Theory Comput. Systems **40**, 379–395 (2007)
DOI: 10.1007/s00224-006-1313-z

**Theory of Computing Systems**


# Robust Polynomials and Quantum Algorithms*

Harry Buhrman,[1,2] Ilan Newman,[3] Hein Röhrig,[4] and Ronald de Wolf[1]

[1]Centrum voor Wiskunde en Informatica,
Kruislaan 413, 1098 SJ Amsterdam, The Netherlands
{harry.buhrman,ronald.de.wolf}@cwi.nl

[2]ILLC, University of Amsterdam,
Plantage Muidergracht 24, 1018 TV Amsterdam, The Netherlands

[3]Department of Computer Science, Haifa University,
Mount Carmel, Haifa 31905, Israel
ilan@cs.haifa.ac.il

[4]Department of Computer Science, University of Calgary,
Calgary, Alberta, Canada T2N 1N4
hroehrig@mail.com

**Abstract.** We define and study the complexity of *robust* polynomials for Boolean functions and the related fault-tolerant quantum decision trees, where input bits are perturbed by noise. We compare several different possible definitions. Our main results are:

- For every $n$-bit Boolean function $f$ there is an $n$-variate polynomial $p$ of degree $O(n)$ that *robustly* approximates it, in the sense that $p(x)$ remains close to $f(x)$ if we slightly vary each of the $n$ inputs of the polynomial.
- There is an $O(n)$-query quantum algorithm that *robustly* recovers $n$ noisy input bits. Hence every $n$-bit function can be quantum computed with $O(n)$ queries in the presence of noise. This contrasts with the classical model of Feige et al., where functions such as parity need $\Theta(n \log n)$ queries.

We give several extensions and applications of these results.

* HB was supported by a Vici grant from the Netherlands Organization for Scientific Research (NWO). RdW was supported by a Veni grant from NWO. HB, HR, and RdW were also supported in part by the EU fifth framework projects QAIP, IST-1999-11234, RESQ, IST-2001-37559, and the sixth framework project QAP. IN was partially supported by ISF Grant 55/0.

## 1. Introduction

In the last two decades, polynomials of many varieties have been used quite successfully in complexity theory, both for upper and lower bounds. We study a variety here that is tailored to analyzing algorithms with *noisy input*.

*Robust Polynomials*. A *robust* polynomial for a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ is a real multivariate polynomial $p(z_1,\ldots,z_n)$ such that for every $x = (x_1,\ldots,x_n) \in \{0,1\}^n$ and every $z = (z_1,\ldots,z_n) \in \mathbb{R}^n$, if $\forall i\colon |x_i - z_i| \le \frac{1}{3}$ then $|f(x) - p(z)| \le \frac{1}{3}$ (the $\frac{1}{3}$ in both cases can be changed to any other positive constant less than $\frac{1}{2}$). The *robust degree* of $f$ is the smallest degree of a robust polynomial for $f$; note that we do not require robust polynomials to be multilinear.

The motivation behind the definition of robust polynomials is twofold. First, it can be viewed as a strengthening (restriction) of the notion of approximating polynomials. An approximating polynomial for $f$ is a multivariate real polynomial $q$ that approximates $f$ within an additive term of $\frac{1}{3}$ for each Boolean input. Approximating polynomials for Boolean functions are of interest in themselves and have been the object of study for quite a while. Their minimal degree is tightly related to the decision tree complexity of $f$ [11], [4]. Indeed, this "polynomial method" [4] is one of the main tools for obtaining lower bounds on the number of queries in *quantum* algorithms. One difficulty, however, is that approximating polynomials do not directly compose: if $f(x_1,\ldots,x_n)$ is a Boolean function with an approximating polynomial $p_f$ and $g(y_1,\ldots,y_m)$ is a Boolean function with an approximating polynomial $p_g$, then the polynomial on $n\cdot m$ variables $p_f(p_g,\ldots,p_g)$ obtained by plugging in a copy of $p_g$ for each of the $x_i$ is not necessarily an approximating polynomial for the composed function $f(g,\ldots,g)$ on $n\cdot m$ variables. This difficulty is avoided with robust polynomials: if $p_f$, $p_g$ are robust for $f$, $g$, respectively, then their composition is a robust polynomial (and thus also approximating) for the composed function.

A second motivation for robust polynomials is the study of quantum decision trees that can tolerate noise in their inputs. We show that a natural quantum analogue of classical fault-tolerant decision trees can be defined. As a result, it will follow that every such algorithm that uses $T$ queries to its input bits (and hence every classical noisy decision tree algorithm as well) implies the existence of a robust degree-$2T$ polynomial for the function. This relates the robust degree to fault-tolerant quantum query algorithms in exactly the same way that approximating polynomials are related to bounded-error quantum query algorithms. Surprisingly, our results imply robust quantum algorithms with a linear number of queries, as well as robust polynomials of linear degree, for *any* Boolean function. This should be contrasted with the result of Feige et al. [5]. They proved that for most Boolean functions, an overhead factor of $\Omega(\log n)$ on the number of queries is needed in the noisy case compared to the non-noisy case. In particular, consider the parity function on $n$ variables. This function can be decided trivially by an $n$-query decision tree, and hence can be represented exactly by a real multilinear polynomial of degree $n$ (which is just the single monomial containing all variables in the $\{-1,1\}$ representation). Feige et al. [5] prove that in the noisy decision tree model any algorithm for PARITY needs $\Theta(n\log n)$ queries. Using standard amplification techniques, this yields an $O(n\log n)$-degree robust polynomial for PARITY. Can one do better? Our results imply that there is a robust polynomial for PARITY of degree $O(n)$. However, we only have an indirect description of this polynomial by means of a quantum algorithm, and we do not know of an explicit simple construction of such a polynomial.

*Noisy Quantum Queries.* We now discuss in more detail the model of noisy decision trees in the quantum world. The notion of a "noisy query" in the quantum case is not as obvious and natural as in the classical case, because one application of a quantum query can address many different $x_i$s in superposition. A first proposal would be that for each quantum query, each of the bits is flipped independently with probability $\varepsilon$. Each such quantum query introduces a lot of randomness and the algorithm's state after the query is a mixed quantum state rather than a pure state. In fact, this model is a concrete (and very destructive) form of decoherence; the effects of various forms of decoherence on oracle algorithms like Grover's have been studied before, see e.g., [10] and [12].

A second model, which we adopt here, is to assume that we have $n$ quantum procedures, $A_1, \ldots, A_n$, such that $A_i$ outputs $x_i$ with probability at least $1 - \varepsilon$. Such a *coherent-noise model* is not unreasonable. For instance, it could be the case that the input bits are actually computed for us by subroutines. Such algorithms can always be made coherent by pushing measurements to the end, which means that we can apply and reverse them at will. To enable us to apply the $A_i$s in superposition, we assume we have a black box that maps

$$\mathcal{A}\colon |i\rangle|0\rangle \mapsto |i\rangle A_i|0\rangle.$$

One application of this will count as one query.

A third model, which we call the *multiple-noisy-copies model*, was studied by Szegedy and Chen [13]. Here, instead of $x_i$, the algorithm can only query "perturbed" copies $y_{i,1}, \ldots, y_{i,m}$ of $x_i$. The $y_{i,j}$ are independent Boolean random variables with $\Pr[x_i = y_{i,j}] \geq 1 - \varepsilon$ for each $i = 1, \ldots, n$ and $j = 1, \ldots, m$. In contrast to the first proposal, this model leaves the queries perfectly reversible, since the perturbed copies are fixed at the start of the algorithm and the same $y_{i,j}$ can be queried more than once. The assumption of this model is also stronger than the second model, since we can construct a 1-query $A_i$ that just outputs a superposition of all $y_{i,j}$. If $m$ is sufficiently large, this $A_i$ will compute $x_i$ with high success probability, satisfying the assumption of the second model (see Section 4.2 for details).

*Robust Quantum Algorithms.* Assuming the second model of noisy queries and some fixed $\varepsilon$, we call a quantum algorithm *robust* if it computes $f$ on $n$ inputs with bounded error probability when the $n$ inputs are given by bounded-error algorithms $A_1, \ldots, A_n$, respectively.

A first observation is that every $T$-query non-robust algorithm can be made robust at a multiplicative cost of $O(\log T)$. With $O(\log T)$ queries, a majority gate, and an uncomputation step, we can construct a unitary $\tilde{U}_x$ that approximates an exact quantum query

$$U_x\colon |i\rangle|b\rangle \mapsto |i\rangle|b \oplus x_i\rangle$$

very well in the standard operator norm: $\|U_x - \tilde{U}_x\| \leq 1/(100T)$. Since errors add linearly in a quantum algorithm [3], replacing $U_x$ by $\tilde{U}_x$ in a non-robust algorithm gives a robust algorithm with almost the same final state. In some cases better constructions are possible. For instance, a recent result by Høyer et al. [7] implies a quantum algorithm that robustly computes the $n$-bit OR function with $O(\sqrt{n})$ queries. This is only a constant factor worse than the noiseless case, which is Grover's algorithm [6]. In fact, we do not know of any function where the robust quantum query complexity is more than a constant factor larger than the non-robust complexity.

Our main result about robust quantum algorithms (made precise in Theorem 3) is the following:

> There exists a quantum algorithm that outputs $x_1, \dots, x_n$, with high probability, using $O(n)$ invocations of the $A_i$ algorithms (i.e., queries).

As already mentioned, this result implies that *every* $n$-bit function $f$ can be robustly quantum computed with $O(n)$ queries. This contrasts with the classical $\Omega(n \log n)$ lower bound for PARITY. It is quite interesting to note that quantum computers, which usually are more fragile than classical computers, are actually more robust in the case of computing PARITY in this model with noisy inputs. The result for PARITY can be extended to every symmetric function: for every such function, the optimal quantum algorithm can be made robust with only a constant factor overhead (see Section 4.1).

Our result has a direct bearing on the *direct-sum problem*, which is the question of how the complexity of computing $n$ independent instances of a function scales with the complexity of one instance. One would expect that computing $n$ instances with bounded error takes no more than $n$ times the complexity of one instance. However, since we want all $n$ instances to be computed correctly *simultaneously* with high probability, the only known general method in the classical world is to compute each instance with error probability reduced to $O(1/n)$. This costs another factor of $O(\log n)$. In fact, it follows from the $\Omega(n \log n)$ bound for PARITY that this factor of $\log n$ is optimal if we can only run algorithms for individual instances in a black-box fashion. In contrast, our result implies that in the quantum world, the bounded-error complexity of $n$ instances is at most $O(n)$ times the bounded-error complexity of one instance. This is a very general result. For example, it also applies to communication complexity [9, Section 4.1.1]. If Alice and Bob have a bounded-error protocol for a distributed function $f$, using $c$ bits (or qubits) of communication, then there is a bounded-error quantum protocol for $n$ instances of $f$, using $O(n(c + \log n))$ qubits of communication. The additive $\log n$ is because Alice and Bob need to communicate (possibly in superposition) the index of the instance that they are computing. In contrast, the best-known general classical solution uses $\Theta(cn \log n)$ bits of communication.

*Note About Related Work.* In their manuscript [8], Iwama et al. study a similar but slightly weaker setting. There, the error probability for each input variable is *exactly* $\varepsilon$. If $\varepsilon$ is known, then one can use a version of exact amplitude amplification to "rotate off" the error using $O(1)$ queries and hence make the algorithm robust. If $\varepsilon$ is unknown, it can be estimated very well using quantum amplitude estimation, after which amplitude amplification can be used as if $\varepsilon$ was known. Iwama et al. derive from this that any quantum algorithm can be made robust (in their model) with only a constant factor overhead. Their model has the disadvantage that it does not cover the subroutine scenario, where each input bit $x_i$ is computed for us by an algorithm or subroutine $A_i$ whose error we can only upper bound. Our model does not need the assumption that the error is the same for all input bits, and hence does not have this disadvantage.

## 2. Robust Polynomials—Preliminaries

In this section we study robust polynomials of two different but essentially equivalent types. The first type arises from the multiple-noisy-copies model; the second type is what we discussed in the introduction.

### 2.1. *Two Definitions*

**Definition 1.** Let $\varepsilon \in [0, \frac{1}{2})$. An $(\varepsilon, m)$*-perturbation* of $x \in \{0, 1\}^n$ is a matrix $y$ of $n \times m$ independent binary random variables $y_{i,j}$ such that $\Pr[y_{i,j} = x_i] \geq 1 - \varepsilon$ for each $1 \leq j \leq m$.

**Definition 2.** A *type*-1 $(\varepsilon, m)$*-robust polynomial* for the Boolean function $f\colon \{0, 1\}^n \to \{0, 1\}$ is a real polynomial $p$ in $nm$ variables $y_{i,j}$ (with $1 \leq i \leq n$ and $1 \leq j \leq m$) so that for every $x \in \{0, 1\}^n$ and $y$ an $(\varepsilon, m)$-perturbation of $x$, we have

$$\Pr[|p(y) - f(x)| > \tfrac{1}{3}] < \tfrac{1}{3},$$

where the probability is taken over the distribution on the $nm$ bits in $y$. Moreover, for every $v \in \{0, 1\}^{nm}$, we require $-\frac{1}{3} \leq p(v) \leq \frac{4}{3}$.

Since $y_{i,j}^2 = y_{i,j}$ for a bit $y_{i,j}$, we can restrict attention to *multilinear* polynomials here.

Notice that the error parameter $\frac{1}{3}$ in our definition of type-1 polynomial is consistent with having *expected* error more than $\frac{1}{2}$ for some $x$: it could be that $|p(y) - f(x)| = \frac{1}{3}$ with probability $\frac{2}{3}$, and $|p(y) - f(x)| = \frac{4}{3}$ with probability $\frac{1}{3}$, giving expected error $\frac{2}{3}$. However, this is not a significant problem, as the next lemma shows that the error parameter $\frac{1}{3}$ can be reduced to any small $\delta > 0$ at only a small multiplicative cost in the degree and the number of perturbations. It employs the following Chernoff bound from Theorem A.1.16 of [1].

**Theorem 1** (Chernoff). *Let $X_i$, $1 \leq i \leq k$, be mutually independent random variables with all* $\mathrm{E}[X_i] = 0$ *and all* $|X_i| \leq 1$. *Set* $S = \sum_{i=1}^{k} X_i$. *Then* $\Pr[S > a] \leq e^{-a^2/2k}$.

**Lemma 1.** *Consider any $\delta > 0$. If there is a* type-1 $(\varepsilon, m)$-robust polynomial *$p$ for $f$ of degree $d$, then there exists a type-1 $(\varepsilon, m')$-robust polynomial $q$ for $f$ of degree* $\mathrm{O}(d \log(1/\delta))$ *and* $m' = \mathrm{O}(m \log(1/\delta))$, *such that for $x \in \{0, 1\}^n$ and $y$ an $(\varepsilon, m')$-perturbation of $x$, we have*

$$\Pr[|q(y) - f(x)| > \delta] < \delta.$$

*Moreover, for every $v \in \{0, 1\}^{nm'}$ we have $q(v) \in [0, 1]$.*

*Proof.* We first analyze the following single-variate "amplification polynomial" of degree $k$:

$$h_k(x) = \sum_{i>k/2} \binom{k}{i} x^i (1-x)^{k-i}.$$

Note that $h_k(x)$ is exactly the probability that among $k$ coin flips with bias $x$ toward 1, more than half come up 1. Since it is a probability, we have $h_k(x) \in [0, 1]$ for all $x \in [0, 1]$. Moreover, applying the Chernoff bound with $X_i$ being the outcome of the $i$th coin flip minus $x$, and $a = (\frac{1}{2} - x)k$, we have $h_k(x) \in [0, 2^{-\Omega(k)}]$ for all $x \in [0, \frac{1}{3}]$. Similarly $h_k(x) \in [1-2^{-\Omega(k)}, 1]$ for $x \in [\frac{2}{3}, 1]$. By "stretching" the domain a bit, we can turn this into a degree-$k$ polynomial $h_k$ such that $h_k(x) \in [0, 2^{-\Omega(k)}]$ for $x \in [-\frac{2}{5}, \frac{2}{5}]$, $h_k(x) \in [0, 1]$ for $x \in [\frac{2}{5}, \frac{3}{5}]$, and $h_k(x) \in [1 - 2^{-\Omega(k)}, 1]$ for $x \in [\frac{3}{5}, \frac{7}{5}]$.

We use $r$ independent $(\varepsilon, m)$-perturbations of $x$, denoted $y = y_1, \dots, y_r$, for some number $r$ to be determined later. For each perturbation $y_i$ it holds that $\Pr[|p(y_i) - f(x)| > \frac{1}{3}] < \frac{1}{3}$. Using the amplification polynomial $h_k$ with $k = \mathrm{O}(1)$ we can get the value of $p$ closer to $f$: $\Pr[|h_k(p(y_i)) - f(x)| > \frac{1}{20}] < \frac{1}{3}$. Note that the expected value of $|h_k(p(y_i)) - f(x)|$ is now at most $(\frac{2}{3})(\frac{1}{20}) + (\frac{1}{3})1 = \frac{11}{30}$. Now define an average polynomial $\overline{p}(y) = \frac{1}{r}\sum_{i=1}^{r} h_k(p(y_i))$. Choosing $r = \mathrm{O}(\log(1/\delta))$, the Chernoff bound (with $k = r$, and $X_i$ being the indicator random variable for the event that $|h_k(p(y_i)) - f(x)| > \frac{23}{60}$ minus its expectation) we have

$$\Pr[|\overline{p}(y) - f(x)| > \tfrac{2}{5}] < \delta.$$

Finally we apply $h_k$ again, this time with degree $k = \mathrm{O}(\log(1/\delta))$, in order to get the value of $\overline{p}$ $\delta$-close to the value $f(x)$: if we define $q(y) = h_k(\overline{p}(y))$ then

$$\Pr[|q(y) - f(x)| > \delta] < \delta.$$

The degree of $q$ is $\mathrm{O}(d \log(1/\delta))$, and $m' = mr = \mathrm{O}(m \log(1/\delta))$. The last property of the lemma is also easily seen. □

The second kind of robust polynomial is the following:

**Definition 3.** For a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$, we call $q$ a *type*-2 *$\varepsilon$-robust polynomial for $f$* if $q$ is a real polynomial in $n$ variables such that for every $x \in \{0,1\}^n$ and every $z \in [0,1]^n$ we have $|q(z) - f(x)| \le \frac{1}{3}$ if $|z_i - x_i| \le \varepsilon$ for all $i \in [n]$. If $\varepsilon = 0$, then $q$ is called an *approximating polynomial* for $f$.

Note that we restrict the $z_i$s to lie in the set $[0, \varepsilon] \cup [1-\varepsilon, 1]$ rather than the less restrictive $[-\varepsilon, \varepsilon] \cup [1-\varepsilon, 1+\varepsilon]$. This facilitates later proofs, because it enables us to interpret the $z_i$s as probabilities. However, with some extra work we could also use the less restrictive definition here. Also note that a minimal-degree type-2 robust polynomial for $f$ need not be multilinear, in contrast to the type-1 variety.

**Definition 4.** For $f\colon \{0,1\}^n \to \{0,1\}$, let $\mathrm{rdeg}_1(f)$ denote the minimum degree of any type-1 $(\frac{1}{3}, \mathrm{O}(\log n))$-robust polynomial for $f$, let $\mathrm{rdeg}_2(f)$ be the minimum degree of any type-2 $\frac{1}{3}$-robust polynomial for $f$, and let $\widetilde{\deg}(f)$ be the minimum degree among all approximating polynomials for $f$.

Strictly speaking, we should fix an explicit constant for the $\mathrm{O}(\log n)$ of the type-1 polynomial, but to simplify proofs we will use the $\mathrm{O}(\cdot)$ instead.

### 2.2. *Relation between Type-1 and Type-2 Robust Polynomials*

We characterize the relation of type-1 and type-2 robust polynomials as follows:

**Theorem 2.** *For every type-2 $\varepsilon$-robust polynomial of degree $d$ for $f$ there is a type-1 $(\varepsilon/2, \mathrm{O}(\log(n)/(\frac{1}{2}-\varepsilon)^2))$-robust polynomial of degree $d$ for $f$.*

*Conversely, for every type-1 $(\varepsilon, m)$-robust polynomial of degree $d$ for $f$ there is a type-2 $\varepsilon$-robust polynomial of degree $\mathrm{O}(d)$ for $f$.*

*Proof.* Let $p$ be a type-2 $\varepsilon$-robust polynomial of degree $d$ for $f$. We choose $m = \mathrm{O}(\log(n)/(\frac{1}{2}-\varepsilon)^2)$. If each $y_{i,j}$ is wrong with probability $\le \varepsilon/2$, then the Chernoff bound implies that the probability that the average $\overline{y}_i = \sum_{j=1}^m y_{i,j}/m$ is more than $\varepsilon$ away from $x_i$ is at most $1/(3n)$. Then, by the union bound, with probability at least $\frac{2}{3}$ we have $|\overline{y}_i - x_i| \le \varepsilon$ for all $i \in [n]$ simultaneously. Hence the polynomial $p(\overline{y}_1, \ldots, \overline{y}_n)$ will be a type-1 $(\varepsilon/2, \mathrm{O}(\log(n)/(\frac{1}{2}-\varepsilon)^2))$-robust polynomial of degree $d$ for $f$.

For the other direction, consider a type-1 $(\varepsilon, m)$-robust polynomial of degree $d$ for $f$. Using Lemma 1, we boost the approximation parameters to obtain a type-1 $(\varepsilon, m')$-robust polynomial $p$ of degree $\mathrm{O}(d)$, with $m' = \mathrm{O}(m)$, such that for any $x \in \{0,1\}^n$ and $(\varepsilon, m')$-perturbation $y$ of $x$, $\Pr[|p(y) - f(x)| > \frac{1}{9}] < \frac{1}{9}$. For $z = (z_1, \ldots, z_n)$ define the formal polynomial $q(z)$ (over the reals) by replacing each appearance of $y_{i,j}$ in $p(y)$ with $z_i$. For $z \in \mathbb{R}^n$ with $0 \le z_i \le 1$ for all $i$, let $y_{i,j}$ ($i \in [n]$, $j \in [m']$) be independent 0/1 random variables, where $\mathrm{E}[y_{i,j}] = z_i$. Then the polynomial $q(z)$ that is defined above can be viewed as $q(z) = \mathrm{E}[p(y)]$ because $\mathrm{E}[p(y)] = p(\mathrm{E}[y])$ and $\mathrm{E}[y_{i,j}] = z_i$. In particular, if for $z$ there exists $x \in \{0,1\}^n$ with $|z_i - x_i| \le \varepsilon$ for all $i$, then for any $y \in \{0,1\}^{nm}$ that is an $(\varepsilon, m)$-perturbation of $x$, we have $q(z) = \mathrm{E}[p(y)]$ (here expectation is according to the distribution induced by $y$). Therefore $V := \{v \in \{0,1\}^{nm} \colon |p(v) - f(x)| < 1/9\}$ has probability $\Pr[y \in V] > \frac{8}{9}$ and

$$|f(x) - q(z)| \le \left|\sum_{v \in V} \Pr[y = v](f(x) - p(v))\right| + \left|\sum_{v \notin V} \Pr[y = v](1 + \tfrac{1}{9})\right| < \tfrac{1}{3}.$$

This means that $q(z)$ is a type-2 $\varepsilon$-robust polynomial for $f$ of degree $\mathrm{O}(d)$. □

Note, in all the above we have discussed total Boolean functions. The definitions above make sense also for partial Boolean functions (or promise problems). The theorem as well as the next corollary are true also for such cases.

**Corollary 1.** $\mathrm{rdeg}_1(f) = \Theta(\mathrm{rdeg}_2(f))$ *for every Boolean function* $f\colon \{0,1\}^n \to \{0,1\}$.

### 2.3. *Polynomials Induced by Quantum Algorithms*

The well-known "polynomial method" [2] allows us to make a connection between "robust" quantum algorithms and robust type-1 polynomials:

**Lemma 2.** *Let* $f\colon \{0,1\}^n \to \{0,1\}$ *be a Boolean function. Let* $Q$ *be a quantum algorithm that makes at most* $T$ *queries on inputs* $y$ *from* $\{0,1\}^{n\times m}$, *and let* $Q(y)$ *denote the binary random variable that is its output. If for every* $x \in \{0,1\}^n$ *and* $y$ *an* $(\varepsilon, m)$-*perturbation of* $x$, *we have that* $\Pr_y[Q(y) = f(x)] \geq \frac{8}{9}$ (*probability taken over the distribution on the* $nm$ *bits in* $y$ *as well as over the algorithm*), *then there exists a degree*-$2T$ *type*-1 $(\varepsilon, m)$-*robust polynomial for* $f$.

*Proof.* By Lemma 4.2 of [2], $Q$ induces a degree-$2T$ multilinear polynomial $p$ on $nm$ variables that gives the acceptance probability of $Q$ on fixed input $y \in \{0,1\}^{nm}$, i.e., $p(y) = \Pr[Q(y) = 1]$ (probability taken only over the algorithm). Fix $x \in \{0,1\}^n$. Suppose $f(x) = 0$, then we want to show that $\Pr_y[p(y) > \frac{1}{3}] < \frac{1}{3}$. Since $\Pr_y[Q(y) = f(x) = 0] \geq \frac{8}{9}$, we have $\mathrm{E}_y[p(y)] = \Pr_y[Q(y) = 1] \leq \frac{1}{9}$. Hence Markov's inequality implies $\Pr_y[p(y) > \frac{1}{3}] < \frac{1}{3}$ and we are done. The case $f(x) = 1$ is similar. □

## 3. Quantum Robust Input Recovery

In this section we prove our main result, that we can recover an $n$-bit string $x$ using $\mathrm{O}(n)$ invocations of algorithms $A_1, \ldots, A_n$ where $A_i$ computes $x_i$ with bounded error. Let $|x|$ denote the Hamming weight of a bit string $x$. Our main theorem says that with high probability we can find $t$ 1-bits in the input $x$ (if they are present) using $\mathrm{O}(\sqrt{nt})$ noisy queries.

**Theorem 3.** *Let* $\varepsilon \in [0, \frac{1}{2})$. *Consider* $\varepsilon$-*error algorithms* $A_1, \ldots, A_n$ *that compute the bits* $x = x_1, \ldots, x_n$. *For every* $t$, $1 \leq t \leq n$, *there is a quantum algorithm that makes* $\mathrm{O}(\sqrt{nt})$ *queries* (*invocations of the* $A_i$) *and that outputs* $\tilde{x} = \tilde{x}_1, \ldots, \tilde{x}_n$ *such that with probability at least* $\frac{2}{3}$

1. *for all* $i$, $\tilde{x}_i = 1 \Rightarrow x_i = 1$,
2. $|\tilde{x}| \geq \min\{t, |x|\}$.

In particular, with $t = n$ we obtain $\tilde{x} = x$ using $\mathrm{O}(n)$ queries.

### 3.1. *Some More Preliminaries*

For simplicity we assume that $0 < \varepsilon < \frac{1}{100}$ is fixed and that $A_i$ is a unitary transformation

$$A_i\colon |0^t\rangle \mapsto \alpha_i|0\rangle|\psi_i^0\rangle + \sqrt{1-\alpha_i^2}|1\rangle|\psi_i^1\rangle$$

for some $\alpha_i \geq 0$ such that $|\alpha_i|^2 \leq \varepsilon$ if $x_i = 1$ and $|\alpha_i|^2 \geq 1 - \varepsilon$ if $x_i = 0$; $|\psi_i^0\rangle$ and $|\psi_i^1\rangle$ are arbitrary norm-1 quantum states. The *output* is the random variable obtained from measuring the first qubit. It equals $x_i$ with probability at least $1 - \varepsilon$. It is standard that any quantum algorithm can be expressed in this form by postponing measurements (i.e., unitarily write the measurement in an auxiliary register without collapsing the state); any classical randomized algorithm can be converted into this form by making it reversible and replacing random bits by states $(|0\rangle + |1\rangle)/\sqrt{2}$.

We define the following notion of closeness:

**Definition 5.** For $\varepsilon \in [0, 1/2)$ and algorithms $\mathcal{A} = (A_1, \ldots, A_n)$, we say $\mathcal{A}$ is *$\varepsilon$-close* to $x \in \{0, 1\}^n$ if $\Pr[A_i \text{ outputs } x_i] \geq 1 - \varepsilon$ for all $i \in [n]$.

We sometimes modify our sequence of algorithms $\mathcal{A} = (A_1, \ldots, A_n)$ as follows: For an $n$-bit string $\widetilde{x}$, we negate the answer of $A_i$ if $\widetilde{x}_i = 1$, and denote the resulting sequence of $n$ algorithms by $\mathcal{A}(\widetilde{x})$. Note that $\mathcal{A}(\widetilde{x})$ is close to $0^n$ if and only if $\widetilde{x} = x$. In other words, by finding ones in $\mathcal{A}(\widetilde{x})$, we find positions where $\widetilde{x}$ differs from $x$. In addition, for a set $S \subseteq [n]$ we use $\mathcal{A}^S(\widetilde{x})$ to denote the vector of algorithms $\mathcal{A}(\widetilde{x})$, except that for all $i \notin S$ the $i$th algorithm always outputs 0 instead of running $A_i$. Also, for $S$ as above and $x \in \{0, 1\}^n$ we denote by $x^S \in \{0, 1\}^n$ the string that is identical to $x$ on indices in $S$ and is 0 on indices in $\bar{S}$.

Our algorithm builds on a robust quantum search algorithm by Høyer et al. [7], which we call RobustFind. This subroutine takes a vector $\mathcal{A}$ of $n$ quantum algorithms and in the good case returns an index $i$ such that the "high probability" output of $A_i$ is 1. Formally, the input/output relation of RobustFind is stated in Theorem 4.

**Theorem 4** [7]. *There is a procedure* **RobustFind($n$, $\mathcal{A}$, $\varepsilon$, $\beta$, $\gamma$, $\delta$)** *where* $n \in \mathbb{N}$, $\mathcal{A}$: *$n$ quantum algorithms*, $\varepsilon, \beta, \gamma, \delta > 0$.

**Output:** $i \in [n] \cup \{\perp\}$ *and with the following properties*:

1. *if $\mathcal{A}$ is $\varepsilon$-close to $x \in \{0, 1\}^n$ and $x$ has Hamming weight $|x| \geq \beta n$, then $i \neq \perp$ with probability $\geq 1 - \delta$*,
2. *if $\mathcal{A}$ is $\varepsilon$-close to $x \in \{0, 1\}^n$ and if $i \neq \perp$, then $x_i = 1$ with probability $\geq 1 - \gamma$*

**Complexity:** $O(1/(\frac{1}{2} - \varepsilon)^2 \cdot \sqrt{1/\beta} \cdot \log 1/\gamma\delta)$ *invocations of the* $A_i$.

### 3.2. *The Algorithm and Its Intuition*

Before we formally prove Theorem 3 we explain the intuition and high level of our algorithm (as defined by the AllInputs pseudo code) and of the proof. Clearly, for $t = O(1)$ Theorem 3 is obvious as we can run RobustFind $t$ times to recover $t$ indices $i$ such that $x_i = 1$ with $O(\sqrt{n})$ queries. Therefore all considerations below will be for $t > t_0$ for some $t_0$ that is independent of $n$ and will be specified later.

An important feature of the robust quantum search is that it can be used to verify a purported solution $\tilde{x} \in \{0, 1\}^n$ by running RobustFind on $\mathcal{A}(\tilde{x})$ to find differences with the real input $x$.

Let $x$ be the unique assignment such that $\mathcal{A}$ is $\varepsilon$-close to $x$. Assume first that the Hamming weight is $|x| < 3t/2$. Our idea is to apply RobustFind repeatedly for about

**Procedure** AllInputs($n$, $t$, $\mathcal{A}$, $\varepsilon$)

$n, t \in \mathbb{N}$, $\mathcal{A}$: $n$ algorithms, $\varepsilon > 0$

$\tilde{x} \leftarrow 0^n$

**Part 1, Aim:** *to find a set of indices* $S \subseteq [n]$ *that contains at least* $\min(|x|, t)$ *and at most* $3t/2$ 1s *of the input.*

**for** $3t/2$ times **do**
  $i \leftarrow$ RobustFind($n, \mathcal{A}(\tilde{x}), \varepsilon, \frac{t}{100n}, \frac{1}{100}, \frac{1}{100}$)
  **if** $i \neq \perp$ **then**
    $\tilde{x}_i \leftarrow 1 - \tilde{x}_i$
$S \leftarrow \{i \mid \tilde{x}_i = 1\}$
**if** $|S| < 5t/4$ **then**
  $S \leftarrow [n]$

**Part 2, Aim:** *correctly find all but* $t/\log^2 t$ 1*s.*

$\beta \leftarrow t/100n$
$\tilde{x} \leftarrow 0^n$
**for** $k \leftarrow 1$ to $\log((\log t)^2)$ **do**
  $\beta_k \leftarrow \beta/2^k$
  $t_k \leftarrow 3t/2^k$
  **for** $\ell \leftarrow 1$ to $t_k$ **do**
    $i \leftarrow$ RobustFind($n, \mathcal{A}^S(\tilde{x}), \varepsilon, \beta_k n, \frac{1}{100}, \frac{1}{100}$)
    **if** $i \neq \perp$ **then**
    $\tilde{x}_i \leftarrow 1 - \tilde{x}_i$

**Part 3, Aim:** *correctly find all other* 1*s and eliminate remaining errors.*

**for** $m \leftarrow t/(\log t)^2$ down to 1 **do**
  $i \leftarrow$ RobustFind($n, \mathcal{A}^S(\tilde{x}), \varepsilon, \frac{m}{n}, \frac{1}{20t}, \frac{1}{20t}$)
  **if** $i \neq \perp$ **then**
    $\tilde{x}_i \leftarrow 1 - \tilde{x}_i$
**return** $\tilde{x}$

$3t/2$ times (with threshold, say, $\beta = t/(100n)$) and error probability $\frac{1}{100}$. We expect that for at least a $\frac{98}{100}$-fraction of the calls, RobustFind will return an index $i$ such that $x_i = 1$, and we expect at most a $\frac{2}{100}$-fraction of wrong indices. The first problem to note is that RobustFind might return the same (correct) index over and over again. This is easily resolved as follows: We set $\tilde{x} \in \{0, 1\}^n$ to be $\tilde{x}_i = 1$ for every index $i$ that we obtained from RobustFind and 0 everywhere else, and we call RobustFind with $\mathcal{A}(\tilde{x})$ rather than with $\mathcal{A}$. This means that the 1s that are to be reported by RobustFind are in $x \oplus \tilde{x}$ which is supported on the erroneous indices of $\tilde{x}$, namely, on those indices that are either 1 in $\tilde{x}$ but are 0 in $x$ (false positive) and those indices that are 0 on $\tilde{x}$ while they are 1 on $x$ (false negative).

Done this, we expect about $3t/200$ errors of both kinds (false positive and false negative) in the $3t/2$ calls to RobustFind, which should result in $\tilde{x}$ being quite close to $x$. We then call RobustFind $3t/4$ times hoping to correct some of the errors while not introducing too many new errors. This would be reasonable as we call RobustFind in this second phase half the times we call it in the first phase. Thus we expect to have half the number of new errors, while good chance of correcting many old errors (as they are 1 in $x \oplus \tilde{x}$ and hence RobustFind is expected to report a $\frac{98}{100}$-fraction of them). We keep doing this until the number of expected errors is smaller than $t/(\log^2 t)$. At this point we can afford to run RobustFind for $t/(\log^2 t)$ times, with error probability as low as $1/(20t)$. This finds all remaining errors with high probability. Indeed this is the structure of Parts 2 and 3 of our algorithm.

However, the idea above fails to work when $|x| \gg t$. To see the problem assume that $t = \sqrt{n}$ while $|x| = n/2$. Then, after the first round above, $\tilde{x}$ will be supported on about $\sqrt{n}$ indices, out of which about $\sqrt{n}/100$ might be false positives. However, in every next call to RobustFind, the procedure has about $n/2 - \sqrt{n}$ false negative indices to report back—those that are 1s in $x$ but still 0 in $\tilde{x}$. Thus, even if all the next $\mathrm{O}(t)$ calls return a correct such index, we still might be left with the same $\sqrt{n}/100$ false positive errors that are introduced in the first round. Note that if $t = n$, which is the case when the algorithm is applied to find all inputs, this last discussion is of no concern. However, for relatively small $t$ (which will be needed for some applications, e.g., Theorem 5) we need to introduce a first part to the algorithm. This part is only meant to find a subset $S \subseteq [n]$ such that $|x^S| < 3t/2$. Once this is done, we can use $x^S$ instead of $x$ in the description above, which will now work for every input.

### 3.3. *Detailed Proof*

We now prove that the success probability of the algorithm is at least $\frac{2}{3}$.

*Success Probability.* The algorithm is composed of three parts. We first prove that after Part 1—that is, prior to line 9—we have $\min(t, |x|) \leq |x^S| \leq 3t/2$ with probability $1 - o(1)$.

Indeed, assume first that just prior to the execution of line 7 we have $|S| \geq 5t/4$. Then the upper bound on $|x^S|$ is trivial. For the lower bound assume (by way of contradiction) that $|x^S| < t$. Then we can have $|S| \geq 5t/4$ only if at least $t/4$ wrong indices have been reported by RobustFind. However, as we call RobustFind with $\gamma = \frac{1}{100}$ we expect at most $3t/200$ errors. Thus by the Chernoff bound we have $|x^S| \geq t$ with probability $1 - o(1)$.

If, on the other hand, we reach line 7 with $|S| < 5t/4$ then $S$ is set to be $[n]$, for which the lower bound on $|x^S|$ certainly holds. For the upper bound assume that $|x| \geq 3t/2$. Then to have $|S| < 5t/4$ at line 7 means that at least $t/4 - t/100$ errors occurred in the $3t/2$ calls for RobustFind (an error here is whenever RobustFind returns either $i = \perp$ or a false negative index; the $t/100$ term comes from the threshold $\beta = t/(100n)$). However, the error probability in this case is at most $\frac{2}{100}$ (as we call RobustFind with $\delta = \gamma = \frac{1}{100}$). Thus we expect at most $3t/100$ errors. Again by Chernoff we are done.

Accordingly, we may assume that with probability $1 - o(1)$, the $S$ we have at line 9 is such that $\min(t, |x|) \leq |x^S| \leq 3t/2$. In Part 2 of the algorithm we want to find *correctly* most of the 1s in $x^S$. We maintain $\tilde{x}$ as our current estimate of $x^S$. Initially $\tilde{x} = 0^n$. Denote by $G_k$, $k = 1, \ldots, \log((\log t)^2)$ the event that $|\tilde{x} \oplus x^S| < 30t_k/100$ at the end of the $k$th run of the loop in line 10; $\bar{G}_k$ denotes the complementary event (the negation of $G_k$). We prove inductively that $\Pr[\bar{G}_k | G_{k-1}] = e^{-\Omega(t_k)}$. This together with an assertion that $\Pr[G_1] = e^{-\Omega(t)}$ will imply that at the end of Part 2, $|x^S \oplus \tilde{x}| \le t/\log^2 t$ with probability at least $\frac{9}{10}$, assuming that $t$ is large enough (such that $e^{-\Omega(t_k)} = e^{-\Omega(t/\log^2 t)} < 1/(10\log(\log^2 t))$).

Indeed, let us examine the situation during the first round, namely for $k = 1$. We call RobustFind in the first round for $t_1 = 3t/2$ times with threshold $\beta_1 n = t/200$. Thus, as long as $|x^S \oplus \tilde{x}| > t/200$ happens, each call to RobustFind gives an $i \in [n]$ with probability at least $\frac{99}{100}$. Moreover, we expect at most a $\frac{1}{100}$-fraction of errors in the reported indices. Assume first that at the beginning of the first round $|x^S \oplus \tilde{x}| > 20t/100$ and let $h = |x^S \oplus \tilde{x}| - t/200$. Then after the first $h$ calls to RobustFind we expect at least $\frac{98}{100}$-fraction of correct indices. Thus with probability $e^{-\Omega(t)}$ we will get less than $\frac{90}{100}$ correct indices. However, if we do get at least $\frac{90}{100} \cdot |x^S \oplus \tilde{x}|$ correct indices after those $h$ calls we get an $\tilde{x}$ for which $|x^S \oplus \tilde{x}| \le \frac{20}{100}h \le 6t/100$. Now, assuming this happens, then $\bar{G}_1$ can happen at the end of the first round only if during the rest of the $3t/2 - h \le 129t/100$ remaining calls at least $39t/100$ incorrect indices have been made. As the probability for an incorrect index is bounded by $\frac{1}{100}$ we expect only at most $1.3t/100$ errors. Thus, by Chernoff $39t/100$ errors will occur with probability $e^{-\Omega(t)}$. If, however, at the beginning of the first round $|x^S \oplus \tilde{x}| \le 20t/100$ then by a similar argument $\bar{G}_1$ can happen at the end of the first round only if during the $3t/2$ calls to RobustFind at least $25t/100$ incorrect indices have been made. Again by Chernoff this will happen with probability $e^{-\Omega(t)}$. This concludes the proof that $\Pr[\bar{G}_1] = e^{-\Omega(t)}$.

We now inductively prove that $\Pr[\bar{G}_k | G_{k-1}] \le e^{-\Omega(t_k)}$.

Indeed, assume that $G_{k-1}$ happens, namely that just prior to the beginning of the $k$th round we have $|\tilde{x} \oplus x^S| < 30t_{k-1}/100 = 60t_k/100$. In round $k$ we call RobustFind with threshold $\beta_k n = t_k/200$; hence, as long as $|\tilde{x} \oplus x^S| > t_k/200$, we expect RobustFind to return an index $i \in [n] \cap S$ with probability at least $\frac{99}{100}$. Moreover, every time it returns a correct index (which occurs with probability at least $\frac{99}{100}$) it is a 1 in $(\tilde{x} \oplus x^S)$, hence it reduces the weight of symmetric difference (the total number of errors) by 1.

Suppose first that prior to round $k$, $|\tilde{x} \oplus x^S| < 30t_k/100$. Then, for $\bar{G}_k$ to happen at the end of round $k$, RobustFind would need to return at least $31t_k/100$ wrong indices, namely $i \in [n] \cap S$ such that $\tilde{x}_i = x_i$. (Returning a $\perp$ here does not count as a false index.) However, as the probability of a wrong index is at most $\frac{1}{100}$ and RobustFind is called $t_k$ times, then, by Chernoff, the probability of $\bar{G}_k$ is $e^{-\Omega(t_k)}$.

Assume now that $|\tilde{x} \oplus x^S| \ge 30t_k/100$ at the beginning of round $k$. Recall also that by the assumption that $G_{k-1}$ occurs, we have $|\tilde{x} \oplus x^S| < 60t_k/100$ at the beginning of the $k$th round. Consider the first $h = |\tilde{x} \oplus x^S| - t_k/200$ calls for RobustFind. In each of those calls $|\tilde{x} \oplus x^S| > t_k/200 = \beta_k n$, hence with probability $\frac{99}{100}$ every such call returns an index $i \in [n] \cap S$ which is then a correct index with probability $\frac{99}{100}$. Thus we expect that at least $\frac{98}{100} \cdot h$ correct indices will be returned in the first $h$ calls. By Chernoff, the probability that the number of correctly returned indices in those $h$ calls is fewer than $90h/100$ is $e^{-\Omega(t_k)}$ (as $h \ge 15t_k/100$). However, if the number of correctly returned indices is at least $90h/100$, then after the first $h$ calls of RobustFind, $|\tilde{x} \oplus x^S| < 0.2h \le 0.2 \cdot 59t_k/100 < 12t_k/100$. Thus, at this point we are still left with $3t_k/2 - h$ calls to RobustFind which will result in $\bar{G}_k$ only if at least $48t_k/100$ wrong indices will be returned. This again will happen with probability $e^{-\Omega(t_k)}$. We conclude that in all cases $\Pr[\bar{G}_k | G_{k-1}] = e^{-\Omega(t_k)}$.

Note that $t_k > t/(\log^2 t)$. Thus if we choose $t > t_0$ such that for every $k$ the probability $\Pr[\bar{G}_k | G_{k-1}] = e^{-\Omega(t_k)} < 1/(10t)$ we get that $\Pr[\bar{G}_k] < \frac{1}{10}$ for $k = \log^2 t$ after the end of Part 2. Hence, with probability at least 0.8, we have $|\tilde{x} \oplus x| < t/(\log t)^2$ bad indices at the end of the `for` loop in lines 11–17.

Finally, in Part 3 we find (with probability close to 1) all remaining wrong indices by making the individual error probability in RobustFind so small that we can use the union bound: we determine each of the remaining bad indices with error probability $1/(10t)$. This implies an overall success probability of at least $0.8 \cdot 0.9 > \frac{2}{3}$.

*Complexity.* Clearly the complexity is determined by Parts 2 and 3 of the algorithm. We bound the number of queries to $f$ in lines 11–17 as follows:

$$\mathrm{O}\left(\sum_{k=1}^{\log(\log^2 t)} t_k \sqrt{1/\beta_k}\right) = \mathrm{O}\left(\sum_{k=1}^{\log(\log^2 t)} \frac{t}{2^k}\sqrt{\frac{n2^k}{t}}\right) = \mathrm{O}\left(\sqrt{nt}\right). \tag{1}$$

The number of queries in lines 18–21 is bounded by

$$\mathrm{O}\left(\sum_{m=1}^{t/(\log t)^2} \sqrt{\frac{n}{m}}\log t\right) = \mathrm{O}\left(\sqrt{nt}\right).$$

Therefore, the total query complexity of AllInputs is $\mathrm{O}(\sqrt{nt})$.

## 4. Making Quantum Algorithms Robust

### 4.1. *Inputs Computed by Quantum Algorithms*

Here we state a few corollaries of Theorem 3. First, once we have recovered the input $x$ we can compute any function of $x$ without further queries, hence

**Corollary 2.** *For every $f\colon \{0,1\}^n \to \{0,1\}$, there is a robust quantum algorithm that computes $f$ using $\mathrm{O}(n)$ queries.*

In particular, PARITY can be robustly quantum computed with $\mathrm{O}(n)$ queries while it takes $\Omega(n \log n)$ queries classically [5].

Second, in the context of the direct-sum problem, the complexity of quantum computing a vector of instances of a function scales linearly with the complexity of one instance.

**Corollary 3** (Direct Sum). *If there exists a $T$-query bounded-error quantum algorithm for $f$, then there is an $\mathrm{O}(Tn)$-query bounded-error quantum algorithm for $n$ independent instances of $f$.*

As mentioned, the best classical upper bound has an additional factor of $\log n$, and this is optimal in a classical black-box setting.

Third, all *symmetric* functions can be computed robustly on a quantum computer with the same asymptotic complexity as non-robustly. A function is symmetric if its value only depends on the Hamming weight of the input. Let $\Gamma(f) := \min\{|2k - n + 1|\colon f$ changes value if the Hamming weight of the input changes from $k$ to $k + 1\}$. Beals et al. [2, Theorem 4.10] exhibited a bounded-error quantum algorithm for $f$ using $\mathrm{O}(\sqrt{n(n - \Gamma(f) + 1)})$ quantum queries, which is optimal. We show that this upper bound remains valid also for *robust* algorithms.

**Theorem 5.** *For every symmetric function $f$, there is a robust quantum algorithm that computes $f$ using $\mathrm{O}(\sqrt{n(n - \Gamma(f) + 1)})$ quantum queries.*

*Proof.* Note that $f$ is constant when the Hamming weight of its input lies in the middle interval $[(n - \Gamma(f))/2, (n + \Gamma(f) - 2)/2]$. Using two applications of Theorem 3 with sufficiently small error probability, we robustly search for $\lceil (n - \Gamma(f))/2 \rceil$ ones and $n - \lceil (n + \Gamma(f) - 2)/2 \rceil$ zeros in the input. If both of these searches succeeded (i.e., found the required zeros and ones), then we know that our input lies in the middle interval. If the search for zeros failed (i.e., ended with fewer zeros) then we know *all* zeros and hence the whole input $x$. Similarly, if the search for ones failed then we know $x$. Either way, we can output $f(x)$. □

### 4.2. *Multiple Noisy Copies*

As mentioned in the Introduction, the assumption that we have a bounded-error algorithm $A_i$ for each of the input bits $x_i$ also covers the model of [13] where we have a sequence $y_{i,1}, \ldots, y_{i,m}$ of noisy copies of $x_i$. These we can query by means of a mapping

$$|i\rangle|j\rangle|0\rangle \mapsto |i\rangle|j\rangle|y_{i,j}\rangle.$$

Here we spell out this connection in some more detail. First, by a Chernoff bound, choosing $m := \mathrm{O}(\log(n)/\varepsilon^2)$ implies that the average $\overline{y}_i := \sum_{j=1}^{m} y_{i,j}/m$ is close to $x_i$ with very high probability:

$$\Pr[|\overline{y}_i - x_i| \geq 2\varepsilon] \leq \frac{1}{100n}.$$

By the union bound, with probability $\frac{99}{100}$ this closeness will hold for all $i \in [n]$ simultaneously. Assuming this is the case, we implement the following unitary mapping using one query:

$$A_i\colon |0^{\log(m)+1}\rangle \mapsto \frac{1}{\sqrt{m}} \sum_{j=1}^{m} |j\rangle|y_{i,j}\rangle.$$

Measuring the last qubit of the resulting state gives $x_i$ with probability at least $1 - 2\varepsilon$. Hence, we can run our algorithm from Section 3 and recover $x$ using $\mathrm{O}(n)$ queries to the $y_{i,j}$. Similarly, all consequences mentioned in Section 4.1 hold for this multiple-noisy-copies model as well.

## 5. Making Approximating Polynomials Robust

The next theorem follows immediately from earlier results.

**Theorem 6.** $\mathrm{rdeg}_{1,2}(f) = O(n)$ *for every* $f\colon \{0,1\}^n \to \{0,1\}$.

*Proof.* By Corollary 2 and the discussion in Section 4.2, $f$ has an $O(n)$-query robust quantum algorithm in the multiple-noisy-copies model that operates on $O(\log n)$ copies. By Lemma 2 this induces a type-1 robust polynomial for $f$ of degree $O(n)$. Finally, by Corollary 1 there also exists a degree-$O(n)$ type-2 robust polynomial for $f$. □

In particular, this shows that for functions with approximate degree $\Theta(n)$ we can make the approximating polynomial robust at only constant factor overhead in the degree. This case includes explicit functions like PARITY and MAJORITY, but also random (hence almost all) functions. It is open whether approximating polynomials can *always* be made robust at only a constant overhead in the degree. The best we can do is show that a non-robust degree-$d$ approximating polynomial can be made robust at a cost of a factor $O(\log d)$. Our proof makes use of the well-known notion of *certificate complexity*.

**Definition 6.** An assignment $C\colon S \to \{0,1\}$ of values to some subset $S \subseteq [n]$ of the $n$ variables is *consistent* with $x \in \{0,1\}^n$ if $x_i = C(i)$ for all $i \in S$. For $b \in \{0,1\}$, a *b-certificate* for $f$ is an assignment $C$ such that $f(x) = b$ whenever $x$ is consistent with $C$. The *size* of $C$ is $|S|$, the cardinality of $S$. The *certificate complexity* $C_x(f)$ *of* $f$ *on* $x$ is the size of a smallest $f(x)$-certificate that is consistent with $x$. The *certificate complexity* of $f$ is $C(f) = \max_x C_x(f)$.

**Lemma 3.** *Let $p$ be an $\varepsilon$-approximating polynomial for $f\colon \{0,1\}^n \to \{0,1\}$, and let $c = C(f)$ be the certificate complexity of $f$. If $x \in \{0,1\}^n$ and $z \in [0,1]^n$ satisfy $|x_i - z_i| \le 1/(10c)$ for all $i \in [n]$, then $|p(z) - f(x)| \le 6\varepsilon/5 + \frac{1}{10}$.*

*Proof.* Consider a certificate $C$ for $x$ of size $c$. We will use $x^C$ and $x^{\overline{C}}$ to denote the parts of $x$ corresponding to $C$ and to its complement, respectively, and write $x = x^C x^{\overline{C}}$. If $y \in \{0,1\}^n$ is chosen according to the $z$-distribution ($y_i = 1$ with probability $z_i$), then

$$p(z) = \mathrm{E}_y[p(y)] = \sum_{y^C y^{\overline{C}}} \Pr[y^C]\Pr[y^{\overline{C}}]p(y^C y^{\overline{C}}) = \sum_{y^{\overline{C}}} \Pr[y^{\overline{C}}] \cdot \mathrm{E}_{y^C}[p(y^C y^{\overline{C}})].$$

Now consider the expectation $\mathrm{E}_{y^C}[p(y^C y^{\overline{C}})]$, where $y^{\overline{C}} \in \{0,1\}^{n-c}$ is fixed, while the $y^C$-bits are still chosen according to the $z$-distribution. Consider the $c$-variate polynomial obtained from $p$ by fixing the bits in $y^{\overline{C}}$. Since the "error" in the $z^C$-variables is at most $1/10c$, we have $\Pr[y^C = x^C] \ge (1 - 1/(10c))^c \ge \frac{9}{10}$. If $y^C \ne x^C$, then the difference between $p(y^C y^{\overline{C}})$ and $p(x^C y^{\overline{C}})$ is at most $1 + 2\varepsilon$, so

$$|\,\mathrm{E}_{y^C}[p(y^C y^{\overline{C}})] - p(x^C y^{\overline{C}})| \le (1 + 2\varepsilon)/10.$$

However, $f(x^C y^{\overline{C}}) = f(x)$, because the input $x^C y^{\overline{C}}$ is consistent with the same certificate as $x$. Hence

$$\begin{aligned} |\,\mathrm{E}_{y^C}[p(y^C y^{\overline{C}})] - f(x)| &\leq |\,\mathrm{E}_{y^C}[p(y^C y^{\overline{C}})] - p(x^C y^{\overline{C}})| + |p(x^C y^{\overline{C}}) - f(x)| \\ &\leq (1+2\varepsilon)/10 + \varepsilon = \tfrac{1}{10} + 6\varepsilon/5, \end{aligned}$$

and also $|p(z) - f(x)| \leq 6\varepsilon/5 + \frac{1}{10}$. □

This lemma implies that we can make a non-robust approximating polynomial robust at the cost of a factor of $\mathrm{O}(\log C(f))$ in the degree: replace each variable by an $\mathrm{O}(\log C(f))$-degree amplification polynomial as used in the proof of Lemma 1. Since it is known that $C(f)$ and $\widetilde{\deg}(f)$ are polynomially related ($C(f) = \mathrm{O}(\widetilde{\deg}(f)^4)$, see [4]), we obtain:

**Theorem 7.** $\mathrm{rdeg}_{1,2}(f) = \mathrm{O}(\widetilde{\deg}(f) \cdot \log \widetilde{\deg}(f))$.

## 6. Open Problems

We mention some open problems. First, in contrast to the classical case (PARITY) we do not know of any function where making a quantum algorithm robust costs more than a constant factor. Such a constant overhead suffices in the case of symmetric functions and functions whose approximate degree is $\Omega(n)$. It is conceivable that quantum algorithms (and polynomials) can *always* be made robust at a constant factor overhead. Proving or disproving this would be very interesting.

Second, we are not aware of a direct "closed form" or other natural way to describe a robust degree-$n$ polynomial for the parity of $n$ bits, but can only infer its existence from the existence of a robust quantum algorithm. Given the simplicity of the non-robust representing polynomial for PARITY, one would hope for a simple closed form for robust polynomials for PARITY as well.

Finally, we have chosen our model of a noisy query such that we can coherently make a query and reverse it. It is not clear to what extent non-robust quantum algorithms can be made resilient against decohering queries, since the usual transformations to achieve fault-tolerant quantum computation do not immediately apply to the query gate, which acts on a non-constant number of quantum bits simultaneously.

## Acknowledgments

We thank Peter Høyer for inspiring initial discussions that led to our main result, and Michele Mosca for sending us a version of [8]. Oded Regev pointed out that when recovering all input bits, the quantum-search subroutine does not need to be robust. Thanks to the anonymous *TOCS* referees for many helpful comments.

## References

[1] N. Alon and J. H. Spencer. *The Probabilistic Method*, second edition. Wiley-Interscience, New York, 2000.

[2] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf. Quantum lower bounds by polynomials. In *Proceedings of* 39*th IEEE FOCS*, pages 352–361, 1998. quant-ph/9802049.

[3] E. Bernstein and U. Vazirani. Quantum complexity theory. *SIAM Journal on Computing*, 26(5):1411–1473, 1997. Earlier version in STOC '93.

[4] H. Buhrman and R. de Wolf. Complexity measures and decision tree complexity: A survey. *Theoretical Computer Science*, 288(1):21–43, 2002.

[5] U. Feige, P. Raghavan, D. Peleg, and E. Upfal. Computing with noisy information. *SIAM Journal on Computing*, 23(5):1001–1018, 1994. Earlier version in STOC '90.

[6] L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of* 28*th ACM STOC*, pages 212–219, 1996. quant-ph/9605043.

[7] P. Høyer, M. Mosca, and R. de Wolf. Quantum search on bounded-error inputs. In *Proceedings of* 30*th International Colloquium on Automata*, *Languages and Programming* (*ICALP'*03), volume 2719 of Lecture Notes in Computer Science, pages 291–299. Springer, Berlin, 2003. quant-ph/0304052.

[8] K. Iwama, R. Putra, and S. Yamashita. Quantum query complexity of biased oracles. Unpublished manuscript, 2003.

[9] E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge University Press, Cambridge, 1997.

[10] G. L. Long, Y. S. Li, W. L. Zhang, and C. C. Tu. Dominant gate imperfection in Grover's quantum search algorithm. *Physical Review A*, 61:042305, 2000. quant-ph/9910076.

[11] N. Nisan and M. Szegedy. On the degree of Boolean functions as real polynomials. *Computational Complexity*, 4(4):301–313, 1994. Earlier version in STOC'92.

[12] N. Shenvi, K. Brown, and K. B. Whaley. Effects of noisy oracle on search algorithm complexity. quant-ph/0304138, 21 Apr 2003.

[13] M. Szegedy and X. Chen. Computing Boolean functions from multiple faulty copies of input bits. In *Proceedings of* 5*th LATIN*, volume 2286 of Lecture Notes in Computer Science, pages 539–553. Springer, Berlin, 2002.

*Received in final form March* 21, 2006. *Online publication March* 1, 2007.