Quantum and Randomised Algorithms for Non-linearity Estimation

DEBAJYOTI BERA and SAPV THARRMASHASTHA, IIIT-Delhi

Non-linearity of a Boolean function indicates how far it is from any linear function. Despite there being several strong results about identifying a linear function and distinguishing one from a sufficiently non-linear function, we found a surprising lack of work on computing the non-linearity of a function. The non-linearity is related to the Walsh coefficient with the largest absolute value; however, the naive attempt of picking the maximum after constructing a Walsh spectrum requires $\Theta(2^n)$ queries to an n-bit function. We improve the scenario by designing highly efficient quantum and randomised algorithms to approximate the non-linearity allowing additive error, denoted λ, with query complexities that depend polynomially on λ. We prove lower bounds to show that these are not very far from the optimal ones. The number of queries made by our randomised algorithm is linear in n, already an exponential improvement, and the number of queries made by our quantum algorithm is surprisingly independent of n. Our randomised algorithm uses a Goldreich-Levin style of navigating all Walsh coefficients and our quantum algorithm uses a clever combination of Deutsch-Jozsa, amplitude amplification and amplitude estimation to improve upon the existing quantum versions of the Goldreich-Levin technique.

CCS Concepts: • **Theory of computation → Quantum query complexity**; • **Security and privacy → Mathematical foundations of cryptography**;

Additional Key Words and Phrases: Boolean function, non-linearity, quantum algorithm, query complexity

ACM Reference format:
Debajyoti Bera and Sapv Tharrmashastha. 2021. Quantum and Randomised Algorithms for Non-linearity Estimation. *ACM Trans. Quantum Comput.* 2, 2, Article 5 (July 2021), 27 pages.
https://doi.org/10.1145/3456509

1 INTRODUCTION

Boolean functions are an indispensable tool to design ciphers, codes and algorithms. Of particular interest are "simple" Boolean functions and the "hard" ones. The most common candidates are linear and bent functions, respectively, and the characterisation used for them is their *non-linearity*. Non-linearity of a function is defined as the smallest (Hamming) distance of that function to *any* affine function and is one of the conceptually simplest metric to evaluate a Boolean function.

Our subjects of investigation are n-bit input one-bit output Boolean functions. For any two such functions $f, g : \{0,1\}^n \to \{0,1\}$, define $dist(f,g)$ as the Hamming distance between the truth-tables of f and g. For any $a \in \{0,1\}^n$, define the function $\chi_a : \{0,1\}^n \to \{0,1\}$ as

Authors' addresses: D. Bera and S. Tharrmashastha, IIIT-Delhi, Okhla Phase-3, New Delhi, India 110020; emails: {dbera, tharrmashasthav}@iiitd.ac.in.

2643-6817/2021/07-ART5 $15.00
https://doi.org/10.1145/3456509

$\chi_a(x) = x \cdot a \pmod 2$ in which $x \cdot a$ denotes the binary bitwise dot-product between a and x. It turns out that these functions $\{\chi_a(\cdot)\}_{a\in\{0,1\}^n}$ exactly represent the set of n-bit linear functions—these are the functions for which $g(x \oplus y) = g(x) \oplus g(y)$. Affine Boolean functions (those with algebraic degree 1) are generalisations of linear functions and satisfy $g(x \oplus y) = g(x) \oplus g(y) \oplus b$ for $b \in \{0, 1\}$; they are represented by functions $\chi'_{a,b}(x) = (x \cdot a) \oplus b$. Of course, $\chi_a = \chi'_{a,0}$.

The Walsh coefficient of a function f at a point a is defined as the average correlation of f with χ_a. We normalise it slightly differently to suit our approaches but that does not affect the main results of this work. We use the following definition:

$$\hat{f}(a) = \frac{1}{2^n} \sum_x (-1)^{f(x)\oplus\chi_a(x)} = \frac{1}{2^n} \sum_x (-1)^{f(x)\oplus x\cdot a}.$$

Any Boolean function whose Walsh coefficients have the same absolute value is called a *bent function*; we know from Parseval's inequality that these functions satisfy $\hat{f}(a) = \pm\frac{1}{\sqrt{2^n}}$ for all $a \in \{0, 1\}^n$. Such functions are considered to be the most "non-linear" ones. However, linear functions satisfy $\hat{f}(a) = 1$ only when $f = \chi_a$, and $\hat{f}(a) = 0$ when f is some different linear function. Observe that $\max_a\{|\hat{f}(a)|\}$ (which we denote $\hat{f}_{\max}$) satisfies $\frac{1}{\sqrt{2^n}} \le \hat{f}_{\max} \le 1$—the first equality holds for Bent functions and the second equality holds for linear functions.

The (normalised) non-linearity of any n-bit Boolean function f is defined as the minimum Hamming distance from f to any affine Boolean function; mathematically,

$$\eta(f) = \min_{a,b} \frac{dist(f, \chi'_{a,b})}{2^n}.$$

Using the above notation $0 \le \eta(f) \le \frac{1}{2} - \frac{1}{2} \cdot \frac{1}{2^{n/2}}$; however, this definition is computationally expensive to operate, since one has to enumerate over all possible (exponentially many) linear functions and then compute distance between those functions and f that also requires exponentially many function evaluations, giving a running time of $\Theta(2^{2n})$.

A seemingly simpler alternative arises from the **Walsh-Hadamard (WH)** transform of f. The WH transform generates a 2^n-dimensional WH spectrum whose ath coefficient is $\hat{f}(a)$. It is easy to see that $\hat{f}(a)$ can be expressed as $1 - 2\frac{dist(f,\chi_a)}{2^n} = 1 - 2\frac{dist(f,\chi'_{a,0})}{2^n}$. That is, $\frac{dist(f,\chi'_{a,0})}{2^n} = \frac{1}{2}(1 - \hat{f}(a))$. Similarly, applying the same steps as above to the function $g(x) = f(x) \oplus 1$, it can be shown that $-\hat{f}(a) = 1 - 2\frac{dist(f,\chi'_{a,1})}{2^n}$, i.e., $\frac{dist(f,\chi'_{a,1})}{2^n} = \frac{1}{2}(1 + \hat{f}(a))$.

Combining the expressions above leads us to another expression for $\eta(f)$:

$$\eta(f) = \min_a \left\{\tfrac{1}{2}(1 + \hat{f}(a)), \tfrac{1}{2}(1 - \hat{f}(a))\right\} = \min_a \tfrac{1}{2}(1 - |\hat{f}(a)|) = \tfrac{1}{2} - \tfrac{1}{2}\max_a |\hat{f}(a)| = \tfrac{1}{2} - \tfrac{1}{2}\hat{f}_{\max}.$$

This expression can be computed using the $\Theta(n2^n)$ fast-Walsh-Hadamard transform algorithm and we get a $\Theta(n2^n)$ exact algorithm for computing $\eta(f)$. However, this algorithm has an additional overhead of $\Theta(2^n)$ space compared to the earlier deterministic approach. To the best of our knowledge, there are no better deterministic approaches known with asymptotically better time or space complexity for arbitrary Boolean functions. Therefore we focus on randomised, sampling-based, approaches. The subject of this article is the `non-linearity estimation` problem where we want to estimate with high probability the non-linearity with λ additive accuracy, for any given λ.

$$\boxed{\text{determine } 0 \le a < b \le 1 \qquad \text{such that } b - a \le \lambda \qquad \text{and} \qquad \Pr[a < \eta(f) < b] \ge 1 - \delta.} \quad (1)$$

Recall that $\hat{f}(a)$ can also be expressed as the *expected* correlation of $f(x)$ and $\chi_a(x)$ for x sampled uniformly at random. Suppose we denote this correlation as $Corr(f)$. We can use standard approaches for estimating $\mathbb{E}[Corr(f)]$; the number of samples of x (and hence the running time

complexity) to estimate $\mathbb{E}[Corr(f)]$ with additive accuracy λ and error at most δ will be $O(\frac{1}{\lambda^2}\log\frac{1}{\delta})$. There is no additional space overhead in these approaches. However, since $\eta(f)$ depends upon the Fourier coefficient with the largest absolute value, the time and query complexity still runs into $\Theta(2^n)$. In this article, we design algorithms with query complexities that are polynomial in $(n, \lambda, \log\frac{1}{\delta})$.

Non-linearity is an important property of Boolean functions and trying to compute or estimate it for a function, given either as a black-box or in some other representation, is a natural question in the realm of Boolean functions. However, beyond this academic curiosity lies the connection of non-linearity to other hardness measures of Boolean functions. In a recent paper, Boyar et al. [6] considered 5 common measures apart from nonlinearity (algebraic degree, annihilator immunity, algebraic thickness, normality, and multiplicative complexity) and obtained relationships among them; for example, they show that low multiplicative complexity implies low non-linearity and vice versa. Many of these measures, including non-linearity, are used to design cryptographic ciphers and hash-functions with interesting properties like collision-resistance [6] and propagation characteristics [9]. Even though we leave out these interesting applications out of the scope of this article, it would be worthwhile to understand the best use of a non-linearity estimation algorithm in cryptography [21].

The design of our quantum algorithm could be of independent interest. We were recently able to use the idea therein to formulate quantum algorithms for a few variants of the element distinctness problem [4].

1.1 Related Work

To highlight the computational challenge of computing, or even estimating, the non-linearity of a Boolean function given as a black-box, recently Bera et al. [2] investigated this question in the context of the well-known linearity testing algorithm that was proposed by **Blum-Luby-Rubinfeld (BLR)** [5]. The BLR test evaluates a Boolean function given in the form of a black-box, always accepting a linear function but sometimes accepting a non-linear function as well. They showed that the probability of false-positive in a BLR test is not monotonic with non-linearity, and hence, found it challenging to *compute* non-linearity by employing BLR.

However, if we want to *estimate* non-linearity, allowing some inaccuracy, then the above observation need not be a show-stopper. Indeed, using p to denote the probability that the BLR test accepts a function f, it can be shown that $p = \frac{1}{2} + \frac{1}{2}\sum_a \hat{f}^3(a) \leq \frac{1}{2} + \frac{1}{2}\max_a \hat{f}(a)$. Suppose one runs the BLR test multiple times to get a close estimate $\hat{p}$ of p. Then it may be possible to estimate the lower bound $\max_a \hat{f}(a) \gtrapprox 2\hat{p} - 1$. However, the trouble is that $2\hat{p} - 1$ can be positive or negative and, if negative, we fail to get any bound on $\hat{f}_{\max}$.

Hillery et al. proposed a property testing quantum algorithm that makes $O(\frac{1}{\epsilon^{2/3}})$ queries [14] and this was subsequently improved to $O(\frac{1}{\epsilon^{1/2}})$ queries [10]. But we faced hurdles when we tried to adapt these property testing algorithms that identify if a function f is linear (i.e., $\eta(f) = 0$) or is ϵ-far from linear (i.e., $\eta(f) \geq \epsilon$). Since we do not consider promise problems *ala* property testing in this article, so, if f is neither linear nor ϵ-far (for any guessed ϵ), the algorithm may erroneously return "linear" or "ϵ-far" and we get no insights whatsoever.

Consider the simpler problem of computing $x^* = \arg\max_x |\hat{f}^2(x)|$, and for simplicity, assume unique x^*; estimating $\eta(f)$ is easy with the knowledge of x^* (a randomised algorithm for this is given by Lemma 3.2 and a quantum algorithm is given by Lemma 4.1). It is known that the quantum circuit used in the Deutsch-Jozsa problem generates the state $\sum_x \hat{f}(x)\,|x\rangle$ that, when observed, gives us a state $|x\rangle$ sampled from the distribution $\{\Pr[x] = \hat{f}^2(x)\}$. Thus it is tempting

Table 1. Our Results on Estimating $\eta(f)$ with Additive Error λ (Ignoring Logarithmic Factors)

Worst-case complexity	Algorithm		Lower bound
	Query complexity	Number of qubits	Query complexity
Randomised algorithm	$O(n/\lambda^6)$ [Theorem 5.2]	—	$\Omega(1/\lambda)$ [Theorem 6.6]
Quantum algorithm	$O(1/\lambda^3)$ [Theorem 5.1]	$2n + 1 + O\left(\log\left(\frac{1}{\lambda^2}\right)\right)$	$\Omega(1/\sqrt{\lambda})$ [Theorem 6.5]

Note the independence of n in the quantum algorithm complexity.

to make multiple observations of independent runs of the Deutsch-Jozsa circuit and return the majority observation; the idea is that x^* has the largest probability in the entire spectrum , and so, may have the largest probability among the observed samples. This is the scenario of using the mode of a few i.i.d. samples as an estimator of the mode of a discrete distribution. However, Dutta et al. showed that if $\min_{y \neq x^*}(\Pr[x^*] - \Pr[y]) \geq g(n)$, then the number of samples required is $O(\frac{n}{g^2} \log \frac{1}{\delta})$ [11, Theorem 4]. But this upper bound can be as large as exponential, since we observed that $g(n)$ could be $O(1/2^{1.5n})$ for n-bit functions.[1]

To the best of our knowledge, there exist very limited attempts toward this problem, even considering the classical computing framework. We are aware of an algorithm for non-linearity computation of a sparse Boolean function (sparsity is with respect to the truth table) [8]; however, neither that approach provides any accuracy guarantees nor it is designed in the usual black-box query model—there the function is required to be given in its algebraic normal form. In a recent pre-print [15], its authors related $\hat{f}^2_{max}$ to the Gower's U_2 norm of f as $||f||^4_{U_2} \leq \hat{f}^2_{max}$ and suggested that $||f||^4_{U_2}$ can be used to obtain a lower bound on $\hat{f}^2_{max}$ (which implies an upper-bound on non-linearity). There is a quantum circuit proposed by us in an earlier work [3] to estimate $||f||^4_{U_2}$ with additive accuracy λ using $\tilde{O}(\frac{1}{\lambda^2})$ queries.[2] However, this approach does not control the accuracy of the estimate of $\hat{f}^2_{\max}$, since there is no known theoretical upper bound on $\hat{f}^2_{max} - ||f||^4_{U_2}$.

1.2 Overview of Results

This article resolves a few important questions in the light of the earlier discussions that non-linearity appears difficult without querying f on exponentially many inputs. The success of quantum query algorithms against Boolean functions motivated us to look into quantum algorithms. Can non-linearity be estimated with an additive constant inaccuracy using exponentially few queries to f? Or, even constant many queries? What is the minimum number of queries needed if accuracy is not a constant? What about classical randomised algorithms? After all, the BLR test is pretty effective.

Our techniques are primarily quantum in nature, but we also obtain results for randomised algorithms along the way. Here we are interested in query complexity, and so our algorithms require access to a unitary representation of a Boolean function f (denoted U_f). λ denotes the accuracy parameter and δ denotes the maximum allowed probability of error. Our results are summarised in Table 1.

Our algorithms are most suitable for estimating non-linearity up to a constant or poly-logarithmic bits of precision (the smallest non-zero non-linearity requires roughly $n/2$ bits after

[1]Choose f with Hamming distance 1 from a bent function, say, h. It is straightforward to show that if $dist(f, h) = 1$, then $\hat{f}(x) = \hat{h}(x) \pm \frac{2}{2^n}$; we obtain that the smallest gap between $\hat{f}^2_{\max}$ and $\hat{f}^2_{2\text{ndmax}}$ is $\frac{8}{2^{3n/2}}$.

[2]The authors claimed a query complexity of $\tilde{O}(\frac{1}{\lambda})$ [15, Equation 28]. However, they used an incorrect form of Hoeffding's inequality. Using the correct inequality gives a query complexity of $\tilde{O}(\frac{1}{\lambda^2})$ to obtain the estimate of the Gower's U_2 norm.

the decimal point). It is easy to show that $|\hat{f}(x) - \hat{f}(y)| \geq \frac{2}{2^n}$ whenever $\hat{f}(x)$ and $\hat{f}(y)$ are distinct. Therefore, non-linearity can be exactly computed if we set $\lambda = 2/2^n$. Our lower bounds say that to compute non-linearity exactly, classically there is nothing better than querying f at all the 2^n points; however, a $O(2^{n/2})$-query quantum algorithm probably exists.

A notable feature of our algorithm is that, unlike the classical "fast Walsh-Hadamard transformation" approaches, our algorithms are iterative in nature requiring little additional space. Further, there is very little overhead in the running time on top of the queries to U_f. Hence the query complexity above directly translates to its time complexity as well, with $poly(n)$ overhead arising from the additional gates required to perform amplitude estimation, amplification and small sub-circuits.

1.3 Overview of Techniques

We repeatedly estimate the probability p of an observation upon measuring the final state of a quantum circuit A. Using quantum amplitude estimation [7] we can obtain an estimate $\tilde{p}$ such that $\Pr[|p - \tilde{p}| \geq \epsilon] \leq \delta$ for any $\epsilon \leq \frac{1}{4}$ using a total of $\Theta(\frac{1}{\epsilon} \log \frac{1}{\delta})$ calls to A (details given as Corollary A.2 in Appendix A). Note that a classical algorithm for the same task would require $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta})$ calls to A, or as we would often state, $\tilde{O}(\frac{1}{\epsilon^2})$ calls after ignoring log-factors.

We also use quantum amplitude amplification, in particular, its fixed point version [22], which gives us a quantum circuit that when measured at the end gives us a good state, if any such exists. If the probability of the original algorithm is lower bounded by p, then the number of iterations, hence queries, can be upper bounded by $O(\sqrt{1/p})$. The fixed point version could have been replaced with other amplitude amplification variations that require only a lower bound on the success probability [7]; however, we prefer the fixed point version, since it does not involve any intermediate measurement and can be used inside another amplitude amplification.

Both our randomised and quantum algorithms to estimate non-linearity with inaccuracy λ actually estimate $\hat{f}^2_{\max}$ with inaccuracy $\Theta(\lambda^2)$. The latter is implemented as a binary search, named `IntervalSearch`, to find the *largest threshold* τ, among a discrete set of thresholds that depend on λ, such that $\hat{f}^2_{\max} \geq \tau$. The difficulty lies in solving the `BoundFMax` decision problem that, given τ, decides if $\hat{f}^2_{\max} \geq \tau$ in sub-exponential time. There is a technical challenge in getting binary search to act, since the algorithms use estimations to guide the search. The estimations have an additive error, and we have to be careful during the comparisons with the estimated values made by the binary search. Our technical contributions here are a randomised algorithm and a quantum algorithm for the `BoundFMax` problem.

The classical randomised approach, named `CBoundFMax`, searches among all $\hat{f}(x)^2$ values; however, it uses the idea of the Goldreich-Levin algorithm [12] to restrict search among only a small subset of values. Value of any specific $\hat{f}(x)^2$ is of course not readily available, but that can be easily estimated using a few $f(x)$ values sampled randomly. The number of queries is linear in n and scales inversely with τ^3. It is possible to convert this algorithm to a quantum one, but we would end up with a complexity that scales as $\frac{n}{\tau^2}$—indeed, that is the complexity of the quantum versions of the Goldreich-Levin algorithms that have been proposed so far [17, 19].

To understand how `QBoundFMax` gets rid of the dependence on n, it will be useful to understand `CBoundFMax`. Think of ϵ to be something that is smaller than τ, say, $\tau/2$. `CBoundFMax` performs a level-order traversal of a binary tree built on all possible binary prefixes of length up to n. At any particular node, say, p, the algorithm estimates $PWC(p^\frown 0)$ where PWC at any prefix q is defined as $PWC(q) = \sum_{x \in \{0,1\}^{n-|q|}} \hat{f}^2(q^\frown x)$ where $q^\frown x$ denotes the concatenation of q with x. Note that for $PWC(p^\frown 0)$, the summation is over all n-bit x with prefix $p^\frown 0$ and $PWC(p^\frown 1)$ is defined similarly.

It identifies those 1-bit extensions of p for which $PWC(p^\frown b) \geq \tau - \epsilon$ and adds them to a queue. Once all the nodes of a level is processed, the nodes in the queue are retrieved and processed in the manner described above. At the final level $l = n$, $PWC(p) = \hat{f}^2(p)$ for any n-bit prefix p. If any prefix at the final level satisfy $PWC(p) \geq \tau$, then the algorithm concludes that $\hat{f}^2_{max} \geq \tau$.

It is immediate that the CBoundFMax algorithm has three components that contribute to its complexity; (a) the estimation of $PWC(p)$ with additive accuracy ϵ that takes $O(\frac{1}{\epsilon^2})$ queries, (b) the total number of prefixes added to the queue at any particular level l, which is $O(\frac{1}{\tau-2\epsilon})$ by applying Parseval's identity, and (c) the outer loop for the level-order traversal, which is $O(n)$. Hence the total query complexity of CBoundFMax is $O(\frac{n}{\epsilon^2(\tau-2\epsilon)})$ queries.

The CBoundFMax algorithm can be improved if we reduce the complexity of any of the three components of CBoundFMax and replace the classical loops and data structures with their quantum equivalent ones. The most trivial way to improve the complexity is to replace the classical estimation with the quantum estimation. While classical estimation uses $O(\frac{1}{\epsilon^2})$ queries, its quantum counterpart (implemented using the Deutsch-Jozsa circuit) uses $O(\frac{1}{\epsilon})$ queries leading to the final query complexity $O(\frac{n}{\epsilon(\tau-2\epsilon)})$ (this algorithm is explained in Appendix C.1). This is exactly what has been proposed earlier as the quantum version of Goldreich-Levin [17, 19] in which we set $\tau = \lambda^2$ and $\epsilon = \tau/2$ to get a list of all x such that $\hat{f}^2_{\max}(x) \geq \lambda^2$.

The above algorithm runs a classical subroutine around a quantum circuit (for estimating $PWC(p)$ on an eligible p) in each level. Its query complexity can be improved by using a single quantum circuit for the entire operations of a level: (i) Estimation of $PWC()$, followed by (ii) filtering based on comparison with τ. This can be implemented by generating a superposition of all eligible p in a level, say, $\sum_p c(p)\,|p\rangle$ where $|c(p)|^2 \approx PWC(p)$, running amplitude estimation without the measurement to store $|c(p)|^2$ in some register and then comparing the value in this register to that of τ to mark some of the $|p\rangle$s in the superposition (see Appendix C.2 for the entire algorithm). The dependence of the query complexity on n remains there, but it nevertheless improves to $\tilde{O}(\frac{n}{\epsilon\sqrt{\tau-2\epsilon}})$.

To remove the dependence on n, we remove the classical level-order traversal altogether and replace the equally superposed initial state by an initial state in which each basis state has amplitude proportional to its Walsh coefficient. Using a clever combination of Deutsch-Jozsa, amplitude amplification and amplitude estimation, QBoundFMax manages to achieve a complexity of $\tilde{O}(\frac{1}{\epsilon\sqrt{\tau-2\epsilon}})$ queries. What is remarkable is that the final algorithm can be implemented as a single quantum circuit (see Figure 2) unlike many quantum algorithms that are essentially classical wrappers around amplitude amplification and amplitude estimation.

2 INTERVAL SEARCH FOR $\hat{F}^2_{MAX}$

In this section, we consider the problem of estimating an interval $J \subseteq (0, 1)$ of length $|J| \leq \epsilon$ such that $\hat{f}^2_{max} \in J$ with high probability. We will use this as stepping stone for estimating $\hat{f}^2_{max}$ with any desired additive accuracy ϵ. Let k be the smallest integer such that $\frac{1}{2^k} \leq \epsilon/2$. For finding J, our IntervalSearch algorithm for the above problem divides the interval $[0, 1]$ into sub-intervals of length $\frac{1}{2^k}$ and then finds the right-most (i.e., toward 1) sub-interval that contains *any* non-zero Fourier coefficient-squared (i.e., $\hat{f}^2(a)$ for any a). Clearly, $\hat{f}^2_{max}$ must belong to the same interval.

To implement the above strategy, we need to first solve the following problem that we call as BoundFMax : Given a function $f(x)$ as a blackbox, a threshold $\tau \in (0, 1)$, and accuracy g (we will refer to this as the "gap") perform the following with probability of error at most δ:

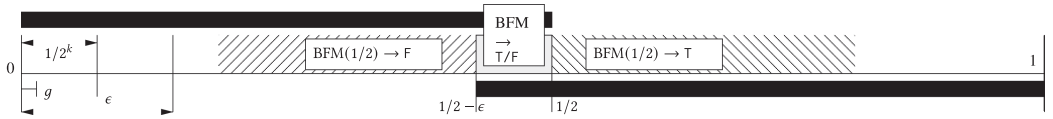

Fig. 1. Explanation of the first binary-search step of the IntervalSearch algorithm for a scenario when ϵ is a power of 2, say, 2^{2-k}. BFM(1/2) denotes the output of the BoundFMax algorithm using $\tau = \frac{1}{2}$. The black rectangles indicate the two possible ranges that will be searched next depending upon BoundFMax(1/2). The shaded regions indicate the outputs of the BFM calls for different values of $\hat{f}^2_{\max}$.

- Return TRUE if $\hat{f}^2_{max} \geq \tau$.
- Return FALSE if $\hat{f}^2_{max} < \tau - 2g$.
- Return anything if $\hat{f}^2_{max} \in [\tau - 2g, \tau)$.

Classical and quantum algorithms for the BoundFMax problem are explained in Sections 3 and 4, respectively. For now, assume that we have an homonymous algorithm for the problem.

The IntervalSearch algorithm is described in Algorithm 1. It uses *binary search* to find the rightmost interval (i.e., with highest boundary values) among all the 2^k sub-intervals that contain $\hat{f}^2_{\max}$. It uses BoundFMax to decide whether to consider the "right-half" or "left-half" of the currently processing interval. To handle the third case of BoundFMax, we equipped IntervalSearch to search among slightly overlapping intervals.

ALGORITHM 1: Algorithm IntervalSearch to find out an ϵ-lengthinterval containing $\hat{f}^2_{\max}$

Require: accuracy ϵ and probability of error δ
Set $k = \left\lceil \log_2 \frac{1}{\epsilon} \right\rceil + 1$ ▷ k is the smallest integer *s.t.* $\frac{1}{2^k} \leq \frac{\epsilon}{2}$; thus, $\frac{\epsilon}{4} < \frac{1}{2^k} \leq \frac{\epsilon}{2}$
Set gap $g = \frac{1}{8}\left(\epsilon - \frac{1}{2^k}\right)$ ▷ $8g + \frac{1}{2^k} = \epsilon \implies \frac{3}{2}\frac{\epsilon}{16} \geq g \geq \frac{\epsilon}{16}$
Set boundaries $lower = \frac{1}{2^n}$, $upper = 1$ and threshold$\tau = \frac{1}{2}$
for $i = 1 \ldots k$ **do**
  if BoundFMax$(\tau, g, \frac{\delta}{k}) \rightarrow$ TRUE **then**
    Update $lower = \tau - 2g$, $\tau = \tau + \frac{1}{2^{i+1}}$; $upper$ is unchanged
  else
    Update $upper = \tau$, $\tau = \tau - \frac{1}{2^{i+1}}$; $lower$ is unchanged
  end if
end for
return $[lower, upper)$

It will be easier to understand IntervalSearch from the illustration given in Figure 1 in which we have assumed ϵ to be a power of $\frac{1}{2}$; in this case $\frac{1}{2^k} = \frac{\epsilon}{2}$ and the gap g used for BoundFMax is $\frac{\epsilon}{16}$. It is easy to verify that the search interval $[lower, upper)$ before round 1 has length $1 - \frac{1}{2^n}$ and before round i has length $\frac{1}{2^{i-1}} - \frac{1}{2^n}$ (if BoundFMax has returned FALSE in all the previous $i-1$ rounds) or $\frac{1}{2^{i-1}} + 2g$ (if BoundFMax has returned TRUE in any of the previous rounds). Note that τ is always "midway" of a search interval excluding g, the overhead due to the gap.

If BoundFMax returns TRUE, then we are sure that $\hat{f}^2_{\max}$ lies in the right-half of the interval (with a slight overhead of $2g$ at the lower boundary)—accordingly, the lower-boundary and the threshold are moved right. However, if BoundFMax returns FALSE, then we are sure that $\hat{f}^2_{\max}$ lies in the

left-half of the interval; so, the upper-boundary and the threshold are moved left. Thus, before and after each round it is ensured that $lower \leq \hat{f}^2_{\max} < upper$ for the current values of *lower* and *upper*.

Binary search ends after the kth round. The interval contains $\hat{f}^2_{\max}$ and its length is at most $\frac{1}{2^k} + 2g = \epsilon - 6g < \epsilon$.

Theorem 2.1. *Let f be an n-bit Boolean function. Given an additive accuracy ϵ and probability of error δ, there is a quantum algorithm of query complexity $\tilde{O}(\frac{1}{\epsilon^{3/2}})$ and a classical algorithm of query complexity $\tilde{O}(\frac{n}{\epsilon^3})$ that outputs an interval of length at most ϵ that, with probability at least $1-\delta$, contains $\hat{f}^2_{\max}$.*

Proof. The Algorithm `IntervalSearch` makes k invocations to `BoundFMax`. Let $c_b(\tau, g, \delta/k)$ be the number of calls that `BoundFMax` makes to the oracle to solve its problem with threshold τ, accuracy g and error δ/k. Then, the total number of calls to U_f is upper bounded by $\sum_{i=1}^{k} c_b(\tau_i, g, \delta/k)$ in which τ_i denotes the value of τ in the i-step of the binary-search.

Using the classical implementation of `BoundFMax` (Lemma 3.3), we have $\sum_{i=1}^{k} c_b(\tau_i, g, \delta) = \frac{n}{g^2}\tilde{O}(\sum_{i=1}^{k} \frac{1}{\tau_i - 2g})$. The way `IntervalSearch` sets k and g ensures that $\frac{\epsilon}{2} \leq 8g = \epsilon - \frac{1}{2^k} < \frac{3\epsilon}{4}$. Furthermore, all $\tau_i \geq \frac{1}{2^k} > \frac{\epsilon}{4} \Rightarrow \tau_i - 2g > \frac{\epsilon}{16}$. This leads to the classical query complexity of $\tilde{O}(\frac{n}{(\epsilon/16)^2 \cdot \epsilon/16}) = \tilde{O}(\frac{n}{\epsilon^3})$ further hiding a $O(\log \frac{1}{\epsilon})$ factor.

With the quantum implementation of `BoundFMax` (Lemma 4.2) we have $\sum_{i=1}^{k} c_b(\tau_i, g, \delta) = \frac{1}{g}\tilde{O}(\sum_{i=1}^{k} \frac{1}{\sqrt{\tau_i - 2g}})$. Therefore, we get the number of calls as $\tilde{O}(\frac{1}{(\epsilon/16)\cdot\sqrt{\epsilon/16}}) = \tilde{O}(\frac{1}{\epsilon^{3/2}})$.

As for the error, there are k calls to `BoundFMax` that is allowed to return an incorrect answer with probability at most $\frac{\delta}{k}$. Therefore, there is an overall probability of δ that any of those calls return an incorrect answer. □

3 CLASSICAL RANDOMISED ALGORITHM FOR BOUNDFMAX

Goldreich and Levin proposed a randomised algorithm for learning the "high" Walsh coefficients of a Boolean function [12]. Our randomised algorithm follows the presentation of this algorithm as a level-order traversal of a binary tree. We borrow from the book by O'Donnell [20] the definition $\mathbf{W}^{S|\bar{J}}[f]$, denoted here as **PrefixWalshCoefficients (PWC)** of f at a prefix a, and Proposition 3.40 as the forthcoming lemma.

Definition 3.1 (PrefixWalshCoefficients). For a prefix $a \in \{0,1\}^s$ such that $0 \leq s \leq n$, $PWC(a) = \sum_{x \in \{0,1\}^n} \hat{f}^2(x)$ where the summation is over all n-bit x with prefix a.

It immediately follows that if a is n-bit, then $PWC(a) = \hat{f}^2(a)$.

Lemma 3.2 ([20]). *There is a $O(\frac{1}{\epsilon^2}\log\frac{1}{\delta})$-query classical algorithm, denoted PWCE, for estimating $PWC(a)$ for a prefix a within $\pm\epsilon$ and with probability at least $1-\delta$. In particular, this algorithm can estimate $\hat{f}^2(b)$ for an n-bit b.*

We include a proof of the lemma for completeness.

Proof. $\hat{f}(b)$ can be expressed as $\mathbb{E}_x(-1)^{f(x)\oplus x\cdot b}$; so it can be estimated by simply averaging $(-1)^{f(x)\oplus x\cdot b}$ for some uniformly chosen random $x \in \{0,1\}^n$—the number of queries required follow from Hoeffding's bound for ± 1 random variables.

Estimating $PWC(a)$ requires expressing it too as the expectation of a ± 1 random variable as shown below; here the length of a is denoted k.

$$\begin{aligned}
PWC(a) &= \sum_{b\in\{0,1\}^{n-k}} \hat{f}(ab)^2 = \tfrac{1}{2^{2n}} \sum_{b\in\{0,1\}^{n-k}} \left(\sum_{x\in\{0,1\}^n} (-1)^{f(x)\oplus x\cdot(ab)} \right)^2 \\
&= \tfrac{1}{2^{2n}} \sum_{b\in\{0,1\}^{n-k}} \left(\sum_{x,\,y\in\{0,1\}^n} (-1)^{f(x)\oplus f(y)\oplus x\cdot(ab)\oplus y\cdot(ab)} \right) \\
&= \tfrac{1}{2^{2n}} \sum_{\substack{x_1,\,y_1\in\{0,1\}^k \\ x_2,\,y_2\in\{0,1\}^{n-k}}} \sum_{b\in\{0,1\}^{n-k}} (-1)^{f(x_1x_2)\oplus f(y_1y_2)\oplus (x_1x_2)\cdot(ab)\oplus(y_1y_2)\cdot(ab)} \qquad \text{(Split } x = x_1x_2,\, y = y_1y_2\text{)} \\
&= \tfrac{1}{2^{2n}} \sum_{\substack{x_1,\,y_1\in\{0,1\}^k \\ x_2,\,y_2\in\{0,1\}^{n-k}}} (-1)^{f(x_1x_2)\oplus f(y_1y_2)\oplus x_1\cdot a\oplus y_1\cdot a} \sum_{b\in\{0,1\}^{n-k}} (-1)^{(x_2\oplus y_2)\cdot b} \\
&= \tfrac{2^{n-k}}{2^{2n}} \sum_{\substack{x_1,\,y_1\in\{0,1\}^k \\ x_2=y_2\in\{0,1\}^{n-k}}} (-1)^{f(x_1x_2)\oplus f(y_1x_2)\oplus x_1\cdot a\oplus y_1\cdot a} \qquad (\textstyle\sum_b (-1)^{z\cdot b} \text{ is 0 if } z\neq 0, \text{ else } 2^{n-k}) \\
&= \tfrac{1}{2^{n-k}2^k2^k} \sum_{\substack{x_1,\,y_1\in\{0,1\}^k \\ x_2\in\{0,1\}^{n-k}}} (-1)^{f(x_1x_2)\oplus f(y_1x_2)\oplus x_1\cdot a\oplus y_1\cdot a} \\
&= \mathbb{E}_{x_1,y_1,x_2}[(-1)^{f(x_1x_2)\oplus f(y_1x_2)\oplus x_1\cdot a\oplus y_1\cdot a}]. \qquad \square
\end{aligned}$$

Our classical algorithm CBoundFMax for the BoundFMax problem is sketched in Algorithm 2.

ALGORITHM 2: Algorithm CBoundFMax

Require: threshold $\tau \in (0,1)$, confidence$\epsilon \in (0,\tau)$, error $\delta \in (0,1)$
  Initialise a FIFO list $Q = \{\varepsilon\}$, where ε denotes the empty string
  while Q is not empty **do**
    remove prefix $\mathfrak{p}$ from Q
    for suffix $\mathfrak{s}$ from $\{0, 1\}$ **do**
      childprefix $\mathfrak{cp} = \mathfrak{p}^\frown\mathfrak{s}$
      obtain estimate $e \leftarrow PWCE(\mathfrak{cp}, \epsilon, \left(\frac{\tau-\epsilon}{2n}\right)\delta)$ ▷ With accuracy ϵ and error $\delta' = \frac{\tau-\epsilon}{2n}\delta$
      if $e \geq \tau - \epsilon$ **then**
        If $len(\mathfrak{cp}) < n$, add $\mathfrak{cp}$to Q
        Else (i.e., $len(\mathfrak{cp}) = n$), **return** TRUE
      end if
    end for
  end while
  return FALSE

Lemma 3.3. *Algorithm* CBoundFMax *solves the* BoundFMax *problem using* $\tilde{O}(\frac{n}{\epsilon^2(\tau-2\epsilon)})$ *queries.*

Proof. The algorithm traverses in level-order a binary tree on all strings of lengths up to n to find some x such that $\hat{f}^2(x) \geq \tau$; children of a node $a \in \{0,1\}^s$ are denoted $a^\frown 0$ and $a^\frown 1$.

It uses the observation that if $PWC(a) = \sum_{b\in\{0,1\}^{n-s}} \hat{f}^2(a^\frown b)$ is less than $\tau - \epsilon$, then there cannot be any x with prefix a for which $\hat{f}^2(x) \geq \tau - \epsilon$, and hence the subtree under a need not be further explored. However, there may be an inaccuracy in estimating $PWC(a)$, which we handle in the two cases below.

Case—CBoundFMax *returns* TRUE. This happens only when the algorithm finds some n-bit $\mathfrak{cp}$ for which the PWC estimate e satisfies $e \geq \tau - \epsilon$. From Lemma 3.2, e satisfies $PWC(\mathfrak{cp}) - \epsilon \leq e \leq$

$PWC(\mathfrak{cp}) + \epsilon$ with probability at least $1 - \delta'$. Furthermore, $PWC(\mathfrak{cp}) = \hat{f}^2(\mathfrak{cp})$. Combining all these results, we see that the following holds with high probability:

$$\tau \leq e + \epsilon \leq (PWC(\mathfrak{cp}) + \epsilon) + \epsilon = \hat{f}^2(\mathfrak{cp}) + 2\epsilon \qquad \Rightarrow \qquad \hat{f}^2_{\max} \geq \tau - 2\epsilon.$$

Case—CBoundFMax *returns* FALSE. For this case, assume that on the contrary $\hat{f}^2_{\max} \geq \tau$, i.e., there is some n-bit x for which $\hat{f}^2(x) \geq \tau$. Therefore, for all prefixes $\mathfrak{p}$ of x, $PWC(\mathfrak{p}) \geq \hat{f}^2(x) \geq \tau$. From Lemma 3.2, with probability at least $1 - \delta'$, e satisfied $PWC(\mathfrak{p}) - \epsilon \leq e \leq PWC(\mathfrak{p}) + \epsilon$. For a moment assume that the estimator of PWC makes no error; thus, $e \geq \tau - \epsilon$ and when that holds, $\mathfrak{p}$ is added to Q and eventually retrieved and processed. Since the above fact holds for all prefixes of x, so, all of them will be stored in Q and retrieved, which means that x will also be added to Q and retrieved and processed. When $\mathfrak{p} = x$, $e \geq PWC(x) - \epsilon = \hat{f}^2(x) - \epsilon \geq \tau - \epsilon$. Thus, CBoundFMax should be returning TRUE when that happens—this leads to a contradiction. Therefore, when CBoundFMax returns FALSE it must be true that, with high probability, $\hat{f}^2_{\max} < \tau$.

Let $PWCE$ be the classical estimator for PWC. The total number of calls to $PWCE$ will be at most $2 \times (\frac{n}{\tau - 2\epsilon})$—there are two suffixes to try for each prefix and due to Parseval's identity, for every length $s \in \{1, \ldots n\}$, there are at most $\frac{1}{\tau - 2\epsilon}$ prefixes $\{\mathfrak{p}_1, \mathfrak{p}_2, \ldots\}$ of length s such that $PWCE(\mathfrak{p}_i, \epsilon, \delta') \geq \tau - \epsilon$ where δ' is some error. This is because for any prefix $\mathfrak{p}$, $PWC(\mathfrak{p}) < \tau - 2\epsilon \Rightarrow PWCE(\mathfrak{p}, \epsilon, \delta) < \tau - \epsilon$. So we have $PWCE(\mathfrak{p}, \epsilon, \delta') \geq \tau - \epsilon \Rightarrow PWC \geq \tau - 2\epsilon$. Since at any level the total number of prefixes $\mathfrak{p}$ such that $PWC(\mathfrak{p}) \geq \tau - 2\epsilon$ is at most $\frac{1}{\tau - 2\epsilon}$, we have that the total number of prefixes $\mathfrak{p}$ such that $PWCE(\mathfrak{p}, \epsilon, \delta') \geq \tau - \epsilon$ is at most $\frac{1}{\tau - 2\epsilon}$. The query-complexity is obtained by combining the number of calls to $PWCE$ with Lemma 3.2.

The algorithm works in a flawless manner as described if all the $PWCE$ calls are within their promised accuracy with no error. Since $PWCE$ is called with error parameter $\delta' = (\frac{\tau - 2\epsilon}{2n})\delta$ therefore the probability of CBoundFMax facing any error is at most δ. □

4 QUANTUM ALGORITHM FOR BOUNDFMAX

In the previous section, we obtain a classical algorithm to solve the BoundFMax problem using $O(\frac{n}{\epsilon^2(\tau - 2\epsilon)})$ queries. Note that the estimation of PWC in Algorithm 2 is done classically. A naive approach to reduce the complexity of the algorithm is to replace the classical estimation by a quantum algorithm for PWC estimation.

This quantum algorithm simply executes the Deutsch-Jozsa circuit and measure the first s qubits in the standard basis. The probability of observing $|a\rangle$ is $\sum_{b \in \{0,1\}^{n-s}} \hat{f}^2(ab) = PWC(a)$. It directly follows that $PWC(a)$ can be estimated using amplitude estimation. The number of calls to U_f that is required to achieve additive accuracy ϵ and probability of error δ is $O(\frac{1}{\epsilon} \log \frac{1}{\delta})$.

Lemma 4.1. *There is a $O(\frac{1}{\epsilon} \log \frac{1}{\delta})$-query quantum algorithm for estimating $PWC(a)$ for a prefix a within $\pm\epsilon$ and with probability at least $1 - \delta$. The algorithm computes $\hat{f}^2(b)$ for an n-bit b.*

Using the quantum estimation subroutine for PWC estimation inside Algorithm 2 gives us a simple quantum algorithm for the BoundFMax problem. The number of queries can easily be shown to be $\tilde{O}(\frac{n}{\epsilon(\tau - 2\epsilon)})$, which is already better compared to the classical algorithm discussed in Lemma 3.3. We now explain how to remove the dependency on n and improve the dependency on $(\tau - 2\epsilon)$.

Our quantum algorithm QBoundFMax is described in Algorithm 3 and a quantum circuit for the same is illustrated in Figure 2. It is to be noted that Ri is used to represent the ith register used in the quantum circuit corresponding to the algorithm. We use DJ to represent the circuit for Deutsch-Jozsa: $H^{\otimes n} \cdot U_f \cdot H^{\otimes n}$ and we use a few smaller circuits listed below.

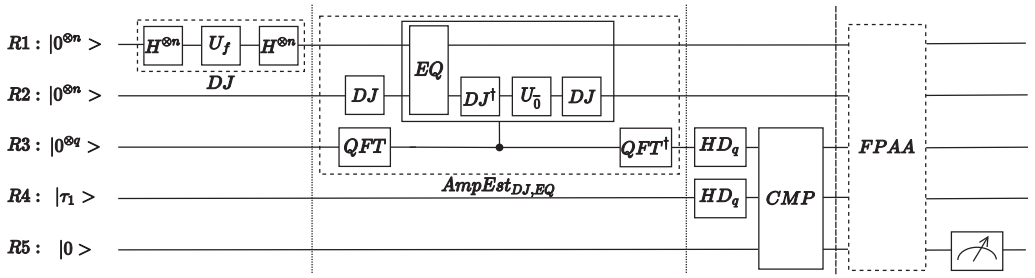

Fig. 2. The QBoundFMax circuit. The four stages are separated using vertical lines.

ALGORITHM 3: Algorithm QBoundFMax

Require: Threshold τ, accuracy ϵ and error δ.

1: Set $\tau' = \tau - \epsilon$ and $q = \lceil \log\left(\frac{1}{\epsilon}\right) \rceil + 3$
2: Set $\tau_1 = \left\lfloor \frac{2^q}{\pi} \sin^{-1}(\sqrt{\tau'}) \right\rfloor$.
3: Initialize the circuit as $|0^n\rangle |0^n\rangle |0^q\rangle |\tau_1\rangle |0\rangle$. Fourth register is on q qubits.
4: **Stage 1:** Apply $DJ = H^{\otimes n} \cdot U_f \cdot H^{\otimes n}$ on $R1$.
5: **Stage 2:** Apply quantum amplitude estimation (*AmpEst*) on DJ with $R2$ as the input register, $R3$ as the precision register and $R1$ is used to determine the "good state." *AmpEst* is called with error at most $\delta/2$ and additive accuracy $\frac{1}{2} \cdot \frac{1}{2^q}$.
6: **Stage 3:** Use $\mathsf{HD_q}$ on $R3$ and $R4$ separately.
7: Use CMP on $R3$ and $R4 = |\tau_1\rangle$ as input registers and $R5$ as output register.
8: **Amplification stage:** Apply Fixed Point Amplitude Amplification (FPAA) $\frac{\sqrt{2}}{\sqrt{\delta(\tau-2\epsilon)}}$ times on $R5$ with error at most $\delta/2$ and measure $R5$ as m.
9: **if** $m = |1\rangle$ **then**
10: **return** TRUE
11: **else**
12: **return** FALSE
13: **end if**

EQ: Checking for equality of two n-bit strings, EQ maps $2n$-qubit basis states $|x, y\rangle$ to $(-1)|x, y\rangle$ if $x = y$ and does nothing otherwise.

HD_q: When the target qubit is $|0^q\rangle$, and with a $q-$bit string y in the control register, HD computes the absolute difference of y_{int} from 2^{q-1} and outputs it as a string where y_{int} is the integer corresponding to the string y. It can be represented as $\mathsf{HD}_q |y\rangle |b\rangle = |b \oplus \tilde{y}\rangle |y\rangle$ where $y, b \in \{0,1\}^q$ and $\tilde{y}$ is the bit string corresponding to the integer $|2^{q-1} - y_{int}|$. Even though the operator HD requires two registers, the second register will always be in the state $|0^q\rangle$ and shall be reused by uncomputing (using $HD^\dagger$) after the CMP gate. Hence, we have not explicitly mentioned it in Algorithm 3 and Figure 2. For all practical purposes, this operator can be treated as the mapping $|y\rangle \mapsto |\tilde{y}\rangle$.

CMP: CMP is defined as $\mathsf{CMP}|y_1\rangle |y_2\rangle |b\rangle = |y_1\rangle |y_2\rangle |b \oplus (y_1 \leq y_2)\rangle$ where $y_1, y_2 \in \{0,1\}^n$ and $b \in \{0,1\}$ and it simply checks if the integer corresponding to the basis state in the first register is at most that in the second register.

The EQ circuit is trivial to implement, but the other two are slightly non-trivial. We have discussed their implementation details in Appendix D.

The algorithm acts in four stages that we first describe. Then we analyse its behaviour.

First stage. In the first stage, DJ is applied to $R1$ to get $R1$ in the state $\sum_z \hat{f}(z)\,|z\rangle$.

Second stage. To understand the second stage, it will be easier to assume that $R1$ is in the state $|z\rangle$ for some fixed $z \in \{0,1\}^n$. The second stage is about estimating the amplitude of $|z\rangle$ in the output of $DJ\,|0^n\rangle$ (to be computed afresh in $R2$—not to be confused with the DJ on $R1$) and putting the estimate, with some precision q in $R3$, basically creating the state $\left|\widetilde{\hat{f}^2(z)}\right\rangle$ in $R3$—here $\widetilde{\hat{f}^2(z)}$ denotes an integer representation of $\hat{f}^2(z)$. The EQ gate is used to select the "good state" as $|z\rangle$ during the amplitude estimation. So at the end of stage-1 and 2 we have in $R1, R3$, roughly speaking, $\sum_z \hat{f}(z)\,|z\rangle\left|\widetilde{\hat{f}^2(z)}\right\rangle$.

Third stage. The third stage is about filtering all those basis states $|z\rangle$ in $R1$ for which $\hat{f}^2(z) \geq \tau$—notice that $\hat{f}^2(z)$ is now available, in some form, in $R3$. The HD and CMP gates do the filtering using $R3$ and $R4$, which has a function of τ. The states R_1, R_3, R_4, and R_5 after stage-3 can be written as $\sum_z \hat{f}(z)\,|z\rangle\left|\widetilde{\hat{f}^2(z)}\right\rangle|\tau_1\rangle\,|p_z \leq p_{\tau_1}\rangle$ where $p_z = \sin^2\left(\frac{\pi}{2^q}\widetilde{\hat{f}^2(z)}\right)$ and $p_{\tau_1} = \sin^2(\frac{\pi}{2^q}\tau_1)$, and we show that $p_z \leq p_{\tau_1}$ is equivalent to $\hat{f}^2 \geq \tau$. Thus, at the end of this stage we get in $R5$ a non-zero amplitude for $|1\rangle$ if $\hat{f}^2(z) \geq \tau$ for some $|z\rangle$ in $R1$. The trick of initialising $R1$ to $\sum_x \hat{f}(x)\,|x\rangle$ comes in handy now, since it ensures that the probability of seeing $R5$ in $|1\rangle$ is lower bounded by $\tau - 2\epsilon$, if at all there are any such z.

Final amplification stage. The lower-bound on the probability of success allows us to reduce the error further using fixed point amplitude amplification in the final stage. It is probably clear that the success of this scheme hinges on the performance of amplitude estimation; unfortunately, amplitude estimation can be imprecise and erroneous and requires careful setting of parameters to control its behaviour.

We now prove that Algorithm 3 manages to iron out the subtleties and solve the BoundFMax problem as desired. For the proof we will assume that the desired probability of error is at most a constant, say 1/2.

Lemma 4.2. *The Algorithm* QBoundFMax *makes* $O(\frac{1}{\epsilon\sqrt{\tau-2\epsilon}})$ *calls to the oracle and with error at most* δ *behaves as follows:*

(1) if QBoundFMax *returns* TRUE *then* $\hat{f}^2_{max} \geq \tau - 2\epsilon$
(2) if QBoundFMax *returns* FALSE *then* $\hat{f}^2_{max} < \tau$.

when given an $n-$*bit Boolean function* f *as an oracle, a threshold* τ*, accuracy parameter* ϵ *and error parameter* δ *such that* $\tau > 2\epsilon$.

Alternatively, we get that (a) if $\hat{f}^2_{\max} \geq \tau$ *then the algorithm returns* TRUE *and (b) if* $\hat{f}^2_{\max} < \tau - 2\epsilon$ *then it returns* FALSE.

Proof. First, we justify the correctness of the algorithm.

The state of the system after the first stage, just before applying amplitude estimation, is

$$\sum_{x\in\{0,1\}^n} \hat{f}(x)\,|x\rangle\,|0^n\rangle\,|0^q\rangle\,|\tau_1\rangle\,|0\rangle\,.$$

On applying amplitude estimation, with $R2$ as input register, $R3$ as precision register and R1 as the register containing the superposition of the good states, we obtain a superposition of strings in $R3$

as shown below:

$$|x\rangle \cdot \sum_w \hat{f}(w)\,|w\rangle \cdot |0^q\rangle \xrightarrow{AmpEst} |x\rangle \cdot \sum_w \hat{f}(w)\,|w\rangle \cdot \Big(\alpha_x\,|y_x\rangle + |E\rangle\Big).$$

Here $|\alpha_x|^2 \geq 1-\delta/2$ is the probability of successful estimation of the probability of $|x\rangle$ in $R2$, and $|E\rangle$ denotes the state of $R3$ when it does not. Let $|y_x\rangle$ be one such state in $R3$, which corresponds to the output of estimating $\hat{f}^2(x)$—the probability of the state $|x\rangle$ in $R1$. By the property of amplitude estimation we have

$$\sin^2(\pi\tfrac{y_x}{2^q}) \in [\hat{f}^2(x) - \tfrac{1}{2^q}, \hat{f}^2(x) + \tfrac{1}{2^q}] \tag{2}$$

with probability at least $1-\delta/2$. We can write the state after the second stage as

$$\sum_{x\in\{0,1\}^n} \hat{f}(x)\,|x\rangle\,|\zeta\rangle\,[\alpha_x\,|y_x\rangle + |E\rangle] \cdot |\tau_1\rangle\,|0\rangle\,,$$

in which $|\zeta\rangle$ indicates the fixed state $\sum_w \hat{f}(w)\,|w\rangle$ that does not play any role in the rest of the circuit and can be ignored for all practical purposes. We will analyse the error arising with probability at most δ toward the end of this proof, but until then assume that there is no error in the amplitude estimation step.

Next, on applying HD on $R3$ we get a q-bit integer $\widetilde{y_x}$ in $R3$ such that $\widetilde{y_x} = |2^{q-1} - y_x|$. Similarly, we obtain $\tilde{\tau_1}$ in $R4$ such that $\tilde{\tau_1} = |2^{q-1} - \tau_1|$. Now, subroutine CMP compares the string $\widetilde{y_x}$ in $R3$ against $\tilde{\tau_1}$ in $R4$ and sets $R5$ as $|1\rangle$ if $\widetilde{y_x} \leq \tilde{\tau_1}$. We claim that CMP actually ends up comparing y_x against τ_1. We need the following propositions to prove this claim. □

Proposition 4.1. *For any two angles $\theta_1, \theta_2 \in [0,\pi]$, $\sin\theta_1 \leq \sin\theta_2 \iff |\frac{\pi}{2} - \theta_1| \geq |\frac{\pi}{2} - \theta_2|$.*

Proof. The proof uses trigonometric identities and transformations.

$$\sin\theta_1 \leq \sin\theta_2 \iff \cos\left(\frac{\pi}{2} - \theta_1\right) \leq \cos\left(\frac{\pi}{2} - \theta_2\right) \equiv \cos\left|\frac{\pi}{2} - \theta_1\right| \leq \cos\left|\frac{\pi}{2} - \theta_2\right|.$$

Now, since $\theta_1 \in [0,\pi]$, we have $(\frac{\pi}{2} - \theta_1) \in [-\frac{\pi}{2}, \frac{\pi}{2}]$. This in turn implies $|\frac{\pi}{2} - \theta_1| \in [0, \frac{\pi}{2}]$. The proposition now follows from the fact that $\cos\theta$ is decreasing in the range $\theta \in [0, \frac{\pi}{2}]$. □

Proposition 4.2. *For any two q-bit integers u, v,*

$$\left|2^{q-1} - v\right| \leq \left|2^{q-1} - u\right| \iff \sin^2\left(\pi\frac{v}{2^q}\right) \geq \sin^2\left(\pi\frac{u}{2^q}\right).$$

Proof. If z is a q-bit integer, then $\pi\frac{z}{2^q} \in [0,\pi]$. Therefore, $\sin^2(\pi\frac{v}{2^q}) \leq \sin^2(\pi\frac{u}{2^q})$ is equivalent to $\sin\left(\pi\frac{v}{2^q}\right) \leq \sin\left(\pi\frac{u}{2^q}\right)$. Using Proposition 4.1, this is equivalent to $|\frac{\pi}{2} - \pi\frac{v}{2^q}| \geq |\frac{\pi}{2} - \pi\frac{u}{2^q}|$, which is same as $|2^{q-1} - v| \geq |2^{q-1} - u|$. □

Using the above proposition we get that $\widetilde{y_x} \leq \tilde{\tau_1}$ holds if and only $\sin^2(\pi\frac{y_x}{2^q}) \geq \sin^2(\pi\frac{\tau_1}{2^q})$ holds, which is equivalent to the inequality $y_x \geq \tau_1$, since both $\sin\left(\pi\frac{y_x}{2^q}\right)$ and $\sin\left(\pi\frac{\tau_1}{2^q}\right)$ are non-negative. The states of the registers after the third stage can be written as

$$\sum_{\substack{x\in\{0,1\}^n \\ y_x \geq \tau_1}} \hat{f}(x)\alpha_x\,|x\rangle\,|\zeta\rangle\,|y_x\rangle\,|\tau_1\rangle\,|1\rangle + \sum_{\substack{x\in\{0,1\}^n \\ y_x < \tau_1}} \hat{f}(x)\alpha_x\,|x\rangle\,|\zeta\rangle\,|y_x\rangle\,|\tau_1\rangle\,|0\rangle + |E'\rangle$$

in which $|E'\rangle$ denotes the states obtained after amplitude estimation went wrong in stage-2.

Next, we want to analyse the state of $R5$ with respect to $\hat{f}^2_{\max}$ and τ. For that we use Equation (2) and the fact that q was chosen to satisfy $\frac{1}{2^q} \leq \frac{\epsilon}{8}$.

First, let $u \in \{0,1\}^n$ be any point such that $\hat{f}^2(u) \geq \tau$. Then amplitude estimation in Line 5 of QBoundFMax will output a q-bit integer y_u such that $\sin^2(\pi \frac{y_u}{2^q}) \geq \hat{f}^2(u) - \frac{1}{2^q} \geq \tau - \frac{1}{2^q} \geq \tau - \frac{\epsilon}{8} \geq \tau - \epsilon$. Further,

$$\sin^2\left(\pi \frac{y_u}{2^q}\right) \geq \tau - \epsilon, \tag{3}$$

$$\equiv \sin\left(\pi \frac{y_u}{2^q}\right) \geq \sqrt{\tau - \epsilon}, \tag{4}$$

$$\equiv y_u \geq \lfloor \tfrac{2^q}{\pi} \sin^{-1}\sqrt{\tau - \epsilon} \rfloor = \tau_1. \tag{5}$$

So for such a u the register $R5$ is going to be set to $|1\rangle$.

Next consider some v such that $\hat{f}^2(v) < \tau - 2\epsilon$. Then amplitude estimation will output a string y_v such that $\sin^2(\pi \frac{y_v}{2^q}) \leq \hat{f}^2(v) + \frac{1}{2^q} < \tau - 2\epsilon + \frac{1}{2^q}$. Here we require the following proposition whose proof involves standard trigonometric identities (see Appendix E).

Proposition 4.3. *τ and τ_1 satisfy* $0 \leq \tau - \epsilon - \sin^2(\pi \frac{\tau_1}{2^q}) \leq \frac{2\pi}{2^q}$.

From this proposition we get $\sin^2(\pi \frac{\tau_1}{2^q}) - (\tau - 2\epsilon) \geq \epsilon - \frac{2\pi}{2^q} \geq \frac{8}{2^q} - \frac{2\pi}{2^q} > \frac{1.5}{2^q}$, which implies that $\sin^2(\pi \frac{\tau_1}{2^q}) > (\tau - 2\epsilon) + \frac{1}{2^q}$, which is used to show that $\sin^2(\pi \frac{y_v}{2^q}) < \tau - 2\epsilon + \frac{1}{2^q} < \sin^2(\pi \frac{\tau_1}{2^q})$. Since y_v and τ_1 belong to $[0, 2^q]$, this implies that $y_v < \tau_1$, so $R5$ will be set to $|0\rangle$ for such a v.

Therefore, we have proved that if the state in $R5$ is $|1\rangle$ then $\hat{f}^2_{\max} \geq \tau - 2\epsilon$ and if the state is $|0\rangle$ then $\hat{f}^2_{\max} < \tau$. The probability of $R5$ to be $|1\rangle$ is $\sum_x |\alpha_x \hat{f}(x)|^2$ summed over all x for which $R5$ is set to $|1\rangle$; as explained above, this is either 0 or at least $(1 - \frac{\delta}{2})(\tau - 2\epsilon)$. The rest of the circuit simply applies fixed-point amplitude amplification that only increases the probability of $R5$ to be in the state $|1\rangle$. So now we move to analysing the query complexity of the Algorithm.

It is straightforward to observe that the number of calls made by amplitude estimation in QBoundFMax is $O(2^q \log(\frac{2}{\delta})) = O(\frac{1}{\epsilon}\log(\frac{1}{\delta}))$. The fixed point amplitude amplification is performed $O(\frac{1}{\sqrt{1-\delta/2}\sqrt{\tau-2\epsilon}})$ iterations and amplification uses $O(\frac{1}{\epsilon}\log(\frac{1}{\delta}))$ many queries to the oracle of the function. Since δ can be assumed to a small constant, say at most $1/2$, hence, the algorithm QBoundFMax makes $O(\frac{1}{\epsilon}\log\left(\frac{1}{\delta}\right)) \cdot (O(\frac{1}{\sqrt{\tau-2\epsilon}}) + 1) = \tilde{O}(\frac{1}{\epsilon\sqrt{\tau-2\epsilon}})$ queries to the oracle in total.

We finish the proof with a quick analysis of the probability of error. Since the HD and CMP subroutines are exact algorithms, the accuracy and the error in these algorithm are solely due to the amplitude estimation and amplification. Now recall that each of amplitude estimation and amplitude amplification are called with $\delta/2$ error. Thus, applying union bound, we get that the probability of error in QBoundFMax is at most δ.

4.1 Fine Tuning of Interval Search

Earlier we saw how to obtain a small interval J that contains $\hat{f}^2_{\max}$ with high probability. However, there may be a requirement to fine-tune this estimation.

Suppose f is linear, i.e., $\eta(f) = 0$. For such a function $\hat{f}^2_{\max} = 1$; however, due to the nature of Algorithm 1 we will get the interval $[1 - \epsilon, 1]$for $\hat{f}^2_{\max}$, and hence an interval for $\eta(f)$. We feel that a non-linearity estimation algorithm should be able to clearly identify a linear function instead of presenting approximate values close to 0.

Consider the other extreme of Bent functions with the largest non-linearity; these would have $\hat{f}^2_{\max} = \frac{1}{2^n}$. However, our Algorithm 1 will, most-likely, return the interval $\approx [\frac{1}{2^n}, \epsilon)$. We wonder if it is possible to obtain an even tighter interval.

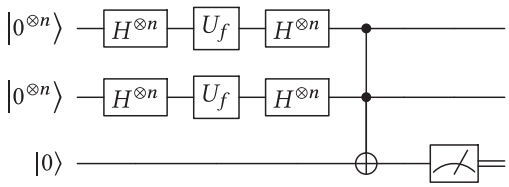

Fig. 3. Circuit for estimating $\sum_x \hat{f}^4(x)$.

For handling these extreme values of $\hat{f}^2_{\max}$ consider the quantum circuit illustrated in Figure 3. Three registers R_1, R_2, and R_3 are initialized to $|0^{\otimes n}\rangle$, $|0^{\otimes n}\rangle$, and $|0\rangle$, respectively. Then, it applies the Deutsch-Jozsa circuit independently on R_1 and R_2, to obtain the state $\sum_x \sum_y \hat{f}(x)\hat{f}(y)\,|x\rangle\,|y\rangle\,|0\rangle$. Finally, the circuit flips the state of R_3 if $x = y$ to obtain the state $\sum_{x,y:x=y} \hat{f}^2(x)\,|x\rangle\,|x\rangle\,|1\rangle + \sum_{x,y:x\neq y} \hat{f}(x)\hat{f}(y)\,|x\rangle\,|y\rangle\,|0\rangle$. Now R_3 is measured in the standard basis. The probability of observing $|1\rangle$ in R_3 is

$$p = \sum_x \hat{f}^4 \leq \hat{f}^2_{max} \sum_x \hat{f}^2 = \hat{f}^2_{max} \quad \text{(using Parseval's equality)}.$$

At this point, amplitude estimation can be used to estimate this probability with any desired additive error ϵ', and with low error probability denoted by δ'. Denoting this estimate by p^*, with probability at least $1 - \delta'$ it satisfies $p^* - \epsilon' \leq p \leq p^* + \epsilon'$, which implies that $\hat{f}^2_{\max} \geq p^* - \epsilon'$. This is used to get a tighter lower bound for $\hat{f}^2_{\max}$. The value of δ' is set to the same δ that is used in IntervalSearch. A tight estimate for $\hat{f}^2_{\max}$ is computed based on the interval J obtained from IntervalSearch.

If J is the rightmost interval, then we want to primarily detect if $\hat{f}^2_{\max} = 1$. We will use $\epsilon' = \epsilon$. Note that amplitude estimation will return 1 in this case without fail. Thus, if $p^* = 1$, then the interval $[1, 1]$ is returned. Otherwise, the left boundary of J can be tightened to $p^* - \epsilon$, provided it is larger than the original left boundary of J. Clearly, we get an interval of length at most ϵ. The number of calls to the circuit, and hence to U_f will be $O(\frac{1}{\epsilon} \log \frac{1}{\delta})$.

If J is the leftmost interval, then we want to primarily get a lower bound on $\hat{f}^2_{\max}$ larger than $\frac{1}{2^n}$. For this reason, we will use $\epsilon' = \epsilon^{3/2}$ and employ the improved amplitude estimation proposed by Montanaro [18]. Observe that the output probability of the circuit is $p = \sum_x \hat{f}^4(x)$; so, its variance can be computed as

$$p(1-p) \leq \hat{f}^2_{\max}\left(1 - \sum_x \hat{f}^4(x)\right) \leq \hat{f}^2_{\max} \leq \epsilon,$$

where the last inequality is implied by the fact that IntervalSearch returned J as the last interval whose right boundary ($\frac{1}{2^k}$) is less than ϵ. Montanaro showed that if variance of an output is bounded, then a better estimation method exists for additive errors; in our case, the number of calls to the circuit, and so also to U_f will be $O(\frac{\sqrt{\epsilon}}{\epsilon^{3/2}} \log \frac{1}{\delta}) = O(\frac{1}{\epsilon} \log \frac{1}{\delta})$. The estimation procedure will return an estimate p^*. If $p^* \leq \epsilon^{3/2}$, then we will can safely use $\frac{1}{2^n}$ as the left boundary of J. Otherwise, we can increase the left boundary of J to $p^* - \epsilon^{3/2}$. In both the cases, the length of J remains at most ϵ.

To summarise, we saw above how to make $O(\frac{1}{\epsilon} \log \frac{1}{\delta})$ queries to U_f and return an estimate for $\hat{f}^2_{\max}$ that is at most ϵ away from the actual value. Moreover, if f is linear, it correctly identifies that $\hat{f}^2_{\max} = 1$.

5 NON-LINEARITY ESTIMATION

Now we combine the earlier results to present our algorithm for estimating $\eta(f)$.

THEOREM 5.1. *Given an oracle U_f to an n-bit function f, an additive accuracy λ and an error δ, there is a quantum algorithm that outputs an interval I of length at most λ such that the normalised nonlinearity of the function f, denoted by $\eta(f)$, belongs to I with probability at least $1-\delta$. The algorithm makes $\tilde{O}(\frac{1}{\lambda^3}\log(\frac{1}{\delta}))$ queries to U_f. If $\eta(f)=0$, then the algorithm always outputs the correct non-linearity.*

The algorithm simply calls `IntervalSearch` (along with fine-tuning described in Section 4.1) using accuracy $\epsilon = 2d$ (for some d to be decided below) and using the error probability $\delta/2$. The combined probability of error, from IntervalSearch and fine-tuning, remains bounded by δ. Let $J = [c-d, c+d)$ be the interval returned by `IntervalSearch`. If $J=[1,1]$, then the algorithm shall output J, else $I = (\frac{1}{2}(1-\sqrt{c}) - \frac{\sqrt{d}}{2}, \frac{1}{2}(1-\sqrt{c}) + \frac{\sqrt{d}}{2}]$. We prove the correctness of this algorithm below.

PROOF. $\eta(f)=0$ is equivalent to $\hat{f}^2_{\max}=1$ and in that case, `IntervalSearch` after fine-tuning returns $[1,1]$ without fail. Thus, this case is correctly handled.

For non-linear functions, suppose `IntervalSearch` returns $J=[c-d,c+d)$. `IntervalSearch` guarantees that $\hat{f}^2_{\max} \in J$. Then, $\hat{f}_{\max} \in J' = [\sqrt{c-d}, \sqrt{c+d})$ and so, $\eta(f) \in I' = (\frac{1}{2}(1-\sqrt{c+d}), \frac{1}{2}(1-\sqrt{c-d})]$. From the fact that $\sqrt{a}-\sqrt{b} \le \sqrt{a-b}$ and $\sqrt{a}+\sqrt{b} \ge \sqrt{a+b}$, we can say that I' is contained in $I = (\frac{1}{2}(1-\sqrt{c}-\sqrt{d}), \frac{1}{2}(1-\sqrt{c}+\sqrt{d})] = (\frac{1}{2}(1-\sqrt{c}) - \frac{\sqrt{d}}{2}, \frac{1}{2}(1-\sqrt{c}) + \frac{\sqrt{d}}{2}]$.

The length of I is $\sqrt{d}$ that we want to be λ. So, we set $d=\lambda^2$. From Theorem 2.1, we know that the complexity of `IntervalSearch` when using `QBoundFMax` is $\tilde{O}(\frac{1}{\epsilon^{3/2}})$. Here, we have $\epsilon = 2d = 2\lambda^2$. Hence, the number of U_f queries in `IntervalSearch` while using `QBoundFMax` is upper bounded by $\tilde{O}(\frac{1}{\lambda^3})$. □

THEOREM 5.2. *There is a classical algorithm that makes $\tilde{O}(\frac{n}{\lambda^6})$ queries to U_f and outputs an interval I of length at most λ such that $\eta(f)$ belongs to I with constant probability.*

The algorithm and the proof are exactly as in Theorem 5.1 but using `CBoundFMax` and without the fine-tuning subroutine.

6 LOWER BOUNDS ON NON-LINEARITY

Now we present our query complexity lower bounds for a non-linearity separation problem. We employ the well-known quantum adversary method proposed by Ambainis [1], which is stated in the form of Theorem 6.1.

THEOREM 6.1. *Let F be a p-bit Boolean function and X and Y be two sets of inputs such that $F(x) \neq F(y)$ for any $x \in X$ and $y \in Y$. Let $R \subseteq X \times Y$ be a relation such that*

(1) for every $x \in X$, $\exists$ at least m different $y \in Y$ such that $(x,y) \in R$.
(2) for every $y \in Y$, $\exists$ at least m' different $x \in X$ such that $(x,y) \in R$.
(3) for every $x \in X$ and $i \in \{1,\ldots,p\}$, $\exists$ at most l different $y \in Y$ such that $x_i \neq y_i$ and $(x,y) \in R$.
(4) for every $y \in Y$ and $i \in \{1,\ldots,p\}$, $\exists$ at most l' different $x \in X$ such that $x_i \neq y_i$ and $(x,y) \in R$.

Then any quantum algorithm uses $\Omega(\sqrt{\frac{m \cdot m'}{l \cdot l'}})$ queries to compute F on $X \bigcup Y$.

Consider the following $\hat{f}_{max}$ decision problem: Given a Boolean function f with a promise that $\hat{f}_{max}=1$ or $\hat{f}_{max}=1-4\lambda$, decide the case of f. Using the notations of Theorem 6.1, consider a

Boolean function $F : \{0,1\}^{2^n} \rightarrow \{0,1\}$ that takes as input a truth-table of an n-bit Boolean function encoded as a 2^n-bit binary string and outputs 1 iff the encoded function, say $f()$, satisfies $\hat{f}_{max} = 1$. The $\hat{f}_{\max}$ decision problem is to compute $F()$ on $X \bigcup Y$ where X contains some n-bit functions with $\hat{f}_{\max} = 1$ and Y contains some n-bit functions with $\hat{f}_{\max} = 1 - 4\lambda$. Note that any algorithm that solves non-linearity estimation with λ accuracy also computes $\hat{f}_{\max}$ with 2λ accuracy and, thus, can solve the above decision problem. So, a lower bound on the $\hat{f}_{max}$ decision problem naturally is a lower bound on the problem of non-linearity estimation.

For the case of $\lambda > \frac{1}{8}$, we have a constant time quantum algorithm for non-linearity estimation and so the lower bound is $\Omega(1)$. So suppose that $\lambda \leq \frac{1}{8}$. Consider the following sets $X = \{0^{2^n}\}$ and $Y = \{y \in \{0,1\}^n : wt(y) = 2\lambda \cdot 2^n\}$ where $wt()$ denotes the Hamming weight. Notice that deciding if an n-bit Boolean function g belongs to X or Y given that g belongs to one of them conforms to the generalized Grover search problem. We present the lower bound of this problem below.

Lemma 6.2. *Any quantum algorithm uses $\Omega(\frac{1}{\sqrt{\lambda}})$ queries to the oracle of the given function g to decide if $g \in X$ or $g \in Y$ given the promise that g belongs to one of them.*

Proof. Consider the sets X and Y. Let the relation R be $R = X \times Y$. Also let $\tilde{\lambda} = 2\lambda \cdot 2^n$. Then for every $x \in X$, $\exists$ exactly $\binom{2^n}{\tilde{\lambda}}$ different $y \in Y$ such that $(x, y) \in R$ and for every $y \in Y$, $\exists$ exactly 1 $x \in X$ such that $(x, y) \in R$. So we have $m = \binom{2^n}{\tilde{\lambda}}$ and $m' = 1$. Now, for any $x \in X$ and $i \in \{1, 2, \ldots, 2^n\}$, fix $y_i = 1$. Naturally, $x_i \neq y_i$. There are exactly $\binom{2^n-1}{\tilde{\lambda}-1}$ different $y \in Y$ such that $y_i = 1$ for any i. Hence, we have $l = \binom{2^n-1}{\tilde{\lambda}-1}$. However, it is trivial that $l' = 1$, since there is only a single element in X. Hence, from Theorem 6.1, we obtain our lower bound as $\Omega(\frac{1}{\sqrt{\lambda}})$ queries. □

Note: Since the problem in Lemma 6.2 is a version of the unstructured search problem, the lower bound in Lemma 6.2 is also a direct consequence of the optimality of the generalized Grover's search [13].

Theorem 6.3. *Any quantum algorithm uses $\Omega(\frac{1}{\sqrt{\lambda}})$ queries to the oracle of the given function f to decide if $\hat{f}_{max} = 1$ or $\hat{f}_{max} = 1 - 4\lambda$ given the promise that it is either of the cases.*

Proof. Consider the set X and Y as defined earlier. We know that for any $a \in \{0,1\}^n$, $\hat{f}(a)$ can be written as $\frac{1}{2^n}(|\{x : f(x) = a \cdot x\}| - |\{x : f(x) \neq a \cdot x\}|)$. Now, since $\lambda \leq \frac{1}{8}$, we have $\hat{f}_y(0^n) = 1 - 4\lambda \geq \frac{1}{2}$ for any $y \in Y$ where y represents the truth table of f_y. Next, as for any $a \neq 0^n$, $f_y(x) = a \cdot x$ for at most $(\frac{1}{2} + 2\lambda)2^n$ many x's. Hence, we have $\hat{f}_y(a) \leq 4\lambda \leq \frac{1}{2}$ for any $y \in Y$. Hence for any $y \in Y$, $(\hat{f}_y)_{max}$ occurs at 0^n and $(\hat{f}_y)_{max} = 1 - 4\lambda$. Now, for $x \in X$, we have $f_x(z) = 0$ for all $z \in \{0,1\}^n$. Hence, $\hat{f}_x(0^n) = 1$ and $\hat{f}_x(a) = 0$ for any $a \neq 0$. So for $x \in X$, $(\hat{f}_x)_{max}$ occurs at 0^n and $(\hat{f}_x)_{max} = 1$.

Thus, the problem of deciding if a given function f belongs to X or Y can be reduced to deciding if for the function f, $\hat{f}_{max} = 1$ or $\hat{f}_{max} = 1 - 4\lambda$. So, using Lemma 6.2, we obtain that any quantum algorithm uses $\Omega(\frac{1}{\sqrt{\lambda}})$ queries to the oracle of the given function f to decide if $\hat{f}_{max} = 1$ or $\hat{f}_{max} = 1 - 4\lambda$ given the promise that it is either of the cases. □

Note that the lower bound in Theorem 6.3 indicates the difficulty in separating functions that are linear and 4λ-close to linear. We also obtain the same lower bound separating functions that are bent and are 4λ-close to bent. Recall that a function f is called a bent function if $|\hat{f}(a)| = \frac{1}{\sqrt{2^n}}$ for all $a \in \{0,1\}^n$. Therefore, for such a function, if $\hat{f}(0^n) = \frac{1}{\sqrt{2^n}}$ then $wt(x_f) = \frac{1}{2}(2^n - \sqrt{2^n})$ where x_f represents the truth table of f.

Characterising functions that are 4λ-close to bent requires a little bit of work, and the following proposition does the heavy-lifting.

Proposition 6.1. *Let f be some n-bit function such that $wt(x_f) \geq k$ and a Boolean function g be obtained from f by flipping k bits in x_f from 1 to 0. Then, $\hat{g}(0^n) = \hat{f}(0^n) + k\frac{2}{2^n}$, and for any $a \neq 0^n$, $\hat{g}(a) = \hat{f}(a) + \sum_{i=1}^{k} \pm\frac{2}{2^n}$.*

Proof. We will actually prove the proposition for $k = 1$, i.e., g is obtained from f by flipping the value at some point, say, y; so $f(y) = 1$ was flipped to $g(y) = 0$. The general case can be applied by repeatedly applying this case, in turn, to the g that is obtained.

For any $a \in \{0,1\}^n$, define $S_a^0(f)$ as the number of elements in the set $\{x : f(x) = a \cdot x\}$ and $S_a^1(f)$ as the number of elements in the set $\{x : f(x) \neq a \cdot x\}$. When $a = 0^n$, $S_a^1(f) = wt(f)$ and $S_a^0(f) = 2^n - wt(f)$. In general, $\hat{f}(a) = \frac{1}{2^n}(S_a^0(f) - S_a^1(f))$.

Now, g is obtained by flipping the value of $f(y) = 1$ to $g(y) = 0$, i.e., $wt(x_g) = wt(x_f) - 1$. First, consider the case of $a = 0^n$. For this a, $S_a^1(g) = S_a^1(f) - 1$, $S_a^0(g) = S_a^1(f) + 1$, and $\hat{g}(0^n) = \hat{f}(0^n) + \frac{2}{2^n}$. This proves the first claim of the proposition.

Now we prove the case of general $a \neq 0^n$. There are two possibilities here. On the one hand, if a is such that $f(y) = a \cdot y$, then $S_a^0(g) = S_a^0(f) - 1$ and $S_a^1(g) = S_a^1(f) + 1$, and thus, $\hat{g}(a) = \hat{f}(a) - \frac{2}{2^n}$. On the other hand, if a is such that $f(y) = 1 \oplus a \cdot y$, then a similar argument shows that $\hat{g}(a) = \hat{f}(a) + \frac{2}{2^n}$. Combining both the cases we get that $\hat{g}(a) = \hat{f}(a) \pm \frac{2}{2^n}$ as required. □

If we choose bent functions such that $\hat{f}(0^n) = \frac{1}{\sqrt{2^n}}$, then we always have $\hat{f}(0^n) \geq \hat{f}(a)$ for any a. Moreover, if g is obtained by flipping some bits of x_f from 1 to 0, then from the above proposition we get that $\hat{g}(0^n) = \hat{f}(0^n) + \frac{2k}{2^n} \geq \hat{f}(0^n) + \sum_{i=1}^{k} \pm\frac{2}{2^n} \geq \hat{f}(a) + \sum_{i=1}^{k} \pm\frac{2}{2^n} = \hat{g}(a)$ for any a. That is, $\hat{g}_{max} = \hat{g}(0^n)$. Clearly, if exactly $2\lambda \cdot 2^n$ 1s in x_f are flipped to 0, then $\hat{g}_{max}$ occurs at 0^n and $\hat{g}(0^n) = \frac{1}{\sqrt{2^n}} + 4\lambda$.

Theorem 6.4. *Any quantum algorithm uses $\Omega(\frac{1}{\sqrt{\lambda}})$ queries to the oracle of the given function f to decide if $\hat{f}_{max} = \frac{1}{\sqrt{2^n}}$ or $\hat{f}_{max} = \frac{1}{\sqrt{2^n}} + 4\lambda$ given the promise that it is either of the cases.*

Proof. Let x be the truth table string of some bent function f_x such that $\hat{f}_x(0^n) = \frac{1}{\sqrt{2^n}}$. Then we know $wt(x) = \frac{1}{2}(2^n - \sqrt{2^n}) = k$. Let $X = \{x\}$ and $\tilde{\lambda} = 2 \cdot \lambda \cdot 2^n$. Now let Y be the set that contains all strings y, which can be obtained by flipping exactly $\tilde{\lambda}$ number of 1s to 0. Then for any $y \in Y$, $wt(y) = \frac{1}{2}(2^n - \sqrt{2^n}) - \tilde{\lambda}$. Let $R = X \times Y$. Now, for $x \in X$, we can see that there are exactly $\binom{k}{\tilde{\lambda}}$ number of $y \in Y$ such that $(x, y) \in R$ and so $m = \binom{k}{\tilde{\lambda}}$. Similarly, we can see that for any $i \in \{1, \ldots, 2^n\}$ there are at most $l = \binom{k-1}{\tilde{\lambda}-1}$ $y \in Y$ such that $x_i \neq y_i$ and $(x, y) \in R$. Since the set X contains only a single element, we have $m' = 1$ and $l' = 1$. So we have $\frac{m \cdot m'}{l \cdot l'} = \frac{k}{\tilde{\lambda}} = \frac{2^n - \sqrt{2^n}}{2\tilde{\lambda}} \geq \frac{2^n/2}{2\tilde{\lambda}} = \frac{1}{8\lambda}$. Thus we have the lower bound as $\Omega(\frac{1}{\sqrt{\lambda}})$. □

These results lead to our required quantum lower bound.

Theorem 6.5. *Any quantum algorithm uses $\Omega(\frac{1}{\sqrt{\lambda}})$ queries to estimate the non-linearity of a given function with λ accuracy.*

Next we show the classical randomised complexity of the non-linearity estimation. The adversary method has been extended to randomised algorithms as well, which is what we use. Using a

theorem given by Laplante et al. [16] in which we use the values obtained in the proof of Theorem 6.5, we obtain the following result. (Complete proof is given in Appendix B.)

Theorem 6.6. *Any classical randomised algorithm uses $\Omega(\frac{1}{\lambda})$ queries to estimate the non-linearity of a given function with λ accuracy.*

7 CONCLUSION

In this article, we looked at the problem of estimating, within additive accuracy λ, the non-linearity of a Boolean function given to us as a black-box. We devised a quantum strategy that makes $\tilde{O}(\frac{1}{\lambda^3})$ queries to the black-box in the worst-case and is independent of the size of the function. In contrast, our classical randomised algorithm makes linear in n many queries apart from depending on λ as $\frac{1}{\lambda^6}$. We proved a lower bound of $\Omega(\frac{1}{\sqrt{\lambda}})$ on the quantum query complexity and a lower bound of $\Omega(\frac{1}{\lambda})$ on the classical randomised query complexity of the non-linearity estimation problem. We conclude with a conjecture that the quantum query complexity of non-linearity estimation is $\Theta(\frac{1}{\sqrt{\lambda}})$—that will allow us to construct a sub-exponential query quantum algorithm for exactly computing $\eta(f)$ using only a few additional qubits.

APPENDICES

A AMPLITUDE ESTIMATION

This section is about estimation of the probability p of an observation upon measuring the final state of a quantum circuit A. Let k and m be some parameters that we shall fix later. A quantum amplitude estimation algorithm (say, named as *AmpEst*) was proposed by Brassard et al. [7] that acts on two registers of m and n qubits, makes 2^m calls to controlled-A and outputs a $\tilde{p} \in [0, 1]$ that is a good approximation of p in the following sense.

Theorem A.1. *The AmpEst algorithm returns an estimate $\tilde{p}$ that has a confidence interval $|p - \tilde{p}| \leq 2\pi k \frac{\sqrt{p(1-p)}}{2^m} + \pi^2 \frac{k^2}{2^{2m}}$ with probability at least $\frac{8}{\pi^2}$ if $k = 1$ and with probability at least $1 - \frac{1}{2(k-1)}$ if $k \geq 2$. If $p = 0$ or 1, then $\tilde{p} = p$ with certainty.*

The *AmpEst* algorithm can be used to estimate p with desired accuracy and error.

Corollary A.2. *Let AmpEst quantum algorithm be run $7(\ln \frac{1}{\delta})^{1/3} = O(\ln \frac{1}{\delta})$ times using $k = 1$ and $2^m \geq \frac{3\pi}{2\epsilon}$ and the median of the obtained estimates of p is returned as $\tilde{p}$. Then we obtain an estimate $\tilde{p}$ such that $\Pr[|p - \tilde{p}| \geq \epsilon] \leq \delta$ for any $\epsilon \leq \frac{1}{4}$ using a total of $\Theta(\frac{1}{\epsilon} \log \frac{1}{\delta})$ calls to A.*

The proof of the corollary is as follows:

Proof. Let M denote 2^m. If we require that $2\pi \frac{\sqrt{p(1-p)}}{M} + \frac{\pi^2}{M^2} \leq \epsilon$, then it suffices to take $M \geq \frac{\pi}{\epsilon}[\sqrt{p(1-p)} + \sqrt{p(1-p) + \epsilon}]$. Since $1/4 \geq p(1-p)$, $\frac{1}{2}e^{2\epsilon} \geq \sqrt{\frac{1}{4} + \epsilon}$ ($\because \epsilon \leq 1/4$) and $3 \geq 1 + e^{2\epsilon}$, it suffices to take $M \geq \frac{3\pi}{2\epsilon}$.

Therefore, for $k = 1$ and using the above value of M we obtain that each individual estimate p_i satisfies $\Pr[|p - p_i| \leq \epsilon] \geq \frac{8}{\pi^2}$. This success probability can be increased to $1 - \delta$ for any arbitrary δ if we use median of $\Theta(\log \frac{1}{\delta})$ values by standard Chernoff bounds. □

B LOWER BOUND FOR RANDOMISED NON-LINEARITY TESTING

Consider the following theorem from Reference [16].

Theorem B.1 (Theorem 2 [16]). *Let S and S' be two sets and let $f : S \to S'$ be a function. Consider a weight scheme where*

(1) *Every pair* $(x, y) \in S \times S$ *is assigned a non-negative weight* $w(x, y)$ *such that* $w(x, y) = 0$ *whenever* $f(x) = f(y)$.
(2) *Every tuple* (x, y, i) *is assigned a non-negative weight* $w'(x, y, i)$ *such that* $w'(x, y, i) = 0$ *whenever* $f(x) = f(y)$ *or* $x_i = y_i$.

Define $wt(x) = \Sigma_y w(x, y)$ *and* $v(x, i) = \Sigma_y w'(x, y, i)$ *for all* x *and for all* i. *If* $w'(x, y, i)w'(y, x, i) \geq w^2(x, y)$ *for all* x, y, i *such that* $x_i \neq y_i$, *then*

$$Q_2(f) = \Omega\left(\min_{\substack{x,y,i \\ w(x,y)\neq 0, x_i \neq y_i}} \sqrt{\frac{wt(x)wt(y)}{v(x,i)v(y,i)}}\right).$$

Moreover, if $w'(x, y, i)w'(y, x, i) \geq w(x, y)$ *for all* x, y, i *such that* $x_i \neq y_i$, *then*

$$R(f) = \Omega\left(\min_{\substack{x,y,i \\ w(x,y)\neq 0, x_i \neq y_i}} \max\left\{\frac{wt(x)}{v(x,i)}, \frac{wt(y)}{v(y,i)}\right\}\right),$$

where $Q_2(f)$ *is the bounded error quantum query complexity and* $R(f)$ *is the classical randomised query complexity of* f.

Define a weight scheme such that

$$w(x, y) = \begin{cases} 0, \text{if } f(x) = f(y) \\ 1, \text{else} \end{cases}; \quad w'(x, y, i) = \begin{cases} 0, \text{if } f(x) = f(y) \text{ or } x_i = y_i \\ 1, \text{else} \end{cases}.$$

Note that in this weight scheme it is always true that $w'(x, y, i) \cdot w'(y, x, i) \geq w^2(x, y)$ and $w'(x, y, i) \cdot w'(y, x, i) \geq w(x, y)$ for all x, y, i such that $x_i \neq y_i$. Let X be a non-empty subset of the set of strings x such that $f(x) = 0$ and Y be a non-empty subset of the set of strings y such that $f(y) = 1$ where f is some decision function that takes inputs of size n. Let $[n] = \{1, \ldots, n\}$. Also let $S = X \cup Y$ and $R = X \times Y$. Define m_x (respectively, m_y) for any $x \in X$ (respectively, $y \in Y$) as the number of $y \in Y$(respectively, $x \in X$) such that $(x, y) \in R$. Similarly, for any $x \in X$ (respectively, $y \in Y$) and $i \in [n]$ define $l_{(x,i)}$ (respectively, $l_{(y,i)}$) as the number of $y \in Y$(respectively, $x \in X$) such that $(x, y) \in R$ and $x_i \neq y_i$. From the definition of the weights from Theorem B.1, we can see that $wt(x) = m_x$, $wt(y) = m_y$, $v(x, i) = l_{(x.i)}$ and $v(y, i) = l_{(y,i)}$ for any $x \in X$, $y \in Y$ and $i \in [n]$. Adapting the definition from Theorem 6.1, let $m = \min_x\{m_x\}$, $m' = \min_y\{m_y\}$, $l = \max_{x,i}\{l_{(x,i)}\}$ and $l' = \max_{y,i}\{l_{(y,i)}\}$. Using these definitions, we can explicitly see the equivalence of the forms of $Q_2(f)$ in Theorem 6.1 and Theorem B.1 as

$$Q_2(f) = \Omega\left(\min_{\substack{x,y,i \\ w(x,y)\neq 0, x_i \neq y_i}} \sqrt{\frac{wt(x)wt(y)}{v(x,i)v(y,i)}}\right) = \Omega\left(\sqrt{\frac{m \cdot m'}{l \cdot l'}}\right).$$

Since X and Y are non-empty and $R = X \times Y$, we know for any y, $m_y \geq 1$. Next, since for any $x \in X$ and $y \in Y$, we have $f(x) \neq f(y)$, it is obvious that $x \neq y$. So for any $y \in Y$ there is at least one index $i \in [n]$ such that $x_i \neq y_i$ and hence $l_{(y,i)} \geq 1$. Now, see in the proofs of Theorem 6.3 and Theorem 6.4 that $m' = l' = 1$. Hence we have $wt(y) = m_y = 1$ and $v(y, i) = l_{(y,i)} = 1$ for any $y \in Y$ and $i \in [n]$. From the proofs of Theorem 6.3 and Theorem 6.4 we also have $\frac{m}{l} = \min_{x,i}(\frac{m_x}{l_{(x,i)}}) \geq 1$.

Hence $\frac{m_x}{l_{(x,i)}} \geq 1$ for any $x \in X$ and $i \in [n]$ and so $\max\{\frac{wt(x)}{v(x,i)}, \frac{wt(y)}{v(y,i)}\} = \frac{wt(x)}{v(x,i)}$. This implies that

$$R(f) = \Omega\left(\min_{\substack{x,y,i \\ w(x,y)\neq 0, x_i \neq y_i}} \frac{wt(x)}{v(x,i)}\right) = \Omega\left(\frac{m}{l}\right),$$

leading us the following theorem.

Theorem B.2. *Any classical randomised algorithm uses $\Omega(\frac{1}{\lambda})$ queries to estimate the non-linearity of a given function with λ accuracy.*

C ALTERNATIVE QUANTUM APPROACHES FOR BOUNDFMAX

Recall that there are three components that contribute to the complexity of CBoundFMax. The innermost component is the classical estimation with complexity $\tilde{O}(\frac{1}{\epsilon^2}\log\frac{1}{\delta})$. This is nested inside the loop over all suffixes of any particular length that contributes $O(\frac{1}{\tau-2\epsilon})$. This in turn is nested inside an outer loop over all levels of the binary tree that gives a $O(n)$ to the complexity. Improving the speed of any of the three components will result in an overall speedup. In the algorithms that follow, we tried to replace the existing classical component with a quantum component that has a speedup over its classical counterpart.

C.1 CBoundFMax-QPWC

The most naive way to introduce "quantumness" in CBoundFMax is by replacing the classical estimation of PWC in Algorithm 2 by a quantum estimation of PWC. The new algorithm that we call CBoundFmax-QPWC is given as Algorithm 4.

ALGORITHM 4: Algorithm CBoundFMax-QPWC

Require: threshold $\tau \in (0,1)$, confidence$\epsilon \in (0,\tau)$, error $\delta \in (0,1)$
  Initialise a FIFO list $Q = \{\varepsilon\}$, where ε denotes the empty string
  while Q is not empty **do**
    remove prefix $\mathfrak{p}$ from Q
    for suffix $\mathfrak{s}$ from $\{0,1\}$ **do**
      childprefix $\mathfrak{cp} = \mathfrak{p}^\frown\mathfrak{s}$
      Estimate $e = \sum_{x \text{ with prefix } \mathfrak{cp}} \hat{f}(x)^2$ with accuracy ϵ and error $(\frac{\tau-2\epsilon}{2n})\delta$. ▷ Lemma 4.1
      if $e \geq \tau - \epsilon$ **then**
        If $len(\mathfrak{cp}) < n$, add $\mathfrak{cp}$to Q
        Else (i.e., $len(\mathfrak{cp}) = n$), **return** TRUE
      end if
    end for
  end while
  return FALSE

The correctness of the algorithm clearly follows from the correctness of Algorithm 2. Now see that the complexity of the quantum estimation can be obtained as $O(\frac{1}{\epsilon}\log(\frac{1}{\delta}))$ from Theorem 4.1. The complexity of the classical outer loop is $O(\frac{n}{\tau-2\epsilon})$ as derived in Section 3. Hence, the query complexity of CBoundFMax-QPWC turns out as $O(\frac{n}{\epsilon(\tau-2\epsilon)}\log(\frac{1}{\delta}))$ queries, a slight improvement over the CBoundFMax.

Using CBoundFMax-QPWC to solve the BoundFMax problem in Algorithm 1, we obtain a hybrid algorithm of query complexity $O(\frac{n}{\epsilon^2}\log\frac{1}{\delta})$ that outputs an interval of length at most ϵ that contains $\hat{f}^2_{max}$ with probability at least $1-\delta$, given ϵ and δ.

C.2 QCBoundFMax

Now we describe another optimisation that changes one more classical operation to a quantum one. Recall that in the level order traversal of the binary tree, at any particular level, there can be at most $\frac{1}{\tau-2\epsilon}$ many prefixes such that the estimate of the PWC at those points are greater than the threshold. Since the estimation of PWC at the next level is performed over all points with these prefixes, the number of estimations due to these prefixes is of the order $O(\frac{1}{\tau-2\epsilon})$. In this version of the algorithm, which we call QCBoundFMax, we replace the classical level order traversal with a quantum level order traversal and use superposition and amplitude amplification to our benefit.

Our quantum algorithm QCBoundFMax is described in Algorithm 5. We use Ri to represent the ith register used in the quantum circuit corresponding to the algorithm. In the algorithm, we also use a few smaller circuits as described in Section 4.

ALGORITHM 5: Algorithm QCBoundFMax

Require: Threshold τ, accuracy ϵ and error δ.

1: Set $\tau' = \tau - \epsilon$ and $q = \lceil \log\left(\frac{1}{\epsilon}\right)\rceil + 3$
2: Set $\tau_1 = \left\lfloor \frac{2^q}{\pi}\sin^{-1}(\sqrt{\tau'})\right\rfloor$ if $\sin^{-1}(\sqrt{\tau'}) \leq \frac{\pi}{2}$ and $\tau_1 = \lceil \frac{2^q}{\pi}\sin^{-1}(\sqrt{\tau'})\rceil$ if $\sin^{-1}(\sqrt{\tau'}) > \frac{\pi}{2}$.
3: Initialise the circuit as $|0^n\rangle|0^n\rangle|0^q\rangle|\tau_1\rangle|0^n\rangle$.
4: Apply $H^{\otimes n}$ on $R1$.
5: **for** i in $\{1,\ldots,n\}$ **do**
6: Apply quantum amplitude estimation (*AmpEst*) on DJ with i qubits of $R2$ as the input register, $R3$ as the precision register and $R1$ is used to determine the "good state". *AmpEst* is called with error at most $\delta/2n$ and additive accuracy $\frac{1}{2}\cdot\frac{1}{2^q}$.
7: Use $\mathsf{HD_q}$ on $R3$ and $R4$ separately.
8: Use CMP on $R3$ and $R4$ as input registers and $R5$ as output register.
9: Apply Fixed Point Amplitude Amplification (FPAA) $\frac{1}{\sqrt{\tau-2\epsilon}}$ times on the ith qubit of $R5$ with error at most $\delta/2n$ and measure ith qubit of $R5$ as m_i.
10: **if** $m_i = |0\rangle$ **then**
11: **return** FALSE
12: **else if** $m_i = |1\rangle$ and $i = n$ **then**
13: **return** TRUE
14: **end if**
15: Reset $R3$ to $|0^q\rangle$ and $R4$ to $|\tau_1\rangle$ by applying $HD_q^\dagger$ individually on them.
16: **end for**

One can see that in QCBoundFMax the selection of desired prefixes (i.e., any prefix p such that $PWC(p) \geq \tau$) for the next round is done quite differently from what is used in CBoundFMax-QPWC. While in CBoundFMax-QPWC the desired prefixes for the round i are selected by estimating PWC of each of the prefixes selected at round $i-1$ one at a time and then comparing them with the threshold, in QCBoundFMax we use a clever combination of amplitude estimation without measurement, amplitude amplification and two constant time subroutines HD_q and CMP. The estimation and comparison using *AmpEst*, HD_q and CMP is exactly as explained in Section 4 except that here the estimate is of the amplitude of the prefixes p that were selected in round i. Also of difference is that the EQ circuit in the amplitude estimation module here works such that in the ith round,

EQ maps the $2n$-qubit basis states $|x, y\rangle$ to $(-1)|x, y\rangle$ if $x_j = y_j$ for all $j \in \{0, 1, \ldots, i-1\}$ where x_j denotes the jth bit of the string x and similarly for y_j. Post estimation and comparison, the ith output bit of any i-bit prefix p contains $|1\rangle$ if $PWC(p)$ is greater than the threshold and $|0\rangle$ otherwise. We next use amplitude amplification to amplify only those states whose ith output qubit is $|1\rangle$. Since at any level there can be at most $\frac{1}{\tau-2\epsilon}$ many prefixes that are greater than the threshold, we use FPAA $O(\frac{1}{\sqrt{\tau-2\epsilon}})$ many times to amplify the desired states. This process is repeated in loop over all possible lengths of prefixes. Hence the total query complexity of QCBoundFMax is $O(n) \times O(\frac{1}{\sqrt{\tau-2\epsilon}}) \times O(1) \times O(\frac{1}{\epsilon}) = O(\frac{n}{\epsilon\sqrt{(\tau-2\epsilon)}})$ queries. Notice that this is an $O(\frac{1}{\sqrt{\tau-2\epsilon}})$ speedup over CBoundFMax-QPWC.

It easily follows that the use of QCBoundFMax to solve the BoundFMax problem in Algorithm 1 gives us a quantum algorithm that given additive accuracy ϵ and error probability δ outputs an interval of length at most ϵ that contains $\hat{f}^2_{max}$ with probability $1-\delta$ using $O(\frac{n}{\epsilon^{3/2}} \log \frac{1}{\delta})$ queries to the oracle of the function.

Finally, observe that by cleverly setting the initial state and ignoring the first $n-1$ rounds in QCBoundFMax algorithm we obtain Algorithm 3 with a better query complexity.

D HD_q AND CMP SUBROUTINES

In this section we present the algorithms of the HD_q and CMP subroutines used in Algorithm 3. The string representations of the integers are indexed in the reverse order, i.e., the bit-representation of an integer z is represented as $z = z_0 z_1 z_2 \ldots$ where z_0 denotes the most significant bit of z. We have used the following quantum operations in those algorithms.

- CX(a,b): Controlled-NOT gate where X gate is applied on qubit b if the control qubit a is in the state $|1\rangle$.
- $\overline{\text{CX}}$(a, b): Controlled-NOT gate where X gate is applied on qubit b if the control qubit a is in the state $|0\rangle$.
- $\text{C}\overline{\text{C}}^{\text{i}}\text{X}(\text{a}, \text{b}_1, \text{b}_2, \ldots, \text{b}_\text{i}, \text{c})$: Multi-controlled NOT gate where X gate is applied on qubit c if the first control qubit a is in the state $|1\rangle$ and the next i control qubits $\text{b}_1, \text{b}_2, \cdots, \text{b}_\text{i}$ are in the state $|0\rangle$.
- SWAP(A,B): Swap gate that swaps register A with register B given that the registers A and B are of the same size.

D.1 Quantum Circuit for HD_q Operation

The HD_q operation was defined as $|y\rangle|b\rangle \mapsto |b \oplus \tilde{y}\rangle|y\rangle$ where $y, b \in \{0,1\}^q$ and $\tilde{y}$ is the bit string corresponding to the integer $|2^{q-1} - y_{int}|$. An algorithm to implement this operator is described in Algorithm 6. We also show the quantum circuit corresponding to HD_4 in Figure 4. Note that the register a is an ancilla register.

The algorithm behind HD_q is based on bitwise manipulations. We use y to denote the input integer ($0 \le y < 2^q$), available in the first register. We will focus on obtaining $\left||y - 2^{q-1}|\right\rangle$ in a register. The simple case is $y \ge 2^{q-1}$ when we need to simply set the most significant bit of y to 0.[3] For the other case, i.e., $y < 2^{q-1}$, we have to subtract y (available in binary) from 2^{q-1} (1 followed by $q-1$ zeroes). The algorithm implements this by first finding out the rightmost bit of y that is set to 1, denoted t; all bits to the right of t are 0 and do not contribute to the difference. The tth bit of the difference is set to 1. And all the bits to the left of t are flipped (they are subtracted from 1

[3]For instance if $y = 111$, then $\tilde{y} = 100 - 111 = 011$.

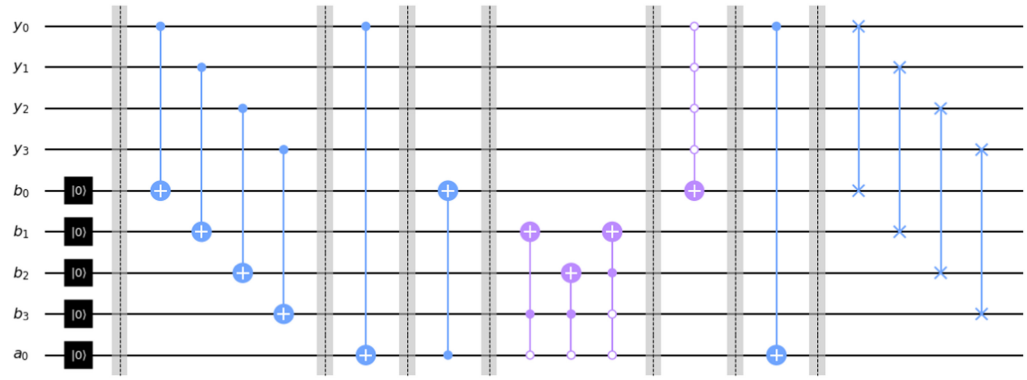

Fig. 4. The $\mathtt{HD_q}$ circuit for a 4 qubit input y (Created using Qiskit).

ALGORITHM 6: Subroutine HD_q

Initialise $\mathtt{R_1R_2R_3} = |y\rangle\,|0^n\rangle\,|0\rangle$.
for i in $\{0, 1, 2, \ldots, n-1\}$ **do**
    Apply $\mathtt{CX}(\mathtt{R_1[i]}, \mathtt{R_2[i]})$.
end for
Apply $\mathtt{CX}(\mathtt{R_1[0]}, \mathtt{R_3})$.
Apply $\mathtt{CX}(\mathtt{R_3}, \mathtt{R_2[0]})$.
Initialise an empty array S.
for i in $\{0, 1, 2, \ldots, n-1\}$ **do**
    Append $R_2[n-i-1]$ to S.
    for k in $\{1, n-i-2\}$ **do**
        Apply $\overline{\mathtt{C}}^{\mathtt{i+1}}\mathtt{CX}(\mathtt{R_3}, \mathtt{S[0]}, \mathtt{S[1]}, \ldots, \mathtt{S[i]}, \mathtt{R_2[k]})$.
    end for
end for
Apply $\overline{\mathtt{C}}^{\mathtt{n}}\mathtt{X}(\mathtt{R_1[0]}, \mathtt{R_1[1]}, \ldots, \mathtt{R_1[n-1]}, \mathtt{R_2[0]})$.
Apply $\mathtt{CX}(\mathtt{R_1[0]}, \mathtt{R_3})$.
Apply $\mathtt{SWAP}(\mathtt{R_1}, \mathtt{R_2})$.

created by the cascading borrowing effect when the t-bit of the difference was calculated).[4] And of course, if all the bits of y are 0, then we just flip the first bit of the answer.

D.2 Quantum Circuit for *CMP* Operation

Next, we present the algorithm for the subroutine CMP in Algorithm 7. Figure 5 contains the circuit for the CMP subroutine with x as the first input and y as the second input. It is to be noted that the register a in the circuit is an ancilla register. The state of the register a before and after the subroutine is applied is $|0^n\rangle$. The *out* register contains the result of the comparison in the form $|x \leq y\rangle$.

The algorithm applies the standard method of comparing two bit-strings. It compares the kth bit of x and y, starting from the most significant bit and moving right, until it finds some k such

[4]For instance, if we have $y = 0110$, then we need $10000 - 00110$. The first bit from the right that is set to one is the second least significant bit. Then all bits between the second least significant bit and the most significant bit is flipped. So we have $10000 - 00110 = 01010$.

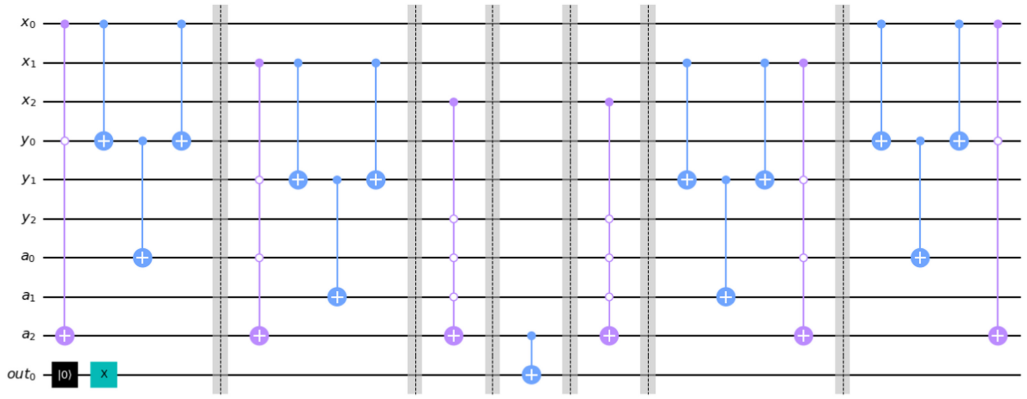

Fig. 5. The CMP circuit of the inputs y and t of size 3 (Created using Qiskit).

ALGORITHM 7: Subroutine CMP

1: Initialise $\mathsf{R}_1\mathsf{R}_2\mathsf{R}_3\mathsf{R}_4 = |x\rangle\,|y\rangle\,|0^n\rangle\,|0\rangle$.
2: Apply $\mathsf{X}(\mathsf{R}_4)$.
3: Initialise an empty array A.
4: **for** i in $\{0, 1, \ldots, n-1\}$ **do**
5: Apply $\overline{\mathsf{CC}}^{\mathsf{i+1}}\mathsf{X}(\mathsf{R}_1[\mathsf{i}], \mathsf{R}_2[\mathsf{i}], \mathsf{A}[\mathsf{0}], \mathsf{A}[\mathsf{1}], \ldots, \mathsf{A}[\mathsf{i}-\mathsf{1}], \mathsf{R}_3[\mathsf{n}-\mathsf{1}])$.
6: **if** $i \neq n-1$ **then**
7: Apply $\mathsf{CX}(\mathsf{R}_1[\mathsf{i}], \mathsf{R}_2[\mathsf{i}])$.
8: Apply $\mathsf{CX}(\mathsf{R}_2[\mathsf{i}], \mathsf{R}_3[\mathsf{i}])$.
9: Apply $\mathsf{CX}(\mathsf{R}_1[\mathsf{i}], \mathsf{R}_2[\mathsf{i}])$.
10: **end if**
11: Append $\mathsf{R}_3[\mathsf{i}]$ to array A.
12: **end for**
13: Apply $\mathsf{CX}(\mathsf{R}_3[\mathsf{n}-\mathsf{1}], \mathsf{R}_4)$.
14: Apply the conjugate transpose of all the operations from line 4 to line 12.

that $x_k \neq y_k$. For this position, the answer bit is flipped (i.e., we claim that $x > y$) if $x_k = 1, y_k = 0$; otherwise we claim that $x \leq y$. We also start the answer qubit in $|1\rangle$, so if $x \leq y$, then the answer qubit remains in the state $|1\rangle$.

E PROOF OF PROPOSITION 4.3

Proposition 4.3. *τ and τ_1 satisfy* $0 \leq \tau - \epsilon - \sin^2(\pi\frac{\tau_1}{2^q}) \leq \frac{2\pi}{2^q}$.

Proof. We defined $\tau' = \tau - \epsilon$ and $\tau_1 = \lfloor \frac{2^q}{\pi} \cdot \sin^{-1}(\sqrt{\tau'}) \rfloor$. Then,

$$\begin{aligned}
&\tau_1 \leq \frac{2^q}{\pi} \cdot \sin^{-1}(\sqrt{\tau'}) \\
&\Rightarrow \sin\left(\frac{\pi}{2^q}\tau_1\right) \leq \sqrt{\tau'} \\
&\Rightarrow \sin^2\left(\frac{\pi}{2^q}\tau_1\right) = p_{\tau_1} \leq \tau' \\
&\Rightarrow \tau' - p_{\tau_1} \geq 0.
\end{aligned}$$

However, we have

$$\begin{aligned}
\tau_1 &> \lfloor (2^q/\pi)\cdot \sin^{-1}(\sqrt{\tau'})\rfloor - 1\\
\Rightarrow \sin\left(\frac{\pi}{2^q}\tau_1\right) &> \sin\left(\sin^{-1}(\sqrt{\tau'}) - \frac{\pi}{2^q}\right)\\
&= \sqrt{\tau'}\cos\left(\frac{\pi}{2^q}\right) - \cos\left(\sin^{-1}(\sqrt{\tau'})\right)\sin\left(\frac{\pi}{2^q}\right)\\
&= \sqrt{\tau'}\cos\left(\frac{\pi}{2^q}\right) - (\sqrt{1-\tau'})\sin\left(\frac{\pi}{2^q}\right) \text{ (Uses } \cos\left(\sin^{-1}(x)\right) = 1-x^2)\\
\Rightarrow p_{\tau_1} &= \sin^2(\frac{\pi\tau_1}{2^q})\\
&\geq \tau'\cos^2\left(\frac{\pi}{2^q}\right) + (1-\tau')\sin^2\left(\frac{\pi}{2^q}\right) - 2\sqrt{\tau'}\sqrt{1-\tau'}\cos\left(\frac{\pi}{2^q}\right)\sin\left(\frac{\pi}{2^q}\right)\\
&\geq \tau'\cos^2\left(\frac{\pi}{2^q}\right) - (1-\tau')\sin^2\left(\frac{\pi}{2^q}\right) - 2\sqrt{\tau'}\sqrt{1-\tau'}\cos\left(\frac{\pi}{2^q}\right)\sin\left(\frac{\pi}{2^q}\right)\\
&= \tau' - \sin^2\left(\frac{\pi}{2^q}\right) - \sqrt{\tau'}\sqrt{1-\tau'}\sin\left(\frac{2\pi}{2^q}\right).
\end{aligned}$$

Now, notice that

$$\begin{aligned}
&\sin^2\left(\frac{\pi}{2^q}\right) + \sqrt{\tau'}\sqrt{1-\tau'}\sin\left(\frac{2\pi}{2^q}\right)\\
&\leq \sin^2\left(\frac{\pi}{2^q}\right) + \frac{1}{2}\sin\left(\frac{2\pi}{2^q}\right) \text{ (Uses } x(1-x)\leq 1/4)\\
&= \sin^2\left(\frac{\pi}{2^q}\right) + \sin\left(\frac{\pi}{2^q}\right)\cos\left(\frac{\pi}{2^q}\right)\\
&= \sin\left(\frac{\pi}{2^q}\right)\left(\sin\left(\frac{\pi}{2^q}\right) + \cos\left(\frac{\pi}{2^q}\right)\right)\\
&\leq \frac{2\pi}{2^q}.
\end{aligned}$$

The last inequality is due to the fact that $\sin\theta \leq \theta$, $\sin(\theta)\leq 1$ and $\cos(\theta)\leq 1$. This gives us the following:

$$\begin{aligned}
p_{\tau_1} &\geq \tau' - \sin^2\left(\frac{\pi}{2^q}\right) - \sqrt{\tau'}\sqrt{1-\tau'}\sin\left(\frac{2\pi}{2^q}\right)\\
&> \tau' - \frac{2\pi}{2^q}.
\end{aligned}$$

So we have

$$0 \leq \tau' - p_{\tau_1} \leq \frac{2\pi}{2^q}.$$ □

REFERENCES

[1] Andris Ambainis. 2002. Quantum lower bounds by quantum arguments. *J. Comput. Syst. Sci.* 64, 4 (2002), 750–767.

[2] Debajyoti Bera, Subhamoy Maitra, Dibyendu Roy, and Pantelimon Stanica. 2019. Limitations of the BLR testing in estimating nonlinearity. In *Proceedings of the 11th International Workshop on Coding and Cryptography (WCC'19).*

[3] Debajyoti Bera, Subhamoy Maitra, and Sapv Tharrmashastha. 2019. Efficient quantum algorithms related to autocorrelation spectrum. In *Proceedings of the International Conference on Cryptology in India.* Springer, 415–432.

[4] Debajyoti Bera and SAPV Tharrmashastha. 2021. Quantum Algorithms for Entropy Testing and Gapped k-Distinctness. arXiv:2103.09033. Retrieved from https://arxiv.org/abs/2103.09033.

[5] Manuel Blum, Michael Luby, and Ronitt Rubinfeld. 1993. Self-testing/correcting with applications to numerical problems. *J. Comput. Syst. Sci.* 47, 3 (1993), 549–595.

[6] Joan Boyar, Magnus Gausdal Find, and René Peralta. 2016. On various nonlinearity measures for Boolean functions. *Cryptogr. Commun.* 8, 3 (2016), 313–330.

[7] Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp. 2002. Quantum amplitude amplification and estimation. *Contemp. Math.* 305 (2002), 53–74.
[8] Çağdaş Çalık. 2013. Nonlinearity computation for sparse Boolean functions. arXiv:1305.0860. Retrieved from https://arxiv.org/abs/1305.0860.
[9] Anne Canteaut, Claude Carlet, Pascale Charpin, and Caroline Fontaine. 2000. Propagation characteristics and correlation-immunity of highly nonlinear Boolean functions. In *Proceedings of the 19th International Conference on Theory and Application of Cryptographic Techniques (EUROCRYPT'00)*. 507–522. https://dl.acm.org/doi/abs/10.5555/1756169.1756219
[10] Kaushik Chakraborty and Subhamoy Maitra. 2013. Improved quantum test for linearity of a Boolean function. arxiv:quant-ph/1306.6195. Retrieved from https://arxiv.org/abs/1306.6195.
[11] Santanu Dutta and Alok Goswami. 2010. Mode estimation for discrete distributions. *Math. Methods Stat.* 19, 12 (2010), 374–384. DOI: https://doi.org/10.3103/S1066530710040046
[12] Oded Goldreich and Leonid A. Levin. 1989. A hard-core predicate for all one-way functions. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*. ACM, 25–32.
[13] Xiaoyu He, Xiaoming Sun, Guang Yang, and Pei Yuan. 2018. Exact quantum query complexity of weight decision problems via Chebyshev polynomials. arXiv:1801.05717. Retrieved from https://arxiv.org/abs/1801.05717.
[14] Mark Hillery and Erika Andersson. 2011. Quantum tests for the linearity and permutation invariance of Boolean functions. *Phys. Rev. A* 84, 6 (Dec. 2011), 062329.
[15] C. A. Jothishwaran, Anton Tkachenko, Sugata Gangopadhyay, Constanza Riera, and Pantelimon Stanica. 2020. A quantum algorithm to estimate the Gowers U_2 norm and linearity testing of Boolean functions. *Quantum Information Processing* 19, 9 (2020), 1–15.
[16] Sophie Laplante and Frédéric Magniez. 2008. Lower bounds for randomized and quantum query complexity using Kolmogorov arguments. *SIAM J. Comput.* 38, 1 (2008), 46–62.
[17] Hongwei Li. 2019. Quantum algorithms for the Goldreich-Levin learning problem. *Quantum Information Processing* 19, 11 (2020), 1–10.
[18] Ashley Montanaro. 2015. Quantum speedup of Monte Carlo methods. *Proc. Roy. Soc. A: Math. Phys. Eng. Sci.* 471, 2181 (2015), 20150301.
[19] Ashley Montanaro and Tobias J. Osborne. 2010. Quantum Boolean functions. *Chic. J. Theor. Comput. Sci.* 1 (2010), 1–45.
[20] Ryan O'Donnell. 2014. *Analysis of Boolean Functions*. Cambridge University Press, Cambridge, MA.
[21] Palash Sarkar and Subhamoy Maitra. 2000. Nonlinearity bounds and constructions of resilient Boolean functions. In *Proceedings of the Annual International Cryptology Conference*. Springer, 515–532.
[22] Theodore J. Yoder, Guang Hao Low, and Isaac L. Chuang. 2014. Fixed-point quantum search with an optimal number of queries. *Phys. Rev. Lett.* 113, 21 (2014), 210501.

Received September 2020; revised March 2021; accepted March 2021