PROBABILISTIC AND POLYNOMIAL METHODS IN ADDITIVE COMBINATORICS AND CODING THEORY

BY JOHN KIM[1]

A dissertation submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Mathematics

Written under the direction of

Swastik Kopparty

and approved by

New Brunswick, New Jersey

May, 2016

ABSTRACT OF THE DISSERTATION

Probabilistic and polynomial methods in additive combinatorics and coding theory

by John Kim[2]

Dissertation Director: Swastik Kopparty

We present various applications of the probabilistic method and polynomial method in additive combinatorics and coding theory.

We first study the effect of addition on the Hamming weight of a positive integer. Consider the first 2^n positive integers, and fix an α among them. We show that if the binary representation of α consists of $\Theta(n)$ blocks of zeros and ones, then addition by α causes a constant fraction of low Hamming weight integers to become high Hamming weight integers. Our result implies that powering by α composed of many blocks requires exponential-size, bounded-depth arithmetic circuits over $\mathbb{F}_2$.

We also prove a version of the Cauchy-Davenport theorem for general linear maps. For subsets A, B of the finite field $\mathbb{F}_p$, the classical Cauchy-Davenport theorem gives a lower bound for the size of the sumset $A + B$ in terms of the sizes of the sets A and B. Our theorem considers a general linear map $L : \mathbb{F}_p^n \to \mathbb{F}_p^m$, and subsets $A_1, \ldots, A_n \subseteq \mathbb{F}_p$, and gives a lower bound on the size of $L(A_1 \times A_2 \times \ldots \times A_n)$ in terms of the sizes of the sets $A_1, \ldots, A_n$. Our proof uses Alon's Combinatorial Nullstellensatz and a variation of the polynomial method.

Lastly, we give a polynomial time algorithm to decode multivariate polynomial codes of degree d up to half their minimum distance, when the evaluation points are an arbitrary product set S^m, for every $d < |S|$. Previously known algorithms can achieve this only if the set S has some very special algebraic structure, or if the degree d is significantly smaller than $|S|$. We also give a near-linear time algorithm, which is based on tools from list-decoding, to decode these codes from nearly half their minimum distance, provided $d < (1 - \epsilon)|S|$ for constant $\epsilon > 0$.

Acknowledgements

I would like to thank my advisor Swastik Kopparty, and my collaborator Simao Herdade for many great mathematical discussions. I would also like to thank my committee (Michael Saks, Doron Zeilberger, and Eli Ben-Sasson) for their support.

Dedication

Dedicated to Isabel.

Table of Contents

Chapter 1
Introduction

Combinatorics, the study of discrete structures, has had applications to nearly every field of mathematics. More recently, methods from probability and algebra have had a major impact on the advancement of combinatorics, hinting at a much deeper connection between combinatorics and these mathematical areas. My research explores this connection, uncovering new results in complexity theory, coding theory, and additive combinatorics.

The two main tools we will be using are the probabilistic method and the polynomial method. The probabilistic method is often used to establish the existence of an object with certain desired properties by considering an appropriate random object and showing that it is likely to have (or be close to having) the desired properties. Additionally, we will use the probabilistic method to formulate a problem in terms of distributions of random variables, so that we gain access to the convergence theorems and concentration bounds in probability theory.

On the other hand, polynomials are simple algebraic objects that have found numerous surprising combinatorial applications. The main idea behind many of these applications is to somehow associate complex objects to low-degree polynomials, so that we can take advantage of the nice properties of polynomials. This strategy is known as the polynomial method.

In this thesis, we will use the probabilistic and polynomial methods to get circuit lower bounds, sumset lower bounds, and efficient decoding algorithms for polynomial-based error correcting codes. For circuit lower bounds, we will analyze which powering maps are hard to compute by constant-depth polynomial-size circuits. For sumset lower bounds, we will generalize the Cauchy-Davenport theorem in additive combinatorics.

Lastly, for polynomial-based error correcting codes, we will find polynomial time (near-linear time) algorithms to decode Reed-Muller codes over product sets to half (nearly half) the minimum distance. We now describe these results in more detail.

1.1 Circuit lower bounds

One early example of the polynomial method is the Razborov-Smolensky method for proving circuit lower bounds. The Razborov-Smolensky method is based on expressing complex circuit computations in terms of low-degree polynomials, and using properties of these low-degree polynomials to argue about the power of such computations. The method was first used by Smolensky in 1993 to give another proof of the fact that the majority function, *Maj*, was not representable by a constant-depth, polynomial-size Boolean arithmetic circuit, also known as an $\mathrm{AC}^0(\oplus)$ circuit [11].

There were two key ingredients to his proof. The first was the 1987 result of Razborov establishing that a low-depth, polynomial-size Boolean arithmetic circuit is well-approximated by a low-degree polynomial [8]. The second ingredient was the fact that *Maj* is *versatile* in the sense that *Maj* can be used to easily compute a large class of other functions $\mathcal{F}$ [11]. If *Maj* was indeed representable as an $\mathrm{AC}^0(\oplus)$ circuit, it turns out that the functions in $\mathcal{F}$ are all well-approximated by low-degree polynomials. However, $\mathcal{F}$ is so large that the low-degree polynomials cannot possibly account for every function in $\mathcal{F}$. Hence, *Maj* cannot be an $\mathrm{AC}^0(\oplus)$ circuit.

In Chapter 2, we will use Kopparty's variation of the Razborov-Smolensky method to greatly expand the class of powering maps over the field $\mathbb{F}_{2^n}$ (mapping $x \mapsto x^\alpha$) that are hard for $\mathrm{AC}^0(\oplus)$ circuits [7]. As in the Razborov-Smolensky argument, the main component will be to understand which powering maps are versatile. We will use a sufficient condition of Kopparty's for a powering map to be versatile, which can be written as a condition on the distribution of a 2-dimensional random variable. We will then apply tools from probability theory to understand what we can about the distribution. This will allow us to give a general sufficient for the map $x \mapsto x^\alpha$ to be hard for $\mathrm{AC}^0(\oplus)$ circuits. Our condition will be general enough to establish the hardness of the map $x \mapsto x^\alpha$ for most values of α.

1.2 Sumset lower bounds

We can get other interesting methods for proving lower bounds by exploiting some of the deeper properties of polynomials. In particular, the polynomial method can also be used to prove lower bounds on set sizes by viewing the elements of the set as the zero set of some low-degree or 'low-complexity' polynomial. We can then obtain bounds on the original set by analyzing the zero set of the polynomial.

One popular tool in additive combinatorics for controlling the zero set of a polynomial is the Combinatorial Nullstellensatz, which gives a non-vanishing criterion for a polynomial on a product set. In 1995, Alon, Nathanson, and Ruzsa used the Combinatorial Nullstellensatz in conjunction with the polynomial method to give a short proof of the Cauchy-Davenport theorem, which provides a lower bound on the size of the sum of two subsets $A, B \in \mathbb{F}_p$ in terms of the sizes of A and B [13]. In Chapter 3, we generalize their strategy to prove a lower bound on the image size of a general linear map on a product set (joint work with Simao Herdade and Swastik Kopparty).

1.3 Polynomial-based error-correcting codes

Polynomial-based error-correcting codes are widely used to ensure the reliability of data storage and communication. They allow us to encode information so that even if the encoded information is corrupted by a few errors, the original information can still be recovered via a process known as decoding. Thus, decoding algorithms are important for facilitating faster data access and communication, enabling larger storage capacities, and handling increased amounts of error.

The Reed-Muller code is a very general type of polynomial-based error-correcting code. To encode a message into a codeword, we view the symbols in the message as coefficients of a low-degree m-variate polynomial and evaluate this polynomial on a finite grid of points S^m. Because distinct low-degree polynomials must differ on many of these evaluations, it takes many errors to confuse one codeword with another. This key property allows us to correct errors by simply searching for nearby codewords. Decoding can then be viewed in the following way:

Decoding Problem: Given a received word R, and a maximum allowable number of errors η, find all codewords that are within Hamming distance η from R.

If the maximum allowable number of errors η (also called the decoding radius) is small, then there is a unique codeword (if it exists) that the received word can be decoded to. More precisely, if we define the minimum distance to be the smallest distance between two codewords, then for η smaller than half the minimum distance, there is at most one codeword within η from R. In this case, the decoding problem is known as decoding to half the minimum distance or unique decoding.

Efficient decoding algorithms up to half the minimum distance are only known for special cases of Reed-Muller codes. Arguably the most useful special case is the Reed-Solomon code, whose codewords are evaluations of low-degree single-variable polynomials on a set of inputs of size n. Reed-Solomon codes are well-studied and are decodable in near-linear ($O(n \operatorname{polylog} n)$) time [31]. For multivariate Reed-Muller codes, existing decoding algorithms require either the evaluation set S^m to have special structure or the degree of the polynomials to be very small.

In Chapter 4, we make significant progress towards the efficient decoding of general Reed-Muller codes, where we are free to choose the finite set $S \subseteq \mathbb{F}$ of grid coordinate values (joint work with Swastik Kopparty). In this setting, we discovered a polynomial-time algorithm for decoding Reed-Muller codes to half the minimum distance. Furthermore, when the rate of the code is constant (degree at most $d = \Theta(|S|)$), we improved the algorithm to decode to almost half the minimum distance in near-linear time.

Chapter 2

Integer addition and Hamming weight

2.1 Introduction

We begin with a natural, but largely unstudied question: How does the Hamming weight of an integer (written in base 2) change under addition? To make this precise, we take $0 < \alpha < 2^n$ to be a fixed integer and let S be chosen uniformly at random from $\{0, 1, \cdots, 2^n - 1\}$. Write S in binary, and take X to be its Hamming weight. Let Y be the Hamming weight of the translation $S + \alpha$. Then what can we say about the joint distribution of initial and final weights, (X, Y)?

Our question is motivated by the problem of determining the complexity of powering maps in $\mathbb{F}_{2^n}$. This problem has been studied extensively in complexity theory [7, 2, 3, 4, 5, 6, 9]. Recently, Kopparty [7] showed that the powering map $x \to x^{\frac{1}{3}}$ from $\mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ cannot be computed with a polynomial-size, bounded-depth arithmetic circuit over $\mathbb{F}_2$ (a.k.a $\mathrm{AC}^0(\oplus)$ circuit). Recall that arithmetic circuits are only allowed addition and multiplication gates of unbounded fan-in). A major advantage of working in $\mathrm{AC}^0(\oplus)$ is that it is basis invariant. That is, determining the $\mathrm{AC}^0(\oplus)$ complexity of powering does not depend on the choice of basis for F_{2^n}. Define $\mathbb{Z}_k := \mathbb{Z}/k\mathbb{Z}$. At the core of Kopparty's argument was the following shifting property of $\frac{1}{3}$: a constant fraction of elements in $\mathbb{Z}_{2^n-1}$ change from low to high Hamming weight under translation by $\frac{1}{3}$.

Definition 2.1.1. *Let $M = \{x \in \mathbb{Z}_{2^n-1} \mid wt(x) \leq \frac{n}{2}\}$, where $wt(x)$ is the Hamming weight of x. We say that $\alpha \in \mathbb{Z}_{2^n-1}$ has the ϵ-*shifting property *if $M \cup (\alpha + M) \geq \left(\frac{1}{2} + \epsilon\right) 2^n$.*

We say that any binary string in M is *light*, and any binary string not in M is *heavy.* Then α has the ϵ-shifting property if translating $\mathbb{Z}_{2^n-1}$ by α takes a constant fraction of light strings to heavy strings.

Kopparty proved that powering by any α with the ϵ-shifting property requires exponential circuit size in $\mathrm{AC}^0(\oplus)$ [7]. We use the term *block* to denote a maximal length sequence of consecutive 0's or consecutive 1's in a binary string. Our main result is that any α with many blocks of 0's and 1's in its binary representation has the ϵ-shifting property, proving a conjecture of Kopparty.

Theorem 2.1.2. *$\forall c > 0$, $\exists \epsilon > 0$, such that the following holds: Let $\sigma \in \{0,1\}^n$ be a bit-string of the form $\sigma = \sigma_1\sigma_2 \cdots \sigma_m$, where $m \geq cn$, σ_i is either 0^{L_i} or 1^{L_i}, and each L_i is chosen to be maximal. Let $\alpha \in \mathbb{Z}_{2^n-1}$ have base 2 representation given by σ. Then α has the ϵ-shifting property.*

Note that the theorem still applies even in the setting of integer addition, not just when doing addition mod $2^n - 1$. Our result states that α with $\Theta(n)$ blocks have the ϵ-shifting property. It is not difficult to show that α with $o(\sqrt{n})$ blocks do not have the ϵ-shifting property. First, observe that $o(\sqrt{n})$-sparse α (i.e. α with Hamming weight $\leq o(\sqrt{n})$) do not have the ϵ-shifting property because addition by α can only increase the weight by $o(\sqrt{n})$. Since there are $O(\frac{2^n}{\sqrt{n}})$ light binary strings of a fixed weight, we get $o(2^n)$ light strings changing to heavy strings under translation by α.

Next, observe that any α with $o(\sqrt{n})$ blocks can be written as a difference of two $o(\sqrt{n})$-sparse strings: $\alpha = \beta - \gamma$. Since translating by α is equivalent to first translating by β and then by $-\gamma$, we find that α with $o(\sqrt{n})$ blocks do not have the ϵ-shifting property. Thus, at least qualitatively, we see a strong connection between the ϵ-shifting property and the number of blocks. Establishing a full characterization of the ϵ-shifting property remains an interesting open question.

2.1.1 Related Work

Kopparty gave a different condition for when α has the ϵ-shifting property: its binary representation consists mostly of a repeating constant-length string that is not all zeros or ones [7]. Note that any integer expressible as $\frac{a \cdot 2^n + b}{q}$, where $a, b, q \in \mathbb{Z}$, $q > 1$ is odd, and $0 < |a|, |b| < q$, has binary representation of this form. As a consequence, taking

q-th roots and computing q-th residue symbols cannot be done with polynomial-size $\mathrm{AC}^0(\oplus)$ circuits. Our main result generalizes Kopparty's condition, as the periodic strings form a small subset of the strings with $\Theta(n)$ blocks.

Beck and Li showed that the q-th residue map is hard to compute in $\mathrm{AC}^0(\oplus)$ by using the concept of algebraic immunity [1]. It is worth noting that their method does not say anything about the complexity of the q-th root map in $\mathrm{AC}^0(\oplus)$. So in this regard, there is something to be gained by analyzing the ϵ-shifting property condition. A more detailed history of the complexity of arithmetic operations using low-depth circuits can be found in [7].

2.2 Application

It is known that powering by sparse α has polynomial-size circuits in $\mathrm{AC}^0(\oplus)$. Kopparty's work shows that powering by α with the ϵ-shifting property requires exponential-size circuits in $\mathrm{AC}^0(\oplus)$. We will use this result, along with our new generalized criterion for when α has the ϵ-shifting property, to expand the class of α whose powers are difficult to compute in $\mathrm{AC}^0(\oplus)$.

The proof resembles the method of Razborov and Smolensky for showing that Majority is not in $\mathrm{AC}^0(\oplus)$ [8, 10, 11]. We can show that for α with the ϵ-shifting property, if powering by α is computable by an $\mathrm{AC}^0(\oplus)$ circuit, then every function $f : F_{2^n} \to F_{2^n}$ is well-approximated by the sum of a low-degree polynomial with a function that sits in a low-dimensional space. The fact that there are not enough such functions provides the desired contradiction. In this way, we show certain powers require exponential-size circuits in $\mathrm{AC}^0(\oplus)$.

As a consequence of Theorem 2.1.2 and the above Razborov-Smolensky method, we get that the powering by any α with $\Theta(n)$ maximal uniform blocks requires an exponential-size $\mathrm{AC}^0(\oplus)$ circuit, thus greatly expanding the class of powers that are hard to compute in $\mathrm{AC}^0(\oplus)$.

Theorem 2.2.1. *Let $\alpha \in \mathbb{Z}_{2^n-1}$ have base 2 representation in the form given by Theorem 2.1.2.*

Define $\Lambda : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ *by* $\Lambda(x) = x^{\alpha}$.

Then for every $AC^0(\oplus)$ *circuit* $C : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ *of depth* d *and size* $M \leq 2^{n^{\frac{1}{5}d}}$, *for sufficiently large* n *we have:*

$$Pr[C(x) = \Lambda(x)] \leq 1 - \epsilon_0,$$

where $\epsilon_0 > 0$ *depends only on* c *and* d.

2.3 The Proof of the Main Result

2.3.1 Outline of Proof

Suppose we have a bit-string of length n. The bit-string is called *light* if its Hamming weight is at most $\frac{n}{2}$. The bit-string is called *heavy* otherwise. It is enough to show that translation by α in $\mathbb{Z}_{2^n-1}$ transforms some positive constant fraction of the light bit-strings into heavy bit-strings.

We choose a binary string of length n uniformly at random, translate it by α, and look at the joint distribution of its initial weight X and final weight Y. Let $(\overline{X}, \overline{Y}) = (X - \mathbb{E}[X], Y - \mathbb{E}[Y])$, so that when plotted, the plane is split into four quadrants. The fraction of strings that shift weight from light to heavy is the proportion of the distribution in the second quadrant. By symmetry, the same proportion of the distribution should lie in the fourth quadrant. We will prove that some constant fraction of the distribution lies in the second or fourth quadrant.

To get a handle on the distribution, we break up α into its m uniform blocks of 0's and 1's, and consider addition on each block separately. Addition on each block is performed from right to left, and if the leftmost sum is at least 2, then we say the addition produces a carry and the *carry bit* is 1. Otherwise, the carry bit is 0. The distribution of the initial weight and final weight of any block is determined by the carry bit from the addition on the previous block and the carry bit going into the next block. Thus, if the carry bits are given, then the weight distributions on the blocks are now independent. Although we will not be able to specify the distribution of the carry bits, we will show that with probability $\frac{1}{6}$, the carry bits have a certain property, and whenever they have this property, then the conditional distribution of (X, Y) has a positive constant fraction of its mass in the second or fourth quadrants.

2.3.2 Notation and Overview

First, observe that it suffices to prove the main result for M as viewed as a subset of $\mathbb{Z}_{2^n}$ instead of $\mathbb{Z}_{2^n-1}$. Note that only one element, $1^n \in \mathbb{Z}_{2^n}$, is not an element of $\mathbb{Z}_{2^n-1}$. Also, when translating by α, the resulting bit-string in $\mathbb{Z}_{2^n-1}$ is either the same or one more than the resulting bit-string in $\mathbb{Z}_{2^n}$. We know that $o(2^n)$ of the light bit-strings of $\mathbb{Z}_{2^n}$ transform into heavy bit-strings under translation by 1, because the only bit-strings that change from light to heavy have weight $n/2$ and end in 0. By symmetry, $o(2^n)$ of the heavy bit-strings transform into light bit-strings under translation by 1, so if $\Theta(2^n)$ light bit-strings become heavy under translation by α in $\mathbb{Z}_{2^n}$, then at least $\Theta(2^n) - o(2^n) = \Theta(2^n)$ light bit-strings become heavy under translation by α in $\mathbb{Z}_{2^n-1}$. This shows that we can work in the symmetric environment of all bit-strings of length n, $\mathbb{Z}_{2^n}$, and still achieve the result we want.

Let $S \in \mathbb{Z}_{2^n}$ be chosen uniformly at random. Let $T = \alpha + S$. Let $X = wt(S)$ and $Y = wt(T)$.

Write $\alpha = \alpha_1\alpha_2 \cdots \alpha_m$, where each α_i is a block of 0's or 1's of length L_i. Write $S = S_1S_2 \cdots S_m$, and $T = T_1T_2 \cdots T_m$, where each of the i-th parts have length L_i. Let $X_i = wt(S_i)$ and $Y_i = wt(T_i)$. Then $(X, Y) = \left(\sum_{i=1}^{m} X_i, \sum_{i=1}^{m} Y_i\right)$. Let $(\overline{X}, \overline{Y}) = (X - \mathbb{E}[X], Y - \mathbb{E}[Y])$. Then the part of the distribution of $(\overline{X}, \overline{Y})$ in the second quadrant corresponds to light bit-strings translating to heavy bit-strings. Similarly, the fourth quadrant corresponds to heavy to light bit-string translation. To avoid having to pass to analogously defined $(\overline{X_i}, \overline{Y_i})$ all the time, any reference to the second or fourth quadrant will be understood to be relative to $\left(\frac{L_i}{2}, \frac{L_i}{2}\right)$, the mean of (X_i, Y_i). We want to show that a positive constant fraction of the distribution lies in the second or fourth quadrants.

The random variables in the sum $\left(\sum_{i=1}^{m} X_i, \sum_{i=1}^{m} Y_i\right)$ are highly dependent. To get around this, we will condition on the fixing of the carry bits. Once the carry bits are fixed, the terms in the sum are independent. We will show that with probability at least $\frac{1}{6}$, we can find $\Theta(n)$ terms with identical distribution. Since the terms are independent, we will use the multidimensional Central Limit Theorem to prove these identical distributions sum to a Gaussian distribution with dimensions of size $\Theta(\sqrt{n})$.

The remaining $O(n)$ terms can be divided into two categories. Either the term has non-zero covariance matrix or it is a translation along the line $y = -x$ relative to the mean, $\left(\frac{L_i}{2}, \frac{L_i}{2}\right)$. By applying the 2-dimensional Chebyshev Inequality to the terms with non-zero covariance matrix, we show that at least half of the distribution lies in a square with dimensions $O(\sqrt{n})$. Any Gaussian with dimensions $\Theta(\sqrt{n})$ centered in the square of dimensions $O(\sqrt{n})$ will have a fixed positive proportion p of its distribution in the second quadrant and p of its distribution in the fourth quadrant. Finally, a translation of any magnitude along the line $y = -x$ still gives at least p of the distribution in the second or fourth quadrant (although we don't know which one!). However, as the addition map is a bijection from $\mathbb{Z}_{2^n}$ to itself, we get that the number of strings that go from light to heavy equals the number of strings that go from heavy to light. So we conclude that at least p of the distribution lies in the second quadrant and at least p of the distribution lies in the fourth quadrant.

2.3.3 Computing the Distribution

We first compute the 2-dimensional distribution of the initial and final weights of the i-th block conditioned on the carry bit from the $(i+1)$-th block. If the carry bit from the i-th block is denoted by c_i, then we want to understand the distribution of (X_i, Y_i) given the carry bit c_{i+1}. Suppose that $\alpha_i = 1^{L_i}$. The case where $\alpha_i = 0^{L_i}$ is similar.

Lemma 2.3.1. *Suppose that $\alpha_i = 1^{L_i}$. The joint distribution of (X_i, Y_i) conditioned on the carry bit c_{i+1} is given by:*

$$p_i(x, y \mid c_{i+1} = 1) = \begin{cases} \frac{1}{2^{L_i}}\binom{L_i}{x} & \textit{if } x = y \, (\textit{then } c_i = 1) \\ 0 & \textit{else} \end{cases};$$

$$p_i(x, y \mid c_{i+1} = 0) = \begin{cases} \frac{1}{2^{L_i}} & \textit{if } (x, y) = (0, L_i) \, (\textit{then } c_i = 0) \\ \frac{1}{2^{L_i}} \binom{L_i - y + x - 2}{x-1} & \textit{if } L_i - 1 \geq y \geq x - 1 \geq 0 \, (\textit{then } c_i = 1) \end{cases}.$$

If $c_{i+1} = 1$, then $X_i = Y_i$ and $c_i = 1$. Hence, the probability mass function for (X_i, Y_i) given $c_{i+1} = 1$ is given by

$$p_i(x, y \mid c_{i+1} = 1) = \begin{cases} \frac{1}{2^{L_i}} \binom{L_i}{x} & \text{if } x = y \\ 0 & \text{else} \end{cases}.$$

If $c_{i+1} = 0$, then the distribution of (X_i, Y_i) depends solely on the number of trailing zeros, Z_i, of S_i:

$$Y_i = \begin{cases} X_i & \text{if } X_{i+1} - Y_{i+1} = L_{i+1} \\ X_i + Z_i - 1 & \text{if } Z_i < L_i \\ L_i & \text{if } Z_i = L_i \end{cases}.$$

We therefore first compute the distribution of Z_i conditioned on X_i and use that to compute the joint distribution of (X_i, Y_i). The distribution of $Z_i \mid X_i$ is given by

$$p_{Z_i}(z \mid x) = \begin{cases} 1 & \text{if } (x, z) = (0, L_i) \\ \frac{\binom{L_i - z - 1}{x-1}}{\binom{L_i}{x}} & \text{if } L_i - z \geq x \\ 0 & \text{else} \end{cases}.$$

Since $p_i(x, y \mid c_{i+1}) = p_{X_i}(x) p_{Y_i}(y \mid x)$, we compute $p_{X_i}(x)$ and $p_{Y_i}(y \mid x)$. As X_i is binomial on L_i trials with success probability $\frac{1}{2}$,

$$p_{X_i}(x) = \frac{1}{2^{L_i}} \binom{L_i}{x} \text{for } x = 0, 1, \cdots, L_i.$$

We can also write the distribution of $Y_i \mid X_i$ in terms of the distribution of $Z_i \mid X_i$:

$$p_{Y_i}(y_i \mid x_i) = \begin{cases} p_{Z_i}(y_i - x_i + 1 \mid x_i) & \text{if } 0 \leq y_i - x_i + 1 < L_i \\ p_{Z_i}(y_i - x_i \mid x_i) & \text{if } (x_i, y_i) = (0, L_i) \end{cases}.$$

Hence, we have the joint distribution of (X_i, Y_i) is

$$p_i(x, y \mid c_{i+1} = 0) = \begin{cases} \frac{1}{2^{L_i}} & \text{if } (x, y) = (0, L_i) \, (\text{then } c_i = 0) \\ \frac{1}{2^{L_i}} \binom{L_i - y + x - 2}{x - 1} & \text{if } L_i - 1 \geq y \geq x - 1 \geq 0 \, (\text{then } c_i = 1) \end{cases}.$$

Similarly, if $\alpha_i = 0^{L_i}$, then the distribution of (X_i, Y_i) is as follows:

If $c_{i+1} = 0$, then $X_i = Y_i$, $c_i = 0$ and

$$p_i(x, y \mid c_{i+1} = 0) = \begin{cases} \frac{1}{2^{L_i}} \binom{L_i}{x} & \text{if } x = y \, (\text{then } c_i = 0) \\ 0 & \text{else} \end{cases}.$$

When the carry bit makes the addition trivial, we call the resulting distribution the trivial distribution. Otherwise, the carry bit $c_{i+1} = 1$. In this case, the distribution of (X_i, Y_i) turns out to be symmetric with the case where $\alpha_i = 1^{L_i}$ and $c_{i+1} = 0$:

$$p_i(x, y \mid c_{i+1} = 1) = \begin{cases} \frac{1}{2^{L_i}} & \text{if } (x, y) = (L_i, 0) \, (\text{then } c_i = 1) \\ \frac{1}{2^{L_i}} \binom{L_i - x + y - 2}{y - 1} & \text{if } L_i - 1 \geq x \geq y - 1 \geq 0 \, (\text{then } c_i = 0) \end{cases}.$$

When the carry bit makes the addition nontrivial, as in this case, we call the resulting distribution the nontrivial distribution.

Fixing the carry bits leads to four types of distributions for the blocks, which are based on the carry bit coming in from the previous block addition and the resulting carry bit from the current block addition.

1. The block distribution is trivial and produces a carry bit that makes the subsequent block distribution non-trivial (Trivial to non-trivial).

2. Non-trivial to trivial.

3. Non-trivial to non-trivial (block length $L = 1$).

4. Non-trivial to non-trivial (block length $L \geq 2$).

We make the distinction between block lengths 1 and 2 for non-trivial to non-trivial distributions, as the latter is the only distribution with invertible covariance matrix. Ideally, we will find many identical distributions of type 4, which will sum to a Gaussian with large enough dimensions. This will not be possible when most of the blocks have length 1, which we deal with separately.

Knowing the weight distribution of a block given the previous carry, it is straightforward to write down the distributions given both the previous carry and the produced carry. Again, we assume the block $\alpha_i = 1^{L_i}$.

As a trivial distribution always produces a non-trivial carry, we get the trivial to non-trivial distribution is the same is the trivial distribution:

$$p_i(x, y \mid c_{i+1} = 1, c_i = 1) = \begin{cases} \frac{1}{2^{L_i}}\binom{L_i}{x} & \text{if } x = y \\ 0 & \text{else} \end{cases}.$$

A non-trivial distribution that produces a trivial carry must have $(X_i, Y_i) = (0, L_i)$. Also, a non-trivial distribution of block length 1 that produces a non-trivial carry must have $(X_i, Y_i) = (1, 0)$.

Finally, a non-trivial distribution of block length greater than 1 that produces a non-trivial carry has distribution:

$$p_i(x, y \mid c_{i+1} = 0, c_i = 1) = \begin{cases} \frac{1}{2^{L_i-1}}\binom{L_i - y + x - 2}{x-1} & \text{if } L_i - 1 \geq y \geq x - 1 \geq 0 \\ 0 & \text{else} \end{cases}.$$

We summarize these distributions in the next lemma:

Lemma 2.3.2. *Suppose that $\alpha_i = 1^{L_i}$. The joint distribution of (X_i, Y_i) conditioned on the carry bits c_{i+1} and c_i is given by:*

$$p_i(x, y \mid c_{i+1} = 1, c_i = 1) = \begin{cases} \frac{1}{2^{L_i}}\binom{L_i}{x} & \textit{if } x = y \\ 0 & \textit{else} \end{cases};$$

$$p_i(x, y \mid c_{i+1} = 0, c_i = 0) = \begin{cases} 1 & \textit{if } (x, y) = (0, L_i) \\ 0 & \textit{else} \end{cases};$$

$$p_i(x, y \mid c_{i+1} = 0, c_i = 1) = \begin{cases} \frac{1}{2^{L_i}-1}\binom{L_i-y+x-2}{x-1} & \textit{if } L_i - 1 \geq y \geq x - 1 \geq 0 \\ 0 & \textit{else} \end{cases}.$$

Observe that the last non-trivial to non-trivial probability distribution works for all lengths $L_i \geq 1$. However, when $L_i = 1$, $(x, y) = (0, 1)$ with probability 1. We will still consider this as a separate type of distribution as its covariance matrix is all zeros, and consequently not invertible, which will be important for analysis.

2.3.4 Computing the Covariance Matrix

Lemma 2.3.3. *The covariance matrix M of the trivial to non-trivial distribution of the random vector (X_i, Y_i) is given by*

$$M = \begin{pmatrix} \frac{L_i}{4} & \frac{L_i}{4} \\ \frac{L_i}{4} & \frac{L_i}{4} \end{pmatrix}.$$

The covariance matrix M of the non-trivial to non-trivial distribution of the random vector (X_i, Y_i) is given by

$$M = \begin{pmatrix} c & d \\ d & c \end{pmatrix},$$

where $c = \frac{L_i}{4}\left(1 + \frac{1}{2^{L_i}-1}\right) - \frac{L_i^2}{4}\left(1 + \frac{1}{2^{L_i}-1}\right)\frac{1}{2^{L_i}-1}$,
and $d = \frac{L_i}{4}\left(1 + \frac{1}{2^{L_i}-1}\right) + \frac{L_i^2}{4}\left(1 + \frac{1}{2^{L_i}-1}\right)\frac{1}{2^{L_i}-1} - 1$.

We will need the following binomial coefficient identities:

Lemma 2.3.4.

$$\begin{aligned} \sum_{n=1}^{L}\binom{L}{n}n^2 &= \frac{L(L+1)}{4}2^L; \\ \sum_{x=1}^{L-1}x(x+1)\binom{L}{x} &= 2^{L-2}L(L+3) - (L^2 + L); \end{aligned}$$

$$\sum_{x=1}^{L-1} x\binom{L}{x-1} = 2^{L-1}(L+2) - (L^2+L+1).$$

Proof. The proofs of these identities are standard and are included for completeness.

We use repeated differentiation of the binomial theorem to compute $\sum_{n=1}^{L}\binom{L}{n}n^2$.

$$\sum_{n=1}^{L}\binom{L}{n}x^n = (x+1)^L - 1.$$

Differentiating with respect to x yields:

$$\begin{aligned}\sum_{n=1}^{L}\binom{L}{n}nx^{n-1} &= L(x+1)^{L-1}\\ \sum_{n=1}^{L}\binom{L}{n}nx^{n} &= L(x+1)^{L-1}x.\end{aligned}$$

Differentiating a second time with respect to x gives:

$$\sum_{n=1}^{L}\binom{L}{n}n^2x^{n-1} = L(L-1)(x+1)^{L-2}x + L(x+1)^{L-1}.$$

Plugging in $x=1$ gives us the sum we want:

$$\begin{aligned}\sum_{n=1}^{L}\binom{L}{n}n^2 &= L(L-1)2^{L-2} + L2^{L-1}\\ &= L2^L\left(\frac{L-1}{4}+\frac{1}{2}\right)\\ &= \frac{L(L+1)}{4}2^L.\end{aligned}$$

Let $B = \sum_{x=1}^{L-1} x(x+1)\binom{L}{x}$, and let $C = \sum_{x=1}^{L-1} x\binom{L}{x-1}$. We can simplify B and C by starting with the binomial theorem and applying standard generating function methods.

$$\begin{aligned}\sum_{k=0}^{L}\binom{L}{k}x^k &= (1+x)^L\\ \sum_{k=0}^{L}\binom{L}{k}x^{k+1} &= x(1+x)^L.\end{aligned}$$

Differentiating both sides with respect to x gives:

$$\sum_{k=0}^{L}(k+1)\binom{L}{k}x^k = (1+x)^{L-1}(1+(L+1)x). \tag{2.1}$$

Substituting $x = 1$ in equation 2.1 gives:

$$\begin{aligned}\sum_{k=0}^{L}(k+1)\binom{L}{k} &= (1+x)^{L-1}(L+2)\\ C + L\binom{L}{L-1} + (L+1)\binom{L}{L} &= 2^{L-1}(L+2)\\ C &= 2^{L-1}(L+2) - (L^2+L+1).\end{aligned}$$

To get B, we differentiate equation 2.1 with respect to x once more:

$$\sum_{k=1}^{L}k(k+1)\binom{L}{k}x^{k-1} = L(1+x)^{L-2}(2+(L+1)x). \tag{2.2}$$

Substituting $x = 1$ in equation 2.2 gives:

$$\begin{aligned}\sum_{k=1}^{L}k(k+1)\binom{L}{k} &= L2^{L-2}(L+3)\\ B + L(L+1)\binom{L}{L} &= 2^{L-2}L(L+3)\\ B &= 2^{L-2}L(L+3) - (L^2+L).\end{aligned}$$

□

Proof of Lemma 2.3.3. To simplify our notation, let $(X(L), Y(L))$ denote some (X_i, Y_i) with $L_i = L$. We begin with the trivial to non-trivial distribution. Since $X(L)$ is binomial on L trials with success probability $\frac{1}{2}$, $Var(X(L)) = \frac{L}{4}$. Since the bit string corresponding to $Y(L)$ can be viewed as a translation of the bit string corresponding to $X(L)$ in $\mathbb{Z}_{2^L}$, the distribution of $Y(L)$ is the same as the distribution of $X(L)$. Hence, $Var(Y(L)) = \frac{L}{4}$. It remains to compute $Cov(X(L), Y(L))$. In the case of the trivial distribution, $X(L) = Y(L)$. So $Cov(X(L), Y(L)) = Var(X(L)) = \frac{L}{4}$.

The case of the non-trivial to non-trivial distribution requires more work. For our computation, we assume $\alpha_i = 1^{L_i}$. As the nontrivial distributions are symmetric in

x and y, the covariances will be the same. We begin by evaluating $Var(X(L)) = \mathbb{E}[X(L)^2] - \mathbb{E}[X(L)]^2$.

As the block with weight $X(L)$ is chosen uniformly at random among all non-zero strings of length L,

$$\begin{aligned}\mathbb{E}[X(L)] &= \frac{L}{2}\left(1+\frac{1}{2^L-1}\right);\\ \mathbb{E}[X(L)^2] &= \frac{1}{2^L-1}\sum_{n=1}^{L}\binom{L}{n}n^2.\end{aligned}$$

Using Lemma 2.3.4, the variance of $X(L)$ is given by:

$$\begin{aligned}Var(X(L)) &= \frac{L(L+1)}{4}\frac{2^L}{2^L-1} - \frac{L^2}{4}\left(1+\frac{1}{2^L-1}\right)^2\\ &= \frac{L^2+L}{4}\left(1+\frac{1}{2^L-1}\right) - \frac{L^2}{4}\left(1+\frac{1}{2^L-1}\right) - \frac{L^2}{4}\left(1+\frac{1}{2^L-1}\right)\frac{1}{2^L-1}\\ &= \frac{L}{4}\left(1+\frac{1}{2^L-1}\right) - \frac{L^2}{4}\left(1+\frac{1}{2^L-1}\right)\frac{1}{2^L-1}.\end{aligned}$$

Observe that Y is the weight of a block of length L chosen uniformly at random from all strings except 1^L. So by symmetry, $Var(Y) = Var(X)$. We now compute $Cov(X,Y) = \mathbb{E}[X(L),Y(L)] - \mathbb{E}[X(L)]\mathbb{E}[Y(L)]$.

$$\begin{aligned}\mathbb{E}[X(L)Y(L)] &= \frac{1}{2^L-1}\sum_{1\le y\le x+1\le L} xy\binom{L-x+y-2}{y-1}\\ &= \frac{1}{2^L-1}\sum_{x=0}^{L-1}x\sum_{y=1}^{x+1}y\binom{L-x+y-2}{y-1}\\ &= \frac{1}{2^L-1}\sum_{x=0}^{L-1}x\sum_{y=0}^{x}(y+1)\binom{L-x+y-1}{y}.\end{aligned}$$

Let $A(x) = \sum_{y=0}^{x}(y+1)\binom{L-x+y-1}{y}$ be the inner summation. Then by repeated application of the hockey stick identity,

$$A(x) = \sum_{y=0}^{x}(x+1)\binom{L-x+y-1}{y} - \sum_{y=0}^{x-1}(x-y)\binom{L-x+y-1}{y}$$

$$\begin{aligned}
&= (x+1)\binom{L}{x} - \sum_{y=0}^{x-1}\sum_{j=0}^{y}\binom{L-x+j-1}{j} \\
&= (x+1)\binom{L}{x} - \sum_{y=0}^{x-1}\binom{L-x+y}{y} \\
&= (x+1)\binom{L}{x} - \binom{L}{x-1}.
\end{aligned}$$

Substituting $A(x)$ back into our expression for $\mathbb{E}[X(L)Y(L)]$ yields:

$$\mathbb{E}[X(L)Y(L)] = \frac{1}{2^L-1}\left[\sum_{x=1}^{L-1} x(x+1)\binom{L}{x} - \sum_{x=1}^{L-1} x\binom{L}{x-1}\right].$$

Using Lemma 2.3.4, we get:

$$\begin{aligned}
\mathbb{E}[X(L)Y(L)] &= \frac{1}{2^L-1}(2^{L-2}L(L+3) - (L^2+L) - 2^{L-1}(L+2) + (L^2+L+1)) \\
&= \frac{2^L}{2^L-1}\left(\frac{L(L+1)}{4} - \frac{L(L+1)}{2^L} - \frac{L+2}{2} + \frac{L^2+L+1}{2^L}\right) \\
&= \left(1+\frac{1}{2^L-1}\right)\left(\frac{L^2+L-4}{4} + \frac{1}{2^L}\right) \\
&= \left(1+\frac{1}{2^L-1}\right)\left(\frac{L^2+L}{4} - \left(1-\frac{1}{2^L}\right)\right) \\
&= \frac{L^2+L}{4}\left(1+\frac{1}{2^L-1}\right) - 1.
\end{aligned}$$

Hence the covariance of X(L) and Y(L) is

$$\begin{aligned}
& Cov(X(L), Y(L)) \\
&= \frac{L^2+L}{4}\left(1+\frac{1}{2^L-1}\right) - 1 - \frac{L}{2}\left(1+\frac{1}{2^L-1}\right)\frac{L}{2}\left(1-\frac{1}{2^L-1}\right) \\
&= \frac{L^2+L}{4}\left(1+\frac{1}{2^L-1}\right) - 1 - \frac{L^2}{4}\left(1+\frac{1}{2^L-1}\right) + \frac{L^2}{4}\left(1+\frac{1}{2^L-1}\right)\frac{1}{2^L-1} \\
&= \frac{L}{4}\left(1+\frac{1}{2^L-1}\right) + \frac{L^2}{4}\left(1+\frac{1}{2^L-1}\right)\frac{1}{2^L-1} - 1.
\end{aligned}$$

$\square$

2.3.5 Breaking up the Weight Distribution

Recall that the joint distribution of initial and final weights is the sum of the joint distribution of initial and final weights for $m = \Theta(n)$ smaller parts: $(X, Y) = \sum_{i=1}^{m} (X_i, Y_i) := \left(\sum_{i=1}^{m} X_i, \sum_{i=1}^{m} Y_i \right)$. However, the terms of this sum are dependent. We can remove the dependence by first sampling the carry bits according to their distribution. Given the carry bits, all of the terms in the sum are independent and have one of four types of distributions given by Lemma 2.3.2. This gives us access to the Central Limit Theorem and the fact that covariance matrices add, both of which will be used in the proof.

We will break up (X, Y) into a sum of a Gaussian (X_G, Y_G), a translation (X_T, Y_T), and some remainder (X_R, Y_R), which we show is well-behaved. As the non-trivial to non-trivial distribution with block length at least 2 (Type 4) is the only type with invertible covariance matrix, our goal will be to find many identical distributions of this type. By the Central Limit Theorem, these sum to a 2-D Gaussian (X_G, Y_G) of dimensions $\Theta(\sqrt{n})$. This will be the main part of the sum that pushes the distribution into the second and fourth quadrants.

It is not always possible to find many identical distributions of type 4. If there are $o(n)$ blocks of length at least 2, then it is trivially impossible. We deal with this case separately with a slightly modified argument. Otherwise, there are $\Theta(n)$ blocks of length at least 2. We will show that with probability at least $\frac{1}{6}$, the carry bits arrange themselves in such a way so that there are $\Theta(n)$ distributions of type 4. This is enough to find many identical distributions of type 4.

We then consider the sum of the remainder of the type 4 distributions along with the trivial to non-trivial type 1 distributions, (X_R, Y_R), and show that it is well-behaved. As the covariance matrices add, we will be able to apply the 2-D Chebyshev inequality to guarantee that half of the distribution lies inside an ellipse of dimensions $O(\sqrt{n})$. This will be enough to guarantee some constant proportion p of the distribution of $(X_G, Y_G) + (X_R, Y_R)$ in the second quadrant, and the same proportion p in the fourth quadrant. The rest of the distribution of (X, Y) is a translation (X_T, Y_T) along the line $y = -x$ relative to the mean. After translation, we still have p-fraction of the distribution in either the second or fourth quadrant.

We first consider the case where there are $m' = \Theta(n)$ blocks of length at least 2. The following lemma says that with probability at least $\frac{1}{6}$, we get many distributions of type 4.

Lemma 2.3.5. *Suppose there are $m' = \Theta(n)$ blocks of length at least 2. Let X be the number of non-trivial to non-trivial distributions with block length at least 2. Then:*

$$\mathbb{P}\left(X > \frac{m'}{4}\right) > \frac{1}{6}.$$

Proof.

$$\begin{aligned}
\mathbb{E}[X] &= \sum_{i=1}^{m'} \mathbb{P}(\text{Block } i \text{ is non-trivial to non-trivial}) \\
&= \sum_{i=1}^{m'} \mathbb{P}(\text{Non-trivial carry out } \mid \text{ non-trivial carry in}) \cdot \mathbb{P}(\text{Non-trivial carry in}) \\
&\geq \sum_{i=1}^{m'} \frac{3}{4} \cdot \frac{1}{2} \\
&= \frac{3}{8} m'.
\end{aligned}$$

Let Y be the number of blocks of length at least 2 that are not non-trivial to non-trivial. Then $\mathbb{E}[Y] \leq \frac{5}{8} m'$. By Markov's inequality,

$$\mathbb{P}(Y \geq t) \leq \frac{\mathbb{E}[Y]}{t} \leq \frac{5}{8} \cdot \frac{m'}{t}.$$

Taking $t = \frac{3}{4} m'$ yields

$$\begin{aligned}
\mathbb{P}(Y \geq \frac{3}{4} m') &\leq \frac{5}{6} \\
\mathbb{P}(Y \leq \frac{3}{4} m') &\geq \frac{1}{6} \\
\mathbb{P}(X \geq \frac{1}{4} m') &\geq \frac{1}{6}.
\end{aligned}$$

□

We now show that many distributions of type 4 implies many identical distributions of type 4.

Lemma 2.3.6. *Suppose we have $m = \Theta(n)$ bit-strings of total length at most n. Then there is some fixed positive length L such that $l = \Theta(n)$ bit-stings have length L.*

Proof. Let n_k be the number of blocks of length k. Then we have the following equations about the total number of blocks and the total length of all the blocks:

$$\begin{aligned}\sum_{k=1}^{n} n_k &= m;\\ \sum_{k=1}^{n} k \cdot n_k &\leq n.\end{aligned}$$

Since $m = \Theta(n)$, we have $\frac{n}{m} = C = \Theta(1)$. Multiplying the first equation by C and subtracting from the second inequality, we get:

$$-C\sum_{k=1}^{C+1} n_k + \sum_{k>C+1} n_k \leq \sum_{k=1} n(k-C)n_k \leq 0.$$

So we have:

$$\begin{aligned}\frac{n}{C} = m &= \sum_{k=1}^{n} n_k\\ &= \sum_{k=1}^{C+1} n_k + \sum_{k>C+1} n_k\\ &\leq (C+1)\sum_{k=1}^{C+1} n_k\\ &\leq (C+1)^2 \max_{k\in[C]} n_k.\end{aligned}$$

So $\max_{k\in[C]} n_k \geq \frac{1}{C(C+1)^2}n$. Taking $a = \frac{1}{C(C+1)^2}$, we get that there exists a fixed length $L \in [C]$ such that an bit-strings have length L, as desired.

□

So with probability at least $\frac{1}{6}$, we can find $l = \Theta(n)$ identical type 4 distributions. Since these identical distributions are independent of each other, the Central Limit Theorem together with Lemma 2.3.3 tells us that the distribution of their sum is a Gaussian with covariance matrix given by

$$M_G = \begin{pmatrix} cl & dl \\ dl & cl \end{pmatrix},$$

where $c = \frac{L}{4}\left(1+\frac{1}{2^L-1}\right) - \frac{L^2}{4}\left(1+\frac{1}{2^L-1}\right)\frac{1}{2^L-1}$,
$d = \frac{L}{4}\left(1+\frac{1}{2^L-1}\right) + \frac{L^2}{4}\left(1+\frac{1}{2^L-1}\right)\frac{1}{2^L-1} - 1$, and $L \geq 2$.

Lemma 2.3.7. *Suppose G is a 2-dimensional Gaussian distribution with covariance matrix given by M_G. Then a fixed positive proportion of the distribution of G lies inside (and outside) an ellipse centered at the mean with dimensions $\Theta(\sqrt{n})$. Furthermore, the probability density function $f_G(x,y) \geq \frac{1}{\pi n}e^{-\frac{144n}{l}}$ inside a circle of radius $4\sqrt{n}$ centered at the mean of G.*

Proof. Observe first that $c-d = 1-\frac{L^2}{2}\left(1+\frac{1}{2^L-1}\right)\frac{1}{2^L-1} \geq \frac{1}{9}$ and $d \geq \frac{1}{9}$ for any value of $L \geq 2$. As $\det(M_G) = (c^2-d^2)l^2 \neq 0$, M_G is invertible. So letting $\overline{G} = (\overline{X_G}, \overline{Y_G})$ denote the distribution obtained by translating G to its mean, we get that a fixed proportion of the distribution lies in the ellipse defined by:

$$\begin{pmatrix} \overline{X_G} & \overline{Y_G} \end{pmatrix} M_G^{-1} \begin{pmatrix} \overline{X_G} \\ \overline{Y_G} \end{pmatrix} = 2.$$

The inverse of M_G is given by:

$$M_G^{-1} = \frac{1}{(c^2-d^2)l^2}\begin{pmatrix} cl & -dl \\ -dl & cl \end{pmatrix}.$$

Substituting M_G^{-1} back into the equation of the ellipse gives:

$$\begin{aligned} \frac{1}{(c^2-d^2)l}[c\overline{X_G}^2 - 2d\overline{X_G Y_G} + c\overline{Y_G}^2] &= 2 \\ \overline{X_G}^2 - \frac{2d}{c}\overline{X_G Y_G} + \overline{Y_G}^2 &= \frac{2(c^2-d^2)}{c}l. \end{aligned}$$

We have an equation of the form $x^2 - 2axy + y^2 = b$, where $a = \frac{d}{c} < 1$. This describes an ellipse rotated by $\frac{\pi}{4}$ counterclockwise. By rotating the ellipse clockwise by $\frac{\pi}{4}$, we can find the dimensions of the ellipse. Making the substitution:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \frac{\sqrt{2}}{2}(x' + y') \\ \frac{\sqrt{2}}{2}(y' - x') \end{pmatrix},$$

we get that the equation of the rotated ellipse is

$$\frac{x^2}{\frac{b}{1+a}} + \frac{y^2}{\frac{b}{1-a}} = 1. \tag{2.3}$$

Taking $a = \frac{d}{c}$ and $b = \frac{2(c^2-d^2)}{c} l$ as in the ellipse for our Gaussian, we find that the squares of the dimensions of the ellipse are given by:

$$\begin{aligned} \frac{b}{1-a} &= \frac{2(c^2-d^2)}{c} l \cdot \frac{c}{c-d} = 2(c+d)l \geq \frac{2}{3} l; \\ \frac{b}{1+a} &= \frac{2(c^2-d^2)}{c} l \cdot \frac{c}{c+d} = 2(c-d)l \geq \frac{2}{9} l. \end{aligned}$$

Hence, both dimensions of the ellipse are $\Theta(\sqrt{n})$. In fact, both dimensions exceed $\frac{1}{3}\sqrt{l}$. So the circle of radius $\frac{1}{3}\sqrt{l}$ centered at the mean lies completely inside the ellipse. Scaling every dimension up by a factor of $\sqrt{\frac{144n}{l}}$ tells us that inside the circle of radius $4\sqrt{n}$ centered at the mean, we have:

$$\begin{pmatrix} \overline{X_G} & \overline{Y_G} \end{pmatrix} M_G^{-1} \begin{pmatrix} \overline{X_G} \\ \overline{Y_G} \end{pmatrix} \leq \frac{144n}{l}.$$

Therefore, we have the following lower bound on the probability distribution function inside the circle of radius $4\sqrt{n}$:

$$\begin{aligned} f_G(x,y) &\geq \frac{1}{2\pi\sqrt{\det M_G}} e^{-\frac{144n}{l}} \\ &= \frac{1}{2\pi\sqrt{(c^2-d^2)l^2}} e^{-\frac{144n}{l}} \\ &\geq \frac{1}{2\pi c l} e^{-\frac{144n}{l}} \\ &\geq \frac{1}{\pi n} e^{-\frac{144n}{l}}. \end{aligned}$$

□

The above sequence of lemmas can be used to show that if we start with many blocks of length at least 2, then with positive constant probability, we can find many identical distributions that sum to a Gaussian of dimensions $\Theta(\sqrt{n})$. Suppose now that the exponent α has a total of m blocks, but fewer than $0.01m$ blocks of length at least 2. Then at least 0.99 fraction of the blocks have length 1. Consider all consecutive block pairs. At most 0.01 fraction of these pairs have their first block with length 2, and at most 0.01 fraction have their second block with length at least 2. So at most 0.02 fraction have a block of length at least 2. Hence, 0.98 fraction of the pairs consists of two blocks of length 1. By the Pigeonhole Principle, at least 0.49 fraction of the pairs are either all 01 or all 10. Without loss of generality, assume that 0.49 fraction of consecutive block pairs are 01. We now treat each block pair 01 as a single block of length 2. The initial and final weight distribution of this larger block, given there is no carry in and no carry out matches the type 4 distribution. We have proven the existence of a large number of modified blocks of length 2:

Lemma 2.3.8. *Suppose there are fewer than* 0.01 *fraction of the blocks have length at least* 2*. Then at least* 0.49 *fraction of consecutive block pairs are* 01 *or at least* 0.49 *fraction of consecutive block pairs are* 10*.*

Lemma 2.3.8 essentially reduces the case of having few blocks of length at least 2 to the case where there are many blocks of length at least 2 by consolidating many of the length 1 blocks. As there are $\Theta(n)$ such consolidated blocks, and each has type 4 distribution with probability at least $\frac{3}{8}$, we can again find $\Theta(n)$ identical type 4 distributions by Lemma 2.3.5. Lemma 2.3.7 then says that these identical distributions sum to a Gaussian of large dimensions. So for any α, we can find many terms in the initial and final weight distribution summing to a large Gaussian.

2.3.6 Distribution of the Sum of the Remaining Terms

Consider the terms remaining in the distribution of $(X, Y) = \sum_{i=1}^{m} (X_i, Y_i)$, when the terms contributing to the Gaussian are removed. The terms with distribution types 2 or 3 are translations in the $< 1, -1 >$ direction relative to $\left(\frac{L}{2}, \frac{L}{2}\right)$. These will contribute to the translation part of the distribution (X_T, Y_T). The rest of the terms of type 4 along with the terms of type 1 sum to the remainder $R = (X_R, Y_R)$. There are $O(n)$ terms remaining. By Lemma 2.3.3, the covariance matrix of each of these terms is one of the following two forms:

$$\begin{pmatrix} \frac{L}{4} & \frac{L}{4} \\ \frac{L}{4} & \frac{L}{4} \end{pmatrix};$$

$$\begin{pmatrix} c & d \\ d & c \end{pmatrix},$$

where $c = \frac{L}{4}\left(1 + \frac{1}{2^L-1}\right) - \frac{L^2}{4}\left(1 + \frac{1}{2^L-1}\right)\frac{1}{2^L-1}$, and
$d = \frac{L}{4}\left(1 + \frac{1}{2^L-1}\right) + \frac{L^2}{4}\left(1 + \frac{1}{2^L-1}\right)\frac{1}{2^L-1} - 1$.

As the terms are independent given a fixing of the carry bits, the covariance matrices add. The total covariance matrix of the sum is:

$$M_R = \begin{pmatrix} C & D \\ D & C \end{pmatrix},$$

where $D \leq C \leq \frac{n}{3}$, with $D = C$ only when the remainder is a sum of type 1 distributions.

Lemma 2.3.9. *At least half of the distribution of the remainder lies in a circle of radius* $\sqrt{2n}$ *centered at the mean.*

Proof. When the remainder is a sum of type 1 distributions, then the remainder has the form (X_R, X_R), where X_R is a binomial distribution with $L_R \leq n$ trials and success probability $\frac{1}{2}$. So by Chebyshev's Inequality,

$$\mathbb{P}\left(|X_R - \frac{L_R}{2}| > \sqrt{\frac{L_R}{2}}\right) \leq \frac{1}{2}$$
$$\mathbb{P}\left(|X_R - \frac{L_R}{2}| > \sqrt{\frac{n}{2}}\right) \leq \frac{1}{2}$$
$$\mathbb{P}\left(|X_R - \frac{L_R}{2}| \leq \sqrt{\frac{n}{2}}\right) \geq \frac{1}{2}.$$

So in this case, at least half of the distribution of the remainder lies in a circle of radius $\sqrt{\frac{n}{2}}$. When the remainder contains some type 4 distributions, then $D < C \leq \frac{n}{3}$. Hence, the covariance matrix M_R is invertible. So we may apply the 2-dimensional Chebyshev inequality to (X_R, Y_R) to get:

$$\mathbb{P}\left\{\overline{R}M_R^{-1}\overline{R}^T > t\right\} \leq \frac{2}{t^2}.$$

Taking $t = 2$ yields

$$\begin{aligned} Pr\left\{\overline{X_R}^2 + 2\frac{D}{C}\overline{X_R Y_R} + \overline{Y_R}^2 > \frac{2(C^2 - D^2)}{C}\right\} &\leq \frac{1}{2} \\ Pr\left\{\overline{X_R}^2 + 2\frac{D}{C}\overline{X_R Y_R} + \overline{Y_R}^2 \leq \frac{2(C^2 - D^2)}{C}\right\} &> \frac{1}{2}. \end{aligned}$$

Chebyshev tells us that at least half of the distribution lies in the ellipse centered at the origin defined by the inequality above. By a similar computation as with the Gaussian distribution, the squares of the dimensions of this ellipse are $2(C + D)$ and $2(C - D)$, both of which are less than $\frac{4n}{3}$. Hence, the ellipse lies inside a circle of radius $2\sqrt{\frac{n}{3}} < \sqrt{2n}$, and therefore over half of the distribution of the remainder must lie inside this circle. $\square$

2.3.7 The Proof

We are ready to prove the main theorem.

Proof of Theorem 2.1.2. Suppose that $\alpha \in \mathbb{Z}_{2^n}$ has $m \geq cn$ blocks in its binary representation, where $0 < c < 1$ is constant. We know that either there are $0.01m$ blocks of length at least 2, or there are fewer than $0.01m$ blocks of length at least 2. In the second case, Lemma 2.3.8 tells us that we can find $0.49m$ identical pairs of consecutive blocks of length 1. Lemma 2.3.5 then says that with probability at least $\frac{1}{6}$, the carry bits arrange themselves in such a way that there are at least $\frac{0.49}{4}m \geq \frac{1}{9}m$ identical type 4 distributions.

If there are $m' > \frac{1}{100}m$ blocks of length at least 2, then Lemma 2.3.5 says that with probability at least $\frac{1}{6}$, the carry bits arrange themselves in such a way that there are at least $\frac{1}{4}m' > \frac{1}{400}m$ type 4 distributions. Since $\frac{1}{9} > \frac{1}{400}$, we conclude that for any α with m blocks, we can find $\frac{1}{400}m$ type 4 distributions with probability $\frac{1}{6}$.

As $m \geq cn$, the number of type 4 distributions exceeds $\frac{1}{400}m \geq \frac{c}{400}n$. By Lemma 2.3.6, we can find $l \geq \left(\frac{c}{400}\right)^3 n$ identical type 4 distributions each with block length L. By Lemma 2.3.7, these sum to a Gaussian whose probability distribution function $f_G(x,y) \geq \frac{1}{\pi n}e^{-\frac{144n}{l}}$ inside a circle of radius $4\sqrt{n}$ centered at the mean of G. As each type 4 distribution has mean $\left(\frac{L}{2}, \frac{L}{2}\right) \pm \frac{L}{2} \cdot \frac{1}{2^L-1}(1,-1)$, we decompose the Gaussian into a Gaussian $G = (X_G, Y_G)$ centered at $\left(\frac{Ll}{2}, \frac{Ll}{2}\right)$ and a translation in the $(1,-1)$ direction which contributes to the translation term (X_T, Y_T).

It is worth noting that every distribution type of block length L can be decomposed into the sum of a distribution centered at $\left(\frac{L}{2}, \frac{L}{2}\right)$ and a translation in the $(1,-1)$ direction. To see this, we will write the mean of each type of distribution as $\left(\frac{L}{2}, \frac{L}{2}\right) + k(1,-1)$, for some k depending on L.

Type 1 distributions have mean $\left(\frac{L}{2}, \frac{L}{2}\right)$. Type 2 distributions have mean $\left(\frac{L}{2}, \frac{L}{2}\right) \pm \frac{L}{2}(1,-1)$. Type 3 distributions have mean $\left(\frac{L}{2}, \frac{L}{2}\right) \pm \frac{L}{2}(1,-1)$. Type 4 distributions have mean $\left(\frac{L}{2}, \frac{L}{2}\right) \pm \frac{L}{2} \cdot \frac{1}{2^L-1}(1,-1)$.

We extract the translation component from each term and call that sum (X_T, Y_T). Let $R = (X_R, Y_R) = (X,Y) - (X_G, Y_G) - (X_T, Y_T)$ be the remainder. If L_T denotes the total length of all blocks contributing to R, we have that $\left(\frac{L_T}{2}, \frac{L_T}{2}\right)$ is the mean of R. Let $\overline{R} = R - \left(\frac{L_T}{2}, \frac{L_T}{2}\right)$. By Lemma 2.3.9, at least half of the distribution of $\overline{R}$ lies in an circle centered at the origin with radius $\sqrt{2n}$. By taking the square W of side length $2\sqrt{2n}$ surrounding the circle, we see that at least half of the distribution of R lies in W.

For any point (p,q) in the square, consider the distribution of the Gaussian $\overline{G}+(p,q) = (\overline{X_G}+p, \overline{Y_G}+q)$, which is centered at (p,q). Lemma 2.3.7 guarantees that the probability distribution function exceeds $\frac{1}{\pi n}e^{-\frac{144n}{l}}$ inside a circle of radius $4\sqrt{n}$ centered at (p,q). Contained within this circle is a square with side length $\sqrt{2n}$ in the second quadrant. Hence, the probability that $\overline{G} + (p,q)$ lies in the second quadrant is at least $\frac{2}{\pi}e^{-\frac{144n}{l}}$.

Recall that $l \geq \frac{c^3}{400^3}n$, where $m \geq cn$. So we have:

$$\begin{aligned}\frac{2}{\pi}e^{-\frac{144n}{l}} &\geq \frac{2}{\pi}e^{-\frac{144\cdot 64000000}{c^3}} \\ &= \frac{2}{\pi}e^{-\frac{9216000000}{c^2}}.\end{aligned}$$

Take $C = \frac{2}{\pi}e^{-\frac{9216000000}{c^2}}$. Then with probability $\frac{1}{6}$, at least C fraction of the distribution of $\overline{G} + \overline{R}$ conditioned on the carry bits lies in the second quadrant, where C is a constant depending only on c. By symmetry, the same fraction lies in the fourth quadrant. Finally, we must add the translation (X_T, Y_T) in the $(1, -1)$ direction. No matter the size of the translation, we are guaranteed C fraction in either the second or fourth quadrant. Hence, we have at least $\frac{C}{6}$ of the unconditioned distribution of $(X, Y) = (X_G, Y_G) + (X_R, Y_R) + (X_T, Y_T)$ lying in the second or fourth quadrants relative to the mean $\left(\frac{n}{2}, \frac{n}{2}\right)$, and so at least $\frac{C}{12}$ lying in either the second or fourth quadrant. By symmetry, we get at least $\frac{C}{12}$ lying in both the second and fourth quadrants. $\square$

2.4 Heavily Shifting Numbers

We have shown that α with many uniform blocks of 0's and 1's have the shifting property. An interesting related question is whether there is an α that is heavily shifting: that is, α shifts almost all of the light strings to heavy strings. More precisely, $o(1)$ fraction of light strings remain light under translation by α. We already know that when α has $o(\sqrt{n})$ blocks, α does not have the ϵ-shifting property, and can therefore not be heavily shifting.

Our current understanding of the joint initial and final weight distribution cannot quite show that α with $\Theta(n)$ blocks are also not heavily shifting. The reason is that we have no handle on the size of the translation term (X_T, Y_T) in the $(1, -1)$ direction. It is possible that the translation is so large most of the time to make α heavily shifting, though we suspect this does not happen.

It is an open problem to figure out which α are heavily shifting.

2.5 Acknowledgements

The author thanks Swastik Kopparty for many useful discussions. The author also would like to thank the referee for pointing out an elementary proof of Lemma 2.3.6.

Chapter 3

A Cauchy-Davenport theorem for general linear maps

3.1 Introduction

Let p be a prime, and let $\mathbb{F}_p$ denote the finite field of integers modulo p. The classical Cauchy-Davenport theorem states that if $A, B \subseteq \mathbb{F}_p$, then the sumset $A + B$ (defined to equal $\{a + b \mid a \in A, b \in B\}$) satisfies the inequality: $|A + B| \geq |A| + |B| - 1$, provided $p \geq |A| + |B| - 1$. It is instructive to compare this with the elementary inequality $|A + B| \geq |A| + |B| - 1$ for $A, B \subseteq \mathbb{R}$ (this has a simple proof using the natural order on $\mathbb{R}$). The Cauchy-Davenport theorem says that this inequality continues to hold mod p, for p large enough.

The Cauchy-Davenport theorem can be seen as a statement about the size of the image of the product set $A \times B$ under the the map $+ : \mathbb{F}_p \times \mathbb{F}_p \to \mathbb{F}_p$. Here we study a similar phenomenon for general linear maps. Let $L : \mathbb{F}_p^n \to \mathbb{F}_p^m$ be an $\mathbb{F}_p$-linear map. For subsets $A_1, \dots, A_n \subseteq \mathbb{F}_p$, we define

$$L(A_1, \dots, A_n) = \{L(a_1, \dots, a_n) \mid a_i \in A_i \text{ for each } i\}.$$

(Equivalently, this is the image of $A_1 \times A_2 \times \ldots \times A_n$ under L.) We are interested in a Cauchy-Davenport theorem for L: given integers $k_1, \dots, k_n$, what is the minimum possible size, over subsets $A_i \subseteq \mathbb{F}_p$ with $|A_i| = k_i$, of $|L(A_1, \dots, A_n)|$? This question is already interesting for the map $L^* : \mathbb{F}_p^3 \to \mathbb{F}_p^2$, given by $L(x, y, z) = (x + y, x + z)$.

Our main theorem, Theorem 3.2.2, gives a lower bound on the size of $L(A_1, \dots, A_n)$. For now we just state an interesting special case of this theorem, where all the $|A_i| = k$. While the bound itself is quite complex, the bound (surprisingly) turns out to be tight for every linear map L when $m = n - 1$.

Theorem 3.1.1. *Let $m < n$, and let $L : \mathbb{F}_p^n \to \mathbb{F}_p^m$ be a linear map with rank m. Let v*

be a nonzero vector in $\ker(L)$ *with minimal support, and let* s *be the size of its support. Let* k *be an integer with* $p \geq 2k-1$.

Then for every $A_1, \ldots, A_n \subseteq \mathbb{F}_p$, *with* $|A_i| = k$ *for all* $i \leq n$, *we have:*

$$|L(A_1, \ldots, A_n)| \geq (k^s - (k-1)^s) \cdot k^{m-s+1}.$$

Some remarks about this theorem:

- If $m = n-1$ and $p \geq 2k-1$, this lower bound is optimal for *every* linear map L. See Lemma 3.2.3.

  If $m = n-1$ and $p < 2k-1$, this lower bound can be violated for *every* linear map L.

- If our sets are taken to be subsets of $\mathbb{R}$ instead of $\mathbb{F}_p$, then for $m = n-1$, an identical lower bound holds for every linear map $L : \mathbb{R}^n \to \mathbb{R}^m$, and this lower bound is optimal for every L. As in the case of the Cauchy Davenport theorem, the lower bound also has an elementary proof using the natural order on $\mathbb{R}$.

- If m is small, and k is large, then the lower bound is approximately $s \cdot k^m$.

Thus for the map $L^* : \mathbb{F}_p^3 \to \mathbb{F}_p^2$ mentioned above, if $p \geq 2k-1$, then for every three sets A_1, A_2, A_3 with $|A_i| = k$, we get that

$$|L^*(A_1, A_2, A_3)| \geq k^3 - (k-1)^3 = 3k^2 - 3k + 1,$$

and this is the best bound possible in term of k.

3.1.1 Proof Outline

Our proof is based on the Combinatorial Nullstellensatz [12], generalizing one of the known proofs of the Cauchy-Davenport theorem.

The Combinatorial Nullstellensatz is an algebraic statement characterizing multivariate polynomials $Q(Y_1, \ldots, Y_n)$ which vanish on a given product set $A_1 \times \ldots \times A_n$ as those polynomials which lie in a certain explicitly given ideal. Let us recall the Combinatorial Nullstellensatz proof [13, 12] of the Cauchy-Davenport theorem. For given sets $A_1, A_2 \subseteq \mathbb{F}_p$, one wants to prove a lower bound on the size of the sumset $C = A_1 + A_2$. Suppose C was small. The key step of this proof is to consider the univariate polynomial $T(X) \in \mathbb{F}_p[X]$, given by:

$$T(X) = \prod_{c \in C} (X - c),$$

and the bivariate polynomial $Q(Y_1, Y_2) \in \mathbb{F}_p[Y_1, Y_2]$ given by:

$$Q(Y_1, Y_2) = T(Y_1 + Y_2) = \prod_{c \in C} (Y_1 + Y_2 - c).$$

Since C is small, T and Q are of low degree. By design, the polynomial Q vanishes on every point $(a_1, a_2) \in A_1 \times A_2$. Thus, by the Combinatorial Nullstellensatz, one concludes that $Q(Y_1, Y_2)$ must lie in a certain ideal. Then, inspecting monomials and using the upper-triangular criterion for linear independence, one shows that no low-degree polynomial of the form $R(Y_1 + Y_2)$ (with $R(X) \in \mathbb{F}_p[X]$) can lie this ideal. Since $Q(Y_1, Y_2) = T(Y_1 + Y_2)$, this a contradiction.

Our proof will follow the same high-level strategy, but with some important differences. If $L(A_1, \ldots, A_n)$ is small, we will find a multivariate polynomial Q of low "complexity" which vanishes on $A_1 \times A_2 \times \ldots \times A_n$, and thus by the Combinatorial Nullstellensatz, it must lie in a certain ideal I. We then use some linear algebra arguments, along with the low complexity of Q, to show that Q cannot lie in I, thus deriving a contradiction.

There are two new technical ingredients that enter the proof. The first ingredient appears in the construction of the polynomial Q. Since the range of L is a high-dimensional vector space, there is no natural way of explictly giving a polynomial vanishing on $C = L(A_1, \ldots, A_n)$. Instead, we will use a dimension argument to show the existence of a suitable polynomial $T(X_1, \ldots, X_m)$ vanishing on C, and define $Q(Y_1, \ldots, Y_n)$ to be $T(L(Y_1, \ldots, Y_n))$. The second ingredient appears in the linear algebra argument showing that Q does not lie in I. In order to make this argument, we will need Q to have a very special kind of monomial structure. This monomial structure is enforced when we choose T; it is because of this requirement that we do not simply take T to be a low-degree polynomial, but instead choose T from a larger space of polynomials satisfying some constraints (this is what we have termed low complexity in the above description).

Organization of this paper

In the next section we give a formal statement of our main result. In Section 3.3 we prove our main result. In Section 3.4 we discuss limitations of our methods to prove an optimal bound in the $m < n-1$ case. We conclude with some open problems.

Notation

We use $[n]$ to denote the set $\{1, 2, \dots, n\}$. For a vector $v \in \mathbb{F}^n$, we define its support, denoted $\mathsf{supp}(v)$ to be the set of its nonzero coordinates, namely $\{i \in [n] \mid v_i \neq 0\}$. We use $\deg(h)$ to denote the total degree of a polynomial h, and $\deg_Y(h)$ to denote the degree in the variable Y of the polynomial h. We say a monomial $\mathcal{M}$ *appears* in a polynomial h if in the standard representation of h as a linear combination of monomials, $\mathcal{M}$ has a nonzero coefficient.

3.2 The main result

We first state our main theorem. It gives, for every linear map $L : \mathbb{F}_p^n \to \mathbb{F}_p^m$, a lower bound on the size of $L(A_1, \dots, A_n)$, in terms of the sizes of $A_1, \dots, A_n$.

Definition 3.2.1. *For a linear map $L : \mathbb{F}_p^n \to \mathbb{F}_p^m$, we define the support-kernel of L to be the set:*

$$\mathsf{suppker}(L) = \{S \subseteq [n] \mid \exists v \in \ker(L), v \neq 0, \text{ with } \mathsf{supp}(v) = S\}.$$

Theorem 3.2.2. *Let p be prime. Let $n \geq 2$ be an integer. Let $m < n$.*

Let $L : \mathbb{F}_p^n \to \mathbb{F}_p^m$ be a linear map of rank m. Let S be a minimal element of $\mathsf{suppker}(L)$. Let S' be a maximal subset of $[n] \setminus S$ such that $2^{S' \cup S} \cap \mathsf{suppker}(L) = \{S\}$.

Let $1 \leq k_1, \dots, k_n \leq p$. Let $k_{\max} = \max_{i \in S} k_i$ and $k_{\min} = \min_{i \in S} k_i$. Suppose $p \geq k_{\max} + k_{\min} - 1$.

Define

$$\lambda = \left(\left(\prod_{i \in S} k_i \right) - \left(\prod_{i \in S} (k_i - 1) \right) \right) \cdot \left(\prod_{i \in S'} k_i \right).$$

Then for every $A_1, \dots, A_n \subseteq \mathbb{F}_p$ with $|A_i| = k_i$, we have:

$$|L(A_1, \dots, A_n)| \geq \lambda.$$

Taking all the k_i to equal k, and observing that S' has size $m+1-s$, we get the theorem stated in the introduction.

The following lemma shows that when $m = n-1$, and $|A_1| = |A_2| = \ldots = |A_n|$, then the above lower bound is the best possible.

Lemma 3.2.3. *Let p be prime. Let $n \geq 2$ be an integer. Let $m = n-1$.*

Let $L : \mathbb{F}_p^n \to \mathbb{F}_p^m$ be a linear map of rank m. Let $S \subseteq [n]$ be the unique element of $\mathsf{suppker}(L)$. Let $S' = [n] \setminus S$, and observe that $2^{S' \cup S} \cap \mathsf{suppker}(L) = S$.

Let $k_1 = k_2 = \ldots = k_n = k$.

Define

$$\lambda = \left(\left(\prod_{i \in S} k_i \right) - \left(\prod_{i \in S} (k_i - 1) \right) \right) \cdot \left(\prod_{i \in S'} k_i \right).$$

Then:

1. *If $p \geq 2k-1$, there exist $A_1, \ldots, A_n \subseteq \mathbb{F}_p$ with $|A_i| = k_i$, such that:*

$$|L(A_1, \ldots, A_n)| = \lambda.$$

2. *If $p < 2k-1$, there exist $A_1, \ldots, A_n \subseteq \mathbb{F}_p$ with $|A_i| = k_i$, such that:*

$$|L(A_1, \ldots, A_n)| < \lambda.$$

3.3 Proof of the main theorem

For a linear map $L : \mathbb{F}_p^n \to \mathbb{F}_p^m$ and integers $k_1, \ldots, k_n$, define:

$$\mu(L, k_1, \ldots, k_n) \overset{\text{def}}{=} \min_{\substack{A_1, A_2, \ldots, A_n \subseteq \mathbb{F}_p \\ |A_i| = k_i}} |L(A_1, \ldots, A_n)|.$$

The proof of the main theorem, Theorem 3.2.2 has two steps. The first step performs elementary operations on the linear map L to bring it into a simple form, while preserving the value of $\mu(L, k_1, \ldots, k_n)$. The second step applies the polynomial method to give a lower bound on $\mu(L, k_1, \ldots, k_n)$ for these simple L. The allowable operations to simplify the linear map are listed in Lemma 3.3.1 and the lower bound for the simpler map is the subject of Theorem 3.3.2.

Lemma 3.3.1. *Let $L : \mathbb{F}_p^n \to \mathbb{F}_p^m$ be a linear map, and let $1 \leq k_1, \ldots, k_n \leq p$.*

1. *Let $L' : \mathbb{F}_p^m \to \mathbb{F}_p^m$ be a full rank linear transformation. Then $\mu(L, k_1, \ldots, k_n) = \mu(L' \circ L, k_1, \ldots, k_n)$.*

2. *Let $L'' : \mathbb{F}_p^n \to \mathbb{F}_p^n$ be a linear map whose matrix is a diagonal matrix with all diagonal entries nonzero. Then $\mu(L, k_1, \ldots, k_n) = \mu(L \circ L'', k_1, \ldots, k_n)$.*

3. *Let $\pi : [n] \to [n]$ be a permutation. Let $L_\pi : \mathbb{F}_p^n \to \mathbb{F}_p^n$ be the linear map that permutes coordinates according to π (i.e.; $L_\pi(e_i) = e_{\pi(i)}$). Then $\mu(L, k_1, \ldots, k_n) = \mu(L \circ L_\pi, k_{\pi^{-1}(1)}, \ldots, k_{\pi^{-1}(n)})$.*

Proof. 1. L' is an isomorphism, so

$$|L' \circ L(A_1, \ldots, A_n)| = |L(A_1, \ldots, A_n)|.$$

Taking the minimum over the choices of the sets $A_i, i \in [n]$, we get $\mu(L, k_1, \ldots, k_n) = \mu(L' \circ L, k_1, \ldots, k_n)$.

2. Applying L'' to $(A_1, \ldots, A_n)$ simply scales the set A_i by a factor of $L''_{i,i}$. In particular, L'' preserves the sizes of the sets. So we have:

$$|L \circ L''(A_1, \ldots, A_n)| = |L(L''_{1,1}A_1, \ldots, L''_{n,n}A_n)| \geq \mu(L, k_1, \ldots, k_n).$$

Taking the minimum over the choices of the sets $A_i, i \in [n]$, we get $\mu(L \circ L'', k_1, \ldots, k_n) \geq \mu(L' \circ L, k_1, \ldots, k_n)$.

For the other direction, observe that any scaling is reversible by an inverse scaling:

$$|L(A_1, \ldots, A_n)| = |L \circ L''(\frac{1}{L''_{1,1}}A_1, \ldots, \frac{1}{L''_{n,n}}A_n)| \geq \mu(L \circ L'', k_1, \ldots, k_n).$$

Taking the minimum over the $A_i, i \in [n]$ gives the reverse inequality.

3. L_π permutes the indices of the sets, and so permutes the sizes of the sets. Taking this into account, the size of the image should remain the same:

$$\begin{aligned} |L \circ L_\pi(A_{\pi^{-1}(1)}, \ldots, A_{\pi^{-1}(n)})| &= |L(A_1, \ldots, A_n)| \geq \mu(L, k_1, \ldots, k_n), \\ |L(A_1, \ldots, A_n)| &= |L \circ L_\pi(A_{\pi^{-1}(1)}, \ldots, A_{\pi^{-1}(n)})| \end{aligned}$$

$$\geq \mu(L \circ L_\pi, k_{\pi^{-1}(1)}, \ldots, k_{\pi^{-1}(n)}).$$

Taking the minimum over the $A_i, i \in [n]$ gives both directions of the inequality.

□

Theorem 3.3.2. *Let p be prime. Let $m \geq 1$ be an integer.*

Let $U_1, U_2, \ldots, U_m, V \subseteq F_p$ be subsets of size $|U_i| = k_i$ for $1 \leq i \leq m$, and $|V| = \hat{k}$. Suppose $p \geq \hat{k} + k_i - 1$ for each i.

Let

$$C = \{(u_1 + v, u_2 + v, \ldots, u_m + v) | u_i \in U_i \text{ for each } i, v \in V\}.$$

Then

$$|C| \geq \hat{k} \cdot \prod_{i=1}^{m} k_i - (\hat{k} - 1) \cdot \prod_{i=1}^{m} (k_i - 1).$$

3.3.1 Preliminaries: multivariate polynomials and Combinatorial Nullstellensatz

In preparation for our proof of Theorem 3.3.2, we recall the statement of the Combinatorial Nullstellensatz, along with some important facts about reducing multivariate polynomials modulo ideals of the kind that arise in the Combinatorial Nullstellensatz.

Lemma 3.3.3 (Combinatorial Nullstellensatz [12])**.** *Let $\mathbb{F}$ be a field, and let $A_1, \ldots, A_n \subseteq \mathbb{F}$. For $i \in [n]$, let $P_i(T) \in \mathbb{F}[T]$ be given by $P_i(T) = \prod_{\alpha \in A_i} (T - \alpha)$.*

Let $h(Y_1, \ldots, Y_n) \in \mathbb{F}[Y_1, \ldots, Y_n]$. Then $h(Y_1, \ldots, Y_n)$ vanishes on $A_1 \times \ldots \times A_n$ if and only if h lies in the ideal generated by $P_1(Y_1), P_2(Y_2), \ldots, P_n(Y_n)$.

Now let $P_1(T), \ldots, P_n(T) \in \mathbb{F}[T]$ be polynomials, with $\deg(P_i) = k_i$. Let I be the ideal generated by $\langle P_i(Y_i) \rangle_{i \in [n]}$.

Given this setup, we now discuss the operation of *reducing a polynomial mod I*. A monomial $\prod_{i=1}^{n} Y_i^{e_i}$ is called *legal* for I if $e_i < k_i$ for each $i \in [n]$. Given a polynomial h, there is a canonical reduction mod I, denoted $\overline{h}$, with the property that $h \equiv \overline{h} \mod I$, and that every monomial appearing in the expansion of $\overline{h}$ is legal for I (equivalently, for each i we have $\deg_{Y_i}(\overline{h}) < k_i$). This canonical reduction can be obtained as follows.

Reducing a polynomial mod $P_i(Y_i) = Y_i^{k_i} - \sum_{j=0}^{k_i-1} a_j Y_i^j$ is simply the act of repeatedly replacing every occurrence of $Y_i^{k_i}$ with $\sum_{j=0}^{k_i-1} a_j Y_i^j$, until the Y_i degree is less than k_i. Reducing the polynomial h mod $P_i(Y_i)$ in succession for each $i \in [n]$ gives the canonical reduction $\overline{h}$.

Here are some important (and easy to verify) points about canonical reduction:

1. $h \in I$ if and only if $\overline{h} = 0$.
2. The map $h \mapsto \overline{h}$ is $\mathbb{F}$-linear.

It will be important for us to understand the degrees of the monomials in $\overline{h}$. Let $\mathcal{M} = \prod_{i=1}^{n} Y_i^{e_i}$ be a monomial, and consider its reduction $\overline{\mathcal{M}}$ mod I. If $e_i < k_i$ for each $i \in [n]$, then we have $\overline{\mathcal{M}} = \mathcal{M}$. Furthermore, if there is some $e_i \geq k_i$, then $\deg \overline{\mathcal{M}} < \deg(\mathcal{M})$. This is because the act of replacing $Y_i^{k_i}$ with a lower degree polynomial in Y_i strictly decreases the degree. Combining these two facts, we get the following fact.

Fact 3.3.4. *With notation as above, let $h(Y_1, \dots, Y_n) \in \mathbb{F}[Y_1, \dots, Y_n]$. Suppose $\mathcal{M}$ is a monomial that (1) appears in h, (2) has $\deg(\mathcal{M}) = \deg(h)$, and (3) is legal for I.*

Then $\mathcal{M}$ appears in the canonical reduction $\overline{h}$.

This is because $\overline{\mathcal{M}} = \mathcal{M}$, and the canonical reductions of the other monomials will have smaller degree than $\mathcal{M}$, and will therefore leave $\mathcal{M}$ untouched.

Very similar considerations give us the following related fact.

Fact 3.3.5. *With notation as above, let $h(Y_1, \dots, Y_n) \in \mathbb{F}[Y_1, \dots, Y_n]$. Suppose $\mathcal{M}$ is a monomial that (1) appears in $\overline{h}$, (2) has $\deg(\mathcal{M}) = \deg(h)$, and (3) is legal for I.*

Then $\mathcal{M}$ appears in h.

3.3.2 Correlated sumsets and the polynomial method

We now prove Theorem 3.3.2.

Proof. We begin by defining some sets of monomials which will be useful to us.

In the polynomial ring $\mathbb{F}_p[Y_1, \dots, Y_m, Z]$, consider the following set of monomials:

$$\Gamma = \{Y_1^{e_1}Y_2^{e_2}\cdots Y_m^{e_m}Z^e | 0 \le e_i \le k_i - 1 \text{ for each } i, \text{ and } 0 \le e \le \hat{k}-1,$$
$$\text{and } e > 0 \Rightarrow e_i = k_i - 1 \text{ for some } i\}.$$

We will also consider the polynomial ring $\mathbb{F}_p[X_1, \ldots, X_m]$. To each monomial $\mathcal{M}(Y_1, \ldots, Y_m, Z) \in \Gamma$, we associate a monomial $\phi(\mathcal{M}) \in \mathbb{F}_p[X_1, \ldots, X_m]$ as follows. If $\mathcal{M}(Y_1, \ldots, Y_m, Z) = Y_1^{e_1}Y_2^{e_2}\cdots Y_m^{e_m}Z^e$, then define:

$$\phi(\mathcal{M}) = \begin{cases} \prod_{i=1}^m X_i^{e_i} & \text{if } e = 0 \\ \left(\prod_{i=1}^m X_i^{e_i}\right) \cdot X_j^e & \text{if } e > 0, \text{ where } j \text{ is the first index} \\ & \quad \text{so that } e_j = k_j - 1. \end{cases}$$

Let $\Delta = \{\phi(\mathcal{M}) \mid \mathcal{M} \in \Gamma\}$ be the set of all such monomials constructed in this way.

Note that ϕ is a bijection, and ϕ preserves the degree of each monomial. Thus, ϕ also gives a bijection when we restrict to monomials in Γ and Δ of fixed total degree. We defined ϕ so that ϕ^{-1} would have the following description: Let $X_1^{f_1}X_2^{f_2}\cdots X_m^{f_m} \in \Delta$. Let $e_i = \min\{f_i, k_i - 1\}$ for each $i \in [m]$. Let $e = \sum_{i=1}^m f_i - \sum_{i=1}^m e_i$. Then

$$\phi^{-1}(X_1^{f_1}X_2^{f_2}\cdots X_m^{f_m}) = Y_1^{e_1}Y_2^{e_2}\cdots Y_m^{e_m} \cdot Z^e.$$

Note that by choice of e, ϕ^{-1} preserves degree.

With these definitions in hand, we proceed with the main parts of the proof.

Interpolating a polynomial

Suppose for contradiction that $|C| < \hat{k} \cdot \left(\prod_{i=1}^m k_i\right) - (\hat{k}-1) \cdot \left(\prod_{i=1}^m (k_i - 1)\right)$. Since $|\Delta| = |\Gamma| = \hat{k} \cdot \prod_{i=1}^m k_i - (\hat{k}-1) \cdot \prod_{i=1}^m (k_i - 1)$, there is a non-zero polynomial $f(X_1, \ldots, X_m) = \sum_{\mathcal{K} \in \Delta} c_{\mathcal{K}} \mathcal{K}(X_1, \ldots, X_m)$ which vanishes on C. By the definition of C, this means that $g(Y_1, \ldots, Y_m, Z) \overset{\text{def}}{=} f(Y_1 + Z, \ldots, Y_m + Z)$ is a non-zero polynomial vanishing on every point $(u_1, u_2, \ldots, u_m, v) \in \prod_{i=1}^m U_i \times V$.

Application of the Combinatorial Nullstellensatz

For each $1 \leq i \leq m$, let $P_i(Y_i) = \prod_{a \in U_i}(Y_i - a)$. Also let $P(Z) = \prod_{a \in V}(Z - a)$.

By the Combinatorial Nullstellensatz,

$$g(Y_1, \ldots, Y_m, Z) \equiv 0 \pmod{I},$$

where I is the ideal generated by the $P_i(Y_i), i \in [m]$ and $P(Z)$.

Explicitly, we have that:

$$\sum_{\mathcal{K} \in \Delta} c_{\mathcal{K}} \mathcal{K}(Y_1 + Z, Y_2 + Z, \ldots, Y_m + Z) \equiv 0 \pmod{I},$$

where at least one $c_{\mathcal{K}}$ is nonzero.

Consider the canonical reduction $\overline{g}$ of $g \mod I$: since $g \in I$ we get that $\overline{g} = 0$. On the other hand, we have by linearity of canonical reduction:

$$\overline{g} = \sum_{\mathcal{K} \in \Delta} c_{\mathcal{K}} \overline{\mathcal{K}}(Y_1, Y_2, \ldots, Y_m, Z),$$

where $\overline{\mathcal{K}}(Y_1, Y_2, \ldots, Y_m, Z)$ is the canonical reduction mod I of $\mathcal{K}(Y_1+Z, Y_2+Z, \ldots, Y_m+Z)$. By Fact 3.3.4, any monomial $\mathcal{M}$ that appears in the expansion of $\mathcal{K}(Y_1 + Z, Y_2 + Z, \ldots, Y_m+Z)$ with $\deg(\mathcal{M}) = \deg(\mathcal{K})$ and is legal for I, also appears in $\overline{\mathcal{K}}(Y_1, Y_2, \ldots, Y_m, Z)$.

Arriving at a contradiction

We may now summarize the strategy for the rest of the proof. We will first find an ordering of the monomials in Δ such that:

1. If $\mathcal{K}, \mathcal{K}'$ are monomials in Δ with $\deg(\mathcal{K}') < \deg(\mathcal{K})$, then $\mathcal{K}'$ is smaller than $\mathcal{K}$ in the ordering.

2. For each $\mathcal{K} \in \Delta$, there is some monomial $\mathcal{M}_{\mathcal{K}}(Y_1, \ldots, Y_m, Z)$ with the following four properties:

   (a) $\mathcal{M}_{\mathcal{K}}$ appears the expansion of $\mathcal{K}(Y_1 + Z, Y_2 + Z, \ldots, Y_m + Z)$,

   (b) $\deg(\mathcal{M}_{\mathcal{K}}) = \deg(\mathcal{K})$,

   (c) $\mathcal{M}_{\mathcal{K}}$ is legal for I,

(d) $\mathcal{M}_{\mathcal{K}}$ does not appear in the expansion of $\mathcal{K}'(Y_1+Z, Y_2+Z, \ldots, Y_m+Z)$ for any $\mathcal{K}' \in \Delta$ smaller than $\mathcal{K}$ in the ordering.

Once we have such an ordering, consider the largest $\mathcal{K}$ in the ordering for which $c_{\mathcal{K}} \neq 0$. By Fact 3.3.4, $\mathcal{M}_{\mathcal{K}}$ appears in $\overline{\mathcal{K}}(Y_1, \ldots, Y_m, Z)$. For every other $\mathcal{K}' \in \Delta$ with $c_{\mathcal{K}'} \neq 0$, we will show that $\overline{\mathcal{K}'}(Y_1, \ldots, Y_m, Z)$ does not include the monomial $\mathcal{M}_{\mathcal{K}}$; this then shows that $\mathcal{M}_{\mathcal{K}}$ appears in $\overline{g}$ with a nonzero coefficient, contradicting our equation $\overline{g} = 0$. This gives the desired contradiction.

Monomial $\mathcal{M}_{\mathcal{K}}$ does not appear in $\overline{\mathcal{K}'}$ (for $\mathcal{K}' \neq \mathcal{K}$ with $c_{\mathcal{K}'} \neq 0$)

Suppose $\mathcal{K}' \in \Delta$, $\mathcal{K}' \neq \mathcal{K}$ and $c_{\mathcal{K}'} \neq 0$. We will show that $\mathcal{M}_{\mathcal{K}}$ does not appear in $\overline{\mathcal{K}'}$. By choice of $\mathcal{K}$, we have that $\mathcal{K}'$ is smaller than $\mathcal{K}$ in the ordering, and hence that $\deg(\mathcal{K}') \leq \deg(\mathcal{K})$.

Suppose $\mathcal{M}_{\mathcal{K}}$ appeared in $\overline{\mathcal{K}'}$. Then the following chain of inequalities:

$$\deg(\mathcal{M}_{\mathcal{K}}) \leq \deg(\overline{\mathcal{K}'}) \leq \deg(\mathcal{K}'(Y_1+Z, \ldots, Y_m+Z) \leq \deg(\mathcal{K}') \leq \deg(\mathcal{K}) = \deg(\mathcal{M}_{\mathcal{K}}),$$

(because of the equality of the endpoints, this is a chain of equalities), shows that $\deg(\mathcal{M}_{\mathcal{K}}) = \deg(\mathcal{K}'(Y_1+Z, \ldots, Y_m+Z))$. Thus by Fact 3.3.5, we can conclude that $\mathcal{M}_{\mathcal{K}}$ appears in $\mathcal{K}'(Y_1+Z, \ldots, Y_m+Z)$. But this contradicts the property that $\mathcal{M}_{\mathcal{K}}$ does not appear in $\mathcal{K}'(Y_1+Z, \ldots, Y_m+Z)$ for any $\mathcal{K}' \in \Delta$ that is smaller than $\mathcal{K}$ in the ordering. Thus $\mathcal{M}_{\mathcal{K}}$ cannot appear in $\overline{\mathcal{K}'}$.

The ordering of Δ

All that remains now is to define the ordering of Δ, and to prove the desired properties of this ordering.

Arrange the monomials in Δ in order of increasing total degree. Within each fixed total degree, order by decreasing $\deg_Z(\phi^{-1}(\mathcal{K}(X_1, \ldots, X_m)))$. Then for $\mathcal{K}(X_1, \ldots, X_m) \in \Delta$ in that ordering, set $\mathcal{M}_{\mathcal{K}} = \phi^{-1}(\mathcal{K}(X_1, \ldots, X_m)) = Y_1^{e_1} Y_2^{e_2} \cdots Y_m^{e_m} Z^e$. We claim that $\mathcal{M}_{\mathcal{K}}$ satisfies the four properties listed above.

(a) $\mathcal{M}_{\mathcal{K}}$ *appears the expansion of* $\mathcal{K}(Y_1+Z, Y_2+Z, \ldots, Y_m+Z)$:

We show that the coefficient of $\mathcal{M}_{\mathcal{K}}$ in $\mathcal{K}(Y_1 + Z, Y_2 + Z, \ldots, Y_m + Z)$ is non-zero. By the definition of Δ, $\deg_{X_i}(\mathcal{K}) \leq \hat{k} + k_i - 2 < p$ for $i \in [m], \mathcal{K} \in \Delta$. Also, there is at most one i such that $\deg_{X_i}(\mathcal{K}) > k_i - 1$. Call this index j if it exists, and let $l = \deg_{X_j}(\mathcal{K})$. Since $\phi^{-1}(\mathcal{K})$ extracts the largest powers of Y_i in $\mathcal{K}(Y_1 + Z, Y_2 + Z, \ldots, Y_m + Z)$ up to $k_i - 1$ for $i \in [m]$, we get that the coefficient of $\mathcal{M}_{\mathcal{K}}$ is 1 if j does not exist and $\binom{l}{k_j-1}$ if j exists. In both cases, the coefficient of $\mathcal{M}_{\mathcal{K}}$ is non-zero in F_p as $l < p$.

(b) $\deg(\mathcal{M}_{\mathcal{K}}) = \deg(\mathcal{K})$:

Recall that ϕ is a bijection from one set of monomials to another which preserves the degree of the monomials. So $\deg(\mathcal{M}_{\mathcal{K}}) = \deg \phi^{-1}(\mathcal{K}(X_1, \ldots, X_m)) = \deg(\mathcal{K})$.

(c) *$\mathcal{M}_{\mathcal{K}}$ is legal for I*:

Recall that writing $\mathcal{K} = X_1^{f_1} X_2^{f_2} \cdots X_m^{f_m}$, we have

$$\phi^{-1}(\mathcal{K}) = Y_1^{e_1} Y_2^{e_2} \cdots Y_m^{e_m} \cdot Z^e,$$

where $e_i = \min\{f_i, k_i - 1\}$ for each $i \in [m]$, and $e = \sum_{i=1}^{m} f_i - \sum_{i=1}^{m} e_i$. So $e_i \leq k_i - 1$, $\forall i \in [m]$. It remains to show that $e \leq \hat{k} - 1$. Suppose $f_i \leq k_i - 1$, $\forall i \in [m]$. Then $e_i = f_i$, $\forall i \in [m]$ and so $e = 0$. Otherwise, $f_i \leq k_i - 1$ for all but one $i \in [m]$, call this index j. We have $e_i = f_i$ for $i \neq j$ and $e_j = k_j - 1$. So

$$e = f_j - e_j \leq \hat{k} + k_j - 2 - (k_j - 1) = \hat{k} - 1.$$

(d) *$\mathcal{M}_{\mathcal{K}}$ does not appear in the expansion of $\mathcal{K}'(Y_1 + Z, Y_2 + Z, \ldots, Y_m + Z)$ for any $\mathcal{K}' \in \Delta$ smaller than $\mathcal{K}$ in the ordering*:

To show that the monomials selected by ϕ^{-1} do not appear in any previous entries of the ordering, first note that the degree of $\mathcal{M}_{\mathcal{K}}$ is too large to have appeared in any previous $\mathcal{K}' \in \Delta$ of lower total degree. Next, consider the expansion of a previous $\mathcal{K}'(Y_1 + Z, Y_2 + Z, \ldots, Y_m + Z)$ in the ordering of the same total degree, then $\mathcal{M}_{\mathcal{K}'} = \phi^{-1}(\mathcal{K}'(X_1, \ldots, X_m)) = Y_1^{e'_1} Y_2^{e'_2} \cdots Y_m^{e'_n} Z^{e'}$ must have $e'_i < e_i \leq k_i - 1$ for some $i \in [m]$, as $e' \geq e$. By the way ϕ is defined, this means that $\deg_{X_i}(\mathcal{K}') = e'_i$, so $\deg_{Y_i}(\mathcal{K}'(Y_1+Z, Y_2+Z, \ldots, Y_m+Z)) = e'_i$. But $\deg_{Y_i}(\mathcal{M}_{\mathcal{K}}) = e_i > e'_i$. So $\mathcal{M}_{\mathcal{K}}$ cannot be a monomial in the expansion of $\mathcal{K}'(Y_1+Z, Y_2+Z, \ldots, Y_m+Z)$.

This completes the proof that the ordering of Δ has the desired properties, and hence we arrive at a contradiction.

Thus we must have that $|C| \geq \hat{k} \cdot \left(\prod_{i=1}^{m} k_i\right) - (\hat{k}-1) \cdot \left(\prod_{i=1}^{m}(k_i-1)\right)$. □

3.3.3 Proving the main result

We now combine Lemma 3.3.1 and Theorem 3.3.2 to prove our main theorem, Theorem 3.2.2.

Proof of Theorem 3.2.2:

By basic linear algebra, we have that $|S| \leq m+1$, and $|S \cup S'| = m+1$.

We first get rid of the coordinates in $[n] \setminus (S \cup S')$. Observe that taking away elements from any of the sets A_i cannot increase the size of the image $|L(A_1, \ldots, A_n)|$. Let $\mathbf{a} \in \prod_{i \in [n] \setminus (S \cup S')} A_i$. Fix the coordinates in $[n] \setminus (S \cup S')$ to $\mathbf{a}$ and consider the resulting map $M : \mathbb{F}_p^{m+1} \to \mathbb{F}_p^m$ (i.e., $M(\mathbf{x}) = L(\mathbf{x}, \mathbf{a})$). If $L' : \mathbb{F}_p^{m+1} \to \mathbb{F}_p^m$ is the linear map obtained by restricting the coordinates of $[n] \setminus (S \cup S')$ to 0, then the image $L'\left(\prod_{i \in S \cup S'} A_i\right)$ is a translate of the image of $M\left(\prod_{i \in S \cup S'} A_i\right)$. So we have:

$$|L(A_1, \ldots, A_n)| \geq \left| M\left(\prod_{i \in S \cup S'} A_i\right)\right| = L'\left(\prod_{i \in S \cup S'} A_i\right).$$

Then a lower bound on $L'\left(\prod_{i \in S \cup S'} A_i\right)$ gives a lower bound on $|L(A_1, \ldots, A_n)|$.

The next step is to use the simple transformations in Lemma 3.3.1 to greatly simplify our linear map L', while preserving $\mu(L, k_1, \ldots, k_n)$. The transformations allow us to apply elementary row operations on L', scale the columns of L', and rearrange the columns of L'.

As L' has rank m, $\ker(L')$ has rank 1. Consider a nonzero vector $v \in \ker(L')$. Then S must be the support of v. Let $\hat{\imath}$ be the index in S that minimizes k_i, i.e. $\hat{\imath} = \arg\min_{i \in S} k_i$.

With the above row and column operations at our disposal, we perform the following reduction of the problem. First, permute the columns so that the columns with indices in S are on the left and move column $\hat{\imath}$ so that it is the first column. Then the last m columns are now linearly independent. This is because if they were linearly dependent, there would be a nonzero vector in the kernel of L whose support does not include $\hat{\imath}$. So there would be two nonzero vectors in $\ker(L)$ with different supports, which is impossible. Next, apply the sequence of elementary row operations that turns the last m columns into the identity matrix. Scale each row so that the first element is either 0 or 1. Finally, scale each of the last m columns so that they again form the identity matrix. We are left with a column of 1's and 0's followed by the m by m identity matrix. We will call this matrix $\hat{L}'$, the reduction of L'.

$$\hat{L}' = \left(\begin{array}{c|c} 1 & \\ \vdots & \\ 1 & I_m \\ 0 & \\ \vdots & \\ 0 & \end{array}\right).$$

Considering the projection P of the image of $\hat{L}'(\prod_{i\in S} A_i, \prod_{i\in S'} A_i)$ onto the first $|S|-1$ coordinates, we find ourselves in the setting of Theorem 3.3.2. Letting $U = A_{\hat{\imath}}$, and $\{V_1, \ldots, V_{|S|-1}\} = \{A_i \mid i \in S - \{\hat{\imath}\}\}$, Theorem 3.3.2 tells us that

$$|P| \geq \prod_{i\in S} k_i - \prod_{i\in S}(k_i - 1).$$

Finally, note that as $\mathbf{a}'$ varies in the set $\prod_{i\in S'} A_i$, the sets $\hat{L}'(U, V_1, \ldots, V_{|S|-1}, \mathbf{a}')$ are all translates of P and are disjoint (the disjointness follows from the fact that $\mathsf{suppker}(L) \cap 2^{S\cup S'} = \{S\}$). Hence, the total size of the image of $\hat{L}'$ is at least $|P| \cdot \prod_{i\in S'} |A_i|$, which is at least:

$$\left(\left(\prod_{i\in S} k_i\right) - \left(\prod_{i\in S}(k_i - 1)\right)\right) \cdot \left(\prod_{i\in S'} k_i\right),$$

as desired.

□

Proof of Lemma 3.2.3:

We first provide a tight example for our lower bound when $p \geq 2k-1$. Using the same transformations as above, we produce the simple linear transformation $\hat{L}$ from L. Lemma 3.3.1 implies that providing a tight example for $\hat{L}$ implies the existence of a tight example for L. We claim that setting $A_i = \{0, \ldots, k_i - 1\}$ attains the smallest possible image size $\left(\prod_{i \in S} k_i - \prod_{i \in S}(k_i - 1)\right) \cdot \prod_{i \in S'} k_i$.

As before, every choice of $\mathbf{a} \in \prod_{i \notin S} A_i$ yields $|P|$ distinct points in the image of $\hat{L}$, where P is the projection of $\hat{L}(\prod_{i \in S} A_i, \prod_{i \in S'} A_i)$ onto the first $|S| - 1$ coordinates. So it suffices to show that $|P| \geq \left(\prod_{i \in S} k_i - \prod_{i \in S}(k_i - 1)\right)$. This is equivalent to showing that equality is attained in Theorem 3.3.2 when the sets are all taken to be intervals starting from 0.

Suppose we have sets $U_i = \{0, \ldots, k_i - 1\}, i \in [m]$ and $V = \{0, \ldots, \hat{k} - 1\}$. We want to show that $C = \{(u_1 + v, u_2 + v, \ldots, u_m + v) | u_i \in U_i$ for each $i \in [m], v \in V\}$ has size exactly equal to $\hat{k} \cdot \prod_{i=1}^{m} k_i - (\hat{k} - 1) \cdot \prod_{i=1}^{m} (k_i - 1)$ as long as $p \geq \hat{k} + k_i - 1$. In particular, this will give a tight example when the set sizes are all the same.

Let $C_j = \{(u_1 + j, u_2 + j, \ldots, u_m + j) | u_i \in U_i$ for each $i \in [m]\}, j = 0, \ldots, \hat{k} - 1$. Then $C = \bigcup_{j=0}^{\hat{k}-1} C_j$. We start with $|C_0| = \prod_{i=1}^{m} k_i$, and ask how many additional elements we add when we take the union with C_1:

$$\begin{aligned} |C_1 - C_0| &= |C_1| - |C_1 \cap C_0| \\ &= \prod_{i=1}^{m} k_i - \prod_{i=1}^{m} (k_i - 1). \end{aligned}$$

Since $p \geq \hat{k} + k_i - 1$, none of the sums that we take exceed $p - 1$, so we will continue to add $\prod_{i=1}^{m} k_i - \prod_{i=1}^{m} (k_i - 1)$ for each successive C_j. Total this gives $\prod_{i=1}^{m} k_i + (\hat{k} - 1) \cdot \left(\prod_{i=1}^{m} k_i - \prod_{i=1}^{m} (k_i - 1) \right)$, which is equal to $\hat{k} \cdot \prod_{i=1}^{m} k_i - (\hat{k} - 1) \cdot \prod_{i=1}^{m} (k_i - 1)$.

We now show that the lower bound is not tight when $p < 2k - 1$. In fact, the same example of taking the sets to be intervals will produce an image whose size is strictly smaller than our lower bound. Let $U_i = \{0, \ldots, k - 1\}, i \in [m]$ and $V = \{0, \ldots, k - 1\}$

in the statement of Theorem 3.3.2. We want to show that $C = \{(u_1 + v, u_2 + v, \ldots, u_m + v) | u_i \in U_i \text{ for each } i \in [m], v \in V\}$ has size strictly less than $k^{m+1} - (k-1)^{m+1}$.

As before, let $C_j = \{(u_1 + j, u_2 + j, \ldots, u_m + j) | u_i \in U_i \text{ for each } i \in [m]\}, j = 0, \ldots, k-1$. Then $C = \bigcup_{j=0}^{k-1} C_j$. Note that the element $(k-1+p-k+1, \ldots, k-1+p-k+1, k-1+p-k+1) = (0, \ldots, 0) \in C_{p-k+1}$ is in C_0. But this was one of the "new" elements of C_{p-k+1} that we counted in the argument for the tight example, which was previously not in any of the C_i, for $i < p-k+1$. Hence, the number $k^{m+1} - (k-1)^{m+1}$ is a strict overcount for the number of elements in the image.

$\square$

3.4 Linear maps of smaller rank

Our lower bound in the general case $n > m-1$ is not tight for every linear map. The main reason for this is that our proof strategy only uses information about the support of vectors in the kernel of L (and not the actual vectors). As the following example shows, if $m < n-1$ the optimal lower bound for $L(A_1, \ldots, A_n)$ may not be determined solely be the set of all supports of vectors in $\ker(L)$.

Example 3.4.1. *Let p be a large prime, and let $k \ll p$. Consider the following 2×4 matrices over $\mathbb{F}_p$:*

$$M = \begin{bmatrix} 1 & 0 & 2 & 1 \\ 0 & 1 & 1 & 2 \end{bmatrix},$$

$$M' = \begin{bmatrix} 1 & 0 & 100 & 1 \\ 0 & 1 & 1 & 100 \end{bmatrix}.$$

Define $L : \mathbb{F}_p^4 \to \mathbb{F}_p^2$ and $L' : \mathbb{F}_p^4 \to \mathbb{F}_p^2$ by $L(x) = Mx$ and $L'(x) = M'x$. Observe that $\mathsf{suppker}(L)$ and $\mathsf{suppker}(L')$ both equal $\binom{[4]}{\geq 3}$.

Letting $A_1, A_2, A_3, A_4 = \{1, 2, \ldots, k\} \subseteq \mathbb{F}_p$, then $|L(A_1, A_2, A_3, A_4)| \leq 16k^2$.

In contrast, we will show in Lemma 3.4.3 that $|L'(A_1', A_2', A_3', A_4')| \geq 100k^2$, for any k-elements sets $A_1', A_2', A_3', A_4' \subseteq \mathbb{F}_p$,

Our analysis of this example will use some results on "sums of dilates". For a constant λ and a set A, we define the *dilate* λA denote the set $\{\lambda a \mid a \in A\}$. We will use the following result of Pontiveros [15] (which builds on a beautiful result of Bukh [14]) on sums of dilates in $\mathbb{Z}_p$.

Lemma 3.4.2. *For every coprime $\lambda_1, \cdots, \lambda_n \in \mathbb{Z}$, there exists a constant $\alpha > 0$ such that $|\lambda_1 X + \lambda_2 X + \cdots + \lambda_n X| \geq \left(\sum \lambda_i\right) \cdot |X| - o(|X|)$, for sufficiently large prime p, and every $X \subseteq \mathbb{Z}_p$, with $|X| \leq \alpha p$.*

We use this estimate on the size of the sum of dilates, to construct linear maps with arbitrarily large image.

Lemma 3.4.3. *For every positive integer constant c, there is a linear map $L : \mathbb{F}_p^4 \to \mathbb{F}_p^2$ such that for every $A_1, A_2, A_3, A_4 \subseteq \mathbb{F}_p$ with $|A_i| = k$, and any prime p sufficiently larger than k, we have:*

$$L(A_1, A_2, A_3, A_4) \geq ck^2.$$

Proof. Consider the linear map

$$L(A_1, A_2, A_3, A_4) = \{(a_1 + c \cdot a_3 + a_4,\ a_2 + a_3 + c \cdot a_4) | \,(a_1, a_2, a_3, a_4) \in A_1 \times A_2 \times A_3 \times A_4\}$$

and let $A_1, A_2, A_3, A_4 \subseteq F_p$ be any k-elements sets.

By Ruzsa triangle inequality [16],

$$|A_4 + c^2 A_4| \leq \frac{|A_4 + cA_3|\,|cA_3 + c^2 A_4|}{|cA_3|} = \frac{|A_4 + cA_3|\,|A_3 + cA_4|}{|A_3|}$$

From Lemma 3.4.2 we know that $|A_4 + c^2 A_4| \geq c^2 |A_4|$, assuming p sufficiently larger than $k = |A_4|$.

Hence $|A_4 + cA_3| \cdot |A_3 + cA_4| \geq c^2 |A_4||A_3| = c^2 k^2$.

Without loss of generality, assume that $|A_3 + cA_4| \geq ck$.

In particular, fixing $a_2 \in A_2$, and an element $(a_3 + c \cdot a_4) \in A_3 + cA_4$, the subset $\{(a_1 + (c \cdot a_3 + a_4), a_2 + a_3 + c \cdot a_4) | a_1 \in A_1\}$ has at least k elements, all with the same second coordinate. Therefore holding some element $a_2 \in A_2$ fixed, and letting $a_3 \in A_3, a_4 \in A_4$ be any elements, we obtain $|A_3 + cA_4|$ distinct second coordinates, and so

$$|\{(a_1 + c \cdot a_3 + a_4, a_2 + a_3 + c \cdot a_4)\} | a_1 \in A_1, a_3 \in A_3, a_4 \in A_4\}| \geq ck^2$$

We conclude that $|L(A_1, A_2, A_3, A_4)| \geq ck^2$. □

3.5 Questions

We conclude with some interesting open questions.

1. The main open question is to obtain the best bound for the Cauchy-Davenport problem for every linear map.

2. Even for the case $m = n - 1$ and all the k_i equal to k, we do not know the optimal bound for the Cauchy-Davenport problem when $p < 2k - 1$. Our method can be extended to give a better bound, but we believe that this is not the optimal bound.

3. What can be said about the "symmetric" Cauchy-Davenport problem: what is smallest possible size of $L(A, A, \ldots, A)$ over all sets A with $|A| = k$? This seems to be closely related to the theory of sums of dilates.

4. Even over $\mathbb{R}$, finding the optimal bound for the Cauchy-Davenport problem for every linear map seems nontrivial.

5. It will be interesting to study analogues of other theorems of additive combinatorics in the setting of linear maps.

Chapter 4

Decoding Reed-Muller codes over product sets

4.1 Introduction

Error-correcting codes based on polynomials have played an important role throughout the history of coding theory. The mathematical phenomenon underlying these codes is that distinct low-degree polynomials have different evaluations at many points. More recently, the intimate relation between polynomials and computation has led to polynomial-based error-correcting codes having a big impact on complexity theory. Notable applications include PCPs, interactive proofs, polynomial identity testing and property testing.

Our main result is a decoding algorithm for multivariate polynomial codes. Let $\mathbb{F}$ be a field, let $S \subseteq \mathbb{F}$, let $d < |S|$ and let $m \geq 1$. Consider the code of all m-variate polynomials of total degree at most d, evaluated at all points of S^m:

$$\mathcal{C} = \{\langle P(\mathbf{a})\rangle_{\mathbf{a}\in S^m} \mid P(X_1, \ldots, X_m) \in \mathbb{F}[X_1, \ldots, X_m], \deg(P) \leq d\}.$$

When $m = 1$, this code is known as the Reed-Solomon code [19], and for $m > 1$ this code is known as the Reed-Muller code [17, 18][1].

The code $\mathcal{C}$ above is a subset of $\mathbb{F}^{S^m}$, which we view as the space of functions from S^m to $\mathbb{F}_q$. Given two functions $f, g : S^m \to \mathbb{F}$, we define their (relative Hamming) distance $\Delta(f, g) = \Pr_{\mathbf{a}\in S^m}[f(\mathbf{a}) \neq g(\mathbf{a})]$. To understand the error-correcting properties of $\mathcal{C}$, we recall the following well known lemma, often called the Schwartz-Zippel lemma:

Lemma 4.1.1. *Let $\mathbb{F}$ be a field, and let $P(X_1, \ldots, X_m)$ be a nonzero polynomial over*

[1]The family of Reed-Muller codes also includes polynomial evaluation codes where the individual degree d is larger than $|S|$, and the individual degree is capped to be at most $|S| - 1$. We do not consider the $d \geq |S|$ case in this paper.

$\mathbb{F}$ *with degree at most d. Then for every $S \subseteq \mathbb{F}$,*

$$\Pr_{\mathbf{a} \in S^m}[P(\mathbf{a}) = 0] \leq \frac{d}{|S|}.$$

This lemma implies that for any two polynomials P, Q of degree at most d, $\Delta(P, Q) \geq (1 - \frac{d}{|S|})$. In other words the minimum distance of $\mathcal{C}$ is at least $(1 - \frac{d}{|S|})$. It turns out that the minimum distance of $\mathcal{C}$ is in fact exactly $(1 - \frac{d}{|S|})$, and we let $\delta_{\mathcal{C}}$ denote this quantity.

For error-correcting purposes, if we are given a "received word" $r : S^m \to \mathbb{F}$ such that there exists a polynomial P of degree at most d with $\Delta(r, P) \leq \delta_{\mathcal{C}}/2$, then we know that there is a unique such P. The problem that we consider in this paper, "decoding $\mathcal{C}$ upto half its minimum distance", is the algorithmic task of finding this P.

4.1.1 Our Results

There is a rich history with several deep algebraic ideas surrounding the problem of decoding multivariate polynomial codes. We first state our main results, and then discuss its relationship to the various other known results.

Theorem 4.1.2 (Efficient decoding of multivariate polynomial codes upto half their minimum distance)**.** *Let $\mathbb{F}$ be a finite field, let S, d, m be as above, and let $\delta_{\mathcal{C}} = (1 - \frac{d}{|S|})$.*

There is an algorithm, which when given as input a function $r : S^m \to \mathbb{F}$, runs in time $\mathsf{poly}(|S|^m, \log |\mathbb{F}|)$ finds the polynomial $P(X_1, \ldots, X_m) \in \mathbb{F}[X_1, \ldots, X_m]$ of degree at most d (if any) such that:

$$\Delta(r, P) < \delta_{\mathcal{C}}/2.$$

As we will discuss below, previously known efficient decoding algorithms for these codes only either worked for (1) very algebraically special sets S, or (2) very low degrees d, or (3) decoded from a much smaller fraction of errors ($\approx \frac{1}{m+1}\delta_{\mathcal{C}}$ instead of $\frac{1}{2}\delta_C$).

Using several further ideas, we also show how to implement the above algorithm in near-linear time to decode upto almost half the minimum distance, provided d is not $(1 - o(1))|S|$.

Theorem 4.1.3 (Near-linear time decoding)**.** *Let $\mathbb{F}$ be a finite field, let S, d, m be as above, and let $\delta_C = (1 - \frac{d}{|S|})$. Assume $\delta_C > 0$ is a constant.*

There is an algorithm, which when given as input a function $r : S^m \to \mathbb{F}$, runs in time $|S|^m \cdot \mathsf{poly}(\log |S|^m, \log |\mathbb{F}|)$ finds the polynomial $P(X_1, \ldots, X_m) \in \mathbb{F}[X_1, \ldots, X_m]$ of degree at most d (if any) with:

$$\Delta(r, P) < (1 - o(1)) \cdot \delta_C/2.$$

Over the rational numbers, we get a version of Theorem 4.1.2 where the running time is $\mathsf{poly}(|S|^m, t)$, where t is the maximum bit-complexity of any point in S or in the image of r. This enables us to decode multivariate polynomial codes upto half the minimum distance in the natural special case where the evaluation set S equals $\{1, 2, \ldots, n\}$.

We also mention that decoding Reed-Muller codes over an arbitrary product set S^m appears as a subroutine in the local decoding algorithm for multiplicity codes [33] (see Section 4 on "Solving the noisy system"). Our results allow the local decoding algorithms there to run efficiently over all fields ([33] could only do this over fields of small characteristic, where algebraically special sets S are available).

4.1.2 Related work

There have been many works studying the decoding of multivariate polynomial codes, which prove (and improve) various special cases of our main theorem.

Reed-Solomon codes ($m = 1$):

When $m = 1$, our problem is also known as the problem of decoding Reed-Solomon codes upto half their minimum distance. That this problem can be solved efficiently is very classical, and a number of algorithms are known for this (Mattson-Solomon [21], Berlekamp-Massey [20], Berlekamp-Welch [28]). The underlying algorithmic ideas have subsequently had a tremendous impact on algebraic algorithms.

For Reed-Solomon codes, it is in fact known how to list-decode beyond half the minimum distance, upto the Johnson bound (Guruswami-Sudan [22]). This has had numerous further applications in coding theory, complexity theory and pseudorandomness.

Special sets S:

For very special sets S, it turns out that there are some algebraic ways to reduce the decoding of multivariate polynomial codes over S^m to the decoding of univariate polynomial codes. This kind of reduction is possible when S equals the whole field $\mathbb{F}$, or more generally when S equals an affine subspace over the prime subfield of $\mathbb{F}$.

When $S = \mathbb{F}_q$, then $S^m = \mathbb{F}_q^m$ and S^m can then be identified with the large field $\mathbb{F}_{q^m}$ in a natural $\mathbb{F}_q$-linear way (this understanding of Reed-Muller codes was discovered by [24]). This converts the multivariate setting into univariate setting, identifies the multivariate polynomial code as a subcode of the univariate polynomial code, and (somewhat miraculously), the minimum distance of the univariate polynomial code equals the minimum distance of the multivariate polynomial code. Thus the classical Reed-Solomon decoding algorithms can then be used, and this leads to an algorithm for the multivariate setting decoding upto half the minimum distance. In fact, Pellikaan-Wu [23] observed that this connection allows one to decode multivariate polynomial codes beyond half the minimum distance too, provided S is special in the above sense.

Another approach which works in the case of $S = \mathbb{F}_q$ is based on local decoding. Here we use the fact that $S^m = \mathbb{F}_q^m$ contains many lines (not just the axis-parallel ones), and then use the univariate decoding algorithms to decode on those lines from $(1 - \frac{d}{q})/2$ fraction errors. This approach manages to decode multivariate polynomial codes with $S = \mathbb{F}_q$ from $(\frac{1}{2} - o(1))$ of the minimum distance. Again, this approach does not work for general S, since a general S^m usually contains only axis-parallel lines (while $\mathbb{F}_q^m$ has many more lines).

Low degree d:

When the degree d of the multivariate polynomial code is significantly smaller than $|S|$, then a number of other list-decoding based methods come into play.

The powerful Reed-Muller list-decoding algorithm of Sudan [25] and its multiplicity-based generalization, based on $(m+1)$-variate interpolation and root-finding, can decode from $1 - (\frac{d}{|S|})^{\frac{1}{m+1}}$ fraction errors. With small degree $d = o(|S|)$ and $m = O(1)$, this decoding radius equals $1 - o(1)$! However when d is much larger (say $0.9 \cdot |S|$), then the fraction of errors decodable by this algorithm is around $\frac{1}{m+1} \cdot (1 - \frac{d}{|S|}) = \frac{1}{m+1} \cdot \delta_C$.

Another approach comes from the list-decoding of tensor codes [26]. While the multivariate polynomial codes we are interested in are not tensor codes, they are subcodes of the code of polynomials with *individual degree* at most d. Using the algorithm of [26] for decoding tensor codes, we get an algorithm that can decode from a $1 - o(1)$ fraction of errors when $d = o(|S|)$, but fails to approach a constant fraction of the minimum distance when d approaches $|S|$.

In light of all the above, to the best of our knowledge, for multivariate polynomial codes with $d > 0.9 \cdot |S|$ (i.e., $\delta_C < 0.1$), and S generic, the largest fraction of errors which could be corrected efficiently was about $\frac{1}{m+1}\delta_C$. In particular, the correctable fraction of errors is a vanishing fraction of the minimum distance, as the number of variables m grows.

We thus believe it is worthwhile to investigate this problem, not only because of its basic nature, but also because of the many different powerful algebraic ideas that only give partial results towards it.

4.1.3 Overview of the decoding algorithm

We now give a brief overview of our decoding algorithms. Let us first discuss the bivariate ($m = 2$) case. Here we are given a received word $r : S^2 \to \mathbb{F}$ such that there exists a codeword $P(X, Y) \in \mathbb{F}[X, Y]$ of degree at most $d = (1 - \delta_C)|S|$ with $\Delta(P, r) < \frac{\delta_C}{2}$. Our goal is to find $P(X, Y)$.

First some high-level strategy. An important role in our algorithm is played by the following observation: the restriction of a degree $\leq d$ bivariate polynomial $P(X, Y)$ to a vertical line (fixing $X = \alpha$) or a horizontal line (fixing $Y = \beta$) gives a degree $\leq d$ univariate polynomial. Perhaps an even more important role is played by the following disclaimer: *the previous observation does not characterize bivariate polynomials of degree d!* The set of functions $f : S^2 \to \mathbb{F}$ for which the horizontal restrictions and vertical restrictions are polynomials of degree $\leq d$ is the code of polynomials with *individual degree* at most d (this is the tensor Reed-Solomon code, with much smaller distance than the Reed-Muller code). For such a function f to be in the Reed-Muller code, the different univariate polynomials that appear as horizontal and vertical restrictions must be related in some way. The crux of our algorithm is to exploit these relations.

It will also help to recap the standard algorithm to decode tensor Reed-Solomon codes upto half their minimum distance (this scheme actually works for general tensor codes). Suppose we are given a received word $r : S^2 \to \mathbb{F}$, and we want to find a polynomial $P(X, Y)$ with individual degrees at most d which is close to r. One then takes the rows of this new received word (after having corrected the columns) and decodes them to the nearest degree $\leq d$ polynomial. The key point is to pass some "soft information" from the column decodings to the row decodings; the columns which were decoded from more errors are treated with lower confidence. This decodes the tensor Reed-Solomon code from 1/2 the minimum distance fraction errors. Several ingredients from this algorithm will appear in our Reed-Muller decoding algorithm.

Now we return to the problem of decoding Reed-Muller codes. Let us write $P(X, Y)$ as a single variable polynomial in Y with coefficients in $\mathbb{F}[X]$: $P(X, Y) = \displaystyle\sum_{i=0}^{d} P_i(X)Y^{d-i}$, where $\deg(P_i) \leq i$. For each $\alpha \in S$, consider the restricted univariate polynomial $P(\alpha, Y)$. Since $\deg(P_0) = 0$, $P_0(\alpha)$ must be the same for each α. Thus all the polynomials $\langle P(\alpha, Y)\rangle_{\alpha \in S}$ have the same coefficient for Y^d. Similarly, the coefficients of Y^{d-i} in the polynomials $\langle P(\alpha, Y)\rangle_{\alpha \in S}$ fit a degree i polynomial.

As in the tensor Reed-Solomon case, our algorithm begins by decoding each column $r(\alpha, \cdot)$ to the nearest degree $\leq d$ univariate polynomial. Now, instead of trying to use these decoded column polynomials to recover $P(X, Y)$ in one shot, we aim lower and just try to recover $P_0(X)$. The advantage is that $P_0(X)$ is only a degree 0 polynomial, and is thus resilient to many more errors than a degree d polynomial. Armed with $P_0(X)$, we then proceed to find $P_1(X)$. The knowledge of $P_0(X)$ allows us to decode the columns $r(\alpha, \cdot)$ to a slightly larger radius; in turn this improved radius allows us to recover the degree 1 polynomial $P_1(X)$. At the i^{th} stage, we have already recovered $P_0(X)$, $P_1(X)$, $\dots$, $P_{i-1}(X)$. Consider, for each $\alpha \in S$, the function $f_\alpha(Y) = r(\alpha, Y) - \sum_{j=0}^{i-1} P_j(\alpha)Y^{d-j}$. Our algorithm decodes $f_\alpha(Y)$ to the nearest degree $d - i$ polynomial: note that as i increases, we are decoding to a lower degree polynomial, and hence we are able to handle a larger fraction of errors. Define $h(\alpha)$ to be the coefficient of Y^{d-i} in the polynomial so obtained; this "should" equal the evaluation of the degree i polynomial $P_i(\alpha)$. So we next decode $h(\alpha)$ to the nearest degree i polynomial (using the appropriate soft information), and it turns out that this decoded polynomial must equal $P_i(X)$. By the time i reaches d, we would have recovered $P_0(X), P_1(X), \ldots, P_d(X)$, and hence all of $P(X, Y)$. Summarizing, the algorithm repeatedly decodes the columns $r(\alpha, \cdot)$, and at each stage it uses the relationship between the different univariate polynomial $P(\alpha, Y)$ to: (1) learn a little bit more about the polynomial $P(X, Y)$, and (2) increase the radius to which we can decode $r(\alpha, \cdot)$ in the next stage. This completes the description of the algorithm in the $m = 2$ case.

The case of general m is very similar, with only a small augmentation needed. Decoding m-variate polynomials turns out to reduce to decoding $m-1$-variate polynomials with soft information; thus in order to make a sustainable recursive algorithm, we aim a little higher and instead solve the more general problem of decoding multivariate polynomial codes with uncertainties (where each coordinate of the received word has an associated "confidence" level).

To implement the above algorithms in near-linear time, we use some tools from list-decoding. The main bottleneck in the running time is the requirement of having to decode the same column $r(\alpha, \cdot)$ multiple times to larger and larger radii (to lower and lower degree polynomials). To save on these decodings, we can instead list-decode $r(\alpha, \cdot)$ to a large radius using a near-linear time list-decoder for Reed-Solomon codes; this reduces the number of required decodings of the same column from d to $O(1)$ (provided $d < (1 - \Omega(1))|S|$). For the $m = 2$ case this works fine, but for $m > 2$ case this faces a serious obstacle; in general it is impossible to efficiently list-decode Reed-Solomon codes *with uncertainties* beyond half the minimum distance of the code (the list size can be superpolynomial). We get around this using some technical ideas, based on speeding-up the decoding of Reed-Muller codes with uncertainties when the fraction of errors is significantly smaller than half the minimum distance. For details, see Section 4.6.

4.1.4 Organization of this paper

In Section 2, we cover the notion of weighted distance, which will be used in handling Reed-Solomon and Reed-Muller decoding with soft information on the reliability of the symbols in the encoding. In Section 3, we state and prove a polynomial time algorithm for decoding bivariate Reed-Muller codes to half the minimum distance. We then generalize the proof to decode multivariate Reed-Muller codes in Section 4. Finally, in sections 5 and 6, we show that decoding Reed-Muller codes to almost half the minimum distance can be done in near-linear time by improving on the algorithms in Section 3 and 4.

4.2 Preliminaries

At various stages of the decoding algorithm, we will need to deal with symbols and received words in which we have varying amounts of confidence. We now introduce some language to deal with such notions.

Let Σ denote an alphabet. A *weighted symbol* of Σ is simply an element of $\Sigma \times [0,1]$. In the weighted symbol (σ, u), we will be thinking of $u \in [0,1]$ as our uncertainty that σ is the symbol we should be talking about.

For a weighted symbol (σ, u) and a symbol σ', we define their distance $\Delta((\sigma, u), \sigma')$ by:

$$\Delta((\sigma, u), \sigma') = \begin{cases} 1 - u/2 & \sigma \neq \sigma' \\ u/2 & \sigma = \sigma' \end{cases}$$

For a weighted function $r : T \to \Sigma \times [0,1]$, and a (conventional) function $f : T \to \Sigma$, we define their Hamming distance by

$$\Delta(r, f) = \sum_{t \in T} \Delta(r(t), f(t)).$$

The key inequality here is the triangle inequality.

Lemma 4.2.1 (Triangle inequality for weighted functions)**.** *Let $f, g : T \to \Sigma$ be functions, and let $r : T \to \Sigma \times [0,1]$ be a weighted function. Then:*

$$\Delta(r, f) + \Delta(r, g) \geq \Delta(f, g).$$

Proof. We will show that if $t \in T$ is such that $f(t) \neq g(t)$, then $\Delta(r(t), f(t)) + \Delta(r(t), g(t)) \geq 1$. This will clearly suffice to prove the lemma.

Let $r(t) = (\sigma, u)$. Suppose $f(t) = \sigma_1$ and $g(t) = \sigma_2$. Then either $\sigma \neq \sigma_1$ or $\sigma \neq \sigma_2$, or both. Thus either we have $\Delta(r(t), f(t)) + \Delta(r(t), g(t)) = (1 - u/2) + u/2$ or we have $\Delta(r(t), f(t)) + \Delta(r(t), g(t)) = u/2 + (1 - u/2)$, or we have $\Delta(r(t), f(t)) + \Delta(r(t), g(t)) = (1-u/2)+(1-u/2)$. In all cases, we have $\Delta(r(t), f(t))+\Delta(r(t), g(t)) \geq 1$, as desired. $\square$

The crucial property that this implies is the unique decodability up to half the minimum distance of a code for *weighted* received words.

Lemma 4.2.2. *Let $\mathcal{C} \subseteq \Sigma^T$ be a code with minimum distance Δ. Let $r : T \to \Sigma \times [0, 1]$ be a weighted function. Then there is at most one $f \in \mathcal{C}$ satisfying*

$$\Delta(r, f) < \Delta/2.$$

Furthermore, for this particular definition of weighted distance, there is a very natural decoding algorithm, due to Forney, to find the unique $f \in \mathcal{C}$ in Lemma 4.2.2 [29]. For each weighted symbol (x, u), we erase x with probability u. We then apply a standard decoding algorithm that handles both errors and erasures. This successfully finds the unique codeword f as long as $2E + F < \Delta$, where E denotes the number of errors and F denotes the number of erasures. With this definition of weighted distance, the condition that $\Delta(r, f) < \Delta/2$ is equivalent to the expected value of $2E + F$ being at most Δ.

4.3 Bivariate Reed-Muller Decoding

In this section, we provide an algorithm for decoding bivariate Reed-Muller codes to half the minimum distance. Consider the bivariate Reed-Muller decoding problem. We are given a received word $r : S^2 \to \mathbb{F}$. Suppose that there is a codeword $C \in \mathbb{F}[X, Y]$ with $\deg(C) \leq d$, whose distance $\Delta(r, C)$ from the received word is at most half the minimum distance $|S|(|S| - d)/2$. The following result says that there is a polynomial time algorithm in the size of the input $|S|^2$ to find C:

Theorem 4.3.1. *Let $\mathbb{F}$ be a finite field and let $S \subseteq \mathbb{F}$ be a nonempty subset of size $|S| = n$. Given a received word $r : S^2 \to \mathbb{F}$, there is a $O(n^3 \operatorname{polylog}(n, |\mathbb{F}|))$ time* *algorithm to find the unique polynomial (if it exists)* $C \in \mathbb{F}[X,Y]$ *with* $\deg(C) \leq d$ *such that*

$$\Delta(r, C) < \frac{n^2}{2}\left(1 - \frac{d}{n}\right).$$

4.3.1 Outline of Algorithm

The general idea of the algorithm is to write $C(X,Y) = \sum_{i=0}^{d} P_i(X)Y^{d-i} \in \mathbb{F}[X][Y]$ as a polynomial in Y with coefficients as polynomials in $\mathbb{F}[X]$, and attempt to uncover the coefficients $P_i(X)$ one at a time.

We outline the first iteration of the algorithm, which uncovers the coefficient $P_0(X)$ of degree 0. View the encoded message as a matrix on $S \times S$, where the rows are indexed by $x \in S$ and the columns by $y \in S$. We first Reed-Solomon decode the rows $r(x,Y), x \in S$ to half the minimum distance $(n-d)/2$ and extract the coefficient of Y^d in those decodings. This gives us guesses for what $P_0(x)$ is for $x \in S$. However, this isn't quite enough to determine $P_0(X)$. So we will also include some soft information which tells us how uncertain we are that the coefficient is correct. The uncertainty is a number in $[0,1]$ that is based on how far the decoded codeword $G_x(Y)$ is from the received word $r(x,Y)$. The farther apart, the higher the uncertainty. A natural choice for the uncertainty is simply the ratio of the distance $\Delta(G_x(Y), r(x,Y))$ to half the minimum distance $(n-d)/2$. In the event that the Reed-Solomon decoding finds no codeword, we make an arbitrary guess and set the uncertainty to be 1. Let $f : S \to F \times [0,1]$ be the function of guesses for $P_0(x)$ and their uncertainties. We then use a Reed-Solomon decoder with uncertainties to find the degree 0 polynomial that is closest to $f(X)$. This will give us $P_0(X)$. Finally, subtract $P_0(X)Y^d$ from $r(X,Y)$ and repeat to get the subsequent coefficients.

In the algorithm, we use REED-SOLOMON-DECODER(r,d) to denote the $O(n \operatorname{polylog} n)$ time algorithm that performs Reed-Solomon decoding of degree d to half the minimum distance [27, 28]. We use RS-SOFT-DECODER(r,d) to denote the $O(n^2 \operatorname{polylog} n)$ time algorithm that performs Reed-Solomon decoding of degree d with uncertainties to half the minimum distance, which is based on Forney's generalized minimum distance decoding algorithm for concatenated codes [29].

Algorithm 1 Decoding Bivariate Reed Muller

```
1: Input: $r : S^2 \to \mathbb{F}$.
2: for $i = 0, 1, \ldots, d$ do
3:     Define $r_i : S \times S \to \mathbb{F}$ by
           $$r_i(X, Y) = r(X, Y) - \sum_{j=0}^{i-1} Q_j(X) Y^{d-j}.$$
4:     for $x \in S$ do
5:         Define $r_{i,x} : S \to \mathbb{F}$ by
               $$r_{i,x}(Y) = r_i(x, Y).$$
6:         Define $G_x(Y) \in \mathbb{F}[Y]$ by
               $$G_x(Y) = \text{REED-SOLOMON-DECODER}(r_{i,x}(Y), d - i).$$
7:         $\sigma_x \leftarrow \text{Coeff}_{Y^{d-i}}(G_x)$.
8:         $\delta_x \leftarrow \Delta(r_{i,x}, G_x)$.
9:     end for
10:    Define the weighted function $f_i : S \to \mathbb{F} \times [0, 1]$ by
           $$f_i(x) = \left(\sigma_x, \frac{\delta_x}{(n - d + i)/2}\right).$$
11:    Define $Q_i : S \to \mathbb{F}$ by
           $$Q_i(X) = \text{RS-SOFT-DECODER}(f_i(X), i).$$
12: end for
13: Output: $\sum_{i=0}^{d} Q_i(X) Y^{d-i}$.
```

4.3.2 Proof of Theorem 4.3.1

Proof. **Correctness of Algorithm** It suffices to show that $Q_i(X) = P_i(X)$ for $i = 0, 1, \ldots, d$, which we prove by induction. For this proof, the base case and inductive step can be handled by a single proof. We assume the inductive hypothesis that we have $Q_j(X) = P_j(X)$ for $j < i$. Note that the base case is $i = 0$ and in this case, we assume nothing.

It is enough to show $\Delta(f_i(X), P_i(X)) < \frac{n}{2}\left(1 - \frac{i}{n}\right)$. Then $P_i(x)$ is the unique polynomial within weighted distance $\frac{n}{2}\left(1 - \frac{i}{n}\right)$ of $f_i(X)$. So RS-SOFT-DECODER$(f_i(X), i)$ will output $Q_i(X) = P_i(X)$.

We first show that $r_i(X,Y)$ is close to $C_i(X,Y) = \sum_{j=i}^{d} P_j(X)Y^{d-j}$. Observe that:

$$\begin{aligned}
& r_i(X,Y) - C_i(X,Y) \\
= \; & (r_i(X,Y) + \sum_{j=1}^{i-1} P_j(X)Y^{d-j}) - (C_i(X,Y) + \sum_{j=1}^{i-1} P_j(X)Y^{d-j})) \\
= \; & (r_i(X,Y) + \sum_{j=1}^{i-1} Q_j(X)Y^{d-j}) - C(X,Y) \\
= \; & r(X,Y) - C(X,Y).
\end{aligned}$$

Hence,

$$\Delta(r_i(X,Y), C_i(X,Y)) = \Delta(r(X,Y), C(X,Y)) < \frac{n^2}{2}\left(1 - \frac{d}{n}\right).$$

For each $x \in S$, define $C_{i,x}(Y) = C_i(x,Y)$. Define $\Delta_x = \Delta(r_{i,x}(Y), C_{i,x}(Y))$. Let $A = \{x \in S | G_x(Y) = C_{i,x}(Y)\}$ be the set of choices of x such that $G_x(Y) =$ REED-SOLOMON-DECODER$(r_{i,x}(Y), d - i)$ produces $C_{i,x}(Y)$.

Then, for $x \in A$, we have

$$\delta_x = \Delta(r_{i,x}(Y), G_x(Y)) = \Delta(r_{i,x}(Y), C_{i,x}(Y)) = \Delta_x,$$

which gives us an uncertainty value of

$$u_{i,x} = \frac{\Delta_x}{(n - d + i)/2}.$$

For $x \notin A$, either we have $G_x \neq C_{i,x}$, or the Reed-Solomon decoder does not find a polynomial. In the first case, Lemma 4.2.1 tells us:

$$\delta_x = \Delta(r_{i,x}(Y), G_x(Y)) \geq n - d + i - \Delta(r_{i,x}(Y), C_{i,x}(Y)) = n - d + i - \Delta_x,$$

which gives us an uncertainty value of

$$u_{i,x} = \frac{n-d+i-\Delta_x}{(n-d+i)/2}.$$

Finally, in the case where the Reed-Solomon decoder does not find a polynomial, we get an uncertainty value of

$$u_{i,x} = 1.$$

This means that the contribution of the corresponding guess to the weighted distance $\Delta(f_i(X), P_i(X))$ is 1/2, However, we know that since no polynomial was found, $\Delta_x \geq \frac{n-d+i}{2}$, so the contribution to the weighted distance had the Reed-Solomon decoder found an incorrect polynomial not matching the true codeword is $1 - \frac{1}{2}\frac{n-d+i-\Delta_x}{(n-d+i)/2} \geq 1/2$. So for the purposes of upper bounding the weighted distance $\Delta(f_i(X), P_i(X))$, we treat this case the same as decoding to the wrong polynomial.

We now upper bound $\Delta(f_i(X), P_i(X))$:

$$\begin{aligned}
\Delta(f_i(X), P_i(X)) &\leq \sum_{x\in A} \frac{1}{2}\frac{\Delta_x}{(n-d+i)/2} + \sum_{x\notin A} 1 - \frac{1}{2}\frac{n-d+i-\Delta_x}{(n-d+i)/2} \\
&\leq \sum_{x\in A} \frac{\Delta_x}{n-d+i} + \sum_{x\notin A} 1 - \frac{n-d+i-\Delta_x}{n-d+i} \\
&= \sum_{x\in A} \frac{\Delta_x}{n-d+i} + \sum_{x\notin A} \frac{\Delta_x}{n-d+i} \\
&= \sum_{x\in S^m} \frac{\Delta_x}{n-d+i} \\
&= \frac{\Delta(r_i(X,Y), C_i(X,Y))}{n-d+i} \\
&< \frac{n^2}{2}\left(1-\frac{d}{n}\right)\frac{1}{n-d+i} \\
&= \frac{n}{2}\cdot\frac{n-d}{n-d+i} \\
&\leq \frac{n}{2}\cdot\frac{n-i}{n} \\
&= \frac{n}{2}\left(1-\frac{i}{n}\right).
\end{aligned}$$

Runtime of Algorithm

We claim that the runtime of our algorithm is $O(n^3 \operatorname{polylog} n)$, ignoring the polylog $|\mathbb{F}|$ factor from field operations. The algorithm has $d+1$ iterations. In each iteration, we update r_i, apply REED-SOLOMON-DECODER to n rows and apply RS-SOFT-DECODER a single time to get the leading coefficient. As updating takes $O(n^2)$ time, REED-SOLOMON-DECODER takes $O(n \operatorname{polylog} n)$ time, and RS-SOFT-DECODER takes $O(n^2 \operatorname{polylog} n)$ time, we get $O(n^2 \operatorname{polylog} n)$ for each iteration. $d+1$ iterations gives a total runtime of $O(dn^2 \operatorname{polylog} n) < O(n^3 \operatorname{polylog} n)$.

□

4.4 Reed-Muller Decoding for General m

We now generalize the algorithm for decoding bivariate Reed-Muller codes to handle Reed-Muller codes of any number of variables. As before, we write the codeword as a polynomial in one of the variables and attempt to uncover its coefficients one at a time. Interestingly, this leads us to a Reed-Muller decoding on one fewer variable, but with uncertainties. This lends itself nicely to an inductive approach on the number of variables, however, the generalization requires us to be able to decode Reed-Muller codes with uncertainties. This leads us to our main theorem:

Theorem 4.4.1. *Let $\mathbb{F}$ be a finite field and let $S \subseteq \mathbb{F}$ be a nonempty subset of size $|S| = n$. Given a received word with uncertainties $r : S^m \to \mathbb{F} \times [0,1]$, there is a $O(n^{m+2} \operatorname{polylog}(n, |\mathbb{F}|))$ time algorithm to find the unique polynomial (if it exists) $C \in \mathbb{F}[X_1, \ldots, X_m]$ with $\deg(C) \leq d$ such that*

$$\Delta(r, C) < \frac{n^m}{2}\left(1 - \frac{d}{n}\right).$$

Note that to decode a Reed-Muller code without uncertainties, we may just set all the initial uncertainties to 0. The algorithm slows by a factor of n from the bivariate case due to having to use the RS-SOFT-DECODER instead of the faster REED-SOLOMON-DECODER on the rows of the received word.

Proof. The proof is by induction on the number of variables, and closely mirrors the proof of the bivariate case.

Base Case

We are given a received word with uncertainties $r : S \to \mathbb{F} \times [0,1]$ and asked to find the unique polynomial $C \in \mathbb{F}[X]$ with $\deg(C) \leq d$ within weighted distance $\frac{n-d}{2}$ of r. This is just Reed-Solomon decoding with uncertainty, which can be done in time $O(n^2 \operatorname{polylog} n)$.

Inductive Step

Assume that the result holds for m variables. That is, assume we have access to an algorithm REED-MULLER-DECODER(r, m, d) which takes as input a received word with uncertainties $r : S^m \to \mathbb{F} \times [0,1]$, and outputs the unique polynomial of degree at most d (if it exists) within weighted distance $\frac{n^m}{2}\left(1 - \frac{d}{n}\right)$ from r. We want to produce an algorithm for $m+1$ variables. Before we progress, we set up some definitions to make the presentation and analysis of the algorithm cleaner. We are given $r : S^{m+1} \to \mathbb{F} \times [0,1]$. View r as a map from $S^m \times S \to \mathbb{F} \times [0,1]$, and write $r(\boldsymbol{X}, Y) = (\overline{r}(\boldsymbol{X}, Y), u(\boldsymbol{X}, Y))$.

Suppose that there exists a polynomial $C \in \mathbb{F}[\boldsymbol{X}, Y]$ with $\deg(C) \leq d$ such that

$$\Delta(r, C) < \frac{n^{m+1}}{2}\left(1 - \frac{d}{n}\right).$$

View C as a polynomial in Y with coefficients in $\mathbb{F}[\boldsymbol{X}]$, $C(\boldsymbol{X}, Y) = \sum_{i=0}^{d} P_i(\boldsymbol{X}) Y^{d-i}$. The general strategy of the algorithm is to determine the P_i's inductively by performing $d+1$ iterations from $i = 0$ to $i = d$, and recovering $P_i(\boldsymbol{X})$ at the i-th iteration.

For the i-th iteration, consider the word

$$r_i(\boldsymbol{X}, Y) = \left(\overline{r}(\boldsymbol{X}, Y) - \sum_{j=0}^{i-1} P_j(\boldsymbol{X}) Y^{d-j}, u(\boldsymbol{X}, Y)\right).$$

Since r is close to $\sum_{j=0}^{d} P_j(\boldsymbol{X}) Y^{d-j}$, r_i will be close to $C_i = \sum_{j=i}^{d} P_j(\boldsymbol{X}) Y^{d-j}$. Our goal is to find $P_i(\boldsymbol{X})$, the leading coefficient of C_i when viewed as a polynomial in Y. For each $\boldsymbol{x} \in S^m$, we decode the Reed-Solomon code with uncertainties given by $r_i(\boldsymbol{x}, Y)$ and extract the coefficient of Y^{d-i} along with how uncertain we are about the correctness of this coefficient. This gives us a guess for the value $P_i(\boldsymbol{x})$ and our uncertainty for this guess. We construct the function $f_i : S^m \to F \times [0,1]$ of guesses for P_i with their uncertainties. We then apply the induction hypothesis of Theorem 4.4.1 to f_i to recover P_i.

Algorithm 2 Decoding Reed Muller with Uncertainties

```
1: Input: $r : S^{m+1} \to \mathbb{F} \times [0,1]$.
2: for $i = 0, 1, \ldots, d$ do
3:     Define $r_i : S^m \times S \to \mathbb{F} \times [0,1]$ by
           $$r_i(\boldsymbol{X}, Y) = \left(\overline{r}(\boldsymbol{X}, Y) - \sum_{j=0}^{i-1} Q_j(\boldsymbol{X}) Y^{d-j}, u(\boldsymbol{X}, Y)\right).$$
4:     for $\boldsymbol{x} \in S^m$ do
5:         Define $r_{i,\boldsymbol{x}} : S \to \mathbb{F} \times [0,1]$ by
               $$r_{i,\boldsymbol{x}}(Y) = r_i(\boldsymbol{x}, Y).$$
6:         Define $G_{\boldsymbol{x}}(Y) \in \mathbb{F}[Y]$ by
               $$G_{\boldsymbol{x}}(Y) = \text{RS-SOFT-DECODER}(r_{i,\boldsymbol{x}}(Y), d - i).$$
7:         $\sigma_{\boldsymbol{x}} \leftarrow \text{Coeff}_{Y^{d-i}}(G_{\boldsymbol{x}})$.
8:         $\delta_{\boldsymbol{x}} \leftarrow \Delta(r_{i,\boldsymbol{x}}, G_{\boldsymbol{x}})$.
9:     end for
10:    Define the weighted function $f_i : S^m \to \mathbb{F} \times [0,1]$ by
           $$f_i(\boldsymbol{x}) = \left(\sigma_{\boldsymbol{x}}, \frac{\delta_{\boldsymbol{x}}}{(n - d + i)/2}\right).$$
11:    Define $Q_i : S^m \to \mathbb{F}$ by
           $$Q_i(\boldsymbol{X}) = \text{REED-MULLER-DECODER}(f_i(\boldsymbol{X}), m, i).$$
12: end for
13: Output: $\sum_{i=0}^{d} Q_i(\boldsymbol{X}) Y^{d-i}$.
```

Correctness of Algorithm

Suppose there is a polynomial $C(\boldsymbol{X}, Y) = \sum_{i=0}^{d} P_i(\boldsymbol{X}) Y^{d-i}$ such that

$$\Delta(r(\boldsymbol{X}, Y), C(\boldsymbol{X}, Y)) < \frac{n^{m+1}}{2}\left(1 - \frac{d}{n}\right).$$

We will show by induction that the i-th iteration of the algorithm produces $Q_i(\boldsymbol{X}) = P_i(\boldsymbol{X})$. For this proof, the base case and inductive step can be handled by a single proof. We assume the inductive hypothesis that we have $Q_j(\boldsymbol{X}) = P_j(\boldsymbol{X})$ for $j < i$. Note that the base case is $i = 0$ and in this case, we assume nothing.

It is enough to show $\Delta(f_i(\boldsymbol{X}), P_i(\boldsymbol{X})) < \frac{n^m}{2}\left(1 - \frac{i}{n}\right)$. Then $P_i(\boldsymbol{X})$ is the unique polynomial within weighted distance $\frac{n^m}{2}\left(1 - \frac{i}{n}\right)$ of $f_i(\boldsymbol{X})$. This means that REED-MULLER-DECODER$(f_i(\boldsymbol{X}), m, i)$ will output $Q_i(\boldsymbol{X}) = P_i(\boldsymbol{X})$.

We first show that $r_i(\boldsymbol{X}, Y)$ is close to $C_i(\boldsymbol{X}, Y) = \sum_{j=i}^{d} P_j(\boldsymbol{X}) Y^{d-j}$. Observe that:

$$\begin{aligned}
& r_i(\boldsymbol{X}, Y) - C_i(\boldsymbol{X}, Y) \\
= \; & (r_i(\boldsymbol{X}, Y) + \sum_{j=1}^{i-1} P_j(\boldsymbol{X}) Y^{d-j}) - (C_i(\boldsymbol{X}, Y) + \sum_{j=1}^{i-1} P_j(\boldsymbol{X}) Y^{d-j})) \\
= \; & (r_i(\boldsymbol{X}, Y) + \sum_{j=1}^{i-1} Q_j(\boldsymbol{X}) Y^{d-j}) - C(\boldsymbol{X}, Y) \\
= \; & r(\boldsymbol{X}, Y) - C(\boldsymbol{X}, Y).
\end{aligned}$$

Hence,

$$\Delta(r_i(\boldsymbol{X}, Y), C_i(\boldsymbol{X}, Y)) = \Delta(r(\boldsymbol{X}, Y), C(\boldsymbol{X}, Y)) < \frac{n^{m+1}}{2}\left(1 - \frac{d}{n}\right).$$

For each $\boldsymbol{x} \in S^m$, define $C_{i,\boldsymbol{x}}(Y) = C_i(\boldsymbol{x}, Y)$. Define $\Delta_{\boldsymbol{x}} = \Delta(r_{i,\boldsymbol{x}}(Y), C_{i,\boldsymbol{x}}(Y))$. Let $A = \{\boldsymbol{x} \in S^m | G_{\boldsymbol{x}}(Y) = C_{i,\boldsymbol{x}}(Y)\}$ be the set of choices of $\boldsymbol{x}$ such that $G_{\boldsymbol{x}}(Y) =$ RS-SOFT-DECODER$(r_{i,\boldsymbol{x}}(Y), d - i)$ produces $C_{i,\boldsymbol{x}}(Y)$.

Then, for $\boldsymbol{x} \in A$, we have

$$\delta_{\boldsymbol{x}} = \Delta(r_{i,\boldsymbol{x}}(Y), G_{\boldsymbol{x}}(Y)) = \Delta(r_{i,\boldsymbol{x}}(Y), C_{i,\boldsymbol{x}}(Y)) = \Delta_{\boldsymbol{x}}.$$

And for $\boldsymbol{x} \notin A$, we have $G_{\boldsymbol{x}} \neq C_{i,\boldsymbol{x}}$, so

$$\delta_{\boldsymbol{x}} = \Delta(r_{i,\boldsymbol{x}}(Y), G_{\boldsymbol{x}}(Y)) \geq n - d + i - \Delta(r_{i,\boldsymbol{x}}(Y), C_{i,\boldsymbol{x}}(Y)) = n - d + i - \Delta_{\boldsymbol{x}}.$$

We now upper bound $\Delta(f_i(\boldsymbol{X}), P_i(\boldsymbol{X}))$:

$$\Delta(f_i(\boldsymbol{X}), P_i(\boldsymbol{X})) \;\leq\; \sum_{\boldsymbol{x} \in A} \frac{1}{2} \frac{\delta_{\boldsymbol{x}}}{(n - d + i)/2} + \sum_{\boldsymbol{x} \notin A} 1 - \frac{1}{2} \frac{\delta_{\boldsymbol{x}}}{(n - d + i)/2}$$

$$
\begin{aligned}
&\le \sum_{\boldsymbol{x}\in A} \frac{\Delta_{\boldsymbol{x}}}{n-d+i} + \sum_{\boldsymbol{x}\notin A} 1 - \frac{n-d+i-\Delta_{\boldsymbol{x}}}{n-d+i} \\
&= \sum_{\boldsymbol{x}\in A} \frac{\Delta_{\boldsymbol{x}}}{n-d+i} + \sum_{\boldsymbol{x}\notin A} \frac{\Delta_{\boldsymbol{x}}}{n-d+i} \\
&= \sum_{\boldsymbol{x}\in S^m} \frac{\Delta_{\boldsymbol{x}}}{n-d+i} \\
&= \frac{\Delta(r_i(\boldsymbol{X},Y), C_i(\boldsymbol{X},Y))}{n-d+i} \\
&< \frac{n^{m+1}}{2}\left(1-\frac{d}{n}\right)\frac{1}{n-d+i} \\
&= \frac{n^m}{2}\cdot\frac{n-d}{n-d+i} \\
&\le \frac{n^m}{2}\cdot\frac{n-i}{n} \\
&= \frac{n^m}{2}\left(1-\frac{i}{n}\right).
\end{aligned}
$$

Runtime of Algorithm

We claim the runtime of our m-variate Reed-Muller decoder is $O(n^{m+2}\operatorname{polylog} n)$, ignoring the polylog $|\mathbb{F}|$ factor from field operations. We again proceed by induction on m. In the base case of $m = 1$, we simply run the Reed-Solomon decoder with uncertainties, which runs in $O(n^2 \operatorname{polylog} n)$ time. Now suppose the m-variate Reed-Muller decoder runs in time $O(n^{m+2}\operatorname{polylog} n)$. We need to show that the $m+1$-variate Reed-Muller decoder runs in time $O(n^{m+3}\operatorname{polylog} n)$.

The algorithm makes $d + 1$ iterations. In each iteration, we perform n^m Reed-Solomon decodings with uncertainties, and extract the leading coefficient along with its uncertainty for each one. Each Reed-Solomon decoding takes $O(n^2 \operatorname{polylog} n)$ time, while computing an uncertainty of a leading coefficient takes $O(n \operatorname{polylog} n)$. So in this step, we have cumulative runtime $O(n^{m+2}\operatorname{polylog} n)$. Next we do a single m-variate Reed-Muller decoding with uncertainties, which takes $O(n^{m+2}\operatorname{polylog} n)$ by our induction hypothesis. This makes the total runtime $O(dn^{m+2}\operatorname{polylog} n) \le O(n^{m+3}\operatorname{polylog} n)$, as desired.

□

4.5 Near-Linear Time Decoding in the Bivariate Case

In this section, we present our near-linear time decoding algorithm for bivariate Reed-Muller codes.

Theorem 4.5.1. *Let $\alpha \in (0,1)$ be a constant. Let $\mathbb{F}$ be a finite field and let $S \subseteq \mathbb{F}$ be a nonempty subset of size $|S| = n$. Let $d = \alpha n$. Given a received word $r : S^2 \to \mathbb{F}$, there is a $O(n^2 \operatorname{polylog}(n, |\mathbb{F}|))$ time algorithm to find the unique polynomial (if it exists) $C \in \mathbb{F}[X,Y]$ with $\deg(C) \leq d$ such that*

$$\Delta(r, C) < \frac{n^2}{2}\left(1 - \frac{d}{n} - \frac{1}{\sqrt{n}}\right).$$

4.5.1 Outline of Improved Algorithm

Recall that the decoding algorithms we presented in the previous sections make $d+1$ iterations, where $d = \alpha n$, revealing a single coefficient of the nearest codeword during one iteration. In a given iteration, we decode each row of $r_i(X,Y)$ to the nearest polynomial of degree $d-i$, extracting the coefficient of Y^{d-i} and its uncertainty. Then we Reed-Solomon decode with uncertainties to get the leading coefficient of $C(X,Y)$, when viewed as a polynomial in Y.

The runtime of this algorithm is $O(n^3 \operatorname{polylog} n)$. Each iteration has n Reed-Solomon decodings and a single Reed-Solomon decoding with uncertainties. As Reed-Solomon decoding takes $O(n \operatorname{polylog} n)$ time and Reed-Solomon decoding with uncertainties takes $O(n^2 \operatorname{polylog} n)$ time, we get a runtime of $O(n^3 \operatorname{polylog} n)$ with $d+1$ iterations. To achieve near-linear time, we need to shave off a factor of n on both the number of Reed-Solomon decodings and the runtime of Reed-Solomon decoding with uncertainties.

To save on the number of Reed-Solomon decodings, we will instead list decode beyond half the minimum distance (using a near-linear time Reed-Solomon list-decoder), and show that the list we get is both small and essentially contains all of the decoded polynomials we require for $\Omega(n)$ iterations of i. So we will do $O(n)$ Reed-Solomon list-decodings total instead of $O(n^2)$ Reed-Solomon unique decodings to half the minimum distance.

To save on the runtime of Reed-Solomon decoding with uncertainties, we will use a probabilistic variant of Forney's generalized minimum distance decoding algorithm, which runs in near-linear time, but reduces the decoding radius from $1/2$ the minimum distance to $1/2 - o(1)$ of the minimum distance.

4.5.2 Proof of Fast Bivariate Reed-Muller Decoding

Proof of Theorem 4.5.1. As in the proof of Theorem 4.3.1, we write $C = \sum_{j=0}^{d} P_j(X)Y^{d-j}$, and inductively find the $P_i(X)$. Suppose that we have successfully found the first i of the $P_j(X)$ and are now trying to find $P_i(X)$. Also as before, we fix $x \in S$ and guess the value of $P_i(x)$ by Reed-Solomon decoding $r_{i,x} = r(x,Y) - \sum_{j=0}^{i-1} P_j(x)Y^{d-j}$ to the nearest polynomial of degree at most i within distance $(n-d+i)/2$.

Reducing the Number of Decodings

To reduce the number of Reed-Solomon decodings, we will instead list decode past half the minimum distance, and use the small list of polynomials we get to guess $P_i(x)$ for the next $\Omega(n)$ values of j. In the above setting, we have that $r_{i,x} : S \to \mathbb{F}$ is a received word for a Reed-Solomon code $\mathcal{C}_i$ of degree at most $d_i = d - i$. Let t be the radius to which we list decode, and let $L_{i,x} = \{C \in \mathcal{C}_i | \Delta(C, r_{i,x}) < t\}$ be the list of codewords within distance t of $r_{i,x}$. The radius to which we can decode while maintaining a polynomial-size list is given by the Johnson bound:

$$n(1 - \sqrt{1 - \delta_i}),$$

where $\delta_i = 1 - \frac{d-i}{n} > 1 - \frac{d}{n} = 1 - \alpha$ is the relative distance of the code. By Taylor approximating the square root, we see that the Johnson bound exceeds half the minimum distance by $\Omega(n)$:

$$\begin{aligned} n(1 - \sqrt{1-\delta_i}) &> n(1 - (1 - \delta_i/2 + \delta_i^2/8 + 3\delta_i^3/16)) \\ &= n(\delta_i/2 + (1-\alpha)^2/8 + 3(1-\alpha)^3/16) \\ &= (n - d + i)/2 + ((1-\alpha)^2/8)n + cn, \end{aligned}$$

where $c = 3(1-\alpha)^3/16$ is a positive constant. By a standard list-size bound as in the one in Cassuto and Bruck [30], we see that if we set the list decoding radius $t = (n-d+i)/2 + ((1-\alpha)^2/8)n$, then the size of the list $|L_{i,x}| < \frac{1}{c}$ is constant. So the list decoding radius exceeds half the minimum distance by $\Omega(n)$, and the list size is constant. By Aleknovich's fast algorithm for weighted polynomial construction [31], the list $L_{i,x}$ can be produced in time $(1/\alpha)^{O(1)} n \log^2 n \log\log n = O(n \operatorname{polylog} n)$. We will let RS-LIST-DECODER(r, d, t) denote the Reed-Solomon list decoder that outputs a list of all ordered pairs of polynomials of degree at most d within distance t to the received word r along with their distances to r. Since the list size is constant, all of the distances can be computed in $O(n \operatorname{polylog} n)$ time.

For the next cn values of j, we search the $O(1)$-size list $L_{i,x}$ to find the nearest polynomial of degree at most $n-d+j$ within distance $(n-d+j)/2$ from $r_{j,x}$.

Faster Reed-Solomon Decoding with Uncertainties

Once we have all the guesses for $P_i(x), x \in S$ along with their uncertainties, we want to apply a near-linear time decoding algorithm to find $P_i(x)$. In Appendix A, we give a description of the probabilistic GMD algorithm that gives a faster Reed-Solomon decoder with uncertainties. We will refer to this algorithm as FAST-RS-DECODER(f, i), where $f : S \to \mathbb{F} \times [0,1]$ is a received word with uncertainties, and i is the degree of the code. FAST-RS-DECODER(f, i) will output the codeword within distance $(n - i - \sqrt{n})/2$ (if it exists) with probability at least $1 - \frac{1}{n^{\Omega(1)}}$ (the $\Omega(1)$ can be chosen to be an arbitrary constant, by simply repeating the algorithm independently several times).

Algorithm 3 Decoding Bivariate Reed Muller

```
1: Input: $r : S^2 \to \mathbb{F}$.
2: Let $c = ((1-\alpha)^2/8)$.
3: for $j = 0, 1, \ldots, \frac{d}{2cn}$ do
4:     Let $t_j = \frac{n-d+j\cdot 2cn}{2} + cn$.
5:     Define $r_{j\cdot 2cn} : S \times S \to \mathbb{F}$ by
           $$r_{j\cdot 2cn}(X,Y) = r(X,Y) - \sum_{i=0}^{j\cdot 2cn-1} Q_i(X) Y^{d-i}.$$
```

Algorithm 3 Decoding Bivariate Reed Muller (Continued)

```
 6:     for $x \in S$ do
 7:         Define $r_{j\cdot 2cn,x} : S \to \mathbb{F}$ by
                $$r_{j\cdot 2cn,x}(Y) = r_{j\cdot 2cn}(x, Y).$$
 8:         Define $\mathcal{C}_{j\cdot 2cn}$ by
                $$\mathcal{C}_{j\cdot 2cn} = \{C(Y) \in \mathbb{F}[Y] | \deg(C) < d - j \cdot 2cn\}.$$
 9:         Define $L_{j,0,x} = \text{RS-LIST-DECODER}(r_{j\cdot 2cn,x}(Y), d - j \cdot 2cn, t_j)$.
10:     end for
11:     for $k = 0, 1, \ldots, 2cn - 1$ do
12:         for $x \in S$ do
13:             Define $(G_x(Y), \delta_x) \in L_{j,k,x}$ to be the unique codeword (if any) with
                    $$\delta_x < \frac{n - d + j \cdot 2cn + k}{2}$$
14:             $\sigma_x \leftarrow \text{Coeff}_{Y^{d-j\cdot 2cn-k}}(G_x)$.
15:         end for
16:         Define the weighted function $f_{j\cdot 2cn+k} : S \to \mathbb{F} \times [0,1]$ by
                $$f_{j\cdot 2cn+k}(x) = \left(\sigma_x, \frac{\delta_x}{(n - d + j \cdot 2cn + k)/2}\right).$$
17:         Define $Q_{j\cdot 2cn+k} : S \to \mathbb{F}$ by
                $$Q_{j\cdot 2cn+k}(X) = \text{FAST-RS-DECODER}(f_{j\cdot 2cn+k}(X), j \cdot 2cn + k).$$
18:         for $x \in S$ do
19:             Define
                    $$L_{j,k+1,x} = \{(C - Q_{j\cdot 2cn+k}(x)Y^{d-j\cdot 2cn-k}, \delta_{C,x})$$
                    $$|C \in L_{j,k,x}, \text{Coeff}_{Y^{d-j\cdot 2cn-k}}(C) = Q_{j\cdot 2cn+k}(x)\}.$$
20:         end for
21:     end for
22: end for
23: Output: $\sum_{i=0}^{d} Q_i(\boldsymbol{X})Y^{d-i}$.
```

Correctness of Algorithm

View the received word as a matrix on $S \times S$, where the rows are indexed by $x \in S$ and the columns by $y \in S$. For correctness, we have to show two things. First, that Algorithm 3 produces the same row decodings $G_x(Y)$ as Algorithm 2. Second, that the algorithm actually extracts the coefficients of $C(X,Y) = \sum_{i=0}^{d} P_i(X)Y^{d-i}$ when viewed as a polynomial in Y, i.e. $Q_i(X) = P_i(X)$ for $i = 0, \ldots, d$. Define $r_{j\cdot 2cn+k} : S \times S \to \mathbb{F}$ by

$$r_{j\cdot 2cn+k}(X,Y) = r(X,Y) - \sum_{i=0}^{j\cdot 2cn+k-1} Q_i(X)Y^{d-i},$$

and define $r_{j\cdot 2cn+k,x} : S \to \mathbb{F}$ by

$$r_{j\cdot 2cn+k,x}(Y) = r_{j\cdot 2cn+k}(x,Y).$$

Then we want to show that in each of the $d+1$ iterations of (j,k), we have

$$G_x(Y) = \text{REED-SOLOMON-DECODER}\left(r_{j\cdot 2cn+k,x}(Y), d - j\cdot 2cn - k\right).$$

It is enough to instead show that the list $L_{j,k,x}$ contains all the polynomials of degree at most $d-j\cdot 2cn-k$ within distance $t_j = (n-d+j\cdot 2cn)/2 + cn > (n-d+j\cdot 2cn+k)/2$ of $r_{j\cdot 2cn+k,x}(Y)$. Furthermore, we want to show $Q_{j\cdot 2cn+k}(X) = P_{j\cdot 2cn+k}(X)$.

We prove this by induction on (j,k). The base case is $j = k = 0$. For each row $x \in S$, we have

$$L_{0,0,x} = \text{RS-LIST-DECODER}(r_{j\cdot 2cn,x}(Y), d - j\cdot 2cn, t_0).$$

The induction hypothesis is that for every $(j',k') < (j,k)$ in the lexicographic order, we have $L_{j',k',x} = \{(\overline{C}, \Delta(\overline{C}, r_{j'\cdot 2cn+k',x})) | \overline{C} \in \mathcal{C}_{j'\cdot 2cn+k'}, \Delta(\overline{C}, r_{j'\cdot 2cn+k',x}) < t_{j'}\}$ and that $Q_{j'\cdot 2cn+k'}(X) = P_{j'\cdot 2cn+k'}(X)$. We will show the corresponding statements hold true for (j,k).

If $k = 0$, then the fact that the algorithm extracted the correct coefficients thus far means that the $r_{j\cdot 2cn}$ are the same in both Algorithm 2 and Algorithm 3. Since $L_{j,0,x} = \text{RS-LIST-DECODER}(r_{j\cdot 2cn,x}(Y), d - j\cdot 2cn, t_j)$, the induction hypothesis on $L_{j,0,x}$ is met by the definition of RS-LIST-DECODER.

If $k \neq 0$, then we know from the induction hypothesis that

$$L_{j,k-1,x} = \{(\overline{C}, \Delta(\overline{C}, r_{j\cdot 2cn+k-1,x}))|\overline{C} \in \mathcal{C}_{j\cdot 2cn+k-1}, \Delta(\overline{C}, r_{j\cdot 2cn+k-1,x}) < t_j\}.$$

We want to say that

$$L_{j,k,x} = \{(\overline{C}, \Delta(\overline{C}, r_{j\cdot 2cn+k,x}))|\overline{C} \in \mathcal{C}_{j\cdot 2cn+k}, \Delta(\overline{C}, r_{j\cdot 2cn+k,x}) < t_j\}.$$

We defined $L_{j,k,x}$ in terms of $L_{j,k-1,x}$ to be:

$$\{(\overline{C} - Q_{j\cdot 2cn+k-1}(x)Y^{d-j\cdot 2cn-k+1}, \Delta(\overline{C}, r_{j\cdot 2cn+k-1,x})) \\ |\overline{C} \in L_{j,k-1,x}, \mathrm{Coeff}_{Y^{d-j\cdot 2cn-k+1}}(\overline{C}) = Q_{j\cdot 2cn+k-1}(x)\}.$$

As $Q_{j\cdot 2cn+k-1}(X) = P_{j\cdot 2cn+k-1}(X)$, $L_{j,k,x}$ is essentially obtained by taking the codewords with the correct leading coefficients and subtracting off the leading term. We claim that what we get is the set of all polynomials of degree at most $d - j \cdot 2cn - k$ within distance t_j of $r_{j\cdot 2cn+k,x}$.

Consider any $(G, \delta) \in L_{j,k,x}$. By definition of $L_{j,k,x}$, we know there exists a $\overline{C} \in L_{j,k-1,x}$ with $\mathrm{Coeff}_{Y^{d-j\cdot 2cn-k+1}}(\overline{C}) = Q_{j\cdot 2cn+k-1}(x)$ such that

$$(G, \delta) = (\overline{C} - Q_{j\cdot 2cn+k-1}(x)Y^{d-j\cdot 2cn-k+1}, \Delta(\overline{C}, r_{j\cdot 2cn+k-1,x})).$$

So we have

$$\begin{aligned} \overline{C} &= G + Q_{j\cdot 2cn+k-1}(x)Y^{d-j\cdot 2cn-k+1} \\ \delta &= \Delta(\overline{C}, r_{j\cdot 2cn+k-1,x}) < t_j. \end{aligned}$$

As $\mathrm{Coeff}_{Y^{d-j\cdot 2cn-k+1}}(\overline{C}) = Q_{j\cdot 2cn+k-1}(x)$, we have deg$(G)$ is at most $d - j \cdot 2cn - k$. Also, as $r_{j\cdot 2cn+k-1,x} = r_{j\cdot 2cn+k,x} + Q_{j\cdot 2cn+k-1}(x)Y^{d-j\cdot 2cn-k+1}$, we have $\Delta(G, r_{j\cdot 2cn+k,x}) = \Delta(\overline{C}, r_{j\cdot 2cn+k-1,x}) = \delta < t_j$. Hence, G is a polynomial of degree at most $d - j \cdot 2cn - k$ within distance t_j of $r_{j\cdot 2cn+k,x}$.

For the reverse inclusion, suppose G is a polynomial of degree at most $d - j \cdot 2cn - k$ at distance $\delta < t_j$ of $r_{j\cdot 2cn+k,x}$. Then $\overline{C} := G + Q_{j\cdot 2cn+k-1}(x)Y^{d-j\cdot 2cn-k+1} \in L_{j,k-1,x}$. Since

$\text{Coeff}_{Y^{d-j\cdot 2cn-k+1}}(\overline{C}) = Q_{j\cdot 2cn+k-1}(x)$, we get that $G = \overline{C} - Q_{j\cdot 2cn+k-1}(x)Y^{d-j\cdot 2cn-k+1} \in L_{j,k,x}$, as desired.

It remains to show that $Q_{j\cdot 2cn+k}(X) = P_{j\cdot 2cn+k}(X)$. As in the proof of Theorem 4.4.1, we show that $\Delta(f_{j\cdot 2cn+k}(X), P_{j\cdot 2cn+k}(X)) < \frac{n-j-\sqrt{n}}{2}$, so that the output of FAST-RS-DECODER$(f_{j\cdot 2cn+k}(X), j)$ is $P_{j\cdot 2cn+k}(X)$. Using the first part of the induction we just proved, we get the same $f_{j\cdot 2cn+k}(X)$ as in Algorithm 2. This means we can adopt a nearly identical argument to get to this step:

$$\Delta(f_{j\cdot 2cn+k}(X), P_{j\cdot 2cn+k}(X)) \le \frac{\Delta(r_{j\cdot 2cn+k}(X,Y), C_{j\cdot 2cn+k}(X,Y))}{n-d+j\cdot 2cn+k}.$$

From here, we get:

$$\begin{aligned}\Delta(f_{j\cdot 2cn+k}(X), P_{j\cdot 2cn+k}(X)) &< \frac{n^2}{2}\left(1-\frac{d}{n}-\frac{1}{\sqrt{n}}\right)\frac{1}{n-d+j\cdot 2cn+k}\\ &= \frac{n}{2}\cdot\frac{n-d-\sqrt{n}}{n-d+j\cdot 2cn+k}\\ &\le \frac{n}{2}\cdot\frac{n-j\cdot 2cn-k-\sqrt{n}}{n}\\ &= \frac{n-j\cdot 2cn-k-\sqrt{n}}{2}.\end{aligned}$$

Analysis of Runtime of Bivariate Reed-Muller Decoder

We run RS-LIST-DECODER $\frac{d}{2cn}n = \frac{\alpha}{2c}n = \frac{4\alpha}{(1-\alpha)^2}n$ times. Also, we run FAST-RS-DECODER $d = \alpha n$ times. As both of these algorithms run in $O(n \operatorname{polylog} n)$ time, the total runtime of the algorithm is $O(n^2 \operatorname{polylog}(n, |\mathbb{F}|))$, after accounting for field operations. As the input is of size n^2, this is near-linear in the size of the input.

$\square$

4.6 Near-Linear Time Decoding in the General Case

A more involved variation of the near-linear time decoding algorithm for bivariate Reed-Muller codes can be used to get a near-linear time algorithm for decoding Reed-Muller codes on any number of variables:

Theorem 4.6.1. *Let $\mathbb{F}$ be a finite field and let $S \subseteq \mathbb{F}$ be a nonempty subset of size $|S| = n$. Let $\beta > \frac{1}{2}$. Given a received word $r : S^m \to \mathbb{F}$, there is a $O\left(n^m \cdot \operatorname{polylog}(n, |\mathbb{F}|)\right)$ time* *algorithm to find the unique polynomial (if it exists)* $C \in \mathbb{F}[X_1, \ldots, X_m]$ *with* $\deg(C) \leq d$ *such that*

$$\Delta(r, C) < \frac{n^m}{2}\left(1 - \frac{d + (m-1)\beta\sqrt{n}}{n}\right).$$

As part of the algorithm for near linear time Reed-Muller decoding, we will need to decode Reed-Muller codes with uncertainties to various radii less than half their minimum distance. We require the following theorem to do such decodings efficiently.

Theorem 4.6.2. *Let* $\mathbb{F}$ *be a finite field and let* $S \subseteq \mathbb{F}$ *be a nonempty subset of size* $|S| = n$*. Let* $\beta > \frac{1}{2}$*, and let* e *be an integer satisfying* $0 \leq e < n - d - m\beta\sqrt{n}$*. Given a received word with uncertainties* $r : S^m \to \mathbb{F} \times [0, 1]$*, there is a* $O\left(\frac{n^{m+1}}{e+1} \cdot \mathrm{polylog}(n, |\mathbb{F}|)\right)$ *time algorithm to find the unique polynomial (if it exists)* $C \in \mathbb{F}[X_1, \ldots, X_m]$ *with* $\deg(C) \leq d$ *such that*

$$\Delta(r, C) < \frac{n^m}{2}\left(1 - \frac{d + m\beta\sqrt{n} + e}{n}\right).$$

Remark 4.6.3. *The algorithm requires the use of the FAST-RS-DECODER to a radius that is* $\beta\sqrt{n}$ *less than half the minimum distance. As long as* $\beta > \frac{1}{2}$*, the FAST-RS-DECODER runs in* $O(n \,\mathrm{polylog}\, n)$ *time.*

Proof of Theorem 4.6.2. The proof is by induction on the number of variables m. The proof of the base case of $m = 2$ is similar to the proof of the inductive step and will be handled last. Assume the theorem statement is true for m, and let RM-UNC-DECODER(f, d, s) denote the $O\left(\frac{n^{m+1}}{e+1} \cdot \mathrm{polylog}(n, |\mathbb{F}|)\right)$ time algorithm that finds the unique polynomial (if it exists) of degree at most d within distance s from f, where $f : S^m \to \mathbb{F} \times [0, 1]$ and s can be written as $\frac{n^m}{2}\left(1 - \frac{d+m\beta\sqrt{n}+e}{n}\right)$. We want to show that the theorem statement holds for $m + 1$ variables.

Algorithm 4 Decoding Reed Muller with Uncertainties

```
1: Input: $r : S^{m+1} \to \mathbb{F} \times [0,1]$.
2: for $j = 0, 1, \ldots, \frac{d}{e+1}$ do
3:     Let $t_j = \frac{n-d+j\cdot(e+1)-\beta\sqrt{n}}{2}$.
4:     Define $r_{j\cdot(e+1)} : S^m \times S \to \mathbb{F}$ by
           $$r_{j\cdot(e+1)}(\boldsymbol{X}, Y) = r(\boldsymbol{X}, Y) - \sum_{i=0}^{j\cdot(e+1)-1} Q_i(\boldsymbol{X})Y^{d-i}.$$
5:     for $\boldsymbol{x} \in S^m$ do
6:         Define $r_{j\cdot(e+1),\boldsymbol{x}} : S \to \mathbb{F}$ by
               $$r_{j\cdot(e+1),\boldsymbol{x}}(Y) = r_{j\cdot(e+1)}(\boldsymbol{x}, Y).$$
7:         Define $D_{j,0,\boldsymbol{x}}(Y) = \text{FAST-RS-DECODER}(r_{j\cdot(e+1),\boldsymbol{x}}(Y), d - j\cdot(e+1), t_j)$.
8:         Define $\delta_{\boldsymbol{x}} = \Delta(D_{j,0,\boldsymbol{x}}(Y), r_{j\cdot(e+1),\boldsymbol{x}}(Y))$.
9:     end for
10:    for $k = 0, 1, \ldots, e$ do
11:        for $\boldsymbol{x} \in S^m$ do
12:            if $\deg(D_{j,k,\boldsymbol{x}}(Y)) \le d - j\cdot(e+1) - k$ then
                   $$\sigma_{\boldsymbol{x}} \leftarrow \text{Coeff}_{Y^{d-j\cdot(e+1)-k}}(D_{j,k,\boldsymbol{x}}(Y)).$$
13:            end if
14:        end for
15:        Define the weighted function $f_{j\cdot(e+1)+k} : S^m \to \mathbb{F} \times [0,1]$ by
               $$f_{j\cdot(e+1)+k}(\boldsymbol{x}) = \left(\sigma_{\boldsymbol{x}}, \min\left\{1, \frac{\delta_{\boldsymbol{x}}}{(n-d+j\cdot(e+1)+k-\beta\sqrt{n}-e)/2}\right\}\right).$$
16:        Define $Q_{j\cdot(e+1)+k} : S^m \to \mathbb{F}$ by
               $$Q_{j\cdot(e+1)+k}(\boldsymbol{X}) = \text{RM-UNC-DECODER}\left(f_{j\cdot(e+1)+k}(\boldsymbol{X}), j\cdot(e+1)+k, \frac{n^m}{2}\left(1 - \frac{j\cdot(e+1)+k+m\beta\sqrt{n}}{n-d+j\cdot(e+1)+k}\right)\right).$$
```

Algorithm 4 Decoding Reed Muller with Uncertainties (Continued)

```
17:         for $\boldsymbol{x} \in S^m$ do
18:             Define $D_{j,k+1,\boldsymbol{x}} : S \to \mathbb{F}$ by
                    $$D_{j,k+1,\boldsymbol{x}} = D_{j,k,\boldsymbol{x}} - Q_{j\cdot(e+1)+k}(\boldsymbol{x})Y^{d-j\cdot(e+1)-k}.$$
19:         end for
20:     end for
21: end for
22: Output: $\sum_{i=0}^{d} Q_i(\boldsymbol{X})Y^{d-i}$.
```

The algorithm proceeds as follows: As before, we write $C(\boldsymbol{X}, Y) = \sum_{i=0}^{d} P_i(\boldsymbol{X})Y^{d-i}$, and find the P_i iteratively. In the i-th iteration, decode row $r_{i,\boldsymbol{x}}$, $\boldsymbol{x} \in S^m$ to a degree $d-i$ polynomial within radius $\frac{1}{2}(n-d+i-\beta\sqrt{n}-e)$ to get $D_{i,\boldsymbol{x}}(Y)$. To reduce the number of times we decode, we will instead decode to the larger radius $\frac{1}{2}(n-d+i-\beta\sqrt{n})$ and use this decoding for $e+1$ iterations. Construct the function $f_i : S^m \to \mathbb{F} \times [0,1]$ of (leading coefficient, uncertainty) $= \left(\text{Coeff}_{Y^{d-i}}(D_{i,\boldsymbol{x}}), \frac{\Delta(r_{i,\boldsymbol{x}}, D_{i,\boldsymbol{x}})}{(n-d+i-\beta\sqrt{n}-e)/2}\right)$. Finally, decode $f_i(\boldsymbol{X})$ to a degree i polynomial within radius $\frac{n^m}{2}\left(1 - \frac{i+m\beta\sqrt{n}}{n-d+i}\right)$ to get $Q_i(\boldsymbol{X})$.

Proof of Correctness

We have to show $Q_i(\boldsymbol{X}) = P_i(\boldsymbol{X})$. It is enough to show that

$$\Delta(f_i, P_i) < \frac{n^m}{2}\left(1 - \frac{i+m\beta\sqrt{n}}{n-d+i}\right) < \frac{n^m}{2}\left(1 - \frac{i}{n}\right).$$

Then P_i will be the unique polynomial of degree i within distance $\frac{n^m}{2}\left(1 - \frac{i+m\beta\sqrt{n}}{n-d+i}\right)$ of f_i. Since Q_i is a polynomial of degree i within distance $\frac{n^m}{2}\left(1 - \frac{i+m\beta\sqrt{n}}{n-d+i}\right)$ of f_i, Q_i must be equal to P_i.

When we decode $r_{i,\boldsymbol{x}}$ to radius $\frac{1}{2}(n-d+i-\beta\sqrt{n}-e)$, there are four possibilities:

1. The decoding is unsuccessful. In this case, we set $D_{i,\boldsymbol{x}}$ to be any polynomial of degree $n-d+i$ and set the uncertainty $u_i = 1$. The contribution to $\Delta(f_i, P_i)$ is $\Delta(f_i(\boldsymbol{x}), P_i(\boldsymbol{x})) = 1/2$, which is bounded above by $\frac{1}{2}\frac{\Delta(r_{i,\boldsymbol{x}}, C_{i,\boldsymbol{x}})}{(n-d+i-\beta\sqrt{n}-e)/2}$.

2. The decoding succeeds and is correct. In this case, $D_{i,\boldsymbol{x}} = C_{i,\boldsymbol{x}}$, so $\Delta(f_i(\boldsymbol{x}), P_i(\boldsymbol{x})) = \frac{1}{2}\frac{\Delta(r_{i,\boldsymbol{x}}, C_{i,\boldsymbol{x}})}{(n-d+i-\beta\sqrt{n}-e)/2}$.

3. The decoding succeeds, but is the wrong codeword, whose leading coefficient disagrees with that of the correct codeword. In this case, $D_{i,\boldsymbol{x}} \neq C_{i,\boldsymbol{x}}$, so

$$\begin{aligned}\Delta(f_i(\boldsymbol{x}), P_i(\boldsymbol{x})) &= 1 - \frac{1}{2}\frac{\Delta(r_{i,\boldsymbol{x}}, D_{i,\boldsymbol{x}})}{(n-d+i-\beta\sqrt{n}-e)/2} \\ &\leq 1 - \frac{(n-d+i) - \Delta(r_{i,\boldsymbol{x}}, C_{i,\boldsymbol{x}})}{(n-d+i-\beta\sqrt{n}-e)} \\ &\leq 1 - \frac{(n-d+i-\beta\sqrt{n}-e) - \Delta(r_{i,\boldsymbol{x}}, C_{i,\boldsymbol{x}})}{(n-d+i-\beta\sqrt{n}-e)} \\ &\leq \frac{\Delta(r_{i,\boldsymbol{x}}, C_{i,\boldsymbol{x}})}{(n-d+i-\beta\sqrt{n}-e)}.\end{aligned}$$

4. The decoding succeeds, but is the wrong codeword, whose leading coefficient matches that of the correct codeword. As in the previous case, $D_{i,\boldsymbol{x}} \neq C_{i,\boldsymbol{x}}$, and we have:

$$\begin{aligned}\Delta(f_i(\boldsymbol{x}), P_i(\boldsymbol{x})) &= \frac{1}{2}\frac{\Delta(r_{i,\boldsymbol{x}}, D_{i,\boldsymbol{x}})}{(n-d+i-\beta\sqrt{n}-e)/2} \\ &\leq 1 - \frac{1}{2}\frac{\Delta(r_{i,\boldsymbol{x}}, D_{i,\boldsymbol{x}})}{(n-d+i-\beta\sqrt{n}-e)/2} \\ &\leq \frac{\Delta(r_{i,\boldsymbol{x}}, C_{i,\boldsymbol{x}})}{(n-d+i-\beta\sqrt{n}-e)}.\end{aligned}$$

Putting it all together, we have:

$$\begin{aligned}\Delta(f_i, P_i) &\leq \sum_{\boldsymbol{x}\in S^m} \frac{\Delta(r_{i,\boldsymbol{x}}, C_{i,\boldsymbol{x}})}{n-d+i-\beta\sqrt{n}-e} \\ &= \frac{\Delta(r_i, C_i)}{n-d+i-\beta\sqrt{n}-e} \\ &= \frac{\Delta(r, C)}{n-d+i-\beta\sqrt{n}-e} \\ &\leq \frac{\frac{n^{m+1}}{2}\left(1 - \frac{d+(m+1)\beta\sqrt{n}+e}{n}\right)}{n-d+i-\beta\sqrt{n}-e} \\ &= \frac{n^m}{2}\frac{n-d-(m+1)\beta\sqrt{n}-e}{n-d+i-\beta\sqrt{n}-e}\end{aligned}$$

$$\begin{aligned} &\leq \frac{n^m}{2}\frac{n-d-m\beta\sqrt{n}}{n-d+i} \\ &= \frac{n^m}{2}\left(1-\frac{i+m\beta\sqrt{n}}{n-d+i}\right). \end{aligned}$$

Analysis of Runtime

The algorithm can be divided into two parts:

1. Constructing the f_i, $i = 0, \dots, d$.
2. Decoding the f_i to get the P_i, $i = 0, \dots, d$.

The dominant contribution to the runtime when constructing f_i comes from all the Reed-Solomon decodings with uncertainties we have to do to get the $D_{i,\boldsymbol{x}}(Y)$. For every $e+1$ iterations, we have to decode each row $x \in S^m$ again. The total number of such decodings is given by $\frac{n}{e+1} \cdot n^m = \frac{n^{m+1}}{e+1}$. Since each Reed-Solomon decoding with uncertainty can be done in $O(n \operatorname{polylog} n)$ time via the FAST-RS-DECODER, we have that the runtime of this part of the algorithm is $O\left(\frac{n^{m+2}}{e+1} \operatorname{polylog} n\right)$.

To understand the runtime of the second part of the algorithm, we will compute the runtime of decoding f_i for some fixed i. Note that decoding f_i is a Reed-Muller decoding with uncertainties problem with m variables. So we will write the decoding radius $\frac{n^m}{2}\left(1-\frac{i+m\beta\sqrt{n}}{n-d+i}\right)$ in the form $\frac{n^m}{2}\left(1-\frac{i+m\beta\sqrt{n}+e_i}{n}\right)$ and apply the induction hypothesis to get a $O\left(\frac{n^{m+1}}{e_i+1} \cdot \operatorname{polylog} n\right)$ runtime. We now need to compute e_i:

$$\begin{aligned} e_i &= n\frac{i+m\beta\sqrt{n}}{n-d+i} - (i+m\beta\sqrt{n}) \\ &= (i+m\beta\sqrt{n})\left(\frac{n}{n-d+i}-1\right) \\ &= \frac{(i+m\beta\sqrt{n})(d-i)}{n-d+i}. \end{aligned}$$

The runtime for all $d+1$ iterations from $i = 0, \dots, d$ is then

$$O\left(\sum_{i=0}^{d} \frac{1}{e_i+1} \cdot n^{m+1} \operatorname{polylog} n\right).$$

It remains to bound $\displaystyle\sum_{i=0}^{d} \frac{1}{e_i+1}$ from above:

$$\begin{aligned}
& \sum_{i=0}^{d} \frac{1}{e_i+1} \\
\leq\ & \sum_{i=0}^{d} \min\left(1, \frac{1}{e_i}\right) \\
\leq\ & 4 + \sum_{i=2}^{d-2} \frac{1}{e_i} \\
\leq\ & 4 + \int_1^{d-1} \frac{n-d+t}{(t+m\beta\sqrt{n})(d-t)} dt.
\end{aligned}$$

The last inequality is a simple Riemann sum bound using the fact that the function $\frac{n-d+t}{(t+m\beta\sqrt{n})(d-t)}$ decreases then increases continuously on $[1, d-1]$. Computing the integral is a straightforward partial fraction decomposition:

$$\begin{aligned}
& \frac{n-d+t}{(t+m\beta\sqrt{n})(d-t)} \\
=\ & \frac{n}{(t+m\beta\sqrt{n})(d-t)} - \frac{1}{t+m\beta\sqrt{n}} \\
=\ & \frac{n}{d+m\beta\sqrt{n}} \left(\frac{1}{t+m\beta\sqrt{n}} + \frac{1}{d-t}\right) - \frac{1}{t+m\beta\sqrt{n}} \\
\leq\ & \frac{1}{\alpha} \left(\frac{1}{t+m\beta\sqrt{n}} + \frac{1}{d-t}\right) - \frac{1}{t+m\beta\sqrt{n}} \\
=\ & \left(\frac{1}{\alpha} - 1\right) \frac{1}{t+m\beta\sqrt{n}} + \frac{1}{\alpha} \cdot \frac{1}{d-t}
\end{aligned}$$

So we have:

$$\begin{aligned}
& \int_1^{d-1} \frac{n-d+t}{(t+m\beta\sqrt{n})(d-t)} dt \\
\leq\ & \int_1^{d-1} \left[\left(\frac{1}{\alpha} - 1\right) \frac{1}{t+m\beta\sqrt{n}} + \frac{1}{\alpha} \cdot \frac{1}{d-t}\right] dt \\
\leq\ & O\left(\left(\frac{1}{\alpha} - 1\right) \log n + \frac{1}{\alpha} \log n\right) \\
=\ & O\left(\left(\frac{2}{\alpha} - 1\right) \log n\right) \\
=\ & O(\log n).
\end{aligned}$$

So the runtime for all $d+1$ iterations is:

$$O\left((4 + O(\log n)) \cdot n^{m+1} \operatorname{polylog} n\right) = O(n^{m+1} \operatorname{polylog} n).$$

This means the runtime for both parts of the algorithm is just $O\left(\frac{n^{m+2}}{e+1} \operatorname{polylog} n\right)$.

Base Case

The algorithm for $m = 2$ is almost identical to that for general m, except that we decode $f_i(X)$ to a degree i polynomial within the larger radius $\frac{n}{2}\left(1 - \frac{i+\beta\sqrt{n}}{n}\right)$ to get $Q_i(\boldsymbol{X})$. Note that this radius is still less than half the minimum distance of the Reed-Solomon code of degree i. The correctness of the algorithm follows from the fact that P_i is still the unique polynomial within distance $\frac{n}{2}\left(1 - \frac{i+\beta\sqrt{n}}{n}\right)$ of f_i.

We can again analyze the runtime of the two parts of the algorithm. The runtime for finding the f_i follows the same analysis as before and is $O(\frac{n^3}{e+1} \operatorname{polylog} n)$. For decoding the f_i, we simply call the FAST-RS-DECODER for $d+1$ different values of i. This has a runtime of $O(dn \operatorname{polylog} n) \leq O(n^2 \operatorname{polylog} n)$. So we get a total runtime of $O(\frac{n^3}{e+1} \operatorname{polylog} n)$.

□

The algorithm for general Reed-Muller decoding follows the same strategy as the algorithm for Reed-Muller decoding with uncertainties to a radius less than half the minimum distance. Recall that to get the f_i in the algorithm, we only needed to Reed-Solomon decode to a radius significantly less than half the minimum distance. We then saved on the number of Reed-Solomon decodings by instead decoding to half the minimum distance and reusing that decoding for many iterations. We now want to Reed-Muller decode to near half the minimum distance. Using the same algorithm doesn't save on enough Reed-Solomon decodings to achieve near linear time. However, when there are no uncertainties in the original received word, we can list decode efficiently to a radius significantly larger than half the minimum distance. We then use the lists for many iterations to generate the f_i before list decoding again.

Proof of Theorem 4.6.1. In the case where the number of variables is 2, we are in the setting of decoding bivariate Reed-Muller codes to near half the minimum distance, which can be done in near-linear time by Theorem 4.5.1. Assume now that $m \geq 2$ and that we have a Reed-Muller code in $m+1$ variables.

Algorithm 5 Decoding Reed Muller

```
1: Input: $r : S^{m+1} \to \mathbb{F}$.
2: Let $c = ((1-\alpha)^2/8)$.
3: for $j = 0, 1, \dots, \frac{d}{2cn}$ do
4:     Let $t_j = \frac{n-d+j\cdot 2cn}{2} + cn$.
5:     Define $r_{j\cdot 2cn} : S^m \times S \to \mathbb{F}$ by
           $$r_{j\cdot 2cn}(\boldsymbol{X}, Y) = r(\boldsymbol{X}, Y) - \sum_{i=0}^{j\cdot 2cn-1} Q_i(\boldsymbol{X}) Y^{d-i}.$$
6:     for $\boldsymbol{x} \in S^m$ do
7:         Define $r_{j\cdot 2cn,\boldsymbol{x}} : S \to \mathbb{F}$ by
               $$r_{j\cdot 2cn,\boldsymbol{x}}(Y) = r_{j\cdot 2cn}(\boldsymbol{x}, Y).$$
8:         Define $L_{j,0,\boldsymbol{x}} = \text{RS-LIST-DECODER}(r_{j\cdot 2cn,\boldsymbol{x}}(Y), d - j\cdot 2cn, t_j)$.
9:     end for
10:    for $k = 0, 1, \dots, 2cn-1$ do
11:        for $\boldsymbol{x} \in S^m$ do
12:            Define $(G_x(Y), \delta_x) \in L_{j,k,x}$ to be the unique codeword (if any) with
                   $$\delta_x < \frac{n-d+j\cdot 2cn+k}{2}$$
13:            $\sigma_x \leftarrow \text{Coeff}_{Y^{d-j\cdot 2cn-k}}(G_x)$.
14:        end for
15:        Define the weighted function $f_{j\cdot 2cn+k} : S^m \to \mathbb{F} \times [0,1]$ by
               $$f_{j\cdot 2cn+k}(\boldsymbol{x}) = \left(\sigma_{\boldsymbol{x}}, \min\left\{1, \frac{\delta_{\boldsymbol{x}}}{(n-d+j\cdot 2cn+k)/2}\right\}\right).$$
16:        Define $Q_{j\cdot 2cn+k} : S^m \to \mathbb{F}$ by
               $$Q_{j\cdot 2cn+k}(\boldsymbol{X}) = \text{RM-UNC-DECODER}\left(f_{j\cdot 2cn+k}(\boldsymbol{X}), j\cdot 2cn+k, \frac{n^{m-1}}{2}\left(1 - \frac{j\cdot 2cn+k+(m-1)\beta\sqrt{n}}{n-d+j\cdot 2cn+k}\right)\right).$$
```

Algorithm 5 Decoding Reed Muller (Continued)

```
17:             for $\boldsymbol{x} \in S^m$ do
18:
                    $L_{j,k+1,\boldsymbol{x}} \leftarrow \{(C - Q_{j\cdot 2cn+k}(\boldsymbol{x})Y^{d-j\cdot 2cn-k}, \delta_{C,\boldsymbol{x}})$
                        $|(C, \delta_{C,\boldsymbol{x}}) \in L_{j,k,\boldsymbol{x}}, \mathrm{Coeff}_{Y^{d-j\cdot 2cn-k}}(C) = Q_{j\cdot 2cn+k}(\boldsymbol{x})\}.$
19:             end for
20:         end for
21:     end for
22:     Output: $\sum_{i=0}^{d} Q_i(\boldsymbol{X})Y^{d-i}$.
```

The decoding algorithm for a $m+1$-variate Reed-Muller code is as follows: In the i-th iteration, list decode row $r_{i,\boldsymbol{x}}$, $\boldsymbol{x} \in S^m$ to obtain a list $L_{i,\boldsymbol{x}}$ of all degree $\leq d-i$ polynomials within radius $\frac{1}{2}(n-d+i+cn)$ along with their distances from $r_{i,\boldsymbol{x}}$, where $c = \frac{(1-\alpha)^2}{8}$. Search the list to get the degree $\leq d-i$ polynomial within distance $\frac{1}{2}(n-d+i)$ from $r_{i,\boldsymbol{x}}$, call it $D_{i,\boldsymbol{x}}(Y)$. We use the lists for cn iterations before list decoding again. Construct function $f_i : S^m \to \mathbb{F} \times [0,1]$ of (leading coefficient, uncertainty) $= \left(\mathrm{Coeff}_{Y^{d-i}}(D_{i,\boldsymbol{x}}), \frac{\Delta(r_{i,\boldsymbol{x}}, D_{i,\boldsymbol{x}})}{(n-d+i)/2}\right)$. Decode $f_i(\boldsymbol{X})$ to a degree i polynomial within radius $\frac{n^m}{2}\left(1 - \frac{i+m\beta\sqrt{n}}{n-d+i}\right)$ to get $Q_i(\boldsymbol{X})$.

Proof of Correctness

As before, we want to show that $Q_i(\boldsymbol{X}) = P_i(\boldsymbol{X})$. It is enough to show

$$\Delta(f_i, P_i) < \frac{n^m}{2}\left(1 - \frac{i+m\beta\sqrt{n}}{n-d+i}\right).$$

We can use a similar analysis of $\Delta(f_i, P_i)$ to the one in Theorem 4.6.2 to get to the following step:

$$\Delta(f_i, P_i) \leq \frac{\Delta(r, C)}{n-d+i}.$$

So we have:

$$\Delta(f_i, P_i) \quad \leq \quad \frac{\frac{n^{m+1}}{2}\left(1 - \frac{d+m\beta\sqrt{n}}{n}\right)}{n-d+i}$$

$$\begin{aligned} &= \frac{n^m}{2} \frac{n-d-m\beta\sqrt{n}}{n-d+i} \\ &= \frac{n^m}{2}\left(1 - \frac{i+m\beta\sqrt{n}}{n-d+i}\right). \end{aligned}$$

Analysis of Runtime

Decoding the f_i over the $d+1$ values of i can be done in $O(n^{m+1} \operatorname{polylog} n)$ following the same runtime analysis from Theorem 4.6.2. For constructing the f_i, we do $O(n^m)$ Reed-Solomon list decodings taking $O(n \operatorname{polylog} n)$ time each. Within any given list, we need to compute uncertainties for each element of the list. This also takes $O(n \operatorname{polylog} n)$ time for each list. Finally, we update the lists at each iteration by identifying the elements with the correct leading coefficient and taking away their leading terms. Since the list size is constant, and there are $O(n^m)$ lists to update in each iteration, the updating takes $O(n^m d) = O(n^{m+1})$ over $d+1$ iterations. Hence the total runtime is $O(n^{m+1} \operatorname{polylog} n)$ as desired.

□

4.7 Open Problems

We conclude with some open problems.

1. The problem of list-decoding multivariate polynomial codes up to the Johnson radius is a very interesting open problem left open by our work. Generalizing our approach seems to require progress on another very interesting open problem, that of list-decoding Reed-Solomon concatenated codes. See [32] for the state of the art on this problem.

2. It would be interesting to understand the relationship between our algorithms and the $m+1$-variate interpolation-based list-decoding algorithm of Sudan [25]. Their decoding radii are incomparable, and perhaps there is some insight into the polynomial method, which is known to face some difficulties in > 2 dimensions, that can be gained here.

3. It would be interesting to see if one can decode multiplicity codes [33] on arbitrary

product sets upto half their minimum distance. Here too, we know algorithms that decode upto the minimum distance only in the case when S is very algebraically special (from [34]), or if the degree d is very small compared to $|S|$ (via an $m + 1$-variate interpolation algorithm, similar to [25]).

References

[1] C. Beck and Y. Li. Represent MOD function by low degree polynomial with unbounded one-sided error, arXiv:1304.0713, (2013).

[2] W. Eberly. Very fast parallel polynomial arithmetic, *SIAM J. Comput.*, **18** (1989), 955-976.

[3] F. E. Fich and M. Tompa. The parallel complexity of exponentiating polynomials over finite fields, *J. AMC*, **35** (1988), 651-667.

[4] J. von zur Gathen. Efficient exponentiation in finite fields (extended abstract), *FOCS*, (1991), 384-391.

[5] A. Healy and E. Viola. Constant-depth circuits for arithmetic in finite fields of characteristic two, *STACS*, volume **3884** of *Lecture Notes in Computer Science*, Springer, New York, 2006.

[6] W. Hesse, E. Allender, and D. A. M. Barrington. Uniform constant-depth circuits for division and iterated multiplication, *J. Comput. Syst. Sci.*, **65** (2002) 695-716.

[7] S. Kopparty. On the complexity of powering in finite fields, *STOC* (2011).

[8] A. Razborov. Lower bounds on the size of bounded depth circuits over a complete basis with logical addition, *MATHNASUSSR: Mathematical Notes of the Academy of Sciences of the USSR*, **41**, (1987).

[9] I. E. Shparlinsky. Number theoretic methods in cryptography: complexity lower bounds, volume **17** of *Progress in computer science and applied logic*, Birkhäuser Verlag, 1999.

[10] R. Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity, *STOC* (1987), 77-82.

[11] R. Smolensky, On representations by low-degree polynomials, *FOCS* (1993), 130-138.

[12] N. Alon, *Combinatorial nullstellensatz*, Combinatorics, Probabability and Computing, 8: 7-29, 1999.

[13] N. Alon, M. B. Nathanson, and I. Z. Ruzsa, *Adding distinct congruence classes modulo a prime.* American Math. Monthly 102:250255, 1995.

[14] B. Bukh, *Sums of dilates.* Combinatorics, Probability and Computing, 17:627-639, 2008.

[15] G.F. Pontiveros, *Sums of dilates in* $\mathbb{Z}_p$, Combinatorics, Probability and Computing, 22:282-293, 2013.

[16] T. Tao, and V. Vu, *Additive Combinatorics*, Cambridge studies in advanced Mathematics, Cambridge University Press, 2006.

[17] D. E. Muller, *Application of boolean algebra to switching circuit design and to error detection*, Electronic Computers, Transactions of the I.R.E. Professional Group on EC-3, 3:6-12, 1954.

[18] I. Reed, *A class of multiple-error-correcting codes and the decoding scheme*, Information Theory, Transactions of the I.R.E. Professional Group on 4, 4:38-49, 1954.

[19] I. S. Reed and G. Solomon, *Polynomial Codes over Certain Finite Fields*, Journal of the Society for Industrial and Applied Mathematics (SIAM) 8:300-304, 1960.

[20] J. L. Massey, *Shift-register synthesis and BCH decoding*, IEEE Transactions on Information Theory, 15:122-127, 1969.

[21] F. J. MacWilliams and N. J. A. Sloane, *The theory of error correcting codes*, Elsevier, 1977.

[22] V. Guruswami and M. Sudan, *Improved decoding of Reed-Solomon and Algebraic-geometric codes* IEEE Transactions on Information Theory, 45:1757-1767, 1999.

[23] R. Pellikaan and X. Wu, *List decoding of q-ary Reed-Muller codes* IEEE Transactions on Information Theory, 50:679-682, 2004.

[24] T. Kasami, S. Lin, and W. Peterson, *Polynomial Codes*, IEEE Transactions on Information Theory, 14:807-814, 2006.

[25] M. Sudan, *Decoding of Reed Solomon Codes beyond the Error-Correction Bound*, J. Complexity, 13:180-193, 1997.

[26] P. Gopalan, V. Guruswami, and P. Raghavendra, *List Decoding Tensor Products and Interleaved Codes*, SIAM J. Comput., 40:1432-1462, 2011.

[27] E. Berlekamp, *Bounded distance +1 soft-decision Reed-Solomon decoding*, IEEE Transactions on Information Theory, 42:704-720, 1996.

[28] L. R. Welch and E. R. Berlekamp, *Error correction of algebraic block codes*, US Patent Number 4633470, 1986.

[29] G. D. Forney, *Generalized Minimum Distance decoding*, IEEE Transactions on Information Theory, 12:125-131, 1966.

[30] Y. Cassuto and J. Bruck, *A Combinatorial Bound on the List Size*, Paradise Laboratory Technical report, California Institute of Technology, 2004.

[31] M. Alekhnovich, *Linear Diophantine Equations Over Polynomials and Soft Decoding of Reed-Solomon Codes*, IEEE Transactions on Information Theory, 51:2257-2265, 2005.

[32] V. Guruswami and M. Sudan, *Decoding concatenated codes using soft information*, Proceedings of the 17th IEEE Annual Conference on Computational Complexity, 122-131, 2002.

[33] S. Kopparty, S. Saraf, and S. Yekhanin, *High-rate Codes with Sublinear-time Decoding*, Journal of the ACM, 61, 28:1-20, 2014.

[34] S. Kopparty, *List decoding multiplicity codes*, Theory of Computing, 11:149-182, 2015.

A Near-Linear Time Soft Decoding of Reed-Solomon Codes

In this section, we present a near-linear time algorithm to soft decode Reed-Solomon codes to almost half the minimum distance. This result can be used to achieve near-linear time decoding of Reed-Muller codes to almost half the minimum distance.

Lemma A.1. *Let $\mathbb{F}$ be a finite field and let $S \subseteq \mathbb{F}$ be a nonempty subset of size $|S| = n$. There is a randomized algorithm FAST-RS-DECODER(r, d) that given a received word with uncertainties $r : S \to \mathbb{F} \times [0, 1]$, finds the unique polynomial (if it exists) $C \in \mathbb{F}[X]$ satisfying $\deg(C) \leq d$ and $\Delta(r, C) < \frac{n-d-\sqrt{n}}{2}$ with probability $3/4$ in time $O(n \operatorname{polylog}(n))$.*

Proof. The near-linear time algorithm for FAST-RS-DECODER(r, d) is based on Forney's generalized minimum distance decoding of concatenated codes.

Given a received word $r : S \to \mathbb{F} \times [0, 1]$, suppose there is a polynomial f of degree at most d such that $\Delta(f, r) < \frac{n-d-\sqrt{n}}{2}$. Let $S = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$, and write $r(\alpha_i) = (\beta_i, u_i), i \in [n]$. We may view r as a set of n points (α_i, β_i) with uncertainties u_i. The general idea of the algorithm is to erase the i-th point with probability u_i, and perform errors and erasures decoding of the resulting Reed-Solomon code. We denote the errors and erasures Reed-Solomon decoder by EE-DECODER(r', d), which takes a received word $r' : S \to \mathbb{F} \times [0, 1] \cup \{?\}$ and a degree d and returns the polynomial of degree at most d that is within $\frac{n-d}{2}$ of r'.

Algorithm 6 Fast Reed-Solomon Decoding with Uncertainties

```
1: Input: $r : S \to \mathbb{F} \times [0,1]$.
2: for $i = 1, 2, \ldots, n$ do
3:     $p_i \leftarrow$ RANDOM($[0,1]$).
4:     Define $r' : S \to (\mathbb{F} \cup \{?\})$ by
           $$r'(\alpha_i) = \begin{cases} \beta_i & \text{if } p_i \leq u_i \\ ? & \text{if } p_i > u_i \end{cases}.$$
5: end for
6: $g \leftarrow$ EE-DECODER$(r', d)$.
7: Output: $g$.
```

We say that a point is an *erasure* if it is erased by the algorithm. We say that a point (α_i, β_i) is an *error* if (α, β) is not an erasure and $f(\alpha_i) \neq \beta_i$. Let E be the number of errors, and let F be the number of erasures. As the resulting $n - F$ points form a Reed-Solomon code of block length $n - F$ and degree d, the algorithm outputs f as long as

$$2E + F < n - d.$$

We will use Chebyshev's inequality to show that $2E + F < n - d$ with probability at least $\frac{3}{4}$. To help us compute the expectation and variance of $2E + F$, we write E and F as a sum of indicator random variables. Let $A = \{i \in [n] | f(\alpha_i) = \beta_i\}$ be the set of agreeing indices, and let $D = \{i \in [n] | f(\alpha_i) \neq \beta_i\}$ be the set of disagreeing indices. Let $T = \{i \in [n] | (\alpha_i, \beta_i) \text{ is erased}\}$ be the set of erasure indices.

Then we can write

$$\begin{aligned} E &= \sum_{i \in D} 1_{i \notin T} \\ F &= \sum_{i \in [n]} 1_{i \in T}. \end{aligned}$$

We then can show $\mathbb{E}[2E + F]$ is less than $n - d$ by a significant amount $\sqrt{n}$:

$$\begin{aligned}
\mathbb{E}[2E+F] &= 2\sum_{i\in D}(1-u_i) + \sum_{i\in[n]} u_i \\
&= 2\sum_{i\in D}(1-u_i) + \sum_{i\in D} u_i + \sum_{i\in A} u_i \\
&= 2\left(\sum_{i\in D}\left(1-\frac{u_i}{2}\right) + \sum_{i\in A}\frac{u_i}{2}\right) \\
&= 2\Delta(f,r) \\
&< n - d - \sqrt{n}.
\end{aligned}$$

Finally, we show that $\mathrm{Var}(2E + F)$ is small:

$$\begin{aligned}
&\mathrm{Var}(2E+F) \\
= \;& 4\mathrm{Var}(E) + 4\mathrm{Cov}(E,F) + \mathrm{Var}(F) \\
= \;& 4\sum_{i\in D} u_i(1-u_i) + 4\left(\mathbb{E}\left[\sum_{i\in D}\sum_{j\in[n]} 1_{i\notin T\cap j\in T}\right] - \sum_{i\in D}(1-u_i)\sum_{j\in[n]} u_j\right) + \sum_{i\in[n]} u_i(1-u_i) \\
= \;& 4\sum_{i\in D} u_i(1-u_i) + 4\left(\mathbb{E}\left[\sum_{i\in D}\sum_{j\neq i}(1-u_i)u_j\right] - \sum_{i\in D}\sum_{j\in[n]}(1-u_i)u_j\right) + \sum_{i\in[n]} u_i(1-u_i) \\
= \;& 4\sum_{i\in D} u_i(1-u_i) - 4\sum_{i\in D} u_i(1-u_i) + \sum_{i\in[n]} u_i(1-u_i) \\
= \;& \sum_{i\in[n]} u_i(1-u_i) \\
\leq \;& \frac{n}{4}.
\end{aligned}$$

By Chebyshev's inequality, $\Pr(2E + F \geq n - d) \leq \frac{1}{4}$. Hence we have $\Pr(2E + F < n - d) \geq \frac{3}{4}$. That is, with probability at least $\frac{3}{4}$, the algorithm outputs f.

We now analyze the runtime of our fast Reed-Solomon decoder. The erasures can be done in $O(n)$ time. Also, as the EE-DECODER is essentially a Reed-Solomon decoder to half the minimum distance, it runs in time $O(n \operatorname{polylog} n)$ [27, 28]. This gives a total runtime of $O(n \operatorname{polylog} n)$.

$\square$

Note that by running the algorithm $\log n$ times, we get that with probability at least $1 - (1/4)^{\log n} = 1 - 1/n^{\log 4}$, we still find f in $O(n \operatorname{polylog} n)$ time.