Electronic Colloquium on Computational Complexity, Report No. 32 (2026)

Advances in List Decoding of Polynomial Codes

Mrinal Kumar[1] and Noga Ron-Zewi[2]

[1]School of Technology and Computer Science, Tata Institute of Fundamental Research,
Email: mrinal.kumar@tifr.res.in
[2]Department of Computer Science, University of Haifa.
Email: noga@cs.haifa.ac.il

Abstract

Error-correcting codes are a method for representing data, so that one can recover the original information even if some parts of it were corrupted. The basic idea, which dates back to the revolutionary work of Shannon and Hamming about a century ago, is to encode the data into a redundant form, so that the original information can be decoded from the redundant encoding even in the presence of some noise or corruption. One prominent family of error-correcting codes are Reed-Solomon Codes which encode the data using evaluations of low-degree polynomials. Nearly six decades after they were introduced, Reed-Solomon Codes, as well as some related families of polynomial-based codes, continue to be widely studied, both from a theoretical perspective and from the point of view of applications.

Besides their obvious use in communication, error-correcting codes such as Reed-Solomon Codes are also useful for various applications in theoretical computer science. These applications often require the ability to cope with many errors, much more than what is possible information-theoretically. List-decodable codes are a special class of error-correcting codes that enable correction from more errors than is traditionally possible by allowing a small list of candidate decodings. These codes have turned out to be extremely useful in various applications across theoretical computer science and coding theory.

In recent years, there have been significant advances in list decoding of Reed-Solomon Codes and related families of polynomial-based codes. This includes efficient list decoding of such codes up to the information-theoretic capacity, with optimal list-size, and using fast nearly-linear time, and even sublinear-time, algorithms. In this book, we survey these developments.

ISSN 1433-8092

Contents

1 Introduction

The problems of communicating and storing data, robustly and efficiently, in the presence of errors or noise is a fundamental problem of interest at the intersection of mathematics and engineering. The theory of error correcting codes (or coding theory) was developed in the landmark works of Shannon [Sha48] and Hamming [Ham50] in the 1940s/50s as an attempt towards formalizing and understanding these problems. The central notion in this theory is that of an *error correcting code* (or simply a *code*). An error correcting code C of *block length* n over a finite *alphabet* Σ is a subset of Σ^n. The words in C are referred to as *codewords*, and typically, we also associate a set M of strings (called *messages*) with the code C such that there is a bijection Φ (called an *encoding map*) from M to C.

The *rate* of a code C is defined as $R(C) := \frac{\log(|C|)}{n \log(|\Sigma|)}$, and it measures the amount of the *redundancy* in the code. Specifically, to use a code to communicate in the presence of noise, the idea is that if we want to communicate a message $m \in M$ over a noisy channel, we instead *encode* it as $\Phi(m)$ and send $\Phi(m)$ over the channel. The rate of the code then measures the number of possible messages one can encode using the code, out of all possible length n strings over Σ. The *minimum (Hamming) distance* of a code C is defined as $\Delta(C) := \min_{c \neq c' \in C} \Delta(c, c')$, where $\Delta(c, c')$, referred to as the *Hamming distance* between c and c', is the number of entries on which c and c' differ. The minimum distance of a code is a measure of how *far* in Hamming distance any two distinct codewords in C are from each other, and consequently, how noise tolerant the code might be. Specifically, the channel might corrupt some of the entries of $\Phi(m)$ during transmission and the receiver might receive a string $w \in \Sigma^n$. However, if w and $\Phi(m)$ differ on less than $\frac{\Delta(C)}{2}$ entries, then $\Phi(m)$ (and hence m) can be recovered uniquely by the decoder by searching for the unique codeword in C that is *closest* to the *received word* w.

The above discussion also highlights the innate tension between the rate and distance of codes - a larger rate appears to force a lower distance (and hence, lower error tolerance). Understanding the precise tradeoff between rate and distance of codes continuous to be a fundamental open problem at the heart of coding theory. In particular, the classical *Singleton Bound* from the 60's states that for a linear code $C \subseteq \Sigma^n$ of size $|C| = \Sigma^k$ it must hold that $\Delta(C) \leq n - k + 1$. This in particular implies that $\delta(C) \leq 1 - R(C)$ for any code C, where $\delta(C) := \frac{\Delta(C)}{n}$ is the *relative distance* of C. Codes satisfying the Singleton bound with equality are called *maximum distance separable* (MDS) codes, and by now we know of various families of codes satisfying this property.

The aforementioned framework of using codes for communication in the presence of noise clearly also highlights *computational questions* of great interest in coding theory, namely that of explicitly constructing families of codes with non-trivial rate and distance and designing efficient encoding and decoding algorithms for these codes. Moreover, such computational aspects are also of great interest to computer science, because of the numerous surprising connections and applications that error-correcting codes have had to areas such as complexity theory, cryptography, pseudorandomness, and algorithm design, which often require explicit and efficiently decodable codes.

1.1 List decoding

If a code C has distance $\Delta(C)$, then clearly, for any word $w \in \Sigma^n$, the number of codewords at Hamming distance less than $\frac{\Delta(C)}{2}$ from w is at most one. Consequently, $\frac{\Delta(C)}{2}$ is called the *unique decoding radius* of the code. The notion of *list decoding*, introduced in the works of Elias [Eli57] and Wozencraft [Woz57] relaxes this error bound to beyond $\frac{\Delta(C)}{2}$. At this distance, the codeword closest to w need not be unique, but we can still hope that the *list* of such codewords is small. More formally, for $\alpha > 0$ and a positive integer

L, a code $C \subseteq \Sigma^n$ is said to be (α, L)-*list decodable* if for every $w \in \Sigma^n$, there are at most L different codewords $c \in C$ such that the Hamming distance $\Delta(w, c)$ is at most αn.

One of the ways to see that this notion of list decoding offers something more than the notion of unique decoding is via a theorem of Johnson, which (informally) shows that *every* code C is (α, L)-list decodable with *constant* list size L as long as the parameter α is less than a function $J(\Delta)$ of the distance Δ of the code. $J(\Delta)$ is referred to as the *Johnson Bound* (or the Johnson Radius) of the code, and it is always as large as $\frac{\Delta}{2}$, but can be much larger. Furthermore, an extension of the aforementioned Singleton Bound to the setting of list decoding shows that any (α, L)-list decodable code must satisfy that $\alpha \leq \frac{L}{L+1}(1 - R)$. Note that the $L = 1$ case corresponds to the classical Singleton Bound which implies that $\alpha \leq \frac{\delta}{2} \leq \frac{1-R}{2}$, but for a growing L, the bound comes close to twice as large. We say that a code achieves *list-decoding capacity* if it approaches this bound, i.e., if it is $(1 - R - \epsilon, O_\epsilon(1))$-list decodable for any $\epsilon > 0$ (We will say that the code achieves list-decoding capacity even if the list is bounded as a function of the block length). Once more, by now we know of several families of capacity-achieving list-decodable codes, which by allowing a small (constant-size) list, can tolerate almost twice as many errors than what is possible in the unique decoding setting!

In the communication setting, list decoding may be useful when the receiver has some *side information* about the transmitted message which enables the receiver to eliminate some of the codewords in the list, and list decodable codes are also a useful building block in the construction of uniquely decodable codes. Furthermore, list decodable codes serve as a *bridge* between the *adversarial error model* described above (pioneered by Hamming [Ham50]) in which an adversary may corrupt an arbitrary subset of the codeword entries, and the *random error model* (pioneered by Shannon [Sha48]) which assumes that each codeword entry is corrupted independently with some fixed probability. It is well known that one can tolerate more (roughly twice as many) errors in the random error model than in the adversarial model, and thus using list-decodable codes, one can handle more errors than in the adversarial unique decoding setting in the presence of both random errors (in which case the list will typically contain a single codeword) and adversarial errors (by allowing a small list). List decodable codes also turned to be highly useful for applications in computer science, like hardness results in complexity theory and explicit construction of various pseudorandom objects like expanders, extractors and pseudorandom generators.

The applications described above highlight two main fundamental questions about list-decodable codes. The first is a *combinatorial question*, where the goal is to bound the list size of various codes for a given decoding radius. The second is a *computational question* about designing explicit list-decodable codes with efficient list decoding algorithms which given a received word, output the list of close-by codewords. A substantial fraction of such results were obtained for families of algebraic error correcting codes based on *low-degree polynomials* (also motivated by the applications described above). Our goal in this survey is to discuss some of these developments, with the focus being on polynomial-based families of codes that we discuss next.

1.2 Polynomials in coding theory

Algebraic techniques, and in particular, the properties of low degree polynomials over finite fields have played an outsized role in the developments in coding theory since the early years of this research area. Indeed, the families of error correcting codes based on the evaluation of low degree polynomials (univariate or multivariate) are among the most well studied families of codes.

The flag bearer of such a code family is undoubtedly the family of *Reed-Solomon Codes*. The codewords of this code correspond to the evaluations of all univariate polynomials of degree less than k with coefficients in a finite field $\mathbb{F}_q$ on a specified set $E \subseteq \mathbb{F}_q$ of n field elements. The distance of these codes stems from the basic algebraic fact that two distinct univariates of degree less than k cannot agree on more than k inputs. It follows that Reed-Solomon Codes are MDS codes.

The mathematical properties of low degree polynomials at the core of Reed-Solomon Codes have found vast generalizations and have led to numerous other fundamental and well studied code families. Some classical examples are *Reed-Muller Codes* whose codewords correspond to evaluations of low degree *multivariate* polynomials over a product set, BCH Codes where the codewords correspond to low-degree univariate polynomials whose evaluations lie in a small subfield, and *Algebraic Geometry* (AG) *codes* where the codewords correspond to evaluations of certain functions on rational points of some algebraic curve.

Besides their elegance and their fundamental mathematical properties, polynomial-based codes as above in many cases achieve the best possible tradeoff between their rate and error-correction radius, in various communication scenarios, and their algebraic structure also lends itself to fast encoding and decoding algorithms. Furthermore, polynomial-based codes such as Reed-Solomon Codes are also widely used in practice due to their good concrete parameters. The algebraic properties of these codes are also often useful for computer science applications. For example, one such useful property is the *multiplication property*, which guarantees that the point-wise multiplication of a pair of codewords (for example, evaluations of a pair of degree k polynomials) is a codeword in another related code (corresponding to evaluations of degree $2k$ polynomials).

In this survey, we will focus on the list-decoding properties of polynomial-based codes, which we discuss next.

1.3 List decoding with polynomial codes

The celebrated work of Sudan [Sud97] and Guruswami-Sudan [GS99] from the 1990s gave efficient (polynomial-time) algorithms for list decoding Reed-Solomon Codes beyond the unique decoding radius, *up to the Johnson Bound*. Note that up to this bound the list size is guaranteed to be constant, so the question is only *computational*, namely, designing efficient list decoding algorithms which given a received word, output the list of close-by codewords. This result led to significant interest in understanding the list decodability of Reed-Solomon Codes *beyond the Johnson Bound*. Beyond this radius, the question is both *combinatorial* and *computational*, namely, showing upper bounds on the list size, and if such upper bounds exist, then also searching for efficient list decoding algorithms. While this continues to be an extremely active area of research with many open research directions, the interest in this problem has been an important driving force behind many other significant developments in this area.

This includes the discovery of variations of Reed-Solomon Codes that were shown to achieve *list decoding capacity*, with a list size that matches the *generalized singleton Bound*, together with efficient list decoding algorithms up to capacity for these codes. Some prominent examples of such codes are *Folded Reed-Solomon Codes*, where one bundles *correlated evaluations* of a low-degree polynomial into a single codeword entry, and *multiplicity codes*, where the encoding also includes evaluations of *derivatives* of a low-degree polynomial.

Recent work in this area has also led to significant advances in the understanding of the *combinatorial* list decodability of Reed-Solomon Codes beyond the Johnson Bound. Specifically, while it was shown that certain Reed-Solomon Codes are *not* list-decodable well beyond the Johnson Bound with a list size that is polynomial in the block length (let alone, a constant list size), it was also shown that Reed-Solomon Codes over *random* evaluation points achieve *list-decoding capacity*, with a list size that matches the *generalized singleton Bound*, with high probability. Finding explicit evaluation points satisfying this property, together with an efficient list-decoding algorithm, constitutes a major open problem in this area by the time of writing of this survey.

A distadvantage of all the aforementioned polynomial-based codes is that their alpahbet is very large (polynomial in the block length). The alphabet size can be brought down to a *constant* by resorting to appropriate variants of AG codes. With regard to efficiency, by now we know of fast *near-linear time* implementation of the list-decoding algorithms mentioned above. Moreover, using *Reed-Muller Codes* and *multivariate multiplicity codes* one can also obtain *local list-decoding* algorithms that are able to list-decode individual codeword entries in *sublinear time*.

1.4 Organization and scope of this survey

The survey is organized as follows:

- In Section 2, we begin by presenting formal definitions and basic facts about list-decodable codes and low-degree polynomials, and introducing some families of polynomial-based codes that will be the focus of this survey.
- In Section 3, we present efficient (polynomial-time) algorithms for list decoding Reed-Solomon Codes up to the Johnson Bound, and we also present combinatorial lower bounds on the list size of Reed-Solomon Codes beyond the Johnson bound.
- In Section 4, we present efficient (polynomial-time) algorithms for list decoding multiplicity codes up to capacity, and we also show that a similar algorithm can be used to list decode Reed-Solomon codes over subfield evaluation points in slightly non-trivial sub-exponential time. These algorithms in particular show that the list size of multiplicity codes at capacity is at most polynomial in the block length, while the list-size of Reed-Solomon codes over subfield evaluation points is at most sub-exponential in the block length.
- In Section 5, we present combinatorial upper bounds on the list size of polynomial-based codes at capacity. Specifically, we show that the list size of Reed-Solomon codes over random evaluation points and multiplicity codes is in fact a constant, independent of the block length (and even matches the generalized Singleton bound), and that the list-size of Reed-Solomon codes over subfield evaluation points can also be reduced to a constant (and the running time of the list decoding algorithm to a polynomial) by passing to an appropriate subcode.
- In Section 6, we present faster near-linear time implementations for some of the list-decoding algorithms presented in the previous sections.
- In Section 7, we present local list decoding algorithms for multivariate polynomial-based codes that are able to decode individual codeword entries in sublinear time.

Some of the material presented in this survey is also covered by other surveys. Specifically, the unique decoding algorithm for Reed-Solomon Codes presented in Section 3.1 is also covered in the online textbook [GRS, Section 12.1] about error-correcting codes, and the list-decoding algorithms for Reed-Solomon Codes presented in Sections 3.2 and 3.3 are also covered in [GRS, Section 12.2], and in Guruswami's survey on algorithmic list decoding [Gur06b, Section 4]. The algorithm presented in Section 4.1 for list decoding of multiplicity codes beyond unique decoding radius follows the presentation of [GRS, Section 17] for the related family of Folded Reed-Solomon Codes. The local correction algorithm for Reed-Muller Codes presented in Section 7.1 is also covered in Yekhanin's survey on locally decodable codes [Yek12, Section 2].

This survey focuses solely on list-decoding properties of polynomial-based codes. Parallel to the time of writing of this survey, it was discovered that certain families of *expander-based codes* have list-decoding properties competitive to those of polynomial-based codes, in particular they can be list-decoded up to capacity with optimal list-size over a constant-size alphabet, and in nearly-linear time [ST25, JMST25].

Finally, we do not discuss in this survey any of the applications of list-decodable codes. Some applications in coding theory are surveyed in [Gur04, Section 11], and applications in theoretical computer science are surveyed in [Sud00], [Tre04, Section 4], [Gur04, Section 12], [Gur06a], [Vad12, Section 5], and [GRS, Sections 20 and 24]. List recovery, a certain generalization of the notion of list-decoding that is useful for applications in coding theory and theoretical computer science, is surveyed in [RV25].

2 Notation and definitions

In this section, we first present some basic notations, and then we provide formal definitions and state some basic facts about list-decodable codes and low-degree polynomials in Sections 2.1 and 2.2, respectively. Finally, in Section 2.3 we introduce some families of polynomial-based codes that will be the focus of this survey.

Notations. For a pair of strings $u, v \in \Sigma^n$, the (Hamming) distance between u and v is the number of entries on which u and v defer, and is denoted $\Delta(u,v) := |\{i \in [n] : u_i \neq v_i\}|$, and the relative (Hamming) distance between u and v is the *fraction* of entries on which u and v differ, and is denoted by $\delta(u,v) := \frac{\Delta(u,v)}{n}$. We say that a string $u \in \Sigma^n$ is α-close (α-far, respectively) to a string $v \in \Sigma^n$ if $\delta(v,u) \leq \alpha$ ($\delta(v,u) > \alpha$, respectively). For $w \in \Sigma^n$ and $\alpha \in (0,1)$, we define the Hamming ball of radius α centered at w as the set $B(w,\alpha) := \{u \in \Sigma^n \mid \delta(u,w) \leq \alpha\}$.

For any prime power q, we denote by $\mathbb{F}_q$ the finite field of q elements. For a field $\mathbb{F}$ and $u \in \mathbb{F}^n$, the (Hamming) weight $|u|$ of u is the number of non-zero entries of u, that is, $|u| := |\{i \in [n] : u_i \neq 0\}|$. Throughout, we let $\mathbb{N} := \{0,1,2,\ldots\}$ denote the set of non-negative integers. For a vector $\mathbf{i} = (i_1, \ldots, i_m) \in \mathbb{N}^m$, the weight $\mathsf{wt}(\mathbf{i})$ of $\mathbf{i}$, equals $\mathsf{wt}(\mathbf{i}) = \sum_{j=1}^m i_j$, and for $\mathbf{i} = (i_1, \ldots, i_m), \mathbf{j} = (j_1, \ldots, j_m) \in \mathbb{N}^m$, we let $\binom{\mathbf{i}}{\mathbf{j}} := \prod_{k=1}^m \binom{i_k}{j_k}$.

2.1 List-decodable codes

Before formally defining list-decodable codes, we start with some basic notation and definitions about error-correcting codes.

An (Error-correcting) code is a subset $C \subseteq \Sigma^n$. We call Σ and n the alphabet and the block length of the code, respectively, and the elements of C are called codewords. The rate of a code $C \subseteq \Sigma^n$ is the ratio $R := \frac{\log(|C|)}{n \cdot \log(|\Sigma|)}$, the (Hamming) distance of C is $\Delta(C) := \min_{c \neq c' \in C} \Delta(c,c')$, and its relative distance is $\delta(C) := \frac{\Delta(C)}{n}$.

We say that a code $C \subseteq \Sigma^n$ is $\mathbb{F}$-linear if $\Sigma = \mathbb{F}^s$ for some field $\mathbb{F}$ and a positive integer s, and C is an $\mathbb{F}$-linear subspace of Σ^n. If $\Sigma = \mathbb{F}$, and C is an $\mathbb{F}$-linear code, then we simply say that C is linear. For an $\mathbb{F}$-linear code $C \subseteq \Sigma^n$, the rate of C equals $\frac{\dim_{\mathbb{F}}(C)}{n \cdot \dim_{\mathbb{F}}(\Sigma)}$, and the distance of C equals $\min_{0 \neq c \in C} |c|$. A generator matrix for a linear code C of dimension k is a (full-rank) matrix $G \in \mathbb{F}^{n \times k}$ so that $\mathrm{Image}(G) = C$, and a parity-check matrix for C is a (full-rank) matrix $H \in \mathbb{F}^{(n-k) \times n}$ so that $\ker(H) = C$. The dual code of C is the code $C^{\perp} \subseteq \mathbb{F}^n$ containing all strings $c' \in \mathbb{F}^n$ satisfying that $\sum_{i=1}^n c'_i \cdot c_i = 0$ for all $c \in C$. It follows by definition that $(C^\perp)^\perp = C$, and that H is a parity-check matrix for C if and only if H^T is a generator matrix for $C^\perp$.

For a code $C \subseteq \Sigma^n$ of relative distance δ, and a given $\alpha < \frac{\delta}{2}$, the problem of (unique) decoding from an α-fraction of errors is the task of finding, given a string $w \in \Sigma^n$, the unique $c \in C$ (if any) which satisfies $\delta(c, w) \leq \alpha$.

List decoding is a paradigm that allows one to correct more than a $\frac{\delta}{2}$ fraction of errors by returning a small list of close-by codewords. More formally, for $\alpha \in (0, 1)$ and a positive integer L we say that a code $C \subseteq \Sigma^n$ is (α, L)-list decodable if for any $w \in \Sigma^n$ there are at most L different codewords $c \in C$ which satisfy that $\delta(c, w) \leq \alpha$. The problem of list-decoding from an α-fraction of errors is the task of finding, given a string $w \in \Sigma^n$, the list of codewords $c \in C$ which satisfy that $\delta(c, w) \leq \alpha$.

The Johnson Bound. The Johnson Bound states that *any* code of relative distance δ is list-decodable from a fraction of $\alpha < 1 - \sqrt{1-\delta}$ errors with a list size which depends on the proximity of α to $1 - \sqrt{1-\delta}$. Note that $1 - \sqrt{1-\delta} \in (\frac{\delta}{2}, \delta)$ for any $\delta \in (0, 1)$, so this bound improves on the unique decoding bound.

Theorem 2.1 (Johnson Bound)**.** *Let $C \subseteq \Sigma^n$ be a code of relative distance δ. Then for any $\alpha < 1 - \sqrt{1-\delta}$, C is (α, L)-list decodable with list size $L = \frac{\delta - \alpha}{(1-\alpha)^2 - (1-\delta)}$.*

Proof. Let $w \in \Sigma^n$ be a string, and let $\mathcal{L} = \{c \in C \mid \delta(c, w) \leq \alpha\}$. Our goal will be to show that $L := \mid \mathcal{L} \mid \leq \frac{\delta - \alpha}{(1-\alpha)^2 - (1-\delta)}$. The proof follows by providing upper and lower bounds on the average agreement between a pair of distinct codewords in the list, given by $\Pr_{c \neq c' \in \mathcal{L}, i \in [n]} [c_i = c'_i]$, and comparing the two bounds.

For the upper bound, note that since C has relative distance δ, we have that $\delta(c, c') \geq \delta$ for any distinct $c, c' \in \mathcal{L}$. Consequently, we have that

$$\Pr_{c \neq c' \in \mathcal{L}, i \in [n]} [c_i = c'_i] \leq 1 - \delta. \tag{1}$$

Next we show a lower bound on the above expression. For $i \in [n]$, let t_i be the number of codewords in $\mathcal{L}$ which agree with w on the i-th entry, that is, $t_i := |\{c \in \mathcal{L} \mid c_i = w_i\}|$. Note that since $\delta(c, w) \leq \alpha$ for any $c \in \mathcal{L}$, we have that $\mathbb{E}_{i \in [n]}[t_i] \geq L(1 - \alpha)$. Consequently, we have that:

$$\begin{aligned} \Pr_{c \neq c' \in \mathcal{L}, i \in [n]} [c_i = c'_i] &\geq \Pr_{c \neq c' \in \mathcal{L}, i \in [n]} [c_i = c'_i = w_i] \\ &\geq \frac{\mathbb{E}_{i \in [n]} \binom{t_i}{2}}{\binom{L}{2}} \geq \frac{\binom{\mathbb{E}_{i \in [n]} t_i}{2}}{\binom{L}{2}} \geq \frac{\binom{L(1-\alpha)}{2}}{\binom{L}{2}}, \end{aligned} \tag{2}$$

where the one before last inequality follows by convexity.

The combination of the bounds given in (1) and (2) gives the following inequality:

$$\binom{L(1-\alpha)}{2} \leq \binom{L}{2} \cdot (1 - \delta),$$

which rearranging gives the desired bound of $L \leq \frac{\delta - \alpha}{(1-\alpha)^2 - (1-\delta)}$.

□

The generalized Singleton Bound. The classical Singleton Bound implies that any code of rate R and relative distance δ must satisfy that $\delta \leq 1 - R$. The generalized Singleton Bound extends this fact by showing that any (α, L)-list decodable code of rate R must satisfy that $\alpha \lesssim \frac{L}{L+1} \cdot (1 - R)$. Note that the classical Singleton Bound for unique decoding corresponds to the special case of $L = 1$, in which case $\alpha \leq \frac{\delta}{2} \leq \frac{1-R}{2}$.

Theorem 2.2 (Generalized Singleton Bound). *Let $C \subseteq \Sigma^n$ be a code of rate R that is (α, L)-list decodable. Then*

$$\alpha \leq \frac{L}{L+1} \cdot (1 - R + o_L(1)),$$

where for a fixed L, the term $o_L(1)$ tends to zero as the block length n tends to infinity.

Proof. Let $b := \frac{L+1}{L} \cdot \alpha n$, for simplicity assume that b is an integer.

We first claim that for any string $u \in \Sigma^{n-b}$, there are at most L distinct codewords in C whose prefix is u. To see this, suppose on the contrary that there exists a string $u \in \Sigma^{n-b}$ and $L+1$ distinct codewords $c_0, c_1, \ldots, c_L \in C$ whose prefix is u. Let $w \in \Sigma^n$ be a string whose first $n - b$ entries agree with u, and for $i = 0, 1, \ldots, L$, w and c_i agree on the i-th subsequent block of length $\frac{b}{L+1}$ (once more, assume that $\frac{b}{L+1}$ is an integer for simplicity). Then by construction for any $i \in \{0, 1, \ldots, L\}$,

$$\delta(w, c_i) \leq \frac{L}{L+1} \cdot \frac{b}{n} = \alpha,$$

which contradicts the assumption that C is (α, L)-list decodable.

Now, since for any string $u \in \Sigma^{n-b}$, there are at most L distinct codewords in C whose prefix is u, we have that $|C| \leq \Sigma^{n-b} \cdot L$, which rearranging and taking logarithms gives

$$\frac{b}{n} \leq 1 - \frac{\log(|C|)}{n \cdot \log(|\Sigma|)} + \frac{\log(L)}{n \cdot \log(|\Sigma|)}.$$

Plugging into the above inequality the values $R = \frac{\log(|C|)}{n \cdot \log(|\Sigma|)}$ and $b = \frac{L+1}{L} \cdot \alpha n$ gives

$$\alpha \leq \frac{L}{L+1} \cdot \left(1 - R + \frac{\log(L)}{n \cdot \log(|\Sigma|)}\right) = \frac{L}{L+1} \cdot (1 - R + o_L(1)),$$

which completes the proof of the theorem. □

The above generalized Singleton Bound in particular implies that (α, L)-list decodable codes must have $\alpha \leq 1 - R - \epsilon$ for some $\epsilon = \epsilon(L)$, which approaches zero as L grows (specifically, $\epsilon = O(\frac{1}{L})$, or equivalently, $L = O(\frac{1}{\epsilon})$). We say that a code $C \subseteq \Sigma^n$ attains list-decoding capacity if it matches this coarser bound, i.e., it is $(1 - R - \epsilon, O_\epsilon(1))$-list decodable for any $\epsilon > 0$[1]

2.2 Polynomials over finite fields

This survey is mainly concerned with algebraic codes, based on low-degree polynomials over finite fields. In what follows, we introduce some definitions and gather several known facts about polynomials over finite fields that will be used for defining these codes and analyzing their list decoding properties.

[1]We will sometimes abuse notation and say that the code achieves list-decoding capacity even if the list is bounded as a function of the block length.

2.2.1 Univariate polynomials

For a field $\mathbb{F}$, we let $\mathbb{F}[X]$ denote the *ring* of univariate polynomials over $\mathbb{F}$, and we let $\mathbb{F}(X)$ denote the *field* of rational functions over $\mathbb{F}$ in the indeterminate X. The degree $\deg(f)$ of a polynomial $f(X) = \sum_{i\in\mathbb{N}} f_i X^i \in \mathbb{F}[X]$ is the maximum i so that $f_i \neq 0$. We let $\mathbb{F}_{<d}[X]$ ($\mathbb{F}_{\leq d}[X]$, respectively) denote the linear space of all univariate polynomials over $\mathbb{F}$ of degree smaller than d (at most d, respectively). A root of a polynomial $f(X) \in \mathbb{F}[X]$ is a point $a \in \mathbb{F}$ so that $f(a) = 0$. It is well-known that a non-zero polynomial $f(X) \in \mathbb{F}[X]$ of degree d has at most d roots in $\mathbb{F}$.

It will sometimes be useful for us to also count roots with *multiplicities*. To define this notion, we first need to define the notion of a (Hasse) derivative, which is a variant of derivatives that is more suitable for finite fields. Let $f(X) \in \mathbb{F}[X]$ be a non-zero univariate polynomial over a field $\mathbb{F}$. For a non-negative integer i, the i'th order (Hasse) derivative $f^{(i)}(X)$ of f is defined as the coefficient of Z^i in the expansion

$$f(X+Z) = \sum_{i\in\mathbb{N}} f^{(i)}(X) Z^i.$$

In particular, for any $a \in \mathbb{F}$, substituting $a, X-a$ into X, Z respectively in the above expression gives the expansion

$$f(X) = \sum_{i\in\mathbb{N}} f^{(i)}(a)(X-a)^i. \tag{3}$$

The above expansion should be compared with the more familiar Taylor Expansion with respect to classical derivatives, where the coefficient of $(X-a)^i$ is divided by an additional factor of $i!$. Eliminating this additional factor of $i!$, which could be zero over a finite field, turns out to be convenient, and tends to make theorem statements cleaner.

The multiplicity of a polynomial $f(X) \in \mathbb{F}[X]$ at a point $a \in \mathbb{F}$, denoted $\mathsf{mult}(f,a)$, is the largest integer s such that for any non-negative integer $i < s$, we have that $f^{(i)}(a) = 0$. Note that $\mathsf{mult}(f,a) \geq 0$ for any $a \in \mathbb{F}$, and that $\mathsf{mult}(f,a) > 0$ for any root a of f. The following fact, which is a direct consequence of the expansion given in (3), generalizes the well-known bound on the number of roots of a low-degree univariate polynomial, taking into account also multiplicities.

Fact 2.3. *Let $f(X) \in \mathbb{F}[X]$ be a non-zero univariate polynomial of degree d. Then*

$$\sum_{a\in\mathbb{F}} \mathsf{mult}(f,a) \leq d.$$

In particular, for any positive integer s, $\mathsf{mult}(f,\mathbf{a}) \geq s$ for at most a $\frac{d}{s|\mathbb{F}|}$-fraction of the points $a \in \mathbb{F}$.

Note that the $s = 1$ case of the 'in particular' part of the above fact corresponds to the standard bound on the number of roots of a low-degree univariate polynomial.

We note the following basic properties of the Hasse Derivative, which follow directly from the definition of the Hasse derivative.

Lemma 2.4. *The following holds for any field $\mathbb{F}$, univariate polynomials $f(X), g(X) \in \mathbb{F}[X]$, and non-negative integers i, j:*

1. *(Linearity)* $(f(X) + g(X))^{(i)} = f^{(i)}(X) + g^{(i)}(X)$.
2. *(Product rule)* $(f(X) \cdot g(X))^{(i)} = \sum_{k=0}^{i} f^{(k)}(X) \cdot g^{(i-k)}(X)$.

3. *(Iterative application)* $(f^{(i)}(X))^{(j)} = \binom{i+j}{i} \cdot f^{(i+j)}(X)$.

Note that, as opposed to classical derivatives, the Hasse derivative is only iterative up to multiplication by an additional binomial coefficient, but on the other hand, the product rule does not require multiplication by an additional binomial coefficient.

Finally, for univariate polynomials $f(X), h(X) \in \mathbb{F}[X]$, we use $(f(X) \mod h(X))$ to denote the unique remainder obtained by dividing $f(X)$ by $h(X)$. We also say that $f(X)$ and $g(X)$ are congruent modulo $h(X)$ (denoted by $(f(X) \equiv g(X) \mod h(X))$) if $(f(X) - g(X))$ is divisible by $h(X)$.

2.2.2 Multivariate polynomials

The notions defined above can also be generalized to the setting of multivariate polynomials. Specifically, for a field $\mathbb{F}$ and a positive integer m, we let $\mathbb{F}[X_1, \ldots, X_m]$ denote the *ring* of m-variate polynomials over $\mathbb{F}$, and we let $\mathbb{F}(X_1, \ldots, X_m)$ denote the *field* of rational functions over $\mathbb{F}$ in the indeterminates $X_1, \ldots, X_m$. For ease of notation, we let $\mathbf{X} := (X_1, X_2, \ldots)$, and throughout the survey we also use the convention that bold letters $\mathbf{a}, \mathbf{b}, \ldots$ denote vectors over $\mathbb{F}$. For $\mathbf{i} = (i_1, \ldots, i_m) \in \mathbb{N}^m$, we let $\mathbf{X}^{\mathbf{i}}$ denote the monomial $\mathbf{X}^{\mathbf{i}} := X_1^{i_1} \cdots X_m^{i_m}$.

The (total) degree $\deg(f)$ of a polynomial $f(\mathbf{X}) = \sum_{\mathbf{i}} f_{\mathbf{i}} \mathbf{X}^{\mathbf{i}} \in \mathbb{F}[\mathbf{X}]$ is the maximum weight $\mathsf{wt}(\mathbf{i}) = \sum_j \mathbf{i}_j$ of $\mathbf{i}$ so that $f_{\mathbf{i}} \neq 0$. As before, we let $\mathbb{F}_{<d}[X_1, \ldots, X_m]$ ($\mathbb{F}_{\leq d}[X_1, \ldots, X_m]$, respectively) denote the linear space of all m-variate polynomials over $\mathbb{F}$ of (total) degree smaller than d (at most d, respectively). In the multivariate setting, it will sometimes be useful for us to also consider a notion of weighted degree. Specifically, for a vector $\mathbf{b} \in \mathbb{N}^m$, the $\mathbf{b}$-weighted degree $\deg_{\mathbf{b}}(f)$ of a polynomial $f(\mathbf{X}) = \sum_{\mathbf{i}} f_{\mathbf{i}} \mathbf{X}^{\mathbf{i}} \in \mathbb{F}[\mathbf{X}]$ is the maximum weighted sum $\sum_j \mathbf{b}_j \cdot \mathbf{i}_j$ of $\mathbf{i}$ so that $f_{\mathbf{i}} \neq 0$.

As in the univariate setting, we define a root of a polynomial $f(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$ as a point $\mathbf{a}$ so that $f(\mathbf{a}) = 0$. The following well-known Schwartz-Zippel Lemma generalizes the well-known bound on the number of roots of a low-degree univariate polynomial to the multivariate setting.

Lemma 2.5 ([Sch80, Zip79, DL78])**.** *Let $f(X_1, \ldots, X_m) \in \mathbb{F}[X_1, \ldots, X_m]$ be a non-zero m-variate polynomial over a finite field $\mathbb{F}$ of (total) degree d. Then f has at most $d \cdot |\mathbb{F}|^{m-1}$ roots in $\mathbb{F}^m$.*

The notion of a Hasse derivative can also be extended to the multivariate setting as follows. Let $f(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$ be a non-zero m-variate polynomial over a field $\mathbb{F}$. For a vector $\mathbf{i} \in \mathbb{N}^m$, we define the $\mathbf{i}$'th (Hasse) derivative $f^{(\mathbf{i})}(\mathbf{X})$ of f as the coefficient of $\mathbf{Z}^{\mathbf{i}}$ in the expansion

$$f(\mathbf{X} + \mathbf{Z}) = \sum_{\mathbf{i} \in \mathbb{N}^m} f^{(\mathbf{i})}(\mathbf{X}) \mathbf{Z}^{\mathbf{i}}.$$

We define the multiplicity of f at a point $\mathbf{a} \in \mathbb{F}^m$, denoted $\mathsf{mult}(f, \mathbf{a})$, to be the largest integer s such that for any $\mathbf{i} \in \mathbb{N}^m$ with $\mathsf{wt}(\mathbf{i}) < s$, we have that $f^{(\mathbf{i})}(\mathbf{a}) = 0$.

The following theorem extends the above Lemma 2.5 to also account for multiplicities of roots.

Theorem 2.6 ([DKSS13], Lemma 2.7)**.** *Let $f(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$ be a non-zero m-variate polynomial over a finite field $\mathbb{F}$ of (total) degree d. Then*

$$\sum_{\mathbf{a} \in \mathbb{F}^m} \mathsf{mult}(f, \mathbf{a}) \leq d \cdot |\mathbb{F}|^{m-1}.$$

In particular, for any positive integer s, $\mathsf{mult}(f, \mathbf{a}) \geq s$ for at most a $\frac{d}{s|\mathbb{F}|}$-fraction of the points $\mathbf{a} \in \mathbb{F}^m$.

Note that the $m = 1$ case of the above theorem corresponds to the above Fact 2.3, while the $s = 1$ case of the 'in particular' part corresponds to the above Lemma 2.5.

We also have the following extension of Lemma 2.4.

Lemma 2.7. *The following holds for any field $\mathbb{F}$, m-variate polynomials $f(\mathbf{X}), g(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$, and $\mathbf{i}, \mathbf{j} \in \mathbb{N}^m$:*

1. *(Linearity)* $(f(\mathbf{X}) + g(\mathbf{X}))^{(\mathbf{i})} = f^{(\mathbf{i})}(\mathbf{X}) + g^{(\mathbf{i})}(\mathbf{X})$.
2. *(Product rule)* $(f(\mathbf{X}) \cdot g(\mathbf{X}))^{(\mathbf{i})} = \sum_{\mathbf{r},\mathbf{k}\,:\,\mathbf{r}+\mathbf{k}=\mathbf{i}} f^{(\mathbf{r})}(\mathbf{X}) \cdot g^{(\mathbf{k})}(\mathbf{X})$.
3. *(Iterative application)* $(f^{(\mathbf{i})}(\mathbf{X}))^{(\mathbf{j})} = \binom{\mathbf{i}+\mathbf{j}}{\mathbf{i}} \cdot f^{(\mathbf{i}+\mathbf{j})}(\mathbf{X})$.

2.3 Some families of polynomial codes

Next we introduce some families of polynomial codes that will be the focus of this survey. All codes are defined over a finite field $\mathbb{F}_q$ of q elements, where q is some prime power. We begin with the most basic family of Reed-Solomon Codes.

Reed-Solomon (RS) Codes. Let $a_1, \ldots, a_n$ be distinct points in $\mathbb{F}_q$ (called evaluation points), and let $k < n$ be a positive integer (the degree parameter). The Reed-Solomon (RS) Code $\mathrm{RS}_q(a_1, \ldots, a_n; k)$ is a code which associates with any univariate polynomial $f(X) \in \mathbb{F}_q[X]$ of degree smaller than k a codeword $(f(a_1), \ldots, f(a_n)) \in \mathbb{F}_q^n$. In the special case where $n = q$, we let $\mathrm{RS}_q(k) := \mathrm{RS}_q(a_1, \ldots, a_n; k)$.

It follows by definition that the Reed-Solomon Code $\mathrm{RS}_q(a_1, \ldots, a_n; k)$ is a linear code of rate $\frac{k}{n}$. Furthermore, since any two distinct univariate polynomials of degree smaller than k can agree on at most $k-1$ points in $\mathbb{F}_q$ (since their difference is a non-zero univariate polynomial of degree smaller than k, and so has at most $k-1$ roots in $\mathbb{F}_q$), it follows that $\mathrm{RS}_q(a_1, \ldots, a_n; k)$ has distance at least $n - k + 1$, and relative distance at least $1 - \frac{k}{n}$. So the Reed-Solomon Code attains the Singleton Bound (and so is an MDS code).

Another family of polynomial codes that will be at the focus of this survey are *multiplicity codes*, which extend Reed-Solomon Codes by also evaluating *derivatives* of polynomials.

Multiplicity codes. As before, let $a_1, \ldots, a_n$ be distinct evaluation points in $\mathbb{F}_q$, and let s be a positive integer (the multiplicity parameter). Setting the multiplicity parameter s to be large allows us to choose a degree parameter that is larger than the block length, specifically, we can choose the degree parameter k to be any positive integer so that $k < sn$. The (univariate) multiplicity code $\mathsf{MULT}_q^{(s)}(a_1, \ldots, a_n; k)$ is defined similarly to the corresponding Reed-Solomon Code $\mathrm{RS}_q(a_1, \ldots, a_n; k)$, except that each codeword of the form $(f(a_1), \ldots, f(a_n))$ is now replaced with the codeword $\left(f^{(<s)}(a_1), \ldots, f^{(<s)}(a_n)\right)$, where for $a \in \mathbb{F}_q$, we use $f^{(<s)}(a) \in \mathbb{F}_q^s$ to denote the vector which includes all (Hasse) derivatives of f at a of order smaller than s, that is, $f^{(<s)}(a) = (f(a), f^{(1)}(a), \ldots, f^{(s-1)}(a))$.

It follows by definition that the multiplicity code $\mathsf{MULT}_q^{(s)}(a_1, \ldots, a_n; k)$ is an $\mathbb{F}_q$-linear code over the alphabet $\mathbb{F}_q^s$ of rate $\frac{k}{sn}$. Furthermore, by Fact 2.3, $\mathsf{MULT}_q^{(s)}(a_1, \ldots, a_n; k)$ has distance at least $n - \frac{k-1}{s}$, and relative distance at least $1 - \frac{k}{sn}$. Finally, note that Reed-Solomon Codes correspond to the special case of multiplicity parameter $s = 1$.

Next we present multivariate extensions of the above codes. The first such extension are the *Reed-Muller Codes* which extend Reed-Solomon codes to the setting of multivariate polynomials.

Reed-Muller (RM) Codes. For simplicity, we only present here the definition of Reed-Muller codes evaluated over the whole field. More specifically, as in the case of Reed-Solomon Codes, let k be a degree parameter so that $k < q$, and let m be a positive integer. The Reed-Muller (RM) Code $\mathrm{RM}_{q,m}(k)$ is a code which associates with any m-variate polynomial $f(X_1, \ldots, X_m) \in \mathbb{F}_q[X_1, \ldots, X_m]$ of degree smaller than k a codeword $(f(\mathbf{a}))_{\mathbf{a}\in\mathbb{F}_q^m}$.

It follows by definition that the Reed-Muller Code $\mathrm{RM}_{q,m}(k)$ is a linear code of block length q^m. Its dimension is the number of monomials in m variables of degree smaller than k which equals $\binom{m+k-1}{m}$, and so this code has rate $\frac{\binom{m+k-1}{m}}{q^m}$. Furthermore, by the Schwartz-Zippel Lemma (Lemma 2.5), $\mathrm{RM}_{q,m}(k)$ has relative distance at least $1 - \frac{k}{q}$. Further note that the $m = 1$ case of the Reed-Muller Code defined above corresponds to the Reed-Solomon code $\mathrm{RS}_q(k)$, evaluated over the whole field.

Finally, we also present an extension of multiplicity codes to the multivariate setting.

Multivariate multiplicity codes. Once more, for simplicity, we only present here the definition of multivariate multiplicity codes evaluated over the whole field. As in the case of univariate multiplicity codes, let k be a degree parameter and let s be a multiplicity parameter so that $k < s \cdot q$, and as in the case of Reed-Muller codes, let m be a positive integer. The multivariate multiplicity code $\mathsf{MULT}_{q,m}^{(s)}(k)$ is defined similarly to the corresponding Reed-Muller Code $\mathrm{RM}_{q,m}(k)$, except that each codeword of the form $(f(\mathbf{a}))_{\mathbf{a}\in\mathbb{F}_q^m}$ is now replaced with the codeword $(f^{(<s)}(\mathbf{a}))_{\mathbf{a}\in\mathbb{F}_q^m}$, where similarly to the univariate multiplicity code case, we let $f^{(<s)}(\mathbf{a})$ denote the vector which includes all m-variate (Hasse) derivatives of f at $\mathbf{a}$ of order smaller than s, that is, $f^{(<s)}(\mathbf{a}) = (f^{(\mathbf{i})}(\mathbf{a}))_{\mathbf{i}\in\mathbb{N}^m \,:\, \mathsf{wt}(\mathbf{i})<s}$. Note that the length of this vector is the number of vectors $\mathbf{i} \in \mathbb{N}^m$ of weight smaller than s which equals $\binom{m+s-1}{m}$.

It follows by definition that the multivariate multiplicity code $\mathsf{MULT}_{q,m}^{(s)}(k)$ is an $\mathbb{F}_q$-linear code over the alphabet $\mathbb{F}_q^{\binom{m+s-1}{m}}$ and of dimension $\binom{m+k-1}{m}$, and so has rate $\frac{\binom{m+k-1}{m}}{\binom{m+s-1}{m}\cdot q^m}$. Furthermore, by Theorem 2.6, $\mathsf{MULT}_{q,m}^{(s)}(k)$ has relative distance at least $1 - \frac{k}{sq}$.

2.4 Bibliographic notes

List decoding. The model of list decoding was first introduced by Elias [Eli57] and Wozencraft [Woz57]. The Johnson Bound is a classical bound in coding theory, and a bound in a similar form to the one stated in Theorem 2.1 appeared in the context of list decoding in [GRS00, Section 4.1]. Our proof of Theorem 2.1 follows the proof of [Gur06b, Theorem 3.1] which is attributed there to Radhakrishnan. The Singleton Bound is another classical bound in coding theory that is attributed to Singleton [Sin64], but also previously appeared in [Jos58]. The Generalized Singleton Bound was stated and proved much more recently by Shangguan and Tamo [ST20], and further improvements and extensions of this bound were given by Roth [Rot22] and by Goldberg, Shangguan and Tamo [GST24].

In this section we only stated and proved simplified *alphabet-independent* versions of both the Johnson Bound and the generalized Singleton Bound, as our focus in this survey is on polynomial-based codes that are defined over a large alphabet, and achieve the best possible list-decoding radius. Alphabet-dependent versions of the Johnson Bound can be found in [Gur06b, Section 3.1]. It is also known that over a q-ary alphabet, the list-decoding capacity (i.e., the best possible list-decoding radius for codes of rate R) is

$H_q^{-1}(1-R)$, where

$$H_q(\alpha) = \alpha \log_q(q-1) + \alpha \log_q\left(\frac{1}{\alpha}\right) + (1-\alpha)\log_q\left(\frac{1}{1-\alpha}\right)$$

denotes the *q-ary entropy function* (see e.g., [Gur06b, Section 3.2]). It can be checked that $H_q^{-1}(1-R)$ approaches $1-R$ for a growing alphabet.

Polynomial-based codes. Theorem 2.6 which bounds the number of roots of a low-degree polynomial, *counted with multiplicities*, was stated and proved by Dvir, Kopparty, Saraf, and Sudan [DKSS13]. This bound extends the classical bound for the number of roots without multiplicities, stated in Lemma 2.5, that was proven independently by Schwartz [Sch80], Zippel [Zip79], and DeMillo and Lipton [DL78].

Reed-Solomon Codes are classical codes that were introduced by Reed and Solomon [RS60], while Reed-Muller Codes were introduced by Muller [Mul54], and a fast decoder for these codes was presented by Reed [Ree54]. The univariate version of multiplicity codes was introduced by Rosenbloom and Tsfasman [RT97], while their multivariate extension was introduced more recently by Kopparty, Saraf, and Yekhanin [KSY14].

3 Algorithmic list decoding up to Johnson Bound

In this section we focus on the most basic family of *Reed-Solomon* (RS) *Codes*, and show how to efficiently list-decode these codes beyond the unique decoding radius, up to the *Johnson Bound* (cf., Theorem 2.1). Note that up to this bound the list size is guaranteed to be constant, so the question is only *computational*, namely, designing efficient list decoding algorithms which given a received word, output the list of close-by codewords.

As a warmup, we first present in Section 3.1 below an efficient (polynomial-time) algorithm for *unique decoding* of Reed-Solomon Codes up to half their minimum distance. Then, in Section 3.2 we present an efficient algorithm for list decoding of Reed-Solomon Codes *beyond* half their minimum distance, while in Section 3.3 we show how to extend this algorithm up to the *Johnson Bound*. Finally, in Section we present combinatorial lower bounds on the list size of Reed-Solomon Codes beyond the Johnson bound, which in particular imply that there is no efficient algorithm which list decodes general Reed-Solomon codes well beyond the Johnson bound.

3.1 Unique decoding of Reed-Solomon Codes

In this section, we first present, as a warmup, an efficient (polynomial time) algorithm for *unique decoding* of Reed-Solomon Codes up to half their minimum distance.

More specifically, fix a prime power q, distinct evaluation points $a_1, \ldots, a_n \in \mathbb{F}_q$, and a positive integer $k < n$. Recall that the *Reed-Solomon* (RS) *Code* $\mathrm{RS}_q(a_1, \ldots, a_n; k)$ is the code which associates with each polynomial $f(X) \in \mathbb{F}_q[X]$ of degree smaller than k a codeword $(f(a_1), \ldots, f(a_n)) \in \mathbb{F}_q^n$. In order to avoid annoying rounding issues, in what follows we shall assume for simplicity that k, n have the same parities. Suppose that $w \in \mathbb{F}_q^n$ is a received word, and suppose that there exists a (unique) polynomial $f(X) \in \mathbb{F}_q[X]$ of degree smaller than k so that $f(a_i) \neq w_i$ for at most $e := \frac{n-k}{2}$ indices $i \in [n]$. That is, $f(a_i) = w_i$ for at least $t := n - e = \frac{n+k}{2}$ indices $i \in [n]$. Below we shall present an algorithm which finds $f(X)$ (and so also the associated codeword $(f(a_1), f(a_2), \ldots, f(a_n))$). In particular, if we let $R := \frac{k}{n}$ denote that rate of the $\mathrm{RS}_q(a_1, \ldots, a_n; k)$ code, then this algorithm will be able to decode from $\frac{1-R}{2}$ fraction of errors.

Overview. As will typically be the case, the decoding algorithm is divided into two main steps. The first step is an *interpolation* step in which we find a non-zero low-degree bivariate polynomial $Q(X,Y) \in \mathbb{F}_q[X,Y]$ so that $Q(a_i, w_i) = 0$ for any $i \in [n]$. The second step is a *root finding* step in which we find a univariate polynomial $g(X) \in \mathbb{F}_q[X]$ satisfying that $Q(X, g(X)) = 0$. To show correctness, we need to prove that such a non-zero low-degree polynomial Q exists, and that $g(X) = f(X)$. We also need to ensure that both steps can be performed efficiently (in time polynomial in n and $\log(q)$). In what follows, we describe separately each of these steps, and analyze their correctness and efficiency.

Step 1: Interpolation. We start by describing the first step of interpolation, in which we would like to find a non-zero low-degree bivariate polynomial $Q(X,Y) \in \mathbb{F}_q[X,Y]$ so that $Q(a_i, w_i) = 0$ for any $i \in [n]$.

How can such a polynomial Q be found? Let

$$E(X) := \prod_{i : f(a_i) \neq w_i} (X - a_i)$$

denote the error-locating polynomial whose roots are all the error-locations, i.e., all evaluation points a_i for which $f(a_i) \neq w_i$.

The main observation is that if we let

$$Q(X,Y) = E(X) \cdot f(X) - E(X) \cdot Y,$$

then for any $i \in [n]$ we have that

$$Q(a_i, w_i) = E(a_i) \cdot f(a_i) - E(a_i) \cdot w_i = 0.$$

To see that the above indeed holds, note that if $f(a_i) = w_i$, then we clearly have that $Q(a_i, w_i) = 0$, and if $f(a_i) \neq w_i$, then $E(a_i) = 0$ by the definition of E, and so we also clearly have that $Q(a_i, w_i) = 0$ in this case.

Motivated by the above, we would like to search for a non-zero bivariate polynomial $Q(X,Y)$ over $\mathbb{F}_q$ of the form

$$Q(X,Y) = A(X) + B(X) \cdot Y,$$

where $A(X)$ has degree at most $e + k - 1 = \frac{n+k}{2} - 1$ and $B(X)$ has degree at most $e = \frac{n-k}{2}$, and which satisfies that $Q(a_i, w_i) = 0$ for any $i \in [n]$. By the above, such a non-zero polynomial $Q(X,Y)$ exists by the choice of $A(X) = E(X) \cdot f(X)$ and $B(X) = -E(X)$. Next we give an alternative linear-algebraic argument for the existence of Q that will later be useful when attempting to extend the unique decoding algorithm to the setting of list decoding.

Lemma 3.1. *There exists a non-zero bivariate polynomial $Q(X,Y)$ over $\mathbb{F}_q$ of the form*

$$Q(X,Y) = A(X) + B(X) \cdot Y,$$

where $A(X)$ has degree at most $\frac{n+k}{2} - 1$ and $B(X)$ has degree at most $\frac{n-k}{2}$, and which satisfies that $Q(a_i, w_i) = 0$ for any $i \in [n]$.

Furthermore, such a polynomial $Q(X,Y)$ can be found by solving a system of linear equations which is obtained by viewing the coefficients of the monomials in $A(X)$ and $B(X)$ as unknowns, and viewing each requirement of the form $Q(a_i, w_i) = A(a_i) + B(a_i) \cdot w_i = 0$ as a homogeneous linear constraint on these unknowns.

Proof. If we view the coefficients of the monomials in $A(X)$ and $B(X)$ as unknowns, then each requirement of the form $Q(a_i, w_i) = A(a_i)+B(a_i)\cdot w_i = 0$ imposes a homogeneous linear constraint on these unknowns. Further note that the number of unknowns is $(\deg(A) + 1) + (\deg(B) + 1) = n + 1$, while the number of linear constraints is n, and consequently this linear system has a non-zero solution. □

Step 2: Root finding. In this step, we would like to find a univariate polynomial $g(X) \in \mathbb{F}_q[X]$ satisfying that $Q(X, g(X)) = 0$, and we would like to show that $g(X) = f(X)$. To this end, we first show that $Q(X, f(X)) = 0$. This will follow by showing that the *univariate* polynomial $P_f(X) := Q(X, f(X))$ has more roots than its degree.

Lemma 3.2. *Suppose that $Q(X, Y)$ is a non-zero bivariate polynomial over $\mathbb{F}_q$ of the form*

$$Q(X, Y) = A(X) + B(X) \cdot Y,$$

where $A(X)$ has degree at most $\frac{n+k}{2} - 1$ and $B(X)$ has degree at most $\frac{n-k}{2}$, and which satisfies that $Q(a_i, w_i) = 0$ for any $i \in [n]$. Let $f(X) \in \mathbb{F}_q[X]$ be a univariate polynomial of degree smaller than k so that $f(a_i) = w_i$ for at least $\frac{n+k}{2}$ indices $i \in [n]$. Then $Q(X, f(X)) = 0$.

Proof. The polynomial $P_f(X) := Q(X, f(X)) \in \mathbb{F}_q[X]$ is a univariate polynomial of degree at most $\frac{n+k}{2} - 1$, which satisfies that $P_f(a_i) = Q(a_i, f(a_i)) = Q(a_i, w_i) = 0$ for any $i \in [n]$ for which $f(a_i) = w_i$. Thus $P_f(X)$ is a univariate polynomial of degree at most $\frac{n+k}{2} - 1$ with at least $\frac{n+k}{2}$ roots, and so it must be the zero polynomial. □

The next lemma shows that $f(X)$ is the unique polynomial satisfying that $Q(X, f(X)) = 0$.

Lemma 3.3. *Suppose that $Q(X, Y)$ is a non-zero bivariate polynomial over $\mathbb{F}_q$ of the form*

$$Q(X, Y) = A(X) + B(X) \cdot Y.$$

Then there exists at most one univariate polynomial $f(X) \in \mathbb{F}_q[X]$ so that $Q(X, f(X)) = 0$. Furthermore, if $Q(X, f(X)) = 0$, then $B(X) \neq 0$ and $f(X) = -\frac{A(X)}{B(X)}$.

Proof. Assume that $Q(X, f(X)) = 0$ for some univariate polynomial $f(X) \in \mathbb{F}_q[X]$. We first show that $B(X) \neq 0$. Suppose on the contrary that $B(X) = 0$. Then by our assumption, we also have that $0 = Q(X, f(X)) = A(X)$, which implies in turn that $Q(X, Y) = A(X) + B(X) \cdot Y = 0$, contradicting the assumption that $Q(X, Y)$ is non-zero. By assumption that $0 = Q(X, f(X)) = A(X) + B(X) \cdot f(X)$, this implies in turn that $f(X) = -\frac{A(X)}{B(X)}$, and so $f(X)$ is uniquely determined. In particular, there exists at most one univariate polynomial $f(X) \in \mathbb{F}_q[X]$ so that $Q(X, f(X)) = 0$. □

The full description of the algorithm appears in Figure 1 below, followed by correctness and efficiency analysis.

Correctness. Suppose that $f(X) \in \mathbb{F}_q[X]$ is a univariate polynomial of degree smaller than k so that $f(a_i) = w_i$ for at least $t := \frac{n+k}{2}$ indices $i \in [n]$. By Lemma 3.1, there exists a non-zero bivariate polynomial $Q(X, Y)$ satisfying the requirements on Step 1. By Lemma 3.2, we have that $Q(X, f(X)) = 0$, and by Lemma 3.3 this implies in turn that $B(X) \neq 0$ and $f(X) = -\frac{A(X)}{B(X)}$, and so the algorithm will compute the correct $f(X)$ on Step 2. Clearly, $f(X)$ will also pass the checks on Step 3, and so the algorithm will return $f(X)$ as required.

Unique decoding of $\mathrm{RS}_q(a_1, \ldots, a_n; k)$:

$\triangleright$ For simplicity assume that n, k have the same parities.

- **INPUT:** $w \in \mathbb{F}_q^n$.
- **OUTPUT:** A polynomial $f(X) \in \mathbb{F}_q[X]$ of degree smaller than k so that $f(a_i) = w_i$ for at least $\frac{n+k}{2}$ indices $i \in [n]$ if such a polynomial exists, or $\perp$ otherwise.

1. Find a non-zero bivariate polynomial $Q(X, Y)$ over $\mathbb{F}_q$ of the form
$$Q(X, Y) = A(X) + B(X) \cdot Y,$$
where $A(X)$ has degree at most $\frac{n+k}{2} - 1$ and $B(X)$ has degree at most $\frac{n-k}{2}$, and which satisfies that $Q(a_i, w_i) = 0$ for any $i \in [n]$.
Such a polynomial $Q(X, Y)$ can be found by solving a system of linear equations which is obtained by viewing the coefficients of the monomials in $A(X)$ and $B(X)$ as unknowns, and viewing each requirement of the form $Q(a_i, w_i) = A(a_i) + B(a_i) \cdot w_i = 0$ as a homogeneous linear constraint on these unknowns.
2. If $B(X) = 0$ return $\perp$; Otherwise, find a univariate polynomial $f(X)$ so that $Q(X, f(X)) = 0$ by letting $f(X) := -\frac{A(X)}{B(X)}$.
3. If $f(X)$ is a polynomial of degree smaller than k so that $f(a_i) = w_i$ for at least $\frac{n+k}{2}$ indices $i \in [n]$, return $(f(a_1), f(a_2), \ldots, f(a_n))$; Otherwise, return $\perp$.

Figure 1: Unique decoding of Reed-Solomon Codes

Efficiency. Step 1 can be performed in time $\mathrm{poly}(n, \log(q))$ by solving a system of n linear equations in $n+1$ variables using Gaussian elimination. Step 2 can be performed in time $\mathrm{poly}(n, \log(q))$ by performing polynomial division using Euclid's algorithm. Step 3 can also clearly be performed in time $\mathrm{poly}(n, \log(q))$.

3.2 List decoding Reed-Solomon Codes beyond unique decoding radius

In this section, we present an efficient algorithm for list decoding of Reed-Solomon Codes *beyond* half their minimum distance.

As before, fix a prime power q, distinct evaluation points $a_1, \ldots, a_n \in \mathbb{F}_q$, and a positive integer $k < n$, and let $\mathrm{RS}_q(a_1, \ldots, a_n; k)$ be the corresponding Reed-Solomon Code. Suppose that $w \in \mathbb{F}_q^n$ is a received word. Our goal is to compute the *list* of all polynomials $f(X) \in \mathbb{F}_q[X]$ of degree smaller than k so that $f(a_i) \neq w_i$ for at most $e := n - \sqrt{2kn} - 1$ indices $i \in [n]$. That is, $f(a_i) = w_i$ for at least $t := n - e = \sqrt{2kn} + 1$ indices $i \in [n]$.

In particular, if we let $R := \frac{k}{n}$ denote the rate of the $\mathrm{RS}_q(a_1, \ldots, a_n; k)$ code, then this algorithm will be able to list-decodable from up to roughly $(1 - \sqrt{2R})$-fraction of errors. It can be verified that this is strictly larger than the unique decoding radius of $\frac{1-R}{2}$ whenever the rate R is smaller than 0.17.

Overview. Recall that in the unique decoding setting, the decoding algorithm was divided into two main steps: The interpolation step in which we searched for a non-zero low-degree bivariate polynomial $Q(X, Y) \in \mathbb{F}_q[X, Y]$ of the form $Q(X, Y) = A(X) + B(X) \cdot Y$ so that $Q(a_i, w_i) = 0$ for any $i \in [n]$, and the root finding step in which we searched for the (unique) univariate polynomial $f(X) \in \mathbb{F}_q[X]$ satisfying that

$Q(X, f(X)) = 0$.

Inspecting the proofs of Lemmas 3.1 and 3.2, we observe that the main properties we needed out of Q were that the number of unknown coefficients in Q is greater than the number n of constraints of the form $Q(a_i, w_i) = 0$, and that the $(1, k)$-*weighted* $\deg_{(1,k)}(Q)$ of Q is smaller than the *agreement parameter* t.[2] The first property guarantees the existence of a non-zero Q. The second property guarantees that for any univariate polynomial $f(X) \in \mathbb{F}_q[X]$ of degree smaller than k so that $f(a_i) = w_i$ for at least t of the indices $i \in [n]$, it holds that $P_f(X) := Q(X, f(X))$ is a univariate polynomial of degree smaller than t that has at least t roots, and consequently $P_f(X) = Q(X, f(X)) = 0$.

In the list-decoding setting, the agreement parameter t is smaller, which means that the $(1, k)$-weighted degree of Q is also required to be smaller. Consequently, Q may potentially have less than n monomials, and so we may not be able to argue the existence of a non-zero polynomial Q. To cope with this, we now no longer require that Q has the special form $Q(X, Y) = A(X) + B(X) \cdot Y$, but we only require that Q has a low $(1, k)$-weighted degree, which can potentially increase the number of monomials in Q. It turns out that this change indeed suffices for setting the agreement parameter t to be significantly smaller than in the unique decoding setting. We also need to ensure that the root finding step can be performed efficiently when Q has this general form.

In what follows, we describe separately the changes made in the interpolation and root finding steps, and analyze their correctness and efficiency.

Step 1: Interpolation. In this step we would now like to find a non-zero bivariate polynomial $Q(X, Y) \in \mathbb{F}_q[X, Y]$ with low $(1, k)$-weighted degree, so that $Q(a_i, w_i) = 0$ for any $i \in [n]$. As opposed to the unique decoding setting, we now do not impose any additional structure on Q beyond the requirement on its weighted degree. The following lemma shows the existence of such a non-zero polynomial Q with $(1, k)$-weighted degree that is significantly smaller than in the unique decoding setting.

Lemma 3.4. *There exists a non-zero bivariate polynomial $Q(X, Y)$ over $\mathbb{F}_q$ with $(1, k)$-weighted degree at most $\sqrt{2kn}$, and which satisfies that $Q(a_i, w_i) = 0$ for any $i \in [n]$. Furthermore, such a polynomial $Q(X, Y)$ can be found by solving a system of linear equations which is obtained by viewing the coefficients of the monomials in $Q(X, Y)$ as unknowns, and viewing each requirement of the form $Q(a_i, w_i) = 0$ as a homogeneous linear constraint on these unknowns.*

Proof. If we view the coefficients of the monomials in Q as unknowns, then each requirement of the form $Q(a_i, w_i) = 0$ imposes a homogeneous linear constraint on these unknowns. It thus suffices to show that the number of monomials in Q is greater than the number of linear constraints n, and so this linear system has a non-zero solution.

Let M denote the number of monomials in $Q(X, Y)$, and assume for simplicity that $\ell := \sqrt{2n/k}$ is an integer. First note that by our requirement that $\deg_{(1,k)}(Q) \leq \ell \cdot k$, the degree of the Y variable in $Q(X, Y)$ can be any $j \in \{0, 1, \ldots, \ell\}$, and for any monomial in which the degree of the Y variable is j, the degree of the X variable can be any $i \in \{0, 1, \ldots, k(\ell - j)\}$. Thus we have that

$$M = \sum_{j=0}^{\ell} (k\ell - kj + 1) = \left(\frac{k\ell}{2} + 1\right) \cdot (\ell + 1) > \frac{k\ell^2}{2} = n.$$

We conclude that the number of unknown coefficients in Q is greater than the number of linear constraints n, and so there exists a non-zero polynomial Q satisfying the requirements of the lemma. $\square$

[2]Recall that for a vector $\mathbf{b} \in \mathbb{N}^m$, the $\mathbf{b}$-*weighted degree* $\deg_{\mathbf{b}}(f)$ of a polynomial $f(\mathbf{X}) = \sum_{\mathbf{i}} f_{\mathbf{i}} \mathbf{X}^{\mathbf{i}} \in \mathbb{F}[\mathbf{X}]$ is the maximum weighted sum $\sum_j \mathbf{b}_j \cdot \mathbf{i}_j$ of $\mathbf{i}$ so that $f_{\mathbf{i}} \neq 0$.

Step 2: Root finding. In this step, we would like to find all univariate polynomials $f(X) \in \mathbb{F}_q[X]$ satisfying that $Q(X, f(X)) = 0$, and we would like to show that any univariate polynomial $f(X) \in \mathbb{F}_q[X]$ of degree smaller than k so that $f(a_i) = w_i$ for at least $t = \sqrt{2kn} + 1$ indices $i \in [n]$ satisfies that $Q(X, f(X)) = 0$.

Lemma 3.5. *Suppose that $Q(X,Y)$ is a non-zero bivariate polynomial over $\mathbb{F}_q$ which satisfies that $\deg_{(1,k)}(Q) \leq \sqrt{2kn}$ and $Q(a_i, w_i) = 0$ for any $i \in [n]$. Let $f(X) \in \mathbb{F}_q[X]$ be a univariate polynomial of degree smaller than k so that $f(a_i) = w_i$ for at least $\sqrt{2kn} + 1$ indices $i \in [n]$. Then $Q(X, f(X)) = 0$.*

Proof. By assumption that $\deg_{(1,k)}(Q) \leq \sqrt{2kn}$, the polynomial $P_f(X) := Q(X, f(X)) \in \mathbb{F}_q[X]$ is a univariate polynomial of degree at most $\sqrt{2kn}$. Additionally, it satisfies that $Q(a_i, f(a_i)) = Q(a_i, w_i) = 0$ for any $i \in [n]$ for which $f(a_i) = w_i$. Thus $P_f(X)$ is a univariate polynomial of degree at most $\sqrt{2kn}$ with at least $\sqrt{2kn} + 1$ roots, and so it must be the zero polynomial. $\square$

In order to ensure that the root finding step can be performed efficiently when Q has a general form, we need to show that there are not too many polynomials $f(X)$ of degree smaller than k satisfying that $Q(X, f(X)) = 0$, and that these polynomials can be found efficiently. To this end we observe the following.

Lemma 3.6. *Suppose that $Q(X,Y)$ is a non-zero bivariate polynomial over $\mathbb{F}_q$, and $f(X) \in \mathbb{F}_q[X]$ is a univariate polynomial so that $Q(X, f(X)) = 0$. Then $Y - f(X)$ divides $Q(X,Y)$ in the bivariate polynomial ring $\mathbb{F}_q[X,Y]$.*

Proof. Let D be the degree of Y in Q. So, we can write Q as $Q(X,Y) = \sum_{i=0}^{D} Q_i(X) Y^i$. Clearly, $Q(X, f(X))$ equals $\sum_{i=0}^{D} Q_i(X)(f(X))^i$. Therefore,

$$Q(X,Y) - Q(X, f(X)) = \sum_{i=1}^{D} Q_i(X)(Y^i - f(X)^i)\,.$$

For every positive integer i, we note that $(Y^i - f(X)^i)$ factors as $(Y - f(X))(Y^{i-1} + Y^{i-2} f(X) + \cdots + Y f(X)^{i-2} + f(X)^{i-1})$. In other words, every term in the sum $\sum_{i=1}^{D} Q_i(X)(Y^i - f(X)^i)$ is divisible by $(Y - f(X))$, and furthermore, the quotient is a polynomial in $\mathbb{F}_q[X,Y]$. Thus, $Q(X,Y) - Q(X, f(X))$ is divisble by $(Y - f(X))$ over $\mathbb{F}_q[X,Y]$. So, if $f(X)$ is such that $Q(X, f(X))$ is zero, then we get that $Q(X,Y)$ is divisble by $(Y - f(X))$ over $\mathbb{F}_q[X,Y]$. $\square$

Thus, in order to find all univariate polynomials $f(X) \in \mathbb{F}_q[X]$ satisfying that $Q(X, f(X)) = 0$, it suffices to factorize $Q(X,Y)$ which can be performed in polynomial time by well known algorithms for bivariate/multivariate polynomial factorization. We summarise the result we need in the following theorem and refer to the cited references and [Gur06b, Section 4.5] for a description of the algorithm.

Theorem 3.7 (Bivariate factorization [Kal85, Len85])**.** *Let $\mathbb{F}$ be any finite field. Then, there is a randomized algorithm that takes as input a bivariate polynomial $Q(X,Y) \in \mathbb{F}[X,Y]$ and outputs all its factors of the form $Y - f(X)$ where $f \in \mathbb{F}[X]$ in time* $\mathrm{poly}(\log(|\mathbb{F}|), \deg(Q))$.

The full description of the algorithm appears in Figure 2 below, followed by correctness and efficiency analysis.

List decoding of $\mathrm{RS}_q(a_1, \ldots, a_n; k)$:

- **INPUT:** $w \in \mathbb{F}_q^n$.
- **OUTPUT:** The list $\mathcal{L}$ of all polynomials $f(X) \in \mathbb{F}_q[X]$ of degree smaller than k so that $f(a_i) = w_i$ for at least $\sqrt{2kn} + 1$ indices $i \in [n]$.

1. Find a non-zero bivariate polynomial $Q(X, Y)$ over $\mathbb{F}_q$ with $(1, k)$-weighted degree at most $\sqrt{2nk}$, and which satisfies that $Q(a_i, w_i) = 0$ for any $i \in [n]$.

   Such a polynomial $Q(X, Y)$ can be found by solving a system of linear equations which is obtained by viewing the coefficients of the monomials in $Q(X, Y)$ as unknowns, and viewing each requirement of the form $Q(a_i, w_i) = 0$ as a homogeneous linear constraint on these unknowns.
2. Find the list $\mathcal{L}'$ of all univariate polynomials $f(X)$ so that $Q(X, f(X)) = 0$ by factoring $Q(X, Y)$ using the algorithm in Theorem 3.7, and including in $\mathcal{L}'$ all univariate polynomials $f(X)$ so that $Y - f(X)$ is a factor of $Q(X, Y)$.
3. Output the list $\mathcal{L}$ of all codewords $(f(a_1), f(a_2), \ldots, f(a_n))$ corresponding to univariate polynomials $f(X) \in \mathcal{L}'$ of degree smaller than k which satisfy that $f(a_i) = w_i$ for at least $\sqrt{2nk} + 1$ indices $i \in [n]$.

Figure 2: List decoding of Reed-Solomon Codes

Correctness. Suppose that $f(X) \in \mathbb{F}_q[X]$ is a univariate polynomial of degree smaller than k so that $f(a_i) = w_i$ for at least $\sqrt{2nk} + 1$ indices $i \in [n]$, we shall show that $f(X) \in \mathcal{L}$. By Lemma 3.4, there exists a non-zero bivariate polynomial $Q(X, Y)$ satisfying the requirements on Step 1. By Lemma 3.5, we have that $Q(X, f(X)) = 0$, and by Lemma 3.6 this implies in turn that $Y - f(X)$ divides $Q(X, Y)$, and so $f(X)$ will be included in $\mathcal{L}'$ on Step 2. Clearly, $f(X)$ will also pass the checks on Step 3, and thus $f(X)$ will also be included in $\mathcal{L}$.

Efficiency. Step 1 can be performed in time $\mathrm{poly}(n, \log(q))$ by solving a system of n linear equations in $O(n)$ variables using Gaussian elimination. Step 2 can be performed in time $\mathrm{poly}(n, \log(q))$ by the algorithm in Theorem 3.7. Step 3 can also clearly be performed in time $\mathrm{poly}(n, \log(q))$.

3.3 List decoding Reed-Solomon Codes up to the Johnson Bound

In this section, we present an efficient algorithm for list decoding of Reed-Solomon Codes *up to the Johnson Bound* (cf., Theorem 2.1).

Once more, fix a prime power q, distinct evaluation points $a_1, \ldots, a_n \in \mathbb{F}_q$, and a positive integer $k < n$, and let $\mathrm{RS}_q(a_1, \ldots, a_n; k)$ be the corresponding Reed-Solomon code. Suppose that $w \in \mathbb{F}_q^n$ is a received word. We shall show how to compute the list of all polynomials $f(X) \in \mathbb{F}_q[X]$ of degree smaller than k so that $f(a_i) \neq w_i$ for at most $e := n - \sqrt{kn} - 1$ indices $i \in [n]$. That is, $f(a_i) = w_i$ for at least $t := n - e = \sqrt{kn} + 1$ indices $i \in [n]$.

In particular, if we let $R := \frac{k}{n}$ denote the rate of the $\mathrm{RS}_q(a_1, \ldots, a_n; k)$ code, then this algorithm will be able to list-decodable from up to roughly $(1 - \sqrt{R})$-fraction of errors. This roughly shaves off a factor of $\sqrt{2}$ from the agreement needed by the algorithm discussed in the previous section and matches the Johnson Bound of Theorem 2.1 (recalling that the relative distance δ of the $\mathrm{RS}_q(a_1, \ldots, a_n; k)$ code satisfies that

$R \leq 1 - \delta$).

Overview. The algorithm builds upon the ideas in the previous algorithms that we saw, and adds one new technical ingredient to the framework, namely, the method of *multiplicities*. To describe this idea, we first briefly recall a high level sketch of the algorithm in the last section.

The algorithm in Figure 2 essentially proceeds by finding a non-zero, low degree (in an appropriate sense) bivariate polynomial $Q(X, Y)$ with the following property: for every polynomial $f(X)$ of degree smaller than k, if the i^{th} entry w_i of the received word satisfies $w_i = f(a_i)$, then the univariate polynomial $P_f(X) := Q(X, f(X))$ has a zero at a_i. Therefore, for an f whose encoding is close to the received word w, the univariate P_f has a *lot* of distinct zeroes and if the number of such agreements exceeds the degree of P_f (which is at most the $(1, k)$-weighted degree of Q), then P_f must be identically zero, or equivalently, $f(X)$ must be a root of Q (when Q is viewed as a univariate polynomial in the indeterminate Y, with coefficients coming from the univariate polynomial ring $\mathbb{F}_q[X]$).

The polynomial Q is obtained by solving an appropriate linear system in its coefficients which guarantees that $Q(a_i, w_i) = 0$ for any i, and we need to set the degree of Q to be not too low in order to guarantee a non-zero solution. On the other hand, it is also crucial that the degree of Q is not to high, since the minimum number of agreements needed for this algorithm to work is essentially one more than the degree of P_f, and hence the error tolerance is higher if Q has low degree. These two requirements, that are in tension with each other, determine the quantitative bounds needed for the algorithm to succeed.

At a high level, the main new idea that makes it possible to tolerate more errors is to obtain a $Q(X, Y)$ in the interpolation step with the stronger property that if a polynomial $f(X)$ and the received word w agree on some entry i, i.e., $f(a_i) = w_i$, then, the univariate polynomial $P_f(X)$ vanishes with *high* multiplicity at a_i, i.e., $P_f^{(j)}(a_i) = 0$ for $j = 0, 1, \ldots, r - 1$ for a sufficiently large parameter r, where $P_f^{(j)}(X)$ denotes the j-th *Hasse Derivative* (cf., Section 2.2). This intuitively seems to indicate that we now need fewer agreements between f and w to ensure that $P_f(X)$ is identically zero. To obtain a non-zero bivariate $Q(X, Y)$ that satisfies this property, the idea is to strengthen the interpolation step and look for a $Q(X, Y)$ that (essentially) vanishes on every point (a_i, w_i) with sufficiently high multiplicity. Since the number of constraints has increased in the process, we now need to work with bivariates $Q(X, Y)$ of somewhat higher $(1, k)$-weighted degree for this system to have a non-zero solution.

Once again, these two steps are in tension with each other - the interpolation step is likely to succeed if the $(1, k)$-weighted degree of Q is not too low, while for $P_f(X)$ to be identically zero for a polynomial $f(X)$ of degree smaller than k, whose encoding is close to w, it helps to have the degree of P_f, which is at most the $(1, k)$-weighted degree of Q, to be low. It is apriori unclear that the quantitative bounds obtained using this approach will be an improvement over the bounds obtained in the algorithm of Figure 2 that was presented in the previous section. However, as we shall see, this is indeed the case, and the new algorithm list decodes all the way up to the Johnson Bound.

We now describe the main steps of the algorithm formally.

Step 1: Interpolation. The following lemma is a high multiplicity variant of Lemma 3.4 from the previous section.

Lemma 3.8. *There exists a non-zero bivariate polynomial $Q(X, Y)$ over $\mathbb{F}_q$ with $(1, k)$-weighted degree at most $\sqrt{kn(r+1)r}$, such that for every $i \in [n]$, Q vanishes with multiplicity at least r on (a_i, w_i). Furthermore, such a polynomial $Q(X, Y)$ can be found by solving a system of linear equations which is obtained by viewing the coefficients of the monomials in $Q(X, Y)$ as unknowns, and viewing each vanishing constraint for a Hasse Derivative of Q as a homogeneous linear constraint on these unknowns.*

Proof. The proof is essentially the same as that of Lemma 3.4, which corresponds to the $r = 1$ case. We view the coefficients of Q as variables, and notice that the property that any Hasse Derivative of Q vanishes at (a_i, w_i) is just a homogeneous linear constraint on the coefficients of Q. Thus, if the parameters are set in a way that the number of unknowns is more than the number of constraints, then this system must have a non-zero solution. Thus, to complete the proof, we just count the number of variables and constraints in the linear system.

The number of Hasse Derivatives of order less than r of a bivariate polynomial is equal to the number of monomials in two variables of total degree at most $(r-1)$, which is at most $\binom{r+1}{2}$. Thus, for every $i \in [n]$, we have $\binom{r+1}{2}$ homogeneous linear constraints, which gives $n\binom{r+1}{2}$ constraints in total. Let D be a parameter (to be set soon), which denotes the $(1, k)$-weighted degree of $Q(X, Y)$. If D is large enough to ensure that the number of bivariate monomials of $(1, k)$-weighted degree at most D exceeds $n\binom{r+1}{2}$, then the above homogeneous linear system has more variables than constraints, and hence has a non-zero solution. Thus, by the same computation as in Lemma 3.4 (setting $\ell = \frac{D}{k}$), it suffices to choose D such that

$$\frac{(D+2)(D+k)}{2k} > n\binom{r+1}{2}.$$

Setting D to be equal to $\sqrt{kn(r+1)r}$ (which we assume is an integer for simplicity) satisfies this constraint. $\square$

Step 2: Root finding. We now argue that any polynomial $f(X)$ of degree smaller than k whose encoding has large agreement with the received word w satisfies that $Q(X, f(X))$ is identically zero. To this end, we first observe that if $Q(X, Y)$ vanishes on a point $(a_i, f(a_i))$ with multiplicity at least r, then $P_f(X) := Q(X, f(X))$ vanishes on a_i with multiplicity at least r.

Claim 3.9. *Let $Q(X, Y) \in \mathbb{F}_q[X, Y]$ and $f(X) \in \mathbb{F}_q[X]$ be polynomials, and let $P_f(X) := Q(X, f(X))$. Suppose that $Q(X, Y)$ vanishes on a point $(a, f(a))$ for some $a \in \mathbb{F}_q$ with multiplicity at least r. Then $P_f(X)$ vanishes on a with multiplicity at least r.*

Proof. The statement follows by a standard application of the chain rule for Hasse Derivatives. In more detail, recall from the definition of Hasse Derivatives that

$$f(a+Z) = \sum_{i=0}^{k-1} f^{(i)}(a)Z^i = f(a) + Z \cdot \tilde{f}(Z),$$

for some univariate polynomial $\tilde{f}(Z)$. From the definition of Hasse Derivatives, we also get

$$P_f(a+Z) = Q(a+Z, f(a) + Z \cdot \tilde{f}(Z)) = \sum_{u,v} Q^{(u,v)}(a, f(a))Z^{u+v}\tilde{f}(Z)^v.$$

Now, by assumption that all Hasse Derivatives of order less than r of Q are zero at $(a, f(a))$, we have that

$$P_f(a+Z) = \sum_{u,v \in \mathbb{Z}_{\geq 0}: u+v \geq r} Q^{(u,v)}(a, f(a))Z^{u+v}\tilde{f}(Z)^v.$$

Thus, the lowest degree of a monomial in Z that can possibly have a non-zero coefficient in $P_f(a+Z)$ is at least r, and hence, by the definition of multiplicity, we get that the multiplicity of P_f at a is at least r. $\square$

The following lemma is analogous to Lemma 3.5 from the previous section.

Lemma 3.10. *Suppose that $Q(X,Y)$ is a non-zero bivariate polynomial over $\mathbb{F}_q$ which satisfies that $\deg_{(1,k)}(Q) \leq \sqrt{kn(r+1)r}$ for $r = 2kn$. Suppose furthermore that Q vanishes on each point (a_i, w_i) for $i \in [n]$ with multiplicity at least r. Let $f(X) \in \mathbb{F}_q[X]$ be a univariate polynomial of degree smaller than k so that $f(a_i) = w_i$ for at least $\sqrt{nk}+1$ indices $i \in [n]$. Then $Q(X, f(X)) = 0$.*

Proof. From the bound on the $(1,k)$-weighted degree of Q, we have that $P_f(X) := Q(X, f(X))$ is a univariate polynomial of degree at most $D := \sqrt{kn(r+1)r}$. From Claim 3.9, we know that for any $i \in [n]$, if $f(a_i) = w_i$, then P_f has a zero of multiplicity at least r at a_i. Thus, if f and w agree on greater than $\frac{D}{r} = \frac{\sqrt{kn(r+1)r}}{r} = \sqrt{nk(1+\frac{1}{r})}$ many distinct a_i's, we have from Fact 2.3 that P_f must be identically zero. For sufficiently large r, e.g. $r \geq kn$, we have that $\sqrt{nk(1+\frac{1}{r})} < \sqrt{nk}+1$. Thus, by picking large enough r (for example, $r = 2kn$), we get that more than $\sqrt{nk}$ agreements suffice for P_f to be identically zero. □

Thus, as in the previous section, we get that any univariate polynomial of degree smaller than k with large agreement with w is contained as a root of $Q(X,Y)$ and can be recovered using Theorem 3.7. Once again, the time complexity of the list decoding algorithm is polynomially bounded in the input size, since the main steps just involve solving a linear system of polynomially bounded size and a special case of bivariate polynomial factorization. The full description of the algorithm is given in Figure 9 below.

List decoding up to Johnson Bound of $\mathrm{RS}_q(a_1, \ldots, a_n; k)$:

▷ Let $r := 2nk$.

- **INPUT:** $w \in \mathbb{F}_q^n$.
- **OUTPUT:** The list $\mathcal{L}$ of all polynomials $f(X) \in \mathbb{F}_q[X]$ of degree smaller than k so that $f(a_i) = w_i$ for at least $\sqrt{nk}+1$ indices $i \in [n]$.

1. Find a non-zero bivariate polynomial $Q(X,Y)$ over $\mathbb{F}_q$ with $(1,k)$-weighted degree at most $\sqrt{nk(r+1)r}$, and which satisfies that $Q^{(u,v)}(a_i, w_i) = 0$ for any $i \in [n]$, and for any $u, v \in \mathbb{N}$ with $u + v < r$.

   Such a polynomial $Q(X,Y)$ can be found by solving a system of linear equations which is obtained by viewing the coefficients of the monomials in $Q(X,Y)$ as unknowns, and viewing each requirement of the form $Q^{(u,v)}(a_i, w_i) = 0$ as a homogeneous linear constraint on these unknowns.
2. Find the list $\mathcal{L}'$ of all univariate polynomials $f(X)$ so that $Q(X, f(X)) = 0$ by factoring $Q(X,Y)$ using the algorithm in Theorem 3.7, and including in $\mathcal{L}'$ all univariate polynomials $f(X)$ so that $Y - f(X)$ is a factor of $Q(X,Y)$.
3. Output the list $\mathcal{L}$ of all codewords $(f(a_1), f(a_2), \ldots, f(a_n))$ corresponding to univariate polynomials $f(X) \in \mathcal{L}'$ of degree smaller than k which satisfy that $f(a_i) = w_i$ for at least $\sqrt{nk}+1$ indices $i \in [n]$.

Figure 3: List decoding of Reed-Solomon Codes up to Johnson Bound

3.4 Limitations on list decoding of Reed-Solomon Codes beyond the Johnson bound

In the previous section, we showed that Reed-Solomon codes are efficiently list-decodable up to the Johnson Bound. That is, we presented an algorithm that efficiently outputs all polynomials of degree smaller than k

whose encoding agrees with the received word on at least $\sqrt{R}n$ points. In this section, we shall present a combinatorial lower bound on the list size of Reed-Solomon Codes, which shows that Reed-Solomon Codes are generally *not* list-decodable from a slightly smaller agreement of $R^{\frac{1}{2}+\epsilon}n$ with a list-size that is polynomial of n. This implies in turn that there is no polynomial-time algorithm that can list-decode Reed-Solomon Codes up to this amount of agreement, and in particular, from an agreement arbitrarily close to R. This bound follows as a consequence of the following theorem.

Theorem 3.11. *For any $\gamma \in (0,1)$, $\beta \in (\gamma, \sqrt{\gamma})$, a sufficiently large integer m, and $n = 2^m$, there is a word $w \in (\mathbb{F}_n)^n$, so that there are at least $n^{(\gamma-\beta^2)\log(n)}$ codewords in $\mathrm{RS}_n(n^\gamma)$ that agree with w on at least n^β entries.*

Setting $\gamma := 1-\epsilon$ and $\beta := (\frac{1}{2}-\epsilon)+\gamma(\frac{1}{2}+\epsilon)$ in the above lemma (noting that $\beta \in (\gamma, \sqrt{\gamma})$), shows, for infinitely many n, the existence of a word $w \in (\mathbb{F}_n)^n$, for which there exist a super-polynomial number of codewords in $\mathrm{RS}_n(k)$ that agree with w on at least $n^{\frac{1}{2}-\epsilon}k^{\frac{1}{2}+\epsilon} = R^{\frac{1}{2}+\epsilon}n$ entries, where $k = n^{1-\epsilon}$.

The proof of the above Theorem 3.11 is based on *subspace polynomials*, defined as follows.

Definition 3.12 (Subspace polynomials)**.** *Let $V \subseteq \mathbb{F}_{2^m}$ be an $\mathbb{F}_2$-linear subspace of $\mathbb{F}_{2^m}$. The subspace polynomial for V is defined by $P_V(X) := \prod_{v\in V}(X-v)$.*

The two main properties of subspace polynomials we shall use are that they have relatively many roots (the whole subspace V), and on the other hand, these polynomials are rather sparse, as shown by the following claim.

Claim 3.13. *Let $V \subseteq \mathbb{F}_{2^m}$ be an $\mathbb{F}_2$-linear subspace of $\mathbb{F}_{2^m}$ of dimension d. Then there exist $b_0, b_1, \ldots, b_{d-1} \in \mathbb{F}_{2^m}$ so that $P_V(X) = X^{2^d} + \sum_{i=0}^{d-1} b_i X^{2^i}$.*

Proof. For a non-negative integer i, let $f_i(X) : V \to \mathbb{F}_{2^m}$ denote the $\mathbb{F}_2$-linear function given by $f_i(v) := v^{2^i}$ for $v \in V$. Next observe that the collection of $\mathbb{F}_2$-linear functions from V to $\mathbb{F}_{2^m}$ forms an $\mathbb{F}_{2^m}$-linear subspace of dimension d, and so $f_0, f_1, \ldots, f_d$ must have a non-trivial linear dependency. That is, there exist scalars $b_0, b_1, \ldots, b_d \in \mathbb{F}_{2^m}$, not all zero, so that $P(X) := \sum_{i=0}^{d} b_i f_i(X) = \sum_{i=0}^{d} b_i X^{2^i}$ evaluates to zero on any $v \in V$. Furthermore, since V has 2^d elements, we must have that $\deg(P(X)) = 2^d$, and we may further assume that $P(X)$ is monic. So $P(X)$ and $P_V(X)$ are two monic polynomials of degree 2^d that vanish on all 2^d elements of V, and therefore we must have that $P_V(X) = P(X) = X^{2^d} + \sum_{i=0}^{d-1} b_i X^{2^i}$ (since $P(X) - P_V(X)$ is a polynomial of degree smaller than 2^d with at least 2^d roots, and so must be the zero polynomial). $\square$

We now turn to the proof of the above Theorem 3.11.

Proof of Theorem 3.11. Let $\gamma \in (0,1)$, $\beta \in (\gamma, \sqrt{\gamma})$, let m be a sufficiently large integer. Let $\mathcal{V}$ be the collection of $\mathbb{F}_2$-linear subspaces $V \subseteq \mathbb{F}_{2^m}$ of dimension βm. Then the size of $\mathcal{V}$ is $\prod_{i=0}^{\beta m-1} \frac{2^m-2^i}{2^{\beta m}-2^i} \geq 2^{\beta(1-\beta)m^2}$ (since to choose a subspace V of dimension d in $\mathbb{F}_{2^m}$, we need for each $i = 0, 1, \ldots, d-1$ to choose a vector which is not in the linear span of all previous i vectors, and we also need to divide by the number of options to choose d linearly independent vectors in V). Furthermore, by Claim 3.13, for each $V \in \mathcal{V}$ there exist $b_0, b_1, \ldots, b_{\beta m-1} \in \mathbb{F}_{2^m}$ so that $P_V(X) = X^{2^{\beta m}} + \sum_{i=0}^{\beta m-1} b_i X^{2^i}$.

By the pigeonhole principle, there exist $b_{\gamma m+1}, b_{\gamma m+2}, \ldots, b_{\beta m-1} \in \mathbb{F}_{2^m}$ such that the number of subspace polynomials $P_V(X)$ for $V \in \mathcal{V}$ with their coefficient of X^{2^i} equal to b_i for $i = \gamma m+1, \ldots, \beta m-1$ is at least $2^{\beta(1-\beta)m^2}/2^{(\beta-\gamma)m^2} \geq 2^{(\gamma-\beta^2)m^2}$. Let $\mathcal{V}' \subseteq \mathcal{V}$ denote the collection of subspaces $V \in \mathcal{V}$ for which $P_V(X)$ has this property.

Let $P(X) := \sum_{i=\gamma m+1}^{\beta m-1} b_i X^{2^i}$, and let $w \in (\mathbb{F}_n)^n$ be the evaluation table of $P(X)$ over $\mathbb{F}_{2^m}$, where $n = 2^m$. For $V \in \mathcal{V}'$, let $P'_V(X) = P_V(X) + P(X)$. Then the polynomials $P'_V(X)$ for $V \in \mathcal{V}'$ are a collection of $2^{(\gamma-\beta^2)m^2} = n^{(\gamma-\beta^2)\log(n)}$ distinct polynomials of degree at most $2^{\gamma m} = n^\gamma$. Moreover, for each $V \in \mathcal{V}'$, $P'_V(X)$ agrees with $P(X)$ on all $2^{\beta m} = n^\beta$ elements of V. We conclude that there are at least $n^{(\gamma-\beta^2)\log(n)}$ codewords in $\mathrm{RS}_n(n^\gamma)$ (the evaluations of $P'_V(X)$ over $\mathbb{F}_{2^m}$ for $V \in \mathcal{V}'$) that agree with w on at least n^β entries. □

3.5 Bibliographic notes

The first efficient algorithms for unique decoding of Reed-Solomon codes were given by Peterson [Pet60], Gorenstein and Zierler [GZ61], Berlekamp [Ber68b], and Massey [Mas69]. The unique decoding algorithm for Reed-Solomon Codes described in Section 3.1 is due to Berlekamp and Welch [BW87], with the exposition based on that of Gemmell and Sudan [GS92] (see also [GRS, Section 12.1]). The list-decoding algorithm for Reed-Solomon codes beyond unique decoding radius described in Section 3.2 is due to Sudan [Sud97], and the list-decoding algorithm for Reed-Solomon Codes up to the Johnson Bound is due to Guruswami and Sudan [GS99] (see also [Gur06b, Section 4] and [GRS, Section 12.2]).

The root finding algorithm (Theorem 3.7) used in the algorithms for list decoding Reed-Solomon Codes follows from the more general algorithms known for factorization of bivariate (and multivariate) polynomials, for instance from the works of Kaltofen [Kal85] and that of Lenstra [Len85].

The limitation on list-decoding of Reed-Solomon Codes beyond the Johnson Bound, presented in Section 3.4, was discovered by Ben-Sasson, Kopparty, and Radhakrishnan [BKR10].

4 Algorithmic list decoding up to capacity

In the previous section, we presented efficient (polynomial-time) algorithms for list decoding of Reed-Solomon Codes *up to the Johnson Bound*, and we also showed that Reed-Solomon Codes are generally *not* list-decodable much beyond the Johnson Bound, and in particular, up to capacity.

In this section, we present algorithms for list-decoding variants of Reed-Solomon codes *up to capacity*. Specifically, in Sections 4.1 and 4.2 we present efficient algorithms for list decoding *multiplicity codes* up to capacity, and in Section 4.3 we also show that a similar algorithm can be used to list decode Reed-Solomon codes over subfield evaluation points in slightly non-trivial sub-exponential time. These algorithms in particular show that the list size of multiplicity codes at capacity is at most polynomial in the block length, while the list-size of Reed-Solomon codes over subfield evaluation points is at most sub-exponential in the block length. In Section 5, we shall present combinatorial upper bounds on the list size of these codes, which effectively reduce the list size to a constant, independent of the block length.

4.1 List decoding multiplicity codes beyond unique decoding radius

Next we present efficient (polynomial-time) algorithms for list decoding the family of *multiplicity codes*, which extend Reed-Solomon Codes by also evaluating *derivatives* of polynomials, *up to capacity*. Towards this, we first present in this section, as a warmup, a simple list-decoding algorithm, similar to the *unique decoding* algorithm for Reed-Solomon Codes presented in Section 3.1, that list decodes multiplicity codes *beyond the unique decoding radius*. Then in Section 4.2, we shall show that augmenting this algorithm with multiplicities, similarly to the list-decoding algorithm for Reed-Solomon Codes *up to the Johnson Bound* presented in Section 3.3, leads to an algorithm that list decodes multiplicity codes *up to capacity*.

In what follows, fix a prime power q, distinct evaluation points $a_1, \ldots, a_n \in \mathbb{F}_q$, and positive integers s, k so that $k < sn$. Recall that the *multiplicity code* $\mathsf{MULT}_q^{(s)}(a_1, \ldots, a_n; k)$ is the code which associates with each polynomial $f(X) \in \mathbb{F}_q[X]$ of degree smaller than k a codeword $(f^{(<s)}(a_1), \ldots, f^{(<s)}(a_n)) \in (\mathbb{F}_q^s)^n$, where for $a \in \mathbb{F}_q$, $f^{(<s)}(a) = (f(a), f^{(1)}(a), \ldots, f^{(s-1)}(a))$. For simplicity, in what follows we shall assume that $\max\{k, s\} \leq \mathsf{char}(\mathbb{F}_q)$. Suppose that $w \in (\mathbb{F}_q^s)^n$ is a received word, where $w_i = (w_{i,0}, \ldots, w_{i,s-1}) \in \mathbb{F}_q^s$ for any $i \in [n]$. We shall show an algorithm, running in time roughly q^s, which computes the list of all polynomials $f(X) \in \mathbb{F}_q[X]$ of degree smaller than k so that $f^{(<s)}(a_i) \neq w_i$ for at most $e := \frac{s \cdot (n-k+1)-1}{s+1}$ indices $i \in [n]$. That is, $f^{(<s)}(a_i) = w_i$ for at least $t := n - e = \frac{n+s(k-1)+1}{s+1}$ indices $i \in [n]$.

Note that the special case of $s = 1$ corresponds to the unique decoding algorithm for Reed-Solomon codes presented in Section 3.1. Moreover, if we let $R := \frac{k}{s \cdot n}$ denote the rate of the $\mathsf{mult}_q^{(s)}(a_1, \ldots, a_n; k)$ code, then this algorithm will be able to list decode from up to roughly $\frac{s}{s+1}(1 - s \cdot R)$-fraction of errors in time roughly q^s. It can be verified that this is strictly larger than the unique decoding radius of $\frac{1-R}{2}$ for any $s > 1$ and $R \leq \frac{1}{3s}$. Furthermore, for any constant s and $q = \mathrm{poly}(n)$ the running time (and list size) is polynomial in the block length.

Overview. Recall that in the unique decoding algorithm for Reed-Solomon Codes (Algorithm 1), the decoding algorithm was divided into two main steps: The interpolation step in which we searched for a non-zero low-degree bivariate polynomial $Q(X, Y) \in \mathbb{F}_q[X, Y]$ of the form $Q(X, Y) = A(X) + B(X) \cdot Y$ so that $Q(a_i, w_i) = 0$ for any $i \in [n]$, and the root finding step in which we searched for the univariate polynomial $f(X) \in \mathbb{F}_q[X]$ satisfying that $Q(X, f(X)) = 0$.

Furthermore, the main properties we needed out of Q were that the number of unknown coefficients in Q is greater than the number n of constraints of the form $Q(a_i, w_i) = 0$, and that the $(1, k)$-weighted degree $\deg_{(1,k)}(Q)$ of Q is smaller than the agreement parameter t. The first property guarantees the existence of a non-zero Q, while the second property guarantees that for any univariate polynomial $f(X) \in \mathbb{F}_q[X]$ of degree smaller than k so that $f(a_i) = w_i$ for at least t of the indices $i \in [n]$, it holds that $Q(X, f(X))$ is a univariate polynomial of degree smaller than t that has at least t roots, and consequently $Q(X, f(X)) = 0$.

In the list-decoding setting, we would like the agreement parameter t to be smaller, which means that the $(1, k)$-weighted degree of Q is required to be smaller. Consequently, Q may potentially have less than n monomials, and so we may not be able to argue the existence of a non-zero polynomial Q. In the list-decoding algorithm for Reed-Solomon Codes beyond the unique decoding radius (Algorithm 2), we coped with this by no longer requiring that Q has the form $Q(X, Y) = A(X) + B(X) \cdot Y$ that is linear in Y, which can potentially increase the number of monomials in Q. For list decoding of multiplicity codes beyond unique decoding radius we deal with this differently by setting Q to be a low-degree $(s+1)$-*variate polynomial* $Q(X, Y_0, \ldots, Y_{s-1})$ that is *linear in* $Y_0, \ldots, Y_{s-1}$, which can also potentially increase the number of monomials in Q.

More specifically, in the interpolation step, we search for a non-zero $(s+1)$-variate polynomial $Q(X, Y_0, \ldots, Y_{s-1}) \in \mathbb{F}_q[X, Y_0, \ldots, Y_{s-1}]$ of the form

$$Q(X, Y_0, \ldots, Y_{s-1}) = A(X) + B_0(X) \cdot Y_0 + \cdots + B_{s-1}(X) \cdot Y_{s-1},$$

of $(1, k, \ldots, k)$-weighted degree smaller than t, so that $Q(a_i, w_i) = 0$ for any $i \in [n]$ (where we view $w_i = (w_{i,0}, \ldots, w_{i,s-1}) \in \mathbb{F}_q^s$ as an assignment to s variables in $\mathbb{F}_q$). Then, in the root finding step, we search for all univariate polynomials $f(X) \in \mathbb{F}_q[X]$ satisfying that $Q(X, f(X), f^{(1)}(X), \ldots, f^{(s-1)}(X)) = 0$.

The analysis is similar to the algorithm for unique decoding of Reed-Solomon Codes, but a crucial point exploited in the proof is that there are not too many polynomials $f(X)$ solving the *differential equation*

$Q(X, f(X), f^{(1)}(X), \ldots, f^{(s-1)}(X)) = 0$. In what follows, we describe separately the interpolation and root finding steps, and analyze their correctness and efficiency.

Step 1: Interpolation. In this step, we search for a non-zero low-degree $(s+1)$-variate polynomial $Q(X, Y_0, \ldots, Y_{s-1})$ so that $Q(a_i, w_i) = 0$ for any $i \in [n]$.

Lemma 4.1. *Let $t = \frac{n+s(k-1)+1}{s+1}$. Then there exists a non-zero $(s+1)$-variate polynomial $Q(X, Y_0, \ldots, Y_{s-1})$ over $\mathbb{F}_q$ of the form*

$$Q(X, Y_0, \ldots, Y_{s-1}) = A(X) + B_0(X) \cdot Y_0 + \cdots + B_{s-1}(X) \cdot Y_{s-1},$$

where $A(X)$ has degree smaller than t and each $B_\ell(X)$ has degree smaller than $t-k+1$, and which satisfies that $Q(a_i, w_i) = 0$ for any $i \in [n]$ (where we view $w_i = (w_{i,0}, \ldots, w_{i,s-1}) \in \mathbb{F}_q^s$ as an assignment to s variables in $\mathbb{F}_q$).

Furthermore, such a polynomial $Q(X, Y_0, \ldots, Y_{s-1})$ can be found by solving a system of linear equations which is obtained by viewing the coefficients of the monomials in $A(X), B_0(X), \ldots, B_{s-1}(X)$ as unknowns, and viewing each requirement of the form $Q(a_i, w_i)$ as a homogeneous linear constraint on these unknowns.

Proof. If we view the coefficients of the monomials in $A(X), B_0(X), \ldots, B_{s-1}(X)$ as unknowns, then each requirement of the form $Q(a_i, w_i) = 0$ imposes a homogeneous linear constraint on these unknowns. Further note that the number of unknowns is

$$t + s \cdot (t-k+1) = (s+1) \cdot t - s \cdot (k-1) = n+1,$$

while the number of linear constraints is n, and consequently this linear system has a non-zero solution. $\square$

Step 2: Root finding. In this step, we search for all univariate polynomials $f(X) \in \mathbb{F}_q[X]$ of degree smaller than k that solve the differential equation

$$Q\left(X, f(X), f^{(1)}(X), \ldots, f^{(s-1)}(X)\right) = 0,$$

and we would like to show that any univariate polynomial $f(X) \in \mathbb{F}_q[X]$ of degree smaller than k so that $f^{(<s)}(a_i) = w_i$ for at least t indices $i \in [n]$ satisfies the above differential equation. We start with the latter.

Lemma 4.2. *Suppose that $Q(X, Y_0, \ldots, Y_{s-1})$ is a non-zero $(s+1)$-variate polynomial over $\mathbb{F}_q$ of the form*

$$Q(X, Y_0, \ldots, Y_{s-1}) = A(X) + B_0(X) \cdot Y_0 + \cdots + B_{s-1}(X) \cdot Y_{s-1},$$

where $A(X)$ has degree smaller than t and each $B_\ell(X)$ has degree smaller than $t-k+1$, and which satisfies that $Q(a_i, w_i) = 0$ for any $i \in [n]$ (where we view $w_i = (w_{i,0}, \ldots, w_{i,s-1}) \in \mathbb{F}_q^s$ as an assignment to s variables in $\mathbb{F}_q$). Let $f(X) \in \mathbb{F}_q[X]$ be a univariate polynomial of degree smaller than k so that $f^{(<s)}(a_i) = w_i$ for at least t indices $i \in [n]$. Then

$$Q(X, f(X), f^{(1)}(X), \ldots, f^{(s-1)}(X)) = 0.$$

Proof. The polynomial $P_f(X) := Q(X, f(X), f^{(1)}(X), \ldots, f^{(s-1)}(X))$ is a univariate polynomial of degree smaller than t, which satisfies that

$$P_f(a_i) = Q(a_i, f(a_i), f^{(1)}(a_i), \ldots, f^{(s-1)}(a_i)) = Q(a_i, w_i) = 0$$

for any $i \in [n]$ for which $f^{(<s)}(a_i) = w_i$. Thus $P_f(X)$ is a univariate polynomial of degree smaller than t with at least t roots, and so it must be the zero polynomial. $\square$

The main new ingredient in the list decoding algorithm for multiplicity codes is the next lemma which bounds the number of univariate polynomials $f(X) \in \mathbb{F}_q[X]$ satisfying the differential equation.

Lemma 4.3. *Suppose that $Q(X, Y_0, \ldots, Y_{s-1})$ is a non-zero $(s+1)$-variate polynomial over $\mathbb{F}_q$ of the form*

$$Q(X, Y_0, \ldots, Y_{s-1}) = A(X) + B_0(X) \cdot Y_0 + \cdots + B_{s-1}(X) \cdot Y_{s-1},$$

and let $\mathcal{L}$ be the list containing all polynomials $f(X) \in \mathbb{F}_q[X]$ of degree smaller than k so that

$$Q(X, f(X), f^{(1)}(X), \ldots, f^{(s-1)}(X)) = 0.$$

Then if $\max\{k, s\} \leq \mathsf{char}(\mathbb{F}_q)$, then $|\mathcal{L}| \leq q^{s-1}$.

Furthermore, $\mathcal{L}$ forms an affine subspace over $\mathbb{F}_q$ of dimension at most $s-1$. A basis for this subspace can be found by solving a system of linear equations which is obtained by viewing the coefficients of $f(X)$ as unknowns, and viewing the requirement that each of the coefficients of $Q(X, f(X), f^{(1)}(X), \ldots, f^{(s-1)}(X))$ is zero as a (non-homogeneous) linear constraint on these unknowns.

Proof. First note that if there is no polynomial $f(X) \in \mathbb{F}_q[X]$ satisfying that $Q(X, f(X), f^{(1)}(X), \ldots, f^{(s-1)}(X)) = 0$, then we are done, so we may assume that there exists a polynomial $f(X) \in \mathbb{F}_q[X]$ satisfying that $Q(X, f(X), f^{(1)}(X), \ldots, f^{(s-1)}(X)) = 0$. Next we observe that in this case $B_0(X), \ldots, B_{s-1}(X)$ are not all zero. To see this, suppose on the contrary that $B_0(X), \ldots, B_{s-1}(X)$ are all zero. Then by assumption that $Q(X, f(X), f^{(1)}(X), \ldots, f^{(s-1)}(X)) = A(X) + \sum_{\ell=0}^{s-1} B_\ell(X) \cdot f^{(\ell)}(X) = 0$, we also have that $A(X) = 0$, which implies in turn that $Q(X, Y_0, \ldots, Y_{s-1}) = A(X) + \sum_{\ell=0}^{s-1} B_\ell(X) \cdot Y_\ell = 0$, contradicting the assumption that $Q(X, Y_0, \ldots, Y_{s-1})$ is non-zero. Without loss of generality, we may further assume that $B_{s-1}(X) \neq 0$, otherwise we prove the statement for the maximal value $s' < s$ for which $B_{s'}(X) \neq 0$, which implies the statement also for the value s. Finally, since $B_{s-1}(X)$ is a non-zero polynomial over $\mathbb{F}_q$, there exists an element $a \in \mathbb{F}_q$ so that $B_{s-1}(a) \neq 0$. The main observation is the following.

Claim 4.4. *For any $u \in \mathbb{F}_q^{s-1}$, there is a* unique *polynomial $f(X) \in \mathbb{F}_q[X]$ of degree smaller than k so that $f^{(<s-1)}(a) = u$ and $Q(X, f(X), f^{(1)}(X), \ldots, f^{(s-1)}(X)) = 0$.*

Proof. To show this, fix $u \in \mathbb{F}_q^{s-1}$, and let $f(X) \in \mathbb{F}_q[X]$ be a polynomial of degree smaller than k so that $f^{(<s-1)}(a) = u$ and

$$P_f(X) := Q(X, f(X), f^{(1)}(X), \ldots, f^{(s-1)}(X)) = A(X) + \sum_{\ell=0}^{s-1} B_\ell(X) \cdot f^{(\ell)}(X) = 0.$$

By the definition of the Hasse Derivative, we can write $f(X) = \sum_{i=0}^{k-1} f^{(i)}(a) \cdot (X-a)^i$. It therefore suffices to show that the values of $f^{(<s-1)}(a)$ determine the rest of the derivatives $f^{(s-1)}(a), \ldots, f^{(k-1)}(a)$.

To show the above, we first compute the derivatives of $P_f(X)$. By the properties of the Hasse Derivative (Lemma 2.4), for any $j \in \mathbb{N}$ we have that:

$$\begin{aligned} P_f^{(j)}(X) &= A^{(j)}(X) + \sum_{\ell=0}^{s-1} (B_\ell(X) \cdot f^{(\ell)}(X))^{(j)} \\ &= A^{(j)}(X) + \sum_{\ell=0}^{s-1} \sum_{h=0}^{j} B_\ell^{(j-h)}(X) \cdot (f^{(\ell)})^{(h)}(X) \\ &= A^{(j)}(X) + \sum_{\ell=0}^{s-1} \sum_{h=0}^{j} \binom{h+\ell}{\ell} B_\ell^{(j-h)}(X) \cdot f^{(h+\ell)}(X). \end{aligned} \tag{4}$$

Fix $j \in \{0, 1, \ldots, k-s\}$. Since $P_f(X)$ is the zero polynomial, we have that $P_f^{(j)}(a) = 0$, and so by the above,

$$A^{(j)}(a) + \sum_{\ell=0}^{s-1} \sum_{h=0}^{j} \binom{h+\ell}{\ell} B_\ell^{(j-h)}(a) \cdot f^{(h+\ell)}(a) = 0.$$

Inspecting the above expression, we see that the highest order derivative of f that appears there is $f^{(j+s-1)}(a)$, which is only obtained for the choice of $\ell = s-1$ and $h = j$. Furthermore, the coefficient multiplying $f^{(j+s-1)}(a)$ is $\binom{j+s-1}{s-1} B_{s-1}(a)$, which is non-zero by assumptions that $\max\{k, s\} \leq \mathsf{char}(\mathbb{F}_q)$ and that $B_{s-1}(a) \neq 0$. Consequently, the derivative $f^{(j+s-1)}(a)$ is a linear combination of lower derivatives $f^{(<j+s-1)}(a)$. In particular, $f^{(s-1)}(a), \ldots, f^{(k-1)}(a)$, and so also the polynomial $f(X)$, are uniquely determined by $f^{(<s-1)}(a) = u$. $\square$

By the above claim, we clearly have that $|\mathcal{L}| \leq q^{s-1}$. The moreover part follows by the linear structure of Q. $\square$

The full description of the algorithm appears in Figure 4 below, followed by correctness and efficiency analysis.

List decoding of $\mathsf{mult}_q^{(s)}(a_1, \ldots, a_n; k)$:

- **INPUT:** $w \in (\mathbb{F}_q^s)^n$, where $w_i = (w_{i,0}, \ldots, w_{i,s-1}) \in \mathbb{F}_q^s$ for each $i \in [n]$.
- **OUTPUT:** The list $\mathcal{L}$ of all polynomials $f(X) \in \mathbb{F}_q[X]$ of degree smaller than k so that $f^{(<s)}(a_i) = w_i$ for at least $t = \frac{n+s(k-1)+1}{s+1}$ indices $i \in [n]$.

1. Find a non-zero $(s+1)$-variate polynomial $Q(X, Y_0, \ldots, Y_{s-1})$ over $\mathbb{F}_q$ of the form

$$Q(X, Y_0, \ldots, Y_{s-1}) = A(X) + B_0(X) \cdot Y_0 + \cdots + B_{s-1}(X) \cdot Y_{s-1},$$

   where $A(X)$ has degree smaller than t and each $B_\ell(X)$ has degree smaller than $t-k+1$, and which satisfies that $Q(a_i, w_i) = 0$ for any $i \in [n]$ (where we view $w_i = (w_{i,0}, \ldots, w_{i,s-1}) \in \mathbb{F}_q^s$ as an assignment to s variables in $\mathbb{F}_q$).

   Such a polynomial $Q(X, Y_0, \ldots, Y_{s-1})$ can be found by solving a system of linear equations which is obtained by viewing the coefficients of the monomials in Q as unknowns, and viewing each requirement of the form $Q(a_i, w_i) = 0$ as a homogeneous linear constraint on these unknowns.

2. Find a basis for the list $\mathcal{L}'$ of all univariate polynomials $f(X)$ of degree smaller than k so that

$$Q(X, f(X), f^{(1)}(X), \ldots, f^{(s-1)}(X)) = 0.$$

   This basis can be found by solving a system of linear equations which is obtained by viewing the k coefficients of $f(X)$ as unknowns, and viewing the requirement that each of the t coefficients of $Q(X, f(X), f^{(1)}(X), \ldots, f^{(s-1)}(X))$ is zero as t (non-homogeneous) linear constraints on these unknowns.

3. Output the list $\mathcal{L}$ of all codewords $(f(a_1), f(a_2), \ldots, f(a_n))$ corresponding to univariate polynomials $f(X) \in \mathcal{L}'$ of degree smaller than k which satisfy that $f^{(<s)}(a_i) = w_i$ for at least t indices $i \in [n]$.

Figure 4: List decoding of multiplicity codes

Correctness. Suppose that $f(X) \in \mathbb{F}_q[X]$ is a univariate polynomial of degree smaller than k so that $f^{(<s)}(a_i) = w_i$ for at least t indices $i \in [n]$, we shall show that $f(X) \in \mathcal{L}$. By Lemma 4.1, there exists a non-zero $(s+1)$-variate polynomial $Q(X, Y_0, \ldots, Y_{s-1})$ satisfying the requirements on Step 1. By Lemma 4.2, we have that $Q(X, f(X), f^{(1)}(X), \cdots, f^{(s-1)}(X)) = 0$, and so $f(X)$ will be included in $\mathcal{L}'$ on Step 2. Clearly, $f(X)$ will also pass the checks on Step 3, and thus $f(X)$ will also be included in $\mathcal{L}$.

Efficiency. Step 1 can be performed in time $\mathrm{poly}(n, s, \log(q))$ by solving a system of n linear equations in $n+1$ variables using Gaussian elimination. Step 2 can be performed in time $\mathrm{poly}(n, s, \log(q))$ by solving a system of t linear equations in k variables. By Lemma 4.3, Step 3 can be performed in time $|\mathcal{L}| \cdot \mathrm{poly}(n, s, \log(q)) \leq q^s \cdot \mathrm{poly}(n, s, \log(q))$, where the latter bound is polynomial in the block length for a constant s and $q = \mathrm{poly}(n)$.

4.2 List decoding multiplicity codes up to capacity

We now show how to extend the list decoding algorithm of the previous section to an algorithm that list decodes multiplicity codes up to capacity.

As before, let $\mathsf{mult}_q^{(s)}(a_1, \ldots, a_n; k)$ be the multiplicity code, where q is a prime power, $a_1, a_2, \ldots, a_n$ are distinct points in $\mathbb{F}_q$, and s, k are positive integers so that $k < sn$ and $\max\{k, s\} \leq \mathsf{char}(\mathbb{F}_q)$. Suppose also that $w \in (\mathbb{F}_q^s)^n$ is a received word, where $w_i = (w_{i,0}, \ldots w_{i,s-1}) \in \mathbb{F}_q^s$ for any $i \in [n]$. Let $r \in \{1, 2, \ldots, s\}$ be a parameter to be determined later on. We shall show how to compute the list of all polynomials $f(X) \in \mathbb{F}_q[X]$ of degree smaller than k so that $f^{(<s)}(a_i) \neq w_i$ for at most $e := \frac{(s-r+1)rn - r(k-1) - 1}{(s-r+1)(r+1)}$ indices $i \in [n]$. That is, $f^{(<s)}(a_i) = w_i$ for at least $t := n - e = \frac{(s-r+1)n + r(k-1) + 1}{(s-r+1)(r+1)}$ indices $i \in [n]$.

Note that the algorithm from the previous section corresponds to the special case that $r = s$. Moreover, if we let $R := \frac{k}{s \cdot n}$ denote the rate of the $\mathsf{mult}_q^{(s)}(a_1, \ldots, a_n; k)$ code, then this algorithm will be able to list decode from up to roughly a $\frac{r}{r+1}\left(1 - \frac{s}{s-r+1} \cdot R\right)$-fraction of errors in time roughly q^r. Thus, for any $\epsilon > 0$, we can choose $r = \Theta(1/\epsilon)$ and $s = \Theta(1/\epsilon^2)$ so that the fraction of errors is at least $1 - R - \epsilon$ and the running time (and list size) is roughly $q^r = q^{1/\epsilon}$, which is polynomial in the block length for $q = \mathrm{poly}(n)$. Later, in Section 5.2 we shall show that the list size is in fact a constant independent of the block length (and even matches the generalized Singlton Bound).

Overview. The improved algorithm is based on the *method of multiplicities*, similarly to the way the algorithm presented in Section 3.3 for list decoding of Reed-Solomon Codes up to the Johnson Bound improves on the algorithm presented in Section 3.2.

In more detail, recall that in the previous Algorithm 4 for list-decoding of the multiplicity code, in the interpolation step, we searched for a non-zero low-degree $(s+1)$-variate polynomial $Q(X, Y_0, \ldots, Y_{s-1}) = A(X) + \sum_{\ell=0}^{s-1} B_\ell(X) Y_\ell$ so that $Q(a_i, w_i) = 0$ for any $i \in [n]$, and in the root finding step, we searched for all univariate polynomials $f(X) \in \mathbb{F}_q[X]$ of degree smaller than k satisfying the differential equation $Q(X, f(X), f^{(1)}(X), \cdots, f^{(s-1)}(X)) = 0$.

Furthermore, the main properties we needed out of Q were that the number of unknown coefficients in Q is greater than the number n of constraints of the form $Q(a_i, w_i) = 0$, and that the $(1, k, \ldots, k)$-weighted degree of Q is smaller than the agreement parameter t. The first property guarantees the existence of a non-zero Q, while the second property guarantees that for any univariate polynomial $f(X) \in \mathbb{F}_q[X]$ of degree smaller than k so that $f^{(<s)}(a_i) = w_i$ for at least t of the indices $i \in [n]$, it holds that

$Q(X, f(X), f^{(1)}(X), \cdots, f^{(s-1)}(X))$ is a univariate polynomial of degree smaller than t that has at least t roots, and consequently it is the zero polynomial.

In order to list-decode the multiplicity code up to capacity, we would like to set the agreement parameter t to be even smaller than in Algorithm 4. As was previously the case, this means that $Q(X, f(X), f^{(1)}(X), \cdots, f^{(s-1)}(X))$ has a smaller number of roots, which forces the degree of Q to be smaller in order to argue that it is the zero polynomial. Consequently, Q may potentially have less than n monomials, and so we may not be able to argue the existence of a non-zero polynomial Q.

To overcome this, we define Q to be an $(r+1)$-variate polynomial $Q(X, Y_0, \ldots, Y_{r-1})$ for some $r < s$, and we require that $Q(X, f(X), f^{(1)}(X), \ldots, f^{(r-1)}(X))$ vanishes on a_i with *multiplicity* $s - r + 1$ for any point $i \in [n]$ for which $f^{(<s)}(a_i) = w_i$. Thus $Q(X, f(X), f^{(1)}(X), \cdots, f^{(s-1)}(X))$ has more roots, counted with multiplicities, and consequently it can have higher degree and more coefficients, which may be helpful in arguing the existence of a non-zero polynomial Q. However, this also comes with a price, as we also increased the number of constraints on Q by a multiplicative factor of $s - r + 1$, and consequently we now require that Q has more coefficients than before. It turns out however that this trade-off is favorable, and allows to decrease the agreement parameter t to the minimum possible for an appropriate setting of r and s.

In what follows, we describe separately the changes made in the interpolation and root finding steps, and analyze their correctness and efficiency.

Step 1: Interpolation. In the interpolation step, we would now like to find a non-zero low degree $(r+1)$-variate polynomial $Q(X, Y_0, \ldots, Y_{r-1})$ which satisfies that $P_f(X) := Q(X, f(X), f^{(1)}(X), \ldots, f^{(r-1)}(X))$ vanishes on a_i with multiplicity $s - r + 1$ for any point $i \in [n]$ for which $f^{(<s)}(a_i) = w_i$. In the algorithm for list decoding of Reed-Solomon Codes up to the Johnson Bound (cf., Figure 3) this was guaranteed by requiring that Q vanishes with high *multivariate* multiplicity on any point (a_i, w_i) (cf., Claim 3.9). However, requiring that Q vanishes with high multivariate multiplicity on all these points turns out to be too costly in the current setting, and instead we directly enforce the property that $P_f(X)$ vanishes on each a_i for which $f^{(<s)}(a_i) = w_i$ with high *univariate* multiplicity.[3]

To understand what requirement this imposes on Q, recall that by (4), for any $j \in \mathbb{N}$, we have that

$$P_f^{(j)}(X) = A^{(j)}(X) + \sum_{\ell=0}^{r-1} \sum_{h=0}^{j} \binom{h+\ell}{\ell} B_\ell^{(j-h)}(X) \cdot f^{(h+\ell)}(X).$$

Further note that in the case that $f^{(<s)}(a_i) = w_i$, for any $j \in \{0, 1, \ldots, s-r\}$, we have all derivatives of the form $f^{(h+\ell)}(a_i)$ for $\ell \in \{0, 1, \ldots, r-1\}$ and $h \in \{0, 1, \ldots, j\}$ available for us.

Lemma 4.5. *Let* $t = \frac{(s-r+1)n + r(k-1) + 1}{(s-r+1)(r+1)}$. *Then there exists a non-zero* $(r+1)$*-variate polynomial* $Q(X, Y_0, \ldots, Y_{r-1})$ *over* $\mathbb{F}_q$ *of the form*

$$Q(X, Y_0, \ldots, Y_{r-1}) = A(X) + B_0(X) \cdot Y_0 + \cdots + B_{r-1}(X) \cdot Y_{r-1},$$

where $A(X)$ *has degree smaller than* $(s-r+1) \cdot t$ *and each* $B_\ell(X)$ *has degree smaller than* $(s-r+1) \cdot t - k + 1$, *and which satisfies that*

$$A^{(j)}(a_i) + \sum_{\ell=0}^{r-1} \sum_{h=0}^{j} \binom{h+\ell}{\ell} B_\ell^{(j-h)}(a_i) \cdot w_{i,h+\ell} = 0 \tag{5}$$

[3] Specifically, as we show below, enforcing that P_f vanishes with univariate multiplicity at least $s-r+1$ on a_i imposes $s-r+1$ linear constraints on the coefficients of Q, while enforcing that Q vanishes with mulivariate multiplicity at least $s-r+1$ on a_i imposes $\binom{(r+1)+(s-r+1)-1}{r+1} = \binom{s+1}{r+1} \approx s^r$ linear constraints on the coefficients of Q.

for any $i \in [n]$ and $j = 0, 1, \ldots, s-r$.

Furthermore, such a polynomial $Q(X, Y_0, \ldots, Y_{r-1})$ can be found by solving a system of linear equations which is obtained by viewing the coefficients of the monomials in $A(X), B_0(X), \ldots, B_{r-1}(X)$ as unknowns, and viewing each requirement of the form (5) as a homogeneous linear constraint on these unknowns.

Proof. If we view the coefficients of the monomials in $A(X), B_0(X), \ldots, B_{r-1}(X)$ as unknowns, then each requirement of the form (5) imposes a homogeneous linear constraint on these unknowns.

Further note that the number of unknowns is

$$(s-r+1)\cdot t + r\cdot((s-r+1)\cdot t - k + 1) = (r+1)\cdot(s-r+1)\cdot t - r\cdot(k-1) = (s-r+1)\cdot n + 1,$$

while the number of linear constraints is $(s-r+1)\cdot n$, and consequently this linear system has a non-zero solution. $\square$

Step 2: Root finding. In this step, we search for all univariate polynomials $f(X) \in \mathbb{F}_q[X]$ of degree smaller than k satisfying that $Q(X, f(X), f^{(1)}(X), \ldots, f^{(r-1)}(X)) = 0$, and we would like to show that any univariate polynomial $f(X) \in \mathbb{F}_q[X]$ of degree smaller than k so that $f^{(<s)}(a_i) = w_i$ for at least t indices $i \in [n]$ satisfies that $Q(X, f(X), f^{(1)}(X), \ldots, f^{(r-1)}(X)) = 0$. We have already shown how to perform the former in Lemma 4.3, the following lemma shows the latter.

Lemma 4.6. *Suppose that $Q(X, Y_0, \ldots, Y_{r-1})$ is a non-zero $(r+1)$-variate polynomial over $\mathbb{F}_q$ of the form*

$$Q(X, Y_0, \ldots, Y_{r-1}) = A(X) + B_0(X)\cdot Y_0 + \cdots + B_{r-1}(X)\cdot Y_{r-1},$$

where $A(X)$ has degree smaller than $(s-r+1)\cdot t$ and each $B_\ell(X)$ has degree smaller than $(s-r+1)\cdot t-k+1$, and which satisfies (5) for any $i \in [n]$ and $j = 0, 1, \ldots, s-r$. Let $f(X) \in \mathbb{F}_q[X]$ be a univariate polynomial of degree smaller than k so that $f^{(<s)}(a_i) = w_i$ for at least t indices $i \in [n]$. Then

$$Q(X, f(X), f^{(1)}(X), \ldots, f^{(r-1)}(X)) = 0.$$

Proof. The polynomial $P_f(X) := Q(X, f(X), f^{(1)}(X), \ldots, f^{(r-1)}(X))$ is a univariate polynomial of degree smaller than $(s-r+1)\cdot t$. Furthermore, by (4), for any $i \in [n]$ for which $f^{(<s)}(a_i) = w_i$, it holds that:

$$\begin{aligned}
P_f^{(j)}(a_i) &= A^{(j)}(a_i) + \sum_{\ell=0}^{r-1}\sum_{h=0}^{j}\binom{h+\ell}{\ell} B_\ell^{(j-h)}(a_i)\cdot f^{(h+\ell)}(a_i) \\
&= A^{(j)}(a_i) + \sum_{\ell=0}^{r-1}\sum_{h=0}^{j}\binom{h+\ell}{\ell} B_\ell^{(j-h)}(a_i)\cdot w_{i,h+\ell}
\end{aligned}$$

for any $j = 0, 1, \ldots, s-r$. So $P_f(X)$ vanishes on a_i with multiplicity $s-r+1$ for any $i \in [n]$ for which $f^{(<s)}(a_i) = w_i$. Thus $P_f(X)$ is a univariate polynomial of degree smaller than $(s-r+1)\cdot t$ with at least t roots of multiplicity $s-r+1$, and so by Fact 2.3 it must be the zero polynomial. $\square$

The full description of the algorithm appears in Figure 5 below, followed by correctness and efficiency analysis.

List decoding up to capacity of $\mathsf{MULT}_q^{(s)}(a_1, \ldots, a_n; k)$:

- **INPUT:** $w \in (\mathbb{F}_q^s)^n$, where $w_i = (w_{i,0}, \ldots, w_{i,s-1}) \in \mathbb{F}_q^s$ for each $i \in [n]$.
- **OUTPUT:** The list $\mathcal{L}$ of all polynomials $f(X) \in \mathbb{F}_q[X]$ of degree smaller than k so that $f^{(<s)}(a_i) = w_i$ for at least $t = \frac{(s-r+1)n+r(k-1)+1}{(s-r+1)(r+1)}$ indices $i \in [n]$.

1. Find a non-zero $(r+1)$-variate polynomial $Q(X, Y_0, \ldots, Y_{r-1})$ over $\mathbb{F}_q$ of the form

$$Q(X, Y_0, \ldots, Y_{r-1}) = A(X) + B_0(X) \cdot Y_0 + \cdots + B_{r-1}(X) \cdot Y_{r-1},$$

where $A(X)$ has degree smaller than $(s-r+1) \cdot t$ and each $B_\ell(X)$ has degree smaller than $(s-r+1) \cdot (t-k+1)$, and which satisfies that

$$A^{(j)}(a_i) + \sum_{\ell=0}^{r-1} \sum_{h=0}^{j} \binom{h+\ell}{\ell} B_\ell^{(j-h)}(a_i) \cdot w_{i,h+\ell} = 0$$

for any $i \in [n]$ and $j = 0, 1, \ldots, s-r$.

Such a polynomial $Q(X, Y_0, \ldots, Y_{r-1})$ can be found by solving a system of linear equations which is obtained by viewing the coefficients of the monomials in Q as unknowns, and viewing each requirement as above as a homogeneous linear constraint on these unknowns.

2. Find a basis for the list $\mathcal{L}'$ of all univariate polynomials $f(X)$ of degree smaller than k so that

$$Q(X, f(X), \ldots, f^{(r-1)}(X)) = 0.$$

This basis can be found by solving a system of linear equations which is obtained by viewing the k coefficients of $f(X)$ as unknowns, and viewing the requirement that each of the t coefficients of $Q(X, f(X), f^{(1)}(X), \ldots, f^{(r-1)}(X))$ is zero as t (non-homogeneous) linear constraints on these unknowns.

3. Output the list $\mathcal{L}$ of all codewords $(f(a_1), f(a_2), \ldots, f(a_n))$ corresponding to univariate polynomials $f(X) \in \mathcal{L}'$ of degree smaller than k which satisfy that $f^{(<s)}(a_i) = w_i$ for at least t indices $i \in [n]$.

Figure 5: List decoding of multiplicity codes up to capacity

Correctness. Suppose that $f(X) \in \mathbb{F}_q[X]$ is a univariate polynomial of degree smaller than k so that $f^{(<s)}(a_i) = w_i$ for at least t indices $i \in [n]$, we shall show that $f(X) \in \mathcal{L}$. By Lemma 4.5, there exists a non-zero $(r+1)$-variate polynomial $Q(X, Y_0, \ldots, Y_{r-1})$ satisfying the requirements on Step 1. By Lemma 4.6, we have that $Q(X, f(X), f^{(1)}(X), \ldots, f^{(r-1)}(X)) = 0$, and so $f(X)$ will be included in $\mathcal{L}'$ on Step 2. Clearly, $f(X)$ will also pass the checks on Step 3, and thus $f(X)$ will also be included in $\mathcal{L}$.

Efficiency. Step 1 can be performed in time $\mathrm{poly}(n, s, \log(q))$ by solving a system of $(s-r+1) \cdot n$ linear equations in $(s-r+1) \cdot n + 1$ variables using Gaussian elimination. Step 2 can be performed in time $\mathrm{poly}(n, s, \log(q))$ by solving a system of t linear equations in k variables. By Lemma 4.3, Step 3 can be performed in time $|\mathcal{L}| \cdot \mathrm{poly}(n, s, \log(q)) \le q^r \cdot \mathrm{poly}(n, s, \log(q))$, where the latter bound is polynomial in the block length for a constant r and $q = \mathrm{poly}(n)$.

4.3 List decoding Reed-Solomon Codes over subfield evaluation points up to capacity

We now turn back to the basic family of Reed-Solomon Codes, and consider a special subclass of these codes which are defined over an extension field, and *evaluated over a subfield*. We show that an algorithm similar to the one presented in Section 4.1 can be used to list decode these codes up to capacity, albeit with only a slightly non-trivial sub-exponential running time. This in particular implies a sub-exponential upper bound on the list size of these codes, and later on, in Section 5.3 we shall show that the list-size can be reduced to a constant independent of the block length (and the running time to polynomial in the block length) by bypassing to a (carefully chosen) subcode.

In more detail, in what follows fix a prime power q, distinct points $a_1, \ldots, a_n \in \mathbb{F}_q$, a positive integer k such that $k < n$, and a positive integer s, and consider the Reed-Solomon Code $\mathrm{RS}_{q^s}(a_1, \ldots, a_n; k)$ that is defined over the *extension field* $\mathbb{F}_{q^s}$. We shall show that $\mathrm{RS}_{q^s}(a_1, \ldots, a_n; k)$ can be list-decoded with similar guarantees to that of the list-decoding algorithm for multiplicity codes beyond the unique decoding radius (Algorithm 4). Suppose that $w \in (\mathbb{F}_{q^s})^n$ is a received word. Let $r \in \{1, \ldots, s\}$ be a parameter to be determined later on. We shall show how to compute the list of all polynomials $f(X) \in \mathbb{F}_{q^s}[X]$ of degree smaller than k so that $f(a_i) \neq w_i$ for at most $e = \frac{r(n-k+1)-1}{r+1}$ indices $i \in [n]$. That is, $f(a_i) = w_i$ for at least $t := n - e = \frac{n+r(k-1)+1}{r+1}$ indices $i \in [n]$.

The main advantage however of the Reed-Solomon Code $\mathrm{RS}_{q^s}(a_1, \ldots, a_n; k)$ over the corresponding multiplicity code $\mathsf{MULT}_q^{(s)}(a_1, \ldots, a_n; k)$ is that the rate of $\mathrm{RS}_{q^s}(a_1, \ldots, a_n; k)$ is higher by a multiplicative factor of s. More specifically, if we let $R := \frac{k}{n}$ denote the rate of the $\mathrm{RS}_{q^s}(a_1, \ldots, a_n; k)$ code, then this algorithm will be able to list decode from up to roughly a $\frac{r}{r+1}(1 - R)$-fraction of errors in time roughly q^{rk}. In particular, for any constant $\epsilon > 0$, we can choose $r = \Theta(1/\epsilon)$ and $s = \Theta(1/\epsilon^2)$ so that the fraction of errors is at least $1 - R - \epsilon$, and the running time (and list size) is roughly $q^{rk} = q^{\epsilon sk}$, which is sub-polynomial in the number q^{sk} of distinct codewords. Later, in Section 5.3, we shall show that the list size can be reduced to a constant independent of the block length (and the running time to polynomial in the block length) by passing to an appropriate subcode.

Overview. The list-decoding algorithm is very similar to the list-decoding algorithm for multiplicity codes beyond the unique decoding radius (Algorithm 4). Recall that in Algorithm 4, in the interpolation step, we searched for a non-zero low-degree $(s + 1)$-variate polynomial $Q(X, Y_0, \ldots, Y_{s-1})$ of the form

$$Q(X, Y_0, \ldots, Y_{s-1}) = A(X) + B_0(X) \cdot Y_0 + \cdots + B_{s-1}(X) \cdot Y_{s-1},$$

so that $Q(a_i, w_i) = 0$ for any $i \in [n]$, and in the root finding step, we searched for all univariate polynomials $f(X) \in \mathbb{F}_q[X]$ satisfying that

$$Q(X, f(X), f^{(1)}(X), \ldots, f^{(s-1)}(X)) = 0.$$

The main observation we shall use for list decoding of Reed-Solomon Codes over subfield evaluation points is that for any polynomial $f(X) = \sum_{i=0}^{k-1} f_i X^i$ over the extension field $\mathbb{F}_{q^s}$ and for any point a in the base field $\mathbb{F}_q$,

$$(f(a))^q = \left(\sum_{i=0}^{k-1} f_i a^i \right)^q = \sum_{i=0}^{k-1} (f_i)^q (a^q)^i = \sum_{i=0}^{k-1} (f_i)^q a^i,$$

where the second equality uses the fact that raising to the power of q is an $\mathbb{F}_q$-linear transformation, and the third equality uses the fact that $a^q = a$ for any $a \in \mathbb{F}_q$. Motivated by this, for a polynomial

$f(X) = \sum_{i=0}^{k-1} f_i X^i \in \mathbb{F}_{q^s}[X]$, we let $\sigma(f) = \sum_{i=0}^{k-1} (f_i)^q X^i \in \mathbb{F}_{q^s}[X]$. Under this notation, we have that $(f(a))^q = \sigma(f)(a)$ for any polynomial $f(X) \in \mathbb{F}_{q^s}[X]$ and point $a \in \mathbb{F}_q$.

Using the above, in the list-decoding algorithm for $\mathrm{RS}_{q^s}(a_1, \ldots, a_n; k)$, in the interpolation step, we once more search for a non-zero low degree $(r+1)$-variate polynomial of the form $Q(X) = A(X) + \sum_{\ell=0}^{r-1} B_\ell(X) \cdot Y^\ell$ (with coefficients in the extension field $\mathbb{F}_{q^s}$). But now in the interpolation step, we require that

$$Q\left(a_i, w_i, (w_i)^q, \ldots, (w_i)^{q^{r-1}}\right) = 0$$

for any $i \in [n]$. Then in the root finding step, we we search for all univariate polynomials $f(X) \in \mathbb{F}_{q^s}[X]$ satisfying that

$$Q\left(X, f(X), \sigma(f)(X), \ldots, \sigma^{r-1}(f)(X)\right) = 0.$$

In what follows, we describe separately the interpolation and root finding steps, and analyze their correctness and efficiency.

Step 1: Interpolation. In this step, we would like to find a non-zero low-degree $(r+1)$-variate polynomial $Q(X, Y_0, \ldots, Y_{r-1})$ satisfying that $Q\left(a_i, w_i, (w_i)^q, \ldots, (w_i)^{q^{r-1}}\right) = 0$. The proof of the following lemma is identical to the proof of Lemma 4.1.

Lemma 4.7. *Let $t = \frac{n+r(k-1)+1}{r+1}$. Then there exists a non-zero $(r+1)$-variate polynomial $Q(X, Y_0, \ldots, Y_{r-1})$ over $\mathbb{F}_{q^s}$ of the form*

$$Q(X, Y_0, \ldots, Y_{r-1}) = A(X) + B_0(X) \cdot Y_0 + \cdots + B_{r-1}(X) \cdot Y_{r-1},$$

where $A(X)$ has degree smaller than t and each $B_\ell(X)$ has degree smaller than $t - k + 1$, and which satisfies that

$$Q\left(a_i, w_i, (w_i)^q, \ldots, (w_i)^{q^{r-1}}\right) = 0$$

for any $i \in [n]$.

Furthermore, such a polynomial $Q(X, Y_0, \ldots, Y_{r-1})$ can be found by solving a system of linear equations which is obtained by viewing the coefficients of the monomials in $A(X), B_0(X), \ldots, B_{r-1}(X)$ as unknowns, and viewing each requirement of the form $Q\left(a_i, w_i, (w_i)^q, \ldots, (w_i)^{q^{r-1}}\right) = 0$ as a homogeneous linear constraint on these unknowns.

Step 2: Root finding. In this step, we search for all univariate polynomials $f(X) \in \mathbb{F}_{q^s}[X]$ satisfying $Q\left(X, f(X), \sigma(f)(X), \ldots, \sigma^{r-1}(f)(X)\right) = 0$, and we would like to show that any univariate polynomial $f(X) \in \mathbb{F}_{q^s}[X]$ of degree smaller than k so that $f(a_i) = w_i$ for at least t indices $i \in [n]$ satisfies that $Q\left(X, f(X), \sigma(f)(X), \ldots, \sigma^{r-1}(f)(X)\right) = 0$. We start by showing the latter.

Lemma 4.8. *Suppose that $Q(X, Y_0, \ldots, Y_{r-1})$ is a non-zero $(r+1)$-variate polynomial over $\mathbb{F}_{q^s}$ of the form*

$$Q(X, Y_0, \ldots, Y_{r-1}) = A(X) + B_0(X) \cdot Y_0 + \cdots + B_{r-1}(X) \cdot Y_{r-1},$$

where $A(X)$ has degree smaller than t and each $B_\ell(X)$ has degree smaller than $t - k + 1$, and which satisfies that

$$Q\left(X, f(X), \sigma(f)(X), \ldots, \sigma^{r-1}(f)(X)\right) = 0$$

for any $i \in [n]$. Let $f(X) \in \mathbb{F}_{q^s}[X]$ be a univariate polynomial of degree smaller than k so that $f(a_i) = w_i$ for at least t indices $i \in [n]$. Then

$$Q\left(X, f(X), \sigma(f)(X), \ldots, \sigma^{r-1}(f)(X)\right) = 0.$$

Proof. The polynomial $P_f(X) := Q\left(X, f(X), \sigma(f)(X), \ldots, \sigma^{r-1}(f)(X)\right)$ is a univariate polynomial over $\mathbb{F}_{q^s}$ of degree smaller than t, which satisfies that

$$\begin{aligned} P_f(a_i) &= Q\left(a_i, f(a_i), \sigma(f)(a_i), \ldots, \sigma^{r-1}(f)(a_i)\right) \\ &= Q\left(a_i, f(a_i), (f(a_i))^q, \ldots, (f(a_i))^{q^{r-1}}\right) \\ &= Q\left(a_i, w_i, (w_i)^q, \ldots, (w_i)^{q^{r-1}}\right) = 0 \end{aligned}$$

for any $i \in [n]$ for which $f(a_i) = w_i$. Thus $P_f(X)$ is a univariate polynomial of degree smaller than t with at least t roots, and so it must be the zero polynomial. □

The next lemma bounds the number of univariate polynomials $f(X) \in \mathbb{F}_{q^s}[X]$ satisfying the equation $Q\left(X, f(X), \sigma(f)(X), \ldots, \sigma^{r-1}(f)(X)\right) = 0$.

Lemma 4.9. *Suppose that $Q(X, Y_0, \ldots, Y_{r-1})$ is a non-zero $(r+1)$-variate polynomial over $\mathbb{F}_{q^s}$ of the form*

$$Q(X, Y_0, \ldots, Y_{r-1}) = A(X) + B_0(X) \cdot Y_0 + \cdots + B_{r-1}(X) \cdot Y_{r-1},$$

and let $\mathcal{L}$ be the list containing all polynomials $f(X) \in \mathbb{F}_{q^s}[X]$ of degree smaller than k so that

$$Q\left(X, f(X), \sigma(f)(X), \ldots, \sigma^{r-1}(f)(X)\right) = 0.$$

Then $|\mathcal{L}| \leq q^{(r-1)\cdot k}$.

Furthermore, $\mathcal{L}$ forms an affine subspace over $\mathbb{F}_q$ of dimension at most $(r-1)k$. A basis for this subspace can be found by solving a system of linear equations over $\mathbb{F}_q$ which is obtained by viewing the coefficients of $f(X)$ (represented as strings in $\mathbb{F}_q^s$) as unknowns, and viewing the requirement that each of the coefficients of $Q\left(X, f(X), \sigma(f)(X), \ldots, \sigma^{r-1}(f)(X)\right)$ is zero as a (non-homogeneous) $\mathbb{F}_q$-linear constraint on these unknowns.

Proof. As in the proof of Lemma 4.3, we may assume that there exists an element $a \in \mathbb{F}_q^s$ so that $B_{r-1}(a) \neq 0$. Without loss of generality, we may further assume that $a = 0$, since otherwise we can translate all polynomials $A(X), B_0(X), \ldots, B_{r-1}(X)$ by a, which corresponds to translating all polynomials in $\mathcal{L}$ by a, which does not change the size of $\mathcal{L}$.

Let $f(X) = \sum_{j=0}^{k-1} f_j X^j$ be a polynomial of degree smaller than k over $\mathbb{F}_{q^s}$ satisfying that

$$\begin{aligned} P_f(X) &:= Q\left(X, f(X), \sigma(f)(X), \ldots, \sigma^{r-1}(f)(X)\right) \\ &= A(X) + \sum_{\ell=0}^{r-1} B_\ell(X) \cdot \sigma^\ell(f)(X) \\ &= A(X) + \sum_{\ell=0}^{r-1} B_\ell(X) \sum_{j=0}^{k-1} (f_j)^{q^\ell} X^j \\ &= 0. \end{aligned}$$

We shall show that for any $j \in \{0, 1, \ldots, k-1\}$, given the first $j-1$ coefficients $f_0, f_1, \ldots, f_{j-1}$, there are at most q^{r-1} options for f_j, and consequently $|\mathcal{L}| \leq q^{(r-1)k}$.

To show the above, fix $j \in \{0, 1, \ldots, k-1\}$. Since $P_f(X)$ is the zero polynomial, the coefficient of X^j in $P_f(X)$ must be zero. On the other hand, inspecting the above expression for $P_f(X)$, we see that the coefficient of X^j in $P_f(X)$ can be written as $y_j + \sum_{\ell=0}^{r-1} B_\ell(0) \cdot (f_j)^{q^\ell}$, where $y_j \in \mathbb{F}_{q^s}$ is determined by $f_0, f_1, \ldots, f_{j-1}$.

Let

$$B(X) := B_0(0) \cdot X + B_1(0) \cdot X^q + \cdots + B_{r-1}(0) \cdot X^{q^{r-1}}.$$

Then by the above, $f_j \in \mathbb{F}_{q^s}$ must satisfy that $B(f_j) = -y_j$. On the other hand, by assumption that $B_{r-1}(0) \neq 0$, $B(X)$ is a non-zero polynomial of degree q^{r-1}, and so it attains any value for at most q^{r-1} assignments in $\mathbb{F}_{q^s}$. Consequently, for any $j \in \{0, 1, \ldots, k-1\}$, given prior coefficients $f_0, \ldots, f_{j-1}$, there are at most q^{r-1} options for the coefficient f_j, which implies in turn that $|\mathcal{L}| \leq q^{(r-1)k}$.

The moreover part follows by the linear structure of Q, and by the fact that raising to a power of q is an $\mathbb{F}_q$-linear transformation. □

The full description of the algorithm appears in Figure 6 below, followed by correctness and efficiency analysis.

List decoding of $\mathrm{RS}_{q^s}(a_1, \ldots, a_n; k)$ for $a_1, \ldots, a_n \in \mathbb{F}_q$:

- **INPUT:** $w \in (\mathbb{F}_{q^s})^n$.
- **OUTPUT:** The list $\mathcal{L}$ of all polynomials $f(X) \in \mathbb{F}_{q^s}[X]$ of degree smaller than k so that $f(a_i) = w_i$ for at least $t = \frac{n+r(k-1)+1}{r+1}$ indices $i \in [n]$.

1. Find a non-zero $(r+1)$-variate polynomial $Q(X, Y_0, \ldots, Y_{r-1})$ over $\mathbb{F}_q$ of the form

$$Q(X, Y_0, \ldots, Y_{r-1}) = A(X) + B_0(X) \cdot Y_0 + \cdots + B_{r-1}(X) \cdot Y_{r-1},$$

where $A(X)$ has degree smaller than t and each $B_\ell(X)$ has degree smaller than $t-k+1$, and which satisfies that $Q(a_i, (w_i)^q, \ldots, (w_i)^{q^{r-1}}) = 0$ for any $i \in [n]$.

Such a polynomial $Q(X, Y_0, \ldots, Y_{r-1})$ can be found by solving a system of linear equations which is obtained by viewing the coefficients of the monomials in Q as unknowns, and viewing each requirement of the form $Q(a_i, (w_i)^q, \ldots, (w_i)^{q^{s-1}}) = 0$ as a homogeneous linear constraint on these unknowns.

2. Find a basis for the list $\mathcal{L}'$ of all univariate polynomials $f(X)$ over $\mathbb{F}_{q^s}$ of degree smaller than k so that

$$Q\left(X, f(X), \sigma(f)(X), \ldots, \sigma^{r-1}(f)(X)\right) = 0.$$

This basis can be found by solving a system of linear equations over $\mathbb{F}_q$ which is obtained by viewing the k coefficients of $f(X)$ (represented as strings over $\mathbb{F}_q^s$) as unknowns, and viewing the requirement that each of the t coefficients of $Q\left(X, f(X), \sigma(f)(X), \ldots, \sigma^{r-1}(f)(X)\right)$ is zero as t (non-homogeneous) $\mathbb{F}_q$-linear constraints on these unknowns.

3. Output the list $\mathcal{L}$ of all codewords $(f(a_1), f(a_2), \ldots, f(a_n))$ corresponding to univariate polynomials $f(X) \in \mathcal{L}'$ of degree smaller than k which satisfy that $f(a_i) = w_i$ for at least t indices $i \in [n]$.

Figure 6: List decoding of Reed-Solomon Codes over subfield evaluation points

Correctness. Suppose that $f(X) \in \mathbb{F}_{q^s}[X]$ is a univariate polynomial of degree smaller than k so that $f(a_i) = w_i$ for at least t indices $i \in [n]$, we shall show that $f(X) \in \mathcal{L}$. By Lemma 4.7, there exists a non-zero $(r+1)$-variate polynomial $Q(X, Y_0, \ldots, Y_{r-1})$ satisfying the requirements on Step 1. By Lemma 4.8, we have that $Q\left(X, f(X), \sigma(f)(X), \ldots, \sigma^{r-1}(f)(X)\right) = 0$, and so $f(X)$ will be included in $\mathcal{L}'$ on Step 2. Clearly, $f(X)$ will also pass the checks on Step 3, and thus $f(X)$ will also be included in $\mathcal{L}$.

Efficiency. Step 1 can be performed in time $\mathrm{poly}(n, s, \log(q))$ by solving a system of n linear equations in $n+1$ variables using Gaussian elimination. Step 2 can be performed in time $\mathrm{poly}(n, s, \log(q))$ by solving a system of t linear equations in ks variables. By Lemma 4.9, Step 3 can be performed in time $|\mathcal{L}| \cdot \mathrm{poly}(n, s, \log(q)) \leq q^{(r-1)k} \cdot \mathrm{poly}(n, s, \log(q))$, where the latter bound is sub-polynomial in the number q^{sk} of distinct codewords for $r \leq \epsilon s$.

4.4 Bibliographic notes

In [GR08], Guruswami and Rudra introduced the family of Folded Reed-Solomon (FRS) Codes, and gave an efficient algorithm for list decoding these codes up to capacity, which gave the first explicit family of codes that can be efficiently list-decoded up to capacity. Folded Reed-Solomon Codes are defined similarly to Reed-Solomon Codes, except that each codeword entry $f(a_i)$ is replaced with a *tuple* of evaluations $(f(a_i), f(\gamma \cdot a_i), \ldots, f(\gamma^{s-1} \cdot a_i)) \in \mathbb{F}_q^s$, where γ is a generator of the multiplicative group $\mathbb{F}_q^*$, $s \geq 1$ is a *folding parameter*, and all sets $\{a_i, \gamma a_i, \ldots, \gamma^{s-1} a_i\}$ for $i \in [n]$ are pairwise disjoint. Guruswami and Rudra showed that for a sufficiently large s (depending on the gap to capacity ϵ), these codes are list-decodable up to capacity, with an algorithm similar to the one presented in Section 3.3 for list decoding of Reed-Solomon Codes up to the Johnson Bound. Later, Kopparty [Kop15] showed that a similar algorithm can also be used to list decode multiplicity codes up to capacity.

The algorithm for list decoding multiplicity (and FRS) codes up to capacity with an interpolating polynolmial Q with a *linear structure*, presented in Section 4.2, was discovered by Vadhan [Vad12, Section 5.2.4] and Guruswami and Wang [GW13]. The simpler algorithm for list decoding multiplicity codes beyond the unique decoding radius, presented in Section 4.1 as a warmup, follows the presentation of [GRS, Section 17.2] for list-decoding of Folded Reed-Solomon Codes.

It was shown by Bhandari, Harsha, Kumar, and Sudan [BHKS24b] that list-decoding algorithms for Folded Reed-Solomon Codes and multiplicity codes can be extended to the more general family of polynomial-ideal codes that includes both Folded Reed-Solomon Codes and multiplicity codes as a special case. The algorithm for list-decoding of Reed-Solomon Codes over subfield evaluation points up to capacity, presented in Section 4.3, was discovered by Guruswami and Xing [GX22].

All the aforementioned list-decoding algorithms can also be extended to the setting of list recovery, where one is given for each codeword entry a small list of ℓ possible values, and the goal is to return the list of all codewords that are consistent with most of these lists (list-decoding corresponds to the special case of $\ell = 1$). We refer the reader to the survey [RV25] for motivation and more information on the list recovery model and the extension of the above algorithms to the setting of list recovery.

5 Combinatorial upper bounds on list size

In the previous section, we presented efficient (polynomial-time) algorithms for list decoding multiplicity codes up to capacity, and we also showed that a similar algorithm can be used to list decode Reed-Solomon codes over subfield evaluation points in slightly non-trivial sub-exponential time. These algorithms in particular imply combinatorial upper bounds on the list sizes of these codes, however, the obtained bounds were quite large. Specifically, the list size was a large polynomial $n^{\Theta(1/\epsilon)}$ for multiplicity codes, and only slightly sub-exponential N^{ϵ} for Reed-Solomon codes over subfield evaluation points, where n, N denote the block length and code size, respectively, and ϵ denotes the gap to capacity.

In this section, we present tighter combinatorial upper bounds on the list size of these codes. Specifically, we first show in Section 5.1 below that Reed-Solomon Codes over *random* evaluation points are (combinatorially) list decodable *up to the generalized Singleton Bound*, and so are in particular list-decodable up to capacity with list-size $O(1/\epsilon)$. Then, in Section 5.2, we show that multiplicity codes attain the generalized Singleton Bound over *any* set of evaluation points. Finally, in Section 5.3, we show that a certain *subcode* (i.e., a code formed by only picking a subset of the codewords) of Reed-Solomon Codes over subfield evaluation points is (efficiently) list-decodable up to capacity with a *constant* list size (depending on ϵ).

5.1 Reed-Solomon Codes over random evaluation points

In this section, we show that Reed-Solomon Codes, evaluated over *random* points, chosen from a sufficiently large (exponential-size) field, are list-decodable all the way up to the *generalized Singleton Bound* (cf., Theorem 2.2), with high probability. The proof relies on two main ingredients: higher-order MDS codes and hypergraph connectivity. We first review each of these ingredients in Sections 5.1.1 and 5.1.2 below, and then use these to establish the result about random Reed-Solomon Codes in Section 5.1.3.

5.1.1 Higher-order MDS codes

The *Singleton Bound* states that for any code $C \subseteq \Sigma^n$ of distance Δ and size $|C| = \Sigma^k$ it must hold that $\Delta \leq n - k + 1$ (which in particular implies that $\delta \leq 1 - R$). The code C is said to be maximum distance separable (MDS) if it attains the Singleton Bound, that is, $\Delta = n - k + 1$. It is well-known that a *linear* code C is an MDS code if and only if its *dual code* $C^{\perp}$ is an MDS code. In Section 2.1, we presented the *generalized Singleton Bound* (cf., Theorem 2.2) which extends the notion of MDS codes to the setting of list decoding. In this section, we shall present another extension of the notion of *linear* MDS codes which we shall refer to as *higher-order MDS codes*. Later, in Section 5.1.3, we shall connect the two notions by showing that if C is a higher-order MDS code then its *dual code* $C^{\perp}$ attains the generalized Singleton Bound.

Let $C \subseteq \mathbb{F}^n$ be a linear code of dimension k, and let $G \in \mathbb{F}^{n \times k}$ be a generator matrix for C, where $G_1, \ldots, G_k \in \mathbb{F}^n$ denote the columns of G. For a string $w \in \mathbb{F}^n$, let $\mathcal{Z}(w) := \{i \in [n] \mid w_i = 0\}$ denote the set of zero entries of w. Note that any column G_j of G is a non-zero codeword of C, and so, if C is an MDS code, then G_j has weight at least $n - k + 1$, or equivalently, $|\mathcal{Z}(G_j)| \leq k - 1$. The following fact extends this observation by stating that for a linear MDS code, any t columns of G have at most $k - t$ common zeros.

Fact 5.1. *Suppose that $C \subseteq \mathbb{F}^n$ is a linear MDS code with generating matrix $G \in \mathbb{F}^{n \times k}$ with columns $G_1, \ldots, G_k$. Then $\left| \bigcap_{j \in J} \mathcal{Z}(G_j) \right| \leq k - |J|$ for any non-empty subset $J \subseteq [k]$.*

Proof. Assume towards a contradiction that there exists a non-empty subset $J \subseteq [k]$ of size t so that $\left| \bigcap_{j \in J} \mathcal{Z}(G_j) \right| \geq k - t + 1$. Without loss of generality we may assume that $J = \{1, 2, \ldots, t\}$ and $\bigcap_{j \in J} \mathcal{Z}(G_j) \supseteq \{1, 2, \ldots, k - t + 1\}$. Let A denote the top $k \times k$ minor of G. Then A has a $(k - t + 1) \times t$ block of zeros on its top left corner, and so its first t columns have rank at most $t - 1$, and so are linearly dependent. Consequently, A is not full rank, and so there exists $0 \neq m \in \mathbb{F}^k$ so that $A \cdot m = 0$. But this implies in turn that $G \cdot m$ is a non-zero codeword of C of weight at most $n - k$, which contradicts the assumption that C is an MDS code. $\square$

Higher-order MDS codes are codes which satisfy the converse of the above fact, i.e., for any zero patterns satisfying the intersection condition given by the above fact, there exists a generator matrix for the code with this zero pattern. More formally, we say that a series of (not necessarily distinct) subsets $Z_1, Z_2, \ldots, Z_k \subseteq [n]$ is a generic zero pattern (GZP) if $\left|\bigcap_{j\in J} Z_j\right| \leq k - |J|$ for any $\emptyset \neq J \subseteq [k]$. Further, we say that a matrix $G \in \mathbb{F}^{n\times k}$ with columns $G_1, \ldots, G_k \in \mathbb{F}^n$ attains $Z_1, \ldots, Z_k$ if $Z_j \subseteq \mathcal{Z}(G_j)$ for all $j \in [k]$.

Definition 5.2 (Higher-order MDS codes). *Let $C \subseteq \mathbb{F}^n$ be a linear* MDS *code of dimension k. We say that C is a higher-order MDS code if for any* GZP *$Z_1, \ldots, Z_k \subseteq [n]$, there exists a generator matrix G for C which attains $Z_1, \ldots, Z_k$.*

5.1.2 Hypergraph connectivity

To relate the notion of higher-order MDS codes to list decoding we shall use a certain notion of *hypergraph connectivity*. To describe this notion, we shall first need to introduce some notation and definitions. A hypergraph $H = (V, E)$ consists of a ground set of vertices $V = \{v_1, \ldots, v_n\}$ and a collection (possibly a multiset) of (hyper-)edges $E = \{e_1, \ldots, e_m\}$ which are (possibly empty) subsets of the vertices V. We define the weight $\mathsf{wt}(e)$ of an edge $e \in E$ as $\mathsf{wt}(e) := \max\{|e| - 1, 0\}$, and the weight $\mathsf{wt}(H)$ of the hypergraph H as the total edge weight $\mathsf{wt}(H) := \sum_{i=1}^m \mathsf{wt}(e_i)$. For $U \subseteq V$, let $H|_U$ denote the *sub-hypergrpah* $H|_U = (U, E|_U)$, where $E|_U := \{e_1 \cap U, \ldots, e_m \cap U\}$.

A hypergraph $H = (V, E)$ is said to be k-edge connected if for any non-empty proper subset $U \subsetneq V$, the number of *crossing edges* between U and $V \setminus U$ (i.e., the number of edges $e \in E$ with at least one vertex in U and at least one vertex in $V \setminus U$) is at least k. The $k = 1$ case corresponds to the standard notion of hypergraph connectivity which requires that there is a path between any pair of vertices in V.[4] By the *Hypergraph Menger's Theorem* [Kir03, Theorem 1.11], the $k \geq 2$ case is equivalent to the property that every pair of vertices in V have k edge-disjoint paths between them.

The notion of k-partition connectivity is a strengthening of the above notion of k-edge connectivity which requires that for any integer $t \geq 2$, and for any partition $\mathcal{P}$ of V into t parts, the number of crossing edges (i.e., the number of edges which are not contained in a single part of the partition) is at least $k(t-1)$. Note that k-edge connectivity corresponds to the special case of $t = 2$. We shall use the following weakening of partition connectivity.

Definition 5.3 (Weak partition connectivity). *A hypergraph $H = (V, E)$ is k-weakly partition connected if for every partition $\mathcal{P}$ of V into t parts, it holds that:*

$$\sum_{e\in E} \max\{|\mathcal{P}(e)| - 1, 0\} \geq k(t-1),$$

where $|\mathcal{P}(e)|$ denotes the number of parts in the partition intersecting e.

Note that for any partition $\mathcal{P}$, any crossing edge $e \in E$ contributes at least 1 to the left-hand side of the above inequality, and so k-partition connectivity implies k-weak partition connectivity. Moreover, for a partition $\mathcal{P}$ into two parts, the left-hand side of the above inequality equals the number of crossing edges, and so k-weak partition connectivity also implies k-edge connectivity.

Next we show that any hypergraph of large weight has a sub-hypergraph that is weakly partition connected.

Lemma 5.4. *Let $H = (V, E)$ be a hypergraph with at least two vertices and of weight at least $k(|V| - 1)$. Then there exists a subset $U \subseteq V$ of at least two vertices so that $H|_U$ is k-weakly partition connected.*

[4] A path in a hypergraph $H = (V, E)$ is a sequence $v_1, e_1', v_2, e_2', \ldots, v_{\ell-1}, e_{\ell-1}', v_\ell$, where $v_1, \ldots, v_\ell \in V$, $e_1', \ldots, e_{\ell-1}' \in E$, and $v_i, v_{i+1} \in e_i'$ for all $i = 1, \ldots, \ell - 1$.

Proof. Let $U \subseteq V$ be an inclusion-minimal subset with $|U| \geq 2$ so that

$$\mathsf{wt}(H|_U) \geq k(|U| - 1). \tag{6}$$

Note that for any subset $U' \subseteq V$ of size one, $\mathsf{wt}(H|_{U'})$ and $k(|U'| - 1)$ are both zero, and so by minimality of U, for any non-empty $U' \subsetneq U$, we have that

$$\mathsf{wt}(H|_{U'}) \leq k(|U'| - 1). \tag{7}$$

Let $\mathcal{P}$ be a partition of U into t parts $P_1, P_2, \ldots, P_t$, we shall show that

$$\sum_{e \in E|_U} \max\{|\mathcal{P}(e)| - 1, 0\} \geq k(t-1),$$

and so $H|_U$ is k-weakly partition connected. If $t = 1$, then we clearly have that both sides in the above inequality are zero, and so the inequality holds. Hence we may assume that $t \geq 2$.

Then we have that:

$$\begin{aligned}
\sum_{e \in E|_U} \max\{|\mathcal{P}(e)| - 1, 0\} &= \sum_{e \in E|_U} \Big(\mathsf{wt}(e) - \sum_{i=1}^{t} \mathsf{wt}(e \cap P_i)\Big) \\
&= \sum_{e \in E|_U} \mathsf{wt}(e) - \sum_{i=1}^{t} \sum_{e \in E|_U} \mathsf{wt}(e \cap P_i) \\
&= \mathsf{wt}(H|_U) - \sum_{i=1}^{t} \mathsf{wt}(H|_{P_i}) \\
&\geq k(|U| - 1) - \sum_{i=1}^{t} k(|P_i| - 1) \\
&= k\Big((|U| - 1) - \sum_{i=1}^{t} (|P_i| - 1)\Big) \\
&= k(t-1),
\end{aligned}$$

where the inequality follows by (6) and (7) (noting that P_i is a non-empty proper subset of U for any $i \in [t]$ by assumption that $t \geq 2$). $\square$

The main property of weakly-partition-connected hypergraphs we shall use is that they can be *oriented*. A **directed hypergraph** $H = (V, E)$ is a hypergraph where in each edge $e \in E$, one vertex is assigned as the **head**, and the rest of the vertices are assigned as **tails**. The **in-degree** $\deg_{\text{in}}(v)$ of a vertex $v \in V$ is the number of edges for which v is the head. A **path** in H is a sequence $v_1, e'_1, v_2, e'_2, \ldots, v_{\ell-1}, e'_{\ell-1}, v_\ell$, where $v_1, \ldots, v_\ell \in V$, $e'_1, \ldots, e'_{\ell-1} \in E$, and for all $i = 1, \ldots, \ell - 1$, vertex v_i is a tail of the edge e'_i, and vertex v_{i+1} is the head of e'_i. An **orientation** of an (undirected) hypergraph is obtained by assigning head to each hyperedge.

Theorem 5.5. *A hypergraph $H = (V, E)$ is k-weakly-partition-connected if and only if it has an orientation such that for some vertex v (the 'root'), every other vertex u has k edge-disjoint paths to v.*

The proof of the above theorem is beyond the scope of the current survey. The interested reader is referred to [Fra11, Theorem 9.4.13 and 15.4.4].

Finally, the following lemma relates the notion of weakly-partition-connected-hypergraphs to the notion of GZPs, discussed in the previous section.

Lemma 5.6. *Let $H = (V, E)$ be a k-weakly-partition-connected hypergraph with $V = \{v_1, \ldots, v_n\}$ and $E = \{e_1, \ldots, e_m\}$, where $n \geq 2$. For $i \in [n]$, let $Z_i = \{j \in [m] \mid v_i \notin e_j\} \subseteq [m]$. Then there exist non-negative integers $\delta_1, \ldots, \delta_n$ summing to $m - k$ so that taking δ_i copies of each Z_i gives a GZP.*

Proof. Consider the orientation of H given by Theorem 5.5. Without loss of generality, we may assume that v_1 is the root in this orientation. Let $\delta_1 := \deg_{\text{in}}(v_1) - k$ and $\delta_i := \deg_{\text{in}}(v_i)$ for any $i > 1$. Then we clearly have that $\delta_i \geq 0$ for $i > 1$, and we also have that $\delta_1 \geq 0$ since there are k edge-disjoint paths from v_2 to v_1, and so $\deg_{\text{in}}(v_1) \geq k$. Moreover, since there are m edges, the δ_i's must sum to $m - k$.

It remains to show the GZP property. Consider an arbitrary non-empty multiset $U \subseteq V$ such that each vertex v_i appears at most δ_i times in U. We shall show that $\left|\bigcap_{i:v_i \in U} Z_i\right| \leq \sum_{i:v_i \notin U} \delta_i$, and so

$$\left|\bigcap_{i:v_i \in U} Z_i\right| \leq \sum_{i:v_i \notin U} \delta_i = m - k - \sum_{i:v_i \in U} \delta_i \leq m - k - |U|,$$

which gives the GZP property.

To show that $\left|\bigcap_{i:v_i \in U} Z_i\right| \leq \sum_{i:v_i \notin U} \delta_i$, first observe that the left-hand side in this inequality equals the number of edges in E which do not contain any vertex from U. This number is clearly at most the sum of the indegrees in vertices not in U, which equals the right-hand side in the case that $v_1 \in U$. Next assume that $v_1 \notin U$, and fix an arbitrary vertex $v_i \in U$. Then there are k edge-disjoint paths from v_o to v_1. Each of these paths must contain an edge whose head is not in U, but contains some vertex from U. Thus, the left-hand side is at most $\left(\sum_{i:v_i \notin U} \deg_{\text{in}}(v_i)\right) - k$, which also equals the right-hand side in the case that $v_1 \notin U$. $\square$

5.1.3 Reed-Solomon Codes over random evaluation points are list-decodable up to the generalized Singleton Bound

Next we use the notions of higher-order MDS codes and hypergraph connectivity, introduced in the previous sections, to show that Reed-Solomon Codes over random evaluation points, chosen from an exponentially-large field, are list-decodable up to the generalized Singleton Bound. To do so, we shall use the following notion of agreement hypergraph.

Definition 5.7 (Agreement hypergraph)**.** *For strings $c_0, c_1, \ldots, c_\ell, w \in \Sigma^n$, we define the agreement hypergraph of $c_0, c_1, \ldots, c_\ell$ and w as the hypergraph on vertex set $\{0, 1, \ldots, \ell\}$ whose edges are $e_i := \{j \in \{0, 1, \ldots, \ell\} \mid (c_j)_i = w_i\} \subseteq \{0, 1, \ldots, \ell\}$ for $i \in [n]$ (i.e., each edge e_i corresponds to all strings c_j which agree with w on the i-th entry).*

Claim 5.8. *Suppose that $c_0, c_1, \ldots, c_\ell, w \in \Sigma^n$ are strings so that $\Delta(c_j, w) \leq \frac{\ell}{\ell+1} \cdot (n - k)$ for any $j \in \{0, 1, \ldots, \ell\}$. Then the agreement hypergraph H of $c_0, c_1, \ldots, c_\ell$ and w has weight at least $\ell \cdot k$.*

Proof. We have that:

$$\begin{aligned}
\mathsf{wt}(H) &= \sum_{i=1}^{n} \mathsf{wt}(e_i) \\
&\geq \sum_{i=1}^{n} (|e_i| - 1) \\
&= \sum_{i=1}^{n} \left(\left(\sum_{j=0}^{\ell} \mathbf{1}_{(c_j)_i = w_i} \right) - 1 \right) \\
&= -n + \sum_{j=0}^{\ell} (n - \Delta(w, c_j)) \\
&\geq -n + (\ell+1)n - \ell(n-k) = \ell \cdot k.
\end{aligned}$$

□

The following theorem says that the *dual code* of a higher-order MDS code attains the generalized Singleton Bound.

Theorem 5.9. *Let $C \subseteq \mathbb{F}^n$ be a linear* MDS *code of rate R, and suppose that $C^\perp$ is a higher-order MDS code. Then C is $(\frac{L}{L+1}(1-R), L)$-list decodable for any positive integer L.*

Proof. The theorem clearly holds for $L = 1$ by assumption that C is MDS. Next assume that C is not $(\frac{L}{L+1}(1-R), L)$-list decodable for some integer $L \geq 2$. Then there exist a string $w \in \mathbb{F}^n$ and distinct codewords $c_0, c_1, \ldots c_L \in C$ so that $\Delta(w, c_j) \leq \frac{L}{L+1}(1-R)n = \frac{L}{L+1}(n-k)$ for any $j \in \{0, 1, \ldots, L\}$, where $k = \dim(C) = Rn$. Let $H = (\{0, 1, \ldots, L\}, E)$ be the agreement hypergraph of $c_0, c_1, \ldots, c_L$ and w. Then by Claim 5.8 we have that $\mathsf{wt}(H) \geq L \cdot k$. By Lemma 5.4, there exists a subset $J \subseteq \{0, 1, \ldots, L\}$ of at least two vertices so that $H|_J$ is k-weakly partition connected. Without loss of generality, we may assume that $J = \{0, 1, \ldots, \ell\}$ for some $\ell \in [L]$. For $j \in \{0, 1, \ldots, \ell\}$, let $Z_j = \{i \in [n] \mid (c_j)_i \neq w_i\} \subseteq [n]$. By Lemma 5.6, there exist non-negative integers $\delta_0, \delta_1, \ldots, \delta_\ell$ summing to $n - k$ so that taking δ_j copies of each Z_j gives a GZP.

By assumption that $C^\perp$ is a higher-order MDS code, there exists a generating matrix $G \in \mathbb{F}^{n \times (n-k)}$ for $C^\perp$ so that G attains the above GZP. Then $H := G^T \in \mathbb{F}^{(n-k) \times n}$ is a parity-check matrix for C. Since $Hc = 0$ for any $c \in C$, we have that $Hw = H(w - c_j)$ for any $j \in \{0, 1, \ldots, \ell\}$. Furthermore, by the definition of the Z_j's, for any $j \in \{0, 1, \ldots, \ell\}$, the entries in $Hw = H(w - c_j)$ which correspond to Z_j are zero. We conclude that $Hw = 0$, and so $w \in C$. Finally, since $\ell \geq 1$, there must exist some $j \in \{0, 1, \ldots, \ell\}$ so that $w \neq c_j$. So w and c_j are two distinct codewords so that $\Delta(w, c_j) \leq \frac{L}{L+1}(n-k) < n - k + 1$, which contradicts the assumption that C is MDS. □

Next we would like to apply the above theorem to Reed-Solomon Codes. To do so, we need to show that the *dual* of a Reed-Solomon Code is higher-order MDS. It is well-known that the dual of a Reed-Solomon Code $\mathrm{RS}_q(k)$, evaluated over the whole field, is also a Reed-Solomon Code $\mathrm{RS}_q(n-k)$. For a Reed-Solomon Code $\mathrm{RS}_q(a_1, \ldots, a_n; k)$, evaluated over a subset of field elements, it is known that there exist non-zero scalers $b_1, \ldots, b_n \in \mathbb{F}_q$ so that the dual code contains all codewords of the form $(b_1 \cdot c_1, \ldots, b_n \cdot c_n)$ for $(c_1, \ldots, c_n) \in \mathrm{RS}_q(a_1, \ldots, a_n; n-k)$. Note that if G is a generator matrix for $\mathrm{RS}_q(a_1, \ldots, a_n; n-k)$, then $B \cdot G$ is a generator matrix for $\mathrm{RS}_q(a_1, \ldots, a_n; k)^\perp$, where B is a diagonal matrix with diagonal entries

$b_1, \ldots, b_n$. Since G and $B \cdot G$ have the same zero entries, to show that $\mathrm{RS}_q(a_1, \ldots, a_n; k)^\perp$ is higher-order MDS, it suffices to show that $\mathrm{RS}_q(a_1, \ldots, a_n; n-k)$ is higher-order MDS.

We shall show that with high probability, a Reed-Solomon code with random evaluation points is higher-order MDS. The proof relies on the following theorem, known as the 'Generator-Matrix-MDS (GM-MDS)' Theorem (the proof of this theorem is beyond the scope of the survey, and we refer the reader to [Lov21, YH19] for more details). In what follows, for $Z \subseteq [n]$, let f_Z denote the $(n+1)$-variate polynomial given by $f_Z(X, Y_1, \ldots, Y_n) = \prod_{i \in Z}(X - Y_i)$. Also, for a field $\mathbb{F}$, let $\mathbb{F}(Y_1, \ldots, Y_n)$ denote the *field of rational functions* over $\mathbb{F}$ in the indeterminates $Y_1, \ldots, Y_n$.

Theorem 5.10 (GM-MDS Theorem, [Lov21, YH19])**.** *Let $\mathbb{F}$ be a field, and let $Z_1, \ldots, Z_k \subseteq [n]$ be a GZP. Then the polynomials $f_{Z_1}, f_{Z_2}, \ldots, f_{Z_k}$, when viewed as univariate polynomials in X with coefficients in $\mathbb{F}(Y_1, \ldots, Y_n)$, are linearly independent over $\mathbb{F}(Y_1, \ldots, Y_n)$.*

Corollary 5.11. *Let q be a prime power, and let k, n be positive integers so that $k < n$ and $2^{kn} \cdot (kn) < q$. Then with probability at least $1 - \frac{2^{kn} \cdot (kn)}{q}$ over the choice of independent and uniform $a_1, \ldots, a_n \in \mathbb{F}_q$, it holds that $\mathrm{RS}_q(a_1, \ldots, a_n; k)$ is a higher-order MDS code.*

Proof. Fix a GZP $Z_1, \ldots, Z_k \subseteq [n]$, and note that by the GZP property, $|Z_j| \leq k-1$ for any $j \in [k]$. For $j \in [k]$, let $f'_{Z_j}(X) = f_{Z_j}(X, a_1, \ldots, a_n) = \prod_{i \in Z_j}(X - a_i)$, and note that f'_{Z_j} is a univariate polynomial over $\mathbb{F}_q$ of degree smaller than k. We shall show that with probability at least $1 - \frac{k \cdot n}{q}$ over the choice of $a_1, \ldots, a_n \in \mathbb{F}_q$, $f'_{Z_1}(X), \ldots, f'_{Z_k}(X)$ are linearly independent over $\mathbb{F}_q$.

To see the above, consider the $k \times k$ matrix M whose j-th column is the coefficient vector of the polynomial f_{Z_j}, when viewed as a univariate polynomial in X with coefficients in $\mathbb{F}_q(Y_1, \ldots, Y_n)$. By Theorem 5.10, $f_{Z_1}, \ldots, f_{Z_k}$, when viewed as univariate polynomials in X with coefficients in $\mathbb{F}_q(Y_1, \ldots, Y_n)$, are linearly independent over $\mathbb{F}_q(Y_1, \ldots, Y_n)$, and so M is a full-rank matrix over $\mathbb{F}_q(Y_1, \ldots, Y_n)$. Further note that each entry in M is a multilinear polynomial in $\mathbb{F}_q[Y_1, \ldots, Y_n]$, and consequently, $\det(M)$ over $\mathbb{F}_q(Y_1, \ldots, Y_n)$ is a non-zero polynomial in $\mathbb{F}_q[Y_1, \ldots, Y_n]$ of total degree at most $k \cdot n$. By the Schwartz-Zippel Lemma (Lemma 2.5), with probability at least $1 - \frac{k \cdot n}{q}$ over the choice of $a_1, \ldots, a_n \in \mathbb{F}_q$, the matrix $M' \in \mathbb{F}_q^{k \times k}$, obtained from M by assigning $a_1, \ldots, a_n$ to the indeterminates $Y_1, \ldots, Y_n$, has a non-zero determinant over $\mathbb{F}_q$, and so is full rank. But note that the columns of M' are precisely the coefficient vectors of $f'_{Z_1}(X), \ldots, f'_{Z_k}(X)$, and so if M' is full rank, then these polynomials are linearly independent over $\mathbb{F}_q$.

Next assume that $f'_{Z_1}(X), \ldots, f'_{Z_k}(X)$ are linearly independent, and let $G \in \mathbb{F}_q^{n \times k}$ be the matrix whose columns are the codewords of $\mathrm{RS}_q(a_1, \ldots, a_n; k)$ corresponding to these polynomials. Then the columns of G are k linearly independent codewords in $\mathrm{RS}_q(a_1, \ldots, a_n; k)$, and consequently we clearly have that $\mathrm{Image}(G) = \mathrm{RS}_q(a_1, \ldots, a_n; k)$, and so G is a generating matrix for this code. Furthermore, by the definition of $f'_{Z_1}(X), \ldots, f'_{Z_k}(X)$, we also clearly have that G attains $Z_1, \ldots, Z_k$.

So we conclude that for any GZP $Z_1, \ldots, Z_k \subseteq [n]$, with probability at least $1 - \frac{k \cdot n}{q}$ over the choice of $a_1, \ldots, a_n \in \mathbb{F}_q$, there exists a generator matrix for $\mathrm{RS}_q(a_1, \ldots, a_n; k)$ which attains $Z_1, \ldots, Z_k$. Since the number of GZPs $Z_1, \ldots, Z_k \subseteq [n]$ is at most 2^{kn}, by a union bound, we conclude that with probability at least $1 - 2^{kn} \cdot \frac{k \cdot n}{q}$ over the choice of $a_1, \ldots, a_n \in \mathbb{F}_q$, for any GZP $Z_1, \ldots, Z_k \subseteq [n]$, there exists a generator matrix for $\mathrm{RS}_q(a_1, \ldots, a_n; k)$ which attains $Z_1, \ldots, Z_k$. □

Combining the above corollary with Theorem 5.9, we conclude that Reed-Solomon Codes over random evaluation points, chosen from an exponentially large field, attain the generalized Singleton Bound with high probability (and in particular, are list-decodable up to capacity with high probability). With some more effort, it can be shown that the field size can be reduced to polynomial, and even linear, in n, but at the cost of only *approximately* attaining the generalized Singleton Bound, that is, achieving a list-decoding radius of $\frac{L}{L+1}(1-\frac{k}{n}-\epsilon)$ for any positive itneger L and constant $\epsilon > 0$ (which can be shown to be necessary), we refer the reader to [AGG$^+$25] for more details. Interesting open problems are to find *explicit* evaluation points so that Reed-Solomon Codes evaluated on these points achieve list-decoding capacity, as well as *efficient* list-decoding algorithms for Reed-Solomon Codes up to capacity.

5.2 Multiplicity codes

In this section, we show that multiplicity codes attain the *generalized Singleton bound* over *any* set of evaluation points. More specifically, we first present in Section 5.2.1 below a simple probabilistic argument which shows that multiplicity codes are list-decodable up to capacity with a constant list size that depends *exponentially* on $1/\epsilon$. This argument is quite general and applies to *any* linear code that is list-decodable up to capacity with a list that is contained in a low-dimensional subspace. Later, in Section 5.2.2, we shall show that multiplicity codes even attain the *generalized Singleton Bound* (and in particular the list size only depends *linearly* on $1/\epsilon$), using more specific properties of multiplicity codes.

5.2.1 Constant list size

In this section, we show that multiplicity codes are list decodable up to capacity with a *constant* list size (depending on the gap to capacity ϵ), based on a simple probabilistic argument which shows that for a linear code of large distance, the list cannot contain many codewords coming from a low dimensional subspace.

More specifically, we shall prove the following fairly general lemma which applies to *any* linear code, and which says that if C is a linear code of relative distance δ that is list-decodable from a $(\delta-\epsilon)$-fraction of errors with a list that is contained in a linear subspace of dimension at most r, then the list size is in fact upper bounded by a quantity that only depends on δ, ϵ, and r. In particular, if δ, ϵ, r are all constants, independent of the block length, than so is the list size.

Lemma 5.12. *Let $\mathbb{F}$ be a finite field, and let $C \subseteq \Sigma^n$ be an $\mathbb{F}$-linear code of relative distance δ. Suppose furthermore that C is list decodable from a $(\delta-\epsilon)$-fraction of errors with a list $\mathcal{L}$ that is contained in an $\mathbb{F}$-linear subspace $V \subseteq C$ of dimension r. Then*

$$|\mathcal{L}| \leq \left(\frac{r}{\epsilon(1-\delta)}\right)^{O\left(\frac{r}{\epsilon(1-\delta)}\cdot\log\left(\frac{1}{1-\delta}\right)\right)}.$$

Recall that multiplicity codes of rate R have relative distance at least $\delta = 1-R$, and that by Lemma 4.3, these codes are list-decodable from a $(\delta-\epsilon)$-fraction of errors with a list that is contained in an $\mathbb{F}$-linear subspace of dimension $r = O(1/\epsilon)$. The above lemma then implies that these codes are list-decodable from an $(1-R-\epsilon)$-fraction of errors with a constant list size on the order of $(1/\epsilon)^{O(1/\epsilon^2)}$. Later, in Section 5.2.2, we shall show that the list size can be further reduced to $O(1/\epsilon)$ (and even all the way up to the generalized Singleton Bound!), using some additional properties that are more specific to multiplicity codes. We now turn to the proof of the above Lemma 5.12.

Proof of Lemma 5.12. The proof is algorithmic: we will give a simple randomized algorithm Prune, which when given the received word $w \in \Sigma^n$, either outputs a vector $v \in V$ or outputs $\perp$. The guarantee is that for any $v \in \mathcal{L}$, v is output by the algorithm Prune with probability at least $p_0 = p_0(\delta, \epsilon, r)$, which implies in turn that $|\mathcal{L}| \leq \frac{1}{p_0}$.

The algorithm Prune works as follows. For some parameter t, to be determined later on, it picks entries $i_1, i_2, \ldots, i_t \in [n]$ uniformly and independently at random, and lets $I := \{i_1, \ldots, i_t\}$. Then the algorithm checks if there is a unique $v \in V$ that agrees with w on I (that is, $v_i = w_i$ for all $i \in I$). If so, it outputs that unique element v; otherwise (i.e., either there are zero or greater than one such v's) it outputs $\perp$.

We would like to show that for any $v \in \mathcal{L}$, the algorithm outputs v with constant probability p_0 (depending only on δ, ϵ, and r). Fix such a $v \in \mathcal{L}$. Let E_1 be the event that v agrees with w on I, and let E_2 be the event that two different codewords in V agree on I. Note that the algorithm will output v if and only if the event E_1 holds and the event E_2 does not hold, so the probability that v is output is at least $\Pr[E_1] - \Pr[E_2]$. The following two claims give lower and upper bounds on the probabilities of the events E_1 and E_2, respectively.

Claim 5.13. $\Pr[E_1] \geq (1 - \delta + \epsilon)^t$.

Proof. Follows since $v \in \mathcal{L}$, and so $v_i = w_i$ for at least a $(1 - \delta + \epsilon)$-fraction of the entries $i \in [n]$, and since $i_1, \ldots, i_t \in [n]$ are chosen uniformly and independently at random. □

Claim 5.14. $\Pr[E_2] \leq (1 - \delta)^t \cdot \left(\frac{t}{1-\delta}\right)^r$.

Proof. For $i \in [n]$, let $A_i := \{v \in C \mid v_i = 0\}$, let $V_0 := V$, and for $j = 1, \ldots, t$, let

$$V_j := V \cap A_{i_1} \cap A_{i_2} \cap \cdots \cap A_{i_j},$$

and $r_j := \dim_{\mathbb{F}}(V_j)$. Observe that $r = r_0 \geq r_1 \geq \ldots \geq r_t$, and that the event E_2 holds if and only if $r_t > 0$. Note also that $|v| \geq \delta n$ for any non-zero $v \in V$, since $V \subseteq C$ and C has relative distance δ.

Next we claim that for any $j = 0, 1, \ldots, t-1$, it holds that $r_{j+1} \leq \max\{0, r_j - 1\}$ with probability at least δ over the choice of i_{j+1}. To see this, note first that if $r_j = 0$, then $r_{j+1} \leq r_j \leq 0$ and so we are done. Otherwise, if $r_j \neq 0$ then there exists a non-zero vector $v \in V_j$. Recalling that $V_j \subseteq V$, we have that $|v| \geq \delta n$, and so $v_{i_{j+1}} \neq 0$ with probability at least δ over the choice of i_{j+1}. But recalling that $V_{j+1} \subseteq A_{i_{j+1}}$, this implies in turn that $v \notin V_{j+1}$. So in this case there exists a non-zero $v \in V_j$ so that $v \notin V_{j+1}$, and so the dimension of V_{j+1} is strictly smaller than that of V_j.

Finally, note that if $r_t > 0$, then it must hold that $r_{j+1} > \max\{0, r_j - 1\}$ for at least $t - r + 1$ of the j's in $0, 1, \ldots, t-1$. By the above and the union bound, the probability of this event is at most

$$\binom{t}{t-r+1} \cdot (1-\delta)^{t-r+1} \leq (1-\delta)^t \cdot \left(\frac{t}{1-\delta}\right)^r.$$

□

By the above Claims 5.13 and 5.14, we conclude that any $v \in \mathcal{L}$ is output by the algorithm Prune with probability at least

$$p_0 \geq \Pr[E_1] - \Pr[E_2] \geq (1-\delta+\epsilon)^t - (1-\delta)^t \cdot \left(\frac{t}{1-\delta}\right)^r.$$

Finally, by setting $t := 3 \cdot \frac{r}{\epsilon(1-\delta)} \cdot \log\left(\frac{r}{\epsilon(1-\delta)}\right)$ in the above expression we get that

$$\begin{aligned} p_0 : &= (1-\delta+\epsilon)^t - (1-\delta)^t \cdot \left(\frac{t}{1-\delta}\right)^r \\ &\geq (1-\delta)^t \cdot \left[(1+\epsilon)^t - \left(\frac{t}{1-\delta}\right)^r\right] \\ &\geq (1-\delta)^t \cdot \left[\left(\frac{r}{\epsilon(1-\delta)}\right)^{3\cdot r} - \left(3 \cdot \frac{r}{\epsilon(1-\delta)^2} \cdot \log\left(\frac{r}{\epsilon(1-\delta)}\right)\right)^r\right] \\ &\geq (1-\delta)^t \\ &\geq (1-\delta)^{O\left(\frac{r}{\epsilon(1-\delta)} \cdot \log\left(\frac{r}{\epsilon(1-\delta)}\right)\right)}, \end{aligned}$$

where the penultimate inequality holds when the ratio $\frac{r}{\epsilon(1-\delta)}$ is sufficiently large. This implies in turn that

$$|\mathcal{L}| \leq \frac{1}{p_0} \leq \left(\frac{1}{1-\delta}\right)^{O\left(\frac{r}{\epsilon(1-\delta)} \cdot \log\left(\frac{r}{\epsilon(1-\delta)}\right)\right)} = \left(\frac{r}{\epsilon(1-\delta)}\right)^{O\left(\frac{r}{\epsilon(1-\delta)} \cdot \log\left(\frac{1}{1-\delta}\right)\right)},$$

which concludes the proof of the lemma. □

5.2.2 Generalized Singleton Bound

In this section, we show that multiplicity codes (approximately) attain the *generalized Singleton Bound* of Theorem 2.2, which in particular implies that the list size of multiplicity codes only depends *linearly* on $1/\epsilon$.

More specifically, in what follows, fix a prime power q, distinct evaluation points $a_1, \ldots, a_n \in \mathbb{F}_q$, and positive integers s, k so that $k < sn$ and $\max\{k, s\} \leq \mathsf{char}(\mathbb{F}_q)$, and let $\mathsf{mult}_q^{(s)}(a_1, \ldots, a_n; k)$ be the corresponding multiplicity code. We shall show that for any positive integer $L < s$, $\mathsf{mult}_q^{(s)}(a_1, \ldots, a_n; k)$ is (α, L)-list decodable for $\alpha < \frac{L}{L+1}(1 - \frac{k}{n(s-L+1)})$. In particular, if we let $R := \frac{k}{s \cdot n}$ denote the rate of this code, then it is list decodable from a $\frac{L}{L+1}(1 - \frac{s}{s-L+1} \cdot R)$-fraction of errors with list size L. So for any $\epsilon > 0$, we can choose $s = \Theta(L/\epsilon)$, so that the fraction of errors is at least $\frac{L}{L+1}(1 - R - \epsilon)$. In particular, we can choose $s = \Theta(1/\epsilon^2)$ so that the multiplicity code $\mathsf{mult}_q^{(s)}(a_1, \ldots, a_n; k)$ is list decodable from an $(1 - R - \epsilon)$-fraction of errors with list size $O(1/\epsilon)$.

By Claim 5.8, this would be a consequence of the following lemma.

Lemma 5.15. *Let q be a prime power, let $a_1, \ldots, a_n$ be distinct points in $\mathbb{F}_q$, and let s, k, ℓ be positive integers so that $\ell < s$, $k < sn$, and $\max\{k, s\} \leq \mathsf{char}(\mathbb{F}_q)$. Let $w \in (\mathbb{F}_q^s)^n$ be a string, let $c_0, c_1, \ldots, c_\ell$ be distinct codewords in $\mathsf{mult}_q^{(s)}(a_1, \ldots, a_n; k)$, and let H be the agreement hypergraph of $c_0, c_1, \ldots, c_\ell$ and w (cf., Definition 5.7). Then $\mathsf{wt}(H) < \frac{\ell}{s-\ell+1} \cdot k$.*

Before we prove the above lemma, we show that this lemma implies that $\mathsf{mult}_q^{(s)}(a_1, \ldots, a_n; k)$ is (α, L)-list decodable for $\alpha < \frac{L}{L+1}(1 - \frac{k}{n(s-L+1)})$. To see this, suppose on the contrary that $\mathsf{mult}_q^{(s)}(a_1, \ldots, a_n; k)$ is not (α, L)-list decodable. Then there exist a string $w \in (\mathbb{F}_q^s)^n$ and codewords $c_0, c_1 \ldots, c_L \in \mathsf{mult}_q^{(s)}(a_1, \ldots, a_n; k)$ so that $\Delta(w, c_j) \leq \frac{L}{L+1}(n - \frac{k}{s-L+1})$ for any $j \in \{0, 1, \ldots, L\}$. Let H be the agreement hypergraph of $c_0, c_1, \ldots, c_L$ and w. Then by Claim 5.8, H has weight at least $L \cdot \frac{k}{s-L+1}$ which contradicts the above Lemma 5.15.

For the proof of Lemma 5.15, we first recall that in Section 5.2.1 we showed that for *any* $\mathbb{F}$-linear code $C \subseteq \Sigma^n$, and for any $\mathbb{F}$-linear subspace $V \subseteq C$, the dimension of the intersection $V \cap A_i$ for a random $i \in [n]$ is typically small, where $A_i := \{v \in C \mid v_i = 0\}$. For the proof of Lemma 5.15, we first show a tighter bound on the expected dimension of $V \cap A_i$ for a random $i \in [n]$ for the special case of multiplicity codes.

Lemma 5.16. *Let q be a prime power, let $a_1, \ldots, a_n$ be distinct points in $\mathbb{F}_q$, and let s, k be positive integers so that $k < sn$ and $\max\{k, s\} \leq \textsf{char}(\mathbb{F}_q)$. Let $V \subseteq \mathsf{mult}_q^{(s)}(a_1, \ldots, a_n; k)$ be an $\mathbb{F}_q$-linear subspace of dimension $r \leq s$, and for $i \in [n]$, let $A_i := \{v \in \mathsf{mult}_q^{(s)}(a_1, \ldots, a_n; k) \mid v_i = 0\}$. Then we have that:*

$$\sum_{i=1}^{n} \dim(A_i \cap V) \leq \frac{r}{s - r + 1} \cdot k.$$

Proof. Let $c_1, \ldots, c_r$ be a basis for V, and let $f_1(X), \ldots, f_r(X) \in (\mathbb{F}_q)_{<k}[X]$ be the polynomials corresponding to these codewords. Let $W(X)$ be the $r \times r$ **Wronskian matrix** over the polynomial ring $\mathbb{F}_q[X]$ given by

$$W(X) = \begin{pmatrix} f_1(X) & f_2(X) & \cdots & f_r(X) \\ f_1^{(1)}(X) & f_2^{(1)}(X) & \cdots & f_r^{(1)}(X) \\ \vdots & \vdots & \vdots & \vdots \\ f_1^{(r-1)}(X) & f_2^{(r-1)}(X) & \cdots & f_r^{(r-1)}(X) \end{pmatrix},$$

and let $D(X) := \det(W(X)) \in \mathbb{F}_q[X]$.

Then $D(X)$ is a polynomial of degree at most $(k-1) \cdot r$. Next we claim that $D(X) \neq 0$. Interestingly, one way to see this is using Lemma 4.3 which shows that the list of multiplicity codes is contained in a low-dimensional subspace. In more detail, suppose on the contrary that $D(X) \neq 0$. Then $W(X)$ is singular over the field of rational functions $\mathbb{F}_q(X)$, and so there exists a non-trivial linear combination of rows that sums to zero. That is, there exist $B_0(X), B_1(X), \ldots, B_{r-1}(X) \in \mathbb{F}_q[X]$, not all zeros, so that for any $t = 1, \ldots, r$,

$$B_0(X) \cdot f_t(X) + B_1(X) \cdot f_t^{(1)}(X) + \cdots + B_{r-1}(X) \cdot f_t^{(r-1)}(X) = 0.$$

Lemma 4.3 then implies that $f_1(X), \ldots, f_r(X)$ lie in an $(r-1)$-dimensional subspace, which contradicts our assumption that they span an r-dimensional subspace.

Next we show that for any $i \in [n]$, $D(X)$ vanishes on each element a_i with multiplicity at least $(s - r + 1) \cdot \dim(A_i \cap V)$. To this end, fix $i \in [n]$, and let $r_i := \dim(A_i \cap V)$. We would like to show that $D(X)$ vanishes on a_i with multiplicity at least $(s - r + 1) \cdot r_i$, which by the definition of the Hasse Derivative, is equivalent to the property that $(X - a_i)^{(s-r+1) \cdot r_i}$ divides $D(X)$.

To show the above, first note that performing elementary operations on the columns of $W(X)$ only changes $D(X)$ by a constant multiplicative factor, and hence we may assume without loss of generality that $f_1(X), \ldots, f_{r_i}(X)$ are a basis for $A_i \cap V$, and so $f_t^{(j)}(a_i) = 0$ for any $t = 1, 2, \ldots, r_i$ and $j = 0, 1, \ldots, s-1$. So $f_1(X), \ldots, f_{r_i}(X)$ all vanish on a_i with multiplicity at least s, and so $(X - a_i)^s$ divides all these polynomials. Furthermore, $(X - a_i)^{s-r+1}$ divides $f_t^{(j)}(X)$ for any $t = 1, 2, \ldots, r_i$ and $j = 0, 1, \ldots, r-1$ since taking derivative reduces the degree of each monomial by at most 1. So we conclude that $(X - a_i)^{s-r+1}$ divides each entry in the first r_i columns, which implies in turn that $(X - a_i)^{(s-r+1) \cdot r_i}$ divides the determinant $D(X)$ of $W(X)$.

So we conclude that $D(X)$ is a non-zero polynomial of degree at most $(k-1)\cdot r$, which vanishes on each a_i with multiplicity at least $(s-r+1)\cdot \dim(A_i \cap V)$. Consequently, we have that

$$\sum_{i=1}^{n} (s-r+1)\cdot \dim(A_i \cap V) \le (k-1)\cdot r.$$

Dividing both sides in the above inequality by $s-r+1$ gives the desired conclusion. □

We now turn to the proof of Lemma 5.15, based on the above Lemma 5.16.

Proof of Lemma 5.15. The proof is by induction on ℓ. For the base case $\ell = 1$, consider a string $w \in (\mathbb{F}_q^s)^n$ and a pair of distinct codewords $c_0, c_1 \in \mathsf{mult}_q^{(s)}(a_1, \ldots, a_n; k)$, and let $H = (\{0,1\}, E)$ be the corresponding agreement hypergraph. Then we have that

$$\mathsf{wt}(H) = \sum_{e \in E} \max\{|e|-1, 0\} = |\{i \in [n] \mid (c_0)_i = (c_1)_i = w_i\}| \le n - \Delta(c_0, c_1) < \frac{k}{s},$$

which proves the $\ell = 1$ case.

For the induction step, fix $1 < \ell < s$. We shall assume that Lemma 5.15 holds for any $1 \le \ell' < \ell$, and we shall show that it also holds for ℓ. Let $w \in (\mathbb{F}_q^s)^n$ be a string, and let $c_0, c_1, \ldots, c_\ell$ be distinct codewords in $\mathsf{mult}_q^{(s)}(a_1, \ldots, a_n; k)$. Without loss of generality, we may assume that $c_0 = 0$, since otherwise we can translate w and $c_0, c_1, \ldots, c_\ell$ by c_0. Let $H = (\{0, 1, \ldots, \ell\}, E = \{e_1, \ldots, e_n\})$ be the corresponding agreement hypergraph, and assume towards a contradiction that $\mathsf{wt}(H) \ge \frac{\ell}{s-\ell+1}\cdot k$.

Let $V := \mathrm{span}\{c_0, c_1, \ldots, c_\ell\}$, let $r := \dim(V)$, and note that $1 \le r \le \ell$ by assumption that $c_0 = 0$. Without loss of generality, we may further assume that $c_1, \ldots, c_r$ is a basis for V. Let $\mathcal{P}$ be a partition of $\{0, 1, \ldots, \ell\}$ into $r+1$ parts $P_0, P_1, \ldots, P_r$, where for $t \in \{0, 1, \ldots, r\}$,

$$P_t := \{j \in [\ell] \mid c_j \in \mathrm{span}\{c_0, c_1, \ldots, c_t\} \setminus \mathrm{span}\{c_0, c_1, \ldots, c_{t-1}\}\}.$$

For $i \in [n]$, let $A_i := \{v \in \mathsf{mult}_q^{(s)}(a_1, \ldots, a_n; k) \mid v_i = 0\}$. Then the main observation is the following.

Claim 5.17. *For any $i \in [n]$, it holds that $\dim(A_i \cap V) \ge \max\{|\mathcal{P}(e_i)| - 1, 0\}$, where $|\mathcal{P}(e_i)|$ denotes the number of parts in the partition intersecting e_i.*

Proof. Fix $i \in [n]$. The claim clearly holds if $e_i = \emptyset$, so assume next that $e_i \ne \emptyset$, and let $V_i = \mathrm{span}\{c_j | j \in e_i\}$ denote the span of all codewords in e_i.

Assume first that $w_i = 0$. In this case the i-th entry of all codewords in V_i is zero, and so $V_i \subseteq A_i \cap V$. Furthermore, we have that $c_0 = 0 \in e_i$, and so e_i intersects P_0. Suppose that e_i intersects additional t parts in the partition $\mathcal{P}$. Then e_i contains t linearly independent vectors, and so $t \le \dim(V_i) \le \dim(A_i \cap V)$. So in this case we have that $\dim(A_i \cap V) \ge t = |\mathcal{P}(e_i)| - 1$.

Next assume that $w_i \ne 0$, and let c_j be an arbitrary codeword in e_i. Then since all codewords in e_i agree on the i-th entry, we have that all codewords in e_i are contained in $c_j + A_i \cap V$, and so $\dim(V_i) \le \dim(A_i \cap V) + 1$. Furthermore, $0 \notin e_i$ and so e_i does not intersect P_0. Suppose that e_i intersects t parts in the partition $\mathcal{P}$. Then e_i contains t linearly independent vectors, and so $t \le \dim(V_i) \le \dim(A_i \cap V) + 1$. So in this case we also have that $\dim(A_i \cap V) \ge t - 1 = |\mathcal{P}(e_i)| - 1$. □

By the above claim, and using the same calculation as in the proof of Lemma 5.4, we have that

$$\begin{aligned}
\sum_{i\in[n]} \dim(A_i\cap V) &\geq \sum_{e\in E}\max\{|\mathcal{P}(e)|-1,0\} \\
&= \mathsf{wt}(H)-\sum_{t=0}^{r}\mathsf{wt}(H|_{P_t}) \\
&> \frac{\ell}{s-\ell+1}\cdot k-\sum_{t=0}^{r}\frac{|P_t|-1}{s-|P_t|+2}\cdot k \\
&> \frac{\ell}{s-\ell+1}\cdot k-\frac{\sum_{t=0}^{r}(|P_t|-1)}{s-\ell+2}\cdot k \\
&= \frac{\ell}{s-\ell+1}\cdot k-\frac{\ell-r}{s-\ell+1}\cdot k \\
&= \frac{r}{s-\ell+1}\cdot k,
\end{aligned}$$

where in the second inequality we used our assumption that $\mathsf{wt}(H)\geq\frac{\ell}{s-\ell+1}\cdot k$, and that by the induction hypothesis $\mathsf{wt}(H|_{P_t})<\frac{|P_t|-1}{s-|P_t|+2}\cdot k$ for any $t\in\{0,1,\dots,r\}$ (since $\mathcal{P}$ is a proper partition). Finally, recalling that $r\leq\ell$, the above inequality contradicts Lemma 5.16, which concludes the proof of this lemma. □

5.3 Reed-Solomon Codes over subfield evaluation points

In this section, we show that a certain *subcode* of Reed-Solomon Codes over subfield evaluation points is (efficiently) list decodable up to capacity with a *constant* list size (depending on the gap to capacity ϵ). For Reed-Solomon Codes over subfield evaluation points, Lemma 4.9 only gave a *linear* upper bound on the dimension of the list, and so the methods for reducing the list size from Section 5.2.1 do not apply in this setting.

The main observation that lets us reduce the list size in this setting is that the proof of Lemma 4.9 in fact shows that the list has a more refined low-dimensional *periodic* structure. Namely, there exists an $\mathbb{F}_q$-linear subspace $\hat{V}\subseteq\mathbb{F}_{q^s}$ of constant dimension $r=O(1/\epsilon)$ so that the following holds: Any polynomial $f(X)=\sum_{i=0}^{k-1}f_i\in\mathbb{F}_{q^s}[X]$ whose associated codeword is in the list satisfies that for any $i\in\{0,1,\dots,k-1\}$, given the first i coefficients $f_0,f_1,\dots,f_{i-1}\in\mathbb{F}_{q^s}$, the next coefficient f_i belongs to an affine shift of $\hat{V}$ (where the affine shift only depends on $f_0,f_1,\dots,f_{i-1}$).[5] This motivates the following definition. In what follows, for a vector $v\in\mathbb{F}^{sk}$, we let $v=(v_{s,1},\dots,v_{s,k})$, where $v_{s,i}$ denotes the i-th block of v of length s.

Definition 5.18 (Periodic subspace)**.** *Let $\mathbb{F}$ be a finite field, and let k,s,r be positive integers. A linear subspace $V\subseteq\mathbb{F}^{sk}$ is a (k,s,r)-periodic subspace if there exists a linear subspace $\hat{V}\subseteq\mathbb{F}^s$ of dimension r, so that the following holds: For any $v\in V$ and $i\in\{1,2,\dots,k\}$, there exists $z_i\in\mathbb{F}^s$, that only depends on $v_{s,1},\dots,v_{s,i-1}$, so that $v_{s,i}\in z_i+\hat{V}$.*

[5]This follows by recalling from the proof of Lemma 4.9 that for any $i\in\{0,1,\dots,k-1\}$, f_i must satisfy that $B(f_i)=-y_i$, where y_i only depends on $f_0,f_1,\dots,f_{i-1}$, and $B(X)=\sum_{\ell=0}^{r-1}B_\ell(0)X^{q^\ell}$. Consequently, we have that $f_i\in z_i+\hat{V}$, where $\hat{V}\subseteq\mathbb{F}_{q^s}$ denotes the set of roots of $B(X)$, and $z_i\in\mathbb{F}_{q^s}$ is such that $B(z_i)=-y_i$. Finally, note that by the linear structure of $B(X)$, $\hat{V}$ is an $\mathbb{F}_q$-linear subspace of dimension at most $r-1$.

By the above, the Reed-Solomon Code over subfield evaluation points $\mathrm{RS}_{q^s}(a_1, \ldots, a_n; k)$ for $a_1, \ldots, a_n \in \mathbb{F}_q$ is list decodable from an $(1-R-\epsilon)$-fraction of errors, where the polynomials whose associated codewords are in the list are contained in a (k, s, r)-periodic subspace for $r = O(1/\epsilon)$ (when viewing each polynomial $f(X) \in \mathbb{F}_{q^s}[X]$ of degree smaller than k as a concatenation of k coefficients in $\mathbb{F}_{q^s}$, identifying $\mathbb{F}_{q^s}$ with $\mathbb{F}_q^s$ via some $\mathbb{F}_q$-linear bijection, and identifying $(\mathbb{F}_q^s)^k$ with $\mathbb{F}_q^{sk}$ in the natural way). This periodic structure can lead to an $\mathbb{F}_q$-linear subcode with lists of *constant dimension* (and so also of *constant size* by Lemma 5.12) when only keeping in the code codewords whose associated polynomials fall in a *periodic evasive subspace*, which is a large subspace whose intersection with any periodic subspace has a low dimension.

Definition 5.19 (Periodic evasive subspace)**.** *Let $\mathbb{F}$ be a finite field, and let k, s, r, t be positive integers. A linear subspace $W \subseteq \mathbb{F}^{sk}$ is a (k, s, r, t)-periodic evasive subspace if $\dim(W \cap V) \leq t$ for any (k, s, r)-periodic subspace $V \subseteq \mathbb{F}^{sk}$.*

One way to obtain a periodic evasive subspace is via the notion of a *subspace design*, which is a collection of large subspaces of $\mathbb{F}^s$, whose intersection with any low-dimensional subspace has low dimension *on average*.

Definition 5.20 (Subspace design)**.** *Let $\mathbb{F}$ be a finite field, and let k, s, r, t be positive integers. A collection of k subspaces $H_1, \ldots, H_k \subseteq \mathbb{F}^s$ is an (r, t)-subspace design if $\sum_{i=1}^k \dim(\hat{V} \cap H_i) \leq t$ for any linear subspace $\hat{V} \subseteq \mathbb{F}^s$ of dimension r.*

The next claim shows that if $H_1, \ldots, H_k$ is a subspace design, then $H_1 \times H_2 \times \cdots \times H_k$ is a periodic evasive subspace with the same parameters.

Claim 5.21. *Let $\mathbb{F}$ be a finite field, and let k, s, r, t be positive integers. Suppose that a collection of k subspaces $H_1, \ldots, H_k \subseteq \mathbb{F}^s$ is an (r, t)-subspace design. Then $W := H_1 \times H_2 \times \cdots \times H_k \subseteq \mathbb{F}^{sk}$ is a (k, s, r, t)-periodic evasive subspace.*

Proof. Let $V \subseteq \mathbb{F}^{sk}$ be a (k, s, r)-periodic subspace, and let $V' := V \cap W$. Since both V and W are linear subspaces over $\mathbb{F}$, then so is V'. Thus, to show that V' has dimension at most t, it suffices to show that V' has size at most q^t.

To see the above, recall that since V is a (k, s, r)-periodic subspace, there exists a subspace $\hat{V} \subseteq \mathbb{F}^s$ of dimension at most r, so that for any $v \in V$ and $i \in \{1, \ldots, k\}$, there exists $z_i \in \mathbb{F}^s$ that only depends on $v_{s,1}, \ldots, v_{s,i-1}$, so that $v_{s,i} \in z_i + \hat{V}$. By the definition of W, this implies in turn that for any $v \in V' = V \cap W$ and $i \in \{1, \ldots, k\}$, there exists $z_i \in \mathbb{F}^s$ that only depends on $v_{s,1}, \ldots, v_{s,i-1}$, so that $v_{s,i} \in (z_i + \hat{V}) \cap H_i$. Thus, given the values of $v_{s,1}, \ldots, v_{s,i-1}$, there are at most $q^{\dim(\hat{V} \cap H_i)}$ possible values for $v_{s,i}$. Consequently, V' has total size at most

$$\prod_{i=1}^{k} q^{\dim(\hat{V} \cap H_i)} = q^{\sum_{i=1}^k \dim(\hat{V} \cap H_i)} \leq q^t,$$

where the upper bound follows since $H_1, \ldots, H_k$ is a (k, s, r, t)-subspace design. □

The next theorem gives an explicit construction of a subspace design with large subspaces $H_1, \ldots, H_k$ whose total intersection with any low-dimensional subspace is small. Interestingly, the construction relies on the list decoding properties of multiplicity codes!

Theorem 5.22. *For any prime power q, a prime d, and positive integers k, s, r satisfying that $s < 2rk$, $\max\{2r, s\} <$ char$(\mathbb{F}_q)$, and $k \leq \frac{q^d - q}{d}$, there exists an explicit construction of an $(r, \frac{s}{d})$-subspace design $H_1, \ldots, H_k \subseteq \mathbb{F}_q^s$, where each subspace H_i has co-dimension $2rd$.*

Proof. We first note that Lemma 5.16 implies a version of this theorem for $d = 1$ and $k \leq q$. To see this, let $a_1, a_2, \ldots, a_k$ be distinct elements in $\mathbb{F}_q$, and for $i \in [k]$, let

$$H_i := \left\{ f(X) \in (\mathbb{F}_q)_{<s}[X] \;\mid\; f(a_i) = f^{(1)}(a_i) = \cdots = f^{(2r-1)}(a_i) = 0 \right\},$$

where we view a polynomial $f(X) \in (\mathbb{F}_q)_{<s}[X]$ as a vector of coefficient in $\mathbb{F}_q^s$. Then each H_i has co-dimension at most $2r$ since any requirement of the form $f^{(j)}(a_i) = 0$ imposes a single linear constraint on the coefficients of polynomials in H_i. Let $\hat{V} \subseteq (\mathbb{F}_q)_{<s}[X]$ be an r-dimensional subspace. For $i \in [n]$, let A_i be the set of codewords in $\mathsf{mult}_q^{(2r)}(a_1, \ldots, a_k; s)$ corresponding to the polynomials in H_i, and let V be the set of codewords in this code corresponding to the polynomials in $\hat{V}$. Then applying Lemma 5.16 with k distinct evaluation points, multiplicity parameter $2r$, and degree parameter s, we have that

$$\sum_{i=1}^{n} \dim(H_i \cap \hat{V}) = \sum_{i=1}^{n} \dim(A_i \cap V) \leq \frac{r}{r+1} \cdot s \leq s.$$

To extend the proof to $d > 1$, and obtain a larger collection of subspaces H_i, one can pick the a_i's from the extension field $\mathbb{F}_{q^d}$. In more detail, let $a_1, a_2, \ldots, a_k$ be elements in $\mathbb{F}_{q^d}$ so that all elements of the form $a_i^{q^\ell}$ for $i \in [k]$ and $\ell \in \{0, 1, \ldots, d-1\}$ are distinct. Note that if d is a prime, then any element $a \in \mathbb{F}_{q^d} \setminus \mathbb{F}_q$ has distinct powers $a, a^q, \ldots, a^{q^{d-1}}$. So in this case, one can find at least $k = \frac{q^d - q}{d}$ elements $a_1, \ldots, a_k$ inside $\mathbb{F}_{q^d}$ which satisfy the above requirement.

Similarly to the $d = 1$ case, for $i \in [k]$, let

$$H_i := \left\{ f(X) \in (\mathbb{F}_q)_{<s}[X] \;\mid\; f(a_i^{q^\ell}) = f^{(1)}(a_i^{q^\ell}) = \cdots = f^{(2r-1)}\left(a_i^{q^\ell}\right) = 0 \text{ for any } \ell = 0, 1, \ldots, d-1 \right\},$$

where we view a polynomial $f(X) \in (\mathbb{F}_q)_{<s}[X]$ as a vector of coefficient in $\mathbb{F}_q^s$. Then it can be verified that each H_i has co-dimension at most $2rd$. Moreover, the proof of Lemma 5.16 shows that

$$\sum_{i \in [k]} \dim(H_i \cap \hat{V}) \leq \frac{r}{d \cdot (r+1)} \cdot s \leq \frac{s}{d}$$

for any r-dimensional subspace $\hat{V} \subseteq (\mathbb{F}_q) < s[X]$. $\square$

By setting $d = \frac{\epsilon s}{2r}$ (assuming that $r \leq \frac{\epsilon s}{2}$) in the above Theorem 5.22, we get the following corollary.

Corollary 5.23 (Explicit subspace design)**.** *For any prime power q, $\epsilon > 0$, and positive integers k, s, r satisfying that $s < 2rk$, $s <$ char$(\mathbb{F}_q)$, $q^s \geq (k \cdot \frac{\epsilon s}{2r})^{2r/\epsilon}$, and $r < \frac{\epsilon s}{2}$, there exists an explicit construction of an $(r, \frac{2r}{\epsilon})$-subspace design $H_1, \ldots, H_k \subseteq \mathbb{F}_q^s$, where each subspace H_i has co-dimension ϵs.*

Recall that by Lemma 4.9, the Reed-Solomon Code over subfield evaluation points $\mathrm{RS}_{q^s}(a_1, \ldots, a_n; k)$ for $a_1, \ldots, a_n \in \mathbb{F}_q$ and $s = \Theta(1/\epsilon^2)$ is list decodable from an $(1 - R - \epsilon)$-fraction of errors, where the polynomials whose associated codewords are in the list are contained in a (k, s, r)-periodic subspace for $r = O(1/\epsilon)$. Let $H_1, \ldots, H_k$ be the subspace design given by the above corollary for these parameters, and recall that by Claim 5.21, $W := H_1 \times H_2 \times \cdots \times H_k$ is a $(k, s, r, \frac{2r}{\epsilon})$-periodic evasive subspace. Let $C \subseteq \mathrm{RS}_{q^s}(a_1, \ldots, a_n; k)$ be the subcode which consists of all codewords associated with polynomials $f(X) = \sum_{i=0}^{k-1} f_i X^i \in \mathbb{F}_{q^s}[X]$ so that $f_i \in H_{i+1}$ for any $i \in \{0, 1, \ldots, k-1\}$. Then C has rate $R - \epsilon$ and is list-decodable from an $(1 - R - \epsilon)$-fraction of errors with list of *dimension* $O(1/\epsilon^2)$ (which by Lemma 5.12, also implies that this subcode has a list of *size* at most $\exp(\mathrm{poly}(1/\epsilon))$).

Finally, note that C can be list decoded up to capacity in *polynomial time* as follows. First note that by Lemma 4.9, one can find a basis for the (k, s, r)-periodic subspace V containing the list of $\mathrm{RS}_{q^s}(a_1, \ldots, a_n; k)$ in polynomial time by solving a system of linear equations. Then one can find a basis for the $O(1/\epsilon^2)$-dimensional subspace V' containing the list of C in polynomial time by solving another system of linear equations to find the intersection of V with the subspace containing the encodings of all degree k polynomials with coefficients in $W = H_1 \times \cdots \times H_k$. Finally, since the size of V' is $q^{O(1/\epsilon^2)}$, one can find all elements in the list in time $q^{O(1/\epsilon^2)}$, which is polynomial in the block length for a constant ϵ and $q = \mathrm{poly}(n)$.

5.4 Bibliographic notes

Higher-order MDS codes. The notion of higher-order MDS codes was introduced independently by Brakensiek, Gopi, and Makam [BGM22] and Roth [Rot22]. In [Rot22], higher-order MDS codes were defined as codes achieving the *generalized Singleton Bound*, while in [BGM22], they were defined based on the *dimension of the intersection of the subspaces* spanned by subsets of columns of their generator matrix. The definition of higher-order MDS codes presented in Section 5.1.1, based on *size of the intersection of the zero-patterns* of columns of the generator matrix, was given later in another paper of Brakensiek, Gopi, and Makam [BGM25]. In the same paper it was also shown that all the above three definitions are equivalent (up to duality). Fact 5.1 was called the MDS condition in [DSY14], and its proof seems to be folklore.

Hypergraph connectivity. Király [Kir03] introduced the notion of edge connectivity for hypergraphs, and proved the Menger Theorem for hypergraphs which says that edge connectivity is equivalent to having multiple edge-disjoint paths between any pair of vertices (this theorem generalizes the well-known Menger theorem for graphs [Men27]). The notion of (weak) partition connectivity for hypergraphs is well-studied in combinatorics [FKK03a, FKK03b, Kir03] and optimization [JMS03, FK08, Fra11, CX18], and it generalizes the well-known notion of graph partition connectivity (In particular, the well-known Nash-Williams-Tutte Tree-Packing theorem shows that in graphs, partition connectivity is equivalent to having edge-disjoint *packing trees*). Our proof of Lemma 5.4 follows the proof of [AGG+25, Lemma 2.4]. Theorem 5.5 about hypergraph orientaitons is stated most explicitly in [Fra11], but is also implicit in [Kir03, FKK03b].

Reed-Solomon Codes over random evaluation points. Shangguan and Tamo [ST20] conjectured that random Reed-Solomon Codes are higher-order MDS codes and proved several special cases of this conjecture, and this conjecture was proven in full in [BGM25]. We present an alternative view of their proof, based on *hypergraph connectivity*, that was presented by Alrabiah, Guo, Guruswami, Li, and Zihan [AGG+25], and was based in turn on a hypergraph perspective on list-decoding that was first suggested by Guo, Li, Shangguan, Tamo, and Wootters [GLS+24]. Lemma 5.6 which connects weak partition connectivity with *generalized zero-patterns* follows [AGG+25, Corollary A.4]. Theorem 5.9 which says that duals of higher-order MDS codes (according to the GZP-based definition) satisfy the generalized Singleton Bound follows the proof of [AGG+25, Theorem 2.11]. The GM-MDS Theorem (Theorem 5.10) was proven independently by Lovett [Lov21] and Yildiz and Hassibi [YH19], following a conjecture made by Dau, Song, and Yuen [DSY14]. Corollary 5.11 follows the proof of [BGM25, Proposition 4.5].

An exponential lower bound on the field size of higher-order MDS codes was shown in [BGM22, Corollary 4.2], while in [AGG+25] it was shown that random Reed-Solomon Codes over a *linear-size alphabet approximately* attain the generalized Singleton Bound [GZ23, AGG+25]. An alternative approach for showing that random Reed-Solomon Codes attain the generalized Singleton Bound was given by Levi, Mosheiff, and Shagrithaya [LMS25]. This approach was based on showing that random Reed-Solomon Codes behave similarly to random linear codes with respect to list-decodability properties (and more generally, any *local* property, see below).

Multiplicity codes and Reed-Solomon Codes with subfield evaluation points. The probabilistic argument, presented in Section 5.2.1, which shows that any linear code that is list decodable up to capacity with a constant dimensional list is also list-decodable up to capacity with a *constant list size* was discovered by Kopparty, Ron-Zewi, Saraf, and Wootters [KRSW23, Lemma 3.1] (some improvements and extensions were later given by Tamo [Tam24]). The proof that multiplicity codes (as well and FRS codes) attain the *generalized Singleton Bound*, presented in Section 5.2.2, was discovered by Chen and Zhang [CZ25], and an algorithmic version was given in [AHS26]. Lemma 5.16 that is used in this proof was proven by Guruswami and Kopparty [GK16b]. Brakensiek, Chen, Dhar, and Zhang [BCDZ25] extended the proof of [CZ25] by showing that multiplicity codes (as well as Folded Reed-Solomon Codes) satisfy any local property that is satisfied by random linear codes (Informally speaking, local properties [MRR+24, LMS25] are properties for which non-membership can be certified using a small number of codewords, and they include list decoding and list recovery as special cases).

The approach for reducing the list size for a subcode of Reed-Solomon Codes over subfield evaluation points using subspace designs, presented in Section 5.3, was suggested by Guruswami and Xing [GX22].

AG codes. A disadvantage of all the explicit capacity-achieving list decodable codes we discussed so far is that their alphabet is a large polynomial in the block length (at least n^{1/ϵ^2}, where n is the block length, and ϵ is the gap to capacity). To reduce the alphabet size to a constant, one can resort to algebraic-geometric (AG) codes. Loosely speaking, AG codes are an extension of Reed-Solomon Codes to the setting of *function fields*, in which codewords correspond to the evaluation of certain functions on rational points of some *algebraic curve* over some finite field $\mathbb{F}$. Reed-Solomon Codes then correspond to the special case in which the algebraic curve is the affine line over $\mathbb{F}$ and the functions are low degree polynomials on the line. As the affine line has only $|\mathbb{F}|$ rational points, this forces the alphabet of the Reed-Solomon Code to be at least as large as its block length. The advantage of AG codes is that algebraic curves can generally have many more rational points, and so the alphabet can potentially be much smaller than the block length. In particular, by a certain choice of parameters, the alphabet size can be made constant for increasing block lengths.

Algebraic-geometric codes were first introduced by Goppa [Gop83]. Guruswami and Sudan [GS99] showed that their algorithm for list-decoding of Reed-Solomon Codes up to the *Johnson Bound* , presented in Section 3.3, can be extended to the setting of AG codes. Later, Guruswami and Xing [GX22] introduced versions of *folded* AG codes, as well as AG codes with *subfield evaluation points*, and used these to extend the results that Folded Reed-Solomon Codes and Reed-Solomon Codes over subfield evaluation points achieve list-decoding capacity, presented in Section 4.2 (for the related family of multiplicity codes) and Section 4.3, respectively, to the setting of AG codes. More recently, Brakensiek, Dhar, Gopi, and Zhang [BDGZ25] extended the result that random Reed-Solomon Codes achieve the *generalized Singleton Bound*, presented in Section 5.1, to the AG code setting. The approach for reducing the list size for a subcode of Reed-Solomon Codes using subspace designs, presented in Section 5.3, was also used by Guruswami and Xing [GX22] to reduce the list-size of a subcode of AG codes to nearly-constant (on the order of $\log^*(n)$), and later to a constant by Guo and Ron-Zewi [GR22].

6 Near-linear time list decoding

In this section, we discuss faster versions of the list decoding algorithms for Reed-Solomon Codes and multiplicity codes. Recall that we have already seen versions of such algorithms in Sections 3 and 4 that run in polynomial time in the input size, and decode Reed-Solomon Codes up to the Johnson Bound and multiplicity codes up to list decoding capacity. The goal now is to see alternative algorithms for these problems whose time complexity is much faster - they run in nearly linear time in the input size, in certain regime of parameters. We start with such an algorithm for Reed-Solomon Codes in Section 6.1 below, followed by a fast list decoding algorithm for multiplicity codes in Section 6.2.

6.1 Fast list decoding of Reed-Solomon Codes up to Johnson Bound

In this section, we present a near-linear time list decoding algorithm for Reed-Solomon Codes up to a radius approaching the Johnson Bound. Specifically, in what follows, fix a prime power q, distinct evaluation points $a_1, \ldots, a_n \in \mathbb{F}_q$, and a positive integer $k < n$, and let $\mathrm{RS}_q(a_1, \ldots, a_n; k)$ be the corresponding Reed-Solomon Code. We shall show an algorithm which for any $\epsilon > 0$, list decodes this code from $(1 - \sqrt{\frac{k}{n}} - \epsilon)$-fraction of errors in time $n \cdot \mathrm{poly}(\frac{n}{k}, \log(q), \log(n), \frac{1}{\epsilon})$. Thus, when both ϵ and the rate $R := \frac{k}{n}$ of the code are constant, the running time of this algorithm is $n \cdot \mathrm{poly}(\log(n), \log(q))$. Note that as opposed to the polynomial-time list-decoding algorithm for Reed-Solomon Codes presented in Section 3.3, the algorithm we present in this section does not achieve the Johnson Bound, but it only approaches it.

Overview. Recall that the polynomial-time algorithm for list decoding of Reed-Solomon Codes up to the Johnson Bound, presented in Section 3.3 (cf., Figure 3), consists of two main steps, namely, that of constructing a non-zero bivariate polynomial that *encodes* all the close enough codewords in its belly, and that of finding roots of a bivariate polynomial. Moreover, the first step is done by setting up and solving a linear system, while the second step relies on some off the shelf root computation algorithm for bivariate polynomials. While the time complexity of both these steps turns out to be polynomially bounded, it is unclear whether the time complexity can be improved to something that is nearly linear in the input size. In fact, even for the seemingly simpler linear system that appears in the interpolation step of the unique decoding algorithm for Reed-Solomon Codes, presented in Section 3.1, it is unclear if the system can be solved in nearly linear time. Indeed, even fast unique decoding algorithms for Reed-Solomon Codes, that have been known since the 60's rely on some non-trivial insights.

The fast list decoding algorithm we present in this section has two main technical components. The first component shows that the interpolation step in the algorithm of Figure 3 can indeed be performed in nearly linear time using appropriate lattices over the univariate polynomial ring. The second component, essentially uses (off the shelf) an algorithm of Roth and Ruckenstein [RR00] for computing low degree roots of bivariate polynomials. We now discuss these ideas in some more detail.

6.1.1 Fast implementation of the interpolation step

Let $w \in \mathbb{F}_q^n$ be the received word. Our goal in this step is to find, in nearly linear time, a non-zero bivariate polynomial $Q(X, Y)$ of small $(1, k)$-weighted degree, such that for every i, Q has a high multiplicity zero at (a_i, w_i). To this end, we start with some definitions that, apriori, might appear to be slightly mysterious.

Let $R(X)$ be the unique polynomial of degree at most $n - 1$ such that for every i, $R(a_i) = w_i$. R can be obtained in nearly linear time using the standard fast interpolation algorithms for univariate polynomials based on the Fast Fourier Transform (e.g Algorithm 10.22 in [vzGG13]). The utility of R stems from the following simple observation: if $f(X)$ is such that $f(a_i) = w_i$, then $X - a_i$ divides $f(X) - R(X)$, or equivalently, $f(X) \equiv R(X) \mod (X - a_i)$. Let $\ell, m \in \mathbb{N}$ be parameters with $\ell \leq m$ to be set later, and for every $j \in \{0, 1, \ldots, m-1\}$, let $P_j(X, Y)$ be the bivariate polynomial defined as follows.

$$\begin{aligned} \forall j \in \{0, 1, \ldots, \ell - 1\}, \qquad & P_j(X, Y) := (Y - R(X))^j \cdot \prod_{i=1}^{n} (X - a_i)^{\ell - j} \\ \forall j \in \{\ell, \ell + 1, \ldots, m - 1\}, \quad & P_j(X, Y) := (Y - R(X))^j \end{aligned}$$

Let us now consider the set of bivariate polynomials in the $\mathbb{F}_q[X]$-linear span of $P_0, P_1, \ldots, P_{m-1}$, that is,

$$\mathcal{L}^{(\ell,m)} := \left\{ \sum_{j=0}^{m-1} g_j(X) \cdot P_j(X, Y) : g_j \in \mathbb{F}_q[X] \right\} .$$

The set $\mathcal{L}^{(\ell,m)}$ could be alternatively thought of as follows. We view each P_j as a univariate in Y with coefficients from the ring $\mathbb{F}_q[X]$; and thus, P_j can be thought of as a vector of dimension m (namely the coefficient vector, when viewed as a univariate in Y), where every entry is an element of $\mathbb{F}_q[X]$. The set $\mathcal{L}^{(\ell,m)}$ is then the $\mathbb{F}_q[X]$ linear span of these vectors. Furthermore, we note that since the Y-degree of P_j equals j (and hence is distinct), the coefficient vectors of P_j (when viewed as univariates in Y) are linearly independent over the field $\mathbb{F}_q(X)$.

Recall that over the ring of integers, the notion of a lattice is defined as follows: for any postive integer n and linearly independent n dimensional vectors $v_0, \ldots, v_m$ over real numbers, the lattice generated by $v_0, v_1, \ldots, v_m$ is defined as $\{\sum_i a_i v_i : a_i \in \mathbb{Z}\}$. There are some obvious syntactic similarities in the above definition of integer lattices and the definition of the set $\mathcal{L}^{(\ell,m)}$: in both cases, we have *integral domains* (integers and $\mathbb{F}_q[X]$ respectively) and we take a set of vectors that are linearly independent over an underlying field containing these domains (the field of real numbers and the field of fractions $\mathbb{F}_q(X)$ respectively) and consider their weighted linear combinations, where the weights come from the respective integral domains. Furthermore, both of the integral domains are in fact *Euclidean*. Given these syntactic similarities, we follow the convention of referring to $\mathcal{L}^{(\ell,m)}$ as a lattice generated by $P_0, P_1, \ldots, P_{m-1}$ throughout this survey.

The following lemma shows the relevance of the lattice $\mathcal{L}^{(\ell,m)}$ to the interpolation step of the list decoding algorithm for Reed-Solomon Codes.

Lemma 6.1. *Let $Q(X, Y)$ be any non-zero polynomial in the lattice $\mathcal{L}^{(\ell,m)}$. Then for every $i \in [n]$, $Q(X, Y)$ vanishes with multiplicity at least ℓ at (a_i, w_i).*

Proof. Let u, v be non-negative integers such that $u + v < \ell$. Then, for the proof of the lemma, it suffices to show that the Hasse Derivative $Q^{(u,v)}$ of Q vanishes at (a_i, w_i). Recall that Q is of the form $\sum_{j=0}^{m-1} g_j(X) P_j(X, Y)$. Thus, by linearity of Hasse Derivatives, it suffices to show that the (u, v)-Hasse Derivative of every summand $g_j(X) P_j(X, Y)$ vanishes at (a_i, w_i). From the product rule of Hasse Derivatives (cf., Lemma 2.7), we have that

$$(g_j(X) P_j(X, Y))^{(u,v)} = \sum_{0 \leq u' \leq u, 0 \leq v' \leq v} P_j(X, Y)^{(u',v')} \cdot g_j(X)^{(u-u', v-v')},$$

and it therefore suffices to show that $P_j^{(u',v')}$ vanishes at (a_i, w_i) for any $u' + v' < \ell$.

To see the above, we consider two cases depending on j, since the structure of P_j changes when $j \geq \ell$.

- $j \geq \ell$: From its definition, we have $P_j = (Y - R(X))^j$. We now argue that for any non-negative u', v' such that $u' + v' < \ell$, $P_j^{(u',v')}$ is divisible by $(Y - R(X))$. This would complete the argument since $(Y - R(X))$ (and hence $P_j^{(u',v')}$) evaluates to zero at (a_i, w_i).

  To see the divisibility claim, we consider $P_j(X + Z_1, Y + Z_2) = (Y + Z_2 - R(X + Z_1))^j$. Viewing $R(X + Z_1)$ as a polynomial in Z_1, we get that $R(X + Z_1)$ equals $R(X) + Z_1 \cdot \Gamma(X, Z_1)$ for some bivariate polynomial Γ. Thus, $P_j(X + Z_1, Y + Z_2) = ((Y - R(X)) - (Z_1\Gamma(X, Z_1) - Z_2))^j$. From the binomial theorem, we further get that

$$P_j(X + Z_1, Y + Z_2) = \sum_{t=0}^{j} \binom{j}{t} (Y - R)^{j-t} (Z_1\Gamma - Z_2)^t.$$

  By definition of Hasse Derivatives, we have that $P_j^{(u',v')}$ equals the coefficient of $Z_1^{u'} Z_2^{v'}$ in the above expansion. Note that every monomial in the polynomial $(Z_1\Gamma - Z_2)^t$ has total degree at least t in the Z variables. Thus, the coefficient of $Z_1^{u'} Z_2^{v'}$ in $P_j(X + Z_1, Y + Z_2)$ equals the coefficient of $Z_1^{u'} Z_2^{v'}$ in the truncated sum $\sum_{t=0}^{u'+v'} \binom{j}{t} (Y - R)^{j-t}(Z_1\Gamma - Z_2)^t$. Furthermore, from $t \leq u' + v' < \ell$ and $\ell \leq j$, we get that each of the terms in the sum above is divisible by $(Y - R)$ (which has Z-degree zero). Thus, $P_j^{(u',v')}$ must be divisible by $(Y - R)$.

- $j < \ell$: This case also proceeds along the lines of the above case, albeit with a little more care. For $j < \ell$, we have from the definition of P_j that $P_j(X, Y) = (Y - R(X))^j \prod_{t=1}^{n} (X - a_t)^{\ell - j}$. For notational convenience we can re-write P_j as $P_j(X, Y) = (Y - R(X))^j \cdot (X - a_i)^{\ell-j} \cdot \prod_{t \neq i} (X - a_t)^{\ell-j}$. The advantage of this rearrangement is that both $(Y - R)$ and $(X - a_i)$ are zero at (a_i, w_i). Thus, counted with multiplicities, P_j has at least ℓ factors that vanish at (a_i, w_i). Intuitively, this should indicate that the multiplicity of P_j at (a_i, w_i) must be at least ℓ and hence its (u', v')-Hasse Derivative should vanish at (a_i, w_i) for all $u' + v' < \ell$. This intuition can be made formal by using the definition of Hasse Derivatives to conclude that $P_j^{(u',v')}$ can be written as a sum of terms, each of which is divisible by either $(Y - R(X))$ or $(X - a_i)$ (or both). We skip the formal proof of this claim here.

This completes the proof of the lemma. □

Lemma 6.1 shows that every non-zero polynomial in the lattice $\mathcal{L}^{(\ell,m)}$ vanishes with high multiplicity at (a_i, w_i) and hence satisfies the interpolation conditions in the algorithm of Figure 3 (for a careful choice of ℓ). Thus, if we can somehow find a polynomial in this lattice that is non-zero and has low $(1, k)$-weighted degree in nearly linear time, we would have made excellent progress towards a fast list decoder for Reed-Solomon Code.

The main insight is to associate a notion of *norm* or *length* to the vectors in the lattice $\mathcal{L}^{(\ell,m)}$ such that under this norm, the question of finding a non-zero low $(1, k)$-weighted degree polynomial $Q(X, Y)$ in this lattice is precisely the question of finding a *short* (under this norm) non-zero vector in this lattice. Thus, proving a bound on the norm of this short vector and finding one in nearly-linear time will together complete a fast implementation of the interpolation step. We now discuss the details, starting with the definition of the notion of lattices over the univariate polynomial ring and the length of the vectors in the lattice.

Lattices over the univariate polynomial ring. We start with a couple of definitions. Let $\mathbb{F}$ be a finite field, and let $\mathcal{P} = \{P_0, P_1, \ldots, P_{m-1}\} \subseteq \mathbb{F}[X]^r$ be a set of m vectors in $\mathbb{F}[X]^r$. A lattice $\mathcal{L}$ generated by

$\mathcal{P}$ over the ring $\mathbb{F}[X]$ is defined as

$$\mathcal{L} := \left\{ \sum_{i=0}^{m-1} g_i(X) \cdot P_i : g_i(X) \in \mathbb{F}[X] \right\} .$$

A set $\mathcal{P} \subseteq \mathbb{F}[X]^r$ is said to be a basis for a lattice $\mathcal{L}$ in $\mathbb{F}[X]^r$ if $\mathcal{P}$ generates the lattice $\mathcal{L}$ in the above sense, and the vectors in $\mathcal{P}$ are linearly independent over the field $\mathbb{F}(X)$ of rational functions in X over $\mathbb{F}$. The size of a basis of a lattice is a fundamental property of the lattice and in particular, all bases of a lattice have the same size. Note that the vectors in $\mathcal{P}$ can alternatively be thought of as coefficient vectors of bivariate polynomials in $\mathbb{F}[X, Y]$ when they are viewed as univariates in Y. We switch between these views in the course of discussion in this survey as per convenience.

For our applications in this survey, we confine ourselves to lattices that are full rank in the sense that they are generated by a basis $\mathcal{P} \subseteq \mathbb{F}[X]^m$ such that $|\mathcal{P}| = m$, i.e. the size of the basis equals the dimension of the ambient space. For instance, we note that the lattice $\mathcal{L}^{(\ell,m)}$ defined earlier in this section is full rank. To see this, observe that the generators of this lattices, i.e the bivariate polynomials $P_0, P_1, \ldots, P_{m-1}$ have distinct Y-degrees, and so they are linearly independent over $\mathbb{F}_q(X)$.

For a basis $\mathcal{P}$ of a full rank lattice $\mathcal{L}$ over $\mathbb{F}[X]$, we define the determinant of $\mathcal{P}$, denoted by $\det(\mathcal{P})$ to be the determinant of the square matrix whose columns are the vectors in $\mathcal{P}$. Recall that $\det(\mathcal{P})$ is a polynomial in the variable X. As is the case with full rank lattices over the ring of integers, the determinants of all bases for a lattice $\mathcal{L}$ are all equal to each other, and this is a fundamental property of the lattice $\mathcal{L}$, that we denote by $\det(\mathcal{L})$.

Next we define a notion of norm of vectors in these lattices.

Definition 6.2 (Degree norm)**.** *For any vector $P = (p_0, p_1, \ldots, p_{r-1}) \in \mathbb{F}[X]^r$, we define the degree norm of P, denoted as $\mu(P)$ as*

$$\mu(P) := \max_{j \in \{0,1,\ldots,r-1\}} \deg(p_j) ,$$

where $\deg(p_j)$ is the maximum degree (in X) of any entry $p_j \in \mathbb{F}[X]$ of P.

In other words, $\mu(P)$ is the degree in the variable X of the bivariate polynomial $\sum_{j=0}^{r-1} p_j(X) Y^j$.

The following theorem is a combination of the key technical result of Alekhnovich in [Ale05] and a classical theorem of Minkowski for lattices. The theorem is implicit in [Ale05] and we refer to Section 3.3 in [GHKS24] for a more detailed discussion on this.

Theorem 6.3 ([Ale05])**.** *Let $\mathcal{L} \subseteq \mathbb{F}[X]^m$ be a full rank lattice over $\mathbb{F}[X]$ generated by a basis $\mathcal{P}$. Then, there is a non-zero vector v in $\mathcal{L}$ such that*

$$\mu(v) \leq \frac{1}{m} \deg(\det(\mathcal{L})) .$$

Moreover, there is an algorithm that when given the basis $\mathcal{P}$ as input finds a non-zero vector $v \in \mathcal{L}$ which satisfies the above inequality, and runs in time $\tilde{O}(\mu(\mathcal{P})) \cdot \mathrm{poly}(m)$, where $\mu(\mathcal{P})$ equals $\max_{P \in \mathcal{P}} \mu(P)$.

Given this brief digression to the properties of lattices, we now get back to the task of obtaining a fast implementation of the interpolation step of the algorithm of Figure 3, and prove the following lemma.

Lemma 6.4. *Let $P_0(X, Y), \ldots, P_{m-1}(X, Y)$ be the polynomials and let $\mathcal{L}^{(\ell,m)}$ be the lattice as defined earlier in this section with respect to a received word $w \in \mathbb{F}_q^n$ and evaluation points $a_1, \ldots, a_n \in \mathbb{F}_q$, and* *let $k < n$ be a positive integer. Then, there is a non-zero polynomial $Q(X,Y)$ in $\mathcal{L}^{(\ell,m)}$ of $(1,k)$-weighted degree at most*

$$\left(\frac{n(\ell+1)\ell}{2m} + \frac{(m-1)k}{2}\right).$$

Moreover, there is an algorithm that when given the coefficient vectors of $P_0, P_1, \ldots, P_{m-1}$ as input, outputs the coefficient vector of Q in time $\tilde{O}(n) \cdot \mathrm{poly}(m)$.

Proof. In order to argue about the $(1,k)$-weighted degree of polynomials in $\mathcal{L}^{(\ell,m)}$, it would be helpful to instead work with the following related lattice $(\mathcal{L}')^{(\ell,m)}$ generated by the $\mathbb{F}_q[X]$ span of the bivariates $P'_0, P'_1, \ldots, P'_{m-1}$ defined as

$$\begin{aligned}
\forall j \in \{0,1,\ldots,\ell-1\}, \qquad & P'_j(X,Y) := (X^kY - R(X))^j \cdot \prod_{i=1}^{n}(X-a_i)^{\ell-j} \\
\forall j \in \{\ell,\ell+1,\ldots,m-1\}, \qquad & P'_j(X,Y) := (X^kY - R(X))^j .
\end{aligned}$$

We again view these polynomials as univariates in Y with coefficients in $\mathbb{F}_q[X]$. Thus, P'_j is a polynomial with Y degree equal to j and for $j < \ell$, its leading coefficient equals $X^{kj} \cdot \prod_{i=1}^{n}(X-a_i)^{\ell-j}$, whereas for $j \geq \ell$, its leading coefficient equals X^{kj}. Thus, if we look at the $m \times m$ matrix consisting of the coefficient vectors of P'_j (we think of each of them as a polynomial in Y of degree at most $m-1$), we get a triangular matrix, with the diagonal entries corresponding to precisely the leading coefficients of $P'_0, P'_1, \ldots, P'_{m-1}$. Hence, the lattice generated by them is full rank and its determinant is the product of the diagonal entries of this matrix, which equals

$$\left(\prod_{j=0}^{\ell-1} X^{kj} \cdot \prod_{i=1}^{n}(X-a_i)^{\ell-j}\right) \cdot \left(\prod_{j=\ell}^{m-1} X^{kj}\right).$$

Thus, the degree of the determinant of $\mathcal{L}^{(\ell,m)}$ equals $\left(\frac{n(\ell+1)\ell}{2} + \frac{m(m-1)k}{2}\right)$. We now invoke Theorem 6.3 to $(\mathcal{L}')^{(\ell,m)}$ to get that there is a non-zero polynomial $\tilde{Q}(X,Y)$ in $(\mathcal{L}')^{(\ell,m)}$ such that $\mu(\tilde{Q})$, which equals its X-degree, is at most

$$\frac{1}{m} \cdot \left(\frac{n(\ell+1)\ell}{2} + \frac{m(m-1)k}{2}\right) = \left(\frac{n(\ell+1)\ell}{2m} + \frac{(m-1)k}{2}\right).$$

Moreover, since the X-degree of any P'_j is at most $(mk + n\ell)$, there is an algorithm that when given $P'_0, \ldots, P'_{m-1}$ as inputs, outputs $\tilde{Q}$ in $\tilde{O}(mk+n\ell) \cdot \mathrm{poly}(m) \leq \tilde{O}(n) \cdot \mathrm{poly}(m)$ time. Since $\tilde{Q}$ is in $(\mathcal{L}')^{(\ell,m)}$, and $P'_0, P'_1, \ldots, P'_{m-1}$ are linearly independent over the field $\mathbb{F}_q(X)$, we have that there are unique univariate polynomials $g_0, g_1, \ldots, g_{m-1} \in \mathbb{F}_q[X]$ such that $\tilde{Q} = \sum_{j=0}^{m-1} g_j(X)P'_j(X,Y)$. Let $Q(X,Y)$ be the polynomial defined as

$$Q(X,Y) := \sum_{j=0}^{m-1} g_j(X)P_j(X,Y).$$

From the definition of $Q, P_0, P_1, \ldots, P_{m-1}, P'_0, P'_1, \ldots, P'_{m-1}$, it follows immediately that the $(1,k)$-weighted degree of Q is at most the X-degree of $\tilde{Q}$, which is at most $\left(\frac{n(\ell+1)\ell}{2m} + \frac{(m-1)k}{2}\right)$. Moreover, there is an algorithm that when given the coefficient vector of $\tilde{Q}$, outputs the coefficient vector of Q in nearly linear time in the input size.

To see the moreover part, we just observe that in each P'_j, we have the property that the coefficient of Y^i is divisible by X^{ik}. Since $\tilde{Q}$ is obtained by taking an $\mathbb{F}_q[X]$-linear combination of such polynomials, this property continues to be true for $\tilde{Q}$. Finally, to obtain Q from $\tilde{Q}$, we just need to divide the coefficient of Y^i in $\tilde{Q}$ by X^{ik}, for every i, and this operation can be done in nearly linear time in the description of $\tilde{Q}$, and hence has time complexity at most $\tilde{O}(n) \cdot \mathrm{poly}(m)$.

Thus, the overall time complexity of obtaining Q, given $P_0, P_1, \ldots, P_{m-1}$ includes the time complexity of obtaining $P'_0, P'_1, \ldots, P'_{m-1}$ from $P_0, P_1, \ldots, P_{m-1}$, invoking Theorem 6.3 to obtain $\tilde{Q}$, and then doing the aforementioned transformation on $\tilde{Q}$ to obtain Q. We have already bounded the time complexity of each of these steps, apart from the complexity of constructing $P'_0, \ldots, P'_{m-1}$, by $\tilde{O}(n) \cdot \mathrm{poly}(m)$. Finally, since $P'_j(X, Y)$ equals $P_j(X, X^k Y)$, we can obtain the coefficient vector of P'_j by multiplying the coefficient of Y^i in $P_j(X, Y)$ by X^{ki} for every $i < m$. This takes an additional $\tilde{O}(n) \cdot \mathrm{poly}(m)$ time, and gives us the desired bound of $\tilde{O}(n) \cdot \mathrm{poly}(m)$ on the total complexity. $\square$

To complete the interpolation step of the algorithm of Figure 3, we now invoke the above Lemma 6.4 with the right setting of parameters, and account for the time complexity of obtaining P_j.

Lemma 6.5. *Let q be a prime power, let $a_1, \ldots, a_n$ be distinct elements of $\mathbb{F}_q$, and let $k < n$ and $\ell \in \mathbb{N}$ be parameters. Then there is an algorithm that runs in time $n \cdot \mathrm{poly}(\log(n), \log(q), \ell, \frac{n}{k})$, and on any input $w \in \mathbb{F}_q^n$, outputs a non-zero bivariate polynomial $Q(X, Y)$ with $(1, k)$-weighted degree at most $\sqrt{nk(\ell+1)\ell}$, such that for every $i \in [n]$, Q vanishes with multiplicity at least ℓ on (a_i, w_i).*

Proof. The algorithm proceeds by first setting the parameter m such that m satisfies $n(\ell+1)\ell = m(m-1)k$. For simplicity, we assume that m can be chosen to be a positive integer while satisfying this equation. So, $m = O(\ell\sqrt{\frac{n}{k}})$.

Next, we construct, via fast univariate polynomial interpolation, the polynomial $R(X)$ of degree at most $n-1$ such that for every $i \in [n]$, $R(a_i) = w_i$. This can be done in $n \cdot \mathrm{poly}(\log(q), \log(n))$ time. We now construct the basis $P_0, P_1, \ldots, P_{m-1}$ of the lattice $\mathcal{L}^{(\ell,m)}$ as follows. We focus on $j < \ell$ since the argument for larger j is similar (and only easier). For $j < \ell$, P_j is defined as $(Y - R(X))^j \cdot \prod_{i=1}^n (X - a_i)^{\ell-j}$. By expanding $(Y - R(X))^j$ and rearranging the terms, we get

$$P_j = \sum_{t=0}^{j} Y^t \left(\binom{j}{t} R^{j-t} \prod_{i=1}^{n} (X - a_i)^{\ell-j} \right) .$$

Thus, for every t, the coefficient of Y^t in P_j is a polynomial in X of degree at most $O(nm)$. Furthermore, given the coefficient vector of R (that we have already computed above) and $a_1, a_2, \ldots, a_n$, we can compute the polynomial $R^{j-t} \prod_{i=1}^n (X - a_i)^{\ell-j}$ using $\tilde{O}(nm)$ arithmetic operations over the underlying field using the standard FFT based algorithms for fast polynomial multiplcation. Thus, each P_j (and hence, the entire basis) can be computed in time $n \cdot \mathrm{poly}(m, \log(n), \log(q))$, which is at most $n \cdot \mathrm{poly}(\log(n), \log(q), \ell, \frac{n}{k})$ for our choice of $m = O(\ell\sqrt{\frac{n}{k}})$.

Now, from Lemma 6.4, we have that there is an algorithm that takes the basis of the lattice $\mathcal{L}^{(\ell,m)}$ as input and outputs the coefficient vector of a non-zero $Q(X, Y)$ in this lattice in time $n \cdot \mathrm{poly}(m) \le n \cdot \mathrm{poly}(\ell, \frac{n}{k})$, such that the $(1, k)$-weighted degree of Q is at most $\left(\frac{n(\ell+1)\ell}{2m} + \frac{(m-1)k}{2}\right)$. For our choice of m, the two terms in this weighted degree bound are equal to each other, and hence the $(1, k)$-degree of Q is at most $\sqrt{\frac{nk(\ell+1)\ell(m-1)}{m}} \le \sqrt{nk(\ell+1)\ell}$.

Finally, from Lemma 6.1, we get that this polynomial Q vanishes with multiplicity at least ℓ on (a_i, w_i) for every $i \in [n]$. This observation together with the fact that the overall running time of the algorithm is at most $n \cdot \mathrm{poly}(\log(n), \log(q), \ell, \frac{n}{k})$ completes the proof of the lemma. $\square$

6.1.2 Fast implementation of the root finding step

We first note that from the properties of the polynomial Q constructed in Lemma 6.5, the following observation follows.

Lemma 6.6. *Let $Q(X,Y)$ be the polynomial constructed in Lemma 6.5, and let $f(X) \in \mathbb{F}_q[X]$ be a univariate polynomial of degree less than k so that $f(a_i) = w_i$ for at least $\frac{\sqrt{nk(\ell+1)\ell}}{\ell}$ many indices $i \in [n]$. Then, $Q(X, f(X)) = 0$.*

Proof. The lemma follows immediately from Claim 3.9, analogously to the proof of Lemma 3.10. We just sketch the details, while keeping track of the changed parameters. From the same argument as in the proof of Lemma 3.10, we have that $Q(X, f(X))$, which is a univariate polynomial of degree at most $\sqrt{nk(\ell+1)\ell}$, vanishes with multiplicity at least ℓ on every a_i where $f(a_i) = w_i$. Thus, if the number of agreements of f and w is at least $\frac{\sqrt{nk(\ell+1)\ell}}{\ell}$, it must be the case that $Q(X, f(X))$ is identically zero. $\square$

One final technical tool needed for the fast variant of a list decoding algorithm for Reed-Solomon Codes is the following theorem of Roth & Ruckenstein [RR00]. The theorem gives a faster algorithm for finding roots of bivariate polynomials than what is given by Theorem 3.7 in certain settings of parameters. We refer to Theorem 1.2 in [Ale05] for a proof of the theorem.

Theorem 6.7 ([RR00], [Ale05])**.** *Let $\mathbb{F}$ be any finite field. Then, there is a randomized algorithm that takes as input the coefficient vector of a bivariate polynomial $Q(X,Y) \in \mathbb{F}[X,Y]$ and outputs all its factors of the form $Y - f(X)$ where $f \in \mathbb{F}[X]$ in time $\tilde{O}(\deg_X(Q)) \cdot \mathrm{poly}(\deg_Y(Q), \log(|\mathbb{F}|))$.*

6.1.3 Fast list decoding of Reed-Solomon Codes up to the Johnson Bound

By setting the parameters appropriately, we can now prove the following theorem which gives a fast list-decoding algorithm for Reed-Solomon Codes approaching the Johson bound.

Theorem 6.8. *Let q be a prime power, let $a_1, a_2, \ldots, a_n$ be distinct points in $\mathbb{F}_q$, let $k < n$ be a positive integer, and let $\epsilon > 0$ be a parameter. Then the Reed-Solomon Code $\mathrm{RS}_q(a_1, \ldots, a_n; k)$ can be list decoded from $(1 - \sqrt{\frac{k}{n}} - \epsilon)$-fraction of errors in time $n \cdot \mathrm{poly}(\frac{n}{k}, \log(q), \log(n), \frac{1}{\epsilon})$.*

Proof. Given the parameter $\epsilon > 0$ in the theorem, we set the parameter ℓ to be the smallest natural number greater than $\frac{1}{\epsilon}$. Note that for this choice of ℓ, we have $\frac{\sqrt{nk(\ell+1)\ell}}{\ell} < (1+\epsilon)\sqrt{nk}$.

We now invoke Lemma 6.5 with the given received word as input. This gives us a non-zero bivariate polynomial $Q(X,Y)$ that, as Lemma 6.6 shows, has the property that any polynomial $f(X) \in \mathbb{F}_q[X]$ of degree less than k with agreement at least $(1+\epsilon)\sqrt{nk}$ with the received word satisfies $Q(X, f(X)) = 0$. We now invoke Theorem 6.7 to compute all such roots $f(X)$ of $Q(X,Y)$ and output the ones that have degree less than k and have sufficiently large agreement with the received word.

The time complexity of the algorithm depends on the time complexity of the interpolation step (Lemma 6.5), which is

$$n \cdot \mathrm{poly}\left(\log(n), \log(q), \ell, \frac{n}{k}\right) \le n \cdot \mathrm{poly}\left(\log(n), \log(q), \frac{n}{k}, \frac{1}{\epsilon}\right)$$

and the time complexity of the root finding step (Theorem 6.7), which is

$$\tilde{O}(\deg_X(Q)) \cdot \mathrm{poly}(\deg_Y(Q), \log(q)) \leq \sqrt{nk} \cdot \mathrm{poly}\left(\frac{n}{k}, \log(nk), \log(q), \frac{1}{\epsilon}\right),$$

where we used the fact that $\deg_X(Q)$ is at most the $(1,k)$-weighted degree of Q and hence is at most $O(\sqrt{nk}/\epsilon)$ and $\deg_Y(Q)$ is at most $\frac{1}{k}$ of the $(1,k)$-weighted degree of Q and hence is at most $O(\sqrt{\frac{n}{k}} \cdot \frac{1}{\epsilon})$. Finally, recalling that $k < n$ gives us the upper bound on the time complexity of $n \cdot \mathrm{poly}(\frac{n}{k}, \log(q), \log(n), \frac{1}{\epsilon})$ of the algorithm, and completes the proof of the theorem.

□

6.2 Fast list decoding of multiplcity codes up to capacity

In this section, we build upon the ideas discussed in the preceding section to get an algorithm that list decodes multiplicity codes up to capacity in nearly linear time (in an appropriate parameter regime). In what follows, fix a prime power q, distinct evaluation points $a_1, \ldots, a_n \in \mathbb{F}_q$, and positive integers k, s so that $k < sn$ and $\max\{k, s\} \leq \mathsf{char}(\mathbb{F}_q)$, and let $\mathsf{MULT}_q^{(s)}(a_1, \ldots, a_n; k)$ denote the corresponding multiplicity code. We shall show that for any constant $\delta := 1 - \frac{k}{sn}$ and constant $\epsilon > 0$, and sufficiently large s (depending on ϵ), the multiplicity code $\mathsf{MULT}_q^{(s)}(a_1, \ldots, a_n; k)$ can be list decoded from $(1 - R - \epsilon)$-fraction of errors in time $\tilde{O}(n) \cdot \mathsf{polylog}(q)$, where $R := \frac{k}{sn}$ is the rate of the code.

Overview. The fast algorithm for list decoding multiplicity codes up to capacity is essentially a fast implementation of the polynomial-time algorithm for list decoding multiplicity codes up to capacity that was presented in Section 4.2. We will proceed by first describing a faster algorithm for the interpolation step that lets us construct an ordinary differential equation of high order satisfied by all close enough codewords. This is based on the machinary of lattices over the univariate polynomial ring that we have already seen. The second part of the proof is a fast algorithm for solving these differential equations whose solutions are low dimensional subspaces, and recovering a basis for the subspace of solutions. Eventually, we will observe that a constant size (depending on ϵ) list of close enough codewords can be recovered in nearly linear time using the results of Section 5.2.1.

6.2.1 Fast implementation of the interpolation step

Let $w \in (\mathbb{F}_q^s)^n$ be the received word, where $w_i = (w_{i,0}, \ldots, w_{i,s-1})$ for any $i \in [n]$, and let $r \leq s$ be a parameter. Our goal in this step is to find, in nearly-linear time, a non-zero low-degree $(r+1)$-variate polynomial $Q(X, Y_0, \ldots, Y_{r-1})$ that is linear in $Y_0, \ldots, Y_{r-1}$ which satisfies that $P_f(X) := Q(X, f(X), f^{(1)}(X), \ldots, f^{(r-1)}(X))$ vanishes on a_i with high multiplicity for any point $i \in [n]$ for which $f^{(<s)}(a_i) = w_i$. The following definition will be helpful in stating some of the technical statements succinctly.

Definition 6.9 (Derivative operator)**.** *The* ***derivative operator*** τ *is an* $\mathbb{F}_q$*-linear map that maps polynomials in* $\mathbb{F}_q[X, Y_0, \ldots, Y_{r-1}]$ *that are linear in* $Y_0, \ldots, Y_{r-1}$ *to polynomials in* $\mathbb{F}_q[X, Y_0, \ldots, Y_r]$ *that are linear in* $Y_0, \ldots, Y_r$ *as follows:*

$$\tau\left(A(X) + \sum_{\ell=0}^{r-1} B_\ell(X) Y_\ell\right) := A^{(1)}(X) + \sum_{\ell=0}^{r-1} \left(B_\ell^{(1)}(X) \cdot Y_\ell + (\ell+1) B_\ell(X) \cdot Y_{\ell+1}\right).$$

More generally, for a non-negative integer j, let $\tau^{(j)}$ denote the following linear operator:

$$\tau^{(j)}\left(A(X)+\sum_{\ell=0}^{r-1}B_\ell(X)Y_\ell\right) := A^{(j)}(X)+\sum_{\ell=0}^{r-1}\sum_{h=0}^{j}\binom{h+\ell}{\ell}B_\ell^{(j-h)}(X)\cdot Y_{\ell+h}.$$

Thus, the image of $\tau^{(j)}$ is contained in $\mathbb{F}_q[X,Y_0,\dots,Y_{r-1+j}]$.

Note that by (4), for any polynomials $Q(X)=A(X)+\sum_{\ell=0}^{r-1}B_\ell(X)Y_\ell$ and $f(X)\in\mathbb{F}_q[X]$, if we let $P_f(X):=Q(X,f(X),f^{(1)}(X),\dots,f^{(r-1)}(X))$, then we have that

$$\tau^{(j)}(Q)(X,f(X),f^{(1)}(X),\dots,f^{(r-1+j)}(X)) = P_f^{(j)}(X),$$

for any non-negative integer j.

The following lemma is a faster version of Lemma 4.5.

Lemma 6.10. *There is an algorithm that takes as input a received word $w\in(\mathbb{F}_q^s)^n$, runs in time $\tilde{O}(n)\cdot\mathrm{poly}(s)$, and outputs a non-zero $(r+1)$-variate polynomial $Q(X,Y_0,\dots,Y_{r-1})$ over $\mathbb{F}_q$ of the form*

$$Q(X,Y_0,\dots,Y_{r-1}) = A(X)+B_0(X)\cdot Y_0+\dots+B_{r-1}(X)\cdot Y_{r-1}$$

such that:

1. *The X-degree of Q is at most $\frac{n(s-r+1)}{r+1}$.*
2. *For any $i\in[n]$ and $j\in\{0,1,\dots,s-r\}$, we have that*

$$\tau^{(j)}(Q)(a_i,w_{i,0},w_{i,1},\dots,w_{i,s-1}) = 0\,.$$

Proof. We will start by defining an appropriate lattice over the ring $\mathbb{F}_q[X]$ such that all polynomials in the lattice have degree at most one in $Y_0,Y_1,\dots,Y_{r-1}$ and satisfy the constraints in the second item in the lemma above. We then show that there is a polynomial in this lattice with X-degree at most $\frac{n(s-r+1)}{r+1}$, and hence the shortest vector with respect to X-degree being the measure satisfies this property. Based on these observations, the algorithm is quite natural - it will construct a basis for this lattice and then run the algorithm given in Theorem 6.3 to find a shortest vector in this lattice.

In more detail, let $R_0,R_1,\dots,R_{r-1}\in\mathbb{F}_q[X]$ be polynomials with degree at most $sn-1,(s-1)n-1,\dots,(s-r+1)n-1$ respectively such that for every $\ell\in\{0,1\dots,r-1\}$, $j\in\{0,1,\dots,s-\ell-1\}$ and $i\in[n]$, we have that

$$R_\ell^{(j)}(a_i) = \binom{\ell+j}{j}w_{i,\ell+j}.$$

Clearly, such polynomials exist due to degree considerations, and in fact are unique and can be found in time $\tilde{O}(ns)$ using FFT based algorithms (algorithm 10.22 in [vzGG13]). Given these R_ℓ's, let us consider the lattice $\mathcal{L}$ defined as follows.

$$\mathcal{L} := \left\{\tilde{g}(X)\cdot\prod_{i=1}^{n}(X-a_i)^{s-r+1}+\sum_{\ell=0}^{r-1}g_\ell(X)\cdot(Y_\ell-R_\ell(X)) : \tilde{g},g_0,\dots,g_{r-1}\in\mathbb{F}_q[X]\right\}$$

We first show that for every polynomial $Q\in\mathcal{L}$, $i\in\{1,2,\dots,n\}$, and $j\in\{0,1,\dots,s-r\}$, it holds that $\tau^{(j)}(Q)(a_i,w_{i,0},w_{i,1},\dots,w_{i,s-1})=0$. To prove the claim, it suffices to show that the property sought is true for every polynomial of the form $\tilde{g}(X) \cdot \prod_{t=1}^{n}(X - a_t)^{s-r+1}$ and of the form $g_\ell(X) \cdot (Y_\ell - R_\ell(X))$ for $\ell \in \{0, 1, \ldots, r-1\}$. Since every polynomial in the lattice is an $\mathbb{F}_q$-linear combination of polynomials of these form, the claim will follow via the $\mathbb{F}_q$-linearity of τ. Moreover, the property is immediate for polynomials of the form $\tilde{g}(X) \cdot \prod_{t=1}^{n}(X - a_t)^{s-r+1}$ since for polynomials with no Y variables, the operator τ is just the derivative operator with respect to X. So we just focus on polynomials of the form $g_\ell(X) \cdot (Y_\ell - R_\ell(X))$. The property is clear for $j = 0$ by the definition of the R_ℓ's, next we discuss the argument for $j = 1$, since the extension for larger j is very similar.

By definition of τ, $\tau^{(1)}(g_\ell(Y_\ell - R_\ell))$ equals

$$\tau^{(1)}(g_\ell(Y_\ell - R_\ell)) = g_\ell^{(1)}(X)(Y_\ell - R_\ell(X)) + g_\ell(X)((\ell + 1)Y_{\ell+1} - R_\ell^{(1)}(X)) .$$

Setting X to a_i, Y_ℓ to $w_{i,\ell}$ and $Y_{\ell+1}$ to $w_{i,\ell+1}$, we notice that both $(w_{i,\ell} - R_\ell(a_i))$ and $((\ell+1)w_{i,\ell+1} - R_\ell^{(1)}(a_i))$ are zero from the definition of R_ℓ. Thus, $\tau^{(1)}(g_\ell(Y_\ell - R_\ell))$ is zero on the input $(a_i, w_{i,0}, w_{i,1}, \ldots, w_{i,s-1})$. This argument naturally extends to $\tau^{(j)}$ for larger values of j by carefully following the definition of the operator $\tau^{(j)}$ and the interpolation constraints on R_ℓ.

Next we show that there exists $Q \in \mathcal{L}$ of X-degree at most $\frac{n(s-r+1)}{r+1}$. The generators of the lattice $\mathcal{L}$, when viewed as an $(r+1)$-dimensional vector space with entries from $\mathbb{F}_q[X]$ (where entries $0, 1, 2 \ldots, r-1$ correspond to the coefficient of $Y_0, Y_1, \ldots, Y_{r-1}$ respectively, and the r-st entry is the Y-free term) form a triangular system. Thus, we have a full rank lattice at our hand. Thus, the determinant of the $(r+1) \times (r+1)$ dimensional matrix whose columns correspond to these generators has a non-zero determinant. Moreover, this determinant is a polynomial in $\mathbb{F}_q[X]$ of degree at most the degree of $\prod_{i=1}^{n}(X - a_i)^{s-r+1}$, and hence is at most $n(s - r + 1)$. Therefore, from Theorem 6.3, we get that there is a polynomial in $\mathcal{L}$ that is non-zero and has X-degree at most $\frac{n(s-r+1)}{r+1}$. Moreover, there is an algorithm that takes the generators of this lattice as input and outputs this non-zero vector in time $\tilde{O}(n) \cdot \mathrm{poly}(s)$. $\square$

6.2.2 Fast implementation of the root finding step

Next observe that from the proof of Lemma 4.6, we have that for any polynomial Q constructed in Lemma 6.10 and any univariate f of degree less than k such that f agrees with w on at least

$$t := \frac{(s - r + 1)n + (r + 1)(k - 1)}{(s - r + 1)(r + 1)} = \frac{n}{r + 1} + \frac{k - 1}{s - r + 1}$$

evaluation points, we have that

$$Q(X, f(X), f^{(1)}(X), \ldots, f^{(r-1)}(X)) = 0 .$$

Moreover, similarly to the discussion in the beginning of Section 4.2, the parameters s, r can be set based on ϵ such that this error tolerance gets us ϵ-close to list decoding capacity as desired.

Therefore, to complete the list decoding task, it suffices to solve the above equation. To be able to do this in nearly linear time, we first recall that by Lemma 4.3, the solution space of polynomials of degree less than k satisfying $Q(X, f(X), f^{(1)}(X), \ldots, f^{(r-1)}(X)) = 0$ is an affine space over the field $\mathbb{F}_q$ of dimension at most r. Hence, this subspace can be described by specifying a basis (whose total description size is at most $O(kr \cdot \log(q))$).

Moreover, by Lemma 5.12 the number of codewords contained in this affine space that are close enough to w is a constant (depending only on ϵ, the relative distance δ of the code, and the dimension r). To complete the algorithm, we shall describe a randomized nearly-linear time algorithm that outputs these codewords.

This is essentially immediate from the proof of Lemma 5.12: Recall that in this lemma we described a randomized algorithm Prune, which when given $w \in \Sigma^n$, outputs any codeword in the list with constant probability (depending on ϵ, δ, and r), and used this to conclude that the list size is constant. Thus, we can simply run Prune for a constant number of times and return the union of the output lists. By a union bound, all close-by codewords will appear in the union of the output lists with high (constant) probability. Finally, note that the algorithm Prune proceeds by sampling constantly many entries in $[n]$ at random, and solving a linear system of equations on these entries, and so this step can be implemented in near linear time in n by using the standard linear system solver.

Thus, it suffices to show how to find the subspace of solutions to the differential equation

$$Q(X, f(X), \ldots, f^{(r-1)}(X)) = 0$$

in nearly-linear time.

Such an algorithm is given in the following lemma.

Lemma 6.11. *Suppose that $Q(X, Y_0, \ldots, Y_{r-1})$ is a non-zero $(r+1)$-variate polynomial over $\mathbb{F}_q$ of the form*

$$Q(X, Y_0, \ldots, Y_{r-1}) = A(X) + B_0(X) \cdot Y_0 + \cdots + B_{r-1}(X) \cdot Y_{r-1}$$

and of degree at most D. Let $\mathcal{L}$ be the list containing all polynomials $f(X) \in \mathbb{F}_q[X]$ of degree smaller than k so that

$$Q(X, f(X), f^{(1)}(X), \ldots, f^{(r-1)}(X)) = 0.$$

Then if $\max\{k, s\} \leq \mathsf{char}(\mathbb{F}_q)$, then $\mathcal{L}$ forms an affine subspace over $\mathbb{F}_q$ of dimension at most $r-1$, and there is an algorithm that takes as input the coefficient vector of Q and k and outputs a basis of this affine space of solutions in time $\tilde{O}(D+k) \cdot \mathrm{poly}(r, \log(q))$.

Note that for our choice of $D = \frac{n(s-r+1)}{r+1}$ and $r < s$, the algorithm runs in time $\tilde{O}(D+k) \cdot \mathrm{poly}(r, \log(q)) = n \cdot \mathrm{poly}(s, \log(q), \log(n))$. In the rest of this section, we sketch some of the main ideas in the proof of the above lemma, and we refer the interested reader to [GHKS24, Theorem 5.1] for a full proof. In particular, for notational simplicity, we only focus on the cases of small r, namely $r = 0$ and $r = 1$, which already contain most of the technical insights.

Solving the $r = 0$ case. For $r = 0$, we are just looking for f of degree less than k such that $A(X) + f(X) \cdot B(X) = 0$. Thus, in this case, the solution $f(X)$ to this equation, if it exists is unique and is equal to the quotient when $A(X)$ is divided by $B(X)$. Thus, a natural way of solving this problem will be to compute the quotient and remainder when $A(X)$ is divided by $B(X)$ and output the quotient if the remainder is zero and the quotient has degree less than k. This task can be done in nearly linear time in the input size using known classical algorithms in computational algebra that are based on the Fast Fourier Transform. We describe the sketch of one such algorithm below based on divide and conquer. As we shall see later, some of the ideas behind this algorithm will naturally lead us towards the case of larger r.

For the discussion below, let us assume that k is a power of 2. First observe that any polynomial f of degree smaller than k can be uniquely written as $f = f_0(X) + X^{k/2} f_1(X)$ where f_0, f_1 are (possibly zero) polynomials of degree less than $\frac{k}{2}$. The idea is to try and recover f_0 and f_1 by solving equations of smaller size, and then put them together to compute f. The first point is to note that if f satisfies $A(X) + f(X) \cdot B(X) = 0$, then it satisfies

$$A + f \cdot B \equiv 0 \mod X^k \,. \tag{8}$$

Moreover, since f has degree less than k, it suffices to recover f modulo X^k. So, to solve the original equation, we will instead solve the above modular equation and then check if the solution satisfies the original equation. To solve the modular equation above, we try to understand the properties that f_0, f_1 must satisfy.

By substituting $f = f_0 + X^{k/2} f_1$ in $(A + fB \equiv 0 \mod X^k)$, we get that

$$A + f_0 \cdot B + X^{k/2} f_1 \cdot B \equiv 0 \mod X^k.$$

Now, every monomial in the term $X^{k/2} f_1 \cdot B$ has degree at least $\frac{k}{2}$. Thus, the above equation holds if and only if

$$A + f_0 B \equiv 0 \mod X^{k/2} \tag{9}$$

and

$$(A + f_0 B) + X^{k/2} f_1 B \equiv 0 \mod X^k$$

Moreover, for any f_0 satisfying $(A + f_0 B \equiv 0 \mod X^{k/2})$, we have that $A + f_0 B$ equals $\tilde{A} X^{k/2}$ for some polynomial $\tilde{A}(X)$. Thus, f_1 must satisfy

$$X^{k/2} \tilde{A} + X^{k/2} f_1 B \equiv 0 \mod X^k,$$

which is equivalent to

$$\tilde{A} + f_1 B \equiv 0 \mod X^{k/2}. \tag{10}$$

Thus, to solve for f in Equation (8) it suffices to solve Equation (9) and then construct $\tilde{A}$ and solve Equation (10). Moreover, in Equation(9) and Equation (10), the polynomials $A, B, \tilde{A}$ can be replaced by their residues modulo $X^{k/2}$. Thus, we have reduced the original problem of solving Equation (8) to solving two problems of exactly half the size. If at any stage of the recursion, we end up with an equation with no solution, we get that the original equation did not have a solution. Moreover, if any of these equations has a solution, it is unique (modulo X^k).

Finally, to argue about the running time, we note that to solve instances where we work modulo X^k, we make two recursive calls sequentially, each involving working modulo $X^{k/2}$. Since we are only interested in solutions modulo X^k (or lower powers of X) in these modular equations, we can assume without loss of generality (by just ignoring all monomials of degree greater than k, since they do not participate in any computation modulo X^k) that A and B are also of degree at most k. Constructing the problem instance for the first recursive call (9) takes $O(k)$ time, and given a solution f_0 to this instance, we need to construct the polynomial $\tilde{A}$ to make the second recursive call (10). Given f_0, we can obtain $\tilde{A}$ by computing the polynomial $(A + f_0 B) \mod X^k$ in time $\tilde{O}(k)$ (using the Fast Fourier Transform for polynomial multiplication). Given solutions f_0 and f_1 of the two subproblems, the solution $f := f_0 + X^{k/2} f_1$ can be again constructed in time $O(k)$. Thus, the overall time complexity $T(k)$ can be upper bounded by the recurrence

$$T(k) \leq 2T(\frac{k}{2}) + k \log^c(k)\,,$$

for some fixed constant c, where c just depends on the complexity of Fast Fourier Transform over the underlying field $\mathbb{F}$.

This gives the desired near linear upper bound on the running time.

Solving the $r = 1$ case. We now consider the case of $r = 1$. This case essentially captures all of the main ideas of the proof of the above Lemma 6.11 for large r, but keeps the exposition simpler in terms of notation.

Recall that for $r = 1$, we are looking for solutions to the following differential equation:

$$A + fB_0 + f^{(1)}B_1 = 0.$$

Without loss of generality, we assume that $B_1(X)$ is a non-zero polynomial (else we are in the $r = 0$ case), and moreover, $B_1(0)$ is non-zero, else, we shift X to $X + a$ for an a which is not a root of B_1. Such a shift can be found in time $\tilde{O}(D)$ deterministically (using fast algorithms for univariate multipoint evaluation, e.g. Chapter 10 in [vzGG13]) or by simply taking a random a from the underlying field.

Once again, it will be helpful to work with the modular equation instead.

$$A + fB_0 + f^{(1)}B_1 \equiv 0 \mod X^k. \tag{11}$$

A natural first attempt for this problem is again to try and recover f by recovering f_0, f_1 recursively as in the $r = 0$ case. However, there is an issue here - unlike in the $r = 0$ case, equations of the form of (11) can have multiple solutions. Thus, even if we set up modular equations for f_0 and f_1 and solve them, the number of candidate solutions f of the original problem that are obtained by a combination of these can be as large as the product size of the solution sets for f_0 and f_1. This explosion of potential solutions appears to be an issue that needs to be handled for this strategy to work.

To get around this issue, we rely on the fact that the space of solutions of (11) is an affine space over $\mathbb{F}_q$ of dimension 1. Furthermore, the proof of Lemma 4.3 (cf., Claim 4.4) tells us more about the structure of this affine space: if $B_1(0) \neq 0$ (which we have assumed here without loss of generality), then any solution f of (11) is uniquely determined by its constant term, i.e. $f(0)$.[6] This simple observation, combined with elementary linear algebra, leads to the following claim that turns out to be very useful.

Claim 6.12. *Let $g(X), h(X)$ be the unique polynomials of degree less than k with constant term zero and one respectively that are solutions of (11) (assuming that $B_1(0) \neq 0$). Then, g, h form a basis of the affine space of solutions of degree less than k (over the field $\mathbb{F}$) of (11). In other words, the set of such solutions equals the set $\{\lambda g + (1-\lambda)h : \lambda \in \mathbb{F}\}$.*

Proof. Let g, h be any solutions to (11). Thus, we have that $A + gB_0 + g^{(1)}B_1 \equiv 0 \mod X^k$ and $A + hB_0 + h^{(1)}B_1 \equiv 0 \mod X^k$. Now, for any $\lambda \in \mathbb{F}$, we claim that $\lambda g + (1-\lambda)h$ is also a solution of (11). To see this, we observe using the linearity of the derivative operator that

$$A + (\lambda g + (1-\lambda)h)B_0 + (\lambda g + (1-\lambda)h)^{(1)}B_1 = A + \lambda\left(gB_0 + g^{(1)}B_1\right) + (1-\lambda)\left(hB_0 + h^{(1)}B_1\right).$$

Since g, h are solutions to (11), we observe that $\left(gB_0 + g^{(1)}B_1 \equiv -A \mod X^k\right)$ and $\left(hB_0 + h^{(1)}B_1 \equiv -A \mod X^k\right)$. Thus,

$$A + \lambda\left(gB_0 + g^{(1)}B_1\right) + (1-\lambda)\left(hB_0 + h^{(1)}B_1\right) \equiv A + \lambda(-A) + (1-\lambda)(-A) \mod X^k,$$

which is 0 modulo X^k. So, we get that every element of the set $\{\lambda g + (1-\lambda)h : \lambda \in \mathbb{F}\}$ is a solution of (11), if g and h are solutions.

Furthermore, we note that

$$\{\lambda g + (1-\lambda)h : \lambda \in \mathbb{F}\} = \{\lambda(g-h) + h : \lambda \in \mathbb{F}\}$$

[6]Note that here, we are still looking at solutions modulo X^k.

and thus this set is indeed an affine space of dimension one whenever $(g - h)$ is non-zero (which is the case here, since the constant terms of g and h are distinct). Since we had already argued that the solutions to (11) form an affine space of dimension at most one, we have that $\{\lambda g + (1-\lambda)h : \lambda \in \mathbb{F}\}$ is indeed the set of all such solutions. $\square$

Given the above claim, a natural strategy towards solving (11) is to try and recover the unique polynomials f_0, f_1 of degree less than k, with constant term zero and one respectively that are solutions of the equation. It turns out that the uniqueness of solutions to (11), once the constant terms are fixed, makes this subproblem amenable to a divide and conquer like strategy.

Let us consider a solution f of degree less than k of (11) with constant term α. Thus, $f(X)$ can be written as $f(X) = \alpha + Xg(X)$ for a polynomial g of degree less than $k-1$. Since f is a solution, we have the following sequence of identities.

$$\begin{aligned} A + fB_0 + f^{(1)}B_1 &\equiv 0 \mod X^k \\ \implies A + (\alpha + Xg)B_0 + (\alpha + Xg)^{(1)}B_1 &\equiv 0 \mod X^k \\ \implies (A + \alpha B_0) + (XB_0 + B_1)g + XB_1 g^{(1)} &\equiv 0 \mod X^k \end{aligned}$$

Thus, for every α such that $f = \alpha + Xg$ is a solution of (11), g must satisfy that

$$(A + \alpha B_0) + (XB_0 + B_1)g + XB_1 g^{(1)} \equiv 0 \mod X^k\,. \tag{12}$$

The above new differential equation, while being similar in structure to (11), has the useful property that there is a unique polynomial $g(X)$ of degree less than $(k-1)$ which satisfies this equation. To see this, note that by the proof of Lemma 4.3 (cf., Claim 4.4), it suffices to show that the constant term $g(0)$ of g is uniquely determined. To see this, we note that evaluating the left hand side of (12) at zero gives the following equation:

$$(A(0) + \alpha B_0(0)) + B_1(0)g(0) = 0\,.$$

Since $B_1(0)$ is a non-zero field element (by our assumption at the beginning of this section), the above degree one equation in $g(0)$ has a unique solution given by $g(0) = -\frac{A(0)+\alpha B_0(0)}{B_1(0)}$.

The uniqueness of solutions now enables us to use a natural divide and conquer style algorithm, almost identical to the one we discussed for the $r = 0$ case. We write g as $g = g_0 + X^{k/2}g_1$ for polynomials g_0, g_1 of degree less than $\frac{k}{2}$ and try to understand the conditions that g_0, g_1 must satisfy. If g is a solution of Equation (12), then, working modulo $X^{k/2}$ and discarding terms that are divisible by $X^{k/2}$, we get that g_0 must satisfy

$$(A + \alpha B_0) + (XB_0 + B_1)g_0 + XB_1 g_0^{(1)} \equiv 0 \mod X^{k/2}\,.$$

Moreover, for any g_0 satisfying the above equation, we have that $(A + \alpha B_0) + (XB_0 + B_1)g_0 + XB_1 g_0^{(1)}$ is divisible by $X^{k/2}$, and hence equals $X^{k/2}\tilde{A}$ for some polynomial $\tilde{A}$, and g_1 must satisfy

$$X^{k/2}\tilde{A} + X^{k/2}\left(XB_0 + \left(1 + \frac{k}{2}\right)B_1\right)g_1 + X^{k/2} \cdot XB_1 g_1^{(1)} \equiv 0 \mod X^k\,.$$

This is equivalent to

$$\tilde{A} + \left(XB_0 + \left(1 + \frac{k}{2}\right)B_1\right)g_1 + XB_1 g_1^{(1)} \equiv 0 \mod X^{k/2}\,.$$

Thus, we have reduced the question of finding g to the question of finding g_0, g_1 that satisfy similar differential equations, but now, we are working modulo $X^{k/2}$. Thus, the instance size has effectively halved. Moreover, these new instances can be generated in nearly linear time in the input size. Thus, a divide and conquer based algorithm will find g in nearly linear time.

As mentioned above, the setting of larger r can be handled using very similar ideas, but with some care. We skip the details here, and refer the interested reader to [GHKS24].

6.3 Bibliographic notes

The question of designing nearly linear time encoding and decoding algorithms has been an important theme of study in algebraic coding theory. The Berlekamp-Massey algorithm (Chapter 7 in [vzGG13]) discovered in the 1960s decodes Reed-Solomon Codes (and BCH Codes) in the unique decoding regime in nearly linear time. The algorithm is based on an algorithm of Berlekamp [Ber68a] for decoding BCH Codes. Massey [Mas69] gave a simplified description of this algorithm and showed that it can also be used to solve other problems in computational algebra, notably the problem of computing minimal polynomials of linear recurrent sequences. These algorithms essentially gave a reduction from the problem of decoding Reed-Solomon Codes (or BCH Codes) in the unique decoding regime to the well studied problem of *rational function reconstruction*. Subsequently, rational function reconstruction was shown to be solvable in nearly linear time using a near linear time algorithm for computing the GCD of univariate polynomials, thereby giving a near linear time algorithm for unique decoding of Reed-Solomon Codes. The half GCD algorithm is also referred to as the Knuth-Schönhage algorithm, and we refer the interested reader to Chapter 11 of [vzGG13] for a description of the algorithm and detailed references.

In the list decoding setting, Roth & Ruckenstein [RR00] gave a faster (though not near linear time) implementation of the algorithms of Sudan and Guruswami-Sudan for list decoding Reed-Solomon Codes. On the way to their algorithm, they designed a faster algorithm for root computation for bivariates stated in Theorem 6.7. In a subsequent work, Alekhnovich [Ale05] introduced lattice based techniques and relied on some of the ideas in [RR00] to give nearly linear time algorithms for list decoding Reed-Solomon Codes of constant rate in parameter regimes approaching the Johnson Bound. For multiplicity codes (as well as folded Reed-Solomon Codes), near linear time algorithms for list decoding up to capacity (again in the constant rate regime) were described in a work of Goyal, Harsha, Kumar and Shankar [GHKS24]. The presentation in this chapter follows the presentation in [GHKS24].

7 Local list decoding

In this section we show how to list decode polynomial codes in *sublinear time*. Sublinear-time algorithms (also known as *local algorithms*) are highly-efficient algorithms which run in time that is much smaller than the input length. In particular, such algorithms cannot even read the whole input, and many times they also cannot afford to write down the whole output. For this to be at all possible, we shall need to relax the computational model, and provide the algorithm with a *query access to the input*, and also in the case the output is too long, only require that the algorithm provides *query access to the output*. Sublinear-time algorithms are also typically *randomized*, and they have a small chance of error.

In the context of (unique) decoding algorithms, this means that the local algorithm runs in time which is much smaller than the block length n of the code. To enable this, we provide the algorithm with a query access to the received word $w \in \Sigma^n$, and additionally only require that for any given codeword entry $i \in [n]$, the algorithm correctly recovers the i-th entry of the closest codeword c with high probability. Formally, we have the following definition.

Definition 7.1 (Locally correctable code (LCC))**.** *Let $C \subseteq \Sigma^n$ be a code of relative distance δ, and let $\alpha \in (0, \frac{\delta}{2})$. We say that C is (Q, α)-locally correctable (LCC) if there exists a randomized algorithm A which satisfies the following requirements:*

- ***Input:*** *A takes as input a codeword entry $i \in [n]$ and also gets query access to a string $w \in \Sigma^n$ so that $\delta(w, c) \le \alpha$ for some codeword $c \in C$.*
- ***Query complexity:*** *A makes at most Q queries to the oracle w.*
- ***Output:*** *A outputs c_i with probability at least $\frac{3}{4}$.*

Remark 7.2. *A few remarks are in order:*

1. *We could have chosen the success probability to be any constant greater than $\frac{1}{2}$, since the success probability can be amplified by repeating the local correction procedure independently for a constant number of times and outputting the majority value (at the cost of increasing the query complexity by a constant multiplicative factor). For simplicity, we fixed the success probability to $\frac{3}{4}$ in the above definition.*
2. *Often one is also interested in the* running time *of the local correction algorithms. As is typically the case, in all the algorithms presented in this section, the running time will be polynomial in the query complexity Q.*
3. *A useful (and arguably more natural) variant of local correction is* local decoding. *Locally decodable codes are defined similarly to locally correctable codes, except that we view the codeword c as the encoding of some message m (by fixing some injective encoding map $\Phi : \Sigma^k \to C$, where k is such that $|C| = |\Sigma|^k$), and the local decoder is required to decode individual entries in m. If we restrict ourselves to* linear codes*, then locally decodable codes are a weaker object than locally correctable codes, since any linear locally correctable code can be converted into a locally decodable code by choosing a systematic encoding map.*[7] *For simplicity, we focus in this section on the model of local correction, but we note that all of the algorithms we present can be easily adapted for the local decoding model as well.*

We will be interested in an extension of the above definition of local correction to the setting of list decoding. However, the actual definition requires some subtlety: we want to return a list of answers corresponding to all near-by codewords in the list, and we want the algorithm to be local, but returning a list of possible symbols in Σ for individual entries of near-by codewords is pretty useless, since typically there would be many entries $i \in [n]$ so that the values of the i-th entry of all near-by codewords cover all of Σ. Furthermore, for applications one typically needs some form of *consistency*, i.e., that for any fixed codeword c in the list it is possible to locally correct individual entries of c with high probability in a consistent way. To ensure this, we require that the local list-decoding algorithm returns a list $A_1, A_2, \ldots, A_L$ of randomized local algorithms, so that for each near-by codeword c in the list there would be some algorithm A_j in the list that locally corrects c. Formally, we have the following definition.

[7] An encoding map $\Phi : \Sigma^k \to C$ is *systematic* if any $m \in \Sigma^k$ is a prefix of $\Phi(m)$. For linear codes, such a map can be found by transforming the generating matrix G into a reduced column echelon form, and letting $\Phi(m) = G \cdot m$.

Definition 7.3 (Locally list decodable code (LLDC))**.** *We say that a code $C \subseteq \Sigma^n$ is (Q, α, L)-locally list decodable (*LLDC*) if there exists a randomized algorithm A which outputs L randomized algorithms $A_1, A_2, \ldots, A_L$ that satisfy the following requirements for any $w \in \Sigma^n$:*

- ***Input:*** *Each A_j takes as input a codeword entry $i \in [n]$ and also gets query access to w.*
- ***Query complexity:*** *Each A_j makes at most Q queries to w.*
- ***Output:*** *For every codeword $c \in C$ so that $\delta(c, w) \leq \alpha$, with probability at least $\frac{1}{2}$ over the randomness of A the following event happens: there exists some $j \in [L]$ such that for all $i \in [n]$,*

$$\Pr[A_j(i) = c_i] \geq \frac{3}{4},$$

where the probability is over the internal randomness of A_j.

Remark 7.4. *The same remarks as in Remark 7.2 also apply to the above definition. We also note the following:*

1. *The success probability of $\frac{1}{2}$ for the local list decoding algorithm A in the above definition can be amplified by invoking the local list decoding algorithm A independently for multiple times, and outputting the union of all randomized algorithms A_j that are output in any of the invocations, at the cost of increasing the list size (Specifically, amplifying the success probability to $1 - 2^{-t}$ requires increasing the list size by a multiplicative factor of $O(t)$).*
2. *In the above definition we required that for any* fixed *close-by codeword c, the list of local algorithm A_j contains a local corrector for c with high probability. We can also guarantee the stronger requirement that with high-probability, the output list contains a local corrector for* any *close-by codeword c, by first amplifying the success probability of A to $1 - \frac{1}{4L}$, and then applying a union bound over all near-by codewords (noting that by the definition of local list decoding, there could be at most $2L$ such codewords).*

Though the above definition of local list decoding may seem a bit strange and convoluted at first glance, it turns out to be just the right definition for various applications in coding theory and theoretical computer science.

For a code to be locally (list) decodable, its codeword entries need to satisfy some non-trivial local relations. In the setting of polynomial-based codes, this property can be achieved by considering codes based on *multivariate* polynomials, where the main property being used is that the restriction of a low-degree multivariate polynomial to any line (or more generally, a curve or a low-dimension subspace) is a low-degree *univariate* polynomial. Thus, given a codeword entry, one can recover the value of the entry by passing a random line through the point, and recovering the closest low-degree univariate polynomial on the line (or more generally, a curve or a low-dimension subspace).

In what follows, we first focus on the family of *Reed-Muller* (RM) *Codes*, which extends Reed-Solomon Codes to the setting of *multivariate* polynomials. We first show in Section 7.1 below, as a warmup, and since this will also be used as a sub-procedure in the local list-decoding algorithm, how to *locally correct* Reed-Muller Codes from a small fraction of errors (below half the minimum distance of the code). Then in Section 7.2 we show how to *locally list decode* Reed-Muller Codes beyond half the minimum distance, and in Section 7.3 we present an improved algorithm that is able to locally list decode Reed-Muller Codes *up to Johnoson bound*. Finally, in Section 7.4 we consider the family of *multivariate multiplicity codes* which extends Reed-Muller Codes by evaluating also *multivariate derivatives* of multivariate polynomials, and outline a local list decoding algorithm for these codes *up to their minimum distance*, and we further explain how this can be used to obtain capacity-achieving locally list decodable codes.

7.1 Local correction of Reed-Muller Codes

In this section, we first show, as a warmup, and since this will also be used as a sub-procedure in the local list-decoding algorithm presented in the next section, how to *locally correct* Reed-Muller Codes from a small fraction of errors (below half the minimum distance of the code). More specifically, fix a prime power q, and positive integers k, m so that $k < q$. Recall that the *Reed-Muller* (RM) *Code* $\mathrm{RM}_{q,m}(k)$ is the code which associates with each m-variate polynomial $f(X_1, \ldots, X_m) \in \mathbb{F}_q[X_1, \ldots, X_m]$ of (total) degree smaller than k a codeword $(f(\mathbf{a}))_{\mathbf{a} \in \mathbb{F}_q^m} \in \mathbb{F}_q^{q^m}$. Let $\delta := 1 - \frac{k}{q}$ denote the relative distance of this code, below we shall show that this code is a (q, α)-LCC for $\alpha := \frac{\delta}{8} - \frac{1}{q}$. Note that for $m > 1$, the query complexity of q is much smaller than the block length q^m of the code.

Suppose that $w \in \mathbb{F}_q^{q^m}$ is a received word, in what follows it will be convenient to view w as a function $w : \mathbb{F}_q^m \to \mathbb{F}_q$. Suppose that there exists a (unique) polynomial $f(X_1, \ldots, X_m) \in \mathbb{F}_q[X_1, \ldots, X_m]$ of degree smaller than k so that $f(\mathbf{a}) \neq w(\mathbf{a})$ for at most an α-fraction of the entries $\mathbf{a} \in \mathbb{F}_q^m$. Below we shall describe a randomized local correction algorithm which given as input an entry $\mathbf{a} \in \mathbb{F}_q^m$, makes a few queries to w, and outputs $f(\mathbf{a})$ with probability at least $\frac{3}{4}$.

Overview. To decode $f(\mathbf{a})$, we pick a random line through $\mathbf{a}$, and query the restriction of w to the line. We then find the *univariate* polynomial $g(X) \in \mathbb{F}_q[X]$ of degree smaller than k that is closest to w on the line (this can be done efficiently using the algorithm of Figure 1), and return the value of g on $\mathbf{a}$. The main properties we use to show correctness are that the restriction of f to any line is a univariate polynomial of degree smaller than k, and also that random lines *sample-well* in the sense that the fraction of errors on a random line is typically close to the total fraction of errors. The formal description of the algorithm is presented in Figure 7 below, followed by correctness analysis.

Local correction of $\mathrm{RM}_{q,m}(k)$**:**

- **INPUT:** Query access to a function $w : \mathbb{F}_q^m \to \mathbb{F}_q$ that is $(\frac{\delta}{8} - \frac{1}{q})$-close to some polynomial $f(X_1, \ldots, X_m) \in (\mathbb{F}_q)_{<k}[X_1, \ldots, X_m]$ (i.e., $f(\mathbf{a}) \neq w(\mathbf{a})$ for at most a $(\frac{\delta}{8} - \frac{1}{q})$-fraction of the points $\mathbf{a} \in \mathbb{F}_q^m$), a point $\mathbf{a} \in \mathbb{F}_q^m$.
- **OUTPUT:** $f(\mathbf{a})$.

1. Pick a random point $\mathbf{b} \in \mathbb{F}_q^m$.
2. Query all points on the line $\lambda_{\mathbf{a},\mathbf{b}}(T) := (1 - T) \cdot \mathbf{a} + T \cdot \mathbf{b}$ passing through $\mathbf{a}$ and $\mathbf{b}$.
3. Find a univariate polynomial $g(T) \in \mathbb{F}_q[T]$ of degree smaller than k that is $\frac{\delta}{2}$-close to $w|_{\lambda_{\mathbf{a},\mathbf{b}}}(T) := (w \circ \lambda_{\mathbf{a},\mathbf{b}})(T)$ using the algorithm given in Figure 1; If such a polynomial does not exist, then output $\perp$.
4. Output $g(0)$.

Figure 7: Local correction of Reed-Muller Codes

Correctness. Note that for a random point $\mathbf{b} \in \mathbb{F}_q^m$, any point on the line $\lambda_{\mathbf{a},\mathbf{b}}(T)$ other than the zero point is uniformly random. Consequently, the expected fraction of errors on the line is at most $(\frac{\delta}{8} - \frac{1}{q}) + \frac{1}{q} = \frac{\delta}{8}$. By Markov's inequality, this implies in turn that with probability at least $\frac{3}{4}$ over the choice of $\mathbf{b}$, the fraction of errors on the line $\lambda_{\mathbf{a},\mathbf{b}}$ is less than $\frac{\delta}{2}$. Lastly, assuming that the latter event happens, by the correctness of the algorithm of Figure 1 (noting that $f|_{\lambda_{\mathbf{a},\mathbf{b}}}$ is a univariate polynomial of degree smaller than k), we have that $g(T) = f|_{\lambda_{\mathbf{a},\mathbf{b}}}(T)$, and in particular, $g(0) = f(\mathbf{a})$. So we conclude that the algorithm outputs $f(\mathbf{a})$ with probability at least $\frac{3}{4}$.

Query complexity. The query complexity is clearly the number q of points on a line.

Remark 7.5. *While we have not made an attempt to optimize the decoding radius of the above local correction algorithm, we note that to get a non-trivial success probability greater than $\frac{1}{2}$, one must set the decoding radius to be smaller than $\frac{\delta}{4}$. This is because the use of* Markov's inequality*, which guarantees that with probability at least $\frac{1}{2}$ over the choice of the random line through $\mathbf{a}$, the number of errors is smaller than $\frac{\delta}{2}$ only below this radius.*

To locally correct Reed-Muller Codes from a larger fraction of errors, one can replace in the algorithm of Figure 7 the random line $\lambda_{\mathbf{a},\mathbf{b}}(T)$ passing through $\mathbf{a}$ with a random curve *passing through $\mathbf{a}$ of the form $\chi_{\mathbf{a},\mathbf{b},\mathbf{c}}(T) = \mathbf{a} + \mathbf{b}T + \mathbf{c}T^2$, where $\mathbf{b}, \mathbf{c} \in \mathbb{F}_q^m$ are uniformly random. The advantage of decoding on curves is that any two points on a random curve through $\mathbf{a}$ are* pairwise independent*, and consequently one can use* Chebyshev inequality *to get a better bound on the number of errors on the curve. However, resorting to a curve also increases the degree of the restriction of f to the curve, and so one can tolerate less errors on the curve. But it turns out that this tradeoff is favorable, and allows one to locally correct up to a radius of roughly $\delta - \frac{1}{2}$, which is close to half the minimum relative distance for δ which is close to 1 (see [Yek12, Section 2.2.3] for a full description and analysis of this algorithm).*

7.2 Local list decoding of Reed-Muller Codes beyond unique decoding radius

Next we show how to *locally list-decode* Reed-Muller Codes beyond the unique decoding radius. Recall that in local list decoding, we require that the local list-decoding algorithm returns a list $A_1, A_2, \ldots, A_L$ of randomized local algorithms, so that for each near-by codeword c in the list there would be some algorithm A_j in the list that locally corrects c. In fact, since we have already shown that Reed-Muller Codes are *locally correctable*, it suffices to show that they are *approximately* locally list decodable, in the sense that the local algorithms A_j are only required to correctly recover *most* of the entries of c. Such an approximate local list decoding algorithm can then be turned into a local list decoding algorithm by composing each local algorithm A_j with the local correction algorithm presented in the previous section. Furthermore, using an averaging argument, it can be shown that in approximate local list decoding each of the local algorithms A_j can be assumed to be *deterministic*. Formally, we have the following definition.

Definition 7.6 (Approximately locally list-decodable code (ALLDC))**.** *We say that a code $C \subseteq \Sigma^n$ is (Q, α, ϵ, L)-**approximately locally list decodable** (ALLDC) if there exists a randomized algorithm A which outputs L* deterministic *algorithms $A_1, A_2, \ldots, A_L$ that satisfy the following requirements for any $w \in \Sigma^n$:*

- ***Input:*** *Each A_j takes as input a codeword entry $i \in [n]$ and also gets query access to w.*
- ***Query complexity:*** *Each A_j makes at most Q queries to w.*

- ***Output:*** *For every codeword $c \in C$ so that $\delta(c, w) \leq \alpha$, with probability at least $\frac{1}{2}$ over the randomness of A the following event happens: there exists some $j \in [L]$ such that*

$$\Pr_{i \in [n]}[A_j(i) = c_i] \geq 1 - \epsilon,$$

where the probability is over the choice of a random entry $i \in [n]$.

Remark 7.7. *The same remarks as in Remark 7.4 also apply to the above definition. We note however that in the case of ALLDC, the success probability of each local algorithm A_j cannot be amplified by repetition, and therefore we have not fixed its value to $\frac{3}{4}$ as in the previous* LLDC *definition (Definition 7.3).*

The following lemma shows that one can turn an approximate local list decoding algorithm into a local list decoding algorithm by composing each local algorithm A_j with a local correction algorithm.

Lemma 7.8 (ALLDC + LCC $\Rightarrow$ LLDC). *Suppose that $C \subseteq \Sigma^n$ is a code that is (Q, α, ϵ, L)-*ALLDC *and (Q', α')-LCC for $\alpha' \geq \epsilon$. Then C is also a $(Q \cdot Q', \alpha, L)$-*LLDC.

Proof. The local list decoding algorithm replaces each of the local algorithms A_j generated by the approximate local list decoding algorithm A with a local algorithm A'_j, operating as follows: on input $i \in [n]$, the local algorithm A'_j invokes the local correction algorithm A' on input i, and forwards each of the queries of A' in $[n]$ to the local algorithm A_j.

Correctness follows since by our assumption that $\alpha' \geq \epsilon$, A_j outputs the correct value on at least a $(1 - \alpha')$-fraction of the entries in $[n]$, and A' outputs the correct value with probability at least $\frac{3}{4}$ provided that at most an α'-fraction of the codeword entries are corrupted. The query complexity is $Q \cdot Q'$, because for each of the Q' queries that A' would make, A'_j runs a copy of A_j which makes Q queries. $\square$

Given the above lemma, our task now is to design an *approximate* local list decoding algorithm for Reed-Muller Codes beyond half their minimum distance. As before, fix a prime power q and positive integers k, m so that $k < q$, and let $\mathrm{RM}_{q,m}(k)$ be the corresponding Reed-Muller Code. Let $\delta := 1 - \frac{k}{q}$ denote the relative distance of this code, let $\sigma > 1$ and $\xi > 0$ be parameters, and let $\alpha := 1 - \sqrt{\sigma \cdot (1 - \delta)} - \xi$. We shall show that this code is a (q, α, ϵ, L)-ALLDC for $\epsilon = \frac{2\delta}{\sigma - 1} + O(\frac{1}{\xi^2 q})$ and $L = q$. Thus, for sufficiently large σ and q, the failure probability ϵ is smaller than $\frac{\delta}{8}$, and so by the results of Section 7.1 and by Lemma 7.8, we also have that this code is a (q^2, α, L)-LLDC. Note that for any fixed $\sigma > 1$, the decoding radius of $1 - \sqrt{\sigma \cdot (1 - \delta)}$ is larger than half the minimum relative distance for a sufficiently large relative distance δ. Note that for any $m > 2$, the query complexity of q^2 is much smaller than the block length q^m of the code.

Below we shall describe a *randomized* approximate local list decoding algorithm A which outputs a list of *deterministic* q-query local algorithms $A_1, \ldots, A_L$ with the following guarantee. Suppose that $w \in \mathbb{F}_q^{q^m}$ is a received word (as before it will be convenient to view w as a function $w : \mathbb{F}_q^m \to \mathbb{F}_q$), and suppose that $f(X_1, \ldots, X_m) \in \mathbb{F}_q[X_1, \ldots, X_m]$ is a polynomial of degree smaller than k so that $f(\mathbf{a}) \neq w(\mathbf{a})$ for at most an α-fraction of the entries $\mathbf{a} \in \mathbb{F}_q^m$. We shall show that with probability at least $\frac{1}{2}$ over the randomness of A, there exists a local algorithm A_j which correctly recovers all but at most an ϵ-fraction of the values $f(\mathbf{a})$ for $\mathbf{a} \in \mathbb{F}_q^m$.

Overview. To generate the list of local algorithms, the algorithm A picks a uniformly random point $\mathbf{b} \in \mathbb{F}_q^m$ and for any value $v \in \mathbb{F}_q$, it outputs a local algorithm $A_{\mathbf{b},v}$ (note that the number of local algorithms is indeed $L = q$). We think of the value v as a "guess" or "advice" for the value of $f(\mathbf{b})$. Once we have this advice, we use it to locally correct f. Specifically, to recover $f(\mathbf{a})$, the local algorithm $A_{\mathbf{b},v}$ first considers the line $\lambda_{\mathbf{a},\mathbf{b}}$ passing through $\mathbf{a}$ and $\mathbf{b}$, and globally list decodes the restriction of w to this line to obtain a list $\mathcal{L} \subseteq \mathbb{F}_q[T]$ of univariate polynomials (this can be done efficiently using the algorithm of Figure 3). These univariate polynomials are candidates for $f|_{\lambda_{\mathbf{a},\mathbf{b}}}$. Which of these univariate polynomials is $f|_{\lambda_{\mathbf{a},\mathbf{b}}}$? We use our guess v for $f(\mathbf{b})$: if there is a unique univariate polynomial in the list with value v at $\mathbf{b}$, then we deem that to be our candidate for $f|_{\lambda_{\mathbf{a},\mathbf{b}}}$, and output its value at the point $\mathbf{a}$ as our guess for $f(\mathbf{a})$.

The above algorithm will correctly recover $f(\mathbf{a})$ if (1) there are not too many errors on the line through $\mathbf{a}$ and $\mathbf{b}$, and (2) no other polynomial in $\mathcal{L}$ takes the same value at $\mathbf{b}$ as f does. As we have already shown in the previous section, the first event is high probability by standard sampling bounds. As to the second event, using that any pair of polynomials in $\mathcal{L}$ differ by at least a δ-fraction of the points, we get that any other polynomial $h(T) \in \mathcal{L}$ will agree with $f|_{\lambda_{\mathbf{a},\mathbf{b}}}$ on $\mathbf{b}$ with probability at most $1-\delta$ over the choice of $\mathbf{b}$. Assuming that the list $\mathcal{L}$ is sufficiently small (which happens if the decoding radius is sufficiently below the Johnson Bound), one can then apply a union bound over all elements of $\mathcal{L}$ to show that with high probability, no other polynomial $h(T) \in \mathcal{L}$ agrees with $f|_{\lambda_{\mathbf{a},\mathbf{b}}}$ on $\mathbf{b}$.

The formal description of the approximate local list decoding algorithm is presented Figure 8 below, followed by correctness analysis.

Approximate local list decoding of $\mathrm{RM}_{q,m}(k)$:

The approximate local list decoding algorithm A:

1. Pick a random point $\mathbf{b} \in \mathbb{F}_q^m$.
2. For any value $v \in \mathbb{F}_q$, output the local algorithm $A_{\mathbf{b},v}$ defined below.

The local algorithms $A_{\mathbf{b},v}$:

- **INPUT:** Query access to a function $w : \mathbb{F}_q^m \to \mathbb{F}_q$, a point $\mathbf{a} \in \mathbb{F}_q^m$.
- **OUTPUT:** A value in $\mathbb{F}_q$.

1. Query all points on the line $\lambda_{\mathbf{a},\mathbf{b}}(T) := (1-T)\cdot \mathbf{a} + T \cdot \mathbf{b}$ passing through $\mathbf{a}$ and $\mathbf{b}$.
2. Find the list $\mathcal{L}$ of all univariate polynomials $g(T) \in \mathbb{F}_q[T]$ of degree smaller than k that are $(1-\sqrt{\sigma\cdot(1-\delta)})$-close to $w|_{\lambda_{\mathbf{a},\mathbf{b}}}(T)$ using the algorithm given in Figure 3.
3. If there exists a unique polynomial $g(T) \in \mathcal{L}$ so that $g(1) = v$, then output $g(0)$; Otherwise, output $\perp$.

Figure 8: Local list decoding of Reed-Muller Codes

Correctness. Let $f(X_1,\ldots,X_m) \in \mathbb{F}_q[X_1,\ldots,X_m]$ be a polynomial of degree smaller than k so that $f(\mathbf{a}) \neq w(\mathbf{a})$ for at most an α-fraction of the entries $\mathbf{a} \in \mathbb{F}_q^m$. We would like to show that with probability at least $\frac{1}{2}$ over the choice of $\mathbf{b}$, the local algorithm $A_{\mathbf{b},f(\mathbf{b})}$ correctly recovers $f(\mathbf{a})$ for all but at most an ϵ-fraction of $\mathbf{a} \in \mathbb{F}_q^m$. To show this, we shall first prove that with high probability over the choice of *both* random $\mathbf{a} \in \mathbb{F}_q^m$ and random $\mathbf{b} \in \mathbb{F}_q^m$, it holds that $A_{\mathbf{b},f(\mathbf{b})}$ correctly computes $f(\mathbf{a})$ on input $\mathbf{a}$.

We start by showing that with high probability over the choice of $\mathbf{a}$, $\mathbf{b}$, there exists a polynomial $g(T) \in \mathcal{L}$ that agrees with f on the line $\lambda_{\mathbf{a},\mathbf{b}}$.

Claim 7.9. *With probability at least* $1 - \frac{1}{\xi^2 q}$ *over the choice of random* $\mathbf{a}, \mathbf{b} \in \mathbb{F}_q^m$*, it holds that* $f|_{\lambda_{\mathbf{a},\mathbf{b}}} \in \mathcal{L}$.

Proof. Note that for random points $\mathbf{a}, \mathbf{b} \in \mathbb{F}_q^m$, the line $\lambda_{\mathbf{a},\mathbf{b}}$ is a uniformly random line. Consequently, any single point on the line is uniformly random, and any pair of points on the line are *pairwaise independent*. Let $Z_1, \ldots, Z_q$ be indicator random variables where $Z_t = 1$ if w differs from f on the t'th point of the line $\lambda_{\mathbf{a},\mathbf{b}}$, let $Z := \frac{\sum_{t=1}^q Z_t}{q}$, and note that by the correctness of the algorithm of Figure 3, we have that $f|_{\lambda_{\mathbf{a},\mathbf{b}}} \in \mathcal{L}$ if $Z \leq 1 - \sqrt{\sigma \cdot (1-\delta)}$. Moreover, by linearity of expectation and since any point on the line is uniformly random, we have that

$$\mathbb{E}[Z] = \frac{\sum_{t=1}^q \mathbb{E}[Z_t]}{q} \leq 1 - \sqrt{\sigma \cdot (1-\delta)} - \xi,$$

while by pairwise independence, we have that

$$\mathrm{Var}(Z) = \frac{\sum_{t=1}^q \mathrm{Var}[Z_t]}{q^2} \leq \frac{\sum_{t=1}^q \mathbb{E}[Z_t]}{q^2} \leq \frac{1}{q}.$$

Consequently, by Chebyshev's inequality,

$$\Pr\left[Z > 1 - \sqrt{\sigma \cdot (1-\delta)}\right] \leq \Pr\left[Z - \mathbb{E}[Z] > \xi\right] \leq \frac{\mathrm{Var}(Z)}{\xi^2} \leq \frac{1}{\xi^2 q}.$$

So we conclude that $Z \leq 1 - \sqrt{\sigma \cdot (1-\delta)}$ with probability at least $1 - \frac{1}{\xi^2 q}$, in which case we have that $f|_{\lambda_{\mathbf{a},\mathbf{b}}} \in \mathcal{L}$. □

Next we show that with high probability over the choice of $\mathbf{a}, \mathbf{b}$, there is no polynomial other than $f|_{\lambda_{\mathbf{a},\mathbf{b}}}$ in $\mathcal{L}$ that agrees with f on $\mathbf{b}$.

Claim 7.10. *With probability at least* $1 - \frac{\delta}{\sigma - 1}$ *over the choice of random* $\mathbf{a}, \mathbf{b} \in \mathbb{F}_q^m$*, there is no polynomial* $h(T) \in \mathcal{L}$ *other than* $f|_{\lambda_{\mathbf{a},\mathbf{b}}}$ *so that* $h(1) = f(b)$.

Proof. We can view the process of picking random $\mathbf{a}, \mathbf{b} \in \mathbb{F}_q^m$ as first picking a random $\mathbf{a} \in \mathbb{F}_q^m$, then picking a random line λ through $\mathbf{a}$, and finally letting $\mathbf{b} \in \mathbb{F}_q^m$ be a random point on the line λ. So in what follows, fix a point $\mathbf{a}$ and a line λ through it, and let $\mathcal{L}$ be the list of univariate polynomials of degree smaller than k that are $(1 - \sqrt{\sigma(1-\delta)})$-close to w on the line. We shall show that with probability at least $1 - \frac{\delta}{\sigma-1}$ over the choice of $\mathbf{b}$, it holds that there is no polynomial $h(T) \in \mathcal{L}$ other than $f|_\lambda$ so that h and f agree on $\mathbf{b}$.

Fix a polynomial $h(T) \in \mathcal{L}$ so that $h(T) \neq f|_\lambda(T)$. Then since both h and $f|_\lambda$ are polynomials of degree smaller than k, they must differ by at least a δ-fraction of the points on the line. Consequently, we have that h and $f|_\lambda$ agree on the point $\mathbf{b}$ with probability at most $1 - \delta$ over the choice of $\mathbf{b}$. Furthermore, note that by the Johnson Bound (cf., Theorem 2.1), we have that

$$|\mathcal{L}| \leq \frac{\delta}{\sigma \cdot (1-\delta) - (1-\delta)} \leq \frac{\delta}{(\sigma-1)(1-\delta)}. \tag{13}$$

Consequently, by a union bound over all elements in $\mathcal{L}$, the probability that there exists a polynomial $h(T) \in \mathcal{L}$ other than $f|_\lambda$ so that h and f agree on $\mathbf{b}$ is at most $\frac{\delta}{\sigma-1}$. □

By the above Claims 7.9 and 7.10, we conclude that with probability at least $1 - \frac{\delta}{\sigma-1} - \frac{1}{\xi^2 q}$ over the choice of *both* $\mathbf{a}$ and $\mathbf{b}$, it holds that $f|_{\lambda_{\mathbf{a},\mathbf{b}}} \in \mathcal{L}$ and there is no polynomial $h(T) \in \mathcal{L}$ other than $f|_{\lambda_{\mathbf{a},\mathbf{b}}}$ so that $h(1) = f(b)$, in which case the local algorithm $A_{\mathbf{b}, f(\mathbf{b})}$ will set $g = f|_{\lambda_{\mathbf{a},\mathbf{b}}}$ in Step 3, and will output $f|_{\lambda_{\mathbf{a},\mathbf{b}}}(0) = f(\mathbf{b})$. By Markov's inequality, this implies in turn that with probability at least $\frac{1}{2}$ over the choice of $\mathbf{b}$, the local algorithm $A_{\mathbf{b}, f(\mathbf{b})}$ outputs $f(\mathbf{a})$ for at least a $(1-\epsilon)$-fraction of the points $\mathbf{a}$ for $\epsilon = \frac{2\delta}{\sigma-1} + \frac{2}{\xi^2 q}$.

Query complexity and list size. The list size is clearly the number q of possible values in $\mathbb{F}_q$, and the query complexity is clearly the number q of points on a line.

7.3 Local list decoding of Reed-Muller Codes up to the Johnson Bound

The local list decoding algorithm for Reed-Muller Codes that we presented in the previous section only gives a decoding radius approaching $1 - \sqrt{\sigma \cdot (1-\delta)}$ for a sufficiently large constant $\sigma > 1$, which is smaller than the Johnson Bound of $1 - \sqrt{1-\delta}$. While we have not made an attempt to optimize the value of σ, we note that it cannot be smaller than 2. The reason is that in order to apply the union bound in Claim 7.10 we need the list size $L' := |\mathcal{L}|$ to be smaller than $\frac{1}{1-\delta}$ (since the failure probability for each individual polynomial in the list can be as large as $1-\delta$), and by (13), such a list size is only guaranteed for a decoding radius smaller than $1 - \sqrt{2(1-\delta)}$.

In this section we show an improved algorithm which can locally list decode Reed-Muller Codes all the way up to the Johnson Bound. In more detail, as before, fix a prime power q and positive integers k, m so that $k < q$, and let $\mathrm{RM}_{q,m}(k)$ be the corresponding Reed-Muller Code. Let $\delta := 1 - \frac{k}{q}$ denote the relative distance of this code, let $\gamma, \xi > 0$ be parameters, and let $\alpha := 1 - \sqrt{(1+\gamma)\cdot(1-\delta)} - \xi$. We shall show that for any $s \geq 1$, this code is a (Q, α, ϵ, L)-ALLDC for $Q = r^m \cdot q$, $\epsilon = \frac{2\delta}{\gamma s} + O\left(\frac{1}{\xi^2 q}\right)$, and $L = (\frac{r}{s})^s$, where $r := \frac{8\delta s}{\gamma(1-\delta)}$. Thus, for $s > \frac{16\delta}{\gamma}$ and sufficiently large q, the failure probability ϵ is smaller than $\frac{\delta}{8}$, and so by the results of Section 7.1 and by Lemma 7.8, we also have that this code is a $(r^m \cdot q^2, \alpha, L)$-LLDC. Note that for constant m, s, γ, and δ, the query complexity is $r^m \cdot q = O(q^2)$ which is much smaller than the block length q^m of the code for $m > 2$ and a growing q.

Overview. The main idea behind the improved algorithm is to also use *derivatives* to disambiguate the list. Specifically, as before the advice is generated by choosing a uniformly random point $\mathbf{b} \in \mathbb{F}_q^m$, but now the guess v is supposed to equal to all multivariate derivatives of order smaller than s of $f(X_1, \ldots, X_m)$ at $\mathbf{b}$ for some integer $s \geq 1$ (cf., Section 2.2 for the definition of multivariate derivatives). To take advantage of derivatives, we recall that by Fact 2.3, any pair of univariate degree k polynomials agree on all derivatives of order smaller than s on at most a fraction of $\frac{k}{sq} = \frac{1-\delta}{s}$ of the points. Consequently, the advice fails to disambiguate any pair of polynomials in $\mathcal{L}$ with probability at most $\frac{1-\delta}{s}$, and taking $s \gg L'$ now allows us to apply a union bound over $\mathcal{L}$.

However, a disadvantage of this approach is that it significantly increases the list size of the local list decoding algorithm, which is the number of different guesses for all m-variate derivatives of order smaller than s at the point $\mathbf{b}$. Specifically, the number of different guesses is $q^{\binom{m+s-1}{m}}$ (which is much larger than the block length q^m of the code!), since the number of m-variate derivatives of order smaller than s is $\binom{m+s-1}{m}$, and each such derivative can take any value in $\mathbb{F}_q$.

To reduce the list size, we slightly extend the model of local list decoding by providing the local list decoding algorithm A with a query access to the received word w (so the algorithm A can make a small number of queries to w before outputting the list of local algorithms $A_1, \ldots, A_L$). To take advantage of this model, the local list decoding algorithm A first chooses t random lines through $\mathbf{b}$, then (globally) list decodes the restriction of f to these lines, and finally only outputs guesses v for $f^{(<s)}(\mathbf{b})$ that are consistent with the directional derivatives of these lines at $\mathbf{b}$ for a significant fraction of the lines. For a careful choice of the joint distribution of the t random lines, bounding the number of guesses can be cast as an instance of a *list recovery* problem for Reed-Muller Codes (a certain generalization of list decoding), and in particular, we can use an extension of the Johnson Bound to the setting of list recovery to bound the number of such

guesses.

The formal description of the approximate local list decoding algorithm is given in Figure 9 below, followed by correctness analysis. Before we formally state the algorithm, we first fix some notation. This notation is motivated by the following *chain rule* for Hasse Derivatives, which relates the directional derivatives of a multivariate polynomial on a line to its global derivatives (the proof follows directly from the definition of the Hasse derivative and is left as an exercise).

Fact 7.11 (Chain rule for Hasse Derivatives)**.** *Let* $f(X_1, \ldots, X_m) \in \mathbb{F}[X_1, \ldots, X_m]$*, let* $\mathbf{a}, \mathbf{u} \in \mathbb{F}^m$*, and let* $\lambda(T) = \mathbf{a} + T\mathbf{u}$ *be the line passing through* $\mathbf{a}$ *in* direction $\mathbf{u}$*. Then for any non-negative integer* i *and* $T \in \mathbb{F}$*,*

$$(f|_\lambda)^{(i)}(T) = \sum_{\mathbf{i}:\mathsf{wt}(\mathbf{i})=i} f^{(\mathbf{i})}(\mathbf{a} + T\mathbf{u}) \cdot \mathbf{u}^{\mathbf{i}}. \tag{14}$$

For integers m, s, let $I_{m,s} := \{\mathbf{i} \in \mathbb{N}^m \mid \mathsf{wt}(\mathbf{i}) < s\}$, and for an element $v \in \mathbb{F}_q^{I_{m,s}}$, and a direction $\mathbf{u} \in \mathbb{F}_q^m$, we define the restriction $v|_{\mathbf{u}} \in \mathbb{F}_q^s$ of v to direction $\mathbf{u}$ to equal $v|_{\mathbf{u}} := \left(\sum_{\mathbf{i}:\mathsf{wt}(\mathbf{i})=i} v_{\mathbf{i}} \cdot \mathbf{u}^{\mathbf{i}}\right)_{i=0,1,\ldots,s-1}$. Note that in this notation, (14) can be rephrased as

$$(f|_\lambda)^{(<s)}(T) = \left(f^{(<s)}(\mathbf{a} + T\mathbf{u})\right)|_{\mathbf{u}}. \tag{15}$$

We now describe the algorithm.

Approximate local list decoding of $\mathrm{RM}_{q,m}(k)$:

$\triangleright$ Let $s \geq 1$ be a parameter, let $r := \frac{8\delta s}{\gamma(1-\delta)}$, and let U be an arbitrary subset of $\mathbb{F}_q$ of size r.

The approximate local list decoding algorithm A:

- **INPUT:** Query access to a function $w : \mathbb{F}_q^m \to \mathbb{F}_q$.
- **OUTPUT:** A list of local algorithms.

1. Pick random points $\mathbf{b} \in \mathbb{F}_q^m$ and $\mathbf{u}_0 \in \mathbb{F}_q^m$.
2. For each $\mathbf{u} \in U^m$:
   (a) Query all points on the line $\lambda_{\mathbf{u}}(T) := \mathbf{b} + T(\mathbf{u}_0 + \mathbf{u})$ through $\mathbf{b}$ in direction $\mathbf{u}_0 + \mathbf{u}$.
   (b) Find the list $\mathcal{L}_{\mathbf{u}}$ of all univariate polynomials $g(T) \in \mathbb{F}_q[T]$ of degree smaller than k that are $(1 - \sqrt{(1+\gamma)\cdot(1-\delta)})$-close to $w|_{\lambda_{\mathbf{u}}}$ using the algorithm given in Figure 3.
   (c) Let $S_{\mathbf{u}} = \left\{ g^{(<s)}(0) \mid g(T) \in \mathcal{L}_{\mathbf{u}} \right\}$.
3. Let V be the set containing all $v \in \mathbb{F}_q^{I_{m,s}}$ which satisfy that $v|_{\mathbf{u}_0+\mathbf{u}} \in S_{\mathbf{u}}$ for at least $\frac{1}{2}$ of the points $\mathbf{u} \in U^m$.
4. For any $v \in V$, output the local algorithm $A_{\mathbf{b},v}$ defined below.

The local algorithms $A_{\mathbf{b},v}$:

- **INPUT:** Query access to a function $w : \mathbb{F}_q^m \to \mathbb{F}_q$, a point $\mathbf{a} \in \mathbb{F}_q^m$.
- **OUTPUT:** A value in $\mathbb{F}_q$.

1. Query all points on the line $\lambda_{\mathbf{a},\mathbf{b}}(T) := (1-T)\cdot \mathbf{a} + T \cdot \mathbf{b} = \mathbf{a} + T(\mathbf{b}-\mathbf{a})$ passing through $\mathbf{a}$ and $\mathbf{b}$.
2. Find the list $\mathcal{L}$ of all univariate polynomials $g(T) \in \mathbb{F}_q[T]$ of degree smaller than k that are $(1 - \sqrt{(1+\gamma)\cdot(1-\delta)})$-close to $w|_{\lambda_{\mathbf{a},\mathbf{b}}}(T)$ using the algorithm given in Figure 3.
3. If there exists a unique polynomial $g(T) \in \mathcal{L}$ so that $g^{(<s)}(1) = v|_{\mathbf{b}-\mathbf{a}}$, then output $g(0)$; Otherwise, output $\perp$.

Figure 9: Local list decoding of Reed-Muller Codes up to Johnson Bound

Correctness. Let $f(X_1, \ldots, X_m) \in \mathbb{F}_q[X_1, \ldots, X_m]$ be a polynomial of degree smaller than k so that $f(\mathbf{a}) \neq w(\mathbf{a})$ for at most an α-fraction of the entries $\mathbf{a} \in \mathbb{F}_q^m$. We first show that with high probability over the choice of $\mathbf{b}, \mathbf{u}_0 \in \mathbb{F}_q^m$, it holds that $f^{(<s)}(\mathbf{b}) \in V$.

Claim 7.12. *With probability at least $1 - \frac{2}{\xi^2 q}$ over the choice of $\mathbf{b}, \mathbf{u}_0 \in \mathbb{F}_q^m$, it holds that $f^{(<s)}(\mathbf{b}) \in V$.*

Proof. First note that for any fixed $\mathbf{u} \in U^m$, since $\mathbf{b}$ and $\mathbf{u}_0$ are chosen uniformly at random, then the line $\lambda_{\mathbf{u}}(T) = \mathbf{b} + T(\mathbf{u}_0 + \mathbf{u})$ is uniformly random. Consequently, for any fixed $\mathbf{u} \in U^m$, the same analysis as in Claim 7.9 shows that with probability at least $1 - \frac{1}{\xi^2 q}$ over the choice of $\mathbf{b}$ and $\mathbf{u}_0$, it holds that $f|_{\lambda_{\mathbf{u}}} \in \mathcal{L}_{\mathbf{u}}$. By Markov's inequality, this implies in turn that with probability at least $1 - \frac{2}{\xi^2 q}$ over the choice of $\mathbf{b}$ and $\mathbf{u}_0$, it holds that $f|_{\lambda_{\mathbf{u}}} \in \mathcal{L}_{\mathbf{u}}$ for at least $\frac{1}{2}$ of the points $\mathbf{u} \in U^m$.

Next observe that for any $\mathbf{u} \in U^m$ so that $f|_{\lambda_{\mathbf{u}}} \in \mathcal{L}_{\mathbf{u}}$ we have that $(f|_{\lambda_{\mathbf{u}}})^{(<s)}(0) \in S_{\mathbf{u}}$. Moreover, by (15), it holds that $(f|_{\lambda_{\mathbf{u}}})^{(<s)}(0) = \left(f^{(<s)}(\mathbf{b})\right)|_{\mathbf{u}_0+\mathbf{u}}$ for each such $\mathbf{u}$. We conclude that with probability at least $1 - \frac{2}{\xi^2 q}$ over the choice of $\mathbf{b}$ and $\mathbf{u}_0$, we have that $\left(f^{(<s)}(\mathbf{b})\right)|_{\mathbf{u}_0+\mathbf{u}} \in S_{\mathbf{u}}$ for at least $\frac{1}{2}$ of $\mathbf{u} \in U^m$, in which case $f^{(<s)}(\mathbf{b})$ will be included in V. □

Next we would like to show that with probability at least $\frac{1}{2}$ over the choice of $\mathbf{b}$, the local algorithm $A_{\mathbf{b},f^{(<s)}(\mathbf{b})}$ correctly recovers $f(\mathbf{a})$ for most points $\mathbf{a} \in \mathbb{F}_q^m$. This part of the anlysis is very similar to the analysis of the algorithm of Figure 8, where the main difference is that we replace the standard bound on the maximum number of zeros of a low-degree polynomial with the stronger version that also accounts for multiplicities given by Fact 2.3, which allows us in turn to handle larger lists $\mathcal{L}$.

In more detail, as before, to show the above, we shall first show that with high probability over the choice of *both* random $\mathbf{a} \in \mathbb{F}_q^m$ and random $\mathbf{b} \in \mathbb{F}_q^m$, it holds that $A_{\mathbf{b},f^{(<s)}(\mathbf{b})}$ correctly computes $f(\mathbf{a})$ on input $\mathbf{a}$. The following claim says that with high probability over the choice of $\mathbf{a}, \mathbf{b}$, there exists a polynomial $g(T) \in \mathcal{L}$ that agrees with f on the line $\lambda_{\mathbf{a},\mathbf{b}}$. The proof is very similar to that of Claim 7.9.

Claim 7.13. *With probability at least* $1 - \frac{1}{\xi^2 q}$ *over the choice of random* $\mathbf{a}, \mathbf{b} \in \mathbb{F}_q^m$*, it holds that* $f|_{\lambda_{\mathbf{a},\mathbf{b}}} \in \mathcal{L}$.

The next claim says that with high probability over the choice of $\mathbf{a}, \mathbf{b}$, there is no polynomial $h(T) \in \mathcal{L}$ other than $f|_{\lambda_{\mathbf{a},\mathbf{b}}}$ so that $h^{(<s)}(1) = f^{(<s)}(\mathbf{b})|_{\mathbf{b}-\mathbf{a}}$.

Claim 7.14. *With probability at least* $1 - \frac{\delta}{\gamma s}$ *over the choice of random* $\mathbf{a}, \mathbf{b} \in \mathbb{F}_q^m$*, there is no polynomial* $h(T) \in \mathcal{L}$ *other than* $f|_{\lambda_{\mathbf{a},\mathbf{b}}}$ *so that* $h^{(<s)}(1) = f^{(<s)}(\mathbf{b})|_{\mathbf{b}-\mathbf{a}}$.

Proof. The proof is similar to that of Claim 7.10, except that we use Fact 2.3 to bound the probability that h and $f|_\lambda$ agree on $\mathbf{b}$ on all derivatives of order smaller than s. In more detail, as in the proof of Claim 7.10, we can view the process of picking random $\mathbf{a}, \mathbf{b} \in \mathbb{F}_q^m$ as first picking a random $\mathbf{a} \in \mathbb{F}_q^m$, then picking a random line λ through $\mathbf{a}$, and finally letting $\mathbf{b} \in \mathbb{F}_q^m$ be a random point on the line λ. So in what follows, fix a point $\mathbf{a}$ and a line λ through it, and let $\mathcal{L}$ be the list of univariate polynomials of degree smaller than k that are $(1 - \sqrt{(1+\gamma)(1-\delta)})$-close to w on the line. We shall show that with probability at least $1 - \frac{\delta}{\gamma s}$ over the choice of $\mathbf{b}$, it holds that there is no polynomial $h(T) \in \mathcal{L}$ other than $f|_\lambda$ so that h and $f|_\lambda$ agree on $\mathbf{b}$ on all derivatives of order smaller than s.

Fix a polynomial $h(T) \in \mathcal{L}$ so that $h(T) \neq f|_\lambda(T)$. Then, since both h and $f|_\lambda$ are polynomials of degree smaller than k, by Fact 2.3, they agree on all of their derivatives of order smaller than s on at most a $\frac{1-\delta}{s}$-fraction of the points on the line. Moreover, note that by the Johnson Bound (cf., Theorem 2.1), we have that

$$|\mathcal{L}| \leq \frac{\delta}{(1+\gamma)\cdot(1-\delta) - (1-\delta)} \leq \frac{\delta}{\gamma(1-\delta)}. \tag{16}$$

Consequently, by a union bound over all elements in $\mathcal{L}$, the probability that there exists a polynomial $h(T) \in \mathcal{L}$ other than $f|_\lambda$ that agrees with $f|_\lambda$ on $\mathbf{b}$ on all derivatives of order smaller than s is at most $\frac{\delta}{\gamma s}$. By (15), this implies in turn that with probability at least $1 - \frac{\delta}{\gamma s}$ over the choice of $\mathbf{a}, \mathbf{b} \in \mathbb{F}_q^m$, it holds that $h^{(<s)}(1) \neq f^{(<s)}(\mathbf{b})|_{\mathbf{b}-\mathbf{a}}$. □

By the above Claims 7.12, 7.13, and 7.14, we conclude that with probability at least $1 - \frac{\delta}{\gamma s} - \frac{3}{\xi^2 q}$ over the choice of $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{u}_0$, it holds that $f^{(<s)}(\mathbf{b}) \in V$, and that during the execution of $A_{\mathbf{b},f^{(<s)}(\mathbf{b})}$, it holds that $f|_{\lambda_{\mathbf{a},\mathbf{b}}} \in \mathcal{L}$ and there is no polynomial $h(T) \in \mathcal{L}$ other than $f|_{\lambda_{\mathbf{a},\mathbf{b}}}$ so that $h^{(<s)}(1) = f^{(<s)}(\mathbf{b})|_{\mathbf{b}-\mathbf{a}}$, in which case the local algorithm $A_{\mathbf{b},f^{(<s)}(\mathbf{b})}$ will set $g = f|_{\lambda_{\mathbf{a},\mathbf{b}}}$ in Step 3, and will output $f|_{\lambda_{\mathbf{a},\mathbf{b}}}(0) = f(\mathbf{a})$.

By Markov's inequality, this implies in turn that with probability at least $\frac{1}{2}$ over the choice of $\mathbf{b}$ and $\mathbf{u}_0$, the local algorithm $A_{\mathbf{b},f^{(<s)}(\mathbf{b})}$ will be output by A, and will output $f(\mathbf{a})$ for at least a $(1-\epsilon)$-fraction of the points $\mathbf{a}$ for $\epsilon = \frac{2\delta}{\gamma s} + \frac{6}{\xi^2 q}$.

Query complexity. The query complexity of the approximate local list decoding algorithm A is $r^m \cdot q$ since it queries r^m lines, where each line has q points, while the query complexity of each of the local algorithms $A_{\mathbf{b},v}$ is the number q of points on the line.

List size. It remains to bound the number of local algorithms output by A, which amounts to bounding the size of the set V. To this end, fix $\mathbf{b}, \mathbf{u}_0 \in \mathbb{F}_q^m$, and recall that for any $v \in V$, we have that $v|_{\mathbf{u}_0+\mathbf{u}} = \left(\sum_{\mathbf{i}:\mathsf{wt}(\mathbf{i})=i} v_{\mathbf{i}} \cdot (\mathbf{u}_0+\mathbf{u})^{\mathbf{i}}\right)_{i=0,1,\ldots,s-1} \in S_{\mathbf{u}}$ for at least $\frac{1}{2}$ of the points $\mathbf{u} \in U^m$, where $S_{\mathbf{u}} = \left\{g^{(<s)}(0) \mid g(T) \in \mathcal{L}_{\mathbf{u}}\right\} \subseteq \mathbb{F}_q^s$. For $\mathbf{u} \in U^m$ and $i \in \{0,1,\ldots,s-1\}$, let $S_{\mathbf{u},i} := \left\{g^{(i)}(0) \mid g(T) \in \mathcal{L}_{\mathbf{u}}\right\} \subseteq \mathbb{F}_q$, and let V_i be the set containing all vectors $v \in \mathbb{F}_q^{\{\mathbf{i}|\mathsf{wt}(\mathbf{i})=i\}}$ so that $\sum_{\mathbf{i}:\mathsf{wt}(\mathbf{i})=i} v_{\mathbf{i}} \cdot (\mathbf{u}_0+\mathbf{u})^{\mathbf{i}} \in S_{\mathbf{u},i}$ for at least $\frac{1}{2}$ of the points $\mathbf{u} \in U^m$. Note that by (16) and our choice of $r = \frac{8\delta s}{\gamma(1-\delta)}$, we have that

$$|S_{\mathbf{u},i}| \le |S_{\mathbf{u}}| \le |\mathcal{L}_{\mathbf{u}}| \le \frac{\delta}{(1-\delta)\gamma} = \frac{r}{8s}.$$

We shall show that for any $i \in \{0,1,\ldots,s-1\}$, V_i has size at most $\frac{r}{s}$, and so the size of V is at most $(\frac{r}{s})^s$.

To see the above, fix $i \in \{0,1,\ldots,s-1\}$, and let

$$\mathcal{L}_i := \left\{h \in (\mathbb{F}_q)_{<s}[X_1,\ldots,X_m] \mid h(\mathbf{u}_0+\mathbf{u}) \in S_{\mathbf{u},i} \text{ for at least half of the points } \mathbf{u} \in U^m\right\}.$$

Next we observe that $|V_i| \le |\mathcal{L}_i|$. To see this, map each $v \in V_i$ to a polynomial $h_v(\mathbf{X}) := \sum_{\mathbf{i}:\mathsf{wt}(\mathbf{i})=i} v_{\mathbf{i}} \mathbf{X}^{\mathbf{i}}$, and note that $h_v \in \mathcal{L}_i$ and $h_v \ne h_{v'}$ for any distinct $v, v' \in V_i$. Thus, it suffices to bound the size of $\mathcal{L}_i$.

The problem of bounding the size of $\mathcal{L}_i$ can be cast as follows. For a prime power q, positive integers k, m, and a subset $U \subseteq \mathbb{F}_q$ of size r, let $\mathrm{RM}_{q,m}(U;k)$ be the code over $\mathbb{F}_q$ of block length r^m which associates with any polynomial $f(X_1,\ldots,X_m) \in (\mathbb{F}_q)_{<k}[X_1,\ldots,X_m]$ a codeword $(f(\mathbf{u}))_{\mathbf{u}\in U^m} \in \mathbb{F}_q^{r^m}$. Note that the Reed-Muller Code $\mathrm{RM}_{q,m}(k)$ corresponds to the special case in which $U = \mathbb{F}_q$. It is well-known that $\mathrm{RM}_{q,m}(U;k)$ has relative distance at least $1 - \frac{k}{r}$ [Sch80, Zip79, DL78].

In the above terminology, we are given for each codeword entry $\mathbf{u} \in U^m$ in the code $\mathrm{RM}_{q,m}(U;s)$ (of relative distance $\delta' := 1 - \frac{s}{r}$) a small *input list* $S_{\mathbf{u},i}$ of possible values in $\mathbb{F}_q$ of size at most $\ell := \frac{r}{8s}$, and the goal is to bound the number codewords $c \in \mathrm{RM}_{q,m}(U;k)$ so that $c_{\mathbf{u}} \notin S_{\mathbf{u},i}$ for at most $\alpha' := \frac{1}{2}$ of the codeword entries $\mathbf{u}$. This is a well-known problem called *list recovery*, which extends the problem of list decoding that corresponds to the case where all input lists $S_{\mathbf{u}}$ have size one. In particular, an extension of the Johnson Bound to the setting of list-recovery (see e.g., [GKO+18, Lemma 5.2]) says that the number of codewords is at most

$$\frac{\delta'\ell}{(1-\alpha')^2 - \ell(1-\delta')} \le \frac{\ell}{\frac{1}{4} - \frac{\ell s}{r}} = \frac{r}{s}.$$

7.4 Local list decoding of multivariate multiplicity codes up to minimum distance

In the previous section we showed how to locally list decode Reed-Muller Codes up to the *Johnson Bound*. The main barrier that prevented us from bypassing the Johnson Bound was that the local list decoding algorithm for Reed-Muller Codes used as a subroutine the algorithm for list decoding of Reed-Solomon Codes from roughly the same decoding radius, and we currently do not know how to list decode Reed-Solomon Codes beyond the Johnson Bound with small list size.

However, for (univariate) multiplicity codes of sufficiently large multiplicity s, the results of Sections 4.2 and 5.2 imply that they are list decodable up to their *minimum distance* with constant list size, and thus it is reasonable to expect that we could also locally list decode *multivariate* multiplicity codes of sufficiently large multiplicity s up to their minimum distance. Indeed, an algorithm very similar to that of Figure 9 can be used to locally list decode multivariate multiplicity codes up to their minimum distance. Since the local list decoding algorithm for multivariate multiplicity codes is very similar to that of Reed-Muller Codes, but significantly more notationally heavy, we do not provide here a formal description of the algorithm for multivariate multiplicity codes, but instead just point out the main changes that need to be done in the algorithm of Figure 9 to adapt it to the setting of multivariate multiplicity codes. We refer the reader to [KRSW23, Section 4] for a formal description of the algorithm.

First, we note that in the multivariate multiplicity code setting, in order to list decode the restriction $w|_\lambda$ of w to some line λ, we need to also receive oracle access to the *directional derivatives* on the line, while w only provides oracle access to the *global derivatives*. However, we can use (15) to compute the directional derivative of w at any point on the line from the global derivatives of w at this point.

Second, in the setting of multivariate multiplicity codes, the local algorithm $A_{\mathbf{b},v}$ needs to output the *global derivatives* $f^{(<s)}(\mathbf{a})$ of f at $\mathbf{a}$, instead of just the value $f(\mathbf{a})$. Using the polynomial $g(T)$ determined in Step 3 of the algorithm $A_{\mathbf{b},v}$ in Figure 9, we can recover with high probability the *directional derivatives* of f at $\mathbf{a}$ in the direction of the line $\lambda_{\mathbf{a},\mathbf{b}}$, however these do not generally determine the global derivatives. To cope with this, we can first use exactly the same procedure that was used by the approximate local list decoding algorithm A to obtain a small number of possible guesses V' for the value of $f^{(<s)}(\mathbf{a})$, and then instead of outputting $g(0)$ in Step 3, output the unique $v' \in V'$ such that $v'|_{\mathbf{b}-\mathbf{a}} = g^{(<s)}(0)$ as a guess for $f^{(<s)}(\mathbf{a})$, if such a v' exists.

To show that the above solution works, we first recall that by Claim 7.12, we have that $v_0 := f^{(<s)}(\mathbf{a}) \in V'$ with high probability. So it remains to show that with high probability, for any $v' \in V'$ other than v_0 it holds that $v'|_{\mathbf{b}-\mathbf{a}} \neq v_0|_{\mathbf{b}-\mathbf{a}}$. To see this, fix $v' \in V'$ other than v_0, and recall that $v'|_{\mathbf{b}-\mathbf{a}} = \left(\sum_{\mathbf{i}:\mathsf{wt}(\mathbf{i})=i} v'_{\mathbf{i}} \cdot (\mathbf{b}-\mathbf{a})^{\mathbf{i}}\right)_{i=0,1,\ldots,s-1}$ and $v_0|_{\mathbf{b}-\mathbf{a}} = \left(\sum_{\mathbf{i}:\mathsf{wt}(\mathbf{i})=i} (v_0)_{\mathbf{i}} \cdot (\mathbf{b}-\mathbf{a})^{\mathbf{i}}\right)_{i=0,1,\ldots,s-1}$. Since $v' \neq v_0$, there exists some $i \in \{0,1,\ldots,s-1\}$ so that the polynomials $h_{v'}(\mathbf{X}) := \sum_{\mathbf{i}:\mathsf{wt}(\mathbf{i})=i} v'_{\mathbf{i}} \cdot \mathbf{X}^{\mathbf{i}}$ and $h_{v_0}(\mathbf{X}) := \sum_{\mathbf{i}:\mathsf{wt}(\mathbf{i})=i} (v_0)_{\mathbf{i}} \cdot \mathbf{X}^{\mathbf{i}}$ are distinct. But since both polynomials are of degree at most s, this implies in turn that with probability at least $1-\frac{s}{q}$ over the choice of $\mathbf{a}$ and $\mathbf{b}$ it holds that $h_{v'}(\mathbf{b}-\mathbf{a}) \neq h_{v_0}(\mathbf{b}-\mathbf{a})$, in which case we shall also have that $v'|_{\mathbf{b}-\mathbf{a}} \neq v_0|_{\mathbf{b}-\mathbf{a}}$. By applying a union bound over all elements of V', we conclude that with probability at least $1-\frac{s|V'|}{q}$ there is no $v' \in V'$ other than $v_0 = f^{(<s)}(\mathbf{a})$ so that $v'|_{\mathbf{b}-\mathbf{a}} = v_0|_{\mathbf{b}-\mathbf{a}}$.

Local list decoding up to capacity. Above we outlined an algorithm which can local list decode multivariate multiplicity codes of sufficiently large multiplicity s up to their minimum relative distance of $\delta := 1-\frac{k}{sq}$. Recall however that the rate of m-variate multiplicity codes of multiplicity s is

$$\frac{\binom{m+k-1}{m}}{\binom{m+s-1}{m} \cdot q^m} \leq \frac{\frac{1}{m!}\cdot (m+k)^m}{\frac{1}{m!}\cdot s^m \cdot q^m} = \left(\frac{k+m}{k}\right)^m \cdot \left(\frac{k}{sq}\right)^m = \left(1+\frac{m}{k}\right)^m \cdot (1-\delta)^m,$$

and consequently the decoding radius achieved by the above algorithm is well below list-decoding capacity for fixed $m > 1$ and growing k. Nevertheless, this algorithm can be used to obtain another family of codes that is locally list decodable up to capacity.

In more detail, recall that in Section 7.3 we encountered the notion of *list recovery*, which extends the notion of list decoding. Recall that in list recovery, one is given for each codeword entry a small input list of possible values, and the goal is to return the list of all codewords that are consistent with most of these input lists. More formally, for $\alpha \in (0,1)$ and positive integers ℓ, L we say that a code $C \subseteq \Sigma^n$ is (α, ℓ, L)-list recoverable if for any subsets $S_1, \ldots, S_n \subseteq \Sigma$ of size at most ℓ each there are at most L different codewords $c \in C$ which satisfy that $w_i \notin S_i$ for at most an α-fraction of the indices $i \in [n]$. Note that list decoding corresponds to the special case of $\ell = 1$. The problem of list-recovering from an α-fraction of errors is the task of finding the list of these codewords.

We have already seen in Section 7.3 that the notion of list recovery could be useful for local list decoding. Another motivation to consider the notion of list recovery is that it is known that *high rate* codes (of rate close to 1) that are list-recoverable from a constant fraction of errors and constant-size input lists S_i can be transformed into capacity-achieving list-decodable codes, and furthermore, this transformation also preserves locality (the transformation is based on expander graphs, and is beyond the scope of the current survey, see [KRSW23, Appendix C] for more details).

As is typically the case, all the local list decoding algorithms mentioned above could be extended to the setting of list recovery with similar performance. Furthermore, another advantage of multivariate multiplicity codes over Reed-Muller Codes is that they can achieve a higher rate. Specifically, while the rate of m-variate multiplicity codes of multiplicity s of relative distance $\delta := 1 - \frac{k}{sq}$ is at least

$$\frac{\binom{m+k-1}{m}}{\binom{m+s-1}{m} \cdot q^m} \geq \frac{\frac{1}{m!} \cdot k^m}{\frac{1}{m!} \cdot (s+m)^m \cdot q^m} = \left(\frac{s}{s+m}\right)^m \cdot \left(\frac{k}{sq}\right)^m = \left(1 - \frac{m}{s+m}\right)^m \cdot (1-\delta)^m,$$

the rate of m-variate Reed-Muller Codes of relative distance $\delta' := 1 - \frac{k}{q}$ is only

$$\frac{\binom{m+k-1}{m}}{q^m} \leq \frac{\frac{1}{m!} \cdot (m+k)^m}{q^m} = \frac{1}{m!} \cdot \left(\frac{k+m}{k}\right)^m \cdot \left(\frac{k}{q}\right)^m = \frac{1}{m!} \cdot \left(1 + \frac{m}{k}\right)^m \cdot (1-\delta')^m,$$

In particular, this means that for any $m > 1$, Reed-Muller Codes have rate at most $\frac{1}{2}$, while multivariate multiplicity codes can have rate close to 1 for $\delta \approx \frac{1}{m}$, $s \approx m^2$, and growing m. Consequently, one can use the extension of the local list decoding algorithm for multivariate multiplicity codes to the setting of list recovery to get high-rate locally list recoverable codes, which can be transformed in turn into capacity-achieving locally list decodable codes. Once more, we refer the reader to [KRSW23, Appendix C] for more details.

7.5 Bibliographic notes

Locally decodable and locally correctable codes were introduced in [BFLS91, KT00, STV01] and [Lip90, BK95], respectively, see [Yek12] for a comprehensive survey on locally decodable and locally correctable codes. Locally list decodable codes were introduced in [GL89, STV01], and since their introduction they found various applications in theoretical computer science, in cryptography [GL89], learning theory [KM93], average-to-worst-case reductions [Lip90], and hardness amplification [BFNW93, STV01].

The *local correction* algorithm for Reed-Muller Codes , presented in Section 7.1, was suggested in [Lip90, BF90, GLR$^+$91] (see also [Yek12, Section 2.2.2]), while the improvement using curves, described in Remark 7.5, was suggested by Gemmell and Sudan [GS92] (see also [Yek12, Section 2.2.3]). The *local list decoding* algorithm for Reed-Muller Codes beyond the unique decoding radius, presented in Section 7.2, was suggested by Arora and Sudan [AS03] and Sudan, Trevisan, and Vadhan [STV01]. The local list decoding algrotihm for Reed-Muller Codes *up to the Johnson Bound*, presented in Section 7.3, follows the ideas suggested by Kopparty, Ron-Zewi, Saraf, and Wootters [KRSW23] for local list decoding of multivariate multiplicity codes up to their minimum distance. Here we present a simpler version of their algorithm for local list decoding of Reed-Muller Codes up to the Johnson Bound. Alternative approaches for local list decoding of Reed-Muller Codes up to the Johnson Bound, based on restriction to planes instead of lines, were suggested by Brander and Kopparty [BK09] (see also [GK16a, Kop15]).

A *local correction algorithm* for *multivariate multiplicity codes* was suggested by Kopparty, Saraf, and Yekhanin [KSY14], who also introduced these codes and observed that they can achieve higher rate than the corresponding Reed-Muller Codes. The *local list decoding* algorithm for multivariate multiplicity codes up to their minimum distance, outlined in Section 7.4, was suggested in [KRSW23].

The *expander-based transformation* that turns a high-rate locally list recoverable code into a capacity-achieving locally list decodable code, mentioned in Section 7.4, was suggested by Gopi, Kopparty, Oliveira, Ron-Zewi, and Saraf [GKO$^+$18] and Hemenway, Ron-Zewi, and Wootters [HRW20]. It is based on the expander-based transformation that was suggested by Alon, Edmonds, and Luby in the unique decoding setting [AEL95, AL96], and its extensions to list decoding by Guruswami and Indyk [GI02], and local decoding by Kopparty, Meir, Ron-Zewi, and Saraf [KMRS17]. More recently, this transformation was used to obtain expander-based constructions of capacity-achieving (non-local) list decodable codes (not relying on polynomial-based codes) [ST25], and even ones achieving the generalized Singleton Bound [JMST25].

In this section we discussed *local* list decoding algorithms for multivariate codes. Such algorithms are inherently randomized (since an adversary can easily corrupt the answers to all of the queries of a deterministic algorithm), and only work when the code is evaluated over all of $\mathbb{F}_q^m$ (because of the need to pass a random line – or a related algebraic low-dimensional structure – through the domain). Over $\mathbb{F}_q^m$, Pellikaan and Wu [PW04] gave an efficient *deterministic* (global) list decoding algorithm for Reed-Muller Codes up to the Johnson Bound, and this algorithm was extended by Kopparty [Kop15] to list decoding of multivariate multiplicity codes of arbitrary multiplicity up to the Johnson Bound. More recently, Bhandari, Harsha, Kumar, and Sudan [BHKS24a] gave an efficient deterministic algorithm for list decoding multivariate multiplicity codes, over a constant number of variables and of sufficiently large multiplicity, *up to their minimum distance*.

Interestingly, the bound on the number of roots with multiplicities given by Theorem 2.6 also applies when the domain is a *product set* of the form $A_1 \times \cdots \times A_m$, where $A_1, \ldots, A_m$ are arbitrary subsets of $\mathbb{F}_q$. However, over arbitrary product sets, Reed-Muller Codes were only shown relatively recently to be efficiently uniquely decodable up to half their minimum distance by Kim and Kopparty [KK17], and this algorithm was extended by Bhandari, Harsha, Kumar, and Shanka [BHKS23] to unique decoding of multivariate multiplicity codes of arbitrary multiplicity over arbitrary product sets. We are not aware of any efficient algorithm that list decodes Reed-Muller Codes, or multivariate multiplicity codes of arbitrary multiplicity, over arbitrary product sets beyond half the minimum distance. Curiously, the aforementioned algorithm of [BHKS24a] for list decoding constant-variate multiplicity codes of sufficiently large multiplicity up to their minimum distance does work over arbitrary product sets.

8 Conclusion and open problems

In this book, we surveyed recent advances on list decoding of polynomial codes. Specifically, we investigated the list-decoding properties of Reed-Solomon Codes, showed that related families of polynomial codes such as multiplicity codes are efficiently list decodable up to capacity, and discussed how to obtain improved combinatorial upper bounds on the list sizes using a more refined analysis. We also presented fast (near-linear time) implementations of these list-decoding algorithms, and local (sublinear-time) list-decoding algorithms for multivariate polynomial codes such as Reed-muller Codes and multivariate multiplicity codes.

Open questions. We end this book with a couple of intriguing questions for further research that are still open by the time of writing of this survey.

1. **Explicit and efficiently list decodable Reed-Solomon Codes achieving list-deocding capacity.** In Section 3.4, we showed that Reed-Solomon Codes are generally *not* list decodable much beyond the Johnson bound, while in Section 5.1 we showed that Reed-Solomon Codes over *random evaluation points* are (combinatorially) list decodable up to capacity with optimal list size, with high probability. A very interesting question is to find explicit evaluation points for which Reed-Solomon Codes are list decodable up to capacity, as well as efficient list decoding algorithms.

2. **Explicit and efficiently list decodable codes achieving list-decoding capacity over a fixed-size alphabet.** In Section 5.4, we mentioned that one can obtain capacity-achieving list-decodable codes over a constant-size alphabet, whose size only depends on the gap ϵ to capacity, by resorting to Algebraic-Geometric (AG) codes. A major open problem is to obtain explicit constructions of capacity-achieving list decodable codes, as well as efficient list-decoding algorithms, over a *fixed-size* alphabet (independent of ϵ), for example over the *binary* alphabet. Over a q-ary alphabet, the list-decoding capacity is known to be $H_q^{-1}(1-R)$, where

$$H_q(\alpha) = \alpha \log_q(q-1) + \alpha \log_q\left(\frac{1}{\alpha}\right) + (1-\alpha)\log_q\left(\frac{1}{1-\alpha}\right)$$

   is the q-ary entropy function.

3. **Linear-time encodable and list decodable codes achieving list-decoding capacity.** In Section 6 we presented fast near-linear time algorithms for list deocding multiplicity codes up to capacity. Obtaining *truly linear-time* list-decoding algorithms (as well as truly linear-time encoding algorithms for capacity-acheiving list-decodable codes) is still open. Since in the unique decoding setting, expander codes can be used to obtain linear-time encodable and decodable codes, it could be that the recent line of work on list-decoding of expander codes [ST25, JMST25] could be used to obtain such codes.

4. **Applications in theoretical computer science.** We mentioned in Section 1 that list decodable codes have many applications in theoretical computer science (some references are given in Section 1.4). It would be very interesting to investigate if any of the developments we surveyed in this book could be useful for such applicaitons.

Acknowledgement. We would like to thank Zeyu Guo, Prahladh Harsha, Swastik Koppparty, Ramprasad Saptharishi, Shubhangi Saraf, Madhu Sudan, Amnon Ta-Shma, S. Venkitesh, and Mary Wootters for many enlightening discussions over the years about list decoding of polynomial codes which inspired and influenced the writing of this survey.

Noga Ron-Zewi was partially suppported by the European Union (ERC, ECCC, 101076663) while writing this survey. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

Mrinal Kumar's work is supported by the Department of Atomic Energy, Government of India, under project no. RTI4014, and partially by grants from ANRF, Google Research and Premji Invest.

References

[AEL95] Noga Alon, Jeff Edmonds, and Michael Luby. Linear time erasure codes with nearly optimal recovery. In *proceedings of the 36th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 512–519. IEEE Computer Society, 1995.

[AGG$^+$25] Omar Alrabiah, Zeyu Guo, Venkatesan Guruswami, Ray Li, and Zihan Zhang. Random reed-solomon codes achieve list-decoding capacity with linear-sized alphabets. *Advances in Combinatorics*, 2025. `http://dx.doi.org/10.19086/aic.2025.8`.

[AHS26] Vikrant Ashvinkumar, Mursalin Habib, and Shashank Srivastava. Algorithmic improvements to list decoding of folded reed-solomon codes. In *Proceedings of the 2026 ACM-SIAM Symposium on Discrete Algorithms, SODA*. SIAM, 2026.

[AL96] Noga Alon and Michael Luby. A linear time erasure-resilient code with nearly optimal recovery. *IEEE Transactions on Information Theory*, 42(6):1732–1736, 1996.

[Ale05] Michael Alekhnovich. Linear Diophantine equations over polynomials and soft decoding of reed-solomon codes. *IEEE Trans. Inform. Theory*, 51(7):2257–2265, 2005.

[AS03] Sanjeev Arora and Madhu Sudan. Improved low-degree testing and its applications. *Comb.*, 23(3):365–426, 2003.

[BCDZ25] Joshua Brakensiek, Yeyuan Chen, Manik Dhar, and Zihan Zhang. From random to explicit via subspace designs with applications to local properties and matroids, 2025.

[BDGZ25] Joshua Brakensiek, Manik Dhar, Sivakanth Gopi, and Zihan Zhang. AG codes achieve list-decoding capacity over constant-sized fields. *IEEE Trans. Inf. Theory*, 71(8):5935–5956, 2025.

[Ber68a] E. Berlekamp. Nonbinary bch decoding (abstr.). *IEEE Transactions on Information Theory*, 14(2):242–242, 1968.

[Ber68b] Elwyn R. Berlekamp. *Algebraic coding theory*. McGraw-Hill series in systems science. McGraw-Hill, 1968.

[BF90] Donald Beaver and Joan Feigenbaum. Hiding instances in multioracle queries. In Christian Choffrut and Thomas Lengauer, editors, *STACS 90, 7th Annual Symposium on Theoretical Aspects of Computer Science, Rouen, France, February 22-24, 1990, Proceedings*, volume 415 of *Lecture Notes in Computer Science*, pages 37–48. Springer, 1990.

[BFLS91] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 21–31. ACM Press, 1991.

[BFNW93] László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3(4):307–318, 1993.

[BGM22] Joshua Brakensiek, Sivakanth Gopi, and Visu Makam. Lower bounds for maximally recoverable tensor codes and higher order MDS codes. *IEEE Trans. Inf. Theory*, 68(11):7125–7140, 2022.

[BGM25] Joshua Brakensiek, Sivakanth Gopi, and Visu Makam. Generic reed–solomon codes achieve list-decoding capacity. *SIAM Journal on Computing*, pages STOC23–118–STOC23–154, 2025. To appear, `https://doi.org/10.1137/23M1598064`.

[BHKS23] Siddharth Bhandari, Prahladh Harsha, Mrinal Kumar, and Ashutosh Shankar. Algorithmizing the multiplicity schwartz-zippel lemma. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 2816–2835. SIAM, 2023.

[BHKS24a] Siddharth Bhandari, Prahladh Harsha, Mrinal Kumar, and Madhu Sudan. Decoding multivariate multiplicity codes on product sets. *IEEE Transactions on Information Theory*, 70(1):154–169, 2024.

[BHKS24b] Siddharth Bhandari, Prahladh Harsha, Mrinal Kumar, and Madhu Sudan. Ideal-theoretic explanation of capacity-achieving decoding. *IEEE Trans. Inf. Theory*, 70(2):1107–1123, 2024.

[BK95] Manuel Blum and Sampath Kannan. Designing programs that check their work. *J. ACM*, 42(1):269–291, 1995.

[BK09] Kristian Brander and Swastik Koparty. List-decoding reed-muller over large fields upto the johnson radius. *Manuscript*, 2009.

[BKR10] Eli Ben-Sasson, Swastik Kopparty, and Jaikumar Radhakrishnan. Subspace polynomials and limits to list decoding of reed-solomon codes. *IEEE Trans. Inf. Theory*, 56(1):113–120, 2010.

[BW87] E. R. Berlekamp and L. Welch. Error correction of algebraic block codes. US Patent Number 4,633,470, 1987.

[CX18] Chandra Chekuri and Chao Xu. Minimum cuts and sparsification in hypergraphs. *SIAM J. Comput.*, 47(6):2118–2156, 2018.

[CZ25] Yeyuan Chen and Zihan Zhang. Explicit folded reed-solomon and multiplicity codes achieve relaxed generalized singleton bounds. In Michal Koucký and Nikhil Bansal, editors, *Proceedings of the 57th Annual ACM Symposium on Theory of Computing, STOC 2025, Prague, Czechia, June 23-27, 2025*, pages 1–12. ACM, 2025.

[DKSS13] Zeev Dvir, Swastik Kopparty, Shubhangi Saraf, and Madhu Sudan. Extensions to the method of multiplicities, with applications to Kakeya sets and mergers. *SIAM Journal on Computing*, 42(6):2305–2328, 2013.

[DL78] Richard A. DeMillo and Richard J. Lipton. A probabilistic remark on algebraic program testing. *Inf. Process. Lett.*, 7(4):193–195, 1978.

[DSY14] Son Hoang Dau, Wentu Song, and Chau Yuen. On the existence of MDS codes over small fields with constrained generator matrices. In *2014 IEEE International Symposium on Information Theory*, pages 1787–1791, 2014. `https://doi.org/10.1109/ISIT.2014.6875141`.

[Eli57] Peter Elias. List decoding for noisy channels. *Research Laboratory of Electronics, MIT*, Technical Report 335, 1957.

[FK08] András Frank and Tamás Király. A survey on covering supermodular functions. In William J. Cook, László Lovász, and Jens Vygen, editors, *Research Trends in Combinatorial Optimization, Bonn Workshop on Combinatorial Optimization, November 3-7, 2008, Bonn, Germany*, pages 87–126. Springer, 2008.

[FKK03a] András Frank, Tamás Király, and Matthias Kriesell. On decomposing a hypergraph into k connected sub-hypergraphs. *Discret. Appl. Math.*, 131(2):373–383, 2003.

[FKK03b] András Frank, Tamás Király, and Zoltán Király. On the orientation of graphs and hypergraphs. *Discrete Applied Mathematics*, 131(2):385–400, 2003.

[Fra11] András Frank. *Connections in combinatorial optimization*, volume 38. Oxford University Press, 2011.

[GHKS24] Rohan Goyal, Prahladh Harsha, Mrinal Kumar, and Ashutosh Shankar. Fast list-decoding of univariate multiplicity and folded Reed-Solomon codes. In Santosh Vempala, editor, *Proc.* 65*th IEEE Symp. on Foundations of Comp. Science (FOCS)*, 2024.

[GI02] Venkatesan Guruswami and Piotr Indyk. Near-optimal linear-time codes for unique decoding and new list-decodable codes over smaller alphabets. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC)*, pages 812–821. ACM Press, 2002.

[GK16a] Alan Guo and Swastik Kopparty. List-decoding algorithms for lifted codes. *IEEE Transactions on Information Theory*, 62(5):2719–2725, 2016.

[GK16b] Venkatesan Guruswami and Swastik Kopparty. Explicit subspace designs. *Combinatorica*, 36(2):161–185, 2016.

[GKO$^+$18] Sivakanth Gopi, Swastik Kopparty, Rafael Oliveira, Noga Ron-Zewi, and Shubhangi Saraf. Locally testable and locally correctable codes approaching the gilbert-varshamov bound. *IEEE Transactions on Information Theory*, 64(8):5813–5831, 2018.

[GL89] Oded Goldreich and Leonid A Levin. A hard-core predicate for all one-way functions. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC)*, pages 25–32. ACM, 1989.

[GLR$^+$91] Peter Gemmell, Richard J. Lipton, Ronitt Rubinfeld, Madhu Sudan, and Avi Wigderson. Self-testing/correcting for polynomials and for approximate functions. In Cris Koutsougeras and Jeffrey Scott Vitter, editors, *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 32–42. ACM, 1991.

[GLS$^+$24] Zeyu Guo, Ray Li, Chong Shangguan, Itzhak Tamo, and Mary Wootters. Improved list-decodability and list-recoverability of reed-solomon codes via tree packings. *SIAM J. Comput.*, 53(2):389–430, 2024.

[Gop83] V. D. Goppa. Algebraico-geometric codes. *Math. USSR-Izv.*, 21:75–91, 1983.

[GR08] Venkatesan Guruswami and Atri Rudra. Explicit codes achieving list decoding capacity: Error-correction with optimal redundancy. *IEEE Transactions on Information Theory*, 54(1):135–150, 2008.

[GR22] Zeyu Guo and Noga Ron-Zewi. Efficient list-decoding with constant alphabet and list sizes. *IEEE Trans. Inf. Theory*, 68(3):1663–1682, 2022.

[GRS] Venkatesan Guruswami, Atri Rudra, and Madhu Sudan. Essential coding theory. `https://cse.buffalo.edu/faculty/atri/courses/coding-theory/book/`. Last accessed: August 2025.

[GRS00] Oded Goldreich, Ronitt Rubinfeld, and Madhu Sudan. Learning polynomials with queries: The highly noisy case. *SIAM J. Discret. Math.*, 13(4):535–570, 2000.

[GS92] Peter Gemmell and Madhu Sudan. Highly resilient correctors for polynomials. *Inf. Process. Lett.*, 43(4):169–174, 1992.

[GS99] Venkatesan Guruswami and Madhu Sudan. Improved decoding of Reed-Solomon and algebraic-geometry codes. *IEEE Transactions on Information Theory*, 45(6):1757–1767, 1999.

[GST24] Eitan Goldberg, Chong Shangguan, and Itzhak Tamo. Singleton-type bounds for list-decoding and list-recovery, and related results. *J. Comb. Theory A*, 203:105835, 2024.

[Gur04] Venkatesan Guruswami. *List decoding of error-correcting codes: winning thesis of the 2002 ACM doctoral dissertation competition*, volume 3282. Springer Science & Business Media, 2004.

[Gur06a] V. Guruswami. List decoding in average-case complexity and pseudorandomness. In *2006 IEEE Information Theory Workshop - ITW '06 Punta del Este*, pages 32–36, 2006.

[Gur06b] Venkatesan Guruswami. Algorithmic results in list decoding. *Foundations and Trends in Theoretical Computer Science*, 2(2), 2006.

[GW13] Venkatesan Guruswami and Carol Wang. Linear-algebraic list decoding for variants of Reed-Solomon codes. *IEEE Transactions on Information Theory*, 59(6):3257–3268, 2013.

[GX22] Venkatesan Guruswami and Chaoping Xing. Optimal rate list decoding over bounded alphabets using algebraic-geometric codes. *J. ACM*, 69(2):10:1–10:48, 2022.

[GZ61] Daniel Gorenstein and Neal Zierler. A class of error-correcting codes in pm symbols. *Journal of the Society for Industrial and Applied Mathematics*, 9(2):207–214, 1961.

[GZ23] Zeyu Guo and Zihan Zhang. Randomly punctured reed-solomon codes achieve the list decoding capacity over polynomial-size alphabets. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*, pages 164–176. IEEE, 2023.

[Ham50] Richard Hamming. Error detecting and error correcting codes. *The Bell System Technical Journal*, 29(2):147–160, 1950.

[HRW20] Brett Hemenway, Noga Ron-Zewi, and Mary Wootters. Local list recovery of high-rate tensor codes and applications. *SIAM Journal on Computing*, 49(4):157–195, 2020.

[JMS03] Kamal Jain, Mohammad Mahdian, and Mohammad R. Salavatipour. Packing steiner trees. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 12-14, 2003, Baltimore, Maryland, USA*, pages 266–274. ACM/SIAM, 2003.

[JMST25] Fernando Granha Jeronimo, Tushant Mittal, Shashank Srivastava, and Madhur Tulsiani. Explicit codes approaching generalized singleton bound using expanders. *CoRR*, abs/2502.07308, 2025.

[Jos58] D. D. Joshi. A note on upper bounds for minimum distance codes. *Inf. Control.*, 1(3):289–295, 1958.

[Kal85] Erich L. Kaltofen. Polynomial-time reductions from multivariate to bi- and univariate integral polynomial factorization. *SIAM J. Comput.*, 14(2):469–489, 1985.

[Kir03] Tamás Király. *Edge-connectivity of undirected and directed hypergraphs*. PhD thesis, Eötvös Loránd University, 2003.

[KK17] John Y. Kim and Swastik Kopparty. Decoding reed-muller codes over product sets. *Theory of Computing*, 13(21):1–38, 2017.

[KM93] Eyal Kushilevitz and Yishay Mansour. Learning decision trees using the Fourier spectrum. *SIAM Journal on Computing*, 22(6):1331–1348, 1993.

[KMRS17] Swastik Kopparty, Or Meir, Noga Ron-Zewi, and Shubhangi Saraf. High-rate locally correctable and locally testable codes with sub-polynomial query complexity. *Journal of ACM*, 64(2):11:1–11:42, 2017.

[Kop15] Swastik Kopparty. List-decoding multiplicity codes. *Theory of Computing*, 11(5):149–182, 2015.

[KRSW23] Swastik Kopparty, Noga Ron-Zewi, Shubhangi Saraf, and Mary Wootters. Improved list decoding of folded reed-solomon and multiplicity codes. *SIAM J. Comput.*, 52(3):794–840, 2023.

[KSY14] Swastik Kopparty, Shubhangi Saraf, and Sergey Yekhanin. High-rate codes with sublinear-time decoding. *J. ACM*, 61(5):28:1–28:20, 2014.

[KT00] Jonathan Katz and Luca Trevisan. On the efficiency of local decoding procedures for error-correcting codes. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 80–86. ACM Press, 2000.

[Len85] Arjen K. Lenstra. Factoring multivariate polynomials over finite fields. *J. Comput. Syst. Sci.*, 30(2):235–248, 1985. (Preliminary version in *15th STOC*, 1983).

[Lip90] Richard J. Lipton. Efficient checking of computations. In *proceedings of the 7th Annual ACM Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 415 of *lncs*, pages 207–215. Springer, 1990.

[LMS25] Matan Levi, Jonathan Mosheiff, and Nikhil Shagrithaya. Random reed-solomon codes and random linear codes are locally equivalent. In *proceedings of the 66th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. IEEE Computer Society, 2025.

[Lov21] Shachar Lovett. Sparse MDS Matrices over Small Fields: A Proof of the GM-MDS Conjecture. *SIAM Journal on Computing*, 50(4):1248–1262, 2021. https://doi.org/10.1137/20M1323345.

[Mas69] James L. Massey. Shift-register synthesis and BCH decoding. *IEEE Trans. Inf. Theory*, 15(1):122–127, 1969.

[Men27] Karl Menger. Zur allgemeinen kurventheorie. *Fundamenta Mathematicae*, 10(1):96–115, 1927.

[MRR+24] Jonathan Mosheiff, Nicolas Resch, Noga Ron-Zewi, Shashwat Silas, and Mary Wootters. Low-density parity-check codes achieve list-decoding capacity. *SIAM J. Comput.*, 53(6):S20–38, 2024.

[Mul54] David Muller. Application of boolean algebra to switching circuit design and to error detection. *Transactions of the IRE Professional Group on Electronic Computers*, 3(3):6–12, 1954.

[Pet60] W. Wesley Peterson. Encoding and error-correction procedures for the Bose-Chaudhuri codes. *IRE Transactions on Information Theory*, 6(4):459–470, 1960.

[PW04] R. Pellikaan and Xin-Wen Wu. List decoding of q-ary reed-muller codes. *IEEE Transactions on Information Theory*, 50(4):679–682, 2004.

[Ree54] Irving Reed. A class of multiple-error-correcting codes and the decoding scheme. *Transactions of the IRE Professional Group on Information Theory*, 4:38–49, 1954.

[Rot22] Ron M. Roth. Higher-order MDS codes. *IEEE Trans. Inf. Theory*, 68(12):7798–7816, 2022.

[RR00] Ron M. Roth and Gitit Ruckenstein. Efficient decoding of Reed-Solomon codes beyond half the minimum distance. *IEEE Trans. Inform. Theory*, 46(1):246–257, 2000.

[RS60] Irving S. Reed and Gustave Solomon. Polynomial codes over certain finite fields. *SIAM Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, 1960. `https://doi.org/10.1137/0108018`.

[RT97] Michael Rosenbloom and Michael Tsfasman. Codes for the m-metric. *Problemy Peredachi Informatsii*, 33(1):55–63, 1997.

[RV25] Nicolas Resch and S. Venkitesh. List recoverable codes: The good, the bad, and the unknown (hopefully not ugly). *arXiv preprint 2510.07597*, 2025. `https://arxiv.org/abs/2510.07597`.

[Sch80] Jacob T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, 1980.

[Sha48] Claude Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379—-423, 623—-656, 1948.

[Sin64] Richard C. Singleton. Maximum distance q -nary codes. *IEEE Trans. Inf. Theory*, 10(2):116–118, 1964.

[ST20] Chong Shangguan and Itzhak Tamo. Combinatorial list-decoding of reed-solomon codes beyond the johnson radius. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 538–551. ACM, 2020.

[ST25] Shashank Srivastava and Madhur Tulsiani. List decoding expander-based codes up to capacity in near-linear time. *CoRR*, abs/2504.20333, 2025.

[STV01] Madhu Sudan, Luca Trevisan, and Salil Vadhan. Pseudorandom generators without the XOR lemma. *Journal of Computer and System Sciences*, 62(2):236–266, 2001.

[Sud97] Madhu Sudan. Decoding of Reed Solomon codes beyond the error-correction bound. *Journal of Complexity*, 13(1):180–193, 1997.

[Sud00] Madhu Sudan. List decoding: algorithms and applications. *SIGACT News*, 31(1):16–27, March 2000.

[Tam24] Itzhak Tamo. Tighter list-size bounds for list-decoding and recovery of folded reed-solomon and multiplicity codes. *IEEE Trans. Inf. Theory*, 70(12):8659–8668, 2024.

[Tre04] Luca Trevisan. Some applications of coding theory in computational complexity. *Electron. Colloquium Comput. Complex.*, TR04-043, 2004.

[Vad12] Salil P. Vadhan. Pseudorandomness. *Found. Trends Theor. Comput. Sci.*, 7(1-3):1–336, 2012.

[vzGG13] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, 3 edition, 2013.

[Woz57] John Wozencraft. List decoding. *Quarterly Progress Report, Research Laboratory of Electronics, MIT*, 48:90–95, 1957.

[Yek12] Sergey Yekhanin. Locally decodable codes. *Found. Trends Theor. Comput. Sci.*, 6(3):139–255, 2012.

[YH19] Hikmet Yildiz and Babak Hassibi. Optimum Linear Codes With Support-Constrained Generator Matrices Over Small Fields. *IEEE Transactions on Information Theory*, 65(12):7868–7875, 2019. `https://doi.org/10.1109/TIT.2019.2932663`.

[Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In Edward W. Ng, editor, *Symbolic and Algebraic Computation, EUROSAM '79, An International Symposiumon Symbolic and Algebraic Computation, Marseille, France, June 1979, Proceedings*, volume 72 of *Lecture Notes in Computer Science*, pages 216–226. Springer, 1979.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il