DOI: 10.1111/cgf.14015

COMPUTER GRAPHICS *forum*
Volume 39 (2020), number 6 pp. 243–257

Quantum Coin Method for Numerical Integration

N. H. Shimada and T. Hachisuka

The University of Tokyo, Tokyo, Japan
NHShimada93@gmail.com, thachisuka@siggraph.org

Abstract

Light transport simulation in rendering is formulated as a numerical integration problem in each pixel, which is commonly estimated by Monte Carlo integration. Monte Carlo integration approximates an integral of a black-box function by taking the average of many evaluations (i.e. samples) of the function (integrand). For N queries of the integrand, Monte Carlo integration achieves the estimation error of $O(1/\sqrt{N})$. Recently, Johnston [Joh16] introduced quantum super-sampling (QSS) into rendering as a numerical integration method that can run on quantum computers. QSS breaks the fundamental limitation of the $O(1/\sqrt{N})$ convergence rate of Monte Carlo integration and achieves the faster convergence rate of approximately $O(1/N)$ which is the best possible bound of any quantum algorithms we know today [NW99]. We introduce yet another quantum numerical integration algorithm, quantum coin (QCoin) [AW99], and provide numerical experiments that are unprecedented in the fields of both quantum computing and rendering. We show that QCoin's convergence rate is equivalent to QSS's. We additionally show that QCoin is fundamentally more robust under the presence of noise in actual quantum computers due to its simpler quantum circuit and the use of fewer qubits. Considering various aspects of quantum computers, we discuss how QCoin can be a more practical alternative to QSS if we were to run light transport simulation in quantum computers in the future.

Keywords: quantum computing, Monte Carlo

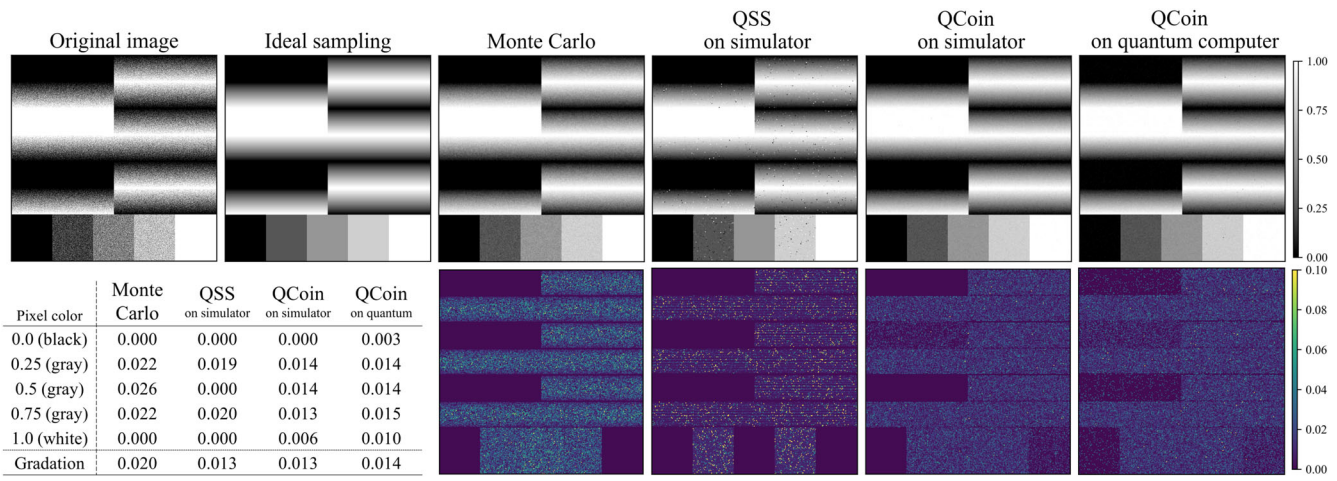
ACM CCS: **Methods and Applications** → Monte Carlo Techniques

1. Introduction

The use of quantum computers for computer graphics is a fascinating idea and potentially leads to a whole new field of research. Lanzagorta and Uhlmann 2005 mentioned this idea for the first time and suggested many interesting directions for further research. Their main focus is on Grover's database search algorithm [Gro96], and they showed how its application could lead to fundamentally more efficient algorithms than those on classical computers for various tasks in rendering, such as rasterization, ray casting and radiosity. Since Lanzagorta and Uhlmann, however, there has been little effort put into this direction, mostly due to the limited availability of actual quantum computers at that time.

Recently, Johnston [Joh16] introduced a quantum algorithm called *quantum super-sampling* (QSS) into computer graphics. Johnston proposed to use this algorithm to perform super-sampling of sub-pixels in rendering. This problem is essentially a numerical integration problem in each pixel, which is commonly done by Monte Carlo integration on classical computers. Johnston showed that the performance of this quantum algorithm is fundamentally better than classical Monte Carlo integration in terms of time complexity. On the other hand, his experiments on an actual quantum computer are not as successful as the simulated results due to the presence of noise in quantum computers. As noise is essentially unavoidable in the current architecture of quantum computers, this issue restricts the use of QSS in practice.

We introduce yet another quantum algorithm for numerical integration which runs well also on *actual* quantum computers; the Quantum Coin method (QCoin). We show that the performance of QCoin is equivalent to QSS both theoretically and numerically, including its convergence rate. We discuss the difference between two algorithms in terms of their implementations on a quantum computer. Unlike QSS, QCoin can be regarded as a hybrid of quantum-classical algorithm [KMT*17]. Being a hybrid algorithm, we show how QCoin is much more practical than QSS in the presence of noise and the various restrictions on actual quantum computers. We tested our QCoin on a real quantum computer and confirmed that QCoin already shows better performance than classical Monte Carlo integration. Figure 1 shows one experiment where we compared Monte Carlo and QCoin with the equal sample counts. QCoin achieves



Pixel color	Monte Carlo	QSS on simulator	QCoin on simulator	QCoin on quantum
0.0 (black)	0.000	0.000	0.000	0.003
0.25 (gray)	0.022	0.019	0.014	0.014
0.5 (gray)	0.026	0.000	0.014	0.014
0.75 (gray)	0.022	0.020	0.013	0.015
1.0 (white)	0.000	0.000	0.006	0.010
Gradation	0.020	0.013	0.013	0.014

Figure 1: *Experiment with super-sampling. We super-sample* 8×8 *sub-pixels of 'Original image'. 'Ideal sampling' shows the ground truth, which takes an average of* 8×8 *sub-pixels. Monte Carlo and our method (QCoin) take samples from sub-pixels to approximate this average for each pixel. The images show the results of numerical experiments using Monte Carlo, Quantum Super-sampling [Joh16](QSS) on a noiseless simulator, QCoin on a noiseless simulator and QCoin on an actual quantum computer with the equal sample counts of 240 queries (only QSS is done by 255 queries). The table shows mean absolute error of five coloured rectangular regions (black, grey and white) in the bottom of super-sampling images and of gradation regions on the right half of the images. QCoin produces more accurate results than Monte Carlo does because of its accurate estimation using quantum computers. Although QSS is comparable to QCoin, it works well only on a noiseless simulator as reported by Johnston [Joh16]. QCoin, on the other hand, works well even on an actual quantum computer for the first time.*

more accurate estimation both on a simulator and an actual quantum computer.

We also discuss several open problems for running rendering tasks on quantum computers in the future.

2. Background

Before diving into the details of our method, we first summarize some basic concepts of quantum computing for readers who are not familiar with them. Although we do cover the basics that are necessary to understand our method in this paper, for some further details, readers might want to refer to a standard textbook of quantum computing [NC11] or an introductory textbook for readers with computer science background [JHS19].

Single-qubit and super-position. On a classical computer, all the information is stored as a set of *bits* where each bit represents only a binary number 0 or 1. We represent a state of a bit having 0 as $|0\rangle$ and 1 as $|1\rangle$. The notation of $|\rangle$ is called *bra-ket*, which is commonly used in the field of quantum computing. On a quantum computer, a single *qubit* can represent a *super-position* of both $|0\rangle$ and $|1\rangle$. For example, we can represent a super-position state, supposing the real number $a \in [-1, 1]$ as an *amplitude* of $|0\rangle$:

$$|\psi\rangle = a|0\rangle + \sqrt{1-a^2}|1\rangle. \tag{1}$$

If we *measure* (read out) a qubit, the state converges to either side of $|0\rangle$ or $|1\rangle$. That is, the only information we can get is either 0 or 1 as in a classical case. This process is *probabilistic* and the probability is given by a squared value of its amplitude. For example, in the case of Equation (1), the measurement of $|\psi\rangle$ returns $|0\rangle$ with the probability a^2 and $|1\rangle$ with the probability $1-a^2$.

Quantum logic gates. Just like logic gates for bits on classical computers, there are several known *quantum logic gates* that are used to manipulate qubits. We summarize some of them here.

Identity gate $\hat{I}$

$$\hat{I}|0\rangle = |0\rangle$$
$$\hat{I}|1\rangle = |1\rangle$$

Pauli $\hat{X}, \hat{Z}$ gates

$$\hat{X}|0\rangle = |1\rangle,\ \hat{Z}|0\rangle = |0\rangle$$
$$\hat{X}|1\rangle = |0\rangle,\ \hat{Z}|1\rangle = -|1\rangle$$

Hadamard gate $\hat{H}$

$$\hat{H}|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}$$
$$\hat{H}|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

Rotation gate $\hat{U}_\theta$

$$U_\theta|0\rangle = \cos\theta|0\rangle + \sin\theta|1\rangle$$
$$U_\theta|1\rangle = -\sin\theta|0\rangle + \cos\theta|1\rangle$$
(θ is a rotation angle)

Multi-qubits. We express a multi-qubit state by concatenating single-qubit states. For example, a two-qubits state whose qubits are both $|0\rangle$ are expressed as $|0\rangle \otimes |0\rangle$ or $|00\rangle$. The symbol $\otimes$ represents a tensor product which means the concatenation of qubits in this case. In the following, we omit the symbol $\otimes$ for simplicity when it is obvious. In general, a two-qubits state whose qubits are both super-position states as Equation (1) can be written as

$$|\psi\rangle_a = a_0|0\rangle + a_1|1\rangle,\ |\psi\rangle_b = b_0|0\rangle + b_1|1\rangle$$
$$\rightarrow\ |\psi\rangle_a \otimes |\psi\rangle_b = (a_0|0\rangle + a_1|1\rangle) \otimes (b_0|0\rangle + b_1|1\rangle)$$
$$= a_0b_0|00\rangle + a_0b_1|01\rangle + a_1b_0|10\rangle + a_1b_1|11\rangle.$$

As this explicit binary notation quickly becomes tedious for many qubits, we use another notation $|i)$ for a decimal number i in the binary representation $i_{n-1}\cdots i_1 i_0$ as

$$|i) \equiv |i_{n-1}\rangle \otimes \cdots \otimes |i_1\rangle \otimes |i_0\rangle. \tag{2}$$

For example, in the case of four qubits, we write as

$$|0) = |0000\rangle, |1) = |0001\rangle, |2) = |0010\rangle, \cdots, |15) = |1111\rangle.$$

Quantum operation as a tensor product. In quantum computing, tensor products are also used to represent logic gate operations. For example, given the initial two-qubits state $|0\rangle \otimes |0\rangle$, the application of the Hadamard $\hat{H}$ gate for the first qubit and the Pauli $\hat{Z}$ gate for the second qubit can be written as

$$(\hat{H} \otimes \hat{Z})(|0\rangle \otimes |0\rangle) = \hat{H}|0\rangle \otimes \hat{Z}|0\rangle. \tag{3}$$

If we only operate the $\hat{H}$ gate for the first qubit and leave the second qubit unchanged, we can use the identity gate $\hat{I}$:

$$(\hat{H} \otimes \hat{I})|0\rangle \otimes |0\rangle. \tag{4}$$

When we apply the same gate to all the qubits, we omit the $\otimes$ symbol and simplify the notation as

$$\hat{H}|00\rangle \equiv \hat{H}|0\rangle \otimes \hat{H}|0\rangle. \tag{5}$$

This notation is also adopted in the case of the decimal representation in Equation (2). For the four qubits case,

$$\begin{aligned}\hat{H}|0) &\equiv \hat{H}|0000\rangle = \hat{H}|0\rangle \otimes \hat{H}|0\rangle \otimes \hat{H}|0\rangle \otimes \hat{H}|0\rangle \\ &= \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} \\ &= \frac{1}{\sqrt{2^4}}(|0000\rangle + |0001\rangle + \cdots |1111\rangle) = \frac{1}{\sqrt{2^4}} \sum_{i=0}^{15} |i).\end{aligned} \tag{6}$$

Oracle gate. In quantum computing, it is usually assumed that we have a (quantum) circuit which converts the information of an input data for each specific application as a quantum state. This circuit is commonly called an *oracle* gate. For example, in a database-search problem [Gro96] with the input data $[a_0, a_1, a_2, a_3]$, the oracle $\hat{O}$ gate works as using a normalization constant C:

$$\hat{O}|00\rangle \to \frac{1}{C}(a_0|00\rangle + a_1|01\rangle + a_2|10\rangle + a_3|11\rangle), \tag{7}$$

which converts the input data into the amplitudes. The exact design of the quantum circuit of an oracle gate is usually omitted in the design of each quantum algorithm, but the computational universality [DBE95] almost guarantees the existence of such an circuit.

In the context of ray tracing, $\hat{O}$ can be considered as a *ray trace function*. Given the (sub-)pixel index (i.e. quantized pixel coordinate) x, a ray trace function $F(x)$ traces a ray from camera through the pixel x and returns the light throughput along this ray, which can model many rendering algorithms such as path tracing [Kaj86]. In advanced algorithms like path tracing, x is defined as a quantized high-dimensional coordinate including the pixel coordinate. For M (sub-)pixels, a classical computer needs to repeat this process M times by evaluating the ray trace function for all the (sub-)pixels.

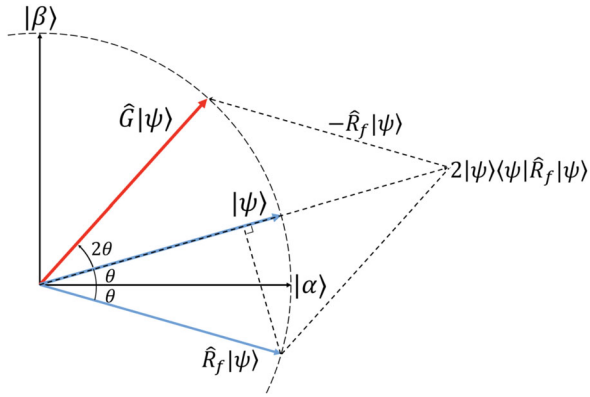


Figure 2: *Amplitude amplification from $|\psi\rangle$ to $\hat{G}|\psi\rangle$. The amplitude of $|\beta\rangle$ is amplified from $\cos\theta$ to $\cos 3\theta$ in the case that θ is a small value. The initial state $|\psi\rangle$ is sequentially changed as $|\psi\rangle \to \hat{R}_f|\psi\rangle \to 2|\psi\rangle\langle\psi|\hat{R}_f|\psi\rangle \to \hat{G}|\psi\rangle$.*

On a quantum computer, however, one can evaluate the ray trace function for all the (sub-)pixels in *one shot*:

$$\hat{O}|0) \to \frac{1}{C}\sum_{x=1}^{M} F(x)|x). \tag{8}$$

We assume the existence of such a ray tracing oracle gate, which is equivalent to the fact that Monte Carlo integration assumes that one can evaluate the integrand, without specifying how to evaluate.

Products using the bra-ket notation. Under the *bra-ket* notation [NC11], a *bra* vector $\langle A|$ denotes as a complex transpose of *ket* vector $|A\rangle$. For example, when $|A\rangle = \hat{U}|00\ldots0\rangle$, we have

$$\langle A| = |A\rangle^\dagger = (\hat{U}|00\ldots0\rangle)^\dagger = \langle 00\ldots0|\hat{U}^{-1}, \tag{9}$$

where $\hat{U}$ is a unitary matrix which represents a gate operation, and the complex transpose of a unitary matrix is an inverse matrix. One can think of $\langle A|$ ($|A\rangle$) as a row-vector (column-vector) representation. Using this notation, inner product (scalar) can be expressed as $\langle A|B\rangle$ and outer product (matrix) can be expressed as $|B\rangle\langle A|$.

3. Quantum Mean Estimation

Let us consider the problem of computing the mean of $F(x)$ in Equation (8). This problem corresponds to super-sampling M sub-pixels (or M quantized bins in high-dimensional integrands) in the context of ray tracing. When M is large, a popular algorithm on a classical computer is Monte Carlo integration; we randomly sample multiple sub-pixels and use their average as the estimate of the correct average. The estimation error of Monte Carlo integration is $O(1/\sqrt{M})$ for M samples.

On a quantum computer, we can evaluate $F(x)$ at *all the possible* x in one-shot using the oracle gate. As we explain later, it is also trivial

to transform the resulting state into another state whose amplitude is the correct average value $f \equiv \frac{1}{M}\sum_{x=1}^{M} F(x)$ as

$$|\psi\rangle = \sqrt{1-f^2}|0\rangle + f|1\rangle. \tag{10}$$

Unlike classical computers, it does not fundamentally matter how large M is on quantum computer as all the M values are computed in one shot. The remaining problem, however, is to estimate the amplitude f using this state.

One naive solution to this problem is to simply prepare N instances of $|\psi\rangle$ by querying the oracle N times and measure all of them (we cannot simply copy $|\psi\rangle$ N times just by querying the oracle 1 time at the beginning, due to the no-cloning theorem [Par70]). We then count the number of measured states belonging to $|1\rangle$ and deduce the value of f from that. This naive solution is essentially classical Monte Carlo integration, hence the convergence rate for N queries (i.e. samples) is $O(1/\sqrt{N})$, and does not provide any benefit compared to classical Monte Carlo integration. It is thus important to design a more efficient estimation algorithm which outperforms the classical calculation. We focus on two quantum algorithms in this paper: QSS and QCoin, which almost achieve $O(1/N)$ error with N queries. They use two other basic quantum algorithms called *amplitude amplification* and *quantum Fourier transformation*.

3.1. Amplitude amplification

The idea of amplitude amplification (AA) was first introduced in the context of a quantum database-search algorithm which is commonly known as *Grover's algorithm* [Gro96]. We consider an oracle $\hat{O}$ which results in

$$|\psi\rangle = \hat{O}|00\ldots0\rangle = \cos\theta|\alpha\rangle + \sin\theta|\beta\rangle, \tag{11}$$

where $|\beta\rangle$ is a set of target states and $|\alpha\rangle$ is a set of the other states. The state $|\psi\rangle$ is represented as a vector $(\cos\theta, \sin\theta)$ within a plane spanned by $|\alpha\rangle$ and $|\beta\rangle$ as shown in Figure 2. The goal of AA is to increase the small probability of observing the target state $|\beta\rangle$. The idea is to rotate $|\psi\rangle$ counter-clockwise as $|\psi\rangle \to \hat{G}|\psi\rangle$ in Figure 2. Note that AA, despite its name, does not necessarily amplify the amplitude when θ is larger than $\pi/2$, and thus one can treat AA just as a rotation operator.

As detailed in Figure 2, we first apply a flip operation $\hat{R}_f$ which flips the state $|\psi\rangle$ against the $|\alpha\rangle$ vector. It can be realized by flipping the sign of target states as $|\beta\rangle \to -|\beta\rangle$. We then project the resulting flipped state $\hat{R}_f|\psi\rangle$ onto the original $|\psi\rangle$, and multiply the length of the projected vector $|\psi\rangle\langle\psi|\hat{R}_f|\psi\rangle$ by two. Finally, we subtract $\hat{R}_f|\psi\rangle$ from it. The resulting state is

$$|\psi_{\text{result}}\rangle = \cos 3\theta|\alpha\rangle + \sin 3\theta|\beta\rangle. \tag{12}$$

The formula of AA operation $\hat{G}$ is derived as

$$\begin{aligned}|\psi_{\text{result}}\rangle = \hat{G}|\psi\rangle &\equiv (2|\psi\rangle\langle\psi|)\hat{R}_f|\psi\rangle - \hat{R}_f|\psi\rangle \\ &= \big(2|\psi\rangle\langle\psi| - \hat{I}\big)\hat{R}_f|\psi\rangle \qquad (13)\\ &= \big(2\hat{O}|00\ldots0\rangle\langle00\ldots0|\hat{O}^{-1} - \hat{I}\big)\hat{R}_f|\psi\rangle \\ &= \hat{O}\big(2|00\ldots0\rangle\langle00\ldots0| - \hat{I}\big)\hat{O}^{-1}\hat{R}_f|\psi\rangle. \qquad (14)\end{aligned}$$

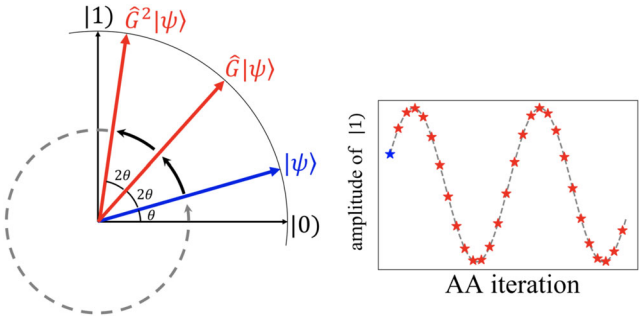


Figure 3: *Example of repeated AA operations. θ is initially defined as $f = \sin\theta$. The degree of state vector evolves as $\theta \to 3\theta \to 5\theta \cdots$ (left). Therefore, the trace of f values tracks a sin curve (right).*

The $(2|00\ldots0\rangle\langle00\ldots0| - \hat{I})$ operation corresponds to flipping the amplitude of all the states except the state $|00\ldots0\rangle$. As $\hat{G}$ includes two oracle gates ($\hat{O}$ and $\hat{O}^{-1}$), the AA algorithm makes two queries (i.e. $\hat{O}$ is called two times) to perform one $\hat{G}$ operator. Note that AA does not need to know the actual value of θ.

3.2. Quantum fourier transformation

Quantum Fourier transformation (QFT) can be thought as an analogy to classical discrete Fourier transformation. Given a data set $\{a_0, a_1, a_2, \cdots, a_{N-1}\}$, classical Fourier transformation $\{a_k \mid 0 \le k \le N-1\} \to \{b_j \mid 0 \le j \le N-1\}$ conducts the calculation as $b_j = \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} e^{-i\frac{2\pi}{N}jk}a_k$. The resulting set $\{b_i\}$ is a set of frequency components of the input data series $\{a_i\}$, and one can view that Fourier transform is an algorithm which converts $\{a_i\}$ into $\{b_i\}$. In QFT, the input data series is given by the amplitudes:

$$|\psi\rangle = a_0|0) + a_1|1) + \cdots + a_{N-1}|N-1). \tag{15}$$

The idea of QFT is to turn this input quantum state into a superposition of frequency components $\{b_i\}$ as

$$\big|\psi_{\text{QFT}}\big\rangle = b_0|0) + b_1|1) + \cdots + b_{N-1}|N-1). \tag{16}$$

We will not explain the detailed process of QFT in this paper as it is not important for our discussion. Interested readers can refer to a textbook of quantum computing [NC11].

3.3. Quantum super-sampling

Grover [Gro98] was the first to introduce a quantum algorithm for estimating a mean $f = \frac{1}{M}\sum_{i=1}^{M} F(i)$. The idea is to combine AA with QFT as we explain later. Many theoretical developments have followed since then [BHT06, BdSGT11], but few numerical experiments using simulation have been done so far [TKI99]. Johnston [Joh16] implemented this original idea by Grover to conduct numerical experiments in the context of rendering. The problem addressed there is super-sampling of an image, which can be seen as a mean estimation per pixel. We explain QSS by Johnston in the following, to contrast it to our QCoin. In the original work by Johnston [Joh16], the values of $F(i)$ are assumed to be binary $\{0, 1\}$. We

modified it to be able to handle continuous values of $F(i)$. As the algorithm essentially stays the same, we refer to our modified QSS simply as QSS in the following.

Main idea. The main idea of QSS is to exploit the existence of a periodic cycle when we keep applying amplitude amplification on $|\psi\rangle$. As we explained before, AA rotates the state within the plane spanned by $|\alpha\rangle$ and $|\beta\rangle$, thus the state actually rotates fully after sufficiently many AA operations. It turns out that there is a unique periodic cycle to each corresponding θ value. Figure 3 shows the movement of the state vector $|\psi\rangle$ (left) and the trace of the amplitude value of $|1\rangle$ (right). Applying QFT on the history of rotated $|\psi\rangle$, we can extract the frequency of this periodic cycle, which then allows us to calculate the corresponding θ (and therefore f).

Problem setting. In QSS, given a black-box function $F(a) : a \to [0, 1]$ and a quantum oracle operator

$$\hat{Q}_F : |0\rangle \otimes |i\rangle \to \left(\sqrt{1 - F(i)}|0\rangle + \sqrt{F(i)}|1\rangle\right) \otimes |i\rangle, \tag{17}$$

the objective is to get the average f of $F(a)$ with $N(= 2^n)$ samples:

$$f \equiv \frac{1}{N} \sum_{i=0}^{N-1} F(i). \tag{18}$$

Algorithm. In QSS, we use the oracle $\hat{Q}_F$ and make a super-position state $|\psi_0\rangle$ from the initial state whose all qubits (= register, target, and input qubits) are $|0\rangle$, where the numbers of qubits for each are $\log_2 P$, 1, $\log_2 N$. We thus write the initial state as

$$\left|\underbrace{0 \cdots 0}_{\log_2 P}\right\rangle \otimes |0\rangle \otimes \left|\underbrace{0 \cdots 0}_{\log_2 N}\right\rangle = |0\rangle \otimes |0\rangle \otimes |0\rangle \tag{19}$$

We generate a super-position state $|\psi_0\rangle$ as

$$|\psi_0\rangle = \hat{Q}_F(\hat{H} \otimes \hat{I} \otimes \hat{H})|0\rangle \otimes |0\rangle \otimes |0\rangle \tag{20}$$

$$= \frac{1}{\sqrt{PN}} \sum_{m=0}^{P-1} \sum_{i=0}^{N-1} |m\rangle \otimes \hat{Q}_F(|0\rangle \otimes |i\rangle) \tag{21}$$

$$= \frac{1}{\sqrt{PN}} \sum_{m=0}^{P-1} \sum_{i=0}^{N-1} |m\rangle \otimes \left(\sqrt{1 - F(i)}|0\rangle + \sqrt{F(i)}|1\rangle\right) \otimes |i\rangle, \tag{22}$$

where $\hat{Q}_F$ in Equation (20) operates the latter two states. The total measurement probability of $\frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} \sqrt{F(i)}|1\rangle \otimes |i\rangle$ states is $\sum_{i=0}^{N-1} \sqrt{\frac{F(i)}{N}}^2 = f$. If we define $|0\rangle' \equiv \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |0\rangle \otimes |i\rangle$ and $|1\rangle'$ in the same manner, the amplitude of $|1\rangle'$ is $\sqrt{f}$

$$|\psi_0\rangle = \frac{1}{\sqrt{P}} \sum_{m=0}^{P-1} |m\rangle \otimes \left(\sqrt{1 - f}|0\rangle' + \sqrt{f}|1\rangle'\right).$$

We can define $\cos\theta$ and $\sin\theta$ as $\sqrt{1 - f}$ and $\sqrt{f}$, and $|\psi_0\rangle$ is

$$|\psi_0\rangle = \frac{1}{\sqrt{P}} \sum_{m=0}^{P-1} |m\rangle \otimes \left(\cos\theta|0\rangle' + \sin\theta|1\rangle'\right). \tag{23}$$

We then apply AA to the $(\cos\theta|0\rangle' + \sin\theta|1\rangle')$ state for P times:

$$|\psi_1\rangle = \frac{1}{\sqrt{P}} \sum_{m=0}^{P-1} |m\rangle\left(\cos(2m+1)\theta|0\rangle' + \sin(2m+1)\theta|1\rangle'\right). \tag{24}$$

We then measure the target qubits. We assume that the state is converged to $|1\rangle'$:

$$|\psi_2\rangle = \frac{1}{C} \sum_{m=0}^{P-1} \sin(2m+1)\theta \ |m\rangle|1\rangle'. \tag{25}$$

Finally, we perform QFT on $|\psi_2\rangle$. With a sufficiently large probability [BHT06], the result of measurement after QFT will be

$$t \simeq \frac{P\theta}{\pi}, \frac{P(\pi - \theta)}{\pi}. \tag{26}$$

If the measured and converged state is $|0\rangle'$, we get the same result. Therefore, we can deduce the estimated average f' by

$$f \approx f' = \sin^2\left(\frac{t\pi}{P}\right). \tag{27}$$

As $\frac{t\pi}{P}$ can be determined by the precision $O(1/P)$ in this process, f' also has the precision of $O(1/P)$. Johnston [Joh16] proposed to use a pre-computed table instead of the analytical expression in Equation (27) by considering only discrete values of f. The estimation error $|f - f'|$ is inversely proportional to the number of AA operations P. As AA uses two queries per operation, we perform $O(N)$ queries to achieve $O(1/N)$ error. Note that this convergence rate is faster than $O(1/\sqrt{N})$ of Monte Carlo integration.

Example. We show how the whole process works for the five qubits case where $P = 4$ and $N = 4$. The initial state of five qubits is $(|0\rangle \otimes |0\rangle) \otimes |0\rangle \otimes (|0\rangle \otimes |0\rangle) = |0\rangle \otimes |0\rangle \otimes |0\rangle$. At first, we apply $\hat{H} \otimes \hat{I} \otimes \hat{H}$ as in Equation (20):

$$\begin{aligned}&\left(\hat{H} \otimes \hat{I} \otimes \hat{H}\right)|0\rangle \otimes |0\rangle \otimes |0\rangle \\ &= \left(\frac{|0\rangle + |1\rangle + |2\rangle + |3\rangle)}{\sqrt{4}}\right) \otimes |0\rangle \otimes \left(\frac{|0\rangle + |1\rangle + |2\rangle + |3\rangle)}{\sqrt{4}}\right).\end{aligned}$$

This transformation is as Equation (6). Then, the oracle $\hat{Q}_F$ works as (omitted the register qubits):

$$\begin{aligned}&\hat{Q}_F|0\rangle \otimes \left(\frac{|0\rangle + |1\rangle + |2\rangle + |3\rangle)}{\sqrt{4}}\right) \\ &= \frac{1}{\sqrt{4}}\left(\sqrt{1 - F(0)}|0\rangle + \sqrt{F(0)}|1\rangle\right) \otimes |0\rangle \\ &+ \frac{1}{\sqrt{4}}\left(\sqrt{1 - F(1)}|0\rangle + \sqrt{F(1)}|1\rangle\right) \otimes |1\rangle \\ &+ \frac{1}{\sqrt{4}}\left(\sqrt{1 - F(2)}|0\rangle + \sqrt{F(2)}|1\rangle\right) \otimes |2\rangle \\ &+ \frac{1}{\sqrt{4}}\left(\sqrt{1 - F(3)}|0\rangle + \sqrt{F(3)}|1\rangle\right) \otimes |3\rangle.\end{aligned}$$

The probability of observing $|1\rangle$ is calculated as

$$\left|\sqrt{\frac{F(0)}{4}}\right|^2 + \left|\sqrt{\frac{F(1)}{4}}\right|^2 + \left|\sqrt{\frac{F(2)}{4}}\right|^2 + \left|\sqrt{\frac{F(3)}{4}}\right|^2 = f$$

hence the total amplitude of $|1\rangle$ is $\sqrt{f}$. By grouping a set of states with $|1\rangle$ as $|1\rangle'$ (and those with $|0\rangle$ as $|0\rangle'$) for brevity, the resulting state vector can be written as $\sqrt{1-f}\,|0\rangle' + \sqrt{f}\,|1\rangle' = \cos\theta\,|0\rangle' + \sin\theta\,|1\rangle'$ where we write $\sqrt{f} = \sin\theta$. The state $|\psi_0\rangle$ is

$$|\psi_0\rangle = \left(\frac{|0\rangle + |1\rangle + |2\rangle + |3\rangle}{\sqrt{4}}\right) \otimes \cos\theta|0\rangle' + \sin\theta|1\rangle'.$$

We perform AA operations ($\hat{G}$) corresponding to the decimal number of the register qubits' state

$$\begin{aligned}
|\psi_1\rangle = {} & \frac{1}{\sqrt{4}}|0\rangle \otimes \hat{G}^0\left(\cos\theta|0\rangle' + \sin\theta|1\rangle'\right) \\
& + \frac{1}{\sqrt{4}}|1\rangle \otimes \hat{G}^1\left(\cos\theta|0\rangle' + \sin\theta|1\rangle'\right) \\
& + \frac{1}{\sqrt{4}}|2\rangle \otimes \hat{G}^2\left(\cos\theta|0\rangle' + \sin\theta|1\rangle'\right) \\
& + \frac{1}{\sqrt{4}}|3\rangle \otimes \hat{G}^3\left(\cos\theta|0\rangle' + \sin\theta|1\rangle'\right) \\
= {} & \frac{1}{\sqrt{4}}|0\rangle \otimes \left(\cos\theta|0\rangle' + \sin\theta|1\rangle'\right) \\
& + \frac{1}{\sqrt{4}}|1\rangle \otimes \left(\cos 3\theta|0\rangle' + \sin 3\theta|1\rangle'\right) \\
& + \frac{1}{\sqrt{4}}|2\rangle \otimes \left(\cos 5\theta|0\rangle' + \sin 5\theta|1\rangle'\right) \\
& + \frac{1}{\sqrt{4}}|3\rangle \otimes \left(\cos 7\theta|0\rangle' + \sin 7\theta|1\rangle'\right).
\end{aligned}$$

We then measure the target qubit $|1\rangle'$ to obtain $|\psi_2\rangle = \frac{1}{C}(\sin\theta|0\rangle + \sin 3\theta|1\rangle + \sin 5\theta|2\rangle + \sin 7\theta|3\rangle) \otimes |1\rangle'$ which allows us to estimate θ (thus f) value using QFT.

4. Quantum Coin Method

We introduce another mean-estimation quantum algorithm, which we call as the QCoin. Although the theory of QCoin was introduced by Abrams and Williams 20 years ago [AW99], its actual implementation was not discussed and no numerical experiment has been done so far. We provide the first practical implementation of this algorithm by identifying practical issues and performed the first set of numerical experiments.

Quantum coin. QCoin uses a *quantum coin* as its core. A quantum coin is a quantum state as described in Equation (10), which returns the target state $|1\rangle$ ('head') with the probability of f^2, and other states $|0\rangle$ ('tail') with the probability $1 - f^2$. By counting the number of 'heads' out of the total number of trials, we can estimate f^2 (and f) with δ error with $O(1/\delta^2)$ queries. As we discussed before, this process alone is equivalent to Monte Carlo integration, thus it will not provide any benefit.

Main idea. Suppose that we have a rough estimate f' by running Monte Carlo integration using a quantum coin as described above with N queries. According to the error analysis of Monte Carlo integration, with a certain confidence probability, one can say that the actual value of f is in the interval of $[f' - \frac{\delta}{2}, f' + \frac{\delta}{2}]$ where $\delta = O(1/\sqrt{N})$. The idea of QCoin is to repeatedly shrink this interval by shifting and scaling it using quantum computation until we are sufficiently close to f. Figure 4 illustrates this process.

Problem setting. QCoin considers a black-box function

$$F(a) : a \to [0, 1] \tag{28}$$

and a quantum oracle operator $\hat{Q}_{F,E}$ which includes the function $F(a)$ and the offset (shifting) parameter E:

$$\hat{Q}_{F,E}|0\rangle \otimes |i\rangle \to \left(\sqrt{1 - (F(i) - E)^2}|0\rangle + (F(i) - E)|1\rangle\right) \otimes |i\rangle. \tag{29}$$

Our goal is to estimate the average value f similar to QSS.

Algorithm. For the first step, using oracle $\hat{Q}_{F,0}$, we make the initial super-position state (the number of qubits of input is $\log_2 N$):

$$\begin{aligned}
|\psi_0\rangle &= \hat{Q}_{F,0}(\hat{I} \otimes \hat{H})|0\rangle \otimes |0\rangle \\
&= \hat{Q}_{F,0} \sum_{i=0}^{N-1} |0\rangle \otimes |i\rangle \\
&= \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} \left(\sqrt{1 - F(i)^2}|0\rangle + F(i)|1\rangle\right) \otimes |i\rangle.
\end{aligned} \tag{30}$$

The construction of a quantum coin is in fact simple; we perform $\hat{H}$ operators for all the qubits after the oracle gate operation. After this process, each state is distributed with $\frac{1}{\sqrt{N}}$ amplitude to a $|0\rangle$ state and any amplitude to all the other states:

$$\begin{aligned}
|\psi_0\rangle' &= (\hat{I} \otimes \hat{H})|\psi_0\rangle \\
&= \frac{1}{N} \sum_{i=0}^{N-1} F(i)|1\rangle \otimes |0\rangle + \cdots = f|1\rangle \otimes |0\rangle + \cdots
\end{aligned} \tag{31}$$

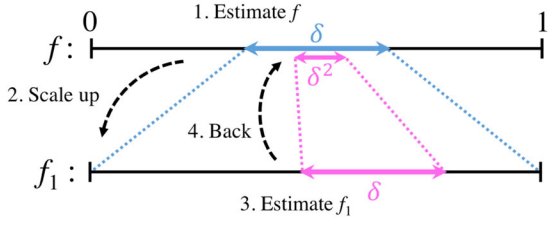


Figure 4: *Shifting-Scaling process of QCoin: (a) We estimate the value of f and decide bounded-error range: $[f' - \frac{\delta}{2}, f' + \frac{\delta}{2}]$. (b) We scale up quantum coin to [0,1]. (c) Now the target value f is changed to f_1, we can estimate f_1 with δ error. (d) We can estimate f with δ^2 error via calculating back from estimated f_1 value.*

We show the construction of a quantum coin for the three qubits case. The initial state of three qubits is $|0\rangle \otimes (|0\rangle \otimes |0\rangle) = |0\rangle \otimes |0)$. Applying $\hat{I} \otimes \hat{H}$ operation, we have

$$\big(\hat{I} \otimes \hat{H}\big)(|0\rangle \otimes |0)) = |0\rangle \otimes \left(\frac{|0) + |1) + |2) + |3)}{\sqrt{4}}\right).$$

We get $|\psi_0\rangle$ in Equation (30) using $\hat{Q}_{F,0}$:

$$\begin{aligned}
|\psi_0\rangle &= \hat{Q}_{F,0}|0\rangle \otimes \left(\frac{|0) + |1) + |2) + |3)}{\sqrt{4}}\right) \\
&= \frac{1}{\sqrt{4}}\Big(\sqrt{1 - F(0)^2}|0\rangle + F(0)|1\rangle\Big) \otimes |0) \\
&+ \frac{1}{\sqrt{4}}\Big(\sqrt{1 - F(1)^2}|0\rangle + F(1)|1\rangle\Big) \otimes |1) \\
&+ \frac{1}{\sqrt{4}}\Big(\sqrt{1 - F(2)^2}|0\rangle + F(2)|1\rangle\Big) \otimes |2) \\
&+ \frac{1}{\sqrt{4}}\Big(\sqrt{1 - F(3)^2}|0\rangle + F(3)|1\rangle\Big) \otimes |3).
\end{aligned}$$

Now, if we operate $\hat{H}$ on the input qubits, the states are changed to

$$\hat{H}|0) = \frac{|0) + |1) + |2) + |3)}{\sqrt{4}}, \quad \hat{H}|1) = \frac{|0) - |1) + |2) - |3)}{\sqrt{4}}$$
$$\hat{H}|2) = \frac{|0) + |1) - |2) - |3)}{\sqrt{4}}, \quad \hat{H}|3) = \frac{|0) - |1) - |2) + |3)}{\sqrt{4}}.$$

All states are distributed to $|0)$ with $+\frac{1}{\sqrt{4}}$ amplitude, hence

$$\begin{aligned}
|\psi_0\rangle' &= \big(\hat{I} \otimes \hat{H}\big)|\psi_0\rangle \\
&= \left(\frac{F(0) + F(1) + F(2) + F(3)}{4}\right)|1\rangle \otimes |0) + \cdots \\
&= f|1\rangle \otimes |0) + \cdots .
\end{aligned}$$

Note that the amplitude of $|1\rangle \otimes |0)$ is equal to f. This $\big|\psi_0\rangle'$ state thus can be regarded as a quantum coin. We use $\big|\psi_0\rangle'$ to perform a rough estimate of f within δ error using $O(1/\delta^2)$ queries just like Monte Carlo integration. Suppose that the estimated value is f_0, then we can say that the correct value f is in the interval $[f_0 - \frac{\delta}{2}, f_0 + \frac{\delta}{2}]$ with a certain high probability (first process in Figure 4).

For the next step, we set $E \equiv f_0 - \frac{\delta}{2}$ and the oracle gate as $\hat{Q}_{F,E}$. We make the quantum coin $\big|\psi_1\rangle'$ using $\hat{Q}_{F,E}$ as above:

$$|\psi_1\rangle' = (f - E)|1\rangle \otimes |0) + \cdots . \tag{32}$$

Now, the amplitude of $|1\rangle \otimes |0)$ is $f - E$. This value is in the interval $[0, \delta]$. If we define $\sin\theta \equiv f - E$,

$$|\psi_1\rangle' \equiv \sin\theta|1\rangle \otimes |0) + \cdots . \tag{33}$$

We then operate AA for $O(1/\delta)$ times to make the error range from $[0, \delta]$ to $[0, 1 - \epsilon]$ (upper limit is not always precisely 1). It can be done without knowing the exact value of f. If we conduct m times AA operations, the state is changed as

$$|\psi_1\rangle'' = \sin(2m + 1)\theta\,|1\rangle \otimes |0) + \cdots . \tag{34}$$

This corresponds to the second process in Figure 4. Now, we can estimate the value of $\sin(2m + 1)\theta$ within δ error measuring the state for $O(1/\delta^2)$ times (third process in Figure 4).

We assume the estimated value is f_1. Then, we can easily calculate back to the original scale: calculate the value of θ from m and $\sin(2m + 1)\theta$ values, and f is calculated by the relation '$f = \sin\theta + E$'. As a result, we get to estimate f value with the error range δ^2 (fourth process in Figure 4). If this step is repeatedly for k times, we achieve the error δ^{k+1}.

Algorithm 1. Our implementation of Qcoin **(*F*, *k*, *L*)**

```
// 1st step
f_0 ← 0
for i = 1 to L do
   make QCoin : Q̂_{√F,0} |0⟩ |0⟩
   if Measure(QCoin) == |1⟩ then
      f_0 += 1
   end if
end for
f_0 /= L
// The other steps
E_− ← 0.0, E_+ ← 1.0
for i = 1 to k do
   δ   ← sin(π/2^{i+1})            // hypothetical error
   E_− ← Max(f_{i−1} − δ/2, E_−)   // lower bound of error range
   E_+ ← Min(f_{i−1} + δ/2, E_+)   // upper bound
   f_i ← 0
   for j = 1 to L do
      make QCoin : Ĝ^{2^{i−1}}_{F,E_−} |0⟩ |0⟩
      if Measure(QCoin) == |1⟩ |0⟩ then
         f_i += 1
      end if
   end for
   f_i /= L
   f_i ← Min ( E_− + sin ( asin(f_i) / 2^i ) , E_+ )
end for
// Output
print f_k
```

Convergence rate. In the case of $k = 1$ step as above, the estimation error is δ^2, and the total number of queries is calculated as

$$O(1/\delta^2) + (1 + 2O(1/\delta)) \times O(1/\delta^2) = O(1/\delta^3) \tag{35}$$

The convergence rate is improved from 'δ error with $O(1/\delta^2)$ queries' to 'δ^2 error with $O(1/\delta^3)$ queries'. For comparison, assuming the numbers of queries are both N_{all}, the estimation error is reduced from $O(\frac{1}{N_{\text{all}}{}^{0.5}})$ to $O(\frac{1}{N_{\text{all}}{}^{0.66\cdots}})$.

As for the case of $k \gg 1$, we show the convergence rate here. If we use M queries in the Monte Carlo integration part of QCoin, we achieve $O(1/\sqrt{M})$ as the error value of δ (equivalently, $\delta = O(1/\sqrt{M})$). For a total of $k - 1$ iterations, QCoin achieves the final error value $O(\delta^k)$ as described above. On the other hand, the

total number of queries N_{all} in this case is asymptotically defined as

$$N_{\text{all}} = M \cdot O\left(1 + M^{1/2} + \cdots + M^{(k-1)/2}\right) = O\left(M^{k/2}\right) \tag{36}$$

for a large enough k. Given that we have $\delta = O(1/\sqrt{M})$, we can conclude that QCoin achieves the final error value of $O(\delta^k) = O(1/\sqrt{M}^k) = O(1/N_{\text{all}})$ using N_{all} queries using QCoin.

Our contributions over Abrams and Williams. Compared to the original work by Abrams and Williams 1999, our work provides the following contributions.

- We conducted numerical experiments to clarify the followings:
  - Although Abrams and Williams 1999 showed that the convergence rate of QCoin approaches to $O(1/N)$ as k increases, it has not been clear how the convergence rate changes for a finite (practical) k as we have done.
  - Similar to classical Monte Carlo integration, there is non-zero possibility that f resides outside the estimated interval at each step. Its influence is difficult to investigate just by looking at the theory, which we have shown by numerical experiments.
  - In the QCoin algorithm, Monte Carlo estimates are done by estimating f^2 (i.e. the probability of 'heads') and then taking its square-root, making its estimation more error-prone towards $f \approx 0$. This causes the fluctuation of an estimation error in accordance with the observed values. How this influences the efficiency of the algorithm is unknown.
- We redesigned and implemented the whole algorithm of QCoin as shown in Algorithm 1. Some technical points we implemented are as follows:
  - We showed how to deal with the cases where Monte Carlo estimation at each step is outside the range $[0, 1]$ which has been ignored so far, yet certainly happens in practice.

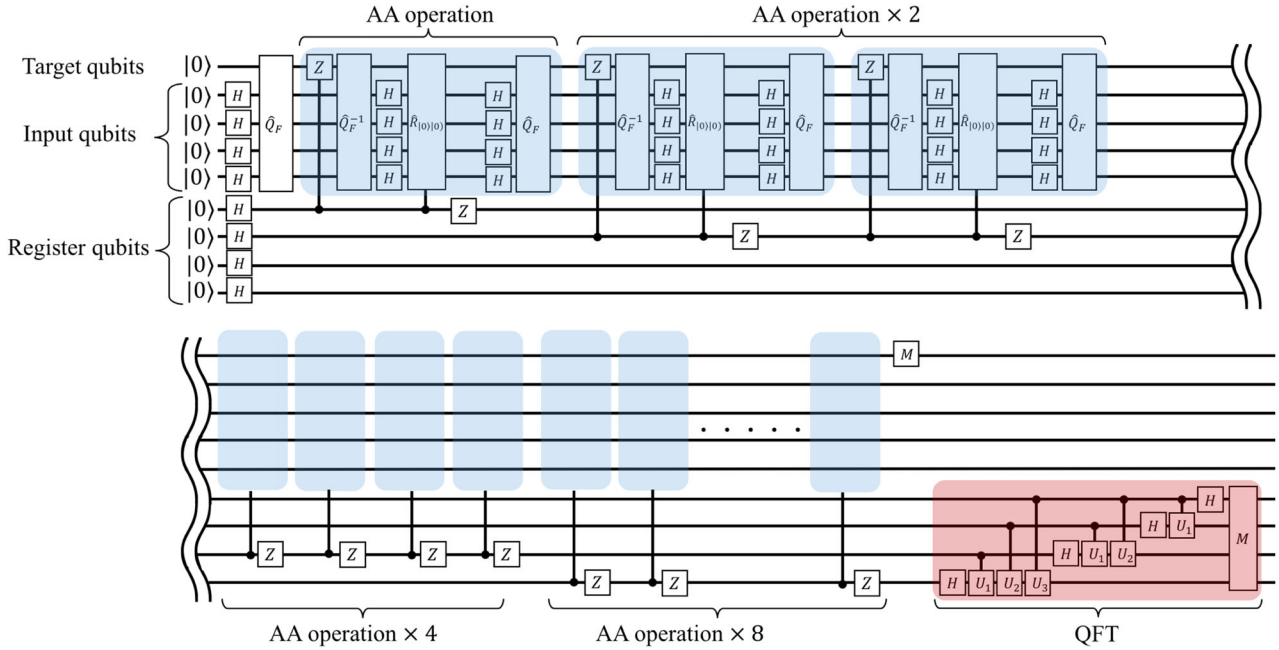


Figure 5: *Example of the quantum circuit of QSS with four input qubits and four register qubits case. (X, Z: Pauli gates, $\hat{H}$: Hadamard gate, $\hat{Q}_F$ and $\hat{Q}_F^{-1}$: oracle gate and inverse oracle gate, M: measurement gate, $R_{|0\rangle|0\rangle}$: phase flip gate only for the state $|0\rangle|0\rangle$), $U_n (n = 1, 2, 3)$: $e^{i\frac{\pi}{2^n}}$ phase shift gate for $|1\rangle$ state.)*

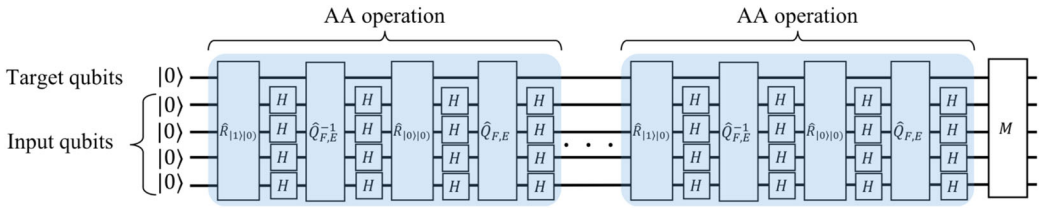


Figure 6: *Example of the quantum circuit of QCoin with four input qubits. ($\hat{Q}_{F,E}$ and $\hat{Q}_{F,E}^{-1}$: oracle gate and its inverse, $R_{|0\rangle|0\rangle}$ and $R_{|1\rangle|0\rangle}$: phase flip gate only for the state $|0\rangle|0\rangle$ and $|1\rangle|0\rangle$, respectively.)*

  - We proposed to directly estimate f for the first estimate using a different quantum coin than the rest.
  - We fixed the number of scaling-shifting operations per step.
- We pointed out the superiority of QCoin against QSS, for the first time, in terms of usefulness on an actual quantum computer.
  - We pointed out that QCoin belongs to a class of hybrid quantum-classical algorithms [KMT*17] and demonstrate its usefulness on actual quantum computers in the presence of noise (shown and discussed later).

5. On a Simulator

We now explain our implementations of QSS and QCoin on a simulator of quantum computing using Microsoft Q# [SO18]. Our source code is available on Github [Shi].

QSS. The AA operation $\hat{G}_F$ using from Equation (23) to Equation (24) in QSS is defined as Equation (13):

$$\hat{G}_F \equiv (2|\psi_0\rangle\langle\psi_0| - \hat{I})\,\hat{Z}. \tag{37}$$

$|\psi_0\rangle$ can be decomposed as Equation (20):

$$|\psi_0\rangle = \hat{Q}_F(\hat{I}\otimes\hat{H})\,|0\rangle|0\rangle) \tag{38}$$

$$\langle\psi_0| = \langle 0|(0|\,(\hat{I}\otimes\hat{H})\hat{Q}_F^{-1}. \tag{39}$$

We substitute Equation (38) and (39) to Equation (37):

$$\begin{aligned}\hat{G}_F &= \hat{Q}_F(\hat{I}\otimes\hat{H})\left(2|0\rangle|0\rangle\langle 0|(0| - \hat{I}\right)(\hat{I}\otimes\hat{H})\hat{Q}_F^{-1}\,\hat{Z}\\ &= -\hat{Q}_F(\hat{I}\otimes\hat{H})\left(\hat{I} - 2|0\rangle|0\rangle\langle 0|(0|\right)(\hat{I}\otimes\hat{H})\hat{Q}_F^{-1}\,\hat{Z},\end{aligned}$$

where we omit irrelevant register qubits here. If $(\hat{I} - 2|0\rangle|0\rangle\langle 0|(0|)$ is defined to be represented by $\hat{R}_{|0\rangle|0\rangle}$, we have

$$\hat{G}_F = -\hat{Q}_F(\hat{I}\otimes\hat{H})\,\hat{R}_{|0\rangle|0\rangle}\,(\hat{I}\otimes\hat{H})\hat{Q}_F^{-1}\,\hat{Z}. \tag{40}$$

The example of an quantum circuit of QSS using $\hat{G}_F$ is shown in Figure 5 where the number of input qubits is 4. Each blue-coloured region of the circuit corresponds to Equation (40). The operators in Equation (40) are lined up in the reverse order in the circuit (operators are like matrix operations, hence they are indeed conducted from the back of an equation). AA operations are controlled by register qubits, which allows us to store the history of the rotating state vector; only if a control-register qubit is $|1\rangle$, $\hat{G}_F$ is run and the state vector rotates.

Finally, the red region of the circuit performs QFT, which operates on the register qubits and extracts the period.

QCoin. The quantum circuit of QCoin is described in Figure 6, where the number of input qubits is 4. The exact operator $\hat{G}_{F,E}$ of AA for Qcoin $|\psi_1\rangle'$ in Equation (33) is defined as the Equation (13):

$$\hat{G}_{F,E} \equiv (2|\psi_1\rangle'\langle\psi_1|' - \hat{I})\,\hat{R}_{|1\rangle|0\rangle}, \tag{41}$$

where operator $\hat{R}_{|1\rangle|0\rangle}$ flips the amplitude of $|1\rangle|0\rangle$. $|\psi_1\rangle'$ is decomposed by $|\psi_1\rangle$ and some gate operations like Equation (31):

$$\begin{aligned}\hat{G}_{F,E} &= (\hat{I}\otimes\hat{H})\,(2|\psi_1\rangle\langle\psi_1| - \hat{I})\,(\hat{I}\otimes\hat{H})\,\hat{R}_{|1\rangle|0\rangle}\\ &= (\hat{I}\otimes\hat{H})\,(2|\psi_1\rangle\langle\psi_1| - \hat{I})\,(\hat{I}\otimes\hat{H}),\ \hat{R}_{|1\rangle|0\rangle}\end{aligned} \tag{42}$$

and $|\psi_1\rangle$ is also deconstructed from Equation (30):

$$\begin{aligned}2|\psi_1\rangle\langle\psi_1| - \hat{I} &= \hat{Q}_{\hat{F},E}(\hat{I}\otimes\hat{H})\,(2|0\rangle|0\rangle)(0|\langle 0| - \hat{I})\,(\hat{I}\otimes\hat{H})\hat{Q}_{\hat{F},E}^{-1}\\ &= \hat{Q}_{\hat{F},E}(\hat{I}\otimes\hat{H})\,\hat{R}_{|0\rangle|0\rangle}\,(\hat{I}\otimes\hat{H})\hat{Q}_{\hat{F},E}^{-1},\end{aligned} \tag{43}$$

where $(2|0\rangle|0\rangle(0|\langle 0| - \hat{I})$ is represented by $\hat{R}_{|0\rangle|0\rangle}$ for simplicity. Substituting Equation (43) to (42), we get the explicit form of $\hat{G}_{F,E}$:

$$\hat{G}_{F,E} = (\hat{I}\otimes\hat{H})\hat{Q}_{\hat{F},E}(\hat{I}\otimes\hat{H})\,\hat{R}_{|0\rangle|0\rangle}\,(\hat{I}\otimes\hat{H})\hat{Q}_{\hat{F},E}^{-1}(\hat{I}\otimes\hat{H})\,\hat{R}_{|1\rangle|0\rangle}. \tag{44}$$

5.1. Results

Convergent behaviour against target value. Figure 7 shows the behaviours of estimation error against the increasing number of queries with various target mean values f in three methods: Monte Carlo, QSS and QCoin. We conducted numerical experiments with 3000 samples for each point in Monte Carlo and QCoin, and calculated its theoretical error for QSS. In Monte Carlo, all the reduction rates of errors are almost uniform regardless of f, while QSS returns almost zero error at specific f. This distinctively different behaviour is also showed by Johnston, which arises from QFT. Fourier

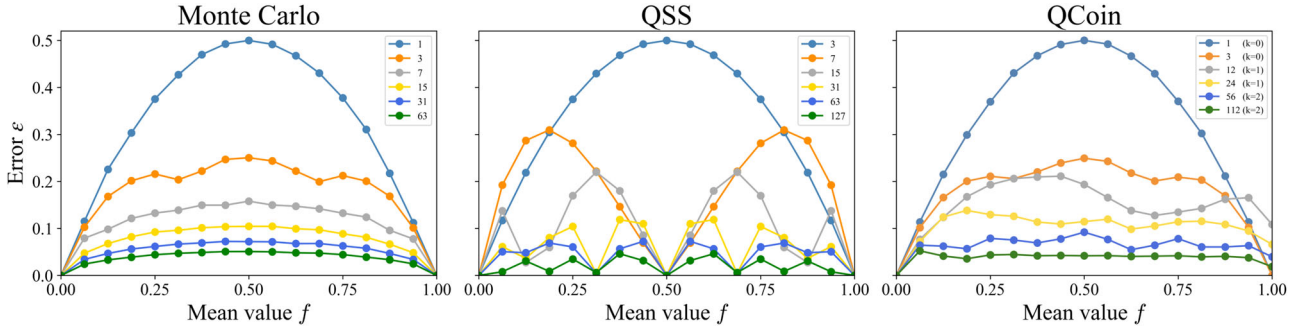


Figure 7: *Error plots against query times (represented by colours indicated in the legends) with various target mean f in three methods; Monte Carlo, QSS and QCoin.*

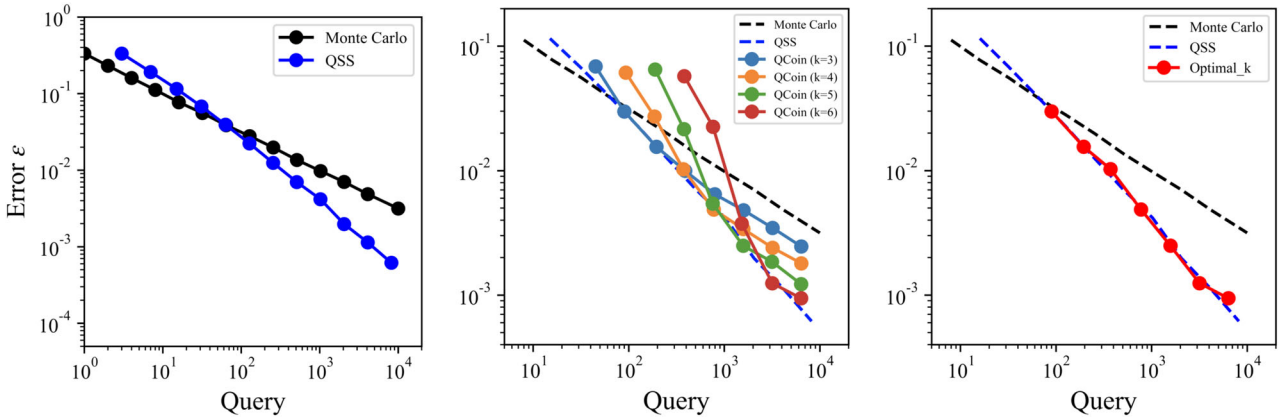


Figure 8: *(Left and Centre) Mean absolute error plots with query times in Monte Carlo, QSS and QCoin($k = 3, 4, 5, 6$). (Right) The best performance of QCoin with selected optimal k values is plotted.*

transformation extracts a period of data series, therefore it can definitely detect the frequency of wave whose period just matches the data length. In QCoin, however, we see almost uniform reduction of error just like Monte Carlo integration. One minor difference occurs at $f = 1.0$ where QCoin has non-zero error while Monte Carlo integration has zero error. This difference arises from the fact that the QCoin algorithm scales the bounded-error of quantum coin $[0, \delta]$ to $[0, a]$ (a is not always 1). If a is always 1, the QCoin with $f = 1.0$ only returns $|1\rangle$ at any steps. This experiment shows that the estimation error of QCoin has a similar characteristic as that of Monte Carlo. On the other hand, QSS behaves quite differently from Monte Carlo. As we discuss later, this similarity between MC and QCoin may allow us to use the existing error reduction methods (e.g. denoising) with QCoin.

Convergent behaviour against the number of queries. Figure 8 shows the mean error of QCoin with random f samplings for each k step. Figure 8 (Left) plots the results of Monte Carlo and QSS. Monte Carlo integration took 10 000 samples (f is randomly selected for each sample) for each point, and calculate theoretical error of QSS with uniformly selected 200 f values for each point. In Monte Carlo, the slope of the curve is -0.50 in the logarithmic scale which matches the theoretical convergence rate of $O(1/\sqrt{N})$ with $O(N)$ queries. In QSS, the slope is -0.85, hence it achieves $O(1/N^{0.85})$ error with $O(N)$ queries. This result is close to the theoretical rate of $O(1/N)$. Figure 8 (centre) shows the results of QCoin with $k = 3, 4, 5, 6$ cases. For all the k values, the error of few queries is large because the trials of a quantum coin in each step is too small for the estimation value be reasonably accurate for the succeeding shifting-scaling operations. Other than that, the slope for the same k value first quickly becomes close to -1.0, but asymptotically approaches to -0.5 after many queries while fixing k. We can thus observe that there is an optimal number of shifting and scaling operations k for a given total number of queries. Figure 8 (right) plots the results of QCoin with the those optimal k values for each number of queries. This optimal k results in almost the same performance as QSS, and we use this optimal k for the remaining experiments. This experiment thus demonstrates that QCoin performs as well as QSS for a finite number of queries on a noiseless simulator. Although its theoretical $O(1/N)$ convergence predicts that QCoin asymptotically outperforms Monte Carlo, we are the first to numerically verify its performance for a finite N.

Super-sampling. Figure 1 shows an application of our method to a rendering task. The task is super-sampling which estimates the average of sub-pixel values. One can think of this task as a numerical integration problem where the integrand is a function of sub-pixel values. This experiment is inspired by similar experiments done by Johnston [Joh16]. In our experiment, each pixel contains 8×8 sub-pixels. The ground truth image ('Ideal sampling') is computed by simply taking the average of all 64 sub-pixels. We used Monte Carlo, QSS on a simulator, QCoin on a simulator and QCoin on an actual quantum computer (later discussed for the last one) with the same 240 queries ($k = 3$ for QCoin) (255 queries only for QSS) by considering sub-pixels as the values of the integrand. The table in Figure 1 shows mean absolute error of the five rectangular regions at the bottom of each and of the gradation parts at the top right of the images, which highlight errors for particular pixel values.

In the gradation part of the images, the mean absolute error of 'QCoin on simulator' is almost the same as that of 'QSS on simulator', and about half as much as Monte Carlo's, which is consistent with the error plots in Figure 8. However, we can see a striking difference in the images. Although Monte Carlo and QCoin show uniform reduction of error in the region, QSS produces more error in some pixels and less error in other pixels, which is consistent with the convergence behaviour seen in Figure 7. This is also confirmed by coloured rectangular regions; in QSS, some regions of particular pixel-colour (0.0,0.5,1.0) show no error result, but the other parts indicate more estimation error than in QCoin.

6. On an Actual Quantum Computer

We use IBM Q5 Yorktown [IBM] and Qiskit [AO19] to run QSS and QCoin on an actual quantum computer. The hardware resources are

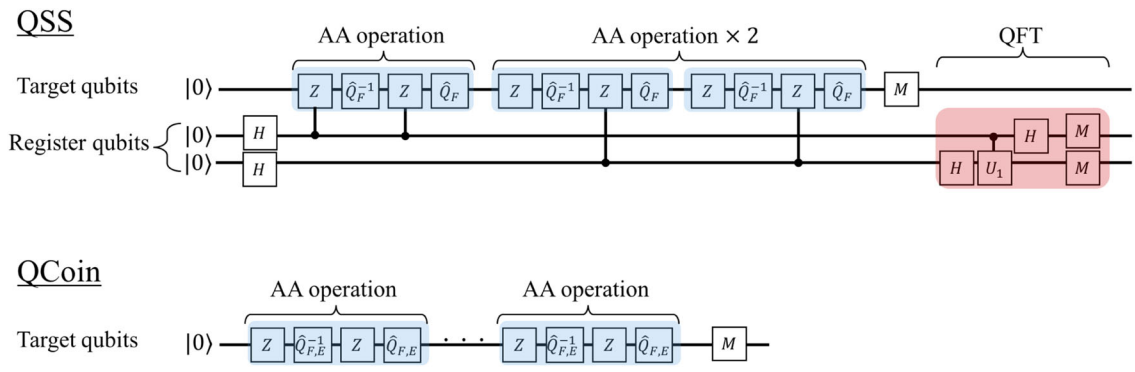


Figure 9: *(Top) QSS's quantum circuit for 6 queries in minimum setting; no input qubit and two register qubits. (Bottom) QCoin's quantum circuit in minimum setting; no input qubit.*

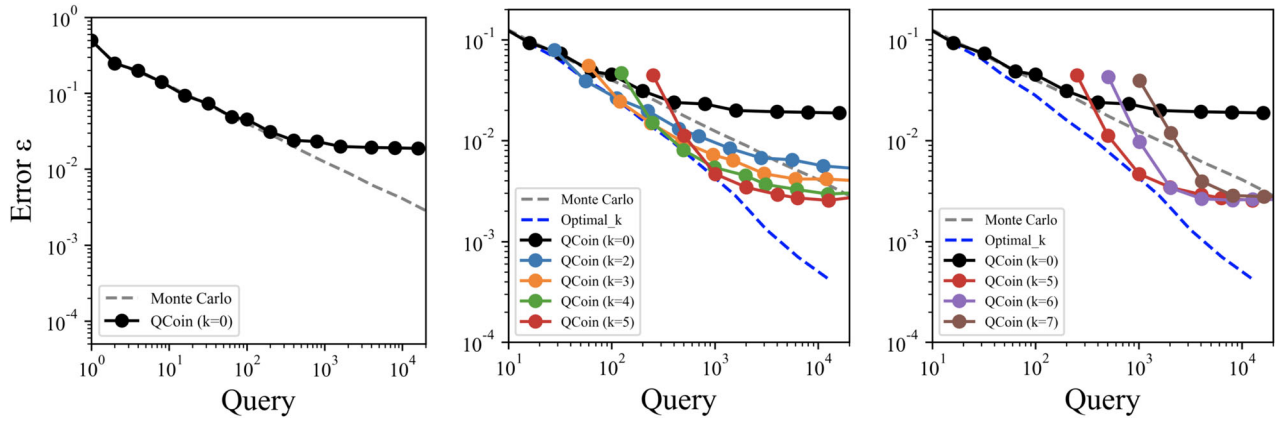


Figure 10: *Results of QCoin with $f = 0.50$ on an actual quantum computer. We plot $k = 0$ on the left, additionally $k = 2, 3, 4, 5$ in the centre and $k = 5, 6, 7$ on the right. Monte Carlo plot data points are calculated with 3000 samples, and Optimal_k data are plotted with the data points calculated by 500 samples on simulator.*

very limited, hence we simplified settings for both QSS and QCoin. Our source code is available on Github [Shi].

QSS. To implement QSS, we removed the circuit for input qubits and assumed that the oracle $\hat{Q}_F$ operates as

$$\hat{Q}_F|0\rangle = \sqrt{1-f}|0\rangle + \sqrt{f}|1\rangle.$$

The AA operation $\hat{G}$ is expressed as

$$\begin{aligned}\hat{G} &= \hat{Q}_F(2|0\rangle\langle 0| - \hat{1})\hat{Q}_F^{-1}\hat{R}_f \\ &= \hat{Q}_F\hat{Z}\hat{Q}_F^{-1}\hat{Z}. \end{aligned} \tag{45}$$

In one qubit case, the R_f flip gate counterparts to a Pauli Z gate, and the $(2|0\rangle\langle 0| - \hat{1})$ flip operation also does to $\hat{Z}$ gate. Despite its very simple implementation, we can create at most three qubits circuit (expressed in Figure 9) on the IBM Q5 quantum computer due to its requirement for the architecture of qubits as discussed in the previous section.

QCoin. In QCoin, we also simplified the quantum circuit by eliminating the circuit for input qubits and set the oracle as

$$\hat{Q}_{F,E}|0\rangle = \sqrt{1-f^2}|0\rangle + f|1\rangle.$$

In this case, the oracle can be regarded as the one including the process of making a quantum coin. AA operation $\hat{G}$ is almost the same as QSS:

$$\begin{aligned}\hat{G} &= \hat{Q}_{F,E}(2|0\rangle\langle 0| - \hat{1})\hat{Q}_{F,E}^{-1}\hat{R}_f \\ &= \hat{Q}_{F,E}\hat{Z}\hat{Q}_{F,E}^{-1}\hat{Z}. \end{aligned} \tag{46}$$

The quantum circuit is shown in Figure 9.

6.1. Results

QCoin. Figure 10 shows the error performance of QCoin on the quantum computer with $f = 0.50$. All the data points are calculated by 300 simulations. Figure 10 (left) shows the results of Monte Carlo integration and QCoin of $k = 0$ (equivalent to Monte Carlo

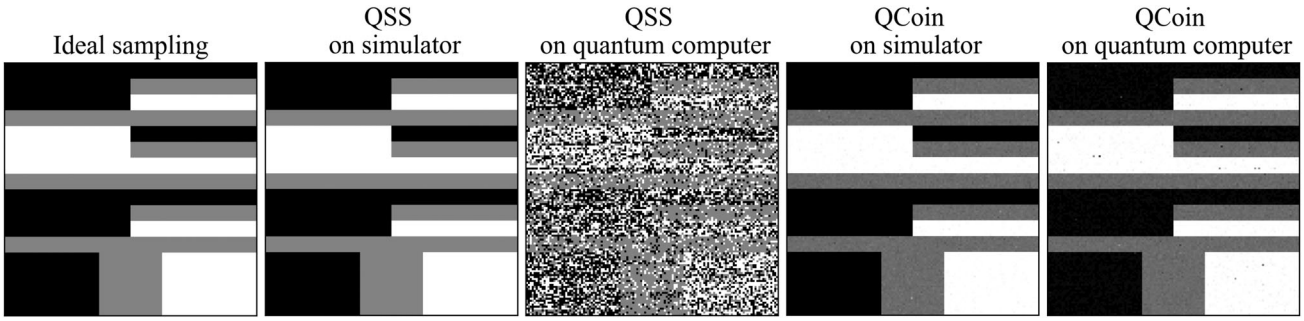


Figure 11: *Super-sampling with QSS on an actual quantum computer using seven queries (third form the left). Ideal sampling image is shown in first column, and QSS on simulator produces no error result as a second image in this case of the limited pixel colours. QCoin's results are also shown in the right, and the settings of experiments are the same as Figure 1.*

integration) on the quantum computer. The convergence rate is almost the same at small number of queries ($\lesssim 100$), while the reduction of error stops in the range of more query. This error seems to mainly arise from the readout error of qubits. Hence, we cannot improve this error by more trials. However, we are able to overcome this limitation by using the AA steps of quantum coin. We scale-up the bounded error and measure the enlarged quantum coin in each step as shown in Algorithm 1. We only need to estimate f value roughly for each step, hence the influence of readout error becomes relatively smaller than Monte Carlo method. Figure 10 (centre) shows $k = 2, 3, 4, 5$ cases of QCoin. We confirm that error performances are better than Monte Carlo and compatible to QCoin's on simulator even on real quantum computer in the range of rather small number of queries. For larger queries, the convergences of error reduction are seen, which seem to be also due to readout error. Figure 10 (right) additionally shows the plots of $k = 6, 7$ cases. The error does not reduce with $k = 6, 7$ compared to $k = 5$, though the scale-up steps of a quantum coin increases. This means that the scaling-up process over almost 16 AA iterations (corresponds to the $k = 5$ case) becomes meaningless due to accumulation of decoherence and gate error in large circuit calculation.

Super-sampling. We also conducted the experiment of super-sampling for QCoin on an actual quantum computer (the right image in Figure 1). Note that as we cannot prepare the oracle gates which convert all sub-pixels value into quantum states due to hardware limitation, we now set and use the oracle gate which directly have a target value as Equation (10). For grey-coloured regions (where the pixel colour is 0.25, 0.50 or 0.75), QCoin on IBMQ produces similar results as QCoin on a simulator, which is consistent with the results in Figure 10. On the other hand, in the black and white regions, QCoin on IBMQ shows a rather large estimation error than the simulation. These regions are sensitive to the noise of an actual quantum computer as the integrand in those regions should be either strictly zero or one, which can be easily corrupted by the noise. This disadvantage in the limited narrow region does not decrease overall performance of QCoin's algorithm so much, as we can see the mean absolute error of the gradation part is not almost reduced compared with QCoin on simulator.

We also show another super-sampling image via QSS in Figure 11. We cannot conduct QSS's algorithm with such large number of queries as in Figure 1 because of the hardware limitation of IBMQ. Then we now treat the case where the target pixel colour is limited to 0.0, 0.5 and 1.0 (seeing 'Ideal sampling' in the figure), and the number of queries for QSS is limited to seven. We, however, note that seven queries are enough to estimate the specific pixel colours of this case with no error on a simulator, which is confirmed by seeing QSS's result in Figure 7. As was also observed by Johnston [Joh16], the result of QSS on IBMQ is significantly influenced by the noise in an actual quantum computer, and the mean absolute error increases 0.30.

The dominant error in QSS on an actual quantum computer seems to be a multi-qubit gate error. Multi-qubit operations are more difficult to execute than single-qubit ones because they must additionally operate controlled functions. All the gate errors are publicly disclosed [IBM], and the multi-qubit gate errors are almost 5%. Therefore, even if we operate it for only a few times, accumulation error quickly reaches to the visible level and thus appears as noise in the image. Thanks to its simpler circuit design, QCoin does not suffer from this issue. One can thus see a striking difference between QSS and QCoin when we run both on an actual quantum computer.

7. Discussions

Ray tracing oracle gate. One might argue that our numerical experiments (inspired by Johnston [Joh16]) are too simple compared to the actual use cases of ray tracing, thus it is not really demonstrating the applicability of QCoin in rendering. Although we admit that it is not as complex as the actual use cases, the basic idea of QCoin is readily applicable to arbitrary complex integrands just like Monte Carlo integration. The remaining challenge is to design an oracle gate which efficiently performs ray tracing. Although it is theoretically possible to design such an oracle gate, we found it infeasible to perform any numerical experiment (*even on a simulator*) at this moment for two major reasons.

First, the maximum number of qubits we can have at the moment is about 50 qubits [PO18]. It is incorrect to assume that we can get away from this limitation on a simulator. Simulation of 50 qubits already takes roughly 16 peta byte of RAM if we store the full quantum states, and a 64 qubits simulator (on a cluster of 128 nodes) is possible only by limiting the complexity of quantum

circuits [CZX*18]. Given that even *one* floating point number consumes 32 bits, we concluded that it is currently infeasible to conduct any numerical experiment (both on actual and simulated quantum computers). Note that computation such as square root of floating point numbers adds up to the required number of qubits. Although we should be able to theoretically design such a quantum circuit, if we were to perform numerical experiments, even for *simple* cases like ray tracing of a sphere, we need either a better hardware or a simulator, which are both out of the scope in this paper.

Second, there is currently no research done on how to *appropriately* represent typical data for ray tracing. For example, due to the limited number of input qubits, it will not be immediately possible to handle triangle meshes on a quantum computer. Although Lanzagorta and Uhlmann 2005 mentioned a theoretical possibility of using Grover's method for ray tracing, implementation of this idea for any practical scene configurations is currently impossible. We thus believe that further research on a suitable data representation for rendering on quantum computers is deemed necessary, and this topic alone can lead to a series of many research questions and thus cannot be a short addendum in our paper. We focused on a numerical integration algorithm which serves as a basic building block for ray tracing on quantum computers. Our work should be useful as a stepping stone to conduct further research along this line.

Error distribution over the image. A unique requirement of solving many integrals on the image place in rendering highlights another important difference between QSS and QCoin. In QCoin, the distribution of errors over the image is essentially noise due to random sampling, which is the same as Monte Carlo integration. Being a hybrid quantum-classical method as we explain later, one can easily apply many exiting tools developed for Monte Carlo integration, such as denoising via image-space filtering [ZJL*15], to QCoin. QCoin can thus directly replace Monte Carlo integration while being asymptotically faster. In QSS, however, erroneous pixels appear as completely wrong pixel values [Joh16] which cannot be easily recovered or identified by the existing tools for Monte Carlo integration. For example, denoising for Monte Carlo rendering would not work as is for rendered images via QSS. We thus believe that QCoin is more readily applicable to rendering than QSS.

Required number of qubits. Quantum computers at this moment have a limited number of usable qubits. As mentioned above, the maximum number of qubits we can have is ~ 50 qubits [PO18] at the moment. Let us consider that the size of input is N and the maximum AA iterations is P. In this case, QSS needs $\log N + \log P$ qubits to run the algorithm, while QCoin needs only $\log N$ (left in Figure 12). The examples of quantum circuits for QSS and QCoin also verify this fact (Figures 5 and 6). For example, when the input $N = 2^{10} = 1024$ is given, using the current architecture of quantum computers, the number of AA iterations in QSS is limited to less than only $P = 2^5 = 32$ times, whereas QCoin can has no such limitation by construction. This severely limits the applicability of QSS, making QCoin an attractive alternative in practice.

Connections among qubits. Another well-known limitation of quantum computer architecture is the number of connections among qubits. In many quantum algorithms, controlled-gates are important.

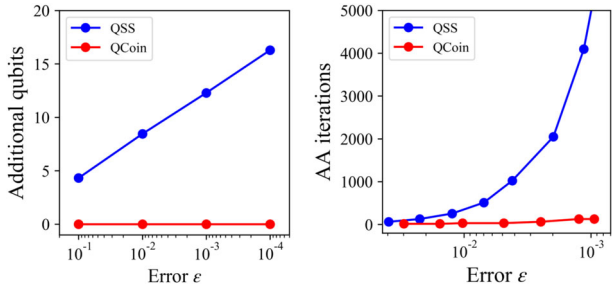


Figure 12: *Relation plots between the number of additional register qubits (AA iterations) and error ϵ in left (right).*

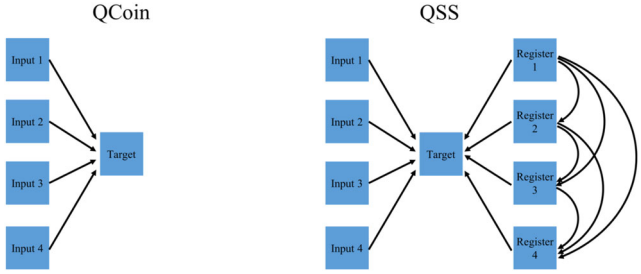


Figure 13: *Minimum qubits' architectures for the quantum circuits in Figure 6 (QCoin) and Figure 5 (QSS). The arrow represents a connection with controlled gate; a root qubit at the arrow is a control qubit, and an end-qubit is a target.*

However, it is currently difficult to prepare all interacting the qubits. For example, IBM Q20 Tokyo [IBM] has a total of 20 qubits, while the size of fully connected qubits set is only up to 4.

Due to the use of QFT, QSS needs a sequence of qubits which has a connection between target and the other qubits and full connections among the register (used for computation) qubits. On the other hand, QCoin needs connections only between target and input qubits. Figure 13 shows the minimum qubits' architectures (connections) for the quantum circuits in Figures 5 and 6. In general, we need not only more qubits, but also more connections among qubits for QSS than QCoin, which further prevents the use of QSS in real quantum computers.

Quantum error correction and NISQs. Aside from errors due to the algorithm (i.e. noise due to a limited number of samples or iterations), the error in quantum computation arises from various factors; bit-flip errors, decoherence of super-position states, errors on logic gates and so on. Note that simulators currently do not include such errors. Although it is theoretically possible to perform error corrections [KLV00], its implementation on current quantum computers is still considered challenging [RDN*12]. It is thus generally assumed that one cannot perform calculation accurately as the scale of a quantum circuit (computation time, the number of qubits, and the number of gate operations) is becoming larger.

Such an 'unscalable' quantum computer is called an 'NISQ' (Noisy Intermediate-Scale Quantum Computer) [Pre18]. On NISQs, quantum algorithms which require many qubits and a long

computation time will not work due to the errors in actual quantum computers. However, NISQs are more realistic models for actual quantum computers in the near future. Therefore, researchers have been vigorously investigating a novel class of quantum algorithm called 'hybrid quantum-classical' [KMT*17]. In this class of algorithms, an algorithm alternately repeats quantum calculations in a small circuit and adjusting parameters of the quantum circuit based on the classical calculation. A hybrid quantum-classical algorithm generally needs fewer qubits and lower depth of quantum circuit, thus suitable to run on NISQs.

In QSS, the relations of AA iterations and estimation error appear as in Figure 12 (right). It is obvious that much more time and larger circuit for one continuous quantum operation are required for QSS than for QCoin. Therefore, we infer that QCoin performs better than QSS due to its shorter computation time. In our experiments and the original experiments by Johnston [Joh16], QSS is not really performing well on NISQs. Based on its performance on a simulator, we hypothesize that its inferior performance on an actual quantum computer is not owing to the limited number of AA iterations, but its use of many qubits and controlled-gate operations. We, however, should mention that we could not run a large enough quantum circuit for QSS due to the restriction of the qubit architecture to fully confirm the influence of decoherence alone.

On the other hand, our QCoin method performs well even on an actual quantum computer. We think that it is because QCoin is hybrid quantum-classical; the use of a quantum coin is done as in classic Monte Carlo integration, while shifting and scaling of the error interval are done by AA in quantum computation. Hybrid quantum-classical algorithms generally need fewer qubits and lower depth of quantum circuit, which is considered suitable to run on NISQs. Although the idea of QCoin was invented a while ago [AW99], there has been no effort to investigate whether QCoin is executable on NISQs, and we think that this finding alone is novel in the field of quantum computing.

8. Limitations

Aside from the limited complexity of integrands due to the current architecture of quantum computers, we have a few more limitations. In classic computers, by giving up the use of random numbers, it is possible to perform quasi Monte Carlo integration [MC95] to achieve the convergence rate of $O(\log(N)^s/N)$ for s-dimension integrands. Although the QCoin's convergence rate of $O(1/N)$ is still better, it is unclear if and how we can incorporate quasi-Monte Carlo to achieve an even better convergence rate in quantum computation, or whether it is possible. The classical part of QCoin is still limited to Monte Carlo integration. Moreover, due to the constraints of hardware, our experiments for both QSS and QCoin on an actual quantum computer omitted the input circuit part, which generally involves controlled-gate operations. As such, if we could have included the omitted input part, errors in our experiments might potentially go up due to the use of more controlled-gate operations.

9. Conclusion

We proposed a concrete algorithm of QCoin and performed numerical experiments for the first time after 20 years of its theoretical introduction 1999. Our implementation of QCoin shows a faster convergence rate than that of classical Monte Carlo integration. This performance is equivalent to QSS. We formulated QCoin as a hybrid quantum-classical method and explained why QCoin is more stable than QSS in the presence of noise in actual quantum computers. We discussed hardware limitations of near-term quantum computers and concluded that QCoin needs fewer qubits and simpler architecture, thus being much more practical than QSS. Our experiments on a quantum computer confirmed this robustness against noise and faster convergence rate than classical Monte Carlo integration. We believe that QCoin is a practical alternative to QSS if we were to run rendering algorithms on quantum computers in the future.

References

[AO19] Aleksandrowicz G., Alexander T., Barkoutsos P., Bello L., Ben-Haim Y., Bucher D., Cabrera-Hernández F. J., Carballo-Franquis J., Chen A., Chen C-F., Chow J M., Córcoles-Gonzales A D., Cross A J., Cross A., Cruz-Benito J., Culver C., González S D. L. P, Torre E D. L., Ding D., Dumitrescu E., Duran I., Eendebak P., Everitt M., Sertage I. F., Frisch A., Fuhrer A., Gambetta J., Gago B. G., Gomez-Mosquera J., Greenberg D., Hamamura I., Havlicek V., Hellmers J., Herok Ł., Horii H., Hu S., Imamichi T., Itoko T., Javadi-Abhari A.. Kanazawa N., Karazeev A., Krsulich K., Liu P., Luh Y., Maeng Y., Marques M., Martín-Fernández F. J., McClure D T., McKay D., Meesala S., Mezzacapo A., Moll N., Rodríguez D. M., Nannicini G., Nation P., Ollitrault P., O'Riordan L. J., Paik H., Pérez J., Phan A., Pistoia M., Prutyanov V., Reuter M., Rice J., Davila A. R., Rudy R. H. P., Ryu M., Sathaye N., Schnabel C., Schoute E., Setia K., Shi Y., Silva A., Siraichi Y., Sivarajah S., Smolin J A., Soeken M., Takahashi H., Tavernelli I., Taylor C., Taylour P., Trabing K., Treinish M., Turner W., Vogt-Lee D., Vuillot C., Wildstrom J. A., Wilson J., Winston E., Wood C., Wood S., Wörner S., Akhalwaya I. Y., Zoufal C.: Qiskit: An open-source framework for quantum computing, 2019.

[AW99] Abrams D. S., Williams C. P.: Fast quantum algorithms for numerical integrals and stochastic processes. *arXiv* (August 1999).

[BdSGT11] Brassard G., Sebastien G., Frederic D., Tapp A.: An optimal quantum algorithm to approximate the mean and its application for approximating the median of a set of points over an arbitrary distance. *arXiv* (June 2011).

[BHT06] Brassard G., Høyer P., Tapp A.: Quantum counting. In *Proceedings of the 25th International Colloquium on Automata, Languages, and Programming* (May 2006), pp. 820–831.

[CZX*18] Chen Z.-Y., Zhou Q., Xue C., Yang X., Guo G.-C., Guo G.-P.: 64-qubit quantum circuit simulation. *Science Bulletin 63*, 15 (2018), 964–971.

[DBE95] Deutsch D. E., Barenco A., Ekert A.: Universality in quantum computation. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences 449*, 1937 (1995), 669–677.

[Gro96] Grover L. K.: A fast quantum mechanical algorithm for database search. In *STOC '96 Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing* (Philadelphia, PA, 1996).

[Gro98] Grover L. K.: A framework for fast quantum mechanical algorithms. In *STOC '98 Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing* (Dallas, TX, 1998).

[IBM] IBMQ: Ibmq experience. https://quantumexperience.ng.bluemix.net/qx/devices (accessed 28 February 2020).

[JHS19] Johnston E. R., Harrigan N., Segovia M. G.: *Programming Quantum Computers: Essential Algorithms and Code Samples*. O'Reilly, Sebastopol, CA, 2019.

[Joh16] Johnston E. R.: Quantum supersampling. In *Proceedings of the SIGGRAPH Talks '16* (2016). https://vimeo.com/180284417.

[Kaj86] Kajiya J. T.: The rendering equation. In *Proceedings of the ACM SIGGRAPH Computer Graphics 20*, (1986), 143–150.

[KLV00] Knill E., Laflamme R., Viola L.: Theory of quantum error correction for general noise. *Physical Review Letters 84*, 11 (2000), 2525.

[KMT*17] Kandala A., Mezzacapo1 A., Temme K., Takita M., brink M., Chow J. M., Gambetta J. M.: Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature 549*, (September 2017), 242.

[LU05] Lanzagorta M., Uhlmann J.: Quantum rendering: An introduction to quantum computing, quantum algorithms and their applications to computer graphics. In *ACM SIGGRAPH 2005 Courses* (2005).

[MC95] Morokoff W. J., Caflisch R. E.: Quasi-Monte Carlo integration. *Journal of Computational Physics 122*, 2 (1995), 218–230.

[NC11] Nielsen M. A., Chuang I. L.: *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, Cambridge, 2011.

[NW99] Nayak A., Wu F.: The quantum query complexity of approximating the median and related statistics. In *STOC '99 Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing* (Atlanta, GA, 1999).

[Par70] Park J. L.: The concept of transition in quantum mechanics. *Foundations of Physics 1*, (1970), 23–33.

[PO18] Pednault E., Gunnels J. A., Nannicini G., Horesh L., Magerlein T., Solomonik E., Draeger E. W., Holland E. T., Wisnieff R.: Breaking the 49-qubit barrier in the simulation of quantum circuits. arXiv:1710.05867 (December 2018).

[Pre18] Preskill J.: Quantum computing in the NISQ era and beyond. *Quantum 2*, (August 2018), 79.

[RDN*12] Reed M. D., DiCarlo L., Nigg S. E., Sun L., Frunzio L., Girvin S. M., Schoelkopf R. J.: Realization of three-qubit quantum error correction with superconducting circuits. *Nature 482*, 7385 (2012), 382.

[Shi] Shimada N. H.: Qcoin's source code on github. (Under construction). https://github.com/N-H-Shimada/QCoin-CGF-2020.

[SO18] Svore K. M., Others: Q#: Enabling scalable quantum computing and development with a high-level DSL. In *Proceedings of the Real World Domain Specific Languages Workshop 2018* (Vienna, Austria, February 2018).

[TKI99] Tokunaga Y., Kobayashi H., Imai H.: Applications or Grover's quantum search algorithm. *IPSJ SIG Notes 70*, 5 (November 1999), 33–40.

[ZJL*15] Zwicker M., Jarosz W., Lehtinen J., Moon B., Ramamoorthi R., Rousselle F., Sen P., Soler C., Yoon S.-E.: Recent advances in adaptive sampling and reconstruction for Monte Carlo rendering. *Computer Graphics Forum 34*, (2015), 667–681.