Modified Iterative Quantum Amplitude Estimation is Asymptotically Optimal

Shion Fukuzawa* Christopher Ho* Sandy Irani* Jasen Zion*

Abstract
In this work, we provide the first QFT-free algorithm for Quantum Amplitude Estimation (QAE) that is asymptotically optimal while maintaining the leading numerical performance. QAE algorithms appear as a subroutine in many applications for quantum computers. The optimal query complexity achievable by a quantum algorithm for QAE is \(O\left(\frac{1}{\epsilon} \log \frac{1}{\alpha}\right)\) queries, providing a speedup of a factor of \(1/e\) over any other classical algorithm for the same problem. The original algorithm for QAE utilizes the quantum Fourier transform (QFT) which is expected to be a challenge for near-term quantum hardware. To solve this problem, there has been interest in designing a QAE algorithm that avoids using QFT. Recently, the iterative QAE algorithm (IQAE) [1] was introduced with a near-optimal \(O\left(\frac{1}{\epsilon} \log \frac{1}{\alpha} \log \frac{1}{\epsilon}\right)\) query complexity and small constant factors. In this work, we combine ideas from the preceding line of work to introduce a QFT-free QAE algorithm that maintains the asymptotically optimal \(O\left(\frac{1}{\epsilon} \log \frac{1}{\alpha}\right)\) query complexity while retaining small constant factors. We supplement our analysis with numerical experiments comparing our performance with IQAE where we find that our modifications retain the high performance, and in some cases even improve the numerical results.

1 Introduction
As we see the rise of the first commercially available programmable quantum computers, quantum algorithms designed over the last few decades that promise various speedups over their classical counterparts are gaining prevalence. However, these devices are still not very large in scale, and use noisy qubits which make it challenging to implement useful algorithms that would require a large number of robust qubits in highly complex circuits using many controlled gates. Because of these restrictions, algorithms that can be run effectively in this so-called NISQ era [2] have been of great interest, where the objective is to find algorithms which can either reduce the number of qubits, the depth of the circuit, and/or the number of controlled gates required.

Two decades ago, Brassard, Hoyer, Mosca, and Tapp [3] introduced the problem of quantum amplitude estimation. In this problem, we are given a unitary operator \(\mathcal{A}\) that acts on \(n + 1\) qubits such that

\[
\mathcal{A}\ket{0}_n \ket{0} = \sqrt{1-a} \ket{\psi_0}_n \ket{0} + \sqrt{a} \ket{\psi_1}_n \ket{1},
\]

where \(a \in [0, 1]\) is unknown and \(\ket{\psi_0}, \ket{\psi_1}\) are two normalized states. Given this setting, the objective is to find a good approximation for \(a\) with as few applications of \(\mathcal{A}\) as possible. They presented the problem as a generalization of Grover’s algorithm [4], and demonstrate that techniques used by Grover could be used to solve QAE. Brassard et al.’s algorithm enables a quadratic speedup for many applications that are solved classically using Monte Carlo simulation, which itself proves useful in finance for applications such as risk analysis [5, 6] or option pricing [7, 8, 9], as well as other general tasks like numerical integration [10]. Furthermore, amplitude estimation is a widely used subroutine in many quantum algorithms, making it an algorithm of high interest to implement in the near future.

Though established as a fundamental algorithm in the field, a challenge in implementing this algorithm is that it requires the ability for the quantum device to perform a large number of controlled gates as it requires the ability to implement controlled unitary gates as well as the quantum Fourier transform (QFT), both of which have been challenging to realize on physical devices. Due to this, there has recently been an increased interest in algorithms that can match the performance of QAE without requiring QFT [1, 11, 12, 13, 14, 15, 16]. It is unlikely that this is sufficient for QAE to become a NISQ compatible algorithm, but our algorithm significantly reduces the width required for the circuit. This implies that if physical devices that can run deep circuits, these algorithms can be beneficial even with a small number of qubits.

Aaronson and Rall [11] introduced QAES, the first QFT-free version of QAE which matches the query complexity of the original algorithm. Though asymptotically tight, the constant overhead of QAES will likely

*University of California, Irvine fukuzawa@uci.edu, christb@uci.edu, irani@ics.uci.edu, jzio@uci.edu
require better quantum hardware to be implemented effectively. Within the same year, two other QFT-free algorithms were introduced by Suzuki et al. [12] and Wie [13]. Suzuki et al.’s algorithm, which is often referred to in the literature as MLAE (Maximum Likelihood Amplitude Estimation), replaces QFT with a classical maximum likelihood estimation to analyze the combined results of the quantum circuit. Numerical experiments were performed to demonstrate the performance of the algorithm, but it is not clear how to prove an upper bound on query complexity for their algorithm.

In 2021, Grinko, Gacon, Zoufal, and Woerner [1] introduce Iterative Quantum Amplitude Estimation (IQAE) which helped remedy the weaknesses of the three above algorithms. Its experimental performance beats that of MLAE, and the query complexity was proven to be just within a log-log factor of the desired complexity. As the name suggests, IQAE is an iterative algorithm, where the approximation of the target value is improved at each round.

In their work, it was shown that IQAE is the best performer numerically amongst the QFT-free algorithms mentioned above [1]. In this work, we show that a modified version of IQAE matches the lower bound on the query complexity of amplitude estimation, while also maintaining the competitive numerical performance that IQAE provides. To achieve this, instead of requiring the same probability of success in each round, we adopt a more intricate sequence for the target probability of success in each round to compensate for the fact that the trials in later rounds require more queries. This allows us to shave off the extra log-log factor from the bound of the original IQAE, and match the lower bound query complexity of $O\left(\frac{1}{c} \log \frac{1}{\alpha}\right)$ for amplitude estimation without requiring QFT. Here, the parameter $c$ is the desired precision of the estimate, and $1 - \alpha$ is the probability that the true amplitude $a$ is $c$ away from the output estimate. In addition to matching the theoretical lower bound, we also demonstrate that our algorithm achieves comparable and, in certain important cases, better numerical performance than the original IQAE.

A recent development of interest to this paper is Rall and Fuller’s work [14] which showed that by a modification in the overshooting condition laid out in IQAE, it was possible to get an increase in query complexity by a factor of two. This amounts to a choice in a hyperparameter for the algorithm. We discuss our experimental observations at the end of section 2.1 and in the results section, but we find that a simple selection for hyperparameter shows a similar improvement in numerical query complexity across all variants of IQAE. There may be other ways to analyze the optimal hyperparameter depending on the experimental setup and other factors, but we believe our suggestion is a reasonably good choice. Furthermore, our work resolves a question they open demonstrating that it is possible to achieve optimal query complexity without a large numerical penalty.

Other notable works on amplitude estimation include Zhao et al. [15] who showed that they are able to reduce the classical cost of the algorithm.

### 1.1 Quantum Amplitude Estimation

A core procedure for all amplitude estimation algorithms mentioned in this paper is amplitude amplification. To describe this, we begin by defining $\theta_a \in [0, \pi/2]$ as the angle that satisfies $\sin^2(\theta_a) = a$, where $a$ is defined in (1.1). Then, given any unitary $A$ defined by (1.1) we see that

$$P[1] = \sin^2((2k + 1)\theta_a). \quad (1.2)$$

As described by Brassard et al. [3] and Grover [4], we can define and easily construct the operator $Q := A S_0 A^\dagger S_{\psi_0}$, where $S_0 := I - 2 |\psi_0\rangle \langle \psi_0| \otimes |0\rangle \langle 0|$ and $S_{\psi_0} := I - 2 |0\rangle_{n+1} \langle 0|_{n+1}$. It has been shown that if we measure the circuit $Q^k A |0\rangle_n |0\rangle$, we have

$$P[1] = \sin^2((2k + 1)\theta_a). \quad (1.3)$$

This process of boosting the probability of measuring a $|1\rangle$ state lies at the core of all QAE algorithms mentioned in this paper, leading to the various speed-ups over naive sampling of the distribution defined by $A$.

To compare our results to the original IQAE paper, we define the query complexity of a QAE algorithm as the total number of times this operator $Q$ is applied. Note that this definition of query complexity can be easily modified to count the number of times $A$ is applied.

### 2 Algorithm

#### 2.1 Overview of IQAE

We will start with an overview of the IQAE algorithm and then describe our modifications to improve the asymptotic query complexity. Note that our description of IQAE is somewhat different from [1]. We believe that the explanation given here distills the essential features of the algorithm and clarifies how IQAE differs from previous approaches.

Like the other QFT-free algorithms, the IQAE algorithm uses the quantum computer to approximate $P[1] = \sin^2((2k+1)\theta_a)$ for the last qubit in $Q^k A |0\rangle_n |0\rangle$ for different powers of $k$. Thus our measurements of the circuits provide us with an approximation of $\theta_a$, which can be converted to an approximation of $a$ within a
desired precision. The algorithm we describe closely follows the structure of IQAE, but we will state all elements of the algorithm and proofs for completeness.

The algorithm has two nested loops and we refer to each iteration of the outer loop as a round. As with all the QFT-free QAE algorithms, IQAE maintains a confidence interval \([\theta_l, \theta_u]\) for \(\theta\) so that it can be proven that \(\theta \in [\theta_l, \theta_u]\) with a certain probability. QAES explicitly narrows down the size of the interval by a constant factor in each iteration. IQAE makes progress towards narrowing the interval more indirectly. In a given round \(i\), the algorithm maintains an odd integer \(K_i\) and implements measurements of the form \(Q^{K_i}A|0\rangle_{ae}|0\rangle\), for \(K_i = 2k_i + 1\). IQAE maintains the invariant that the scaled interval \([K_i\theta_l, K_i\theta_u]\) lies in a single quadrant, described mathematically as

\[
\frac{K_i\theta_l}{\pi/2} = \frac{K_i\theta_u}{\pi/2} - 1. \tag{2.4}
\]

We refer to the value \(R_i = \lfloor K_i\theta_l/(\pi/2) \rfloor\) as the quadrant count, which is the number of complete quadrants passed through starting at an angle of 0 and slowly increasing the angle until \(K_i\theta_l\) or \(K_i\theta_u\) is reached. The rounding is done slightly differently for the upper bound than the lower bound. The only difference occurs when \(K_i\theta_l/(\pi/2)\) or \(K_i\theta_u/(\pi/2)\) are integers. In this case, we would like the upper bound to map to the previous quadrant. This ensures, for example, that for \([\theta_l, \theta_u] = [\pi, 5\pi/4]\) and \([\theta_l, \theta_u] = [5\pi/4, 3\pi/2]\) both angles will have a quadrant count of 2 as can be seen in Figure 1.

![Figure 1: Edge cases for intervals of \([\theta_l, \theta_u]\). For each of the cases pictured above, the quadrant counts \(R_i\) will be 2. Note that the quadrant count begins at 0. Our rounding scheme ensures that this is indeed the case, and the code we used further handles numerical floating errors that were occurring in these regions.](image)

Note that Invariant \((2.4)\) implies that \(K_i\theta_l\) and \(K_i\theta_u\) can both be expressed as

\[
K_i\theta_l = R_i \cdot (\pi/2) + \gamma_l \quad \text{for } \gamma_l \in [0, \pi/2)
\]

\[
K_i\theta_u = R_i \cdot (\pi/2) + \gamma_u \quad \text{for } \gamma_u \in (0, \pi/2]
\]

The invariant also implies that \(\theta_u - \theta_l < \pi/2K_i\). The algorithm makes progress from round to round by increasing the value of \(K_i\) for which the invariant holds. The analysis shows that the value of \(K_i\) will increase by at least factor of 3 in each round.

If Invariant \((2.4)\) holds and \(\theta_a \in [\theta_l, \theta_u]\), then \(K_i\theta_a\) can also be expressed as \(R_i \cdot (\pi/2) + \gamma_a\), for \(\gamma_a \in [0, \pi/2]\). Since \(K_i\) is an odd integer, we can measure \(Q^{K_i}A|0\rangle_{ae}|0\rangle\), for \(k_i = (K_i - 1)/2\), which will return a state ending in \(|1\rangle\) with probability \(\sin^2(K_i\theta_a)\). The value \(\sin^2(K_i\theta_a)\) is equal to \(\sin^2(\gamma_a)\) or \(1 - \sin^2(\gamma_a)\), depending on whether the quadrant count \(R_i\) is odd or even, as can be seen in Figure 2. Using repeated measurements, we obtain a confidence interval \([a_{\min}, a_{\max}]\) for \(\sin^2(K_i\theta_a)\) which can be in turn translated to a confidence interval \([\gamma_{\min}, \gamma_{\max}]\) for \(\gamma_a\). The Chernoff-Hoeffding inequality is used to upper bound the probability that \(\sin^2(K_i\theta_a) \notin [a_{\min}, a_{\max}]\), which is equal to the probability that \(\gamma_a \notin [\gamma_{\min}, \gamma_{\max}]\). Table 1 shows the mapping from the confidence interval \([a_{\min}, a_{\max}]\) to the confidence interval \([\gamma_{\min}, \gamma_{\max}]\) based on the parity of the quadrant count \(R_i\).

The new interval for \(\gamma_a\) gives rise to new values for \(\theta_l\) and \(\theta_u\):

\[
\theta_l \leftarrow \frac{R_i(\pi/2) + \gamma_{\min}}{K_i}\quad \theta_u \leftarrow \frac{R_i(\pi/2) + \gamma_{\max}}{K_i}
\]

Invariant \((2.4)\) trivially holds for the new \(\theta_l\) and \(\theta_u\) as well as the old value of \(K_i\). However, if the interval \([\theta_l, \theta_u]\) has narrowed sufficiently, then the invariant may hold for a larger value of \(K\). Note that if we have a fixed target for the probability that \(\gamma_a \in [\gamma_{\min}, \gamma_{\max}]\), the Chernoff-Hoeffding bound can be used to trade off the number of queries made against the size of the interval. For round \(i\), we denote by \(a_i\) the desired upper bound on probability of failure conditioned on success in all of the previous rounds. The heart of the analysis is to show that the expression \(N_{\max}\), given in Line 11 of

![Figure 2: If the interval \([\gamma_l, \gamma_u]\) lies in an even quadrant, as \(\gamma\) increases, the value \(\sin^2(\gamma_i)\) increases for \(i \in \{l, u\}\). On the other hand, for odd quadrants, as \(\gamma\) increases, the value \(\sin^2(\gamma_i)\) decreases. This drove the calculations in the conversion described in Table 1.](image)
Algorithm 1 Modified IQAE

Require: \( \epsilon > 0, \alpha > 0, N_{\text{shots}} \in \mathbb{N}, \text{Unitary } \mathcal{A} \)

\begin{algorithm}
\begin{align*}
1: & \quad i = 0, k_i = 0 \\
2: & \quad [\theta_l, \theta_u] = [0, \pi/2] \\
3: & \quad K_{\text{max}} := \frac{\pi}{\alpha} \\
4: & \quad \text{while } \theta_u - \theta_l > 2\epsilon \text{ do} \\
5: & \quad \quad N := i, i := i + 1, k_i = k_{i-1} \\
6: & \quad \quad K_i := 2k_i + 1 \\
7: & \quad \quad \alpha_i := 2\alpha K_i^{2} \\
8: & \quad \quad N_i^{\text{max}} = \sin^{-1}(\sqrt{2}/21) \sin^{-1}(8\pi/21) \ln\left(\frac{2}{\alpha_i}\right) \\
9: & \quad \quad R_i = \left\lceil K_i \theta_u / \pi \right\rceil > \text{The number of quadrants passed} \\
10: & \quad \quad \text{to get to current angle} \\
11: & \quad \quad \text{while } k_i = k_{i-1} \text{ do} \\
12: & \quad \quad \quad \text{Prepare circuit } Q^{k_i} \mathcal{A} |0\rangle |0\rangle \\
13: & \quad \quad \quad \text{Measure min}\{N_{\text{shots}}, N_i^{\text{max}} - N\} \text{ times} \\
14: & \quad \quad \quad \text{Combine results in this round to approximate} \\
15: & \quad \quad \quad \hat{a}_i \approx \mathbb{P} [1]. \\
16: & \quad \quad \quad N = N + \min\{N_{\text{shots}}, N_i^{\text{max}} - N\} \\
17: & \quad \quad \quad \epsilon_{a_i} := \sqrt{\frac{1}{2\alpha_i} \ln\left(\frac{2}{\alpha_i}\right)} \\
18: & \quad \quad \quad a_i^{\text{max}} = \min(1, \hat{a}_i + \epsilon_{a_i}) \\
19: & \quad \quad \quad a_i^{\text{min}} = \max(0, \hat{a}_i - \epsilon_{a_i}) \\
20: & \quad \quad \quad \text{Calculate the confidence interval } [\gamma_i^{\text{min}}, \gamma_i^{\text{max}}] \\
21: & \quad \quad \quad \text{for } \{K_i, \theta_i\} \text{ from } [a_i^{\text{min}}, a_i^{\text{max}}] \text{ and } R_i \text{ according} \\
22: & \quad \quad \quad \text{to table } 1 \text{ and} \\
23: & \quad \quad \quad \theta_i = \frac{K_i \theta_u}{\pi} + \gamma_i^{\text{min}} \\
24: & \quad \quad \quad \theta_u = \frac{K_i \theta_u}{\pi} + \gamma_i^{\text{max}} \\
25: & \quad \quad \quad \text{if } \theta_u - \theta_l < 2\epsilon \text{ then} \\
26: & \quad \quad \quad \quad \text{Break} \\
27: & \quad \quad \quad \text{end if} \\
28: & \quad \quad k_i := \text{FindNextK}(k_i, \theta_i, \theta_u) \\
29: & \quad \quad \text{end while} \\
30: & \quad \text{end while} \\
31: & \quad [a_i, a_u] = [\sin^{2}(\theta_i), \sin^{2}(\theta_u)] \\
32: & \quad \text{return } [a_i, a_u] \\
\end{align*}
\end{algorithm}

Algorithm 2 FindNextK

Require: \( k_i, \theta_l, \theta_u \)

\begin{algorithm}
\begin{align*}
1: & \quad K_i = 2k_i + 1 \\
2: & \quad K = \left\lceil \frac{\pi}{2\theta_u - \theta_l} \right\rceil \\
3: & \quad \text{if } K \text{ is even then} \\
4: & \quad \quad K = K - 1 \\
5: & \quad \text{end if} \\
6: & \quad \text{while } K \geq 3K_i \text{ do} \\
7: & \quad \quad \text{if } \left\lceil K \theta_u / \pi \right\rceil = \left\lceil K \theta_u / \pi \right\rceil - 1 \text{ then} \\
8: & \quad \quad \quad \text{return } (K - 1)/2 \\
9: & \quad \quad \text{end if} \\
10: & \quad \quad K = K - 2 \\
11: & \quad \text{end while} \\
12: & \quad \text{return } k_i \\
\end{align*}
\end{algorithm}

Table 1: A table describing the conversion from \( a_i^{\text{min}}, a_i^{\text{max}} \) to \( \gamma_i^{\text{min}}, \gamma_i^{\text{max}} \) respectively. \( \sin^{-1} \) denotes the arcsine function. The top right and bottom left quadrants correspond to an even quadrant count. The top left and bottom right quadrants correspond to an odd quadrant count.
sets the target probability of failure to be the same numerical query complexity. Instead of taking query cost becomes very large which can blow up the depth of the final round. These stages are a rough heuristic characterization of the circuit depth (size of $k_i$) to be sampled. When the circuit is deep, the query cost becomes very large which can blow up the numerical query complexity. Instead of taking $N_{\text{shots}}$ samples per iteration, they introduce a heuristic telling the algorithm to take a smaller number of shots at a time once the circuit reaches a certain depth. This has the effect of reducing the required confidence interval. If the algorithm achieves the confidence of $1 - \alpha$ for all the previous rounds is upper bounded by

$$\mathbb{P}[a \not\in [a_i, a_u]] \leq \alpha$$

where $a_u - a_i < 2\epsilon$, leading to an estimate $\hat{a}$ for a such that $|a - \hat{a}| < \epsilon$ with a confidence of $1 - \alpha$, using $O\left(\frac{1}{\epsilon} \log \frac{1}{\epsilon}\right)$ applications of $A$.

### 3.1 Upper Bound on the Number of Rounds

The algorithm begins with $K_0 = 1$. By definition, the algorithm advances to the next round when FindNextK successfully finds a new value of $K$, and FindNextK only returns a new value if the new $K$ is at least $3K_i$, where $K_i$ is the current value of $K$. Therefore, $K_i \geq 3K_{i-1}$ for each round. The upper bound proven below for the maximum value of $K$ establishes that the number of rounds will be at most $\log_3(\pi/4\epsilon)$.

**Lemma 3.1.** $K_i \leq \frac{\pi}{4\epsilon}$ for all $i$.

**Proof.** By the termination conditions of our algorithm, if FindNextK is called at the end of a round, then we know $\theta_u - \theta_l \geq 2\epsilon$. Furthermore, by the invariant condition for the scaled confidence interval described in equation (2.4), if this call to FindNextK returns a new $K_i$, it must be the case that

$$K_i \leq \frac{\pi/2}{\theta_u - \theta_l}.$$  \hfill (3.5)

We show in Lemma 3.2 that such a $K_i$ always exists. Combining the above, we get

$$K_i \leq \frac{\pi/2}{2\epsilon} = \frac{\pi}{4\epsilon}.$$  \hfill (3.6)

\[ \square \]

### 3.2 Proof that Each Round Successfully Terminates

**Lemma 3.3** establishes that each round will successfully terminate by proving that by the time $N$
Figure 3: An illustration of the inner loop of Algorithm 1. In a given round $i$, we store a variable $k_i$ and $K_i := 2^{k_i} + 1$. We first construct the circuit $Q^{k_i} A |0\rangle_n |0\rangle$ and sample from it $N$ shots times, which gives us a confidence interval for the variable $a_i$. The endpoints of this confidence interval are then converted to variables $\gamma_{i}^{\min}$ and $\gamma_{i}^{\max}$. Dividing these values by $K_i$ gives us a confidence interval for $a$, and we exit the algorithm once the desired precision is reached. If not, we call FindNextK (Algorithm 2) and continue sampling from the circuit if it returns the same $K$, or advance to round $i + 1$ if it returns a new $K$.

reaches $N_{\max}$, FindNextK will successfully find a new value for $K$ which allows the algorithm to advance to the next round. The proof of Lemma 3.3 depends on the following lemma whose result can be stated independent of the algorithm. The proof of Lemma 3.2 is slightly involved and distracting from understanding the dynamics of the algorithm, so it has been moved to the appendix.

LEMMA 3.2. Given $\theta_a, \theta_b$ such that
1. $\theta_b - \theta_a > 0$,
2. $|\sin^2 \theta_b - \sin^2 \theta_a| \leq \sin \frac{\pi}{2} \sin \frac{8\pi}{21}$,
3. the two angles lie in the same quadrant, i.e.,
   \[ \left\lfloor \frac{\theta_a}{\pi/2} \right\rfloor = \left\lfloor \frac{\theta_b}{\pi/2} \right\rfloor - 1, \]

then there exists an odd integer $q \in \left[3, \frac{\pi}{2} / (\theta_u - \theta_l)\right]$ such that
\[ \left\lfloor \frac{q\theta_a}{\pi/2} \right\rfloor = \left\lfloor \frac{q\theta_b}{\pi/2} \right\rfloor - 1. \] (3.7)

Proof. Appendix A. 

LEMMA 3.3. Round $i$ terminates within $N \leq N_{\max}^i$ shots, where
\[ N_{\max}^i := \frac{2}{\sin^2 \left(\frac{\pi}{21}\right) \sin^2 \left(\frac{8\pi}{21}\right)} \ln \left(\frac{2}{\alpha_i}\right). \] (3.8)

Proof. FindNextK returns the largest odd integer in the range $[3K_i, (\pi/2)/(\theta_u - \theta_l)]$ such that
\[ \left\lfloor \frac{K\theta_i}{\pi/2} \right\rfloor = \left\lfloor \frac{K\theta_u}{\pi/2} \right\rfloor - 1. \]

If no such $K$ exists, then FindNextK returns the old value of $K$ (which is $K_i$). We will show that if $N = N_{\max}$, then such a $K$ does in fact exist. This means a new $K$ is found and the algorithm will advance to the next round within the allotted number of shots.

Plugging in $N_{\max}$ for $N$ into the expression in Line 17 in Algorithm 1 for $\epsilon_{a_i}$ gives that $\epsilon_{a_i} = \frac{1}{2} \sin(\pi/21) \sin(8\pi/21)$ and therefore $a_i^{max} - a_i^{min} \leq \sin(\pi/21) \sin(8\pi/21)$. Furthermore, since $\epsilon_{a_i} > 0$, the values of $a_i^{max}$ and $a_i^{min}$ are chosen so that $1 \geq a_i^{max} > \frac{1}{2} \sin(\pi/21) \sin(8\pi/21)$.
$a_i^{min} \geq 0$, which in turn implies that $\gamma_i^{min}$ and $\gamma_i^{max}$ are chosen in Table 1 so that $\pi/2 \geq \gamma_i^{max} > \gamma_i^{min} \geq 0$ and

$$|\sin^2(K_i\theta_u) - \sin^2(K_i\theta_l)| = |\sin^2 \gamma_i^{max} - \sin^2 \gamma_i^{min}| = a_i^{max} - a_i^{min}.$$ 

Also, since $\pi/2 \geq \gamma_i^{max} > \gamma_i^{min} \geq 0$, the values of $\theta_u$ and $\theta_l$ are chosen in Lines 21 and 22 so that

$$\frac{K_i\theta_l}{\pi/2} = \left\lfloor \frac{K_i\theta_u}{\pi/2} \right\rfloor - 1.$$ 

We can now apply Lemma 3.2 with $\theta_b = K_i\theta_u$ and $\theta_a = K_i\theta_l$. Therefore, there is an odd integer in the range from 3 to $(\pi/2)/(\theta_a - \theta_l)$ such that

$$\frac{qK_i\theta_l}{\pi/2} = \left\lfloor \frac{qK_i\theta_u}{\pi/2} \right\rfloor - 1.$$ 

Since $q$ and $K_i$ are both odd integers, $qK_i$ is also an odd integer. Since $q \geq 3$, then $qK_i \geq 3K_i$. Since $q \leq (\pi/2)/(\theta_a - \theta_l)$, then $qK_i \leq (\pi/2)/(\theta_a - \theta_l)$. Therefore the integer $qK_i$ is a viable candidate for a new $K_i$ which will allow the algorithm to advance to the next round.

### 3.3 Proof for the Probability of Success

The sequence $K_0, \ldots, K_t$ has the property that each $K_i$ is in the range from 1 to $K_{max} = \pi/4\epsilon$ and $K_i \geq 3K_{i-1}$. The following lemma will assist in the analysis by allowing us to assume fixed worst-case values for the $K_i$’s. The proof of Lemma 3.3 is given in the appendix.

**Lemma 3.4.** Consider a finite sequence $K_0 = 1, K_1, \ldots, K_t$ such that $K_i \leq K_{max}$ and $K_i \geq 3K_{i-1}$ for all $1 \leq i \leq t$. Suppose that $f$ is a function that is positive and increasing over the range $[1, K_{max}]$, then

$$\sum_{i=1}^{t} f(K_i) \leq \sum_{i=0}^{t-1} f \left( \frac{K_{max}}{3^i} \right).$$  

**Proof.** Appendix B.

We can now show the algorithm succeeds with the desired probability. Lemma 3.5 establishes that with probability at least $1 - \alpha$, the true value of $\theta_a$ lies within the final interval $[\theta_a, \theta_u]$, where $\theta_u - \theta_l \leq 2\epsilon$. This in turn implies that the true value of $a = \sin^2 \theta_u$ lies in the interval $[a_u, a_u] = [\sin^2(\theta), \sin^2(\theta_u)]$. Finally, Lemma 3.6 establishes that $[a_u, a_u]$ is sufficiently small to give the desired bound on the approximation for $a$, namely that $a_u - a_l \leq 2\epsilon$.

**Lemma 3.5.** With probability $\geq 1 - \alpha$ the algorithm returns an estimate satisfying $\theta_a \in [\theta_a, \theta_u]$.

**Proof.** There are a total of $t$ rounds in which the index $i$ goes from 1 to $t$. Consider the last iteration of the inner loop in round $i$, and let $\theta_l^{(i)}$ and $\theta_u^{(i)}$ be the value of $\theta_l$ and $\theta_u$ as assigned in Lines 21 and 22 of the algorithm. The round ends either because FindNextK succeeds in finding a new value for $K$ or because $\theta_u - \theta_l < 2\epsilon$ in which case the algorithm terminates. The final values $\theta_u$ and $\theta_l$ are equal to $\theta_l^{(i)}$ and $\theta_u^{(i)}$. Define $E_i$ to be the event that $\theta_u \in [\theta_l^{(i)}, \theta_u^{(i)}]$. Note that the algorithm starts with $\theta_u^{(0)} = \pi/2$ and $\theta_l^{(0)} = 0$, so $\mathbb{P}[E_0] = 1$.

We will first show that the probability that $\mathbb{P}[E_i \cap E_{i-1}] \leq \alpha_i$. At the beginning of round $i$, the current values for $\theta_l$ and $\theta_u$ are $\theta_l^{(i-1)}$ and $\theta_u^{(i-1)}$, respectively. Furthermore, as a result of the conditions of FindNextK at the end of round $i-1$, $[K_i\theta_l/(\pi/2)] = 1 = [K_i\theta_u/(\pi/2)] = R_i$. If $\theta_u \in [\theta_l^{(i-1)}, \theta_u^{(i-1)}]$, then $K_i\theta_u$ can be expressed as $R_i(\pi/2) + \gamma_u$, where $0 \leq \gamma_u \leq \pi/2$. The measurements $Q^{\alpha_i} A[0]_{u} | 0 \rangle$ return a 1 with probability $a_i = \sin^2(K_i\theta_u)$ which is equal to $\sin^2(\gamma_u)$ or $1 - \sin^2(\gamma_u)$ depending on whether the quadrant $R_i$ is odd or even. After $N$ measurements, we have an estimate for $a_i$, which is denoted by $\hat{a}_i$. By the Chernoff-Hoeffding bound for i.i.d. Bernoulli trials with $N$ samples,

$$\mathbb{P}[|\hat{a}_i - a_i| \geq \epsilon, \hat{a}_i + \epsilon] \leq 2 \exp(-2N\epsilon^2).$$  

(3.10)

Selecting $\epsilon = \sqrt{(1/2N)\ln(2/\alpha_i)}$ gives a probability at most $\alpha_i$. The angles $\gamma^{min}$ and $\gamma^{max}$ are chosen so that the interval between $\sin^2(\gamma^{min})$ and $\sin^2(\gamma^{max})$ is equal to $[\hat{a}_i - \epsilon, \hat{a}_i + \epsilon]$. Therefore, the probability that $\gamma_u$ is not in $[\gamma^{min}, \gamma^{max}]$ is at most $\alpha_i$. If $\gamma_u \in [\gamma^{min}, \gamma^{max}]$, then assigning

$$\theta_u \leftarrow \frac{R_i(\pi/2) + \gamma_u}{K_i}, \quad \theta_l \leftarrow \frac{R_i(\pi/2) + \gamma^{max}}{K_i}$$

guarantees that $\theta_u \in [\theta_l, \theta_u]$. Now we can use the union bound to show that the total error probability is bounded by $\alpha$ as required by Algorithm 1.

$$\mathbb{P}[\theta_u \notin [\theta_l, \theta_u]] = \mathbb{P}[\text{error}]$$

(3.11)

$$\leq \mathbb{P}[\exists i \in \{1, \ldots, t\} : E_i \cap E_{i-1}]$$

(3.12)

$$\leq \sum_{i=1}^{t} \mathbb{P}[E_i \cap E_{i-1}]$$

(3.13)

$$\leq \sum_{i=1}^{t} \mathbb{P}[E_i \mid E_{i-1}] \leq \sum_{i=1}^{t} \alpha_i$$

(3.14)

Now plugging in the chosen values for $\alpha_i$ and using Lemma 3.3 with $f(x) = x$, we get that

$$\sum_{i=1}^{t} \alpha_i = \frac{2\alpha}{3K_{max}} \sum_{i=1}^{t} K_i$$

(3.15)
Therefore the probability of failure is bounded above by \( \alpha \).

**Lemma 3.6.** A confidence interval for \( \theta \) with width at most \( 2\epsilon \) is sufficient to get a confidence interval for \( \alpha \) with width at most \( 2\epsilon \).

**Proof.** Given a confidence interval \([a_l, a_u]\) for \( \alpha \), we can derive a relation to the confidence interval for the parameter \( \theta \) using \( a = \sin^2(\theta) \), and defining \( \sin^2(\theta_i) := a_l \) and \( \sin^2(\theta_u) := a_u \).

\[
\frac{|a_u - a_l|}{2} = \frac{|\sin(\theta_u + \theta_l)||\sin(\theta_u - \theta_l)|}{2} \leq \frac{|\theta_u - \theta_l|}{2}.
\]

So to achieve a confidence interval for \( \alpha \) that is smaller than \( \epsilon \), i.e. \( |a_u - a_l|/2 \leq \epsilon \), it suffices to achieve \( |\theta_u - \theta_l|/2 \leq \epsilon \) for our estimate of \( \theta \).

### 3.4 Bounding the Total Number of Queries

We have proven that the algorithm does indeed succeed with the desired probability. In the next lemma we prove an upper bound on the total number of queries required, which concludes the proof of Theorem 3.1.

**Lemma 3.7.** Algorithm 1 requires at most \( \frac{62}{\epsilon} \ln \frac{6}{\alpha} \) applications of \( A \), achieving the desired query complexity of \( O\left( \frac{1}{\epsilon} \ln \frac{1}{\alpha} \right) \).

**Proof.** The query complexity is defined as the number of times the operator \( Q \) is applied. Each circuit \( Q^k, A \ket{0}, \ket{0} \) applies the operator \( Q \) a total of \( k \) times, and in each round we measure the state created by this state at most \( N_{max} \) times. For simplicity, we define the constant \( C = 1/\left( \sin^2(\pi/21) \sin^2(8\pi/21) \right) \).

\[
N_{queries} \leq \sum_{i=1}^{t} k_i N_{i}^{max}
\]
\[
< \frac{1}{2} \sum_{i=1}^{t} k_i N_{i}^{max}
\]
\[
= C \sum_{i=1}^{t} K_i \ln \left( \frac{2}{\alpha_i} \right)
\]

To invoke Lemma 3.4 notice that \( f(x) = x \ln \left( \frac{x}{2} \right) \) is increasing for all \( x \leq c/e \):

\[
x \leq \frac{c}{e} \Rightarrow 1 \leq \ln \left( \frac{c}{x} \right) \Rightarrow f(x) = \ln \left( \frac{c}{x} \right) - 1 \geq 0.
\]

To apply the lemma to the expression in Equation (3.22), we set \( c = 3K_{max}/\alpha \) and note that for \( 1 \leq K_i \leq K_{max} \), \( K_i \leq K_{max}/\alpha \). So we can use Lemma 3.4 to upperbound the sum:

\[
C \sum_{i=1}^{t} K_i \ln \left( \frac{3K_{max}}{K_i \alpha} \right)
\]
\[
\leq C \sum_{j=0}^{t-i} K_{max} \ln \left( \frac{3j+1}{K_{max} \alpha} \right)
\]
\[
= C \sum_{j=0}^{t-i} K_{max} \ln \left( \frac{3j+1}{\alpha} \right)
\]
\[
= CK_{max} \left( \ln \left( \frac{3}{\alpha} \right) \sum_{j=0}^{t-i} \frac{1}{3j} + \ln\left(3\right) \sum_{j=0}^{t-i} \frac{1}{3j} \right)
\]
\[
\leq CK_{max} \left( \ln \left( \frac{3}{\alpha} \right) \frac{3}{2} + 3 \ln 3 \right)
\]
\[
= CK_{max} \left( \ln \left( \frac{3}{\alpha} \right) \frac{3}{2} + 3 \ln 3 \right)
\]
\[
= \frac{3}{2} \ln \left( \frac{27}{\alpha} \right)
\]
\[
\approx \frac{62}{\epsilon} \ln \left( \frac{6}{\alpha} \right)
\]
\[
= O \left( \frac{1}{\epsilon} \ln \left( \frac{1}{\alpha} \right) \right)
\]

### 3.5 Absolute and Relative Error

Algorithm [1] can be modified to output an approximation of \( \alpha \) to within a \( (1+\epsilon) \) relative error as well, by adding an extra \( 1/\alpha \) overhead. To accomplish this, an extra outer loop is added to Algorithm [1] such that the algorithm will terminate when \( \alpha > 0.5 \). If the algorithm completes the inner loop without achieving our desired approximation, the upper bound is continually halved until the algorithm terminates.
Figure 4: Plots comparing the target precision $\epsilon$ with number of queries made to $A$. The main graph is drawn in a log-log plot over the full range of $\epsilon$’s tested, and the close-up uses a log-lin plot to increase the visibility of the respective lines. As described in the main text, the amplitudes are averaged over varied amplitudes. The other two hyperparameter selections are $\alpha = 0.05$, and $N_{\text{shots}} = 100$.

Figure 5: Experimental results comparing the numerical query complexity when varying the hyperparameter $N_{\text{shots}}$. Regardless of the confidence interval computation method of choice, we see that there is a large improvement in numerical performance across the IQAE variants tested. In (a), the comparison is made between the two variants using the Chernoff-Hoeffding bound, whereas in (b) we used the Clopper-Pearson method (Beta). A greater improvement is seen with the Chernoff method for MIQAE, while the improvement is comparable when using the Beta method.
successfully approximates the targeted amplitude to the correct precision. Details of the modification and proof are outlined in Appendix C.

4 Results
In this section, we present the results of Algorithm 1. We are particularly interested in visualizing the empirical query complexity, and comparing our performance to the original IQAE algorithm. Our code can be found at https://github.com/shifubear/Modified-IQAE.

Our experiments were run over the 3-tuples of parameters from the Cartesian product of the sets \{\texttt{epsilons, amplitudes, confint_methods}\}. Here, \texttt{epsilons} = \{10^{-2}, 10^{-3}, \ldots, 10^{-6}\}, \texttt{amplitudes} = \{0, \frac{1}{16}, \ldots, 1\}, and \texttt{confint_methods} = \{\texttt{chernoff}, \texttt{beta}\}. The term \texttt{beta} refers to the Clopper-Pearson method for determining the confidence interval. The confidence level \(\alpha\) is set to .05 for all experiments.

Figure 4 shows the query complexity of the four IQAE algorithms: Modified IQAE (MIQAE) and the original IQAE with the two difference methods for determining the confidence interval (Chernoff and Beta). Experiments were run using the parameter space described above. The plot shows the total number of queries as a function of the precision parameter epsilon. To discount any anomalies that arise from random sampling, we repeat each parameter combination 10,000 times and average the total number of queries per \(\epsilon\). We also average the results over all amplitudes, since their query complexity should not change based on the ground-truth amplitude. While testing the original IQAE algorithm, we found that amplitudes expressed using powers of 2 (i.e. 0, \frac{1}{2} \ldots) were noticeably easier to estimate. We slightly perturb the ground truth amplitude by sampling \(\tilde{a} \sim \mathcal{N}(a, \frac{1}{16})\). Here, the standard deviation is chosen such that \(\tilde{a}\) does not vary more than 5\% of our 1/16 step distance from \(a\). This reduces any biases in performance due to ground-truth amplitudes that use powers of 2.

Using the Chernoff confidence interval, there is a distinct improvement in both the query complexity and the total number of queries used. The modified IQAE algorithm (green) always makes fewer queries by about half an order of magnitude, when compared to the original (blue).

On the log-log scale, the plotted query complexity for MIQAE+Beta and IQAE+Beta are nearly identical. For finer detail, a second set of error tolerances \{10^{-4}, 7.5 \times 10^{-5}, \ldots, 10^{-6}\} is also tested with the same amplitudes. On a linear-log scale, the original algorithm using the Clopper-Pearson (Beta) interval (orange) appears to use marginally less queries for higher values of \(\epsilon\).

It is possible that the optimal value for each intermediate confidence level \(\alpha_i\) using Clopper-Pearson intervals takes a different form, since \(\alpha_i\) is specifically designed to optimize the Chernoff confidence interval. However, an analytic proof using the beta distribution is intractable and we leave these potential improvements to be discovered.

In Figure 5, we calculate the numerical query complexity when varying the hyperparameter \(N_{\text{shots}}\). As described in an earlier section, this parameter determines how many samples we take from the quantum computer in a given iteration before using the results to recompute a confidence interval. As demonstrated in the plot, there is a substantial improvement for both IQAE and MIQAE regardless of the method of computing the confidence interval, when \(N_{\text{shots}} = 1\). This is true even when taking into account IQAE’s no-overshooting condition, showing that this sampling method is the most general way to extend that heuristic. Because of this, unless there are some resource constraints around how the quantum computer and classical computer communicate that would benefit from batch sampling, we suggest setting this parameter to 1 for the lowest query complexity.

Furthermore, there is a substantial improvement to the query complexity when using MIQAE with the Chernoff-Hoeffding bound as demonstrated in Figure 5 (a). Since the analysis in the paper was centered around this case, this may suggest that our algorithm doesn’t necessitate the no-overshooting condition proposed in IQAE and ChebAE. Instead, the gradual relaxation for the confidence interval requirement implicitly serves this purpose, leading to an improved numerical performance. We comment that this may mean a careful analysis of the algorithm using the Clopper-Pearson (Beta) method could lead to a similar improvement using that technique.

We also analyze the behavior of the algorithms in each individual round. To accomplish this, we record the number of shots \(N\), the Grover power \(k_i\), and the number of queries made at each \(k_i\). As shown in Figure 6, the number of queries per \(k\) increases at the same rate for all combinations of algorithms and confidence intervals. Therefore, the modified algorithm must reduce the total queries made by using fewer shots at higher values of \(k_i\), as Aaronson and Rall observed. This behavior can also be experimentally verified in Figure 6(c) where the modified IQAE uses progressively fewer shots for higher values of \(k\).

Figure 6(c) illustrates the values of the Grover power \(k\) in each round. In all runs, the value of \(k\) never surpasses \(k_{\text{max}}\), supporting the correctness of the IQAE...
algorithm. $k$ increases most slowly between rounds with IQAE+Chernoff, whereas all other configurations increase their $k$’s at similarly fast rates. This explains the results that are shown in each configuration’s query complexity (Figure 4), where the IQAE+Chernoff uses the most queries and all other configurations use a similar number of queries. Evidently, the evolution of $k$ is central to the algorithm’s overall performance. However, it remains unclear whether this relationship is responsible for the improvement seen in the other algorithms over IQAE+Chernoff, or a side effect of controlling the evolution of $N$.

Figure 7 plots the number of queries made for each 3-tuple of experiment parameters against the ground-truth input amplitude. Like Figure 4, the results are averaged across 10,000 runs. The number of queries for any amplitude generally stays the same in both the modified and original algorithms for a specific $\epsilon$. The query count for algorithms using Clopper-Pearson intervals appears to have higher variance between amplitudes, but still averages out to the same level trend. The performance of the Chernoff interval also closely matches that of the Clopper-Pearson interval in our modified algorithm. When using the original IQAE

(a) Number of measurements used as a function of $k_i$.
(b) Number of queries used as a function of $k_i$.
(c) Scaling of $k_i$ as a function of the round number.

Figure 6: Per-round results from running the algorithms with $\epsilon = 10^{-5}$, averaged over 10,000 runs. The red dotted line in the rightmost graph represents $k_{max}$.

(a) Original IQAE
(b) Modified IQAE

Figure 7: Query count vs. amplitude for each combination of algorithm and confidence interval method, recorded at different values of $\epsilon$ described in the legends. The solid lines and dotted lines correspond to the Chernoff-Hoeffding and Clopper-Pearson confidence intervals respectively. On average, the number of queries do not depend on the input amplitude in either algorithm.
algorithm, however, there is a significant gap between the algorithm’s query count depending on which confidence interval method is used. These observations justify our claim that input amplitudes do not affect the algorithm’s query complexity.

Finally, we verified that the algorithms return correct estimations with the desired frequency. According to the proofs, both IQAE algorithms should return an interval that contains the ground-truth amplitude with probability at least $1 - \alpha$. In other words, both algorithms should return an interval that does not contain the true amplitude with frequency at most $\alpha$. We use our previous data, which tests all error tolerance, amplitude, and confidence interval combinations using $\alpha = .05$. We plot the number of failures for a specific amplitude and confidence interval method for each input $\epsilon$ in Figure 8. Overall, the modified algorithm has a higher failure rate than the original, but all failure rates are clearly well below the theoretical bounds.

5 Conclusion and Future Work
In this paper, we show that a modification to Grinko et al.’s IQAE algorithm produces an asymptotically optimal QFT-free quantum amplitude estimation algorithm, and furthermore demonstrate experimentally that this modification is not only a theoretical improvement, but makes the algorithm more suitable to run in the upcoming NISQ era. Future work may include experimenting with this algorithm on real quantum hardware, and analyzing its performance in the presence of noise and noise mitigation techniques.

References

[1] Dmitry Grinko, Julien Gacon, Christa Zoufal, and Stefan Woerner. Iterative quantum amplitude estimation. npj Quantum Information, 7(1):52, December 2021.
[2] John Preskill. Quantum Computing in the NISQ era and beyond. Quantum, 2:79, August 2018.
[3] Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. In Samuel J. Lomonaco and Howard E. Brandt, editors, Contemporary Mathematics, pages 305, 53–74. American Mathematical Society, Providence, Rhode Island, 2002.
[4] Lov K. Grover. A fast quantum mechanical algorithm for database search. In Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing - STOC ’96, pages 212–219, Philadelphia, Pennsylvania, United States, 1996. ACM Press.
[5] Stefan Woerner and Daniel J. Egger. Quantum risk analysis. npj Quantum Information, 5(1):15, December 2019.
[6] Daniel J. Egger, Ricardo Gacía Gutiérrez, Jordi Cahué Mestre, and Stefan Woerner. Credit Risk Analysis using Quantum Computers. arXiv:1907.03044 [quant-ph], 2019.
[7] Patrick Rebentrost, Brajesh Gupt, and Thomas R. Bromley. Quantum computational finance: Monte Carlo pricing of financial derivatives. Physical Review A, 98(2):022321, August 2018.
[8] Nikitas Stamatopoulos, Daniel J. Egger, Yue Sun, Christa Zoufal, Raban Iten, Ning Shen, and Stefan Woerner. Option Pricing using Quantum Computers. Quantum, 4:291, July 2020.
[9] Christa Zoufal, Aurélien Lucchi, and Stefan Woerner. Quantum Generative Adversarial Networks for learning and loading random distributions. npj Quantum Information, 5(1):103, December 2019.
[10] Ashley Montanaro. Quantum speedup of Monte Carlo methods. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, 471(2181):20150301, September 2015.
[11] Scott Aaronson and Patrick Rall. Quantum Approximate Counting, Simplified. arXiv:1908.10846 [quant-ph], January 2020.
[12] Yoichi Suzuki, Shumpei Uno, Rudy Raymond, Tomoki Tanaka, Tamiya Onodera, and Naoki Yamamoto. Amplitude estimation without phase estimation. Quantum Information Processing, 19(2):75, February 2020.
[13] Chu-Ryang Wie. Simpler quantum counting. Quantum Information and Computation, 19(11&12):967–983, September 2019.
[14] Patrick Rall and Bryce Fuller. Amplitude Estimation from Quantum Signal Processing. 2022.

Copyright © 2023 by SIAM
Unauthorized reproduction of this article is prohibited
6 Appendix A: Proof of Lemma 3.2

This proof follows very closely from the proof of Lemma 1 in [1].

Proof. This lemma characterizes the conditions in which a pair of angles that lie in the same quadrant can be multiplied by an odd integer $q$ such that the resulting angle still lies in a single quadrant.

We begin by describing a geometric intuition on when such an odd integer can be found. Consider a circle in which each quadrant has been subdivided into $j \in [3, 5, 7]$ slices. If the current interval $[\theta_a, \theta_b]$ lies within one of these slices, then multiplying that interval by $j$ will guarantee that the scaled interval will still lie inside of a quadrant.

We now define a set of piecewise-linear functions $f_j$ on $[0, \pi/2]$ for integers $j \in [3, 5, 7]$ to capture this geometric intuition as follows. There are a total of $j$ pieces, and the $m$-th piece where $m \in \{1, \ldots, j\}$ is constructed on the interval $\left[\frac{(m-1)\pi}{j}, \frac{m\pi}{j}\right]$. On this interval, the function of interest is defined as

$$f_j(\theta) := \min \left\{ \theta - \frac{(m-1)\pi}{2j}, \frac{m\pi}{2j} - \theta \right\}. \quad (6.34)$$

Figure 9 illustrates the quantity being computed in the case that $j = 3$, and similar images can be used to visualize the case for other values of $j$ as well. This set of functions characterizes the closest distance from a point on a circle to one of the $j$ subdivided points in

![Figure 9: An illustration describing the quantity computed by $f_3$. We only show the first quadrant for illustration purposes. The circular arc is subdivided into 3 sections (green markers). $f_3(\theta)$ returns the angle corresponding to the red arc, describing the angular distance to the nearest subdivided marker from $\theta$.](image-url)
the quadrant it lies in. By combining these functions, it is possible to find the maximum possible interval width centered at the input point such that the scaled interval still lies within a quadrant. The combined function is as follows:

\[
\epsilon_a := \frac{|\sin^2 \theta_b - \sin^2 \theta_a|}{2}, \quad \epsilon_\theta := \frac{\theta_b - \theta_a}{2}.
\]

(6.36)

(6.37)

The following derivation relates \(\epsilon_a\) with \(\theta\) and \(\epsilon_\theta\):

\[
\epsilon_a := \frac{|\sin^2 \theta_b - \sin^2 \theta_a|}{2}
\]

(6.38)

Without loss of generality, the remaining analysis will assume that \(\theta_a, \theta_b\) lie in the first quadrant. The same analysis holds for the other quadrants by symmetry. With this assumption, our analysis of (6.40) will proceed without its absolute values. From \(|\sin^2 \theta_b - \sin^2 \theta_a| \leq \sin \frac{\pi}{21} \sin \frac{8\pi}{21}\) and (6.36), we have an upper-bound for \(\epsilon_a\) given by \(\epsilon_a \leq \frac{1}{2} \sin \frac{\pi}{21} \sin \frac{8\pi}{21}\). Given this upper-bound, the goal is to determine the maximum tolerable \(\epsilon_\theta\) that still allows us to scale the interval such that it lies within a quadrant. To achieve this, we define the following function which is constructed by inverting (6.40):

\[
\epsilon_\theta(\epsilon_a, \theta) := \min \left( \frac{1}{2 \sin(2\theta)} \arcsin \left( \frac{2\epsilon_a}{\sin(2\theta)} \right), \theta, \pi/2 - \theta \right).
\]

(6.41)

The right two terms in the minimum function follow from the constraint that the interval must lie within a quadrant.
single quadrant. Naturally, this constraint forces the interval to be upperbounded by the boundaries of the quadrant. Since we have an upperbound on $\epsilon_a$, the following is the maximum value that $\epsilon_\theta$ can take:

$$
\epsilon_\theta(\theta) := \min \left( \frac{1}{2} \arcsin \left( \frac{\sin \frac{\pi}{21} \sin \frac{3\pi}{21}}{\sin(2\theta)} \right), \theta, \pi/2 - \theta \right). \tag{6.42}
$$

We want to show that $\epsilon_\theta \leq f_{\text{max}}$ for all values of $\theta$ in $[0, \pi/2]$, which means that whenever the size of the interval $[\theta_a, \theta_b]$ is $\epsilon_\theta$, there exists an odd integer $q \geq 3$ such that the interval $[q\theta_a, q\theta_b]$ lies within a single quadrant. The functions $\epsilon_\theta(\theta)$ and $f_{\text{max}}(\theta)$ evaluated near the endpoints of the interval $[0, \pi/2]$ are identical linear functions.

What remains is to show that the inequality holds for the middle section of the function, defined by the first term inside the minimum function from (6.42). The interval where this term takes the minimum can be calculated by solving for the values of $\theta$ where the functions are equal. This interval is

$$
\theta \in \left[ \frac{1}{2} \arcsin \sqrt{S}, \frac{\pi}{2} - \frac{1}{2} \arcsin \sqrt{S} \right], \tag{6.43}
$$

where $S := \sin \frac{\pi}{21} \sin \frac{3\pi}{21}$.

A simple analysis of the piecewise linear function $f_{\text{max}}$ shows that it takes its minimum value of $\frac{\pi}{42}$ when $\theta = \frac{4\pi}{21}$. Evaluating $\epsilon_\theta$ at this value of $\theta$ gives

$$
\epsilon_\theta \left( \frac{4\pi}{21} \right) = \frac{1}{2} \arcsin \left( \frac{\sin \frac{\pi}{21} \sin \frac{3\pi}{21}}{\sin \frac{\pi}{7}} \right) \tag{6.44}
$$

$$
= \frac{\pi}{42}. \tag{6.45}
$$

For all other values of $\theta$, it is clear that $\epsilon_\theta \leq f_{\text{max}}$ which can be seen figuratively in Figure 10. This can be verified by testing that $\epsilon_\theta \leq f_{\text{max}}$ holds within each of the intervals that define the piecewise linear parts of the function manually. Therefore, $\epsilon_\theta \leq f_{\text{max}}$ in the possible values of $\theta$, showing that there indeed exists an odd integer $q \geq 3$ such that the multiplied interval lies in a single quadrant.

\[ \square \]

7 Appendix B: Proof of Lemma 3.4

Proof. To prove this lemma, we first show that for all $j = 1, \ldots, t$, we have

$$
K_j \leq \frac{K_{\text{max}}}{3^{t-j}}, \tag{7.46}
$$

which by the assumption that $f$ is an increasing function implies that

$$
f(K_j) \leq f \left( \frac{K_{\text{max}}}{3^{t-j}} \right). \tag{7.47}
$$

We can then sum over all $j$ and rewrite the indices of the right-hand side to get

$$
\sum_{j=1}^{t} f(K_j) = \sum_{j=1}^{t} f \left( \frac{K_{\text{max}}}{3^{t-j}} \right) = \sum_{i=0}^{t-1} f \left( \frac{K_{\text{max}}}{3^i} \right). \tag{7.48}
$$

Now we show how to prove equation 7.46. We prove this statement inductively backward over the indices $j$. By definition, we know that $K_{j-1} \leq K_j/3$. The base case when $j = t$ holds by definition since $K_t \leq K_{\text{max}}$. By inductive hypothesis, assume that $K_k \leq \frac{K_{\text{max}}}{3^k}$ holds for some $1 < k \leq t$. Then,

$$
K_{k-1} \leq \frac{K_k}{3} \leq \frac{K_{\text{max}}}{3^{k-(k-1)}} \tag{7.49}
$$

$$
\leq \frac{K_{\text{max}}}{3^{t-(k-1)}} \tag{7.50}
$$

which is what we wanted. \[ \square \]

8 Appendix C: Relative Error Algorithm

A key difference between QACS \cite{11} and IQAE \cite{12} were the type of error bounds they used to define their algorithms. IQAE used an absolute error, so our modification to the algorithm also adopts the absolute error bound. QACS used a relative error bound, which is a commonly used error bound when defining approximation algorithms. We show here that the relative error bound can be achieved with an extra $1/a$ overhead, which is optimal as shown in QACS.

**Theorem 8.1.** Given a confidence level $1 - \alpha \in (0, 1)$, a target accuracy $\epsilon > 0$, and an $(n + 1)$-qubit unitary $A$ satisfying

$$
A |0\rangle_n |0\rangle = \sqrt{a} |\psi_0\rangle |0\rangle + \sqrt{1-a} |\psi_1\rangle |1\rangle
$$

where $|\psi_0\rangle$ and $|\psi_1\rangle$ are arbitrary $n$-qubit states and $a \in (0, 1)$, Algorithm 3 outputs a confidence interval for $a$ that satisfies

$$
P[a \not\in [a_l, a_u]] \leq \alpha
$$

where $a_u := a(1+\epsilon)$ and $a_l := a(1-\epsilon)$, leading to a $(1+\epsilon)$-approximation for $a$ with a confidence of $1 - \alpha$, using $O \left( \frac{1}{\epsilon} \log \frac{1}{\alpha} \right)$ applications of $A$.  

Copyright © 2023 by SIAM
Unauthorized reproduction of this article is prohibited
Algorithm 3 Relative IQAE

Require: $\epsilon > 0$, $\alpha > 0$, Unitary $\mathcal{A}$
1: $[a_l, a_u] = [0, 1]$
2: $\epsilon_i = \epsilon$
3: $\hat{a} = 0.5$
4: while $a_u - a_l > 2\hat{a}\epsilon$ do
5: $\epsilon_i = \epsilon_i / 2$
6: $[a_l, a_u] = \text{ModifiedIQAE}(\epsilon_i, \alpha, \mathcal{A})$
7: $\hat{a} = (a_u - a_l) / 2$
8: end while
9: return $[a_l, a_u]$

Proof. We know that Algorithm 1 given an error parameter $\epsilon > 0$ outputs an absolute error confidence interval $[\hat{a} + \epsilon, \hat{a} - \epsilon]$ that contains $a$ with confidence $1 - \alpha$. In the $i$-th iteration of the loop of algorithm 3, ModifiedIQAE is called using $\epsilon_i := \epsilon / 2^i$, which will output an interval satisfying

$$[\hat{a} + \epsilon / 2^i, \hat{a} - \epsilon / 2^i].$$

For $a \in [1 / 2^i, 1 / 2^{i-1}]$, this is a $(1 + \epsilon)$ relative error confidence interval.

From (3.31), we can write the exact query complexity of ModifiedIQAE given $\epsilon_i$ as a parameter. We define the constant $C = 1 / \left( \sin^2 (\pi / 21) \sin^2 (8\pi / 21) \right)$ to simplify the presentation.

$$\frac{3\pi C}{4} \frac{1}{\epsilon_i} \ln \left( \frac{\sqrt{27}}{\alpha} \right) = 3\pi C \frac{2^i}{\epsilon} \ln \left( \frac{\sqrt{27}}{\alpha} \right).$$

If we required $R$ total iterations to achieve the target approximation error, then $a < 1 / 2^R - 1$, and we can sum over this to get the query complexity

$$\frac{3\pi C}{4} \frac{1}{\epsilon} \ln \left( \frac{\sqrt{27}}{\alpha} \right) \sum_{i=1}^{R} 2^i = 3\pi C \frac{1}{\epsilon} \ln \left( \frac{\sqrt{27}}{\alpha} \right) (2^{R+1} - 2) < \frac{3\pi C}{\sin^2 (\pi / 21) \sin^2 (8\pi / 21)} \frac{1}{a} \epsilon \ln \left( \frac{\sqrt{27}}{\alpha} \right) = O \left( \frac{1}{a} \epsilon \ln \left( \frac{1}{\alpha} \right) \right).$$

The inequality follows from $4 / a > 2^{R+1}$. Therefore Algorithm 3 returns a relative error confidence interval with confidence $\alpha$ in the desired query complexity.