Chapter 3

LIST DECODING OF FOLDED REED-SOLOMON CODES

3.1 Introduction

Even though list decoding was defined in the late 1950s, there was essentially no algorithmic progress that could harness the potential of list decoding for nearly forty years. The work of Sudan [97] and improvements to it by Guruswami and Sudan in [63], achieved efficient list decoding up to a $\rho_{\mathrm{GS}}(R) = 1 - \sqrt{R}$ fraction of errors for Reed-Solomon codes of rate R. Note that $1 - \sqrt{R} > \rho_U(R) = (1 - R)/2$ for every rate R, $0 < R < 1$, so this result showed that list decoding can be effectively used to go beyond the unique decoding radius for every rate (see Figure 3.1). The ratio $\rho_{\mathrm{GS}}(R)/\rho_U(R)$ approaches 2 for rates $R \to 0$, enabling error-correction when the fraction of errors approaches 100%, a feature that has found numerous applications outside coding theory, see for example [98], [49, Chap. 12].

Unfortunately, the improvement provided by [63] over unique decoding diminishes for larger rates, which is actually the regime of greater practical interest. For rates $R \to 1$, the ratio $\frac{\rho_{\mathrm{GS}}(R)}{\rho_U(R)}$ approaches 1, and already for rate $R = 1/2$ the ratio is at most 1.18. Thus, while the results of [97, 63] demonstrated that list decoding always, for every rate, enables correcting more errors than unique decoding, they fell short of realizing the full quantitative potential of list decoding (recall that the list-decoding capacity promises error correction up to a $1 - R = 2\rho_U(R)$ fraction of errors).

The bound $\rho_{\mathrm{GS}}(R)$ stood as the best known decoding radius for efficient list decoding (for any code) for several years. In fact constructing (ρ, L)-list decodable codes of rate R for $\rho > \rho_{\mathrm{GS}}(R)$ and polynomially bounded L, regardless of the complexity of actually performing list decoding to radius ρ, itself was elusive. Some of this difficulty was due to the fact that $1 - \sqrt{R}$ is the largest radius for which small list size can be shown generically, via the so-called Johnson bound which argues about the number of codewords in Hamming balls using only information on the relative distance of the code, cf. [48].

In a recent breakthrough paper [85], Parvaresh and Vardy presented codes that are list-decodable beyond the $1 - \sqrt{R}$ radius for low rates R. The codes they suggest are variants of Reed-Solomon (or simply RS) codes obtained by evaluating $m \geqslant 1$ correlated polynomials at elements of the underlying field (with $m = 1$ giving RS codes). For any $m \geqslant 1$, they achieve the list-decoding radius $\rho_{\mathrm{PV}}^{(m)}(R) = 1 - \sqrt[m+1]{m^m R^m}$. For rates $R \to 0$, choosing m large enough, they can list decode up to radius $1 - O(R \log(1/R))$, which approaches the capacity $1 - R$. However, for $R \geqslant 1/16$, the best choice of m (the one that maximizes $\rho_{\mathrm{PV}}^{(m)}(R)$) is in fact $m = 1$, which reverts back to RS codes and the list-decoding radius $1 - \sqrt{R}$. (See Figure 3.1 where the bound $1 - \sqrt[3]{4R^2}$ for the case $m = 2$ is plotted

— except for very low rates, it gives a small improvement over $\rho_{\mathrm{GS}}(R)$.) Thus, getting arbitrarily close to capacity for some rate, as well as beating the $1 - \sqrt{R}$ bound for every rate, both remained open before our work[1].

In this chapter, we describe codes that get arbitrarily close to the list-decoding capacity for every rate (for large alphabets). In other words, we give explicit codes of rate R together with polynomial time list decoding up to a fraction $1 - R - \varepsilon$ of errors for every rate R and arbitrary $\varepsilon > 0$. As mentioned in Section 2.2.1, this attains the best possible trade-off one can hope for between the rate and list-decoding radius. This is the first result that approaches the list-decoding capacity for *any* rate (and over any alphabet).

Our codes are simple to describe: they are *folded Reed-Solomon codes*, which are in fact *exactly* Reed-Solomon codes, but viewed as codes over a larger alphabet by careful bundling of codeword symbols. Given the ubiquity of RS codes, this is an appealing feature of our result, and in fact our methods directly yield better decoding algorithms for RS codes when errors occur in *phased bursts* (a model considered in [75]).

Our result extends easily to the problem of *list recovery* (recall Definition 2.4). The biggest advantage here is that we are able to achieve a rate that is independent of the size of the input lists. This is an extremely useful feature that will be used in Chapters 4 and 5 to design codes over smaller alphabets. In particular, we will construct new codes from folded Reed-Solomon codes that achieve list-decoding capacity over constant sized alphabets (the folded Reed-Solomon codes are defined over alphabets whose size increases with the block length of the code).

Our work builds on existing work of Guruswami and Sudan [63] and Parvaresh and Vardy [85]. See Figure 3.1 for a comparison of our work with previous known list-decoding algorithms (for various codes).

We start with the description of our code in Section 3.2 and give some intuition why these codes might have good list decodable properties. We present the main ideas in our list-decoding algorithms for the folded Reed-Solomon codes in Section 3.3. In Section 3.4, we present and analyze a polynomial time list-decoding algorithm for folded RS codes of rate R that can correct roughly $1 - \sqrt[3]{R^2}$ fraction of errors . In Section 3.5, we extend the results in Section 3.4 to present codes that can be efficiently list decoded up to the list-decoding capacity. Finally, we extend our results to list recovery in Section 3.6.

3.2 Folded Reed-Solomon Codes

In this section, we will define a simple variant of Reed-Solomon codes called folded Reed-Solomon codes. By choosing parameters suitably, we will design a list-decoding algorithm that can decode close to the optimal fraction $1 - R$ of errors with rate R.

[1]Independent of our work, Alex Vardy (personal communication) constructed a variant of the code defined in [85] which could be list decoded with fraction of errors more than $1 - \sqrt{R}$ for all rates R. However, his construction gives only a small improvement over the $1 - \sqrt{R}$ bound and does not achieve the list-decoding capacity.

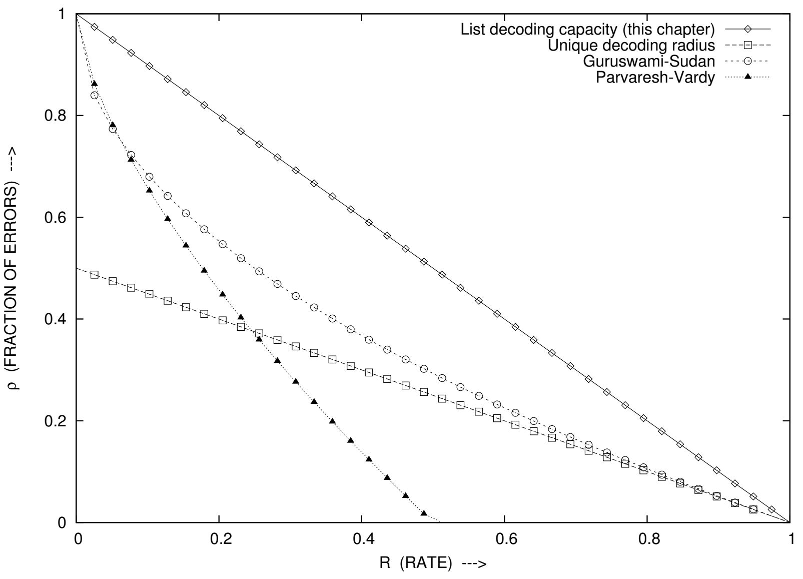

Figure 3.1: List-decoding radius ρ plotted against the rate R of the code for known algorithms. The best possible trade-off, i.e., list-decoding capacity, is $\rho = 1 - R$, and our work achieves this.

3.2.1 Description of Folded Reed-Solomon Codes

Consider a $[n, k+1]_q$ Reed-Solomon code C consisting of evaluations of degree k polynomials over $\mathbb{F}_q$ at the set $\mathbb{F}_q^*$. Note that $q = n + 1$. Let γ be a generator of the multiplicative group $\mathbb{F}_q^*$, and let the evaluation points be ordered as $1, \gamma, \gamma^2, \ldots, \gamma^{n-1}$. Using all nonzero field elements as evaluation points is one of the most commonly used instantiations of Reed-Solomon codes.

Let $m \geqslant 1$ be an integer parameter called the *folding parameter*. For ease of presentation, we will assume that m divides $n = q - 1$.

Definition 3.1 (Folded Reed-Solomon Code). *The m-folded version of the RS code C, denoted $\mathrm{FRS}_{\mathbb{F}_q,\gamma,m,k}$, is a code of block length $N = n/m$ over $\mathbb{F}_q^m$, where $n = q - 1$. The encoding of a message $f(X)$, a polynomial over $\mathbb{F}_q$ of degree at most k, has as its j'th symbol, for $0 \leqslant j < n/m$, the m-tuple $(f(\gamma^{jm}), f(\gamma^{jm+1}), \cdots, f(\gamma^{jm+m-1}))$. In other words, the codewords of $C' = \mathrm{FRS}_{\mathbb{F}_q,\gamma,m,k}$ are in one-one correspondence with those of the RS code C and are obtained by bundling together consecutive m-tuple of symbols in codewords of C.*

The way the above definition is stated the message alphabet is $\mathbb{F}_q$ while the codeword alphabet is $\mathbb{F}_q^m$ whereas in our definition of codes, both the alphabets were the same. This

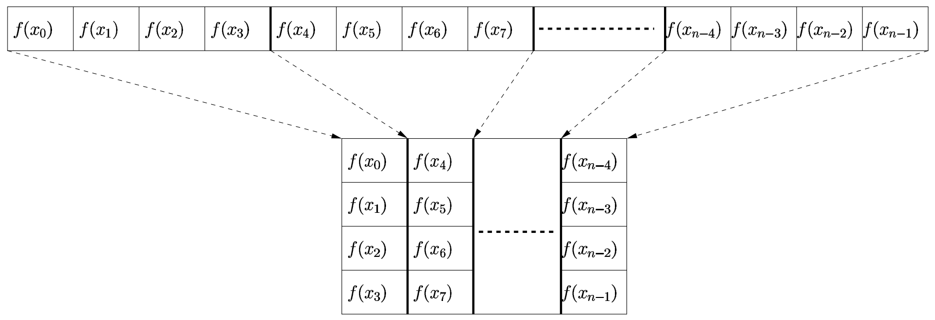

Figure 3.2: Folding of the Reed-Solomon Code with Parameter $m = 4$.

can be easily taken care of by bundling m consecutive message symbols from $\mathbb{F}_q$ to make the message alphabet to be $\mathbb{F}_q^m$. We will however, state our results with the message symbols as coming from $\mathbb{F}_q$ as this simplifies our presentation.

We illustrate the above construction for the choice $m = 4$ in Figure 3.2. The polynomial $f(X)$ is the message, whose Reed-Solomon encoding consists of the values of f at $x_0, x_1, \ldots, x_{n-1}$ where $x_i = \gamma^i$. Then, we perform a folding operation by bundling together tuples of 4 symbols to give a codeword of length $n/4$ over the alphabet $\mathbb{F}_q^4$.

Note that the folding operation does not change the rate R of the original Reed-Solomon code. The relative distance of the folded RS code also meets the Singleton bound and is at least $1 - R$.

Remark 3.1 (Origins of term "folded RS codes"). *The terminology of folded RS codes was coined in [75], where an algorithm to correct random errors in such codes was presented (for a noise model similar to the one used in [27, 18]: see Section 3.7 for more details). The motivation was to decode RS codes from many random "phased burst" errors. Our decoding algorithm for folded RS codes can also be likewise viewed as an algorithm to correct beyond the $1 - \sqrt{R}$ bound for RS codes if errors occur in large, phased bursts (the actual errors can be adversarial).*

3.2.2 Why Might Folding Help?

Since folding seems like such a simplistic operation, and the resulting code is essentially just a RS code but viewed as a code over a large alphabet, let us now understand why it can possibly give hope to correct more errors compared to the bound for RS codes.

Consider the folded RS code with folding parameter $m = 4$. First of all, decoding the folded RS code up to a fraction ρ of errors is certainly not harder than decoding the RS code up to the same fraction ρ of errors. Indeed, we can "unfold" the received word of the folded RS code and treat it as a received word of the original RS code and run the RS list-decoding algorithm on it. The resulting list will certainly include all folded RS codewords within distance ρ of the received word, and it may include some extra codewords which we can, of course, easily prune.

In fact, decoding the folded RS code is a strictly easier task. To see why, say we want to correct a fraction $1/4$ of errors. Then, if we use the RS code, our decoding algorithm ought to be able to correct an error pattern that corrupts every 4'th symbol in the RS encoding of $f(X)$ (i.e., corrupts $f(x_{4i})$ for $0 \leqslant i < n/4$). However, after the folding operation, this error pattern corrupts every one of the symbols over the larger alphabet $\mathbb{F}_q^4$, and thus need not be corrected. In other words, for the same fraction of errors, the folding operation reduces the total number of error patterns that need to be corrected, since the channel has less flexibility in how it may distribute the errors.

It is of course far from clear how one may exploit this to actually correct more errors. To this end, algebraic ideas that exploit the specific nature of the folding and the relationship between a polynomial $f(X)$ and its shifted counterpart $f(\gamma X)$ will be used. These will become clear once we describe our algorithms later in the chapter.

We note that the above simplification of the channel is not attained for free since the alphabet size increases after the folding operation. For folding parameter m that is an absolute constant, the increase in alphabet size is moderate and the alphabet remains polynomially large in the block length. (Recall that the RS code has an alphabet size that is linear in the block length.) Still, having an alphabet size that is a large polynomial is somewhat unsatisfactory. Fortunately, existing alphabet reduction techniques, which are used in Chapter 4, can handle polynomially large alphabets, so this does not pose a big problem.

3.2.3 *Relation to Parvaresh Vardy Codes*

In this subsection, we relate folded RS codes to the Parvaresh-Vardy (PV) codes [85], which among other things will help make the ideas presented in the previous subsection more concrete.

The basic idea in the PV codes is to encode a polynomial f of degree k by the evaluations of $s \geqslant 2$ polynomials $f_0 = f, f_1, \ldots, f_{s-1}$ where $f_i(X) = f_{i-1}(X)^d \mod E(X)$ for an appropriate power d (and some irreducible polynomial $E(X)$ of some appropriate degree) — let us call s the *order* of such a code. Our first main idea is to pick the irreducible polynomial $E(X)$ (and the parameter d) in such a manner that every polynomial f of degree at most k satisfies the following identity: $f(\gamma X) = f(X)^d \mod E(X)$, where γ is the generator of the underlying field. Thus, a folded RS code with bundling using an γ as above is in fact exactly the PV code of order $s = m$ for the set of evaluation points $\{1, \gamma^m, \gamma^{2m}, \ldots, \gamma^{(n/m-1)m}\}$. This is nice as it shows that PV codes can meet the Singleton bound (since folded RS codes do), but as such does not lead to any better codes for list decoding.

We now introduce our second main idea. Let us compare the folded RS code to a PV code of order 2 (instead of order m where m divides n) for the set of evaluation points $\{1, \gamma, \ldots \gamma^{m-2}, \gamma^m, \gamma^{m+1}, \ldots, \gamma^{2m-2}, \ldots, \gamma^{n-m}, \gamma^{n-m+1} \ldots, \gamma^{n-2}\}$. We find that in the PV encoding of f, for every $0 \leqslant i \leqslant n/m - 1$ and every $0 < j < m - 1$, $f(\gamma^{mi+j})$

appears exactly twice (once as $f(\gamma^{mi+j})$ and another time as $f_1(\gamma^{-1}\gamma^{mi+j})$), whereas it appears only once in the folded RS encoding. (See Figure 3.3 for an example when $m = 4$ and $s = 2$.) In other words, the PV and folded RS codes have the same information, but the

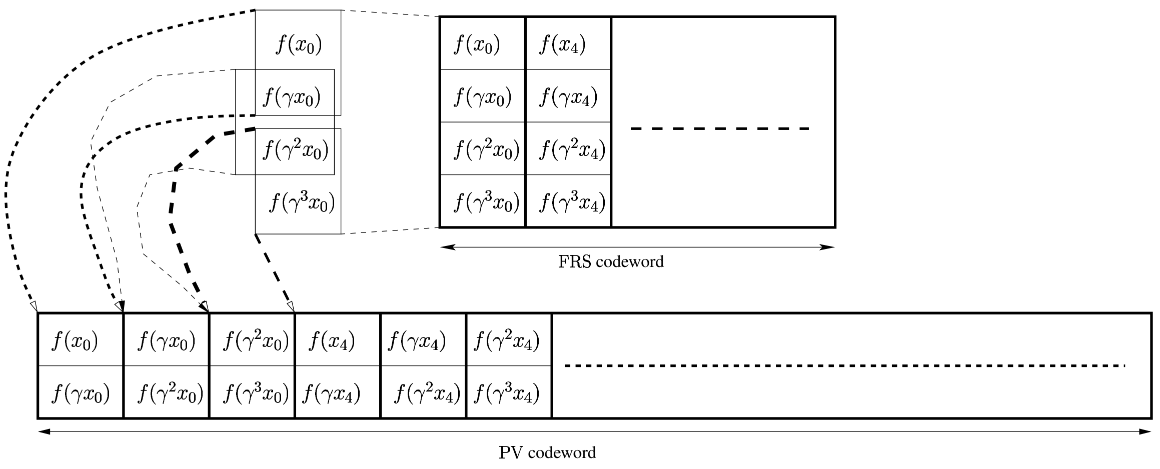

Figure 3.3: The correspondence between a folded Reed-Solomon code (with $m = 4$ and $x_i = \gamma^i$) and the Parvaresh Vardy code (of order $s = 2$) evaluated over $\{1, \gamma, \gamma^2, \gamma^4, \ldots, \gamma^{n-4}, \ldots, \gamma^{n-2}\}$. The correspondence for the first block in the folded RS codeword and the first three blocks in the PV codeword is shown explicitly in the left corner of the figure.

rate of the folded RS codes is bigger by a factor of $\frac{2m-2}{m} = 2 - \frac{2}{m}$. Decoding the folded RS codes from a fraction ρ of errors reduces to correcting the same fraction ρ of errors for the PV code. But the rate vs. list-decoding radius trade-off is better for the folded RS code since it has (for large enough m, almost) twice the rate of the PV code.

In other words, our folded RS codes are chosen such that they are compressed forms of suitable PV codes, and thus have better rate than the corresponding PV code for a similar error-correction performance. This is where our gain is, and using this idea we are able to construct folded RS codes of rate R that are list decodable up to radius roughly $1 - \sqrt[s+1]{R^s}$ for any $s \geqslant 1$. Picking s large enough lets us get within any desired ε of list-decoding capacity.

3.3 Problem Statement and Informal Description of the Algorithms

We first start by stating more precisely the problem we will solve in the rest of the chapter. We will give list-decoding algorithms for the folded Reed-Solomon code $\mathrm{FRS}_{\mathbb{F}_q,\gamma,m,k}$ of rate R. More precisely, for every $1 \leqslant s \leqslant m$ and $\delta > 0$, given a received word $\mathbf{y} =$

$\langle (y_0, \ldots, y_{m-1}), \ldots, (y_{n-m}, \ldots, y_{n-1}) \rangle$ (where recall $n = q - 1$), we want to output all codewords in $\mathrm{FRS}_{\mathbb{F}_q,\gamma,m,k}$ that disagree with $\mathbf{y}$ in at most $1 - (1+\delta)\left(\frac{m}{m-s+1}\right) R^{s/(s+1)}$ fraction of positions in polynomial time. In other words, we need to output all degree k polynomials $f(X)$ such that for at least $(1+\delta)\left(\frac{m}{m-s+1}\right) R^{s/(s+1)}$ fraction of $0 \leqslant i \leqslant n/m - 1$, $f(\gamma^{im+j}) = y_{im+j}$ (for every $0 \leqslant j \leqslant m-1$). By picking the parameters m, s and δ carefully, we will get folded Reed-Solomon codes of rate R that can be list decoded up to a $1 - R - \varepsilon$ fraction of errors (for any $\varepsilon > 0$).

We will now present the main ideas need to design our list-decoding algorithm. Readers familiar with list-decoding algorithms of [97, 63, 85] can skip the rest of this section.

For the ease of presentation we will start with the case when $s = m$. As a warm up, let us consider the case when $s = m = 1$. Note that for $m = 1$, we are interested in list decoding Reed-Solomon codes. More precisely, given the received word $\mathbf{y} = \langle y_0, \ldots, y_{n-1} \rangle$, we are interested in all degree k polynomials $f(X)$ such that for at least $(1+\delta)\sqrt{R}$ fraction of positions $0 \leqslant i \leqslant n-1$, $f(\gamma^i) = y_i$. We now sketch the main ideas of the algorithms in [97, 63]. The algorithms have two main steps: the first is an *interpolation* step and the second one is a *root finding* step. In the interpolation step, the list-decoding algorithm finds a bivariate polynomial $Q(X, Y)$ that *fits* the input. That is,

for every position i, $Q(\gamma^i, y_i) = 0$.

Such a polynomial $Q(\cdot, \cdot)$ can be found in polynomial time if we search for one with large enough total degree (this amounts to solving a system of linear equations). After the interpolation step, the root finding step finds all factors of $Q(X, Y)$ of the form $Y - f(X)$. The crux of the analysis is to show that

for every degree k polynomial $f(X)$ that satisfies $f(\gamma^i) = y_i$ for at least $(1 + \delta)\sqrt{R}$ fraction of positions i, $Y - f(X)$ is indeed a factor of $Q(X, Y)$.

However, the above is not true for every bivariate polynomial $Q(X, Y)$ that satisfies $Q(\gamma^i, y_i) = 0$ for all positions i. The main ideas in [97, 63] were to introduce more constraints on $Q(X, Y)$. In particular, the work of Sudan [97] added the constraint that a certain weighted degree of $Q(X, Y)$ is below a fixed upper bound. Specifically, $Q(X, Y)$ was restricted to have a non-trivially bounded $(1, k)$-weighted degree. The $(1, k)$-weighted degree of a monomial $X^i Y^j$ is $i + jk$ and the $(1, k)$-weighted degree of a bivariate polynomial $Q(X, Y)$ is the maximum $(1, k)$-weighted degree among its monomials. The intuition behind defining such a weighted degree is that given $Q(X, Y)$ with weighted $(1, k)$ of D, the *univariate* polynomial $Q(X, f(X))$, where $f(X)$ is some degree k polynomial, has total degree at most D. The upper bound D is chosen carefully such that if $f(X)$ is a codeword that needs to be output, then $Q(X, f(X))$ has more than D zeroes and thus $Q(X, f(X)) \equiv 0$, which in turn implies that $Y - f(X)$ divides $Q(X, Y)$. To get to the bound of $1 - (1+\delta)\sqrt{R}$, Guruswami and Sudan in [63], added a further constraint on $Q(X, Y)$ that required it to have r roots at (γ^i, y_i), where r is some parameter (in [97] $r = 1$ while in [63], r is roughly $1/\delta$).

We now consider the next non-trivial case of $m = s = 2$ (the ideas for this case can be easily generalized for the general $m = s$ case). Note that now given the received word $\langle (y_0, y_1), (y_2, y_3), \ldots, (y_{n-2}, y_{n-1}) \rangle$ we want to find all degree k polynomials $f(X)$ such that for at least $2(1+\delta)\sqrt[3]{R^2}$ fraction of positions $0 \leqslant i \leqslant n/2 - 1$, $f(\gamma^{2i}) = y_{2i}$ and $f(\gamma^{2i+1}) = y_{2i+1}$. As in the previous case, we will have an interpolation and a root finding step. The interpolation step is a straightforward generalization of $m = 1$ case: we find a trivariate polynomial $Q(X, Y, Z)$ that fits the received word, that is, for every $0 \leqslant i \leqslant n/2 - 1$, $Q(\gamma^{2i}, y_{2i}, y_{2i+1}) = 0$. Further, $Q(X, Y, Z)$ has an upper bound on its $(1, k, k)$-weighted degree (which is a straightforward generalization of the $(1, k)$-weighted degree for the bivariate case) and has a multiplicity of r at every point. These straightforward generalization and their various properties are recorded in Section 3.4.1. For the root finding step, it suffices to show that for every degree k polynomial $f(X)$ that needs to be output $Q(X, f(X), f(\gamma X)) \equiv 0$. This, however does not follow from weighted degree and multiple root properties of $Q(X, Y, Z)$. Here we will need two new ideas, the first of which is to show that for some irreducible polynomial $E(X)$ of degree $q - 1$, $f(X)^q \equiv f(\gamma X) \mod (E(X))$ (this is Lemma 3.4). The second idea, due to Parvaresh and Vardy [85], is the following. We first obtain the bivariate polynomial (over an appropriate extension field) $T(Y, Z) \equiv Q(X, Y, Z) \mod (E(X))$. Note that by our first idea, we are looking for solutions on the curve $Z = Y^q$ (Y corresponds to $f(X)$ and Z corresponds to $f(\gamma X)$ in the extension field). The crux of the argument is to show that all the polynomials $f(X)$ that need to be output correspond to (in the extension field) some root of the equation $T(Y, Y^q) = 0$. See Section 3.4.3 for the details.

As was mentioned earlier, the extension of the $m = s = 2$ case to the general $m = s > 2$ case is fairly straightforward (and is presented in part as Lemma 3.6). To go from $s = m$ to any $s \leqslant m$ requires another simple idea: We will reduce the problem of list decoding folded Reed-Solomon code with folding parameter m to the problem of list decoding folded Reed-Solomon code with folding parameter s. We then use the algorithm outlined in the previous paragraph for the folded Reed-Solomon code with folding parameter s. A careful tracking of the agreement parameter in the reduction, brings down the final agreement fraction (that is required for the original folded Reed-Solomon code with folding parameter m) from $m(1+\delta)\sqrt[m+1]{R^m}$ (which can be obtained without the reduction) to $(1+\delta)\left(\frac{m}{m-s+1}\right)\sqrt[s+1]{R^s}$. This reduction is presented in detail in Section 3.4 for the $s = 2$ case. The generalization to any $s \leqslant m$ is presented in Section 3.5.

3.4 Trivariate Interpolation Based Decoding

As mentioned in the previous section, the list-decoding algorithm for RS codes from [97, 63] is based on bivariate interpolation. The key factor driving the agreement parameter t needed for the decoding to be successful was the ($(1, k)$-weighted) degree D of the interpolated bivariate polynomial. Our quest for an improved algorithm for folded RS codes will be based on trying to lower this degree D by using more degrees of freedom in the interpolation. Specifically, we will try to use *trivariate interpolation* of a polynomial $Q(X, Y_1, Y_2)$ through n points in $\mathbb{F}_q^3$. This enables us to perform the interpolation with D in $O((k^2n)^{1/3})$, which is much smaller than the $\Theta(\sqrt{kn})$ bound for bivariate interpolation. In principle, this could lead to an algorithm that works for agreement fraction $R^{2/3}$ instead of $R^{1/2}$. Of course, this is a somewhat simplistic hope and additional ideas are needed to make this approach work. We now turn to the task of developing a trivariate interpolation based decoder and proving that it can indeed decode up to a $1 - R^{2/3}$ fraction of errors.

3.4.1 Facts about Trivariate Interpolation

We begin with some basic definitions and facts concerning trivariate polynomials.

Definition 3.2. *For a polynomial $Q(X, Y_1, Y_2) \in \mathbb{F}_q[X, Y_1, Y_2]$, its $(1, k, k)$-weighted degree is defined to be the maximum value of $\ell + kj_1 + kj_2$ taken over all monomials $X^\ell Y_1^{j_1} Y_2^{j_2}$ that occur with a nonzero coefficient in $Q(X, Y_1, Y_2)$. If $Q(X, Y_1, Y_2) \equiv 0$ then its $(1, k, k)$-weighted degree is 0.*

Definition 3.3 (Multiplicity of zeroes). *A polynomial $Q(X, Y_1, Y_2)$ over $\mathbb{F}_q$ is said to have a zero of multiplicity $r \geqslant 1$ at a point $(\alpha, \beta_1, \beta_2) \in \mathbb{F}_q^3$ if $Q(X + \alpha, Y_1 + \beta_1, Y_2 + \beta_2)$ has no monomial of degree less than r with a nonzero coefficient. (The degree of the monomial $X^i Y_1^{j_1} Y_2^{j_2}$ equals $i + j_1 + j_2$.)*

Lemma 3.1. *Let $\{(\alpha_i, y_{i1}, y_{i2})\}_{i=1}^n$ be an arbitrary set of n triples from $\mathbb{F}_q^3$. Let $Q(X, Y_1, Y_2) \in \mathbb{F}_q[X, Y_1, Y_2]$ be a nonzero polynomial of $(1, k, k)$-weighted degree at most D that has a zero of multiplicity r at $(\alpha_i, y_{i1}, y_{i2})$ for every $i \in [n]$. Let $f(X), g(X)$ be polynomials of degree at most k such that for at least $t > D/r$ values of $i \in [n]$, we have $f(\alpha_i) = y_{i1}$* and *$g(\alpha_i) = y_{i2}$. Then, $Q(X, f(X), g(X)) \equiv 0$.*

Proof. If we define $R(X) = Q(X, f(X), g(X))$, then $R(X)$ is a univariate polynomial of degree at most D, and for every $i \in [n]$ for which $f(\alpha_i) = y_{i1}$ *and* $g(\alpha_i) = y_{i2}$, $(X - \alpha_i)^r$ divides $R(X)$. Therefore if $rt > D$, then $R(X)$ has more roots (counting multiplicities) than its degree, and so it must be the zero polynomial. □

Lemma 3.2. *Given an arbitrary set of n triples $\{(\alpha_i, y_{i1}, y_{i2})\}_{i=1}^n$ from $\mathbb{F}_q^3$ and an integer parameter $r \geqslant 1$, there exists a nonzero polynomial $Q(X, Y_1, Y_2)$ over $\mathbb{F}_q$ of $(1, k, k)$-weighted degree at most D such that $Q(X, Y_1, Y_2)$ has a zero of multiplicity r at $(\alpha_i, y_{i1}, y_{i2})$ for all $i \in [n]$, provided $\frac{D^3}{6k^2} > n\binom{r+2}{3}$. Moreover, we can find such a $Q(X, Y_1, Y_2)$ in time polynomial in n, r by solving a system of homogeneous linear equations over $\mathbb{F}_q$.*

Proof. We begin with the following claims. (i) The condition that $Q(X, Y_1, Y_2)$ has a zero of multiplicity r at $(\alpha_i, y_{i1}, y_{i2})$ for all $i \in [n]$ amounts to $n\binom{r+2}{3}$ homogeneous linear conditions in the coefficients of Q; and (ii) The number of monomials in $Q(X, Y_1, Y_2)$ equals the number, say $N_3(k, D)$, of triples (i, j_1, j_2) of nonnegative integers that obey

$i + kj_1 + kj_2 \leqslant D$ is at least $\frac{D^3}{6k^2}$. Hence, if $\frac{D^3}{6k^2} > n\binom{r+2}{3}$, then the number of unknowns exceeds the number of equations, and we are guaranteed a nonzero solution.

To complete the proof, we prove the two claims. To prove the first claim, it suffices to show that for any arbitrary tuple (α, β, γ), the condition that $Q(X,Y,Z)$ has multiplicity r at point (α,β,γ) amounts to $\binom{r+2}{3}$ many homogeneous linear constraints. By the definition of multiplicities of roots, this amounts to setting the coefficients of all monomials of total degree r in $Q(X+\alpha, Y+\beta, Z+\gamma)$ to be zero. In particular, the coefficient of the monomial $X^{i_1}Y^{i_2}Z^{i_3}$ is given by $\sum_{i_1' \geqslant i_1} \sum_{i_2' \geqslant i_2} \sum_{i_3' \geqslant i_3} \binom{i_1'}{i_1}\binom{i_2'}{i_2}\binom{i_3'}{i_3} q_{i_1',i_2'.i_3'} \alpha^{i_1'-i_1}\beta^{i_2'-i_2}\gamma^{i_3'-i_3}$, where $q_{i_1',i_2',i_3'}$ is the coefficient of $X^{i_1'}Y^{i_2'}Z^{i_3'}$ in $Q(X,Y,Z)$. Thus, the condition on multiplicities on roots of $Q(X,Y,Z)$ at (α,β,γ) follows if the following is satisfied by for every triple (i_1,i_2,i_3) such that $i_1+i_2+i_3 \leqslant r$:

$$\sum_{i_1' \geqslant i_1} \sum_{i_2' \geqslant i_2} \sum_{i_3' \geqslant i_3} \binom{i_1'}{i_1}\binom{i_2'}{i_2}\binom{i_3'}{i_3} q_{i_1',i_2',i_3'} \alpha^{i_1'-i_1}\beta^{i_2'-i_2}\gamma^{i_3'-i_3} = 0.$$

The claim follows by noting that the number of integral solutions to $i_1+i_2+i_3 \leqslant r$ is $\binom{r+2}{3}$.

To prove the second claim, following [85], we will first show that the number $N_3(k,D)$ is at least as large as the volume of the 3-dimensional region $\mathcal{P} = \{x + ky_1 + ky_2 \leqslant D \mid x, y_1, y_2 \geqslant 0\} \subset \mathbb{R}^3$. Consider the correspondence between monomials in $\mathbb{F}_q[X,Y,Z]$ and unit cubes in $\mathbb{R}^3$: $X^{i_1}Y^{i_2}Z^{i_3} \to \mathcal{C}(i_1,i_2,i_3)$, where $\mathcal{C}(i_1,i_2,i_3) = [i_1, i_1+1) \times [i_2, i_2+1) \times [i_3, i_3+1)$. Note that the volume of each such cube is 1. Thus, $N_3(k,D)$ is the volume of the union of cubes $\mathcal{C}(i_1,i_2,i_3)$ for positive integers i_1,i_2,i_3 such that $i_1 + ki_2 + ki_3 \leqslant D$: let $\mathcal{U}$ denote this union. It is easy to see that $\mathcal{P} \subset \mathcal{U}$. To complete the claim we will show that the volume of $\mathcal{P}$ equals $\frac{D^3}{6k^2}$. Indeed the volume of $\mathcal{P}$ is

$$\begin{aligned}\int_0^D \int_0^{(D-x)/k} \int_0^{(D-x)/k-y_1} dy_2\, dy_1\, dx &= \int_0^D \int_0^{(D-x)/k} \left(\frac{D-x}{k} - y_1\right) dy_1\, dx \\ &= \int_0^D \frac{(D-x)^2}{2k^2}\, dx \\ &= \frac{1}{2k^2}\int_0^D z^2\, dz \\ &= \frac{D^3}{6k^2},\end{aligned}$$

where the third equality follows by substituting $z = D - x$. □

3.4.2 *Using Trivariate Interpolation for Folded RS Codes*

Let us now see how trivariate interpolation can be used in the context of decoding the folded RS code $C' = \mathrm{FRS}_{\mathbb{F}_q,\gamma,m,k}$ of block length $N = (q-1)/m$. (Throughout this section, we denote $n = q - 1$.) Given a received word $\mathbf{z} \in (\mathbb{F}_q^m)^N$ for C' that needs to be list decoded, we define $\mathbf{y} \in \mathbb{F}_q^n$ to be the corresponding "unfolded" received word. (Formally, let the j'th symbol of $\mathbf{z}$ be $(z_{j,0}, \ldots, z_{j,m-1})$ for $0 \leqslant j < N$. Then $\mathbf{y}$ is defined by $y_{jm+l} = z_{j,l}$ for $0 \leqslant j < N$ and $0 \leqslant l < m$.)

Suppose that $f(X)$ is a polynomial whose encoding agrees with $\mathbf{z}$ on at least t locations (note that the agreement is on symbols from $\mathbb{F}_q^m$). Then, here is an obvious but important observation:

> For at least $t(m-1)$ values of i, $0 \leqslant i < n$, *both* the equalities $f(\gamma^i) = y_i$ and $f(\gamma^{i+1}) = y_{i+1}$ hold.

Define the notation $g(X) = f(\gamma X)$. Therefore, if we consider the n triples $(\gamma^i, y_i, y_{i+1}) \in \mathbb{F}_q^3$ for $i = 0, 1, \ldots, n-1$ (with the convention $y_n = y_0$), then for at least $t(m-1)$ triples, we have $f(\gamma^i) = y_i$ *and* $g(\gamma^i) = y_{i+1}$. This suggests that interpolating a polynomial $Q(X, Y_1, Y_2)$ through these n triples and employing Lemma 3.1, we can hope that $f(X)$ will satisfy $Q(X, f(X), f(\gamma X)) = 0$, and then somehow use this to find $f(X)$. We formalize this in the following lemma. The proof follows immediately from the preceding discussion and Lemma 3.1.

Lemma 3.3. *Let $\mathbf{z} \in (\mathbb{F}_q^m)^N$ and let $\mathbf{y} \in \mathbb{F}_q^n$ be the unfolded version of $\mathbf{z}$. Let $Q(X, Y_1, Y_2)$ be any nonzero polynomial over $\mathbb{F}_q$ of $(1, k, k)$-weighted degree at D that has a zero of multiplicity r at (γ^i, y_i, y_{i+1}) for $i = 0, 1, \ldots, n-1$. Let t be an integer such that $t > \frac{D}{(m-1)r}$. Then every polynomial $f(X) \in \mathbb{F}_q[X]$ of degree at most k whose encoding according to $\mathrm{FRS}_{\mathbb{F}_q,\gamma,m,k}$ agrees with $\mathbf{z}$ on at least t locations satisfies $Q(X, f(X), f(\gamma X)) \equiv 0$.*

Lemmas 3.2 and 3.3 motivate the following approach to list decoding the folded RS code $\mathrm{FRS}_{\mathbb{F}_q,\gamma,m,k}$. Here $\mathbf{z} \in (\mathbb{F}_q^m)^N$ is the received word and $\mathbf{y} = (y_0, y_1, \ldots, y_{n-1}) \in \mathbb{F}_q^n$ is its unfolded version. The algorithm uses an integer multiplicity parameter $r \geqslant 1$, and is intended to work for an agreement parameter $1 \leqslant t \leqslant N$.

Algorithm Trivariate-FRS-decoder:

Step 1 (Trivariate Interpolation) Define the degree parameter

$$D = \lfloor \sqrt[3]{k^2 n r (r+1)(r+2)} \rfloor + 1 \,. \tag{3.1}$$

Interpolate a nonzero polynomial $Q(X, Y_1, Y_2)$ with coefficients from $\mathbb{F}_q$ with the following two properties: (i) Q has $(1, k, k)$-weighted degree at most D, and (ii) Q has a zero of multiplicity r at (γ^i, y_i, y_{i+1}) for $i = 0, 1, \ldots, n-1$ (where $y_n = y_0$). (Lemma 3.2 guarantees the feasibility of this step as well as its computability in time polynomial in n, r.)

Step 2 (Trivariate "Root-finding") Find a list of all degree $\leqslant k$ polynomials $f(X) \in \mathbb{F}_q[X]$ such that $Q(X, f(X), f(\gamma X)) = 0$. Output those whose encoding agrees with $\mathbf{z}$ on at least t locations.

Ignoring the time complexity of Step 2 for now, we can already claim the following result concerning the error-correction performance of this algorithm.

Theorem 3.1. *The algorithm* Trivariate-FRS-decoder *successfully list decodes the folded Reed-Solomon code* $\mathrm{FRS}_{\mathbb{F}_q,\gamma,m,k}$ *up to a radius of* $N - \left\lfloor N \frac{m}{m-1} \sqrt[3]{\frac{k^2}{n^2}\left(1+\frac{1}{r}\right)\left(1+\frac{2}{r}\right)} \right\rfloor - 2$.

Proof. By Lemma 3.3, we know that any $f(X)$ whose encoding agrees with $\mathbf{z}$ on t or more locations will be output in Step 2, provided $t > \frac{D}{(m-1)r}$. For the choice of D in (3.1), this condition is met for the choice $t = 1 + \left\lfloor \sqrt[3]{\frac{k^2 n}{(m-1)^3}\left(1+\frac{1}{r}\right)\left(1+\frac{2}{r}\right)} + \frac{1}{(m-1)r} \right\rfloor$. Indeed, we have

$$\begin{aligned}
\frac{D}{(m-1)r} &\leqslant \frac{1}{(m-1)r}\left(\sqrt[3]{k^2 nr(r+1)(r+2)} + 1\right) \\
&= \frac{1}{m-1}\sqrt[3]{k^2 n\left(1+\frac{1}{r}\right)\left(1+\frac{2}{r}\right)} + \frac{1}{(m-1)r} \\
&< 1 + \left\lfloor \frac{1}{m-1}\sqrt[3]{k^2 n\left(1+\frac{1}{r}\right)\left(1+\frac{2}{r}\right)} + \frac{1}{(m-1)r} \right\rfloor \\
&= t,
\end{aligned}$$

where the first inequality follows from (3.1) and the fact that for any real $x \geqslant 0$, $\lfloor x \rfloor \leqslant x$ while the second inequality follows from the fact that for any real $x \geqslant 0$, $x < \lfloor x \rfloor + 1$. The decoding radius is equal to $N - t$, and recalling that $n = mN$, we get bound claimed in the lemma. $\square$

The rate of the folded Reed-Solomon code is $R = (k+1)/n > k/n$, and so the fraction of errors corrected (for large enough r) is $1 - \frac{m}{m-1}R^{2/3}$. Letting the parameter m grow, we can approach a decoding radius of $1 - R^{2/3}$.

3.4.3 Root-finding Step

In light of the above discussion, the only missing piece in our decoding algorithm is an efficient way to solve the following trivariate "root-finding" problem:

> Given a nonzero polynomial $Q(X, Y_1, Y_2)$ with coefficients from a finite field $\mathbb{F}_q$, a primitive element γ of the field $\mathbb{F}_q$, and an integer parameter $k < q-1$, find the list of all polynomials $f(X)$ of degree at most k such that $Q(X, f(X), f(\gamma X)) \equiv 0$.

The following simple algebraic lemma is at the heart of our solution to this problem.

Lemma 3.4. *Let γ be a primitive element that generates $\mathbb{F}_q^*$. Then we have the following two facts:*

1. *The polynomial* $E(X) \overset{\text{def}}{=} X^{q-1} - \gamma$ *is irreducible over* $\mathbb{F}_q$.

2. *Every polynomial* $f(X) \in \mathbb{F}_q[X]$ *of degree less than* $q-1$ *satisfies* $f(\gamma X) = f(X)^q$ mod $E(X)$.

Proof. The fact that $E(X) = X^{q-1} - \gamma$ is irreducible over $\mathbb{F}_q$ follows from a known, precise characterization of all irreducible binomials, i.e., polynomials of the form $X^a - c$, see for instance [77, Chap. 3, Sec. 5]. For completeness, and since this is an easy special case, we now prove this fact. Suppose $E(X)$ is not irreducible and some irreducible polynomial $f(X) \in \mathbb{F}_q[X]$ of degree b, $1 \leqslant b < q-1$, divides it. Let ζ be a root of $f(X)$ in the extension field $\mathbb{F}_{q^b}$. We then have $\zeta^{q^b-1} = 1$. Also, $f(\zeta) = 0$ implies $E(\zeta) = 0$, which implies $\zeta^{q-1} = \gamma$. These equations together imply $\gamma^{\frac{q^b-1}{q-1}} = 1$. Now, γ is primitive in $\mathbb{F}_q$, so that $\gamma^a = 1$ iff a is divisible by $(q-1)$. We conclude that $q-1$ must divide $\frac{q^b-1}{q-1} = 1+q+q^2+\cdots+q^{b-1}$. This is, however, impossible since $1+q+q^2+\cdots+q^{b-1} \equiv b \pmod{(q-1)}$ and $0 < b < q-1$. This contradiction proves that $E(X)$ has no such factor of degree less than $q-1$, and is therefore irreducible.

For the second part, we have the simple but useful identity $f(X)^q = f(X^q)$ that holds for all polynomials in $\mathbb{F}_q[X]$. Therefore, $f(X)^q - f(\gamma X) = f(X^q) - f(\gamma X)$. Since $X^q = \gamma X$ implies $f(X^q) = f(\gamma X)$, $f(X^q) - f(\gamma X)$ is divisible by $X^q - \gamma X$, and thus also by $X^{q-1} - \gamma$. Hence $f(X)^q \equiv f(\gamma X) \pmod{E(X)}$ which implies that $f(X)^q$ mod $E(X) = f(\gamma X)$ since the degree of $f(\gamma X)$ is less than $q-1$. $\square$

Armed with this lemma, we are ready to tackle the trivariate root-finding problem.

Lemma 3.5. *There is a deterministic algorithm that on input a finite field* $\mathbb{F}_q$, *a primitive element* γ *of the field* $\mathbb{F}_q$, *a nonzero polynomial* $Q(X, Y_1, Y_2) \in \mathbb{F}_q[X, Y_1, Y_2]$ *of degree less than* q *in* Y_1, *and an integer parameter* $k < q-1$, *outputs a list of all polynomials* $f(X)$ *of degree at most* k *satisfying the condition* $Q(X, f(X), f(\gamma X)) \equiv 0$. *The algorithm has run time polynomial in* q.

Proof. Let $E(X) = X^{q-1} - \gamma$. We know by Lemma 3.4 that $E(X)$ is irreducible. We first divide out the largest power of $E(X)$ that divides $Q(X, Y_1, Y_2)$ to obtain $Q_0(X, Y_1, Y_2)$ where $Q(X, Y_1, Y_2) = E(X)^b Q_0(X, Y_1, Y_2)$ for some $b \geqslant 0$ and $E(X)$ does not divide $Q_0(X, Y_1, Y_2)$. Note that as $E(X)$ is irreducible, $f(X)$ does not divide $E(X)$. Thus, if $f(X)$ satisfies $Q(X, f(X), f(\gamma X)) \equiv 0$, then $Q_0(X, f(X), f(\gamma X)) \equiv 0$ as well, so we will work with Q_0 instead of Q. Let us view $Q_0(X, Y_1, Y_2)$ as a polynomial $T_0(Y_1, Y_2)$ with coefficients from $\mathbb{F}_q[X]$. Further, reduce each of the coefficients modulo $E(X)$ to get a polynomial $T(Y_1, Y_2)$ with coefficients from the extension field $\mathbb{F}_{q^{q-1}}$ (which is isomorphic to $\mathbb{F}_q[X]/(E(X))$ as $E(X)$ is irreducible over $\mathbb{F}_q$). We note that $T(Y_1, Y_2)$ is a nonzero polynomial since $Q_0(X, Y_1, Y_2)$ is not divisible by $E(X)$.

In view of Lemma 3.4, it suffices to find degree $\leqslant k$ polynomials $f(X)$ satisfying $Q_0(X, f(X), f(X)^q) \pmod{E(X)} \equiv 0$. In turn, this means it suffices to find elements

$\Gamma \in \mathbb{F}_{q^{q-1}}$ satisfying $T(\Gamma, \Gamma^q) = 0$. If we define the univariate polynomial $R(Y_1) \overset{\text{def}}{=} T(Y_1, Y_1^q)$, this is equivalent to finding all $\Gamma \in \mathbb{F}_{q^{q-1}}$ such that $R(\Gamma) = 0$, or in other words the roots in $\mathbb{F}_{q^{q-1}}$ of $R(Y_1)$.

Now $R(Y_1)$ is a nonzero polynomial since $R(Y_1) = 0$ iff $Y_2 - Y_1^q$ divides $T(Y_1, Y_2)$, and this cannot happen as $T(Y_1, Y_2)$ has degree less than q in Y_1. The degree of $R(Y_1)$ is at most dq where d is the total degree of $Q(X, Y_1, Y_2)$. The characteristic of $\mathbb{F}_{q^{q-1}}$ is at most q, and its degree over the base field is at most $q \lg q$. Therefore, by Theorem 2.4 we can find all roots of $R(Y_1)$ by a deterministic algorithm running in time polynomial in d, q. Each of the roots will be a polynomial in $\mathbb{F}_q[X]$ of degree less than $q-1$. Once we find all the roots, we prune the list and only output those roots $f(X)$ that have degree at most k and satisfy $Q_0(X, f(X), f(\gamma X)) = 0$. □

With this, we have a polynomial time implementation of the algorithm Trivariate-FRS-decoder. There is the technicality that the degree of $Q(X, Y_1, Y_2)$ in Y_1 should be less than q. This degree is at most D/k, which by the choice of D in (3.1) is at most $(r+3)\sqrt[3]{n/k} < (r+3)q^{1/3}$. For a fixed r and growing q, the degree is much smaller than q. (In fact, for constant rate codes, the degree is a constant independent of n.) By letting m, r grow in Theorem 3.1, and recalling that the running time is polynomial in n, r, we can conclude the following main result of this section.

Theorem 3.2. *For every $\delta > 0$ and R, $0 < R < 1$, there is a family of m-folded Reed-Solomon codes for m in $O(1/\delta)$ that have rate at least R and that can be list decoded up to a fraction $1 - (1+\delta)R^{2/3}$ of errors in time polynomial in the block length and $1/\delta$.*

Remark 3.2 (Optimality of degree q of relation between $f(X)$ and $f(\gamma X)$). *Let $\mathbb{F}_{q^{q-1}}$ be the extension field $\mathbb{F}_q[X]/(E(X))$ — its elements are in one-one correspondence with polynomials of degree less than $q-1$ over $\mathbb{F}_q$. Let $\Gamma : \mathbb{F}_{q^{q-1}} \to \mathbb{F}_{q^{q-1}}$ be such that for every $f(X) \in \mathbb{F}_{q^{q-1}}$, $\Gamma(f(X)) = f(G(X))$ for some polynomial G over $\mathbb{F}_q$. (In the above, we had $\Gamma(f(X)) = f(X)^q \mod (E(X))$ and $G(X) = \gamma X$; as a polynomial over $\mathbb{F}_{q^{q-1}}$, $\Gamma(Z) = Z^q$, and hence had degree q.) Any such map Γ is an $\mathbb{F}_q$-linear function on $\mathbb{F}_{q^{q-1}}$, and is therefore a* linearized *polynomial, cf. [77, Chap. 3, Sec. 4], which has only terms with exponents that are powers of q (including $q^0 = 1$). It turns out that for our purposes Γ cannot have degree 1, and so it must have degree at least q.*

3.5 Codes Approaching List Decoding Capacity

Given that trivariate interpolation improved the decoding radius achievable with rate R from $1 - R^{1/2}$ to $1 - R^{2/3}$, it is natural to attempt to use higher order interpolation to improve the decoding radius further. In this section, we discuss the quite straightforward technical changes needed for such a generalization.

Consider again the m-folded RS code $C' = \mathrm{FRS}_{\mathbb{F}_q,\gamma,m,k}$. Let s be an integer in the range $1 \leqslant s \leqslant m$. We will develop a decoding algorithm based on interpolating an $(s+1)$-variate polynomial $Q(X, Y_1, Y_2, \ldots, Y_s)$. The definitions of the $(1, k, k, \ldots, k)$-weighted degree (with k repeated s times) of Q and the multiplicity at a point $(\alpha, \beta_1, \beta_2, \ldots, \beta_s) \in \mathbb{F}_q^{s+1}$ are straightforward extensions of Definitions 3.2 and 3.3.

As before let $\mathbf{y} = (y_0, y_1, \ldots, y_{n-1})$ be the unfolded version of the received word $\mathbf{z} \in (\mathbb{F}_q^m)^N$ of the folded RS code that needs to be decoded. For convenience, define $y_j = y_{j \bmod n}$ for $j \geqslant n$. Following algorithm Trivariate-FRS-decoder, for suitable integer parameters D, r, the interpolation phase of the $(s+1)$-variate FRS decoder will fit a nonzero polynomial $Q(X, Y_1, \ldots, Y_s)$ with the following properties:

1. It has $(1, k, k, \ldots, k)$-weighted degree at most D

2. It has a zero of multiplicity r at $(\gamma^i, y_i, y_{i+1}, \ldots, y_{i+s-1})$ for $i = 0, 1, \ldots, n-1$.

The following is a straightforward generalization of Lemmas 3.2 and 3.3.

Lemma 3.6. *(a) Provided $\frac{D^{s+1}}{(s+1)!k^s} > n\binom{r+s}{s+1}$, a nonzero polynomial $Q(X, Y_1, \ldots, Y_s)$ with the following properties exists. $Q(X, Y_1, \ldots, Y_s)$ has $(1, k, \ldots, k)$ weighted degree at most D and has roots with multiplicity r at $(\gamma^i, y_i, y_{i+1}, \ldots, y_{i+s-1})$ for every $i \in \{0, \ldots, n-1\}$. Moreover such a $Q(X, Y_1, \ldots, Y_s)$ can be found in time polynomial in n, r^s and D^{s+1}/k^s.*

(b) Let t be an integer such that $t > \frac{D}{(m-s+1)r}$. Then every polynomial $f(X) \in \mathbb{F}_q[X]$ of degree at most k whose encoding according to $\mathrm{FRS}_{\mathbb{F}_q,\gamma,m,k}$ agrees with the received word $\mathbf{z}$ on at least t locations satisfies $Q(X, f(X), f(\gamma X), \ldots, f(\gamma^{s-1}X)) \equiv 0$.

Proof. The first part follows from (i) a simple lower bound on the number of monomials $X^a Y_1^{b_1} \cdots Y_s^{b_s}$ with $a + k(b_1 + b_2 + \cdots + b_s) \leqslant D$, which gives the number of coefficients of $Q(X, Y_1, \ldots, Y_s)$, and (ii) an estimation of the number of $(s+1)$-variate monomials of total degree less than r, which gives the number of interpolation conditions per $(s+1)$-tuple. We now briefly justify these claims. By a generalization of the argument in Lemma 3.2, one can lower bound the number of monomials $X^a Y_1^{b_1} \cdots Y_s^{b_s}$ such that $a + k(b_1 + \cdots b_s) \leqslant D$ by the volume of $\mathcal{P}_{s,D} = \{x + ky_1 + ky_2 + \cdots + ky_s \leqslant D | x, y_1, y_2, \ldots y_s \geqslant 0\}$. We will use induction on s to prove that the volume of $\mathcal{P}_{s,D}$ is $\frac{D^{s+1}}{(s+1)!k^s}$. The proof of Lemma 3.2 shows this for $s = 2$. Now assume that the volume of $\mathcal{P}_{s-1,D}$ is exactly $\frac{D^s}{s!k^{s-1}}$. Note that the subset of $\mathcal{P}_{s,D}$ where the value of $y_s = \alpha$ is fixed is exactly $\mathcal{P}_{s-1,D-k\alpha}$ Thus, the volume of $\mathcal{P}_{s,D}$ is exactly

$$\int_0^{D/k} \frac{(D - ky_s)^s}{s!k^{s-1}}\, dy_s = \frac{1}{s!k^s} \int_0^D z^s\, dz = \frac{D^{s+1}}{(s+1)!k^s},$$

where the second equality follows by substituting $z = D - ky_s$. Further, a straightforward generalization of the argument in the proof of Lemma 3.2, shows that the condition on the multiplicity of the polynomial $Q(X, Y_1, \ldots, Y_s)$ is satisfied if for every $i \in \{0, \ldots, n-1\}$ and every tuple $(l, j_1, \ldots, j_s)$ such that $l + j_1 + j_2 \cdots + j_s \leqslant r$ the following is 0

$$\sum_{l' \geqslant l} \sum_{j_1' \geqslant j_1} \sum_{j_2' \geqslant j_2} \cdots \sum_{j_s' \geqslant j_s} \binom{l'}{l} \binom{j_1'}{j_1} \binom{j_2'}{j_2} \cdots \binom{j_s'}{j_s} q_{l', j_1', i_2', \ldots, j_s'} \gamma^{i(l'-l)} y_i^{j_1' - g_1} y_{i+1}^{j_2' - j_2} \cdots y_{i+s-1}^{j_s' - j_s},$$

where $q_{l', j_1', j_2', \ldots, j_s'}$ is the coefficient of the monomial $X^{l'} Y_1^{j_1'} \cdots Y_s^{j_s'}$ in $Q(X, Y_1, \ldots, Y_s)$. The number of positive integral solutions for $i + j_1 + j_2 \cdots + j_s \leqslant r$ is exactly $\binom{r+s}{s+1}$. Thus, the total number of constraints is $n\binom{r+s}{s+1}$. Thus, the condition in part (a) of the lemma, implies that the set of homogeneous linear equations have more variables than constraints. Hence, a solution can be found in time polynomial in the number of variables ($\leqslant D^{s+1}/k^s$) and constraints (at most $nr^{O(s)}$).

The second part is similar to the proof of Lemma 3.3. If $f(X)$ has agreement on at least t locations of $\mathbf{z}$, then for at least $t(m-s+1)$ of the $(s+1)$-tuples $(\gamma^i, y_i, y_{i+1}, \ldots, y_{i+s-1})$, we have $f(\gamma^{i+j}) = y_{i+j}$ for $j = 0, 1, \ldots, s-1$. As in Lemma 3.1, we conclude that $R(X) \stackrel{\text{def}}{=} Q(X, f(X), f(\gamma X), \ldots, f(\gamma^{s-1} X))$ has a zero of multiplicity r at γ^i for each such $(s+1)$-tuple. Also, by design $R(X)$ has degree at most D. Hence if $t(m-s+1)r > D$, then $R(X)$ has more zeroes (counting multiplicities) than its degree, and thus $R(X) \equiv 0$. $\square$

Note the lower bound condition on D above is met with the choice

$$D = \left\lfloor \left(k^s n r (r+1) \cdots (r+s)\right)^{1/(s+1)} \right\rfloor + 1 \,. \tag{3.2}$$

The task of finding the list of all degree k polynomials $f(X) \in \mathbb{F}_q[X]$ satisfying $Q(X, f(X), f(\gamma X), \ldots, f(\gamma^{s-1} X)) = 0$ can be solved using ideas similar to the proof of Lemma 3.5. First, by dividing out by $E(X)$ enough times, we can assume that not all coefficients of $Q(X, Y_1, \ldots, Y_s)$, viewed as a polynomial in $Y_1, \ldots, Y_s$ with coefficients in $\mathbb{F}_q[X]$, are divisible by $E(X)$. We can then go modulo $E(X)$ to get a nonzero polynomial $T(Y_1, Y_2, \ldots, Y_s)$ over the extension field $\mathbb{F}_{q^{q-1}} = \mathbb{F}_q[X]/(E(X))$. Now, by Lemma 3.4, we have $f(\gamma^j X) = f(X)^{q^j} \mod E(X)$ for every $j \geqslant 1$. Therefore, the task at hand reduces to the problem of finding all roots $\Gamma \in \mathbb{F}_{q^{q-1}}$ of the polynomial $R(Y_1)$ where $R(Y_1) = T(Y_1, Y_1^q, \ldots, Y_1^{q^{s-1}})$. There is the risk that $R(Y_1)$ is the zero polynomial, but it is easily seen that this cannot happen if the total degree of T is less than q. This will be the case since the total degree is at most D/k, which is at most $(r+s)(n/k)^{1/(s+1)} \ll q$.

The degree of the polynomial $R(Y_1)$ is at most q^s, and therefore all its roots in $\mathbb{F}_{q^{q-1}}$ can be found in $q^{O(s)}$ time (by Theorem 2.4). We conclude that the "root-finding" step can be accomplished in polynomial time.

The algorithm works for agreement $t > \frac{D}{(m-s+1)r}$, which for the choice of D in (3.2) is satisfied if

$$t \geqslant \left(1 + \frac{s}{r}\right) \frac{(k^s n)^{1/(s+1)}}{m-s+1} + 2 \,.$$

Indeed,

$$\begin{aligned}
\frac{D}{(m-s+1)r} &\leqslant \frac{1}{(m-s+1)r}\cdot\left(\sqrt[s+1]{k^s nr(r+1)\cdots(r+s)}+1\right)\\
&\leqslant \frac{1}{(m-s+1)r}\cdot\left((r+s)\sqrt[s+1]{k^s n}+1\right)\\
&= \left(1+\frac{s}{r}\right)\frac{(k^s n)^{1/(s+1)}}{m-s+1}+\frac{1}{r(m-s+1)}\\
&< \left(1+\frac{s}{r}\right)\frac{(k^s n)^{1/(s+1)}}{m-s+1}+2\\
&\leqslant t,
\end{aligned}$$

where the first inequality follows from (3.2) along with the fact that for any real $x \geqslant 0$, $\lfloor x \rfloor \leqslant x$ while the second inequality follows by upper bounding $r+i$ by $r+s$ for every $0 \leqslant i \leqslant s$. We record these observations in the following, which is a multivariate generalization of Theorem 3.1.

Theorem 3.3. *For every integer $m \geqslant 1$ and every s, $1 \leqslant s \leqslant m$, the $(s+1)$-variate FRS decoder successfully list decodes the m-folded Reed-Solomon code $\mathrm{FRS}_{\mathbb{F}_q,\gamma,m,k}$ up to a radius $n/m - t$ as long as the agreement parameter t satisfies*

$$t \geqslant \left(1+\frac{s}{r}\right)\frac{(k^s n)^{1/(s+1)}}{m-s+1}+2\,. \tag{3.3}$$

The algorithm runs in $n^{O(s)}$ time and outputs a list of size at most $|F|^s = (n+1)^s$.

Recalling that the block length of $\mathrm{FRS}_{\mathbb{F}_q,\gamma,m,k}$ is $N = n/m$ and the rate is $(k+1)/n$, the above algorithm can decode a fraction of errors approaching

$$1-\left(1+\frac{s}{r}\right)\frac{m}{m-s+1}R^{s/(s+1)} \tag{3.4}$$

using lists of size at most q^s. By picking r, m large enough compared to s, the decoding radius can be made larger than $1-(1+\delta)R^{s/(s+1)}$ for any desired $\delta > 0$. We state this result formally below.

Theorem 3.4. *For every $0 < \delta \leqslant 1$, integer $s \geqslant 1$ and $0 < R < 1$, there is a family of m-folded Reed-Solomon codes for $m \leqslant 4s/\delta$ that have rate at least R and which can be list decoded up to a $1-(1+\delta)R^{s/(s+1)}$ fraction of errors in time $(Nm)^{O(s)}$ and outputs a list of size at most $(Nm)^{O(s)}$ where N is the block length of the code. The alphabet size of the code as a function of the block length N is $(Nm)^{O(m)}$.*

Proof. We first instantiate the parameters r and m in terms of s and δ:

$$r = \frac{3s}{\delta} \qquad m = \frac{(s-1)(3+\delta)}{\delta}\,.$$

Note that as $\delta \leqslant 1$, $m \leqslant 4s/\delta$. With the above choice, we have

$$\left(1+\frac{s}{r}\right)\frac{m}{m-s+1} = \left(1+\frac{\delta}{3}\right)^2 < 1+\delta\,.$$

Together with the bound (3.4) on the decoding radius, we conclude that the $(s+1)$-variate decoding algorithm certainly list decodes up to a fraction $1-(1+\delta)R^{s/(s+1)}$ of errors.

The worst case list size is q^s and the claim on the list size follows by recalling that $q = n+1$ and $N = n/m$. The alphabet size is $q^m = (Nm)^{O(m)}$. The running time has two major components: (1) Interpolating the $s+1$-variate polynomial $Q(\cdot)$, which by Lemma 3.6 is $(nr^s)^{O(1)}$; and (2) Finding all the roots of the interpolated polynomial, which takes $q^{O(s)}$ time. Of the two, the time complexity of the root finding step dominates, which is $(Nm)^{O(s)}$. □

In the limit of large s, the decoding radius approaches the list-decoding capacity $1-R$, leading to our main result.

Theorem 3.5 (Explicit capacity-approaching codes). *For every $0 < R < 1$ and $0 < \varepsilon \leqslant R$, there is a family of folded Reed-Solomon codes that have rate at least R and which can be list decoded up to a $1-R-\varepsilon$ fraction of errors in time (and outputs a list of size at most) $(N/\varepsilon^2)^{O(\varepsilon^{-1}\log(1/R))}$ where N is the block length of the code. The alphabet size of the code as a function of the block length N is $(N/\varepsilon^2)^{O(1/\varepsilon^2)}$.*

Proof. Given ε, R, we will apply Theorem 3.4 with the choice

$$s = \left\lceil \frac{\log(1/R)}{\log(1+\varepsilon)} \right\rceil \quad \text{and} \quad \delta = \frac{\varepsilon(1-R)}{R(1+\varepsilon)}\,. \tag{3.5}$$

Note that as $\varepsilon \leqslant R$, $\delta \leqslant 1$. Thus, the list-decoding radius guaranteed by Theorem 3.4 is at least

$$\begin{aligned} 1-(1+\delta)R^{s/(s+1)} &= 1-R(1+\delta)(1/R)^{1/(s+1)} \\ &\geqslant 1-R(1+\delta)(1+\varepsilon) \text{ (by the choice of } s \text{ in (3.5))} \\ &= 1-(R+\varepsilon) \text{ (using the value of } \delta)\,. \end{aligned}$$

We now turn our attention to the time complexity of the decoding algorithm and the alphabet size of the code. To this end we first claim that m is $O(1/\varepsilon^2)$. First we note that by the choice of s,

$$s \leqslant \frac{2\ln(1/R)}{\ln(1+\varepsilon)} \leqslant \frac{4\ln(1/R)}{\varepsilon},$$

where the second inequality follows from the fact that for $0 < x \leqslant 1$, $\ln(1+x) \geqslant x/2$. Thus, we have

$$m \leqslant \frac{4s}{\delta} = 4s \cdot \frac{R(1+\varepsilon)}{\varepsilon(1-R)} \leqslant 8s \cdot \frac{R}{\varepsilon(1-R)} \leqslant \frac{32}{\varepsilon^2} \cdot \frac{R\ln(1/R)}{1-R} \leqslant \frac{32}{\varepsilon^2}\,,$$

where for the last step we used $\ln(1/R) \leqslant \frac{1}{R} - 1$ for $0 < R \leqslant 1$. The claims on the running time, worst case list size and the alphabet size of the code follow from Theorem 3.4 and the facts that m is $O(1/\varepsilon^2)$ and s is $O(\varepsilon^{-1}\log(1/R))$. $\square$

Remark 3.3 (Upper bound on ε in Theorem 3.5). *A version of Theorem 3.5 can also be proven for $\varepsilon > R$. The reason it is stated for $\varepsilon \leqslant R$, is that we generally think of ε as much smaller than R (this is certainly true when we apply the generalization of Theorem 3.5 (Theorem 3.6) in Chapters 4 and 5). However, if one wants $\varepsilon \geqslant R$, first note that the theorem is trivial for $\varepsilon \geqslant 1 - R$. Then if $R < \varepsilon < 1 - R$, can do a proof similar to the one above. However, in this range $\delta = \frac{\varepsilon(1-R)}{R(1+\varepsilon)}$ can be strictly greater than 1. In such a case we apply Theorem 3.4 with $\delta = 1$ (note that applying Theorem 3.4 with a smaller δ than what we want only increases the decoding radius). This implies that we have $m \leqslant 4s$, in which case both the worst case list and the alphabet size become $(N\log(1/R)/\varepsilon)^{\varepsilon^{-1}\log(1/R)}$.*

Remark 3.4 (Minor improvement to decoding radius). *It is possible to slightly improve the bound of (3.4) to $1 - \left(1 + \frac{s}{r}\right)\left(\frac{mR}{m-s+1}\right)^{s/(s+1)}$ with essentially no effort. The idea is to not use only a fraction $(m-s+1)/m$ of the n $(s+1)$-tuples for interpolation. Specifically, we omit tuples with γ^i for $i \mod m > m - s$. This does not affect the number of $(s+1)$-tuples for which we have agreement (this remains at least $t(m-s+1)$), but the number of interpolation conditions is reduced to $N(m-s+1) = n(m-s+1)/m$. This translates into the stated improvement in list-decoding radius. For clarity of presentation, we simply chose to use all n tuples for interpolation.*

Remark 3.5 (Average list size). *Theorem 3.5 states that the worst case list size (over all possible received words) is polynomial in the block length of the codeword (for fixed R and ε). One might also be interested in what is the* average *list size (over all the possible received words within a distance ρn from some codeword). It is known that for Reed-Solomon codes of rate R the average list size is $\ll 1$ even for ρ close to $1 - R$ [81]. Since folded Reed-Solomon codes are just Reed-Solomon codewords with symbols bundled together, the arguments in [81] extend easily to show that even for folded Reed-Solomon codes, the average list size is $\ll 1$.*

3.6 Extension to List Recovery

We now present a very useful generalization of the list decoding result of Theorem 3.5 to the setting of list recovery. Recall that under the list recovery problem, one is given as input for each codeword position, not just one but a set of several, say ℓ, alphabet symbols. The goal is to find and output all codewords which agree with some element of the input sets for several positions. Codes for which this more general problem can be solved turn out to be extremely valuable as outer codes in concatenated code constructions. In short, this is because one can pass a set of possibilities from decodings of the inner codes and then list recover the outer code with those sets as the input. If we only had a list-decodable code at the outer level, we will be forced to make a unique choice in decoding the inner codes thus losing valuable information.

This is a good time to recall the definition of list recoverable codes (Definition 2.4).

Theorem 3.5 can be generalized to list recover the folded RS codes. Specifically, for a FRS code with parameters as in Section 3.5, for an arbitrary constant $\ell \geqslant 1$, we can (ζ, ℓ)-list recover in polynomial time provided

$$(1-\zeta)N \geqslant \left(1+\frac{s}{r}\right)\frac{\sqrt[s+1]{n\ell k^s}}{m-s+1}, \tag{3.6}$$

where $N = n/m$. We briefly justify this claim. The generalization of the list-decoding algorithm of Section 3.5 is straightforward: instead of one interpolation condition for each symbol of the received word, we just impose $|S_i| \leqslant \ell$ many interpolation conditions for each position $i \in \{1, 2, \ldots, n\}$ (where S_i is the i'th input set in the list recovery instance). The number of interpolation conditions is at most $n\ell$, and so replacing n by $n\ell$ in the bound of Lemma 3.6 guarantees successful decoding[2]. This in turn implies that the condition on the number of agreement of (3.3) generalizes to the one in (3.6).[3] This simple generalization to list recovery is a positive feature of all interpolation based decoding algorithms [97, 63, 85] beginning with the one due to Sudan [97].

Picking $r \gg s$ and $m \gg s$ in (3.6), we get (ζ, ℓ)-list recoverable codes with rate R for $\zeta \leqslant 1 - \left(\ell R^s\right)^{1/(s+1)}$. Now comes the remarkable fact: we can pick a suitable $s \gg \ell$ and perform (ζ, ℓ)-list recovery with $\zeta \leqslant 1 - R - \varepsilon$ which is independent of ℓ ! We state the formal result below (Theorem 3.5 is a special case when $\ell = 1$).

Theorem 3.6. *For every integer $\ell \geqslant 1$, for all R, $0 < R < 1$ and $0 < \varepsilon \leqslant R$, and for every prime p, there is an* explicit *family of folded Reed-Solomon codes over fields of characteristic p that have rate at least R and which are $(1-R-\varepsilon, \ell, L(n))$-list recoverable in polynomial time, where $L(n) = (N/\varepsilon^2)^{O(\varepsilon^{-1}\log(\ell/R))}$. The alphabet size of a code of block length N in the family is $(N/\varepsilon^2)^{O(\varepsilon^{-2}\log \ell/(1-R))}$.*

Proof. (Sketch) Using the exact same arguments as in the proof of Theorem 3.4 to the agreement condition of (3.6), we get that one can list recover in polynomial time as long as $\zeta \leqslant 1-(1+\delta)(\ell R^s)^{1/(s+1)}$, for any $0 < \delta \leqslant 1$. The arguments to obtain an upper bound of $1 - R - \varepsilon$ are similar to the ones employed in the proof of theorem 3.5. However, s needs to be defined in a slightly different manner:

$$s = \left\lceil \frac{\log(\ell/R)}{\log(1+\varepsilon)} \right\rceil .$$

[2]In fact, this extension also works when the average size of the size is at most ℓ, that is $\sum_{i=1}^{n} |S_i| \leqslant \ell n$.

[3]We will also need the condition that $(r+s)(n\ell/k)^{1/(s+1)} < q$. This condition is required to argue that in the "root finding" step, the "final" polynomial $R(Y_1)$ is not the zero polynomial. The condition is met for constant rate codes if $\ell \ll q^s$ (recall that we think of q as growing while r and s are fixed). In all our applications of list recovery for folded Reed-Solomon codes, the parameter ℓ will be a constant, so this is not a concern.

Also this implies that m is $O\left(\frac{\log \ell}{(1-R)\varepsilon^2}\right)$, which implies the claimed bound on the alphabet size of the code as well as $L(n)$. □

We also note that increasing the folding parameter m only helps improve the result (at the cost of a larger alphabet). In particular, we have the following corollary of the theorem above.

Corollary 3.7. *For every integer $\ell \geqslant 1$, for all constants $0 < \varepsilon \leqslant R$, for all R, R'; $0 < R \leqslant R' < 1$, and for every prime p, there is an* explicit *family of folded Reed-Solomon codes, over fields of characteristic p that have rate at least R and which can be $(1 - R - \varepsilon, \ell, L(N))$-list recovered in polynomial time, where for codes of block length N, $L(N) = (N/\varepsilon^2)^{O(\varepsilon^{-1}\log(\ell/R))}$ and the code is defined over alphabet of size $(N/\varepsilon^2)^{O(\varepsilon^{-2}\log \ell/(1-R'))}$.*

Note that one can trivially increase the alphabet of a code by thinking of every symbol as coming from a larger alphabet. However, this trivial transformation decreases the rate of the code. Corollary 3.7 states that for folded Reed-Solomon codes, we can increase the alphabet while retaining the rate and the list recoverability properties. At this point this extra feature is an odd one to state explicitly, but we will need this result in Chapter 4.

Remark 3.6 (Soft Decoding). *The decoding algorithm for folded RS codes from Theorem 3.5 can be further generalized to handle soft information, where for each codeword position i the decoder is given as input a non-negative weight $w_{i,z}$ for each possible alphabet symbol z. The weights $w_{i,z}$ can be used to encode the confidence information concerning the likelihood of the the i'th symbol of the codeword being z. For any $\varepsilon > 0$, for suitable choice of parameters, our codes of rate R over alphabet Σ have a soft decoding algorithm that outputs all codewords $c = \langle c_1, c_2, \ldots, c_N\rangle$ that satisfy*

$$\sum_{i=1}^{N} w_{i,c_i} \geqslant \left((1+\varepsilon)(RN)^s\Big(\sum_{i=1}^{N}\sum_{z\in\Sigma} w_{i,z}^{s+1}\Big)\right)^{1/(s+1)} .$$

For $s = 1$, this soft decoding condition is identical to the one for Reed-Solomon codes in [63].

3.7 Bibliographic Notes and Open Questions

We have solved the qualitative problem of achieving list-decoding capacity over large alphabets. Our work could be improved with some respect to some parameters. The size of the list needed to perform list decoding to a radius that is within ε of capacity grows as $n^{O(1/\varepsilon)}$ where n is the block length of the code. It remains an open question to bring this list size down to a constant independent of n, or even to $f(\varepsilon)n^c$ with an exponent c independent of ε (we recall that the existential random coding arguments work with a list size of $O(1/\varepsilon)$).

These results in this chapter were first reported in [58]. We would like to point out that the presentation in this chapter is somewhat different from the original papers [85, 58] in terms of technical details, organization, as well as chronology. Our description closely follows that of a survey by Guruswami [50]. With the benefit of hindsight, we believe this alternate presentation to be simpler and more self-contained than the description in [58], which used the results of Parvaresh-Vardy as a black-box. Below, we discuss some technical aspects of the original development of this material, in order to shed light on the origins of our work.

Two independent works by Coppersmith and Sudan [27] and Bleichenbacher, Kiayias and Yung [18] considered the variant of RS codes where the message consists of two (or more) independent polynomials over some field $\mathbb{F}_q$, and the encoding consists of the joint evaluation of these polynomials at elements of $\mathbb{F}_q$ (so this defines a code over $\mathbb{F}_q^2$).[4] A naive way to decode these codes, which are also called "interleaved Reed-Solomon codes," would be to recover the two polynomials individually, by running separate instances of the RS decoder. Of course, this gives no gain over the performance of RS codes. The hope in these works was that something can possibly be gained by exploiting that errors in the two polynomials happen at "synchronized" locations. However, these works could not give any improvement over the $1 - \sqrt{R}$ bound known for RS codes for worst-case errors. Nevertheless, for *random errors*, where each error replaces the correct symbol by a uniform random field element, they were able to correct well beyond a fraction $1 - \sqrt{R}$ of errors. In fact, as the order of interleaving (i.e., number of independent polynomials) grows, the radius approaches the optimal value $1 - R$. This model of random errors is not very practical or interesting in a coding-theoretic setting, though the algorithms are interesting from an algebraic viewpoint.

The algorithm of Coppersmith and Sudan bears an intriguing relation to multivariate interpolation. Multivariate interpolation essentially amounts to finding a non-trivial linear dependence among the rows of a certain matrix (that consists of the evaluations of appropriate monomials at the interpolation points). The algorithm in [27], instead finds a non-trivial linear dependence among the *columns* of this same matrix! The positions corresponding to columns not involved in this dependence are erased (they correspond to error locations) and the codeword is recovered from the remaining symbols using erasure decoding.

In [84], Parvaresh and Vardy gave a heuristic decoding algorithm for these interleaved RS codes based on multivariate interpolation. However, the provable performance of these codes coincided with the $1 - \sqrt{R}$ bound for Reed-Solomon codes. The key obstacle in improving this bound was the following: for the case when the messages are pairs $(f(X), g(X))$ of degree k polynomials, two algebraically independent relations were needed to identify both $f(X)$ and $g(X)$. The interpolation method could only provide one such relation in general (of the form $Q(X, f(X), g(X)) = 0$ for a trivariate polynomial $Q(X, Y, Z)$). This still left too much ambiguity in the possible values of $(f(X), g(X))$. (The approach

[4]The resulting code is in fact just a Reed-Solomon code where the evaluation points belong to the subfield $\mathbb{F}_q$ of the extension field over $\mathbb{F}_q$ of degree two.

in [84] was to find several interpolation polynomials, but there was no guarantee that they were not all algebraically dependent.)

Then, in [85], Parvaresh and Vardy put forth the ingenious idea of obtaining the extra algebraic relation essentially "for free" by enforcing it as an *a priori* condition satisfied at the encoder. Specifically, instead of letting the second polynomial $g(X)$ to be an independent degree k polynomial, their insight was to make it correlated with $f(X)$ by a specific algebraic condition, such as $g(X) = f(X)^d \mod E(X)$ for some integer d and an irreducible polynomial $E(X)$ of degree $k+1$.

Then, once we have the interpolation polynomial $Q(X,Y,Z)$, $f(X)$ can be obtained as follows: Reduce the coefficients of $Q(X,Y,Z)$ modulo $E(X)$ to get a polynomial $T(Y,Z)$ with coefficients from $\mathbb{F}_q[X]/(E(X))$ and then find roots of the univariate polynomial $T(Y,Y^d)$. This was the key idea in [85] to improve the $1-\sqrt{R}$ decoding radius for rates less than $1/16$. For rates $R \to 0$, their decoding radius approached $1 - O(R\log(1/R))$.

The modification to using independent polynomials, however, does not come for free. In particular, since one sends at least twice as much information as in the original RS code, there is no way to construct codes with rate more than $1/2$ in the PV scheme. If we use $s \geqslant 2$ correlated polynomials for the encoding, we incur a factor $1/s$ loss in the rate. This proves quite expensive, and as a result the improvements over RS codes offered by these codes are only manifest at very low rates.

The central idea behind our work is to avoid this rate loss by making the correlated polynomial $g(X)$ essentially identical to the first (say $g(X) = f(\gamma X)$). Then the evaluations of $g(X)$ can be inferred as a simple cyclic shift of the evaluations of $f(X)$, so intuitively there is no need to explicitly include those too in the encoding.