# Fast List Recovery of Univariate Multiplicity Codes

**Rohan Goyal**
Massachusetts Institute of Technology, Cambridge, MA, USA

**Prahladh Harsha**
Tata Institute of Fundamental Research, Mumbai, India

**Mrinal Kumar**
Tata Institute of Fundamental Research, Mumbai, India

**Ashutosh Shankar**
Tata Institute of Fundamental Research, Mumbai, India

## Abstract

Recent work gave near-linear time algorithms for list decoding Folded Reed–Solomon codes and univariate multiplicity codes up to capacity in their natural parameter regimes. Unlike most known list decoding algorithms, these techniques appeared inherently tied to list decoding, and it was unclear whether they could be extended to list recovery in near-linear time.

In this work, we resolve this question by giving $\widetilde{O}(n)$-time algorithms for list recovery of Folded Reed–Solomon codes and univariate multiplicity codes up to capacity, where $n$ is the block length. Our algorithms build on the lattice-based framework of the prior work, augmented with a new technical ingredient: the construction of suitably structured lattices over the univariate polynomial ring that capture the list recovery problem for these codes.

**2012 ACM Subject Classification** Theory of computation → Error-correcting codes; Theory of computation → Design and analysis of algorithms

**Keywords and phrases** list-recovery, multiplicity-codes, near-linear-algorithm

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2026.67

**Category** RANDOM

**Related Version** *Full Version*: arXiv:2512.00248

**Funding** *Rohan Goyal*: 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).
*Prahladh Harsha*: Supported by the Department of Atomic Energy, Government of India, under project number RTI400112 and in part by a Google research grant.
*Mrinal Kumar*: Supported by the Department of Atomic Energy, Government of India, under project number RTI400112, and in part by research grants from Google, SERB and Premji Invest.
*Ashutosh Shankar*: Supported by the Department of Atomic Energy, Government of India, under project number RTI400112.

## 1 Introduction

An error-correcting code $\mathcal{C} \subseteq \Sigma^n$ is said to be $(\eta, L)$ list decodable if every Hamming ball of radius $\eta n$ in $\Sigma^n$ contains at most $L$ codewords of $\mathcal{C}$. When $\eta < \delta/2$, where $\delta$ is the minimum distance of the code, we are in the unique decoding regime, in which at most one codeword lies in such a ball. List decoding is of fundamental interest both from a combinatorial perspective, where one studies tradeoffs between parameters such as $\eta$, $L$, $\delta$, and the rate of the code, and from an algorithmic perspective, where the goal is to design efficient decoding algorithms for increasingly large values of $\eta$. Since Sudan's breakthrough work showing that Reed–Solomon

codes can be efficiently list decoded beyond the unique decoding radius [19], there has been substantial progress in our understanding of list decoding, culminating in explicit codes that achieve list decoding capacity with constant list size [9, 8, 10, 14, 15, 17, 5, 11] and, in several cases, near-linear time decoding algorithms [6, 2, 18]. Beyond coding theory, list decoding has deep and fruitful connections to complexity theory and pseudorandomness, including randomness extraction and hardness amplification [21].

Our focus in this paper is on the related notion of *list recovery*. A code $\mathcal{C} \subseteq \Sigma^n$ is said to be $(\eta, \ell, L)$ list recoverable if for all subsets $E_1, \ldots, E_n \subseteq \Sigma$ 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 $\eta n$ coordinates $i \in [n]$. Clearly, the case $\ell = 1$ recovers the definition of list decoding. List recovery has attracted considerable attention both because of its close connection to list decoding and because of its numerous applications, including the construction of pseudorandom objects such as extractors and expanders, cryptographic constructions, and algorithmic problems such as group testing and compressed sensing. We refer the reader to the survey of Resch and Venkitesh [16] and the lecture notes of Wootters [23] for further background.

A notable feature of many list decoding algorithms is that they naturally extend to the more general setting of list recovery. This phenomenon holds for the classical Sudan and Guruswami–Sudan algorithms for Reed–Solomon codes [19, 9], for algorithms achieving capacity for Folded Reed–Solomon (FRS) and (univariate and multivariate) multiplicity codes [8, 10, 14, 15, 3, 4], as well as for recent results on expander-based codes approaching list decoding capacity [11, 18, 12]. Consequently, one might expect fast list decoding algorithms to extend automatically to fast list recovery.

A central direction in the study of list decoding and list recovery is the design of near-linear time algorithms. This includes both the construction of codes admitting such fast decoding algorithms and the development of efficient algorithms for several well-studied and natural code families. A key milestone in this direction is the work of Alekhnovich [1], which gave a near-linear time algorithm for list decoding (and list recovery) Reed–Solomon codes up to the Johnson radius. Alekhnovich's algorithm can be viewed as a fast implementation of the Guruswami–Sudan algorithm that exploits natural connections between polynomial interpolation-based decoding algorithms and lattices over the univariate polynomial ring. More recently, a lattice-based framework building on these ideas was used to obtain near-linear time list decoding algorithms up to capacity for FRS codes and univariate multiplicity codes [6]. In addition, this work developed fast algorithms for solving certain families of differential and functional equations that arise naturally in this context.

Perhaps surprisingly, the techniques underlying these near-linear time list decoding results do not appear to extend directly to the setting of list recovery. This obstruction already arises for $\ell = 2$, and stands in contrast to the behavior of most known list decoding algorithms. In this work, we show that this limitation can in fact be overcome[1].

Informally, our main result gives a near-linear time algorithm for list recovery up to capacity. In the standard parameter regime for univariate multiplicity codes, where the list-size parameter $\ell$, the error parameter $\varepsilon$, and the multiplicity parameter $s$ are treated as constants, the running time of our algorithm is $\widetilde{O}(n)$.

More formally, we prove the following theorem.

▶ **Theorem 1** (main result for multiplicity codes). *For every $\varepsilon > 0$ and $\ell \in \mathbb{N}$, there exists $s_0 = \Theta(\ell/\varepsilon^2) \in \mathbb{N}$ such that for all $s > s_0$, degree parameter $k$, block length $n$, and fields $\mathbb{F}$ of characteristic zero or characteristic greater than $k$, satisfying $k < ns$, the following holds.*

[1] In this paper we focus on univariate multiplicity codes. We believe our results extend to FRS codes and defer that case to the full version.

*There is a randomized algorithm that, given a set $S \subseteq \mathbb{F}$ of size $n$ and sets $E_\alpha \subseteq \mathbb{F}^s$ with $|E_\alpha| \le \ell$ for each $\alpha \in S$, takes*

$$O\Big(n \cdot \mathsf{poly}\Big(s, \ell, \log n, (\ell/\varepsilon)^{\frac{1+\log \ell}{\varepsilon}}\Big)\Big)$$

*field operations and with high probability outputs all univariate polynomials $f(X) \in \mathbb{F}[X]$ of degree less than $k$ such that for at least $(\frac{k}{sn} + \varepsilon)$ fraction of $\alpha \in S$,*

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

## 1.1 An overview of the proof

We give a high-level overview of the proof, focusing on the list-recovery problem for univariate multiplicity codes. Our algorithm follows the same two-step structure as prior near-linear time list-decoding algorithms, but the interpolation step requires a fundamentally new construction.

The input to the list-recovery problem consists of a set $S \subseteq \mathbb{F}$ of size $n$, where each $\alpha \in S$ is associated with a list $\{\beta(\alpha)_j\}_{j\in[\ell]}$ of $s$-tuples in $\mathbb{F}^s$. As in the list-decoding setting, we refer to this input as the received word.

We begin by recalling the structure of the near-linear time list-decoding algorithm of [6], which is a fast implementation of the algorithm of Guruswami and Wang [10]. This algorithm has two main components:

- **Interpolation via a differential equation.** One constructs, for an appropriately chosen parameter $m = \Theta(\sqrt{s})$, a nonzero polynomial $Q(X, Y_0, \ldots, Y_m) = \widetilde{Q}(X) + \sum_i Q_i(X)Y_i$, linear in the $Y$-variables and of bounded degree, such that every polynomial $f$ whose encoding agrees sufficiently with the received word satisfies the differential equation:

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

- **Solving the differential equation.** The solution space of this linear differential equation is an affine space of dimension at most $m$. A near-linear time algorithm computes a basis for this space, followed by a pruning step [15, 20] to recover the constant-sized list of close enough codewords.

The fast list-recovery algorithm in this paper follows the same high-level outline. This is consistent with the polynomial-time list-recovery algorithm of Guruswami and Wang [10], where the list-decoding and list-recovery procedures are structurally identical, differing only in parameter settings.

To extend the results of [6] to the list-recovery setting, we use the fast differential equation solver from [6] as is. The main new ingredient is in the first step: constructing an appropriate differential equation in near-linear time.

We briefly recall the relevant interpolation framework from [6]. In the list-decoding setting, each $\alpha \in S$ is associated with a unique received vector $\beta(\alpha)$. The algorithm constructs in near-linear time univariate polynomials $A_0(X), A_1(X), \ldots, A_m(X)$ satisfying $(A_i)^{(j)}(\alpha) = \beta(\alpha)^{(i+j)}$ for all $\alpha \in S$, $i \in [m], j \in [s-m]$. One then considers the $\mathbb{F}[X]$-module (lattice) of polynomials in $X, Y_0, \ldots, Y_m$ generated by

$$\{\, Y_i - A_i(X) : i \in [m] \,\} \;\cup\; \left\{ \prod_{\alpha \in S} (X - \alpha)^{s-m} \right\}. \tag{1}$$

This lattice has the property that any nonzero element of this lattice of sufficiently small degree yields a polynomial $Q(X, Y_0, \ldots, Y_m) = \widetilde{Q}(X) + \sum_i Q_i(X)Y_i$ such that every polynomial $f$ whose encoding agrees with the received word on sufficiently many $\alpha$'s satisfies

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

In [6], existence of such a "short" nonzero lattice element is proved via an analogue of Minkowski's theorem, and it is found in near-linear time using Alekhnovich's algorithm for shortest vectors (with respect to the degree norm) in polynomial lattices. The "short" guarantee here matches that [10] achieve via variable and constraint counting.

The difficulty in moving to list recovery is that each $\alpha$ now comes with $\ell$ candidate vectors $\beta(\alpha)_1, \ldots, \beta(\alpha)_\ell$, so the received word is no longer a function $\alpha \mapsto \beta(\alpha)$. In particular, the definition of the polynomials $A_i$ above breaks down. A natural attempt is to arbitrarily "split" the input into $\ell$ received words in the list-decoding sense: for $t \in [\ell]$, let $r_t = (\alpha, \beta(\alpha)_t)_{\alpha \in S}$, and construct the corresponding polynomials $A_{t,0}, \ldots, A_{t,m}$. One could then try to work with the lattice generated by products such as $\prod_{t=1}^{\ell}(Y_j - A_{t,j}(X))$ (over $\mathbb{F}[X]$), thereby capturing the disjunction "$Y_j$ matches one of the $\ell$ possibilities." However, this immediately leads to polynomials $Q$ with $Y$-degree $\ell$, and hence to nonlinear differential equations. This is the main obstacle. Even if one could find such a low-degree $Q$ in near-linear time, it is unclear how to solve the resulting nonlinear differential equation and recover all close codewords efficiently: the solution set need not be an affine subspace, and the linear-algebraic techniques of [6] do not apply. While high $Y$-degree interpolation is standard in other contexts (e.g., [19, 9, 1, 14, 13]), we do not know how to make this approach work here[2].

Motivated by this, it is natural to insist on keeping $Q$ *linear* in the $Y$-variables even in the list-recovery setting. Guruswami and Wang show that this can be done in polynomial time. Our main technical contribution is to show that it can in fact be done in near-linear time: we construct a carefully structured lattice over $\mathbb{F}[X]$ whose generators are linear in $Y_0, \ldots, Y_m$, and whose short nonzero vectors correspond precisely to linear differential equations that *capture* all close codewords. In the list-decoding setting the received word is a function $\alpha \mapsto \beta(\alpha)$ leading to an almost-diagonal basis of the form (1) for the resulting $(m+1)$-dimensional lattice. In list recovery, the received word no longer has such a functional structure and hence one cannot expect the resulting lattice to admit a similar simple basis. What we show is the next best thing: there exists a lattice whose basis has a structured almost-triangular form, which is sufficient for computing short nonzero vectors efficiently and for deriving the required linear differential equation. The basis we obtain has the following form.

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

for some polynomials $\widetilde{C_i}, C_{i,r} \in \mathbb{F}[X]$ for $\ell - 1 \le i \le m$ and $0 \le r \le i - 1$. The proof that such a basis exists is considerably involved. We first show that such an almost-triangular

[2] Note that the Guruswami–Sudan list-decoding algorithm, and Alekhnovich's near-linear time implementation thereof, already employ a $Y$-degree larger than one; consequently, Alekhnovich's approach readily extends to list recovery.

basis exists for each $\alpha \in S$ and then combine the almost-triangular bases for the different $\alpha$ to obtain the basis of the above type using the Chinese remainder theorem. See Section 3 for details.

### Organization

The rest of this paper is organized as follows. Section 2 introduces univariate multiplicity codes and the subroutines used in the lattice construction. Section 3 gives the fast lattice construction and the resulting differential equation. Setting up this lattice is the key novelty of the paper. Section 4 describes the list-extraction procedure from the differential equation, completing the proof of Theorem 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 zero vector. We may drop the subscript if the length is clear from context.

### 2.2 Univariate multiplicity codes

We recall the formal definition of the univariate multiplicity code.

▶ **Definition 2** (Multiplicity codes). *Let $n, k, s$ be positive integers satisfying $k < ns$, $\mathbb{F}$ be a field of characteristic zero or at least $s$ 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, the univariate multiplicity code with multiplicity parameter $s$ and degree parameter $k$ 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 less than $k$ in $\mathbb{F}[x]$. 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[3] of $f$ on the input $\alpha$. The rate of this code is $k/ns$ while the distance is $1 - (k-1)/ns$.*

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

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

[3] This definition in terms of (standard) derivatives requires the characteristic of the field to be 0 or at least $k$. All known capacity-achieving list-decoding results for univariate multiplicity codes require large characteristic (previous works [14, 10, 15, 6] as well as our work). For this reason as well as ease of presentation, we work with standard derivatives.

## 2.3 Polynomial operators and lattices

We recall below the $\tau$ operator, used in [10, 13], and [6].

▶ **Definition 3** (Tau operator). *Let $s \in \mathbb{N}$ be a parameter. Then, for every $m$ such that $0 \leq m < s$, the function $\tau$ 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 $i \leq (s-m-1)$, we use $\tau^{(i)}$ to denote the linear map from $\mathbb{F}[X, Y_0, Y_1, \ldots, Y_m]$ to $\mathbb{F}[X, Y_0, Y_1, \ldots, Y_{m+i}]$ obtained by applying the operator $\tau$ iteratively $i$ times.*

The operator is defined this way so that

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

It will be convenient to view polynomials in $\mathbb{F}[X, Y_0, \ldots, Y_{m+i}]$ for $i \leq s-m-1$ as polynomials in $\mathbb{F}[X, Y_0, \ldots, Y_{s-1}]$. Thus, when evaluating a polynomial $R \in \mathbb{F}[X, Y_0, \ldots, Y_{m+i-1}]$ at the point $(\alpha, \beta) \in \mathbb{F}^{s+1}$, $R(\alpha, \beta)$ refers to $R(\alpha, \beta_{[m+i]}) = R(\alpha, \beta_0, \ldots, \beta_{m+i-1})$.

We will extensively use the following simple property of the $\tau$ operator.

▶ **Proposition 4.** *Let $\alpha$ be a field element and $\beta = (\beta_0, \ldots, \beta_i, \ldots, \beta_{s-1}) \in \mathbb{F}^s$. Then, for any integer $j$ with $1 \leq j \leq s - m$,*

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

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

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

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

▶ **Definition 5** (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 $\mathbb{F}$. A basis $B$ can be viewed as an $m \times \ell$ matrix (the $\ell$ basis vectors written side by side as columns).*

*For a full-rank lattice (i.e., when $\ell = m$), the determinant of a basis $\mathcal{B}$ refers to the determinant of this matrix Since this quantity is independent of the choice of basis, it is also referred to as the determinant of the lattice, denoted by* $\det \mathcal{L}$.

*Let $\beta = (\beta(1), \beta(2), \ldots, \beta(m)) \in \mathbb{F}[X]^m$ be a vector of polynomials. We define the max-degree norm as follows $\beta := \max_{j \in [m]} \deg \beta(j)$.*

▶ **Theorem 6** (polynomial version of Minkowski's theorem)**.** *Let $\mathcal{L}$ be an $m$-dimensional polynomial lattice. Then, there is a nonzero vector $\mathbf{v}$ in the lattice $\mathcal{L}$ 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 [6].

▶ **Theorem 7** ([1, Theorem 2.1],[7, 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 $\omega$ is the exponent in the complexity of matrix multiplication.*

Here "shortest vector" is in the max-degree norm. See [6] for details.

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

▶ **Theorem 8** ([22, 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 $O(d \mathsf{poly} \log(d))$.*

## 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 this can be done in near-linear time. Below is the main theorem for this section.

▶ **Theorem 9.** *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 nonzero 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,*

$$Q\left(X, f(X), f^{(1)}(X), \ldots, f^{(m)}(X)\right) \equiv 0. \tag{2}$$

*Furthermore, there is a deterministic algorithm that takes as input $R$ and returns $Q$ as a list of coefficients that carries out $\widetilde{O}(n\mathsf{poly}(s+m+\ell))$ field operations – 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 (2), is to ensure that for each $\alpha \in S$, we have the following conditions

$$\tau^{(t)}(Q)\left(\alpha, \beta_j^0, \beta_j^1, \ldots, \beta_j^{m+t}\right) = 0, \qquad \forall t \in [s-m],\ j \in [\ell]. \tag{3}$$

This will ensure that every close enough codeword satisfies (2) (see Lemma 14 for exact details). We will refer to these conditions as the "$\tau$-conditions". The set of polynomials $Q$ satisfying (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 9, we will first construct a structured 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 structured basis for a sub-lattice of the lattice of polynomials satisfying the "$\tau$-conditions".

▶ **Proposition 10.** *Let $\ell$ 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 $m \le s-1$ be a parameter.*

*Then, there exist $m+2$ polynomials in $\mathbb{F}[X, Y_0, \ldots, Y_m]$ satisfying the $\tau$-conditions* (3) *for all $\alpha \in S$ of the following type*

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

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

The analogue of Proposition 10 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 $\ell$ is more than 1. In order to address this, we first show that a version of Proposition 10 can be proved for each $\alpha \in S$ and then combine these different bases for the different $\alpha$ into a single basis that works for all the $\alpha$. The following proposition is the version of Proposition 10 for a single $\alpha \in S$ (which is proved in Section 3.1).

▶ **Proposition 11.** *Let $\alpha \in S$ and $\ell$ be the list size and let $m \le s-1$ be a parameter.*

*Given $\ell$ 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 $\alpha$, there exist $m+2$ polynomials in $\mathbb{F}[X, Y_0, \ldots, Y_m]$ satisfying the $\tau$-conditions* (3) *for this specific $\alpha \in S$ of the following type*

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

*for some polynomials* $\widetilde{C_i}, C_{i,r} \in \mathbb{F}[X]$ *for* $\ell-1 \le i \le m$ *and* $0 \le r \le i-1$. *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, carrying out* $\widetilde{O}(\ell(s-m)^2)$ *field operations.*

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 Proposition 11 for each $\alpha \in S$ to obtain a basis that works simultaneously for all $\alpha \in S$ to yield Proposition 10.

## 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 11. More precisely, we want to show that there exist $m+2$ polynomials of the following type that satisfy the $\tau$-conditions for $\alpha$. Since $\alpha$ will be fixed throughout this section, we will drop $\alpha$ 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 $\beta = (\beta^{(0)}, \beta^{(1)}, \ldots, \beta^{(s-1)})$ we will find it convenient to use the following notation for the vector in $\mathbb{F}^{s-m}$ composed of evaluations of iterated $\tau$-operator of the polynomial $\mathbf{B}$ at the point $(\alpha, \beta)$:

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

The linearity of the $\tau$ operator shows that the function $\tau(\cdot,\cdot)$ is linear in its first argument.

**Proof of Proposition 11.** In the notation defined above, we want to show that $\tau(\mathbf{B}, \beta) = \mathbf{0}_{s-m}$ for all $\beta \in \{\beta_j : j \in [\ell]\}$ 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 $\ell$, the list size.

**Base case $\ell = 1$:** This is the list-decoding setting, where we have only $\beta$ in the list corresponding to $\alpha$, namely $\beta_0 = (\beta_0^{(0)}, \beta_0^{(1)}, \ldots, \beta_0^{(s-1)})$. The proof in this case is similar to the corresponding step in [6], 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 $\ell - 1 = 0 \le i \le m$. $\mathbf{B}_i$ clearly satisfies the $\tau$-conditions (3) for $\alpha$.

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

Fix an $i$ such that $\ell - 1 \le i \le m$. 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 4, we have that $\tau(\mathbf{B}''_i, \beta_j) = \mathbf{0}$ for all $j \in [\ell-1]$. Now, if $\tau(\mathbf{B}''_i, \beta_{\ell-1})$ is also $\mathbf{0}$, we can set $\mathbf{B}_i := \mathbf{B}''_i$ and be done.

Suppose $\tau(\mathbf{B}''_i, \beta_{\ell-1}) \neq \mathbf{0}$. Then, let $r_* \in [s-m]$ be the least $r$ such that $\tau^{(<r)}(\mathbf{B}''_i)(\alpha, \beta_{\ell-1}) = \mathbf{0}_r$ but $\tau^{(r)}(\mathbf{B}''_i)(\alpha, \beta_{\ell-1}) \neq 0$. By Proposition 4, definition of $\mathbf{B}''_i$ and the fact that $s-m$ is less than the characteristic of $\mathbb{F}$, we also have $\tau^{(<r_*)}(\mathbf{B}'_{i-1})(\alpha, \beta_{\ell-1}) = \mathbf{0}_{r_*}$ but $\tau^{(r_*)}(\mathbf{B}'_{i-1})(\alpha, \beta_{\ell-1}) \neq 0$. Let us consider the $(s-m-r_*)$ polynomials $\{(X-\alpha)^t \cdot \mathbf{B}'_{i-1} \colon t \in [s-m-r_*]\}$. The $t$-th polynomial in this set, namely $(X-\alpha)^t \cdot \mathbf{B}'_{i-1}$ satisfies $\tau^{(<r_*+t)}((X-\alpha)^t \cdot \mathbf{B}'_{i-1})(\alpha, \beta_{\ell-1}) = \mathbf{0}_{r_*+t}$ but $\tau^{(r_*+t)}((X-\alpha)^t \cdot \mathbf{B}'_{i-1})(\alpha, \beta_{\ell-1}) \neq 0$. Hence, the $(s-m-r_*)$ vectors $\{\tau\left((X-\alpha)^t \cdot \mathbf{B}'_{i-1}, \beta_{\ell-1}\right) : t \in [s-m-r_*]\}$ 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 $\tau(\mathbf{B}''_i, \beta_{\ell-1})$ and hence there exist scalars $c_t \in \mathbb{F}$ for $t \in [s-m-r_*]$ such that

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

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 $\tau(\mathbf{B}_i, \beta_{\ell-1}) = \mathbf{0}$. Since $\tau(\mathbf{B}''_i, \beta_j) = \mathbf{0}$ and $\tau(\mathbf{B}'_{i-1}, \beta_j) = \mathbf{0}$ for all $j \in [\ell-1]$, we also have that $\tau(\mathbf{B}_i, \beta_j) = \mathbf{0}$ for all $j \in [\ell-1]$. Combining, we have that $\tau(\mathbf{B}_i, \beta_j) = \mathbf{0}$ 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 with $O(s-m)^2$ operations; giving an overall total of $\widetilde{O}(\ell(s-m)^2)$ operations. This concludes the proof of the proposition. ◀

## 3.2 Combining the bases

In the previous section, we constructed a structured 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 $p_\alpha(X) = (X-\alpha)^{s-m}$ for multiplicity codes.

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

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

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[4] 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:

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

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 nonzero factors of the polynomial $P_{\mathcal{U}}$ and similarly $\widetilde{V}, V_{i,i}, i \in [m]$ are nonzero 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 a 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:

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

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 $\mathbf{W}_i \equiv V_{i,i}(X) \cdot \mathbf{U}_i \bmod P_{\mathcal{U}}$ and symmetrically, $\mathbf{W}_i \equiv U_{i,i}(X) \cdot \mathbf{V}_i \bmod P_{\mathcal{V}}$.

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

[4] 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 12. If $\mathbf{H} = \widetilde{H}(X) + \sum_{i\in[m]} Y_i \cdot H_i(X)$ or equivalently $(\widetilde{H}, H_0, H_1, \ldots, H_m)$ satisfies $\mathbf{H} \bmod P_{\mathcal{U}} \in \mathcal{M}_{\mathcal{U}}$ and $\mathbf{H} \bmod P_{\mathcal{V}} \in \mathcal{M}_{\mathcal{V}}$, then $\mathbf{H} \bmod P_{\mathcal{W}} \in \mathcal{M}_{\mathcal{W}}$.

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

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

Define polynomials

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

We claim that

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

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 $\mathbf{H} \bmod P_{\mathcal{U}}$. The case mod $P_{\mathcal{V}}$ is symmetric and the equivalence follows from the Chinese remainder theorem. ◁

For the specific setting of multiplicity codes, 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 $\ell$ is the list size.
2. Every basis element $\mathbf{U} \in \mathcal{B}_{\mathcal{U}}$ satisfies the $\tau$-conditions (3) for every $\alpha \in \mathcal{U}$.

Clearly, the basis generated by Proposition 11 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 13. 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 $\mathcal{B}_{\mathcal{W}}$, carrying out $\widetilde{O}(\max\{|\mathcal{U}|, |\mathcal{V}|\}m^2(s-m))$ field operations.

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 the Chinese remainder theorem (Theorem 8): 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 $\tau$-conditions for all $\alpha \in \mathcal{W}$. Suppose $\mathbf{W} = \mathbf{W}_i$ for some $0 \leq i \leq m$. 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 $V_{i,i} \cdot \mathbf{U}_i + \prod_{\alpha \in \mathcal{U}} (X - \alpha)^{s-m} \cdot \mathbf{H}$ for some polynomial $\mathbf{H} = \widetilde{H} + \sum_{i=0}^{m} Y_i \cdot H_i(X)$. Since $\mathcal{B}_{\mathcal{U}}$ is $\mathcal{U}$-good, $\mathbf{U}_i$ satisfies the $\tau$-conditions with respect to $\alpha$. 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 10.

**Proof of Proposition 10.** 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 11.

It follows from Claim 13, 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 10. 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 13 states that the bases for the two instances can be combined in time $\widetilde{O}(nm^2(s-m))$. Finally, Proposition 11 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 9

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

▶ **Lemma 14.** *Let $Q(X, Y_0, Y_1, \ldots, Y_m) \in \mathcal{M}_S$ have $X$-degree at most $D \leq n\ell(s-m)/m + n\ell$ 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, $Q(X, f(X), f^{(1)}(X), \ldots, f^{(m)}(X)) \equiv 0$.*

**Proof.** Let $P(X) = Q(X, f(X), f^{(1)}(X), \ldots, f^{(s-1)}(X))$. Since $f$ has degree less than $k$ and $Q$ is linear in the $Y$-variables, $P(X)$ has degree less than $D + k$. The definition of the $\tau$ 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 Proposition 10 implies that for an agreement point $\alpha$,

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

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 at least $(D+k)/(s-m)$, $P$ must be identically zero. ◀

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

**Proof of Theorem 9.** 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 10. Since each of them satisfies the $\tau$-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 6, $\mathcal{M}_S$ 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 $\ell$ 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 14. 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 10 and Theorem 7, 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 for multiplicity codes

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

### 4.1 Fast algorithms for solving differential equations

We will invoke the following algorithms for solving differential equations from [6]. (Their paper also contains an analogous result for functional equations, for FRS codes.)

▶ **Theorem 15** ([6, Theorem 1.6]). *Let $\mathbb{F}$ be a finite field of characteristic greater than $d$ or zero, and let*

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

*be a nonzero 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 of 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 16. The [6, Theorem 1.6] paper presents a complexity of $\widetilde{O}((D+d)\mathsf{poly}(m))$. 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 [6, Algorithm 1]. The time complexity for this step is analyzed in [6, 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 [15], which was simplified by [20], to obtain the codewords. The list-size bounds we present below are due to [20].

▶ **Theorem 17** ([15, Lemma 3.1], [20, 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 $\widetilde{O}(n)\mathsf{poly}(\log q, s, L(\delta, \varepsilon, \ell, r), \log(1/\gamma))$ 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 18.** *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 $O(n \log|\mathbb{F}|(\ell(s^2 + sm^3) + \frac{k}{n}m^4)\mathsf{polylog}(n, s, \log\log|\mathbb{F}|))$ 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 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 9.
2. Solve the resulting differential equation using the algorithm in Theorem 15, to obtain an affine space $A$ containing all the codewords.

Observe that by Theorem 9, 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 15 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** (main result for multiplicity codes). *For every $\varepsilon > 0$ and $\ell \in \mathbb{N}$, there exists $s_0 = \Theta(\ell/\varepsilon^2) \in \mathbb{N}$ such that for all $s > s_0$, degree parameter $k$, block length $n$, and fields $\mathbb{F}$ of characteristic zero or characteristic greater than $k$, satisfying $k < ns$, the following holds. There is a randomized algorithm that, given a set $S \subseteq \mathbb{F}$ of size $n$ and sets $E_\alpha \subseteq \mathbb{F}^s$ with $|E_\alpha| \leq \ell$ for each $\alpha \in S$, takes*

$$O\Big(n \cdot \mathsf{poly}\Big(s, \ell, \log n, (\ell/\varepsilon)^{\frac{1+\log \ell}{\varepsilon}}\Big)\Big)$$

*field operations and with high probability outputs all univariate polynomials $f(X) \in \mathbb{F}[X]$ of degree less than $k$ such that for at least $(\frac{k}{sn} + \varepsilon)$ 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 9 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 18.
2. Using the algorithm in Theorem 17, 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 18, $f$ is contained in the output affine space. The algorithm in Theorem 17 finds $f$ with high probability.

We saw in Theorem 18 that step 1 can be done in $\widetilde{O}(n\mathsf{poly}(s + m + \ell))$ time. Combining with the runtime of Theorem 17, we get a runtime of $\widetilde{O}\left(n\mathsf{poly}(s, \ell, (\ell/\varepsilon)^{\frac{1+\log \ell}{\varepsilon}})\right)$. ◀

## References


1 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). doi:10.1109/TIT.2005.850097.

2 Vikrant Ashvinkumar, Mursalin Habib, and Shashank Srivastava. Algorithmic improvements to list decoding of folded Reed-Solomon codes. In Kasper Green Larsen and Barna Saha, editors, *Proc. 37th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 880–898, 2026. doi:10.1137/1.9781611978971.35.

3 Siddharth Bhandari, Prahladh Harsha, Mrinal Kumar, and Madhu Sudan. Decoding multivariate multiplicity codes over product sets. *IEEE Trans. Inform. Theory*, 70(1):154–169, 2024. (Preliminary version in *53rd STOC*, 2021). doi:10.1109/TIT.2023.3306849.

4 Joshua Brakensiek, Yeyuan Chen, Manik Dhar, and Zihan Zhang. Combinatorial bounds for list recovery via discrete Brascamp–Lieb inequalities. In Aditya Bhaskara and Artur Czumaj, editors, *Proc. 58th ACM Symp. on Theory of Computing (STOC)*, pages 365–376. ACM, 2026. doi:10.1145/3798129.3800756.

5 Yeyuan Chen and Zihan Zhang. Explicit folded Reed-Solomon and multiplicity codes achieve relaxed generalized singleton bound. In Michal Koucký and Nikhil Bansal, editors, *Proc. 57th ACM Symp. on Theory of Computing (STOC)*, pages 1–12, 2025. doi:10.1145/3717823.3718114.

6 Rohan Goyal, Prahladh Harsha, Mrinal Kumar, and Ashutosh Shankar. Fast list-decoding of univariate multiplicity and folded Reed-Solomon codes. In Santosh Vempala, editor, *Proc. 65th IEEE Symp. on Foundations of Comp. Science (FOCS)*, pages 328–343, 2024. doi:10.1109/FOCS61266.2024.00028.

7 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. doi:10.1016/J.JSC.2011.09.006.

8 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). doi:10.1109/TIT.2007.911222.

9 Venkatesan Guruswami and Madhu Sudan. Improved decoding of Reed-Solomon and algebraic-geometry codes. *IEEE Trans. Inform. Theory*, 45(6):1757–1767, 1999. (Preliminary version in *39th FOCS*, 1998). doi:10.1109/18.782097.

10 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). doi:10.1109/TIT.2013.2246813.

11 Fernando Granha Jeronimo, Tushant Mittal, Shashank Srivastava, and Madhur Tulsiani. Explicit codes approaching generalized Singleton bound using expanders. In Michal Koucký and Nikhil Bansal, editors, *Proc. 57th ACM Symp. on Theory of Computing (STOC)*, pages 843–854, 2025. doi:10.1145/3717823.3718302.

12 Fernando Granha Jeronimo and Nikhil Shagrithaya. Probabilistic guarantees to explicit constructions: Local properties of linear codes. In Aditya Bhaskara and Artur Czumaj, editors, *Proc. 58th ACM Symp. on Theory of Computing (STOC)*, pages 806–813, 2026. doi:10.1145/3798129.3800795.

13 Itay Kalev and Amnon Ta-Shma. Unbalanced expanders from multiplicity codes. In Amit Chakrabarti and Chaitanya Swamy, editors, *Proc. 26th International Conf. on Randomization and Computation (RANDOM)*, volume 245 of *LIPIcs*, pages 12:1–12:14. Schloss Dagstuhl, 2022. doi:10.4230/LIPIcs.APPROX/RANDOM.2022.12.

14 Swastik Kopparty. Some remarks on multiplicity codes. In Alexander Barg and Oleg R. Musin, editors, *Discrete Geometry and Algebraic Combinatorics*, volume 625 of *Contemporary Mathematics*, pages 155–176. AMS, 2014. doi:10.1090/conm/625/12497.

15 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). doi:10.1137/20M1370215.

16 Nicolas Resch and S. Venkitesh. List recoverable codes: The good, the bad, and the unknown (hopefully not ugly). (manuscript).

17 Shashank Srivastava. Improved list size for folded Reed-Solomon codes. In Yossi Azar and Debmalya Panigrahi, editors, *Proc. 36th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 2040–2050, 2025. doi:10.1137/1.9781611978322.64.

18 Shashank Srivastava and Madhur Tulsiani. List decoding expander-based codes up to capacity in near-linear time. In Rotem Oshman and Ran Raz, editors, *Proc. 66th IEEE Symp. on Foundations of Comp. Science (FOCS)*, pages 1465–1487, 2025. doi:10.1109/FOCS63196.2025.00077.

19 Madhu Sudan. Decoding of Reed-Solomon codes beyond the error-correction bound. *J. Complexity*, 13(1):180–193, 1997. (Preliminary version in *37th FOCS*, 1996). doi:10.1006/jcom.1997.0439.

20 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. doi:10.1109/TIT.2024.3402171.

21 Salil P. Vadhan. Pseudorandomness. *Found. Trends Theor. Comput. Sci.*, 7(1-3):1–336, 2012. doi:10.1561/0400000010.

22 Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, 3 edition, 2013. doi:10.1017/CBO9781139856065.

23 Mary Wootters. CS250/EE387: Algebraic error correcting codes, 2019. (A course on coding theory at Stanford, Winter 2019). URL: https://web.stanford.edu/~marykw/classes/CS250_W19/index.html.