Electronic Colloquium on Computational Complexity, Revision 2 of Report No. 91 (2018)

# Improved List Decoding of Folded Reed-Solomon and Multiplicity Codes[*]

Swastik Kopparty[†] Noga Ron-Zewi[‡] Shubhangi Saraf[§] Mary Wootters[¶]

February 22, 2023

## Abstract

We show new and improved list decoding properties of folded Reed-Solomon (RS) codes and multiplicity codes. Both of these families of codes are based on polynomials over finite fields, and both have been the sources of recent advances in coding theory: Folded RS codes were the first known explicit construction of *capacity-achieving* list decodable codes (Guruswami and Rudra, *IEEE Trans. Information Theory*, 2010), and multiplicity codes were the first construction of *high-rate* locally decodable codes (Kopparty, Saraf, and Yekhanin, *J. ACM*, 2014).

In this work, we show that folded RS codes and multiplicity codes are in fact better than was previously known in the context of list decoding and local list decoding. Our first main result shows that folded RS codes achieve list decoding capacity with *constant* list sizes, independent of the block length. Prior work with constant list sizes first obtained list sizes that are polynomial in the block length, and relied on pre-encoding with subspace evasive sets to reduce the list sizes to a constant (Guruswami and Wang, *IEEE Trans. Information Theory*, 2012; Dvir and Lovett, *STOC*, 2012). The list size we obtain is $(1/\varepsilon)^{O(1/\varepsilon)}$ where $\varepsilon$ is the gap to capacity, which matches the list size obtained by pre-encoding with subspace evasive sets.

For our second main result, we observe that *univariate* multiplicity codes exhibit similar behavior, and use this, together with additional ideas, to show that *multivariate* multiplicity codes are *locally* list decodable *up to their minimum distance*. By known reductions, this gives in turn capacity-achieving locally list decodable codes with query complexity $\exp(\tilde{O}((\log N)^{5/6}))$. This improves on the tensor-based construction of (Hemenway, Ron-Zewi, and Wootters, *SICOMP*, 2019), which gave capacity-achieving locally list decodable codes of query complexity $N^{\tilde{O}(1/\log\log N)}$, and is close to the best known query complexity of $\exp(\tilde{O}(\sqrt{\log N}))$ for high-rate locally (uniquely) decodable codes (Kopparty, Meir, Ron-Zewi, and Saraf, *J. ACM*, 2017).

[*]This is an updated and revised version of an extended abstract that has appeared at FOCS 2018 (see [KRSW18] for a full version). The current version contains some improvements and simplifications over the previous version. For simplicity and clarity, we have also decided to omit some of the results from [KRSW18].

[†]Department of Mathematics and Department of Computer Science, Rutgers University. Research supported in part by NSF grants CCF-1253886 and CCF-1540634. `swastik.kopparty@gmail.com`.

[‡]Department of Computer Science, Haifa University. `noga@cs.haifa.ac.il`. Research supported in part by BSF grants 2017732 and 2021683 and ISF grant 735/20.

[§]Department of Mathematics and Department of Computer Science, Rutgers University. Research supported in part by NSF grants CCF-1350572 and CCF-1540634. `shubhangi.saraf@gmail.com`.

[¶]Department of Computer Science and Department of Electrical Engineering, Stanford University. `marykw@stanford.edu`. Research supported in part by NSF grants CCF-1657049, CCF/BSF-1814629, and CAREER grant CCF-1844628, and by a Sloan Research Fellowship.

ISSN 1433-8092

## 1 Introduction

An error correcting code $C \subset \Sigma^n$ is a collection of *codewords* $c$ of length $n$ over an alphabet $\Sigma$. The goal in designing $C$ is to enable the recovery of a codeword $c \in C$ given a corrupted version $\tilde{c}$ of $c$, while at the same time making $C$ as large as possible. In the classical unique decoding problem, the goal is to efficiently recover $c$ from any $\tilde{c} \in \Sigma^n$ so that $c$ and $\tilde{c}$ differ in at most $\alpha n$ places; this requires the *relative distance* $\delta$ of the code (that is, the minimum fraction of places on which any two codewords differ) to exceed $2\alpha$.

Modern applications of error correcting codes, both in coding theory and theoretical computer science, have highlighted the importance of variants of the unique decoding problem, including *list decoding,* and *local decoding.* In list decoding, the amount of error $\alpha$ is large enough that unique recovery of the codeword $c$ is impossible (that is, $\alpha > \delta/2$), and instead the goal is to return a short list $\mathcal{L} \subset C$ with the guarantee that $c \in \mathcal{L}$. In local decoding, we still have $\alpha < \delta/2$, but the goal is to recover a single symbol $c_i$ of a codeword $c$, after querying not too many positions of the corrupted codeword $\tilde{c}$. In a variant known as *local list decoding,* we seek local information about a symbol even when $\alpha > \delta/2$. List-decoding, local decoding, and local list decoding are important primitives in error correcting codes, with applications in coding theory and theoretical computer science, for example in cryptography [GL89], learning theory [KM93], and hardness amplification and derandomization [STV01].

Algebraic codes have been at the heart of the study of list decoding, local-decoding and local list decoding. One classical example of this is Reed-Solomon (RS) codes, whose codewords are comprised of evaluations of low-degree polynomials.[1] In the late 1990's, Guruswami and Sudan [Sud97, GS99] gave an algorithm for efficiently list decoding Reed-Solomon codes well beyond half the distance of the code, and this kicked off the field of algorithmic list decoding. A second example is Reed-Muller (RM) codes, the multivariate analogue of Reed-Solomon codes. The structure of Reed-Muller codes is very amenable to local algorithms: a codeword of a Reed-Muller code corresponds to a multivariate low-degree polynomial, and considering the restriction of that polynomial to a line yields a univariate low-degree polynomial, *a.k.a.* a Reed-Solomon codeword. This local structure is the basis for Reed-Muller codes being locally testable [RS96] and locally decodable [Lip90, BFLS91]. Using this locality in concert with the Guruswami-Sudan algorithm leads to local list decoding algorithms for these codes [AS03, STV01].

More recently, variants of Reed-Solomon and Reed-Muller codes have emerged to obtain improved list decoding and local-decoding properties. Two notable examples, which are the focus of this work, are *Folded Reed-Solomon* (FRS) *codes* and *multiplicity codes.* Both of these constructions have led to recent advances in coding theory. We introduce these codes informally here, and give formal definitions in Section 2.

Folded Reed-Solomon codes, introduced by Krachkovsky in [Kra03], are a simple variant of Reed-Solomon codes. If the codeword of a Reed-Solomon code is $(c_0, c_2, \ldots, c_{n-1}) \in \Sigma^n$, then the

[1] That is, a codeword of an RS code has the form $(f(x_0), f(x_1), \ldots, f(x_{n-1})) \in \mathbb{F}^n$ for some low-degree polynomial $f \in \mathbb{F}[X]$.

codeword of the folded version (with *folding parameter* $s$) is

$$\left( \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{s-1} \end{bmatrix}, \begin{bmatrix} c_s \\ c_{s+1} \\ \vdots \\ c_{2s-1} \end{bmatrix}, \ldots, \begin{bmatrix} c_{n-s} \\ c_{n-s+1} \\ \vdots \\ c_{n-1} \end{bmatrix} \right) \in (\Sigma^s)^{n/s}.$$

The main property of these codes, that makes them interesting is that they admit much better list decoding algorithms than the original Guruswami-Sudan algorithm: more precisely, it was shown by Guruswami and Rudra [GR08] that it allows for the error tolerance $\alpha$ to be much larger for a code of the same rate,[2] asymptotically obtaining the *optimal* trade-off.

Multiplicity codes, introduced in the univariate setting by Rosenbloom and Tsfasman in [RT97] and in the multivariate setting by Kopparty, Saraf and Yekhanin in [KSY14], are variants of polynomial codes that also include evaluations of derivatives. That is, while a symbol of a RS codeword is of the form $f(x) \in \mathbb{F}_q$ for some low-degree polynomial $f \in \mathbb{F}[X]$ and some $x \in \mathbb{F}_q$, a symbol in a univariate multiplicity code codeword is of the form $(f(x), f^{(1)}(x), f^{(2)}(x), \ldots, f^{(s-1)}(x)) \in \mathbb{F}_q^s$, where $s$ is the *multiplicity parameter*. Similarly, while a symbol of an RM codeword is of the form $f(\mathbf{x})$ for $\mathbf{x} \in \mathbb{F}_q^m$ for some low-degree multivariate polynomial $f \in \mathbb{F}[X_1, \ldots, X_m]$, a symbol in a multivariate multiplicty code includes all partial derivatives of order less than $s$. Multivariate multiplicity codes were shown in [KSY14] to have strong locality properties, and were the first constructions known of *high-rate* locally decodable codes. Meanwhile, univariate multiplicity codes were shown in [Kop15, GW13] to be list decodable in the same parameter regime as folded Reed-Solomon codes[3], also achieving asymptotically optimal trade-off between rate and error-tolerance.

In this work, we show that Folded Reed-Solomon codes, univariate multiplicity codes, and multivariate multiplicity codes are even more powerful than was previously known in the context of list decoding and local list decoding. Our motivations for this work are threefold:

1. First, FRS codes and multiplicity codes are basic and natural algebraic codes, central to many recent results in coding theory ([GR08, KSY14, Kop15, GW13, DL12, KMRS17], to name a few), and understanding their error-correcting properties is important in its own right.

2. Second, while there have been improved constructions of list decodable and locally list decodable codes building on FRS and multiplicity codes (discussed more below), those constructions involve significant additional pseudorandom ingredients. As a consequence, these constructions eliminate many of the original structural properties of the codes, such as linearity. Our results give simpler and more structured constructions of capacity-achieving list decodable and locally list decodable codes with the best known parameters.

3. Third, by composing our new results with known reductions, we obtain capacity-achieving locally list decodable codes with significantly improved query complexity than was previously known.

[2]The *rate* of a code $C \subseteq \Sigma^n$ is defined as $R = \frac{1}{n}\log_{|\Sigma|}(|C|)$ and quantifies how much information can be sent using the code. We always have $R \in (0,1)$, and we would like $R$ to be as close to 1 as possible.

[3]Univariate multiplicity codes were previously shown to be list decodable up to the Johnson bound by Nielsen [Nie01].

In what follows, we state our results and contributions more precisely, and provide a more detailed account of related work.

## 1.1 Our results

We start by fixing some notation. Let $C \subseteq \Sigma^n$ be a code. We define the *rate* of $C$ as $R := \frac{1}{n}\log_{|\Sigma|}(|C|)$. The *relative distance* $\delta$ of $C$ is the minimum fraction of coordinates on which any two codewords of $C$ differ. The code $C$ is $(\alpha, L)$-*list decodable* if for any received word $w \in \Sigma^n$, there are at most $L$ codewords $c \in C$ which satisfy that $c_i = w_i$ for all but an $\alpha$ fraction of the coordinates $i$.

### 1.1.1 Improved list decoding of FRS codes

The celebrated Guruswami-Sudan list decoding algorithm [Sud97, GS99], mentioned above, can efficiently list decode Reed-Solomon codes up to radius $\alpha = 1 - \sqrt{1-\delta}$, with polynomial list sizes $L$; this radius is known as the *Johnson bound*, and as Reed-Solomon codes satisfy $R = 1 - \delta$, this amounts to a decoding radius of $\alpha = 1 - \sqrt{R}$. It is a classical result that there are codes that go beyond a decoding radius of $1 - \sqrt{R}$, while keeping the list size polynomial in $n$, or even constant: for large alphabet sizes, the "correct" limit, called the *list decoding capacity*, is $\alpha = 1 - R$, and this is achieved by uniformly random codes. More precisely, it is well-known that a random code of rate $R$ and alphabet size $\exp(1/\varepsilon)$ is $(1 - R - \varepsilon, O(1/\varepsilon))$-list decodable [Eli91]. For a decade it was open whether or not one could construct explicit codes which efficiently achieve list decoding capacity.

In a breakthrough result, Guruswami and Rudra [GR08] (building on the work of Parvaresh and Vardy [PV05]) showed that the folding operation described above can make RS codes approach capacity with polynomial list-sizes. More precisely, they showed that an FRS code of rate $R$ and folding parameter $s \approx 1/\varepsilon^2$ is $\left(1 - R - \varepsilon, n^{O(1/\varepsilon)}\right)$-list decodable. Note that the list size is polynomial in $n$ for any constant $\varepsilon > 0$, while ideally one could expect the list size to be independent of $n$.

Towards reducing the list size, it was later shown by Guruswami and Wang [GW13] that, surprisingly, the list of FRS is contained in a linear subspace of *constant* dimension $O(1/\varepsilon)$.[4] Note that this still does not give an improvement on the list size, as the alphabet size of FRS is at least $n$. To reduce the list size to a constant, [GW13] proposed to pre-encode these codes with pseudorandom objects called *subspace evasive sets* that intersect constant dimensional subspaces in a constant number of points. They further showed that such objects exist non-explicitly, and posed the question of searching for an explicit construction. Subsequently, their program was carried out by Dvir and Lovett [DL12] who gave an explicit construction of subspace evasive sets of co-dimension at most $\varepsilon n$ that intersect any $t$-dimensional subspace in at most $(t/\varepsilon)^{O(t)}$ points, which led to a list size of $(1/\varepsilon)^{O(1/\varepsilon)}$. Note that subspace evasive sets are inherently non-linear, and so the resulting code is non-linear as well.

In this work, we show that in fact FRS codes are *already* list decodable with constant list-sizes,

[4]Many codes in this paper have alphabet $\Sigma = \mathbb{F}_q^s$, where $\mathbb{F}_q$ is a finite field. For a subset $V \subseteq \Sigma^n$, we use the term "linear" to mean "$\mathbb{F}_q$-linear", and we measure the dimension over $\mathbb{F}_q$.

with no additional modification needed! Interestingly, the list size we obtain is $(1/\varepsilon)^{O(1/\varepsilon)}$, which matches the list size obtained via the subspace evasive approach. In particular, this gives the first construction of *linear* capacity-achieving list decodable codes with constant list size.

**Theorem 1.1** (List decoding FRS with constant list size (Informal, see Theorem 3.11))**.** *Let $C$ be an* FRS *code of constant rate $R \in (0,1)$ and folding parameter $s \geq \frac{16}{\varepsilon^2}$. Then $C$ is $\left(1 - R - \varepsilon, (1/\varepsilon)^{O(1/\varepsilon)}\right)$-list decodable.*

As mentioned above, it is known that it is possible for capacity-achieving list decodable codes to achieve a list size of $O(1/\varepsilon)$, and it would be very interesting to strengthen the above theorem to this bound.

### 1.1.2 Improved local list decoding of multiplicity codes

For some time, FRS codes were the only known route to capacity-achieving list decodable codes, until it was shown in [GW13, Kop15] that univariate multiplicity codes also do the job (again, with polynomial list sizes). We first observe that our approach to prove Theorem 1.1 above also applies to univariate multiplicity codes, albeit with a larger (but still constant) list size.

**Theorem 1.2** (List decoding univariate multiplicity codes with constant list size (Informal, see Theorem 3.8))**.** *Let $C$ be a univariate multiplicity code of constant rate $R \in (0,1)$ and multiplicity parameter $s \geq \frac{16}{\varepsilon^2}$, defined over a sufficiently large prime field $\mathbb{F}_q$. Then $C$ is $\left(1 - R - \varepsilon, (s/\varepsilon)^{O(s/\varepsilon^2)}\right)$-list decodable.*

Our second main result uses the above theorem (together with some additional ideas) to obtain improved *local* list decoding algorithms for *multivariate* multiplicity codes. The definition of local list decoding (given formally as Definition 2.3 below) is a bit involved, but intuitively the idea is as follows. As with list decoding, we have a received word $w \in \Sigma^n$, and the goal is to obtain information about a single symbol $c_i$ of a close-by codeword $c$, given query access to $w$. More precisely, we will require that the decoder outputs a short list of randomized algorithms $A_1, \ldots, A_L$, each of which corresponds to a codeword $c$ with $|\{i : c_i \neq w_i\}| \leq \alpha n$. The requirement is that if $A_r$ corresponds to a codeword $c$, then on input $i$, $A_r(i)$ outputs $c_i$ with high probability, and using no more than $t$ queries to $w$. If such a decoder exists, we say that the code is $(t, \alpha, L)$-locally-list decodable.

Perhaps the most natural algebraic approach for local list decoding is via Reed-Muller codes, which have a natural local structure. As discussed above, a Reed-Muller codeword corresponds to a low-degree multivariate polynomial, and restricting such a polynomial to a line yields a low-degree univariate polynomial, which corresponds to a Reed-Solomon codeword. Using this connection, along with the Guruswami-Sudan list decoding algorithm for Reed-Solomon codes, Arora and Sudan [AS03] and Sudan, Trevisan and Vadhan [STV01] gave algorithms for locally list decoding Reed-Muller codes up to the Johnson bound.[5]

[5]Technically these algorithms were only able to list decode up to radius of $1 - \sqrt{2(1-\delta)}$. To go all the way to the Johnson bound of $1 - \sqrt{1-\delta}$, one needs some additional ideas [BK09]; see [GK16a, Kop15] for further variations on this. There is another regime, where the field size $q$ is constant, and the degree $d$ is much larger than $q$, in which the Reed-Muller codes can be locally list decoded well beyond the Johnson bound, up to the minimum distance [GKZ08, Gop13, BL18]. Note that in this regime the rate of Reed-Muller codes is extremely low, whereas we are interested in the regime of $d < q$ where both rate and relative distance can be made constant.

One might hope to be able to surpass the Johnson bound, and local list decode multivariate multiplicity codes up to their minimum distance $\delta$; after all, the univariate versions are (globally) list decodable up to their minimum distance $\delta = 1 - R$. However, the natural approach (as in [AS03, STV01]) is to rely on the univariate case, and the fact that the list sizes were large for the univariate case was an obstacle to this approach (see Section 1.2.2 for further discussion). Thus, previous work on the local list decodability of multivariate multiplicity codes also only worked up to the Johnson bound [Kop15]. In this work, we use our Theorem 1.2 above, along with some additional ideas, to give a local list decoding algorithm for multivariate multiplicity codes up to their minimum distance.

**Theorem 1.3** (Local list decoding multivariate multiplicity codes up to minimum distance (Informal, see Theorem 4.1))**.** *Let $C$ be an $m$-variate multiplicity code of relative distance $\delta \in (0,1)$ and multiplicity parameter $s \geq \frac{64}{\varepsilon^2}$, defined over a sufficiently large prime field $\mathbb{F}_q$. Then $C$ is $(t, \delta - \varepsilon, L)$-locally list decodable for $L = \exp(\mathsf{poly}(s/\varepsilon))$ and $t = \mathsf{poly}(q) \cdot \left(\frac{s \cdot m \cdot L}{\delta - \varepsilon}\right)^{O(m)}$.*

Note that an $m$-variate multiplicity code has length $N := q^m$, and so for any constant $m, \delta, \varepsilon$, and $s$, the above theorem gives a query complexity of the form $N^{O(1/m)}$. Moreover, by compromising on a *sub-constant* relative distance, $m$ and $s$ could also be taken to be super-constant, leading to a *sub-polynomial* query complexity of $N^{o(1)}$.

### 1.1.3 Capacity-achieving locally list decodable codes

List decoding algorithms for algebraic codes typically extend to the setting of *list recovery.* A code $C \subseteq \Sigma^n$ is $(\alpha, \ell, L)$-*list recoverable* if for any sequence of input lists $S_1, \ldots, S_n \subseteq \Sigma$, each of size at most $\ell$, there are at most $L$ codewords $c \in C$ which satisfy that $c_i \in S_i$ for all but an $\alpha$ fraction of the coordinates $i$. List-decoding is the special case of list recovery when $\ell = 1$. The definition can also be extended naturally to the setting of *local* list recovery (see Definition 2.3).

In the setting of list recovery, the Johnson bound translates into a decoding radius of $\alpha = 1 - \sqrt{\ell \cdot (1 - \delta)}$, which in particular requires the rate $R \leq 1 - \delta$ to be smaller than $1/\ell$ for a non-trivial decoding radius. On the other hand, list recovery up to capacity corresponds to a decoding radius of $\alpha = 1 - R$ (for any $\ell$, provided that the alphabet size is sufficiently large compared to $\ell$). Thus the Guruswami-Sudan list decoding algorithm for Reed-Solomon codes gives a list recovery algorithm for Reed-Solomon codes of rate $R < 1/\ell$, while the Guruswami-Rudra list decoding algorithm for FRS codes gives a list recovery algorithm for FRS codes of *high rate* (arbitrarily close to 1).

However, until recently, we did not know of *any* high-rate *locally* list recoverable codes. The significance of these codes is that, as shown in [HRW20], such codes can be transformed, using the expander-based distance amplification technique of Alon, Edmonds, and Luby (AEL) [AEL95, AL96], into capacity-achieving locally list decodable codes with roughly the same parameters (note that as opposed to the univariate setting, multivariate multiplicity codes do not achieve list-decoding capacity, and therefore Theorem 1.3 does not directly give capacity-achieving locally list decodable codes). The only previous construction of high-rate locally list recoverable codes was the tensor-based construction of [HRW20], and applying the AEL distance amplification method mentioned above, this gave the first construction of capacity-achieving locally list decodable codes. This construction achieved arbitrarily small polynomial query complexity, and even slightly sub-polynomial query complexity $N^{\tilde{O}(1/\log\log N)}$.[6] In the recent work [KRR+21], it was shown that using the tensor-based approach, one cannot reduce the query complexity below $N^{\Omega(1/\log\log N)}$.

In this work we observe that the above Theorem 1.3 extends also to the setting of list recovery, and gives an alternative construction of high-rate locally list recoverable codes with significantly lower query complexity of $\exp(\tilde{O}((\log N)^{5/6}))$. Combined with the AEL distance amplification, this gives capacity-achieving locally list decodable codes with the same query complexity. This brings the query complexity for capacity-achieving local list decodability close to the best known query complexity for high-rate locally (uniquely) decodable codes [KMRS17], which is $\exp(\tilde{O}(\sqrt{\log n}))$ (for the same codes).

**Theorem 1.4** (Capacity-achieving locally list decodable codes (Informal, see Theorem 5.1))**.** *For any constant $R \in (0,1)$ and $\varepsilon > 0$, there exists an infinite family $\{C_N\}_N$ of codes that satisfy the following.*

1. *$C_N$ is a code of block length $N$ and rate at least $R$.*
2. *$C_N$ is $(t, 1-R-\varepsilon, L)$-locally list decodable for*
$$t = \exp\left(\tilde{O}\left((\log N)^{5/6}\right)\right) \qquad \textit{and} \qquad L = \exp\left(\tilde{O}\left((\log N)^{2/3}\right)\right).$$
3. *The alphabet size of $C_N$ is $O(1)$.*

**Results omitted in the current version.** As noted above, for simplicity and clarity we have decided to omit some of the results presented in our previous version [KRSW18]. For completeness, we list below all omitted results.

1. **Reducing alphabet size:** The FRS and univariate multiplicity codes in Theorems 1.1 and 1.2 have large alphabet sizes. It turns out that this can be ameliorated using concatenation and expander-based distance amplification [AEL95, GI04], at the cost of increasing the dependency of the list size on $1/\varepsilon$ to *quadruply-exponential*.[7] The details can be found in [KRSW18, Section 6]. In the current version we omit these details and the corresponding theorem statement, as it is technical, and follows similarly to the use of these techniques for alphabet-reduction in prior work (see Appendix C for a similar transformation in the setting of local list decoding).

2. **A tighter bound on the list size for low-degree univariate multiplicity codes:** As shown in [KRSW18, Section 4.1], when the degree $d$ is smaller than the characteristic of the underlying field, one can obtain the same quantitative bound on the list size of univariate multiplicity codes as is given for FRS codes in Theorem 1.1. However, when the degree $d$

[6]Here, the $\tilde{O}$ notation hides logarithmic factors in the argument.

[7]The reason for the large list size is that the construction roughly uses four levels of encodings, two of these via FRS codes, and two other via random linear codes, and for both codes, the best-known list size is exponential in $1/\varepsilon$. It may be possible to reduce the list size by replacing the random linear codes with other codes of smaller list size and succinct representation, e.g., the *pseudo-linear codes* of [GI01]. However, the list size would still be at least doubly-exponential in $1/\varepsilon$.

is larger than the characteristic—which is what is relevant for the application to local list decoding of multivariate multiplicity codes — we only obtain the weaker bound given in Theorem 1.2. For simplicity, in the current version we only state and prove the weaker bound on the list size.

3. **Reducing the query complexity of local list decoder:** In [KRSW18], we have shown a slightly lower upper bound of $\exp(\tilde{O}((\log N)^{3/4}))$ on the query complexity of capacity-achieving locally list decodable codes. This improvement followed from a tighter bound on the list size of univariate multiplicity codes given in Theorem 1.2 for the special case of list recovery from a small fraction of errors. As the analysis is quite involved, and for clarity, we present in the current paper a simpler argument which obtains only the weaker bound of $\exp(\tilde{O}((\log N)^{5/6}))$ given in Theorem 1.4.

Next we give an overview of the proofs of our main results.

## 1.2 Overview of techniques

### 1.2.1 List decoding FRS codes with constant list size

Let $C \subseteq \Sigma^n$ be a folded Reed-Solomon code with constant rate $R \in (0,1)$, relative distance $\delta = 1 - R$, and folding parameter $s$, so that $\Sigma = \mathbb{F}_q^s$. We begin by describing a simple argument that gives a slightly worst list size of $(1/\varepsilon)^{O(1/\varepsilon^2)}$. Then we will explain how it can be tightened to give a list size of $(1/\varepsilon)^{O(1/\varepsilon)}$, as stated in Theorem 1.1.

Recall that we are given a received word $w \in \Sigma^n$, and we want to find the list $\mathcal{L}$ of all codewords $c \in C$ such that $\mathrm{dist}(w,c) \le 1 - R - \varepsilon = \delta - \varepsilon$ (i.e., $c_i \neq w_i$ for at most a $(\delta - \varepsilon)$-fraction of the coordinates). Recall also that by [GW13], we know that $\mathcal{L}$ is contained in an $\mathbb{F}_q$-linear subspace $V$ of dimension at most $O(1/\varepsilon)$. How many elements $c$ of the subspace $V \subseteq C$ can have $\mathrm{dist}(c,w) \le \delta - \varepsilon$? We show that there cannot be too many such $c$.

The proof is algorithmic: we will give a simple randomized algorithm PRUNE, which when given $w$ and the low dimensional subspace $V$, either outputs a codeword $c \in V$ or outputs Fail. The guarantee is that for any $c \in \mathcal{L}$, $c$ is output by the algorithm PRUNE with probability at least $p_0 = \Omega_\varepsilon(1)$. This implies that $|\mathcal{L}| \le \frac{1}{p_0} = O_\varepsilon(1)$.

The algorithm PRUNE works as follows. For some parameter $t = O_\varepsilon(1)$, to be determined later on, we pick coordinates $i_1, i_2, \ldots, i_t \in [n]$ uniformly at random, and let $I := \{i_1, \ldots, i_t\}$. Then the algorithm PRUNE checks if there is a unique codeword $c \in V$ that agrees with $w$ on $I$ (that is, $c_i = w_i$ for all $i \in I$). If so, we output that unique element $c$; otherwise (i.e., either there are zero or greater than one such $c$'s) we output Fail.

It remains to show that for any $c \in \mathcal{L}$, the algorithm outputs $c$ with constant probability. Fix such a $c$. Let $E_1$ be the event that $c$ agrees with $w$ on $I$, and let $E_2$ be the event that two different codewords in $V$ agree on $I$. Note that the algorithm will output $c$ if and only if the event $E_1$ holds and the event $E_2$ does not hold, so the probability that $c$ is output is at least $\Pr[E_1] - \Pr[E_2]$.

By assumption, $c_i = w_i$ for at least a $(1 - \delta + \varepsilon)$-fraction of the coordinates $i \in [n]$, and so

$\Pr[E_1] \geq (1-\delta+\varepsilon)^t$. Next, note that by linearity, the event $E_2$ can be alternatively expressed as $\dim\left(V \cap \left(\bigcap_{i\in I} H_i\right)\right) > 0$, where $H_i := \{v \in \Sigma^n | v_i = 0\}$. Lemma 2 from [SY11] (stated in a different language, and proven for a very different purpose) shows that for any linear space $V$ with dimension $r$ and relative distance at least $\delta$, for a large enough $t$ (depending on $r$ and $\delta$), it is very unlikely that $\dim\left(V \cap \left(\bigcap_{i\in I} H_i\right)\right) > 0$.

One way to prove (a version of) Lemma 2 from [SY11] is as follows. First observe that for a random $i \in [n]$, we have that $\dim(V \cap H_i) < \dim(V)$ with probability at least $\delta$. To see this, fix a non-zero element $v \in V$. By assumption, $v$ has at least a $\delta$-fraction of non-zero coordinates, so with probability at least $\delta$ we have that $v_i \neq 0$. Conditioned on this, we have that $v \notin V \cap H_i$, and so the dimension reduces by at least 1 when intersecting with $H_i$. Iterating over this, we conclude that the probability that $\dim\left(V \cap \left(\bigcap_{i\in I} H_i\right)\right) > 0$ is at most

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

We conclude that $c$ is output by the algorithm with probability at least

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

where $\delta \in (0,1)$ is a constant and $r = O(1/\varepsilon)$. Finally, it can be verified that the right hand term is comparable to the left hand term when setting $t \approx \frac{1}{\varepsilon^2}\log\left(\frac{1}{\varepsilon}\right)$. In this case, we get that the algorithm $\mathsf{PRUNE}$ outputs any codeword $c \in \mathcal{L}$ with probability at least $p_0 = \varepsilon^{O(1/\varepsilon^2)}$, and so the list size is at most $|\mathcal{L}| \leq \frac{1}{p_0} = (1/\varepsilon)^{O(1/\varepsilon^2)}$.

Note that in the above argument, the only special property of FRS (besides linearity and distance) we use is that its list is contained in a low-dimensional subspace. Specifically, the above argument shows more generally (see Lemma 3.1 for a formal statement) that if $C$ is *any* $\mathbb{F}_q$-linear code of relative distance $\delta$ that is list decodable up to a radius of $\delta - \varepsilon$ with a list of dimension $r$, then $C$ is $(\delta-\varepsilon, L)$-list decodable, where $L$ depends only on $\delta, \varepsilon$, and $r$.

In particular, this argument applies not only to FRS codes but also to univariate multiplicity codes. This results in Theorem 1.2.

In Theorem 1.1, we get an improved bound of $(1/\varepsilon)^{O(1/\varepsilon)}$ on the list size of FRS codes. To obtain this, we use a tighter bound on the probability that $\dim\left(V \cap \left(\bigcap_{i\in I} H_i\right)\right) > 0$ for the special case where $V$ is a subspace of FRS. Such a bound was shown in [GK16b] (once again, in a different language, and for a very different purpose), and it roughly says that the expected dimension of $V \cap H_i$, for a random $i \in [n]$, is at most $(1-\delta)\dim(V)$. Thus, with $t$ applications, we get that the probability that $\dim\left(V \cap \left(\bigcap_{i\in I} H_i\right)\right) > 0$ is at most $(1-\delta)^t \dim(V)$. Consequently, the bound in (1) becomes

$$p_0 \geq \Pr[E_1] - \Pr[E_2] \geq (1-\delta+\varepsilon)^t - (1-\delta)^t \cdot r,$$

for constant $\delta \in (0,1)$ and $r = O(1/\varepsilon)$, and setting this time $t \approx \frac{1}{\varepsilon}\log\left(\frac{1}{\varepsilon}\right)$ gives $p_0 = \varepsilon^{O(1/\varepsilon)}$ and $|\mathcal{L}| \leq \frac{1}{p_0} = (1/\varepsilon)^{O(1/\varepsilon)}$.

**Remark 1.5** (A tighter bound on the list size for low-degree univariate multiplicity codes.)**.** The tighter bound of $(1/\varepsilon)^{O(1/\varepsilon)}$ on the list size can also be shown to hold for univariate multiplicity codes whose degree $d$ is smaller than the characteristic of the field $\mathbb{F}_q$. The idea is to use a different but analogous lemma from [GK16b]. The precise statement and proof can be found in our preliminary version [KRSW18, Section 4.1]. However, for our application to local list decoding of multivariate multiplicity codes, we need to deal with univariate multiplicity codes where the degree $d$ is larger than $q$. In that case the lemma from [GK16b] does not apply. Thus, in this work, for simplicity we only state and prove the weaker bound of $(s/\varepsilon)^{O(s/\varepsilon^2)}$ stated in Theorem 1.2. This is what follows from the approach described above using the lemma from [SY11] and the bound of $O(s/\varepsilon)$ on the dimension of the list of univariate multiplicity codes.

### 1.2.2 Local list decoding multivariate multiplicity codes up to minimum distance

We now describe the high-level view of our local list decoding algorithm for multivariate multiplicity codes. Our algorithm follows the general paradigm of Arora and Sudan [AS03] and Sudan, Trevisan and Vadhan [STV01], who gave a local list-decoding algorithm for Reed-Muller codes slightly below the Johnson bound, up to a radius of $1 - \sqrt{2(1-\delta)}$. To locally list decode multiplicity codes up to the minimum distance, we use our improved bound on the list size for univariate multiplicity codes (Theorem 1.2), together with some additional ideas elaborated on below. These ideas can also be used to locally list decode Reed-Muller codes all the way up to the Johnson bound of $1 - \sqrt{1-\delta}$ (in particular, this can remove the restriction from [AS03, STV01] that the degree $d$ needs to be at most $1/2$ the size of the field $\mathbb{F}_q$), and we first describe our ideas in this context.[8]

**Local list decoding of Reed-Muller codes below the Johnson bound.** We start by describing the approach of [AS03, STV01] for local list-decoding of Reed-Muller codes slightly below the Johnson bound, up to a radius of $1 - \sqrt{2(1-\delta)}$.

Local list decoding of Reed-Muller codes is the following problem: we are given a function $f : \mathbb{F}_q^m \to \mathbb{F}_q$ which is promised to be close to the evaluation table of some degree $d$ polynomial $Q(X_1, \dots, X_m)$. At a high level, the local list decoding algorithm of [AS03, STV01] for Reed-Muller codes has two phases: generating advice, and decoding with advice. To generate the advice, we pick a uniformly random $\mathbf{a} \in \mathbb{F}_q^m$ and "guess" a value $z \in \mathbb{F}_q$ (this guessing can be done by going over all $z \in \mathbb{F}_q$; each possible guess corresponds to an element in the list). Our hope for this guess is that $z$ equals $Q(\mathbf{a})$.

Once we have this advice, we use it to decode. We define an oracle machine $M^f[\mathbf{a}, z]$, which takes as advice $[\mathbf{a}, z]$, has query access to $f$, and given an input $\mathbf{x} \in \mathbb{F}_q^m$, tries to compute $Q(\mathbf{x})$. The algorithm first considers the line $\lambda$ passing through $\mathbf{x}$ and the advice point $\mathbf{a}$, and globally list decodes the restriction of $f$ to this line to obtain a list $\mathcal{L} \subseteq \mathbb{F}_q[T]$ of univariate polynomials. These univariate polynomials are candidates for $Q|_\lambda$. Which of these univariate polynomials is $Q|_\lambda$? We use our guess $z$ (which is supposed to be $Q(\mathbf{a})$): if there is a unique univariate polynomial in the list with value $z$ at $\mathbf{a}$, then we deem that to be our candidate for $Q|_\lambda$, and output its value at the point $\mathbf{x}$ as our guess for $Q(\mathbf{x})$.

[8]An alternative approach for local list decoding of Reed-Muller codes up to the Johnson bound was given in [BK09] (see also [GK16a, Kop15]). Jumping ahead, this approach will not work for us since it is based on global unique decoding on planes, and *a priori* we do not have a good bound on the list size of bivariate multiplicity codes.

The above algorithm will be correct on the point $\mathbf{x}$ if (1) there are not too many errors on the line through $\mathbf{x}$ and $\mathbf{a}$, and (2) no other polynomial in $\mathcal{L}$ takes the same value at $\mathbf{a}$ as $Q|_\lambda$ does. The first event is high probability by standard sampling bounds. As to the second event, using the random choice of $\mathbf{a}$, and that any pair of polynomials in $\mathcal{L}$ differ by at least a $\delta := 1 - \frac{d}{q}$ fraction of the points, we get that any other polynomial $P \in \mathcal{L}$ will agree with $Q|_\lambda$ on $\mathbf{a}$ with probability at most $1 - \delta$. Assuming that $L := |\mathcal{L}| \ll \frac{1}{1-\delta}$, one can then apply a union bound over all elements of $\mathcal{L}$ to show that with high probability, no other polynomial $P \in \mathcal{L}$ agrees with $Q|_\lambda$ on $\mathbf{a}$. The Johnson bound tells us that $L < \frac{1}{1-\delta}$ as long as the decoding radius $\alpha$ is smaller than $1 - \sqrt{2(1-\delta)}$, and thus this approach can be used to locally list decode Reed-Muller codes up to this radius.

**Local list decoding of Reed-Muller codes up to the Johnson bound.** Suppose we wanted to locally list decode Reed-Muller codes all the way up to the Johnson bound of $\alpha < 1 - \sqrt{1-\delta}$, in which case $L$ may be larger.

Our first idea is to use *derivatives* to disambiguate the list. Specifically, as before the advice is generated by choosing a uniformly random point $\mathbf{a} \in \mathbb{F}_q^m$, but now the guess $z$ is supposed to equal to all derivatives up to order $s$ of $Q$ at $\mathbf{a}$ for some integer $s > 0$. To take advantage of derivatives, we recall the “Schwartz-Zippel lemma with multiplicities” from [DKSS13], which says that two univariate degree $d$ polynomials agree on all derivatives up to order $s$ on at most a $\frac{d}{sq}$-fraction of points. This implies in turn that the advice fails to disambiguate any pair of polynomials in $\mathcal{L}$ with probability at most $\frac{d}{sq}$, and taking $s \gg L$ allows us to apply a union bound over $\mathcal{L}$. Thus, considering derivatives could be useful even in the context of Reed-Muller codes!

However, a disadvantage of this approach is that it significantly increases the list size, which is the number of different guesses for all derivatives up to order $s$ at the point $\mathbf{a}$. Specifically, the number of different guesses is $q^{\binom{m+s-1}{m}}$, since the number of $m$-variate derivatives up to order $s$ is $\binom{m+s-1}{m}$, and each such derivative can evaluate to any point in $\mathbb{F}_q$. Note that this number is greater than the code length $N := q^m$ for any $s > 0$ (and generally, is very large for our choice of $s \gg L$).

To get a *sublinear* list size, we generate the guess $z$ more cleverly. Specifically, we first choose $t$ random lines through $\mathbf{a}$, then (globally) list decode on the restriction of $f$ to these lines, and finally aggregate the results to obtain a short list $Z$ of guesses for all derivatives up to order $s$ of $Q$ at $\mathbf{a}$. This aggregation turns out to be a *list recovery* problem for Reed-Muller codes evaluated on product sets, and in particular the Johnson bound for list recovery implies that by choosing $t = L^{O(m)}$, the size of $Z$ can be made as small as $O(L)$. Thus, list recovery comes up naturally even if one is only interested in the list decoding task!

Note however that the latter procedure increases the number of queries to $L^{O(m)} \cdot q$ (since we are querying $t = L^{O(m)}$ lines, each containing $q$ points), so this approach can give sublinear query complexity only if $L \ll q$. Thus, this approach can be used to locally list decode the Reed-Muller code up to a radius of $1 - \sqrt{1-\delta}$, where the Johnson bound guarantees a constant size list.

**Local list decoding of multiplicity codes up to minimum distance.** For Reed-Muller codes $L \gg q$ beyond the Johnson bound, and so above this radius, the number of queries would be super-linear in the code length $N = q^m$. However, at this point we can resort to our improved bound on the list size of univariate multiplicity codes (Theorem 1.2), which gives a constant list size well beyond the Johnson bound, up to the minimum distance of the code. We show that the above approach, with some modifications, can be adapted to locally list decode multiplicity codes up to their minimum distance with sublinear query complexity and list size.

**Subsequent work.** Subsequent work by Guo and Ron-Zewi [GR22] obtained capacity-achieving (globally) list-decodable codes with both constant alphabet and constant list sizes, based on pre-encoding of algebraic-geometric codes. Compared to our Theorem 1.1, combined with an alphabet reduction (see Footnote 7), the list-size obtained in [GR22] has smaller dependency on $1/\varepsilon$ that is only *singly-exponential*. Another subsequent work by Bhandari, Harsha, Kumar, and Sudan [BHKS21] gave a list-decoding algorithm for multivariate multiplicity codes over a *grid* (i.e., a product-set) up to the minimum distance. Their techniques are completely different than ours, in particular their algorithm is non-local, and it only applies in the regime of parameters in which the number of variables is constant, and the characteristic of the field is larger than the degree, which is less useful in the local setting. However, their algorithm works over arbitrary product-sets, as opposed to our local list-decoding algorithm for multivariate multiplicity codes that only works over the whole field. Finally, a recent work by Karliner, Salama, and Ta-Shma [KST22] has shown that multivariate multiplicity codes are also locally *testable*.

**Organization.** We begin in Section 2 with notation and preliminary definitions. Once these are in place, in Section 3 we show that folded Reed-Solomon codes and univariate multiplicity codes achieve list decoding capacity with constant list size. In Section 4, we present our local list decoding algorithm for multivariate multiplicity codes up to their minimum distance. Finally, in Section 5 we explain how one can use an extension of the local list decoding algorithm to the setting of list recovery, combined with the expander-based machinery of [AEL95], to give capacity-achieving locally list decodable codes with low query complexity.

## 2 Notation and Preliminaries

We begin by formally defining the coding-theoretic notions we will need, and by setting notation. For every prime power $q$, we denote by $\mathbb{F}_q$ the finite field of $q$ elements. For any finite alphabet $\Sigma$ and for any string $x \in \Sigma^n$ the relative weight $\mathsf{wt}(x)$ of $x$ is the fraction of non-zero coordinates of $x$, that is, $\mathsf{wt}(x) := |\{i \in [n] : x_i \neq 0\}| / n$ (assuming that $0 \in \Sigma$). For any pair of strings $x, y \in \Sigma^n$, the relative distance between $x$ and $y$ is the fraction of coordinates on which $x$ and $y$ differ, and is denoted by $\mathrm{dist}(x, y) := |\{i \in [n] : x_i \neq y_i\}| / n$. For a positive integer $\ell$ we denote by $\binom{\Sigma}{\ell}$ the set containing all subsets of $\Sigma$ of size $\ell$, and for any pair of strings $x \in \Sigma^n$ and $S \in \binom{\Sigma}{\ell}^n$ we denote by $\mathrm{dist}(x, S)$ the fraction of coordinates $i \in [n]$ for which $x_i \notin S_i$, that is, $\mathrm{dist}(x, S) := |\{i \in [n] : x_i \notin S_i\}| / n$. Throughout the paper, we use $\exp(n)$ to denote $2^{\Theta(n)}$. Whenever we use log, it is to the base 2. The notation $O_a(\cdot)$, $\Theta_a(\cdot)$, $\mathsf{poly}_a(\cdot)$, $\exp_a(\cdot)$ means that we treat $a$ as a constant; in particular, $\mathsf{poly}_a(n) = n^{O_a(1)}$ and $\exp_a(n) = 2^{\Theta_a(n)}$.

## 2.1 Error-correcting codes

Let $\Sigma$ be an alphabet and let $n$ be a positive integer (the block length). A code is simply a subset $C \subseteq \Sigma^n$. The elements of a code $C$ are called codewords. If $\mathbb{F}$ is a finite field and $\Sigma$ is a vector space over $\mathbb{F}$, we say that a code $C \subseteq \Sigma^n$ is $\mathbb{F}$-linear if it is an $\mathbb{F}$-linear subspace of the $\mathbb{F}$-vector space $\Sigma^n$. The rate of a code $C$ is the ratio $R := \frac{\log |C|}{\log(|\Sigma|^n)}$, which for $\mathbb{F}$-linear codes equals $\frac{\dim_{\mathbb{F}}(C)}{n \cdot \dim_{\mathbb{F}}(\Sigma)}$. The relative distance $\mathrm{dist}(C)$ of $C$ is the minimum $\delta > 0$ such that for every pair of distinct codewords $c_1, c_2 \in C$ it holds that $\mathrm{dist}(c_1, c_2) \geq \delta$, which for $\mathbb{F}$-linear codes equals the minimum $\delta > 0$ such that $\mathsf{wt}(c) \geq \delta$ for every non-zero $c \in C$.

Given a code $C \subseteq \Sigma^n$, we will occasionally abuse notation and think of $c \in C$ as a map $c : \mathcal{D} \to \Sigma$, where $\mathcal{D}$ is some domain of size $n$. With this notation, the map $c : \mathcal{D} \to \Sigma$ corresponds to the vector $(c(x))_{x \in \mathcal{D}} \in \Sigma^n$. For a code $C \subseteq \Sigma^n$ of relative distance $\delta$, a given parameter $\alpha < \delta/2$, and a string $w \in \Sigma^n$, the problem of decoding from $\alpha$ fraction of errors is the task of finding the unique $c \in C$ (if any) which satisfies $\mathrm{dist}(c, w) \leq \alpha$.

List decoding is a paradigm that allows one to correct more than a $\delta/2$ fraction of errors by returning a small list of close-by codewords. More formally, for $\alpha \in [0, 1]$ and an integer $L$ we say that a code $C \subseteq \Sigma^n$ is $(\alpha, L)$-list decodable if for any $w \in \Sigma^n$ there are at most $L$ different codewords $c \in C$ which satisfy that $\mathrm{dist}(c, w) \leq \alpha$. List recovery is a more general notion where one is given as input a small list of candidate symbols for each of the coordinates and is required to output a list of codewords that are consistent with many of the input lists. Formally we say that a code $C \subseteq \Sigma^n$ is $(\alpha, \ell, L)$-list recoverable if for any $S \in \binom{\Sigma}{\ell}^n$ there are at most $L$ different codewords $c \in C$ which satisfy that $\mathrm{dist}(c, S) \leq \alpha$. Note that list decoding corresponds to the special case of $\ell = 1$.

## 2.2 Locally correctable and locally list recoverable codes

Intuitively, a code is said to be locally correctable [BFLS91, STV01, KT00] if, given a codeword $c \in C$ that has been corrupted by some errors, it is possible to decode any coordinate of $c$ by reading only a small part of the corrupted version of $c$. Formally, it is defined as follows.

**Definition 2.1** (Locally correctable code (LCC))**.** *We say that a code $C \subseteq \Sigma^n$ is $(t, \alpha)$-locally correctable if there exists a randomized algorithm $A$ that satisfies the following requirements:*

- ***Input:*** *$A$ takes as input a coordinate $i \in [n]$ and also gets oracle access to a string $w \in \Sigma^n$ that is $\alpha$-close to a codeword $c \in C$.*
- ***Query complexity:*** *$A$ makes at most $t$ queries to the oracle $w$.*
- ***Output:*** *$A$ outputs $c_i$ with probability at least $\frac{2}{3}$.*

**Remark 2.2.** In the above definition we only required that the local corrector succeeds with probability at least $2/3$. However, the success probability can be increased to any constant $\beta < 1$ by repeating the test independently a constant number of times, and returning the most common output. Note that this only incurs a constant multiplicative overhead in query complexity.

The following definition generalizes the notion of locally correctable codes to the setting of list decoding / recovery. In this setting the algorithm $A$ is required to find all the nearby codewords in an implicit sense.

**Definition 2.3** (Locally list recoverable code)**.** *We say that a code $C \subseteq \Sigma^n$ is $(t, \alpha, \ell, L)$-locally list recoverable if there exists a randomized algorithm $A$ that satisfies the following requirements:*

- ***Input:*** *$A$ gets oracle access to a string $S \in \binom{\Sigma}{\ell}^n$.*
- ***Query complexity:*** *$A$ makes at most $t$ queries to the oracle $S$.*
- ***Output:*** *$A$ outputs $L$ randomized algorithms $A_1, \ldots, A_L$, where each $A_j$ takes as input a coordinate $i \in [n]$, makes at most $t$ queries to the oracle $S$, and outputs a symbol in $\Sigma$.*
- ***Correctness:*** *For every codeword $c \in C$ for which $\mathrm{dist}(c, S) \leq \alpha$, with probability at least $\frac{2}{3}$ over the randomness of $A$ the following event happens: there exists some $j \in [L]$ such that for all $i \in [n]$,*
$$\Pr[A_j(i) = c_i] \geq \frac{2}{3},$$
*where the probability is over the internal randomness of $A_j$.*

We say that a code is $(t, \alpha, L)$-locally list decodable if it is $(t, \alpha, 1, L)$-locally list recoverable.

**Remark 2.4.** For simplicity, in the above definition we only required that for any *fixed* close-by codeword $c$, the output list contains a local corrector for $c$ with high probability. We note that by increasing the query complexity and output list size by a multiplicative factor of $O(\log L)$, we can also guarantee the stronger requirement that with high-probability, the output list contains a local corrector for *any* close-by codeword $c$.

To see this, let $B \subseteq C$ denote the subset of all near-by codewords $c \in C$ with $\mathrm{dist}(c, S) \leq \alpha$. First note that since for any $c \in B$, the list of local algorithms $A_1, \ldots, A_L$ contains a local corrector for $c$ with probability at least $\frac{2}{3}$, we can deduce that $|B| \leq \frac{3}{2} \cdot L$. Moreover, by repeating the local list recovery algorithm for $O(\log L)$ times, and outputting the union of all output lists, we can guarantee that for any codeword $c \in B$, the output list contains a local corrector for $c$ with probability at least $1 - \frac{2}{9} \cdot \frac{1}{L}$. By a union bound over all $\frac{3}{2} \cdot L$ codewords in $B$, this implies in turn that with probability at least $\frac{2}{3}$, the output list contains a local corrector for any $c \in B$. Note that this process indeed only increases the query complexity and output list size by a multiplicative factor of $O(\log L)$.

## 2.3 Some families of polynomial codes

In this section, we formally define the families of codes we will study: folded Reed-Solomon codes [Kra03, GR08], univariate multiplicity codes [RT97, KSY14, GW13], and multivariate multiplicity codes [KSY14].

**Folded Reed-Solomon codes.** Let $q$ be a prime power, and let $s, d, n$ be nonnegative integers such that $n \leq (q-1)/s$. Let $\gamma \in \mathbb{F}_q$ be a primitive element of $\mathbb{F}_q$, and let $a_1, a_2, \ldots, a_n$ be distinct elements in $\{\gamma^{si} \mid 0 \leq i \leq (q-1)/s - 1\}$. Let $\mathcal{D} = \{a_1, \ldots, a_n\}$.

For a polynomial $P(X) \in \mathbb{F}_q[X]$ and $a \in \mathbb{F}_q$, let $P^{[s]}(a) \in \mathbb{F}_q^s$ denote the vector:

$$P^{[s]}(a) = \begin{bmatrix} P(a) \\ P(\gamma a) \\ \vdots \\ P(\gamma^{s-1} a) \end{bmatrix}.$$

The folded Reed-Solomon code $\mathrm{FRS}_{q,s}(n,d)$ is a code over alphabet $\mathbb{F}_q^s$. To every polynomial $P(X) \in \mathbb{F}_q[X]$ of degree at most $d$, there corresponds a codeword $c$:

$$c : \mathcal{D} \to \mathbb{F}_q^s,$$

where for each $a \in \mathcal{D}$:

$$c(a) = P^{[s]}(a).$$

Explicitly,

$$\begin{aligned} P(x) &\mapsto \left(P^{[s]}(a_1), P^{[s]}(a_2), \ldots, P^{[s]}(a_n)\right) \\ &= \left( \begin{bmatrix} P(a_1) \\ P(\gamma a_1) \\ \vdots \\ P(\gamma^{s-1} a_1) \end{bmatrix}, \begin{bmatrix} P(a_2) \\ P(\gamma a_2) \\ \vdots \\ P(\gamma^{s-1} a_2) \end{bmatrix}, \ldots, \begin{bmatrix} P(a_n) \\ P(\gamma a_n) \\ \vdots \\ P(\gamma^{s-1} a_n) \end{bmatrix} \right). \end{aligned}$$

Note that Reed-Solomon codes correspond to the special case of $s = 1$. The following claim summarizes the basic properties of folded Reed-Solomon codes.

**Claim 2.5** ([GR08])**.** *The folded Reed-Solomon code* $\mathrm{FRS}_{q,s}(n,d)$ *is an* $\mathbb{F}_q$*-linear code over alphabet* $\mathbb{F}_q^s$ *of block length* $n$*, rate* $(d+1)/(sn)$*, and relative distance at least* $1 - d/(sn)$*.*

**Univariate multiplicity codes.** Let $P(x)$ be a univariate polynomial over $\mathbb{F}_q$. For $i \in \mathbb{N}$, we define the $i$'th (Hasse) derivative $P^{(i)}(X)$ as the coefficient of $Z^i$ in the expansion

$$P(X+Z) = \sum_i P^{(i)}(X) Z^i.$$

Let $q$ be a prime power, and let $s, d, n$ be nonnegative integers such that $n \leq q$. Let $a_1, a_2, \ldots, a_n$ be distinct elements in $\mathbb{F}_q$. Let $\mathcal{D} = \{a_1, \ldots, a_n\}$.

For a polynomial $P(X) \in \mathbb{F}_q[X]$, let $P^{(<s)}(a) \in \mathbb{F}_q^s$ denote the vector:

$$P^{(<s)}(a) = \begin{bmatrix} P(a) \\ P^{(1)}(a) \\ \vdots \\ P^{(s-1)}(a) \end{bmatrix}.$$

The $\textsf{univariate multiplicity code}$ $\mathsf{MULT}^{(1)}_{q,s}(n,d)$ is a code over alphabet $\mathbb{F}_q^s$. To every polynomial $P(X) \in \mathbb{F}_q[X]$ of degree at most $d$, there corresponds a codeword $c$:

$$c : \mathcal{D} \to \mathbb{F}_q^s,$$

where for each $a \in \mathcal{D}$:

$$c(a) = P^{(<s)}(a).$$

Explicitly,

$$\begin{aligned} P(x) &\mapsto \left(P^{(<s)}(a_1), P^{(<s)}(a_2), \dots, P^{(<s)}(a_n)\right) \\ &= \left( \begin{bmatrix} P(a_1) \\ P^{(1)}(a_1) \\ \vdots \\ P^{(s-1)}(a_1) \end{bmatrix}, \begin{bmatrix} P(a_2) \\ P^{(1)}(a_2) \\ \vdots \\ P^{(s-1)}(a_2) \end{bmatrix}, \dots, \begin{bmatrix} P(a_n) \\ P^{(1)}(a_n) \\ \vdots \\ P^{(s-1)}(a_n) \end{bmatrix} \right). \end{aligned}$$

Once again, Reed-Solomon codes correspond to the special case of $s = 1$.

**Claim 2.6** ([KSY14]). *The univariate multiplicity code $\mathsf{MULT}^{(1)}_{q,s}(n,d)$ is an $\mathbb{F}_q$-linear code over alphabet $\mathbb{F}_q^s$ of block length $n$, rate $(d+1)/(sn)$, and relative distance at least $1 - d/(sn)$.*

Of particular importance is the setting where $q = n$ and $\mathcal{D}$ equals the whole field $\mathbb{F}_q$. We refer to this code as the *whole-field univariate multiplcity code*, and denote it by $\mathsf{MULT}^{(1)}_{q,s}(d)$. This will be relevant to multivariate multiplicity codes, which we define next.

**Multivariate multiplicity codes.** Multivariate multiplicity codes are a generalization of whole-field univariate multiplicity codes to the multivariate setting. For multivariate polynomials $Q \in \mathbb{F}_q[X_1, \dots, X_m]$, we use the notation $\mathbf{X} = (X_1, \dots, X_m)$ and $\mathbf{X}^{\mathbf{i}} = X_1^{i_1} \cdots X_m^{i_m}$ where $\mathbf{i} = (i_1, \dots, i_m) \in \mathbb{N}^m$. For $\mathbf{i} \in \mathbb{N}^m$, we define the $\textsf{i'th (Hasse) derivative}$ $Q^{(\mathbf{i})}(\mathbf{X})$ as the coefficient of $\mathbf{Z}^{\mathbf{i}}$ in the expansion

$$Q(\mathbf{X} + \mathbf{Z}) = \sum_{\mathbf{i}} Q^{(\mathbf{i})}(\mathbf{X}) \mathbf{Z}^{\mathbf{i}}.$$

Let $q$ be a prime power, and let $s, d, m$ be nonnegative integers. Let $U_{m,s}$ denote the set $\{\mathbf{i} \in \mathbb{N}^m \mid \mathsf{wt}(\mathbf{i}) < s\}$. Note that $|U_{m,s}| = \binom{m+s-1}{m}$. Let $\Sigma_{m,s} = \mathbb{F}_q^{U_{m,s}}$.

For a polynomial $Q(X_1, \dots, X_m) \in \mathbb{F}_q[X_1, \dots, X_m]$, and a point $\mathbf{a} \in \mathbb{F}_q^m$, define $Q^{(<s)}(\mathbf{a}) \in \Sigma_{m,s}$ by:

$$Q^{(<s)}(\mathbf{a}) = (Q^{(\mathbf{i})}(\mathbf{a}))_{\mathbf{i} \in U_{m,s}}.$$

The $\textsf{multiplicity code}$ $\mathsf{MULT}^{(m)}_{q,s}(d)$ is a code over alphabet $\Sigma_{m,s}$. To every polynomial $Q(X_1, \dots, X_m) \in \mathbb{F}_q[X_1, \dots, X_m]$ of (total) degree at most $d$, there corresponds a codeword $c$:

$$c : \mathbb{F}_q^m \to \Sigma_{m,s},$$

where for each $\mathbf{a} \in \mathbb{F}_q^m$,

$$c(\mathbf{a}) = Q^{(<s)}(\mathbf{a}).$$

Note that Reed-Muller codes correspond to the special case of $s = 1$.

**Claim 2.7** ([KSY14]). *The multivariate multiplicity code $\mathsf{MULT}_{q,s}^{(m)}(d)$ is an $\mathbb{F}_q$-linear code over alphabet $\Sigma_{m,s}$ of block length $q^m$, rate at least $(1 - m^2/s)(d/(sq))^m$, and relative distance at least $1 - d/(sq)$.*

## 3 Constant-size output list for folded Reed-Solomon and univariate multiplicity codes

In this section we show that folded Reed-Solomon and univariate multiplicity codes are list decodable up to capacity with constant list size, independent of the block length. For this, we first show in Section 3.1 that the list cannot contain many codewords from a low dimensional subspace. Then in Section 3.2 we instantiate this with the results of [GW13] showing that the list of both folded Reed-Solomon and univariate multiplicity codes is contained in a low-dimensional subspace. Finally in Section 3.3 we show a tighter bound on the list size of FRS codes. Towards our results on local list decoding, we prove a more general result that applies also to list recovery.

### 3.1 Output list has small intersection with low dimensional subspaces

Our main lemma shows that when list recovering a linear code of large distance, the output list cannot contain many codewords from a low dimensional subspace.

**Lemma 3.1.** *Suppose that $C \subseteq (\mathbb{F}_q^s)^n$ is an $\mathbb{F}_q$-linear code of relative distance $\delta$. Suppose furthermore that $C$ is list recoverable from a $(\delta - \varepsilon)$-fraction of errors and input list size $\ell$, with an output list $\mathcal{L}$ that is contained in an $\mathbb{F}_q$-linear subspace $V \subseteq C$ of dimension $r$. Then*

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

*Moreover, for any $\gamma > 0$, there is a randomized algorithm that, given a basis for $V$, outputs $\mathcal{L}$ with probability at least $1 - \gamma$ in time* $\mathsf{poly}(\log q, s, n, |\mathcal{L}|, \log(1/\gamma))$.

To show that the output list $\mathcal{L}$ of $C$ cannot contain too many elements from $V$ (and to find $\mathcal{L}$ in the process), we first give a preliminary randomized algorithm $\mathsf{PRUNE}$ that outputs a small size list $\hat{\mathcal{L}}$ such that any codeword of $\mathcal{L}$ appears in $\hat{\mathcal{L}}$ with probability at least $p_0$. This implies that $|\mathcal{L}| \leq \mathbb{E}[|\hat{\mathcal{L}}|]/p_0$, proving the first part of Lemma 3.1. Now that we know that $|\mathcal{L}|$ is small, our final algorithm simply runs $\mathsf{PRUNE}$ $O\left(\frac{\log(1/\gamma)}{p_0} \cdot \log|\mathcal{L}|\right)$ times and returns the union of the output lists. By a union bound, all elements of $\mathcal{L}$ will appear in the union of the output lists with probability at least $1 - \gamma$. This will complete the proof of the second part of Lemma 3.1.

We start by describing the algorithm PRUNE and analyzing it. The algorithm is given as input a collection of input lists $S \in \binom{\mathbb{F}_q^s}{\ell}^n$, (a basis for) an $\mathbb{F}_q$-linear subspace $V \subseteq C$ of dimension $r$ containing the output list, and a parameter $t \in \mathbb{N}$ to be determined later on.

```
Algorithm PRUNE$(S, V, t)$

  1. Initialize $\hat{\mathcal{L}} = \emptyset$.
  2. Pick $i_1, i_2, \ldots, i_t \in [n]$ independently and uniformly at random.
  3. For each choice of $y_1 \in S_{i_1}, y_2 \in S_{i_2}, \ldots, y_t \in S_{i_t}$:
       • If there is exactly one codeword $c \in V$ such that $c_{i_j} = y_j$ for all $j \in [t]$, then:
             $\hat{\mathcal{L}} \leftarrow \hat{\mathcal{L}} \cup \{c\}$.
  4. Output $\hat{\mathcal{L}}$.
```

**Lemma 3.2.** *The algorithm* PRUNE *runs in time* $\mathsf{poly}(\log q, s, n, \ell^t)$, *and outputs a list* $\hat{\mathcal{L}}$ *containing at most* $\ell^t$ *codewords of* $C$, *such that any codeword* $c \in V$ *with* $\mathrm{dist}(c, S) \leq \delta - \varepsilon$ *appears in* $\hat{\mathcal{L}}$ *with probability at least*

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

*Proof.* We clearly have that $|\hat{\mathcal{L}}| \leq \ell^t$, and that the algorithm has the claimed running time. Fix a codeword $\hat{c} \in V$ such that $\mathrm{dist}(\hat{c}, S) \leq \delta - \varepsilon$, we shall show below that $\hat{c}$ belongs to $\hat{\mathcal{L}}$ with probability at least

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

Let $E_1$ denote the event that $\hat{c}_{i_j} \in S_{i_j}$ for all $j \in [t]$. Let $E_2$ denote the event that two different codewords $v, v' \in V$ agree on $i_1, \ldots, i_t$ (that is, $v_{i_j} = v'_{i_j}$ for all $j \in [t]$). By the assumption that $\mathrm{dist}(\hat{c}, S) \leq \delta - \varepsilon$, we readily have that

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

Claim 3.3 below also shows that

$$\Pr[E_2] \leq (1-\delta)^t \cdot \left(\frac{t}{1-\delta}\right)^r. \tag{2}$$

So we have that both $E_1$ occurs and $E_2$ does not occur with probability at least

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

If $E_2$ does not occur, then for every choice of $y_1 \in S_{i_1}, y_2 \in S_{i_2}, \ldots, y_t \in S_{i_t}$, there can be at most one codeword $c \in V$ such that $c_{i_j} = y_j$ for all $j \in [t]$. If $E_1$ also occurs, then in the iteration of

Step 3 where $y_j = \hat{c}_{i_j}$ for each $j \in [t]$, the algorithm will take $c = \hat{c}$, and thus $\hat{c}$ will be included in $\hat{\mathcal{L}}$. This completes the proof of the lemma. $\square$

It remains to prove the following claim.

**Claim 3.3.**

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

*Proof.* For $i \in [n]$, let

$$H_i := \{v \in (\mathbb{F}_q^s)^n \mid v_i = 0\},$$

let $V_0 := V$, and for $j = 1, \ldots, t$, let

$$V_j := V \cap H_{i_1} \cap H_{i_2} \cap \ldots \cap H_{i_j},$$

and $r_j := \dim_{\mathbb{F}_q}(V_j)$. Observe $r = r_0 \ge r_1 \ge \ldots \ge r_t$, and that the event $E_2$ holds if and only if $r_t > 0$.

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

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

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

$\square$

Our main Lemma 3.1 now follows as an immediate corollary of the above Lemma 3.2.

*Proof of Lemma 3.1.* Setting

$$t := 3 \cdot \frac{r}{\varepsilon(1-\delta)} \cdot \log\left(\frac{r}{\varepsilon(1-\delta)}\right)$$

in Lemma 3.2, we have that

$$|\hat{\mathcal{L}}| \le \ell^t \le \ell^{O\left(\frac{r}{\varepsilon(1-\delta)} \cdot \log\left(\frac{r}{\varepsilon(1-\delta)}\right)\right)},$$

and moreover, any codeword in the output list $\mathcal{L}$ of $C$ appears in $\hat{\mathcal{L}}$ with probability at least

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

where the penultimate inequality holds when the ratio $\frac{r}{\varepsilon(1-\delta)}$ is sufficiently large.

This implies in turn that

$$|\mathcal{L}| \leq \frac{\mathbb{E}[|\hat{\mathcal{L}}|]}{p_0} \leq \frac{\ell^t}{p_0} \leq \left(\frac{\ell}{1-\delta}\right)^{O\left(\frac{r}{\varepsilon(1-\delta)} \cdot \log\left(\frac{r}{\varepsilon(1-\delta)}\right)\right)},$$

proving the first part of Lemma 3.1.

To find $\mathcal{L}$, our final algorithm simply runs PRUNE $O\left(\frac{\log(1/\gamma)}{p_0} \cdot \log|\mathcal{L}|\right)$ times and returns the union of the output lists. By a union bound, all elements of $\mathcal{L}$ will appear in the union of the output lists with probability at least $1-\gamma$. This completes the proof of the second part of Lemma 3.1. □

## 3.2 Constant-size output list for folded Reed-Solomon and univariate multiplicity codes

Our first corollary is obtained by instantiating Lemma 3.1 with the following result from [GW13], showing that the output list of folded Reed-Solomon codes is contained in a low dimensional subspace (which can also be found efficiently).

**Theorem 3.4** (Constant-dimensional output list for FRS, [GW13], Theorem 7)**.** *Let $q$ be a prime power, and let $s, d, n$ be nonnegative integers such that $n \leq (q-1)/s$. Let $\delta := 1 - \frac{d}{sn}$ be a lower bound on the relative distance of the folded Reed-Solomon code $\mathrm{FRS}_{q,s}(n,d)$. Let $\varepsilon > 0$ and $\ell \in \mathbb{N}$ be such that $\frac{16\ell}{\varepsilon^2} \leq s$. Then $\mathrm{FRS}_{q,s}(n,d) \subseteq (\mathbb{F}_q^s)^n$ is list recoverable from a $(\delta-\varepsilon)$-fraction of errors and input list size $\ell$, with an output list that is contained in an $\mathbb{F}_q$-linear subspace $V \subseteq \mathrm{FRS}_{q,s}(n,d)$ of dimension at most $r = \frac{4\ell}{\varepsilon}$.*

*Moreover, there is a (deterministic) algorithm that outputs a basis for $V$ in time* $\mathsf{poly}(\log q, s, d, n)$.

**Remark 3.5.** Theorem 7 of [GW13] only deals with the case where $a_i = \gamma^{s(i-1)}$ for all $i = 1, \ldots, n$, and $\ell = 1$. However, it can be verified that the proof goes through for any choice of distinct $a_1, a_2, \ldots, a_n$ in $\{\gamma^{si} \mid 0 \leq i \leq (q-1)/s - 1\}$, and $\ell \in \mathbb{N}$ (see discussion at end of [GW13, Section 2.D.]). In this setting, the bound on the decoding radius in Theorem 7 of [GW13] becomes

$$\alpha := 1 - \frac{\ell}{r+1} - \frac{r}{r+1} \cdot \frac{s}{s-r+1} \cdot \frac{d}{sq},$$

and the above Theorem 3.4 then follows by noting that $\alpha \geq \delta - \varepsilon$ when setting $s \geq \frac{16\ell}{\varepsilon^2}$ and $r = \frac{4\ell}{\varepsilon}$.

We now obtain the following theorem as an immediate corollary of Lemma 3.1 and Theorem 3.4.

**Theorem 3.6** (Constant-size output list for FRS)**.** *Let $q$ be a prime power, and let $s, d, n$ be nonnegative integers such that $n \leq (q-1)/s$. Let $\delta := 1 - \frac{d}{sn}$ be a lower bound on the relative distance of the folded Reed-Solomon code $\mathrm{FRS}_{q,s}(n,d)$. Let $\varepsilon > 0$ and $\ell \in \mathbb{N}$ be such that $\frac{16\ell}{\varepsilon^2} \leq s$. Then $\mathrm{FRS}_{q,s}(n,d)$ is $(\delta - \varepsilon, \ell, L)$-list recoverable with*

$$L = \left(\frac{\ell}{\varepsilon(1-\delta)}\right)^{O\left(\frac{\ell}{\varepsilon^2(1-\delta)} \cdot \log\left(\frac{\ell}{1-\delta}\right)\right)}.$$

*In particular, if $\delta \in (0,1)$ and $\ell \in \mathbb{N}$ are constant, then the output list size is $\left(\frac{1}{\varepsilon}\right)^{O(1/\varepsilon^2)}$.*

*Moreover, for any $\gamma > 0$, there is a randomized algorithm that list recovers $\mathrm{FRS}_{q,s}(n,d)$ with the above parameters, and with success probability at least $1-\gamma$, in time $\mathsf{poly}(\log q, s, d, n, L, \log(1/\gamma))$.*

Our second corollary is obtained by replacing Theorem 7 of [GW13] with Theorem 17 of that paper, showing that the output list of univariate multiplicity codes is contained in a low-dimensional subspace. However, towards our application for local list decoding, we shall need a slight modification of that result, specifically: (1) we need to talk about list recovery, not just list decoding, (2) we allow the degree to be larger than the field size, while [GW13] assumes that the degree is smaller than the characteristic of the field, and (3) we work with Hasse derivatives, while [GW13] works with standard derivatives. All differences are minor; for completeness we include a full proof in Appendix A.

**Theorem 3.7** (Constant-dimensional output list for UniMULT)**.** *Let $q$ be a prime power, and let $s, d, n$ be nonnegative integers such that $n \leq q$. Let $\delta := 1 - \frac{d}{sn}$ be a lower bound on the relative distance of the univariate multiplicity code $\mathsf{MULT}^{(1)}_{q,s}(n,d)$. Let $\varepsilon > 0$ and $\ell \in \mathbb{N}$ be such that $\frac{16\ell}{\varepsilon^2} \leq s$ and $\frac{4\ell}{\varepsilon} \leq \mathsf{char}(\mathbb{F}_q)$. Then $\mathsf{MULT}^{(1)}_{q,s}(n,d) \subseteq (\mathbb{F}_q^s)^n$ is list recoverable from a $(\delta-\varepsilon)$-fraction of errors and input list size $\ell$, with output list is contained in an $\mathbb{F}_q$-linear subspace $V \subseteq \mathsf{MULT}^{(1)}_{q,s}(n,d)$ of dimension at most $\frac{4\ell}{\varepsilon} \cdot \left(1 + \frac{d}{\mathsf{char}(\mathbb{F}_q)}\right)$.*

*Moreover, there is a (deterministic) algorithm that outputs a basis for $V$ in time $\mathsf{poly}(\log q, s, d, n)$.*

Instantiating Lemma 3.1 with the above theorem gives the following.

**Theorem 3.8** (Constant-size output list for UniMult)**.** *Let $q$ be a prime power, and let $s, d, n$ be nonnegative integers such that $n \leq q$. Let $\delta := 1 - \frac{d}{sn}$ be a lower bound on the relative distance of the univariate multiplicity code $\mathsf{MULT}^{(1)}_{q,s}(n,d)$. Let $\varepsilon > 0$ and $\ell \in \mathbb{N}$ be such that $\frac{16\ell}{\varepsilon^2} \leq s$ and $\frac{4\ell}{\varepsilon} \leq \mathsf{char}(\mathbb{F}_q)$. Then $\mathsf{MULT}^{(1)}_{q,s}(n,d)$ is $(\delta-\varepsilon, \ell, L)$-list recoverable with*

$$L = \left(\frac{d}{\mathsf{char}(\mathbb{F}_q)} \cdot \frac{\ell}{\varepsilon(1-\delta)}\right)^{O\left(\frac{d}{\mathsf{char}(\mathbb{F}_q)} \cdot \frac{\ell}{\varepsilon^2(1-\delta)} \cdot \log\left(\frac{\ell}{1-\delta}\right)\right)}.$$

*Moreover, for any $\gamma > 0$, there is a randomized algorithm that list recovers $\mathsf{MULT}^{(1)}_{q,s}(n,d)$ with the above parameters, and with success probability at least $1-\gamma$, in time $\mathsf{poly}(\log q, s, d, n, L, \log(1/\gamma))$.*

## 3.3 Tighter bound on output list size of folded Reed-Solomon codes

For the case of folded Reed-Solomon codes we can obtain the following stronger version of Lemma 3.1, showing that the output list of folded Reed-Solomon codes contains even fewer elements from a low-dimensional subspace.

**Lemma 3.9.** *Let $q$ be a prime power, and let $s, d, n$ be nonnegative integers such that $n \leq (q-1)/s$. Let $\delta := 1 - \frac{d}{sn}$ be a lower bound on the relative distance of the folded Reed-Solomon code $\mathrm{FRS}_{q,s}(n,d)$. Suppose that $\mathrm{FRS}_{q,s}(n,d)$ is list recoverable from a $(\delta-\varepsilon)$-fraction of errors and input list size $\ell$, with an output list $\mathcal{L}$ that is contained in an $\mathbb{F}_q$-linear subspace $V \subseteq \mathrm{FRS}_{q,s}(n,d)$ of dimension $r \leq \frac{\varepsilon s}{4}$. Then $|\mathcal{L}| \leq \left(\frac{\ell}{1-\delta}\right)^{O((\log r)/\varepsilon)}$.*

*Moreover, for any $\gamma > 0$, there is a randomized algorithm that, given a basis for $V$, outputs $\mathcal{L}$ with probability at least $1-\gamma$ in time $\mathsf{poly}\left(\log q, s, n, |\mathcal{L}|, \log(1/\gamma)\right)$.*

**Remark 3.10.** Note that the main difference between the above lemma and Lemma 3.1 is the bound on the size of the output list $\mathcal{L}$, where for constant $\delta \in (0,1)$, the main difference is that in the above lemma the exponent is reduced by a factor of $r$.

Combining the above lemma with Theorem 3.4, we obtain the following corollary.

**Theorem 3.11** (Tighter bound on output list size for FRS)**.** *Let $q$ be a prime power, and let $s, d, n$ be nonnegative integers such that $n \leq (q-1)/s$. Let $\delta := 1 - \frac{d}{sn}$ be a lower bound on the relative distance of the folded Reed-Solomon code $\mathrm{FRS}_{q,s}(n,d)$. Let $\varepsilon > 0$ and $\ell \in \mathbb{N}$ be such that $\frac{16\ell}{\varepsilon^2} \leq s$. Then $\mathrm{FRS}_{q,s}(n,d)$ is $(\delta-\varepsilon, \ell, L)$-list recoverable with*

$$L = \left(\frac{\ell}{\varepsilon}\right)^{O\left(\frac{1}{\varepsilon}\cdot\log\left(\frac{\ell}{1-\delta}\right)\right)}.$$

*In particular, if $\delta \in (0,1)$ and $\ell \in \mathbb{N}$ are constant, then the output list size is $\left(\frac{1}{\varepsilon}\right)^{O(1/\varepsilon)}$.*

*Moreover, for any $\gamma > 0$, there is a randomized algorithm that list recovers $\mathrm{FRS}_{q,s}(n,d)$ with the above parameters, and with success probability at least $1-\gamma$, in time $\mathsf{poly}(\log q, s, d, n, L, \log(1/\gamma))$.*

**Remark 3.12.** Note that the only difference between the above theorem and Theorem 3.6 is the bound on the output list size $L$, where for constant $\delta \in (0,1)$, the main difference is that in the above theorem the exponent is reduced by a factor of $\ell/\varepsilon$. Note that in addition to reducing the dependency of the output list size on $\varepsilon$ from $(1/\varepsilon)^{O(1/\varepsilon^2)}$ to $(1/\varepsilon)^{O(1/\varepsilon)}$, this also reduces the dependency on $\ell$ from exponential to quasi-polynomial.

The proof of Lemma 3.9 is identical to that of Lemma 3.1, except that we obtain a better bound on the probability of the event $E_2$ using the following theorem from [GK16b].

**Theorem 3.13** ([GK16b], Theorem 14)**.** *Let $q$ be a prime power, and let $s, d, n$ be nonnegative integers such that $n \leq (q-1)/s$. Let $\delta := 1 - \frac{d}{sn}$ be a lower bound on the relative distance of the folded Reed-Solomon code $\mathrm{FRS}_{q,s}(n,d)$. Let $V \subseteq \mathrm{FRS}_{q,s}(n,d)$ be an $\mathbb{F}_q$-linear subspace of dimension $r \leq s$. For $i \in [n]$, let*

$$H_i = \left\{ v \in (\mathbb{F}_q^s)^n \mid v_i = 0 \right\}.$$

*Then*

$$\mathbb{E}_{i\in[n]}\left[\dim(V\cap H_i)\right] \le \frac{1-\delta}{1-r/s}\cdot r.$$

*Proof of Lemma 3.9.* We first use the above Theorem 3.13 to deduce a better bound on the probability of the event $E_2$, compared to the bound given in Claim 3.3. As in the proof of Claim 3.3, for $i \in [n]$, let

$$H_i := \{v \in (\mathbb{F}_q^s)^n \mid v_i = 0\},$$

let $V_0 := V$, and for $j = 1, \ldots, t$, let

$$V_j := V \cap H_{i_1} \cap H_{i_2} \cap \ldots \cap H_{i_j},$$

and $r_j := \dim_{\mathbb{F}_q}(V_j)$. Recall once more that $r = r_0 \ge r_1 \ge \ldots \ge r_t$, and that event $E_2$ holds if and only if $r_t > 0$.

By Theorem 3.13,

$$\mathbb{E}\left[r_{j+1} \mid r_j = r'\right] = \mathbb{E}_{i\in[n]}\left[\dim(V_j \cap H_i) \mid \dim(V_j) = r'\right] \le \frac{1-\delta}{1-r'/s}\cdot r' \le \frac{1-\delta}{1-r/s}\cdot r'.$$

Thus

$$\mathbb{E}\left[r_{j+1}\right] \le \mathbb{E}\left[r_j\right]\cdot\frac{1-\delta}{1-r/s},$$

and

$$\mathbb{E}\left[r_t\right] \le \mathbb{E}\left[r_0\right]\cdot\left(\frac{1-\delta}{1-r/s}\right)^t = (1-\delta)^t\cdot\frac{r}{(1-r/s)^t}.$$

Finally, by Markov's inequality this implies in turn that

$$\Pr\left[E_2\right] = \Pr\left[r_t > 0\right] \le (1-\delta)^t\cdot\frac{r}{(1-r/s)^t} \le (1-\delta)^t\cdot\frac{r}{(1-\varepsilon/4)^t},$$

where the last inequality follows by assumption that $r \le \frac{\varepsilon s}{4}$.

Plugging the above bound in the proof of Lemma 3.2, we obtain that any codeword $c \in V$ with $\mathrm{dist}(c, S) \le \delta - \varepsilon$ appears in $\hat{\mathcal{L}}$ with probability at least

$$(1-\delta+\varepsilon)^t - (1-\delta)^t\cdot\frac{r}{(1-\varepsilon/4)^t}. \tag{3}$$

Setting $t := \frac{4\log r}{\varepsilon}$ in Lemma 3.2, we have that $|\hat{\mathcal{L}}| \le \ell^t \le \ell^{O((\log r)/\varepsilon)}$. Moreover, using (3) any codeword in the output list $\mathcal{L}$ of $C$ appears in $\hat{\mathcal{L}}$ with probability at least

$$\begin{aligned}
p_0 : \;&=\; (1-\delta+\varepsilon)^t - (1-\delta)^t\cdot\frac{r}{(1-\varepsilon/4)^t}\\
&\ge\; (1-\delta+\varepsilon)^t - \frac{1}{2}\cdot\left(\frac{1+\varepsilon/4}{1-\varepsilon/4}\cdot(1-\delta)\right)^t\\
&\ge\; \frac{1}{2}(1-\delta+\varepsilon)^t\\
&\ge\; (1-\delta)^{O((\log r)/\varepsilon)},
\end{aligned}$$

where the first inequality holds for sufficiently small $\varepsilon > 0$ by our choice of $t = \frac{4 \log r}{\varepsilon}$.

Finally, this implies in turn that

$$|\mathcal{L}| \leq \frac{\mathbb{E}[|\hat{\mathcal{L}}|]}{p_0} \leq \frac{\ell^t}{p_0} \leq \left(\frac{\ell}{1-\delta}\right)^{O((\log r)/\varepsilon)},$$

as well as the claimed running time. □

## 4 Local list decoding multivariate multiplicity codes

In this section we use our results from Section 3 on *global* list decoding of *univariate* multiplicity codes with constant output list size to show that *multivariate* multiplicity codes can be *locally* list decoded up to their minimum distance. As before, we show a more general version that applies also to list recovery.

**Theorem 4.1** (Local list recovery up to minimum distance for MultiMult)**.** *There exist absolute constants $c_0, c_1 > 0$ so that the following holds. Let $q$ be a prime, let $s, d, m$ be nonnegative integers so that $d + 6s \leq sq$, and let $\delta := 1 - \frac{d}{sq}$. Let $\varepsilon > 0$ and $\ell, L \in \mathbb{N}$ be such that $s \geq \frac{64\ell}{\varepsilon^2}$, $L \geq \left(\frac{s\ell}{\varepsilon}\right)^{c_0 \cdot \frac{s\ell}{\varepsilon^2} \cdot \log\left(\frac{\ell}{1-\delta}\right)}$, and $q \geq c_1 \cdot \frac{s \cdot m^2 \cdot L}{(\delta-\varepsilon)\cdot\varepsilon^2}$. Then the multivariate multiplicity code $\mathsf{MULT}^{(m)}_{q,s}(d)$ is $\left(t, \delta - \varepsilon, \ell, O(m^2 L)\right)$-locally list recoverable for $t = q^2 \cdot \left(\frac{s\cdot m\cdot L}{\delta-\varepsilon}\right)^{O(m)}$.*

*Moreover, the local list recovery algorithm for $\mathsf{MULT}^{(m)}_{q,s}(d)$ runs in time $\mathsf{poly}(t)$.*

The above theorem is a consequence of the following lemma which relates the parameters of the global list recovery algorithm for univariate multiplicity codes to that of the local list recovery algorithm for the corresponding multivariate multiplicity codes.

**Lemma 4.2.** *Let $q$ be a prime power, let $s, d, m$ be nonnegative integers so that $d + 6s \leq sq$, and let $\delta := 1 - \frac{d}{sq}$. Let $\alpha \in (0, \delta)$ and $\ell \in \mathbb{N}$, and suppose that the univariate multiplicity code $\mathsf{MULT}^{(1)}_{q,s}(d)$ is $(\alpha, \ell, L)$-(globally) list recoverable. Let $\varepsilon > 0$ be a parameter, and suppose that $q \geq \frac{30{,}000 \cdot s \cdot m^2 \cdot L}{\alpha\cdot\varepsilon^2}$. Then the multivariate multiplicity code $\mathsf{MULT}^{(m)}_{q,s}(d)$ is $\left(t, \alpha - \varepsilon, \ell, O(m^2 L)\right)$-locally list recoverable for $t = q^2 \cdot \left(\frac{s\cdot m\cdot L}{\alpha}\right)^{O(m)}$.*

*Moreover, if $\mathsf{MULT}^{(1)}_{q,s}(d)$ can be probabilistically list-recovered with success probability at least $1 - \frac{1}{q}$ in time $T$, then the local list recovery algorithm for $\mathsf{MULT}^{(m)}_{q,s}(d)$ runs in time $T \cdot \mathsf{poly}(t)$.*

Theorem 4.1 follows by instantiating the above lemma with the global list recovery algorithm for univariate multiplicity codes from Theorem 3.8.

*Proof of Theorem 4.1.* Applying Theorem 3.8 with $n = q$ being a prime, and with parameters $\varepsilon/2$ and $\ell$, and noting that $s \geq \frac{64\ell}{\varepsilon^2} = \frac{16\ell}{(\varepsilon/2)^2}$ and $q \geq s \geq \frac{8\ell}{\varepsilon} = \frac{4\ell}{\varepsilon/2}$ by the assumptions of Theorem 4.1, we conclude that the whole-field univariate multiplicity code $\mathsf{MULT}^{(1)}_{q,s}(d)$ is $(\delta - \frac{\varepsilon}{2}, \ell, L)$-(globally) list recoverable with

$$L = \left(\frac{d}{q} \cdot \frac{\ell}{\varepsilon(1-\delta)}\right)^{O\left(\frac{d}{q} \cdot \frac{\ell}{\varepsilon^2(1-\delta)} \cdot \log\left(\frac{\ell}{1-\delta}\right)\right)} \leq \left(\frac{s\ell}{\varepsilon}\right)^{c_0 \cdot \frac{s\ell}{\varepsilon^2} \cdot \log\left(\frac{\ell}{1-\delta}\right)},$$

where the inequality holds for a sufficiently large constant $c_0$, recalling that $\delta = 1 - \frac{d}{sq}$.

Applying Lemma 4.2 with parameters $\alpha = \delta - \varepsilon/2$, $\ell$, and $\varepsilon/2$, we conclude that the multivariate multiplicity code $\mathsf{MULT}^{(m)}_{q,s}(d)$ is $\left(t, \delta - \varepsilon, \ell, O(m^2 L)\right)$-locally list recoverable for $t = q^2 \cdot \left(\frac{s \cdot m \cdot L}{\delta - \varepsilon}\right)^{O(m)}$.

Moreover, by Theorem 3.8 and Lemma 4.2, the local list-recovery algorithm runs in time at most

$$\mathsf{poly}(s, d, q, L) \cdot \mathsf{poly}(t) \leq \mathsf{poly}(q, s, L) \cdot \mathsf{poly}(t) \leq \mathsf{poly}(t),$$

where the first inequality follows by assumption that $d < sq$, and the second inequality follows by our setting of $t = q^2 \cdot \left(\frac{s \cdot m \cdot L}{\delta - \varepsilon}\right)^{O(m)}$. $\square$

The rest of this section is devoted to the proof of Lemma 4.2. We first give a short overview of the approach, and state the required preliminaries. We then flesh out the details in the subsequent three subsections.

**Overview.** Recall that in the local list recovery problem for the multivariate multiplicity code $\mathsf{MULT}^{(m)}_{q,s}(d)$ one is given an oracle access to a function $S : \mathbb{F}_q^m \to \binom{\Sigma_{m,s}}{\ell}$ (think of this function as assigning to each element of $\mathbb{F}_q^m$ a list of size $\ell$ of alphabet symbols $\Sigma_{m,s}$ of the multiplicity code), and the goal is to output a list of oracle machines, satisfying the following requirement: For any degree $d$ polynomial $Q(X_1, \ldots, X_m)$ that "agrees" with most of the input lists of $S$ (think of $Q$ to represent a true codeword of the multiplicity code), with high probability there exists an oracle machine in the output list that is a local corrector for $Q$ (that is, for any input coordinate $\mathbf{x} \in \mathbb{F}_q^m$, the oracle machine makes a few queries to the input lists $S$, and outputs $Q^{(<s)}(\mathbf{x})$ with high probability).

The local list recovery algorithm has three main subroutines that we will describe and analyze in the next three subsections. Briefly, the three components are the following:

1. A subroutine $\mathsf{RecoverCandidates}$, given in Section 4.1. $\mathsf{RecoverCandidates}$ takes as input a point $\mathbf{y} \in \mathbb{F}_q^m$, has query access to the collection of input lists $S$, and returns a short list $Z$ of guesses for $Q^{(<\tilde{s})}(\mathbf{y})$, where $\tilde{s} \geq s$ is some parameter. The goal of this subroutine will be two-fold. First, it will be used by the oracle machine $M^S[\mathbf{a}, z]$, described in Step 2 below, with $\mathbf{y} = \mathbf{x}$ and $\tilde{s} = s$, to generate a list that is more likely to contain the value $Q^{(<s)}(\mathbf{x})$ of the input coordinate $\mathbf{x}$, than the initial input list $S(\mathbf{x})$. Second, it will be used by the main local list recovery algorithm $\mathsf{LocalListRecoverMULT}$, described in Step 3 below, with $\mathbf{y} = \mathbf{a}$ and $\tilde{s} \gg s$ to generate a short list of possibilities for the value $Q^{(<\tilde{s})}(\mathbf{a})$ of the advice point $\mathbf{a}$.
2. A randomized oracle machine $M^S[\mathbf{a}, z]$ to evaluations of $Q^{(<s)}$, given in Section 4.2. The oracle machine $M^S[\mathbf{a}, z]$ is defined using an advice string $(\mathbf{a}, z)$, has query access to $S$, and

with high probability over the choice of a random point $\mathbf{a}$, we will have that $M^S[\mathbf{a}, Q^{(<\tilde{s})}(\mathbf{a})]$ recovers most of the points of $Q^{(<s)}$.

3. The final local list recovery algorithm $\mathsf{LocalListRecoverMULT}$, given in Section 4.3. The algorithm $\mathsf{LocalListRecoverMULT}$ first runs $\mathsf{RecoverCandidates}$ on a random point $\mathbf{a} \in \mathbb{F}_q^m$ to generate a short list $Z$ of possibilities for $Q^{(<\tilde{s})}(\mathbf{a})$. Then, for each $z \in Z$, it forms the oracle machine $M^S[\mathbf{a}, z]$. At this point it would be tempting to output the list of these oracle machines, but we are not quite done: even if $z = Q^{(<\tilde{s})}(\mathbf{a})$ corresponds to the correct advice and the choice of $\mathbf{a}$ is good, for some small fraction of points $\mathbf{x}$, we may still have $M^S[\mathbf{a}, z](\mathbf{x}) \neq Q^{(<s)}(\mathbf{x})$. Fortunately, for most $\mathbf{x}$ this will not be the case, and so we can implement the local correction algorithm of [KSY14] for multiplicity codes on top of $M^S[\mathbf{a}, z]$. This will give us our final list of randomized oracle algorithms that the local list recovery algorithm returns.

**Preliminaries.** Next we fix some notation, and state a couple of claims that relate the *global* properties of a multivariate polynomial to its *local* properties when restricted to a line. In what follows, for a multivariate polynomial $Q(X_1, \ldots, X_m) \in \mathbb{F}_q[X_1, \ldots, X_m]$, $\mathbf{a}, \mathbf{b} \in \mathbb{F}_q^m$, and a line $\lambda(T) = \mathbf{a} + T\mathbf{b}$, we let $(Q|_\lambda)(T) = Q(\lambda(T)) \in \mathbb{F}_q[T]$ denote the restriction of $Q$ to the line $\lambda(T)$.

The following claim gives a *chain rule* for Hasse derivatives, which relates the directional derivatives of a multivariate polynomial on a line to its global derivatives. Recall that for $\mathbf{i} = (i_1, \ldots, i_m) \in \mathbb{N}^m$ and $\mathbf{b} = (b_1, \ldots, b_m) \in \mathbb{F}_q^m$, we use the notation $\mathbf{b}^{\mathbf{i}} = b_1^{i_1} \cdots b_m^{i_m}$.

**Claim 4.3** (Chain rule for Hasse derivatives). *Let $q$ be a prime power, and let $m$ be a nonnegative integer. Let $Q(X_1, \ldots, X_m) \in \mathbb{F}_q[X_1, \ldots, X_m]$, let $\mathbf{a}, \mathbf{b} \in \mathbb{F}_q^m$, and let $\lambda(T) = \mathbf{a} + T\mathbf{b}$ be the line passing through $\mathbf{a}$ in direction $\mathbf{b}$.*

*Then for any integer $i \geq 0$ and $T \in \mathbb{F}_q$,*

$$(Q|_\lambda)^{(i)}(T) = \sum_{\mathbf{i}:\mathsf{wt}(\mathbf{i})=i} Q^{(\mathbf{i})}(\mathbf{a} + T\mathbf{b}) \cdot \mathbf{b}^{\mathbf{i}}. \tag{4}$$

Recall that we denote the alphabet of the multivariate multiplicity code $\mathsf{MULT}_{q,s}^{(m)}(d)$ by $\Sigma_{m,s} = \mathbb{F}_q^{U_{m,s}}$, where $U_{m,s} = \{\mathbf{i} \in \mathbb{N}^m \mid \mathsf{wt}(\mathbf{i}) < s\}$. Motivated by the above chain rule, for an element $z \in \Sigma_{m,s}$, and a direction $\mathbf{b} \in \mathbb{F}_q^m$, we define the restriction $z|_{\mathbf{b}}$ of $z$ to direction $\mathbf{b}$ to equal:

$$z|_{\mathbf{b}} := \left(h^{(0)}, \ldots, h^{(s-1)}\right) \in \Sigma_{1,s}, \qquad h^{(i)} = \sum_{\mathbf{i}:\mathsf{wt}(\mathbf{i})=i} z^{(\mathbf{i})}\mathbf{b}^{\mathbf{i}} \text{ for } i = 0, 1, \ldots, s-1. \tag{5}$$

Note that in the above notation, (4) can be rephrased as

$$(Q|_\lambda)^{(<s)}(T) = \left(Q^{(<s)}(\mathbf{a} + T\mathbf{b})\right)|_{\mathbf{b}}. \tag{6}$$

Finally, for a function $S : \mathbb{F}_q^m \to \binom{\Sigma_{m,s}}{\ell}$, assigning to each element of $\mathbb{F}_q^m$ a list of size $\ell$ of alphabet symbols $\Sigma_{m,s}$, and a line $\lambda(T) = \mathbf{a} + T\mathbf{b}$, we let $S|_\lambda : \mathbb{F}_q^m \to \binom{\Sigma_{1,s}}{\ell}$ equal

$$(S|_\lambda)(T) := \{z|_{\mathbf{b}} \mid z \in S(\lambda(T))\}. \tag{7}$$

The next claim relates the agreement of a multivariate polynomial $Q(X_1, \dots, X_m)$ with the input lists $S : \mathbb{F}_q^m \to \binom{\Sigma_{m,s}}{\ell}$, to the agreement of $Q|_\lambda$ with $S|_\lambda$ on a random line $\lambda$.

**Claim 4.4.** *Let $q$ be a prime power, and let $s, m$ be nonnegative integers. Let $\alpha \in (0,1)$ and $\ell \in \mathbb{N}$, and let $\varepsilon > 0$ be a parameter. Let $S : \mathbb{F}_q^m \to \binom{\Sigma_{m,s}}{\ell}$, and suppose that $Q(X_1, \dots, X_m) \in \mathbb{F}_q[X_1, \dots, X_m]$ is a polynomial such that:*

$$\Pr_{\mathbf{x} \in \mathbb{F}_q^m}\left[Q^{(<s)}(\mathbf{x}) \in S(\mathbf{x})\right] > 1 - \alpha + \varepsilon.$$

*Let $\mathbf{a}, \mathbf{b}$ be uniform random points in $\mathbb{F}_q^m$, and let $\lambda(T) = \mathbf{a} + T\mathbf{b}$ be the line passing through $\mathbf{a}$ in direction $\mathbf{b}$.*

*Then with probability at least $1 - \frac{1}{\varepsilon^2 q}$ over the choice of uniform random $\mathbf{a}, \mathbf{b} \in \mathbb{F}_q^m$, we have that:*

$$\Pr_{T \in \mathbb{F}_q}\left[(Q|_\lambda)^{(<s)}(T) \in (S|_\lambda)(T)\right] > 1 - \alpha. \tag{8}$$

*Proof.* By (6) and (7) we have that $(Q|_\lambda)^{(<s)}(T) \in (S|_\lambda)(T)$ if $Q^{(<s)}(\mathbf{a} + T\mathbf{b}) \in S(\mathbf{a} + T\mathbf{b})$. Thus, the event (8) holds if $Q^{(<s)}(\mathbf{x}) \in S(\mathbf{x})$ for more than $(1-\alpha)$-fraction of points $\mathbf{x}$ on the line $\lambda(T)$. The claim then follows from a standard application of Chebyshev's inequality, using the fact that the points on a uniformly random line are pairwise independent.

More precisely, for $t \in \mathbb{F}_q$, let $Y_t$ be the indicator random variable for the event that $Q^{(<s)}(\lambda(t)) \notin S(\lambda(t))$, and note that these random variables are pairwise independent with $\mathbb{E}[Y_t] \le \alpha - \varepsilon$ for any $t \in \mathbb{F}_q$. Then we have that the probability over the choice of uniform random $\mathbf{a}, \mathbf{b} \in \mathbb{F}_q^m$ that (8) does not hold is at most:

$$\Pr_{\mathbf{a},\mathbf{b} \in \mathbb{F}_q^m}\left[\sum_{t \in \mathbb{F}_q} \mathbf{1}\left\{Q^{(<s)}(\lambda(t)) \notin S(\lambda(t))\right\} > \alpha q\right] = \Pr_Y\left[\sum_{t \in \mathbb{F}_q} Y_t > \alpha q\right] \le \Pr_Y\left[\sum_{t \in \mathbb{F}_q} (Y_t - \mathbb{E}Y_t) > \varepsilon \cdot q\right].$$

Then by Chebyshev's inequality, this last quantity is at most

$$\frac{\sum_{t \in \mathbb{F}_q} \mathbb{E}(Y_t - \mathbb{E}Y_t)^2}{\varepsilon^2 q^2} \le \frac{1}{\varepsilon^2 q}.$$

□

## 4.1 The algorithm RecoverCandidates

As an important subroutine of the local list recovery algorithm, we will implement an algorithm, which we call RecoverCandidates. The main feature of this algorithm is that given oracle access to small input lists, which agree with the evaluations of order $s$ derivatives of $Q$ on a large fraction of the coordinates, it can output for *most* coordinates (in particular, a larger fraction than the initial agreement of the input lists with $Q$), a small list that agrees with evaluations of order $\tilde{s}$ derivatives of $Q$ (where $\tilde{s} \ge s$ is some parameter). This subroutine will later be used for two different purposes. First, it will be used by the oracle machine $M^S[\mathbf{a}, z]$ with $\tilde{s} = s$ to generate a list that is more likely to contain the value $Q^{(<s)}(\mathbf{x})$ of the input coordinate $\mathbf{x}$, than the initial input list $S(\mathbf{x})$. Second, it will be used by the main local list recovery algorithm $\mathsf{LocalListRecoverMULT}$ with $\tilde{s} \gg s$ to generate a short list of possibilities for the value $Q^{(<\tilde{s})}(\mathbf{a})$ of the advice point $\mathbf{a}$.

Specifically, the algorithm $\mathsf{RecoverCandidates}$ will have oracle access to a function $S : \mathbb{F}_q^m \to \binom{\Sigma_{m,s}}{\ell}$. Now suppose that $Q$ is an $m$-variate polynomial of degree at most $d$ that "agrees" with at least $1 - \alpha + \varepsilon$ fraction of these lists. On input $\mathbf{x}$, a random element of $\mathbb{F}_q^m$, and for some parameter $\tilde{s}$, the algorithm $\mathsf{RecoverCandidates} \geq s$ will make few queries to $S$ and output a small list $Z \subseteq \Sigma_{m,\tilde{s}}$, such that with high probability (over the choice of $\mathbf{x}$ and the randomness of the algorithm), the list $Z$ contains $Q^{(<\tilde{s})}(\mathbf{x})$.

**Lemma 4.5.** *Let $q$ be a prime power, and let $s, d, m$ be nonnegative integers so that $d < sq$. Let $\alpha \in (0, 1)$ and $\ell \in \mathbb{N}$, and suppose that $\mathsf{MULT}_{q,s}^{(1)}(d)$ is $(\alpha, \ell, L)$-(globally) list recoverable. Let $\tilde{s} \geq s$ be a parameter, and suppose that $q > 8 \cdot m^2 \cdot L \cdot \tilde{s}$. Let $\varepsilon > 0$ be a parameter.*

*Let $S : \mathbb{F}_q^m \to \binom{\Sigma_{m,s}}{\ell}$, and suppose that $Q(X_1, \ldots, X_m) \in \mathbb{F}_q[X_1, \ldots, X_m]$ is a polynomial of degree at most $d$ such that:*

$$\Pr_{\mathbf{x}\in\mathbb{F}_q^m}\left[Q^{(<s)}(\mathbf{x}) \in S(\mathbf{x})\right] > 1 - \alpha + \varepsilon.$$

*There is an algorithm $\mathsf{RecoverCandidates}$ which on input $\mathbf{x} \in \mathbb{F}_q^m$, and given oracle access to $S$, makes at most $t := q \cdot (\tilde{s} \cdot m \cdot L)^{O(m)}$ queries to $S$, and outputs a list $Z \subseteq \Sigma_{m,\tilde{s}}$ of size at most $8m^2L$ such that*

$$\Pr\left[Q^{(<\tilde{s})}(\mathbf{x}) \in Z\right] \geq 1 - \frac{4}{\varepsilon^2 q},$$

*where the probability is over the choice of a uniform random $\mathbf{x} \in \mathbb{F}_q^m$ and the randomness of the algorithm $\mathsf{RecoverCandidates}$.*

*Moreover, if $\mathsf{MULT}_{q,s}^{(1)}(d)$ can be probabilistically list-recovered with success probability at least $1 - \frac{1}{q}$ in time $T$, then the procedure $\mathsf{RecoverCandidates}$ runs in time $T \cdot \mathsf{poly}(t)$.*

The high level idea of the algorithm is as follows. On input $\mathbf{x}$, we take several random lines passing through $\mathbf{x}$, whose directions lie on a *random grid*, and run the univariate multiplicity list recovery algorithm on the restrictions of the input lists to those lines. This gives us, for each of these lines, a list of univariate polynomials. For a given line, this list of univariate polynomials contains candidates for $Q$ restricted to that line. In particular, this gives us candidate values for $Q(\mathbf{x})$ and all the higher order directional derivatives of $Q$ at $\mathbf{x}$ in the directions of those lines. We combine this information about the different directional derivatives to reconstruct $Q^{(<\tilde{s})}(\mathbf{x})$.

This combination turns out to be a certain kind of polynomial list recovery problem: namely list recovery for *tuples* of polynomials on a *grid*. The following lemma, which is proved in Appendix B, shows how this can be done algorithmically.

**Lemma 4.6** (List-recovering tuples of polynomials on a grid)**.** *Let $q$ be a prime power, let $d, m, \ell, t$ be nonnegative integers, and let $U$ be an arbitrary subset of $\mathbb{F}_q$ of size $r$.*

*Let $S : U^m \to \binom{\mathbb{F}_q^t}{\ell}$, and let $\mathcal{L}$ be the list of all $t$-tuples of polynomials $(H_1, \ldots, H_t)$, so that each $H_i \in \mathbb{F}_q[Y_1, \ldots, Y_m]$ is a polynomial of degree at most $d$, and*

$$\Pr_{\mathbf{u}\in U^m}\left[(H_1(\mathbf{u}), \ldots, H_t(\mathbf{u})) \in S(\mathbf{u})\right] > \sqrt{\frac{2d\ell}{r}}.$$

*Then* $|\mathcal{L}| \le \frac{r}{d}$.

*Moreover, if* $\mathcal{L}$ *is the list of all* $t$*-tuples of polynomials* $(H_1, \ldots, H_t)$*, so that each* $H_i \in \mathbb{F}_q[Y_1, \ldots, Y_m]$ *is a polynomial of degree at most* $d$*, and*

$$\Pr_{\mathbf{u} \in U^m} [(H_1(\mathbf{u}), \ldots, H_t(\mathbf{u})) \in S(\mathbf{u})] > m \cdot \sqrt{\frac{2d\ell}{r}},$$

*then there is a* $\mathsf{poly}(\log q, (d \cdot m \cdot r)^m, \ell, t)$*-time algorithm* GridListRecover *which computes* $\mathcal{L}$*.*

We will use GridListRecover as a subroutine of RecoverCandidates. Now we present our main subroutine RecoverCandidates, and analyze it below.

**Main Subroutine** RecoverCandidates**.**

- Oracle access to $S : \mathbb{F}_q^m \to \binom{\Sigma_{m,s}}{\ell}$.
- **INPUT:** $\mathbf{x} \in \mathbb{F}_q^m$, parameter $\tilde{s} \in \mathbb{N}$.
- The goal is to recover a small list of candidates for $Q^{(<\tilde{s})}(\mathbf{x})$.

1. Let $U$ be an arbitrary subset of $\mathbb{F}_q$ of size $r := 8 \cdot \tilde{s} \cdot m^2 \cdot L$.
2. Let $\mathbf{b} \in \mathbb{F}_q^m$ be picked uniformly at random.
3. For each $\mathbf{u} \in U^m$:
   (a) Let $\lambda_{\mathbf{u}}(T)$ be the line $\lambda_{\mathbf{u}}(T) = \mathbf{x} + T(\mathbf{b} + \mathbf{u})$.
   (b) Run the randomized $(\alpha, \ell, L)$-list recovery algorithm for $\mathsf{MULT}^{(1)}_{q,s}(d)$ of success probability at least $1 - \frac{1}{q}$ given by Theorem 3.8 on $S|_{\lambda_{\mathbf{u}}}$, and let $\mathcal{L}_{\mathbf{u}} \subseteq \mathbb{F}_q[T]$ denote the resulting output list of univariate polynomials.
4. Define a function $\tilde{S} : U^m \to \binom{\mathbb{F}_q^{\tilde{s}}}{L}$ by

$$\tilde{S}(\mathbf{u}) = \left\{ P^{(<\tilde{s})}(0) \mid P(T) \in \mathcal{L}_{\mathbf{u}} \right\} \tag{9}$$

   for any $\mathbf{u} \in U^m$.
5. Run the GridListRecover algorithm given by Lemma 4.6 to recover the list $\tilde{\mathcal{L}}$ of all $\tilde{s}$-tuples of polynomials $(H_0, \ldots, H_{\tilde{s}-1})$, so that each $H_i \in \mathbb{F}_q[Y_1, \ldots, Y_m]$ is a polynomial of degree at most $\tilde{s}$, and

$$\Pr_{\mathbf{u} \in U^m} \left[ (H_0(\mathbf{u}), \ldots, H_{\tilde{s}-1}(\mathbf{u})) \in \tilde{S}(\mathbf{u}) \right] \ge \frac{1}{2}. \tag{10}$$

6. Let $Z$ be the set containing all $z \in \Sigma_{m,\tilde{s}}$ so that $(H_0, \ldots, H_{\tilde{s}-1}) \in \tilde{\mathcal{L}}$, where

$$H_i(\mathbf{Y} - \mathbf{b}) = \sum_{\mathbf{i} : \mathsf{wt}(\mathbf{i}) = i} z^{(\mathbf{i})} \mathbf{Y}^{\mathbf{i}} \tag{11}$$

```
    for $i = 0, 1, \ldots, \tilde{s} - 1$.
7. Return $Z$.
```

To analyze the above algorithm, we first prove the following two claims. The first claim uses Claim 4.4 to show that with high probability, for a large fraction of $\mathbf{u} \in U^m$ the restriction $(Q|_{\lambda_{\mathbf{u}}})(T)$ of $Q$ to the line $\lambda_u$ is contained in the output list $\mathcal{L}_{\mathbf{u}}$.

**Claim 4.7.** *With probability at least* $1 - \frac{4}{\varepsilon^2 q}$ *over the choice of uniform random* $\mathbf{x}, \mathbf{b} \in \mathbb{F}_q^m$*, and the randomness of the univariate list-recovery algorithms, we have that*

$$\Pr_{\mathbf{u} \in U^m} \left[ (Q|_{\lambda_{\mathbf{u}}})(T) \in \mathcal{L}_{\mathbf{u}} \right] \geq \frac{1}{2}.$$

*Proof.* First note that for any $\mathbf{u} \in \mathbb{F}_q^m$, and for a uniform random choice of $\mathbf{x}, \mathbf{b} \in \mathbb{F}_q^m$, the line $\lambda_{\mathbf{u}}(T) = \mathbf{x} + T(\mathbf{b} + \mathbf{u})$ is uniform random. Thus, by Claim 4.4 we have that for any $\mathbf{u} \in \mathbb{F}_q^m$, with probability at least $1 - \frac{1}{\varepsilon^2 q}$ over the choice of $\mathbf{x}, \mathbf{b} \in \mathbb{F}_q^m$, it holds that:

$$\Pr_{T \in \mathbb{F}_q} \left[ (Q|_{\lambda_{\mathbf{u}}})^{(<s)}(T) \in (S|_{\lambda_{\mathbf{u}}})(T) \right] > 1 - \alpha,$$

in which case $(Q|_{\lambda_{\mathbf{u}}})(T) \in \mathcal{L}_{\mathbf{u}}$ with probability at least $1 - \frac{1}{q}$ over the randomness of the univariate list-recovery algorithm on $\lambda_{\mathbf{u}}$. We conclude that for any $\mathbf{u} \in \mathbb{F}_q^m$, $(Q|_{\lambda_{\mathbf{u}}})(T) \in \mathcal{L}_{\mathbf{u}}$ with probability at least $1 - \frac{2}{\varepsilon^2 q}$ over the random choice of $\mathbf{x}, \mathbf{b} \in \mathbb{F}_q^m$, and the randomness of the univariate list-recovery algorithms.

By Markov's inequality, the above implies in turn that $(Q|_{\lambda_{\mathbf{u}}})(T) \in \mathcal{L}_{\mathbf{u}}$ for at least a $\frac{1}{2}$-fraction of $\mathbf{u} \in U^m$ with probability at least $1 - \frac{4}{\varepsilon^2 q}$ over the choice of uniform random $\mathbf{x}, \mathbf{b} \in \mathbb{F}_q^m$, and the randomness of the univariate list-recovery algorithms. □

The second claim uses the chain rule (Claim 4.3) to define a tuple of polynomials $(H_0, \ldots, H_{\tilde{s}-1})$ so that $(H_0(\mathbf{u}), \ldots, H_{\tilde{s}-1}(\mathbf{u})) \in \tilde{S}(\mathbf{u})$ for any $\mathbf{u} \in U^m$ which satisfies that $(Q|_{\lambda_{\mathbf{u}}})(T) \in \mathcal{L}_{\mathbf{u}}$.

**Claim 4.8.** *For any* $i = 0, 1, \ldots, \tilde{s} - 1$*, let*

$$H_i(\mathbf{Y}) = \sum_{\mathbf{i}:\mathsf{wt}(\mathbf{i})=i} Q^{(\mathbf{i})}(\mathbf{x}) \cdot (\mathbf{Y} + \mathbf{b})^{\mathbf{i}}. \tag{12}$$

*Then for any* $\mathbf{u} \in U^m$ *which satisfies that* $(Q|_{\lambda_{\mathbf{u}}})(T) \in \mathcal{L}_{\mathbf{u}}$*, we have that*

$$(H_0(\mathbf{u}), \ldots, H_{\tilde{s}-1}(\mathbf{u})) \in \tilde{S}(\mathbf{u}).$$

*Proof.* Fix any $\mathbf{u} \in U^m$ which satisfies that $(Q|_{\lambda_{\mathbf{u}}})(T) \in \mathcal{L}_{\mathbf{u}}$. First note that by the definition of $\tilde{S}$ given in (9), we have that $(Q|_{\lambda_{\mathbf{u}}})^{(<\tilde{s})}(0) \in \tilde{S}(\mathbf{u})$. Moreover, by the chain rule (Claim 4.3), we have that

$$(Q|_{\lambda_{\mathbf{u}}})^{(i)}(0) = \sum_{\mathbf{i}:\mathsf{wt}(\mathbf{i})=i} Q^{(\mathbf{i})}(\mathbf{x}) \cdot (\mathbf{b} + \mathbf{u})^{\mathbf{i}} = H_i(\mathbf{u})$$

for any $i = 0, 1, \ldots, \tilde{s} - 1$. Consequently, we have that

$$(H_0(\mathbf{u}), \ldots, H_{\tilde{s}-1}(\mathbf{u})) = (Q|_{\lambda_{\mathbf{u}}})^{(<\tilde{s})}(0) \in \tilde{S}(\mathbf{u}).$$

□

We now prove Lemma 4.5, based on the above Claims 4.7 and 4.8.

*Proof of Lemma 4.5.* By Claim 4.7, we have that with probability at least $1 - \frac{4}{\varepsilon^2 q}$ over the choice of uniform random $\mathbf{x}, \mathbf{b} \in \mathbb{F}_q^m$, and the randomness of the univariate list-recovery algorithms, it holds that

$$\Pr_{\mathbf{u} \in U^m}\left[(Q|_{\lambda_\mathbf{u}})(T) \in \mathcal{L}_\mathbf{u}\right] \geq \frac{1}{2}.$$

By Claim 4.8, we have that $(H_0(\mathbf{u}), \ldots, H_{\tilde{s}-1}(\mathbf{u})) \in \tilde{S}(\mathbf{u})$ for each $\mathbf{u} \in U^m$ satisfying that $(Q|_{\lambda_\mathbf{u}})(T) \in \mathcal{L}_\mathbf{u}$, where $(H_0, \ldots, H_{\tilde{s}-1})$ are as given in (12). We conclude that with probability at least $1 - \frac{4}{\varepsilon^2 q}$ over the choice of uniform random $\mathbf{x}, \mathbf{b} \in \mathbb{F}_q^m$, and the randomness of the univariate list-recovery algorithms, it holds that

$$\Pr_{\mathbf{u} \in U^m}\left[(H_0(\mathbf{u}), \ldots, H_{\tilde{s}-1}(\mathbf{u})) \in \tilde{S}(\mathbf{u})\right] \geq \frac{1}{2} = m \cdot \sqrt{\frac{2\tilde{s}L}{r}},$$

where the last equality follows by choice of $r = 8 \cdot \tilde{s} \cdot m^2 \cdot L$.

By the definition of $\tilde{\mathcal{L}}$ given in (10), and applying Lemma 4.6 with $\tilde{S} : U^m \to \binom{\mathbb{F}_q^{\tilde{s}}}{L}$, and with parameters $d = \tilde{s}$, $m$, $\ell = L$, $t = \tilde{s}$, and $r = 8 \cdot \tilde{s} \cdot m^2 \cdot L$, the above implies in turn that $(H_0, \ldots, H_{\tilde{s}-1}) \in \tilde{\mathcal{L}}$, where $H_i(\mathbf{Y}) = Q^{(\mathbf{i})}(\mathbf{x}) \cdot (\mathbf{Y} + \mathbf{b})^{\mathbf{i}}$ for $i = 0, 1, \ldots \tilde{s} - 1$. It then follows by the definition of $Z$ given in (11) that the list $Z$ will contain $Q^{(<\tilde{s})}(\mathbf{x})$.

It remains to show the claimed query complexity, output list size, and running time. The algorithm $\mathsf{RecoverCandidates}$ runs the global list recovery algorithm for the univariate multiplicity codes on $|U|^m = (\tilde{s} \cdot m \cdot L)^{O(m)}$ lines, and so the query complexity is at most $t := q \cdot (\tilde{s} \cdot m \cdot L)^{O(m)}$. The output list size is at most the size of $\tilde{\mathcal{L}}$, which can be bounded by $8m^2L$ by applying Lemma 4.6 with $\tilde{S} : U^m \to \binom{\mathbb{F}_q^{\tilde{s}}}{L}$, and with parameters $d = \tilde{s}$, $m$, $\ell = L$, $t = \tilde{s}$, and $r = 8 \cdot \tilde{s} \cdot m^2 \cdot L$.

We now turn to compute the running time. On Step 3, for each $\mathbf{u} \in U^m$, the algorithm needs to compute the restriction $S|_{\lambda_\mathbf{u}}$ of $S$ to $\lambda_\mathbf{u}$, which takes time $O(q \cdot \log(|\Sigma_{m,s}|)) \leq \mathsf{poly}(q, (sm)^m) \leq \mathsf{poly}(t)$ (recalling our assumption that $\tilde{s} \geq s$), and to run the univariate list-recovery algorithm on $S|_{\lambda_\mathbf{u}}$, which takes time $T$. Hence the total running time of Step 3 is at most $|U|^m \cdot (T + \mathsf{poly}(t)) \leq T \cdot \mathsf{poly}(t)$ (recalling that $t > |U|^m$). On Step 4, for each $\mathbf{u} \in U^m$ and $P \in \mathcal{L}_\mathbf{u}$, the algorithm needs to compute $P^{(<\tilde{s})}(0)$, which takes time at most $\mathsf{poly}(\log q, \tilde{s}, d) \leq \mathsf{poly}(q, \tilde{s}) \leq \mathsf{poly}(t)$ (recalling our assumptions that $\tilde{s} \geq s$ and $d < qs$). So the total running time of Step 4 is at most $|U|^m \cdot L \cdot \mathsf{poly}(t) \leq \mathsf{poly}(t)$. By Lemma 4.6, Step 5 can be performed in time $(\tilde{s} \cdot m \cdot L)^{O(m)} \cdot \mathsf{polylog}(q) \leq \mathsf{poly}(t)$. Finally, Step 6 can be performed in time at most $O(|\tilde{\mathcal{L}}| \cdot \log(|\Sigma_{m,\tilde{s}}|)) \leq \mathsf{poly}(t)$. This results in the claimed running time of at most $T \cdot \mathsf{poly}(t)$. $\square$

## 4.2 The Oracle Machine $M$

The oracle machine $M$ is defined by an advice $(\mathbf{a}, z)$, where $\mathbf{a} \in \mathbb{F}_q^m$, and $z \in \Sigma_{m,\tilde{s}}$ is meant to be a guess for $Q^{(<\tilde{s})}(\mathbf{a})$. Given this advice, the oracle machine works as follows: on input $\mathbf{x}$, it will run the univariate list recovery algorithm on the line $\lambda(T) = \mathbf{x} + T(\mathbf{a} - \mathbf{x})$ through $\mathbf{x}$ and $\mathbf{a}$ to obtain a list $\mathcal{L}$ of univariate polynomials $P(T)$. We will show that with high probability (assuming the advice is good), there will be a unique polynomial $P_*(T)$ in $\mathcal{L}$ so that $(P_*)^{(<\tilde{s})}(1)$ is consistent with $z$. The oracle machine then runs the RecoverCandidates procedure on input $\mathbf{x}$ with parameter $\tilde{s} = s$ to generate a list $Y \subseteq \Sigma_{m,s}$ that is more likely to contain the value $Q^{(<s)}(\mathbf{x})$ of the input coordinate $\mathbf{x}$, than the initial input list $S(\mathbf{x})$. We then show that with high probability, there will be a unique value in $Y$ that is consistent with $(P_*)^{(<s)}(0)$. The oracle machine will then output the symbol in $Y$ that $(P_*)^{(<s)}(0)$ agrees with.

The key later will be that the advice $z$ will not vary over all possibilities in $\Sigma_{m,\tilde{s}}$; this would result in too long a list. Rather, we will use RecoverCandidates in order to generate this advice.

Formally, we will prove the following lemma about our oracle machine, which we define below.

**Lemma 4.9.** *Let $q$ be a prime power, and let $s, d, m$ be nonnegative integers so that $d < sq$. Let $\alpha \in (0,1)$ and $\ell \in \mathbb{N}$, and suppose that $\mathsf{MULT}^{(1)}_{q,s}(d)$ is $(\alpha, \ell, L)$-(globally) list recoverable. Let $\tilde{s} \geq s$ be a parameter, and suppose that $q > 8 \cdot m^2 \cdot L \cdot s$. Let $\varepsilon > 0$ be a parameter.*

*Let $S : \mathbb{F}_q^m \to \binom{\Sigma_{m,s}}{\ell}$, and suppose that $Q(X_1, \ldots, X_m) \in \mathbb{F}_q[X_1, \ldots, X_m]$ is a polynomial of degree at most $d$ such that:*

$$\Pr_{\mathbf{x} \in \mathbb{F}_q^m} \left[ Q^{(<s)}(\mathbf{x}) \in S(\mathbf{x}) \right] > 1 - \alpha + \varepsilon. \tag{13}$$

*There is an algorithm $M^S[\mathbf{a}, z](\mathbf{x})$ which on input $\mathbf{x} \in \mathbb{F}_q^m$, given as advice a point $\mathbf{a} \in \mathbb{F}_q^m$, and $z \in \Sigma_{m,\tilde{s}}$, and given oracle access to $S$, makes at most $t := q \cdot (s \cdot m \cdot L)^{O(m)}$ queries to $S$, and outputs an element of $\Sigma_{m,s} \cup \{\perp\}$ such that*

$$\Pr \left[ M^S[\mathbf{a}, Q^{(<\tilde{s})}(\mathbf{a})](\mathbf{x}) = Q^{(<s)}(\mathbf{x}) \right] \geq 1 - \frac{6}{\varepsilon^2 q} - \frac{8 \cdot s \cdot m^2 \cdot L}{q} - \frac{sL}{\tilde{s}}, \tag{14}$$

*where the probability is over the choice of uniform random $\mathbf{a}, \mathbf{x} \in \mathbb{F}_q^m$ and the randomness of the algorithm $M^S[\mathbf{a}, Q^{(<\tilde{s})}(\mathbf{a})](\mathbf{x})$.*

*Moreover, if $\mathsf{MULT}^{(1)}_{q,s}(d)$ can be probabilistically list-recovered with success probability at least $1 - \frac{1}{q}$ in time $T$, then the procedure $M^S[\mathbf{a}, z](\mathbf{x})$ runs in time $T \cdot \mathsf{poly}(t, (\tilde{s})^m)$.*

We will first describe the algorithm and then show that it satisfies the required properties.

**Oracle machine $M$.**

- Oracle access to $S : \mathbb{F}_q^m \to \binom{\Sigma_{m,s}}{\ell}$.
- **INPUT:** $\mathbf{x} \in \mathbb{F}_q^m$.
- **ADVICE:** Point $\mathbf{a} \in \mathbb{F}_q^m$, and $z \in \Sigma_{m,\tilde{s}}$.
- The goal is to recover $Q^{(<s)}(\mathbf{x})$.

1. Let $\mathbf{b} := \mathbf{a} - \mathbf{x}$, let $\lambda$ be the line $\lambda(T) = \mathbf{x} + T(\mathbf{a} - \mathbf{x}) = \mathbf{x} + T\mathbf{b}$.
2. Run the randomized $(\alpha, \ell, L)$-list recovery algorithm for $\mathsf{MULT}^{(1)}_{q,s}(d)$ of success probability

at least $1-\frac{1}{q}$ given by Theorem 3.8 on $S|_\lambda$, and let $\mathcal{L} \subseteq \mathbb{F}_q[T]$ denote the resulting output list of univariate polynomials.

3. If there exists exactly one polynomial $P \in \mathcal{L}$ with $P^{(<\tilde{s})}(1) = z|_{\mathbf{b}}$, let $P_*$ be that polynomial. Otherwise, output $\perp$ and exit.
4. Let $Y \subseteq \Sigma_{m,s}$ be the output of $\mathsf{RecoverCandidates}^S(\mathbf{x}, s)$.
5. If there exists exactly one value $y \in Y$ such that $(P_*)^{(<s)}(0) = y|_{\mathbf{b}}$, output that $y$. Otherwise, output $\perp$.

The correctness of the above algorithm relies on the following pair of claims. The first claim shows that with high probability, $P_* = Q|_\lambda$.

**Claim 4.10.** *Suppose that $z = Q^{(<\tilde{s})}(\mathbf{a})$. Then with probability at least $1 - \frac{2}{\varepsilon^2 q} - \frac{sL}{\tilde{s}}$ over the choice of uniform random $\mathbf{a}, \mathbf{x} \in \mathbb{F}_q^m$, and the randomness of the univariate list-recovery algorithm, it holds that $P_* = Q|_\lambda$.*

*Proof.* We first claim that with probability at least $1 - \frac{2}{\varepsilon^2 q}$ over the random choice of $\mathbf{a}, \mathbf{x} \in \mathbb{F}_q^m$, and the randomness of the univariate list-recovery algorithm, it holds that $(Q|_\lambda)(T) \in \mathcal{L}$. To see this, note first that when $\mathbf{a}, \mathbf{x}$ are uniformly random elements of $\mathbb{F}_q^m$, then $\lambda(T) = \mathbf{x} + T(\mathbf{a} - \mathbf{x})$ is a uniformly random line. Thus, by Claim 4.4 we have that with probability at least $1 - \frac{1}{\varepsilon^2 q}$ over the choice of $\mathbf{x}, \mathbf{a} \in \mathbb{F}_q^m$, it holds that:

$$\Pr_{T \in \mathbb{F}_q}\left[(Q|_\lambda)^{(<s)}(T) \in (S|_\lambda)(T)\right] > 1 - \alpha,$$

in which case $(Q|_\lambda)(T) \in \mathcal{L}$ with probability at least $1 - \frac{1}{q}$ over the randomness of the univariate list-recovery algorithm on $\lambda$. We conclude that $(Q|_\lambda)(T) \in \mathcal{L}$ with probability at least $1 - \frac{2}{\varepsilon^2 q}$ over the random choice of $\mathbf{a}, \mathbf{x} \in \mathbb{F}_q^m$, and the randomness of the univariate list-recovery algorithm.

By our assumption that $z = Q^{(<\tilde{s})}(\mathbf{a})$, we clearly have that $(Q|_\lambda)^{(<\tilde{s})}(1) = z|_{\mathbf{b}}$. Next we claim that with probability at least $1 - \frac{sL}{\tilde{s}}$ over the random choice of $\mathbf{a}, \mathbf{x} \in \mathbb{F}_q^m$, there is no $P \in \mathcal{L}$ so that $P \neq Q|_\lambda$ but $P^{(<\tilde{s})}(1) = (Q|_\lambda)^{(<\tilde{s})}(1)$. To see this, note that because of the way $\mathbf{x}$, $\mathbf{a}$ and the line $\lambda$ are sampled, equivalently one could let $\mathbf{x}$ be picked uniformly at random from $\mathbb{F}_q^m$, $\lambda$ be a uniformly random line through $\mathbf{x}$, and $\mathbf{a}$ be a uniformly random point on $\lambda$. Now fix any polynomial $P \in \mathcal{L}$ so that $P \neq Q|_\lambda$. We want to bound the probability that $P^{(<\tilde{s})}(\alpha) = (Q|_\lambda)^{(<\tilde{s})}(\alpha)$ where $\alpha \in \mathbb{F}_q$ is picked uniformly at random. But $P$ and $Q|_\lambda$ are fixed distinct polynomials of degree at most $d$. Thus the probability that they agree with multiplicity $\tilde{s}$ on a random point of $\mathbb{F}_q$ is at most $\frac{d}{\tilde{s}q} \leq \frac{s}{\tilde{s}}$, where the inequality follows by assumption that $d < sq$. By a union bound over all $P \in \mathcal{L}$ with $P \neq Q|_\lambda$, we conclude that with probability at least $1 - \frac{sL}{\tilde{s}}$, there is no $P \in \mathcal{L}$ so that $P \neq Q|_\lambda$ but $P^{(<\tilde{s})}(1) = (Q|_\lambda)^{(<\tilde{s})}(1)$.

Finally, by a union bound, we conclude that with probability at least $1 - \frac{2}{\varepsilon^2 q} - \frac{sL}{\tilde{s}}$ over the random choice of $\mathbf{a}, \mathbf{x} \in \mathbb{F}_q^m$, and the randomness of the univariate list-recovery algorithm, both events that $Q|_\lambda \in \mathcal{L}$ and that there is no $P \in \mathcal{L}$ so that $P \neq Q|_\lambda$ but $P^{(<\tilde{s})}(1) = (Q|_\lambda)^{(<\tilde{s})}(1)$ hold, in which case $P_* = Q|_\lambda$. $\square$

The next claim shows that with high probability, $Q^{(<s)}(\mathbf{x}) \in Y$, and there is no $y \in Y$ so that $y \neq Q^{(<s)}(\mathbf{x})$ but $y|_{\mathbf{b}} = (Q^{(<s)}(\mathbf{x}))|_{\mathbf{b}}$.

**Claim 4.11.** *With probability at least $1-\frac{4}{\varepsilon^2 q}-\frac{8\cdot s\cdot m^2\cdot L}{q}$ over the choice of uniform random $\mathbf{a}, \mathbf{x} \in \mathbb{F}_q^m$, and the randomness of the* RecoverCandidates *procedure, it holds that $Q^{(<s)}(\mathbf{x}) \in Y$, and there is no $y \in Y$ so that $y \neq Q^{(<s)}(\mathbf{x})$ but $y|_{\mathbf{b}} = (Q^{(<s)}(\mathbf{x}))|_{\mathbf{b}}$.*

*Proof.* First note that by the guarantees of the RecoverCandidates procedure (Lemma 4.5), with probability at least $1 - \frac{4}{\varepsilon^2 q}$ over the random choice of $\mathbf{x} \in \mathbb{F}_q^m$ and over the randomness of the RecoverCandidates procedure, it holds that $Q^{(<s)}(\mathbf{x}) \in Y$.

Fix $\mathbf{x} \in \mathbb{F}_q^m$, and let $y_0 := Q^{(<s)}(\mathbf{x})$. Let $Y \subseteq \Sigma_{m,s}$ be the output of $\mathsf{RecoverCandidates}^S(\mathbf{x}, s)$ on Step 4. Next we claim that with probability at least $1 - \frac{8\cdot s\cdot m^2\cdot L}{q}$ over the random choice of $\mathbf{a} \in \mathbb{F}_q^m$, it holds that there is no $y \in Y$ so that $y \neq y_0$ but $y|_{\mathbf{b}} = y_0|_{\mathbf{b}}$. To see this, first observe that for any fixed choice of $\mathbf{x}$, the randomness of $\mathbf{a}$ implies that $\mathbf{b} = \mathbf{a} - \mathbf{x}$ is a uniformly random element of $\mathbb{F}_q^m$. Now fix any $y \in Y$ so that $y \neq y_0$. We want to bound the probability that $y|_{\mathbf{b}} = y_0|_{\mathbf{b}}$ when $\mathbf{b} \in \mathbb{F}_q^m$ is chosen uniformly at random.

To this end, recall that by (5), for an element $y \in \Sigma_{m,s}$, and a direction $\mathbf{b} \in \mathbb{F}_q^m$, $y|_{\mathbf{b}} = \left(h^{(0)}, \ldots, h^{(s-1)}\right) \in \Sigma_{1,s}$, where:

$$h^{(i)} = \sum_{\mathbf{i}:\mathsf{wt}(\mathbf{i})=i} y^{(\mathbf{i})}\mathbf{b}^{\mathbf{i}}$$

for each $i$ such that $0 \leq i < s$. Note that $h^{(i)}$ can be viewed as a polynomial of degree at most $i$ evaluated at $\mathbf{b}$, where the coefficients of the polynomial depend only on $y$. Since $y \neq y_0$, the corresponding tuples of polynomials (each of degree at most $s$) will differ in at least one coordinate. Thus in the coordinate where the tuples of polynomials differ, the evaluations at a random element $\mathbf{b} \in \mathbb{F}_q^m$ will be distinct with probability at least $1 - \frac{s}{q}$ by Claim 2.7. By a union bound over all $y \in Y$ with $y \neq y_0$, we conclude that with probability at least $1 - \frac{|Y|s}{q}$, there is no $y \in Y$ so that $y \neq y_0$ but $y|_{\mathbf{b}} = y_0|_{\mathbf{b}}$. Finally, note that this probability is at least $1 - \frac{8\cdot s\cdot m^2\cdot L}{q}$, since by Lemma 4.5 the procedure RecoverCandidates returns a list $Y$ of size at most $8m^2L$.

Finally, by a union bound, we conclude that with probability at least $1 - \frac{4}{\varepsilon^2 q} - \frac{8\cdot s\cdot m^2\cdot L}{q}$ over the random choice of $\mathbf{a}, \mathbf{x} \in \mathbb{F}_q^m$, and the randomness of the RecoverCandidates procedure, it holds that $Q^{(<s)}(\mathbf{x}) \in Y$, and there is no $y \in Y$ so that $y \neq Q^{(<s)}(\mathbf{x})$ but $y|_{\mathbf{b}} = (Q^{(<s)}(\mathbf{x}))|_{\mathbf{b}}$.

$\square$

We now prove Lemma 4.9, based on the above Claims 4.10 and 4.11.

*Proof of Lemma 4.9.* Claim 4.10 shows that with probability at least $1 - \frac{2}{\varepsilon^2 q} - \frac{sL}{\tilde{s}}$ over the random choice of $\mathbf{a}, \mathbf{x} \in \mathbb{F}_q^m$, and the randomness of the univariate list-recovery algorithm, it holds that $P_* = Q|_\lambda$. Moreover, by Claim 4.11, with probability at least $1 - \frac{4}{\varepsilon^2 q} - \frac{8\cdot s\cdot m^2\cdot L}{q}$ over the random choice of $\mathbf{a}, \mathbf{x} \in \mathbb{F}_q^m$, and the randomness of the RecoverCandidates procedure, $Q^{(<s)}(\mathbf{x}) \in Y$, and there is no $y \in Y$ so that $y \neq Q^{(<s)}(\mathbf{x})$ but $y|_{\mathbf{b}} = (Q^{(<s)}(\mathbf{x}))|_{\mathbf{b}}$. By a union bound, we

have that both events happen with probability at least $1 - \frac{6}{\varepsilon^2 q} - \frac{8 \cdot s \cdot m^2 \cdot L}{q} - \frac{sL}{\tilde{s}}$, in which case $(P_*)^{(<s)}(0) = (Q^{(<s)}(\mathbf{x}))|_{\mathbf{b}}$, and $Q^{(<s)}(\mathbf{x})$ is the unique $y \in Y$ with $y|_{\mathbf{b}} = (Q^{(<s)}(\mathbf{x}))|_{\mathbf{b}}$, and so the oracle machine will output $Q^{(<s)}(\mathbf{x})$, as required.

It remains to show the claimed query complexity and running time. To this end, note that the oracle machine makes $q$ queries in Step 2, while in Step 4 the procedure RecoverCandidates, which is run with parameter $s$, makes at most $q \cdot (s \cdot m \cdot L)^{O(m)}$ queries. Hence the total query complexity is at most $t := q \cdot (s \cdot m \cdot L)^{O(m)}$.

We now turn to compute the running time. On Step 2, the algorithm needs to compute the restriction $S|_\lambda$ of $S$ to $\lambda$, which takes time $O(q \cdot \log(|\Sigma_{m,s}|)) \leq \mathsf{poly}(q, (s \cdot m)^m) \leq \mathsf{poly}(t)$, and to run the univariate list-recovery algorithm on $S|_\lambda$, which takes time $T$. Hence the total running time of Step 2 is at most $T + \mathsf{poly}(t)$. On Step 3, the algorithm needs to compute $z|_{\mathbf{b}}$, which takes time at most $O(\log(|\Sigma_{m,\tilde{s}}|)) \leq (m\tilde{s})^{O(m)} \cdot \log q \leq \mathsf{poly}(t, (\tilde{s})^m)$, and for each $P \in \mathcal{L}$, the algorithm needs to compute $P^{(<\tilde{s})}(1)$, which takes time at most $\mathsf{poly}(\log q, \tilde{s}, d) \leq \mathsf{poly}(q, s, \tilde{s}) \leq \mathsf{poly}(t, \tilde{s})$ (recalling our assumption that $d < qs$). So the total running time of Step 4 is at most $\mathsf{poly}(t, (\tilde{s})^m) + |L| \cdot \mathsf{poly}(t, \tilde{s}) \leq \mathsf{poly}(t, (\tilde{s})^m)$. By Lemma 4.5, Step 4 can be performed in time $T \cdot \mathsf{poly}(t)$. Finally, on Step 5, the algorithm needs to compute $(P_*)^{(<s)}(0)$, which takes time at most $\mathsf{poly}(\log q, s, d) \leq \mathsf{poly}(t)$, and for each $y \in Y$, the algorithm needs to compute $y|_{\mathbf{b}}$, which takes time at most $O(\log(|\Sigma_{m,s}|)) \leq \mathsf{poly}(t)$, and so the running time of this step is at most $L \cdot \mathsf{poly}(t) \leq \mathsf{poly}(t)$. This results in a running time of at most $T \cdot \mathsf{poly}(t, (\tilde{s})^m)$, as claimed.

□

## 4.3 Main local list recovery algorithm

Together, Lemmas 4.5 and 4.9 inspire a local list recovery algorithm for multivariate multiplicity codes. The idea is that RecoverCandidates will first obtain a list of possibilities, $Z$, for $Q^{(<\tilde{s})}(\mathbf{a})$. Then for each possibility $z \in Z$, we will create an oracle machine as in Lemma 4.9 which guesses $Q^{(<\tilde{s})}(\mathbf{a}) = z$. Unfortunately, this will still have some amount of error; that is, there will be some small fraction of $\mathbf{x} \in \mathbb{F}_q^m$ so that the approach above will not be correct on $\mathbf{x}$. To correct the remaining errors we use the following local correction algorithm for multivariate multiplicity codes from [KSY14].

**Theorem 4.12** ([KSY14], Theorem 3.6)**.** *Let $q$ be a prime power, let $s, d, m$ be nonnegative integers so that $d + 6s \leq sq$ and $q \geq \max\{10m, 12(s+1)\}$, and let $\delta := 1 - \frac{d}{sq}$.*

*Let $f : \mathbb{F}_q^m \to \Sigma_{m,s}$, and suppose that $Q(X_1, \ldots, X_m) \in \mathbb{F}_q[X_1, \ldots, X_m]$ is a polynomial of degree at most $d$ such that:*

$$\Pr_{\mathbf{x} \in \mathbb{F}_q^m}\left[Q^{(<s)}(\mathbf{x}) = f(x)\right] > 1 - \frac{\delta}{10}.$$

*There is an oracle machine* LocalCorrect*, which on input $\mathbf{x} \in \mathbb{F}_q^m$, and given oracle access to $f$, makes at most $t := (O(s)^m \cdot q)$ queries to $f$, and outputs an element of $\Sigma_{m,s}$ such that*

$$\Pr\left[\mathsf{LocalCorrect}^f(\mathbf{x}) = Q^{(<s)}(\mathbf{x})\right] \geq 0.9,$$

*where the probability is over the randomness of* LocalCorrect*.*

*Moreover, the local corrector can be*[9] *made to run in time* $\mathsf{poly}(t)$*.*

Assuming the above local-correction algorithm $\mathsf{LocalCorrect}$ for multivariate multiplicity codes in hand, we define our final local list recovery algorithm as follows.

```
Algorithm LocalListRecoverMULT.

  • Oracle access to $S : \mathbb{F}_q^m \to \binom{\Sigma_{m,s}}{\ell}$.

  1. Pick $\mathbf{a} \in \mathbb{F}_q^m$ uniformly at random, and set $\tilde{s} = \frac{3000sL}{\alpha}$.
  2. Let $Z \subseteq \Sigma_{m,\tilde{s}}$ be the output of $\mathsf{RecoverCandidates}^S(\mathbf{a}, \tilde{s})$.
  3. For $z \in Z$, define $\mathcal{A}_z$ by:
       • INPUT: $\mathbf{x} \in \mathbb{F}_q^m$
     (a) Pick a randomness string $R$ for the oracle machine $M^S[\mathbf{a}, z]$.
     (b) Let $M_R$ denote the oracle machine $M^S[\mathbf{a}, z]$ with the randomness string fixed to $R$.
     (c) Return $\mathsf{LocalCorrect}^{M_R}(\mathbf{x})$, where $\mathsf{LocalCorrect}^{M_R}$ denotes the algorithm $\mathsf{LocalCorrect}$
         with oracle access to the output of the oracle machine $M_R$.
  4. Return $\mathcal{L} = \{\mathcal{A}_z : z \in Z\}$.
```

The following lemma shows that this algorithm works, which implies in turn our main Lemma 4.2.

**Lemma 4.13.** *Let $q$ be a prime power, let $s, d, m$ be nonnegative integers so that $d + 6s \le sq$, and let $\delta := 1 - \frac{d}{sq}$. Let $\alpha \in (0, \delta)$ and $\ell \in \mathbb{N}$, and suppose that the univariate multiplicity code $\mathsf{MULT}_{q,s}^{(1)}(d)$ is $(\alpha, \ell, L)$-(globally) list recoverable. Let $\varepsilon > 0$ be a parameter, and suppose that $q \ge \frac{30{,}000 \cdot s \cdot m^2 \cdot L}{\alpha \varepsilon^2}$.*

*Let $S : \mathbb{F}_q^m \to \binom{\Sigma_{m,s}}{\ell}$, and suppose that $Q(X_1, \ldots, X_m) \in \mathbb{F}_q[X_1, \ldots, X_m]$ is a polynomial of degree at most $d$ such that:*

$$\Pr_{\mathbf{x} \in \mathbb{F}_q^m}\left[Q^{(<s)}(\mathbf{x}) \in S(\mathbf{x})\right] > 1 - \alpha + \varepsilon.$$

*Then with probability at least $2/3$ over the choice of a uniform random $\mathbf{a} \in \mathbb{F}_q^m$ and the randomness of the* $\mathsf{RecoverCandidates}$ *procedure, there exists an oracle machine $\mathcal{A}_z \in \mathcal{L}$ so that for all $\mathbf{x} \in \mathbb{F}_q^m$,*

$$\Pr\left[\mathcal{A}_z(\mathbf{x}) = Q^{(<s)}(\mathbf{x})\right] \ge \frac{2}{3}, \tag{15}$$

*where the probability is over the randomness of $\mathcal{A}_z$.*

[9]This claim about the running time in [KSY14] was only proved for fields of small characteristic. There, in the discussion about "Solving the Noisy System" in Section 4.3, it was shown that the running time can be made $\mathsf{poly}(O(s)^m \cdot q)$ provided one could efficiently decode Reed-Muller codes over certain product sets in $\mathbb{F}_q$, and remarked that this was known over fields of small characteristic. Recently [KK17] showed that this Reed-Muller decoding problem could be solved over all fields. This justifies the running time claim over all fields.

*Moreover, the output list $\mathcal{L}$ has size $|\mathcal{L}| = O(m^2L)$; and* LocalListRecoverMULT *makes $t := q \cdot \left(\frac{s \cdot m \cdot L}{\alpha}\right)^{O(m)}$ queries to $S$, and each $\mathcal{A}_z$ makes $q^2 \cdot (s \cdot m \cdot L)^{O(m)}$ queries to $S$.*

*Moreover, if $\mathsf{MULT}^{(1)}_{q,s}(d)$ can be probabilistically list-recovered with success probability at least $1 - \frac{1}{q}$ in time $T$, then* LocalListRecoverMULT *runs in time $T \cdot \mathsf{poly}(t)$, and each $\mathcal{A}_z$ runs in time $T \cdot \mathsf{poly}(t)$.*

The proof of the above lemma relies on the following two claims. The first claim shows that with high probability, there exists a "good" $z \in Z$ satisfying a certain desired property.

**Claim 4.14.** *With probability at least $2/3$ over the choice of a uniform random $\mathbf{a} \in \mathbb{F}_q^m$ and the randomness of the* RecoverCandidates *procedure, there exists $z \in Z$ so that*

$$\Pr_{\mathbf{x},R}\left[M^S\left[\mathbf{a}, z\right](\mathbf{x}) = Q^{(<s)}(\mathbf{x})\right] \geq 1 - 10\gamma, \tag{16}$$

*where the probability is over the choice of a uniform random $\mathbf{x} \in \mathbb{F}_q^m$ and the randomness $R$ of the oracle machine $M^S\left[\mathbf{a}, z\right]$, and*

$$\gamma = \frac{6}{\varepsilon^2 q} + \frac{8 \cdot s \cdot m^2 \cdot L}{q} + \frac{sL}{\tilde{s}}. \tag{17}$$

*Proof.* By the properties of the RecoverCandidates procedure (Lemma 4.5), we have that

$$\Pr\left[Q^{(<\tilde{s})}(\mathbf{a}) \in Z\right] \geq 1 - \frac{4}{\varepsilon^2 q},$$

where the probability is over the choice of a uniform random $\mathbf{a} \in \mathbb{F}_q^m$ and the randomness of the RecoverCandidates procedure. Recalling our assumption that $q \geq 40/\varepsilon^2$, this implies in turn that

$$Q^{(<\tilde{s})}(\mathbf{a}) \in Z \tag{18}$$

with probability at least 0.9 over the choice of a uniform random $\mathbf{a} \in \mathbb{F}_q^m$ and the randomness of the RecoverCandidates procedure.

By the properties of the oracle machine $M^s[\mathbf{a}, z]$ (Lemma 4.9), we have that

$$\Pr_{\mathbf{a},\mathbf{x},R}\left[M^S\left[\mathbf{a}, Q^{(<\tilde{s})}(\mathbf{a})\right](\mathbf{x}) = Q^{(<s)}(\mathbf{x})\right] \geq 1 - \gamma,$$

where the probability is over the choice of uniform random $\mathbf{a}, \mathbf{x} \in \mathbb{F}_q^m$ and the randomness $R$ of the oracle machine $M^S[\mathbf{a}, z]$, and $\gamma$ is as given in (17). By Markov's inequality, this implies in turn that with probability at least 0.9 over the choice of a uniform random $\mathbf{a} \in \mathbb{F}_q^m$, it holds that

$$\Pr_{\mathbf{x},R}\left[M^S\left[\mathbf{a}, Q^{(<\tilde{s})}(\mathbf{a})\right](\mathbf{x}) = Q^{(<s)}(\mathbf{x})\right] \geq 1 - 10\gamma. \tag{19}$$

By a union bound, we have that both (18) and (19) hold with probability at least 2/3 over the choice of a uniform random $\mathbf{a} \in \mathbb{F}_q^m$ and the randomness of the RecoverCandidates procedure. So with probability at least 2/3 over the choice of a uniform random $\mathbf{a} \in \mathbb{F}_q^m$ and the randomness of the RecoverCandidates procedure, there exists $z = Q^{(<\tilde{s})}(\mathbf{a}) \in Z$ so that (16) holds, as desired. □

The next claim shows that a “good” $z$ satisfying (16) leads to the desired conclusion (15).

**Claim 4.15.** *Assume that $z$ is such that (16) holds. Then for all $\mathbf{x} \in \mathbb{F}_q^m$,*

$$\Pr\left[\mathcal{A}_z(\mathbf{x}) = Q^{(<s)}(\mathbf{x})\right] \geq \frac{2}{3}, \tag{20}$$

*where the probability is over the choice of $R$ and the randomness of the algorithm* $\mathsf{LocalCorrect}$*.*

*Proof.* By our assumption (16) and Markov's inequality, with probability at least 0.9 over the choice of $R$, it holds that

$$\Pr_{\mathbf{x}}\left[M_R(\mathbf{x}) = Q^{(<s)}(\mathbf{x})\right] \geq 1 - 100\gamma, \tag{21}$$

where $M_R$ denotes the oracle machine $M^S[\mathbf{a}, z]$ with the randomness string fixed to $R$. Recalling the definition of $\gamma$ in (17), and our choice of $\tilde{s} = \frac{3000sL}{\alpha}$ and $q \geq \frac{30{,}000 \cdot s \cdot m^2 \cdot L}{\alpha \cdot \varepsilon^2}$, we can ensure that the probability in (21) is at least $1 - \frac{\alpha}{10}$.

Assume that $R$ is such that (21) holds. Then $M_R$ computes correctly $Q^{(<s)}(\mathbf{x})$ for all but an $\frac{\alpha}{10}$-fraction of the points $\mathbf{x} \in \mathbb{F}_q^m$. Recalling our assumption that $\alpha \in (0, \delta)$, Theorem 4.12 implies that for all $\mathbf{x} \in \mathbb{F}_q^m$, $\mathsf{LocalCorrect}^{M_R}$ computes $Q^{(<s)}(\mathbf{x})$ correctly with probability at least 0.9.

Consequently, for all $\mathbf{x} \in \mathbb{F}_q^m$, with probability at least $(0.9)^2 \geq 2/3$ over the choice of $R$ and the randomness of the algorithm $\mathsf{LocalCorrect}$, $\mathcal{A}_z(\mathbf{x}) = Q^{(<s)}(\mathbf{x})$, as desired. □

We now prove Lemma 4.13, based on the above Claims 4.14 and 4.15.

*Proof of Lemma 4.13.* By Claim 4.14, with probability at least 2/3 over the choice of a uniform random $\mathbf{a} \in \mathbb{F}_q^m$ and the randomness of the $\mathsf{RecoverCandidates}$ procedure, there exists $z \in Z$ which satisfies (16). By Claim 4.15, the resulting oracle machine $\mathcal{A}_z \in \mathcal{L}$ satisfies the desired conclusion (15).

It remains to show the claimed output list size, query complexity, and running time. The output list size is clearly $O(m^2L)$, because this is the list size returned by $\mathsf{RecoverCandidates}$. For the query complexity, the algorithm $\mathsf{LocalListRecoverMULT}$ has the same query complexity of $q \cdot (\tilde{s} \cdot m \cdot L)^{O(m)}$ as $\mathsf{RecoverCandidates}$, which is at most $t := q \cdot \left(\frac{s \cdot m \cdot L}{\alpha}\right)^{O(m)}$ by our choice of $\tilde{s} = \frac{3000sL}{\alpha}$, while each $\mathcal{A}_z$ has query complexity which is the product of the query complexity of $q \cdot (s \cdot m \cdot L)^{O(m)}$ of the oracle machines $M^S[\mathbf{a}, z]$ and the query complexity of $(O(s)^m \cdot q)$ of $\mathsf{LocalCorrect}$, and together these give the claimed values. The runtime calculation is similar. □

## 5 Capacity-achieving locally list decodable codes

In this section we use the results from the previous section to construct capacity-achieving locally list decodable (in fact, recoverable) codes with low (sub-polynomial) query complexity.

**Theorem 5.1** (Capacity-achieving locally list-recoverable codes)**.** *For any $R \in (0, 1)$, $\gamma > 0$, and $\ell \in \mathbb{N}$, there exists an infinite family $\{C_N\}_N$ of codes that satisfy the following.*

1. *$C_N$ is a code of block length $N$ and rate at least $R$.*
2. *$C_N$ is $(t, 1-R-\gamma, \ell, L)$-locally list recoverable for*
$$t = \exp_{R,\gamma,\ell}\left((\log N)^{5/6} \cdot (\log\log N)^{1/6}\right) \qquad \text{and} \qquad L = \exp_{R,\gamma,\ell}\left((\log N)^{2/3} \cdot (\log\log N)^{1/3}\right).$$
3. *The alphabet size of $C_N$ is $O_{R,\gamma,\ell}(1)$.*

*Moreover, $C_N$ can be encoded in time $\mathsf{poly}(N)$, and the local list recovery algorithm for $C_N$ runs in time $\mathsf{poly}(t)$.*

To prove the above theorem, we first use Theorem 4.1 from the previous section to show that *high-rate* multivariate multiplicity codes are locally list recoverable with low query complexity from a small (sub-constant) fraction of errors.

**Lemma 5.2** (High-rate locally list-recoverable codes)**.** *For any $\gamma > 0$ and $\ell \in \mathbb{N}$, there exists an infinite family $\{C_N\}_N$ of codes that satisfy the following.*

1. *$C_N$ is a code of block length $N$ and rate at least $1-\gamma$.*
2. *$C_N$ is $(t, \alpha, \ell, L)$-locally list recoverable for*
$$t = \exp_{\gamma,\ell}\left((\log N)^{5/6} \cdot (\log\log N)^{1/6}\right), \qquad \alpha = \Omega_{\gamma,\ell}\left(\left(\frac{\log\log N}{\log N}\right)^{1/6}\right),$$
$$\text{and} \qquad L = \exp_{\gamma,\ell}\left((\log N)^{2/3} \cdot (\log\log N)^{1/3}\right).$$
3. *The alphabet size of $C_N$ is $\exp_{\gamma,\ell}\exp\left((\log N)^{1/6} \cdot (\log\log N)^{5/6}\right)$.*

*Moreover, $C_N$ can be encoded in time $\mathsf{poly}(N)$, and the local list recovery algorithm for $C_N$ runs in time $\mathsf{poly}(t)$.*

Theorem 5.1 follows from the above lemma by applying the AEL distance amplification procedure [AEL95, AL96] to transform these codes into capacity-achieving locally list recoverable codes, while roughly preserving the query complexity and output list size. As this part is fairly standard by now, and has been applied in a similar way in a sequence of recent papers [KMRS17, GKO$^+$18, HRW20, KRR$^+$21], we defer the proof of Theorem 5.1 to Appendix C. The rest of this section is devoted to the proof of Lemma 5.2.

*Proof of Lemma 5.2.* Fix $\gamma > 0$ and $\ell \in \mathbb{N}$, we shall let $C_N := \mathsf{MULT}^{(m)}_{q,s}(d)$, as per the following choice of parameters.

Let $m$ be a parameter to be determined later on, and let
$$\delta := \frac{\gamma}{2m} \quad \text{and} \quad \varepsilon := \frac{\delta}{2} = \frac{\gamma}{4m}. \tag{22}$$

Let

$$s := \frac{64\ell}{\varepsilon^2} = \frac{1024\ell}{\gamma^2} \cdot m^2 = \Theta_{\gamma,\ell}(m^2), \tag{23}$$

$$L := \left(\frac{s\ell}{\varepsilon}\right)^{c_0 \cdot \frac{s\ell}{\varepsilon^2} \cdot \log\left(\frac{\ell}{1-\delta}\right)} = \exp_{\gamma,\ell}\left(m^4 \log m\right), \tag{24}$$

and

$$q := \max\left\{L^m, c_1 \cdot \frac{s \cdot m^2 \cdot L}{(\delta - \varepsilon)\cdot \varepsilon^2}\right\} = \exp_{\gamma,\ell}(m^5 \log m), \tag{25}$$

where $c_0, c_1$ are the constants guaranteed by Theorem 4.1. Jumping ahead, the reason for the above choice of $q$ is so that the query complexity expression $t \geq L^{O(m)}$ from Theorem 4.1 is substantially smaller than the codeword length $q^m$.

Let $d := (1-\delta)sq$, and note that as per Claim 2.7, the rate of $C_N$ is at least

$$\begin{aligned} R &\geq \left(1 - \frac{m^2}{s}\right)(1-\delta)^m \\ &= \left(1 - \frac{\gamma^2}{1024\ell}\right)\left(1 - \frac{\gamma}{2m}\right)^m \\ &\geq \left(1 - \frac{\gamma^2}{1024\ell}\right)\left(1 - \frac{\gamma}{2}\right) \\ &\geq 1 - \gamma, \end{aligned}$$

where the first equality follows by choices of $s$ and $\delta$ in (23) and (22), respectively, and the last two inequalities hold for sufficiently small $\gamma$.

Finally, to guarantee block length $N$ we must have that $N = q^m = \exp_{\gamma,\ell}(m^6 \log m)$, which implies in turn that

$$m = \Theta_{\gamma,\ell}\left(\left(\frac{\log N}{\log\log N}\right)^{1/6}\right). \tag{26}$$

Next observe that our choice of $s$, $L$, and $q$ in (23), (24), and (25) guarantess that all conditions of Theorem 4.1 are satisfied. Consequently, we have that $C_N$ is locally list recoverable with query complexity

$$t = q^2 \cdot \left(\frac{s \cdot m \cdot L}{\delta - \varepsilon}\right)^{O(m)} = \exp_{\gamma,\ell}(m^5 \log m) = \exp_{\gamma,\ell}\left((\log N)^{5/6} \cdot (\log\log N)^{1/6}\right),$$

decoding radius

$$\alpha = \frac{\delta}{2} = \frac{\gamma}{4m} = \Omega_{\gamma,\ell}\left(\left(\frac{\log\log N}{\log N}\right)^{1/6}\right),$$

and output list size

$$O(m^2 L) = \exp_{\gamma,\ell}(m^4 \log m) = \exp_{\gamma,\ell}\left((\log N)^{2/3} \cdot (\log\log N)^{1/3}\right),$$

where the equalities follow by our choice of $\delta, \varepsilon, s, L, q$, and $m$ in (22)–(26).

Finally, note that the alphabet size is

$$q^{\binom{s+m-1}{m}} = \exp_{\gamma,\ell}\left(m^{O(m)}\right) = \exp_{\gamma,\ell} \exp\left((\log N)^{1/6} \cdot (\log\log N)^{5/6}\right),$$

where the equalities follow by our choice of $q, s$, and $m$ in (25), (23), and (26), respectively. It can also be observed that the encoding and local list-recovery times are as claimed. $\square$

## Acknowledgements

We would like to thank Atri Rudra and Venkatesan Guruswami for helpful discussions.

## References

[AEL95] Noga Alon, Jeff Edmonds, and Michael Luby. Linear time erasure codes with nearly optimal recovery. In *proceedings of the 36th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 512–519. IEEE Computer Society, 1995.

[AL96] Noga Alon and Michael Luby. A linear time erasure-resilient code with nearly optimal recovery. *IEEE Transactions on Information Theory*, 42(6):1732–1736, 1996.

[AS03] Sanjeev Arora and Madhu Sudan. Improved low-degree testing and its applications. *Combinatorica*, 23(3):365–426, 2003.

[BFLS91] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 21–31. ACM Press, 1991.

[BHKS21] Siddharth Bhandari, Prahladh Harsha, Mrinal Kumar, and Madhu Sudan. Decoding multivariate multiplicity codes on product sets. In *Proceedings of the 53rd Symposium on Theory of Computing Conference (STOC)*, pages 1489–1501. ACM Press, 2021.

[BK09] Kristian Brander and Swastik Koparty. List-decoding reed-muller over large fields upto the johnson radius. *Manuscript*, 2009.

[BL18] Abhishek Bhowmick and Shachar Lovett. The list decoding radius for reed-muller codes over small fields. *IEEE Transactions on Information Theory*, 64(6):4382–4391, 2018.

[DKSS13] Zeev Dvir, Swastik Kopparty, Shubhangi Saraf, and Madhu Sudan. Extensions to the method of multiplicities, with applications to kakeya sets and mergers. *SIAM Journal on Computing*, 42(6):2305–2328, 2013.

[DL12] Zeev Dvir and Shachar Lovett. Subspace evasive sets. In *Proceedings of the 44th Symposium on Theory of Computing Conference (STOC)*, pages 351–358. ACM Press, 2012.

[Eli91] Peter Elias. Error-correcting codes for list decoding. *IEEE Transactions on Information Theory*, 37(1):5–12, 1991.

[GI01] Venkatesan Guruswami and Piotr Indyk. Expander-based constructions of efficiently decodable codes. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 658–667. IEEE Computer Society, 2001.

[GI04] Venkatesan Guruswami and Piotr Indyk. Linear-time list decoding in error-free settings. In *proceedings of the 31st International Colloquium on Automata, Languages and Programming (ICALP)*, volume 3142 of *Lecture Notes in Computer Science*, pages 695–707. Springer, 2004.

[GK16a] Alan Guo and Swastik Kopparty. List-decoding algorithms for lifted codes. *IEEE Transactions on Information Theory*, 62(5):2719–2725, 2016.

[GK16b] Venkatesan Guruswami and Swastik Kopparty. Explicit subspace designs. *Combinatorica*, 36(2):161–185, 2016.

[GKO$^+$18] Sivakanth Gopi, Swastik Kopparty, Rafael Oliveira, Noga Ron-Zewi, and Shubhangi Saraf. Locally testable and locally correctable codes approaching the gilbert-varshamov bound. *IEEE Transactions on Information Theory*, 64(8):5813–5831, 2018.

[GKZ08] Parikshit Gopalan, Adam Klivans, and David Zuckerman. List-decoding reed-muller codes over small fields. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC)*, pages 265–274. ACM Press, 2008.

[GL89] Oded Goldreich and Leonid A Levin. A hard-core predicate for all one-way functions. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC)*, pages 25–32. ACM Press, 1989.

[Gop13] Parikshit Gopalan. A fourier-analytic approach to reed-muller decoding. *IEEE Transactions on Information Theory*, 59(11):7747–7760, 2013.

[GR08] Venkatesan Guruswami and Atri Rudra. Explicit codes achieving list decoding capacity: Error-correction with optimal redundancy. *IEEE Transactions on Information Theory*, 54(1):135–150, 2008.

[GR22] Zeyu Guo and Noga Ron-Zewi. Efficient list-decoding with constant alphabet and list sizes. *IEEE Transactions on Information Theory*, 68(3):1663–1682, 2022.

[GS99] Venkatesan Guruswami and Madhu Sudan. Improved decoding of reed-solomon and algebraic-geometry codes. *IEEE Transactions on Information Theory*, 45(6):1757–1767, 1999.

[GW13] Venkatesan Guruswami and Carol Wang. Linear-algebraic list decoding for variants of reed-solomon codes. *IEEE Transactions on Information Theory*, 59(6):3257–3268, 2013.

[HRW20] Brett Hemenway, Noga Ron-Zewi, and Mary Wootters. Local list recovery of high-rate tensor codes and applications. *SIAM Journal on Computing*, 49(4):157–195, 2020.

[KK17] John Kim and Swastik Kopparty. Decoding reed–muller codes over product sets. *Theory of Computing*, 13(21):1–38, 2017.

[KM93] Eyal Kushilevitz and Yishay Mansour. Learning decision trees using the fourier spectrum. *SIAM Journal on Computing*, 22(6):1331–1348, 1993.

[KMRS17] Swastik Kopparty, Or Meir, Noga Ron-Zewi, and Shubhangi Saraf. High-rate locally correctable and locally testable codes with sub-polynomial query complexity. *Journal of ACM*, 64(2):11:1–11:42, 2017.

[Kop15] Swastik Kopparty. List-decoding multiplicity codes. *Theory of Computing*, 11(5):149–182, 2015.

[Kra03] Victor Krachkovsky. Reed-solomon codes for correcting phased error bursts. *IEEE Transactions on Information Theory*, 49(11):2975–2984, 2003.

[KRR$^+$21] Swastik Kopparty, Nicolas Resch, Noga Ron-Zewi, Shubhangi Saraf, and Shashwat Silas. On list recovery of high-rate tensor codes. *IEEE Transactions on Information Theory*, 67(1):296–316, 2021.

[KRSW18] Swastik Kopparty, Noga Ron-Zewi, Shubhangi Saraf, and Mary Wootters. Improved decoding of folded reed-solomon and multiplicity codes. *Electronic Colloquium on Computational Complexity (ECCC)*, TR18-091, 2018.

[KST22] Dan Karliner, Roie Salama, and Amnon Ta-Shma. The plane test is a local tester for multiplicity codes. In *Proceedings of the 37th Computational Complexity Conference (CCC)*, volume 14, pages 1–33, 2022.

[KSY14] Swastik Kopparty, Shubhangi Saraf, and Sergey Yekhanin. High-rate codes with sublinear-time decoding. *Journal of ACM*, 61(5):28, 2014.

[KT00] Jonathan Katz and Luca Trevisan. On the efficiency of local decoding procedures for error-correcting codes. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 80–86. ACM Press, 2000.

[Lip90] Richard Lipton. Efficient checking of computations. In *Proceedings of the 7th Annual ACM Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 207–215. Springer, 1990.

[Nie01] Rasmus Nielsen. *List decoding of linear block codes*. PhD thesis, Technical University of Denmark, 2001.

[PV05] Farzad Parvaresh and Alexander Vardy. Correcting errors beyond the Guruswami–Sudan radius in polynomial time. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 285–294. IEEE Computer Society, 2005.

[RS96] Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):252–271, 1996.

[RT97] Michael Rosenbloom and Michael Tsfasman. Codes for the m-metric. *Problemy Peredachi Informatsii*, 33(1):55–63, 1997.

[STV01] Madhu Sudan, Luca Trevisan, and Salil P. Vadhan. Pseudorandom generators without the xor lemma. *Journal of Computer and System Sciences*, 62(2):236–266, 2001.

[Sud97] Madhu Sudan. Decoding of reed solomon codes beyond the error-correction bound. *Journal of Complexity*, 13(1):180–193, 1997.

[SY11] Shubhangi Saraf and Sergey Yekhanin. Noisy interpolation of sparse polynomials, and applications. In *Proceedings of the IEEE 26th Annual Conference on Computational Complexity (CCC)*, pages 86–92. IEEE Computer Society, 2011.

## A Constant-dimensional output list for univariate multiplicity codes

In this section we prove the following lemma which implies Theorem 3.7.

**Lemma A.1.** *Let $q$ be a prime power, and let $s, d, n$ be nonnegative integers such that $n \leq q$. Let $r$ be a nonnegative integer such that $r \leq \min\{s, \mathsf{char}(\mathbb{F}_q)\}$. Then the univariate multiplicity code $\mathsf{MULT}^{(1)}_{q,s}(n,d)$ is $(\alpha, \ell, L)$-list recoverable for any $\alpha \in (0,1)$ satisfying that*

$$\alpha < 1 - \frac{\ell}{r+1} - \frac{r}{r+1} \cdot \frac{s}{s-r+1} \cdot \frac{d}{sq},$$

*where the output list is contained in an $\mathbb{F}_q$-affine subspace $v_0 + V$ of dimension at most $r \cdot \left(1 + \frac{d}{\mathsf{char}(\mathbb{F}_q)}\right)$. Moreover, there is a (deterministic) algorithm that outputs a basis for $V$ in time $\mathsf{poly}(\log q, s, d, n, L)$.*

Theorem 3.7 follows by setting $s \geq \frac{16\ell}{\varepsilon^2}$ and $r = \frac{4\ell}{\varepsilon}$, and noting that in this setting of parameters we have that $\alpha \geq \delta - \varepsilon$.

The proof of Lemma A.1 above adapts Theorem 17 of [GW13] to our setting. We refer the reader to [GW13] for more context and intuition.

We begin by giving the algorithm.

**Algorithm** FindSubspace.

- **INPUT:** Access to input lists $S \subseteq \binom{\mathbb{F}_q^s}{\ell}^n$
- **OUTPUT:** (A basis for) an $\mathbb{F}_q$-affine subspace $v_0 + V$ containing all codewords $c \in \mathsf{MULT}^{(1)}_{q,s}(n,d)$ with $\mathrm{dist}(c, S) \leq \alpha$.

1. Set $D = (s - r + 1)(1 - \alpha)q - 1$.
2. By solving a linear system of equations over $\mathbb{F}_q$, find polynomials

$$A(X), B_0(X), \ldots, B_{r-1}(X) \in \mathbb{F}_q[X],$$

not all zero, such that:

(a) $\deg(A) \leq D$, and for all $i = 0, \ldots, r-1$, $\deg(B_i) \leq D - d$.

(b) For each $\lambda$ with $0 \leq \lambda \leq s-r$, for each evaluation point $a_i \in \mathbb{F}_q$, and for each $\beta = (\beta_0, \ldots, \beta_{s-1}) \in S_i$:

$$A^{(\lambda)}(a_i) + \sum_{i=0}^{r-1} \sum_{j=0}^{\lambda} \binom{i+j}{i} \beta_{i+j} B_i^{(\lambda-j)}(a_i) = 0.$$

3. Let $v_0 + V$ be the affine space containing the encoding of all polynomials $f(X)$ satisfying

$$A(X) + \sum_{i=0}^{r-1} f^{(i)}(X) B_i(X) = 0.$$

We need to show:

1. The linear system has a nonzero solution,
2. Any codeword $c \in \mathsf{MULT}_{q,s}^{(1)}(n,d)$ with $\mathrm{dist}(c, S) \leq \alpha$ is contained in $v_0 + V$.

We show these two items below.

**Item (1):** To see that the linear system has a nonzero solution, we show that the homogeneous system of linear equations in Step 2 of the algorithm has more variables than constraints. The total number of free coefficients in $A(X), B_0(X), \ldots, B_{r-1}(X)$ equals:

$$\begin{aligned}
(D+1) + r(D-d+1) &= (D+1)(r+1) - d \cdot r \\
&= (s-r+1)(1-\alpha)q(r+1) - d \cdot r \\
&> (s-r+1)\left(\frac{\ell}{r+1} + \frac{r}{r+1} \cdot \frac{s}{s-r+1} \cdot \frac{d}{sq}\right) q(r+1) - d \cdot r \\
&= (s-r+1)\ell q + d \cdot r - d \cdot r \\
&= (s-r+1)\ell q.
\end{aligned}$$

The total number of constraints equals:

$$q \cdot (s-r+1) \cdot \ell.$$

Thus the number of free coefficients is larger than the number of constraints, and this proves that the algorithm can find a nonzero solution in Step 2.

**Item (2):** Let $c$ be a codeword with $\mathrm{dist}(c, S) \leq \alpha$ that is the encoding of a polynomial $g(X)$. We will show that $c \in v_0 + V$. Define $Q(X) = A(X) + \sum_{i=0}^{r-1} g^{(i)}(X) B_i(X)$. Observe that $\deg(Q) \leq D$.

Now take any evaluation point $a_i \in \mathbb{F}_q$ and $\beta = (\beta_0, \ldots, \beta_{s-1}) \in S_i$ for which

$$g^{(<s)}(a_i) = \beta. \tag{27}$$

Let $t$ be an integer with $0 \le t \le s-r$. Then by the chain rule for Hasse derivatives:

$$\begin{aligned}
Q^{(t)}(a_i) &= A^{(t)}(a_i) + \sum_{i=0}^{r-1} \left(g^{(i)} \cdot B_i\right)^{(t)}(a_i) \\
&= A^{(t)}(a_i) + \sum_{i=0}^{r-1}\sum_{j=0}^{t} \left(g^{(i)}\right)^{(j)}(a_i) B_i^{(t-j)}(a_i) \\
&= A^{(t)}(a_i) + \sum_{i=0}^{r-1}\sum_{j=0}^{t} \binom{i+j}{i} g^{(i+j)}(a_i) B_i^{(t-j)}(a_i) \\
&= A^{(t)}(a_i) + \sum_{i=0}^{r-1}\sum_{j=0}^{t} \binom{i+j}{i} \beta_{i+j} B_i^{(t-j)}(a_i) \\
&= 0,
\end{aligned}$$

where the last equality is due to the requirement on Step 2b of Algorithm FindSubspace. Since this holds for every $t$ with $0 \le t \le s-r$, we get that:

$$\mathsf{mult}(Q, a_i) \ge s - r + 1.$$

By assumption on $g$, there are at least $(1-\alpha)q$ evaluation points $a_i \in \mathbb{F}_q$ such that there exists some $\beta \in S_i$ for which Equation (27) holds. Thus there are at least $(1-\alpha)q$ points where $Q$ vanishes with multiplicity at least $s-r+1$. Since $\deg(Q) \le D < (s-r+1)(1-\alpha)q$, we conclude that $Q(X) = 0$.

By definition of $Q(X)$ and $v_0 + V$, this implies that $c \in v_0 + V$, as desired.

Finally, we turn to bound the dimension of $V$.

**Dimension of affine subspace:** Let $f = \sum_{i=0}^{d} f_i X^i$ be a polynomial satisfying that $Q(X) := A(X) + \sum_{i=0}^{r-1} f^{(i)}(X) B_i(X) = 0$. Let $0 \le \hat{r} \le r-1$ be maximal such that $B_{\hat{r}}(X) \neq 0$, and note that such an $\hat{r}$ exists by the requirement that $A(X), B_0(X), \ldots, B_{r-1}(X)$ are not all zero, and by assumption that $Q(X) = A(X) + \sum_{i=0}^{r-1} f^{(i)}(X) B_i(X) = 0$. Let $a \in \mathbb{F}_q$ be such that $B_{\hat{r}}(a) \neq 0$, and note that such an $a$ exists by assumption that $\deg(B_{\hat{r}}) < D < q$. Let $\hat{f}(X) = \sum_{i=0}^{d} \hat{f}_i X^i := f(X+a)$, $\hat{A}(X) := A(X+a)$, and $\hat{B}_i(X) := B_i(X+a)$ for each $0 \le i \le \hat{r}$. Let

$$\hat{Q}(X) := \hat{A}(X) + \sum_{i=0}^{\hat{r}} \hat{f}^{(i)}(X) \hat{B}_i(X),$$

and note that our assumptions imply that $\hat{Q}(X) = 0$ and $\hat{B}_{\hat{r}}(0) = B_{\hat{r}}(a) \neq 0$.

Then for each $i = \hat{r}, \ldots, d$, the coefficient of $X^{i-\hat{r}}$ in $\hat{Q}(X)$ is $\binom{i}{\hat{r}} \cdot \hat{B}_{\hat{r}}(0) \cdot \hat{f}_i + \ell_i$, where $\ell_i$ is an affine linear combination of $\hat{f}_1, \ldots, \hat{f}_{i-1}$. Thus, whenever $\binom{i}{\hat{r}} = \frac{i\cdot(i-1)\cdots(i-\hat{r}+1)}{(\hat{r})!} \neq 0$ we get that the coefficient $\hat{f}_i$ is determined by prior coefficients. By assumptions that $\hat{r} < r \le \mathsf{char}(\mathbb{F}_q)$ we get that $(\hat{r})! \neq 0$. Moreover, $i \cdot (i-1) \cdots (i-\hat{r}+1)$ can be non-zero for at most $\hat{r} \cdot \frac{d+1}{\mathsf{char}(\mathbb{F}_q)}$ values of $i$. We conclude that the number of undetermined coefficients is at most $\hat{r} + \hat{r} \cdot \frac{d+1}{\mathsf{char}(\mathbb{F}_q)} \leq r \cdot \left(1 + \frac{d}{\mathsf{char}(\mathbb{F}_q)}\right)$, and this also gives a bound on the dimension of $V$.

## B List-recovering tuples of polynomials on a grid

In this section, we prove Lemma 4.6 about list-recovery of tuples of polynomials on a grid.

To this end, we first note that the case of tuples of polynomials can be reduced to the case of a single polynomial over a large field. Specifically, let $\mathbb{K}$ be the degree $t$ field extension of $\mathbb{F} := \mathbb{F}_q$, and let $\phi : \mathbb{F}^t \to \mathbb{K}$ be an arbitrary $\mathbb{F}$-linear bijection. Then to every function $f : U^m \to \mathbb{F}^t$, we can associate a function $\tilde{f} : U^m \to \mathbb{K}$, where $\tilde{f} = \phi \circ f$. This identifies the underlying Hamming metric spaces. The key observation is that under this identification, $f : U^m \to \mathbb{F}^t$ is the evaluation table of a tuple of $t$ polynomials in $\mathbb{F}[Y_1, \ldots, Y_m]$ of degree $\leq d$ if and only if $\tilde{f} : U^m \to \mathbb{K}$ is the evaluation table of a degree $\leq d$ polynomial in $\mathbb{K}[Y_1, \ldots, Y_m]$.

Through this connection, Lemma 4.6 is a consequence of the following lemma (and the fact that $\mathbb{K}$ can be constructed in randomized $\mathsf{poly}(\log |\mathbb{K}|) = \mathsf{poly}(t, \log q)$ time).

**Lemma B.1** (List-recovering polynomials on a grid)**.** *Let $q$ be a prime power, let $d, m, \ell$ be non-negative integers, and let $U$ be an arbitrary subset of $\mathbb{F}_q$ of size $r$.*

*Let $S : U^m \to \binom{\mathbb{F}_q}{\ell}$, and let $\mathcal{L}$ be the list of all polynomials $Q(Y_1, \ldots, Y_m)$ of degree at most $d$, so that*

$$\Pr_{\mathbf{u} \in U^m} [Q(\mathbf{u}) \in S(\mathbf{u})] > \sqrt{\frac{2d\ell}{r}}.$$

*Then $|\mathcal{L}| \leq \frac{r}{d}$.*

*Moreover, if $\mathcal{L}$ is the list of all polynomials $Q(Y_1, \ldots, Y_m)$ of degree at most $d$, so that*

$$\Pr_{\mathbf{u} \in U^m} [Q(\mathbf{u}) \in S(\mathbf{u})] > m \cdot \sqrt{\frac{2d\ell}{r}},$$

*then there is a $\mathsf{poly}(\log q, (d \cdot m \cdot r)^m, \ell)$-time algorithm which computes $\mathcal{L}$.*

*Proof.* The bound on $|\mathcal{L}|$ follows from the Johnson bound for list recovery, which is a general statement implying good list recoverability for codes with large distance. Specifically, Lemma 21 in [GKO+18] states that a code of relative distance $\delta$ is $(\alpha, \ell, L)$-list recoverable for $\alpha < 1 - \sqrt{\ell(1-\delta)}$ and $L = \frac{\ell}{(1-\alpha)^2 - \ell(1-\delta)}$. In our setting, the code of polynomials of degree at most $d$ on $U^m$ has relative distance $\delta$ at least $1 - \frac{d}{r}$. Thus for $\alpha = 1 - \sqrt{\frac{2d\ell}{r}}$, the Johnson bound for list recovery implies that we can take $L = \frac{r}{d}$, as desired.

We prove the 'moreover' part by induction on $m$. The $m = 1$ case is simply the Sudan list-recovery algorithm [Sud97] for Reed-Solomon codes, which works with the claimed parameters (since the total number of points is $n := \ell \cdot r$, and the number of agreement points is at least $\sqrt{\frac{2d\ell}{r}} \cdot r = \sqrt{2dn}$, which is the requirement for the Sudan algorithm to work). Moreover, this algorithm has running time $\mathsf{poly}(\log q, d, r, \ell)$.

For general $m$, we first do list-recovery on $(m-1)$-dimensional grids, and then combine the results using list-recovery for vector-valued univariate polynomials. We crucially use the previous combinatorial bound on the output list size to ensure that the intermediate output list size is under control (as the recursion unfolds).

More concretely, the algorithm proceeds as follows:

1. First, for each setting of $u \in U$, we consider the collection of input lists $S_u : U^{m-1} \to \binom{\mathbb{F}_q}{\ell}$, given by $S_u(\mathbf{y}) = S(\mathbf{y}, u)$.

   Now recursively list-recover $S_u$ from a $(m-1) \cdot \sqrt{\frac{2d\ell}{r}}$ fraction of agreement to find the set of nearby $(m-1)$-variate polynomials $\mathcal{L}_u$. By the previous combinatorial bound, we can assume (after confirming that all elements of $\mathcal{L}_u$ are indeed close to $S_u$) that $|\mathcal{L}_u| \leq \frac{r}{d}$.

2. Next we combine all output lists $\mathcal{L}_u$ for $u \in U$. Let $M_1(Y_1, \ldots, Y_{m-1}), \ldots, M_t(Y_1, \ldots, Y_{m-1})$ be all the $(m-1)$-variate monomials of total degree at most $d$. Define a function $\tilde{S} : U \to \binom{\mathbb{F}_q^t}{r/d}$ as follows: for each $u \in U$ and each element $P$ of $\mathcal{L}_u$, include the vector of coefficients of $P$ into $\tilde{S}(u)$.

   Then, using a vector-valued Sudan list-recovery algorithm for univariate polynomials (obtained from the standard scalar-valued Sudan list-recovery algorithm via the connection described above), we find all tuples of univariate polynomials $(P_1(Z), \ldots, P_t(Z)) \in (\mathbb{F}_q[Z])^t$ such that:
$$\Pr_{u \in U}[(P_1(u), \ldots, P_t(u)) \in \tilde{S}(u)] > \sqrt{\frac{2d\ell}{r}}.$$

3. For each tuple of polynomials $(P_1(Z), \ldots, P_t(Z))$ found in the previous step, we construct the polynomial:
$$R(Y_1, \ldots, Y_{m-1}, Y_m) = \sum_{i=1}^{t} M_i(Y_1, \ldots, Y_{m-1}) P_i(Y_m).$$
   If this polynomial has total degree at most $d$, and it satisfies that
$$\Pr_{\mathbf{u} \in U^m}[R(\mathbf{u}) \in S(\mathbf{u})] > m \cdot \sqrt{\frac{2d\ell}{r}},$$
   then we include it in the output list.

To prove correctness of this algorithm, consider any $Q(Y_1, \ldots, Y_m) \in \mathcal{L}$. Let $Q_u(Y_1, \ldots, Y_{m-1}) = Q(Y_1, \ldots, Y_{m-1}, u)$. Then we have:
$$\Pr_{u \in U,\, \mathbf{y} \in U^{m-1}}[Q_u(\mathbf{y}) \in S_u(\mathbf{y})] > m \cdot \sqrt{\frac{2d\ell}{r}},$$
and so by Markov's inequality, with probability at least $\sqrt{\frac{2d\ell}{r}}$ over the choice of uniform random $u \in U$, we have that:
$$\Pr_{\mathbf{y} \in U^{m-1}}[Q_u(\mathbf{y}) \in S_u(\mathbf{y})] > (m-1) \cdot \sqrt{\frac{2d\ell}{r}}.$$

This implies that for at least a $\sqrt{\frac{2d\ell}{r}}$-fraction of $u \in U$, we have that $Q_u \in \mathcal{L}_u$.

Write $Q(Y_1, \ldots, Y_m)$ as $\sum_{i=1}^{t} M_i(Y_1, \ldots, Y_{m-1}) G_i(Y_m)$. Then the above discussion means that for at least a $\sqrt{\frac{2d\ell}{r}}$-fraction of $u \in U$, we have that $(G_1(u), \ldots, G_t(u)) \in \tilde{S}(u)$. This implies that $(G_1(Z), \ldots, G_t(Z))$ will be included in the list returned by the univariate list-recovery algorithm in Step 2, and thus that $Q$ will be included in the output of the algorithm in Step 3.

This completes the proof of correctness. The bound on the running time follows immediately from the description of the algorithm (using the fact that the number of $m$-variate degree $d$ monomials is at most $(d+m)^m$). $\square$

## C Capacity-achieving locally list decodable codes

In this section we prove Theorem 5.1, showing the existence of capacity-achieving locally list recoverable codes with low (sub-polynomial) query complexity. To this end, we apply the Alon-Edmonds-Luby (AEL) distance amplification method [AEL95, AL96] on the high-rate locally list recoverable codes given by Lemma 5.2.

More specifically, we use the following version of the AEL transformation for local list recovery from [GKO+18] which roughly says the following. Given an "outer" code $C$ of rate approaching 1 that is locally list recoverable from a tiny fraction of errors, and a short "inner" code $C'$ that is a capacity-achieving (globally) list recoverable code, they can be combined to get a new code $C_{\mathrm{AEL}}$ that on the one hand, inherits the tradeoff between rate and error correction that $C'$ enjoys, and on the other hand, inherits the locality of $C$.

**Lemma C.1** (Amplification for local list recovery, [GKO+18], Lemma 23)**.** *There exists an absolute constant $b_0$ so that the following holds for any $\alpha, \gamma > 0$ and $s \geq (\alpha \cdot \gamma)^{-b_0}$.*

*Suppose that $C \subseteq (\Sigma^{R' \cdot s})^n$ is a code of rate $R$ that is $(t, \alpha, \ell, L)$-locally list recoverable, and $C' \subseteq \Sigma^s$ is a code of rate $R'$ that is $(\alpha', \ell', \ell)$-globally list recoverable. Then there exists a code $C_{AEL} \subseteq (\Sigma^s)^n$ of rate $R \cdot R'$ that is $(t \cdot \mathsf{poly}(s), \alpha' - \gamma, \ell', L)$-locally list recoverable.*

*Moreover, if $C, C'$ can be encoded in times $T, T'$, respectively, then $C_{AEL}$ can be encoded in time $T + \mathsf{poly}(n, s) \cdot T'$. If the local list recovery algorithm for $C$ runs in time $T$, and $C'$ can be globally list recovered in time $T'$, then the local list recovery algorithm for $C_{AEL}$ runs in time $T + \mathsf{poly}(t, \log n, s) \cdot T'$.*

To reduce the alphabet size of our codes to a constant, we shall also use the following *concatenation* procedure for local list recovery, which reduces the alphabet size of a given outer code $C$ by encoding each alphabet symbol of $C$ with a short inner code $C'$ of smaller alphabet.

**Lemma C.2** (Concatenation for local list recovery, [KRR+21], Lemma 15.13)**.** *Suppose that $C \subseteq (\Sigma^{R' \cdot s})^n$ is a code of rate $R$ that is $(t, \alpha, \ell, L)$-locally list recoverable, and $C' \subseteq \Sigma^s$ is a code of rate $R'$ that is $(\alpha', \ell', \ell)$-globally list recoverable. Let $C_{concat} \subseteq \Sigma^{s \cdot n}$ be the code obtained by applying $C'$ on each coordinate of $C$. Then $C_{concat}$ is a code of rate $R \cdot R'$ that is $(t \cdot s, \alpha \cdot \alpha', \ell', L)$-locally list recoverable.*

*Moreover, if $C, C'$ can be encoded in times $T, T'$, respectively, then $C_{concat}$ can be encoded in time $T + O(n \cdot T')$. If the local list recovery algorithm for $C$ runs in time $T$, and $C'$ can be globally list recovered in time $T'$, then the local list recovery algorithm for $C_{concat}$ runs in time $T + O(t \cdot T')$.*

As the inner code in the above procedures, we shall use the following two families of capacity-achieving (globally) list-recoverable codes. The first family of codes are capacity-achieving (globally) list-recoverable codes over *constant-size alphabets*, albeit with *non-efficient* encoding and list-recovery algorithms (such codes can be obtained for example by random linear codes, see e.g., [HRW20, Corollary 2.2]).

**Fact C.3.** *For any $R \in (0,1)$, $\gamma > 0$, and $\ell \in \mathbb{N}$, and sufficiently large $q$ and $n$, there exists a code $C \subseteq \Sigma^n$ of rate $R$ and alphabet size $q$ that is $(1 - R - \gamma, \ell, O_{R,\gamma,\ell}(1))$-list recoverable. Moreover, $C$ can be encoded in time $\exp_{R,\gamma,\ell}(n^2)$, and list-recovered in time $\exp_{R,\gamma,\ell}(n)$.*

The second family of codes are *efficient* capacity-achieving (globally) list-recoverable codes, albeit over *polynomially-large alphabets* (such codes can be obtained for example as a consequence of our Lemma 3.9 and Theorem 3.11 about global list recovery of folded Reed-Solomon codes).

**Fact C.4.** *For any $R \in (0,1)$, $\gamma > 0$, and $\ell \in \mathbb{N}$, and a sufficiently large $n$, there exists a code $C \subseteq \Sigma^n$ of rate $R$ and alphabet size $\mathsf{poly}_{R,\gamma,\ell}(n)$ that is $(1 - R - \gamma, \ell, O_{R,\gamma,\ell}(1))$-list recoverable in time $\mathsf{poly}_{R,\gamma,\ell}(n)$. Moreover, $C$ can be encoded in time $\mathsf{poly}(n)$.*

We now turn to the proof of Theorem 5.1.

*Proof of Theorem 5.1.* We shall construct the required capacity-achieving locally list-recoverable code $C$ via the following steps:

1. In the first step, we apply the amplification procedure with the outer code being the high-rate locally list-recoverable code $C_1$ given by Lemma 5.2 , and the inner code being an *efficient* capacity-achieving globally list-recoverable code $C_1'$. This gives a capacity-achieving locally list-recoverable code $C_2$ with the desired properties, albeit with an *exponentially-large* alphabet size.

2. To reduce the alphabet size, we first concatenate the resulting code $C_2$ with an *efficient* capacity-achieving globally list-recoverable code $C_2'$, to exponentially reduce the alphabet size. This gives a high-rate locally list-recoverable code $C_3$ with a *sub-polynomial* alphabet size.

3. Once the alphabet size is sufficiently small, we can further reduce the alphabet size to a constant by concatenating the resulting code $C_3$ with a *non-efficient* capacity-achieving globally list-recoverable code $C_3'$ over a constant-size alphabet. This gives a high-rate locally list-recoverable code $C_4$ with a *constant* alphabet size.

4. Finally, we apply once more the amplification procedure with the outer code being the code $C_4$ obtained in the previous step, and the inner code being a *non-efficient* capacity-achieving globally list-recoverable code $C_4'$ over a constant-size alphabet. This gives the capacity-achieving locally list recoverable code $C$ with the desired properties.

In more detail, fix $R \in (0,1)$, $\gamma > 0$, and $\ell \in \mathbb{N}$, we shall construct the code $C := C_N$ as follows.

**The code $C_1$:** Let $C_1$ be the length $N$ high-rate locally list-recoverable code given by Lemma 5.2 with parameters $\frac{\gamma}{5}$, and $\ell_1 = O_{R,\gamma,\ell}(1)$ to be determined later on.

Then by Lemma 5.2, $C_1$ is a code of block length $N$, rate $1-\frac{\gamma}{5}$, and alphabet size

$$q_1 = \exp_{R,\gamma,\ell} \exp\left((\log N)^{1/6} \cdot (\log\log N)^{5/6}\right),$$

that is $(t_1, \alpha_1, \ell_1, L_1)$-locally list recoverable for

$$t_1 = \exp_{R,\gamma,\ell}\left((\log N)^{5/6} \cdot (\log\log N)^{1/6}\right), \qquad \alpha_1 = \Omega_{R,\gamma,\ell}\left(\left(\frac{\log\log N}{\log N}\right)^{1/6}\right),$$

$$\text{and} \qquad L_1 = \exp_{R,\gamma,\ell}\left((\log N)^{2/3} \cdot (\log\log N)^{1/3}\right).$$

Moreover, $C_1$ can be encoded in time $\mathsf{poly}(N)$, and the local list-recovery algorithm for $C_1$ has running time $\mathsf{poly}(t_1)$.

**The code $C_1'$:** Let $C_1'$ be the efficient capacity-achieving (globally) list-recoverable code given by Fact C.4 with parameters $1-\frac{\gamma}{5}$, $\frac{\gamma}{10}$, and $\ell_1' = O_{R,\gamma,\ell}(1)$ to be determined later on. We choose the block length $s_1$ of $C_1'$ so that $q_1 = (q_1')^{(1-\gamma/5)\cdot s_1}$, where $q_1' = \mathsf{poly}_{R,\gamma,\ell}(s_1)$ is the alphabet size of $C_1'$ guaranteed by Fact C.4. Note that under this choice, we have that

$$s_1 = \exp_{R,\gamma,\ell}\left((\log N)^{1/6} \cdot (\log\log N)^{5/6}\right). \tag{28}$$

Then by Fact C.4, $C_1'$ is a code of block length $s_1$, rate $1-\frac{\gamma}{5}$, and alphabet size $q_1'$, that is $(\frac{\gamma}{10}, \ell_1', L_1')$-globally list recoverable for $L_1' = O_{R,\gamma,\ell}(1)$. Moreover, $C_1'$ can be encoded in time $\mathsf{poly}(s_1)$, and globally list-recovered in time $\mathsf{poly}_{R,\gamma,\ell}(s_1)$.

**The code $C_2$:** Next set $\ell_1 := L_1' = O_{R,\gamma,\ell}(1)$, and note that our choice of $s_1$ in (28) guarantees that $q_1 = (q_1')^{(1-\gamma/5)\cdot s_1}$, and $s_1 \geq \left(\alpha_1 \cdot \frac{\gamma}{20}\right)^{-b_0}$, where $b_0$ is the constant guaranteed by Lemma C.1. Let $C_2$ be the code obtained by applying the amplification procedure of Lemma C.1 with the outer code $C_1$ and with the inner code $C_1'$, and with parameter $\frac{\gamma}{20}$.

Then by Lemma C.1, we have that $C_2$ is a code of block length $N$, rate at least $1-\frac{2\gamma}{5}$, and alphabet size

$$q_2 = \exp_{R,\gamma,\ell} \exp\left((\log N)^{1/6} \cdot (\log\log N)^{5/6}\right),$$

that is $(\mathsf{poly}(t_1), \frac{\gamma}{20}, \ell_1', L_1)$-locally list recoverable. Moreover, $C_2$ can be encoded in time $\mathsf{poly}(N)$, and the local list-recovery algorithm for $C_2$ has running time $\mathsf{poly}(t_1)$.

**The code $C_2'$:** Let $C_2'$ be the efficient capacity-achieving (globally) list-recoverable code given by Fact C.4 with parameters $1-\frac{\gamma}{5}$, $\frac{\gamma}{10}$, and $\ell_2' = O_{R,\gamma,\ell}(1)$ to be determined later on. Similarly to the above, we choose the block length $s_2$ of $C_2'$ so that $q_2 = (q_2')^{(1-\gamma/5)\cdot s_2}$, where $q_2' = \mathsf{poly}_{R,\gamma,\ell}(s_2)$ is the alphabet size of $C_2'$ guaranteed by Fact C.4. Note that under this choice, we have that

$$s_2 = \exp_{R,\gamma,\ell}\left((\log N)^{1/6} \cdot (\log\log N)^{5/6}\right). \tag{29}$$

Then by Fact C.4, $C_2'$ is a code block length $s_2$, rate $1-\frac{\gamma}{5}$, and alphabet size $q_2'$, that is $(\frac{\gamma}{10}, \ell_2', L_2')$-globally list recoverable for $L_2' = O_{R,\gamma,\ell}(1)$. Moreover, $C_2'$ can be encoded in time $\mathsf{poly}(s_2)$, and globally list-recovered in time $\mathsf{poly}_{R,\gamma,\ell}(s_2)$.

**The code $C_3$:** Next set $\ell_1' := L_2' = O_{R,\gamma,\ell}(1)$, and note that our choice of $s_2$ in (29) guarantees that $q_2 = (q_2')^{(1-\gamma/5)\cdot s_2}$. Let $C_3$ be the code obtained by concatenating the outer code $C_2$ with the inner code $C_2'$.

Then by Lemma C.2, we have that $C_3$ is a code of block length at least $N$, rate at least $1-\frac{3\gamma}{5}$, and alphabet size

$$q_3 = \exp_{R,\gamma,\ell}\left((\log N)^{1/6} \cdot (\log\log N)^{5/6}\right),$$

that is $(\mathsf{poly}(t_1), \frac{\gamma^2}{200}, \ell_2', L_1)$-locally list recoverable. Moreover, $C_3$ can be encoded in time $\mathsf{poly}(N)$, and the local list-recovery algorithm for $C_3$ has running time $\mathsf{poly}(t_1)$.

**The code $C_3'$:** Let $C_3'$ be the capacity-achieving (globally) list-recoverable code over constant-size alphabet given by Fact C.3 with parameters $1-\frac{\gamma}{5}$, $\frac{\gamma}{10}$, and $\ell_3' = O_{R,\gamma,\ell}(1)$ to be determined later on. Similarly to the above, we choose the block length $s_3$ of $C_3'$ so that $q_3 = (q_3')^{(1-\gamma/5)\cdot s_3}$, where $q_3' = O_{R,\gamma,\ell}(1)$ is to be determined later on. Note that under this choice, we have that

$$s_3 = \Theta_{R,\gamma,\ell}\left((\log N)^{1/6} \cdot (\log\log N)^{5/6}\right). \tag{30}$$

Then by Fact C.3, $C_3'$ is a code of block length $s_3$, rate $1-\frac{\gamma}{5}$, and alphabet size $q_3'$, that is $(\frac{\gamma}{10}, \ell_3', L_3')$-globally list recoverable for $L_3' = O_{R,\gamma,\ell}(1)$. Moreover, $C_3'$ can be encoded in time $\exp_{R,\gamma,\ell}(s_3^2) \leq \mathsf{poly}(N)$, and globally list-recovered in time $\exp_{R,\gamma,\ell}(s_3)$.

**The code $C_4$:** Next set $\ell_2' := L_3' = O_{R,\gamma,\ell}(1)$, and note that our choice of $s_3$ in (30) guarantees that $q_3 = (q_3')^{(1-\gamma/5)\cdot s_3}$. Let $C_4$ be the code obtained by concatenating the outer code $C_3$ with the inner code $C_3'$.

Then by Lemma C.2, we have that $C_4$ is a code of block length at least $N$, rate at least $1-\frac{4\gamma}{5}$, and alphabet size

$$q_4 = q_3' = O_{R,\gamma,\ell}(1),$$

that is $(\mathsf{poly}(t_1), \frac{\gamma^3}{2000}, \ell_3', L_1)$-locally list recoverable. Moreover, $C_4$ can be encoded in time $\mathsf{poly}(N)$, and the local list-recovery algorithm for $C_4$ has running time $\mathsf{poly}(t_1)$.

**The code $C_4'$:** Let $C_4'$ be the capacity-achieving (globally) list-recoverable code over constant-size alphabet given by Fact C.3 with parameters $R + \frac{4\gamma}{5}$, $\frac{\gamma}{10}$, and $\ell$. We choose the block length $s_4$ of $C_4'$ to be

$$s_4 = \left(\frac{\gamma^4}{20,000}\right)^{-b_0} = O_\gamma(1), \tag{31}$$

where $b_0$ is the constant guaranteed by Lemma C.1. We set $q_4 = q_3' = O_{R,\gamma,\ell}(1)$ to be the maximum between the alphabet size of $C_3'$ guaranteed by Fact C.3, and $(\hat{q}_4)^{(R+4\gamma/5)\cdot s_4}$, where $\hat{q}_4 = O_{R,\gamma,\ell}(1)$ is the alphabet size of $C_4'$ guaranteed by Fact C.3. Finally, we choose the alphabet size of $C_4'$ to be $q_4' \geq \hat{q}_4$ so that $q_4 = (q_4')^{(R+4\gamma/5)\cdot s_4}$, noting that $q_4' = O_{R,\gamma,\ell}(1)$.

Then by Fact C.3, $C_4'$ is a code of block length $s_4$, rate $R + \frac{4\gamma}{5}$, and alphabet size $q_4'$, that is $(1 - R - \frac{9\gamma}{10}, \ell, L_4')$-globally list recoverable for $L_4' = O_{R,\gamma,\ell}(1)$. Moreover, $C_4'$ can be encoded and globally list-recovered in time $O_{R,\gamma,\ell}(1)$.

**The code $C$:** Finally, set $\ell_3' := L_4' = O_{R,\gamma,\ell}(1)$, and note that our choice of $s_4$, $q_4$, and $q_4'$ in (31) guarantees that $q_4 = (q_4')^{(R+4\gamma/5)\cdot s_4}$, and $s_4 \geq \left(\frac{\gamma^4}{20,000}\right)^{-b_0}$, where $b_0$ is the constant guaranteed by Lemma C.1. Let $C$ be the code obtained by applying the amplification procedure of Lemma C.1 with the outer code $C_4$ and with the inner code $C_4'$, and with parameter $\frac{\gamma}{10}$.

Then by Lemma C.1, we have that $C$ is a code of block length at least $N$, rate at least $R$, and alphabet size $O_{R,\gamma,\ell}(1)$, that is $(\mathsf{poly}(t_1), 1 - R - \gamma, \ell, L_1)$-locally list recoverable. Moreover, $C_2$ can be encoded in time $\mathsf{poly}(N)$, and the local list-recovery algorithm for $C_2$ has running time $\mathsf{poly}(t_1)$.

□