On the use of quantum computation in exploration seismology

Shahpoor Moradi, Daniel Trad and Kris Innanen

ABSTRACT

Recent advances in quantum computer hardware and software have led to a jump in the number of discipline areas of pure and applied science identifying themselves as stakeholders in quantum computation (QC) technology. Areas such as chemistry, biology, machine learning and finance are clearly on this list, and it is the purpose of this article to advocate that geophysics should be too. Seismic exploration and monitoring practitioners and researchers in particular stand to gain an enormously powerful tool when QC comes online. Meaningful advances in seismic exploration methods depend on progress in computer hardware and software technology. In our view, it is essential for geophysicists to start to become familiar with the ideas and the potential within the computers and algorithms in the quantum regime, in order to properly take advantage of these tools as they become available. Along this line, the effectiveness in principle of quantum algorithms for seismic wave modeling and seismic imaging is discussed, at the same time introducing in geo-scientific terms the opportunities and challenges of QC. We examine the extent to which QC will be able to solve both the seismic modeling and imaging problems, by exploiting quantum algorithms such as quantum linear systems of equations, quantum Fourier transform and quantum database search.

THE NEED FOR QUANTUM COMPUTING

The idea of quantum computation was first introduced by Nobel Prize winner Richard Feynman (Feynman, 1982). He proposed that quantum systems be simulated through a quantum mechanical version of computation, whereas classical computation becomes very complicated for systems of more than a few particles.

Quantum computers have the potential to support the simulation and modeling of many complex physical systems, not just quantum ones, significantly more rapidly than conventional supercomputers (DiVincenzo, 1995). The increasing demand for computational resources in a variety of disciplines in science, based on this promise, motivates the development of quantum computers in addition to that of classical supercomputers. In geophysics, simulation is critical; ultimately, we would like to use QC for numerical modeling of seismic wave propagation for earthquake modeling and reservoir characterization (Moczo et al., 2007).

Seismologists have for many years relied on high performance computing (HPC) either to locate earthquake sources or search for hydrocarbon deposits. In exploration seismology, geophysicists analyze the echo data from seismic waves interacting with underground layers in the context of complex wave propagation models. In reservoir characterization, accurate monitoring of subsurface structures and fluid movements requires similar simulation of seismic wave propagation (Igel, 2017). A good indicator of the computational resource requirements in geophysics is produced by enumerating the number of discrete elements needed to define and interpret waves in an earth model. In an elastic $10 \times 10 \times 10\ km^3$ geological volume discretized into $1000\ m^3$ sub-cells ($10m$ in each spatial direction), a total of 3×10^9 parameters are required for an isotropic earth,

and 10^{10} parameters for the more realistic anisotropic case. For a numerically stable simulation of a wave within this volume, the time sampling should be small in comparison to the grid spacing which results in a prohibitive amount of CPU time as higher temporal frequencies are required (Moczo et al., 2007). Finally, inversion of seismic data is at least one order of magnitude more expensive as it requires several forward modeling steps for each iteration.

QUANTUM COMPUTING: DEVELOPMENTS AND CHALLENGES

A quantum computer is a machine that uses the principles of quantum mechanics to solve certain tasks more efficiently than its classical counterpart (Nielsen and Chuang, 2010). QC processes use quantum bits (qubits) as the basic unit of information. Unlike the classical bit, which is at any point in either one or the other of two possible states, '0' or '1', a qubit can exist in an inexactly specified mixture of these states, '0+1'. This is a consequence of the quantum superposition principle, which allows a quantum system to be in two different states simultaneously, provided it is not measured. If the system is measured, it collapses to one of the states, taking on either the value '0' or '1'. On atomic scales, several properties of quantum systems can be meaningfully assigned to represent these '0' and '1' states. For example, '0' can be the ground state and '1' the excited state of an atom. Or '0' and '1' can represent the magnetic moment of a particle in opposite directions (i.e., spin directions up or down). In quantum mechanics, in fact, '0' and '1' are not regular numbers, but are two-dimensional vectors, represented in Dirac's notation respectively as $|0\rangle \equiv (1 \quad 0)^T$ and $|1\rangle \equiv (0 \quad 1)^T$, where T means the conjugate transpose. A general qubit, which is the superposition of these zero and one states, is defined by $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where the complex numbers α and β are referred to as *probability amplitudes*: $|\alpha|^2(|\beta|^2)$ is the probability of finding the qubit in the state $|0\rangle$ ($|1\rangle$). A more complex two-level quantum system, involving, for instance, two coupled atoms, contains 2 qubits of information. Such a system can be in a superposition of all possible states, i.e., $\alpha|00\rangle + \beta\,|01\rangle + \gamma\,|10\rangle + \delta\,|11\rangle$, which means 2-qubits can store four different states simultaneously. This 'storage capacity' increases exponentially with the number of qubits. For example, 50 qubits can hold $2^{50}{\sim}10^{15}$ states. A single quantum CPU can simultaneously operate on all of these states. This property is often referred to as *natural parallelism* in the quantum world.

In the classical regime, a linear increase in the number of processors results in a linear increase of computational power. In the quantum setting, however, the computational power increases exponentially in response to a linear increase in the number of qubits. The key conceptual for QC is, that, despite the fact that calculations can be done on all states simultaneously, this superposition collapses to a single state if we apply a measurement. As a result, at the moment of measurement, only one out of 2^n states can be obtained. This means we cannot take advantage of quantum parallelism directly. However, we can design algorithms capable of increasing the probability amplitude of a particular target before the measurement. This procedure is called amplitude amplification and forms the core of many QC algorithms.

One way to understand QC is to compare its practical design to a comparable example in classical computation. For example, we can compare how classical Random Access Memory (RAM) compares with quantum RAM, which in turn can be understood by using the superposition principle. In a classical computer, the RAM is connected to the

CPU through a group of address lines called a *bus*. As each line carries one bit of information, either 0 or 1, with an n-bit address bus, 2^n memory locations can be addressed. Figure 1 illustrates the RAM addressing. Each row contains an n-bit word with 2^n memory locations. In quantum RAM, each word is registered as a quantum state $|m_j\rangle$ with a corresponding quantum address $|j\rangle$. If $\sqrt{p_j}$ corresponds to the probability amplitude of the word address $|j\rangle$, the superposition of query addresses would be $\sum_j \sqrt{p_j}\,|j\rangle$. As a result, the quantum state of RAM would be $\sum_j \sqrt{p_j}\,|j\rangle|m_j\rangle$. In this way, Quantum RAM allows queries to occur in superposition.

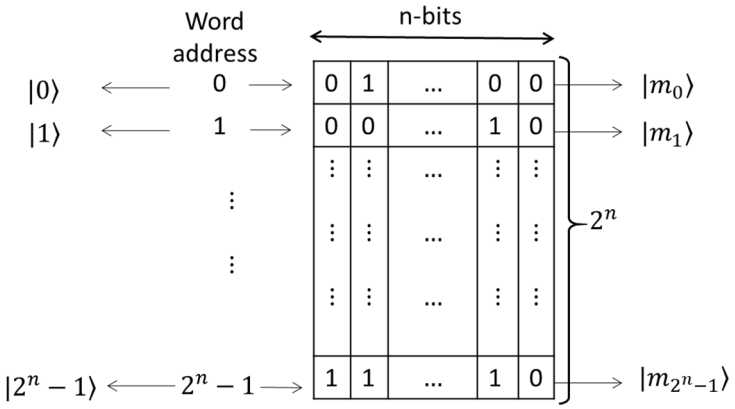

Figure 1. Classical versus quantum memory addressing.

Let us consider a 2-qubit quantum processor. In this case we have $2^2 = 4$ memory locations and the state that describes the quantum RAM is given by

$$|\text{qRAM}_2\rangle = \sqrt{p_0}|0\rangle|m_0\rangle + \sqrt{p_1}|1\rangle|m_1\rangle + \sqrt{p_2}|2\rangle|m_2\rangle + \sqrt{p_3}|0\rangle|m_3\rangle \quad (1)$$

This means accessibility to each memory location with probabilities p_0, p_1, p_2 and p_3.
The architecture of quantum RAM is described in Figures 2 and 3. In a quantum setting, a switch in each node is a three level system called qutrit (Figure 2a). A qutrit can be any quantum system with three levels of energy: ground state, first excited state and second excited state. As long as there is no incoming signal, the qutrit remains in the ground state. After interaction with an incoming signal, it changes to $|0\rangle$ or $|1\rangle$ for 0 and 1 qubit addresses respectively. Whether the qutrit changes to $|0\rangle$ or $|1\rangle$ the path is routed to right or left (Figures 2b & c). The first register interacts with one node, the second register interacts with two nodes and the kth address qubit interacts with k nodes. In this way, the number of time steps to address an n-qubit register is 1+2+3+…n=$n(n+1)/2 \sim O(n^2)$. As a result, we are able to address N=2^n memory cells with a number of operations proportional to $(\log N)^2$. This means if we load classical data on a quantum RAM, their accessibility is exponentially faster than within classical RAM. Each memory address is constructed by a sequence of qubit addresses. For example, consider the memory cell $|a_0 a_1 a_2\rangle$. Three qubit addresses $|a_0\rangle, |a_1\rangle$ and $|a_2\rangle$ are sent through the binary tree sequentially. Each qubit register changes the state of the qutrit at the nodes based on their

values and eventually the routing path is assigned through the several interactions between the qubits and qutrits (Figure 3).

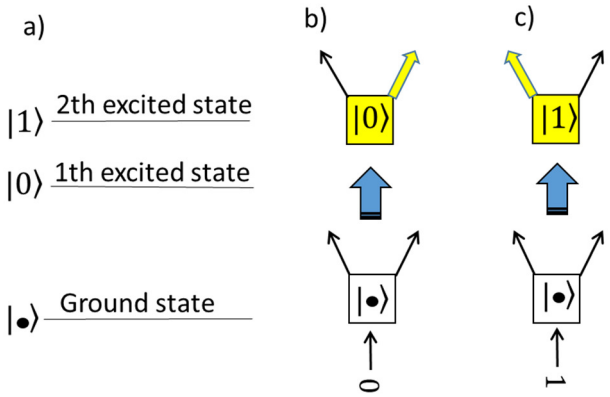

Figure 2. a) Energy levels corresponding to a qutrit. The ground state is separated with an energy gap larger than the energy difference between the first and second excited states. b) Qubit address $|0\rangle$ interacts with the ground state of the qutrit switch and changes its state to $|0\rangle$, turning the right spatial direction for the next qubit register. c) Qubit address $|1\rangle$ interacts with the ground state of the qutrit switch, changes its state to $|1\rangle$, and turns the left spatial direction for the next qubit register.

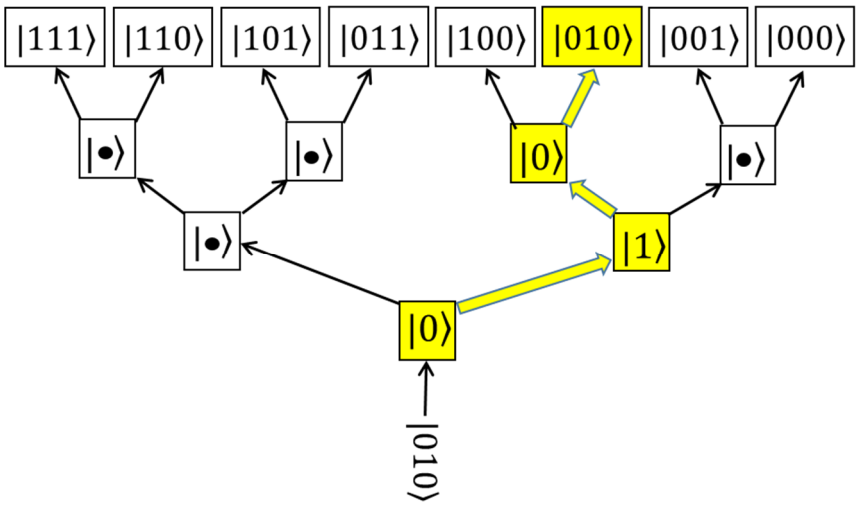

Figure 3. Schematic description of memory cell addressing in Bucket Bridge RAM (Giovannetti et al., 2008). At the first stage, qubit register 0 changes the ground state of the qutrit switch into the $|0\rangle$ state and turns on the right root for the second register, which is 1. In the same fashion, the second and third registers change the switches sequentially into $|0\rangle$ and $|1\rangle$. Finally, the routing path is assigned from the root to the desired memory cell $|010\rangle$ (Figure is adapted from Arunachala et al., 2015).

Quantum computation maps a collection of qubits called a quantum register into another quantum register through operators called *quantum gates*. The most common gate is the Hadamard gate, defined as $H|j\rangle = 2^{-1/2}(|0\rangle + (-1)^j|1\rangle)$ with $j = 0, 1$. These gates can be used to understand more completely what is meant by quantum parallelism. Consider a 4-bit classical register. Within such an object there are $2^4 = 16$ possible different 4-bit words. A single classical computer can process only one of these words at a time. To process each of the different 4-bit possible words simultaneously, we require 16 processors in parallel. This is an easy task nowadays. However, for a 50-bit register, processing simultaneously all possible combinations requires 10^{15} processors, something unfeasible now and in the foreseeable future. At least in theory, generating a quantum register as a superposition of all possible 50-qubit words is, conversely, achievable. Consider the application of a Hadamard gate 50 times to the initial register $\underbrace{|0\ldots00\rangle}_{50\ qubits}$. We get

$$H^{50}\underbrace{|0\ldots00\rangle}_{50\ qubits} \equiv \frac{1}{\sqrt{2^{50}}}\left(\underbrace{|0\ldots01\rangle}_{50\ qubits} + \underbrace{|0\ldots10\rangle}_{50\ qubits} + \cdots \underbrace{|1\ldots11\rangle}_{50\ qubits}\right), \tag{2}$$

The above expression has 10^{15} terms! Any operation on this superposition state can be applied to all 10^{15} registers simultaneously. In the classical mode, to achieve this we would need 10^{15} processors operating on all of the registers at the same time.

The first step in any quantum computation task is the preparation of a well-defined initial state. Figure 4 illustrates the initial state preparation for a quantum computer. In Figure 4a the system is in its lowest level, wherein all atoms are in the ground energy state. By application of a Hadamard gate on this state N times (experimentally with a laser light being smoothly switched on and off), 2^N input configurations are generated as a single entangled state (Figure 4b).

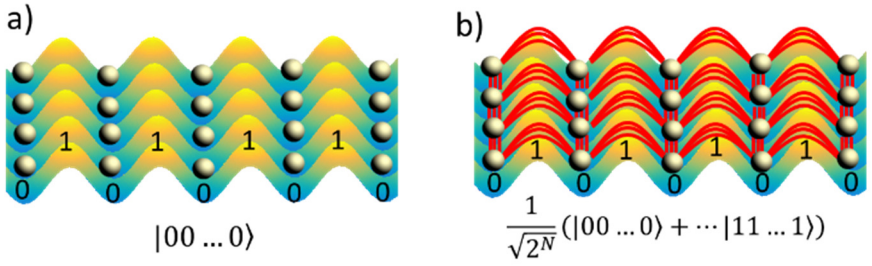

Figure 4. Preparation of the initial state for quantum computation. a) All qubits are in the ground energy state. b) Multiple applications of the Hadamard operator generates entanglement between the qubits. The resulting quantum state has equal probabilities for each register.

Quantum gates are then applied to the initial state, changing it to a final state:

$$|\psi_f\rangle = \alpha_0|0,0,\ldots,0\rangle + \alpha_1|0,0,\ldots,1\rangle + \cdots \alpha_N|1,1,\ldots,1\rangle \tag{3}$$

where $N = 2^n$ for n qubits. During the quantum computation process, the amplitude related to the correct answer must be amplified as a consequence of the instructions given by the algorithm; in fact, this amplification process defines what it means to design a quantum computing algorithm. The final state is not the solution to the problem; however, it is used to achieve the solution by the readout process. When measured, the final state collapses to a classical state. Which state depends on probabilities. Mathematically, the transition from an initial to a final state in a quantum computer is governed by the time dependent Schrödinger equation

$$i\frac{\partial|\psi(\mathrm{t})\rangle}{\partial t} = \mathcal{H}|\psi(\mathrm{t})\rangle$$

Here, $|\psi(\mathrm{t})\rangle$ is the state of the quantum computer at time t and $\mathcal{H}$, the Hamiltonian, is the energy operator containing every gate applied to the initial state. Since the Schrödinger equation is invariant under a time reversal, all quantum gates are reversible. This means that the number of outputs is not less than the number of inputs. Given a Hamiltonian, the time evolution of the quantum computer from an initial state $|\psi_i\rangle$ at time t_i to a final state $|\psi_f\rangle$ at time t_f can be determined by the invertible relationship $|\psi_f\rangle = e^{-i\mathcal{H}(t_f - t_i)}|\psi_i\rangle$. The technique for implementing and constructing the time evolution operator is called *Hamiltonian simulation* (Berry et al., 2007). A QC algorithm pushes the system from an initial state to a final state at a later time. Practically, it is impossible to derive the quantum state from point A to B directly, but it is possible to break up the Hamiltonian into a set of sub-Hamiltonians, $\mathcal{H} = \sum_{k=1}^{m}\mathcal{H}_k$, each of which acts during a period of time $(t_k - t_{k-1})$. In each time step, the quantum system moves slightly in one direction and then in another direction and so on until it reaches the desired final state with accuracy of ε. This process is illustrated in Figure 5.

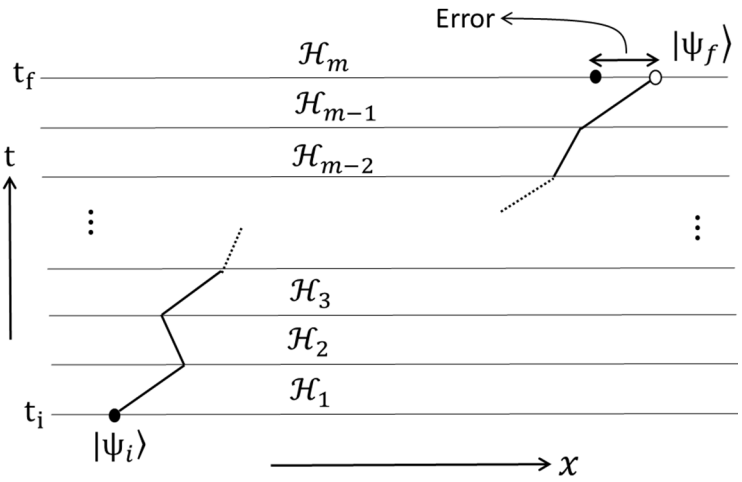

Figure 5. Graph illustrating the Hamiltonian simulation process. A quantum system with initial state $|\psi_i\rangle$ at time t_i moves to another point in space-time. In each level of transition, the corresponding Hamiltonian contributes to the time evolution. After several sub-transitions, the quantum state reaches its final state $|\psi_f\rangle$ with error ε.

Discipline experts refer to examples in which QC greatly exceeds even the possible theoretical performance of classical computers as *quantum supremacy* (Harrow and Montanaro, 2017; Boixo et al., 2018).

One way to compare quantum versus classical computer power (though not efficiency), and to flesh out where quantum supremacy might hold, is to consider how powerful a classical computer needs to be to simulate a quantum computer of a given number of qubits. To date, the largest number of simulated qubits is 45. This means that the fastest existing classical computers is limited to simulating a system with roughly 45 atoms. In such as system, all 45 atoms have magnetic moments and interact with each other, involving 2^{45} configurations. Because the number of configurations doubles with each new atom, the simulation of 46 atoms would be nonviable for a classical computer. However, given a 45-qubit quantum computer the addition of more qubits is in principle straightforward. The simulation of a 45-qubit quantum computer was done on the Cori II supercomputer by using 8,192 nodes and 0.5 petabytes of memory (Häner and Steiger, 2017). This means that a simulation of 60 qubits requires a supercomputer with nearly an exabyte of memory: this is far beyond current technology.

Tempering excitement about the potential of QC are a set of serious technological barriers. One of these is referred to as *decoherence* (Nielsen and Chuang, 2010). The superposition states, which store the information in QC, tend to be fragile, and to decay in very short periods of time. In principle, quantum calculations must be performed in this limited time.

Another challenge comes from the unavoidably defective calculations occasionally executed by quantum gates; just like in classical computations, quantum memory errors occur frequently. To address this issue, quantum algorithms require quantum error corrections. To understand how these corrections work and why they are necessary, we can think of a simple example in the classical world of a bit-flip error: $0 \leftrightarrow 1$. To diagnose this error, one can keep multiple copies of bits, for example '00' and '11'. If a bit-flip error occurs, the states '01' or '10' would be detected in the output. For three reasons similar error diagnosis is unavailable in the quantum domain. First, the *no cloning theorem* (Wootters and Zurek, 1982) prevents copies of qubits from being made. Second, the quantum state must be measured in order to find and correct an error, a process which collapses the quantum state, reducing the qubit to a classical bit. Third, quantum errors are continuous: a single bit '0' or '1' can undergo an arbitrary error becoming a superposition of '0' and '1'.

Although multiple copies of qubits are forbidden, there are ways to generate copies of each bit in a quantum bit. For example, consider the cryptography example of a single qubit state $\alpha|\mathbf{0}\rangle + \beta|\mathbf{1}\rangle$ that a sender (Alice) would like to transfer to a receiver (Bob) through a noisy channel, say a bit-flip channel. By using multiple quantum gates, Alice can generate multiple copies of each bit without violating the cloning theorem. For example, one of these states could be $\alpha|\mathbf{0}00\rangle + \beta|\mathbf{1}11\rangle$. In this way, without being stopped by the no-cloning theorem one could generate two copies of each bit in the qubit. The idea of quantum error correction involves transferring the errors to auxiliary qubits, measuring and discarding them without touching the original qubits. Error correcting in

the quantum regime leads to an increase in the number of qubits, and subsequently requires more memory.

Although in principle straightforward, a large fraction of computational resources involved in QC – more than 90% – are consumed by quantum error correction (Nielsen and Chuang, 2010). This means that hundreds of thousands of qubits are required to execute quantum algorithms for real-world problems.

The design of quantum algorithms is a complex task; it should be anticipated that in any area of applied science, including geophysics, this algorithm design will involve a steep learning curve, and significant time and thought. To be able to make progress in the design and debugging of quantum algorithms prior to the availability of large scalable quantum computers, scientists have been using simulators of quantum computers that work in classical computers. This kind of simulation is also critical for calibration of quantum computers. Many quantum programming languages have been designed or adapted from existing languages in this manner, for the purpose of the execution of algorithms on quantum hardware (Chong et al., 2017). But, effective as they are, these simulators are for the purpose of software design only—no speed-up occurs when one runs a quantum algorithm on a classical computer.

QUANTUM ALGORITHMS: NATURAL PARALLELISM

An algorithm is a step-by-step instruction set for solving a specific problem. One of the key concepts in algorithm theory is computational complexity, which refers to the minimum computational resources required to solve a problem. Analysis of computational complexity is a way to measure the efficiency of an algorithm in its ability to solve a particular problem. There are two famous quantum algorithms often used to compare classical and quantum complexity: the search algorithm (Grover, 1997) and the Fast Fourier Transform (FFT). Searching and updating a list of unordered elements is an important task in many data-processing problems. Suppose we have N given objects and wish to find an item that satisfies a particular condition. If we scan the items one-by-one, on average it takes $N/2$ operations to find the desired item.

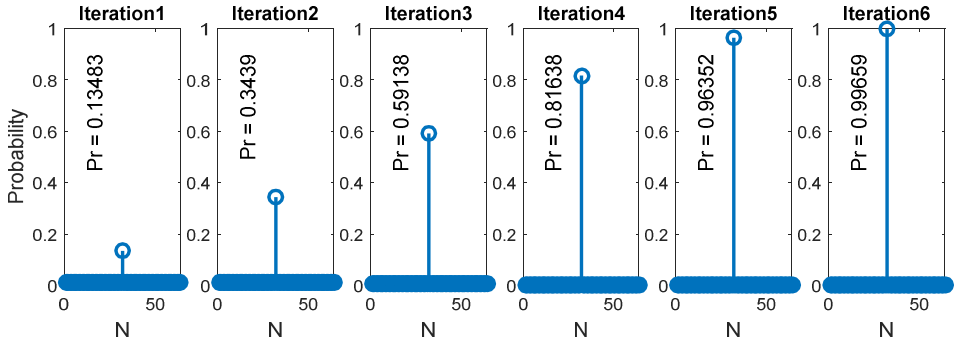

Figure 6: Simulation of amplitude amplification in the search algorithm for a 6-qubit system, chosen to be equivalent to a search space with size N = 2^6 = 64. The initial probability to find the right answer is Pr = 0.013483, but after 6 iterations it increases to Pr = 0.99659.

A well-known quantum search algorithm is *Grover's algorithm.* Like all QC algorithms, it takes advantage of the superposition principle, as a consequence of which multiple operations on multiple inputs can be performed simultaneously. At each iteration, the probability of finding the correct answer is increased, and becomes optimal after $\sqrt{N}$ iterations, after which the probability decreases (Figure 6). This means maximal efficiency is achieved in the QC search problem in $\sqrt{N}$ steps, implying a quadratic speed-up compared to the classical search.

In Figure 6 we illustrate a simulation of a quantum search algorithm for 64 items. It can be seen that the algorithm runs on average 6 times to find the correct answer. This example was calculated with a Matlab simulation tool developed in the CREWES research group.

Another important quantum algorithm is the Quantum Fourier Transform (QFT), which while important in its own right is also one of the key components of many other more complex quantum methods. Given a set of N points $(x_0, \ldots, x_{N-1})$, the discrete Fourier transform (DFT) is another set given by $(y_0, \ldots, y_{N-1})$ with $y_j = \sum_{i=1}^{N-1} F_{ji} x_i$, $j = 0, .., N-1$. Direct calculation of the DFT requires N^2 arithmetical operations, which is not efficient for a large data set. When $N = 2^n$, the Fast Fourier Transform (FFT) algorithm reduces the computational complexity to $N \log N$ by taking advantages of periodic nature of sine and cosine functions.

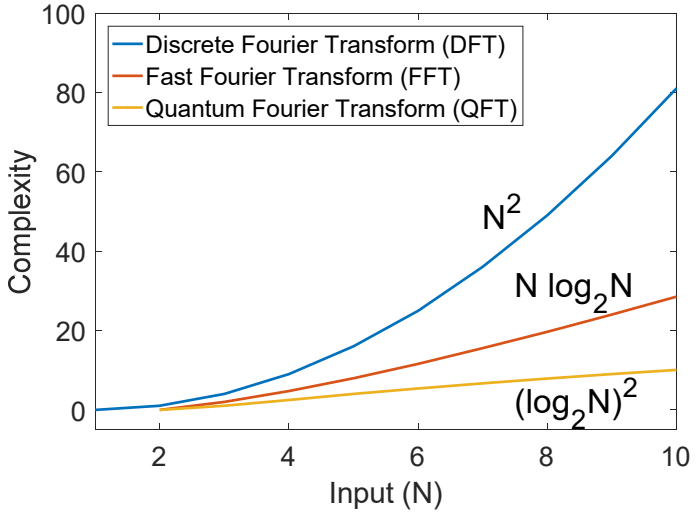

Figure 7: Comparison of the computational complexities of DFT, FFT and QFT algorithms.

Taking advantage of the superposition principle in the quantum regime, 2^n points (states) can be registered by a n qubit system. As a result, technically the DFT is computed over $n = \log N$ points with only $(\log N)^2$ operations. This amounts to an exponential speed-up over FFT. The transition between the time and frequency domains is exponentially faster in the quantum computing world.

QUANTUM LINEAR SYSTEM ALGORITHM (QLSA)

The QLSA, or Harrow–Hassidim–Lloyd (HHL), algorithm (Harrow et al., 2009) is the most recently-developed quantum algorithm concerning the solution of linear system of equations, i.e. $\mathbf{Ax} = \mathbf{b}$.

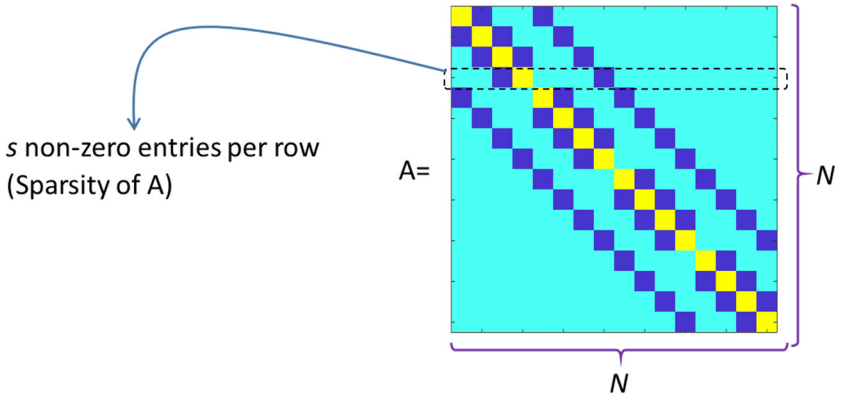

Figure 8. The structure of a sparse matrix used in QLSA.

For a linear system of dimension N, the best classical algorithm is the conjugate gradient (CG) algorithm in multigrids, which can solve this problem for a sparse matrix A in a running time proportional to N. In comparison, QLSA offers an exponential speed up over the CG method, running the algorithm in a time proportional to $\log_2 N$. Let us analyze QLSA in more detail to expose its restrictions and benefits.

The first step in QLSA is loading the N-dimensional vector $\boldsymbol{b} = (b_0, .., b_{N-1})$ into the quantum RAM (qRAM) discussed above. In this procedure, $\log_2 N$ qubits are used to store the components of the vector $\boldsymbol{b}$ in the form of amplitudes for the quantum state $|b\rangle = \sum_{i=1}^{N} b_i |i\rangle$. All of the components of the vector can be read at once via the superposition principle. The state of a quantum system at the current time is related to its value at a later time through a unitary operator called the *time evolution operator*. If we consider $|b\rangle$ to be the initial state and $|x\rangle$ to be the final state (at some later time), the time evolution operator is given by e^{-iAt}. The matrix A corresponds to the interaction that maps the initial state $|b\rangle$ to the solution $|x\rangle$. This is called the Hamiltonian simulation; it requires $\mathcal{O}(ts^2 \log_2 N)$ operations, where t is the time and $s \ll N$ is the sparsity of the matrix (Figure 8), i.e. the largest number of non-zero entries per row. The output is the quantum state $|x\rangle = x_0|0\rangle + x_1|2\rangle + \cdots + x_{N-1}|N-1\rangle$, with amplitudes x_i which encode the components of the vector $\boldsymbol{x} = (x_0, .., x_{N-1})$. This leads to a disappointment: reading all components of the output requires N queries, which results in the loss of the exponential speed-up.

Figure 9 illustrates the probability distribution of $|x\rangle$, for $N = 2^{10} = 1024$. This resembles the output of the QLSA that is not fully accessible for a quantum computer user. Measurement of the superposition collapses the state to one of its basis states $|i\rangle$ with probability of $|x_i|^2$. Evidently, by running the algorithm several times and measuring the final solution each time, we derive the components of $\boldsymbol{x}$ with greater probabilities. However, if we are interested in some statistical features of $\boldsymbol{x}$ rather than a determination of all of its components, we do not need to read all components of $\boldsymbol{x}$. For

example, we can examine whether two sets of linear systems of equations have the same solution. Furthermore, the QLSA allows us to efficiently classify the $|x\rangle$ as an N-dimensional vector into one of two clusters of M-dimensional vectors say $|y\rangle$ and $|z\rangle$ by estimation of the inner products $\langle y|x\rangle$ and$\langle z|x\rangle$ in about $O(\log_2 MN)$ steps, exponentially faster than the best classical algorithm with the $O\big(\text{poly}(MN)\big)$ time steps (Rebentrost et al., 2014). Another example of a calculable quantity utilizing $|x\rangle$ without reading all of its amplitudes lies in the scattering problem via the Finite Element method (FEM) (Clader et al., 2013).

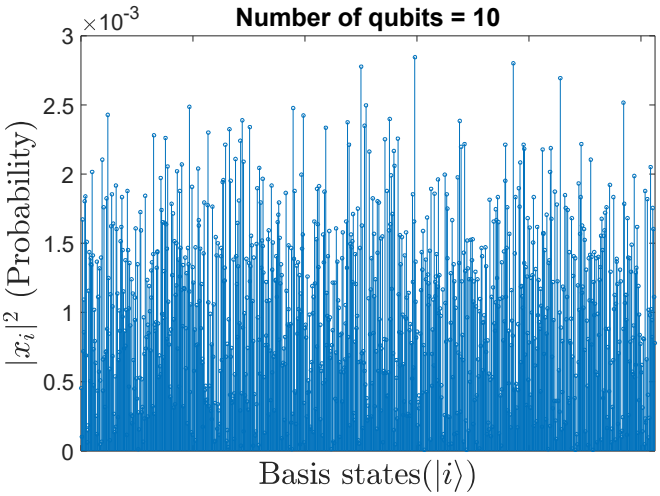

Figure 9. The probability distribution related to the possible solution of the linear system of equation for n=10 or N=1024. The solution is normal to one($\|x\|^2 = 1$). The vertical axis refers to the probability corresponds to each component of the solution i.e. $|x_0|^2$, $|x_1|^2, \ldots |x_{N-1}|^2$. The horizontal axis shows the basis states, i.e. $|0\rangle,\ldots, |i\rangle$.

The differential scattering cross section (DSCS) is defined as a ratio of the outgoing energy-flux density over the incident energy-flux. The total scattering cross section which is the average of DSCS over all directions is proportional to the scattering attenuation factor Q_c^{-1} known as the coda-attenuation factor (Q_c is called 'coda Q'). This factor can be used to estimate the scattering attributes of the lithosphere from the seismic data.

POSSIBLE ALGORITHMS FOR SEISMIC WAVEFORM MODELING

The numerical computation of synthetic seismic data via Finite Difference (FDM) and Finite Element (FEM) methods is a critical part of most modern methods of imaging of subsurface Earth structures. Solution of seismic wave equation on a computer requires both the equation and the space-time domain to be discretized. In FDM and FEM methods, solutions are computed only on specific points called nodes. By interpolating the solution on the nodes, a continuous solution can be achieved. A desired seismic wave equation can be solved in either the time domain or the frequency domain. In time domain, variables such as pressure, velocity and viscosity are extrapolated at each grid point based on the values of a previous time step. Frequency domain solvers transform

the wave equation into the frequency domain and computes the wave field for each frequency.

Method	operations
Gaussian elimination	N^4
Conjugate gradients	N^3
Fast Fourier Transform	$N^2 \log N$
Multigrid iterative	N^2
Quantum	$\log N$

Table. 1 Order of complexity of solving 2D Poisson's equation on an N-by-N grid (Demmel, 1997).

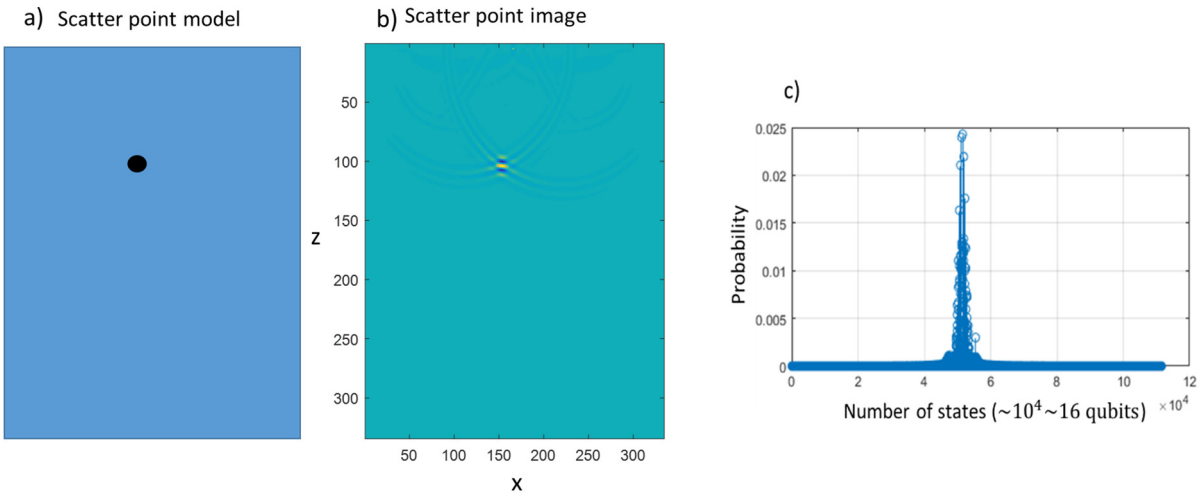

Figure 10. Classical versus quantum model of imaging. a) A scatter point in homogeneous background. b) Imaging the scatter point using RTM. c) Probability distribution corresponding to the location of the scatter.

Let us consider how a quantum computer would solve the wave equation in the frequency domain. Since the wave equation after discretization reduces to a linear system of equations, the final output would be a quantum state with amplitude probability distribution similar to Figure 9. As we discussed in previous section, we would not be able to produce a shot record of the wave field due to the nature of measurement in the quantum regime which would collapse the superposition state into one single state. This means at the end, we retrieve a single value that has the largest amplitude with highest probability. How can we utilize the quantum state solution without measuring and exploring its components? One possibility is to produce the image of the reflector using the imaging condition like in reverse time migration. This condition is the cross-correlation of the forward wavefield from source to reflection point with backward field from reflection point to receiver in time domain; or multiplication of forward with backward wavefield in frequency domain. Figure 10 illustrate the simple example of a scatter point imaging in a homogeneous background. The image is a $N \times N$ matrix with nonzero components at the location of scatter point. In quantum setting this matrix can be

transformed into a N^2 dimensional vector with a probability distribution shown in fig 10c. Our model is 350×350, so the quantum state corresponds to the image is in order of 10^5. To store this image on a quantum computer we need roughly 17 qubits.
Now, let us examine the number of floating point operations required for a seismic modeling problem of standard size. Assume, for example, $h = 10\,m$ as the grid-spacing and a model with dimensions of $L_x = L_y = L_z = 10\,km$. We need around 10^9 grids to simulate the wave propagation in the entire area. Assume we run a simulation with a number of time steps $N_t = 10^4$ and a number of shots $N_s = 10^4$. With an accurate 8-point first derivative operator ($N_{tn} = 250$ operations per node per time step for 3D modeling and 170 for 2D modeling), we need to perform $N_{float} \sim N_{tn} N_s N_t L_x L_y L_z / h^3 \sim 2.5 \times 10^{17}$ floating-point operations to finish the simulation.
Assuming a computer with 1 Gflops power (giga-flops, 10^9 floating-point operations per second), the simulation would take around 8 years. With a Tflops (tera-flops, 10^{12} floating-point operations per second), a supercomputer such as *Nvidia Titan V* (15 Tflops) will take around 5 hours to run this simulation. If we want to simulate the wave propagation with higher resolution, the memory requirement will increase with mesh refinement. For example, with $h = 1\,m$, which also requires finer time sampling and therefore more time steps, say $N_t = 10^5$, the number of floating-point operations to finish the simulation would be 2.5×10^{21}. Even with the *Nvidia Titan V,* this simulation would take 5 years and 8 months!

Table 1 gives the computational times required for solving the 2D Poisson equation (which is similar to the 2D acoustic wave equation in the frequency domain). Amongst classical solvers, the Gaussian elimination method is the most expensive and the Multigrid iterative method is the most efficient (Demmel, 1997). We see that the computational time for a quantum solver is reduced logarithmically in terms of the numbers of grid points in each direction due to the quantum superposition.
Let us next compare one on the fastest classical algorithms for the Poisson equation with the quantum solver. Suppose that the number of grid points in each direction is $N = 10^4$, the 2D Poisson equation using Conjugate gradients on a classical computer needs roughly 10^{12} Flops with $10^8 \sim 0.1$ Gb of RAM. The same technique, if implemented on a quantum computer, requires only 2.3×10^3 Flops with 176 qubits.

The above-discussed CPU computational time is an example of a measure for computational complexity for a seismic problem. To solve the Helmholtz Equation (HE) there are a variety of techniques available, and one of the motivations to choose a specific approach is its computational complexity. A typical instance of seismic modeling requires the discretization of the HE with at least $N^3 = 10^9$grid points (N=1000 grid point in each direction), which means a solution of a linear system of equations with millions or billions of unknowns. These kinds of algebraic tasks are very expensive and sometimes impossible to accomplish. Consider the 3D problem in the frequency domain with $N = 1000$ grid points in each direction. The computational complexity for an iterative solver for this problem is proportional to $O(N_f N_{it} N_s N^3)$, where $N_f \sim N$ is the number of frequency, $N_t \sim N$ is the number of iteration and $N_s \sim N^2$ is the number of shots in the surface. Therefore, this modeling requires $N_{float} \sim N^7 = 10^{21}$ Flops!

Seismic wave modeling is at the heart of any FWI algorithm. It is used to generate the synthetic data in each iteration in the inversion procedure, as well as to compute the gradient for optimization of the objective function.
Although the 3D FWI and RTM are generally implemented in the time domain, it is possible to formulate them in the frequency domain. Forward modeling for the inversion procedure can be computed by numerical simulation of the Helmholtz Equation (HE), which is the wave equation in the frequency domain. The advantage is that only a few frequencies, rather than the full spectrum, needs to be calculated.

SUMMARY

Quantum computation have been developed as a computationally efficient paradigm to solve problems that are intractable with conventional classical computers.

The idea of using quantum computation in seismology should be very exciting to both physicists and seismologist. Researchers in quantum computation are looking for new quantum algorithms to demonstrate the substantial speed-up over the classical counterparts. At the present, the interplay between quantum theory and geophysics remains unexplored. However, we believe that quantum computation shows a significant potential to deal with the large-scale modeling and inversion problems geophysicists currently encountered.

Among the developed quantum algorithms, the HHL algorithm can act as a rapid solver for seismic wave equation, which is a linear systems of equations. In this paper we have identified several opportunities for seismologist at the intersection of seismology and quantum computation. Among them, the development of quantum algorithms to solve the wave equation and seismic imaging. As an example, by borrowing the idea of probabilistic interpretation of a quantum state, we propose an effective method to image the subsurface reflectors.

ACKNOWLEDGEMENTS

The authors would like to thank CREWES industrial sponsors and NSERC under the grant CRDPJ 461179-13 for funding this work. We also would like to thank Dr. Samual Gray for his valuable and constructive comments on this research.

REFERENCES

Feynman, R.P., 1982. Simulating physics with computers. *International journal of theoretical physics*, *21*(6-7), pp.467-488.

DiVincenzo, D.P., 1995. Quantum computation. *Science*, *270*(5234), pp.255-261.

Moczo, P., Robertsson, J.O. and Eisner, L., 2007. The finite-difference time-domain method for modeling of seismic wave propagation. *Advances in geophysics*, *48*, pp.421-516.

Igel, H., 2017. *Computational seismology: a practical introduction*. Oxford University Press.

Nielsen, M.A. and Chuang, I.L., 2010. *Quantum Computation and Quantum Information*. Cambridge University Press.

Giovannetti, V., Lloyd, S. and Maccone, L., 2008. Architectures for a quantum random access memory. *Physical Review A*, *78*(5), p.052310.

Arunachala, S., Gheorghiu, V., Jochym-O'Connor, T., Mosca, M. and Srinivasan, P.V., 2015. On the robustness of bucket brigade quantum RAM. *New Journal of Physics*, *17*(12), p.123010.

Berry, D.W., Ahokas, G., Cleve, R. and Sanders, B.C., 2007. Efficient quantum algorithms for simulating sparse Hamiltonians. *Communications in Mathematical Physics*, *270*(2), pp.359-371.

Harrow, A.W. and Montanaro, A., 2017. Quantum computational supremacy. *Nature*, *549*(7671), p.203.

Boixo, S., Isakov, S.V., Smelyanskiy, V.N., Babbush, R., Ding, N., Jiang, Z., Bremner, M.J., Martinis, J.M. and Neven, H., 2018. Characterizing quantum supremacy in near-term devices. *Nature Physics*, *14*(6), p.595.

Häner, T. and Steiger, D.S., 2017, November. 0.5 petabyte simulation of a 45-qubit quantum circuit. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (p. 33). ACM.

Wootters, W.K. and Zurek, W.H., 1982. A single quantum cannot be cloned. *Nature*, *299*(5886), pp.802-803.

Chong, F.T., Franklin, D. and Martonosi, M., 2017. Programming languages and compiler design for realistic quantum hardware. *Nature*, *549*(7671), p.180.

Grover, L.K., 1997. Quantum mechanics helps in searching for a needle in a haystack. *Physical review letters*, *79*(2), p.325.

Harrow, A.W., Hassidim, A. and Lloyd, S., 2009. Quantum algorithm for linear systems of equations. *Physical review letters*, *103*(15), p.150502.

Rebentrost, P., Mohseni, M. and Lloyd, S., 2014. Quantum support vector machine for big data classification. *Physical review letters*, *113*(13), p.130503.

Clader, B.D., Jacobs, B.C. and Sprouse, C.R., 2013. Preconditioned quantum linear system algorithm. *Physical review letters*, *110*(25), p.250504.

Demmel, J.W., 1997. *Applied numerical linear algebra* (Vol. 56). Siam.