Algorithmizing the Multiplicity Schwartz-Zippel Lemma

Received Mar 29, 2024
Revised Apr 22, 2025
Accepted May 23, 2025
Published Dec 18, 2025

Key words and phrases
Error-correcting codes, coding theory, algebraic coding theory, information theory

Siddharth Bhandari [a]
Prahladh Harsha [b]
Mrinal Kumar [b]
Ashutosh Shankar [b]

a Toyota Technological Institute, Chicago, USA

b Tata Institute of Fundamental Research, Mumbai, India

ABSTRACT. The multiplicity Schwartz-Zippel lemma asserts that over a field, a low-degree polynomial cannot vanish with high multiplicity very often on a sufficiently large product set. Since its discovery in a work of Dvir, Kopparty, Saraf and Sudan [6], the lemma has found numerous applications in both math and computer science; in particular, in the definition and properties of multiplicity codes by Kopparty, Saraf and Yekhanin [15].

In this work, we show how to algorithmize the multiplicity Schwartz-Zippel lemma for arbitrary product sets over any field. In other words, we give an efficient algorithm for unique decoding of multivariate multiplicity codes from half their minimum distance on arbitrary product sets over any field and any constant multiplicity parameter. Previously, such an algorithm was known either when the underlying product set had a nice algebraic structure (for instance, was a subfield) [14] or when the underlying field had large (or zero) characteristic, the multiplicity parameter was sufficiently large and the multiplicity code had distance bounded away from 1 [4].

Our algorithm builds upon a result of Kim & Kopparty [13] who gave an algorithmic version of the Schwartz-Zippel lemma (without multiplicities) or equivalently, an efficient algorithm for unique decoding of Reed-Muller codes over arbitrary product sets. We introduce a refined notion of distance based on the multiplicity Schwartz-Zippel lemma and design a unique decoding

A preliminary version of this article appeared at SODA 2023 [3]. Research of the first, second and fourth authors supported in part by the Department of Atomic Energy, Government of India, under project 12-R&D-TFR-5.01-0500. Work done when the first author was a PhD student at TIFR where his research was supported in part by the Google PhD Fellowship, and the third author was at IIT Bombay. Research of the second author supported by the Swarnajayanti Fellowship.

Cite as Siddharth Bhandari, Prahladh Harsha, Mrinal Kumar, Ashutosh Shankar. Algorithmizing the Multiplicity Schwartz-Zippel Lemma. TheoretiCS, Volume 4 (2025), Article 29, 1-53.

https://theoretics.episciences.org
DOI 10.46298/theoretics.25.29

algorithm for this distance measure. On the way, we give an alternative analysis of Forney's classical generalized minimum distance decoder that might be of independent interest.

1. Introduction

The *degree-mantra* states that any univariate non-zero polynomial $P \in \mathbb{F}[x]$ of degree d, where $\mathbb{F}$ is a field, has at most d zeros even including multiplicities. A generalization of this basic maxim to the multivariate setting is the well-known *Schwartz-Zippel* Lemma, also sometimes referred to as the *Polynomial-Identity* Lemma, (due to Ore [20], Schwartz [23], Zippel [26] and DeMillo & Lipton [5]). This Lemma states that if $\mathbb{F}$ is a field, and $P \in \mathbb{F}[x_1, x_2, \ldots, x_m]$ is a *non-zero* polynomial of degree d, and $T \subseteq \mathbb{F}$ is an arbitrary finite subset of $\mathbb{F}$, then the number of points on the grid T^m where P is zero is upper bounded by $d|T|^{m-1}$. A generalization of this lemma that incorporates the *multiplicity* aspect of the (univariate) degree-mantra was proved by Dvir, Kopparty, Saraf and Sudan [6]. This multiplicity Schwartz-Zippel Lemma (henceforth, referred to as the multiplicity SZ lemma for brevity) states that the number of points on the grid T^m where f is zero with *multiplicity*[1] at least s is upper bounded by $\frac{d|T|^{m-1}}{s}$.[2]

This innately basic statement about low degree polynomials is crucial to many applications in both computer science and math, e.g. see [12]. Of particular interest to this work is the significance of this lemma and its variants to algebraic error-correcting codes that we now discuss in more detail.

1.1 The coding theoretic context and main theorem

SZ lemma and its variants play a fundamental role in the study of error-correcting codes. The univariate version (referred to as the *degree mantra* in the first paragraph) implies a lower bound on the distance of Reed-Solomon codes whereas for the general multivariate setting, the SZ lemma gives the distance of Reed-Muller codes. In its multiplicity avatar (both the univariate and multivariate settings), the lemma implies a lower bound on the distance of the so-called *multiplicity codes*. Informally, multiplicity codes are generalizations of Reed-Muller codes where in addition to the evaluation of a message polynomial at every point on a sufficiently large grid, we also have the evaluations of all partial derivatives of the polynomial up to some order at every point on a grid. Thus, the number of agreements between two distinct codewords of this linear code is exactly the number of points on a grid where a low-degree polynomial vanishes with high multiplicity, and thus, is precisely captured by the multiplicity SZ lemma.

[1] This means that all the partial derivatives of P of order at most $s-1$ are zero at this point. See Section 3.2 for a formal definition.

[2] This bound is only interesting when $|T| > d/s$ so that $\frac{d|T|^{m-1}}{s}$ is less than the trivial bound of $|T|^m$.

A natural algorithmic question in the context of error-correcting codes is that of decoding: given a received word that is sufficiently close to a codeword in Hamming distance, can we efficiently find the codeword (or the message corresponding to the closest codeword). For Reed-Solomon codes, this decoding question is equivalent to the question of designing efficient algorithms for recovering a univariate polynomial from its evaluations, with the additional complication that some of the evaluations might be erroneous. For Reed-Muller codes, this is the question of multivariate polynomial interpolation with errors over grids, and for multiplicity codes, this is the question of recovering a multivariate polynomial from its evaluations and evaluations of all its partial derivatives up to some order on a grid, where some of the data is erroneous. If the number of coordinates with errors is at most half the minimum distance of the code, then we know that there is a *unique* codeword close to any such received word, and what the minimum distance actually is implied by the SZ lemma or its variants. In particular, the multiplicity SZ lemma can be restated as the following combinatorial statement about unique decodability of multiplicity codes.

Unique Decoding (Combinatorial Statement). *Let $\mathbb{F}$ be a field, and $T \subseteq \mathbb{F}$, d the degree parameter, m the dimension and s the multiplicity parameter be as above. Given an arbitrary function $f\colon T^m \to \mathbb{F}_{<s}[z_1, \ldots, z_m]$, where $\mathbb{F}_{<s}[z_1, \ldots, z_m]$ denotes the set of all polynomials of degree less than s in the variables $z_1, z_2, \ldots, z_m$, there exists* at most one *polynomial $P \in \mathbb{F}[x_1, \ldots, x_m]$ of degree at most d such that the function* $\mathsf{Enc}^{(s)}(P)\colon T^m \to \mathbb{F}_{<s}[z_1, \ldots, z_m]$ *defined as*

$$\mathsf{Enc}^{(s)}(P)(\mathbf{a}) := P(\mathbf{a}+\mathbf{z}) \mod \langle \mathbf{z} \rangle^s$$

differs from f on less than $\frac{1}{2}\left(1 - \frac{d}{s|T|}\right)$ fraction of points on T^m. Here $\langle \mathbf{z} \rangle^s$ denotes the ideal in $\mathbb{F}[\mathbf{z}]$ generated by monomials of total degree s.

REMARK 1.1. We note that the encoding function $\mathsf{Enc}^{(s)}(P)(\mathbf{a})$ is sometimes also defined as being given by the evaluation of all partial derivatives of P of order at most $s-1$ on input $\mathbf{a}$. These definitions are equivalent as can be seen by looking at the Taylor expansion of $P(\mathbf{a}+\mathbf{z})$ and truncating the series at monomials of degrees less than s in $\mathbf{z}$.

Thus, the unique decoding question for multiplicity codes is equivalent to asking whether there is an algorithmic equivalent of the multiplicity SZ lemma. More precisely, given a function f, can one *efficiently* find the (unique) polynomial P such that $\Delta(f, \mathsf{Enc}^{(s)}(P)) < \frac{1}{2}\left(1 - \frac{d}{s|T|}\right)$ (if one exists).

Given the central role that these polynomial evaluation codes play in coding theory and in complexity theory, it is not surprising to note that this question of efficient unique decoding of Reed-Solomon codes and Reed-Muller codes in particular has been widely investigated. For Reed-Solomon codes, we know multiple algorithms for this task, starting with the result of Peterson from the 50s [21] and the subsequent algorithms of Berlekamp and Massey [2, 18] and

Welch and Berlekamp [25]. However, the situation is a bit more complicated for Reed-Muller and multivariate multiplicity codes.

Till recently, all known efficient decoding algorithms for the multivariate setting (both Reed-Muller and larger order multiplicity codes), worked only when the underlying set T had a nice algebraic structure (e.g., $T = \mathbb{F}$) or when the degree d was very small (cf, the Reed-Muller list-decoding algorithm of Sudan [24] and its multiplicity variant due to Guruswami & Sudan [11]). In particular, even for Reed-Muller codes (equivalently multiplicity codes with multiplicity parameter $s = 1$), where the unique decoding question corresponds to an algorithmic version of the standard SZ lemma, no *efficient* unique decoding algorithm was known. In fact, the problem was open even in the bivariate setting! This seems a bit surprising since the distance of these codes just depends on the non-vanishing of polynomials on arbitrary grids, whereas the decoding algorithms appear to crucially use some very specific algebraic properties of the grid. Even beyond the immediate connection to questions in coding theory, this seems like a very natural algorithmic question in computational algebra that represents a gap in our understanding of a very fundamental property of low-degree polynomials.

For Reed-Muller codes, this question was resolved in a beautiful work of Kim and Kopparty [13] who gave an efficient algorithm for this problem of unique decoding Reed-Muller codes on arbitrary grids. Their algorithm was essentially an algorithmic version of the standard induction-based proof of the SZ lemma. However, the algorithm of Kim and Kopparty does not seem to generalize for the case of higher multiplicity ($s > 1$), and indeed, this problem is mentioned as an open problem of interest in [13].

Since the work of Kim and Kopparty, Bhandari, Harsha, Kumar and Sudan [4] made some partial progress towards this problem for the case of $s > 1$ and $m > 1$. In particular, they designed an efficient decoding algorithm albeit requiring the following conditions to be satisfied:

- the field $\mathbb{F}$ has characteristic zero or is larger than the degree d
- the distance of the code is constant, and
- the multiplicity parameter s is sufficiently large (in terms of the dimension m)

Under these special conditions, they in fact obtained a list-decoding algorithm. Yet, the original algorithmic challenge of obtaining an efficient unique decoding for *all* dimensions and *all* multiplicity parameters remained unresolved. In particular, even unique decoding of bivariate multiplicity codes with multiplicity two from half their minimum distance was not known over arbitrary product sets over any field.

Our main result in this work is a generalization of the result of Kim and Kopparty to *all* constant multiplicities or equivalently, an algorithmic version of the multiplicity SZ lemma (for constant multiplicity parameter). More formally, we prove the following theorem.

Main result. *Let $s, d, m \in \mathbb{N}$ and $T \subseteq \mathbb{F}$ of size n be such that $d < sn$. Then, there is a deterministic algorithm that runs in time $(sn)^{O(s+m)} \cdot \binom{s-1+m}{m}$ and on input $f\colon T^m \to \mathbb{F}_{<s}[\mathbf{x}]$ outputs the unique polynomial $P \in \mathbb{F}_{\leq d}[\mathbf{x}]$ (if such a polynomial exists) such that $\mathsf{Enc}^{(s)}(P)$ differs from f on less than $\frac{1}{2}\left(1 - \frac{d}{s|T|}\right)$ fraction of points on T^m.*

We remark that our algorithm is not polynomial-time in the input size for all settings of the multiplicity parameter s; our algorithm runs in time $(sn)^{O(s+m)} \cdot \binom{s-1+m}{m}$ while the input-size is $n^m \cdot \binom{s-1+m}{m}$. That said, it is polynomial when s is a constant, the typical setting for coding-theoretic applications. While we are chiefly interested in the constant multiplicity parameter regime, we note that our algorithm is polynomial for certain super-constant settings of s also, which might be useful in some applications. We conjecture that it can be made to run in polynomial time for all settings of s.

The main theorem here differs from the results of [4] in the following sense: our main theorem gives a polynomial-time algorithm for unique decoding multiplicity codes for multiplicity parameter s being any arbitrary constant and over any underlying field, whereas [4] give an efficient algorithm for list decoding multiplicity codes up to the list decoding capacity, provided that the multiplicity parameter is sufficiently large (given by the dimension m) and the underlying field has large or zero characteristic.

1.2 Alternative analysis of Forney's GMD Decoding

Forney designed the Generalized Minimum Distance (GMD) decoding to decode concatenated codes from half its minimum distance [7, 8]. A key step in our algorithm (as in Kim and Kopparty's Reed-Muller decoder) is reminiscent of Forney's GMD decoding. Forney analysed the GMD decoding using a convexity argument, which in modern presentations is usually expressed as a probabilistic or a random threshold argument. Kim and Kopparty used a direct adaptation of this random threshold argument for their Reed-Muller decoder. This argument unfortunately fails in our setting. To get around this, we first give an alternative analysis of Forney's GMD decoding using a matching argument, which we explain in detail in the next section. The alternative analysis of Forney's GMD decoding is given in detail in Section 5.

1.3 Further discussion and open problems

We conclude the introduction with some open problems.

Faster algorithms. Observe that the running time of the main result as stated above depends polynomially on $(sn)^{s+m}$. Since the input to the decoding problem has size $\left(n^m \cdot \binom{s-1+m}{m}\right)$, strictly speaking, the running time of the algorithm in the main theorem is not polynomially bounded in the input size for all choices of the parameters n, s, m. It would be extremely interesting to obtain an algorithm for this problem which runs in polynomial time in the input size for all regimes of parameters.

List-decoding. Another natural question in the context of the main result in this paper is to obtain efficient algorithms for decoding multiplicity codes on arbitrary grids when the amount of error exceeds half the minimum distance of the code. The results in [4] provide such algorithms when s is sufficiently large, m is a constant and we are over fields of sufficiently large (or zero) characteristic. However, when s is small, for instance, even when $s = 1$ and $m = 2$, we do not have any such list decoding results. In fact, for small s, we do not even have a good understanding of the combinatorial limits of list decoding of these (Reed-Muller/Multiplicity) codes on arbitrary grids.

Polynomial-method-based algorithms. It would also be interesting to understand if the results in this paper and those in [13] can be proved via a more direct application of the polynomial method. By this we mean a decoding algorithm along the lines of the classical Welch-Berlekamp [25] decoding algorithm as presented by Gemmell and Sudan [9] or the generalizations used for list-decoding: for instance, as in the work of Guruswami and Sudan [11]. Note that for large s, constant m, and fields of sufficiently large or zero characteristic, the algorithm in [4] is indeed directly based on a clean application of the polynomial method. It would be aesthetically appealing to have an analogous algorithm for the case of small s (and perhaps over all fields).

1.4 Organization

The rest of the paper is organized as follows.

We begin with an overview of our algorithm in Section 2. In Section 3, we discuss the necessary preliminaries including the definition and properties of multiplicity codes and describe a fine-grained notion of distance for multiplicity codes that plays a crucial role in our proofs. We build up the necessary machinery for the analysis of the bivariate decoder in Section 4 and Section 6 where we describe and analyse efficient decoding algorithms for univariate multiplicity codes with varying multiplicities and weighted univariate multiplicity codes respectively. In Section 7, we put it all together and describe our algorithm for decoding bivariate multiplicity codes, and give its proof of correctness. Using a suitable induction on the number of variables, we generalize the algorithm for the bivariate case to the multivariate case in Section 8. An alternative analysis of Forney's generalized minimum distance decoder, which inspires our analysis of our algorithm, is discussed in Section 5.

2. Overview of algorithm

As one would expect, our multivariate multiplicity code decoder for $s > 1$ is a generalization of the Kim-Kopparty decoder for Reed-Muller codes (i.e., the $s = 1$ case). However, a straightforward generalization does not seem to work and several subtle issues arise. Some of these issues are definitional and conceptual; for instance, one needs to work with a finer notion of distance while others are more technical, for instance the need for an alternative analysis of Forney's GMD decoding.

To explain these issues and discuss how we circumvent them, we first recall the Kim-Kopparty decoder and then mention how we generalize it to the $s > 1$ setting, highlighting the various issues that arise along the way.

2.1 An overview of the Kim-Kopparty Reed-Muller decoder

We begin by recalling the Kim-Kopparty decoder for Reed-Muller codes on arbitrary grids. Their algorithm is, in some sense, an "algorithmization" of the standard inductive proof of the classical Schwartz-Zippel lemma (without any multiplicities). For simplicity, we will focus on the bivariate case ($m = 2$) initially.

The setting is as follows: we are given a received word $f\colon T^2 \longrightarrow \mathbb{F}$ and have to find the unique polynomial $P(x, y) \in \mathbb{F}[x, y]$ (if one exists) such that $\Delta(f, P) < \frac{1}{2}n^2(1 - \frac{d}{n})$, where Δ is the standard Hamming distance and $n = |T|$.

The high-level approach is to write P as $\sum_{i=0}^{d} P_i(x)y^{d-i}$ and iteratively recover the univariate polynomials $P_0, P_1, \ldots, P_d$ in this order. The goal of the ℓ^{th} iteration is to recover $P_\ell(x)$ assuming we have correctly recovered $P_0, P_1, \ldots, P_{\ell-1}$ in the previous iterations. For the ℓ^{th} iteration, we work with $f_\ell\colon T^2 \longrightarrow \mathbb{F}$ given by $f_\ell(a, b) := f(a, b) - \sum_{i=0}^{\ell-1} P_i(a) \cdot b^{d-i}$ as the received word. Clearly, the Hamming distance between f_ℓ and the evaluation of the bivariate polynomial $Q_\ell(x, y) = \sum_{i=\ell}^{d} P_i(x)y^{d-i}$ on the grid is exactly $\Delta(f, P)$ which is at most $\frac{1}{2}n^2(1 - \frac{d}{n})$.

Very naturally, we view f_ℓ as values written on a two-dimensional grid $T \times T$, with the columns indexed by $x = a \in T$ and the rows indexed by $y = b \in T$, and each entry on a grid point of $T \times T$ is an element of $\mathbb{F}$. As a first step of the algorithm, for every $a \in T$, a Reed-Solomon decoder is used to find a univariate polynomial $G^{(a)}(y)$ of degree at most $d - \ell$ such that the tuple of evaluations of this polynomial on the set T is close to the restriction of f_ℓ on the column $x = a$. For columns where Q_ℓ and f_ℓ have a high agreement, $G^{(a)}(y)$ would be equal to $Q_\ell(a, y)$ and hence, the leading coefficient of $G^{(a)}(y)$ must equal $P_\ell(a)$. A natural suggestion here would be to collect the leading coefficients of the polynomials $G^{(a)}(y)$'s, namely $g(a) := \mathrm{Coeff}_{y^{d-\ell}}(G^{(a)}(y))$ and try and run another Reed-Solomon decoder on $g\colon T \longrightarrow \mathbb{F}$ to get $P_\ell(x)$. Unfortunately, with $\Delta(f, P) < \frac{1}{2}n^2(1 - \frac{d}{n})$, it can be shown that a lot of these polynomials $G^{(a)}$ could be incorrect and hence, the number of errors in the received word for this second step Reed-Solomon decoding is too large for it to correctly output P_ℓ.

The key idea of Kim and Kopparty is to observe that at the end of the first step, not only do we have the decoded column polynomials $G^{(a)}(y)$ but we also have hitherto unused information about the function f_ℓ, namely, the Hamming distance between the received word (the restriction of f_ℓ) on a column $x = a$ and the evaluation of $G^{(a)}(y)$. Formally, for every $a \in T$, they associate a weight

$$w(a) := \min\left\{\Delta\left(f_\ell(a,\cdot), G^{(a)}\right), \frac{n-(d-\ell)}{2}\right\} . \tag{1}$$

Note that if $\Delta\left(f_\ell(a,\cdot), G^{(a)}\right) > \frac{n-(d-\ell)}{2}$, then there is no guarantee that $G^{(a)}(y)$ is the closest codeword to $f_\ell(a,y)$ and hence the minimum is taken in the above expression to cap the distance to $\frac{n-(d-\ell)}{2}$ (this plays a technical reason in the proof, but let us ignore it for now). Based on these weights, they define a modified distance between the pair $(g,w)\colon T \to \mathbb{F}\times\mathbb{Z}_{\geq 0}$, which they refer to as a "fractional word", and a regular word $h\colon T \to \mathbb{F}$ as follows:

$$\Delta((g,w),h) := \sum_{a\in A_0(g,h)} (n-(d-\ell)-w(a)) + \sum_{a\in A_1(g,h)} w(a) , \tag{2}$$

where $T = A_0(g,h) \cup A_1(g,h)$ is the following partition of T

$$A_1(g,h) := \{a \in T\colon g(a) = h(a)\} , \qquad \text{(agreement points)}$$
$$A_0(g,h) := \{a \in T\colon g(a) \neq h(a)\} . \qquad \text{(disagreement points)}$$

We note that the above is a scaled version of the modified distance used by Kim and Kopparty. We find it convenient to express it in the above equivalent form for the purpose of this presentation as this alternate form generalizes to the $s > 1$ setting more naturally.

They then prove that this modified distance satisfies the following two important properties.

- The fractional word (g,w) and the "correct" polynomial P_ℓ satisfy

$$\Delta((g,w),P_\ell) \leq \Delta(f,P) < \frac{1}{2}n^2\left(1-\frac{d}{n}\right) . \tag{3}$$

- For any two distinct polynomials Q_ℓ and R_ℓ of degree ℓ, we have

$$\begin{aligned}\Delta((g,w),Q_\ell) + \Delta((g,w),R_\ell) &\geq (n-(d-\ell))\cdot\Delta(Q_\ell,R_\ell)\\ &\geq (n-(d-\ell))\cdot(n-\ell)\\ &\geq n^2\left(1-\frac{d}{n}\right) .\end{aligned} \tag{4}$$

These two properties imply that the pair (g,w) uniquely determines the polynomial P_ℓ since P_ℓ satisfies $\Delta((g,w),P_\ell) < \frac{1}{2}n^2\left(1-\frac{d}{n}\right)$ (by the first property) while every other polynomial Q_ℓ satisfies $\Delta((g,w),Q_\ell) \geq n^2\left(1-\frac{d}{n}\right) - \Delta((g,w),P_\ell) > \frac{1}{2}n^2\left(1-\frac{d}{n}\right)$ (by the second property).

We are still left with the problem of *efficiently* determining P_ℓ from the pair (g,w). We will defer that discussion to later, but first show how the above steps can be generalized to the $s > 1$ setting.

2.2 Generalizing the Kim-Kopparty decoder to $s > 1$

To begin with, we will need to work with a more fine-grained notion of distance which incorporates the multiplicity information. For two functions $f, g \colon T^m \to \mathbb{F}_{<s}[\mathbf{z}]$ and a point $\mathbf{a} \in T^m$, the Hamming distance $\Delta(f(\mathbf{a}), g(\mathbf{a}))$ measures if the polynomials $f(\mathbf{a})$ and $g(\mathbf{a})$ are different. We will consider a refined notion of $\Delta^{(s)}_{\mathsf{mult}}(f(\mathbf{a}), g(\mathbf{a}))$, which measures the "multiplicity distance" at the point $\mathbf{a}$ (here s is the multiplicity parameter). Formally,

$$\Delta^{(s)}_{\mathsf{mult}}(f(\mathbf{a}), g(\mathbf{a})) := (s - d^{(s)}_{\mathrm{min}}(f(\mathbf{a}) - g(\mathbf{a}))) \ ,$$

where $d^{(s)}_{\mathrm{min}}(R)$ is defined as follows for any polynomial R: $d^{(s)}_{\mathrm{min}}(R)$ is the minimum of s and the degree of the minimum degree monomial with a non-zero coefficient in R. Note that if R is identically zero, $d^{(s)}_{\mathrm{min}}(R)$ is set to s. Recall that here $f(\mathbf{a})$ and $g(\mathbf{a})$ are both polynomials in $\mathbf{z}$ of degree strictly less than s. The multiplicity distance between the functions f and g is defined to be the sum of the above quantity over all $\mathbf{a} \in T^m$. More precisely,

$$\Delta^{(s)}_{\mathsf{mult}}(f, g) := \sum_{\mathbf{a} \in T^m} \Delta^{(s)}_{\mathsf{mult}}(f(\mathbf{a}), g(\mathbf{a})) \ .$$

Observe that for $s = 1$, this matches with the usual notion of Hamming distance. Furthermore, this fine-grained distance lower-bounds the Hamming distance as follows:

$$\Delta^{(s)}_{\mathsf{mult}}(f, g) \leq s \cdot \Delta(f, g) \ .$$

The multiplicity SZ lemma, in terms of this fine-grained multiplicity distance, states that for two distinct m-variate degree d polynomials P, Q, we have

$$\Delta^{(s)}_{\mathsf{mult}}(\mathsf{Enc}^{(s)}(P), \mathsf{Enc}^{(s)}(Q)) \geq n^m \cdot \left(s - \frac{d}{n}\right) .$$

The Kim-Kopparty decoder for Reed-Muller codes on a grid required as a basic primitive a Reed-Solomon decoder. Correspondingly, our algorithm will require as a basic primitive a univariate multiplicity code decoder. Such a decoder was first obtained by Nielsen [19]. Later on in the algorithm, we will actually need a generalization of this decoder which can handle the case when the multiplicity parameters at different evaluation points are not necessarily the same. Such a decoder can be obtained by a suitable modification of the standard Welch-Berlekamp decoder for Reed-Solomon codes. This modification is presented in Section 4 (Algorithm 1 specifically).

We are now ready to generalize the Kim-Kopparty decoder to the $s > 1$ setting. To keep things simple, let us focus on the $s = 2$ and $m = 2$ setting. The problem stated in terms of the multiplicity-distance is as follows. We are given a received word $f \colon T^2 \to \mathbb{F}_{<2}[u, v]$ and have to find the unique polynomial $P(x, y) \in \mathbb{F}[x, y]$ (if one exists) such that $\Delta^{(2)}_{\mathsf{mult}}(f, \mathsf{Enc}^{(2)}(P)) < \frac{1}{2}n^2(2 - \frac{d}{n})$, where $n = |T|$.

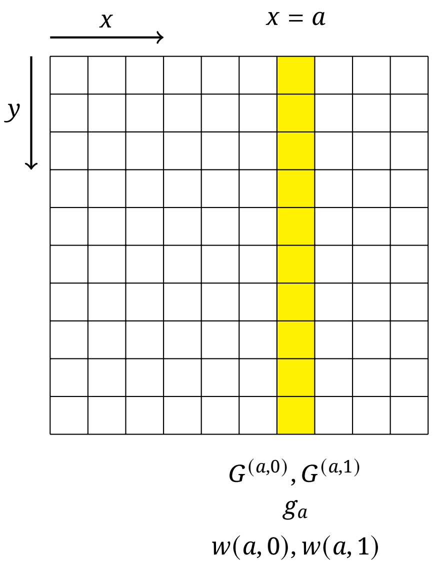


Figure 1. Bivariate algorithm, visualised

As before we write $P = \sum_{i=0}^{d} P_i(x) y^{d-i}$ and recover the polynomials $P_0, \ldots, P_d$ in successive iterations. Let us assume we are in the ℓ^{th} iteration where we have to recover P_ℓ and have already correctly recovered $P_0, \ldots, P_{\ell-1}$ in the previous iterations. Similar to the $s = 1$ setting, we consider the function $f_\ell \colon T^2 \to \mathbb{F}_{<2}[u,v]$ given by $f_\ell(a,b) := f(a,b) - \mathsf{Enc}^{(2)}\left(\sum_{i=0}^{\ell-1} P_i(x) \cdot y^{d-i}\right)$ as the received word. Clearly, the distance between f_ℓ and $\mathsf{Enc}^{(2)}\left(\sum_{i=\ell}^{d} P_i(x) y^{d-i}\right)$ is exactly $\Delta^{(2)}_{\mathsf{mult}}(f, \mathsf{Enc}^{(2)}(P))$ which is at most $\frac{1}{2}n^2(2 - \frac{d}{n})$.

As before, we view f_ℓ as values written on a two-dimensional grid $T \times T$, with the columns indexed by $x = a \in T$ and the rows indexed by $y = b \in T$, and each entry on a grid point of $T \times T$ is an element of $\mathbb{F}_{<2}[u,v]$. It will be convenient to view this degree-one polynomial $f_\ell(a,b)$ at the grid point (a,b) as $f_\ell^{(0)}(a,b)(v) + u \cdot f_\ell^{(1)}(a,b)$ where $f_\ell^{(0)}(a,b) \in \mathbb{F}_{<2}[v]$ and $f_\ell^{(1)}(a,b) \in F_{<1}[v] \equiv \mathbb{F}$. For every $a \in T$, we look at the restriction of the functions $f_\ell^{(0)}$ and $f_\ell^{(1)}$ to the column $x = a$. We could use a Reed-Solomon decoder to find the univariate polynomial $G^{(a,1)}(y)$ of degree at most $d - \ell$ such that the tuple of evaluations of this polynomial on the set T is close to the restriction of $f_\ell^{(1)}$ on the column $x = a$. Similarly, we could use the univariate multiplicity code decoder (mentioned above) to find the univariate polynomial $G^{(a,0)}(y)$ of degree at most $d - \ell$ such that encoding of this polynomial is close to the restriction of $f_\ell^{(0)}$ on the column $x = a$. We can for each a, define $g(a) \in \mathbb{F}_{<2}[u]$ to be the polynomial $g(a) := \mathsf{Coeff}_{y^{d-\ell}}(G^{(a,0)}) + u \cdot \mathsf{Coeff}_{y^{d-\ell}}(G^{(a,1)})$.

We also have the corresponding weight functions (similar to (1)) which measure how close the encodings of the polynomials $G^{(a,0)}(y)$ and $G^{(a,1)}(y)$ are close to the corresponding functions $f_\ell^{(0)}$ and $f_\ell^{(1)}$ respectively. Namely,

$$w(a,0) := \min\left\{\Delta^{(2)}_{\mathsf{mult}}\left(f_\ell^{(0)}(a,\cdot), \mathsf{Enc}^{(2)}(G^{(a,0)})\right), \frac{2n - (d-\ell)}{2}\right\} ,$$

$$w(a,1) := \min\left\{\Delta^{(1)}_{\mathsf{mult}}\left(f^{(1)}_{\ell}(a,\cdot), \mathsf{Enc}^{(1)}(G^{(a,1)})\right), \frac{n-(d-\ell)}{2}\right\} .$$

The quantities $2n-(d-\ell)$ and $n-(d-\ell)$ are the distances of univariate multiplicity codes of degree $(d-\ell)$ and multiplicity parameters 2 and 1 respectively. We will refer to these pairs of weight functions as $\mathbf{w} = (w(\cdot,0), w(\cdot,1))$. As in the $s=1$ setting, we will view the pair $(g,\mathbf{w})$ as a "fractional word" and would like to define the distance between the fractional word $(g,\mathbf{w})$ and h. In the $s=1$ Reed-Muller setting ((2)), the set T was partitioned into two sets: the set $A_1(g,h)$ of agreement points and the set $A_0(g,h)$ of disagreement points. However, for the $s=2$ case, agreement/disagreement come in more flavours. We have points in T in which g and h agree in both the evaluation and derivative, points in which they agree only in the evaluation but not the derivative and finally points where they disagree even on the evaluation. Based on this, we have the following partition of the set $T = A_0(g,h) \cup A_1(g,h) \cup A_2(g,h)$:

$$A_2(g,h) := \{a \in T \colon g(a) \equiv h(a)\} , \qquad \text{(agreement points with multiplicity 2)}$$

$$A_1(g,h) := \{a \in T \colon g(a) \equiv h(a) \mod \langle u\rangle\} \setminus A_2(g,h) , \qquad \text{(agreement points with multiplicity 1)}$$

$$A_0(g,h) := T \setminus (A_2(g,h) \cup A_1(g,h)) . \qquad \text{(disagreement points)}$$

Inspired by the distance definition of Kim and Kopparty between a fraction word and a regular word (see (2)), we define a similar modified distance between the fractional word $(g,\mathbf{w})\colon T \to \mathbb{F}_{<2}[u] \times (\mathbb{Z}_{\geq 0} \times \mathbb{Z}_{\geq 0})$ and and a word $h\colon T \to \mathbb{F}_{<2}[u]$ as follows.

$$\Gamma\left((g,\mathbf{w}),h\right) := \sum_{a\in A_0(g,h)} (2n-(d-\ell)-w(a,0)) + \sum_{a\in A_1(g,h)} \max\left\{(n-(d-\ell)-w(a,1)), w(a,0)\right\} + \sum_{a\in A_2(g,h)} \max\left\{w(a,0), w(a,1)\right\} , \tag{5}$$

(See Definition 6.1 for the exact definition of this modified distance.) This definition might seem complicated, however it is chosen so that it satisfies the following two important properties.

- [Lemma 7.3] The fractional word $(g,\mathbf{w})$ and the "correct" polynomial P_ℓ satisfy

$$\Gamma((g,\mathbf{w}),P_\ell) \leq \Delta^{(2)}_{\mathsf{mult}}(f,\mathsf{Enc}^{(2)}(P)) < \frac{1}{2}n^2\left(2-\frac{d}{n}\right) . \tag{6}$$

- [Lemma 6.2] For any two distinct polynomials Q_ℓ and R_ℓ of degree ℓ, we have

$$\Gamma((g,\mathbf{w}),Q_\ell) + \Gamma((g,\mathbf{w}),R_\ell) \geq n^2\left(2-\frac{d}{n}\right) . \tag{7}$$

These two properties imply that the pair $(g,\mathbf{w})$ uniquely determines the polynomial P_ℓ, which is in fact an alternative statement of the multiplicity SZ lemma.

As in the case $s = 1$, we are now left with the problem of efficiently extracting the polynomial P_ℓ from the fractional word $(g, \mathbf{w})$. In the next section, we describe how this was done by Kim and Kopparty and our modification of the same.

2.3 Extracting P_ℓ from the fractional word

Let us restate the problem of P_ℓ extraction: we are given a fractional word $(g, w) \colon T \to \mathbb{F} \times \mathbb{Z}_{\geq 0}$ that satisfies (3), namely $\Delta((g, w), P_\ell) \leq \frac{1}{2} n^2 \left(1 - \frac{d}{n}\right)$ (note that by (4), this uniquely defines P_ℓ) and we have to find the polynomial P_ℓ. Kim and Kopparty designed the weighted Reed-Solomon decoder, inspired by Forney's generalized minimum distance (GMD) decoder for this purpose. To see this connection to Forney's GMD decoding, we first rewrite (3) as follows:

$$\frac{1}{2} \cdot (n - (d - \ell)) \cdot \left(\sum_{a \in A_1} \widetilde{w}(a) + \sum_{a \in A_0} (2 - \widetilde{w}(a)) \right) \leq \frac{1}{2} n^2 \left(1 - \frac{d}{n}\right) ,$$

where $\widetilde{w}(a) := w(a) / \left(\frac{1}{2} \cdot (n - (d - \ell))\right) \in [0, 1]$. Or equivalently,

$$\sum_{a \in A_1} \widetilde{w}(a) + \sum_{a \in A_0} (2 - \widetilde{w}(a)) \leq \frac{n^2 \left(1 - \frac{d}{n}\right)}{(n - (d - \ell))} \leq n \left(1 - \frac{\ell}{n}\right) .$$

This can be further rewritten as

$$\sum_{a \in T} \widetilde{w}(a) + \sum_{a \in A_0} 2(1 - \widetilde{w}(a)) \leq n \left(1 - \frac{\ell}{n}\right) . \tag{8}$$

The above inequality is very reminiscent of Forney's analysis of GMD decoding (c.f., [10, Section 12.3]). Consider a uniformly random threshold $\theta \in [0, 1]$ and erase all coordinates $a \in T$ such that $\widetilde{w}(a) \geq \theta$. (8) can be now rewritten as

$$\mathbb{E}_\theta \left[\# \text{ erasures}(\theta) + 2 \cdot \# \text{ errors}(\theta)\right] \leq n \left(1 - \frac{\ell}{n}\right) . \tag{9}$$

This implies that there exists a threshold $\theta \in [0, 1]$ that satisfies $\# \text{ erasures}(\theta) + 2 \cdot \# \text{ errors}(\theta) \leq n \left(1 - \frac{\ell}{n}\right)$. This is precisely the condition under which Reed-Solomon decoding (under erasures and errors) is feasible. Thus, the weighted Reed-Solomon decoder of Kim and Kopparty is as follows: consider every possible threshold $\theta \in [0, 1]$ and the corresponding erased word $g|_{\{a \in T \colon \widetilde{w}(a) < \theta\}}$. Observe that there are at most n distinct thresholds to consider. We can now run a standard Reed-Solomon decoder on this partially erased word to extract the polynomial P_ℓ.

This completes the description of the Kim-Kopparty decoder for Reed-Muller codes on grids.

We would now like to adapt this weighted Reed-Solomon decoder to the multiplicity setting. We are now given the fractional word $(g, \mathbf{w}) \colon T \to \mathbb{F}_{<2}[u] \times (\mathbb{Z}_{\geq 0} \times \mathbb{Z}_{\geq 0})$ with the promise of (6),

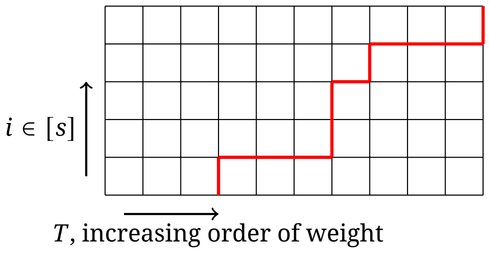


Figure 2. Step-threshold in weighted univariate multiplicity decoding

namely $\Gamma((g, \mathbf{w}), P_\ell) \le \frac{1}{2}n^2(2 - \frac{d}{n})$ and we have to find the polynomial P_ℓ. Recall that (7) states that the promise uniquely determines P_ℓ.

A natural "weighted univariate multiplicity code decoder" would be the following: consider all possible step-thresholds $\boldsymbol{\theta} = (\theta_0, \theta_1)$ (or for general s, $(\theta_0, \theta_1, \ldots, \theta_{s-1})$) and erase coordinates of g accordingly to obtain a univariate multiplicity received word of varying multiplicities[3]. If this step-threshold $\boldsymbol{\theta}$ is indeed correct (i.e., it retains a large fraction of agreements with P_ℓ), we can show that we can adapt the classical Welch-Berlekamp decoder for Reed-Solomon codes to decode even in this setting. And we do precisely this in Section 4. We are still left with the problem of showing that there exists a step-threshold $\boldsymbol{\theta}$ that works.

How did Kim and Kopparty show that such a threshold exists in the $s = 1$ setting? They showed by scaling the promise (3) by $\frac{1}{2}(n - (d - \ell))$ to obtain (8) which is equivalent to the statement (9) that a random threshold θ works. Observe that the scaling $\frac{1}{2}(n-(d-\ell))$ is precisely the unique-decoding radius of the $(d - \ell)$ degree Reed-Solomon code $G^{(a)}(y)$. In our setting $(s > 1)$, there is not one code $G^{(a)}(y)$, but a whole family of them ($G^{(a,0)}(y)$ and $G^{(a,1)}(y)$ for the $s = 2$ case). Recall that these are members of multiplicity codes of degree $d - \ell$ with multiplicity parameter 2 and 1 respectively. Their corresponding unique-decoding radii are $\frac{1}{2}(2n - (d - \ell))$ and $\frac{1}{2}(n - (d - \ell))$ (in terms of the multiplicity distance). In general, we might have s different unique-decoding radii. It is unclear which of these scalings we should use.

There is a more fundamental reason why one does not expect such a random-threshold argument to hold. Consider the following instance of the problem. Let P be a degree d polynomial for $2n/3 < d < 2n - 2$. Let $B \subseteq T$ be a set of points of size $\left\lceil \frac{n}{2}\left(1 - \frac{d}{2n}\right)\right\rceil - 1$. Consider a word $f \colon T^2 \to \mathbb{F}_{<2}[u, v]$ formed by corrupting $\mathsf{Enc}^{(2)}(P)$ on $T \times B$. The fraction of errors is $|B|/n < \frac{1}{2}\left(1 - \frac{d}{2n}\right)$ and hence we must be able to uniquely decode P from f. Let us see how our suggested algorithm proceeds when it decodes along each column $x = a$. For each $a \in T$, the function $f_\ell^{(0)}(a, y)$ is corrupted on less than $\frac{1}{2}\left(1 - \frac{d}{2n}\right)$ and hence the univariate multiplicity code decoder for multiplicity parameter 2 correctly decodes this column. Furthermore, $w(a, 0) = |B|$. However, while decoding $f_\ell^{(1)}(a, y)$ we observe that the number of errors is $|B|$ while the distance of the underlying code (multiplicity code with multiplicity parameter 1) is only $n\left(1 - \frac{d}{n}\right)$ which

[3] We are intentionally vague at this point. For more details, see Section 6.

is less than $|B|$ for our choice of parameters (since $d > 2n/3$). So we could have corrupted enough points so that for each $a \in T$, $f_\ell^{(1)}(a, y)$ is a valid codeword and yet different from the correct codeword in $\mathsf{Enc}^{(2)}(P)$. In this case, all the $G^{(a,1)}(y)$ polynomials are erroneous, however $w(a, 1) = 0$. We thus have for this setting $w(a, 1) = 0$ and $w(a, 0) = |B|$ for all $a \in T$. Consider the step-threshold $(|B| + 1, 0)$ (by this we mean that we use the threshold $|B| + 1$ for $f_\ell^{(0)}$ and 0 for $f_\ell^{(1)}$). Then the $f_\ell^{(1)}$ level is completely erased, while the $f_\ell^{(0)}$ level is completely un-erased. This object would be a received word for univariate multiplicity code decoding with varying multiplicities, in general; in this case since the first-derivative level is completely erased, it is an instance of Reed-Solomon decoding. Then, in fact, it is correctly decoded to obtain P_ℓ, since each column $x = a$ was correctly decoded at the $f_\ell^{(0)}$ level, the only available level. Yet it is easy to check that the expected number of erasures plus two times the number of errors is too large for any reasonable choice of a random threshold.

How do we then show a step-threshold $\boldsymbol{\theta}$ works without resorting to the random-threshold argument?

For starters, we could ask if we could give an alternative analysis of Forney's GMD decoding without appealing to the random-threshold argument. We show that this is indeed true. More formally, we show that if every threshold fails for Forney's GMD decoding, then the promise for GMD decoding is violated (i.e., the received word is more than half the distance of the concatenated code away from a codeword). We prove this as follows. Assume every threshold fails. We then construct a matching between agreement points and disagreement points (that is, correctly and incorrectly decoded blocks, following the inner decoding) such that every matched agreement-disagreement pair has the property that the weight in the disagreeing coordinate is at least that in the agreeing coordinate. Such a matching then immediately implies that the promise for GMD decoding is violated. See Section 5 for further details. This alternative analysis of Forney's GMD decoding is not needed for our algorithm. It only serves as an inspiration for our "weighted univariate multiplicity code decoder" just as the original analysis inspired the "weighted Reed-Solomon decoder" of Kim and Kopparty.

Equipped with this alternative analysis of Forney's GMD decoding, we show a similar result for our setting. For contradiction, let us assume that every step-threshold $\boldsymbol{\theta}$ fails to decode the polynomial P_ℓ. This lets us construct a matching between "correct" and "incorrect" coordinate-multiplicity pairs. The construction of this matching is considerably more involved than the construction of the corresponding matching in Forney's GMD decoding as we need to respect the monotonicity of the multiplicity coordinates. The matching is constructed via a delicate analysis using a generalization of Hall's Theorem (see Theorem 6.10). As in the Forney analysis, the matched agreement-disagreement pairs have the property that the multiplicity-distance of the disagreeing member is at least that of the agreeing member. Once we have constructed such a matching, it is not hard to show that the promise (6) is violated. Hence, at least one of the step-thresholds work and our "weighted univariate multiplicity code decoder" works as suggested. This part of the analysis happens to be the most technically-challenging part of the paper and indeed this “weighted univariate multiplicity code decoder” forms the work-horse of our bivariate decoder (for more details, see Section 6). The factor of n^s in the running time comes because this step runs over all possible step-thresholds and there are $O(n^s)$ of them. We conjecture that this step can be improved to $\mathrm{poly}(n, s)$.

2.4 The multivariate setting for $m > 2$

The above presentation gave an outline of both the Kim-Kopparty decoder as well as ours in the bivariate setting. Kim and Kopparty extended the bivariate Reed-Muller decoder to larger m by designing a weighted version of the Reed-Muller decoder which they then used inductively as suggested by the proof of the classical SZ Lemma. We however do not have a similar “weighted multivariate multiplicity code decoder”. In particular, we do not even have a weighted version of the bivariate decoder. We get around this issue by performing a slightly different proof of the SZ Lemma from the textbook proof (see Section 8.1). This alternative proof of the SZ Lemma proceeds by viewing the polynomial as an $(m - 1)$-variate polynomial with the coefficients coming from a univariate polynomial ring $\mathbb{F}[x_m]$ instead of as a univariate polynomial in x_m with the coefficients coming from the $(m-1)$-variate polynomial ring $\mathbb{F}[x_1, \ldots, x_{m-1}]$. This slight modification allows us to work with just a “weighted univariate multiplicity code decoder”. We note that a similar construction could have been obtained in the Reed-Muller setting too. We discuss this multivariate generalization in more detail in Section 8.

3. Multiplicity codes and main result

In this section, we give a formal statement of our main result. We start by building up the requisite notation and preliminaries including the notion of multiplicity codes and their properties which provide the key underlying motivation for the results in this paper.

3.1 Notation

- We use boldface letters like $\mathbf{x}, \mathbf{y}, \mathbf{z}$ for m-tuples of variables.
- $\mathbb{F}$ denotes the underlying field.
- For a positive integer ℓ, $[\ell] = \{0, \ldots, \ell - 1\}$ (note $\ell \notin [\ell]$).
- For $\ell \in \mathbb{N}$, $\mathbb{F}_{\leq \ell}[\mathbf{x}]$ (and $\mathbb{F}_{<\ell}[\mathbf{x}]$) denotes the set of polynomials in $\mathbf{x}$, with coefficients in $\mathbb{F}$ of degree at most (strictly less than) ℓ.
- For $\mathbf{e} \in \mathbb{Z}^m_{\geq 0}$, $\mathbf{x}^{\mathbf{e}}$ denotes the monomial $\prod_{j \in [m]} x_j^{e_j}$ and $|\mathbf{e}|_1 := \sum e_i$.

3.2 Multiplicity code

We now state the definition of multiplicity codes.

DEFINITION 3.1 (Multiplicity code). Let $s, m \in \mathbb{N}$, $d \in \mathbb{Z}_{\geq 0}$, $\mathbb{F}$ be a field and $T \subseteq \mathbb{F}$ be a non-empty finite subset of $\mathbb{F}$. The m-variate order-s multiplicity code for degree-d polynomials over $\mathbb{F}$ on the grid T^m is defined as follows.

Let $E := \{\mathbf{e} \in \mathbb{Z}_{\geq 0}^m \mid 0 \leq |\mathbf{e}|_1 < s\}$. Note that $|E| = \binom{s+m-1}{m}$, which is the number of distinct monomials in $\mathbb{F}_{<s}[\mathbf{z}]$. Hence, we can identify $\mathbb{F}^E$ with $\mathbb{F}_{<s}[\mathbf{z}]$.

The code is defined over the alphabet $\mathbb{F}^E$ and has block length $|T|^m$ with the coordinates being indexed by elements of the grid T^m. The code is an $\mathbb{F}$-linear map from the space of polynomials $\mathbb{F}_{\leq d}[\mathbf{x}]$ to $(\mathbb{F}_{<s}[\mathbf{z}])^{|T|^m}$, where for any $\mathbf{a} \in T^m$, the $\mathbf{a}^{th}$ coordinate of the codeword denoted by $\mathsf{Enc}_{T^m}^{(s)}(P)|_{\mathbf{a}}$ is given by

$$\mathsf{Enc}_{T^m}^{(s)}(P)|_{\mathbf{a}} := P(\mathbf{a}+\mathbf{z}) \mod \langle \mathbf{z} \rangle^s.$$

Thus, a codeword of this code can be naturally viewed as a function from T^m to $\mathbb{F}_{<s}[\mathbf{z}]$.

REMARK 3.2. For the ease of notation, we sometimes drop one or both of s and T^m from $\mathsf{Enc}_{T^m}^{(s)}(P)$ when they are clear from the context. Also, we will use n to refer to the size of T, i.e., $n = |T|$.

Univariate multiplicity codes were first studied by Rosenbloom & Tsfasman [22] and Nielsen [19]. Multiplicity codes for general m and s were introduced by Kopparty, Saraf and Yekhanin [15] where they constructed locally decodable codes with high rate and small query complexity. In the context of local decoding, these codes are typically studied with the set T being the entire field $\mathbb{F}$ (or a subfield of $\mathbb{F}$). However, in this work, we encourage the reader to think of T being an arbitrary subset of $\mathbb{F}$.

REMARK 3.3. We note that for every $\mathbf{a} \in \mathbb{F}^m$, $(P(\mathbf{a}+\mathbf{z}) \mod \langle \mathbf{z} \rangle^s)$ is a polynomial in $\mathbb{F}[\mathbf{z}]$ of degree at most $s-1$, and therefore can be viewed as a function from the set E to $\mathbb{F}$ given by its coefficient vector. Thus, the alphabet of the code is indeed $\mathbb{F}^E$. Moreover, the coefficients of $(P(\mathbf{a}+\mathbf{z}) \mod \langle \mathbf{z} \rangle^s)$ are precisely the evaluation of the Hasse derivatives of order at most $(s-1)$ of P at the input $\mathbf{a}$. Recall that for a polynomial $P \in \mathbb{F}[\mathbf{x}]$, the $\mathbf{e}^{th}$ Hasse derivative of P, denoted by $\frac{\bar{\partial} P}{\bar{\partial} \mathbf{x}^{\mathbf{e}}}$ is the coefficient of $\mathbf{z}^{\mathbf{e}}$ in $P(\mathbf{x}+\mathbf{z})$. Typically, multiplicity codes are defined directly via Hasse derivatives, however, in this paper, it is notationally more convenient to work with Definition 3.1. Observe that with this notation, it is natural to think of a received word (an input to the decoding algorithm for multiplicity codes) as being given as a function $f\colon T^m \to \mathbb{F}_{<s}[\mathbf{z}]$.

The distance of multiplicity codes is guaranteed by a higher order generalization of the Schwartz-Zippel Lemma, that was proved by Dvir, Kopparty, Saraf and Sudan [6]. We need the following notation for the statement of the lemma.

DEFINITION 3.4 (multiplicity at a point). For an m-variate polynomial $P \in \mathbb{F}[\mathbf{x}]$ and a point $\mathbf{a} \in \mathbb{F}^m$, the multiplicity of P at $\mathbf{a}$, denoted by $\mathrm{mult}(P, \mathbf{a})$ is the largest integer ℓ such that $P(\mathbf{a}+\mathbf{z}) = 0 \mod \langle \mathbf{z} \rangle^{\ell}$, or equivalently, the Hasse derivative $\frac{\bar{\partial} P}{\bar{\partial} \mathbf{x}^{\mathbf{e}}}(\mathbf{a})$ is zero for all monomials $\mathbf{x}^{\mathbf{e}}$ of degree at most $\ell - 1$.

We now state the multiplicity Schwartz-Zippel lemma.

LEMMA 3.5 (multiplicity Schwartz-Zippel lemma [6, Lemma 8]). *Let $P \in \mathbb{F}[\mathbf{x}]$ be a nonzero m-variate polynomial of total degree at most d and let $T \subseteq \mathbb{F}$. Then,*

$$\sum_{\mathbf{a} \in T^m} \mathrm{mult}(P, \mathbf{a}) \le d\,|T|^{m-1}.$$

We note that the lemma immediately implies that the multiplicity codes as defined in Definition 3.1 have distance at least $n^m(1 - \frac{d}{ns})$.

3.3 A fine-grained notion of distance for multiplicity codewords

The multiplicity Schwartz-Zippel lemma states that for two distinct polynomials P and Q of total degree d,

$$\sum_{\mathbf{a} \in T^m} (s - \mathrm{mult}(P - Q, \mathbf{a})) > n^m \left(s - \frac{d}{n}\right).$$

Thus, $\sum_{\mathbf{a} \in T^m} (s - \mathrm{mult}(P - Q, \mathbf{a}))$ is a candidate measure of distance between the encodings of P and Q. However, it may be the case that $\mathrm{mult}(P - Q, \mathbf{a})$ exceeds s, making the quantity negative. To get around this, we “cap” $\mathrm{mult}(P - Q, \mathbf{a})$ at s. That is, for each $\mathbf{a} \in T^m$, we take $s - \min\{\mathrm{mult}(P - Q, \mathbf{a}), s\}$.

We certainly have

$$\sum_{\mathbf{a} \in T^m} (s - \min\{\mathrm{mult}(P - Q, \mathbf{a}), s\}) \ge \sum_{\mathbf{a} \in T^m} (s - \mathrm{mult}(P - Q, \mathbf{a})) > n^m \left(s - \frac{d}{n}\right)$$

We then naturally extend this measure of distance to a notion of distance between functions $f, g \colon T^m \to \mathbb{F}_{<s}[\mathbf{z}]$ that might not necessarily be valid codewords of a multiplicity code.

DEFINITION 3.6. Let $T \subseteq \mathbb{F}$ be any set. For functions $f, g \colon T^m \to \mathbb{F}_{<s}[\mathbf{z}]$, we define $\Delta^{(s)}_{\mathrm{mult}}(f, g)$ as

$$\Delta^{(s)}_{\mathrm{mult}}(f, g) := \sum_{\mathbf{a} \in T^m} (s - d^{(s)}_{\min}(f(\mathbf{a}) - g(\mathbf{a}))),$$

where for a polynomial R, $d^{(s)}_{\min}(R)$ is defined to be the minimum of s and the degree of the minimum degree monomial with a non-zero coefficient in R. Note that if R is identically zero, then $d^{(s)}(R)$ is set to s.

As indicated above, the quantity $d^{(s)}_{\text{min}}$ is related to the notion of multiplicity in the following sense:

$$d^{(s)}_{\text{min}}(\mathsf{Enc}^{(s)}_{T^m}(P)(\mathbf{a}) - \mathsf{Enc}^{(s)}_{T^m}(Q)(\mathbf{a})) = \min\{\text{mult}(P - Q, \mathbf{a}), s\}\,.$$

For brevity, we abuse notation slightly and sometimes drop one or more of the parameters r, s, T^m from $\Delta^{(s)}_{\text{mult}}(\mathsf{Enc}^{(r)}_{T^m}(P), \mathsf{Enc}^{(r)}_{T^m}(Q))$ when they are clear from the context. Note that the parameter r in the encoding and s in the Δ_{mult} in $\Delta^{(s)}_{\text{mult}}(\mathsf{Enc}^{(r)}_{T^m}(P), \mathsf{Enc}^{(r)}_{T^m}(Q))$ might be different from each other.

REMARK 3.7. In Definition 3.6, s does not depend on $\mathbf{a}$ and is constant throughout. However, in the course of our analysis we will also encounter a scenario (see Algorithm 1) where $m = 1$ and the multiplicity parameter is not constant and is given by a function $\mathbf{s}\colon T \to \mathbb{Z}_{\geq 0}$. The Definition 3.6 naturally extends to this case as follows.

$$\Delta^{(\mathbf{s})}_{\text{mult},T}(f, g) = \sum_{\mathbf{a}\in T}(\mathbf{s}(\mathbf{a}) - d^{(s)}_{\text{min}}(f(\mathbf{a}) - g(\mathbf{a})))\,.$$

The following observation relates $\Delta^{(s)}_{\text{mult}}$ with the standard definition of Hamming distance for multiplicity codes of order s.

OBSERVATION 3.8. *For any two polynomials $P, Q \in \mathbb{F}[\mathbf{x}]$ of degree at most d,*

$$\Delta^{(s)}_{\text{mult}}(\mathsf{Enc}^{(s)}_{T^m}(P), \mathsf{Enc}^{(s)}_{T^m}(Q)) \leq s \cdot \Delta(\mathsf{Enc}^{(s)}_{T^m}(P), \mathsf{Enc}^{(s)}_{T^m}(Q))\,.$$

Intuitively, if $\mathsf{Enc}^{(s)}_{T^m}(P)$ differs from $\mathsf{Enc}^{(s)}_{T^m}(Q)$ at a point $\mathbf{a} \in T^m$, then the standard Hamming distance $\Delta(\mathsf{Enc}^{(s)}_{T^m}(P), \mathsf{Enc}^{(s)}_{T^m}(Q))$ counts this point with weight one, whereas the new notion of distance $\Delta^{(s)}_{\text{mult}}(\mathsf{Enc}^{(s)}_{T^m}(P), \mathsf{Enc}^{(s)}_{T^m}(Q))$ also takes into account the lowest degree monomial in $\mathbf{z}^{\mathbf{e}}$ such that the coefficients of $\mathbf{z}^{\mathbf{e}}$ in $\mathsf{Enc}^{(s)}_{T^m}(P)$ and $\mathsf{Enc}^{(s)}_{T^m}(Q)$ are not equal to each other. This fine-grained structure will be crucially used in the analysis of our algorithm.

3.4 Main Result

With these definitions in place, our main technical result can be stated as follows.

THEOREM 3.9. *Let $s, d, m \in \mathbb{N}$ and $T \subseteq \mathbb{F}$ be such that $d < s|T|$. Then, there is a deterministic algorithm (Algorithm 4) that runs in time $(sn)^{3m+s+O(1)} \cdot \binom{m+s-1}{s}$ and on input $f\colon T^m \to \mathbb{F}_{<s}[\mathbf{z}]$ outputs the unique polynomial $P \in \mathbb{F}_{\leq d}[\mathbf{x}]$ (if such a polynomial exists) such that*

$$\Delta^{(s)}_{\text{mult}}(f, \mathsf{Enc}^{(s)}_{T^m}(P)) < \frac{n^m}{2}\left(s - \frac{d}{n}\right),$$

where $n = |T|$.

Since this is stated in terms of the fine-grained distance, this is actually a strengthening of the main result as stated in the introduction.

4. Decoding univariate multiplicity codes with varying multiplicities

In this section, we discuss an algorithm for decoding univariate multiplicity codes up to half their minimum distance based on the standard Welch-Berlekamp decoder[4] for Reed-Solomon codes. While such decoders for univariate multiplicity codes are well known, the decoder discussed here handles a slightly more general scenario than off-the-shelf decoders of this kind, namely that the multiplicity parameter at each evaluation point is not necessarily the same. This slight generalization is necessary for our applications in this paper.

4.1 Description of the generalized univariate multiplicity decoder

We start with a description of the algorithm.

```
Input:  $T \subseteq \mathbb{F}$                                        ▷ set of evaluation points
        $e$                                                         ▷ degree of the univariate polynomial
        $\mathbf{s}\colon T \longrightarrow \mathbb{Z}_{\geq 0}$     ▷ number of multiplicities
        $h\colon T \longrightarrow \mathbb{F}[z]$ such that for all $a \in T$, $h_a = \sum_{i \in [\mathbf{s}(a)]} h_a^{(i)} z^i$     ▷ received word
Output: $R \in \mathbb{F}_{\leq e}[x]$ such that $\Delta_{\mathrm{mult}}^{(\mathbf{s})}(h, \mathrm{Enc}^{(\mathbf{s})}(R)) < \frac{1}{2}(\sum_a \mathbf{s}(a) - e)$, if such an $R$
        exists and $\perp$ otherwise
Set $N \longleftarrow \sum_{a \in T} \mathbf{s}(a)$;
Set $D \longleftarrow \lfloor \frac{1}{2}(N + e) \rfloor + 1$ ;
Find a non-zero polynomial $Q(x, y) = B_0(x) + y \cdot B_1(x)$ such that
    • $\deg(B_0) < D$,
    • $\deg(B_1) < D - e$, and
    • $\forall a \in T$, $Q(a + z, h_a) \equiv 0 \mod z^{\mathbf{s}(a)}$ ;
if the following three conditions are satisfied
    • $-B_0/B_1$ is a polynomial,
    • $-B_0/B_1$ has degree $\leq e$, and
    • $\Delta_{\mathrm{mult}}^{(\mathbf{s})}(h, \mathrm{Enc}^{(\mathbf{s})}(-B_0/B_1)) < \frac{1}{2}(\sum_a \mathbf{s}(a) - e)$
 then return $-B_0/B_1$ else return $\perp$.
```

Algorithm 1. Generalized Univariate Multiplicity Decoder

[4] By the Welch-Berlekamp decoder we refer to the description of the Welch-Berlekamp algorithm [25] provided by Gemmell and Sudan [9].

We remark while the overall structure of the algorithm is essentially that of the Welch-Berlekamp based algorithms, the main point of difference is that the number of linear constraints imposed in the interpolation step (Line 3) at any $a \in T$ is $\mathbf{s}(a)$, and might be different for different $a \in T$.

4.2 Correctness and running time of Algorithm 1

We first show that the interpolation step (Line 3) is possible, that is, a non-zero polynomial $Q(x, y)$ satisfying the constraints exists. Then, we argue that if there is a polynomial R of degree at most e such that $\Delta^{(\mathbf{s})}_{\mathsf{mult}}(h, \mathsf{Enc}^{(\mathbf{s})}(R)) < \frac{1}{2}(\sum_a \mathbf{s}(a) - e)$, then R is indeed output by the algorithm. Moreover, from the check in step 4, it is clear that the algorithm never outputs a polynomial that is *far* from the received word. Thus, together, these claims imply the correctness of the algorithm.

We also note that the algorithm is efficient and runs in polynomial time in its input size, since all it needs is to solve a linear system of not-too-large size and a call to an off-the-shelf polynomial division algorithm, both of which can be done efficiently. We now proceed with the proof of correctness.

CLAIM 4.1. *For* $D = \lfloor \frac{1}{2}(N + e) \rfloor + 1$, *there exists a non-zero polynomial* $Q(x, y) = B_0(x) + y \cdot B_1(x)$ *with* $\deg(B_0) < D$, $\deg(B_1) < D - e$ *and* $\forall a \in T$, $Q(a + z, h_a) \equiv 0 \mod z^{\mathbf{s}(a)}$, *where* $N = \sum_a \mathbf{s}(a)$. *Moreover, for any such non-zero solution,* $B_1(x)$ *is non-zero.*

PROOF. As is standard with decoding algorithms for various algebraic codes that are based on the polynomial method, we think of the constraints as a system of homogeneous linear equations in the coefficients of the unknown polynomials B_0 and B_1.

The number of variables in the linear system equals the number of coefficients we need to find across B_0 and B_1, which is $D + (D - e) = 2D - e$. For every $a \in T$, the constraint

$$Q(a + z, h_a) \equiv 0 \mod z^{\mathbf{s}(a)}$$

is really $\mathbf{s}(a)$ many homogeneous linear constraints on the coefficients of Q, with there being one constraint corresponding to the coefficient of z^i in $Q(a + z, h_a)$ being 0 for every $i \in [\mathbf{s}(a)]$. Thus, the total number of homogeneous linear constraints is $\sum_{a \in T} \mathbf{s}(a) = N$. Hence, setting $2D - e > N$, e.g., $D > \frac{1}{2}(N + e)$ ensures the existence of a non-zero solution.

For the *moreover* part, observe that if B_1 is identically zero, then the system of homogeneous linear constraints imposed on Q imply that B_0 vanishes with multiplicity at least $\mathbf{s}(a)$ for every $a \in T$. Thus, $\sum_{a \in T} \mathrm{mult}(B_0, a) = \sum_{a \in T} \mathbf{s}(a) = N > D \geq \deg(B_0)$. But this implies that B_0 must be identically zero, and hence Q must be identically zero, which contradicts the non-zeroness of the solution. ■

We now observe that any non-zero solution of the linear system, as is guaranteed by claim 4.1 becomes identically zero when y is substituted by the *correct* message polynomial R.

CLAIM 4.2. *If $R \in \mathbb{F}_{\leq e}[x]$ is such that $\Delta^{(\mathbf{s})}_{\mathrm{mult}}(h, \mathrm{Enc}^{(\mathbf{s})}(R)) < \frac{1}{2}(N-e)$ and Q is a non-zero polynomial satisfying the set of constraints in the algorithm, then $Q(x,R) \equiv 0$.*

PROOF. The linear constraints imposed in the interpolation step imply that for every $a \in T$,

$$Q(a+z, h_a) \equiv 0 \mod z^{\mathbf{s}(a)}.$$

Now, if $(h_a - \mathrm{Enc}^{(\mathbf{s})}(R)(a)) = 0 \mod z^{u_a}$, i.e. $\mathrm{Enc}^{(\mathbf{s})}(R)$ and h agree with multiplicity at least u_a at $a \in T$, then

$$Q(a+z, R(a+z)) \equiv 0 \mod z^{\min\{u_a, \mathbf{s}(a)\}}.$$

Now, from the definition of Δ_{mult} and from the hypothesis of the claim, we know that

$$\Delta^{(\mathbf{s})}_{\mathrm{mult}}(h, \mathrm{Enc}^{(\mathbf{s})}(R)) = \sum_{a \in T} (\mathbf{s}(a) - \min\{u_a, \mathbf{s}(a)\}) < \frac{1}{2}(N-e)\,.$$

If $Q(x, R(x))$ is a non-zero polynomial, then it is a non-zero polynomial of degree strictly less than D, by construction of Q. We will now show the sum of multiplicities of zeroes at all points is at least D which implies $Q(x, R(x))$ is identically zero.

$$\begin{aligned}\sum_{a \in T} \mathrm{mult}(Q(x,R(x)), a) &\geq \sum_{a \in T} \min\{u_a, \mathbf{s}(a)\} \\ &> \sum_{a \in T} s(a) - \frac{1}{2}(N-e) \\ &= N - \frac{1}{2}(N-e) \\ &= \frac{1}{2}(N+e)\end{aligned}$$

Note that $\sum_{a \in T} \mathrm{mult}(Q(x,R(x)), a)$ is an integer. Then we must have

$$\sum_{a \in T} \mathrm{mult}(Q(x,R(x)), a) > \lfloor \frac{1}{2}(N+e) \rfloor + 1 = D$$ ■

We are now ready to complete the proof of correctness of the algorithm.

THEOREM 4.3. *If $R \in \mathbb{F}_{\leq e}[x]$ is such that $\Delta^{(\mathbf{s})}_{\mathrm{mult}}(h, \mathrm{Enc}^{(\mathbf{s})}(R)) < \frac{1}{2}(N-e)$, then R is correctly output by the algorithm and if there is no such close enough R, then the algorithm outputs $\perp$. Moreover, the algorithm runs in time $N^{O(1)}$.*

PROOF. From claim 4.1, we know that the linear system solver in the algorithm (Line 3) always finds a non-zero polynomial $B_0(x) + yB_1(x)$ satisfying the linear constraints imposed in the algorithm, regardless of the existence of a codeword that is close enough to the received word. Moreover, in any such non-zero solution, B_1 is a non-zero polynomial.

We also know from claim 4.2 that any $R \in \mathbb{F}_{\leq e}[x]$ such that $\Delta^{(\mathbf{s})}_{\mathsf{mult}}(h, \mathsf{Enc}^{(\mathbf{s})}(R)) < 1/2(N-e)$ satisfies

$$B_0(x) + B_1(x) \cdot R(x) = 0\,,$$

or, in other words $R = -B_0/B_1$, and is correctly output by the algorithm. Since the algorithm performs a sanity check in the last line to make sure that $-B_0/B_1$ is indeed a low-degree polynomial such that the corresponding codeword is close to h and outputs the computed solution only if this check is passed; else it outputs $\perp$. Thus, it never outputs an incorrect solution.

Algorithm 1 really only needs to set up and solve a linear system of size $O(N)$ and perform some basic univariate polynomial arithmetic involving polynomials of degree at most $D < N$. Thus, it runs in time $N^{O(1)}$. ■

5. Forney's generalized minimum distance decoding

In this section, we give an alternative analysis of Forney's algorithm for decoding a concatenated code from half its minimum distance, assuming that there is an efficient algorithm for decoding the outer code from errors (scaled by a factor of 2) and erasures up to its minimum distance, and that the inner code has an efficient maximum likelihood decoder. An example of such a setting is when the outer code is the Reed-Solomon code and the inner code has block length logarithmic in the total block length, and is over an alphabet of constant size. In this case, the outer code can be efficiently decoded from errors and erasures using, for instance, the Welch-Berlekamp algorithm, as long as

$$2(\#\text{Errors}) + (\#\text{Erasures}) < (\text{Minimum distance})\,.$$

For the inner code, one can just do a brute-force iteration over all codewords, and find the closest one. We recommend the reader to think of this example going forward even though the discussion here applies to a general concatenated code.

5.1 Concatenated codes

We start with a definition of concatenated codes.

DEFINITION 5.1 (concatenated code). Let $q \geq 2, k \geq 1$ be natural numbers and let $Q = q^k$. Let $C_{out}\colon [Q]^K \to [Q]^N$ be a code of minimum distance D and let $C_{in}\colon [q]^k \to [q]^n$ be a code of minimum distance d. The concatenated code $C = C_{out} \circ C_{in}\colon [Q]^K \to [q]^{Nn}$ is defined as follows: Given a message $m \in [Q]^K$, we first apply C_{out} to m to get a codeword $m' \in [Q]^N$. Since $Q = q^k$, we identify each symbol of m' with a vector of length k over $[q]$, or equivalently, an element in the message space of C_{in}. We now encode each coordinate of m' using C_{in} to get a vector in $[q]^{Nn}$. See Figure 3.

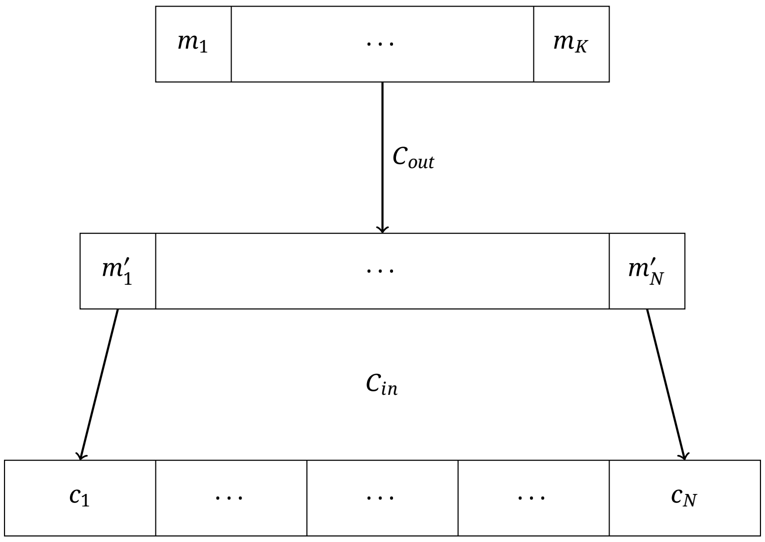


Figure 3. Encoding of a concatenated code

As defined above, the concatenated code C has minimum distance at least Dd. Forney designed an algorithm [7, 8] to uniquely decode C when the number of erroneous coordinates is less than $Dd/2$. Next, we briefly describe Forney's algorithm before discussing a slightly different analysis for it.

Let f be the received word (obtained by making less than $Dd/2$ errors), and let c be the (unique) codeword of the concatenated code C such that $\Delta(f, c) < Dd/2$ where Δ refers to the usual Hamming distance.

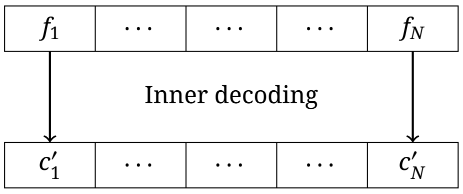


Figure 4. Inner decoding of a concatenated code

Notice that the received word f consists of N blocks of length n each over $[q]$, which we denote by $f_1, \ldots, f_N$. Similarly, let $c_1, c_2, \ldots, c_N$ be the corresponding blocks in the nearest codeword c. The decoding algorithm starts by taking each of the blocks f_i and using the maximum likelihood decoder for C_{in} to find the codeword c'_i of C_{in} that is closest to f_i. This process is shown in Figure 4. For some of the coordinates $i \in [N]$, c'_i equals c_i and for others $c'_i \neq c_i$. For each $i \in [N]$, let m'_i be such that $C_{in}(m'_i) = c'_i$. The natural next step would be to use a decoder for the outer code C_{out} with the input $m' = (m'_1, m'_2, \ldots, m'_N)$ and hope to show that the output equals c as is intended. When the number of errors in f (i.e. $\Delta(f, c)$) is less than $Dd/4$ then this algorithm can indeed be shown to work and output the correct message m corresponding to the codeword c.

In [7, 8], Forney built upon this simple and natural decoder to design an efficient algorithm with error tolerance improved from $Dd/4$ to $Dd/2$. The key idea was to assign a weight

$$w(i) := \min\left\{\frac{\Delta(f_i, c_i')}{d/2}, 1\right\}$$

to each block i; the intuition being that the weight $w(i)$ is an indicator of the number of errors in the block i. Thus, the higher the weight for a coordinate i, the lower is the confidence in c_i' being c_i.

Forney showed that if $\Delta(f, c) < Dd/2$, then there is a threshold $\theta \in \mathbb{Z}_{\geq 0}$ such that the vector $m'' = (m_1'', m_2'', \ldots, m_N'') \in ([Q] \cup \{\perp\})^N$ where m_i'' equals m_i' if $w(i) \leq \theta$ and is an erasure ($\perp$) otherwise has the property that when compared to the correct codeword c, the sum of the number of erased blocks and twice the number or erroneous blocks is less than D. Thus, given an error-erasure decoder for C_{out} from half the minimum distance, we can recover the correct message m.[5]

The key technical task in the correctness of above algorithm is to prove the existence of such a threshold θ given that $\Delta(f, c) < Dd/2$. Forney proves this using a convexity argument which shows that if all the thresholds in $\{w(i) : i \in [N]\}$ fail, then the number of errors in f must be at least $Dd/2$. This convexity argument has a randomized interpretation, which in turn shows that a random threshold succeeds (see the manuscript by Guruswami, Rudra and Sudan [10, Section 12.3] for this randomized interpretation).

In the next section we give an alternative proof of existence of a *good* threshold, thereby proving the correctness of Forney's algorithm. This alternative proof serves as the inspiration for our proof of Theorem 3.9, and as far as we understand appears to be different from the above-mentioned proofs.

5.2 An alternative analysis of Forney's GMD decoding

We now give an alternative proof of the existence of a good threshold.

THEOREM 5.2 (Forney). *Let f, c, c', w, D, d be as defined above. If $\Delta(f, c) < Dd/2$, then there exists a $\theta \in [0, 1]$ such that*

$$2\left|\left\{i \in [N] : w(i) \leq \theta, c_i' \neq c_i\right\}\right| + |\{i \in [N] : w(i) > \theta\}| < D\,.$$

[5] We remark that just showing the existence of a good threshold θ is sufficient, since without loss of generality, θ can be taken to be equal to one of the weights in the set $\{w(i) : i \in [N]\}$. Thus we can efficiently try all these N values and check if the resulting message gives a codeword close enough to the received word f.

We recall that the condition in the conclusion of the theorem is sufficient for the outer decoding to be done by our assumption on the outer code C_{out}. For instance, if the outer code is the Reed-Solomon code, then the Welch-Berlekamp decoder can be used for this.

Notice that all weights $w(i)$ lie between 0 and 1 and, as mentioned earlier, can be thought of as representing the uncertainty in that block. Let A and B be defined as follows:

$$A := \left\{i \in [N] \,:\, c_i = c'_i\right\}, \qquad B := \left\{i \in [N] \,:\, c_i \neq c'_i\right\}.$$

That is, A is the set of blocks where the inner decoding is correct, and B is those where the decoding is incorrect. Further, for a threshold θ, we define $A_\theta = A \cap \{i \in [N] \,:\, w(i) \leq \theta\}$. B_θ is defined similarly. Notice that the number of erased blocks for a given θ is $N - |A_\theta| - |B_\theta|$.

We say that a threshold θ is bad if

$$2\left|\left\{i \in [N] \colon w(i) \leq \theta, c'_i \neq c_i\right\}\right| + |\{i \in [N] \colon w(i) > \theta\}| \geq D\,.$$

Or, in the new notation, $2|B_\theta| + (N - (|A_\theta| + |B_\theta|)) \geq D$, or equivalently,

$$|B_\theta| \geq |A_\theta| - (N - D)\,.$$

We note the above in the following observation.

OBSERVATION 5.3. *For a fixed $\theta \in [0,1]$, if the threshold θ is bad, then $|B_\theta| \geq |A_\theta| - (N - D)$.*

PROOF OF THEOREM 5.2. Suppose for contradiction that all thresholds $\theta \in [0,1]$ are bad. By observation 5.3, we have $|B_\theta| \geq |A_\theta| - (N - D)$ for every $\theta \in [0,1]$.

First, we observe that the size of A, i.e., the number of correctly decoded blocks, must be more than $(N - D)$. Otherwise, since $|A| + |B| = N$, the number of errors $|B|$ is at least D. Further, for every $i \in B$, the i^{th} block in f must have had at least $d/2$ errors for the maximum likelihood decoder for C_{in} to have output a $c'_i \neq c_i$. But this means that $\Delta(f, c)$ is at least $Dd/2$ which is a contradiction. Therefore, we can write $|A| = N - D + u$ for some positive integer u.

Now, write the indices in A in increasing order of their weights, i.e., according to w, to get a sequence $i_1, i_2, \ldots, i_{N-D+u}$. We do the same with B to get a sequence $j_1, j_2, \ldots, j_{D-u}$. If many blocks have the same weight, we place them in some arbitrary order.

We first claim that $u \leq D - u$ and for each $k \in \{1, 2, \ldots, u\}$, $w(j_k) \leq w(i_{N-D+k})$. In other words, if the indices $i \in \{1, 2, \ldots, N\}$ are written left to right in increasing order of weights $w(i)$, then for every $k \in \{1, 2, \ldots, u\}$, the index j_k is to the left of the index i_{N-D+k} as indicated in the picture below.

To see this, let us assume that there is a $k \in \{1, 2, \ldots, u\}$, such that either $w(j_k) > w(i_{N-D+k})$ or j_k does not exist. Consider a threshold θ which picks up everything including and to the left of i_{N-D+k}. Then, we have $|A_\theta| \geq N - D + k$ and $|B_\theta| \leq k - 1$. This contradicts our assumption that $|B_\theta| \geq |A_\theta| - (N - D)$ for every θ.

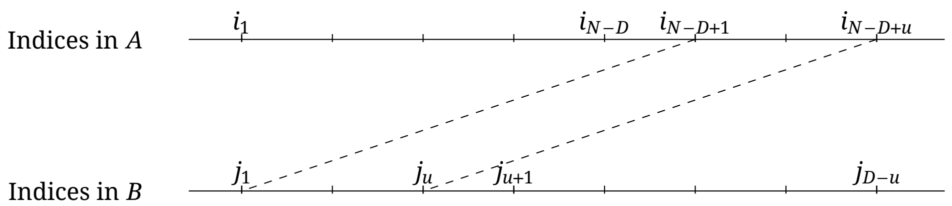


Figure 5. Pairing of indices in the concatenated code

Using the above claim, we can “pair” each of the last u indices in A with the first u of B (ordered by weight). When we pair i_{N-D+k} with j_k, we have $w(j_k) \leq w(i_{N-D+k})$. For the incorrectly decoded index j_k, the distance of the block f_{j_k} from c_{j_k} is at least $\frac{d}{2}(2 - w(j_k))$. This follows from the triangle inequality of the Hamming distance: $\Delta(f_{j_k}, c_{j_k}) + \Delta(f_{j_k}, c'_{j_k}) \geq \Delta(c_{j_k}, c'_{j_k}) \geq d$.

For a correct index i_{N-D+k}, the distance between f and c on this block is by definition $\frac{d}{2}w(i_{N-D+k})$. Thus, the total distance between f and c on the pair (j_k, i_{N-D+k}) of blocks together is at least

$$\frac{d}{2}(2 - w(j_k)) + \frac{d}{2}w(i_{N-D+k}) \geq d,$$

where the inequality follows from the fact that $w(j_k) \leq w(i_{N-D+k})$. We now have u pairs contributing a distance of at least d each. In addition, there are still $D - 2u$ incorrectly decoded blocks with indices in B, each contributing at least $d/2$ to $\Delta(f, c)$. Thus, we have

$$\Delta(f, c) \geq du + (D - 2u)\frac{d}{2} = \frac{Dd}{2},$$

which is a contradiction. ■

As mentioned earlier, Forney’s original proof [7, 8] uses a convexity argument, while more recent presentations of this argument reinterprets this as a randomized argument (erasing the block i with probability $w(i)$ for each i) to show that there is a good threshold θ (see [10, Section 12.3]). For the analysis of our bivariate decoder, we adapt the *pairing* argument used in the proof of Theorem 5.2 to analyse the weighted univariate multiplicity code decoder in Section 6. It is unclear to us if the previous convexity-based proofs of Theorem 5.2 can be adapted for our application.

6. Weighted univariate multiplicity code decoder

In this section, we describe an algorithm (Algorithm 2) for decoding weighted univariate multiplicity codes, i.e. given a received word $g\colon T \to \mathbb{F}_{<r}[z]$, and a weight function $w\colon T \times [r] \to \mathbb{Z}_{\geq 0}$ (indicating the uncertainty in each coordinate), this algorithm finds a low-degree polynomial R such that the encoding of R as a univariate multiplicity code of order r evaluated on T is *close enough* to the received word h.

The notion of *close enough* here is not defined in terms of either the Hamming distance or the multiplicity distance, but a weighted notion of distance parameterized by the weight function w. This notion of distance, denoted by Γ, was previously introduced in Equation (5). It serves as a proxy for distance in our analysis, turns out to be crucial in the overall analysis of Algorithm 2 and will be discussed in detail in Section 6.1.

More precisely, this decoder gets as input a function $g \colon T \to \mathbb{F}_{<r}[z]$ and a set of weights $w \colon T \times [r] \to \mathbb{Z}_{\geq 0}$. Recall that we referred to this pair (g, w) as a "fractional word" in the algorithm overview. It is also given as input a degree parameter ℓ and multiplicity parameter r besides the global degree parameter d, the global dimension m and global multiplicity parameter s. We have two sets of degree and multiplicity parameters (the local and global) as the intermediate algorithms will be running decoders on degree and multiplicity parameters different from the global ones, but they do need to know the global parameters. The weighted decoder then returns a polynomial $R \in \mathbb{F}_{\leq \ell}[x]$ such that $\Gamma_w^{d,\ell,s}(g, R) < \frac{n^2}{2}\left(s - \frac{d}{n}\right)$.

Algorithm 2 will be used as a subroutine by the final decoder (i.e., Algorithm 3 in the bivariate $m = 2$ case or Algorithm 4 in the general multivariate case). Ideally, Algorithm 2 should be oblivious of whether it is being used by the bivariate decoder or the multivariate decoder. Unfortunately, this is not the case for our Algorithm 2 and we need to feed it as input several global parameters m, s, and d[6]. In this section we will work with the global dimension m being 2 for the sake of analysis, though the algorithm is stated in terms of general m. The general m setting will be discussed in Section 8.

6.1 A notion of weighted distance and its properties

We start with the following definition of a distance measure between the received word g and the encoding of a polynomial R, using the weight w. The case for $s = 2$ was discussed in Section 2 (see Equation (5)).

DEFINITION 6.1. Let $d, \ell, s \in \mathbb{N}$ be parameters with $d \geq \ell$ and let $T \subseteq \mathbb{F}$ be a subset of size n. Let $r := s - \lfloor \frac{d-\ell}{n} \rfloor$. Let $R \in \mathbb{F}[x]$ be a univariate polynomial of degree at most ℓ, $g \colon T \to \mathbb{F}_{<r}[z]$ and $w \colon T \times [r] \to \mathbb{Z}_{\geq 0}$ be functions such that for every $(a, i) \in T \times [r]$ we have $w(a, i) \leq \frac{n}{2} \cdot \left((s - i) - \frac{d-\ell}{n}\right)$.

Then, $\Gamma_w^{s,d,\ell}(g, R)$ is defined as follows.

$$\Gamma_w^{s,d,\ell}(g, R) := \left(\sum_{i=0}^{r-1} \sum_{a \in A_i(g,R)} \max\left\{\left(n\left((s-i) - \frac{d-\ell}{n}\right) - w(a,i)\right), \max_{j<i} w(a,j)\right\}\right) + \sum_{a \in A_r(g,R)} \max_{j<r} w(a,j)$$

[6] We believe this dependence on global parameters should be removed, but do not know how to do so.

where for every $i \in [r+1]$

$$A_i(g,R) = \left\{a \in T : \ \max\left\{j \in [r+1] : \ g(a) = R(a+z) \mod \langle z \rangle^j\right\} = i\right\}.$$

Observe that for $i \in [r+1]$, the set A_i collects those locations in T where g and R agree up to derivatives of order exactly $i-1$ and no further. For $i=0$, A_0 is the set of locations where they disagree at the 0^{th} derivative itself, i.e. the evaluation level, and the $\max_{j<i} w(a,j)$ term can be thought of as $-\infty$.

Now, we prove some properties of $\Gamma_w^{s,d,\ell}$. These properties will turn out to be useful in proving the correctness of Algorithm 2.

Below we prove a triangle-like inequality for Γ which we will use to show uniqueness of the output of Algorithm 2. This is the point alluded to in Equation (7).

LEMMA 6.2 (Triangle-like inequality for Γ). *Let $d, \ell, r, s \in \mathbb{N}$ be parameters with $\ell \le d < n$, $r = s - \lfloor \frac{d-\ell}{n} \rfloor$, and let $T \subseteq \mathbb{F}$ be a subset of size n. Let $Q, R \in \mathbb{F}[x]$ be univariate polynomials of degree at most ℓ, and $g \colon T \longrightarrow \mathbb{F}_{<r}[z]$ and $w \colon T \times [r] \longrightarrow \mathbb{Z}_{\ge 0}$ be functions such that for every $(a,i) \in T \times [r]$, $w(a,i) \le \frac{n}{2} \cdot \left((s-i) - \frac{d-\ell}{n}\right)$.*

If $Q \ne R$, then

$$\Gamma_w^{s,d,\ell}(g,Q) + \Gamma_w^{s,d,\ell}(g,R) \ge n^2\left(s - \frac{d}{n}\right).$$

PROOF. For $i \in [r+1]$, let $A_i(Q,R)$ be the set of points $a \in T$ such that

$$A_i(Q,R) = \left\{a \in T : \ \max\left\{j \in [r+1] : \ Q(a+z) = R(a+z) \mod \langle z \rangle^j\right\} = i\right\}.$$

Further, for each $i \in [r+1]$, let $\tau_i = |A_i(Q,R)|$. Since the sets $A_i(Q,R)$ for each i are all disjoint, we have $\sum_{i=0}^r \tau_i = |T| = n$.

In addition, since Q and R are distinct polynomials of degree at most ℓ, using the multiplicity SZ lemma we have:

$$\Delta_{\mathsf{mult}}^{(r)}(\mathsf{Enc}_T^{(r)}(Q), \mathsf{Enc}_T^{(r)}(R)) \ge rn - \ell.$$

Using the definition of $\Delta_{\mathsf{mult}}^{(r)}$, this can be re-written as

$$\sum_{i=0}^{r-1} \tau_i \cdot (r-i) \ge rn - \ell.$$

That is,

$$r\sum_{i=0}^{r} \tau_i - \sum_{i=0}^{r} i \cdot \tau_i \ge rn - \ell.$$

Rearranging and using $\sum_{i=0}^r \tau_i = n$, we obtain

$$\sum_{i=0}^{r} i \cdot \tau_i \le \ell.$$

Now, consider an $a \in A_i(Q,R)$ for some $i \in [r]$. (The case of $a \in A_r(Q,R)$ will be explained shortly.) Hence, $Q(a+z) \neq R(a+z) \mod \langle z\rangle^{i+1}$. Therefore, both $g(a) = Q(a+z) \mod \langle z\rangle^{i+1}$ and $g(a) = R(a+z) \mod \langle z\rangle^{i+1}$ can't simultaneously hold. In other words, there is a $j \leq i$ such that $a \in A_j(g,R)$ or $a \in A_j(g,Q)$.

Without loss of generality, we assume $a \in A_j(g,Q)$ with $j \leq i$. In addition, $a \in A_k(g,R)$ for some k. Again we can assume $j \leq k$ (otherwise, swap the roles of Q and R).

As $a \in A_j(g,Q)$, we claim that the contribution of the term corresponding to a to $\Gamma_w^{s,d,\ell}(g,Q)$, which we denote by $\Gamma_w^{s,d,\ell}(g,Q)_a$ is at least $((s-j)n-(d-\ell)-w(a,j))$, because that is one of the terms in the maximum, in the definition of $\Gamma_w^{s,d,\ell}$.

As for $\Gamma_w^{s,d,\ell}(g,R)_a$, since a is in $A_k(g,Q)$ with $k \geq j$, two cases can happen. If $j \neq k$, then the contribution of a is at least $w(a,j)$, since that is one of the terms in the maximum. Else, if $k = j$, then the contribution is at least $(s-j)n-(d-\ell)-w(a,j)$, which, by the condition on $w(a,j)$ in Definition 6.1 is at least $w(a,j)$.

Then, the sum of the distances at a is at least

$$\begin{aligned}\Gamma_w^{s,d,\ell}(g,Q)_a + \Gamma_w^{s,d,\ell}(g,R)_a &\geq (s-j)n-(d-\ell)-w(a,j)+w(a,j)\\ &= (s-j)n-(d-\ell)\\ &\geq (s-i)n-(d-\ell)\,.\end{aligned}$$

since $j \leq i$.

If $a \in A_r(Q,R)$, we cannot get an expression of this form, since its contribution to Γ will be $\max_{j<r} w(a,j)$. Since the contribution from this set is nonnegative, and we are trying to get a lower bound, it is sufficient to consider the contributions from $A_i(Q,R)$'s for $i < r$.

By the above arguments we arrive at the following.

$$\begin{aligned}\Gamma_w^{s,d,\ell}(g,Q) + \Gamma_w^{s,d,\ell}(g,R) &\geq \sum_{i=0}^{r-1} \tau_i((s-i)n-(d-\ell))\\ &= \sum_{i=0}^{r-1} \tau_i(sn-(d-\ell)) - n\sum_{i=0}^{r-1} i\tau_i\\ &\geq (sn-(d-\ell))(n-\tau_r) - n(\ell - r\tau_r)\\ &= n(sn-d) + \tau_r(rn+(d-\ell)-sn)\\ &\geq n^2(s-d/n)\,.\end{aligned}$$

using the relations $\sum_{i=0}^{r} \tau_i = n$ and $\sum_{i=0}^{r} i\tau_i \leq \ell$, and the definition of r: since $r \geq s - \frac{d-\ell}{n}$, the coefficient of τ_r in the penultimate expression is non-negative. ■

6.2 Weighted Univariate Multiplicity Code Decoder

For the Weighted Univariate Multiplicity Code Decoder we are given a word $g\colon T \to \mathbb{F}_{<r}[z]$ and a weight function $w\colon T \times [r] \to \mathbb{Z}_{\geq 0}$ along with parameters d, ℓ and s with $\ell \leq d$ and

$r = s - \lfloor \frac{d-\ell}{n} \rfloor$. Further, for every $(a,i) \in T \times [r]$: $w(a,i) \le \frac{n}{2} \cdot \left((s-i) - \frac{d-\ell}{n}\right)$. The algorithm returns a polynomial R of degree at most ℓ such that $\Gamma_w^{d,\ell,s}(g,R)$ is smaller than $\frac{n^2}{2}(s - \frac{d}{n})$. Notice that by Lemma 6.2 there can only be one such R.

Recall that higher the value of $w(a,i)$, lower is our confidence on the $(i-1)^{th}$ derivative specified by g at a. For every $a \in T$, we set $\omega(a) \leftarrow \max_{i \in [r]} w(a,i)$, i.e., the maximum distrust in any derivative specified by g at a. For every step-threshold $\boldsymbol{\theta} = (\theta_0, \theta_1, \ldots, \theta_{r-1}) \in [sn/2]^r$ with $\theta_0 \ge \theta_1 \ge \cdots \ge \theta_{r-1}$ and for every $a \in T$ we retain $g(a)$ up to degree i (equivalently derivatives up to order i) such that $\omega(a) \le \theta_i$. Let the retained set of (a,i)'s be $U_{\boldsymbol{\theta}}$; we then call Algorithm 1 on g restricted to $U_{\boldsymbol{\theta}}$. Then, we check whether the polynomial returned by this step satisfies $\Gamma_w^{d,\ell,s}(g,R) < \frac{n^2}{2}(s - \frac{d}{n})$ and output it if it does.

```
Input:   $T \subseteq \mathbb{F}, |T| = n$                                   ▹ set of evaluation points
         $d, s, m$                                   ▹ global degree, multiplicity and dimension resp.
         $\ell, r$ with $\ell \le d$ and $r = s - \lfloor \frac{d-\ell}{n} \rfloor$          ▹ actual degree and multiplicity resp.
         $g : T \to \mathbb{F}_{<r}[z]$                                          ▹ received word
         $w : T \times [r] \to \mathbb{Z}_{\ge 0}$ satisfying $w(a,i) \le \frac{n^{m-1}}{2}\left(s - i - \frac{d-\ell}{n}\right)$, $\forall (a,i)$
                                                                            ▹ weight function
Output:  $R \in \mathbb{F}_{\le \ell}[x]$ such that $\Gamma_w^{d,\ell,s}(g,R) < \frac{n^m}{2}(s - \frac{d}{n})$, if such an $R$ exists and
         0 otherwise.
for $a \in T$ do
    Set $\omega(a) \leftarrow \max_{i \in [r]} w(a,i)$ ;
for every step threshold $\boldsymbol{\theta} = (\theta_0, \theta_1, \ldots, \theta_{r-1})$ such that $\theta_0 \ge \cdots \ge \theta_{r-1}$ do
        ▹ There are at most $\binom{n+r}{r}$ step thresholds and the algorithm goes over each one of them.
    Set $U_{\boldsymbol{\theta}} \leftarrow \{(a,i) \in T \times [r] \ : \ \omega(a) \le \theta_i \ , \ a \in T, \ i \in [r]\}$ ;
    for $a \in T$ do
        Set $\mathbf{s}(a) \leftarrow \max\{i \le r-1 \ : \ \omega(a) \le \theta_i\} + 1$ ;
    Run Generalized Univariate Multiplicity Code Decoder (Algorithm 1) on
    $(T, \ell, \mathbf{s}, g|_{U_{\boldsymbol{\theta}}})$ where $g|_{U_{\boldsymbol{\theta}}} : T \to \mathbb{F}_{<r}[z]$ is defined as $g|_{U_{\boldsymbol{\theta}}}(a) = g(a) \mod z^{s(a)}$
    to obtain $R$ (if $R = \perp$ set $R \leftarrow 0$) ;
    if $\Gamma_w^{d,\ell,s}(g,R) < \frac{n^m}{2}(s - \frac{d}{n})$ then return $R$ else return 0 .
```

Algorithm 2. Weighted Univariate Multiplicity Code Decoder

REMARK 6.3. The number of step-thresholds is $O(\binom{n+s}{s})$. This can be seen by considering $\theta_0, \theta_1 - \theta_0, \theta_2 - \theta_1, \ldots, \theta_{r-1} - \theta_{r-2}, n - \theta_{r-1}$. Each of these $r+1$ quantities is a non-negative integer and their sum is n, so the number of solutions is $\binom{n+r}{r}$, and $r \le s$.

6.3 Proof of correctness of Algorithm 2

The analysis of this decoder is inspired by the alternative analysis of Forney's GMD decoding mentioned in the previous section.

Notice that the algorithm proceeds by trying every monotone threshold vector $\boldsymbol{\theta}$. (We only consider monotone threshold vectors in the algorithm and following analysis.) Hence, it suffices to show that one $\boldsymbol{\theta}$ exists which can be used to do the decoding. We first characterise when a threshold vector can be used in the call to Algorithm 1. We then prove that such a threshold vector exists in Lemma 6.5. For this we need the following notation. Suppose there is a polynomial $R \in \mathbb{F}_{\leq \ell}[x]$ such that $\Gamma_w^{d,\ell,s}(g, R) < \frac{n^2}{2}(s - \frac{d}{n})$: Lemma 6.2 implies that there can be at most one such R.

Given this polynomial R, we partition the set $T \times [r+1] = A \uplus B$ as follows. Let A be the set of (a, i) such that $g(a) = R(a+z) \mod \langle z \rangle^{(i)}$: in other words, at the location a all derivatives of R till order $i-1$ match with g. And let $B = (T \times [r+1]) \setminus A$. Let $\boldsymbol{\theta}$ be a vector of thresholds.

Observe that if $(a, i) \in A$, then $(a, j) \in A$ for all $j < i$. Then, since $B = (T \times [r+1]) \setminus A$, if $(b, j) \in B$, then $(b, i) \in B$ for all $i > j$.

Define $A_{\boldsymbol{\theta}} := U_{\boldsymbol{\theta}} \cap A$ and similarly $B_{\boldsymbol{\theta}} := U_{\boldsymbol{\theta}} \cap B = U_{\boldsymbol{\theta}} \setminus A_{\boldsymbol{\theta}}$.

OBSERVATION 6.4. *If it holds that $|A_{\boldsymbol{\theta}}| > |B_{\boldsymbol{\theta}}| + \ell$, then Algorithm 1 can decode using $\boldsymbol{\theta}$. We call such threshold vectors* good.

This is because, Algorithm 1 will decode using $\boldsymbol{\theta}$ whenever $\Delta^{(\mathsf{s})}_{\mathsf{mult}}(g|_{U_{\boldsymbol{\theta}}}, \mathsf{Enc}^{(\mathsf{s})}(R)) < \frac{1}{2}(|U_{\boldsymbol{\theta}}| - \ell)$. With our notation, this means $|B_{\boldsymbol{\theta}}| < \frac{|U_{\boldsymbol{\theta}}| - \ell}{2}$, since $\Delta^{(\mathsf{s})}_{\mathsf{mult}}(g|_{U_{\boldsymbol{\theta}}}, \mathsf{Enc}^{(\mathsf{s})}(R)) = |B_{\boldsymbol{\theta}}|$. Rearranging this and using $|U_{\boldsymbol{\theta}}| = |A_{\boldsymbol{\theta}}| + |B_{\boldsymbol{\theta}}|$ gives the above characterization.

The following lemma, which we prove in the next section, shows that there is at least one good step-threshold $\boldsymbol{\theta}$.

LEMMA 6.5. *Let g be the received word. If R is such that $\Gamma_w^{d,\ell,s}(g, R) < \frac{n^2}{2}(s - \frac{d}{n})$, then there is a good vector of thresholds $\boldsymbol{\theta}$ such that Algorithm 2 returns R in the iteration corresponding to $\boldsymbol{\theta}$.*

Armed with the above lemma we are now ready to prove the correctness of Algorithm 2.

THEOREM 6.6. *Let $g \colon T \to \mathbb{F}_{<r}[z]$ be a received word and R a degree ℓ polynomial such that $\Gamma_w^{d,\ell,s}(g, R) < \frac{n^2}{2}(s - \frac{d}{n})$. Then, Algorithm 2 returns the polynomial R. Further, Algorithm 2 runs in time $(sn)^{s+O(1)}$ where n is the size of the set of evaluation points, T.*

PROOF. The algorithm proceeds by trying every step-threshold $\boldsymbol{\theta}$. By Lemma 6.5, there is a good vector of thresholds $\boldsymbol{\theta}$ that can be used to find R in the call to Algorithm 1. Hence, the algorithm finds R within the given distance if one exists. Also, the algorithm never outputs an incorrect R due to the check at Line 8 and the fact that if one such R exists then it is unique by Lemma 6.2.

The running time of the Algorithm is determined by the $O(sn)^r$ iterations of the for-loop over all possible step-thresholds. By Theorem 4.3 each such iteration requires $(nr)^{O(1)}$ time. As $r \leq s$, the overall running time is $(sn)^{s+O(1)}$. ■

6.4 Proof of Lemma 6.5

The proof of Lemma 6.5 requires a few claims and definitions. We begin by showing that $|A| > \ell$.

CLAIM 6.7. *Let g be any received word and R be a degree ℓ polynomial with $\Gamma_w^{d,\ell,s}(g,R) < \frac{n^2}{2}(s - \frac{d}{n})$. Let A be defined as above, that is, the set of locations (a,i) such that $g(a) = R(a+z) \mod \langle z\rangle^{(i)}$. Then, $|A| > \ell$.*

PROOF. Suppose for contradiction that $|A| \leq \ell$. We will show that the error is more than promised.

Write A as a disjoint union, $A = \biguplus_{i=0}^{r} A_{\geq i}(g,R)$ where $A_{\geq i}(g,R) = A \cap (T \times \{i\})$.

Informally, $A_{\geq i}(g,R)$ is the set of locations where g and R agree up to derivatives of order $i-1$. Let the size of $A_{\geq i}(g,R)$ be η_{i-1}, so that we have $n = \eta_0$ elements which are (trivially) "correct" up to the "$(-1)^{th}$" order derivative, η_1 up to the 0^{th} order derivative (i.e. evaluation level), η_2 up to the first-order derivative and so on.

Note that $\sum \eta_i \leq \ell$, by our assumption. Additionally, $\eta_0 \geq \eta_1 \geq \cdots \geq \eta_r$ because if $(a,i) \in A$ then for all $0 \leq j < i$, $(a,j) \in A$. Hence the number of coordinates where g and R disagree at the 0^{th} order derivative is $n - \eta_1$ and the total contribution to Γ from such coordinates is at least $\frac{1}{2}(n-\eta_1)(ns-(d-\ell))$. This is because the contribution from each term is at least $\frac{1}{2}(ns-(d-\ell))$. To observe this, note that in Definition 6.1, in the first term in the maximum, the weight being subtracted is at most $\frac{1}{2}(ns-(d-\ell))$, so the maximum never dips below this quantity. Further, there are at least $(n-\eta_2)-(n-\eta_1) = \eta_1 - \eta_2$ coordinates where g and R agree at the 0^{th} order derivative but disagree at the first order derivative: such coordinates contribute to Γ at least $\frac{1}{2}(\eta_1-\eta_2)(n(s-1)-(d-\ell))$. In this manner, we get the total distance to be

$$\begin{aligned}\Gamma_w^{d,\ell,s}(g,R) &\geq \frac{1}{2}(\eta_0-\eta_1)(ns-(d-\ell)) + \frac{1}{2}(\eta_1-\eta_2)(n(s-1)-(d-\ell)) + \\ &\qquad \cdots + \frac{1}{2}(\eta_{r-1}-\eta_r)(n(s-r+1)-(d-\ell)) \\ &= \frac{1}{2}\sum_{i=0}^{r-1} \delta_i(n(s-i)-(d-\ell))\end{aligned}$$

where $\delta_i = \eta_i - \eta_{1+1}$. Note that each $\delta_i \geq 0$ and $\sum_{i=0}^{r-1} \delta_i \leq n$ (by a telescoping sum). In addition, $\sum_{i=0}^{r-1} i\delta_i = (\sum_{i=2}^{r-2} \eta_i) - (r-1)\eta_r \leq \ell$, by assumption.

The rest of this proof is identical to the argument in the proof of Lemma 6.2.

$$\begin{aligned}\Gamma_w^{s,d,\ell}(g,R) &\geq \frac{1}{2}\sum_{i=0}^{r-1}\delta_i((s-i)n-(d-\ell)) \\ &= \frac{1}{2}\sum_{i=0}^{r-1}\delta_i(sn-(d-\ell)) - \frac{n}{2}\sum_{i=0}^{r-1} i\delta_i \\ &\geq \frac{1}{2}(sn-(d-\ell))n - \frac{n}{2}\ell \\ &\geq \frac{1}{2}n^2\left(s-\frac{d}{n}\right).\end{aligned}$$

which contradicts the assumption. ■

Now we know $|A| > \ell$, so we can write $|A| = \ell + u$ for some positive u. For $a \in T$, let $i_R(a)$ be the i such that $a \in A_i(g,R)$. Recall that for every $i \in [r+1]$

$$A_i(g,R) = \left\{a \in T : \ \max\left\{j \in [r+1] : \ g(a) = R(a+z) \mod \langle z\rangle^j\right\} = i\right\}.$$

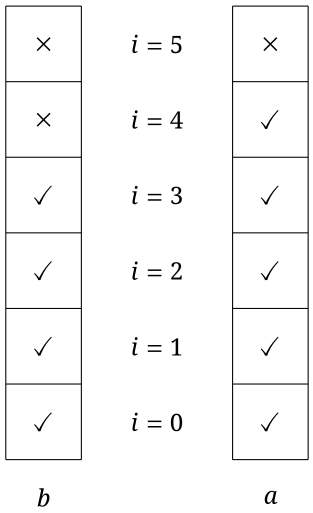
Informally, g and R agree up to derivatives of order $i-1$ at a, but disagree at the i^{th} order derivative. We construct a pairing as in our reproof of Forney's result in Section 5.2; however, we will need it to be a “good” pairing to help in the error analysis. We provide the definition below.

DEFINITION 6.8 (good pairing). Let $a_0,\ldots,a_{k-1},b_0,\ldots,b_{k-1} \in T$. We say the k pairs given by $(a_0,b_0),\ldots,(a_{k-1},b_{k-1})$ is a good pairing if it satisfies the following conditions:

1. $|\{a_i : \ i \in [k]\} \cup \{b_i : \ i \in [k]\}| = 2k$. That is, they are all distinct.
2. $\omega(b_j) \leq \omega(a_j)$ for all $j \in [k-1]$ where as before $\omega(a) = \max_{i\in[r]} w(a,i)$. This is analogous to the weight condition in the Forney case.
3. $i_R(b_j) < i_R(a_j)$ for all $j \in [k]$. This is analogous to one being correct (from A) and the other being corrupted (from B) - here, we need one to be correct up to more derivatives than the other. See Figure 6
4. $\sum_{j=0}^{k-1}(i_R(a_j) - i_R(b_j)) \geq u$. This is analogous to the number of pairs being u (that is, $|A| - \ell$). (While there are $2k$ locations being paired with each other, the quantity of interest is the total difference in the derivatives to which a_i's and b_i's are correct. This has to be at least u).

LEMMA 6.9. *If Algorithm 2 fails to find R for every threshold vector $\boldsymbol{\theta}$, then there exists a good pairing among the elements of T.*

The proof of this lemma requires the following generalization of Hall's theorem (which can be found for instance in the textbook on Matching Theory by Lovász and Plummer).



b		a
×	$i = 5$	×
×	$i = 4$	✓
✓	$i = 3$	✓
✓	$i = 2$	✓
✓	$i = 1$	✓
✓	$i = 0$	✓

Figure 6. When a and b can be in a good pairing, with $\omega(b) \leq \omega(a)$

THEOREM 6.10 (Generalisation of Hall's theorem [17, Theorem 1.3.1]). *In a bipartite graph $G = (\mathcal{L}, \mathcal{R}, E)$, if the maximum matching has size at most m, then there is a subset $U \subseteq \mathcal{L}$ such that*

$$|U| - |N(U)| \geq |\mathcal{L}| - m .$$

PROOF OF LEMMA 6.9. Consider the following bipartite graph with left and right partite sets $\mathcal{L} = A$ and $\mathcal{R} = B$, respectively. It will be useful to stratify $\mathcal{L} = \biguplus_{i=0}^{r} \mathcal{L}_i$ where $\mathcal{L}_i = A_{\geq i}(g, R)$; recall that $A_{\geq i}(g, R) = A \cap (T \times \{i\})$. Similarly, we stratify $\mathcal{R} = \biguplus_{i=0}^{r} \mathcal{R}_i$ where $\mathcal{R}_i = (T \times \{i\}) \setminus \mathcal{L}_i$. From earlier, $(a, i) \in \mathcal{L}_i$ implies that for all $j < i$, $(a, j) \in \mathcal{L}_j$, and inversely for $\mathcal{R}$.

The edge set of the graph is given as (See Figure 7):

$$E = \{((a, i), (b, i)) \,:\, (a, i) \in \mathcal{L}_i \,, (b, i) \in \mathcal{R}_i, \; \omega(a) \geq \omega(b) \,, \; 0 \leq i \leq r\} .$$

It will be useful to visualise the vertices of $\mathcal{L}$ and $\mathcal{R}$, which are of the type (a, i), ordered first according to their strata, i.e., i, and then within each stratum according to the $\omega(a)$ values. See Figure 7.

We first show that this graph has a matching of size u, using the generalization of Hall's theorem from Theorem 6.10. We will then convert this matching into a good pairing using claim 6.12.

CLAIM 6.11. *The graph defined above has a matching of size u, where $|\mathcal{L}| = \ell + u$.*

Proof. Suppose for contradiction that the maximum matching in the graph has size $u - 1$. Appealing to Theorem 6.10 with $m = u - 1$, we have a witness set $U \subseteq \mathcal{L}$ such that $|U| - |N(U)| \geq |\mathcal{L}| - u + 1 = \ell + 1$.

Let U_i be the part of U that lies in each $\mathcal{L}_i$. For each U_i, observe that we may as well take the first (ordered by weight ω — observe that this weight is independent of i) $|U_i|$ elements in $\mathcal{L}_i$, and this does not increase the neighbourhood. This is because (a, i) of weight $\omega(a)$ is

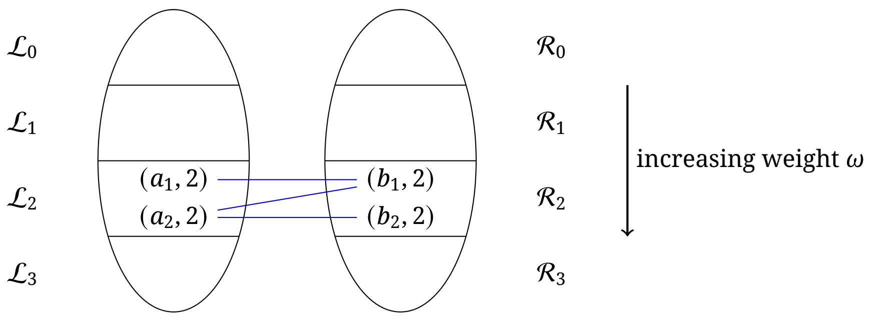


Figure 7. Edge structure of the bipartite graph in Lemma 6.9.

connected only to vertices (b, i) with $\omega(b) \leq \omega(a)$. Hence, we can assume each U_i is a prefix of $\mathcal{L}_i$ (i.e., the first $|U_i|$ elements of $\mathcal{L}_i$).

Further, since $(a, i) \in \mathcal{L}_i$ implies that for all $j < i$, $(a, j) \in \mathcal{L}_j$ we can do the following transformation to U without increasing the size of $N(U)$. If for $i > j$ we have $|U_i| > |U_j|$, then, we can replace U_i and U_j with $U'_i = \left\{(a, i) \colon (a, j) \in U_j\right\}$ and $U'_j = \{(a, j) \colon (a, i) \in U_i\}$ respectively. This does not increase the neighbourhood. Indeed, consider some new neighbour (b, j) caused by an edge $((a, j), (b, j))$. We must have $(a, j) \in U'_j \setminus U_j$. This newly included (a, j) originated from some (a, i) in $U_i \setminus U'_i$. Since $j < i$ and $(b, j) \in B$, we must also have $(b, i) \in B$. Then, the edge $((a, i), (b, i))$ also exists, and (b, i) was previously in the neighbourhood of U but no longer is.

We now have a structure on U such that for every i, the first $|U_i|$ elements of A, ordered by weight ω, are in U. Further, for $i > j$, if $(a, i) \in U_i$ then $(a, j) \in U_j$. With this structure on U we can now create a valid step-threshold $\boldsymbol{\theta}$ so that $A_{\boldsymbol{\theta}}$ contains U (and exactly equals U if the weights ω are all distinct) : indeed, set θ_i to be $\max\{\omega(a) : (a, i) \in U_i\}$ (and if U_i is empty, set θ_i to be $-\infty$). Further, $B_{\boldsymbol{\theta}} = N(U)$. To see this, take $(b, i) \in \mathcal{R}_i$ with $\omega(b) \leq \theta_i$. Then there would be an edge $((a, i), (b, i))$ for the (a, i) with $\omega(a) = \theta_i$ and hence $(b, i) \in N(U)$. Conversely, if $(b, i) \in N(U)$, there is an edge $((a, i), (b, i))$ with $\omega(b) \leq \omega(a) \leq \theta_i$ and hence $(b, i) \in B_{\boldsymbol{\theta}}$. (See Figure 8). However, as $|A_{\boldsymbol{\theta}}| - |B_{\boldsymbol{\theta}}| \geq |U| - |N(U)| \geq \ell + 1$, this contradicts our assumption that there is no good step-threshold $\boldsymbol{\theta}$. Hence, we must have a matching of size at least u in the bipartite graph. ◆

To finish the proof of Lemma 6.9 we will need claim 6.12 which essentially transforms this matching into a good pairing. ■

Next, we proceed to show how to extract a good pairing from a matching obtained in the bipartite graph defined above.

CLAIM 6.12. *Given a maximum matching of size u in the graph defined in Lemma 6.9, a good pairing can be extracted from it.*

PROOF. The given matching M consists of u edges of the form $((a, i), (b, i))$ with $\omega(b) \leq \omega(a)$. We want to extract a good pairing $(a_0, b_0), \ldots, (a_{k-1}, b_{k-1})$.

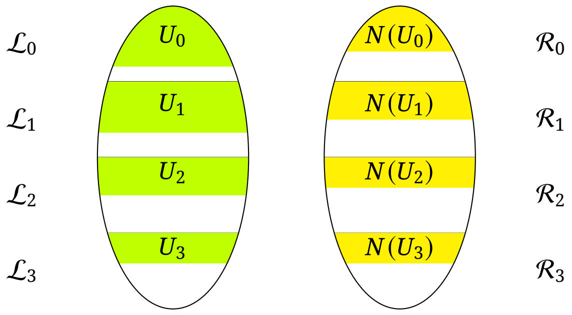


Figure 8. Extracting a threshold from the matching

Recall the conditions from Definition 6.8. For Condition 1, we need to be able to read off the k distinct pairs. We will add (a, b) to the pairing if we have the edges $((a, j), (b, j))$ for every $i_R(b) < j \leq i_R(a)$ in M. This will also take care of Condition 4 since we are given exactly u edges in the matching M. Conditions 2 and 3 are guaranteed by the edge structure.

If M consists of exactly the edges of this type, we are done. But, M is an arbitrary matching and we have no guarantees on its structure. We will now see how to extract a pairing from it regardless, by making some modifications to M while maintaining its size.

We will now use a visualization as in Figure 6, that is, every $a \in T$ has a tower of height s. If the matching M contains an edge $(a, i), (b, i)$, we will think of the a and b towers being connected by an edge at level i.

We need, for every a and b in T, the matching M to either match (a, j) to (b, j) for every j between $i_R(b)$ and $i_R(a)$, or for none of them. (In the first case, (a, b) is in the pairing; otherwise, it is not.) The problem case is if they are matched on some levels but not on others — in this case, we will call a and b a *bad* pair of blocks. We will *correct* this pair by either adding the missing edges or disconnecting a and b completely.

Arrange the elements of T in increasing order of the weight ω. We will process every pair in co-lexicographic order, since for (a, b) to be in a good pairing, we need $\omega(a) \geq \omega(b)$. Hence, when we are trying to correct a pair (a, b), we can assume there are no bad pairs (a', b') with $\omega(b') < \omega(b)$, or $\omega(b') = \omega(b)$ and $\omega(a') < \omega(a)$.

Now, say we are at the stage of processing a bad pair (a, b), and say they are matched at some other levels, but not the level j, that is, the edge $((a, j), (b, j))$ is not in the matching M.

(a, j) could be unmatched in M, or it could be matched to some (b', j). Notice that because of the order in which we are proceeding, we cannot have $\omega(b') < \omega(b)$. Otherwise, (a, b') form a bad pair which we would have encountered earlier. Also, by definition, we cannot have $\omega(b') > \omega(a)$. Hence we must have $\omega(b) \leq \omega(b') \leq \omega(a)$.

As for (b, j), it could be unmatched in M, or matched to some other (a', j). Again, if $\omega(a') < \omega(a)$, there is a bad pair (a', b) which would have been encountered before (a, b), so we can eliminate this case. So we must have $\omega(a) \leq \omega(a')$.

We will now go through the possibilities one by one.

Case 1: (a, j) and (b, j) are both unmatched in M. (shown in Figure 9)

Then they can be matched with each other, which increases the size of the matching which is a contradiction since M was a maximum matching.

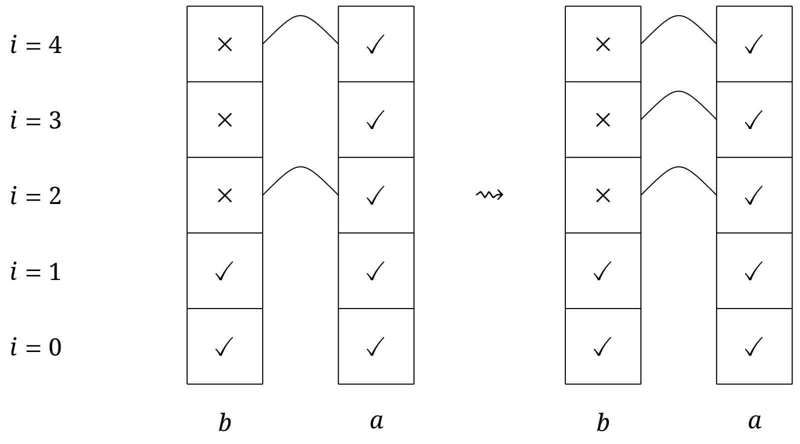


Figure 9. Case 1: $(a, 3)$ and $(b, 3)$ both unmatched

Case 2: (b, j) is matched with some (a', j) with $\omega(a') \geq \omega(a)$. (shown in Figure 10)

Then, we can remove that edge from M and add the edge $((b, j), (a, j))$ to the matching. (If (a, j) is already matched to (b', j) in M, we remove it and add an edge between (b', j) and (a', j).) This may create a new bad pair of higher weight, which can be dealt with subsequently.

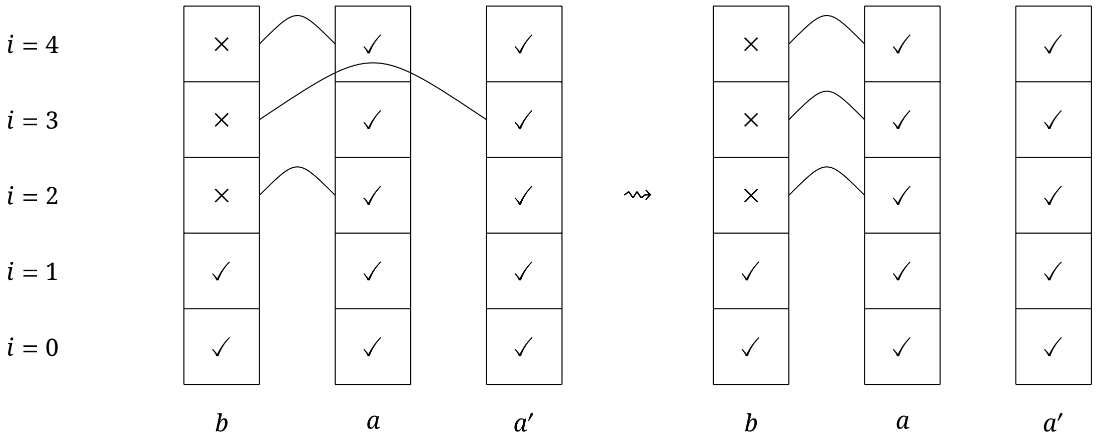


Figure 10. Case 2: $(b, 3)$ is matched with $(c, 3)$ instead of $(a, 3)$

Case 3: (b, j) is unmatched in M, and (a, j) is matched to (b', j). (shown in Figure 11)

Then the edge between (a, j) and (b', j) can be removed and the edge between (a, j) and (b, j) added to M. Again, any new bad pairs created will be dealt with subsequently.

This process will terminate since T is finite. Then, M will be of the form we saw above, and the good pairing can be read off from it. ∎

Now, we show that if there is a good pairing in the elements of T, then, $\Gamma_w^{d,\ell,s}(g, R) \geq \frac{n^2}{2}(s - \frac{d}{n})$. This is shown by transforming the received word g and the weight function w into a different

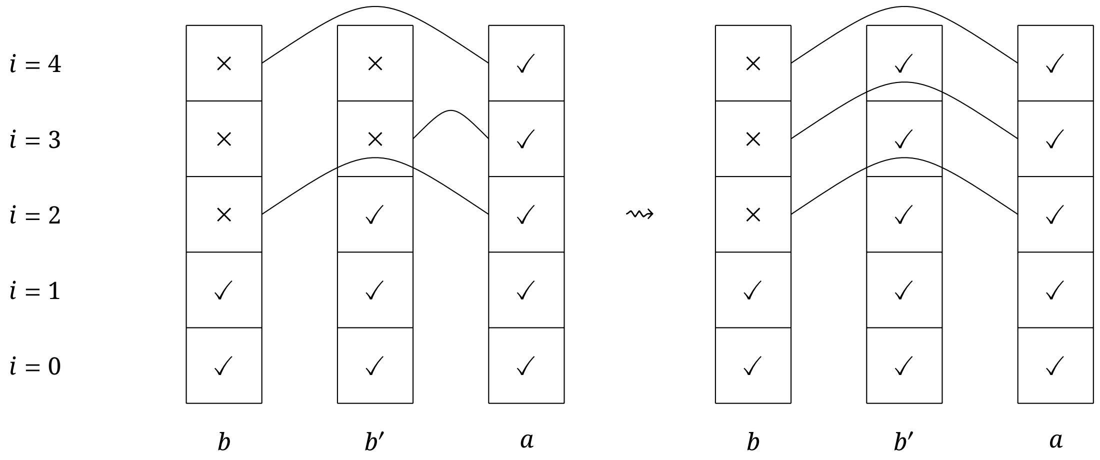


Figure 11. Case 3: $(a, 3)$ is matched with $(b', 3)$ instead of $(b, 3)$

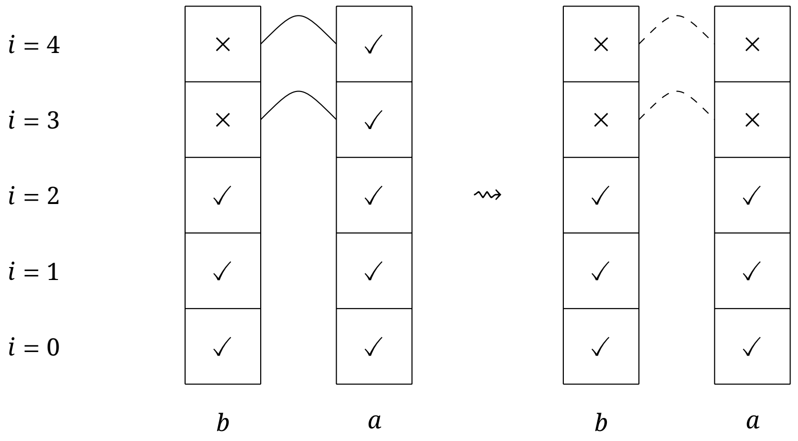


Figure 12. Construction of the new received word

word g' and weight function w', by changing the values of g and w at the locations where the good pairing exists. In doing so, we ensure $\Gamma(g, R)$ is non-increasing. We then reach a scenario where $|A| \le \ell$ for this new word g'. Then, by claim 6.7 we get that $\Gamma_w^{d,\ell,s}(g, R) \ge \Gamma_{w'}^{d,\ell,s}(g', R) \ge \frac{n^2}{2}(s - \frac{d}{n})$.

CLAIM 6.13. *Assume we have a good pairing of size k, $(a_0, b_0), \ldots, (a_{k-1}, b_{k-1})$, in the received word g. Construct a new received word g' as follows: for each j in $[k]$ (recall that $i_R(a_j) > i_R(b_j)$) in the original received word, corrupt the values of $\{g(a_j, i) : i \in \{i_R(b_j) + 1, \ldots, i_R(a_j)\}\}$ such that $i_R(a_j) = i_R(b_j)$. Further, for all $i \in [r]$ let*

$$w'(a_j, i) = w'(b_j, i) = \min\left\{\frac{n}{2} \cdot \left((s - i_R(b_j)) - \frac{d-\ell}{n}\right), \frac{n}{2} \cdot \left((s - i) - \frac{d-\ell}{n}\right)\right\}.$$

At all other locations which are not part of the pairing, g' and w' are exactly the same as g and w. See ?? 12. We note w' is a valid weight function.Then, the error as per our measure actually does not increase, *that is $\Gamma_{w'}^{d,\ell,s}(g', R) \le \Gamma_w^{d,\ell,s}(g, R)$. Furthermore, $\Gamma_{w'}^{d,\ell,s}(g', R) \ge \frac{n^2}{2}(s - \frac{d}{n})$.*

PROOF. Let (a, b) be one of the pairs in the pairing. We count the contribution of a and b to Γ before and after the transformation is applied.

Suppose $i_R(a) = i_a$ (originally) and $i_R(b) = i_b$. Since it is a good pairing, $i_a > i_b$ and $\omega(b) \le \omega(a)$.

Originally, their contribution to $\Gamma_w^{d,\ell,s}(g,R)$ is at least

$$\omega(a) + ((s-i_b)n - (d-\ell) - w(b,i_b)).$$

The reason why the contribution from a is at least $\omega(a)$ is as follows. Let $\omega(a) = w(a,i^*)$. That is, let $i^* = \arg\max_{i\in[r]} w(a,i)$. If $i^* < i_a$, then $\max_{j<i_a} w(a,i) = \omega(a)$. On the other hand, if $i^* \ge i_a$, we have that $w(a,i^*) \le \frac{1}{2}(n(s-i^*) - (d-\ell))$, as well as $w(a,i_a) \le \frac{1}{2}(n(s-i_a) - (d-\ell))$. Then the contribution to Γ is at least $n(s-i_a) - (d-\ell) - w(a,i_a) \ge n(s-i^*) - (d-\ell) - w(a,i^*) \ge \frac{1}{2}(n(s-i^*) - (d-\ell)) \ge w(a,i^*) = \omega(a)$.

Now, since $w(b,i_b) \le \omega(b) \le \omega(a)$, the total contribution is at least $(s-i_b)n - (d-\ell)$.

Now, we apply the corruption to $i_a - i_b$ levels of a and change the weight function w at a and b. Then, to $\Gamma_{w'}^{d,\ell,s}(g',R)$, a and b each contribute $\frac{1}{2}((s-i_b)n - (d-\ell))$, giving a total of $((s-i_b)n - (d-\ell))$, which is the lower bound for the previous expression.

Notice, that after we have applied this transformation to all the pairs present in the good pairing we are left with a word g' such that $|A| \le \ell$. This is because by Condition 4 of Definition 6.8 we have $\sum_{j=0}^{k-1}(i_R(a_j) - i_R(b_j)) \ge u$: hence, we have corrupted at least u locations (a,i) of agreement between g and r to obtain g'. However, now g' and w' satisfy the hypothesis of claim 6.7. Putting everything together we get that $\Gamma_w^{d,\ell,s}(g,R) \ge \Gamma_{w'}^{d,\ell,s}(g,R) \ge \frac{n^2}{2}(s - \frac{d}{n})$. ■

We can finally complete the proof of Lemma 6.5.

PROOF OF LEMMA 6.5. We prove this by contradiction, by assuming that no threshold $\boldsymbol{\theta}$ is good. By observation 6.4, this means we assume $|A_{\boldsymbol{\theta}}| \le |B_{\boldsymbol{\theta}}| + \ell$ for all $\boldsymbol{\theta}$.

claim 6.7 shows that A must have size at least ℓ, or the error is more than promised. Hence, we can write $|A| = \ell + u$ for some positive u.

As in our reproof of Forney's result in Section 5.2, we construct a pairing. Here, the size of the pairing is u - measured not as the number of locations $a \in T$ participating in the pairing but the total difference in the multiplicity levels summed over each pair. We add some conditions for it to be a *good* pairing in definition 6.8: Lemma 6.9 shows that such a pairing exists. Given such a *good* pairing claim 6.13 shows that $\Gamma_w^{d,\ell,s}(g,R) \ge \frac{n^2}{2}(s - \frac{d}{n})$, a contradiction. ■

7. Bivariate multiplicity code decoder

In this section, we describe our decoding algorithm for bivariate multiplicity codes. It relies on algorithms for decoding two variants of a decoder for univariate multiplicity codes; the first where the multiplicity parameters varies with the evaluation point (Algorithm 1) and the second being a decoder for a weighted version of univariate multiplicity codes (Algorithm 2).

7.1 Description of the bivariate decoder

We start with an informal description of this algorithm, along the lines suggested in the overview Section 2.

The bivariate decoder takes as input sets $T_1, T_2 \subseteq \mathbb{F}$ of size n, degree and multiplicity parameters d and s, and the received word $f\colon T_1 \times T_2 \longrightarrow \mathbb{F}_{<s}[z_1, z_2]$. The decoder outputs a polynomial $P \in \mathbb{F}_{\le d}[x_1, x_2]$ such that $\Delta^{(s)}_{\mathsf{mult}}(f, \mathsf{Enc}^{(s)}(P)) < \frac{1}{2}n^2(s - \frac{d}{n})$ if one exists. It will be convenient to write the polynomial P in the following form.

$$P(x_1, x_2) = \sum_{\ell \in [d+1]} P_\ell(x_1) x_2^{d-\ell} \in \mathbb{F}_{\le d}[x_1, x_2] \ .$$

The decoder proceeds in $d+1$ iterations numbered 0 to d where in the ℓ^{th} iteration, the decoder recovers the univariate polynomial $P_\ell(x_1) \in \mathbb{F}_{\le \ell}[x_1]$. Having obtained the polynomials $P_0, P_1, \ldots, P_{\ell-1}$ correctly in the previous iterations, the decoder in the ℓ^{th} iteration peels away these polynomials from the received word f to obtain the word f_ℓ defined as

$$f_\ell := f - \mathsf{Enc}^{(s)}\left(\sum_{i \in [\ell]} P_i(x_1) x_2^{d-i}\right).$$

The appropriate choice of the multiplicity parameter for the ℓ^{th} iteration is $r = s - \lfloor \frac{d-\ell}{n} \rfloor$ (see Remark 7.1). For each $a \in T_1$, the decoder unravels the received word $f_\ell|_{x_1=a}$ along the column $x_1 = a$ into r parts $\left\{f_\ell^{(i,a)}\right\}_{i \in [r]}$ in Line 6. It then runs the univariate decoder on each of these parts to obtain the polynomials $G_\ell^{(i,a)}(x_2) \in \mathbb{F}_{d-\ell}[x_2]$ respectively for each $i \in [r]$. It then constructs the polynomial g_ℓ from the leading coefficients of these polynomials (Line 10) and the corresponding weights $w_\ell(a, i)$ which indicate how close the polynomial $G_\ell^{(i,a)}(x_2)$ is to the word $f_\ell^{(i,a)}$ (Line 9). Finally it extracts the polynomial P_ℓ from the pair (g_ℓ, w_ℓ) using the Weighted Univariate Multiplicity Code Decoder.

REMARK 7.1. We remark that the number of derivatives r we use here (Line 3) is $s - \lfloor \frac{d-\ell}{n} \rfloor$. This is the value that satisfies a few constraints. For the call to Algorithm 1 (Line 7) to work, it must be the case that $(s-i)n - (d-\ell) \ge 0$, for every i in $[r]$. In other words, we must have $r - 1 \le s - (d-\ell)/n$. For the call to Algorithm 2 (Line 11), we must have $rn - \ell \ge 0$. Finally, in the proof of correctness of Algorithm 2 (specifically, Lemma 6.2), we require $rn - \ell \ge sn - d$. (This actually subsumes the previous constraint.) $r := s - \lfloor \frac{d-\ell}{n} \rfloor$ satisfies all these (and is the unique integer that does so unless $\frac{d-\ell}{n}$ is an integer). When $\ell = 0$, $r \ge 1$, and when $\ell = d$, $r = s$.

```
Input:  $T_1, T_2 \subseteq \mathbb{F}$, $|T_1| = |T_2| = n$                          ▷ points of evaluation
        $d, s$                                                              ▷ degree and multiplicity resp.
        $f \colon T_1 \times T_2 \to \mathbb{F}_{<s}[z_1, z_2]$.                               ▷ received word
Output: $P = \sum_{\ell \in [d+1]} P_\ell(x_1)\, x_2^{d-\ell} \in \mathbb{F}_{\leq d}[x_1, x_2]$ such that
        $\Delta^{(s)}_{\text{mult}}\left(f, \text{Enc}^{(s)}(P)\right) < \frac{1}{2}n^2\left(s - \frac{d}{n}\right)$, if such a $P$ exists
for $\ell \leftarrow 0$ to $d$ do
    Define $f_\ell \colon T_1 \times T_2 \to \mathbb{F}_{<s}[z_1, z_2]$ as               ▷ received word for the $\ell^{th}$ iteration
            $f_\ell \leftarrow f - \text{Enc}^{(s)}\left(\sum_{i \in [\ell]} P_i(x_1)\, x_2^{d-i}\right)$;
    $\forall (a, b) \in T_1 \times T_2$, let                                           ▷ some more notation
            $f_\ell(a, b) = \sum_{i,j} f_{\ell,(i,j)}(a, b)\, z_1^i z_2^j$ ;
    Set $r \leftarrow s - \lfloor \frac{d-\ell}{n} \rfloor$;                                      ▷ number of usable derivatives
    for $i \leftarrow 0$ to $r - 1$ do
        for $a \in T_1$ do
            Define $f_\ell^{(i,a)} \colon T_2 \to \mathbb{F}_{<s-i}[z_2]$ as
                    $f_\ell^{(i,a)}(b) \leftarrow \sum_{j \in [s-i]} f_{\ell,(i,j)}(a, b) \cdot z_2^j$;
            Run Generalized Univariate Multiplicity Code Decoder
            (Algorithm 1) on $\left(T_2, d - \ell, s - i, f_\ell^{(i,a)}\right)$ to obtain $G_\ell^{(i,a)} \in \mathbb{F}_{\leq d-\ell}[x_2]$;
            if $G_\ell^{(i,a)}$ is $\perp$ then set $G_\ell^{(i,a)} \leftarrow 0$;
    Define $w_\ell \colon T_1 \times [r] \to \mathbb{Z}_{\geq 0}$ as
            $w_\ell(a, i) \leftarrow \min\left\{\Delta^{(s-i)}_{\text{mult}}\left(f_\ell^{(i,a)}, \text{Enc}^{(s-i)}\left(G_\ell^{(i,a)}\right)\right), \frac{n}{2} \cdot \left((s - i) - \frac{d-\ell}{n}\right)\right\}$;
    Define $g_\ell \colon T_1 \to \mathbb{F}_{<r}[z_1]$ as
            $g_\ell(a) \leftarrow \sum_{i \in [r]} \text{Coeff}_{x_2^{d-\ell}}\left(G_\ell^{(i,a)}\right) z_1^i$;
    Run Weighted Univariate Multiplicity Code Decoder (Algorithm 2) on
    $(T_1, d, s, 2, \ell, r, g_\ell, w_\ell)$ to get $P_\ell(x_1)$;
Set $P(x_1, x_2) \leftarrow \sum_{\ell \in [d+1]} P_\ell(x_1)\, x_2^{d-\ell}$;
if $\Delta^{(s)}_{\text{mult}}\left(f, \text{Enc}^{(s)}(P)\right) < \frac{1}{2}n^2\left(s - \frac{d}{n}\right)$ then return $P(x_1, x_2)$ else return $\perp$.
```

Algorithm 3. Bivariate Multiplicity Code Decoder

7.2 Analysis of the bivariate decoder

We now formally state the claim of correctness and the running time of Algorithm 3. It asserts that Algorithm 3 does indeed decode bivariate multiplicity codes from half their minimum distance on arbitrary product sets, hence giving us the bivariate version of Theorem 3.9.

THEOREM 7.2 (correctness of bivariate decoder (Algorithm 3)). *Let $d, s, n \in \mathbb{N}$ be such that $d < sn$, $\mathbb{F}$ be any field and let T_1, T_2 be arbitrary subsets of $\mathbb{F}$ of size n each. Let $f\colon T_1 \times T_2 \to \mathbb{F}_{<s}[z_1, z_2]$ be any function.*

Then, on input T_1, T_2, s, d, n and f, Algorithm 3 runs in time $(sn)^{s+O(1)}$ and outputs a polynomial $P \in \mathbb{F}[x_1, x_2]$ of total degree at most d such that

$$\Delta^{(s)}_{\mathrm{mult}}(f, \mathrm{Enc}^{(s)}_{T_1\times T_2}(P)) < \frac{1}{2}n^2\left(s - \frac{d}{n}\right),$$

if such a P exists.

Algorithm 3 makes calls to Algorithm 1 and Algorithm 2. We have shown the correctness of both individually.

To prove correctness of Algorithm 2 one of the ingredients we need is that the call on Line 11 always satisfies the promise $\Gamma^{d,\ell,s}_{w_\ell}(g_\ell, R) < \frac{n^2}{2}(s - \frac{d}{n})$, which is required for Algorithm 2 to work. We prove this in the lemma below.

LEMMA 7.3 (Relationship between Γ and multiplicity distance). *Suppose we are in the ℓ^{th} iteration of the for-loop at Line 1 of Algorithm 2. To recall, we have subsets $T_1, T_2 \subseteq \mathbb{F}$ sets of size n each, and natural numbers d, ℓ, s, r with $d \geq \ell$ and $r = s - \lfloor \frac{d-\ell}{n} \rfloor$. Let $\widetilde{P}_\ell = \sum_{i=\ell}^{d} P_i(x_1)x_2^{d-i}$ be the polynomial of degree at most d with $\Delta^{(s)}_{\mathrm{mult}}(f_\ell, \mathrm{Enc}^{(s)}_{T_1\times T_2}(\widetilde{P}_\ell)) < \frac{1}{2}n^2(s - \frac{d}{n})$, that is, the “remaining” portion of P after $P_0, \dots, P_{\ell-1}$ have been peeled off.*

If $g_\ell\colon T_1 \to \mathbb{F}_{<r}[z_1]$ and $w_\ell\colon T_1 \times [r] \to \mathbb{Z}_{\geq 0}$ are as defined in Algorithm 3, i.e., for $a \in T_1$, $g_\ell(a)$ is the guess for $(P_\ell(a + z_1) \mod \langle z_1\rangle^r)$ that comes from using Algorithm 1 on f_ℓ at $x_1 = a$ and $w_\ell(a, i)$ represents the confidence we have in the $(i-1)^{th}$ derivative of $P_\ell(a + z_1)$ at a as specified by g_ℓ, then

$$\Gamma^{s,d,\ell}_{w}(g_\ell, P_\ell) \leq \Delta^{(s)}_{\mathrm{mult}}(f_\ell, \mathrm{Enc}^{(s)}_{T_1\times T_2}(\widetilde{P}_\ell)) < \frac{1}{2}n^2\left(s - \frac{d}{n}\right).$$

PROOF. It will be helpful to recall the definition of $\Gamma^{s,d,\ell}_{w_\ell}$ from Definition 6.1:

$$\Gamma^{s,d,\ell}_{w_\ell}(g_\ell, P_\ell) := \sum_{i=0}^{r-1} \sum_{a\in A_i(g_\ell,P_\ell)} \max\left\{\left(n\cdot\left((s-i) - \frac{d-\ell}{n}\right) - w_\ell(a,i)\right), \max_{j<i} w_\ell(a,j)\right\} + \sum_{a\in A_r(g_\ell,P_\ell)} \max_{j<r} w_\ell(a,j),$$

where for every $i \in [r+1]$

$$A_i\left(g_\ell, P_\ell\right) = \left\{a \in T_1 : \ \max\left\{j \in [r+1] : \ g_\ell(a) = P_\ell(a+z_1) \mod \langle z_1\rangle^j\right\} = i\right\}.$$

We also recall the definition of the weights. In short, for every column $a \in T_1$ and level i, $f_\ell^{(i,a)}$ is the received word and $G_\ell^{(i,a)}$ is the result of univariate multiplicity decoding. The weight is their multiplicity distance, capped at $\frac{n}{2} \cdot ((s-i) - \frac{d-\ell}{n})$.

Formally, $f_\ell^{(i,a)} : T_2 \to \mathbb{F}_{<s-i}[z_2]$ refers to

$$f_\ell^{(i,a)}(b) = \sum_{j \in [s-i]} f_{\ell,(i,j)}(a,b) \cdot z_2^j,$$

while $G_\ell^{(i,a)}$ is the output of Algorithm 1 on input $\left(T_2, d-\ell, s-i, f_\ell^{(i,a)}\right)$.

Then, the weights w_ℓ are defined as

$$w_\ell(a,i) := \min\left\{\Delta_{\mathsf{mult}}^{(s-i)}\left(f_\ell^{(i,a)}, \mathsf{Enc}^{(s-i)}\left(G_\ell^{(i,a)}\right)\right), \frac{n}{2} \cdot \left((s-i) - \frac{d-\ell}{n}\right)\right\}.$$

Having found $G_\ell^{(i,a)}$ for each $a \in T$ and $i \in [s]$, we extract our guess for P_ℓ for each $a \in T$, as the function $g_\ell : T_1 \to \mathbb{F}_{<r}[z_1]$.

$$g_\ell(a) := \sum_{i \in [r]} \mathsf{Coeff}_{x_2^{d-\ell}}\left(G_\ell^{(i,a)}\right) z_1^i.$$

It follows from the definition that $A_i\left(g_\ell, P_\ell\right)$ refers to the set of a's in T_1 such that g_ℓ and P_ℓ agree up to the $(i-1)^{th}$-derivative at a but not at the i^{th}-derivative for $i < r$ while $A_r\left(g_\ell, P_\ell\right)$ refers to the set of a's in T_1 such that g_ℓ and P_ℓ agree up to the $(r-1)^{th}$-derivative. Further, as mentioned earlier, $w_\ell(a,i)$ represents the confidence we have in the $(i-1)^{th}$ derivative of $P_\ell(a+z_1)$ at a as specified by g_ℓ.

Also, define the polynomial $\widetilde{P}_\ell^{(i,a)}(x_2) \in \mathbb{F}_{\le d-\ell}[x_2]$ as

$$\widetilde{P}_\ell^{(i,a)}(x_2) := \mathsf{Coeff}_{z_1^i}\left(\sum_{j=\ell}^{d} P_j\left(a+z_1\right) x_2^{d-j}\right).$$

That is, the i^{th} Hasse derivative with respect to x_1, of $\widetilde{P}_\ell$ at a.

Hence, if the received word f_ℓ and the polynomial $\widetilde{P}_\ell = \sum_{j=\ell}^{d} P_j$ agree completely, then the following three conditions are met

- $f_\ell^{(i,a)}(b) = \left(\mathsf{Enc}^{(s-i)}\left(\widetilde{P}_\ell^{(i,a)}\right)\right)(b) = \widetilde{P}_\ell^{(i,a)}\left(b+z_2\right) \mod \langle z_2\rangle^{(s-i)}, \forall i, a, b.$
- The polynomials $G_\ell^{(i,a)}$ and $P_\ell^{(i,a)}$ are identical, for all i, a.
- $g_\ell = \mathsf{Enc}^{(r)}(P_\ell)$.

Our ultimate goal is to prove $\Gamma_w^{s,d,\ell}(g_\ell, P_\ell) \le \Delta_{\mathsf{mult}}^{(s)}(f_\ell, \mathsf{Enc}_{T_1 \times T_2}^{(s)}(\widetilde{P}_\ell))$. We will first show

$$\Gamma_w^{s,d,\ell}(g_\ell, P_\ell) \le \sum_{a \in T_1} \max_{i \in [s]}\left\{\Delta_{\mathsf{mult}}^{(s-i)}\left(f_\ell^{(i,a)}, \mathsf{Enc}^{(s-i)}\left(\widetilde{P}_\ell^{(i,a)}\right)\right)\right\}$$

and then

$$\sum_{a\in T_1} \max_{i\in[s]} \left\{\Delta_{\mathsf{mult}}^{(s-i)}\left(f_\ell^{(i,a)}, \mathsf{Enc}^{(s-i)}\left(\widetilde{P}_\ell^{(i,a)}\right)\right)\right\} \le \Delta_{\mathsf{mult}}^{(s)}\left(f_\ell, \mathsf{Enc}^{(s)}\left(\widetilde{P}_\ell\right)\right)$$

First, fix some $a \in T_1$ and let $a \in A_i(g_\ell, P_\ell)$ for some $i \in [r+1]$. Recall that its contribution to $\Gamma_w^{s,d,\ell}(g_\ell, P_\ell)$ is at least $n\left((s-i) - \frac{d-\ell}{n}\right) - w_\ell(a,i)$. We will show an upper bound on this quantity.

For every $j \in [r+1]$, since the decoded polynomial $G^{(j,a)}$ is the closest, it is no further than the encoding of $\widetilde{P}_\ell^{(j,a)}$.

$$\Delta_{\mathsf{mult}}^{(s-j)}\left(f_\ell^{(j,a)}, \mathsf{Enc}^{(s-j)}\left(\widetilde{P}_\ell^{(j,a)}\right)\right) \ge \Delta_{\mathsf{mult}}^{(s-j)}\left(f_\ell^{(j,a)}, \mathsf{Enc}^{(s-j)}\left(G_\ell^{(j,a)}\right)\right).$$

Further if $i < r$, we can now apply the triangle inequality to get

$$\Delta_{\mathsf{mult}}^{(s-i)}\left(f_\ell^{(i,a)}, \mathsf{Enc}^{(s-i)}\left(\widetilde{P}_\ell^{(i,a)}\right)\right) + \Delta_{\mathsf{mult}}^{(s-i)}\left(f_\ell^{(i,a)}, \mathsf{Enc}^{(s-i)}\left(G_\ell^{(i,a)}\right)\right) \\ \ge \Delta_{\mathsf{mult}}^{(s-i)}\left(\mathsf{Enc}^{(s-i)}\left(\widetilde{P}_\ell^{(i,a)}\right), \mathsf{Enc}^{(s-i)}\left(G_\ell^{(i,a)}\right)\right).$$

Since $i < r$ and $a \in A_i(g_\ell, P_\ell)$, we have $g_\ell(a) \ne P_\ell(a+z_1) \mod \langle z_1\rangle^i$. Hence, $\widetilde{P}_\ell^{(i,a)} \ne G_\ell^{(i,a)}$ are two distinct polynomials of degree at most $d-\ell$ and by the distance of the $(s-i)^{th}$ order multiplicity code, we have

$$\Delta_{\mathsf{mult}}^{(s-i)}\left(\mathsf{Enc}^{(s-i)}\left(\widetilde{P}_\ell^{(i,a)}\right), \mathsf{Enc}^{(s-i)}\left(G_\ell^{(i,a)}\right)\right) \ge n\left((s-i) - \frac{d-\ell}{n}\right),$$

which in turn yields, combining with the triangle inequality from above,

$$\Delta_{\mathsf{mult}}^{(s-i)}\left(f_\ell^{(i,a)}, \mathsf{Enc}^{(s-i)}\left(\widetilde{P}_\ell^{(i,a)}\right)\right) \ge n\left((s-i) - \frac{d-\ell}{n}\right) - \Delta_{\mathsf{mult}}^{(s-i)}\left(f_\ell^{(i,a)}, \mathsf{Enc}^{(s-i)}\left(G_\ell^{(i,a)}\right)\right).$$

Recall that

$$w_\ell(a,i) = \min\left\{\Delta_{\mathsf{mult}}^{(s-i)}\left(f_\ell^{(i,a)}, \mathsf{Enc}^{(s-i)}\left(G_\ell^{(i,a)}\right)\right), \frac{n}{2}\cdot\left((s-i) - \frac{d-\ell}{n}\right)\right\}$$

First we consider the case that the minimum is attained at the first term. That is,

$$\Delta_{\mathsf{mult}}^{(s-i)}\left(f_\ell^{(i,a)}, \mathsf{Enc}^{(s-i)}\left(G_\ell^{(i,a)}\right)\right) \le \frac{n}{2}\cdot\left((s-i) - \frac{d-\ell}{n}\right).$$

In that case,

$$n\left((s-i) - \frac{d-\ell}{n}\right) - \Delta_{\mathsf{mult}}^{(s-i)}\left(f_\ell^{(i,a)}, \mathsf{Enc}^{(s-i)}\left(G_\ell^{(i,a)}\right)\right) = n\left((s-i) - \frac{d-\ell}{n}\right) - w_\ell(a,i)$$

Otherwise, if $\Delta_{\mathsf{mult}}^{(s-i)}\left(f_\ell^{(i,a)}, \mathsf{Enc}^{(s-i)}\left(G_\ell^{(i,a)}\right)\right) > \frac{n}{2}\cdot\left((s-i) - \frac{d-\ell}{n}\right)$, then,

$$\Delta_{\mathsf{mult}}^{(s-i)}\left(f_\ell^{(i,a)}, \mathsf{Enc}^{(s-i)}\left(\widetilde{P}_\ell^{(i,a)}\right)\right) > \frac{n}{2}\cdot\left((s-i) - \frac{d-\ell}{n}\right).$$

In both cases, we get that if $i < r$, then

$$\Delta^{(s-i)}_{\mathsf{mult}}\left(f_\ell^{(i,a)}, \mathsf{Enc}^{(s-i)}\left(\widetilde{P}_\ell^{(i,a)}\right)\right) \geq n\left((s-i) - \frac{d-\ell}{n}\right) - w_\ell(a,i).$$

Now, by the definition of Γ and w,

$$\begin{aligned}
\Gamma_w^{s,d,\ell}(g_\ell, P_\ell) &= \sum_{i=0}^{r-1} \sum_{a\in A_i(g_\ell,P_\ell)} \max\left\{\left(n\left((s-i)-\frac{d-\ell}{n}\right) - w_\ell(a,i)\right), \max_{j<i} w_\ell(a,j)\right\} \\
&\quad + \sum_{a\in A_r(g_\ell,P_\ell)} \max_{j<r} w_\ell(a,j) \\
&\leq \sum_{a\in T_1} \max_{i\in[s]} \left\{\Delta^{(s-i)}_{\mathsf{mult}}\left(f_\ell^{(i,a)}, \mathsf{Enc}^{(s-i)}\left(\widetilde{P}_\ell^{(i,a)}\right)\right)\right\}.
\end{aligned}$$

This proves the first inequality indicated above. We now prove the second:

$$\sum_{a\in T_1} \max_{i\in[s]} \left\{\Delta^{(s-i)}_{\mathsf{mult}}\left(f_\ell^{(i,a)}, \mathsf{Enc}^{(s-i)}\left(\widetilde{P}_\ell^{(i,a)}\right)\right)\right\} \leq \Delta^{(s)}_{\mathsf{mult}}\left(f_\ell, \mathsf{Enc}^{(s)}\left(\widetilde{P}_\ell\right)\right).$$

To see this, note that the right-hand side is $\sum_{a\in T_1}\sum_{b\in T_2}\left(s - d^{(s)}_{\min}\left(f_\ell(a,b) - \left(\mathsf{Enc}^{(s)}\left(\widetilde{P}_\ell\right)\right)(a,b)\right)\right)$ and the left-hand side is $\sum_{a\in T_1}\max_{i\in[s]}\sum_{b\in T_2}\left((s-i) - d^{(s-i)}_{\min}\left(f_\ell^{(i,a)}(b) - \left(\mathsf{Enc}^{(s-i)}\left(\widetilde{P}_\ell^{(i,a)}\right)\right)(b)\right)\right)$. Since both have a summation over $a \in T_1$, we will show the inequality holds term by term for each $a \in T_1$.

Hence, fix an $a \in T_1$. Let $d^{(s)}_{\min}\left(f_\ell(a,b) - \left(\mathsf{Enc}^{(s)}\widetilde{P}_\ell\right)(a,b)\right) = d_0$. Then the right-hand side is $s - d_0$.

Say the maximum on the left-hand side is attained at some i_0. Then the left-hand side is

$$\sum_{b\in T_2}(s-i_0) - d^{(s-i_0)}_{\min}\left(f_\ell^{(i_0,a)}(b) - \left(\mathsf{Enc}^{(s-i_0)}\widetilde{P}_\ell^{(i_0,a)}\right)(b)\right).$$

Note that $d_0 \leq i_0 + d^{(s-i_0)}_{\min}\left(f_\ell^{(i_0,a)}(b) - \left(\mathsf{Enc}^{(s-i_0)}\widetilde{P}_\ell^{(i_0,a)}\right)(b)\right)$, and hence

$$s - d_0 \geq (s-i_0) - d^{(s-i_0)}_{\min}\left(f_\ell^{(i_0,a)}(b) - \left(\mathsf{Enc}^{(s-i_0)}\widetilde{P}_\ell^{(i_0,a)}\right)(b)\right).$$

This completes the proof. ■

Now that we have all the necessary ingredients, we complete the proof of Theorem 7.2.

PROOF OF THEOREM 7.2. We first observe that Algorithm 3 never outputs an incorrect answer, since towards the end of the algorithm we always check whether the polynomial P that is the potential output indeed satisfies

$$\Delta^{(s)}_{\mathsf{mult}}(f, \mathsf{Enc}^{(s)}_{T_1\times T_2}(P)) < \frac{1}{2}n^2\left(s - \frac{d}{n}\right),$$

and P is output only if the check passes. In particular, Algorithm 3 does not output a polynomial if there is no codeword close enough to the received word.

Thus, to show the correctness of the algorithm, it suffices to assume that there exists a polynomial Q of degree at most d that is close to the received word f and argue that in this case the polynomial P output by the algorithm equals Q. Let $Q(x_1, x_2) = \sum_{\ell=0}^{d} Q_\ell(x_1)x_2^{d-\ell}$ with the polynomials Q_ℓ satisfying $\deg(Q_\ell) \leq \ell$.

Algorithm 3 proceeds in $d+1$ iterations and we now claim that at the end of iteration ℓ, we have correctly recovered $Q_0, Q_1, \ldots, Q_\ell$. More formally, we have the following claim.

CLAIM 7.4. *Let ℓ be any element in $\{0, 1, \ldots, d\}$. Then, at the end of the iteration ℓ of the* for *loop in line* 2 *of Algorithm 3, we have that the polynomial $P_\ell(x_1)$ equals $Q_\ell(x_1)$.*

Clearly, the claim proves the correctness of Algorithm 3. We now prove this claim by a strong induction on ℓ. The argument for the base case of the algorithm, i.e. $\ell = 0$ is essentially the same as the in the induction step. So, we just sketch the argument for the induction step.

To this end, we assume that for every $i \in \{0, 1, \ldots, \ell-1\}$, $P_i(x_1) = Q_i(x_1)$ and prove that $P_\ell(x_1) = Q_\ell(x_1)$. To start with, let

$$\widetilde{Q}_\ell := Q - \sum_{i=0}^{\ell-1} Q_i(x_1)x_2^{d-i} .$$

From the induction hypothesis, note that

$$\widetilde{Q}_\ell = Q - \sum_{i=0}^{\ell-1} P_i(x_1)x_2^{d-i} .$$

Thus, from the definition of the function $f_\ell \colon T_1 \times T_2 \to \mathbb{F}_{<s}[z_1, z_2]$ in Line 2 of Algorithm 3 as

$$f_\ell := f - \mathsf{Enc}^{(s)}\left(\sum_{i=0}^{\ell-1} P_i(x_1)x_2^{d-i}\right) ,$$

and the linearity of the encoding map for multiplicity codes, we have

$$\begin{aligned}\Delta^{(s)}_{\mathsf{mult}}\left(f_\ell, \mathsf{Enc}^{(s)}_{T_1\times T_2}\left(\widetilde{Q}_\ell\right)\right) &= \Delta^{(s)}_{\mathsf{mult}}\left(f - \mathsf{Enc}^{(s)}\left(\sum_{i=0}^{\ell-1} P_i\,(x_1)\,x_2^{d-i}\right), \mathsf{Enc}^{(s)}_{T_1\times T_2}\left(Q - \sum_{i=0}^{\ell-1} Q_i\,(x_1)\,x_2^{d-i}\right)\right)\\ &= \Delta^{(s)}_{\mathsf{mult}}\left(f, \mathsf{Enc}^{(s)}_{T_1\times T_2}\,(Q)\right)\\ &< \frac{1}{2}n^2\left(s - \frac{d}{n}\right)\end{aligned}$$

With this guarantee in hand, we now proceed with the analysis of the ℓ^{th} iteration.

Now, for every $a \in T_1$, and $i \in \{0, 1, \ldots, r-1\}$ for $r = s - \lfloor\frac{d-\ell}{n}\rfloor$, the function $f_\ell^{(i,a)} \colon T_2 \to \mathbb{F}_{<s-i}[z_2]$ defined as

$$f_\ell^{(i,a)}(b) = \sum_{j\in[s-i]} f_{\ell,(i,j)}(a,b) \cdot z_2^j$$

can be viewed as a received word for a univariate multiplicity code with multiplicity $(s-i)$ and degree $d-\ell$ on the set T_2 of evaluation points. Indeed, if the original received word f had no errors, and was in fact the encoding of Q, then $f_{\ell}^{(i,a)}$ must be equal to the encoding of the univariate polynomial obtained by taking the i^{th} order (Hasse) derivative of $\widetilde{Q}_{\ell}$ with respect to x_1 and setting x_1 to a. Since the degree of Q_{ℓ} in x_2 was at most $d - \ell$, the resulting $G_{\ell}^{(i,a)}$ also has degree at most $d - \ell$.

By combining the output of various calls to Algorithm 1, we obtain the function g_{ℓ} and the weight function w_{ℓ} which together are part of an input to the Weighted Univariate Multiplicity Code Decoder (Algorithm 2). Now Lemma 7.3 shows that $\Delta_{\mathsf{mult}}^{(s)}\left(f_{\ell}, \mathsf{Enc}^{(s)}\left(\widetilde{P}_{\ell}\right)\right) < \frac{1}{2}n^2\left(s - \frac{d}{n}\right)$. This is the promise needed to invoke Theorem 6.6, which states that Algorithm 2 returns $\widetilde{P}_{\ell}$.

Since $\Delta_{\mathsf{mult}}^{(s)}(f_{\ell}, \mathsf{Enc}_{T_1 \times T_2}^{(s)}(\widetilde{P}_{\ell}))$ and $\Delta_{\mathsf{mult}}^{(s)}(f_{\ell}, \mathsf{Enc}_{T_1 \times T_2}^{(s)}(\widetilde{Q}_{\ell}))$ are both upper-bounded by half the minimum distance between polynomials, $\widetilde{P}_{\ell}$ must equal $\widetilde{Q}_{\ell}(x_1)$, which completes the induction step.

The upper bound on the running time follows immediately from the time complexity of Algorithm 1 (Theorem 4.3) and Algorithm 2 (Theorem 6.6). ■

8. Multivariate multiplicity code decoder

In this section, we extend the bivariate decoder (Algorithm 3) constructed in Section 7 to the multivariate setting with $m > 2$. The extension to larger m proceeds as suggested by the inductive proof of the multiplicity SZ Lemma. If we perform the induction following the standard textbook proof of the SZ Lemma (e.g., in [1] and the proof in Kim and Kopparty's work [13]), we need a "weighted multivariate multiplicity code decoder". However, we do not even have a weighted version of the bivariate decoder. We get around this issue by performing a slightly different proof of the SZ Lemma. This alternative proof of the SZ Lemma proceeds by viewing the polynomial as an $(m-1)$-variate polynomial with the coefficients coming from a univariate polynomial ring $\mathbb{F}[x_m]$ instead of as a univariate polynomial in x_m with the coefficients coming from the $(m-1)$-variate polynomial ring $\mathbb{F}[x_1, \ldots, x_{m-1}]$. We first present this alternative proof of the classical SZ Lemma (without multiplicities) in Section 8.1 and in the subsequent sections, extend the bivariate decoder (Algorithm 3) to the multivariate decoder (Algorithm 4).

8.1 Multivariate Schwartz-Zippel Lemma

LEMMA 8.1 (Schwartz-Zippel Lemma). *Let $P(x_1, x_2, \ldots, x_m) \in \mathbb{F}[\mathbf{x}]$ be a non-zero m variate polynomial of total degree at most d and let $T_1, T_2, \ldots, T_m$ be subsets of F of size n each. Then, the number of zeroes of P on the product set $T_1 \times T_2 \times \cdots \times T_m$ is at most dn^{m-1}.*

PROOF. The proof is via induction on m as usual. The base case, where $m = 1$ is clear. For the induction step, we view P as a polynomial in the variables $x_1, x_2, \ldots, x_{m-1}$, with the coefficients being from the polynomial ring $\mathbb{F}[x_m]$.

$$P(\mathbf{x}) = \sum_{i=0}^{\ell} P_i(\mathbf{x}) \,,$$

where the polynomial $P_i(\mathbf{x})$ is homogeneous and degree i when viewed as a polynomial in the variables $x_1, \ldots, x_{m-1}$ with coefficients in $\mathbb{F}[x_m]$. Note that ℓ is *equal* to the total degree of P in $x_1, \ldots, x_{m-1}$ and is at most d, and the degree of P_i in the variable x_m is at most $d - i$. Now, for any setting a_m of x_m in T_m, we consider two cases based on whether $P_\ell(x_1, x_2, \ldots, x_{m-1}, a_m)$ is zero or non-zero.

Let $T'_m \subset T_m$ be the set of $a_m \in T_m$ such that $P_\ell(x_1, x_2, \ldots, x_{m-1}, a_m)$ is identically zero. Viewing $P_\ell(x_1, x_2, \ldots, x_{m-1}, x_m)$ as a univariate polynomial in x_m of degree at most $d - \ell$ with coefficients from $\mathbb{F}(x_1, \ldots, x_{m-1})$, we get that $|T'_m| \leq d - \ell$. For every each $a_m \in T'_m$, the total number of $(a_1, \ldots, a_{m-1}) \in T_1 \times \cdots \times T_{m-1}$ such that $P(a_1, \ldots, a_{m-1}, a_m)$ equals zero is trivially at most n^{m-1}.

On the other hand for every $a_m \in T_m \setminus T'_m$, $P_\ell(x_1, \ldots, x_{m-1}, a_m)$ is not identically zero, and thus $P(x_1, x_2, \ldots, x_{m-1}, a_m)$ is a non-zero $(m-1)$ variate polynomial of degree ℓ. Thus, for each $a_m \in T_m \setminus T'_m$ by the induction hypothesis, the total number of $(a_1, \ldots, a_{m-1}) \in T_1 \times \cdots \times T_{m-1}$ such that $P(a_1, \ldots, a_{m-1}, a_m)$ equals zero is at most $\ell \cdot n^{m-2}$.

Therefore, the total number of zeroes of P on the product set $T_1 \times T_2 \times \cdots \times T_m$ is at most

$$|T'_m| \cdot n^{m-1} + (n - |T'_m|) \cdot \ell \cdot n^{m-2} \leq (|T'_m| + \ell) n^{m-1} \,,$$

which is at most dn^{m-1}, since T'_m is of size at most $d - \ell$. ■

We do remark that the way induction is set up in the above proof is different from the way it proceeds in a typical proof of this lemma, where P is viewed as a univariate polynomial in x_m with the coefficients coming from the $(m-1)$-variate polynomial ring $\mathbb{F}[x_1, \ldots, x_{m-1}]$ (as opposed to being viewed as an $(m-1)$-variate polynomial with the coefficients coming from a univariate polynomial ring $\mathbb{F}[x_m]$.). This subtle difference also shows up in the way induction is done in our decoding algorithm for the multivariate case when compared to how Kim-Kopparty proceed in the decoding algorithm for multivariate Reed-Muller codes [13]. In fact, it is not clear to us that the results in this paper can be obtained if we set up the induction as in the work of Kim and Kopparty [13]. The main technical difficulty is that we do not have an analogue of Algorithm 3 when the received word comes with weights.

8.2 Multivariate decoder: description

We now describe our main algorithm for the multivariate case. For this case, the message space consists of m-variate degree d polynomials. For our algorithm, it will be helpful to think of the decomposition of such a polynomial P as follows. For brevity, we use $\mathbf{e}_{-1} = (e_2, e_3, \ldots, e_m)$ to denote an $m-1$ tuple, whose coordinates are indexed from 2 up to m.

$$P(\mathbf{x}) = \sum_{\ell \in [d+1]} \left(\sum_{\substack{\mathbf{e}_{-1} \in \mathbb{Z}_{\geq 0}^{m-1} \\ |\mathbf{e}_{-1}|_1 = d-\ell}} P_{\ell, \mathbf{e}_{-1}}(x_1) \cdot \prod_{j=2}^{m} x_j^{e_j} \right).$$

Since the total degree of P is at most d, the degree of the univariate polynomial $P_{\ell,\mathbf{e}_{-1}}(x_1)$ is at most ℓ.

Multivariate Multiplicity Code Decoder

To prove the correctness of Algorithm 4 we will generalize the analysis of Algorithm 2. Since, all the proofs are direct extensions of the corresponding proofs in the bivariate case, we have skipped them for the sake of brevity.

8.3 Weighted Univariate Multiplicity Code Decoder (multivariate case)

We give the general versions of the statements in Section 6 for the bivariate case. The proofs are identical, except for the new bounds on distances and weights. Accordingly, n^2 in the distance statements is replaced by n^m, and n in the weight bounds after column decoding is replaced by n^{m-1} (since a column is now replaced by recursive decoding over a grid of lower dimension). There is no change in the matching argument.

8.3.1 Properties of weighted distance

The following is the general definition of the distance Γ between a polynomial and a received word. The only difference from Definition 6.1 is in the weight bounds (n is replaced by n^{m-1}) and in the contribution of each element a, the quantity $n(s-i)$ is replaced by $n^{m-1}(s-i)$.

DEFINITION 8.2. Let $d, \ell, s, m \in \mathbb{N}$ be parameters with $d \geq \ell$, $T \subseteq \mathbb{F}$ be a subset of size n. Let $R \in \mathbb{F}[x]$ be a univariate polynomial of degree at most ℓ, $h\colon T \to \mathbb{F}_{<r}[z]$ and $w\colon T \times [r] \to \mathbb{Z}_{\geq 0}$ be functions such that for every $(a,i) \in T \times [r]$, $w(a,i) \leq \frac{n^{m-1}}{2} \cdot \left((s-i) - \frac{d-\ell}{n}\right)$.

Then, $\Gamma_w^{s,d,\ell}(h,R)$ is defined as follows.

$$\Gamma_w^{s,d,\ell}(h,R) = \left(\sum_{i=0}^{r-1} \sum_{a \in A_i(h,R)} \max \left\{ \left(n^{m-1} \left((s-i) - \frac{d-\ell}{n} \right) - w(a,i) \right), \max_{j<i} w(a,j) \right\} \right) + \sum_{a \in A_r(h,R)} \max_{j<r} w(a,j)$$

where for every $i \in [r+1]$

$$A_i(h,R) = \left\{ a \in T : \max \left\{ j \in [r+1] : h(a) = R(a+z) \mod \langle z \rangle^j \right\} = i \right\}.$$

```
Input:  $m; T_1, T_2, \ldots, T_m \subseteq \mathbb{F}$, $|T_1| = \cdots = |T_m| = n$          ▷ #variables & points of evaluation
        $d, s$                                                             ▷ degree and multiplicity resp.
        $f\colon T_1 \times T_2 \times \cdots \times T_m \to \mathbb{F}_{<s}[z_1, z_2, \ldots, z_m]$.        ▷ received word
Output: $P = \sum_{\ell \in [d+1]} \sum_{\mathbf{e} \in \mathbb{Z}_{\geq 0}^{m-1}, |\mathbf{e}|_1 = d-\ell} P_{\ell,\mathbf{e}}(x_1) \cdot \prod_{j=2}^{m} x_j^{e_j} \in \mathbb{F}_{\leq d}[x_1, x_2, \ldots, x_m]$ such that
        $\Delta_{\text{mult}}^{(s)}\left(f, \text{Enc}^{(s)}(P)\right) < \frac{1}{2} n^m \left(s - \frac{d}{n}\right)$, if such a $P$ exists.
if m = 2 then
    Run Bivariate Multiplicity Code Decoder (Algorithm 3) on $(T_1, T_2, d, s, f)$
    to obtain $P$ ;
else
    Set $\widetilde{T} \leftarrow T_2 \times \cdots \times T_m$ ;
    for $\ell \leftarrow 0$ to $d$ do
        Define $f_\ell\colon T_1 \times \widetilde{T} \to \mathbb{F}_{<s}[\mathbf{z}]$ as          ▷ received word for the $\ell^{th}$ iteration
            $f_\ell \leftarrow f - \text{Enc}^{(s)}\left(\sum_{i \in [\ell]} \left(\sum_{\mathbf{e}_{-1} \in \mathbb{Z}_{\geq 0}^{m-1}, |\mathbf{e}_{-1}|_1 = d-i} P_{\ell,\mathbf{e}_{-1}}(x_1) \cdot \prod_{j=2}^{m} x_j^{e_j}\right)\right)$ ;
        $\forall \mathbf{a} \in T_1 \times \widetilde{T}$, let $f_\ell(\mathbf{a}) = \sum_{\mathbf{i}} f_{\ell,\mathbf{i}}(\mathbf{a}) \mathbf{z}^{\mathbf{i}}$ ;          ▷ some more notation
        Set $r \leftarrow s - \lfloor \frac{d-\ell}{n} \rfloor$ ;          ▷ number of usable derivatives
        for $a \in T_1$ do
            for $i \leftarrow 0$ to $r-1$ do
                Define $f_\ell^{(i,a)}\colon \widetilde{T} \to \mathbb{F}_{<s-i}[z_2, \ldots, z_m]$ as
                    $f_\ell^{(i,a)}(\mathbf{a}_{-1}) = \sum_{\mathbf{i}_{-1} \in [s-i]} f_{\ell,(i,\mathbf{i}_{-1})}(a, \mathbf{a}_{-1}) \cdot \mathbf{z}_{-1}^{\mathbf{i}_{-1}}$, where
                    $\mathbf{z}_{-1} = (z_2, \ldots, z_m)$, $\mathbf{a}_{-1} = (a_2, \ldots, a_m)$ and
                    $\mathbf{i}_{-1} = (i_2, \ldots, i_m)$ ;
                (Recursively) run Multivariate Multiplicity Code Decoder
                (Algorithm 4) on $\left(m-1, \widetilde{T}, d-\ell, s-i, f_\ell^{(i,a)}\right)$ to obtain
                $G_\ell^{(i,a)} \in \mathbb{F}_{\leq d-\ell}[x_2, x_3, \ldots, x_m]$ ;
                if $G_\ell^{(i,a)}$ is $\perp$ then set $G_\ell^{(i,a)} \leftarrow 0$ ;
        Define $w_\ell\colon T_1 \times [r] \to \mathbb{Z}_{\geq 0}$ as
            $w_\ell(a, i) \leftarrow$
             $\min\left\{\Delta_{\text{mult}}^{(s-i)}\left(f_\ell^{(i,a)}, \text{Enc}^{(s-i)}\left(G_\ell^{(i,a)}\right)\right), \frac{n^{m-1}}{2} \cdot \left((s-i) - \frac{d-\ell}{n}\right)\right\}$ ;
        For every $\mathbf{e}_{-1} \in \mathbb{Z}_{\geq 0}^{m-1}, |\mathbf{e}_{-1}|_1 = d-\ell$, define $g_{\ell,\mathbf{e}_{-1}}\colon T_1 \to \mathbb{F}_{<r}[z_1]$ as
            $g_{\ell,\mathbf{e}_{-1}}(a) \leftarrow \sum_{i \in [r]} \text{Coeff}_{\mathbf{x}_{-1}^{\mathbf{e}_{-1}}}\left(G_\ell^{(i,a)}\right) z_1^i$ , where
            $\mathbf{x}_{-1} = (x_2, x_3, \ldots, x_m)$, $\mathbf{e}_{-1} = (e_2, \ldots, e_m)$, and
             $\mathbf{x}_{-1}^{\mathbf{e}_{-1}} = x_2^{e_2} \cdot x_3^{e_3} \cdots x_m^{e_m}$ ;
        For every $\mathbf{e}_{-1} \in \mathbb{Z}_{\geq 0}^{m-1}, |\mathbf{e}_{-1}|_1 = d-\ell$, run Weighted Univariate
        Multiplicity Code Decoder (Algorithm 2) on $(T_1, d, s, m, \ell, r, g_{\ell,\mathbf{e}_{-1}}, w_\ell)$
        to get $P_{\ell,\mathbf{e}_{-1}}$ ;
Set $P(x_1, \ldots, x_m) \leftarrow \sum_{\ell \in [d+1]} \sum_{\mathbf{e} \in \mathbb{Z}_{\geq 0}^{m-1}, |\mathbf{e}|_1 = d-\ell} P_{\ell,\mathbf{e}}(x_1) \cdot \prod_{j=2}^{m} x_j^{e_j}$ ;
if $\Delta_{\text{mult}}^{(s)}\left(f, \text{Enc}^{(s)}(P)\right) < \frac{1}{2} n^m \left(s - \frac{d}{n}\right)$ then  return $P(x_1, x_2, \ldots, x_m)$ else return $\perp$ .
```

Algorithm 4. Multivariate Multiplicity Code Decoder

The following is a triangle-inequality-type statement for the general definition of Γ.

LEMMA 8.3 (Triangle-like inequality for Γ). *Let $d, \ell, s, m \in \mathbb{N}$ be parameters with $d \geq \ell$, $T \subseteq \mathbb{F}$ be a subset of size n. Let $Q, R \in \mathbb{F}[x]$ be univariate polynomials of degree at most ℓ, $h\colon T \to \mathbb{F}_{<r}[z]$ and $w\colon T \times [r] \to \mathbb{Z}_{\geq 0}$ be functions such that for every $(a,i) \in T \times [r]$, $w(a,i) \leq \frac{n^{m-1}}{2} \cdot \left((s-i) - \frac{d-\ell}{n}\right)$.*

If $Q \neq R$, then

$$\Gamma_w^{s,d,\ell}(h,Q) + \Gamma_w^{s,d,\ell}(h,R) \geq n^m \left(s - \frac{d}{n}\right).$$

PROOF SKETCH. The proof is identical to that of Lemma 6.2, with the following one difference: where we count a contribution of the form $(s-i)n - (d-\ell)$, that is replaced by $n^{m-1}((s-i) - (d-\ell)/n)$ (in accordance with the new weight bound). ■

The following is the relationship between the distance we work with, Γ, and the multiplicity distance Δ.

LEMMA 8.4 (Relationship between Γ and multiplicity distance). *Let $T_1, T_2, \ldots, T_m \subseteq \mathbb{F}$ be sets of size n each, and d, ℓ, s, r be natural numbers with $d \geq \ell$ and $r = s - \lfloor \frac{d-\ell}{n} \rfloor$. Let*

$$P = \sum_{i=0}^{d} \sum_{\mathbf{x_{-1}}\colon \textit{monomials of degree } d-i} P_{i,\mathbf{x_{-1}}}(x_1)\mathbf{x_{-1}}$$

be a polynomial of degree at most d with $\Delta^{(s)}_{\mathsf{mult}}(\mathsf{Enc}^{(s)}_{T_1 \times T_2 \times \cdots \times T_m}(P), f) < \frac{1}{2}n^m(s - \frac{d}{n})$. If $g_\ell\colon T_1 \to \mathbb{F}_{<r}[z]$ and $w_\ell\colon T_1 \times [r] \to \mathbb{Z}_{\geq 0}$ are as defined in the main multivariate algorithm, then for every fixed monomial $\mathbf{x_{-1}}$ of degree $d-\ell$,

$$\Gamma_w^{s,d,\ell}(g_\ell, P_{\ell,x_{-1}}) \leq \Delta^{(s)}_{\mathsf{mult}}(f, \mathsf{Enc}^{(s)}_{T_1 \times T_2 \times \cdots \times T_m}(P)) < \frac{1}{2}n^m \left(s - \frac{d}{n}\right).$$

PROOF SKETCH. The proof is identical to that of Lemma 7.3, with instances of $(s-i)n$ being once again replaced by $(s-i)n^{m-1}$. ■

8.3.2 Proof of Theorem 3.9 (correctness of Algorithm 4)

Recall that the algorithm proceeds by trying every possible vector of thresholds $\boldsymbol{\theta}$. The following lemma asserts that one of the thresholds can be used to carry out the decoding.

LEMMA 8.5. *Let g be the received word. If R is such that $\Gamma_w^{d,\ell,s}(g,R) < \frac{n^m}{2}(s - \frac{d}{n})$, then there is a vector of thresholds $\boldsymbol{\theta}$ that can be used to find R.*

PROOF SKETCH. The proof is identical to that of Lemma 6.5: it proceeds by contradiction, assuming that no such vector exists. We start by making a claim on the size of A, the set of points where the received word g agrees with the desired polynomial R.

CLAIM 8.6. *Let g be any received word and R be a degree ℓ polynomial with $\Gamma_w^{d,\ell,s}(g,R) < \frac{n^m}{2}(s - \frac{d}{n})$. Let A be the set of points where g and R agree. Then, $|A| > \ell$.*

Proof sketch. The proof is identical to that of claim 6.7, with instances of $(s-i)n$ being once again replaced by $(s-i)n^{m-1}$. ◆

We then define a good pairing as before, the same as Definition 6.8. Lemma 6.9 and claim 6.12 which assert that such a good pairing exists, remain the same, word for word.

Finally, we use this pairing to compute the error, and show it is more than promised, under the contradiction assumption that no threshold vector works. The statement of claim 6.13 remains the same. In its proof, in the error calculation, as expected $(s-i)n$ is replaced by $(s-i)n^{m-1}$. ■

The remainder of the proof is similar to the proof of Theorem 7.2 presented in Section 7.

Running time: Let $W(m)$ denote the running time of the m-variate decoder. Then, by Theorem 7.2, $W(2) = (sn)^{s+O(1)}$. From the structure of Algorithm 4, $W(m)$ satisfies the following recurrence:

$$W(m) \leq (sn)^{s+O(1)} \cdot \binom{m+s-1}{s} + (sn)^2 \cdot W(m-1).$$

From this we conclude that $W(m)$ is upper bounded by $(sn)^{3m+s+O(1)} \cdot \binom{m+s-1}{s}$.

Acknowledgements

We are thankful to Swastik Kopparty, Ramprasad Saptharishi and Madhu Sudan for many helpful discussions and much encouragement. A special thanks to Madhu for sitting through many long presentations of the preliminary versions of the proofs and his comments.

References

[1] **Sanjeev Arora and Boaz Barak**. Computational Complexity: A Modern Approach. Cambridge University Press, 2009. DOI (47)

[2] **Elwyn R. Berlekamp**. Algebraic Coding Theory. World Scientific, revised edition, 2015. DOI (3)

[3] **Siddharth Bhandari, Prahladh Harsha, Mrinal Kumar, and Ashutosh Shankar**. Algorithmizing the Multiplicity Schwartz-Zippel lemma. *Proc. 34th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 2816–2835, 2023. DOI ePrint (1)

[4] **Siddharth Bhandari, Prahladh Harsha, Mrinal Kumar, and Madhu Sudan**. Decoding multivariate multiplicity codes over product sets. *IEEE Trans. Inform. Theory*, 70(1):154–169, 2024. (Preliminary version in *53rd STOC*, 2021). DOI ePrint (1, 4–6)

[5] **Richard A. DeMillo and Richard J. Lipton**. A probabilistic remark on algebraic program testing. *Inform. Process. Lett.* 7(4):193–195, 1978. DOI (2)

[6] **Zeev Dvir, Swastik Kopparty, Shubhangi Saraf, and Madhu Sudan**. Extensions to the method of multiplicities, with applications to Kakeya sets and mergers. *SIAM J. Comput.* 42(6):2305–2328, 2013. (Preliminary version in *50th FOCS*, 2009). DOI ePrint (1, 2, 16, 17)

[7] **George David Forney Jr.** Concatenated Codes. PhD thesis, Massachusetts Institute of Technology, 1965. URL (5, 23, 24, 26)

[8] **George David Forney Jr.** Generalized minimum distance decoding. *IEEE Trans. Inform. Theory*, 12(2):125–131, 1966. DOI (5, 23, 24, 26)

[9] **Peter Gemmell and Madhu Sudan**. Highly resilient correctors for polynomials. *Inform. Process. Lett.* 43(4):169–174, 1992. DOI (6, 19)

[10] **Venkatesan Guruswami, Atri Rudra, and Madhu Sudan**. Essential coding theory. 2025. (draft of book). URL (12, 24, 26)

[11] **Venkatesan Guruswami and Madhu Sudan**. Improved decoding of Reed-Solomon and algebraic-geometry codes. *IEEE Trans. Inform. Theory*, 45(6):1757–1767, 1999. (Preliminary version in *39th FOCS*, 1998). DOI ePrint (4, 6)

[12] **Larry Guth**. Polynomial Methods in Combinatorics, volume 64 of *University Lecture Series*. Amer. Math. Soc., 2016. URL (2)

[13] **John Y. Kim and Swastik Kopparty**. Decoding Reed-Muller codes over product sets. *Theory Comput.* 13(1):1–38, 2017. (Preliminary version in *31st IEEE Conference on Computational Complexity*, 2016). DOI ePrint (1, 4, 6, 47, 48)

[14] **Swastik Kopparty**. List-decoding multiplicity codes. *Theory of Computing*, 11:149–182, 2015. DOI ePrint (1)

[15] **Swastik Kopparty, Shubhangi Saraf, and Sergey Yekhanin**. High-rate codes with sublinear-time decoding. *J. ACM*, 61(5):28:1–28:20, 2014. (Preliminary version in *43rd STOC*, 2011). DOI ePrint (1, 16)

[16] **Rudolf Lidl and Harald Niederreiter**. Finite Fields, volume 2 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, 2nd edition, 1996. DOI (53)

[17] **László Lovász and Michael D. Plummer**. Matchings in bipartite graphs, *Matching Theory*. Volume 121, North-Holland Mathematics Studies, chapter 1, pages 1–40. North-Holland, 1986. DOI (34)

[18] **James L. Massey**. Shift-register synthesis and BCH decoding. *IEEE Trans. Inform. Theory*, 15(1):122–127, 1969. DOI (3)

[19] **Rasmus Refslund Nielsen**. List decoding of linear block codes. PhD thesis, Technical University of Denmark, 2001. URL (9, 16)

[20] **Øystein Ore**. Über höhere kongruenzen (German) [About higher congruences]. *Norsk Mat. Forenings Skrifter*, 1(7):15, 1922. (see [16, Theorem 6.13]) (2)

[21] **W. Wesley Peterson**. Encoding and error-correction procedures for the Bose-Chaudhuri codes. *IRE Trans. Inf. Theory*, 6(4):459–470, 1960. DOI (3)

[22] **M. Yu Rosenbloom and Michael Anatolévich Tsfasman**. Коды для m-метрики (Russian) [Codes for the m-metric]. *Probl. Peredachi Inf.* 33(1):55–63, 1997. (English translation in *Problems Inform. Transmission*, 33(1):45–52, 1997). URL (16)

[23] **Jacob T. Schwartz**. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, 1980. DOI (2)

[24] **Madhu Sudan**. Decoding of Reed-Solomon codes beyond the error-correction bound. *J. Complexity*, 13(1):180–193, 1997. (Preliminary version in *37th FOCS*, 1996). DOI (4)

[25] **Lloyd R. Welch and Elwyn R. Berlekamp**. Error correction of algebraic block codes. (U.S. Patent 4 633 470). 1986. URL (4, 6, 19)

[26] **Richard Zippel**. Probabilistic algorithms for sparse polynomials. *Proc. International Symp. of Symbolic and Algebraic Computation (EUROSAM)*, volume 72 of *LNCS*, pages 216–226. Springer, 1979. DOI (2)