Understanding and compensating for noise on IBM quantum computers

Scott Johnstun and Jean-François Van Huele

Citation: American Journal of Physics **89**, 935 (2021); doi: 10.1119/10.0006204
View online: https://doi.org/10.1119/10.0006204
View Table of Contents: https://aapt.scitation.org/toc/ajp/89/10
Published by the American Association of Physics Teachers

Understanding and compensating for noise on IBM quantum computers

Scott Johnstun[a] and Jean-François Van Huele[b]
Department of Physics and Astronomy, Brigham Young University, Provo, 84602 Utah

(Received 10 September 2020; accepted 21 August 2021)

Quantum algorithms offer efficient solutions to computational problems that are expensive to solve classically. Publicly available quantum computers, such as those provided by IBM, can now be used to run small quantum circuits that execute quantum algorithms. However, these quantum computers are highly prone to noise. Here, we introduce important concepts of quantum circuit noise and connectivity that must be addressed to obtain reliable results on quantum computers. We utilize several examples to show how noise scales with circuit depth. We present Simon's algorithm, a quantum algorithm for solving a computational problem of the same name, explain how to implement it in IBM's Qiskit platform, and compare the results of running it both on a noiseless simulator and on physical hardware subject to noise. We discuss the impact of Qiskit's transpiler, which adapts ideal quantum circuits for physical hardware with limited connectivity between qubits. We show that even circuits of only a few qubits can have their success rate significantly reduced by quantum noise unless specific measures are taken to minimize its impact.

https://doi.org/10.1119/10.0006204

I. INTRODUCTION

In recent years, our understanding of the usefulness of quantum mechanics in the computational and information theory has expanded significantly. From the inception of quantum computing in 1985 when Richard Feynman adapted the idea of reversible classical computing and applied it to quantum systems,[1] the field has grown to become a mainstream research area in the 21st century. Peter Shor showed why quantum computation is more than a frivolous link between the otherwise distinct fields of computer science and quantum mechanics when he published an algorithm using quantum mechanics to factor large numbers exponentially faster than a classical computer.[2,3] Quantum algorithms, therefore, have consequences in the fields of cryptography and information security. Since then, other important quantum algorithms have been developed, such as Grover's efficient quantum algorithm for searching a database;[4,5] Harrow, Hassidim, and Lloyd's algorithm for solving linear systems of equations;[6] and quantum algorithms for use in speeding up machine learning computations.[7] As society evolves to become increasingly dependent on information, it is clear that efficient quantum algorithms can play a major role in optimizing the computations we must perform to acquire and process the desired information.

In addition to these theoretical developments, advances have been made in producing the physical hardware necessary to run these algorithms. Companies, such as IBM, Google, and others,[8] have utilized Josephson junctions[9] to manufacture quantum computers with as many as 53 qubits. At the end of 2019, Google claimed to have achieved quantum supremacy by using such a quantum computer to compute laser scattering distributions in less time than a classical supercomputer would be able to.[10] It is safe to say that we are on the cusp of many more developments in quantum computing hardware.

Quantum computation has commercial applications with financial consequences due to its influence in cryptography, which is essential for companies that deal with communication and information security. The United States government has passed the National Quantum Initiative Act,[11] allocating funding to universities and companies in an effort to stimulate the development of quantum information technology. As of the writing of this paper, the National Institute of Science and Technology is working to replace current encryption algorithms with ones that are quantum secure.[12]

With these exciting developments in software, hardware, and applications, the literature on quantum algorithms has grown correspondingly larger. We refer the interested reader to the quantum information American Journal of Physics Resource Letter for a summary of the subject up to 2016.[13] Textbooks on the subject introduce quantum algorithmic design and cover many of the fundamental algorithms in quantum computation.[16,17] Online textbooks and tutorials are also widely available online.[18–20] Some authors have proposed classroom experiments and simulations to educate students on quantum computation and algorithms,[21] and others have presented methods for introducing students of computer science to quantum algorithms.[22] Other recent treatments of this topic in AJP include a computational project on quantum computing[14] and a realization of the Deutsch algorithm using optical devices.[15] Research groups are using publicly available quantum computers built and maintained by IBM to run and test algorithms.[23–25] In May 2020, the National Science Foundation initiated an effort with the White House Office of Science and Technology to develop additional resources for teaching quantum information science and technology.[26] It is clear that quantum computing is slowly yet surely becoming a staple of the undergraduate physics experience.

In this article, we call attention to the important issue of the difference in performance between quantum circuits in simulators and quantum circuits implemented on physical quantum computers. We specifically apply this performance comparison to Simon's algorithm[27] as it runs on currently available quantum computers. Our illustration is appropriate for introducing discrepancies between the results of simulations of quantum computers and experiments on physical

quantum computers. Our intention is not to provide a comprehensive introduction to quantum computation, since that is available elsewhere;[16–18] instead, we want to guide students towards an understanding of how the physical limitations of real devices affect the success rate of quantum algorithms.

The paper is organized as follows. In Sec. II, we describe sources of noise in IBM quantum computers using a few examples of simple quantum circuits. In Sec. III, we introduce an example problem, Simon's problem, that can be solved more efficiently with a quantum algorithm than a classical algorithm. The algorithmic solution is detailed in Sec. IV, and Sec. V shows the implementation and results of running the algorithm on physical quantum computers in comparison to simulations. In Sec. VI, we discuss several lessons to be learned from our experiments and analysis.[28] We include three suggested problems with solutions in the supplementary material.[29] In the process, we also provide the PYTHON code used to setup and solve Simon's problem and generate data for readers to familiarize themselves with it, to reproduce our algorithmic results, and to expand upon them.[30]

II. NOISE AND TRANSPILATION IN QUANTUM COMPUTERS

In spite of the significant technical progress of quantum computers in recent years, they are still subject to various types of noise. IBM claims that errors due to such noise are fundamental.[33] We consider noise to be any undesired source that changes the quantum system. Because of the prevalence of noise, any quantum algorithm that is to be implemented and put to use in real-life scenarios must be able to perform its task with a high probability of success despite the presence of such noise. It has been shown that an approximation to a quantum algorithm can perform better than the exact version when noise is present: In the case of the quantum algorithm known as the quantum fourier transform (QFT), analytical methods have indicated that in the presence of decoherence, an approximation of the algorithm can provide better performance than its full version.[34] Including random gate defects in numerical calculations has also revealed that the approximate QFT can still perform its task to an acceptable degree of success in spite of such defects.[37]

Noise can come from systematic sources, such as noise introduced by hardware imperfections like incorrect pulse timing. It can also come from stochastic sources, such as thermal noise (also referred to as Johnson–Nyquist noise) causing voltage and current fluctuations proportional to temperature; quantum noise from fluctuations in the phase and amplitude of the physical qubit that have an effect even at zero temperature; and classical 1/f noise from fluctuation in local electromagnetic fields, which causes dephasing in qubits.[31,32]

Public access to physical available quantum computers is provided by IBM through the PYTHON package Qiskit,[38] which provides an excellent opportunity for students and educators to build and execute quantum circuits that implement quantum algorithms. Results from these experiments can easily be compared to the results of Qiskit's QASM simulator,[39] providing excellent insight into the performance of quantum algorithms on current quantum computers. In this paper, we use Qiskit to run our simulations and perform experiments on IBM quantum computers. A guide on installing this software for personal or classroom use can be found online.[40] A tutorial series specific to coding quantum algorithms is also available online.[19,20] We will not detail setting up the environment since instructional resources are plentiful online.[18,41,42]

In physical quantum computers, including those provided by IBM, errors tend to scale with circuit depth. For our purposes, a circuit's depth is defined by the number of quantum gates it contains. Each gate in a quantum computer has an error rate determined by the qubit it acts on. In addition, gates that act on two qubits suffer from errors that depend on both physical qubits being used. Error rates for two-qubit gates are typically higher than those for single-qubit gates, so a circuit made of two-qubit gates will experience more noise than a circuit of the same number of single-qubit gates.

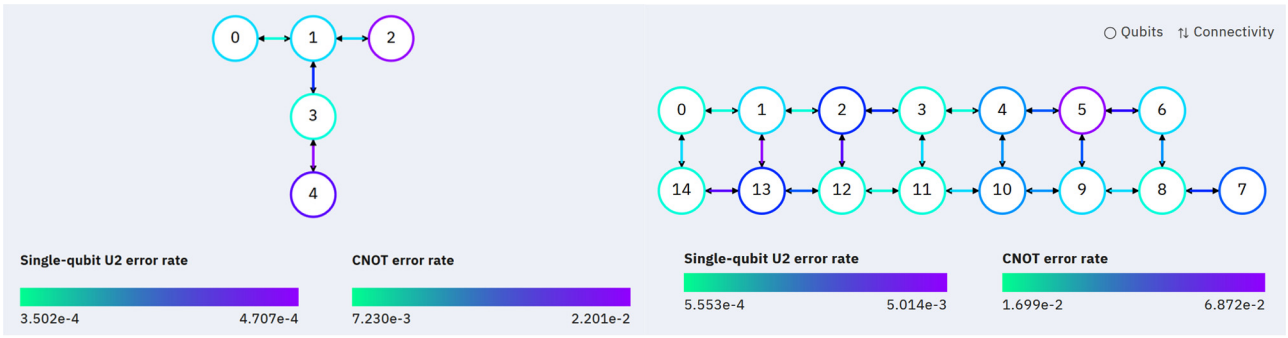
A. Transpilation

The limited connectivity between qubits in physical quantum computers is another challenge preventing two-qubit gates from acting on any arbitrary pair of qubits. Figure 1 shows the connectivity of two quantum computers at IBM: the five-qubit computer `ibmq_london` and the 15-qubit computer `ibmq_melbourne`. Each qubit is represented by a node, and connections between qubits are represented by edges. We can see that the nodes are not fully connected, indicating that some pairs of qubits cannot directly communicate. For example, a two-qubit gate cannot be directly applied on qubits 0 and 2. However, limited connectivity is not an issue that prohibits algorithmic design.[43] Two-qubit gates can be chained between intermediate qubits that *are* connected in a way that the desired operation is applied on two unconnected qubits, and the intermediate qubits are returned to their original state. However, this increases the circuit depth, which increases the error rate of the circuit when run on physical devices.

Qiskit performs the circuit transformations necessary to ensure that the software description of a quantum circuit can be implemented on a physical quantum system through a process called transpilation. This includes the aforementioned fix for limited connectivity as well as a decomposition of common gates into sequences of U2 and controlled-NOT (CNOT) gates, which we refer to as U_2 and cX gates, respectively. The U_2 and cX gates are very important, because they are two gates that form part of a universal set.[44] This means that for any input state, we can build a circuit with gates in this set that maps it to any possible output state. The U_2 gate is a single-qubit gate parameterized by two angles ϕ and λ. When acting on the single-qubit computational basis $\{|0\rangle, |1\rangle\}$, U_2 takes on the form

$$U_2(\phi,\lambda) = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & -e^{i\lambda} \\ e^{i\phi} & e^{i(\phi+\lambda)} \end{bmatrix} \equiv \text{—}\boxed{U_2(\phi,\lambda)}\text{—}, \quad (1)$$

where the symbol on the right depicts the gate as it appears on circuit diagrams per convention. This gate can manipulate a single qubit from one state to another. The cX gate acts on the two-qubit computational basis $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$, which is the tensor product of the single-qubit computational basis with itself. In this basis, cX takes on the form



(a) (b)

Fig. 1. (Color online) Qubit connectivity of `ibmq_london` and `ibmq_melbourne` as published by IBM. The colored circles in the figures indicate the error rate of single-qubit gates applied on the circled qubit. The colored arrows between pairs of qubits indicate the error rate of cX gates applied between the pair. Note the limited connectivity between qubits. In Sec. V D, we will see how transpilation issues can be overcome even on `ibmq_melbourne`. Image obtained from the IBM quantum experience (Ref. 47).

$$cX = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \equiv \quad . \tag{2}$$

Note that we have used Qiskit's little endian convention, which associates the leftmost (more significant) digits in the ket with lower qubits on the circuit diagram (higher qubit index). The qubit with a black dot on the circuit diagram is called the control qubit, and the qubit with a circled cross is the target qubit. The action of a cX gate is to apply a NOT (X) gate to the target qubit in the subspace where the control qubit is in the $|1\rangle$ state. For example, using the circuit in Eq. (2), $cX|10\rangle = |10\rangle$ and $cX|01\rangle = |11\rangle$, since the left digit in the ket is the target qubit and the right digit is the control qubit

$$X = \begin{bmatrix} 01 \\ 10 \end{bmatrix} \equiv --\boxed{X}--. \tag{3}$$

In practice, we will see that the increase in circuit depth resulting from transpilation quickly makes the execution of quantum circuits highly error-prone, rendering their results unreliable.

B. Noise demonstration

In this section, we present a few examples to characterize errors in IBM quantum computers, both from physical noise and from algorithmic compromises performed by Qiskit's transpiler. As we progress through our examples, we will gradually increase the complexity of the circuit, both in number of qubits and number of gates, and see how noise increases simultaneously. For explanations and tutorials about setting up and running the circuits in our examples, see Chapter 2 of the Qiskit Textbook.[18]

Our first example is a simple circuit composed of a single Hadamard gate that acts on one qubit. The circuit then measures that qubit. The base circuit is shown in Fig. 2(a), while Fig. 2(b) shows the circuit after it has been transpiled to run on the real hardware. This illustrates the simplest nontrivial transpilation possible, requiring only a relabeling of the Hadamard gate

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

to $U_2(0, \pi)$. When we run the circuits depicted in Figs. 2(a) and 2(b) 1000 times on a simulator and real quantum computing device, respectively, we obtain the measurement probability distributions shown in Table I. Although the simulator's final state is a uniform superposition of the $|0\rangle$ and $|1\rangle$ states, the measurement probabilities are slightly uneven. This is due to sampling error. The experiment on the quantum computer is also subject to such sampling error, but it additionally experiences physical noise. Judging by the similarity of the simulator results and the device results, it appears that any errors resulting from physical implementation are similar in magnitude to sampling error.

Our second example features a single qubit gate and a two-qubit gate between two physically unconnected qubits. The original circuit, pictured in Fig. 2(c), includes a single cX gate, controlled by qubit 0 and acting on qubit 2. These two gates act together to entangle qubits 0 and 2. These qubits are not directly connected, as seen in Fig. 1(a). After applying the transpilation and obtaining the circuit shown in in Fig. 2(d), we see that Qiskit has added three intermediate cX gates between qubits q_1 and q_2. These three gates swap the states of the two qubits they act on. The resulting effect is that the state that qubit q_2 starts in is moved onto qubit q_1, after which it is the target of the cX gate controlled by qubit q_0. The resulting entangled state of qubits q_0 and q_2 in the simulator is theoretically the same as the entangled state of qubits q_0 and q_1 in the physical implementation, but in practice, it is expected to be slightly different due to errors.

We again ran 1000 trials of the circuits pictured in Figs. 2(c) and 2(d) on both the simulator and real device; results are shown in Table I. In this case, we can see a more significant difference between the simulator results and the experimental results: some of the $|00\rangle$ and $|11\rangle$ states have changed to $|01\rangle$ and $|10\rangle$ states.

Our final example is a simple phase estimation algorithm circuit.[45] All gates and combinations of gates that act on quantum computers are unitary operators. Unitary operators have the special property that their eigenvalues always take on the form $e^{i\varphi}$ for some real angle φ between 0 and 2π. The phase estimation algorithm is used to estimate the phase φ of any unitary operator and, thus, allow the determination of

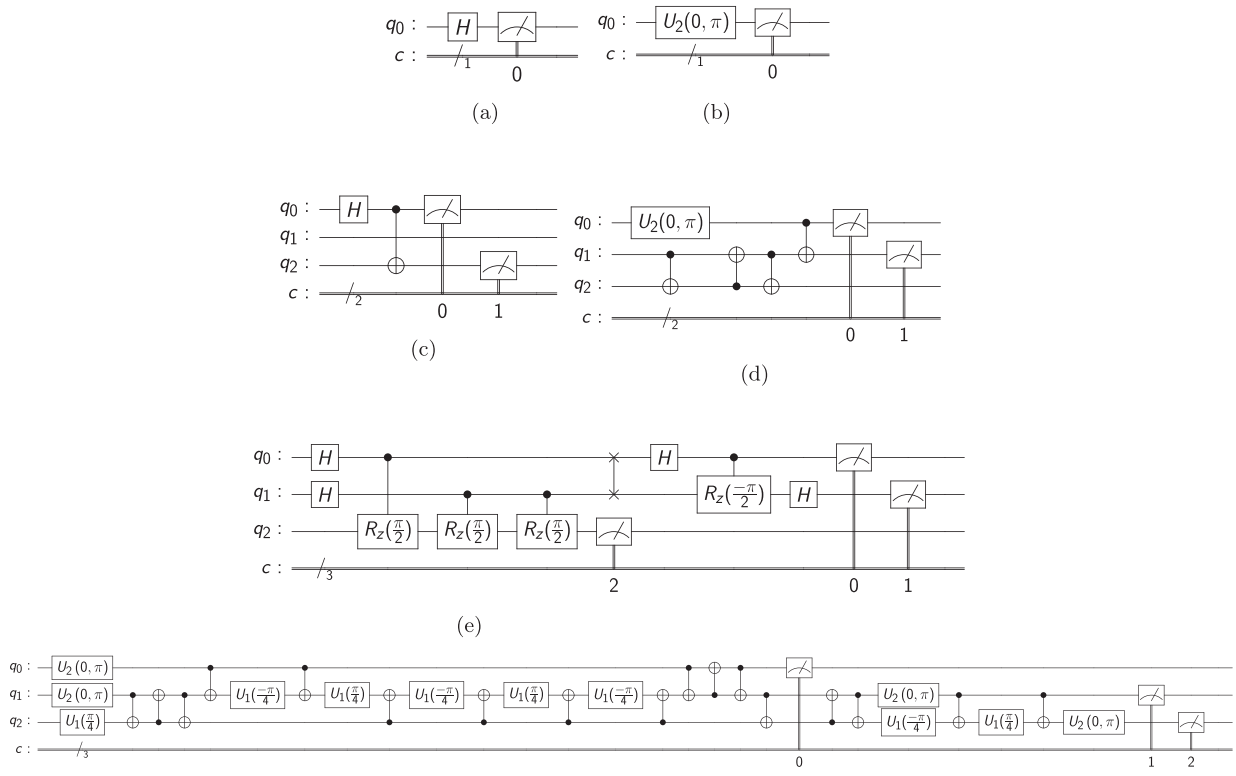


Fig. 2. Effect of transpilation on several representative circuits. (a) Base and (b) transpiled circuit for a single gate that acts on one qubit. (a) Base and (b) transpiled circuit that entangles two qubits that are not physically connected directly. (a) Base and (b) transpiled phase estimator circuit. After each circuit is run, the qubits are measured. The symbols q_0 and c_0 are the names of a qubit and a classical bit, respectively. In this and all figures in this paper, qubits are initialized to the $|0\rangle$ state and classical bits are initialized to 0.

the eigenvalue. This algorithm is incorporated within the aforementioned Shor's algorithm, as well as in algorithms used for simulation of quantum systems[46] (not to be confused with quantum simulators). With larger versions of this algorithm, the eigenvalue can be estimated to better accuracy. An in-depth tutorial on this algorithm can be found in Sec. 3.8 of the Qiskit Textbook.[18]

By comparing Fig. 2(e) and 2(f), we can see that the transpiler has significantly increased the depth of the circuit by inserting a large number of cX gates. A comparison between the simulation and experimental results after 1000 trials, shown in Table I, indicates that a large amount of noise has entered the circuit. Whereas in previous examples, we would have been able to guess the correct probability distributions from the data, in this case, the originally highly probable $|11\rangle$ state of the first two qubits in the circuit has fallen to a less than 10% probability of being observed.

III. SIMON'S PROBLEM

Simon's problem is a toy computational problem introduced in 1994 by Daniel Simon. Its solution is known as Simon's algorithm. We present it as an example of a problem

Table I. Results of the circuits shown in Fig. 1 for runs on both the simulator and the real quantum hardware.

One qubit circuit				
Outcome	$\|0\rangle$	$\|1\rangle$		
Simulation	0.482	0.518		
Experiment	0.538	0.462		
Entangling circuit				
Outcome	$\|00\rangle$	$\|01\rangle$	$\|10\rangle$	$\|11\rangle$
Simulation	0.495	0	0	0.505
Experiment	0.458	0.047	0.061	0.434
Phase estimator circuit				
Outcome	$\|000\rangle$	$\|001\rangle$	$\|010\rangle$	$\|011\rangle$
Simulation	0.405	0.088	0.077	0.430
Experiment	0.484	0.149	0.033	0.085
Outcome	$\|100\rangle$	$\|101\rangle$	$\|110\rangle$	$\|111\rangle$
Simulation	0	0	0	0
Experiment	0.152	0.055	0.024	0.018

that is solved more efficiently by a quantum algorithm than a classical algorithm.

A. Problem description

In Simon's problem, we are given a function f that maps n-bit integers to $(n-1)$-bit integers. This function has the property that there is some nonzero n-bit number a for which, if x and y are any two n-bit numbers, then $f(x) = f(y)$ if and only if $y = x \oplus a$, where $\oplus$ indicates bitwise modulo-2 addition (also known as the XOR operation). In this case, we see that $f(x) = f(x \oplus a)$, so the function is periodic under the $\oplus$ operation. The task of Simon's problem is to find the period a.

B. Classical solution

Classically, if we seek to find the period a, we must test many different values of x and keep track of x and the output $f(x)$ until we find a different input y where $f(x) = f(y)$. Once this happens, we find a by calculating $x \oplus y$. As x is an n-bit number, there are 2^n possible values to use as input, and each x has exactly one y for which $f(x) = f(y)$, so there are $2^n/2 = 2^{n-1}$ pairs of x and y values. This means that we must compute $f(x)$ for up to 2^{n-1} distinct values of x in order to find a, so the algorithm scales exponentially in n, the number of bits in a.

C. Quantum solution

In contrast, the quantum solution to this problem requires an amount of circuit runs that is only linear in n to find a. This is accomplished by putting the n input qubits into a balanced superposition of all possible states before calculating f. Putting a register into this state is a standard quantum computational procedure that allows for speedup in many problems. The result of the quantum circuit that implements Simon's algorithm requires classical postprocessing to find a, since the circuit does not provide the value of a directly. This means that the outcomes measured on the quantum computer need to be manipulated to extract the value of a. The algorithmic solution, Simon's algorithm, is detailed in Sec. IV.

IV. SIMON'S QUANTUM ALGORITHM

A. Quantum operations

We use a quantum circuit of $2n-1$ qubits with readout onto a classical register of n bits. As a reminder, a classical bit can only store a value of 0 or 1, whereas a qubit can store any normalized superposition of the $|0\rangle$ and $|1\rangle$ basis states. A schematic for the operations to be performed is presented in Fig. 3. We represent our basis states with the convention that $|x\rangle_n$ refers to n qubits whose state is the n-bit binary expansion of x. For example, $|4\rangle_3 = |1\rangle|0\rangle|0\rangle$ since $4 = 1*2^2 + 0*2^1 + 0*2^0$. The first n qubits, initialized to the state $|0\rangle_n$, are used as input qubits to f, and the next $n-1$ are used as the output and initialized to the state $|0\rangle_{n-1}$. We assume that the given function f has been provided in a perfect black box that computes the output $f(x)$ via a $(2n-1)$-qubit unitary transformation U_f. We will see in the following derivation that the input register's state changes throughout the course of the algorithm.

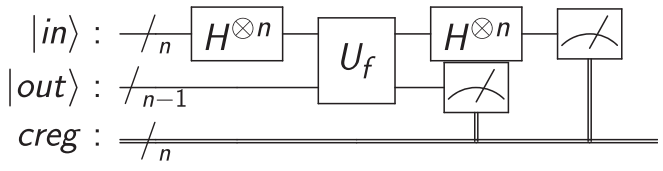


Fig. 3. Schematic for the quantum algorithm solving Simon's problem. Slashes on the horizontal lines with a number indicate a register of that many qubits. The n-qubit register is referred to as the input register and the $(n-1)$-qubit register as the output register. The lines at the bottom are an n-bit classical register.

The algorithm starts by applying a Hadamard gate to each of the n qubits in the input register to produce a normalized, balanced superposition of every state from $|0\rangle_n$ to $|2^n-1\rangle_n$

$$H^{\otimes n}|0\rangle_n = \frac{1}{2^{n/2}}\sum_{x=0}^{2^n-1}|x\rangle_n, \tag{4}$$

where the $\otimes n$ superscript represents the tensor product of the n Hadamard gates operating on the n-qubit register. We then apply U_f on the input and output registers, yielding

$$U_f\left(\frac{1}{2^{n/2}}\sum_{x=0}^{2^n-1}|x\rangle_n|0\rangle_{n-1}\right) = \frac{1}{2^{n/2}}\sum_{x=0}^{2^n-1}|x\rangle_n|f(x)\rangle_{n-1}. \tag{5}$$

We now perform a quantum measurement of the output register and record the resulting $(n-1)$-bit integer onto $n-1$bits of the classical register. The result of this measurement is irrelevant for our solution, as the values of $f(x)$ are completely independent of a. However, the act of measurement itself is critical because it collapses the superposition in the output register. Since there are exactly two inputs, $\hat{x}$ and $\hat{y} = \hat{x} \oplus a$, that correspond to any particular output $\hat{f} = f(\hat{x}) = f(\hat{y})$, this measurement leaves the circuit in the state

$$\frac{1}{\sqrt{2}}\left(|\hat{x}\rangle_n + |\hat{y}\rangle_n\right)|f(\hat{x})\rangle_{n-1}. \tag{6}$$

We can finally apply Hadamard gates to each qubit in the input register, which results[35] in an input register state of

$$H^{\otimes n}\left(\frac{1}{\sqrt{2}}\left(|\hat{x}\rangle_n + |\hat{y}\rangle_n\right)\right) = \frac{1}{2^{\frac{n+1}{2}}}\sum_{x=0}^{2^n-1}\left((-1)^{\hat{x}\cdot x} + (-1)^{\hat{y}\cdot x}\right)|x\rangle_n, \tag{7}$$

where $x \cdot y$ refers to the bitwise dot product between x and y, taken modulo 2,

$$x \cdot y = \sum_{i=0}^{n} x_i y_i \,(\mathrm{mod}\, 2).$$

If we then rewrite the coefficient of $|x\rangle_n$ as

$$\begin{aligned}(-1)^{\hat{x}\cdot x} + (-1)^{\hat{y}\cdot x} &= (-1)^{\hat{x}\cdot x} + (-1)^{(\hat{x}\oplus a)\cdot x} \\ &= (-1)^{\hat{x}\cdot x} + (-1)^{(\hat{x}\cdot x)\oplus(a\cdot x)},\end{aligned} \tag{8}$$

it becomes clear that this coefficient is nonzero only when $a \cdot x = 0$. Let $I = \{x : a \cdot x = 0\}$. We can then rewrite the state in Eq. (7) as

$$\frac{1}{2^{\frac{n-1}{2}}}\sum_{x\in I}(-1)^{\hat{x}\cdot x}|x\rangle_n, \tag{9}$$

where the summation is taken only over those x values for which $a{\cdot}x{=}0$. If we finally measure the input register and record the result onto the n-bit classical register, we are guaranteed to read out a number z that satisfies $z \cdot a = 0$. If the circuit is run m times ($m \geq n$) and n unique results are recorded, we can build a system of n modulo-2 equations, which we can solve to find a. This is the classical postprocessing.

Each of the n individual z values has a probability of $1/2^{n-1}$ of being measured. After m runs of the circuit, the probability of being able to determine a is no smaller than[36]

$$P_{min} = 1 - \frac{1}{2^{m-n+1}}. \tag{10}$$

We can, thus, determine a to a high probability with a linear number of circuit runs, or *shots*, in n; this probability increases as the quantity $m - n$ increases. For example, if $m = n + 4$, we have over 96% probability of determining a after m shots. This reveals the exponential speedup of the quantum algorithm over the classical one.

B. Classical postprocessing

After n unique z values have been obtained, we can turn the system of n equations

$$\begin{aligned} z_0 \cdot a &= 0 \,(\mathrm{mod}\ 2),\\ z_1 \cdot a &= 0 \,(\mathrm{mod}\ 2),\\ &\vdots\\ z_n \cdot a &= 0 \,(\mathrm{mod}\ 2), \end{aligned} \tag{11}$$

into a matrix equation

$$Z\mathbf{a} = \mathbf{0},$$

where the ith row of the matrix Z is the binary expansion of z_i. Applying Gaussian elimination modulo 2 then reveals a single nontrivial solution which is $\mathbf{a}$, the binary expansion of a.

V. QUANTUM ALGORITHM IMPLEMENTATION

A. Constructing the circuit

We described the transformations needed to implement Simon's algorithm in Sec. IV. In Appendix B of the supplementary material,[29] we detail the construction of the quantum circuit that performs these operations. Both the base circuits and transpiled versions of the circuits for $n = 3$ and $n = 6$ are also shown in the supplementary material. As with the examples in Sec. II B, the transpiler significantly increases the quantity of cX gates. The value $n = 6$ was chosen, because it is sufficiently small to allow Simon's problem to be solved on a quantum computer of as low as 11 qubits. We also constructed a circuit for $n = 3$, the largest n value for which Simon's algorithm can fit on a five-qubit quantum computer.

B. Results

Using Qiskit, we can run the circuits on both the simulator and the real quantum computing hardware. We perform m shots, where m ranges from three to ten for the $n = 3$ case and from 6 to 15 in the $n = 6$ case. The lower bounds three and six are chosen, because they are the minimum number of shots after which unique determination of a is possible for the respective problem sizes. The upper bounds of 10 and 15 are chosen somewhat arbitrary and simply represent a point at which further incrementing the number of shots ceases to be useful.

As a measure of success, we introduce the ratio P of trials, in which we are able to uniquely determine a in postprocessing to the total number of trials for that m value. For each value of m, we perform 1000 trials on the simulator with one a value (for $n = 3$) and 100 trials with each of sixteen different a values (for $n = 6$). The experimental results were obtained using the five-qubit computer `ibmq_ourense` for $n = 3$ and the 15-qubit `ibmq_16_melbourne` for $n = 6$.[47] These machines are publicly available and shared with other users, so in our trials we experienced queue times that ranged from a few seconds to an hour. Due to this limitation, we performed only a few trials for each m value; 20 trials of m shots were performed in order to determine the success rate of the algorithm.

The simulator and experimental results as well as a comparison with the theoretical bound from Eq. (10) are plotted in Fig. 4. The simulator results show that for a noiseless circuit, the probability of finding a approaches 1 after a sufficient number of shots. We note that for smaller values of

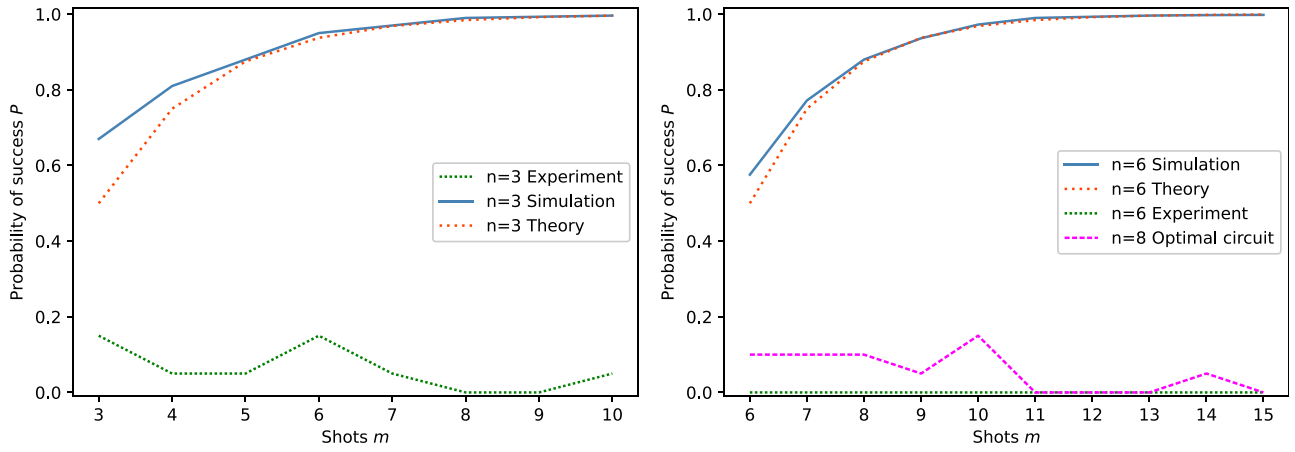


Fig. 4. (Color online) Probability of being able to determine a uniquely after m shots for $n = 3$ (left) and $n = 6$ (right). Each plot compares the theoretical probability of success with the results of running the algorithm on an ideal simulator and a real IBM quantum computing device; the right plot also shows the results for an optimized $n = 8$ circuit on a real device. For the results, each data point represents the proportion of multiple independent trials, in which a is able to be uniquely determined for the given value of m and n.

$m - n$, the probability of success is slightly better for the smaller n value. In addition, the simulation probabilities are always greater than or equal to the theoretical lower bounds, which verifies the correct performance of our implementation in ideal conditions.

The experimental results show a quite low probability of success with the $n = 3$ algorithm for all values of m; the success rate never rises above 15%, and it even reaches zero for some values of m. More significantly, the $n = 6$ algorithm was never able to complete its task for any value of m. In the next section, we will address the cause of such low probabilities.

C. Error analysis

Due to the nonzero error rate of cX gates on IBM quantum computers, we have seen that adding cX gates to the circuit causes it to perform significantly worse than expected. For our n=6 circuit, the transpiler added 57 cX gates to a circuit which originally had only 11, for a total of 68 cX gates. Even if we were to make the most optimistic assumption and take the minimum error rate of 1.669×10^{-2} (as shown in Fig. 1(b)) for *all* cX connections, the probability $P_{no\,error,cX}$ that *no* cX gates error out in the transpiled circuit would be

$$P_{no\ \ errors,cX,transpiled} = (1 - 0.016\,69)^{62} = 31.8\%,$$

compared to

$$P_{noerrors,cX,original} = (1 - 0.01669)^{11} = 98.3\%,$$

for the original, nontranspiled circuit. Since the actual error rate for most of the cX gates in our circuit is actually larger than this rate, it is practically inevitable that some cX gate will not perform its task correctly. In terms of our algorithm, this means that the U_f gate, which applies the function f to the qubits, could be affected so much that it represents an entirely different function.

We have not taken into account the single-qubit U2 error rate depicted in Fig. 1(b). (Recall that Hadamard gates are converted into the equivalent $U_2(0, \pi)$ during transpilation.) Because the circuit requires exactly $2n$ Hadamard gates and the single-qubit nature of these gates avoids any connectivity issues, the U_2 gates contribute to less error in the circuit than the cX gates. In addition, the greatest $U2$ error rate of any qubit on `ibmq_16_melbourne` was 5.014×10^{-3} at calibration time, which is an order of magnitude lower than the cX error rates. For our $n = 6$ circuit, then, the *worst case* estimate of the probability $P_{no\,error,H}$ that no H gate in the circuit has an error is

$$P_{no\ \ error,H} = (1 - 0.005\,014)^{12} = 94.1\%,$$

which is the same pre- and post-transpilation.

The quantity of cX gates added, along with the significant error rate of the cX gates, is the reason for the difference in success between running our algorithm in simulations and on a quantum computer.

D. Overcoming transpilation issues

In an attempt to confirm that the transpiler's introduction of a significant number of cX gates is responsible for the low success rates in our algorithm's performance on a quantum computer, we constructed a circuit with a U_f gate corresponding to the parameter a=10000001. This particular value of a minimizes the number of cX gates necessary, and no ancilla qubits are needed to make up for connectivity restrictions, so this circuit is expected to perform its task as well as possible for its n value. This n=8 circuit is the largest n that can fit on the 15-qubit `ibmq_16_melbourne`. Using an appropriate mapping, we produced a circuit for n=8, which is shown in the supplementary material.[29] The only difference between the pre- and post-transpilation circuits is the change from H gates to U_2 gates; the number of cX gates is the same.

Using the same testing protocol described in the introductory paragraph of Sec. V A, we performed 20 trials of m shots of this $n = 8$ circuit for m ranging from 8 to 17. The results of these trials are also plotted in Fig. 4 in comparison to the $n = 6$ results. The behavior of P as a function of m is rather flat and random, which is indicative of small fluctuations close to zero. We emphasize the fact that we observed a nonzero probability of success for most values of m, which is a significant improvement over the results from the $n = 6$ case. This result suggests that the increased quantity of cX gates was indeed a major factor in the low success rate we saw for the $n = 6$ circuit.

The rate of success for this circuit was still rather low compared to both simulations on smaller problems and theoretical estimates. We suspect that this is simply due to the error rates of gates implemented on `ibmq_16_melbourne` as discussed in Sec. V C.

VI. DISCUSSION

Our simulations of an ideal quantum computer free of noise displayed the correctness of our implementation of Simon's algorithm. In the presence of noise on IBM quantum computers, we see that the algorithm performs very poorly. In addition to this, the benefit of performing many shots diminishes, which means that the algorithm loses its exponential speedup over the classical algorithm; ordinarily, the speedup would originate in the fact that we could guarantee success up to a probability arbitrary close to 1 with a number of shots linear in n.

Qiskit's transpiler changes our circuit in a way that allows for the physical limitations of the quantum computer to be overcome while theoretically leaving the algorithm unaffected, but it also introduces an unruly amount of cX gates into the circuit, which are subject to higher error rates than single-qubit gates. Since the number of cX gates in U_f scales at least linearly with n (being the length of the target number a), the algorithm, therefore, suffers worse performance as n increases. When we minimized the impact of the transpiler with a new circuit in Sec. V D, we saw an improved success rate, which supports this conclusion.

We have not discussed error correction protocols in this paper. Such protocols essentially repeat the algorithm multiple times on multiple sets of data, thus increasing the number of qubits and gates required. However, they may be able to further improve the success of the algorithm when it runs on actual quantum computers. In fact, quantum algorithms to be put to commercial use will likely require such protocols in their implementation. Currently, this requirement of additional qubits means that they are not yet appropriate for addressing the noise found in the quantum computer we used, but they are promising future prospects for helping quantum algorithms perform their tasks more reliably. For

more information on error correction, see the introductory guide in Ref. 48 and relevant chapters in textbooks on quantum computation.[16,17]

In conclusion, we have introduced the reader to challenges due to noise and transpilation. These are challenges that have an important impact on current quantum computers and must be dealt with in order to understand results. We presented an example problem that was solved efficiently with a quantum algorithm, implemented its algorithmic solution on a simulator and a quantum computer, compared the results from simulators and experiments, and demonstrated a way to minimize the effects of transpilation.

ACKNOWLEDGMENTS

The authors acknowledge support from the College of Physical and Mathematical Sciences at Brigham Young University. The authors thank Jason Saunders and other members of the Quantum Information and Dynamics research group at Brigham Young University for useful discussion and helpful feedback, as well as anonymous referees for feedback on the manuscript.

[a]Electronic mail: scottjohnstun@byu.net
[b]Electronic mail: vanhuele@byu.edu

[1]Richard P. Feynman, "Quantum mechanical computers," Opt. News **11**, 11–20 (1985).
[2]Peter W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," SIAM Rev. **41**, 303–332 (1999).
[3]Edward Gerjuoy, "Shor's factoring algorithm and modern cryptography. An illustration of the capabilities inherent in quantum computers," Am. J. Phys. **73**, 521–540 (2005).
[4]Lov K. Grover, "A fast quantum mechanical algorithm for database search," in *STOC 1996: Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing* (1996).
[5]Lov K. Grover, "From Schrödinger's equation to the quantum search algorithm," Am. J. Phys. **69**, 769–777 (2001).
[6]Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd, "Quantum algorithm for linear systems of equations," Phys. Rev. Lett. **103**, 150502 (2009).
[7]Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione, "An introduction to quantum machine learning," Contemp. Phys. **56**, 172–185 (2014).
[8]Wikipedia has an extensive list of companies involved in quantum computing or communication <https://en.wikipedia.org/wiki/List_of_companies_involved_in_quantum_computing_or_communication> (last accessed December 30, 2020).
[9]Michel H. Devoret, John M. Martinis, and John Clarke, "Measurements of macroscopic quantum tunneling out of the zero-voltage state of a current-biased Josephson junction," Phys. Rev. Lett. **55**, 1908–1911 (1985).
[10]Frank Arute *et al.*, "Quantum supremacy using a programmable superconducting processor," Nature **574**, 505–510 (2019).
[11]Christopher Monroe, Michael G. Raymer, and Jacob Taylor, "The U.S. National Quantum Initiative: From act to action," Science **364**, 440–442 (2019).
[12]National Academies of Sciences, Engineering, and Medicine, in *Quantum Computing: Progress and Prospects*, edited by Emily Grumbling and Mark Horowitz (The National Academies Press, Washington, DC, 2019).
[13]Frederick W. Strauch, "Resource letter QI-1: Quantum information," Am. J. Phys. **84**, 495–507 (2016).
[14]D. Candela, "Undergraduate computational physics projects on quantum computing," Am. J. Phys. **83**, 688–702 (2015).
[15]Yohan Vianna, Mariana R. Barros, and Malena Hor-Meyll, "Classical realization of the quantum Deutsch algorithm," Am. J. Phys. **86**, 914–923 (2018).
[16]Isaac Chuang and Michael Nielsen, *Quantum Computation and Quantum Information* (Cambridge U. P., Cambridge, 2000).
[17]N. David Mermin, *Quantum Computer Science* (Cambridge U. P., Cambridge, 2007).
[18]Abraham Asfaw *et al.*, Learn quantum computation using Qiskit <https://qiskit.org/textbook/preface.html> (2020).
[19]Daniel Koch, Laura Wessing, and Paul M. Alsing, "Introduction to coding quantum algorithms: A tutorial series using Qiskit," e-print arXiv:1903.04359v1 (2019).
[20]Daniel Koch *et al.*, "Fundamentals in quantum algorithms: A tutorial series using Qiskit continued," e-print arXiv:2008.10647 (2020).
[21]Javier Rodríguez-Laguna and Silvia N. Santalla, "Building an adiabatic quantum computer simulation in the classroom," Am. J. Phys. **86**, 360–367 (2018).
[22]N. David Mermin, "From Cbits to Qbits: Teaching computer scientists quantum mechanics," Am. J. Phys. **71**, 23–30 (2003).
[23]Chih-Chieh Chen *et al.*, "Hybrid classical-quantum linear solver using Noisy Intermediate-Scale Quantum machines," Sci. Rep. **9**, 16251 (2019).
[24]Sima E. Borujeni *et al.*, "Quantum circuit representation of Bayesian networks," Expert Syst. Appl. **176**, 114768 (2020).
[25]James R. Wootton, "Benchmarking near-term devices with quantum error correction," Quantum Sci. Technol. **5**, 044044 (2020).
[26]Announcement by the National Science Foundation on May 18, 2020 <https://www.nsf.gov/news/special_reports/announcements/051820.jsp> (last accessed December 30, 2020).
[27]Daniel R. Simon, "On the power of quantum computation," in *1994 Proceedings of the 35th Annual Symposium on Foundations of Computer Science* (Institute of Electrical and Electronic Engineers Computer Society Press, 1994), pp. 115–123.
[28]Some preliminary results of this analysis were submitted for presentation at the 2020 Annual Conference of the Utah Academy of Sciences, Arts, and Letters and are included in its proceedings.
[29]See supplementary material at https://www.scitation.org/doi/suppl/10.1119/10.0006204 for several appendixes that expand and clarify our discussion and results.
[30]Supplementary material: Demonstration of transpilation effects in Qiskit <https://github.com/Dot145/QiskitTranspilationNoise/blob/master/QuantumNoiseInQiskit.ipynb>."
[31]Philip Krantz *et al.*, "A quantum engineer's guide to superconducting qubits," Appl. Phys. Rev. **6**, 021318 (2019).
[32]For an enlightening description of noise sources by Will Oliver, see <https://youtu.be/aGAb-GbrvMU?t<983> (last accessed December 30, 2020).
[33]See the IBM Research Blog post on dealing with errors in quantum computers a <https://www.ibm.com/blogs/research/2014/06/dealing-with-errors-in-quantum-computing/> (last accessed December 30, 2020).
[34]Adriano Barenco *et al.*, "Approximate quantum Fourier transform and decoherence," Phys. Rev. A **54**, 139–146 (1996).
[35]See Eq. (2.37) in Ref. 17.
[36]See Eq. (2.40) and the discussion leading up to it in Ref. 17.
[37]Y. S. Nam and R. Blümel, "Robustness of the quantum Fourier transform with respect to static gate defects," Phys. Rev. A **89**, 042337 (2014).
[38]Gadi Aleksandrowicz *et al.*, "Qiskit: An open-source framework for quantum computing," Zenodo (2019).
[39]Information on IBM quantum simulators can be found at <https://www.ibm.com/quantum-computing/simulator/> (last accessed December 30, 2020).
[40]See the Qiskit page with information installing the package at <https://qiskit.org/documentation/install.html#installing-qiskit> (last accessed December 30, 2020).
[41]The Qiskit blog on medium is a helpful source for using Qiskit. It can be found at <https://medium.com/@qiskit> (last accessed December 30, 2020).
[42]Qiskit also has a YouTube channel with video lectures and helpful tutorials at <https://www.youtube.com/c/qiskit> (last accessed December 30, 2020).
[43]Clara R. Woods, "Evaluating IBM's quantum compiler and quantum computer architectures as they pertain to quantum walk simulation algorithms," Honors thesis (University of California, San Diego, 2019).
[44]See the Qiskit webpage on its transpiler, which can be found at <https://qiskit.org/documentation/apidoc/transpiler.html#supplementary-information> (last accessed December 30, 2020).
[45]Krysta M. Svore, Matthew B. Hastings, and Michael Freedman, "Faster phase estimation," arXiv:1304.0741 (2013).
[46]Pedro M. Q. Cruz *et al.*, "Optimizing quantum phase estimation for the simulation of Hamiltonian eigenstates," Quantum Sci. Technol. **5**, 044005 (2020).
[47]These and other quantum computers at IBM can be accessed easily via the IBM Quantum Experience, located at <https://quantum-computing.ibm.com/> (last accessed December 30, 2020).
[48]Joschka Roffe, "Quantum error correction: An introductory guide," Contemp. Phys. **60**, 225–245 (2019).