OPTIMAL QUANTUM ALGORITHM FOR POLYNOMIAL INTERPOLATION

ANDREW M. CHILDS[*,†], WIM VAN DAM[‡], SHIH-HAN HUNG[†], AND IGOR E. SHPARLINSKI[§]

Abstract. We consider the number of quantum queries required to determine the coefficients of a degree-d polynomial over $\mathbb{F}_q$. A lower bound shown independently by Kane and Kutin and by Meyer and Pommersheim shows that $d/2 + 1/2$ quantum queries are needed to solve this problem with bounded error, whereas an algorithm of Boneh and Zhandry shows that d quantum queries are sufficient. We show that the lower bound is achievable: $d/2 + 1/2$ quantum queries suffice to determine the polynomial with bounded error. Furthermore, we show that $d/2 + 1$ queries suffice to achieve probability approaching 1 for large q. These upper bounds improve results of Boneh and Zhandry on the insecurity of cryptographic protocols against quantum attacks. We also show that our algorithm's success probability as a function of the number of queries is precisely optimal. Furthermore, the algorithm can be implemented with gate complexity $\operatorname{poly}(\log q)$ with negligible decrease in the success probability. We end with a conjecture about the quantum query complexity of multivariate polynomial interpolation.

1. Introduction

Let $f(X) = c_d X^d + \cdots + c_1 X + c_0 \in \mathbb{F}_q[X]$ be an unknown polynomial of degree d, specified by its coefficient vector $c \in \mathbb{F}_q^{d+1}$. Suppose q and d are known[1] and we are given a black box that evaluates f on any desired $x \in \mathbb{F}_q$. In the *polynomial interpolation problem*, our goal is to learn f—that is, to determine the vector c—by querying this black box. We would like to determine how many queries are required to solve this problem.

The classical query complexity of polynomial interpolation is well known: $d + 1$ queries to f are clearly sufficient and are also necessary to determine the polynomial, even with bounded error. Shamir [18] used this fact to construct a cryptographic protocol that divides a secret into $d + 1$ parts such that knowledge of all the parts can be used to infer the secret, but any d parts give no information about the secret. The security of this protocol relies on the fact that if f is chosen uniformly at random, and if we only know d function values $f(x_1), \ldots, f(x_d)$, then we cannot guess the value $f(x_{d+1})$ for a point $x_{d+1} \notin \{x_1, \ldots, x_d\}$ with probability greater than $1/q$ (that is, there is no advantage over random guessing). This example motivates understanding the query complexity of polynomial interpolation precisely, since a single query can dramatically increase the amount of information that can be extracted.

The quantum query complexity of polynomial interpolation has also been studied previously. Kane and Kutin [10] and Meyer and Pommersheim [13] independently showed that $d/2 + 1/2$ quantum queries are needed to solve the problem with bounded error. Furthermore, Kane and Kutin conjectured that $d + 1$ quantum queries might be necessary. This was refuted by Boneh and Zhandry, who showed that d quantum queries suffice to solve the problem with probability[2] $1 - O(1/q)$ [3]. To show this, they described a 1-query quantum algorithm that determines a linear

[*]Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland

[†]Joint Center for Quantum Information and Computer Science, University of Maryland

[‡]Departments of Computer Science and Physics, University of California, Santa Barbara

[§]Department of Pure Mathematics, University of New South Wales

[1] We assume $q > d$ so that different coefficients correspond to distinct functions $f\colon \mathbb{F}_q \to \mathbb{F}_q$.

[2] While the notation $O(\cdot)$ only indicates an asymptotic upper bound on the absolute value, we sometimes write $1-O(\cdot)$ to indicate a bound on a quantity that is at most 1.

polynomial with probability $1 - O(1/q)$. The result for general d follows because $d - 1$ classical queries can be used to reduce the case of a degree-d polynomial to that of a linear polynomial. However, this work left a substantial gap between the lower and upper bounds.

Here we present an improved quantum algorithm for polynomial interpolation. We show that the aforementioned lower bounds are tight: with d fixed, $k = d/2 + 1/2$ queries suffice to solve the problem with constant success probability. While the success probability at this value of k has a q-independent lower bound, it decreases rapidly with k, scaling like $1/k!$. This raises the question of how the success probability increases as we make more queries. We show that there is a sharp transition as k is increased: in particular, with $k = d/2 + 1$ queries, the algorithm succeeds with a probability that approaches 1 for large q.

Our algorithm is motivated by the pretty good measurement (PGM) approach to the hidden subgroup problem (HSP) [1]. In this approach, one queries the black box on uniform superpositions to create *coset states* and then makes entangled measurements on several coset states to infer the hidden subgroup. As in the PGM approach (and in other approaches to the HSP using the so-called standard method), our algorithm makes nonadaptive queries to the black box and performs collective postprocessing. Also, similarly to previous analysis of the PGM approach, we can express our success probability in terms of the number of solutions of a system of polynomial equations.

However, our approach to polynomial interpolation also has significant differences from the PGM approach to the HSP. In particular, we introduce a different way to query the black box that simplifies both the algorithm and its analysis. In the PGM approach, we query the black box on a uniform superposition and then uncompute uniform superpositions over certain sets. For polynomial interpolation, we instead query a carefully-chosen non-uniform superposition of inputs so that the subsequent uncomputation is classical. Furthermore, the success probability of our method is higher, and its analysis is more straightforward, than if we used a direct analog of the PGM approach. We hope that these techniques will prove useful for other quantum algorithms, perhaps for the hidden subgroup problem or for other applications of the PGM approach [5,7].

We also show that our strategy is precisely optimal: for any number of queries k, we describe a k-query algorithm with the highest possible success probability. We give a simple algebraic characterization of this success probability, as follows.

Theorem 1. *The maximum success probability of any k-query quantum algorithm for interpolating a polynomial of degree d over $\mathbb{F}_q$ is $|R_k|/q^{d+1}$, where $R_k := Z(\mathbb{F}_q^k \times \mathbb{F}_q^k)$ is the range of the function $Z\colon \mathbb{F}_q^k \times \mathbb{F}_q^k \to \mathbb{F}_q^{d+1}$ defined by $Z(x,y)_j := \sum_{i=1}^k y_i x_i^j$ for $j \in \{0,1,\ldots,d\}$.*

We present an explicit quantum algorithm that achieves this success probability, and we show that no algorithm can do better. We establish optimality with an argument based on the dimension of the space spanned by the possible output states, which appears to be distinct from arguments using the two main approaches to proving limitations on quantum algorithms, the polynomial and adversary methods. Instead, our approach is closely related to a linear-algebraic lower bound technique of Radhakrishnan, Sen, and Venkatesh [17] and to the "rank method" of Boneh and Zhandry [3].

We characterize the query complexity by proving bounds on $|R_k|$, as follows.

Theorem 2. *For any fixed positive integer d, the success probability of Theorem 1 is*

(i) $|R_k|/q^{d+1} = \frac{1}{k!}(1 - O(1/q))$ *if d is odd and $k = \frac{d}{2} + \frac{1}{2}$, and*

(ii) $|R_k|/q^{d+1} = 1 - O(1/q)$ *if d is even and $k = \frac{d}{2} + 1$.*

To show the former bound, we explicitly characterize the possible $(x,y) \in \mathbb{F}_q^k \times \mathbb{F}_q^k$ such that $Z(x,y)$ takes a particular value. We prove the latter bound in a completely different way, using a second moment argument.

Theorem 2 shows that the success probability has a sharp transition as a function of k, from subconstant for $k < d/2 + 1/2$ (by known lower bounds [10, 13]), to a (d-dependent) constant for $k = d/2 + 1/2$, to $1 - o(1)$ for $k = d/2 + 1$. Note that since k must be an integer, the success probability varies differently with k depending on whether d is odd or even. For fixed even d, $k = d/2 + 1$ queries give success probability $1 - o(1)$, whereas $k = d/2$ queries give success probability $o(1)$. For fixed odd d, the success probability is $o(1)$ for $k = d/2 - 1/2$ and constant for $k = d/2 + 1/2$. To achieve higher success probability, we can make $k = d/2 + 3/2$ queries and treat f as a polynomial of degree $d + 1$ with $c_{d+1} = 0$, giving success probability $1 - o(1)$.

In light of these results, polynomial interpolation is reminiscent of the task of computing the parity of n bits, where the classical query complexity is n (even for bounded error) and the quantum query complexity is $n/2$ [2, 8]. More generally, a similar factor-of-two improvement is possible for the oracle interrogation problem, where the goal is to learn the entire n-bit string encoded by a black box [6]. However, polynomial interpolation is qualitatively different in that the oracle returns values over $\mathbb{F}_q$ rather than $\mathbb{F}_2$. Note that for the oracle interrogation problem over $\mathbb{F}_q$, one can only achieve speedup by a factor of about $1 - 1/q$ [3, Section 4], which is negligible for large q.

Our algorithm improves results of Boneh and Zhandry giving quantum attacks on certain cryptographic protocols [3]. For a version of the Shamir secret sharing scheme [18] where the shares can be quantum superpositions, their d-query interpolation algorithm shows that a subset of only d parties can recover the secret. Our algorithm considerably strengthens this, showing that a subset of $d/2 + 1/2$ parties can recover the secret with constant probability, and $d/2 + 1$ can recover it with probability $1 - O(1/q)$. Boneh and Zhandry also formulate a model of quantum message-authentication codes (MACs), where the goal is to tag messages to authenticate the sender. Informally, a MAC is called d-time if, given the ability to create d valid message-tag pairs, an attacker cannot forge another valid message-tag pair. Boneh and Zhandry show that there are $(d + 1)$-wise independent functions that are not d-time quantum MACs. Our result improves this to show that there are $(d + 1)$-wise independent functions that are not $(d/2 + 1/2)$-time quantum MACs.

Finally, we consider the gate complexity of polynomial interpolation. We call an algorithm *gate-efficient* if it can be implemented with a number of 2-qubit gates that is only larger than its query complexity by a factor of $\operatorname{poly}(\log q)$. We construct a gate-efficient variant of our algorithm that achieves almost the same success probability.[3]

Theorem 3. *For any fixed positive integer d, there is a gate-efficient quantum algorithm for interpolating a polynomial of degree d over $\mathbb{F}_q$ using*

(i) $k = \frac{d}{2} + \frac{1}{2}$ *queries, succeeding with probability* $\frac{1}{k!}(1 - O(1/q))$, *if d is odd; and*

(ii) $k = \frac{d}{2} + 1$ *queries, succeeding with probability* $1 - o(1)$, *if d is even.*

The main step in implementing the algorithm is to invert the function Z described in the statement of Theorem 1, i.e., to find some $x, y \in \mathbb{F}_q^k$ so that $Z(x, y)$ takes a given value. We achieve this by characterizing the solutions in terms of a polynomial equation and a system of linear equations.

In Section 5 we discuss the more general case where $f \in \mathbb{F}_q[X_1, \ldots, X_n]$ is a multivariate polynomial of degree d. While our algorithm generalizes straightforwardly, the analysis of its success probability is more complicated. We conjecture that the quantum query complexity of this problem is smaller than the classical query complexity by a factor of $n + 1$.

The remainder of the paper is organized as follows. After introducing some definitions in Section 2.1, we describe our k-query algorithm in Section 2.2. We analyze the success probability of this algorithm for $k = d/2 + 1/2$ in Section 2.3, and for $k = d/2 + 1$ in Section 2.4. We also show in Section 2.5 that essentially the same performance can be achieved using k independent queries to the oracle, each on a uniform superposition of inputs (which might make some cryptographic attacks easier, depending on the model). We establish optimality of our algorithm in

[3] Note that while our algorithm for $k = d/2 + 1/2$ has gate complexity polynomial in both $\log q$ and d, the algorithm for $k = d/2 + 1$ has gate complexity $k! \operatorname{poly}(\log q)$. Improving the dependence on d is a natural open question.

Section 3. In Section 4, we describe the gate-efficient version of our algorithm. Finally, we conclude in Section 5 with a brief discussion of some open questions.

2. Quantum algorithm for polynomial interpolation

2.1. **Preliminaries.** Let $f(X) = c_d X^d + \cdots + c_1 X + c_0 \in \mathbb{F}_q[X]$ be an unknown polynomial of degree d that is specified by the vector of coefficients $c \in \mathbb{F}_q^{d+1}$, where $q = p^r$ a power of a prime p. Access to f is provided by a black box acting as $|x, y\rangle \mapsto |x, y + f(x)\rangle$ for all $x, y \in \mathbb{F}_q$.

Let $e\colon \mathbb{F}_q \to \mathbb{C}$ be the exponential function $e(z) = \mathrm{e}^{2\pi \mathrm{i}\,\mathrm{Tr}(z)/p}$, where the trace function $\mathrm{Tr}\colon \mathbb{F}_q \to \mathbb{F}_p$ is defined by $\mathrm{Tr}(z) = z + z^p + z^{p^2} + \cdots + z^{p^{r-1}}$. The Fourier transform over $\mathbb{F}_q$ is the unitary transformation acting as $|x\rangle \mapsto \frac{1}{\sqrt{q}} \sum_{y \in \mathbb{F}_q} e(xy)|y\rangle$ for all $x \in \mathbb{F}_q$.

We can compute the value of f into the phase by Fourier transforming the second query register. If we apply the inverse Fourier transform, perform a query, and then apply the Fourier transform, we have the transformation

$$|x, y\rangle \mapsto \frac{1}{\sqrt{q}} \sum_{z \in \mathbb{F}_q} e(-yz)|x, z\rangle \tag{1}$$

$$\mapsto \frac{1}{\sqrt{q}} \sum_{z \in \mathbb{F}_q} e(-yz)|x, z + f(x)\rangle \tag{2}$$

$$\mapsto \frac{1}{q} \sum_{z,w \in \mathbb{F}_q} e(-yz + (z + f(x))w)|x, w\rangle \tag{3}$$

$$= e(yf(x))|x, y\rangle \tag{4}$$

for any $x, y \in \mathbb{F}_q$, where we used the fact that $\sum_{z \in \mathbb{F}_q} e(zv) = q\delta_{z,v}$. We call the transformation $|x, y\rangle \mapsto e(yf(x))|x, y\rangle$ a *phase query*. Since a phase query can be implemented with a single standard query and vice versa, the query complexity of a problem does not depend on which type of query we use.

For vectors $x, y \in \mathbb{F}_q^k$, we denote the inner product over $\mathbb{F}_q$ by $x \cdot y := \sum_{i=1}^k x_i y_i$. The k-fold Fourier transform (i.e., the Fourier transform acting independently on each register) acts as $|x\rangle \mapsto \frac{1}{\sqrt{q^k}} \sum_{y \in \mathbb{F}_q^k} e(x \cdot y)|y\rangle$ for any $x \in \mathbb{F}_q^k$.

2.2. **The algorithm.** We now describe our algorithm for polynomial interpolation. An ideal algorithm would produce the Fourier transform of the coefficient vector $c \in \mathbb{F}_q^{d+1}$, that is, the state

$$|\hat{c}\rangle = \frac{1}{\sqrt{q^{d+1}}} \sum_{z \in \mathbb{F}_q^{d+1}} e(c \cdot z)|z\rangle. \tag{5}$$

Instead we use k quantum queries to create the approximate state

$$|\hat{c}_{R_k}\rangle := \frac{1}{\sqrt{|R_k|}} \sum_{z \in R_k} e(c \cdot z)|z\rangle \tag{6}$$

for some set $R_k \subseteq \mathbb{F}_q^{d+1}$. A measurement of this state in the Fourier basis gives c with probability $|\langle \hat{c}_{R_k}|\hat{c}\rangle|^2 = |R_k|/q^{d+1}$.

Our algorithm performs k phase queries in parallel, each acting on a separate register. On input $|x, y\rangle$ for $x, y \in \mathbb{F}_q^k$, these k queries introduce the phase $e(\sum_{i=1}^k y_i f(x_i))$. To define the set R_k, recall the function $Z\colon \mathbb{F}_q^k \times \mathbb{F}_q^k \to \mathbb{F}_q^{d+1}$ defined by

$$Z(x, y)_j := \sum_{i=1}^k y_i x_i^j \text{ for } j \in \{0, 1, \ldots, d\}. \tag{7}$$

Then we have $\sum_{i=1}^k y_i f(x_i) = \sum_{i=1}^k \sum_{j=0}^d y_i c_j x_i^j = c \cdot Z(x,y)$ for all $x, y \in \mathbb{F}_q^k$. The range $R_k := Z(\mathbb{F}_q^k \times \mathbb{F}_q^k)$ of the function Z is the set

$$R_k = \{Z(x,y) : (x,y) \in \mathbb{F}_q^k \times \mathbb{F}_q^k\} \subseteq \mathbb{F}_q^{d+1}. \tag{8}$$

For each $z \in R_k$ we choose a unique $(x,y) \in \mathbb{F}_q^k \times \mathbb{F}_q^k$ such that $Z(x,y) = z$. Let $T_k \subseteq \mathbb{F}_q^k \times \mathbb{F}_q^k$ be the set of these representatives. Clearly, $Z\colon T_k \to R_k$ is a bijection.

To create the state $|\hat{c}_{R_k}\rangle$, we prepare a uniform superposition over T_k, perform k phase queries, and compute Z in place (i.e., perform the unitary transformation $|x,y\rangle \mapsto |Z(x,y)\rangle$), giving

$$\frac{1}{\sqrt{|T_k|}} \sum_{(x,y)\in T_k} |x,y\rangle \mapsto \frac{1}{\sqrt{|T_k|}} \sum_{(x,y)\in T_k} e(c \cdot Z(x,y))|x,y\rangle \tag{9}$$

$$\mapsto \frac{1}{\sqrt{|R_k|}} \sum_{z\in R_k} e(c \cdot z)|z\rangle. \tag{10}$$

The above procedure is a k-query algorithm for polynomial interpolation that succeeds with probability $|R_k|/q^{d+1}$, establishing the lower bound on the success probability stated in Theorem 1. To analyze the algorithm, it remains to lower bound $|R_k|$ as a function of k.

2.3. **Performance using $d/2 + 1/2$ queries.** We now consider the performance of the above algorithm using $k = d/2 + 1/2$ queries. Let

$$Z^{-1}(z) = \{(x,y) \in \mathbb{F}_q^k \times \mathbb{F}_q^k : Z(x,y) = z\} \tag{11}$$

be the set of those $(x,y) \in \mathbb{F}_q^k \times \mathbb{F}_q^k$ corresponding to a particular $z \in \mathbb{F}_q^{d+1}$. Clearly $|R_k|$ is the number of values of z such that $Z^{-1}(z)$ is nonempty. To analyze this, we focus on "good" values of (x,y). Define $X_k^{\text{good}} := \{x \in \mathbb{F}_q^k : x_i \neq x_j \ \forall\, i \neq j\}$ and $Y_k^{\text{good}} := (\mathbb{F}_q^\times)^k$ and let $Z^{-1}(z)^{\text{good}} := Z^{-1}(z) \cap (X_k^{\text{good}} \times Y_k^{\text{good}})$. We claim the following:

Lemma 1. *If $k = d/2 + 1/2$, then for all $z \in \mathbb{F}_q^{d+1}$, either $|Z^{-1}(z)^{\text{good}}| = 0$ or $|Z^{-1}(z)^{\text{good}}| = k!$.*

Proof. We can write the condition $Z(x,y) = z$ in the form $\sum_i y_i \boldsymbol{x}_i = z$, where $\boldsymbol{x}_i := (1, x_i, x_i^2, \ldots, x_i^d)$. We claim that for a given $z \in \mathbb{F}_q^{d+1}$, the values $(x,y) \in X^{\text{good}} \times Y^{\text{good}}$ that satisfy this equation are unique up to a permutation of the indices. To see this, suppose that $Z(x,y) = Z(u,v)$ for some good values $(x,y) \neq (u,v)$. By permuting the indices, we can ensure that $x_i = u_i$ for $i \in \{1, \ldots, m\}$ and $x_i \neq u_i$ for $i \in \{m+1, \ldots, k\}$, where m is the number of positions at which x and u agree. Then we have

$$\sum_{i=1}^m (y_i - v_i)\boldsymbol{x}_i + \sum_{i=m+1}^k y_i \boldsymbol{x}_i + \sum_{i=m+1}^k v_i \boldsymbol{u}_i = 0. \tag{12}$$

It is well known that the Vandermonde matrix

$$\begin{pmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_{d+1} \\ x_1^2 & x_2^2 & \cdots & x_{d+1}^2 \\ \vdots & \vdots & & \vdots \\ x_1^d & x_2^d & \cdots & x_{d+1}^d \end{pmatrix} \tag{13}$$

is invertible provided the values $x_1, x_2, \ldots, x_{d+1}$ are distinct. Because the values x_i for $i \in \{1, \ldots, k\}$ and u_i for $i \in \{m+1, \ldots, k\}$ are all distinct, and because the number of terms in (12) is at most $2k \leq d+1$, the vectors $\boldsymbol{x}_i$ for $i \in \{1, \ldots, k\}$ and $\boldsymbol{u}_i$ for $i \in \{m+1, \ldots, k\}$ are linearly independent. Thus we have $y_i = v_i$ for all $i \in \{1, \ldots, m\}$ and $y_i = v_i = 0$ for all $i \in \{m+1, \ldots, k\}$. Since $y \in Y^{\text{good}}$, we cannot have $y_i = 0$ for any i, so we must have $m = k$. Therefore $x = u$ and $y = v$.

It follows that the only way to obtain a distinct (x,y) is to permute the indices, and therefore we either have $|Z^{-1}(z)^{\text{good}}|=0$ (if there is no $(x,y)\in X^{\text{good}}\times Y^{\text{good}}$ such that $Z(x,y)=z$) or $|Z^{-1}(z)^{\text{good}}|=k!$. □

Using Lemma 1, we can show that $k=d/2+1/2$ queries suffice to perform polynomial interpolation with probability that is independent of q, but that decreases with d.

Proof of Theorem 2(i): $k=d/2+1/2$. We have $|X_k^{\text{good}}|=q!/(q-k)!$ and $|Y_k^{\text{good}}|=(q-1)^k$, so

$$\sum_{z\in\mathbb{F}_q^{d+1}}|Z^{-1}(z)^{\text{good}}|=|X_k^{\text{good}}|\cdot|Y_k^{\text{good}}|=\frac{q!}{(q-k)!}(q-1)^k=q^{2k}(1-O(1/q)). \tag{14}$$

Thus, invoking Lemma 1, the number of values of z for which $|Z^{-1}(z)^{\text{good}}|=k!$ is at least $\frac{q^{2k}}{k!}(1-O(1/q))$. Since $k=d/2+1/2$, it follows that $|R_k|/q^{d+1}$ is at least $\frac{1}{k!}(1-O(1/q))$, as claimed. □

2.4. **Performance using $d/2+1$ queries.** Next we show that with more than $d/2+1/2$ queries, the success probability approaches 1 for large q.

Proof of Theorem 2(ii): $k=d/2+1$. Under the uniform distribution on $z\in\mathbb{F}_q^{d+1}$, we have

$$|R_k|/q^{d+1}=1-\Pr[|Z^{-1}(z)|=0]. \tag{15}$$

We use a second moment argument to upper bound the number of $z\in\mathbb{F}_q^{d+1}$ for which $|Z^{-1}(z)|=0$. The mean of $|Z^{-1}(z)|$ is

$$\mu:=\frac{1}{q^{d+1}}\sum_{z\in\mathbb{F}_q^{d+1}}|Z^{-1}(z)|=q^{2k-(d+1)}. \tag{16}$$

Let $\delta[\mathcal{P}]$ be 1 if $\mathcal{P}$ is true and 0 if $\mathcal{P}$ is false. For the second moment, we compute

$$\begin{align}
\sum_{z\in\mathbb{F}_q^{d+1}}|Z^{-1}(z)|^2 &= \sum_{u,v,x,y\in\mathbb{F}_q^k}\delta[Z(u,v)=Z(x,y)] \tag{17}\\
&= \sum_{u,v,x,y\in\mathbb{F}_q^k}\frac{1}{q^{d+1}}\sum_{\lambda\in\mathbb{F}_q^{d+1}}e(\lambda\cdot(Z(u,v)-Z(x,y))) \tag{18}\\
&= \frac{q^{4k}}{q^{d+1}}+\frac{1}{q^{d+1}}\sum_{\lambda\in\mathbb{F}_q^{d+1}\setminus(0,\ldots,0)}\left(\sum_{x,y\in\mathbb{F}_q}e\left(y\sum_{j=0}^{d}\lambda_j x^j\right)\right)^{2k} \tag{19}\\
&= q^{4k-(d+1)}+\frac{1}{q^{d+1}}\sum_{\lambda\in\mathbb{F}_q^{d+1}\setminus(0,\ldots,0)}\left(q\sum_{x\in\mathbb{F}_q}\delta\left[\sum_{j=0}^{d}\lambda_j x^j=0\right]\right)^{2k} \tag{20}\\
&\le q^{4k-(d+1)}+(qd)^{2k}. \tag{21}
\end{align}$$

Thus for the variance, we have

$$\sigma^2:=\frac{1}{q^{d+1}}\sum_{z\in\mathbb{F}_q^{d+1}}|Z^{-1}(z)|^2-\mu^2\le\frac{(qd)^{2k}}{q^{d+1}}. \tag{22}$$

(note that $\sigma^2\ge 0$ by the Cauchy inequality). Applying the Chebyshev inequality, we find

$$\Pr[Z^{-1}(z)=0]\le\frac{\sigma^2}{\mu^2}\le\frac{(qd)^{2k}/q^{d+1}}{q^{4k-2(d+1)}}=d^{2k}q^{d+1-2k}. \tag{23}$$

Therefore $|R_k|/q^{d+1} = 1 - \Pr[Z^{-1}(z) = 0] \geq 1 - d^{2k}q^{d+1-2k}$. With $k = d/2+1$, we have

$$|R_k|/q^{d+1} \geq 1 - d^{2k}/q = 1 - O(1/q) \tag{24}$$

as claimed. □

Note that one can improve the dependence on d in (22) using results on the distribution of zeros in random polynomials [11].

2.5. **An alternative algorithm.** The algorithm described above queries the oracle nonadaptively, that is, all k queries can be performed in parallel. However, the input state to these queries is correlated across all k copies. In this section, we describe an alternative algorithm that queries the black box on a state that is independent and identical for each of the k queries, namely, a uniform superposition over all inputs. This algorithm is suboptimal, but its performance is not significantly worse than that of the optimal algorithm described in Section 2.2.

Analogous to the so-called standard method for the hidden subgroup problem, querying f on a uniform superposition gives the state $\frac{1}{\sqrt{q}}\sum_{x\in\mathbb{F}_q^k}|x, f(x)\rangle$. If we use k queries to prepare k copies of this state and then perform the Fourier transform on the second register (or equivalently, perform k independent phase queries), we obtain the state

$$\frac{1}{q^k}\sum_{x,y\in\mathbb{F}_q^k} e(c\cdot Z(x,y))|x,y\rangle = \frac{1}{q^k}\sum_{z\in\mathbb{F}_q^{d+1}} e(c\cdot z)\sqrt{|Z^{-1}(z)|}\,|Z^{-1}(z)\rangle \tag{25}$$

where $|Z^{-1}(z)\rangle := \sum_{(x,y)\in Z^{-1}(z)}|x,y\rangle/|Z^{-1}(z)|^{1/2}$. Motivated by the PGM approach to the hidden subgroup problem [1], suppose we perform the transformation $|Z^{-1}(z)\rangle \mapsto |z\rangle$, giving the state

$$|\phi_k^c\rangle := \frac{1}{q^k}\sum_{z\in\mathbb{F}_q^{d+1}} e(c\cdot z)\sqrt{|Z^{-1}(z)|}\,|z\rangle. \tag{26}$$

Measuring this state in the Fourier basis gives the outcome c with probability

$$|\langle\phi_k^c|\hat{c}\rangle|^2 = \frac{1}{q^{2k+d+1}}\left(\sum_{z\in\mathbb{F}_q^{d+1}}\sqrt{|Z^{-1}(z)|}\right)^2. \tag{27}$$

If $k = d/2+1/2$, we claim that this algorithm succeeds with constant probability. From the proof of Theorem 2 for $k = d/2+1/2$, we have that $|Z^{-1}(z)| \geq k!$ for at least $\frac{q^{2k}}{k!}(1-O(1/q))$ values of z. Therefore the success probability is at least $\frac{1}{k!}(1-O(1/q))$.

If $k = d/2+1$, then this algorithm succeeds with probability that approaches 1 for large q. To see this, recall from the proof of Theorem 2 for $k = d/2+1$ that, under a uniform distribution over $z \in \mathbb{F}_q^{d+1}$, the quantity $Z^{-1}(z)$ has mean $\mu = q$ and standard deviation $\sigma = \sqrt{q}d^k$. Thus, by the Chebyshev inequality, we have

$$\Pr\bigl[|Z^{-1}(z)| \leq q - \alpha\sqrt{q}d^k\bigr] \leq \frac{1}{\alpha^2}. \tag{28}$$

It follows that

$$|\langle\phi_k^c|\hat{c}\rangle|^2 \geq \left(1 - \frac{\alpha d^k}{\sqrt{q}}\right)\left(1 - \frac{1}{\alpha^2}\right)^2. \tag{29}$$

Choosing $\alpha = \Theta(q^{1/6})$, this gives a success probability of $|\langle\phi_k^c|\hat{c}\rangle|^2 = 1 - O(q^{-1/3})$, which approaches 1 for large q.

3. Optimality

In this section, we show that the query complexity of our algorithm is precisely optimal: no k-query algorithm can succeed with a probability larger than $|R_k|/q^{d+1}$. We begin with a basic result showing that m states spanning an n-dimensional subspace can be distinguished with probability at most n/m.

Lemma 2. *Suppose we are given a state $|\psi_c\rangle$ with $c \in C$ chosen uniformly at random. Then the probability of correctly determining c with some orthogonal measurement is at most* $\dim\operatorname{span}\{|\psi_c\rangle : c \in C\}/|C|$.

Proof. Consider a measurement with orthogonal projectors E_c, and let Π denote the projection onto $\operatorname{span}\{|\psi_c\rangle : c \in C\}$. Then we have

$$\Pr[\text{success}] = \frac{1}{|C|}\sum_{c\in C}\langle\psi_c|E_c|\psi_c\rangle \le \frac{1}{|C|}\sum_{c\in C}\operatorname{tr}(E_c\Pi) = \frac{\operatorname{tr}(\Pi)}{|C|} = \frac{\dim\operatorname{span}\{|\psi_c\rangle : c\in C\}}{|C|} \tag{30}$$

as claimed. □

We apply this lemma where $|\psi_c\rangle$ is the final state of a given quantum query algorithm when the black box contains $c \in \mathbb{F}_q^{d+1}$. There is no loss of generality in considering an orthogonal measurement at the end of the algorithm since we allow the use of an arbitrary-sized ancilla.

Lemma 3. *Let $|\psi_c\rangle$ be the state of any quantum polynomial interpolation algorithm after k queries, where the black box contains $c \in \mathbb{F}_q^{d+1}$. Then* $\dim\operatorname{span}\{|\psi_c\rangle : c \in \mathbb{F}_q^{d+1}\} \le |R_k|$.

Proof. We claim that

$$|\psi_c\rangle = \sum_{x,y\in\mathbb{F}_q^k} e(Z(x,y)\cdot c)|\phi_{x,y}\rangle \tag{31}$$

for some set of (unnormalized) states $\{|\phi_{x,y}\rangle : x,y \in \mathbb{F}_q^k\}$ that do not depend on c. Then the result follows, since

$$|\psi_c\rangle = \sum_{z\in\mathbb{F}_q^{d+1}} e(z\cdot c)\sum_{x,y\in Z^{-1}(z)}|\phi_{x,y}\rangle \in \operatorname{span}\Bigg\{\sum_{x,y\in Z^{-1}(z)}|\phi_{x,y}\rangle : z\in\mathbb{F}_q^{d+1}\Bigg\}, \tag{32}$$

which has dimension at most $|R_k| = |\{Z(x,y) : (x,y) \in \mathbb{F}_q^k \times \mathbb{F}_q^k\}|$.

To see the claim, consider a general k-query algorithm $U_kQ_cU_{k-1}\dots Q_cU_1Q_cU_0$ acting on states of the form $|x,y,w\rangle$ for an arbitrary-sized workspace register $|w\rangle$, starting in the state $|x_0,y_0,w_0\rangle = |0,0,0\rangle$. Here $Q_c\colon |x,y,w\rangle \mapsto e(yf(x))|x,y,w\rangle$ is a phase query. The final state $|\psi_c\rangle$ equals

$$\sum_{\substack{x,y\in\mathbb{F}_q^k\\ x_{k+1},y_{k+1}\in\mathbb{F}_q\\ w\in I^{k+1}}} e\Bigg(\sum_{j=1}^k y_jf(x_j)\Bigg)\Bigg(\prod_{j=0}^k\langle x_{j+1},y_{j+1},w_{j+1}|U_j|x_j,y_j,w_j\rangle\Bigg)|x_{k+1},y_{k+1},w_{k+1}\rangle, \tag{33}$$

with $x_0 = y_0 = w_0 = 0$, $x = (x_1,\dots,x_k)$, $y = (y_1,\dots,y_k)$, $w = (w_1,\dots,w_{k+1})$, and I some appropriate index set. This expression has the claimed form when we define

$$|\phi_{x,y}\rangle = \sum_{\substack{x_{k+1},y_{k+1}\in\mathbb{F}_q\\ w\in I^{k+1}}}\Bigg(\prod_{j=0}^k\langle x_{j+1},y_{j+1},w_{j+1}|U_j|x_j,y_j,w_j\rangle\Bigg)|x_{k+1},y_{k+1},w_{k+1}\rangle.$$ □

We can now prove our upper bound on the success probability of quantum algorithms for polynomial interpolation.

Proof of Theorem 1 (upper bound on success probability). By combining Lemma 2 with Lemma 3, we see that if the coefficients $c \in \mathbb{F}_q^{d+1}$ are chosen uniformly at random, no algorithm can succeed with probability greater than $|R_k|/q^{d+1}$. Since the minimum cannot be larger than the average, this implies a lower bound on the success probability in the worst case of $|R_k|/q^{d+1}$. □

This result also shows that the exact quantum query complexity of polynomial interpolation is maximal.

Corollary 1. *The exact quantum query complexity of interpolating a degree-d polynomial is $d+1$.*

Proof. This follows from Theorem 1 and the fact that if $k < d+1$, we have $|R_k| < q^{d+1}$. To see this, observe that if $k < d+1$, then vectors of the form $(0, \dots, 0, z_d)$ for $z_d \neq 0$ are not in the range of Z. We can assume there is an $(x, y) \in Z^{-1}(z)$ with $x_1, \dots, x_k$ all distinct, since if $x_i = x_j$ for some $i \neq j$, then we could delete index j and replace y_i by $y_i + y_j$. Then in equation (52), the Vandermonde matrix on the left-hand side is invertible, so $y_1 = \cdots = y_k = 0$. However, this implies that $\sum_i y_i x_i^d = 0 \neq z_d$. □

4. Gate complexity

In Section 2, we analyzed the query complexity of our polynomial interpolation algorithm. Here we describe a $(d/2+1/2)$-query algorithm whose gate complexity is $\mathrm{poly}(\log q)$, and whose success probability is close to that of the best algorithm using this number of queries (in particular, for fixed d it still succeeds with constant probability). We also give an algorithm for the case $k = d/2+1$ whose gate complexity is larger by a factor of $\mathrm{poly}(\log q)$, but with an additional factor of $k!$.

4.1. Algorithm for $k = d/2+1/2$ queries.

To simplify the computation of unique representatives of values $z \in R_k$, we restrict attention to the "good" case considered in Section 2.3. Let

$$R_k^{\text{good}} := \{Z(x, y) : x \in X_k^{\text{good}},\, y \in Y_k^{\text{good}}\}. \tag{34}$$

For any $z \in R_k^{\text{good}}$, we show how to efficiently compute representative values $x \in X_k^{\text{good}}$ and $y \in Y_k^{\text{good}}$ with $Z(x, y) = z$, defining a set of representatives T_k^{good}. Then we consider an algorithm as described in Section 2.2, but with R_k replaced by R_k^{good} and T_k replaced by T_k^{good}. Clearly the success probability of this algorithm is $|R_k^{\text{good}}|/q^{d+1}$. Our lower bound on $|R_k|$ in Section 2.3 was actually a bound on $|R_k^{\text{good}}|$, so this algorithm still succeeds with probability $\frac{1}{k!}(1 + O(1/q))$.

To give a gate-efficient algorithm, it suffices to show how to efficiently compute the function $Z^{-1} \colon R_k^{\text{good}} \to T_k^{\text{good}}$ (that is, to compute this function using $\mathrm{poly}(\log q)$ gates).

Lemma 4. *Suppose there is an efficient algorithm to compute $Z^{-1} \colon R_k^{\text{good}} \to T_k^{\text{good}}$. Then the algorithm of Section 2.2 can be made gate-efficient (with R_k replaced by R_k^{good} and T_k by T_k^{good}).*

Proof. It is trivial to compute $Z \colon T_k^{\text{good}} \to R_k^{\text{good}}$ efficiently. Given an efficient procedure for computing $Z^{-1} \colon R_k^{\text{good}} \to T_k^{\text{good}}$, this gives us the ability to efficiently compute Z in place (that is, to perform the transformation $|x, y\rangle \mapsto |z\rangle$ as required by the algorithm). To do this, we first compute z in an ancilla register by evaluating Z (which only requires arithmetic over $\mathbb{F}_q$) and then uncompute (x, y) by applying the circuit for Z^{-1} in reverse.

It remains to prepare the initial uniform superposition over T_k^{good}. This can also be done using the ability to compute Z^{-1}. Suppose we create a uniform superposition over all of $z \in \mathbb{F}_q^{d+1}$ and then attempt to compute Z^{-1}. If $z \notin R_k^{\text{good}}$, this is detected, and we can set a flag qubit indicating failure. Thus we can prepare a state of the form

$$\frac{1}{\sqrt{q^{d+1}}}\Bigg(\sum_{(x,y)\in T_k^{\text{good}}} |Z(x, y), 0, x, y\rangle + \sum_{z\in \mathbb{F}_q^{d+1}\setminus R_k^{\text{good}}} |z, 1, 0, 0\rangle\Bigg). \tag{35}$$

A measurement of the flag qubit gives the outcome 0 with probability $|R_k^{\text{good}}|/q^{d+1}$. Since this is our lower bound on the success probability of the overall algorithm, we do not have to repeat this process too many times before we successfully prepare the initial state (and by sufficiently many repetitions, we can make the error probability arbitrarily small). When the measurement succeeds, we can uncompute the first register to obtain the state $\sum_{(x,y)\in T_k^{\text{good}}}|x,y\rangle/|T_k^{\text{good}}|^{1/2}$ as desired. □

In the remainder of this section, we describe how to efficiently compute $Z^{-1}(z)$ for $z \in R_k^{\text{good}}$. Our approach appeals to "Prony's method" [15] (a precursor to Fourier analysis) and the theory of linear recurrences. We start with the following technical result, where e_j denotes the jth elementary symmetric polynomial in k variables, i.e.,

$$e_j(x_1,\dots,x_k) = \sum_{1\le i_1<i_2<\cdots<i_j\le k} x_{i_1}x_{i_2}\cdots x_{i_j}. \tag{36}$$

Lemma 5. *We have*

$$x_i^k = -\sum_{j=1}^{k} x_i^{k-j}(-1)^j e_j(x_1,\dots,x_k) \tag{37}$$

for all $i \in \{1,\dots,k\}$.

Proof. Observe that it suffices to prove the lemma for $i = 1$, since if we interchange the roles of x_1 and x_ℓ in (37) with $i = 1$, we obtain (37) with $i = \ell$.

We apply induction on k. If $k = 1$ then the claim is trivial. Now suppose the claim holds for a given value of k. We have

$$e_j(x_1,\dots,x_{k+1}) = e_j(x_1,\dots,x_k) + x_{k+1}e_{j-1}(x_1,\dots,x_k) \tag{38}$$

$$= e_j(x_1,\dots,x_k) + x_{k+1}\tfrac{\partial}{\partial x_{k+1}}e_j(x_1,\dots,x_{k+1}). \tag{39}$$

Therefore

$$-\sum_{j=1}^{k+1} x_1^{k+1-j}(-1)^j e_j(x_1,\dots,x_{k+1}) = -\sum_{j=1}^{k+1} x_1^{k+1-j}(-1)^j \big[e_j(x_1,\dots,x_k) + x_{k+1}\tfrac{\partial}{\partial x_{k+1}}e_j(x_1,\dots,x_{k+1})\big] \tag{40}$$

$$= x_1^{k+1} - x_{k+1}\tfrac{\partial}{\partial x_{k+1}}x_i^{k+1} \tag{41}$$

$$= x_1^{k+1} \tag{42}$$

(where the second equality uses the induction hypothesis). □

Using this fact, we can show that each component of $Z(x,y)$ satisfies a kth-order linear recurrence.

Lemma 6. *If $z_j = \sum_{i=1}^k y_i x_i^j$ for all nonnegative integers j, then we have (for all nonnegative integers n)*

$$z_{n+k} = -\sum_{j=0}^{k-1}(-1)^{k-j}e_{k-j}(x_1,\dots,x_k)z_{n+j}. \tag{43}$$

Proof. The right-hand side of (43) is

$$-\sum_{j=0}^{k-1}(-1)^{k-j}e_{k-j}(x_1,\ldots,x_k)\sum_{i=1}^{k}y_ix_i^{n+j} = -\sum_{i=1}^{k}y_i\sum_{j=0}^{k-1}(-1)^{k-j}e_{k-j}(x_1,\ldots,x_k)x_i^{n+j} \tag{44}$$

$$= -\sum_{i=1}^{k}y_i\sum_{j=1}^{k}(-1)^{j}e_{j}(x_1,\ldots,x_k)x_i^{n+k-j} \tag{45}$$

$$= \sum_{i=1}^{k}y_ix_i^{n+k} = z_{n+k} \tag{46}$$

as claimed, where the third equality uses Lemma 5. □

We are now ready to describe the gate-efficient algorithm for polynomial interpolation.

Proof of Theorem 3(i): $k = d/2 + 1/2$. By Lemma 4, it suffices to give an efficient algorithm for computing a representative $(x, y) \in Z^{-1}(z)^{\text{good}}$ for any given $z \in R_k^{\text{good}}$.

By Lemma 6, the coefficients $a_j = -(-1)^{k-j}e_{k-j}(x_1,\ldots,x_k)$ of the linear recurrence (43) satisfy

$$H_k\begin{pmatrix}a_0\\a_1\\\vdots\\a_{k-1}\end{pmatrix} = \begin{pmatrix}z_k\\z_{k+1}\\\vdots\\z_{2k-1}\end{pmatrix}, \quad \text{where} \quad H_k := \begin{pmatrix}z_0 & z_1 & \cdots & z_{k-1}\\ z_1 & z_2 & \cdots & z_k\\ \vdots & \vdots & \ddots & \vdots\\ z_{k-1} & z_k & \cdots & z_{2(k-1)}\end{pmatrix} \tag{47}$$

is a Hankel matrix. Observe that

$$H_k = V_k^T\begin{pmatrix}y_1 & 0 & \cdots & 0\\ 0 & y_2 & & 0\\ \vdots & & \ddots & \vdots\\ 0 & 0 & \cdots & y_k\end{pmatrix}V_k, \quad \text{where} \quad V_k := \begin{pmatrix}1 & x_1 & x_1^2 & \cdots & x_1^{k-1}\\ 1 & x_2 & x_2^2 & \cdots & x_2^{k-1}\\ \vdots & \vdots & \vdots & & \vdots\\ 1 & x_k & x_k^2 & \cdots & x_k^{k-1}\end{pmatrix}. \tag{48}$$

For $x \in X_k^{\text{good}}$, the Vandermonde matrix V_k (and its transpose) are invertible, and for $y \in Y_k^{\text{good}}$, the diagonal matrix is invertible. Then H_k is invertible, and we have

$$\begin{pmatrix}a_0\\a_1\\\vdots\\a_{k-1}\end{pmatrix} = H_k^{-1}\begin{pmatrix}z_k\\z_{k+1}\\\vdots\\z_{2k-1}\end{pmatrix}. \tag{49}$$

We claim that for any $(x, y) \in Z^{-1}(z)^{\text{good}}$, the values $x_1,\ldots,x_k$ must be roots of the characteristic polynomial

$$\chi(x) := x^k - \sum_{j=0}^{k-1}a_jx^j. \tag{50}$$

To see this, observe that

$$\begin{pmatrix}z_k\\z_{k+1}\\\vdots\\z_{2k-1}\end{pmatrix} - H_k\begin{pmatrix}a_0\\a_1\\\vdots\\a_{k-1}\end{pmatrix} = V_k^T\begin{pmatrix}\chi(x_1)\\\chi(x_2)\\\vdots\\\chi(x_k)\end{pmatrix}. \tag{51}$$

This must be the zero vector, and since the Vandermonde matrix is invertible, we see that $\chi(x_i) = 0$ for all $i \in \{1, \ldots, k\}$.[4]

Finally, observe that the values $y_1, \ldots, y_k$ satisfy

$$V_k^T \begin{pmatrix} y_1 \\ \vdots \\ y_k \end{pmatrix} = \begin{pmatrix} z_0 \\ z_1 \\ \vdots \\ z_{k-1} \end{pmatrix}. \tag{52}$$

Since the Vandermonde matrix is invertible, we see that y is uniquely determined by z and x.

To compute a unique representative of a given $z \in R_k^{\mathrm{good}}$, we use equation (49) to efficiently compute the coefficients $a_0, \ldots, a_{k-1}$ of the characteristic polynomial $\chi(x)$. We can then determine $x \in X_k^{\mathrm{good}}$ by finding the roots of this polynomial, which can be done in time $\mathrm{poly}(k, \log q)$ using a randomized algorithm [9, Chapter 14]. Finally, we can determine $y \in Y_k^{\mathrm{good}}$ by solving a linear system of equations, namely (52).

This procedure does not uniquely specify (x, y) because any permutation of the indices (acting identically on x and y) gives an equivalent solution. To choose a unique $(x, y) \in T_k^{\mathrm{good}}$, we simply require that the entries of x occur in lexicographic order with respect to some fixed representation of $\mathbb{F}_q$. □

4.2. Algorithm for $k = d/2 + 1$ queries.

We now present a similar algorithm for the case $k = d/2 + 1$ that also has gate complexity $\mathrm{poly}(\log q)$, although it has more overhead as a function of d.

To apply the approach of Section 4.1, we again focus on solutions of $Z(x, y) = z$ with $(x, y) \in X^{\mathrm{good}} \times Y^{\mathrm{good}}$. However, recall that our lower bound on the success probability for $k = d/2 + 1$ in Section 2.4 used all solutions $(x, y) \in \mathbb{F}_q^k \times \mathbb{F}_q^k$. Thus we begin by showing that the success probability of the algorithm remains close to 1 even when restricted to good solutions.

Lemma 7. *If $k = d/2 + 1$, then $|R_k^{\mathrm{good}}|/q^{d+1} = 1 - O(1/q)$.*

Proof. We repeat the second moment argument of Section 2.4, but now restricted to good solutions. Under the uniform distribution on $z \in \mathbb{F}_q^{d+1}$, we have

$$\mu^{\mathrm{good}} := \frac{1}{q^{d+1}} \sum_{z \in \mathbb{F}_q^{d+1}} |Z^{-1}(z)^{\mathrm{good}}| = q^{2k-(d+1)}(1 - O(1/q)) \tag{53}$$

by (14). Similarly to the previous second moment calculation, we have

$$\sum_{z \in \mathbb{F}_q^{d+1}} |Z^{-1}(z)^{\mathrm{good}}|^2 = \sum_{u,x \in X_k^{\mathrm{good}}} \sum_{v,y \in Y_k^{\mathrm{good}}} \delta[Z(u, v) = Z(x, y)] \tag{54}$$

$$= q^{d+1}(\mu^{\mathrm{good}})^2 + \frac{1}{q^{d+1}} \sum_{u,x \in X_k^{\mathrm{good}}} \sum_{v,y \in Y_k^{\mathrm{good}}} \sum_{\lambda \in \mathbb{F}_q^{d+1}} e(\lambda \cdot (Z(u, v) - Z(x, y))). \tag{55}$$

[4] Since a polynomial of degree k can have at most k roots, this shows that the values $x_1, \ldots, x_k$ are unique up to permutation, giving $|Z^{-1}(z)^{\mathrm{good}}| = k!$ as shown in Lemma 1.

Thus we have

$$(\sigma^{\text{good}})^2 := \frac{1}{q^{d+1}} \sum_{z\in\mathbb{F}_q^{d+1}} |Z^{-1}(z)^{\text{good}}|^2 - (\mu^{\text{good}})^2 \tag{56}$$

$$= \frac{1}{q^{2(d+1)}} \sum_{\lambda\in\mathbb{F}_q^{d+1}} \sum_{u,x\in X_k^{\text{good}}} \sum_{v,y\in Y_k^{\text{good}}} \prod_{i=1}^{k} e(v_i \textstyle\sum_{j=0}^d \lambda_j u_i^j)\, e(-y_i \sum_{j=0}^d \lambda_j x_i^j) \tag{57}$$

$$= \frac{1}{q^{2(d+1)}} \sum_{\lambda\in\mathbb{F}_q^{d+1}} \sum_{u,x\in X_k^{\text{good}}} \prod_{i=1}^{k} (q\,\delta[\textstyle\sum_{j=0}^d \lambda_j u_i^j = 0] - 1)(q\,\delta[\sum_{j=0}^d \lambda_j x_i^j = 0] - 1) \tag{58}$$

$$\le \frac{1}{q^{2(d+1)}} \sum_{\lambda\in\mathbb{F}_q^{d+1}} \sum_{u,x\in \mathbb{F}_q^k} \prod_{i=1}^{k} (q\,\delta[\textstyle\sum_{j=0}^d \lambda_j u_i^j = 0] + 1)(q\,\delta[\sum_{j=0}^d \lambda_j x_i^j = 0] + 1) \tag{59}$$

$$\le \frac{(q(d+1))^{2k}}{q^{d+1}} \tag{60}$$

(which is identical to the previous bound for σ^2 except that d is replaced by $d+1$). Therefore, by the Chebyshev inequality, we have

$$\Pr[Z^{-1}(z)^{\text{good}}] \le \frac{(\sigma^{\text{good}})^2}{(\mu^{\text{good}})^2} \le (d+1)^{2k} q^{d+1-2k}. \tag{61}$$

With $k = d/2+1$, we find

$$\frac{|R_k^{\text{good}}|}{q^{d+1}} = 1 - \Pr[Z^{-1}(z)^{\text{good}} = 0] \ge 1 - \frac{(d+1)^{2k}}{q} = 1 - O(1/q) \tag{62}$$

as claimed. □

Now consider the problem of computing a value $(x,y) \in X^{\text{good}} \times Y^{\text{good}}$ such that $Z(x,y) = z$ for some given $z \in R_k^{\text{good}}$. We can approach this task using the strategy outlined in Section 4.1. With $k = d/2+1$, we have $2k-1 = d+1$, so the last entry in the vector on the right-hand side of (49) is not specified. Nevertheless, for any fixed $(x,y) \in \mathbb{F}_q^k \times \mathbb{F}_q^k$, the value $z_{d+1} = Z(x,y)_{d+1}$ is well-defined by extending (7) to $j = d+1$, so we can find $(x,y) \in X^{\text{good}} \times Y^{\text{good}}$ by searching for some value of $z_{d+1} \in \mathbb{F}_q$ for which the algorithm of Section 4.1 succeeds at finding k distinct roots $x_1, \ldots, x_k \in \mathbb{F}_q$ of the characteristic polynomial (50).

We claim that choosing a random $z_{d+1} \in \mathbb{F}_q$ gives a solution with probability nearly $1/k!$.

Lemma 8. *Suppose $z = (z_0, \ldots, z_d)$ is chosen uniformly at random from $\mathbb{F}_q^{d+1}$. Then with probability $1 - o(1)$ (over the choice of z), choosing z_{d+1} uniformly at random from $\mathbb{F}_q$ and solving for $(x,y) \in Z^{-1}(z)^{\text{good}}$ as in the proof of Theorem 3(ii) gives a solution with probability $(1-o(1))/k!$ (over the choice of z_{d+1}).*

Proof. For any $z \in R_k^{\text{good}}$, each value of $z_{d+1} \in \mathbb{F}_q$ gives a unique set of roots of the characteristic polynomial (50), and hence corresponds to either 0 or $k!$ solutions $(x,y) \in Z^{-1}(z)^{\text{good}}$. By a similar second moment argument as in (28), but using the mean (53) and variance (60) of $Z^{-1}(z)^{\text{good}}$, we have $|Z^{-1}(z)^{\text{good}}| = q(1-o(1))$ with probability $1-o(1)$ over the uniform choice of $z \in \mathbb{F}_q^{d+1}$. Thus the number of values of z_{d+1} that lead to a valid solution $(x,y) \in Z^{-1}(z)^{\text{good}}$ is at least $q(1-o(1))/k!$ with probability $1-o(1)$ over the choice of z. Since there are q possible values of z_{d+1}, choosing z_{d+1} at random leads to a valid representative $(x,y) \in Z^{-1}(z)^{\text{good}}$ with probability $(1-o(1))/k!$, again with probability $1-o(1)$ over the uniform choice of z. □

Lemma 8 gives a method for computing a representative $(x, y) \in X^{\text{good}} \times Y^{\text{good}}$ such that $Z(x, y) = z$: simply choose $z_{d+1} \in \mathbb{F}_q$ at random until we find a solution. Repeating this process $O(k!)$ times suffices to find a solution with constant probability (for almost all z). However, since this approach constructs a random $(x, y) \in Z^{-1}(z)^{\text{good}}$ rather than a unique representative, it does not define a set T_k^{good}, and it cannot be directly applied to our quantum algorithm as described so far. Instead, we construct an equivalent algorithm that represents the sets $Z^{-1}(z)^{\text{good}}$ using quantum superpositions.

Lemma 9. *Suppose there is an efficient algorithm to generate the quantum state*

$$|Z^{-1}(z)^{\text{good}}\rangle := \frac{1}{\sqrt{|Z^{-1}(z)^{\text{good}}|}} \sum_{(x,y)\in Z^{-1}(z)^{\text{good}}} |x, y\rangle \tag{63}$$

for any given $z \in R_k^{\text{good}}$. Then there is a gate-efficient k-query quantum algorithm for the polynomial interpolation problem, succeeding with probability $|R_k^{\text{good}}|/q^{d+1}$.

Proof. We essentially replace $(x, y) \in T_k^{\text{good}}$ by $|Z^{-1}(Z(x, y))^{\text{good}}\rangle$ throughout the algorithm. More concretely, we proceed as follows.

Observe that the ability to perform the given state generation map $|z\rangle \mapsto |z\rangle|Z^{-1}(z)^{\text{good}}\rangle$ implies the ability to perform the in-place transformation

$$|z\rangle \mapsto |Z^{-1}(z)^{\text{good}}\rangle. \tag{64}$$

After applying the state generation map, we simply uncompute the map Z to erase the register $|z\rangle$.

The algorithm begins by creating a uniform superposition over all of $z \in \mathbb{F}_q^{d+1}$ and applying the map (64). As in the proof of Lemma 4, we can detect whether $z \notin R_k^{\text{good}}$, and we can postselect on the outcomes for which $z \in R_k^{\text{good}}$ with reasonable overhead, giving the state $\sum_{z\in R_k^{\text{good}}} |Z^{-1}(z)^{\text{good}}\rangle/|R_k^{\text{good}}|^{1/2}$. Then perform k phase queries and apply the inverse of the transformation (64), giving the state

$$\frac{1}{\sqrt{|R_k^{\text{good}}|}} \sum_{z\in R_k^{\text{good}}} e(c \cdot z)|Z^{-1}(z)^{\text{good}}\rangle \mapsto \frac{1}{\sqrt{|R_k^{\text{good}}|}} \sum_{z\in R_k^{\text{good}}} e(c \cdot z)|z\rangle. \tag{65}$$

As discussed in Section 2.2, measuring this state gives c with probability $|R_k^{\text{good}}|/q^{d+1}$. □

Finally, we show how to prepare $|Z^{-1}(z)^{\text{good}}\rangle$ and thereby give a gate-efficient quantum algorithm for polynomial interpolation with $k = d/2 + 1$ queries.

Proof of Theorem 3(ii): $k = d/2 + 1$. We use $|Z^{-1}(z)^{\text{good}}\rangle$ as a quantum representative of the set of solutions $Z^{-1}(z)^{\text{good}}$ as described in Lemma 9. We claim that we can efficiently perform the transformation $|z\rangle \mapsto |Z^{-1}(z)^{\text{good}}\rangle$ for a fraction $1 - o(1)$ of those $z \in R_k^{\text{good}}$, which in turn are a fraction $1 - o(1)$ of all $z \in \mathbb{F}_q^{d+1}$ (by Lemma 7), giving the claimed success probability.

To prepare $|Z^{-1}(z)^{\text{good}}\rangle$, we first prepare a uniform superposition over $z_{d+1} \in \mathbb{F}_q$ and use the procedure of Section 4.1 to compute the corresponding (x, y), if it exists. Lemma 8 shows that a fraction $(1-o(1))/k!$ of the values of z_{d+1} correspond to a valid (x, y), so this process can be boosted to prepare a state close to $|Z^{-1}(z)^{\text{good}}\rangle$ with overhead $O(k!)$ (or with amplitude amplification, $O(\sqrt{k!})$), which in particular is independent of q. We can easily uncompute z_{d+1} given (x, y), giving the desired transformation. □

5. Open problems

In this paper, we have precisely characterized the quantum query complexity of polynomial interpolation. We conclude by briefly discussing some possible directions for future work.

In Section 4, we gave an algorithm for the case $k = d/2+1$ whose gate complexity is larger than its query complexity by a factor of $k!\operatorname{poly}(\log q)$. This gate complexity is polynomial in $\log(q)$ but superexponential in d. Is it possible to give an algorithm with gate complexity only $\operatorname{poly}(d, \log q)$?

A natural extension of our results would be to consider the problem of learning a multivariate polynomial $f \in \mathbb{F}_q[X_1, \ldots, X_n]$ of degree at most d. Montanaro gave asymptotically optimal bounds for this problem assuming f is multilinear [14], but it is also natural to consider the more general case where f is not necessarily multilinear. The quantum algorithm described in Section 2.2 can be extended to the multivariate case in a fairly straightforward manner, and we conjecture that it performs as follows.

Conjecture 4. *For any fixed positive integers d and n, there exists a k-query quantum algorithm for interpolating a degree-d multivariate polynomial in n variables that, as q grows, has success probability $1 - o(1)$ provided $k > \binom{n+d}{d}/(n+1)$.*

Note that classically one needs $\binom{n+d}{d}$ queries to solve the same problem, so our conjecture states that the quantum query complexity is smaller by a factor of $n+1$. We now discuss why computing the success probability of the quantum algorithm appears to be a difficult problem in algebraic geometry.

Let $f \in \mathbb{F}_q[X_1, \ldots, X_n]$ be of degree at most d. For $j \in \mathbb{N}^n$ and $x \in \mathbb{F}_q^n$, we let

$$x^j := \prod_{t=1}^{n} x_t^{j_t}. \tag{66}$$

To define the set of possible polynomials, we use the set of allowed exponents

$$\mathcal{J} := \{j \in \mathbb{N}^n : j_1 + \cdots + j_n \le d\}, \tag{67}$$

with size

$$J := |\mathcal{J}| = \binom{n+d}{d}. \tag{68}$$

We now define the function $Z\colon (\mathbb{F}_q^n)^k \times \mathbb{F}_q^k \to \mathbb{F}_q^J$ by

$$Z(x, y)_j = \sum_{i=1}^{k} y_i x_i^j \tag{69}$$

and consider its range

$$\mathcal{R}_k := Z((\mathbb{F}_q^n)^k \times \mathbb{F}_q^k) \subseteq \mathbb{F}_q^J. \tag{70}$$

A straightforward generalization of the univariate interpolation algorithm described in Section 2.2 gives a multivariate interpolation algorithm with success probability $|\mathcal{R}_k|/q^J$. We expect that this algorithm solves the interpolation problem with probability $1 - o(1)$ using $\lfloor J/(n+1)\rfloor + 1$ queries. This would be implied by the following:

Conjecture 5. *With $J := \binom{n+d}{d}$ and $\mathcal{R}_k$ as in (70), we have $|\mathcal{R}_k| = q^J(1 - o(1))$ provided $k > J/(n+1)$.*

Note that this holds for $n = 1$ (according to Lemma 7) and also for $d = 1$. Unfortunately, the approach via exponential sums used in the proof of Lemma 7 only works if $k > J/2$. Thus, while it gives a tight result for $n = 1$, it appears to be inefficient for $n > 1$.

Another way to approach Conjecture 5 is to consider the affine variety

$$\mathcal{V}_k\colon Z(x, y) = z \tag{71}$$

in $kn + k + J$ variables $x \in (\mathbb{F}_q^n)^k$, $y \in \mathbb{F}_q^k$, $z \in \mathbb{F}_q^J$. Clearly $|\mathcal{V}_k(\mathbb{F}_q)| = q^{kn+k}$. It is not hard to show that $\mathcal{V}_k$ is a complete intersection and has only one absolutely irreducible component. Thus it suffices to show that for almost all specializations of $z \in \mathbb{F}_q^J$, the corresponding variety $\mathcal{V}_k(\boldsymbol{z})$ is absolutely irreducible; then provided $k(n+1) > J$, a version of the Lang-Weil bound [12] applies and gives the desired result. Although results of this type are known (see [4, 16] and references therein), unfortunately none of them seems to imply the desired statement. Nevertheless, since a generic variety is absolutely irreducible, the conjecture appears plausible.

Acknowledgments

AMC acknowledges support from ARO (grant W911NF-12-1-0482), CIFAR, IARPA, NSF (grant 1526380), and NRO. WvD acknowledges support from NSF (grant 1314969). IES acknowledges support from ARC (grant DP140100118).

References

[1] Dave Bacon, Andrew M. Childs, and Wim van Dam, *From optimal measurement to efficient quantum algorithms for the hidden subgroup problem over semidirect product groups*, Proceedings of the 46th IEEE Symposium on Foundations of Computer Science, 2005, pp. 469–478, available at arXiv:quant-ph/0504083.

[2] Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald de Wolf, *Quantum lower bounds by polynomials*, Journal of the ACM, Vol. 48, no. 4, pp. 778–797 (2001), available at quant-ph/9802049.

[3] Dan Boneh and Mark Zhandry, *Quantum-secure message authentication codes*, Proceedings of Eurocrypt, 2013, pp. 592–608.

[4] François Charles and Bjorn Poonen, *Bertini irreducibility theorems over finite fields*, Journal of the American Mathematical Society, Vol. 29, pp. 81–94 (2016).

[5] Andrew M. Childs and Wim van Dam, *Quantum algorithm for a generalized hidden shift problem*, Proceedings of the 18th ACM-SIAM Symposium on Discrete Algorithms, 2007, pp. 1225–1234, available at arXiv:quant-ph/0507190.

[6] Wim van Dam, *Quantum oracle interrogation: Getting all information for almost half the price*, Proceedings of the 39th IEEE Symposium on Foundations of Computer Science, 1998, pp. 362–367, available at arXiv:quant-ph/9805006.

[7] Thomas Decker, Jan Draisma, and Pawel Wocjan, *Efficient quantum algorithm for identifying hidden polynomials*, Quantum Information and Computation, Vol. 9, no. 3, pp. 215–230 (2009), available at arXiv:0706.1219.

[8] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser, *Limit on the speed of quantum computation in determining parity*, Physical Review Letters, Vol. 81, no. 24, pp. 5442–5444 (1998), available at quant-ph/9802045.

[9] Joachim von zur Gathen and Jürgen Gerhard, *Modern Computer Algebra*, Cambridge University Press, 2013.

[10] Daniel M. Kane and Samuel A. Kutin, *Quantum interpolation of polynomials*, Quantum Information and Computation, Vol. 11, no. 1, pp. 95–103 (2011), available at arXiv:0909.5683.

[11] Arnold Knopfmacher and John Knopfmacher, *Counting polynomials with a given number of zeros in a finite field*, Linear and Multilinear Algebra, Vol. 26, no. 4, pp. 287–292 (1990).

[12] Serge Lang and André Weil, *Number of points of varieties in finite fields*, American Journal of Mathematics, Vol. 76, pp. 819–827 (1954).

[13] David A. Meyer and James Pommersheim, *On the uselessness of quantum queries*, Theoretical Computer Science, Vol. 412, no. 51, pp. 7068–7074 (2011), available at arXiv:1004.1434.

[14] Ashley Montanaro, *The quantum query complexity of learning multilinear polynomials*, Information Processing Letters, Vol. 112, no. 11, pp. 438–442 (2012), available at arXiv:1105.3310.

[15] Gerald M. Pitstick, João R. Cruz, and Robert J. Mulholland, *A novel interpretation of Prony's method*, Proceedings of the IEEE, Vol. 76, no. 8, pp. 1052–1053 (1988).

[16] Bjorn Poonen, *Bertini theorems over finite fields*, Annals of Mathematics, Vol. 160, pp. 1099–1127 (2004).

[17] Jaikumar Radhakrishnan, Pranab Sen, and S. Venkatesh, *The quantum complexity of set membership*, Algorithmica, pp. 462–479 (2002), available at arXiv:quant-ph/0007021.

[18] Adi Shamir, *How to share a secret*, Communications of the ACM, Vol. 22, no. 11, pp. 612–613 (1979).