A quantum algorithm for traveltime tomography

Jorge E. Monsegny[1], Daniel Trad[1] and Don C. Lawton[1,2]
[1]CREWES, University of Calgary; [2]Carbon Management Canada

ABSTRACT

Quantum computation is described as a promising paradigm for the future. One of their advantages is the quantum parallelism, that consists in solving many instances of the same problem in a single run. This can be done due to the possibility to set a quantum system in a superposition of states. Although its main limitation is that only one of the states can be read at the end, it is possible to increase the chances of the state we are looking for. In this report we show a framework to pose the traveltime tomography problem, usually solved using gradient methods, as a quantum computing algorithm. This algorithm does a global exploration of the model space using the aforementioned quantum parallelism. Then it manipulates the quantum phase of their states to increase the chances of reading the model that produces the global minimum. In that way we can read the tomography answer at the end of the quantum computation. Something important to notice is that there is no need to compute gradients or Hessians but only to do forward modelling and residual calculation. In this report we introduce only the notions needed to solve the inverse problem and we show a small example step by step to illustrate how the quantum algorithm works. The algorithm has been coded and run in a quantum simulator.

INTRODUCTION

In 1981 quantum computation was proposed by physicist Richard Feynman to simulate quantum systems (Preskill, 2021). Later, in 1985, David Deutsch extended the notion of quantum computer to problems not related to quantum physics (Deutsch and Penrose, 1985).

Several applications appeared in the following decade showing that quantum computing is more powerful than classical computing. For example, an exponential speedup was found in the period finding problem (Simon, 1994) that inspired the Shor algorithm for factoring large numbers (Shor, 1997) that threatens cryptological systems. The quantum search algorithm that inspires the algorithm proposed in this report is also from that decade (Grover, 1997) and its speedup is quadratic.

Some applications of quantum computing to geophysical problems have been proposed in the last 10 years. Most of them use a less powerful technique called quantum annealing (de Falco and Tamascelli, 2011). Only in the last couple of years general quantum computing algorithms have been proposed in geophysical problems. Alulaiw and Sen (2015) find that quantum annealing is faster than the simulated annealing and does not get trapped in local minima when performing seismic prestack inversion. Moradi et al. (2018) present the different quantum computing concepts necessary for wave modelling applications in geophysics.

Sarkar and Levin (2018) use quantum annealing to estimate the rock material percent-

ages with traveltime information. Greer and O'Malley (2020) introduce a quantum annealing algorithm to solve constrained optimization problems and applies it to invert the subsurface P-wave velocity. van der Linde (2021) solves the residual statics problem using quantum annealing and quantum-classic hybrid solvers. In Souza et al. (2022) the velocities of a layered model are inverted using quantum annealing techniques. Albino et al. (2022) solve the same problem with a more general gate-based quantum computer but using a variational quantum algorithm.

This report is structured as follows. First we show an overview of the quantum computing theory, illustrating the ideas that will be used in the rest of the report. Then we present in a more formal way the quantum bits, quantum gates, quantum arithmetic and quantum amplitude amplification techniques, needed in the quantum tomography algorithm. In the third section we show the quantum traveltime tomography algorithm by running a small instance of it. At the end we discuss the results and present some conclusions. In the appendix we include the simulation Python code.

QUANTUM COMPUTING OVERVIEW

Classical computers differ from Quantum computers in that the later can be in more than one computation state at the same time. Figure 1 depicts three bits of information in a classical computer. Each column of two circles is a bit that can be in only one of two possible states: 0 or 1, represented in the diagram as filled circles. In this example the three bits are in state 110, from left to right.

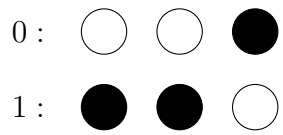

FIG. 1. Three classical bits. Each column of two circles is a bit. The filled circles indicate their states. The states are 110, from left to right.

On the other hand, the quantum equivalent of a bit, called quantum bit or *qubit* can be in a mixture or *superposition* of states. Figure 2 shows three possible states of three qubits. On the left all 0 circles are filled and in consequence the state is 000. On the centre all the 1 circles are filled resulting in a 111 state. On the right is a state that differs from all the possible classical ones. In it, each circle is partially filled, meaning that the corresponding qubit is in a superposition of states 0 and 1. There is also a phase factor that the diagrams do not show but we introduce in the next section.

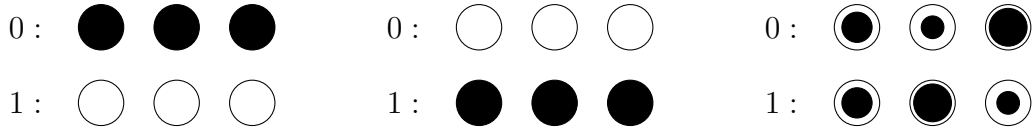

FIG. 2. Three possible states of a set of three qubits. Each column of two circles is a qubit. The shaded portion inside each circle indicates the amount of each possible states, 0 and 1, the qubit can be. On the left all the qubits are in state 0, while in the middle all are in state 1. On the right each qubit is in a different mixture or superposition of states 0 and 1. The figures do not show the phase factor that will be introduced in the following section.

The advantage of being in a superposition of states is that with a quantum computer

is possible to make computations in that superposition and obtain a superposition of the results in a single run. This is called *quantum parallelism*.

To be more specific, consider the wave simulation problem. In a classical computer you have a velocity model made of cells with propagation velocities and you produce a wavefield where cells contain the wave amplitudes. In a single run with a single processor you can compute the outcome for a single velocity model.

In contrast, in a quantum computer you can have a superposition of many velocity models. Each cell in that model contains the superposition of different propagation velocities. In a single run, with a single quantum computer is possible to obtain a superposition of the wavefields that are the result of the wave simulation of all superposed velocity models.

However, there is catch. Although the quantum computer produces a superposition of all the computation results, it is only possible to read one of them. This is due to the properties of *quantum measurement*. Figure 3 illustrates the measuring of a qubit in a non trivial superposition. After the measurement it collapses to 0 or 1. The filled portion of the circles is proportional to the probability of reading the qubit in that state. In this example, the shaded portion corresponding to circle 1 is bigger so there is more chance to read this qubit in state 1.

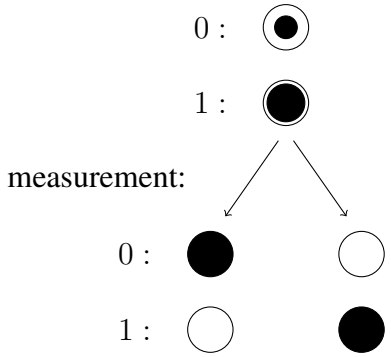

FIG. 3. Above the qubit is in a superposition of states 0 and 1. The shaded portion of the circle indicates the probability of the qubit to turn into that state after being read. Below are the two possible measurement outcomes.

Another example is in Figure 2; the two states on the left and centre of that figure are read with 100% chance in states 000 and 111, respectively. The state on the right of the figure is different; the chance of reading any qubit as 0 or 1 is not 100% because the circles are partially filled.

As it is only possible to read a single final state of a quantum computation, we have to use a quantum computer differently from a classical parallel one. Instead of doing all the calculations and collecting them at the end, we do all the calculations in superposition and increment the probability of one of them so it can be read with higher probability than the others. This single solution must have a special property that solves the problem at hand. In the traveltime tomography problem described in subsequent sections this property is related to the residuals.

Figure 4 shows a general quantum algorithm flow diagram. The first step consists on setting the superposition of states that contribute to the inputs. The second step is to perform computations in this state superpositions. At this point the quantum computer contains all the results in superposition, but as was already mentioned, it is not possible to read but one of them. The third step is in charge of increasing the probability of one subset of those results. This subset depends on the characteristics of the answer we are looking for. Most of the times it is necessary to repeat the second and third steps several times until the probability of the answer is high enough to produce a successful reading. The last step in the flow diagram is to read the answer.

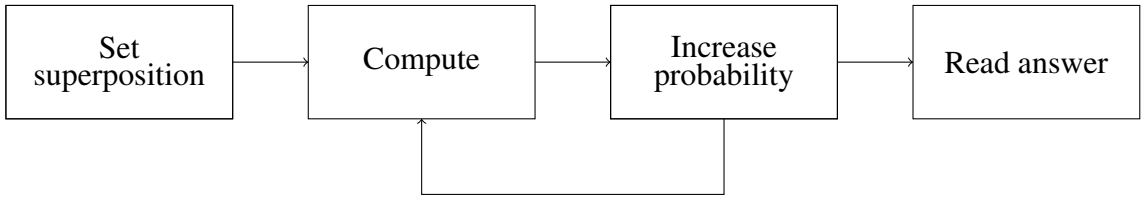

FIG. 4. A general quantum algorithm flow diagram. It starts by setting the states superpositions we want to use as inputs. Then it computes in superposition. The next step is to increase the probability of a subset of the answers. It is possible that this step and the previous one have to be repeated several times until the probability of the answer is high enough. The last step is to read the answer. Diagram modified from Johnston (2019).

In the next section we will explain with more formality the qubits and quantum gates needed to solve the traveltime tomography problem.

THEORY

Quantum bits

Traditional computers are binary due to the high commutation speed between two voltage levels that electronic technology can achieve. These two levels are usually assigned two values, like 0 and 1 or true and false, among others. A system capable of having these two values is called a *bit*. Other pieces of computer information like integer numbers, real numbers or characters are encoded using sets of bits.

On the other hand, quantum systems can take a superposition of states and perform calculations on this ensemble of states simultaneously. There is not yet a definitive hardware implementation of quantum computers but they usually choose two states, following the electronic tradition, and assign the number 0 to one state and 1 to the other.

To be more precise and following the usual quantum notation, one state is written $|0\rangle$ and the other $|1\rangle$, and pronounced ket zero and ket one, respectively. The superposition of these two values is noted like a weighted combination of values:

$$|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{1}$$

where α and β are complex numbers, and $|\alpha|^2 + |\beta|^2 = 1$. A system capable of having this superposition is called a *qubit*.

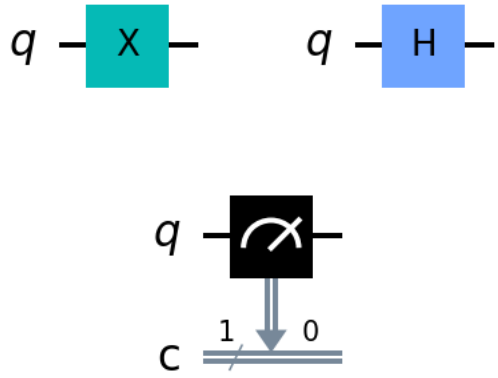

FIG. 5. Three single qubit gates. The inputs are on the left side and the outputs on the right. Above are the not and the Hadamard gates. Below is a measuring gate.

The meaning of the numbers α and β has something to do with the action of measuring or reading the qubit. When you measure a quantum system, it collapses to one of the basis elements you are measuring into. In the qubit case, when measured, it collapses to $|0\rangle$ with probability $|\alpha|^2$ and to $|1\rangle$ with probability $|\beta|^2$. You might have noticed that these squared moduli are proportional to the shaded portions of the circle diagrams in Figure 3.

As explained before, measuring damages the superposition stored in the qubit and with it the quantum parallelism. However, this is something that is done at the very end of a quantum computation and after we make the probability of one of the values, usually the answer to our calculation, as high as possible.

Quantum gates

Quantum computers evolve the superposition stored in the qubits in ways that are useful to perform calculations. The mathematics behind the evolution of quantum systems is the mathematics of unitary complex matrices but it can also be described with *quantum gates* in the style of the logic gates used to describe digital circuits.

Figures 5 and 6 show some of the most common quantum gates. They have the inputs in the left side and the outputs in the right side, with the gate in the middle.

The gate on the left of Figure 5 is called flip gate or *not* gate and is defined by its action on basis qubits. It transforms $|0\rangle$ into $|1\rangle$ and vice versa. Its action on a superposition $\alpha|0\rangle + \beta|1\rangle$ is to transform it into $\beta|0\rangle + \alpha|1\rangle$. This is usually expressed using arrow diagrams:

$$|0\rangle \overset{not}{\longmapsto} |1\rangle \tag{2}$$

$$|1\rangle \overset{not}{\longmapsto} |0\rangle \tag{3}$$

$$\alpha|0\rangle + \beta|1\rangle \overset{not}{\longmapsto} \alpha|1\rangle + \beta|0\rangle \tag{4}$$

The next gate in Figure 5 is called *Hadamard* gate and is very important in the creation of quantum superpositions. It will be used in the first box of the quantum algorithm diagram flow of Figure 4.

This gate turns $|0\rangle$ into $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$, and $|1\rangle$ into $\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$. Note that the difference is the sign of the second term. It is also important to notice that it transforms a basis qubit into an equal superposition of the basis qubits. It is equal because $|\frac{1}{\sqrt{2}}|^2 = |-\frac{1}{\sqrt{2}}|^2 = 1/2$. Using arrow diagrams:

$$|0\rangle \overset{H}{\longmapsto} \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \tag{5}$$

$$|1\rangle \overset{H}{\longmapsto} \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle \tag{6}$$

The gate on the bottom of the same figure is the reading or *measuring* gate. It is used in the last box of the quantum algorithm diagram flow in Figure 4.

This quantum gate has two lines: q for a qubit y c for a classical bit. It reads q, destroying the superposition, and outputs a $|0\rangle$ or a $|1\rangle$ at random in the bit c. After several readings of a qubit generated in the same way, the normalized counts should approximate the probabilities determined by the superposition coefficients.

Multiple qubits

The two gates in Figure 6 work in sets of two qubits. Before continuing we have to see how to group qubits for computations. We know a single qubit has basis elements $|0\rangle$ and $|1\rangle$. We form the basis elements of groups of qubits using the tensor product rules, or basically, concatenating the basis elements of the single qubits in all possible combinations.

For example, two qubits have basis elements $|0\rangle|0\rangle$, $|0\rangle|1\rangle$, $|1\rangle|0\rangle$ and $|1\rangle|1\rangle$. Three quantum bits have basis elements $|0\rangle|0\rangle|0\rangle$, $|0\rangle|0\rangle|1\rangle$ and so on, up to $|1\rangle|1\rangle|1\rangle$. A typical 3-qubit can be expressed as:

$$|\phi\rangle = \sum_{i,j,k=0,0,0}^{1,1,1} \alpha_{ijk}|i\rangle|j\rangle|k\rangle \tag{7}$$

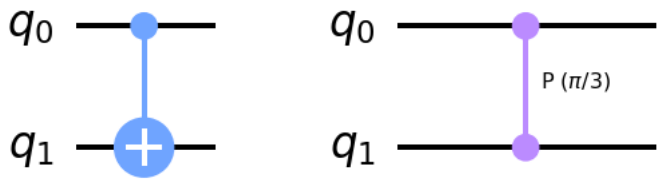

FIG. 6. Two multiple qubit gates. On the left is the controlled not or cnot gate. On the right is a phase gate.

where i, j, k take the values 0 and 1, $\alpha_{ijk} \in \mathbb{C}$ and $\sum_{i,j,k=0,0,0}^{1,1,1} |\alpha_{ijk}|^2 = 1$. Each amplitude $|\alpha_{ijk}|^2$ is the probability the 3-qubit is in state $|i\rangle|j\rangle|k\rangle$.

It is usual to abbreviate the notation by using a single ket symbol around all the qubit values: $|i\rangle|j\rangle|k\rangle$ to $|ijk\rangle$. A further simplification is to transform the binary values to their decimal representation. For example $|011\rangle$ would be $|3\rangle$.

Quantum gates with multiple inputs

The first gate with multiple inputs is the *controlled not* or *cnot*, shown in the left part of Figure 6. The controlling qubit is q_0 and the controlled one is q_1. If and only if q_0 is $|1\rangle$, a not gate is applied to q_1. For example, the action on $|00\rangle$ and $|01\rangle$ is:

$$|0\rangle|0\rangle \xmapsto{cnot} |0\rangle|0\rangle \tag{8}$$

$$|0\rangle|1\rangle \xmapsto{cnot} |1\rangle|1\rangle \tag{9}$$

where the controlling qubit is the rightmost one. An interesting case is when the controlling qubit is a superposition: the not gate is applied and not applied to the controlled qubit at the same time. In this way the cnot gate is performing both actions simultaneously in a quantum parallel way. For example, if the controlling qubit is $\alpha|0\rangle + \beta|1\rangle$ and the controlled qubit is $|0\rangle$, the action is:

$$|0\rangle(\alpha|0\rangle + \beta|1\rangle) = \alpha|00\rangle + \beta|01\rangle \xmapsto{cnot} \alpha|00\rangle + \beta|11\rangle \tag{10}$$

There exist versions of the cnot gate with multiple controlling qubits and multiple controlled ones. They work in a similar way: when all the controlling qubits are one, all the controlled qubits are flipped.

The last gate in Figure 6 is the *phase gate*. It changes the phase of two qubits if both are $|1\rangle$. The way to change the phase is to remember that the coefficient of the qubit is a

complex number that can be expressed in terms of amplitude and phase: $\alpha = |\alpha|e^{i\phi}$. The gate specifies an angle that is added to the coefficient phase. For example, if both qubits are $|1\rangle$ and the gate angle is $\pi/3$, the action is the following:

$$\alpha|11\rangle = |\alpha|e^{i\phi}|11\rangle \xmapsto{pha(\pi/3)} |\alpha|e^{i\phi}e^{i\pi/3}|11\rangle = |\alpha|e^{i(\phi+\pi/3)}|11\rangle \tag{11}$$

Note that the exponential factor has magnitude equal to 1, so it does not change the qubit probabilities. As was the case with the cnot gate, if one of qubits is in superposition, the phase is added only to one of the states:

$$|1\rangle(\alpha|0\rangle + \beta|1\rangle) = \alpha|11\rangle + \beta|10\rangle \xmapsto{pha(\pi/3)} \alpha e^{i\pi/3}|11\rangle + \beta|10\rangle \tag{12}$$

Similar to the cnot case, there are versions of this gate for more qubits that add the phase only when all qubits are $|1\rangle$. The gates just described are not the only ones used in quantum computing (Johnston, 2019), but are the ones necessary for this report.

Encoding numbers with qubits

We are interested in numerical computations. That is why we need to express numbers using qubits. In classical computing the bits can be interpreted as unsigned integers, signed integers or floating numbers, among others. In quantum computing the same encodings can be used.

An set of n qubits $q_{n-1}, q_{n-2}, \ldots, q_0$ can be interpreted as the unsigned number that is a power of two expansion:

$$|q_{n-1}, q_{n-2}, \ldots, q_0\rangle \mapsto \sum_{i=0}^{n-1} q_i 2^i \tag{13}$$

For example, $|101\rangle$ can be interpreted as the number $5 = 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$. A superposition of qubits is interpreted as a superposition of the corresponding unsigned numbers.

In the case of the signed numbers, there are two options: ones' complement and two's complement (There is no typo in the apostrophes). We use two's complement because is more widespread. The set of qubits $q_{n-1}, q_{n-2}, \ldots, q_0$ is interpreted as:

$$|q_{n-1}, q_{n-2}, \ldots, q_0\rangle \mapsto -q_{n-1}2^{n-1} + \sum_{i=0}^{n-2} q_i 2^i \tag{14}$$

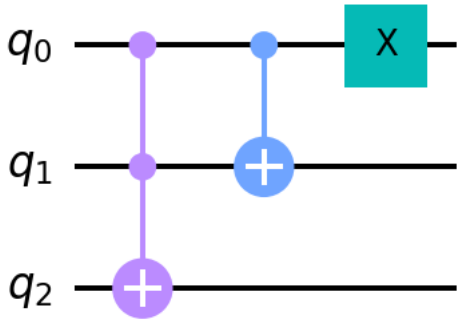

FIG. 7. Increment quantum circuit for three qubits. It is composed of two cnot and one not gates.

Notice that the summation goes up to $n-2$ and that the qubit q_{n-1} is used similarly to a sign: when is $|0\rangle$, the interpreted number is positive and when is $|1\rangle$, the number is negative. Using the same example as before, $|101\rangle$ is interpreted as $-3 = -1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$.

There are also encodings for real numbers, but we are not going to use them in this report. We suggest to consult Johnston (2019) to know more about them.

Quantum arithmetic

With a set of qubits that we can interpret as numbers we can use quantum gates to perform arithmetic operations on them. Remember that if the qubits are in superposition, the arithmetic operations produce a superposition of results.

We are going to introduce only one quantum arithmetic circuit. This circuit is the increment by one circuit shown in Figure 7. This particular circuit operates in sets of three qubits $|q_2 q_1 q_0\rangle$. It first flips q_2 if q_0 and q_1 are both $|1\rangle$, then it flips q_1 if q_0 is $|1\rangle$, and finally, it flips q_0.

As an example, if the input is $|011\rangle$, 3 in binary, the outputs of each stage are:

$$|3\rangle = |011\rangle \xmapsto{ccnot} |111\rangle \xmapsto{cnot} |101\rangle \xmapsto{not} |100\rangle = |4\rangle \tag{15}$$

The first cnot is doubly controlled and called ccnot. It flips q_2 because q_0 and q_1 are $|1\rangle$. The second cnot flips q_1 and the final not flips q_0. The final output is interpreted as $|4\rangle$.

If the input is in the superposition $\alpha|1\rangle + \beta|4\rangle$ the circuit performs the following operations:

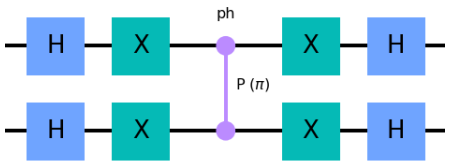

FIG. 8. Quantum phase amplification circuit for two qubits, also known as mirror or Grover's iteration circuit. It rotates the qubits amplitudes around their mean. It is used to increase the amplitude of a subset of terms in the quantum superposition.

$$\begin{aligned} \alpha|1\rangle + \beta|4\rangle = \alpha|001\rangle + \beta|100\rangle &\xmapsto{ccnot} \alpha|001\rangle + \beta|100\rangle \\ &\xmapsto{cnot} \alpha|011\rangle + \beta|100\rangle \\ &\xmapsto{not} \alpha|010\rangle + \beta|101\rangle = \alpha|2\rangle + \beta|5\rangle \end{aligned} \tag{16}$$

Other quantum arithmetic circuits needed in this report are the addition circuit and the multiplication circuit. They are explained in Johnston (2019) and work similarly to the quantum increment circuit just shown, so we are going to depict them as black boxes in quantum circuits. They are all used in the second box, compute, of the quantum algorithm diagram flow of Figure 4.

Quantum phase amplification

The only box in the quantum algorithm diagram flow depicted in Figure 4 we have not addressed yet is the third one, increase probability. The *quantum phase amplification circuit* helps to perform this part of a quantum algorithm. This circuit is also called *Groover's iteration* or *mirror*.

Figure 8 depicts a version of the mirror circuit for two qubits. This circuit performs the following operations:

1. Apply Hadamard gate to every qubit.
2. Flip every qubit.
3. Multiply by $e^{i\pi} = -1$ the element composed of all ones $|1 \ldots 1\rangle$.
4. Flip every qubit.
5. Apply Hadamard gate to every qubit.

The mirror circuit reflects the amplitude coefficients around the mean. Figure 9 shows how this reflection is performed. On the left is a graph of all the amplitudes. There is an entry in the horizontal axis for every term in the state superposition. The vertical axis shows the signed magnitude of the corresponding coefficient in the superposition. They are

equal and positive, except one that is negative. The term with this negative coefficient is the one we are going to amplify. The dashed line is the amplitudes mean. This mean is slightly below the positive amplitudes due to the contribution of the single negative amplitude.

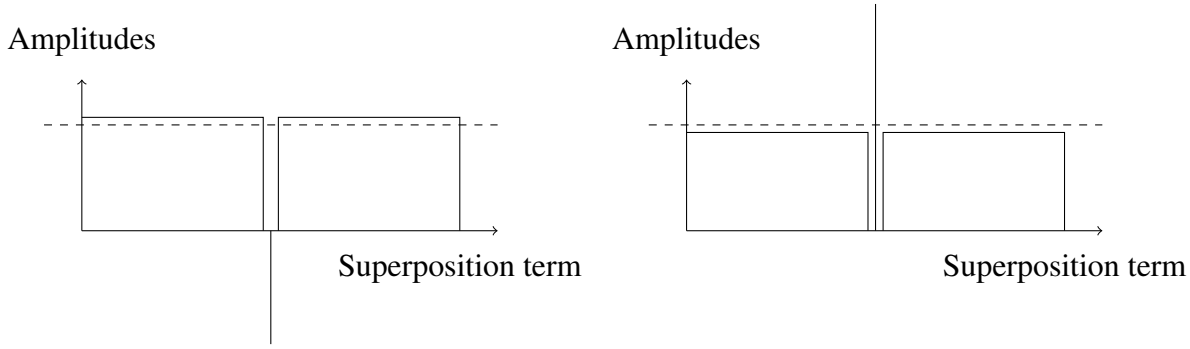

FIG. 9. Quantum phase amplification or Groover's iteration. The left plot are the amplitudes of each term in the superposition. There is one amplitude, corresponding to the term we want to increase its probability, that has the opposite sign or phase. The dashed line is the mean amplitude. Quantum phase amplification reflects all amplitudes around the mean, increasing the amplitude of the target term and decreasing the others as the right part shows.

The right part of the same figure is after the application of the mirror circuit. Imagine the original amplitudes being rotated by using the mean as the rotation axis. The positive amplitudes are now slightly below the original mean while the negative amplitude is now bigger than the rest.

To summarize this circuit, the initial amplitudes were equal except one that was negative, that is, it had the opposite phase. After the mirror operation this phase difference was converted to an amplitude difference. In the quantum algorithm described in the next section we flip the phase of the term that is the solution of our geophysical inversion problem and then we transform that phase difference into an amplitude difference by using the mirror circuit. This higher amplitude of the solution increases its probability of being read as the final answer.

QUANTUM TRAVELTIME TOMOGRAPHY ALGORITHM

First arrivals tomography is the geophysical problem solved by the proposed quantum algorithm. To illustrate it we use the most simple first arrivals tomography instance: one time measurement and one velocity cell. Figure 10 shows this problem configuration where the star is the source, the triangle is the receiver, the square is the cell and the arrow is the ray path. The objective is to estimate the cell slowness knowing the raypath length between source and receiver and the wave traveltime.

The obvious solution of this problem is to divide the traveltime by the ray path length to obtain the cell slowness. However, we are going to show how to solve it doing the calculations in a quantum superpositions of velocity models. This will help understand how to solve a bigger problem with more cells, sources and receivers.

The gradient descend strategy to solve a problem like this is to start with an initial

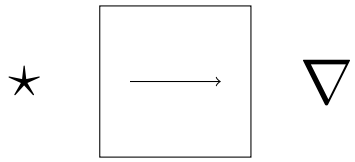

FIG. 10. Tomography problem used to illustrate the quantum inversion algorithm. It is composed of a single velocity cell and a single pair of source and receiver. The source is the star and the receiver the triangle. Only one traveltime is inverted.

model and do forward modelling and evaluate the residual. If the residual is not good enough the gradient and possibly the Hessian are calculated to update the velocity model and start another iteration.

The quantum strategy is to do forward modelling in a superposition of all the velocity models and evaluate the residuals. The smallest residual is used to increase the probability of its corresponding velocity model of being read. There is no need to calculate the gradient or the Hessian and the model space can be explored completely.

Blackbox quantum circuit

A blackbox version of the main quantum circuit is shown in Figure 11. It consists of horizontal lines for each qubit and boxes that will be explained later.

On the left are the qubits and bits used in the computation. We denote each group of qubits as a *quantum register*. Raypath distances are in register $d = [d_1, d_0]$. The superposition of velocity models is in register $v = [v_1, v_0]$. The modelled traveltimes will be placed in register $t^* = [t_3^*, t_2^2, t_1^*, t_0^*]$. Measured traveltimes are input in register $t = [t_3, t_2, t_1, t_0]$. There is a temporary qubit $temp$ and two classical bits out where the output velocity model will be read to at the end.

The box *setup* initializes the input data. Is in charge of writing the known physical quantities to register d and t, and also of initializing the velocity models superposition in register v.

The box labelled *computation* calculates the traveltimes t^* from the input raypath distances d and superposition of velocity models v, then it computes the residuals using the measured traveltimes t, and finally, it reverses the phase of the residual corresponding to the answer velocity model v_m.

The box labelled *mirror* implements the quantum phase amplification circuit described in the previous section. It amplifies the probability to be read of the state whose phase was reversed by the the computation stage, that is, the state corresponding to the answer velocity model v_m. The final stage reads the velocity qubits into the classical bits out.

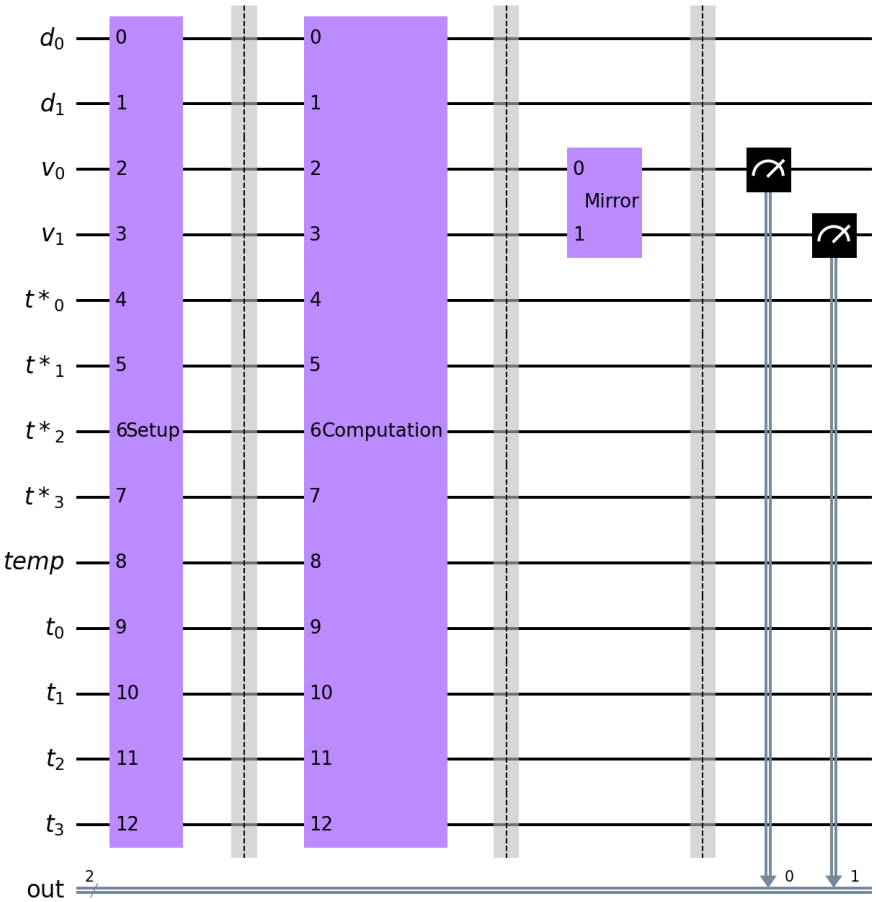

FIG. 11. Blackbox traveltime tomography quantum circuit. Horizontal lines are the qubits and bits. Quantum register $d = [d_1, d_0]$ has the raypath distances. Register $v = [v_1, v_0]$ contains the velocity models in superposition. Register $t^* = [t_3^*\text{-}t_0^*]$ will have the calculated traveltimes. Register $t = [t_3\text{-}t_0]$ has the measured traveltimes. There is one temporary qubit $temp$ and two classical bits out where the output velocity model will be read to. The setup box sets the input data and velocity models superposition. The Computation box calculates the traveltimes and residuals, then it reverses the phase of one velocity model. The mirror box implements the quantum phase amplification circuit described in the previous section. At the end, a velocity model is read into classical bits out.

#	Step	t	t^*	v	t
1	Initial	$\|0\rangle$	$\|0\rangle$	$\|0\rangle$	$\|0\rangle$
2	Setup	$\|-4\rangle$	$\|0\rangle$	$\frac{1}{2}\|0\rangle + \frac{1}{2}\|1\rangle + \frac{1}{2}\|2\rangle + \frac{1}{2}\|3\rangle$	$\|2\rangle$
3		$\|-4\rangle$	$\frac{1}{2}\|00\rangle + \frac{1}{2}\|01\rangle + \frac{1}{2}\|02\rangle + \frac{1}{2}\|03\rangle$		$\|2\rangle$
4	Mult	$\|-4\rangle$	$\frac{1}{2}\|00\rangle + \frac{1}{2}\|21\rangle + \frac{1}{2}\|42\rangle + \frac{1}{2}\|63\rangle$		$\|2\rangle$
5	Add	$\|-4\rangle$	$\frac{1}{2}\|-40\rangle + \frac{1}{2}\|-21\rangle + \frac{1}{2}\|02\rangle + \frac{1}{2}\|23\rangle$		$\|2\rangle$
6	Phase $e^{i\pi}$	$\|-4\rangle$	$\frac{1}{2}\|-40\rangle + \frac{1}{2}\|-21\rangle - \frac{1}{2}\|02\rangle + \frac{1}{2}\|23\rangle$		$\|2\rangle$
7	Add†	$\|-4\rangle$	$\frac{1}{2}\|00\rangle + \frac{1}{2}\|21\rangle - \frac{1}{2}\|42\rangle + \frac{1}{2}\|63\rangle$		$\|2\rangle$
8	Mult†	$\|-4\rangle$	$\frac{1}{2}\|00\rangle + \frac{1}{2}\|01\rangle - \frac{1}{2}\|02\rangle + \frac{1}{2}\|03\rangle$		$\|2\rangle$
9		$\|-4\rangle$	$\|0\rangle$	$\frac{1}{2}\|0\rangle + \frac{1}{2}\|1\rangle - \frac{1}{2}\|2\rangle + \frac{1}{2}\|3\rangle$	$\|2\rangle$

Table 1. Evolution of the quantum circuit states during the setup and computation stages. The columns contain, from left to right, the step number, the step name and the contents of the quantum registers t, t^*, v and d. Some registers are joined together for explanatory motives.

Registers

This is a small instance of the traveltime tomography problem and we are going to use small numbers for the physical quantities. The number of qubits of each register and its encodings indicate the set of possible values that we can process with this configuration. We are going to use only integer numbers to simplify the explanation.

Traveltimes d and velocity models v are two qubits wide and they are non negative quantities. This means the allowed values of these registers are $0, 1, 2$ and 3.

Although the traveltimes are also non negative quantities, we are going to subtract them to calculate the residuals and possibly getting negative values. For this reason the quantum registers t and t^* are signed integers in two's complement encoding. Both have 4 qubits so their allowed values are in the range $[-8, 7]$.

Detailed circuit and example execution

Figure 12 shows a more detailed version of the quantum algorithm of Figure 11. The four parts are separated by vertical dashed lines. Table 1 contains the quantum circuit states during the setup and computation stages. The columns are the step number, the step name and the four quantum registers t, t^*, v and d. We will be referring to each row of this table.

The problem instance that we are going to solve has a raypath of 2 distance units and a measured traveltime of 4 time units. The answer is 2 slowness units.

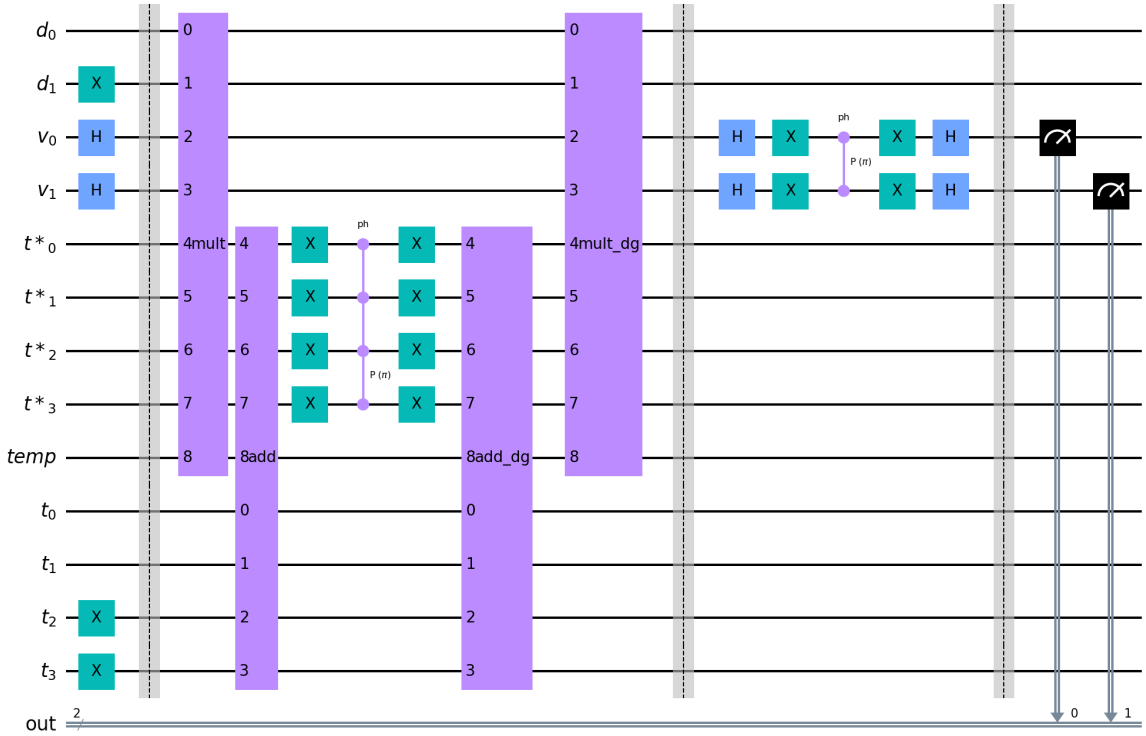

FIG. 12. Detailed version of the blackbox quantum circuit of Figure 11. Before the first vertical dashed line is the setup stage. Between the first and second vertical dashed lines is the computation stage. Between the second and third vertical dashed lines is the mirror stage. The velocity reading is at the end.

Setup stage

By convention the initial of all qubits before the setup stage are $|0\rangle$ as step 1 of table 1 shows. To input these quantities the setup stage sets $d = |2\rangle = |10\rangle$ and $t = |-4\rangle = |1100\rangle$ using not gates in the appropriate qubits. This stage also sets the velocity register to an equal superposition of all the velocity models by using two Hadamard gates:

$$\begin{aligned} |v_1 v_0\rangle = |00\rangle \xmapsto{HH} & \left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right)\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) \\ = & \frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle \\ = & \frac{1}{2}|0\rangle + \frac{1}{2}|1\rangle + \frac{1}{2}|2\rangle + \frac{1}{2}|3\rangle \end{aligned} \tag{17}$$

Step 2 of the table shows the state after the setup stage. To ease the explanation we distribute the register t^* state over the superposition in register v. The $|0\rangle$ in t^* is simply attached in front of every term in v as the result of this distribution. Step 3 of the table depicts this.

Computation stage

The next step consists in performing the forward modelling in superposition. In our simple tomography instance this means to multiply the known value of the raypath lengths in d by the superpositions of the slowness models in v to obtain a superposition of calculated traveltimes in t^*.

The blackbox labelled *mult* in Figure 12 performs this multiplication between the mentioned registers. Details about the implementation of this quantum arithmetic circuit can be found in Häner et al. (2018). Step 4 shows the results of this step. For example, this circuit multiplies $|2\rangle$ in d with the 2 in term $\frac{1}{\sqrt{2}}|02\rangle$ of combined register t^*v obtaining $\frac{1}{\sqrt{2}}|42\rangle$.

The blackbox with the label *add* in Figure 12 calculates the residuals. It subtracts the measured traveltimes in t from the just calculated traveltimes in t^*, storing the answer in t^* itself. Information about this quantum arithmetic circuit is in Cuccaro et al. (2004). Step 5 of table 1 has the state after this step is performed. For example, -4 is added to the first part of the term $\frac{1}{\sqrt{2}}|21\rangle$ resulting in $\frac{1}{\sqrt{2}}|-21\rangle$.

The centre of the computation circuit in the same figure is in charge of flipping the phase of the state with the correct answer. In this idealized noise free problem instance the correct model is the one with residual equal to $|0\rangle = |0000\rangle$. Also remember that the phase gates are only applied to terms where all the qubits are $|1\rangle$. This means the circuit must transform this residual in one composed of just ones by flipping every qubit, then applying a π phase gate and finally flipping all the qubits again. This is shown in step 6 of table 1. Notice that the only the term $\frac{1}{\sqrt{2}}|02\rangle$ changes phase because it first part is all $|1\rangle$ after flipping:

$$|02\rangle = |0\rangle|2\rangle = |0000\rangle|2\rangle \overset{not^4}{\longmapsto} |1111\rangle|2\rangle \overset{pha(\pi)}{\longmapsto} -|1111\rangle|2\rangle \overset{not^4}{\longmapsto} -|02\rangle \tag{18}$$

while the other terms like $\frac{1}{\sqrt{2}}|23\rangle$ does not change phase:

$$|23\rangle = |2\rangle|3\rangle = |0010\rangle|3\rangle \overset{not^4}{\longmapsto} |1101\rangle|3\rangle \overset{pha(\pi)}{\longmapsto} |1101\rangle|3\rangle \overset{not^4}{\longmapsto} |23\rangle \tag{19}$$

Inverse computation stage

Up to this point we have the residuals in the quantum circuit state, but the only important information is the phase that marks the correct velocity model. In quantum computing is normal to undo the computation before proceeding further because the different states can be *entangled* Johnston (2019) and this can damage the following stages.

To break those possible entanglements we perform all the computations in the opposite order, except for the phase change. The two boxes labelled sum_dg and $mult\_dg$ are the inverses of the boxes sum and $mult$. The steps 7 and 8 in table 1 show this inverse computing. Notice that all states return to their original values but the phase attached to the term with the correct velocity model is preserved. In step 9 of the same table we factor the register t^*.

Mirror stage and reading

Now that the state of the circuit has the term with the correct velocity model with a phase different from the other terms, we apply the mirror circuit to the register v.

Figure 13 shows this. Before the computation stage the register v is in an equal superposition of velocity models as the top left part of the figure shows. After the computation stage the amplitude of term $|2\rangle$ changes to $-\frac{1}{2}$ like in the top right part of the figure.

The mean of the amplitudes is $(\frac{1}{2} + \frac{1}{2} - \frac{1}{2} + \frac{1}{2})/4 = \frac{1}{4}$. The bottom left part shows the mean of all the amplitudes as a dashed line. The mirror circuit reflects the amplitudes using the mean as the reflection axis. In this case the amplitude of term $|2\rangle$ becomes 1 while the others vanish. At this point, a reading of the register v would get the velocity model $|2\rangle$ that is the correct answer to this problem instance.

We are lucky in this case that only one iteration of the computation and mirror stages are needed to have 100% chances of reading the correct velocity model. In bigger models the number of iterations needed is proportional to the square root of the number of models Johnston (2019).

In this way we computed in superposition using a quantum algorithm the one cell traveltime tomography problem. Bigger problems can be solved in the same way, but the will

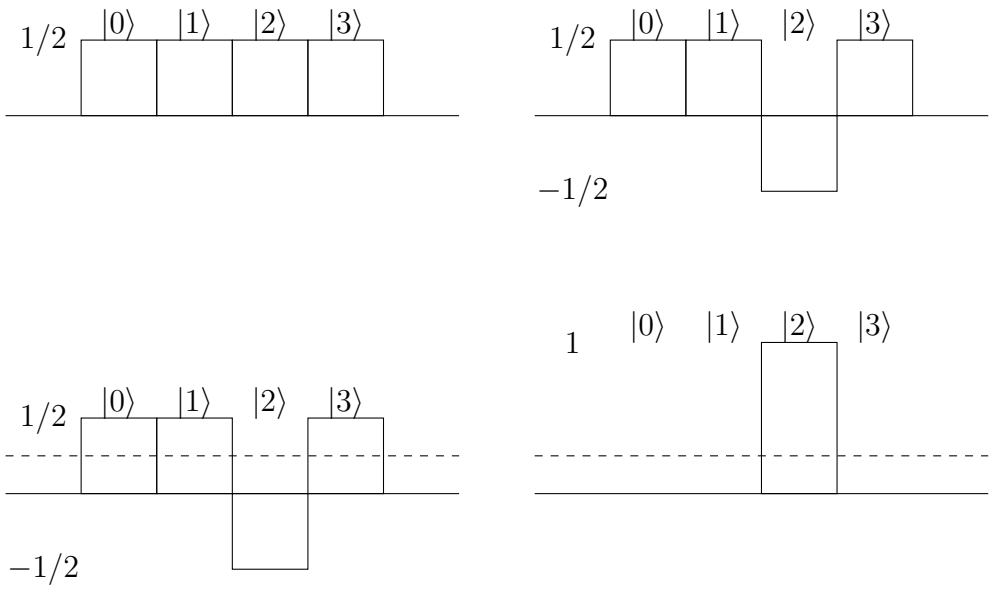

FIG. 13. Mirror stage effect on the velocity models superposition in register v. Top left is v state before computation stage. The four possible models have the same amplitude. Top right is after the computation stage. Model answer $|2\rangle$ amplitude is reversed. On the bottom left, the dashed line is the mean amplitude. Mirror stage reflects amplitudes around the mean on the bottom right. All amplitudes have vanished except for the amplitude of $|2\rangle$ that is 1.

need more qubits to have more cells, more measured traveltimes and more numeric precision. Also more iterations will be needed. However, no gradient or Hessians will be necessary and the model space can be explored more thoroughly.

DISCUSSION AND CONCLUSIONS

The main attractive of using quantum computation for solving inverse problems is using quantum superposition. With quantum superposition we can calculate the residuals of a set of velocity models in parallel. The main drawback is that it is not possible to read all of them but just one, depending on the coefficients of the quantum superposition. It is necessary to increase the probability of one term, the one corresponding to the inversion solution, by applying the mirror or Grover's iteration several times. This approach is the same used in quantum searching algorithms (Grover, 1997) and requires around $O(\sqrt{N})$ iterations when N models are simulated in superposition. It is still a huge advantage because it allows to explore the model space with the same number of iterations of a single gradient based inversion.

In addition, the actual computation done in superposition is simpler than the classical one because there is no need to compute gradients or Hessians. Only forward modelling and residual calculation are needed, and this simplifies the quantum operations.

Classical computers use floating point formats to encode the inverted models and perform the optimization. In the quantum computing case, the number of available qubits, that is scarce for the moment, constraints the range and precision of the numbers in the quantum calculations. We used very few qubits and a very small problem to test the concept of quantum inversion and because simulating a quantum computer with a classical one

is computationally expensive. However, there are reports that predict up to thousands or even millions of qubits by the end of this decade (IBM Computing, 2021a; Google, 2022). Although this refers to physical qubits and not logical qubits, the increase in the size of the applications will be worthwhile.

The traveltime tomography with fixed raypaths required a simple forward modelling. More complex modellings, like full wave simulations, are still a challenge due to their complexity. In quantum computing every operation should be reversible, because quantum systems are, and this means that more effort has to be spent in coding numerical computations in quantum terms. Future quantum compilers can leverage this issue, though.

In our proof of concept problem we found the answer model by looking at the residual that was identically zero. In real applications where noise is present this is not possible. With noise we have to change the residual calculation to include some kind of norm and look for the smallest residual. For the first part, it is possible to calculate absolute values with quantum gates (Johnston, 2019). For the second, there are quantum searching algorithms that find the minimum of a numerical set (Dürr and Hoyer, 1996) that are very similar to the mirror operation.

ACKNOWLEDGEMENTS

We thank the sponsors of CREWES for continued support. This work was funded by CREWES industrial sponsors and NSERC (Natural Science and Engineering Research Council of Canada) through the grant CRDPJ 543578-19. Research at the CaMI field site is supported in part by the Canada First Research Excellence Fund, through the Global Research Initiative at the University of Calgary and the CaMI.FRS Joint Industry Project. The first author (JM) is also supported by Canada First Research Excellence Fund, through the Global Research Initiative at the University of Calgary

CODE AND SIMULATION

The quantum traveltime tomography algorithm was coded in Python using the IBM Qiskit libraries IBM Computing (2021b). The first block of code imports all the packages needed to create the quantum circuit and run the simulation.

```
from qiskit import QuantumCircuit, QuantumRegister, ClassicalRegister,
                   assemble, Aer, transpile
from qiskit.circuit.library import HRSCumulativeMultiplier,
                                   CDKMRippleCarryAdder, MCPhaseGate
from qiskit.visualization import plot_histogram
from math import pi
```

The second block of code creates the quantum and classical registers that compose the quantum circuit.

```
# Register for distances
```

```
d = QuantumRegister(2,'d')

# Register for velocities
v = QuantumRegister(2,'v')

# Register for calculated traveltimes
ct = QuantumRegister(4,'t*')

# Temporary register for arithmetic
temp = QuantumRegister(1,"temp")

# Register for observed traveltimes
t = QuantumRegister(4,'t')

# Register for output velocity model
outv = ClassicalRegister(2,'out')
```

The third block of code instantiates from Qiskit the multiplication, addition and multiple controlled gates used in the computation and mirror stages.

```
# Multiplication circuit
mult = HRSCumulativeMultiplier(2,name="mult")

# Addition circuit
add = CDKMRippleCarryAdder(4,kind='fixed',name="add")

# Controlled phase inversion gate
ph = MCPhaseGate(pi,3,'ph')

# Controlled phase inversion gate for Mirror circuit
groover = MCPhaseGate(pi,1,'ph')
```

The fourth block of code assembles the quantum traveltime tomography circuit stage by stage. At the end it prints the graphical representation shown in Figure 12.

```
# Main circuit
qc = QuantumCircuit(d,v,ct,temp,t,outv)

# Fixed distance inputs: 2
qc.x(d[1])

# Observed times: -4 = 1100_b
qc.x(t[2])
```

```
qc.x(t[3])

# Velocity model superposition
qc.h(v)
qc.barrier()

# Multiplication d*v
qc.append(mult,range(9))

# Add measured and calculated times
qc.append(add,[9,10,11,12,4,5,6,7,8])

# Invert bits
qc.x(ct)

# Phase change
qc.append(ph,[4,5,6,7])

# Invert bits
qc.x(ct)

# Undo adding measured and calculated times
qc.append(add.inverse(),[9,10,11,12,4,5,6,7,8])

# Undo multiplication d*v
qc.append(mult.inverse(),range(9))

qc.barrier()

# Grover mirror
qc.h(range(2,4))
qc.x(range(2,4))
qc.append(groover,range(2,4))
qc.x(range(2,4))
qc.h(range(2,4))

# Read velocities into classical bits
qc.measure(v,outv)

qc.draw(output='mpl',scale=2)
```

The fifth block of code creates the simulator, compiles the circuit and runs the simulation. At the end it gathers the results and print the histogram of Figure 14. This histogram shows the final probabilities of the velocity models that were read into the classical bits. the model $|10\rangle = |2\rangle$ has probability 1, meaning that the circuit produces with certainty the correct velocity model.

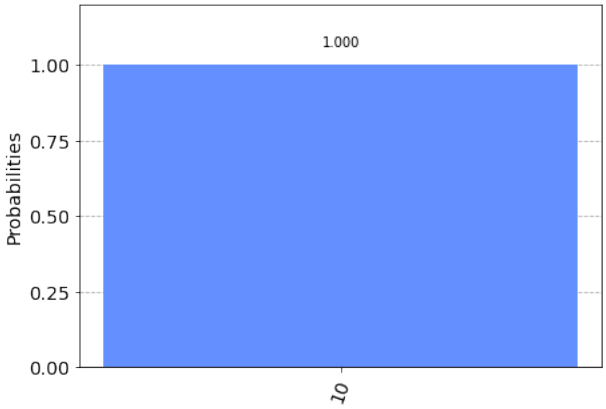

FIG. 14. Simulation result histogram. It shows the final probabilities of the velocity models read into the classical bits. The velocity model $|10\rangle = |2\rangle$ probability is 1 meaning that the circuit produces with certainty the correct velocity model.

```
# Instantiate simulator
sim = Aer.get_backend("aer_simulator")

# Compile and assemble circuit
qc_trans = transpile(qc,sim)
qobj = assemble(qc_trans)

# Run
result = sim.run(qobj).result()

# Get results and plot
counts = result.get_counts()
plot_histogram(counts)
```

REFERENCES

Albino, A., Pires, O., Ferreira de Souza, R., Santos, P., Neto, A., and Sperandio Nascimento, E. G., 2022, Employing gate-based quantum computing for traveltime seismic inversion.

Alulaiw, B., and Sen, M. K., 2015, Prestack Seismic Inversion by Quantum Annealing, 3507–3511, https://library.seg.org/doi/pdf/10.1190/segam2015-5831164.1.
URL https://library.seg.org/doi/abs/10.1190/segam2015-5831164.1

Cuccaro, S., Draper, T., Kutin, S., and Moulton, D., 2004, A new quantum ripple-carry addition circuit: ArXiv, **abs/qiant-ph/0410184**.

de Falco, D., and Tamascelli, D., 2011, An introduction to quantum annealing: RAIRO - Theoretical Informatics and Applications, **45**.

Deutsch, D., and Penrose, R., 1985, Quantum theory, the church–turing principle and the universal quantum computer: Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences, **400**, No. 1818, 97–117, https://royalsocietypublishing.org/doi/pdf/10.1098/rspa.1985.0070.
URL https://royalsocietypublishing.org/doi/abs/10.1098/rspa.1985.0070

Dürr, C., and Hoyer, P., 1996, A quantum algorithm for finding the minimum: CoRR, **quant-ph/9607014**.

Google, 2022, Our quantum computing journey.
URL https://quantumai.google/learn/map

Greer, S., and O'Malley, D., 2020, An approach to seismic inversion with quantum annealing, 2845–2849, https://library.seg.org/doi/pdf/10.1190/segam2020-3424413.1.
URL https://library.seg.org/doi/abs/10.1190/segam2020-3424413.1

Grover, L. K., 1997, Quantum mechanics helps in searching for a needle in a haystack: Physical Review Letters, **79**, No. 2.
URL https://www.osti.gov/biblio/549697

Häner, T., Rötteler, M., and Svore, K. M., 2018, Optimizing quantum circuits for arithmetic: ArXiv, **abs/1805.12445**.

IBM Computing, 2021a, IBM's roadmap for building an open quantum software ecosystem.
URL https://research.ibm.com/blog/quantum-development-roadmap

IBM Computing, 2021b, Qiskit: An open-source framework for quantum computing.

Johnston, E. R., 2019, Programming quantum computers : essential algorithms and code samples: O'Reilly, Beijing, first edition. edn.

Moradi, S., Trad, D., and Innanen, K. A., 2018, Quantum computing in geophysics: Algorithms, computational costs, and future applications, 4649–4653, https://library.seg.org/doi/pdf/10.1190/segam2018-2998507.1.
URL https://library.seg.org/doi/abs/10.1190/segam2018-2998507.1

Preskill, J., 2021, Quantum computing 40 years later.
URL https://arxiv.org/abs/2106.10522

Sarkar, R., and Levin, S. A., 2018, Snell tomography for net-to-gross estimation using quantum annealing, vol. All Days of *SEG International Exposition and Annual Meeting*, sEG-2018-2998409, https://onepetro.org/SEGAM/proceedings-pdf/SEG18/All-SEG18/SEG-2018-2998409/1323383/seg-2018-2998409.pdf.
URL https://doi.org/10.1190/segam2018-2998409.1

Shor, P. W., 1997, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer: SIAM J. Comput., **26**, No. 5, 1484–1509.
URL https://doi.org/10.1137/S0097539795293172

Simon, D., 1994, On the power of quantum computation, *in* Proceedings 35th Annual Symposium on Foundations of Computer Science, 116–123.

Souza, A. M., Martins, E. O., Roditi, I., Sá, N., Sarthour, R. S., and Oliveira, I. S., 2022, An application of quantum annealing computing to seismic inversion: Frontiers in Physics, **9**.
URL https://www.frontiersin.org/articles/10.3389/fphy.2021.748285

van der Linde, S., 2021, Quantum annealing for seismic imaging: Exploring quantum annealing possibilities for residual statics estimation using the d-wave advantage system and hybrid solver: M.Sc. thesis, Delft University of Technology.