Electronic Colloquium on Computational Complexity, Report No. 194 (2025)

Fast list recovery of univariate multiplicity and folded Reed-Solomon codes

Rohan Goyal* Prahladh Harsha† Mrinal Kumar† Ashutosh Shankar†

Abstract

A recent work of Goyal, Harsha, Kumar and Shankar gave nearly linear time algorithms for the list decoding of Folded Reed-Solomon codes (FRS) and univariate multiplicity codes up to list decoding capacity in their natural setting of parameters. A curious aspect of this work was that unlike most list decoding algorithms for codes that also naturally extend to the problem of list recovery, the algorithm in the work of Goyal et al. seemed to be crucially tied to the problem of list decoding. In particular, it wasn't clear if their algorithm could be generalized to solve the problem of list recovery FRS and univariate multiplicity codes in near linear time.

In this work, we address this question and design $\widetilde{O}(n)$-time algorithms for list recovery of Folded Reed-Solomon codes and univariate Multiplicity codes up to capacity, where n is the blocklength of the code. For our proof, we build upon the lattice based ideas crucially used by Goyal et al. with one additional technical ingredient - we show the construction of appropriately structured lattices over the univariate polynomial ring that *capture* the list recovery problem for these codes.

*Massachusetts Institute of Technology, Cambridge, MA, United States, Email: rohan_g@mit.edu. Part of this work was done when the first and second author were visiting the *Simons Institute for the Theory of Computing*, Berkeley. Supported by (Yael Tauman Kalai's) grant from Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-25-C-0300. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA).

†Tata Institute of Fundamental Research, Mumbai, India. Email: {prahladh, mrinal, ashutosh.shankar}@tifr.res.in. Research supported by the Department of Atomic Energy, Government of India, under project number RTI400112 and partially supported by Google Research Awards, SERB grant and Premji Invest.

ISSN 1433-8092

1 Introduction

An error-correcting code (or simply a code) $\mathcal{C}$ of block length n over an alphabet Σ is said to be (ρ, L) list decodable if a Hamming ball of radius ρn around any vector in Σ^n contains at most L codewords from $\mathcal{C}$. When ρ is less than half the minimum distance of the code (denoted by δ), the number of such codewords is at most one, and we are in the so-called unique decoding regime. The notion of list decoding is of great interest both from a combinatorial perspective, where the goal is to understand the tradeoffs between various parameters of the code like ρ, L, δ and rate as well as from an algorithmic perspective, where the goal is to design efficient algorithms for list decoding the code $\mathcal{C}$ for larger and larger ρ. In addition to its considerable inherent interest, list decodability of codes is also of interest due to deep and extremely fruitful connections to other notions in complexity theory, especially in pseudorandomness [Vad12]. While list decoding was defined in the 1950s by Elias, it is fair to say that the research in this area really took off in the mid-90s following the work of Sudan [Sud97] who showed that Reed-Solomon codes can be efficiently list decoded far beyond the unique decoding regime. The subsequent two decades have witnessed significant progress in our understanding of list decoding (cf., [GS99, GR08, GW13, Kop14, KRSW23, Sri25, CZ25, JMST25, LMS25]) and we now know explicit codes that achieve list decoding capacity with constant list size, and modulo some finer details, this is essentially the best that one can hope for! In fact, we also know that for some of these codes, the list decoding algorithms can be assumed to run in nearly linear time! [GHKS24, AHS26, ST25].

Our focus in this paper is on the question of list recovery, a notion closely related to that of list decoding that we now define. A code $\mathcal{C} \subseteq \Sigma^n$ is said to be (ρ, ℓ, L) list recoverable if for all subsets $E_1, E_2, \ldots, E_n \subseteq \Sigma$ with $|E_i| \leq \ell$, there are at most L codewords $c \in \mathcal{C}$ such that $c_i \notin E_i$ for at most ρn coordinates $i \in [n]$. Clearly, for $\ell = 1$, this is precisely the notion of list decoding. For a detailed exposition on recent works on list-recovery, we refer to the survey by Resch and Venkitesh [RV25]. In addition to coding theoretic motivations due to its close connection to list decoding, list recovery has found numerous other applications, for instance to the construction of pseudorandom objects like extractors, expanders and condensers [GUV09, KT22], to cryptography [GL89] and to design of algorithms for problems like group testing, compressed sensing & heavy hitters. We refer the interested reader to detailed expositions of some of these applications in the excellent lecture notes by Mary Wootters [Woo19].

A curious feature of most of the list decoding algorithms in literature is that they also extend to the problem of list recovery. This includes the algorithms of Sudan [Sud97] and Guruswami-Sudan [GS99] for list decoding of Reed-Solomon codes up to the Johnson radius, the algorithms for list decoding Folded Reed-Solomon (FRS) codes, multiplicity codes (both univariate and multivariate versions) [GR08, GW13, Kop14, KRSW23, BHKS24, BCDZ25] up to capacity and the recent results on expander based codes [JMST25, ST25, JS25] that approach list-decoding capacity.

Yet another family of problems of great interest in the list decoding/recovery setting is to construct codes that can be list decoded/recovered in nearly linear time. This line of work includes results that rely on careful construction of codes that have such super-fast decoding algorithms as well as the design of such algorithms for many known and natural families

of codes. A notable milestone in this line of work is the result of Alekhnovich [Ale05] that gave near linear time algorithm for list decoding (and recovery) of Reed-Solomon codes up to the Johnson radius. Alekhnovich's algorithm is essentially a near-linear time implementation of the Guruswami-Sudan algorithm, and introduces and studies some deep connections between the polynomial method-based list decoding algorithms for Reed-Solomon codes in [Sud97, GS99] and appropriate lattices over the univariate polynomial ring. In a recent work, Goyal, Harsha, Kumar and Shankar [GHKS24] built upon the ideas of Alekhnovich to show that Folded Reed-Solomon codes and univariate multiplicity codes codes (which were known to achieve list decoding capacity with constant list size [GW13, Kop14, KRSW23, Tam24]) can be list decoded up to capacity in nearly linear time. In addition to the lattice-based ideas, this work also gave near-linear time algorithms for solving families of differential equations and functional equations that naturally appear in the context of list decoding and list recovery of FRS and univariate multiplicity codes.

Given the fact that for most codes, list decoding algorithms also immediately extend to list recovery, it came as a bit of a surprise that the results in [GHKS24] did not appear to extend to the list recovery problem immediately. Even more surprising was the fact that this already seemed to be the case when the the list size at each coordinate (the parameter ℓ in the definition of list recovery) was equal to 2. In this work, we build upon the techniques of [GHKS24] to show that the list recovery problem for FRS codes and univariate multiplicity codes can indeed be solved in nearly linear time. More formally, we have the following theorem.

Theorem 1.1 (Main result). *For every $\varepsilon > 0, \ell \in \mathbb{N}$, there is an $s_0 \in \mathbb{N}$ such that for all $s > s_0$, degree parameter k, block length n and field $\mathbb{F}$ of characteristic zero or greater than k, the following is true.*

There is a randomized algorithm that when given as input sets $S \subseteq \mathbb{F}$ and $E_\alpha \subseteq \mathbb{F}^s$ with $|E_\alpha| \leq \ell$ for every $\alpha \in S$ and $|S| = n$, runs in time $O\left(n \operatorname{poly}(s, \ell, \log n, \log |\mathbb{F}|, (\ell/\varepsilon)^{\frac{1+\log \ell}{\varepsilon}})\right)$ and with high probability outputs the set of all univariate polynomials $f(X) \in \mathbb{F}[X]$ of degree less than k such that for at least $(1 - k/sn - \varepsilon)$ fraction of $\alpha \in S$,

$$(f(\alpha), f^{(1)}(\alpha), \ldots, f^{(s-1)}(\alpha)) \in E_\alpha \,.$$

Remark 1.2. *Choosing s_0 as $\theta(\ell/\varepsilon^2)$ suffices in the above statement.* ⌟

Remark 1.3. *Even though we state and prove the main theorem for univariate multiplicity codes here, our algorithm extends to Folded Reed-Solomon codes as it is with some cosmetic changes.* ⌟

1.1 An overview of the proof

We now give an overview of our proof. We start by setting up some notation. For this overview, we confine ourselves to the list recovery problem for univariate multiplicity codes.

We have a set $S \subseteq \mathbb{F}$ of size n and the input to the list recovery algorithm is data of the form $(\alpha, \boldsymbol{\beta}(\alpha)_j)_{j \in [\ell], \alpha \in S}$. Here, $\boldsymbol{\beta}(\alpha)_j = (\beta(\alpha)_j^{(0)}, \ldots, \beta(\alpha)_j^{(s-1)})$ for field elements $\beta(\alpha)_j^{(i)}$. For $\ell = 1$, we are in the list decoding regime. We continue to refer the input as a *received word* even for the list recovery setting for this overview. Before proceeding further, we recall the main steps of the

near-linear time list decoding algorithm in [GHKS24], which, in turn is a fast implementation of the algorithm of Guruswami & Wang [GW13]. The algorithm has two main steps.

- **A differential form that explains the received word:** In the first step the goal is to construct a non-zero polynomial $Q(X, Y_0, \ldots, Y_m)$ of the form $Q = \widetilde{Q}(X) + \sum_i Q_i(X)Y_i$ for an appropriately chosen parameter $m = \Theta(\sqrt{s})$, with the property that the degree of Q is not too large and for every polynomial f whose encoding is close enough to the received word, we have that

  $$Q(X, f(X), f^{(1)}(X), \ldots, f^{(m)}(X)) \equiv 0.$$

  In other words, Q gives us a linear differential equation of high order such that its solution space contains all the close enough codewords.

- **Solving the differential equation:** The second part of the algorithm involves solving the differential equation given by the first part and recovering all close enough codewords. This, in turn, is done in two steps. In the first step, a near-linear time algorithm is designed to obtain a basis of this solution space (which has dimension at most m) and then a pruning algorithm of Kopparty, Ron Zewi, Saraf & Wootters [KRSW23, Tam24] is invoked to obtain a constant-sized list of close enough codewords.

The high-level structure of the fast list recovery algorithm in this paper also follows the outline above. This should not be surprising since in the work of Guruswami & Wang [GW13], an algorithm with the above structure is shown to solve the list recovery problem for univariate multiplicity codes in polynomial time. In fact, beyond the setting of parameters, the list recovery and list decoding algorithms in [GW13] are the same.

To extend the results in [GHKS24] to the list recovery setting, we use the fast differential equation solver from [GHKS24] as it is. However, to extend the first step of constructing the differential equations requires some new technical observations. To elaborate a bit more on these differences, we start by recalling some more of the technical details of the construction of Q in [GHKS24]. Recall that for the list decoding setting, each α comes with a unique $\boldsymbol{\beta}(\alpha)$ in the received word. To construct Q, the algorithm in [GHKS24] starts by constructing univariate polynomials $A_0(X), A_1(X), \ldots, A_m(X)$ that satisfy $(A_i)^{(j)}(\alpha) = \beta(\alpha)^{(i+j)}$. These polynomials can be constructed in nearly linear time using standard Fast Fourier Transform-based techniques, quite well known in computational algebra literature. Given these $A_i's$, the authors then look at the set of polynomials in $X, Y_0, Y_1, \ldots, Y_m$ generated by taking $\mathbb{F}[X]$-weighted linear combinations of $\{Y_i - A_i(X) : i \in [m]\} \cup \{\prod_{\alpha \in S}(X - \alpha)^{s-m}\}$. This set of polynomials is a lattice over the ring $\mathbb{F}[X]$ (or equivalently, an $\mathbb{F}[X]$-module) and has some nice properties - any non-zero polynomial of sufficiently low degree in this lattice satisfies the properties desired from Q. This follows from the simple observation that if $f(X)$ agrees with the received word at α, then for every i, $f^{(i)}(X) - A_i(X)$ is zero modulo $(X - \alpha)^{s-i}$. Thus, if we could show that this lattice has a non-zero polynomial of sufficiently low degree and can also construct one such polynomial in nearly linear time, we would be done. This is shown in [GHKS24] using an analogue of Minkowski's theorem for this lattice and an algorithm of Alekhnovich for computing *shortest vectors* (under the degree norm) for such lattices.

The primary difficulty in generalizing this specific approach to the setting of list recovery stems from the fact that in the list recovery setting, we now have many $\boldsymbol{\beta}(\alpha)$ for every α. Thus, the definition of the the polynomials A_i immediately breaks down since the received word can no longer be seen as a function from the evaluation points α to vectors $\boldsymbol{\beta}$. One potential fix for this situation is to consider the given received word as a union of ℓ different received words (with an arbitrary split) in the list decoding setup, and try to work with them together. So, we can try to view the received word $(\alpha, \boldsymbol{\beta}(\alpha)_j)_{j\in[\ell],\alpha\in S}$ as ℓ different received words $r_0, \ldots, r_{\ell-1}$ defined as $r_i = (\alpha, \boldsymbol{\beta}(\alpha)_i)_{\alpha\in S}$. Now, for each of these r_i, we follow the strategy in the list decoding algorithm and construct the polynomials $A_{i,0}, \ldots, A_{i,m}$. To consolidate this information, we could now choose to work over the lattice generated by the polynomials $\prod_{i=0}^{\ell-1}(Y_j - A_{i,j})$ over the ring $\mathbb{F}[X]$. Even if we succeeded in finding a low degree non-zero polynomial Q in this new lattice (in near linear time) and showing that it provides us with a differential equation satisfied by all close enough codewords, we have a technical issue to resolve in the second step: the differential equation at hand is no longer linear and it is unclear how to go about solving it and recovering close enough codewords. The solution space of such a differential equation is not necessarily an affine space, and the techniques in [GHKS24] don't seem to be immediately applicable. We note that in certain coding-theoretic contexts (e.g. [Sud97, GS99]) we do indeed work with polynomials Q with high Y-degree, although for the particular situation at hand, we do not know how to proceed with this approach.

Given the technical difficulties outlined above, it seems natural to try and find a Q that is linear in the Y-variables for the interpolation step even in the list recovery setting. Guruswami & Wang show that this can indeed be done in polynomial time. For our proof, we show that this can in fact be done in nearly linear time. To this end, we give a careful construction of a lattice whose generators are linear in the Y-variables, such that the shortest non-zero vectors in this lattice are precisely differential equations that *capture* all close enough codewords. See section 3 for details.

Organization

The paper is organized as follows. Section 2 contains preliminaries related to the univariate multiplicity code and the subroutines invoked during the lattice construction. Section 3 contains the technical details for construction of the differential equation, via fast construction of the previously mentioned lattice. In Section 4, we mention the subroutines used to extract the list of codewords from the differential equation and give the final proof of Theorem 1.1.

2 Preliminaries

2.1 Notation

- We work in some fixed finite field $\mathbb{F}$.
- For a natural number N, we will use $[N]$ to refer to the set $\{0, 1, 2, \ldots, N-1\}$.
- We use $\mathbf{0}_k$ to denote the k-length vector of all-zeroes. We may drop the subscript if the length is clear from context.

2.2 Univariate multiplicity codes

The following is the formal definition of the univariate multiplicity code.

Definition 2.1 (Multiplicity codes)**.** *Let n, k, s be positive integers satisfying $ks < n$, $\mathbb{F}$ be a field of characteristic zero or at least k and size at least n, and let $S = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ be a set of n distinct elements of $\mathbb{F}$. Then, univariate multiplicity codes with multiplicity parameter s for degree-k polynomials is defined as follows.*

The alphabet of the code is $E = \mathbb{F}^s$ and the block length is $n = |S|$, where the coordinates of a codeword are indexed by $\alpha \in S$. The message space is the set of all univariate polynomials of degree at most k in $\mathbb{F}[x]$. And, the encoding of such a message $f \in \mathbb{F}[x]$, denoted by $\mathsf{Mult}_s(f) \in E^S$ is defined as

$$\mathsf{Mult}_s(f)|_\alpha := \left(f(\alpha), f^{(1)}(\alpha), \ldots, f^{(s-1)}(\alpha)\right),$$

where $f^{(j)}(\alpha)$ is the evaluation of the j^{th}-order derivative[1] of f on the input α. ⌟

In the list recovery setting, the received word R consists of: for every evaluation point $\alpha \in S$, a collection of tuples $\boldsymbol{\beta}(\alpha)_j = (\beta(\alpha)_j^{(0)}, \beta(\alpha)_j^{(1)}, \ldots, \beta(\alpha)_j^{(s-1)})$ for every $j \in [\ell]$. In words $\beta(\alpha)_j^{(i)}$ is the j^{th} option for the i^{th} derivative of the message polynomial at α. Agreement is with respect to inclusion in this list; that is, $\mathsf{Mult}(f)$ agrees with R on point α if there exists $j \in [\ell]$ such that $(f(\alpha), f^{(1)}(\alpha), \ldots, f^{(s-1)}(\alpha)) = \boldsymbol{\beta}(\alpha)_j = (\beta(\alpha)_j^{(0)}, \beta(\alpha)_j^{(1)}, \ldots, \beta(\alpha)_j^{(s-1)})$.

Our goal is to carry out list recovery from an agreement fraction of $k/ns + \varepsilon$, for a given parameter ε.

2.3 Polynomial operators and lattices

We recall below the τ operator, used in [GW13] and [GHKS24].

Definition 2.2 (Tau operator)**.** *Let $s \in \mathbb{N}$ be a parameter. Then, for every m such that $0 \le m < s$, the function τ is an $\mathbb{F}$-linear map from the set of polynomials in $\mathbb{F}[X, Y_0, Y_1, \ldots, Y_m]$ with $\mathbf{Y}$-degree 1 to the set of polynomials in $\mathbb{F}[X, Y_0, Y_1, \ldots, Y_{m+1}]$ with $\mathbf{Y}$-degree 1 and is defined as follows.*

$$\tau\left(\widetilde{Q}(X) + \sum_{i=0}^{m} Q_i(X) \cdot Y_i\right) := \widetilde{Q}^{(1)}(X) + \sum_{i=0}^{m} \left(Q_i^{(1)}(X) \cdot Y_i + Q_i(X) \cdot Y_{i+1}\right).$$

For an integer $i \le (s - m)$, we use $\tau^{(i)}$ to denote the linear map from $\mathbb{F}[X, Y_0, Y_1, \ldots, Y_m]$ to $\mathbb{F}[X, Y_0, Y_1, \ldots, Y_{m+1}]$ obtained by applying the operator τ iteratively i times. ⌟

The operator is defined this way so that

$$\frac{d}{dX} Q(X, f(X), f^{(1)}(X), \ldots, f^{(m)}(X)) = \tau Q(X, f(X), f^{(1)}(X), \ldots, f^{(m+1)})(X).$$

We will extensively use the following simple property of the τ operator.

[1] This definition in terms of (standard) derivatives requires the characteristic of the field to be 0 or at least d. All known capacity-achieving list-decoding results for univariate multiplicity codes require large characteristic (previous works [Kop14, GW13, KRSW23, GHKS24] as well as our work). For this reason as well as ease of presentation, we work with standard derivatives.

Proposition 2.3. *Let α be a field element and $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_i, \ldots, \beta_{s-1}) \in \mathbb{F}^s$. Then, for any integer j with $1 \leq j \leq s - m$,*

$$\tau^{(j)}[(X - \alpha)Q](\alpha, \boldsymbol{\beta}) = j\tau^{(j-1)}(Q)(\alpha, \boldsymbol{\beta}).$$

Proof. We prove this by induction on j. When $j = 1$, the left-hand side is $\tau[(X - \alpha)Q](\alpha, \boldsymbol{\beta})$. From the definition, this equals $[(X - \alpha)\tau Q + Q](\alpha, \boldsymbol{\beta})$. (This can be checked term by term.) The first term $(X - \alpha)\tau Q$ is zero when evaluated at $(\alpha, \boldsymbol{\beta})$ leaving $Q(\alpha, \boldsymbol{\beta})$ as required.

For the induction step, observe similarly that

$$\begin{aligned}\tau^{(j)}[(X - \alpha)Q] &= \tau^{(j-1)}\tau[(X - \alpha)Q] \\ &= \tau^{(j-1)}[(X - \alpha)\tau Q + Q] \\ &= \tau^{(j-1)}[(X - \alpha)\tau Q] + \tau^{(j-1)}Q\end{aligned}$$

Evaluating at $(\alpha, \boldsymbol{\beta})$, by the induction hypothesis,

$$\begin{aligned}\tau^{(j-1)}[(X - \alpha)\tau Q](\alpha, \boldsymbol{\beta}) + \tau^{(j-1)}Q(\alpha, \boldsymbol{\beta}) &= (j-1)\tau^{j-2}[\tau Q](\alpha, \boldsymbol{\beta}) + \tau^{(j-1)}Q(\alpha, \boldsymbol{\beta}) \\ &= (j-1)\tau^{(j-1)}Q(\alpha, \boldsymbol{\beta}) + \tau^{(j-1)}Q(\alpha, \boldsymbol{\beta}) \\ &= j\tau^{(j-1)}Q(\alpha, \boldsymbol{\beta}).\end{aligned}$$

□

We now define lattices and state the version of Minkowski's theorem for polynomial lattices.

Definition 2.4 (polynomial lattice)**.** *Let $\mathcal{B} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N\}$ be a set of m-dimensional vectors of polynomials, that is, $\mathbf{v}_i \in \mathbb{F}[X]^m$ for all i. The* lattice *$\mathcal{L}_\mathcal{B}$ generated by this set over the ring $\mathbb{F}[X]$ is the set of all linear combinations*

$$\mathcal{L}_\mathcal{B} := \{f_i\mathbf{v}_1 + f_2\mathbf{v}_2 + \cdots + f_N\mathbf{v}_N \colon f_i \in \mathbb{F}[X]\}.$$

A basis *of a lattice is a set $\mathcal{B}$ of vectors of polynomials such that $\mathcal{B}$ generates the lattice in the sense of the above definition and the vectors in $\mathcal{B}$ are linearly independent over the field $\mathbb{F}(X)$ of rational functions over $\mathbb{F}$.* ⌟

Theorem 2.5 (polynomial version of Minkowski's theorem)**.** *Let $\mathcal{L}$ be an m-dimensional polynomial lattice. Then, there is a nonzero vector $\mathbf{v}$ satisfying*

$$\deg \mathbf{v} \leq \frac{1}{m} \deg \det \mathcal{L}$$

where deg $\mathbf{v}$ *is the max-degree norm and* det $\mathcal{L}$ *is the determinant of the lattice basis.*

A proof can be found in [GHKS24].

Theorem 2.6 ([Ale05, Theorem 2.1],[GSSV12, Theorem 9])**.** *Let B be a set of N vectors of dimension m. Let d be the maximal degree of the polynomials which are entries of the vectors in B. Then, there*

is an algorithm `ShortVector` *that finds the shortest vector in $\mathcal{L}_B$, the lattice generated by B, in time $\widetilde{O}(d(N+m)^\omega)$ where ω is the exponent in the complexity of matrix multiplication.*

Here "shortest vector" is in the max-degree norm. See [GHKS24] for details.

We will also use the following subroutine off the shelf, for fast Hermite interpolation and for polynomial Chinese remainders.

Theorem 2.7 ([GG13, Algorithm 10.22])**.** *Let $m_1(X), m_2(X), \ldots, m_r(X)$ be univariate polynomials in $\mathbb{F}[x]$ with pairwise GCD equal to 1. Let $v_1(X), v_2(X), \ldots, v_r(X) \in \mathbb{F}[X]$ such that $\deg v_i < \deg m_i$ for all $i \in [r]$. Then, we can find the unique polynomial $f(X) \in \mathbb{F}[X]$ of degree less than $d = \sum_i \deg m_i$ such that $f \equiv v_i \mod m_i$ for all $i \in [r]$, in time $O(d \operatorname{poly}\log(d))$.*

3 Fast construction of differential equation

As in Guruswami-Wang, our first step is to construct an "explanation" polynomial for the received word. In this section, we show that can be done in near-linear time. Below is the main theorem for this section.

Theorem 3.1. *Let $R = \left(\alpha, \beta_j^{(0)}, \beta_j^{(1)}, \ldots, \beta_j^{(s-1)}\right)_{\alpha \in S, j \in [\ell]}$ be a received word and let $m \leq s-1$ be a parameter.*

Then, for $D \leq n\ell(s-m)/m + n\ell$, there exists a non-zero polynomial $Q(X, Y_0, Y_1, \ldots, Y_m)$ of the form $Q = \widetilde{Q}(X) + \sum_i Q_i(X) \cdot Y_i$ of X-degree at most D with the following property: if $f(X) \in \mathbb{F}_{<k}[X]$ is such that for at least $(D+k)/(s-m)$ values of $\alpha \in S$, we have

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

for some $j \in [\ell]$, then,

$$Q\left(X, f(X), f^{(1)}(X), \ldots, f^{(m)}(X)\right) \equiv 0. \tag{3.2}$$

Furthermore, there is a deterministic algorithm that takes as input R and returns Q as a list of coefficients in time $\widetilde{O}(n \operatorname{poly}(s+m+\ell))$ — specifically, $\widetilde{O}(n[\ell(s-m)^2 + m^2(s-m) + (s-m)m^\omega])$.

A natural way to ensure that the polynomial Q satisfies (3.2), is to ensure that for each $\alpha \in S$, we have the following conditions

$$\tau^{(t)}(Q)\left(\alpha, \beta_j^0, \beta_j^1, \ldots, \beta_j^{m+t}\right) = 0, \qquad \forall t \in [s-m],\ j \in [\ell]. \tag{3.3}$$

This will ensure that every close enough codeword satisfies (3.2) (see Lemma 3.8 for exact details). We will refer to these conditions as the "τ-conditions". The set of polynomials Q satisfying (3.3) are closed under addition and multiplication by a polynomial in $\mathbb{F}[X]$. In other words, they form a $\mathbb{F}[X]$-module (or equivalently a lattice with entries from the univariate ring $\mathbb{F}[X]$). Here, it will be convenient to view the polynomial $Q = \widetilde{Q}(X) + \sum_i Q_i(X) \cdot Y_i$ as an $(m+2)$-dimensional vector of the form $(\widetilde{Q}, Q_0, Q_1, \ldots, Q_m) \in \mathbb{F}[X]^{m+2}$ where the first component $\widetilde{Q}$ is the Y-free part of Q and the subsequent components are the coefficients of $Y_0, Y_1, \ldots, Y_m$ respectively. To prove Theorem 3.1, we will first construct a nice-basis for this

lattice and then use Minkowski's theorem to show that there exists a short vector in this lattice. The following proposition gives this nice-basis for a sub-lattice of the lattice of polynomials satisfying the "τ-conditions" and use Minkowski's theorem on this sub-lattice to show a low degree polynomial within this sub-lattice itself.

Proposition 3.4. *Let ℓ be the list-size and $R = \left(\alpha, \beta_j^{(0)}, \beta_j^{(1)}, \ldots, \beta_j^{(s-1)}\right)_{\alpha \in S, j \in [\ell]}$ be the received word and let $m \leq s-1$ be a parameter.*

Then, there exist $m+2$ polynomials in $\mathbb{F}[X, Y_0, \ldots, Y_m]$ satisfying the τ-conditions (3.3) for all $\alpha \in S$ of the following type

$$\begin{aligned}
\widetilde{\mathbf{B}} &= \prod_{\alpha \in S} (X-\alpha)^{s-m}, \\
\mathbf{B}_i &= Y_i \cdot \prod_{\alpha \in S} (X-\alpha)^{s-m}, && \forall i, \text{ such that } 0 \leq i \leq \ell - 2 \\
\mathbf{B}_i &= \widetilde{C}_i(X) + \sum_{r=0}^{i-1} Y_r \cdot C_{i,r}(X) + Y_i \cdot \prod_{\alpha \in S} (X-\alpha)^{\ell-1}, && \forall i, \text{ such that } \ell - 1 \leq i \leq m.
\end{aligned}$$

for some polynomials $\widetilde{C}_i, C_{i,r} \in \mathbb{F}[X]$ for $\ell - 1 \leq i \leq m$ and $0 \leq r \leq i-1$. Furthermore, there is a deterministic algorithm that takes as input R and returns these $(m+2)$-polynomials in time $\widetilde{O}(n[\ell(s-m)^2 + m^2(s-m)])$.

The equivalent proposition of Proposition 3.4 in the list-decoding case was easier to obtain since the conditions for all the evaluation points (i.e., $\alpha \in S$) are identical. However, this is not the case when the list-size ℓ is more than 1. In order to address this, we first show that a version of Proposition 3.4 can be proved for each $\alpha \in S$ and then combine these different bases for the different α into a single basis that works for all the α. The following proposition is the version of Proposition 3.4 for a single $\alpha \in S$ (which is proved in Section 3.1).

Proposition 3.5. *Let $\alpha \in S$ and ℓ be the list-size and let $m \leq s-1$ be a parameter.*

Given ℓ possible evaluations (i.e, $\left(\beta_j^{(0)}, \beta_j^{(1)}, \ldots, \beta_j^{(s-1)}\right)$ for each $j \in [\ell]$) at the point α, There, exist $m+2$ polynomials in $\mathbb{F}[X, Y_0, \ldots, Y_m]$ satisfying the τ-conditions (3.3) for this specific $\alpha \in S$ of the following type

$$\begin{aligned}
\widetilde{\mathbf{B}}_\alpha &= (X-\alpha)^{s-m}, \\
\mathbf{B}_{\alpha,i} &= Y_i \cdot (X-\alpha)^{s-m}, && \forall i, \text{ such that } 0 \leq i \leq \ell - 2 \\
\mathbf{B}_{\alpha,i} &= \widetilde{C}_i(X) + \sum_{r=0}^{i-1} Y_r \cdot C_{i,r}(X) + Y_i \cdot (X-\alpha)^{\ell-1}, && \forall i, \text{ such that } \ell - 1 \leq i \leq m.
\end{aligned}$$

for some polynomials $\widetilde{C}_i, C_{i,r} \in \mathbb{F}[X]$ for $\ell - 1 \leq i \leq m$ and $0 \leq r \leq i-1$. Furthermore, there is a deterministic algorithm that takes as input $\left(\beta_j^{(0)}, \beta_j^{(1)}, \ldots, \beta_j^{(s-1)}\right)_{j \in [\ell]}$ and returns these $(m+2)$ polynomials in time $\widetilde{O}(\ell(s-m)^2)$.

We then show in Section 3.2 that the bases for two disjoint sets U, V of evaluation points (i.e., $U, V \subseteq S$) can be combined into a basis for the set $W = U \cup V$ in time approximately $\widetilde{O}(\max(|U|, |V|))$. We use this procedure recursively over the bases constructed via Propo-

sition 3.5 for each $\alpha \in S$ to obtain a basis that works simultaneously for all $\alpha \in S$ to yield Proposition 3.4.

3.1 Construction of basis for a single evaluation point

In this section, we will fix an evaluation point $\alpha \in S$ and construct a set of $m+2$ linearly independent polynomials (over the ring $\mathbb{F}[X]$) as specified in Proposition 3.5. More precisely, we want to show that there exist $m+2$ polynomials of the following type that satisfy the τ-conditions for α. Since α will be fixed throughout this section, we will drop α from the subscript to declutter the notation.

$$\begin{aligned}
\widetilde{\mathbf{B}} &= (X-\alpha)^{s-m}, \\
\mathbf{B}_i &= Y_i \cdot (X-\alpha)^{s-m}, && \forall i, \text{ such that } 0 \le i \le \ell-2 \\
\mathbf{B}_i &= \widetilde{C}_i(X) + \sum_{r=0}^{i-1} Y_r \cdot C_{i,r}(X) + Y_i \cdot (X-\alpha)^{\ell-1}, && \forall i, \text{ such that } \ell-1 \le i \le m.
\end{aligned}$$

for some polynomials $\widetilde{C}_i, C_{i,r} \in \mathbb{F}[X]$ for $\ell-1 \le i \le m$ and $0 \le r \le i-1$.

To this end, given a polynomial $\mathbf{B}$ of the above form and any $\boldsymbol{\beta} = (\beta^{(0)}, \beta^{(1)}, \ldots, \beta^{(s-1)})$ we will find it convenient to use the following notation for the vector in $\mathbb{F}^{s-m}$ composed of evaluations of iterated τ-operator of the polynomial $\mathbf{B}$ at the point $(\alpha, \boldsymbol{\beta})$:

$$\boldsymbol{\tau}(\mathbf{B}, \boldsymbol{\beta}) := \tau^{(<s-m)}(\mathbf{B})(\alpha, \boldsymbol{\beta}) = \begin{bmatrix} \mathbf{B}(\alpha, \boldsymbol{\beta}) \\ \tau(\mathbf{B})(\alpha, \boldsymbol{\beta}) \\ \tau^{(2)}(\mathbf{B})(\alpha, \boldsymbol{\beta}) \\ \vdots \\ \tau^{(s-m-1)}(\mathbf{B})(\alpha, \boldsymbol{\beta}) \end{bmatrix}.$$

The linearity of the τ operator shows that the function $\boldsymbol{\tau}(\cdot, \cdot)$ is linear it its first argument.

Proof of Proposition 3.5. In the notation defined above, we want to show that $\boldsymbol{\tau}(\mathbf{B}, \boldsymbol{\beta}) = \mathbf{0}_{s-m}$ for all $\boldsymbol{\beta} \in \{\boldsymbol{\beta}_j : j \in [\ell]\}$ and each $\mathbf{B} \in \{\widetilde{\mathbf{B}}, \mathbf{B}_0, \mathbf{B}_1, \ldots, \mathbf{B}_m\}$. Clearly, this is true when $\mathbf{B} \in \{\widetilde{\mathbf{B}}, \mathbf{B}_0, \mathbf{B}_1, \ldots, \mathbf{B}_{\ell-2}\}$. In the rest of the proof, we will construct polynomials $\widetilde{C}_i, C_{i,r} \in \mathbb{F}[X]$, such that this is also true for the polynomials $\mathbf{B} = \{\mathbf{B}_{\ell-1}, \mathbf{B}_\ell, \ldots, \mathbf{B}_m\}$. We will prove this by induction on ℓ, the list size.

Base case $\ell = 1$: This is the list decoding setting, where we have only β in the list corresponding to α, namely $\boldsymbol{\beta}_0 = (\beta_0^{(0)}, \beta_0^{(1)}, \ldots, \beta_0^{(s-1)})$. The proof in this case is similar to the corresponding step in [GHKS24], which is obtained by Hermite interpolation. For each $0 \le i \le m$, we define the polynomial $\widetilde{C}_i$ to be the unique $(s-m-1)$-degree polynomial in $\mathbb{F}[X]$ such that for all $j \in [s-m]$, we have

$$\widetilde{C}_i^{(j)}(\alpha) = -\beta_0^{(i+j)}.$$

The remaining $C_{i,r}(X)$ are set to zero for all $0 \le i \le m$ and $r \in [i]$. In other words, the i^{th}

polynomial $\mathbf{B}_i := Y_i + \widetilde{C}_i(X)$ for $\ell - 1 = 0 \leq i \leq m$. $\mathbf{B}_i$ clearly satisfies the τ-conditions (3.3) for α.

Induction Step $\ell \geq 2$: By induction, let us assume that for the first $(\ell - 1)$ list elements $\boldsymbol{\beta}_j, j \in [\ell - 1]$, we have polynomials $\mathbf{B}'_i$ satisfying the τ-conditions for α, namely $\boldsymbol{\tau}(\mathbf{B}'_i, \boldsymbol{\beta}_j) = \mathbf{0}$ for all i such that $\ell - 2 \leq i \leq m$ and $j \in [\ell - 1]$. We now need to construct polynomials $\mathbf{B}_i$ for $\ell - 1 \leq i \leq m$ such that $\boldsymbol{\tau}(\mathbf{B}_i, \boldsymbol{\beta}_j) = \mathbf{0}$ for all i such that $\ell - 1 \leq i \leq m$ and $j \in [\ell]$. Note that the number of polynomials to be constructed in the ℓ-case is one less than the $(\ell - 1)$-case while the number of $\boldsymbol{\beta}$'s for which we need $\boldsymbol{\tau}(\mathbf{B}_i, \boldsymbol{\beta}_j) = \mathbf{0}$ is one more.

Fix an i such that $\ell - 1 \leq i \leq m$. We will construct the polynomial $\mathbf{B}_i$ from the polynomial $\mathbf{B}'_{i-1}$. Consider the polynomial $\mathbf{B}''_i := \tau\left((X - \alpha) \cdot \mathbf{B}_{i-1}\right)$. Clearly, this polynomial is of the required form $Y_i \cdot (X - \alpha)^{\ell-1} + \sum_{r \in [i]} Y_r \cdot C_{i,r}(X) + \widetilde{C}_i(X)$. By Proposition 2.3, we have that $\boldsymbol{\tau}(\mathbf{B}''_i, \boldsymbol{\beta}_j) = \mathbf{0}$ for all $j \in [\ell - 1]$. Now, if $\boldsymbol{\tau}(\mathbf{B}''_i, \boldsymbol{\beta}_{\ell-1})$ is also $\mathbf{0}$, we can set $\mathbf{B}_i := \mathbf{B}''_i$ and be done.

Suppose $\boldsymbol{\tau}(\mathbf{B}''_i, \boldsymbol{\beta}_{\ell-1}) \neq \mathbf{0}$. Then, let $r_* \in [s - m]$ be the least r such that $\tau^{(<r)}(\mathbf{B}''_i)(\alpha, \boldsymbol{\beta}_{\ell-1}) = \mathbf{0}_r$ but $\tau^{(r)}(\mathbf{B}''_i)(\alpha, \boldsymbol{\beta}_{\ell-1}) \neq 0$. By Proposition 2.3, definition of $\mathbf{B}''_i$ and the fact that $s - m$ is less than the characteristic of $\mathbb{F}$, we also have $\tau^{(<r)}(\mathbf{B}'_{i-1})(\alpha, \boldsymbol{\beta}_{\ell-1}) = \mathbf{0}_r$ but $\tau^{(r)}(\mathbf{B}'_{i-1})(\alpha, \boldsymbol{\beta}_{\ell-1}) \neq 0$. Let us consider the $(s - m - r_*)$ polynomials $\{(X - \alpha)^t \cdot \mathbf{B}'_{i-1} : t \in [s - m - r_*]\}$. The t-th polynomial in this set, namely $(X - \alpha)^t \cdot \mathbf{B}'_{i-1}$ satisfies $\tau^{(<r+t)}((X - \alpha)^t \cdot \mathbf{B}'_{i-1})(\alpha, \boldsymbol{\beta}_{\ell-1}) = \mathbf{0}_{r+t}$ but $\tau^{(r+t)}((X - \alpha)^t \cdot \mathbf{B}_{i-1})(\alpha, \boldsymbol{\beta}_{\ell-1}) \neq 0$. Hence, the $(s - m - r_*)$ vectors $\{\boldsymbol{\tau}\left((X - \alpha)^t \cdot \mathbf{B}'_{i-1}, \boldsymbol{\beta}_{\ell-1}\right) : t \in [s - m - r_*]\}$ form a basis for the space of all vectors in $\mathbb{F}^{s-m}$ which are zero in the first r_* coordinates. This space contains the vector $\boldsymbol{\tau}(\mathbf{B}''_i, \boldsymbol{\beta}_{\ell-1})$ and hence there exist scalars $c_t \in \mathbb{F}$ for $t \in [s - m - r_*]$ such that

$$\boldsymbol{\tau}(\mathbf{B}''_i, \boldsymbol{\beta}_{\ell-1}) = \sum_{t \in [s-m-r_*]} c_t \cdot \boldsymbol{\tau}((X - \alpha)^t \cdot \mathbf{B}'_{i-1}, \boldsymbol{\beta}_{\ell-1}).$$

We can now set

$$\mathbf{B}_i := \mathbf{B}''_i - \sum_{t \in [s-m-r_*]} c_t \cdot (X - \alpha)^t \cdot \mathbf{B}'_{i-1}.$$

By choice of the scalars c_t, we have that $\boldsymbol{\tau}(\mathbf{B}_i, \boldsymbol{\beta}_{\ell-1}) = \mathbf{0}$. Since $\boldsymbol{\tau}(\mathbf{B}''_i, \boldsymbol{\beta}_j) = \mathbf{0}$ and $\boldsymbol{\tau}(\mathbf{B}'_{i-1}, \boldsymbol{\beta}_j) = \mathbf{0}$ for all $j \in [\ell - 1]$, we also have that $\boldsymbol{\tau}(\mathbf{B}_i, \boldsymbol{\beta}_j) = \mathbf{0}$ for all $j \in [\ell - 1]$. Combining, we have that $\boldsymbol{\tau}(\mathbf{B}_i, \boldsymbol{\beta}_j) = \mathbf{0}$ for all $j \in [\ell]$.

It can be seen that $\mathbf{B}_i$ is of the required form $Y_i \cdot (X - \alpha)^{\ell-1} + \sum_{r \in [i]} Y_r \cdot C_{i,r}(X) + \widetilde{C}_i(X)$ since $\mathbf{B}''_i$ is of this form and each of the polynomials $(X - \alpha)^t \cdot \mathbf{B}'_{i-1}$ are of the form $\sum_{r \in [i]} Y_r \cdot C_{i,r}(X) + \widetilde{C}_i(X)$.

We turn to the time complexity. For each list element, we may have to solve a system of equations to find the scalars $\{c_t\}$. Observe that this is a triangular system in at most $s - m$ variables, and hence can be solved in $O(s - m)^2$ time; giving an overall total of $\widetilde{O}(\ell(s - m)^2)$ time. This concludes the proof of the proposition. □

3.2 Combining the bases

In the previous section, we constructed a nice basis for each $\alpha \in S$. In this section, we will show how to combine these bases recursively to obtain a single basis that works for all $\alpha \in S$. To this end, we begin by showing a general procedure to combine the bases of two $\mathbb{F}[X]$-modules to generate the basis for a related $\mathbb{F}[X]$-module.

Let S be the set of evaluation points. For each $\alpha \in S$, let $p_\alpha(X)$ be an associated monic polynomial of degree at most $(s-m)$ such that for any two distinct $\alpha, \beta \in S$, we have $\gcd(p_\alpha, p_\beta) = 1$. We will be setting $p_\alpha(X) = (X-\alpha)^{s-m}$.

Let $\mathcal{U}, \mathcal{V}$ be two disjoint non-empty subsets of S. Define the polynomials

$$P_{\mathcal{U}}(X) := \prod_{\alpha \in \mathcal{U}} p_\alpha(X), \qquad\qquad P_{\mathcal{V}}(X) := \prod_{\alpha \in \mathcal{V}} p_\alpha(X).$$

Observe that since $\mathcal{U} \cap \mathcal{V} = \emptyset$, we have $\gcd(P_{\mathcal{U}}, P_{\mathcal{V}}) = 1$.

Let $\mathcal{M}_{\mathcal{U}}, \mathcal{M}_{\mathcal{V}}$ be two $(m+2)$-dimensional $\mathbb{F}[X]$-modules[2] with lower-triangular bases $\mathcal{B}_{\mathcal{U}} = \{\widetilde{\mathbf{U}}, \mathbf{U}_0, \mathbf{U}_1, \ldots, \mathbf{U}_m\}$ and $\mathcal{B}_{\mathcal{V}} = \{\widetilde{\mathbf{V}}, \mathbf{V}_0, \mathbf{V}_1, \ldots, \mathbf{V}_m\}$ respectively of the following type:

$$\widetilde{\mathbf{U}} = \widetilde{U}(X), \qquad\qquad \widetilde{\mathbf{V}} = \widetilde{V}(X)$$

$$\mathbf{U}_i = \widetilde{U_i}(X) + \sum_{r=0}^{i} Y_r \cdot U_{i,r}(X), \qquad \mathbf{V}_i = \widetilde{V_i}(X) + \sum_{r=0}^{i} Y_r \cdot V_{i,r}(X), \qquad \forall 0 \le i \le m$$

Furthermore, suppose these bases $\mathcal{B}_{\mathcal{U}}, \mathcal{B}_{\mathcal{V}}$ behave "nicely" with their respective polynomials $P_{\mathcal{U}}, P_{\mathcal{V}}$ in the following sense: the diagonal elements $\widetilde{U}, U_{i,i}, i \in [m]$ are non-zero factors of the polynomial $P_{\mathcal{U}}$ and similarly $\widetilde{V}, V_{i,i}, i \in [m]$ are non-zero factors of the polynomial $P_{\mathcal{V}}$.

Let $\mathcal{W} = \mathcal{U} \cup \mathcal{V}$ and $P_{\mathcal{W}} = P_{\mathcal{U}} \cdot P_{\mathcal{V}}$ be the associated polynomial with the set $\mathcal{W}$. We can construct an lower-triangular basis $\mathcal{B}_{\mathcal{W}} = \{\widetilde{\mathbf{W}}, \mathbf{W}_0, \mathbf{W}_1, \ldots, \mathbf{W}_m\}$ which behaves "nicely" with the polynomial $P_{\mathcal{W}}$ as follows:

$$\widetilde{\mathbf{W}} = \widetilde{W}(X)$$

$$\mathbf{W}_i = \widetilde{W_i}(X) + \sum_{r=0}^{i} Y_r \cdot W_{i,r}(X), \qquad\qquad \forall 0 \le i \le m$$

where the diagonal elements are given by $\widetilde{W} = \widetilde{U} \cdot \widetilde{V}$ and $W_{i,i} = U_{i,i} \cdot V_{i,i}, 0 \le i \le m$ and the non-diagonal elements $W_{i,r}(X), r \in [i]$ are the unique polynomials (by the Chinese remainder theorem) such that

$$W_{i,r} \equiv U_{i,r} \cdot V_{i,i} \mod P_{\mathcal{U}},$$
$$W_{i,r} \equiv V_{i,r} \cdot U_{i,i} \mod P_{\mathcal{V}}.$$

This basis satisfies that $\mathbf{W}_i \equiv V_{i,i}(X) \cdot \mathbf{U}_i \mod P_{\mathcal{U}}$ and symmetrically, $\mathbf{W}_i \equiv U_{i,i}(X) \cdot \mathbf{V}_i \mod P_{\mathcal{V}}$.

Let $\mathcal{M}_{\mathcal{W}}$ be the $\mathbb{F}[X]$-module generated by this basis $\mathcal{B}_{\mathcal{W}}$. This basis satisfies the following nice property.

[2] Here, we identify the polynomial $\widetilde{Q} + \sum_{i \in [m]} Y_i \cdot Q_i(X)$ with the $(m+2)$-dimensional vector $(\widetilde{Q}, Q_0, Q_1, \ldots, Q_m)$

Claim 3.6. *If* $\mathbf{R} = \widetilde{R}(X) + \sum_{i\in[m]} Y_i \cdot R_i(X)$ *or equivalently* $(\widetilde{R}, R_0, R_1, \ldots, R_m)$ *satisfies* $\mathbf{R} \bmod P_{\mathcal{U}} \in \mathcal{M}_{\mathcal{U}}$ *and* $\mathbf{R} \bmod P_{\mathcal{V}} \in \mathcal{M}_{\mathcal{V}}$*, then* $\mathbf{R} \bmod P_{\mathcal{W}} \in \mathcal{M}_{\mathcal{W}}$.

Proof. Since $\mathbf{R} \bmod P_{\mathcal{U}} \in \mathcal{M}_{\mathcal{U}}$ and $\mathbf{R} \bmod P_{\mathcal{V}} \in \mathcal{M}_{\mathcal{V}}$, we have polynomials $\widetilde{A}, \widetilde{B}, A_i, B_i, 0 \le i \le m$ such that

$$\mathbf{R} \bmod P_{\mathcal{U}} = \widetilde{A}(X) \cdot \widetilde{\mathbf{U}} + \sum_{i=0}^{m} A_i(X) \cdot \mathbf{U}_i,$$
$$\mathbf{R} \bmod P_{\mathcal{V}} = \widetilde{B}(X) \cdot \widetilde{\mathbf{V}} + \sum_{i=0}^{m} B_i(X) \cdot \mathbf{V}_i.$$

Define polynomials

$$\widetilde{C}(X) := \widetilde{U}^{-1} \cdot P_{\mathcal{U}}^{-1} \cdot \widetilde{B} \bmod P_{\mathcal{V}}, \qquad \widetilde{D}(X) := \widetilde{V}^{-1} \cdot P_{\mathcal{V}}^{-1} \cdot \widetilde{A} \bmod P_{\mathcal{U}}$$
$$C_i(X) := U_{i,i}(X)^{-1} \cdot P_{\mathcal{U}}^{-1} \cdot B_i \bmod P_{\mathcal{V}}, \qquad D_i(X) := V_{i,i}^{-1} \cdot P_{\mathcal{V}}^{-1} \cdot A_i \bmod P_{\mathcal{U}}, \qquad i \in [m]$$

We claim that

$$\mathbf{R} = [\widetilde{C} \cdot P_{\mathcal{U}} + \widetilde{D} \cdot P_{\mathcal{V}}]\widetilde{\mathbf{W}} + \sum_{i=0}^{m} [C_i \cdot P_{\mathcal{U}} + D_i \cdot P_V]\mathbf{W}_i \bmod P_U P_V.$$

Indeed, observe the right-hand side mod $P_{\mathcal{U}}$. All the terms involving $P_{\mathcal{U}}$ vanish, leaving $\widetilde{D} \cdot P_{\mathcal{V}} \cdot \widetilde{\mathbf{W}} + \sum_{i=0}^{m} D_i \cdot P_{\mathcal{V}} \cdot \mathbf{W}_i$. Simplifying using the definition of $\mathbf{W}$, we have $\widetilde{A} \cdot \widetilde{\mathbf{U}} + \sum_{i\in[m]} A_i \cdot \mathbf{U}_i$, which is $\mathbf{R} \bmod P_{\mathcal{U}}$. The case mod P_V is symmetric and the equivalence follows from the Chinese remainder theorem. □

For our specific setting, the polynomial $p_\alpha(X)$ will be $(X - \alpha)^{s-m}$. For any subset $\mathcal{U} \subseteq S$ of the evaluation points, we will say that the basis $\mathcal{B}_{\mathcal{U}} = \{\widetilde{\mathbf{U}}, \mathbf{U}_0, \ldots, \mathbf{U}_m\}$ is $\mathcal{U}$-good, if the following two properties are satisfied.

1. The diagonal elements of the basis satisfy

$$\widetilde{U}(X) := \prod_{\alpha\in\mathcal{U}} (X - \alpha)^{s-m},$$
$$U_{i,i}(X) := \prod_{\alpha\in\mathcal{U}} (X - \alpha)^{s-m}, \qquad 0 \le i \le \ell - 2$$
$$U_{i,i}(X) := \prod_{\alpha\in\mathcal{U}} (X - \alpha)^{\ell-1}, \qquad \ell - 1 \le i \le m$$

   where ℓ is the list size.

2. Every basis element $\mathbf{U} \in \mathcal{B}_{\mathcal{U}}$ satisfies the τ-conditions (3.3) for every $\alpha \in \mathcal{U}$.

Clearly, the basis generated by Proposition 3.5 when $\mathcal{U} = \{\alpha\}$ is a singleton set is $\{\alpha\}$-good. The following claim shows that if $\mathcal{B}_{\mathcal{U}}, \mathcal{B}_{\mathcal{V}}$ are both good, so is $\mathcal{B}_{\mathcal{W}}$.

Claim 3.7. *Let* $\mathcal{U}, \mathcal{V} \subseteq S$ *be two disjoint non-empty subsets of the set of evaluation points* S*. If* $\mathcal{B}_{\mathcal{U}}$ *is* $\mathcal{U}$*-good and* $\mathcal{B}_{\mathcal{V}}$ *is* $\mathcal{V}$*-good, then* $\mathcal{B}_{\mathcal{W}}$ *is* $\mathcal{W}$*-good.*

Furthermore, there is a deterministic algorithm that when given the bases $\mathcal{B}_{\mathcal{U}}$ and $\mathcal{B}_{\mathcal{V}}$ outputs the basis $\mathcal{B}_{\mathcal{W}}$ in time $\widetilde{O}(\max\{|\mathcal{U}|, |\mathcal{V}|\}m^2(s-m))$.

Proof. Item 1 follows from how the diagonal elements of the basis $\mathcal{B}_{\mathcal{W}}$ are defined. The runtime bound follows from the time required for Chinese Remainder theorem (Theorem 2.7): for the $O(m^2)$ off-diagonal elements $W_{i,r}$, we carry out a CRT of polynomials of degree at most $\max\{|\mathcal{U}|, |\mathcal{V}|\}(s-m)$.

We now prove Item 2. Let $\mathbf{W} \in \mathcal{B}_{\mathcal{W}}$. If $\mathbf{W} = \widetilde{\mathbf{W}}$, then $\mathbf{W} = \widetilde{\mathbf{V}} \cdot \widetilde{\mathbf{U}} = \prod_{\alpha \in \mathcal{W}}(X-\alpha)^{s-m}$ and hence satisfies the τ-conditions for all $\alpha \in \mathcal{W}$. Suppose $\mathbf{W} = \mathbf{W}_i$ for some $0 \le i \le m$. Let $\alpha \in \mathcal{W}$. Suppose $\alpha \in \mathcal{U}$. By definition of $\mathbf{W}_i$, we have $\mathbf{W}_i \equiv V_{i,i} \cdot \mathbf{U}_i \bmod P_{\mathcal{U}}$ and is hence of the form $V_{i,i} \cdot \mathbf{U}_i + \prod_{\alpha \in \mathcal{U}}(X-\alpha)^{s-m} \cdot \mathbf{R}$ for some polynomial $\mathbf{R} = \widetilde{R} + \sum_{i=0}^{m} Y_i \cdot R_i(X)$. Since $\mathcal{B}_{\mathcal{U}}$ is $\mathcal{U}$-good, $\mathbf{U}_i$ satisfies the τ-conditions with respect to α. Hence, so does $V_{i,i} \cdot \mathbf{U}_i$ and $\mathbf{W}_i$. The case when $\alpha \in \mathcal{V}$ is similar. □

We are now ready to prove Proposition 3.4.

Proof of Proposition 3.4. Let $\mathcal{B}_S := \{\widetilde{\mathbf{B}}, \mathbf{B}_0, \mathbf{B}_1, \ldots, \mathbf{B}_m\}$ be obtained by the following process: Divide S into two halves $\mathcal{U}$ and $\mathcal{V}$, and recursively compute bases $\mathcal{B}_{\mathcal{U}}$ and $\mathcal{B}_{\mathcal{V}}$ for them, and then combine them as indicated earlier in this section. For a singleton set, we use the basis given by Proposition 3.5.

It follows from Claim 3.7, that since the bases for the singleton sets are good, so are all the bases that are recursively constructed. Thus, the basis $\mathcal{B}_S$ has all the properties mentioned in Proposition 3.4. The only thing left to be verified is the running time.

As for the running time, let $T(n)$ be the running time where n is the size of the set S. We split it into two instances of size $n/2$, each of which can be solved in time $T(n/2)$. Claim 3.7 states that the bases for the two instances can be combined in time $\widetilde{O}(nm^2(s-m))$. Finally, Proposition 3.5 states that the base case when the set is a single point requires time $\widetilde{O}(\ell(s-m)^2)$. Putting all this together, we have $T(n) = 2T(n/2) + \widetilde{O}(nm^2(s-m))$ and $T(1) = \widetilde{O}(\ell(s-m)^2)$. Solving, we get $T(n) = \widetilde{O}(n[\ell(s-m)^2 + m^2(s-m)])$. □

3.3 Proof of Theorem 3.1

Before we prove the main theorem of the section, we have to connect the τ-conditions to the desired property, of close enough codewords satisfying the differential equation.

Lemma 3.8. *Let $Q(X, Y_0, Y_1, \ldots, Y_m) \in \mathcal{M}_S$ have X-degree at most $D \le n\ell(s-m)/m + n\ell$ and Y_i-degree at most 1 for every Y_i. Let $f(X) \in \mathbb{F}_{<k}[X]$ be such that for $(D+k)/(s-m)$ values of $\alpha \in S$, $(f(\alpha), f^{(1)}(\alpha), \ldots, f^{(s-1)}(\alpha)) = (\beta_j^{(0)}, \beta_j^{(1)}, \ldots, \beta_j^{(s-1)})$ for some $j \in [\ell]$. Then, $Q(X, f(X), f^{(1)}(X), \ldots, f^{(s-1)}(X)) \equiv 0$.*

Proof. Let $P(X) = Q(X, f(X), f^{(1)}(X), \ldots, f^{(s-1)}(X))$. Since f has degree at most k and Q is linear in the Y-variables, $P(X)$ has degree at most $D+k$. The definition of the τ operator is set up such that $\tau^{(i)}(Q)(X, f(X), f^{(1)}(X), \ldots, f^{(s-1)}(X)) = \frac{d^i P}{dX^i}$. This combined with Proposi-

tion 3.4 implies that for an agreement point α,

$$\frac{d^i P}{dX^i}(\alpha) = [\tau^{(i)}(Q)](\alpha, f(\alpha), f^{(1)}(\alpha), \ldots, f^{(s-1)}(\alpha)) = \tau^{(i)}(Q)(\alpha, \beta_j^{(0)}, \beta_j^{(1)}, \ldots, \beta_j^{(s-1)}) = 0$$

for any $i \in [s-m]$. That is, at these points, P vanishes with multiplicity at least $(s-m)$. Therefore, if the number of such agreements is more than $(D+k)/(s-m)$, P must be identically zero. □

We are now ready to prove the main theorem of the section.

Proof of Theorem 3.1. We will take Q to be a polynomial in the $\mathbb{F}[X]$-span of the polynomials $\widetilde{\mathbf{B}}, \mathbf{B}_0, \mathbf{B}_1, \ldots, \mathbf{B}_m$ given by Proposition 3.4. Since each of them satisfies the τ conditions, as given by the proposition, so will Q. In addition, Q will have the form $Q = \widetilde{Q}(X) + \sum_i Q_i(X) \cdot Y_i$, inherited from the basis polynomials.

We now have to bound the degree of the coefficients. Following our previous notation, let $\mathcal{M}_S$ be span$\{\widetilde{\mathbf{B}}, \mathbf{B}_0, \mathbf{B}_1, \ldots, \mathbf{B}_m\}$. By Theorem 2.5, $\mathcal{M}_S$ contains a polynomial of degree at most $(\deg \det \mathcal{M}_S)/(m+2)$. Because of the lower triangular structure, to find $\deg \det \mathcal{M}_S$, we need only multiply the degrees of the terms on the diagonal.

The first ℓ terms on the diagonal are $\prod_{\alpha \in S}(X-\alpha)^{s-m}$, and the remaining $m+2-\ell$ are $\prod_{\alpha \in S}(X-\alpha)^{\ell-1}$. This gives $\deg \det \mathcal{M}_S = n\ell(s-m) + (m+2-\ell)n(\ell-1) \leq n\ell(s-m) + n(m+2)\ell$. Applying the Minkowski statement gives the required bound.

The close enough codewords statement is given by Lemma 3.8. This gives all the required properties of Q.

The running time is the time to construct the basis followed by the time to find a short vector in the lattice. From Proposition 3.4 and Theorem 2.6, we get a running time bound of $\widetilde{O}(n[\ell(s-m)^2 + m^2(s-m) + (s-m)m^\omega])$. □

4 List recovery algorithm

Having constructed the differential equation, we now have to solve it in near-linear time. We will be using subroutines off the shelf for this. We mention the relevant algorithms before analyzing the overall list recovery algorithm.

4.1 Fast algorithms for solving differential and functional equations

We will invoke the following algorithms for solving differential equations from [GHKS24]. (Their paper also contains an analogous result for functional equations, for FRS codes.)

Theorem 4.1 ([GHKS24, Theorem 5.1])**.** *Let $\mathbb{F}$ be a finite field of characteristic greater than d or zero, and let*

$$Q(X, Y_0, \ldots, Y_m) = \widetilde{Q}(X) + \sum_{i=0}^{m} Q_i(x) Y_i$$

be a non-zero polynomial with X-degree at most D. Then, the set of polynomials $f(X) \in \mathbb{F}[X]$ of degree at most d that satisfy

$$Q\left(X, f, f^{(1)} \ldots, f^{(m)}\right) \equiv 0$$

is contained in an affine space dimension at most m.

Furthermore, there is a deterministic algorithm `FastDESolver` *that when given Q as an input via its coefficient vector, and the parameter d, performs at most $\widetilde{O}((D+d)m^4)$ field operations and outputs a basis for this affine space.*

Remark 4.2. *The [GHKS24, Theorem 5.1] paper presents a complexity of $\widetilde{O}((D+d)\,\mathrm{poly}(m))$. On closer examination, their recursion contains a factor of at most m^4 as they solve for m different basis elements separately and each is solved using [GHKS24, Algorithm 1]. The time complexity for this step is analyzed in [GHKS24, Lemma 5.8] which their analysis shows has a dependence of at most $O(m^3)$.* ⌟

These algorithms output a subspace containing the codewords. We use the "Prune" subroutine introduced by [KRSW23], which was simplified by [Tam24], to obtain the codewords. The list size bounds we present below are due to [Tam24].

Theorem 4.3 ([KRSW23, Lemma 3.1], [Tam24, Theorem 4.5])**.** *For every $\varepsilon > 0, \ell \in \mathbb{N}$, all integers $s > \frac{16\ell}{\varepsilon^2}$, degree parameter k, block length n and field $\mathbb{F}$ of characteristic zero or greater than k, the following is true.*

Define

$$L(\delta, \varepsilon, \ell, r) = \left(\frac{\ell}{\varepsilon}\right)^{O(\frac{1+\log(\ell)}{\varepsilon})}.$$

Then, for any $\gamma > 0$, there is a randomized algorithm `Prune` *that when given as input sets $S \subseteq \mathbb{F}$ and $E_\alpha \subseteq \mathbb{F}^s$, with $|E_\alpha| \leq \ell$ for every $\alpha \in S$ and $|S| = n$, and a basis for an affine space $\mathcal{A}$ of dimension at most $4\ell/\varepsilon$, outputs*

$$\mathcal{L} = \{f(X) \in \mathbb{F}_{<k}[X] \cap \mathcal{A} \mid |\{\alpha \in S \mid (f(\alpha), f^{(1)}(\alpha), \cdots, f^{(s-1)}(\alpha)) \in E_\alpha\}| > (R+\varepsilon)n\}$$

with probability at least $1-\gamma$ in time $\widetilde{O}(n)\,\mathrm{poly}(\log q, s, L(\delta, \varepsilon, \ell, r), \log(1/\gamma))$ and it is guaranteed that,

$$|\mathcal{L}| \leq L(\delta, \varepsilon, \ell, r)\,.$$

4.2 Algorithm and proof of main theorem

The final algorithm proceeds in 3 steps: finding the differential equation, solving it, and pruning. Before restating and proving the main theorem, we state separately here the combination of the first two steps, as the time complexity in those parts is our main contribution.

Theorem 4.4. *For all integers $\ell, m, s \in \mathbb{N}$ with $m < s$, degree parameter k, block length n and field $\mathbb{F}$ of characteristic zero or greater than k, the following is true.*

There is a deterministic algorithm, that when given as input sets $S \subseteq \mathbb{F}$ and $E_\alpha \subseteq \mathbb{F}^s$ with $|E_\alpha| \leq \ell$ for every $\alpha \in S$ and $|S| = n$, runs in time $O(n \log |\mathbb{F}|(\ell(s^2 + sm^3) + \frac{k}{n}m^4) \operatorname{polylog}(n, s, \log\log|\mathbb{F}|))$ and outputs the basis of an affine space $\mathcal{A} \subseteq \mathbb{F}[X]$ of dimension at most m such that $\mathcal{A}$ contains all polynomials $f(X)$ of degree less than k such that for at at least $(\frac{n\ell+k}{n(s-m)} + \frac{\ell}{m})$ fraction of $\alpha \in S$,

$$(f(\alpha), \cdots, f^{(s-1)}(\alpha)) \in E_\alpha \,.$$

Proof. We begin by describing the algorithm.

Input: $R = \{(\alpha_i, \beta_j^{(0)}, \beta_j^{(1)}, \ldots, \beta_j^{(s-1)})\}_{i\in[n], j\in[\ell]}, k \in \mathbb{Z}$

Task: Return the basis of an affine space $\mathcal{A}$ of dimension at most m containing all $f \in \mathbb{F}_{<k}[X]$ such that for $n\ell/m + (n\ell + k)/(s - m)$ values of i,

$$(f(\alpha_i), f^{(1)}(\alpha_i), \ldots, f^{(s-1)}(\alpha_i)) = (\beta_j^{(0)}, \beta_j^{(1)}, \ldots, \beta_j^{(s-1)})$$

for some $j \in [\ell]$.

1. Construct a multivariate polynomial Q as per Theorem 3.1.
2. Solve the resulting differential equation using the algorithm in Theorem 4.1, to obtain an affine space A containing all the codewords.

Observe that by Theorem 3.1, we can construct a multivariate polynomial $Q(X, Y_0, \cdots, Y_m)$ of the form $Q = \widetilde{Q}(X) + \sum_i Q_i(X) \cdot Y_i$ of X-degree at most $D \leq \frac{n\ell(s-m)}{m} + n\ell$ with the following property: if $f(X) \in \mathbb{F}_{<k}[X]$ is such that for at least $\frac{1}{s-m}\left(\frac{n\ell(s-m)}{m} + n\ell + k\right) = \frac{n\ell}{m} + \frac{n\ell+k}{s-m}$ values of $\alpha \in S$, we have

$$(f(\alpha), \cdots, f^{(s-1)(\alpha)} \in E_\alpha) \,,$$

then,

$$Q(X, f(X), f^{(1)}(X), \cdots, f^{(m)}(X)) \equiv 0$$

in time $\widetilde{O}(n\ell((s-m)^2 + (s-m)m^\omega))$ which is within our time complexity.

Now, Theorem 4.1 outputs the basis of the affine space of dimension at most m of the polynomials of degree $< k$ which are solutions of the above differential equation in time $\widetilde{O}((D + k)m^4)$. Substituting $D \leq \frac{n\ell(s-m)}{m} + n\ell$, the time complexity is within the claimed range. □

We are finally ready to prove the main theorem. We first recall the theorem statement.

Theorem 1.1 (Main result). *For every $\varepsilon > 0, \ell \in \mathbb{N}$, there is an $s_0 \in \mathbb{N}$ such that for all $s > s_0$, degree parameter k, block length n and field $\mathbb{F}$ of characteristic zero or greater than k, the following is true.*

There is a randomized algorithm that when given as input sets $S \subseteq \mathbb{F}$ and $E_\alpha \subseteq \mathbb{F}^s$ with $|E_\alpha| \leq \ell$ for every $\alpha \in S$ and $|S| = n$, runs in time $O\left(n \operatorname{poly}(s, \ell, \log n, \log|\mathbb{F}|, (\ell/\varepsilon)^{\frac{1+\log\ell}{\varepsilon}})\right)$ and with high probability outputs the set of all univariate polynomials $f(X) \in \mathbb{F}[X]$ of degree less than k such that for

at least $(1 - k/sn - \varepsilon)$ *fraction of* $\alpha \in S$,

$$(f(\alpha), f^{(1)}(\alpha), \ldots, f^{(s-1)}(\alpha)) \in E_\alpha \,.$$

Proof. We will first set the parameters. Recall that the agreement fraction for which Theorem 3.1 works is

$$\frac{n\ell + k}{n(s-m)} + \frac{\ell}{m}.$$

The rate R is k/ns. For the above to approximately equal $R + \varepsilon$, we set $s_0 = 16\ell/\varepsilon^2 + \frac{4\ell}{\varepsilon}$ and $m = 4\ell/\varepsilon$ with $s > s_0$. Thus,

$$\frac{n\ell + k}{n(s-m)} + \frac{\ell}{m} \leq \frac{\ell}{(s-m)} + \frac{k}{n(s-m)} + \frac{\varepsilon}{2} \leq \frac{\varepsilon^2}{16} + \frac{\varepsilon}{4} + R \cdot \left(\frac{s}{s-m}\right) \leq R + \frac{\varepsilon \cdot R}{4} + \frac{\varepsilon}{4} + \frac{\varepsilon^2}{16}$$

Since $\varepsilon \leq 1 - R$ and thus $\varepsilon^2 \leq \varepsilon(1-R)$, we get that

$$\frac{n\ell + k}{n(s-m)} + \frac{\ell}{m} < R + \varepsilon/2.$$

The algorithm is given below.
Input: $R = \{(\alpha_i, \beta_j^{(0)}, \beta_j^{(1)}, \ldots, \beta_j^{(s-1)})\}_{i \in [n], j \in [\ell]}, k \in \mathbb{Z}$
Task: Return all $f \in \mathbb{F}_{<k}[X]$ such that for $n\ell/m + (n\ell + k)/(s-m)$ values of i,

$$(f(\alpha_i), f^{(1)}(\alpha_i), \ldots, f^{(s-1)}(\alpha_i)) = (\beta_j^{(0)}, \beta_j^{(1)}, \ldots, \beta_j^{(s-1)})$$

for some $j \in [\ell]$.

1. Find an affine space of dimension at most m which contains all the required codewords using Theorem 4.4.
2. Using the algorithm in Theorem 4.3, obtain a list $\mathcal{L}$ of codewords and return it.

Let f be a polynomial such that for $n\ell/m + (n\ell + k)/(s-m)$ values of i,

$$(f(\alpha_i), f^{(1)}(\alpha_i), \ldots, f^{(s-1)}(\alpha_i)) = (\beta_j^{(0)}, \beta_j^{(1)}, \ldots, \beta_j^{(s-1)})$$

for some $j \in [\ell]$. By Theorem 4.4, f should be contained in the output affine space. The algorithm in Theorem 4.3 finds f with high probability.

We saw in Theorem 4.4 that step 1 can be done in $\widetilde{O}(n \operatorname{poly}(s + m + \ell))$ time. Combining with the runtime of Theorem 4.3, we get a runtime of $\widetilde{O}\left(n \operatorname{poly}(s, \ell, (\ell/\varepsilon)^{\frac{1+\log \ell}{\varepsilon}})\right)$.

□

References

[AHS26] VIKRANT ASHVINKUMAR, MURSALIN HABIB, and SHASHANK SRIVASTAVA. *Algorithmic improvements to list decoding of folded Reed-Solomon codes*. In *Proc. 37th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*. 2026. (To appear). arXiv:2508.12548. 2

[Ale05] MICHAEL ALEKHNOVICH. *Linear Diophantine equations over polynomials and soft decoding of Reed-Solomon codes*. IEEE Trans. Inform. Theory, 51(7):2257–2265, 2005. (Preliminary version in *43rd FOCS*, 2002). 3, 7

[BCDZ25] JOSHUA BRAKENSIEK, YEYUAN CHEN, MANIK DHAR, and ZIHAN ZHANG. *Combinatorial bounds for list recovery via discrete Brascamp–Lieb inequalities*, 2025. (manuscript). arXiv:2510.13775. 2

[BHKS24] SIDDHARTH BHANDARI, PRAHLADH HARSHA, MRINAL KUMAR, and MADHU SUDAN. *Decoding multivariate multiplicity codes over product sets*. IEEE Trans. Inform. Theory, 70(1):154–169, 2024. (Preliminary version in *53rd STOC*, 2021). arXiv:2012.01530, eccc:2020/179. 2

[CZ25] YEYUAN CHEN and ZIHAN ZHANG. *Explicit folded Reed-Solomon and multiplicity codes achieve relaxed generalized singleton bound*. In MICHAL KOUCKÝ and NIKHIL BANSAL, eds., *Proc. 57th ACM Symp. on Theory of Computing (STOC)*, pages 1–12. 2025. arXiv:2408.15925. 2

[GG13] JOACHIM VON ZUR GATHEN and JÜRGEN GERHARD. *Modern Computer Algebra*. Cambridge University Press, 3 edition, 2013. 8

[GHKS24] ROHAN GOYAL, PRAHLADH HARSHA, MRINAL KUMAR, and ASHUTOSH SHANKAR. *Fast list-decoding of univariate multiplicity and folded Reed-Solomon codes*. In SANTOSH VEMPALA, ed., *Proc. 65th IEEE Symp. on Foundations of Comp. Science (FOCS)*, pages 328–343. 2024. eccc:2023/185. 2, 3, 4, 5, 6, 7, 8, 10, 15, 16

[GL89] ODED GOLDREICH and LEONID A. LEVIN. *A hard-core predicate for all one-way functions*. In *Proc. 21st ACM Symp. on Theory of Computing (STOC)*, pages 25–32. 1989. 2

[GR08] VENKATESAN GURUSWAMI and ATRI RUDRA. *Explicit codes achieving list decoding capacity: Error-correction with optimal redundancy*. IEEE Trans. Inform. Theory, 54(1):135–150, 2008. (Preliminary version in *38th STOC*, 2006). arXiv:cs/0511072, eccc:2005/133. 2

[GS99] VENKATESAN GURUSWAMI and MADHU SUDAN. *Improved decoding of Reed-Solomon and algebraic-geometry codes*. IEEE Trans. Inform. Theory, 45(6):1757–1767, 1999. (Preliminary version in *39th FOCS*, 1998). eccc:1998/043. 2, 3, 5

[GSSV12] SOMIT GUPTA, SOUMOJIT SARKAR, ARNE STORJOHANN, and JOHNNY VALERIOTE. *Triangular x-basis decompositions and derandomization of linear algebra algorithms over $k[x]$*. J. Symb. Comput., 47(4):422–453, 2012. 7

[GUV09] VENKATESAN GURUSWAMI, CHRISTOPHER UMANS, and SALIL P. VADHAN. *Unbalanced expanders and randomness extractors from Parvaresh-Vardy codes*. J. ACM,

56(4):20:1–20:34, 2009. (Preliminary version in *22nd CCC*, 2007). eccc:2006/TR06-134. 2

[GW13] VENKATESAN GURUSWAMI and CAROL WANG. *Linear-algebraic list decoding for variants of Reed-Solomon codes*. IEEE Trans. Inform. Theory, 59(6):3257–3268, 2013. (Preliminary version in *26th IEEE Conference on Computational Complexity*, 2011 and *15th RANDOM*, 2011). eccc:2012/073. 2, 3, 4, 6

[JMST25] FERNANDO GRANHA JERONIMO, TUSHANT MITTAL, SHASHANK SRIVASTAVA, and MADHUR TULSIANI. *Explicit codes approaching generalized Singleton bound using expanders*. In MICHAL KOUCKÝ and NIKHIL BANSAL, eds., *Proc. 57th ACM Symp. on Theory of Computing (STOC)*, pages 843–854. 2025. arXiv:2502.07308. 2

[JS25] FERNANDO GRANHA JERONIMO and NIKHIL SHAGRITHAYA. *Probabilistic guarantees to explicit constructions: Local properties of linear codes*, 2025. (manuscript). arXiv:2510.06185. 2

[Kop14] SWASTIK KOPPARTY. *Some remarks on multiplicity codes*. In ALEXANDER BARG and OLEG R. MUSIN, eds., *Discrete Geometry and Algebraic Combinatorics*, volume 625 of *Contemporary Mathematics*, pages 155–176. AMS, 2014. arXiv:1505.07547. 2, 3, 6

[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. (Preliminary version in *59th FOCS*, 2018). arXiv:1805.01498, eccc:2018/091. 2, 3, 4, 6, 16

[KT22] ITAY KALEV and AMNON TA-SHMA. *Unbalanced expanders from multiplicity codes*. In AMIT CHAKRABARTI and CHAITANYA SWAMY, eds., *Proc. 26th International Conf. on Randomization and Computation (RANDOM)*, volume 245 of *LIPIcs*, pages 12:1–12:14. Schloss Dagstuhl, 2022. eccc:2022/073. 2

[LMS25] MATAN LEVI, JONATHAN MOSHEIFF, and NIKHIL SHAGRITHAYA. *Local properties of Reed-Solomon codes and random linear codes are locally equivalent*. In *Proc. 66th IEEE Symp. on Foundations of Comp. Science (FOCS)*. 2025. (To appear). arXiv:2406.02238. 2

[RV25] NICOLAS RESCH and S. VENKITESH. *List recoverable codes: The good, the bad, and the unknown (hopefully not ugly)*, 2025. (manuscript). arXiv:2510.07597. 2

[Sri25] SHASHANK SRIVASTAVA. *Improved list size for folded Reed-Solomon codes*. In YOSSI AZAR and DEBMALYA PANIGRAHI, eds., *Proc. 36th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 2040–2050. 2025. arXiv:2410.09031. 2

[ST25] SHASHANK SRIVASTAVA and MADHUR TULSIANI. *List decoding expander-based codes up to capacity in near-linear time*. In *Proc. 66th IEEE Symp. on Foundations of Comp. Science (FOCS)*. 2025. (To appear). arXiv:2504.20333. 2

[Sud97] MADHU SUDAN. *Decoding of Reed-Solomon codes beyond the error-correction bound*. J. Complexity, 13(1):180–193, 1997. (Preliminary version in *37th FOCS*, 1996). 2, 3, 5

[Tam24] ITZHAK TAMO. *Tighter list-size bounds for list-decoding and recovery of folded Reed-Solomon and multiplicity codes*. IEEE Trans. Inform. Theory, 70(12):8659–8668, 2024. arXiv:2312.17097. 3, 4, 16

[Vad12] SALIL P. VADHAN. *Pseudorandomness*. Found. Trends Theor. Comput. Sci., 7(1-3):1–336, 2012. doi:10.1561/0400000010. 2

[Woo19] MARY WOOTTERS. *CS250/EE387: Algebraic error correcting codes*, 2019. (A course on coding theory at Stanford, Winter 2019). 2

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il