Local Correction of Linear Functions over the Boolean Cube*

Prashanth Amireddy
Harvard University
Boston, USA
pamireddy@g.harvard.edu

Amik Raj Behera
Aarhus University
Aarhus, Denmark
bamikraj@cs.au.dk

Manaswi Paraashar
University of Copenhagen
Copenhagen, Denmark
manpa@di.ku.dk

Srikanth Srinivasan
University of Copenhagen
Copenhagen, Denmark
srsr@di.ku.dk

Madhu Sudan
Harvard University
Boston, USA
madhu@cs.harvard.edu

ABSTRACT

We consider the task of locally correcting, and locally list-correcting, multivariate linear functions over the domain $\{0,1\}^n$ over arbitrary fields and more generally Abelian groups. Such functions form error-correcting codes of relative distance $1/2$ and we give local-correction algorithms correcting up to nearly $1/4$-fraction errors making $\widetilde{O}(\log n)$ queries. This query complexity is optimal up to $\mathrm{poly}(\log\log n)$ factors. We also give local list-correcting algorithms correcting $(1/2-\varepsilon)$-fraction errors with $\widetilde{O}_\varepsilon(\log n)$ queries.

These results may be viewed as natural generalizations of the classical work of Goldreich and Levin whose work addresses the special case where the underlying group is $\mathbb{Z}_2$. By extending to the case where the underlying group is, say, the reals, we give the first non-trivial locally correctable codes (LCCs) over the reals (with query complexity being sublinear in the dimension (also known as message length)).

Previous works in the area mostly focused on the case where the domain is a vector space or a group and this lends to tools that exploit symmetry. Since our domains lack such symmetries, we encounter new challenges whose resolution may be of independent interest. The central challenge in constructing the local corrector is constructing "nearly balanced vectors" over $\{-1,1\}^n$ that span 1^n — we show how to construct $O(\log n)$ vectors that do so, with entries in each vector summing to ± 1. The challenge to the local-list-correction algorithms, given the local corrector, is principally combinatorial, i.e., in proving that the number of linear functions within any Hamming ball of radius $(1/2-\varepsilon)$ is $O_\varepsilon(1)$. Getting this general result covering every Abelian group requires integrating a variety of known methods with some new combinatorial ingredients analyzing the structural properties of codewords that lie within small Hamming balls.

CCS CONCEPTS

• **Theory of computation** → **Error-correcting codes**; • **Mathematics of computing** → **Coding theory**.

KEYWORDS

Local Correction, Local List Correction, Goldreich-Levin, Group-valued polynomials

ACM Reference Format:
Prashanth Amireddy, Amik Raj Behera, Manaswi Paraashar, Srikanth Srinivasan, and Madhu Sudan. 2024. Local Correction of Linear Functions over the Boolean Cube. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing (STOC '24), June 24–28, 2024, Vancouver, BC, Canada.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3618260.3649746

1 INTRODUCTION

In this paper we consider the class of "linear" functions mapping $\{0,1\}^n$ to an Abelian group and give "local correction" and "local list-correction" algorithms for this family (of codes). We describe our problems and results in detail below. We start with some basic notation.

We denote the space of functions mapping $\{0,1\}^n$ to an Abelian group G by $\mathcal{F}(\{0,1\}^n, G)$. Given two functions f, g from this set, we denote by $\delta(f,g)$ the fractional Hamming distance between them, i.e. the fraction of points in $\{0,1\}^n$ on which f and g disagree. In other words,

$$\delta(f,g) = \Pr_{x\sim\{0,1\}^n}[f(x) \neq g(x)].$$

We say that f, g are δ-close if $\delta(f,g) \leq \delta$ and that f, g are δ-far otherwise. Given a set of functions $\mathcal{F} \subseteq \mathcal{F}(\{0,1\}^n, G)$, we denote by $\delta(f,\mathcal{F})$ the minimum distance between f and a function $P \in \mathcal{F}$. The function f is said to be δ-close if $\delta(f,\mathcal{F}) \leq \delta$ and otherwise δ-far from $\mathcal{F}$. We denote by $\delta(\mathcal{F})$ the minimum distance between two distinct functions in $\mathcal{F}$.

The main thrust of this paper is getting efficient local correcting algorithms for some basic classes of functions $\mathcal{F}$ that correct close to $\delta(\mathcal{F})/2$ fraction of errors uniquely (i.e. given $f : \{0,1\}^n \to G$ determine $P \in \mathcal{F}$ such that $\delta(f,P) < \delta(\mathcal{F})/2$), and to list-correct close to $\delta(\mathcal{F})$ fraction of errors with small sized lists (i.e., given f output a small list $P_1, \ldots, P_L \in \mathcal{F}$ containing all functions P

*PA is supported in part by a Simons Investigator Award and NSF Award CCF 2152413 to Madhu Sudan and a Simons Investigator Award to Salil Vadhan. ARB is supported by Srikanth Srinivasan's start-up grant from Aarhus University. MP is supported by Srikanth Srinivasan's start-up grants from Aarhus University and the University of Copenhagen. SS is partially employed by Aarhus University, Denmark. This work was supported by start-up grants from Aarhus University and the University of Copenhagen. MS is supported in part by a Simons Investigator Award and NSF Award CCF 2152413.

STOC '24, June 24–28, 2024, Vancouver, BC, Canada

ACM ISBN 979-8-4007-0383-6/24/06
https://doi.org/10.1145/3618260.3649746

such that $\delta(f,P) < \delta(\mathcal{F}) - \varepsilon$). We start by describing our class of functions.

Group Valued Polynomials. The function spaces we are interested in are defined by polynomials of low-degree over the Boolean cube $\{0,1\}^n$ with coefficients from an Abelian group G, where we view $\{0,1\} \subseteq \mathbb{Z}$. (Thus a monomial function is given by a group element $g \in G$ and subset $S \subseteq [n]$ and takes the value g at points $a \in \{0,1\}^n$ such that $a_i = 1$ for all $i \in S$. The degree of a monomial is $|S|$ and a degree d polynomial is the sum of monomials of degree at most d.) We let $\mathcal{P}_d(\{0,1\}^n, G)$ denote the space of polynomials of degree at most d in this setting. (If n and G are clear from context we refer to this class as simply $\mathcal{P}_d$.) The standard proof of the "Schwartz-Zippel Lemma" (dating back at least to Ore [27]) extends to this setting (see Theorem 2.1) and shows that two distinct degree d polynomials differ on at least a 2^{-d} fraction of $\{0,1\}^n$. Therefore $\delta(\mathcal{P}_d) = 2^{-d}$ and our goal is to correct half this fraction of errors uniquely and close to this fraction of errors with small-sized lists. (We do so for $d = 1$, though many results apply to general values of d.) Next, we describe our notion of efficiency for correction.

Local Correction. In this work we are interested in a particularly strong notion of decoding, namely "local correction". Informally, $\mathcal{F}$ is locally correctable if there exists a probabilistic algorithm C that, given a point $\mathbf{x} \in \{0,1\}^n$ and oracle access to a function f that is close to $\mathcal{F}$, computes $P(\mathbf{x})$ with high probability while making few queries to the oracle f, where P is a function in $\mathcal{F}$ closest to f. In contrast to the usual notion of decoding which would require explicitly outputting a description of P (say the coefficients of P when $\mathcal{F} = \mathcal{P}_d$) this notion thus only requires us to compute $P(\mathbf{x})$ for a given $\mathbf{x}$. The main parameters associated with a local correction algorithm are the fraction of errors it corrects and the number of queries it makes to the oracle f. Formally, we say $\mathcal{F}$ is (δ, q)-locally correctable if there exists a probabilistic algorithm that given $\mathbf{a} \in \{0,1\}^n$ and oracle access to the input polynomial f that is δ-close to $P \in \mathcal{F}$, outputs $P(\mathbf{a})$ correctly with probability at least $3/4$ by making at most q queries to f.

The question of local correction of low-degree polynomials has been widely studied [7, 13, 14, 31]. These works have focused on the setting when the domain has an algebraic structure such as being a vector space over a finite field. In contrast the "Schwartz-Zippel" lemma only requires the domain to be a product space. Kim and Kopparty [22] were the first to study the decoding of low-degree multivariate polynomials when the domain is a product set, though they do not study local correctibility. Bafna, Srinivasan, and Sudan [5] were the first to study the problem of local correctibility of linear polynomials, though their result was mainly a negative result. They showed that if the underlying field $\mathbb{F}$ is large, for example, $\mathbb{F} = \mathbb{R}$, then any $(\Omega(1), q)$-local correction algorithm for $\mathcal{P}_1$ with constant δ requires at least $\tilde{\Omega}(\log n)$ queries. In this work, we consider the task of designing local correction algorithms with nearly matching performance.

Local List Correction. When considering a fraction of errors larger than $\delta(\mathcal{F})/2$, the notion of correction that one usually appeals to is called "list-decoding" or "list-correction" as we will refer to it, to maintain a consistent distinction between the notion of recovering the message (decoding) and recovering its encoding (correction). Here the problem comes in two phases: First a combinatorial challenge of proving that for some parameter $\rho \in [\delta(\mathcal{F})/2, \delta(\mathcal{F})]$ we have an a priori bound $L = L(\rho)$ such that for every function $f : \{0,1\}^n \to G$ there are at most $t \le L$ functions $P_1, \ldots, P_t \in \mathcal{F}$ satisfying $\delta(f, P_i) \le \rho$. We define $\mathcal{F}$ to be (ρ, L)-list-decodable if it satisfies this property. Next comes the algorithmic challenge of finding the list of size of most L that includes all such P_i's. In the non-local setting, this is referred to as the list correction task. In the local setting, the task is subtler to define and was formalized by Sudan, Trevisan, and Vadhan [31] as follows:

A (ρ, L, q)-local-list-corrector for $\mathcal{F}$ is a probabilistic algorithm C that takes as input an index $i \in [L]$ and $\mathbf{x} \in \{0,1\}^n$ along with oracle access to $f : \{0,1\}^n \to G$ such that $C^f(i, \mathbf{x})$ is computed with at most q oracle calls to f and C satisfies the following property: For every polynomial $P \in \mathcal{F}$ such that $\delta(f, P) \le \rho$ there exists an index $i \in [L]$ such that for every $\mathbf{x} \in \{0,1\}^n$, $\Pr[C^f(i, \mathbf{x}) = P(\mathbf{x})] \ge 3/4$. (In other words $C^f(i, \cdot)$ provides oracle access to P and ranging over $i \in [L]$ gives oracle access to every $P_1, \ldots, P_t$ that are ρ-close to f, while some i may output spurious functions.)

The notion of list-decoding dates back to the work of Elias [12]. The seminal work of Goldreich and Levin [16] produced the first non-trivial list-decoders for any non-trivial class of functions. (Their work happens to consider the class $\mathcal{F} = \mathcal{P}_1(\{0,1\}^n, \mathbb{Z}_2)$ and actually give local list-decoders, though not strictly local list-correctors.) List-decoding of Reed-Solomon codes was studied by Sudan [30] and Guruswami and Sudan [20]. Local list-correction algorithms for functions mapping $\mathbb{F}_q^n$ to $\mathbb{F}_q$ for polynomials of degree $d \ll q$ were given in [31]. The setting of $d > q$ has been considered by Gopalan, Klivans and Zuckerman [17] and Bhowmick and Lovett [8]. In a somewhat different direction Dinur, Grigorescu, Kopparty and Sudan [9, 18] consider the class of homomorphisms from G to H for finite Abelian groups G and H, and give local list-correction algorithms for such classes of functions.

In this work, we give local list-correction algorithms for the class $\mathcal{P}_1(\{0,1\}^n, G)$ for every Abelian group G. We explain the significance of this work in the broader context below.

1.1 Motivation for Our Work

The problem of decoding linear polynomials over $\{0,1\}^n$ over an arbitrary Abelian group is a natural generalization of the work of Goldreich and Levin, who consider this problem over $\mathbb{Z}_2$. However, the error-correcting properties of the underlying code (rate and distance) remain the same over any Abelian group G. Further, standard non-local algorithms [29] over $\mathbb{Z}_2$ work almost without change over any G (see Appendix A in full version [2]). The local correction problem is, therefore, a natural next step.

There are also other technical motivations for our work from the limitations of known techniques. Perhaps the clearest way to highlight these limitations is that to date we have no $(\Omega(1), O(1))$-locally correctable codes over the reals with growing dimension. This glaring omission is highlighted in the results of [6, 10, 11]. The work of Barak, Dvir, Wigderson and Yehudayoff [6] and the

followup of Dvir, Saraf, and Wigderson [10] show that there are no $(\Omega(1), 2)$-locally correctable codes of super-constant dimension, while another result of Dvir, Saraf and Wigderson [11] shows that any $(\Omega(1), 3)$-locally correctable codes in $\mathbb{R}^n$ must have dimension at most $n^{1/2-\Omega(1)}$. Indeed, till our work, there has been very little exploration of code constructions over the reals. While our work does not give an $(\Omega(1), O(1))$-locally correctable code either, ours is the first to give n-dimensional codes that are $(\Omega(1), \widetilde{O}(\log n))$-locally correctable. (These are obtained by setting $G = \mathbb{R}$ in our results.)

A technical reason why existing local correction results do not cover any codes over the reals is that almost all such results rely on the underlying symmetries of the domain. Typical results in the area (including all the results cited above) work over a domain that is either a vector space or least a group. Automorphisms of the domain effectively play a central role in the design and analysis of the local correction algorithms; but they also force the range of the function to be related to the domain and this restricts their scope. In our setting, while the domain $\{0,1\}^n$ does have some symmetries, they are much less rich and unrelated to the structure of the range. Thus new techniques are needed to design and analyze local-correction algorithms in this setting. Indeed we identify a concrete new challenge — the design of "balanced" vectors in $\{-1,1\}^n$ (i.e., with the sum of entries being in $\{-1,1\}$) that span the vector 1^n — that enables local correction, and address this challenge. We remark that correcting $\mathcal{P}_d$ for $d > 1$ leads to even more interesting challenges that remain open.

A final motivation is just the combinatorics of the list-decodability of this space. For instance for the class $\mathcal{F} = \mathcal{P}_1(\{0,1\}^n, G)$ for any G, it is straightforward to show $\delta(\mathcal{F}) = 1/2$ and so the unique decoding radius is $1/4$, but the list-decoding radius was not understood prior to this work. The general bound in this setting would be the Johnson bound which only promises a list-decoding radius of $1 - 1/\sqrt{2} \approx 0.29$. Indeed a substantial portion of this work is to establish that the list-decoding radius of all these codes approaches $\delta(\mathcal{F}) = 1/2$.

We describe our results below before turning to their proofs.

1.2 Our Main Results

Our first result is an almost optimal local correction algorithm for degree 1 polynomials up to an error slightly less than $1/4$, which is the unique decoding radius.

Theorem 1.1 (Local correction algorithms for $\mathcal{P}_1$ up to the unique decoding radius). *The space $\mathcal{P}_1$ over any Abelian group G has a (δ, q)-local correction algorithm where $\delta = \frac{1}{4} - \varepsilon$ for any constant $\varepsilon > 0$ and $q = \tilde{O}(\log n)$.*

We remark that Theorem 1.1 is tight up to $\mathrm{poly}(\log \log n)$ factors due to a lower bound of $\Omega(\log n / \log \log n)$ by earlier work of Bafna, Srinivasan, and Sudan [5]. Using further ideas, we also extend the algorithm from Theorem 1.1 to the list decoding regime. For this, we need first a combinatorial list decoding bound.

Theorem 1.2 (Combinatorial list decoding bound for $\mathcal{P}_1$). *For any constant $\varepsilon > 0$, the space $\mathcal{P}_1$ over any Abelian group G is $(1/2 - \varepsilon, \mathrm{poly}(1/\varepsilon))$-list decodable.*

Using the combinatorial list decoding bound, we also give a local list correction algorithm for degree 1 polynomials. We state the result below. For a formal definition of local list correction, refer to Definition 2.3.

Theorem 1.3 (Local List Correction for degree-1). *There exists a fixed polynomial L such that for all Abelian groups G and for every $\varepsilon > 0$ and $n \in \mathbb{Z}^+$ we have that $\mathcal{P}_1(\{0,1\}^n, G)$ is $(1/2 - \varepsilon, L(\varepsilon), \widetilde{O}(\log n))$-locally list correctable.*

Specifically, there is a randomized algorithm $\mathcal{A}$ that, when given oracle access to a polynomial f and a parameter $\varepsilon > 0$, outputs with probability at least $3/4$ a list of randomized algorithms $\phi_1, \ldots, \phi_L$ ($L \le \mathrm{poly}(1/\varepsilon)$) such that the following holds:

For each $P \in \mathcal{P}_1$ that is $(1/2 - \varepsilon)$-close to f, there is at least one algorithm ϕ_i that, when given oracle access to f, computes P correctly on every input with probability at least $3/4$.

The algorithm $\mathcal{A}$ makes $O_\varepsilon(1)$ queries to f, while each ϕ_i makes $\tilde{O}_\varepsilon(\log n)$ queries to f.

Using known local testing results [3, 5], one can show that the local list-correction Theorem 1.3 actually implies Theorem 1.1. Nevertheless, we give the proof of Theorem 1.1 in its entirety, since it is a simpler self-contained proof than the one that goes through Theorem 1.3, and introduces some of the same ideas in an easier setting. A weak version of Theorem 1.1 is also required for Theorem 1.3.

1.3 Proof Overview

1.3.1 Local Correction - Theorem 1.1. We prove Theorem 1.1 in three steps. The first step, which is specific to the space of linear polynomials, shows how to locally correct from any oracle f that $O(1/\log n)$-close to a degree-1 polynomial in n variables using $O(\log n)$ queries. In the second and third steps, we show how to increase the radius of decoding from $O(1/\log n)$ to an absolute constant and then to (nearly) half the unique decoding radius. The latter two steps also work for polynomials of degree greater than 1.

First Step. To motivate the proof of the first step, it is worth recalling the idea behind the lower bound result of [5] mentioned above. For simplicity, let us assume that we are working with *homogeneous* linear polynomials over $\mathbb{R}$. In this situation, we can equivalently replace the domain with $\{-1,1\}^n$. Now, assume the given oracle $f : \{-1,1\}^n \to \mathbb{R}$ agrees with some homogeneous linear polynomial P at all points of Hamming weight in the range $[\frac{n}{2} - n^{0.51}, \frac{n}{2} + n^{0.51}]$, and takes adversarially chosen values outside this set. It is easy to check that f is $\exp\left(-n^{\Omega(1)}\right)$-close to P. Further, assume that we only want to correct the polynomial P at 1^n.

Over $\mathbb{R}$, the space of homogeneous linear polynomials forms a vector space. Hence, given access to an oracle f that is close to a codeword P, it is natural to correct P using a 'linear algorithm' in the following sense. To correct P at 1^n, we choose points $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(q)} \in \{-1,1\}^n$ and output

$$c_1 f(\mathbf{x}^{(1)}) + \cdots + c_q f(\mathbf{x}^{(q)})$$

for some coefficients $c_1, \ldots, c_q \in \mathbb{R}$. (Indeed, it is not hard to argue that if any strategy works, then there must be a linear algorithm that does [5].)

Since this strategy must work when given P itself as an oracle, it must be the case that

$$P(1^n) = c_1 P(\mathbf{x}^{(1)}) + \cdots + c_q P(\mathbf{x}^{(q)})$$

for any linear polynomial P. Further, as the space of homogeneous linear polynomials is a vector space spanned by the coordinate (i.e. dictator) functions, it is necessary and sufficient to have

$$1^n = c_1 \cdot \mathbf{x}^{(1)} + \cdots + c_q \cdot \mathbf{x}^{(q)}. \tag{1}$$

Finally, for such a correction algorithm to work for f as given above, it must be the case that each of the 'query points' $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(q)}$ has Hamming weight in the range $[n/2 - n^{0.51}, n/2 + n^{0.51}]$.

Note that Equation (1) cannot hold for *perfectly balanced* (i.e. Hamming weight exactly $n/2$) query points, no matter what q we choose: this is because the query points lie in a subspace not containing 1^n. The work of [5] showed a robust version of this: for any set of 'nearly-balanced' vectors with Hamming weight in the range $[n/2 - n^{0.51}, n/2 + n^{0.51}]$ that satisfy Equation (1), it must hold that $q = \Omega(\log n / \log\log n)$. At a high level, this lower bound holds because if Equation (1) is true, then the coefficients can be taken to be at most $q^{O(q)}$ in magnitude (via a suitable application of Cramer's rule). The lower bound then follows by adding up the entries of the vectors on both sides of Equation (1).

The first step of the algorithm is based on showing that this lower bound is essentially tight. More formally, we show that we can find $q = O(\log n)$ nearly-balanced (in fact, the vectors we construct have Hamming weight $n/2 \pm 1$) vectors $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(q)} \in \{0,1\}^n$ such that the following (more general) equation holds.

$$1^{n+1} = c_1 \cdot \begin{pmatrix} 1 \\ \mathbf{x}^{(1)} \end{pmatrix} + \cdots + c_q \cdot \begin{pmatrix} 1 \\ \mathbf{x}^{(q)} \end{pmatrix}. \tag{2}$$

This identity allows us to correct any linear (not just homogeneous) polynomial. Moreover, we show that we can take the coefficients to be *integers*, which allows us to apply this algorithm over any Abelian group (it makes sense to multiply a group element g with an integer k, since it amounts to adding either the element g or its inverse $-g$, $|k|$ times).

Finally, we show that this construction also implies a similar algorithm to compute $P(1^n)$ from *any* f that is $O(1/\log n)$-close to P (and not just the special f given above). This is done by constructing random query points $\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(q)}$ where the ith bit of these vectors is picked by choosing a *random* bit of $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(q)}$. The fact that each $\mathbf{x}^{(j)}$ is nearly balanced implies that each $\mathbf{y}^{(j)}$ is nearly uniform over $\{0,1\}^n$ and hence likely not an error point of f. Intuitively, the distance requirement is because we make q (nearly) random queries to f and the algorithm succeeds if none of the query points is in error. So, the algorithm correctly computes $P(1^n)$ when $\delta(f, P)$ is sufficiently smaller than $1/q$. By a suitable 'shift', we can also correct at points other than 1^n.

This construction of the points $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(q)} \in \{0,1\}^n$ is based on ensuring that the coefficients $c_1, \ldots, c_q$ must be exponentially large in q (to ensure that the argument of [5] is tight). This leads to the natural problem of finding a hyperplane whose Boolean solutions cannot be described by an equation with small coefficients. This is a topic that has received much interest in the study of threshold circuits and combinatorics [1, 4, 15, 21, 28].

For the result stated above, we require only a simple construction. Consider the following equation over $\{0,1\}^q$ where $q = 2k$. The first k bits describe an integer $i \in \{0, \ldots, 2^k - 1\}$ and the last k bits describe an integer j. The hyperplane expresses the constraint that $j = i - 1$. This hyperplane can easily be described using coefficients that are exponentially large in k and one can easily show that this is in fact necessary. After some modification to ensure that the coefficients sum to 1, we get Equation (2). See Lemma 3.4 for more details.

Using a more involved construction due to Håstad [21] and its extension due to Alon and Vu [1], it is possible to show that we can achieve $q = O(\log n / \log\log n)$, showing that the lower bound of [5] is in fact tight up to constant factors (see Appendix B of the full version [2]). However, in this case, we don't know how to ensure that the coefficients $c_1, \ldots, c_q$ are integers, meaning that the algorithm does not extend to general Abelian groups (moreover, we also lose $\mathrm{poly}(\log\log n)$ factors in query complexity in the subsequent steps, and so the final algorithm is only tight up to $\mathrm{poly}(\log\log n)$ factors, no matter which construction we use in the first step).

Second and Third Steps. To obtain an algorithm resilient to a larger fraction of errors, we use a process of error reduction. Specifically, we show that, given an oracle $f : \{0,1\}^n \to G$ that is δ-close to a polynomial $P \in \mathcal{P}_1$, we can obtain (with high probability) an oracle $g : \{0,1\}^n \to G$ that is $O(1/\log n)$-close to P; we can then apply the above described local correction algorithm to g to correct P at any given point. The oracle g makes $\mathrm{poly}(\log\log n)$ queries to f and hence the overall number of queries to f is $\tilde{O}(\log n)$.

Interestingly, the error-reduction step is not limited to linear polynomials. We show that this also works for the space of degree-d polynomials, where the number of queries now also depends on the degree parameter d. In general, δ can be arbitrarily close to the unique decoding radius of $\mathcal{P}_d$, which is $2^{-(d+1)}$.

We use two slightly different error-reduction algorithms to handle the case when δ is a small constant, and the case when δ is close to $2^{-(d+1)}$ respectively. We reduce the latter case to the former case and the former to the case of error $O(1/\log n)$. It is simpler to describe the error reduction algorithm when the error is large, i.e. close to the unique decoding radius, so we start there.

The process of error-reduction may be viewed as an *average-case* version of the correction problem, where we are only required to compute P on *most* points in $\{0,1\}^n$ with high probability. Assume, therefore, that we are given a *random* point $\mathbf{a} \in \{0,1\}^n$ and we are required to output $P(\mathbf{a})$ (with high probability).

In the setting where the domain is not $\{0,1\}^n$ but rather a vector space like $\mathbb{F}_q^n$, a natural strategy going back to the works of Beaver and Feigenbaum [7] and Lipton [24] is to choose a random subspace V of appropriate constant (here, we think of all parameters except n as constants) dimension k containing $\mathbf{a}$ and then find the closest k-variate degree-d polynomial to the restriction $f|_V$ of f to this subspace. The reason this works is that the points in a random subspace come from a pairwise independent distribution and hence standard second-moment methods show that $\delta(f|_V, P|_V) \approx \delta$ with high probability, in which case $\delta(f|_V, P|_V)$ is also less than the unique-decoding radius of $\mathcal{P}_d$. A brute-force algorithm (or better ones, such as the Welch-Berlekamp algorithm (see e.g. [19, Chapter 15])) can now be used to find $P|_V$, which also determines $P(\mathbf{a})$.

To adapt this idea to the setting of $\{0,1\}^n$, we note that random subspaces are not available to us since most constant-dimensional subspaces don't have points in $\{0,1\}^n$. However, we observe that we can apply the above idea to a *random subcube* in $\{0,1\}^n$. More specifically, we identify variables randomly into k buckets via a random hash function $h : [n] \to [k]$, reducing the original set of n variables $x_1, \ldots, x_n$ to a set of k variables $y_1, \ldots, y_k$. Further, to ensure that the given point $\mathbf{a}$ is in the chosen subcube, we start by replacing x_i by $x_i \oplus a_i$ before the identification process (the process of XORing a variable x by a Boolean value b is equivalent to either leaving the variable as is when $b = 0$, or replacing x by $1 - x$ when $b = 1$. This does not affect the degree of the polynomial P). This gives rise to a random subcube C containing $\mathbf{a}$ (obtained by setting $y_1 = \cdots = y_k = 0$). We define a random subcube formally in Definition 2.7. We can now apply the above idea by restricting the given f to this subcube.

Having defined a subcube C as above, the non-trivial part of the argument is to show that $\delta(f|_C, P|_C) \approx \delta$. This is not obvious as the points of the subcube C are not pairwise independent. Nevertheless, for random $\mathbf{a}$, the points of C are 'noisy' copies of one another (Definition 2.5). Using this fact and standard hypercontractivity estimates, we can show that most pairs of points of C are 'approximately' pairwise independent (see Theorem 2.6 below) as long as k is a large enough constant. This allows us to use the second-moment method to recover $P(\mathbf{a})$ as before, for all but a small fraction δ' of possible inputs $\mathbf{a}$ (with high probability). The parameter k is $\mathrm{poly}(1/\delta')$ making the query complexity a constant as long as the required error δ' is constant. This is proved in Section 3.2.

To reduce the error further down to $O(1/\log n)$, we modify the above idea. We repeat the above process (Actually, we need to slightly modify the process to ensure that we only query 'balanced' points on the subcube C. We postpone this detail to the proof.) on three randomly chosen subcubes of dimension k' each containing $\mathbf{a}$ and take a plurality vote of their outputs. The probability of error in this algorithm is bounded by the probability that at least two of the iterations query a point of error, which would be $\leq O_{k'}((\delta')^2)$ if the repetitions were independent. However, the iterations here have some dependency - each iteration uses the *same* random input $\mathbf{a}$. Nevertheless, using hypercontractivity, we can again argue that if k' is a large enough constant *depending only on* d, the probability of error is at most $O_{k'}((\delta')^{1.5}) \leq (\delta')^{1.1}$ for small enough δ'. Repeating this process t times, gives an error that is *double-exponentially* small in t, at the expense of $O_{k'}(1)^t$ many queries. Choosing t to be $O(\log\log\log n)$ gives us an oracle that is $O(1/\log n)$-close to P. Since this step uses similar ideas to the previous error-reduction algorithm, we defer the proof to the full version.

1.3.2 Combinatorial List Decoding Bound - Theorem 1.2. We first note that the list size can indeed be as large as $\mathrm{poly}(1/\varepsilon)$, no matter the underlying group G. This is shown by the following example. Fix an integer parameter t and any non-zero element $g \in G$. Let $f = \mathrm{Maj}_G^t(x_1, \ldots, x_t)$ denote the function of the first t input variables that takes the value g when its input has Hamming weight greater than $t/2$ and the value 0 otherwise. A standard calculation (see e.g. [26, Theorem 5.19]) shows that Maj_G^t agrees with the linear functions $g \cdot x_i$ ($i \in [t]$) on a $(\frac{1}{2} + \frac{O(1)}{\sqrt{t}})$-fraction of inputs. Setting $t = \Theta(1/\varepsilon^2)$, we see that this agreement can be made $\frac{1}{2} + \varepsilon$. This implies that for f as defined above, the list size at distance $\frac{1}{2} - \varepsilon$ can be as large as $\Omega((1/\varepsilon)^2))$.

To motivate the proof of Theorem 1.2, it is helpful to start with the case when G is a group $\mathbb{Z}_p$ of prime order. There are two extremes in this case: $p = 2$ and large p (say a large constant or even growing with n).

Case 1: $p = 2$. The case $p = 2$ is the classical setting that has been intensively investigated in the literature, starting with the foundational work of Goldreich and Levin [16] (see also the work of Kushilevitz and Mansour [23]). In this setting, it is well-known that the bound of $1/\varepsilon^2$ is tight. This follows from the standard Parseval identity from basic Fourier analysis of Boolean functions (see e.g. [26]) or as a special case of the binary Johnson bound (see e.g. the appendix of [9]). At a high level, this is because the Boolean Fourier transform identifies each $f : \{0,1\}^n \to \mathbb{Z}_2$ with a real unit vector $\mathbf{v}_f$ such that distinct linear polynomials are mapped to an orthonormal basis. Moreover, if f is $(\frac{1}{2} - \varepsilon)$-close to a linear polynomial P, then the length of projection of the vector $\mathbf{v}_f$ on $\mathbf{v}_P$ is at least ε. Pythagoras' theorem now implies the list bound.

Case 2: Large p. For $p > 2$, it is unclear if we can map distinct linear polynomials to orthogonal real or complex vectors in the above way. Nevertheless, we do expect the list-size bound to hold, as the distance $\delta(\mathcal{P}_1)$ is the same as over $\mathbb{Z}_2$, i.e. $1/2$. Moreover, a *random* pair of linear polynomials have a distance much larger than $1/2$ for large p. This latter fact is a consequence of *anti-concentration* of linear polynomials, which informally means the following. Let $P(\mathbf{x})$ be a non-zero polynomial with many (say, at least 100) non-zero coefficients. Then, on a random input $\mathbf{a}$, the random variable $P(\mathbf{a})$ does not take any given value $b \in \mathbb{Z}_p$ with good probability (say, greater than $1/5$).

In the case of large p, we crucially use anti-concentration to argue the upper bound on the list size. At a high level, in this case, we can show that if a function $f : \{0,1\}^n \to \mathbb{Z}_p$ is $(\frac{1}{2} - \varepsilon)$-close to many (say L) linear polynomials, then there is a large subset (size $L' = L^{\Omega(1)}$) that 'look' somewhat like the example of the Maj_G^t example mentioned above. More precisely, the coefficient vectors of the linear polynomials in this subset are at most a *constant* (independent of p, order of the underlying group) Hamming distance from one another. By shifting the polynomials by one of the linear functions in the subset, we can assume without loss of generality that all the linear functions in fact have a constant number of *non-zero coefficients*, as in the case of the list of polynomials corresponding to Maj_G^t. It now suffices to bound the size L' of this subset by $\mathrm{poly}(1/\varepsilon)$. The bound now reduces to a case analysis based on the number of variables that appear in the coefficients of the L' polynomials in the subset.

Putting it Together. We sketch here how to handle general finite Abelian groups. In the proof, we show that this also implies the same bound for infinite groups such as $\mathbb{R}$.

Recall that any finite group G is a direct product of cyclic groups, each of which has a size that is a prime power. We collect the terms in this product to write $G = G_1 \times G_2 \times G_3$ where G_1 is the product of the factors of sizes that are powers of 2, G_2 is the same with powers of 3, and G_3 is the product of the rest (There is nothing very special

about this decomposition. Essentially, we have one argument that works for 'small' p and another that works for 'large' p. To combine them, we need some formalization of these notions. Here, 'large' could be defined to be larger than any constant $C \geq 5$.). A simple observation shows that it suffices to bound the size of the list in each of these cases by $\mathrm{poly}(1/\varepsilon)$.

For G_3, the argument of large p sketched above works without any change (with some care to ensure that we can handle all the primes greater than or equal to 5). The only part of the argument that is sensitive to the choice of the group is the initial use of anti-concentration, and this works over G_3 since the order of any non-zero element is large (i.e. at least 5).

The argument for G_1 needs more work. While the use of Parseval's identity works over $\mathbb{Z}_2$, it is not clear how to extend it to groups of size powers of 2, such as $\mathbb{Z}_4$. For inspiration, we turn to a different extension of the $\mathbb{Z}_2$-case proved by Dinur, Grigorescu, Kopparty, and Sudan [9]. They deal with the list-decodability of the space of *group homomorphisms* from a group H to a group G. Setting the group H to be $\{0,1\}^n$ (with addition defined by the XOR operation) and G to be $\mathbb{Z}_2$, we recover again the setting of (homogeneous) linear polynomials over $\mathbb{Z}_2$. The work of [9] shows how to extend this result to larger groups G that have order a power of 2. Note that it is not immediately clear that this should carry over to the setting of linear polynomials: for groups of order greater than 2, the space of polynomials is different from the space of homomorphisms. However, we show that the technique of [9] does work in our setting as well.

Finally, the proof for G_2 is a combination of the ideas of the two proofs above. We omit the details here and refer the reader to the actual proof.

1.3.3 Local List Correction - Theorem 1.3. Like the proof of the second and third steps of Theorem 1.1 described above, at the heart of our local list correction algorithm lies an error-reduction algorithm. More precisely, we design an algorithm $\mathcal{A}_1^f$ which, using oracle access to f, produces a list of algorithms $\psi_1, \ldots, \psi_L$ such that, with high probability, for each linear polynomial P that is $(\frac{1}{2} - \varepsilon)$-close to f, there is at least one algorithm ψ_j in the list that agrees with P on *most* inputs, i.e. ψ_j "approximates" P. Here, $L \leq L(\varepsilon)$ denotes the list-size bound proved in Theorem 1.2. Further, each ψ_i makes at most $O_L(1) = O_\varepsilon(1)$ queries to f.

We can now apply the algorithm from the unique correction setting with oracle access to the various ψ_j to produce the desired list $\phi_1, \ldots, \phi_L$ as required.

The proof is motivated by a local list-decoding algorithm for low-degree polynomials over $\mathbb{F}_q^n$ due to Sudan, Trevisan, and Vadhan [31]. In that setting, we are given oracle access to a function $f : \mathbb{F}_q^n \to \mathbb{F}_q$ and we are required to produce a list as above that approximates the set $S = \{P_1, \ldots, P_L\}$ of degree-d polynomials (say $d = o(q)$) that have significant (say $\Omega(1)$) agreement with f. It follows from the Johnson bound that $L = O(1)$ in this case (see, e.g. [19, Chapter 7]). The corresponding algorithm $\mathcal{A}_{\mathrm{STV}}$ chooses a *random* point $\mathbf{a}$ and gets as advice the values of $\alpha_i = P_i(\mathbf{a})$ for each $i \in [L]$. (We can easily get rid of this advice assumption, but let us assume for now that we have it.)

Now, we want to produce an algorithm that approximates P_i. Given a random point $\mathbf{b} \in \mathbb{F}_q^n$, the algorithm constructs the random line ℓ passing through $\mathbf{a}$ and $\mathbf{b}$ and produces the list of univariate polynomials that have significant agreement with the restriction $f|_\ell$ of f to the line. This can be done via brute force with $O(d)$ queries (if one only cares about query complexity) or in $\mathrm{poly}(d, \log q)$ time using Sudan's list decoding algorithm for univariate polynomials [30]. By pairwise independence and standard second-moment estimates, it is easy to argue that for each $j \in [L]$, $P_j|_\ell$ is in this list of univariate polynomials. However, to single out $P_i|_\ell$ in this list, we use advice $\alpha_i = P_i(\mathbf{a})$. Since $\mathbf{a}$ is a *random* point on ℓ (even given ℓ), it follows that, with high probability, α_i uniquely disambiguates $P_i|_\ell$ from the ($O(1)$ many) other polynomials in the list. In particular this also determines $P_i(\mathbf{b})$, since $\mathbf{b}$ lies on ℓ.

Let us now turn to our local list correction algorithm. We use similar ideas to [31] but, as in the proof of Theorem 1.1, with *subcubes* instead of lines. More precisely, the algorithm $\mathcal{A}_1^f$ produces a random $\mathbf{a}$ and a random hash function $h : [n] \to [k]$ ($k = O_\varepsilon(1)$ suitably large), and uses them to produce a random subcube C as in the proof sketch of Theorem 1.1. The advice in this case is the restriction $P|_{\mathrm{C}}$ for each polynomial P in the set $S = \{P_1, \ldots, P_L\}$ of degree-1 polynomials that are $(\frac{1}{2} - \varepsilon)$-close to f.

Now, given a random point $\mathbf{b} \in \{0,1\}^n$, we correct $P_i(\mathbf{b})$ as follows. We first construct the smallest subcube C′ that contains both C and the point $\mathbf{b}$. With high probability, this is a subcube of dimension $2k$. Using a simple brute-force algorithm that uses 2^{2k} queries to f, we can find the set S' of all $2k$-variate linear polynomials that are $(\frac{1}{2} - \frac{\varepsilon}{2})$-close to $f|_{\mathrm{C}'}$. Note that $|S'| \leq L(\varepsilon)$. By a hypercontractivity-based argument (as we did in the error reduction algorithms), we can show that, with high probability, each $P_j|_{\mathrm{C}'}$ is in this list S'. To single out $P_i|_{\mathrm{C}'}$, we use advice $P_i|_{\mathrm{C}}$. The proof that this works needs an understanding of the distribution of C given C′: it turns out that the k-dimension subcube C is obtained by randomly pairing up variables in C′ and identifying them with a single variable. We show that, if k is large enough in comparison to the list bound L, then with high probability, this process does not identify any two distinct elements in the list (there is a small subtlety in the argument that is being hidden here for simplicity). Thus, we are able to single out $P_i|_{\mathrm{C}'}$ and this allows us to compute $P_i(\mathbf{b})$ correctly, with high probability over the choice of $\mathbf{b}$ and the randomness of the algorithm (which includes $\mathbf{a}$ and the hash function h).

Finally, to get rid of the advice, we note that a similar hypercontractivity based argument also shows that each $P_i|_{\mathrm{C}}$ is $(\frac{1}{2} - \frac{\varepsilon}{2})$-close to $f|_{\mathrm{C}}$. So by applying a similar brute-force algorithm on C, we find, with high probability, a set $\tilde{S}$ of polynomials containing $P_i|_{\mathrm{C}}$ for each $i \in [L]$. This is good enough for the argument above. The algorithm $\mathcal{A}_1^f$ first computes $\tilde{S}$ and then outputs the descriptions of the algorithm in the previous paragraph for each $P \in \tilde{S}$ (treating it as a restriction of one of the P_i).

Organization. We start with some preliminaries and then sketch the proof of Theorem 1.1. For lack of space, the proofs of some of the intermediate lemmas and the proofs of Theorem 1.2 and Theorem 1.3 are postponed to the full version of the paper [2].

2 PRELIMINARIES

2.1 Notation

Let $(G, +)$ denote an Abelian group G with addition as the binary operation. For any $g \in G$, let $-g$ denote the inverse of $g \in G$. For any $g \in G$ and integer $a \geq 0$, $a \cdot g$ (or simply ag) is the shorthand notation of $g + \ldots + g$ (added a times) and $-ag$ denotes $a \cdot (-g)$.

For a natural number n, we consider functions $f : \{0,1\}^n \to G$. We denote the set of functions that can be expressed as a multilinear polynomial of degree d, with the coefficients being in G by $\mathcal{P}_d(n, G)$. We will simply write P_d when n and G are clear from the context. For $\mathbf{x}, \mathbf{y} \in \{0,1\}^n$, let $\delta(\mathbf{x}, \mathbf{y})$ denote the relative Hamming distance between $\mathbf{x}$ and $\mathbf{y}$, i.e. $\delta(\mathbf{x}, \mathbf{y}) = |\{i \in [n] \mid x_i \neq y_i\}|/n$.

For any $\mathbf{x} \in \{0,1\}^n$, $|\mathbf{x}|$ denotes the Hamming weight of $\mathbf{x}$. $\tilde{O}(\cdot)$ notation hides factors that are poly-logarithmic in its argument. For a polynomial $P(\mathbf{x})$, let $\mathrm{vars}(P)$ denote the variables on which P depends, i.e. the variables that appear in a monomial with non-zero coefficient in P. For any natural number n, U_n denotes the uniform distribution on $\{0,1\}^n$.

2.2 Basic Definitions and Tools

Probabilistic Notions. For any distribution X on $\{0,1\}^n$, let $\mathrm{supp}(X)$ denote the subset of $\{0,1\}^n$ on which X takes non-zero probability. For two distributions X and Y on $\{0,1\}^n$, the statistical distance between X and Y, denoted by $\mathrm{SD}(X, Y)$ is defined as

$$\mathrm{SD}(X, Y) = \max_{T \subseteq \{0,1\}^n} |\Pr[X \in T] - \Pr[Y \in T]|$$

We say X and Y are ε-close if the statistical distance between X and Y is at most ε.

Coding Theory Notions. Fix an Abelian group G. We use $\mathcal{P}_d$ to denote the space of multilinear polynomials from $\{0,1\}^n$ to G of degree at most d. More precisely, any element $P \in \mathcal{P}_d$ can be described as

$$P(x_1, \ldots, x_n) = \sum_{I \subseteq [n] : |I| \leq d} \alpha_I \prod_{i \in I} x_i$$

where $\alpha_I \in G$ for each I. On an input $\mathbf{a} \in \{0,1\}^n$, each monomial evaluates to a group element in G and the polynomial evaluates to the sum of these group elements.

The following is a summary of standard facts about multilinear polynomials, which also hold true in the setting when the range is an arbitrary Abelian group G. The proofs are standard and omitted.

Theorem 2.1. (1) *(Möbius Inversion) Any function $f : \{0,1\}^n \to G$ has a unique representation as a multilinear polynomial in $\mathcal{P}_n$. Moreover, we have $f = \sum_{I \subseteq [n]} c_I \prod_{i \in I} x_i$ where for any $I \subseteq [n]$, we have*

$$c_I = \sum_{J \subseteq I} (-1)^{|I \setminus J|} f(1_J)$$

where 1_J is the indicator vector of the set J.

(2) *(DeMillo-Lipton-Schwartz-Zippel) Any non-zero polynomial $P \in \mathcal{P}_d$ is non-zero with probability at least 2^{-d} at a uniformly random input from $\{0,1\}^n$. Equivalently, $\delta(\mathcal{P}_d) \geq 2^{-d}$.*

We now turn to the kinds of algorithms we will consider. Below, let $\mathcal{F}$ be any space of functions mapping $\{0,1\}^n$ to G.

Definition 2.2 (Local Correction Algorithm). We say that $\mathcal{F}$ has a (δ, q)-local correction algorithm if there is a probabilistic algorithm that, when given oracle access to a function f that is δ-close to some $P \in \mathcal{F}$, and given as input some $\mathbf{a} \in \{0,1\}^n$, returns $P(\mathbf{a})$ with probability at least $3/4$. Moreover, the algorithm makes at most q queries to its oracle.

Definition 2.3 (Local List-Correction Algorithm). We say that $\mathcal{F}$ has a (δ, q_1, q_2, L)-local list correction algorithm if there is a randomized algorithm $\mathcal{A}$ that, when given oracle access to a function f, produces a list of randomized algorithms $\phi_1, \ldots, \phi_L$, where each ϕ_i has oracle access to f and have the following property: with probability at least $3/4$, for each codeword P that is δ-close to f, there exists some $i \in [L]$ such that the algorithm ϕ_i computes P with error at most $1/4$, i.e. on any input $\mathbf{a}$, the algorithm ϕ_i outputs $P(\mathbf{a})$ with probability at least $3/4$.
Moreover, the algorithm $\mathcal{A}$ makes at most q_1 queries to f, while the algorithms $\phi_1, \ldots, \phi_L$ each make at most q_2 queries to f.

Definition 2.4 (Combinatorial List Decodability). We say that $\mathcal{F}$ is (δ, L)-list decodable if for any function f, the number of elements of $\mathcal{F}$ that are δ-close to f is at most L.

The questions of decoding polynomial-based codes over groups become much more amenable to known techniques if we drop the locality constraint. In the full version of the paper, we show how the standard Majority-logic decoding algorithms also yield non-local unique and list-decoding algorithms for $\mathcal{P}_d$ over any Abelian group.

Hypercontractivity. Next we are going to state a consequence of the standard Hypercontracitivity theorem (Refer to [26, Chapter 9]).

Definition 2.5 (Noise distribution). Let $\rho \in [-1, 1]$. For a fixed $\mathbf{x} \in \{0,1\}^n$, $\mathbf{y} \sim \mathcal{N}_\rho(\mathbf{x})$ denotes a random variable defined as follows: For each $i \in [n]$ independently,

$$y_i := \begin{cases} x_i, & \text{with prob. } (1+\rho)/2 \\ \neg x_i, & \text{with prob. } (1-\rho)/2 \end{cases}$$

In other words, to sample from the distribution $\mathcal{N}_\rho(\mathbf{x})$), we flip each bit of $\mathbf{x}$ independently with probability $(1-\rho)/2$, and keeping it unchanged with probability $(1+\rho)/2$.

Theorem 2.6 ([26, Section 9.5]). *Let $E \subseteq \{0,1\}^n$ be a subset of density δ, i.e. $|E|/2^n = \delta$. Then for any $\rho \in [-1, 1]$,*

$$\Pr_{\substack{\mathbf{x} \sim \{0,1\}^n \\ \mathbf{y} \sim \mathcal{N}_\rho(\mathbf{x})}} [\mathbf{x} \in E \text{ and } \mathbf{y} \in E] \leq \delta^{2/(1+|\rho|)}$$

In particular, if ρ is close to 0, then Theorem 2.6 tells us that the probability that $\mathbf{x}$ and $\mathbf{y}$ are in E is close to the probability when $\mathbf{x}$ and $\mathbf{y}$ are sampled independently and uniformly from U_n.

Subcubes of $\{0,1\}^n$. It will be very useful in our algorithms to be able to restrict the given function to a small-dimensional subcube and analyze this restriction. We construct such subcubes by first negating a subset of the variables, then identifying them into a smaller set of variables. A more precise definition follows.

Definition 2.7 (Embedding a smaller cube into $\{0,1\}^n$). Fix any $k \in \mathbb{N}$ and $k \leq n$. Fix a point $\mathbf{a} \in \{0,1\}^n$ and a function $h : [n] \to$

$[k]$. For every $\mathbf{y} \in \{0,1\}^k$, $x(\mathbf{y})$ is defined with respect to $\mathbf{a}$ and h as follows:

$$x(\mathbf{y})_i = y_{h(i)} \oplus a_i = \begin{cases} a_i, & \text{if } y_{h(i)} = 0 \\ 1 \oplus a_i, & \text{if } y_{h(i)} = 1 \end{cases}$$

$C_{\mathbf{a},h}$ is the subset in $\{0,1\}^n$ consisting of $x(\mathbf{y})$ for every $\mathbf{y} \in \{0,1\}^k$, i.e. $C_{\mathbf{a},h} := \left\{ x(\mathbf{y}) \;\middle|\; \mathbf{y} \in \{0,1\}^k \right\}$.

Given any polynomial $P(x_1, \ldots, x_n)$ and any subcube $C_{\mathbf{a},h}$ as above, P restricts naturally to a degree-d polynomial $Q(y_1, \ldots, y_k)$ on $C_{\mathbf{a},h}$ obtained by replacing each x_i by $y_{h(i)} \oplus a_i$. We use $P|_{C_{\mathbf{a},h}}$ to denote the polynomial Q.

Random Subcubes. Now assume that we choose a subcube $C_{\mathbf{a},h}$ by sampling $\mathbf{a} \sim \{0,1\}^n$ and sampling a random hash function $h : [n] \to [k]$. For any $\mathbf{y} \in \{0,1\}^k$, $x(\mathbf{y})$ is the image of $\mathbf{y}$ in $\{0,1\}^n$ under $\mathbf{a}$ and h and $C_{\mathbf{a},h}$ is the subcube consisting of all 2^k such images. From the Definition 2.7, we can derive following two observations:

(1) For any $\mathbf{y} \in \{0,1\}^k$, distribution of $x(\mathbf{y})$ is the uniform distribution over $\{0,1\}^n$. This is because $\mathbf{a}$ is uniformly distributed over $\{0,1\}^n$.
(2) Fix $\mathbf{y}, \mathbf{y}' \in \{0,1\}^k$. Recall that $\delta(\mathbf{y}, \mathbf{y}')$ denotes the fractional Hamming distance between $\mathbf{y}$ and $\mathbf{y}'$. A simple calculation shows the following: For all $i \in [n]$, $x(\mathbf{y})_i \oplus x(\mathbf{y}')_i$ is 1 with probability $\delta(\mathbf{y}, \mathbf{y}')$, and 0 with probability $1 - \delta(\mathbf{y}, \mathbf{y}')$. Since this is true for any choice of $x(\mathbf{y})$, this means that the distribution of the random variable $x(\mathbf{y}) \oplus x(\mathbf{y}')$ is independent of $x(\mathbf{y})$. In particular, using also our observation in the previous item, we see that the pair $(x(\mathbf{y}), x(\mathbf{y}'))$ has the same distribution as $(\mathbf{z}, \mathbf{z}')$ where $\mathbf{z}$ is chosen uniformly at random from $\{0,1\}^n$ and $\mathbf{z}'$ is sampled from the distribution $\mathcal{N}_\rho(\mathbf{z})$, where $\rho = 1 - 2\delta(\mathbf{y}, \mathbf{y}')$.

Building on the above observation, we have the following sampling lemma for subcubes that will be useful. The proof, based on the second-moment method, is postponed to the full version.

Lemma 2.8 (Sampling lemma for random subcubes). *Sample $\mathbf{a}$ and h uniformly at random, and let $\mathrm{C} = C_{\mathbf{a},h}$ be the subcube of dimension k as described in Definition 2.7. Fix any $T \subseteq \{0,1\}^n$ and let $\mu := |T|/2^n$. Then, for any $\varepsilon, \eta > 0$, and $k \geq \frac{A}{\varepsilon^4 \eta^2} \cdot \log\left(\frac{1}{\varepsilon\eta}\right)$ where $A > 0$ is a large enough absolute constant, we have*

$$\Pr_{\mathbf{a},h}\left[\left|\frac{|T \cap \mathrm{C}|}{2^k} - \mu\right| \geq \varepsilon\right] < \eta.$$

3 LOCAL CORRECTION IN THE UNIQUE DECODING REGIME

In this section, we will prove Theorem 1.1, i.e. we will give a local correction algorithm for degree 1 polynomials with only $\tilde{O}(\log n)$ queries.

We will do this in three steps. We start by proving a slightly weaker statement: we will first give a local correction algorithm that can correct $\mathcal{P}_1$ with the error-parameter $\delta \leq 1/O(\log n)$ (see Theorem 3.1). Then we will show how to handle some $\delta = \Omega(1)$ by reducing to the small error case. Finally, by using a similar argument to the second step, we prove Theorem 1.1, which is a local correction algorithm with δ arbitrarily close to the unique decoding radius.

The first part of this argument works only for linear polynomials, while the latter two reductions also work for higher degree.

3.1 Sub-Constant Error

In this section, we give a correction algorithm for $\mathcal{P}_1$ that can correct for $\delta < O(1/\log n)$. The main result of this section is the following.

Theorem 3.1 (Local correction algorithms for $\mathcal{P}_1$ up to error $O(1/\log n)$). *Let $\mathcal{P}_1$ be the set of degree 1 polynomials from $\{0,1\}^n$ to G. Then $\mathcal{P}_1$ has a (δ, q)-local correction algorithm for any $\delta < O(1/\log n)$ and $q = O(\log n)$.*

We first describe the general framework of the algorithm, which is applicable more generally.

3.1.1 *Framework of Local Correction Algorithm.*

We will now give a formal definition of how we construct a local correction algorithm, namely, via a *correction gadget*. This will be useful in the regime where the distance of the input function to the codeword (in our case, a linear polynomial) is small.

Let $\mathcal{F}$ be a class of functions from $\{0,1\}^n$ to an Abelian group G. Let $P_1, \ldots, P_D$ be functions from $\{0,1\}^n$ to $\mathbb{Z}$ satisfying the following property: for any $P \in \mathcal{F}$, there exist coefficients $\alpha_1, \ldots, \alpha_D \in G$ such that for any $\mathbf{a} \in \{0,1\}^n$

$$P(\mathbf{a}) = \alpha_1 P_1(\mathbf{a}) + \ldots + \alpha_D P_D(\mathbf{a}).$$

In the case when $G = \mathbb{F}_p$ for a prime p and $\mathcal{F}$ is a vector space of functions, $\{P_1, \ldots, P_D\}$ is a standard spanning set for $\mathcal{F}$ in the linear algebraic sense. We extend this definition to this case and say that $\{P_i, \ldots, P_D\}$ is a spanning set for $\mathcal{F}$.

Definition 3.2 (Local Correction Gadget). Let $\mathcal{F}$ be a set of functions from $\{0,1\}^n$ to an Abelian group G with spanning set $\{P_i\}_{i \in [D]}$. For any $\mathbf{a} \in \{0,1\}^n$, an (ε, q)-correction gadget for $\mathbf{a}$ is a distribution $\mathcal{D}$ over $(\{0,1\}^n)^q$ satisfying the following two properties:

(1) There exists $c_1, \ldots, c_q \in \mathbb{Z}$ such that for any $(\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(q)}) \in \mathrm{supp}(\mathcal{D})$, the following holds true for each element of the spanning set P_j ($j \in [D]$).

$$P_j(\mathbf{a}) = c_1 P_j(\mathbf{y}^{(1)}) + \ldots + c_q P_j(\mathbf{y}^{(q)}) \tag{3}$$

(2) For any $i \in [q]$, the distribution of $\mathbf{y}^{(i)}$ is ε-close to U_n.

The next claim shows that if we have an (ε, q)-correction gadget for sufficiently small ε, that immediately gives us a (δ, q)-local correction algorithm for small enough δ. We will use the same notation in Definition 3.2. We omit the easy proof, referring the interested reader to the full version.

Lemma 3.3 (Correction gadget gives local correction algorithm). *If there is an (ε, q)-correction gadget for any $\mathbf{a} \in \{0,1\}^n$ where $q(\delta + \varepsilon) < 1/4$, then there is a (δ, q)-local correction algorithm for $\mathcal{F}$.*

3.1.2 *Local Correction Algorithm for Linear Polynomials.*

We now prove Theorem 3.1. The main technical step in the proof of this theorem is the proof of the following lemma.

Lemma 3.4 (Correction gadget for 1^n). *Fix any odd positive integer q. For any n, there is a choice of $c_1, \ldots, c_q \in \mathbb{Z}$ and a distribution $\mathcal{D}$ over $(\{0,1\}^n)^q$ such that the following properties hold for $c_1, \ldots, c_q$ and any sample $(\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(q)})$ from $\mathcal{D}$.*

- *$c_1 + \ldots + c_q = 1$ and for all $i \in [n]$, $c_1 y_i^{(1)} + \ldots + c_q y_i^{(q)} = 1$.*
- *For each $j \in [q]$, $\mathbf{y}^{(j)}$ is $(1/2^{\Omega(q)} \cdot \sqrt{n})$-close to the U_n.*

We first show how to prove Theorem 3.1 assuming this lemma. The lemma is proved subsequently.

Proof of Theorem 3.1. The space $\mathcal{P}_1$ of linear polynomials over G has as a spanning set the constant function $P_0(\mathbf{x}) = 1$ and the co-ordinate functions $P_j(\mathbf{x}) = x_j$ for each $j \in [n]$.

From Lemma 3.3, it suffices to give a (ε, q)-correction gadget for any $\mathbf{a} \in \{0,1\}^n$, where $\varepsilon = 1/n$. Note that Lemma 3.4 directly yields a correction gadget $\mathcal{D}$ at the point 1^n for $q = O(\log n)$.

To get a correction gadget at a point $\mathbf{a} \neq 1^n$, we simply shift this correction gadget by $\mathbf{b} = 1^n \oplus \mathbf{a}$ and use the fact that the space of linear polynomials is preserved by such shifts.

More precisely, consider the distribution $\mathcal{D}_\mathbf{b}$ obtained by sampling $(\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(q)})$ from $\mathcal{D}$ and shifting each element by $\mathbf{b}$ to get

$$(\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(q)}) = (\mathbf{y}^{(1)} \oplus \mathbf{b}, \ldots, \mathbf{y}^{(q)} \oplus \mathbf{b}).$$

We retain the same coefficients $c_1, \ldots, c_q$ as in Lemma 3.4.

To prove that $(\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(q)})$ is an (ε, q)-correction gadget for $\mathbf{a}$, it remains to verify

$$c_1 + \cdots + c_q = 1$$
$$c_1 z_i^{(1)} + \cdots + c_q z_i^{(q)} = a_i \text{ for each } i \in [n] \tag{4}$$

The first of the above follows from Lemma 3.4. The second equality Equation (4) is also easily verified for i such that $a_i = 1$ since $z_i^{(j)} = y_i^{(j)}$ in this case. For i such that $a_i = 0$, we see that $z_i^{(j)} = 1 - y_i^{(j)}$ for each $j \in [q]$ and hence

$$\sum_{j \in [q]} c_j z_i^{(j)} = \sum_{j \in [q]} c_j - \sum_{j \in [q]} c_j y_i^{(j)} = 1 - 1 = 0 = a_i.$$

We have thus shown that Equation (4) holds for all $i \in [n]$. Further, since $\mathbf{y}^{(j)}$ is $1/n$-close to uniform for each $j \in [q]$, so is $\mathbf{z}^{(j)}$. Overall, this implies that $\mathcal{D}_\mathbf{b}$ is a correction gadget for $\mathbf{a}$.

Lemma 3.3 then gives us the desired local correction algorithm. □

3.1.3 Proof of Lemma 3.4. We first construct a Boolean matrix with some interesting combinatorial and algebraic properties. The distribution $\mathcal{D}$ in Lemma 3.4 is obtained later by sampling n rows of this matrix independently and uniformly at random.

The technical lemma below shows that we can find a small number of nearly balanced-Boolean vectors, whose integer span contains the all 1s vector.

Lemma 3.5 (Construction of a matrix). *For any natural number k, there exists an integer matrix A_k of dimension $(2^k - 1) \times (2k - 1)$ with entries in $\{0,1\}$ and a vector $\mathbf{c} \in \mathbb{Z}^{2k-1}$ such that $A_k \mathbf{c} = 1^{2^k - 1}$ and there is exactly one row in A_k that is $(1, \ldots, 1)$. Additionally, for any column of A_k, the Hamming weight of the column is in $[2^{k-1} - 1, 2^{k-1} + 1]$.*

Remark 1. *The statement of this lemma is, in some sense, the best that we can hope for as the lemma does not hold if each column is required to be* perfectly balanced. *In fact, the above lemma does not hold even in the setting where each column is required to have weight* exactly *w for some $w < 2^k - 1$: in this case, 1^{2^k-1} would not even be in the $\mathbb{Q}$-linear span of the columns of A_k (Consider a vector $\mathbf{v} \in \mathbb{Q}^{2^k-1}$ the entries of which are $1 - 1/w$ or $-1/w$, depending on whether the corresponding row in A_k is the all 1s row or not. The vector $\mathbf{v}$ is orthogonal to the columns of A_k but not the vector 1^{2^k-1}).*

Quantitatively, this lemma exhibits a near-tight converse to a lemma of Bafna, Srinivasan, and Sudan [5] who showed that for any $n \times k$ Boolean matrix with an all-1s row, and columns that have Hamming weights in the range $[n/2 - \sqrt{n}, n/2 + \sqrt{n}]$ and also span the all 1s column, we must have $k = \tilde{\Omega}(\log n)$.

Proof. Fix a $k \in \mathbb{N}$. Given a non-negative integer $i < 2^k$, we denote by $\mathrm{bin}(i)$ the Boolean vector that denotes the k-bit binary expansion of i (with the first entry being the most significant bit).

Defining the Base Matrix. Let M be a $(2^k - 1) \times 2k$ matrix with entries in $\{0,1\}$. For all $i \in [2^k - 1]$ and $j \in [2k]$, let the i^{th} row and the j^{th} column of M be denoted by $\mathrm{row}^{(i)}$ and $\mathrm{col}^{(j)}$, respectively. The i^{th} row of M is $\mathrm{row}^{(i)} := (\mathrm{bin}(i)\,\mathrm{bin}(i-1))$, i.e. in $\mathrm{row}^{(i)}$, the first k coordinates are $\mathrm{bin}(i)$ and the next k entries are $\mathrm{bin}(i-1)$, where for an integer i, $\mathrm{bin}(i)$ denotes its binary representation.

$$M = \begin{bmatrix} \vdots & \vdots \\ \mathrm{bin}(i) & \mathrm{bin}(i-1) \\ \vdots & \vdots \end{bmatrix}_{(2^k-1) \times 2k}$$

Let $\mathbf{w} \in \mathbb{R}^{2k}$ be the following vector:

$$\mathbf{w} = \left(2^{k-1}, \ldots, 2^1, 2^0,\ -2^{k-1}, \ldots, -2^1, -2^0\right)$$

It is easy to see that for any row $\mathrm{row}^{(i)}$ of M, $\langle \mathrm{row}^{(i)}, \mathbf{w} \rangle = i - (i-1) = 1$. Thus, $M\mathbf{w} = 1^{2^k-1}$. Observe that $\mathrm{col}^{(k)} = 1^{2^k-1} - \mathrm{col}^{(2k)}$.

Modifying the Base Matrix. Let $\tilde{M}$ be a $(2^k - 1) \times 2k$ matrix and $\tilde{\mathbf{w}}$ be a column vector of dimension $2k$. Let the i^{th} row and the j^{th} column of $\tilde{M}$ be denoted by $\widetilde{\mathrm{row}}^{(i)}$ and $\widetilde{\mathrm{col}}^{(j)}$, respectively. $\tilde{M}$ and $\tilde{\mathbf{w}}$ are defined as follows:

$$\widetilde{\mathrm{col}}^{(j)} = \begin{cases} 1 - \mathrm{col}^{(j)}, & \text{if } j \neq k \\ \mathrm{col}^{(j)}, & \text{if } j = k \end{cases} \qquad \tilde{w}_j = \begin{cases} w_j, & \text{if } j \neq k \\ -w_j, & \text{if } j = k \end{cases}$$

It is easy to verify the following: for any $i \in [2^k - 1]$, $\langle \widetilde{\mathrm{row}}^{(i)}, \tilde{\mathbf{w}} \rangle = -2$. Thus $\tilde{M}(-\tilde{\mathbf{w}}/2) = 1^{2^k-1}$.

From the observation made above, $\widetilde{\mathrm{col}}^{(k)} = \widetilde{\mathrm{col}}^{(2k)}$. The first row of $\tilde{M}$. i.e. $\widetilde{row}^{(1)} = (1, \ldots, 1)$. This implies that $\sum_{j=1}^{2k} (-\tilde{w}_j/2) = 1$. It's also easy to verify that no row other than the first row of $\tilde{M}$ is $(1, 1, \ldots, 1)$.

Integral Coefficients. We have $-\tilde{w}_k/2 = -\tilde{w}_{2k}/2 = 1/2$. For any i, since $\widetilde{\mathrm{row}}_k^{(i)} = \widetilde{\mathrm{row}}_{2k}^{(i)}$, the following equality holds:

$$\widetilde{\mathrm{row}}_k^{(i)}(-\tilde{w}_k/2) + \widetilde{\mathrm{row}}_{2k}^{(i)}(-\tilde{w}_{2k}/2) = \widetilde{\mathrm{row}}_k^{(i)} \cdot 1 + \widetilde{\mathrm{row}}_{2k}^{(i)} \cdot 0 \tag{5}$$

Let $\mathbf{c} \in \mathbb{Z}^{2k-1}$ be the following vector: $c_j = (-\tilde{w}_j/2)$ if $j \neq k$, otherwise $c_j = 1$. For any row $\widetilde{\text{row}}^{(i)}$, from Equation (5), $\langle \widetilde{\text{row}}^{(i)}, \mathbf{c} \rangle = \langle \widetilde{\text{row}}^{(i)}, (-\tilde{\mathbf{w}}/2) \rangle = 1$. Let A_k denote the matrix $\tilde{M}$ after removing the $2k^{th}$ column. Then $A_k \mathbf{c} = 1^{2^k-1}$.
Since $\sum_{j=1}^{2k} (-\tilde{w}_j/2) = 1$, using Equation (5), we get that $\sum_{j=1}^{2k-1} c_j = 1$.

Columns are Nearly Balanced. Finally, we will prove that for each column $\widetilde{\text{col}}^{(j)}$ of A, the Hamming weight of $\widetilde{\text{col}}^{(j)} \in [2^{k-1} - 1, 2^{k-1} + 1]$. For any $j \in [2k-1]$, the Hamming weight of $\text{col}^{(j)}$ is in $\left\{2^{k-1} - 1, 2^{k-1} + 1\right\}$. This is because if M had an additional row $[\text{bin}(0)\text{bin}(2^k - 1)]$, then each column of M would be exactly balanced, i.e. have Hamming weight of 2^{k-1}. Then by definition of $\widetilde{\text{col}}^{(j)}$, it follows that Hamming weight of each column of A_k is also in $[2^{k-1} - 1, 2^{k-1} + 1]$. □

Next, we are going to describe a distribution $\mathcal{D}$ on $(\{0,1\}^m)^q$, where $m = 2^k - 1$ and $q = 2k - 1$. We will do this by randomly sampling rows of the matrix A_k given by Lemma 3.5.

Proof of Lemma 3.4. Assume that $q = 2k - 1$. To sample $(\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(q)}) \sim \mathcal{D}$ over $(\{0,1\}^n)^q$, we sample n rows independenly and uniformly at random from the rows of A_k as constructed in Lemma 3.5 and define $(y_i^{(1)}, \ldots, y_i^{(q)})$ to be the ith sample for each $i \in [n]$.

We now show that $\mathcal{D}$ has the required properties from the statement of Lemma 3.4. Let $(c_1, \ldots, c_q) = \mathbf{c}$ be as guaranteed by Lemma 3.5. The first property holds from the properties of A_k and $\mathbf{c}$. For each $i \in [n]$, the vector $(y_i^{(1)}, \ldots, y_i^{(q)})$ is a row of A_k, and from Lemma 3.5, we know that the inner product of any row of A_k and $\mathbf{c}$ is 1. Further, since 1^q is also a row of A_k, it follows that the entries of $\mathbf{c}$ sum to 1.

The second property follows from the fact that each column of A_k has relative Hamming weight in the range $[\frac{1}{2} - 2^{-k}, \frac{1}{2} + 2^{-k}]$. Thus, for any fixed $j \in [q]$ and each $i \in [n]$, we have that $|\Pr[y_i^{(j)} = 1] - 1/2| \leq 1/2^k$.
Since for a fixed $j \in [q]$ the bits $\{y_i^{(j)} \mid i \in [n]\}$ are mutually independent, we are now done by the following standard fact (which can easily be proved by, say, following the proof of [25, Theorem 5.5, Claim 5.6]).

Lemma 3.6. *Let $\eta > 0$. Let $\mathcal{D}'$ be a distribution on $\{0,1\}^n$ such that for any $\mathbf{y} \sim \mathcal{D}'$, the co-ordinates of $\mathbf{y}$ are independent and for all $i \in [n]$, $1/2 - \eta \leq \Pr[y_i = 1] \leq 1/2 + \eta$. Then $\mathcal{D}'$ is $O(\eta\sqrt{n})$-close to U_n.*

This concludes the proof of Lemma 3.4. □

3.2 Constant-Error Algorithm via Error-Reduction

We now show how to locally correct degree-1 polynomials in the regime of *constant* error. We will do this by reducing the problem to the case of low error. The results of this section also work for higher-degree polynomials.

We will show that there is a randomized algorithm $\mathcal{A}^f$ that given oracle access to any function f that is δ-close to a degree-d polynomial P (think of δ as being a small enough constant depending on d), has the following property: with high probability over the internal randomness of $\mathcal{A}^f$, the function computed by $\mathcal{A}^f$ is η-close to P, where $\eta < \delta$. We state it formally below.

Lemma 3.7 (Error reduction for constant error). *Fix any Abelian group G and a positive integer d. The following holds for $\delta < 1/2^{O(d)}$ and $K = 2^{O(d)}$ where the $O(\cdot)$ hides a large enough absolute constant.*
For any η, δ, where $\eta < \delta$, there exists a randomized algorithm $\mathcal{A}$ with the following properties: Let $f : \{0,1\}^n \to G$ be a function and let $P : \{0,1\}^n \to G$ be a degree-d polynomial such that $\delta(f, P) \leq \delta$, and let $\mathcal{A}^f$ denotes that $\mathcal{A}$ has oracle access to f.
Then $\Pr[\delta(\mathcal{A}^f, P) > \eta] < 1/10$, where the probability is over the internal randomness of $\mathcal{A}^f$. Further, for every $\mathbf{x} \in \{0,1\}^n$, $\mathcal{A}^f$ makes K^T oracle queries to f and $T = O\left(\log\left(\frac{\log(1/\eta)}{\log(1/\delta)}\right)\right)$.

Putting this together with Theorem 3.1, we immediately get a local corrector that can correct errors up to a small enough constant with $\tilde{O}(\log n)$ queries.

In the rest of this subsection, we will prove Lemma 3.7. The algorithm $\mathcal{A}^f$ in Lemma 3.7 will be a recursive algorithm. Each recursive iteration of the algorithm $\mathcal{A}^f$ uses the same 'base algorithm' $\mathcal{B}$, which will be the core of our error reduction algorithm from small constant error. In the next lemma, we formally state the properties of the base algorithm.

Lemma 3.8 (Base Error Reduction Algorithm). *Fix any Abelian group G and a positive integer d. The following holds for $K = 2^{O(d)}$. For any $0 < \gamma < 1$, there exists a randomized algorithm $\mathcal{B}$ with the following properties: Let $g : \{0,1\}^n \to G$ be a function and let $P : \{0,1\}^n \to G$ be a degree-d polynomial such that $\delta(g, P) \leq \gamma$, and let $\mathcal{B}^g$ denotes that $\mathcal{B}$ has oracle access to g.*
Then $\mathbb{E}[\delta(\mathcal{B}^g, P)] < O(K^2) \cdot \gamma^{1.5}$, where the expectation is over the internal randomness of $\mathcal{B}$. Further, for every $\mathbf{x} \in \{0,1\}^n$, $\mathcal{A}^g$ makes K queries to g.

We defer the construction of the base algorithm and proof of Lemma 3.8 to the next subsection, Section 3.2.1. For now, we assume Lemma 3.8 and proceed to describe the recursive construction of $\mathcal{A}^f$ and prove Lemma 3.7.

Proof of Lemma 3.7. Let $\mathcal{B}$ be the algorithm given by Lemma 3.8. We define a sequence of algorithms $\mathcal{A}_0^f, \mathcal{A}_1^f, \ldots$, as follows (see the boxed text in the next page).

An easy inductive argument shows that $\mathcal{A}^f$ makes at most K^T queries to f. The error probability can be upper bounded by an inductive argument. We will argue inductively that for each $t \leq T$ and $\delta_t := \delta^{(1.1)^t}$, we have

$$\Pr_{\sigma_t}[\underbrace{\delta(\mathcal{A}_t^f(\cdot, \sigma_t), P) > \delta_t}_{:= \mathcal{E}_t}] \leq \sum_{j=1}^{t} \frac{1}{100^j} < \frac{1}{10}. \tag{6}$$

Due to space constraints, we skip the analysis and refer the reader to the full version for a detailed proof.

The algorithm $\mathcal{A}_t^f$ computes a function mapping inputs in $\{0,1\}^n$ along with a uniformly random string from $\{0,1\}^{r_t}$ to a random group element in G.

- $\mathcal{A}_0^f$ just computes the function f. (In particular, $r_0 = 0$.)
- For each $t > 0$, we inductively define $r_t = r_{t-1} + r$, where r is the amount of randomness required by the base error reduction algorithm $\mathcal{B}$. On input $\mathbf{x}$ and random string $\sigma_t \sim U_{r_t}$, the algorithm $\mathcal{A}_t^f$ algorithm runs the algorithm $\mathcal{B}$ on $\mathbf{x}$ using the first r bits of σ_t as its source of randomness, and with oracle access to $\mathcal{A}_{t-1}^f$ using the remaining r_{t-1} bits of σ_t as randomness.

The algorithm $\mathcal{A}^f$ will be $\mathcal{A}_T^f$ for $T = C \cdot \log\left(\frac{\log(1/\eta)}{\log(1/\delta)}\right)$ where C is a large enough absolute constant chosen below.

□

Thus we have shown so far that given the base algorithm $\mathcal{B}$, we do get an error reduction algorithm from small constant error to error $O(1/\log n)$. Now it remains to describe the base error reduction algorithm. In the next subsection, we describe the base algorithm $\mathcal{B}$ and prove Lemma 3.8.

3.2.1 The Base Algorithm and Its Analysis. In this section, we prove Lemma 3.8, which will then complete the proof of Lemma 3.7. Before we describe $\mathcal{B}$, we will define an *error reduction gadget*, which is a variant of the local correction gadget defined above (Definition 3.2).

Definition 3.9 (Error-reduction Gadget for $\mathcal{P}_d$). For $\rho \in (0,1)$, an (ρ, q)-error reduction gadget for $\mathcal{P}_d$ is a distribution $\mathcal{D}$ over $(\{0,1\}^n)^q$ satisfying the following two properties:

(1) There exists $c_1, \ldots, c_q \in \mathbb{Z}$ such that for any $(\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(q)}) \in \mathrm{supp}(\mathcal{D})$, the following holds true for each $P \in \mathcal{P}_d$ and each $\mathbf{a} \in \{0,1\}^n$
$$P(\mathbf{a}) = c_1 P(\mathbf{a} \oplus \mathbf{y}^{(1)}) + \ldots + c_q P(\mathbf{a} \oplus \mathbf{y}^{(q)}). \tag{7}$$
(2) For any $i \in [q]$, the bits of $\mathbf{y}^{(i)}$ are i.i.d. Bernoulli random variables that are ρ-close to uniform. Equivalently, each co-ordinate is 1 with probability $p_i \in [\frac{1-\rho}{2}, \frac{1+\rho}{2}]$.

To prove Lemma 3.8, we need an error-reduction gadget for $\mathcal{P}_d$, the space of degree-d polynomials over a group G. This is given by the following lemma.

Lemma 3.10 (Constructing an error-reduction gadget for $\mathcal{P}_d$). *Fix any Abelian group G and any $\rho > 0$. Then $\mathcal{P}_d$ has a (ρ, q)-error-reduction gadget where $q = 2^{O(d/\rho)}$.*

Assuming the above lemma, we first finish the proof of Lemma 3.8.

In the algorithm, we use the error-reduction gadget to correct the polynomial at a *random point* $\mathbf{a} \in \{0,1\}^n$. This process is likely to give the right answer except with probability $q\gamma$ since, after shifting, each query is now *uniformly* distributed and hence the chance that any of the queried points is an error point of g is at most γ. We reduce the error by repeating this process three times and taking a majority vote. To analyze this algorithm, we need to understand the probability that two iterations of this process both evaluate g at an error point. We do this using hypercontractivity (more specifically Theorem 2.6).

Proof of Lemma 3.8. Let $\mathcal{D}$ be a $(1/10, q)$-error-reduction gadget as given by Lemma 3.10. The algorithm $\mathcal{B}$, given oracle access to $g : \{0,1\}^n \to G$ and $\mathbf{a} \in \{0,1\}^n$, does the following.

- Repeat the following three times independently. Sample $(\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(q)})$ from $\mathcal{D}$ and compute
$$c_1 g(\mathbf{a} \oplus \mathbf{y}^{(1)}) + \cdots + c_q g(\mathbf{a} \oplus \mathbf{y}^{(q)})$$
where $c_1, \ldots, c_q$ are the coefficients corresponding to the error-reduction gadget.
- Output the plurality among the three group elements b_1, b_2, b_3 computed above.

The number of queries made by the algorithm is $K = O(q) = 2^{O(d)}$ as claimed. So it only remains to analyze $\delta(\mathcal{B}^g, P)$. From now on, let $\mathbf{a}$ be a uniformly random input in $\{0,1\}^n$.

For $i \in \{1,2,3\}$, let $\mathcal{E}_i$ denote the event that $b_i \neq P(\mathbf{a})$. We have
$$\mathbb{E}[\delta(\mathcal{B}^g, P)] = \Pr[\mathcal{B}^g(\mathbf{a}) \neq g(\mathbf{a})]$$
$$\leq \Pr[\mathcal{E}_1 \wedge \mathcal{E}_2] + \Pr[\mathcal{E}_2 \wedge \mathcal{E}_3] + \Pr[\mathcal{E}_1 \wedge \mathcal{E}_3]$$
Thus it suffices to show that each of the three terms in the final expression above is at most $O(q^2) \cdot \gamma^{1.5}$.

Without loss of generality, consider the event $\mathcal{E}_1 \wedge \mathcal{E}_2$. Let $(\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(q)})$ and $(\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(q)})$ be the two independent samples from $\mathcal{D}$ in the two corresponding iterations.

It follows from Equation (7) that the algorithm correctly computes $P(\mathbf{a})$ in the first iteration as long as none of the queried points lie in the set T of points where g and P differ. A similar statement also holds for the second iteration. This reasoning implies that
$$\Pr[\mathcal{E}_1 \wedge \mathcal{E}_2] \leq \sum_{i,j=1}^{q} \Pr[\underbrace{\mathbf{a} \oplus \mathbf{y}^{(i)}}_{\mathbf{u}^{(i)}} \in T \wedge \underbrace{\mathbf{a} \oplus \mathbf{z}^{(j)}}_{\mathbf{v}^{(j)}} \in T]. \tag{8}$$
We bound the latter expression using Theorem 2.6.

Fix $i, j \in [q]$. Note that for every fixing of $\mathbf{y}^{(i)}$, the vector $\mathbf{u}^{(i)}$ is distributed uniformly over $\{0,1\}^n$ (because $\mathbf{a}$ is uniform over $\{0,1\}^n$). In particular, this implies that $\mathbf{u}^{(i)}$ is uniformly distributed and moreover that $\mathbf{u}^{(i)}$ and $\mathbf{y}^{(i)}$ are independent random variables. This means that $\mathbf{v}^{(j)} = \mathbf{u}^{(i)} \oplus \mathbf{y}^{(i)} \oplus \mathbf{z}^{(j)}$ is drawn from the noise distribution $\mathcal{N}_\rho(\mathbf{u}^{(i)})$. Further, the parameter $\rho \leq 1/100$ since the co-ordinates of $\mathbf{y}^{(i)}$ and $\mathbf{z}^{(j)}$ are i.i.d. Bernoulli random variables that are each $1/10$-close to uniform.

Using Theorem 2.6, we have
$$\Pr[\mathbf{u}^{(i)} \in T \wedge \mathbf{v}^{(j)} \in T] \leq \gamma^{2/1+|\rho|} \leq \gamma^{1.5}.$$
Plugging this into Equation (8) implies the required bound on the probability of $\mathcal{E}_1 \wedge \mathcal{E}_2$. This concludes the analysis of $\mathcal{B}$. □

We now show how to construct the error-reduction gadget and prove Lemma 3.10. This requires the following standard claim (implied e.g. by Möbius inversion) that shows that any degree-d polynomial over $\{0,1\}^n$ (even with group coefficients) can be interpolated from its values on a Hamming ball of radius d. We omit the proof.

Lemma 3.11. *Fix $d \in \mathbb{N}$. For any natural number $m \geq d$ and any Hamming ball B of radius d,*

$$P(0^m) = \sum_{\mathbf{b} \in B} \alpha_{\mathbf{b}} P(\mathbf{b})$$

where the $\alpha_{\mathbf{b}}$ are integer coefficients.

We end this section by completing the proof of Lemma 3.10.

Proof of Lemma 3.10. The idea is to apply Lemma 3.11 on a random subcube, as defined in Definition 2.7. More precisely, for an even integer $k > 2d$ that we will fix below, let $\mathbf{a} \in \{0,1\}^n$ be arbitrary and let $h : [n] \to [k]$ be chosen uniformly at random. Let $C = C_{\mathbf{a},h}$ be the corresponding subcube of $\{0,1\}^n$. Let $Q(y_1, \ldots, y_k)$ denote $P|_C$, the restriction of P to this subcube.

Fix a Hamming ball B of radius d in $\{0,1\}^k$ centred at a point $\mathbf{c}$ of weight exactly $k/2$. Since Q is a polynomial of degree at most d, applying Lemma 3.11 to Q and the ball B yields an equality

$$Q(0^k) = \sum_{\mathbf{b} \in \mathbf{B}} \alpha_{\mathbf{b}} Q(\mathbf{b}) \implies P(x(0^k)) = \sum_{\mathbf{b} \in \mathbf{B}} \alpha_{\mathbf{b}} P(x(\mathbf{b})),$$

where in the implication we used that Q is a restriction of P. From the definition of the cube C, it follows that $x(0^k) = \mathbf{a}$ and thus the above gives us an equality of the type desired in an error-reduction gadget (Equation (7)). To finish the proof, we only need to argue that each $x(\mathbf{b})$ has the required distribution.

Note that for each $\mathbf{b} \in B$, we have, $x(\mathbf{b}) = \mathbf{a} \oplus \mathbf{b}_h$, where $\mathbf{b}_h$ is the random vector in $\{0,1\}^n$ that at co-ordinate i takes the random value $b_{h(i)}$. Since h is chosen uniformly at random, it follows that the entries of $\mathbf{b}_h$ are independent and the ith co-ordinate is a Bernoulli random variable that takes the value 1 with probability equal to the relative Hamming weight of $\mathbf{b}$.

To conclude the argument, note that $\mathbf{b}$ is at Hamming distance at most d from $\mathbf{c}$, implying that it has relative Hamming weight in $[1/2 - (2d/k), 1/2 + (2d/k)]$.
Setting k larger than $4d/\rho$ gives us the desired value for the parameter of the Bernoulli distribution. Setting $k = O(d/\rho)$ gives us the query complexity as claimed. □

To prove Theorem 1.1, we need another error-reduction algorithm, which builds on the ideas presented in this section. We defer the statement and the proof to the full version [2].

REFERENCES

[1] Noga Alon and Văn H. Vũ. 1997. Anti-Hadamard Matrices, Coin Weighing, Threshold Gates, and Indecomposable Hypergraphs. *J. Comb. Theory Ser. A* 79, 1 (1997), 133–160. https://doi.org/10.1006/jcta.1997.2780

[2] Prashanth Amireddy, Amik Raj Behera, Manaswi Paraashar, Srikanth Srinivasan, and Madhu Sudan. 2024. Local Correction of Linear Functions over the Boolean Cube . *Electron. Colloquium Comput. Complex.* TR24-056 (2024). ECCC:TR24-056 https://eccc.weizmann.ac.il/report/2024/056/

[3] Prashanth Amireddy, Srikanth Srinivasan, and Madhu Sudan. 2023. Low-Degree Testing over Grids. In *Approximation, Randomization, and Combinatorial Optimization (RANDOM)*, Vol. 275. 41:1–41:22. https://doi.org/10.4230/LIPICS.APPROX/RANDOM.2023.41

[4] László Babai, Kristoffer Arnsfelt Hansen, Vladimir V. Podolskii, and Xiaoming Sun. 2010. Weights of exact threshold functions. *Izvestiya: Mathematics* 85 (2010), 1039 – 1059. https://api.semanticscholar.org/CorpusID:7248898

[5] Mitali Bafna, Srikanth Srinivasan, and Madhu Sudan. 2020. Local decoding and testing of polynomials over grids. *Random Struct. Algorithms* 57, 3 (2020), 658–694. https://doi.org/10.1002/rsa.20933

[6] Boaz Barak, Zeev Dvir, Amir Yehudayoff, and Avi Wigderson. 2011. Rank Bounds for Design Matrices with Applications to Combinatorial Geometry and Locally Correctable Codes. In *ACM Symposium on Theory of Computing (STOC)*. 519–528. https://doi.org/10.1145/1993636.1993705

[7] Donald Beaver and Joan Feigenbaum. 1990. Hiding Instances in Multioracle Queries. In *Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, Vol. 415. 37–48. https://doi.org/10.1007/3-540-52282-4_30

[8] Abhishek Bhowmick and Shachar Lovett. 2018. The List Decoding Radius for Reed-Muller Codes Over Small Fields. *IEEE Trans. Inf. Theory* 64, 6 (2018), 4382–4391. https://doi.org/10.1109/TIT.2018.2822686

[9] Irit Dinur, Elena Grigorescu, Swastik Kopparty, and Madhu Sudan. 2008. Decodability of Group Homomorphisms beyond the Johnson Bound. *Electron. Colloquium Comput. Complex.* TR08-020 (2008). ECCC:TR08-020 https://eccc.weizmann.ac.il/eccc-reports/2008/TR08-020/index.html

[10] Zeev Dvir, Shubhangi Saraf, and Avi Wigderson. 2014. Improved rank bounds for design matrices and a new proof of Kelly's theorem. *Forum Math. Sigma* 2 (2014), Paper No. e4, 24. https://doi.org/10.1017/fms.2014.2

[11] Zeev Dvir, Shubhangi Saraf, and Avi Wigderson. 2017. Superquadratic lower bound for 3-query locally correctable codes over the reals. *Theory Comput.* 13 (2017), Paper No. 11, 36. https://doi.org/10.4086/toc.2017.v013a011

[12] Peter Elias. 1957. List decoding for noisy channels. *Technical Report 335, Research Laboratory of Electronics, MIT* (1957).

[13] Peter Gemmell, Richard J. Lipton, Ronitt Rubinfeld, Madhu Sudan, and Avi Wigderson. 1991. Self-Testing/Correcting for Polynomials and for Approximate Functions. In *ACM Symposium on Theory of Computing (STOC)*. 32–42. https://doi.org/10.1145/103418.103429

[14] Peter Gemmell and Madhu Sudan. 1992. Highly Resilient Correctors for Polynomials. *Inf. Process. Lett.* 43, 4 (1992), 169–174. https://doi.org/10.1016/0020-0190(92)90195-2

[15] Mikael Goldmann, Johan Håstad, and Alexander A. Razborov. 1992. Majority gates vs. general weighted threshold gates. *Computational Complexity* 2 (1992), 277–300. https://api.semanticscholar.org/CorpusID:17637868

[16] Oded Goldreich and Leonid A. Levin. 1989. A Hard-Core Predicate for all One-Way Functions. In *ACM Symposium on Theory of Computing (STOC)*. 25–32. https://doi.org/10.1145/73007.73010

[17] Parikshit Gopalan, Adam R. Klivans, and David Zuckerman. 2008. List-Decoding Reed-Muller Codes over Small Fields. In *ACM Symposium on Theory of Computing (STOC)*. 265–274. https://doi.org/10.1145/1374376.1374417

[18] Elena Grigorescu, Swastik Kopparty, and Madhu Sudan. 2006. Local Decoding and Testing for Homomorphisms. In *International Workshop on Approximation Algorithms for Combinatorial Optimization (RANDOM)*, Vol. 4110. 375–385. https://doi.org/10.1007/11830924_35

[19] Venkatesan Guruswami, Atri Rudra, and Madhu Sudan. 2023. *Essential Coding Theory (Book draft)*. http://www.cse.buffalo.edu/atri/courses/coding-theory/book

[20] Venkatesan Guruswami and Madhu Sudan. 1999. Improved decoding of Reed-Solomon and algebraic-geometry codes. *IEEE Trans. Inf. Theory* 45, 6 (1999), 1757–1767. https://doi.org/10.1109/18.782097

[21] Johan Håstad. 1994. On the Size of Weights for Threshold Gates. *SIAM J. Discret. Math.* 7, 3 (1994), 484–492. https://doi.org/10.1137/S0895480192235878

[22] John Y. Kim and Swastik Kopparty. 2017. Decoding Reed-Muller Codes over Product Sets. *Theory Comput.* 13, 1 (2017), 1–38. https://doi.org/10.4086/TOC.2017.V013A021

[23] Eyal Kushilevitz and Yishay Mansour. 1993. Learning Decision Trees Using the Fourier Spectrum. *SIAM J. Comput.* 22, 6 (1993), 1331–1348. https://doi.org/10.1137/0222080 arXiv:https://doi.org/10.1137/0222080

[24] Richard J. Lipton. 1989. New Directions In Testing. In *Distributed Computing And Cryptography*, Vol. 2. 191–202. https://doi.org/10.1090/DIMACS/002/13

[25] Yishay Mansour. 2011. Lecture 5: Lower Bounds using Information Theory Tools. http://www.math.tau.ac.il/~mansour/advanced-agt+ml/scribe5-lower-bound-MAB.pdf. Lecture notes.

[26] Ryan O'Donnell. 2014. *Analysis of Boolean Functions*. Cambridge University Press. https://doi.org/10.1017/CBO9781139814782

[27] Øystein Ore. 1922. Über höhere kongruenzen. *Norsk Mat. Forenings Skrifter* 1, 7 (1922), 15.

[28] Vladimir V. Podolskii. 2009. Perceptrons of large weight. *Probl. Inf. Transm.* 45, 1 (2009), 46–53. https://doi.org/10.1134/S0032946009010062

[29] Irving S. Reed. 1954. A class of multiple-error-correcting codes and the decoding scheme. *Trans. IRE Prof. Group Inf. Theory* 4 (1954), 38–49. https://doi.org/10.1109/TIT.1954.1057465

[30] Madhu Sudan. 1997. Decoding of Reed Solomon Codes beyond the Error-Correction Bound. *J. Complex.* 13, 1 (1997), 180–193. https://doi.org/10.1006/jcom.1997.0439

[31] Madhu Sudan, Luca Trevisan, and Salil P. Vadhan. 2001. Pseudorandom Generators without the XOR Lemma. *J. Comput. Syst. Sci.* 62, 2 (2001), 236–266. https://doi.org/10.1006/JCSS.2000.1730

Received 13-NOV-2023; accepted 2024-02-11