Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 110 (2024)

Time and Space Efficient Deterministic Decoders

Joshua Cook* Dana Moshkovitz†

November 7, 2024

Abstract

Time efficient decoding algorithms for error correcting codes often require linear space. However, locally decodable codes yield more efficient *randomized* decoders that run in time $n^{1+o(1)}$ and space $n^{o(1)}$. In this work we focus on *deterministic* decoding. Gronemeier [Gro06] showed that any *non-adaptive* deterministic decoder for a good code running in time $n^{1+\delta}$ must use space $n^{1-\delta}$.

In sharp contrast, we show that typical locally correctable codes have (non-uniform) time and space efficient *adaptive deterministic* decoders. For instance, the constant rate, constant relative distance codes with sub-linear query complexity of [Kop+17] have non-uniform deterministic decoders running in time $n^{1+o(1)}$ and space $n^{o(1)}$. To obtain the decoders we devise a new time-space efficient derandomization technique that works by iterative correction.

Further, we give a new construction of curve samplers that allow us to *uniformly* decode Reed-Muller codes time and space efficiently. In particular, for any constant $\gamma > 0$, we give asymptotically good Reed-Muller codes that are decodable in time $n^{1+\gamma}$ and space n^{γ} by a uniform, deterministic decoder. A related construction allows us to *uniformly* decode asymptotically good codes based on lifted Reed-Solomon codes in time $n^{1+o(1)}$ and space $n^{o(1)}$.

*jac22855@utexas.edu. Department of Computer Science, UT Austin. This material is based upon work supported by the National Science Foundation under grant number 2200956.

†danama@cs.utexas.edu. Department of Computer Science, UT Austin. This material is based upon work supported by the National Science Foundation under grant numbers 2200956 and 2312573.

ISSN 1433-8092

Contents

1 Introduction

Time efficient decoding algorithms for error correcting codes often require linear space. For instance, efficient decoders for Reed-Solomon codes, starting with [BW86; Sud97], construct and manipulate a polynomial that represents the received word. Efficient decoders for expander codes [SS96; Spi95] iteratively update each index of the supposed codeword. In this work we focus on time-efficient decoders that work in sub-linear space. We believe that this is a natural problem for coding theory. Moreover, in complexity theory error correcting codes are the basis of many important objects, such as pseduorandom generators [KU06; STV01; Uma03], probabilistically checkable proofs [Lun+90; Bab+91; Fei+91; AS98; Aro+98] and delegation schemes [GKR15; KRR21]. The bounded space setting is of interest for all of these objects [Nis90; BS+13; RRR16; HR18; CM23], and therefore motivates a study of error correcting codes in bounded space.

Are there error correcting codes that can be decoded in nearly linear time and sub-linear space? The answer is yes for *randomized* decoders thanks to locally decodable codes (see, e.g., the survey [Yek12]). Given a corrupted codeword, a local decoder can decode any message symbol in sub-linear randomized time. With $O(\log n)$ repetitions we can decrease the error probability sufficiently below $1/n$, so the decoder can recover the whole n bit message with probability at least $2/3$. Meanwhile, the space used is sub-linear as well, depending on the number of queries of the decoder. The Reed-Muller code is locally decodable, and in recent years there have been many other constructions of locally decodable codes, such as multilicity codes [KSY14], lifted codes [GKS13], and expander based constructions [HOW13]. In particular, there are locally decodable codes of constant rate that use sub-linear number of queries $n^{o(1)}$ and therefore yield randomized (global) decoders that run in $n^{1+o(1)}$ time and $n^{o(1)}$ space [Kop+17].

But what about deterministic decoders? Gronemeier [Gro06] showed that *non-adaptive* deterministic decoders that run in nearly linear time $n^{1+\delta}$ must use nearly linear space $n^{1-\delta}$. Are general deterministic decoders, which could be adaptive, ruled out as well? If so, this would be a remarkable demonstration of the power of randomness. We know that randomness can speed up computation thanks to algorithms like local decoders that only read a small portion, at most $n^{o(1)}$, of their input, whereas any deterministic algorithm must read most of the input $\Omega(n)$, so randomized algorithms are faster than deterministic algorithms by a nearly linear factor in the input size. However, randomness vs. deterministic separations for problems like (global) decoding where even the randomized algorithm requires linear time are much harder to obtain. To the best of our knowledge, such a separation (specifically: univariate polynomial identity testing for polynomials of degree n requires time $n^{2-o(1)}$ deterministically but can be solved in time $\tilde{O}(n)$ randomly) is only known under the strong, quite possibly false, Nondeterministic Strong Exponential Time Hypothesis [Wil16]. The main result of our work is that even though there is a nearly linear factor randomness vs. deterministic separation for local decoding, there is no such separation for global decoding.

1.1 Asymptotically Good Codes With Time-Space Efficient Decoders

The main result of this paper is that there exists deterministic decoders that run in nearly linear time $n^{1+o(1)}$ and sub-linear space $n^{o(1)}$ for error correcting codes with constant rate and constant relative distance. Further, these codes are explicit and their decoders are uniform.

Theorem 1.1 (Codes With *Uniform* Time And Space Efficient Decoders)**.** *There exists an infinite family of codes $C : \{0,1\}^n \to \{0,1\}^m$ where $m = O(n)$ and a deterministic uniform algorithm B that computes a function $D : \{0,1\}^m \to \{0,1\}^n$ such that:*

Efficient: *B runs in time $n^{1+o(1)}$ and space $n^{o(1)}$.*

Decodes: *For some decoding radius $d = \Omega(n)$, for any $x \in \{0,1\}^n$ and $w \in \{0,1\}^m$ with $\Delta(w, C(x)) \leq d$ we have that*

$$D(w) = x.$$

In fact, we give efficient code correction, not just decoding. Code correction is correction where one does not output the message, but the codeword corresponding to the message. Correction is often viewed as harder than decoding because correction of linear codes implies non-uniform decoding. Our explicit codes are based on lifted Reed-Solomon codes [GKS13; GK16]. Similar techniques also give efficient uniform decoders for Reed-Muller codes.

1.2 Time-Space Efficient Deterministic Decoders for Locally Correctable Codes

We prove Theorem 1.1 by showing that known constructions of locally *correctable* codes give rise to time-space efficient *deterministic* decoders. A locally correctable code is similar to a locally decodable code, except that it decodes codeword symbols and not just message symbols. Reed-Muller is locally correctable in addition to locally decodable, and so are most of the locally decodable codes with constant rate and sub-linear number of queries. We focus on *typical* locally correctable codes, which are locally correctable codes that satisfy several basic properties: smoothness, perfect completeness, non-adaptivity and systematic encoding (see Definitions 3.1 and 3.6). These properties are satisfied by all the constructions we mention (sometimes with minor modifications). We prove that any typical locally correctable code has time and space efficient deterministic non-uniform decoders.

Theorem 1.2 (Locally Correctable Codes Have Efficient Deterministic Decoders)**.** *Suppose some code $C : \Sigma^n \to \Sigma^m$ is a typical locally correctable code with q queries, correcting radius $\Omega(m)$, and soundness $\frac{1}{3}$ (see Definition 3.6).*

Then for any integer $\ell \geq 1$, the code C has a (non-uniform) deterministic decoder with decoding radius $\Omega(m)$ running in time

$$O(m^{1+1/\ell}(q\log(m))^{O(\ell)}\log(|\Sigma|))$$

and space

$$O(\ell q \log(m)\log(|\Sigma|)).$$

As a direct consequence of this theorem, existing constructions of asymptotically good locally correctable codes that have $q = m^{o(1)}$ are time $m^{1+o(1)}$ and space $m^{o(1)}$ deterministically decodable. Theorem 1.2 can be thought of as a derandomization result that converts a randomized (local) corrector to a deterministic corrector *with essentially the same time and space complexity!*

1.3 New Derandomization Method

The proof of Theorem 1.2 gives rise to a *general derandomization method* that produces deterministic algorithms with essentially the same complexity as the randomized algorithm. In contrast, most derandomization techniques, e.g., via pseudorandom generators (see, e.g., the survey [Gol]) or via Adleman's argument [Adl78], result in deterministic algorithms that are slower by factor at least n (the input size) compared to the randomized algorithm. There are very few techniques, like the method of conditional probabilities (see, e.g., the book [AS98]) or derandomization via sketching [GM20], that result in deterministic algorithms with nearly the same time complexity as the corresponding randomized algorithm. Each of those derandomization techniques can be applied only in a certain setup. Next we'll give an overview of our new derandomization method and the setup in which it is applicable. We hope that the method will find more applications in the future.

Our method is relevant when it is possible to divide the algorithmic task at hand to a large number of smaller tasks, such that the following conditions hold:

1. Each small task can be solved using a small number of random bits.
2. There is an efficient algorithm to check whether a setting of the random bits leads to a correct solution for the small task.

The deterministic algorithm works by sequentially completing the small tasks by enumerating all possible settings of the random bits for each. The point of the method is that if there are r randomness strings per task and ℓ tasks, then there are r^ℓ randomness strings that the algorithm might use overall, but the algorithm only needs to check $r \cdot \ell$ randomness strings.

In Section 1.4 we will explain how our decoding algorithm works in detail, but here is the quick overview of how it fits into this scheme: In the case of decoding, the task is to decrease the number of errors in the received word from $d = \Theta(m)$ to 0. We break the task to a large number ℓ of small tasks, where the i'th task for $i = 0, 1, ..., \ell - 1$ is to decrease the number of errors from roughly $m^{1-i/\ell}$ to less than $m^{1-(i+1)/\ell}$. We show that roughly only $m^{1/\ell}$ randomness strings suffice for each one of the ℓ tasks. Moreover, we show how to verify in nearly linear time and low space that a randomness string decreases the number of errors sufficiently.

This results in an algorithm that runs in nearly linear time, roughly $m^{1+1/\ell}$, and is space-efficient. Note that the total number of randomness strings the algorithm might end up using to decode the message is roughly $\Theta((m^{1/\ell})^\ell) = \Theta(m)$ as we know is necessary for local decoding.

1.4 Efficient Derandomization By Iterative Correction

In this section we give more details about the proof of Theorem 1.2. As discussed before, one can repeat a (randomized) local corrector $\Theta(\log m)$ times to ensure that for each one of the m indices the probability the correct codeword symbol is not computed correctly is smaller than $1/3m$. In this case, the entire length-m codeword that is close to the input word is output with probability at least $2/3$. This randomized corrector runs in time $O(mq \log m)$ and space $O(q \log m)$ for q the number of queries of the local corrector. Every fixing of the randomness to the repeated local corrector defines a deterministic corrector, which may or may not output the nearby codeword. Importantly, we can *test* whether the output word is close to the input word, and whether it is a codeword, in time $O(mq \log m)$ and space $O(q \log m)$. (In Section 3 we prove that a typical locally correctable code also gives rise to an efficient codeword tester for words that are not too far from the code).

Therefore, we can obtain a deterministic corrector by enumerating over all the possible choices of randomness for the repeated local corrector and testing the outcome of each one. Since there are only 2^m possible inputs, only $O(m)$ different randomness strings suffice by a Chernoff bound. Hence, we get a (possibly non-uniform) deterministic corrector that runs in time $O(m^2 q \log m)$ and space $O(q \log m)$. Our goal is to obtain a deterministic algorithm in near linear time, not near quadratic time, but how can we? There are necessarily $\Omega(m)$ possible randomness strings, and testing each randomness string necessarily requires time $\Omega(m)$.

This is where iterative correction enters the picture. In each iteration of iterative correction the corrector decreases the distance of the word from the code by a little, until eventually the distance becomes 0. Specifically, if the distance is d the corrector decreases it to at most $d/m^{1/\ell}$. Since initially the number of corruptions could be $\Omega(m)$, after about ℓ iterations the distance drops below 1 and correction is complete. The heart of the technique is a proof that in order to decrease the distance from the code by a factor of $m^{1/\ell}$ the number of randomness strings we need to check in each iteration is only $O(m^{1/\ell} \log m)$, few enough we can deterministically check them all. We will discuss the proof of this claim at the end of this sub-section, and for now continue assuming it. To test whether a randomness string leads to sufficient correction of the word, the corrector now needs to estimate the distance of the word from the code. In Section 3 we show that this too can be done efficiently for typical locally correctable codes.

Crucially, iterative correction can be implemented in small space. The corrector will only store the randomness strings it identified so far (at most ℓ). Whenever it needs to query the current word, it will use at most $(q \log m)^\ell$ queries to the input word to compute the symbol on the fly. See Figure 1.

The crux of the iterative correction technique is that the total number of *sequences* of random strings that may be needed to go from $\Omega(m)$ corruptions to no corruptions (over all possible corrupted codewords) is $O((m^{1/\ell} \log m)^\ell) = O(m(\log m)^\ell)$, not much larger than the number $\Theta(m)$ we know is necessary, however the number of tests we need to do per corrupted codeword is only $O(\ell m^{1/\ell} \log(m))$.

Finally, let us explain why in order to decrease the distance of a word from the code from d to ηd we need only $O((1/\eta) \log m)$ randomness strings. First, consider the case of $d = \Theta(m)$, which is the case in the first iteration. We will show that only $O(1/\eta)$ randomness strings suffice in this case. Think of a table with rows that are all the $M = \Theta(m)$ randomness strings we found earlier and with columns that are all the m indices we wish to correct. Every entry in the table corresponds to a possible correction of an index by a randomness. We know that at most $(Mm)/m$ of the entries in this table correspond to incorrect outcomes. Thus, if there are more than t randomness strings that fail for at least ηd indices, it must hold that $t \cdot \eta d \leq M$. Therefore, $t \leq (1/\eta)M/d = \Theta(1/\eta)$, and only $O(1/\eta)$ randomness strings suffice. For general d we use the properties of a typical local corrector to argue that we need only start with $O(d \log m)$ randomness strings and not with $O(m)$ randomness strings. This follows from a Chernoff bound, similarly to how we initially derived the $O(m)$ result. The difference is instead of union bounding over all possible 2^m inputs, we union bound over $2^{d \log m}$ possibilities, using that only the pattern of corruption matters for the success of correction, and that there are only $\binom{m}{d} \leq 2^{d \log m}$ possible corruption patterns. More details can be found in Section 5.

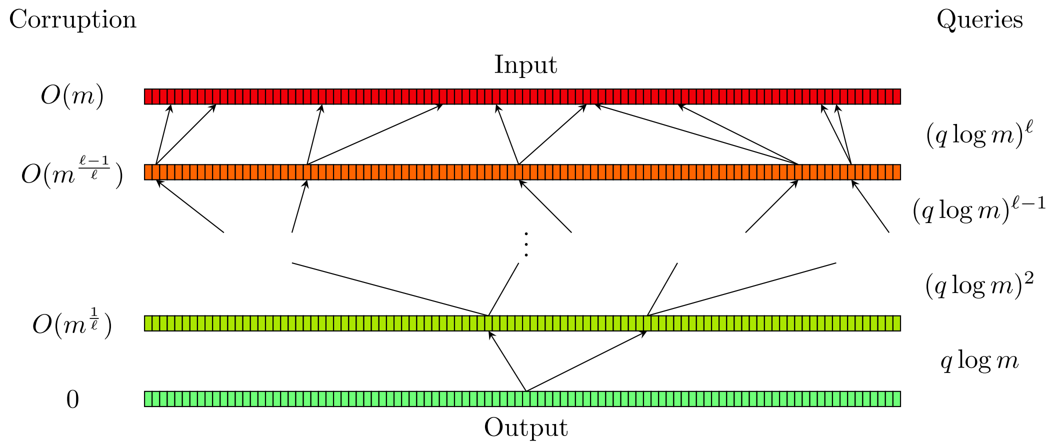

Figure 1: Iterative Correction. There are ℓ layers of codeword improvers built using a local corrector. The first layer is the input with a linear amount of corruption. Each codeword improver reduces the fraction of errors by $m^{1/\ell}$ by making $q \log m$ queries (per symbol) to the layer before it. After ℓ iterations the corruption is reduced to 0, and the final layer can be computed using $(q \log m)^\ell$ queries (per symbol) to the input.

1.5 Explicit Curve Samplers and Uniform Decoding

Our generic derandomization method produces non-uniform algorithms, since it requires a small family of good randomness strings for the local corrector. Such a family exists by a probabilistic argument, and we can hard-wire it into the algorithm when the algorithm is non-uniform.

For specific codes and local correctors it is possible to construct this pseudorandom set explicitly, and thereby obtain a uniform decoder. We demonstrate this using Reed-Muller codes and lifted Reed-Solomon codes.

1.5.1 Reed-Muller Codes and Curve Samplers

We give explicit constructions of *curve samplers* that imply a time and space efficient, uniform, deterministic decoder for Reed-Muller codes. It is known that the family of all degree-k curves in a space $\mathbb{F}_p^{\mathsf{dim}}$ is a good sampler, in the sense that for any μ fraction of points $A \subseteq \mathbb{F}_p^{\mathsf{dim}}$, for a random degree-$k$ curve, c, about μ fraction of the points on the curve falls in A with high probability. Specifically,

$$\Pr_c[\Pr_{t \in \mathbb{F}}[c(t) \in A] > \mu + \epsilon] \leq \mu\delta, \tag{1}$$

where $\epsilon = p^{-\Omega(1)}$ and $\delta = p^{-\Omega(k)}$ (see, e.g., [Mos17]). We call ϵ the accuracy error and δ the strong confidence error. By the probabilistic method, there exists a family of $p^{\mathsf{dim}+O(k)}$ curves (as opposed to the number of all curves: $p^{\mathsf{dim}\cdot(k+1)}$) that satisfies the aforementioned sampling property. Ta-Shma and Umans [TSU06] and Guo [Guo13] constructed explicit curve samplers, however these fall short of the parameters we stated in three ways. First, their degree is $poly(k)$ rather than k. Second, the number of curves is $p^{O(\mathsf{dim}+k)}$ as opposed to $p^{\mathsf{dim}+O(k)}$. Third, their probability bound in Eq. (1) is δ rather than $\mu\delta$. The last two issues are crucial for an efficient uniform decoder.

We construct a new family of efficient curve samplers that overcomes all of these shortcomings.

Theorem 1.3 (Efficient Curve Sampler)**.** *For any prime power p, integers dim and $b \geq 2$ such that $b | \mathsf{dim}$, there is a degree b-curve sampler $\mathcal{C}$ for $\mathbb{F}_p^{\mathsf{dim}}$ such that for every $\epsilon > 0$, the sampler $\mathcal{C}$ has accuracy error ϵ and strong confidence error $2b\left(\frac{2b}{\epsilon\sqrt{p}}\right)^b$. The size of $\mathcal{C}$ is $|\mathcal{C}| = p^{\mathsf{dim}+poly(b)} poly(\mathsf{dim})$. Further, given the index of a curve $c \in \mathcal{C}$ and an element $x \in \mathbb{F}_p$ we can evaluate $c(x)$ in time $poly(p^b \mathsf{dim})$.*

Our construction is to first sample a line in $\mathbb{F}_{p^b}^{\mathsf{dim}/b}$ using ε-biased sets. This line can be interpreted as a subspace of $\mathbb{F}_p^{\mathsf{dim}}$. Then we sub-sample this subspace using a random degree b curve.

To explain the construction, the set of lines whose directions are taken from an ε-biased set is a sampler with confidence error inversely proportional to the field size [BS+03]. Hence, we get a small confidence error by working over a large field, that of $\mathbb{F}_{p^b}$ instead of $\mathbb{F}_p$. The large field gives a large sample, and the sub-sampling step decreases the sample size.

The approach of increasing the field size in order to decrease the confidence error and then sub-sampling is the basis of the Ta-Shma–Umans [TSU06] construction as well. However, their choice of the sample over the large field is a random curve, and they have many sub-sampling iterations. Picking a random curve in the first iteration is the cause of the large number of curves in their final construction, and we obtain our result by picking a smaller family. The higher than 1 degree of the sample is the reason for the higher degree in their final construction, which we eliminate by considering a degree-1 construction. Finally, we achieve a probability bound $\mu\delta$ in Eq. (1) as opposed to δ by a more careful analysis of line and curve samplers.

1.5.2 Lifted Reed-Solomon Codes and Correcting Lines

Our approach to correcting lifted Reed-Solomon codes is similar, except that we cannot use curves to correct lifted Reed-Solomon codes, we have to use lines. A lifted Reed-Solomon code has as codewords polynomials such that restricting that polynomial to any line through its domain gives a low degree polynomial. Reed-Muller codes are a special case of lifted Reed-Solomon codes, but for large degrees and fields with small characteristic lifted Reed-Solomon codes have much better rate than Reed-Muller codes [GKS13].

Unfortunately, since we are only promised that lifted Reed-Solomon codes are low degree polynomials on lines, they may not be low degree polynomials when restricted to curves. Thus our correctors for lifted Reed-Solomon codes must be lines. Unfortunately, a random line does not correct with high enough probability to just correct with lines. However, we can handle this the same way we handle it in the non-uniform case: choose several lines through a point, correct each one and take the majority vote. We just need to choose these lines efficiently.

Inspired by our curve sampler, we first sub-sample a space, and then we will choose lines in that subspace. Unfortunately, to get a good enough choice of decoding lines, we need to choose many lines in the subspace, too many to choose each independently. To get around this, we can use any good enough sampler to choose the lines in the subspace. We use a random curve through the subspace as our sampler to choose the lines.

One last detail is that lifted Reed-Solomon codes with sub polynomial query complexity do not have good distance. We handle this the same way as [Kop+17]: by using graph based distance amplification to increase the correcting radius from $m^{1-o(1)}$ to $\Omega(m)$.

1.6 Related Work

Known constructions of locally correctable codes are not known to be *encodable* in time $n^{1+o(1)}$ and space $n^{o(1)}$. Cook and Moshkovitz [CM24] constructed error correcting codes that can be encoded in time $n^{1+o(1)}$ and space $n^{o(1)}$, but it does not seem like they can be decoded with this complexity. It remains open whether there is a code that can be both encoded and decoded in time $n^{1+o(1)}$ and space $n^{o(1)}$.

Linear time decoders for expander codes [SS96; Spi95] iteratively correct the received word like our iterative corrector, however they cannot be implemented in sub-linear space, since they need to maintain each symbol in the current word, and cannot compute it on the fly. We use local correction to ensure low space.

Our iterative correction technique can be thought of as a more efficient version of Adleman's derandomization [Adl78], which decreases the error probability below 2^{-n} (by repetition, increasing the time; here n is the input size) and then hard-wires to the algorithm a randomness string that works for all 2^n inputs. Prior work [GM20] had a different approach for an efficient Adleman derandomization, one that focused on faster testing of possible randomness strings (via an algorithm for finding a biased coin), and dividing the inputs to groups with similar behavior with respect to the randomness (via sketching). The faster testing approach does not apply to our problem, since it may yield a word that is close to a codeword instead of being a codeword.

2 Preliminaries

Our results are about decoding error correcting codes. An error correcting code is a function that maps any two distinct messages to codewords that are far apart in Hamming distance. The Hamming distance of two strings is the number of indexes where they differ.

Definition 2.1 (Hamming Distance)**.** *For any alphabet Σ and integer n, we define the Hamming distance $\Delta : \Sigma^n \times \Sigma^n \to \mathbb{N}$ by*

$$\Delta(x, y) = |\{i \in [n] : x_i \neq y_i\}|.$$

Now we can define an error correcting code, often just called a code.

Definition 2.2 (Error Correcting Code)**.** *For alphabets Σ_1 and Σ_2, integers $n, m \in \mathbb{N}$ and distance $d \in \mathbb{N}$, a code is a function $C : \Sigma_1^n \to \Sigma_2^m$ such that for all $x, y \in \Sigma_1^n$ with $x \neq y$ we have that*

$$\Delta(C(x), C(y)) \geq d.$$

We call d the distance of C, Σ_2 the alphabet of the code, and Σ_1 the alphabet of the message. We call $\frac{n}{m}$ the rate of the code.

Any $x \in \Sigma_1^n$ we call a message and we call $C(x)$ its codeword. We define the codewords of C to be the set $\{C(x) : x \in \Sigma_1^n\}$.

We say the code is asymptotically good if $d = \Omega(m)$ and $m = O(n)$.

Often an error correcting code is defined in terms of its set of codewords and not its encoding function. In particular

Lemma 2.3 (Codes From Codewords)**.** *Suppose that there is a set $C' \subseteq \Sigma^m$ such that for all $u, v \in \Sigma^m$ we have that $\Delta(x, y) \geq d$. Then there is a code $C : [|C'|] \to \Sigma^m$ with distance d such that the set of codewords for C is C'.*

This comes from just ordering the elements of C' in an arbitrary order and letting $C(i)$ output the ith element.

A closely related concept to Hamming distance is the weight of a string. The weight of a string is the number of non-zero coordinates it has. If two strings have a binary alphabet, then the weight of their difference is their Hamming distance.

Definition 2.4 (Weight)**.** *For any Σ where $0 \in \Sigma$, for any $z \in \Sigma^n$ we define the weight of z by*

$$wt(z) = |\{i \in [n] : z_i \neq 0\}|.$$

We may refer to the distance of a string to a code. This is the distance from that string to the nearest codeword of that code.

Definition 2.5 (Distance To A Set)**.** *For any set $S \subseteq \Sigma_2^m$ and $w \in \Sigma_2^m$ define the distance of w to S by*

$$\Delta(w, S) = \min_{y \in S} \Delta(w, y).$$

If $C : \Sigma_1^n \to \Sigma_2^m$ is a code, we define the distance of w to C by

$$\Delta(w, C) = \min_{x \in \Sigma_1^n} \Delta(w, C(x)).$$

The primary feature of codes we are interested in is decoding.

Definition 2.6 (Decoding)**.** *Suppose we have a code $C : \Sigma_1^n \to \Sigma_2^m$. Then we say a function $D : \Sigma_2^m \to \Sigma_1^n$ is a decoder with decoding radius d if for all messages $x \in \Sigma_1^n$ and strings $y \in \Sigma_2^m$ with $\Delta(C(x), y) \leq d$ we have that $D(y) = x$. We call $\frac{d}{m}$ the relative decoding radius.*

If D can be computed in time T and space S, we say that C is decodable in time T and space S. We say C is time T and space S decodable if such a D exists.

In contrast to decoding, one can also ask for the codeword to be corrected. This is where our decoder, instead of outputting the message x, it instead outputs the codeword $C(x)$.

Definition 2.7 (Correcting)**.** *Suppose we have a code $C : \Sigma_1^n \to \Sigma_2^m$. Then we say a function $D : \Sigma_2^m \to \Sigma_2^m$ is a corrector with correcting radius d if for all messages $x \in \Sigma_1^n$ and strings $y \in \Sigma_2^m$ with $\Delta(C(x), y) \leq d$ we have that $D(y) = C(x)$. We call $\frac{d}{m}$ the relative correcting radius.*

If D can be computed in time T and space S, we say that C is correctable in time T and space S with a correcting radius of d.

Our non-uniform algorithms are RAM algorithms with advice. These are RAM algorithms with:

1. A read only input tape.
2. A write only output tape.
3. A bounded space working tape.
4. A read only advice tape.

Where the algorithm has random access to the input, advice, and working tape. The space of the algorithm is the size of the working tape, and we require the size of the advice tape to be bounded by the run time of the algorithm. We only allow our algorithm to print to the output tape in sequential order.

Note that the output tape can be much longer than the work tape, so the algorithm can output much larger messages than it has space to store itself. The fact that the input tape is read only and output tape is write only is why the standard decoder for Spielman codes are not space efficient.

Another common model of non-uniform computation is the branching program model. Our model of non-uniform computation can be efficiently simulated by branching programs, so we could also state our results in terms of branching programs.

For our explicit codes, we will use alphabets that are finite fields.

Definition 2.8 (Finite Field)**.** *We call an integer p a prime power if there exists a prime number p_0 and positive integer such that $p = p_0^a$. For any prime power p, we denote the field of order p as $\mathbb{F}_p$.*

One of the codes we will consider is the Reed-Muller code. This is the code with codewords that are low degree polynomials over many variables. A special case of Reed-Muller codes are Reed-Solomon codes where the polynomials are over only one variable.

Definition 2.9 (Reed-Muller code)**.** *For prime power p, degree deg, and number of variables dim let $RM_p(\mathsf{deg}, \mathsf{dim})$ be the code with codewords that are polynomials $p : \mathbb{F}_p^{\mathsf{dim}} \to \mathbb{F}_p$ of degree deg represented by evaluating p at all points in $\mathbb{F}_p^{\mathsf{dim}}$. So the alphabet of $RM_p(\mathsf{deg}, \mathsf{dim})$ is $\mathbb{F}_p$ and the codeword length is $m = |\mathbb{F}_p|^{\mathsf{dim}} = p^{\mathsf{dim}}$.*

For a function $f : \mathbb{F}_p^{\mathsf{dim}} \to \mathbb{F}_p$ we say that $f \in RM_p(\mathsf{deg}, \mathsf{dim})$ if f is a degree deg polynomial.

3 Local Code Properties And Their Relationships

Now we define the properties we need for our construction to give a time and space efficient deterministic decoder. While Theorem 1.2 only assumes the code is a typical LCC, we will actually show that a typical LCC has several other useful properties that we will use to give our deterministic decoder. We will first define all the relevant properties and then show relationships between them.

3.1 Local Code Properties

The main local code property we will discuss in this paper is local correction. A locally correctable code is a code with a local corrector. A local corrector is a randomized algorithm, D, that when given oracle access to a string w that is a slightly corrupted version of a codeword y, the corrector D can compute any symbol of y with high probability using few queries to w.

Definition 3.1 (Locally Correctable Codes (LCC))**.** *For any code $C : \Sigma_1^n \to \Sigma_2^m$ with distance greater than $2d$, we say that C is a locally correctable code (LCC) if there exists some local corrector D that takes as input an index $i \in [m]$, a randomness string $r \in \{0,1\}^R$, and oracle access to a string $w \in \Sigma_2^m$ and outputs a single symbol, denoted $D^w(i, r)$, such that*

Locality: *On any input string $w \in \Sigma_2^m$, index $i \in [m]$, and randomness $r \in \{0,1\}^R$ we have that $D^w(i,r)$ can be computed with only q queries to w.*

Soundness: *There is an $s < \frac{1}{2}$ such that for any $w \in \Sigma_2^m$ and $x \in \Sigma_1^n$ with $\Delta(w, C(x)) \leq d$ and any $i \in [m]$ we have*

$$\Pr_{r \in \{0,1\}^R}[D^w(i,r) \neq C(x)_i] \leq s.$$

We call d the correcting radius, q the number of queries, and s the soundness of D. If soundness s is not specified, it is assumed to be $\frac{1}{3}$. If correcting radius d is not specified, it is assumed to be $\Omega(m)$.

While we would like time and space efficient deterministic decoders for any LCC, we are only able to show it for typical LCCs. A typical LCC is a systematic code with a local corrector that is smooth, non-adaptive, and has perfect completeness. We will define each of these individually.

A systematic code is a code such that any message is contained (as plain text) in the associated codeword. We use systematic codes since it gives a straightforward way to get the message from an uncorrupted codeword. Similar results hold for any code with an efficient way to recover the message from an uncorrupted codeword. Systematic codes provide a simple mechanism for doing this.

Definition 3.2 (Systematic Code)**.** *We say that a code $C : \Sigma_1^n \to \Sigma_2^m$ is systematic if for all $i \in [n]$ there exists $j \in [m]$ such that for all $x \in \Sigma_1^n$ we have that $x_i = C(x)_j$.*

A smooth code is a code with a local corrector such that when it corrects any particular symbol, the corrector queries every index of the input with about equal probability. In particular, no index is queried more often then about $\frac{2q}{m}$ times in expectation.

Definition 3.3 (Smooth)**.** *We say that a code $C : \Sigma_1^n \to \Sigma_2^m$ with a local corrector D (where the randomness of D is R) is β smooth if for all $w \in \Sigma_2^m$ and $i, j \in [m]$:*

$$\Pr_{r \in \{0,1\}^R}[D^w(i,r) \text{ queries index } j \text{ of } w] \leq \frac{\beta}{m}.$$

If β is unspecified and D is q query, we assume $\beta = 2q$.

A non-adaptive code is a code with a local corrector such that the indexes of the input which are queried *only* depends on the randomness and the index of the symbol being decoded, *not* on the input being corrected. Put another way, given the index to decode and a random string, the corrector can specify each of the q locations it will query before querying them.

Definition 3.4 (Non-Adaptive)**.** *We say that a code $C : \Sigma_1^n \to \Sigma_2^m$ with a local corrector D (where the randomness of D is R) is non-adaptive if for all $i \in [n]$ and $r \in \Sigma_2^m$, we have that $D^w(i,r)$ always queries the same indexes in w for any $w \in \Sigma_2^m$.*

Finally, a code with a local corrector is said to have perfect completeness if, when it is given a valid codeword, it always outputs the symbols from that codeword. Put another way, local corrections of an uncorrupted codeword always output that codeword.

Definition 3.5 (Perfect Completeness)**.** *We say that a code $C : \Sigma_1^n \to \Sigma_2^m$ with a local corrector D (where the randomness of D is R) has perfect completeness if for all $x \in \Sigma_1^n$, for any $i \in [m]$ and $r \in \{0,1\}^R$ we have that*

$$D^{C(x)}(i,r) = C(x)_i.$$

Now we define a typical LCC to be any systematic code with a corrector that is smooth, non-adaptive and has perfect completeness. These are standard properties of locally testable codes, and all the standard constructions of LCC are typical (with only minor changes). The only assumption on the code in Theorem 1.2 is that it is typical, so the results are very general.

Definition 3.6 (Typical Locally Correctable Codes). *For any systematic code $C : \Sigma_1^n \to \Sigma_2^m$ we say that C is a typical LCC if it is an LCC and has a corrector D that is smooth, non-adaptive, and has perfect completeness.*

An important subroutine in our deterministic decoder is estimating the amount of corruption in an input string. We perform this estimate using a local testing algorithm. The natural property one would hope for is strong local testability.

Definition 3.7 (Strong Locally Testable Codes). *We say that a code $C : \Sigma_1^n \to \Sigma_2^m$ is a strong locally testable code (LTC) with q queries, randomness R, and approximation factor*[1] *α if there exists a tester V that takes as input randomness $r \in \{0,1\}^R$ and oracle access to a string $w \in \Sigma_2^m$ and outputs a single bit, denoted $V^w(r)$, such that:*

Locality: *For any $r \in \{0,1\}^R$ and $w \in \Sigma_2^m$ we have that $V^w(r)$ can be computed with only q queries to w.*

Approximation: *For any $w \in \Sigma_2^m$ we have*

$$\frac{\Delta(w,C)}{m\alpha} \leq \Pr_{r \in \{0,1\}^R}[V^w(r)] \leq \frac{\alpha\Delta(w,C)}{m}.$$

If the approximation factor α is not specified, it is assumed to be 2.

Unfortunately, we don't know how to construct strong local tests for all typical locally correctable codes. But we can get a weaker property, a *vicinity* LTC (VLTC). A VLTC is a strong LTC that is only promised to give a good estimation of the corruption for an input that is already close to a codeword [CY23].

Definition 3.8 (Vicinity Locally Testable Codes). *We say that a code $C : \Sigma_1^n \to \Sigma_2^m$ is a vicinity locally testable code (VLTC) with q queries, randomness R, approximation factor α, and vicinity d if there exists a tester V that takes as input randomness $r \in \{0,1\}^R$ and oracle access to a string $w \in \Sigma_2^m$ and outputs a single bit, denoted $V^w(r)$, such that:*

Locality: *For any $r \in \{0,1\}^R$ and $w \in \Sigma_2^m$ we have that $V^w(r)$ can be computed with only q queries to w.*

Approximation: *for any $w \in \Sigma_2^m$ with $\Delta(w,C) \leq d$ we have*

$$\frac{\Delta(w,C)}{m\alpha} \leq \Pr_{r \in \{0,1\}^R}[V^w(r)] \leq \frac{\alpha\Delta(w,C)}{m}.$$

If the approximation factor α is not specified, it is assumed to be 2.

See that an LTC is just a VLTC with vicinity m.

The last property we will need is that our local corrector has soundness that is only dependent on the corruption, or error pattern, and not on the message itself. We call this property "message oblivious soundness" (MOS). The MOS property allows us to take a union bound over possible corruption patterns instead of possible inputs, which allows us to use fewer randomness strings.

To make this formal, we need to formally define corruptions. The corruption of an input is the difference from the nearest codeword. We emphasize that by difference of two strings, we don't mean the *number* of indexes the strings differ, we mean the actual *indexes* they differ on. That is, difference is not distance, but rather the weight of the difference is the distance.

Definition 3.9 (Difference). *For any strings $w_1, w_2 \in \Sigma_2^m$ the difference between w_1 and w_2 is a vector $z \in \{0,1\}^m$ defined by*

$$z_i = \begin{cases} 1 & C(x)_i \neq w_i \\ 0 & C(x)_i = w_i. \end{cases}$$

Further, for any $w_1 \in \Sigma_1^m$ and $z \in \{0,1\}^m$, we define the set of z differences from w_1 by

$$\mathsf{Diff}(w_1, z) = \{w_2 \in \Sigma_2^m : z \text{ is the difference between } w_1 \text{ and } w_2\}.$$

[1]A closely related property has been called *testability* (e.g. [KM23]) and detection probability (e.g. [Din+22]).

Now we can define message oblivious soundness as an LCC with soundness that is only dependent on the corruption $z \in \{0,1\}^m$ and not on the message $x \in \Sigma_1^n$. This is the property that is easiest to work with for our results. Most standard constructions of locally testable codes also have message oblivious soundness, but message oblivious soundness is a less standard property than those of typical LCCs.

Definition 3.10 (Message Oblivious Soundness Locally Correctable Codes)**.** *For any code $C : \Sigma_1^n \to \Sigma_2^m$ with distance greater than $2d$, we say that C is a message oblivious soundness locally correctable code (MOSLCC) if there exists some local corrector function D that takes as input an index $i \in [m]$, a randomness string $r \in \{0,1\}^R$, and oracle access to a string $w \in \Sigma_2^m$ and outputs a single symbol, denoted $D^w(i, r)$, such that*

Locality: *On any input $i \in [m]$, $r \in \{0,1\}^R$ and $w \in \Sigma_2^m$ we have that $D^w(i, r)$ can be computed with only q queries to w.*

Soundness: *For any $z \in \{0,1\}^m$ with $wt(z) \le d$ and any $i \in [m]$ we have that*

$$\Pr_{r\in\{0,1\}^R}[\exists x \in \Sigma_1^n, w \in \mathit{Diff}(C(x), z) : D^w(i, r) \neq C(x)_i] \le s.$$

We call D a MOS corrector, d the MOS correcting radius, q the number of queries, and s the soundness of D. If soundness s is not specified, it is assumed to be $\frac{1}{3}$.

Intuitively, one might expect local correction success to be a function of the corruption and not the underlying codeword. If the number of corruptions is fixed and small, then most choices of randomness should miss most corruptions, and most locally correctable codes always succeed if they don't see any corruption. This intuition can be made rigorous for typical LCCs, as we will see in the next section. Since the properties of a typical LCC are well known, it may be easier to verify that an LCC is typical than it is to verify it has message oblivious soundness.

3.2 Local Code Property Relationships

One standard fact of LCCs is that one can decrease the soundness (the probability it fails to correct) by correcting several times and taking majority. And if the corrector is a MOS corrector, so is the amplified corrector. These follows from Chernoff bounds and are well known so we don't reprove them.

Lemma 3.11 (Amplification Of LCCs)**.** *Suppose that some code $C : \Sigma_1^n \to \Sigma_2^m$ has a local corrector D with correcting radius d, number of queries q, and soundness $s < 1/2$.*

Then for any odd $k \ge 1$, the code C has a local corrector, D', with correcting radius d, number of queries kq and soundness $s' = e^{-\frac{(1/2-s)^2}{2(1-s)}k}$. If D is a MOS corrector, so is D'.

In particular, if s is constant, then for any $s' < 1/2$, we have that D' has $O_s(q \log(1/s'))$ queries.

Next we show that any LCC with perfect completeness is also a VLTC. The local tests are simple: correct a random symbol and compare it to the symbol in the input. We need the LCC to have perfect completeness because we are not satisfied with our VLTCs only telling us when corruption is present, we want a close *estimate* of that corruption. So if there are no errors, the local test should never fail.

Lemma 3.12 (LCCs with Perfect Completeness are VLTCs)**.** *Suppose some code $C : \Sigma_1^n \to \Sigma_2^m$ is an LCC with perfect completeness, correcting radius d, soundness $\frac{1}{10m}$, and q correcting queries.*

Then C is also an VLTC with vicinity d, $q + 1$ queries, randomness $2\log(m) + O(1)$ and approximation factor 2.

Proof. Let D be the corrector for C and R be the randomness of D. The VLTC first chooses $k = 10m^2$ choices randomness for the decoders, $r_1, \ldots, r_k \in \{0,1\}^R$. For notation, denote $i_j = \lceil jm/k \rceil$. Then the tester V is defined by

$$V^w(j) = \begin{cases} 1 & D'^w(i_j, r_j) \neq w_{i_j} \\ 0 & D'^w(i_j, r_j) = w_{i_j}. \end{cases}.$$

See that V has $q + 1$ queries. Now we just need to show V will estimate the error within a 2 factor with high probability.

So consider any input $w \in \Sigma_2^m$ and any message $x \in \Sigma_1^n$ with $\Delta(w, C(x)) \le d$. We want to show that with very high probability we have

$$\frac{\Delta(w, C(x))}{2m} \le \Pr_{j \in [k]}[V^w(j) = 1] \le \frac{2\Delta(w, C(x))}{m}.$$

If $w = C(x)$, then since the corrector has perfect completeness, $\Pr_{j\in[k]}[V^w(j) = 1] = 0$, so the equation holds. Otherwise $\Delta(w, C(x)) \ge 1$, so we just need to show that the number of times the corrector fails to output the correct codeword symbol is at most $k\frac{\Delta(w,C(x))}{2m} \ge 5m$. So for $j \in [k]$ denote by F_j the event that $D'^w(i_j, r_j) \ne C(x)_{i_j}$.

See that the expectation of $\sum_{j\in[k]} F_j$ is $\mu \le \frac{k}{10m} \le m$. Then by a Chernoff bound

$$\Pr[\sum_{j\in[k]} F_j \ge 5m = (1 + 4)\mu] \le e^{-4^2 m/(2+4)} \le e^{-16m/6} < 2^{-m}.$$

So with probability at most 2^{-m} will $V^w(j)$ not give a 2 approximation of the corruption in w. Thus by a union bound, some choice of V must give a 2 approximation for every such w. □

All typical LCCs have perfect completeness, but some LCCs do not. It is unclear if all LCCs can be transformed into one with perfect completeness and a similar number of queries. In contrast, all MOSLCCs can be efficiently transformed into one with perfect completeness. This is because no corruption is a corruption pattern, so the correction failures on codewords is only a function of the index to correct and the randomness string. If number of randomness strings that fail on any input is very small, we just update our corrector to not use them.

Lemma 3.13 (MOSLCCs Have Perfect Completeness)**.** *Suppose some systematic code $C : \Sigma_1^n \to \Sigma_2^m$ is a MOSLCC with a MOS local corrector D using randomness R, with MOS correcting radius d, soundness $s_1 = \frac{1}{am}$ for $a > 1$, and q correcting queries.*

Then we also have that C is a MOSLCC with a MOS local corrector D' that uses randomness R, MOS correcting radius d, soundness $s_2 = \frac{1}{(a-1)m}$, q correcting queries, and has perfect completeness.

Proof. Call a randomness string $r \in \{0,1\}^R$ good for a corruption pattern $z \in \{0,1\}^m$ and index $i \in [m]$ if for all inputs $w \in \Sigma_2^m$ where z is difference from a codeword $C(x)$ we have that $D^w(i, r) = C(x)_i$. Then the assumption is that for any corruption pattern z with $wt(z) \le d$ and $i \in [m]$ that at least a $\frac{1}{am}$ fraction of randomness strings are good for z and i.

Notice that by a union bound at least $\frac{1}{a}$ of all randomness strings must be good for the zero corruption for all indexes $i \in [m]$. So for any randomness string that is not good for the zero error pattern for all indexes, we simply remove these from the possible randomness strings to get D'. The total number of randomness strings we remove is at most $2^R/a$. This decreases the total possible number of randomness strings for D' to $2^R\left(\frac{a-1}{a}\right)$, so the final randomness is $R - \log(a/(a-1)) \le R$.

Removing these randomness strings may hurt soundness for other inputs, but not too much. For any other corruption pattern $z \in \{0,1\}^m$ with $wt(z) \le d$ and $i \in [m]$ we know that at least $2^R\frac{am-1}{am}$ of the randomness strings are good for z and i. Then in worst case, all the strings removed were some of these good strings. So after removing these strings, at least $2^R\frac{(a-1)m-1}{am}$ of the good strings for this w and index i are left. The total fraction of remaining good strings then are

$$\begin{aligned} 2^R\frac{(a-1)m-1}{am}\frac{1}{2^R}\frac{a}{a-1} =& \frac{(a-1)m-1}{(a-1)m} \\ =& 1 - \frac{1}{(a-1)m}. \end{aligned}$$

So the probability that for any $x \in \Sigma_1^n$ and any $w \in \mathsf{Diff}(C(x), z)$ that $D^w(i, r) \ne C(x)_i$ is at most $\frac{1}{(a-1)m}$. □

Unfortunately, the vicinity local testers of Lemma 3.12 are not randomness efficient enough for us. We would need $\Omega(m^2)$ time to just enumerate through all of the tests. The problem is that we need to estimate the number of errors, and getting a good estimate becomes more difficult as the number of errors becomes smaller. If we only need to approximate the fraction of errors for a large fraction of errors, this can be done randmness efficiently.

Lemma 3.14 (Test for Large Distance From LCC)**.** *Suppose that $C : \Sigma_1^n \to \Sigma_2^m$ is an LCC with correcting radius d_1, soundness $s = \frac{1}{10m}$ and q queries.*

Then for $d_2 \leq d_1$, there is a tester V which takes as input a $w \in \Sigma_2^m$ and $r \in \{0,1\}^R$ and outputs a single bit, denoted $V^w(r)$, such that:

Randomness Efficient: *The randomness R is $\log(\frac{m^2}{d_2}) + O(1)$.*

Locality: *For any $w \in \Sigma_2^m$ and $r \in \{0,1\}^R$, the function $V^w(r)$ only queries w in $q+1$ places.*

Completeness: *For any $w \in \Sigma_2^m$ with $\Delta(w, C) \leq \frac{d_2}{2}$ we have that*

$$\Pr_{r \in \{0,1\}^R}[V^w(r) = 1] \leq \frac{3d_2}{4m}.$$

Soundness: *For any $w \in \Sigma_2^m$ with $\Delta(w, C) > d_2$ and $\Delta(w, C) \leq d_1$ we have that*

$$\Pr_{r \in \{0,1\}^R}[V^w(r) = 1] > \frac{3d_2}{4m}.$$

Proof. Let D be the corrector for C. The VLTC just chooses $k = 100\frac{m^2}{d_2}$ choices randomness for the decoders, $r_1, \ldots, r_k \in \{0,1\}^R$. For notation, denote $i_j = \lceil jm/k \rceil$. Then the tester V is defined by

$$V^w(j) = \begin{cases} 1 & D'^w(i_j, r_j) \neq w_{i_j} \\ 0 & D'^w(i_j, r_j) = w_{i_j}. \end{cases}.$$

See that V has $q+1$ queries.

Now we want to bound the probability too many choices of r_j have $D'^w(i_j, r_j)$ correct incorrectly. Choose any $w \in \Sigma_2^m$ with an $x \in \Sigma_1^n$ such that $\Delta(w, C(x)) \leq d_1$. Let F_j be the event that $D'^w(i_j, r_j) \neq C(x)_{i_j}$. By the soundness of D, the expectation of $\sum_{j \in [k]} F_j$ is at most

$$\mu = ks = \frac{10m}{d_2}.$$

We want to show that it is very unlikely that $\sum_{j \in [k]} F_j$ is greater than $k\frac{d_2}{4m} = 25m$. Set δ such that $25m = (1+\delta)\mu$. See that since $\mu \leq 10m$ we have that $\delta \geq 1.5$ and since $\frac{\delta}{2+\delta}$ is monotone in δ, we also have that $\frac{\delta}{2+\delta} \geq \frac{1.5}{2+1.5}$. We also have that $\delta\mu \geq 15m$. Then by a Chernoff bound, we have that

$$\begin{aligned} \Pr[\sum_{j \in [k]} F_j > k\frac{d_2}{4m}] \leq & e^{-\delta^2 m/(2+\delta)} \\ \leq & e^{-\frac{1.5}{2+1.5}15m} \\ < & e^{-m} \\ < & 2^{-m}. \end{aligned}$$

So in particular, their exists choice of r so that for every $w \in \Sigma_2^m$ with $\Delta(w, C) \leq d_1$ we have $\sum_{j \in [k]} F_j \leq k\frac{d_2}{4m}$. Choose such choices of r.

With this choice of r, the discrepancy between the failure probability of V^w and the actual fraction of corruption in V^w is at most $\frac{d_2}{4m}$. Thus if for some $w \in \Sigma_2^m$ we have both $\Delta(w, C) > d_2$ and $\Delta(w, C) \leq d_1$ then

$$\Pr_{r \in \{0,1\}^R}[V^w(r) = 1] \geq \frac{\Delta(w, C)}{m} - \frac{d_2}{4m} > \frac{3d_2}{4m}.$$

Similarly, if for some $w \in \Sigma_2^m$ we have $\Delta(w, C) \leq \frac{d_2}{2}$, then

$$\Pr_{r \in \{0,1\}^R}[V^w(r) = 1] \leq \frac{\Delta(w, C)}{m} + \frac{d_2}{4m} \leq \frac{3d_2}{4m}.$$

□

If one only wants a randomness efficient test that fails with high probability when the number of errors is high (but still within the correcting radius), then one can get this for any LCC. The difficulty is only for accurately estimating the amount of corruption when the corruption is small. But we can do this randomness efficiently for MOSLCCs. To do this, we union bound over m^d patterns of corruption if our local corrector has message oblivious soundness (MOS). This allows us to more randomness efficiently perform local testing, at least within the correcting radius.

Theorem 3.15 (MOSLCCs are VLTCs)**.** *Suppose some systematic code $C : \Sigma_1^n \to \Sigma_2^m$ is a MOSLCC with MOS correcting radius d, soundness $\frac{1}{11m}$, and q correcting queries.*

Then C is also an VLTC with vicinity d, $q+1$ queries, randomness $\log(m) + \log(\log(m)) + O(1)$ and approximation factor 2.

Proof. First we use Lemma 3.13 to transform our corrector to one with perfect completeness, MOS correcting radius d, q queries, and soundness $\frac{1}{10m}$. Let D be such a MOS corrector for C with randomness R.

The VLTC just chooses $k = 10m\log(m)$ choices of randomness for the decoders, $r_1, \dots, r_k \in \{0,1\}^R$. For notation, denote $i_j = \lceil jm/k \rceil$. Then the tester V is defined by

$$V^w(j) = \begin{cases} 1 & D'^w(i_j, r_j) \neq w_{i_j} \\ 0 & D'^w(i_j, r_j) = w_{i_j}. \end{cases}.$$

See that V has $q+1$ queries. Now we just need to show V will estimate the error within a 2 factor with high probability.

To show this, we first consider a fixed error pattern weight and argue V will give a good approximation for all error patterns of that weight with probability greater than $1 - \frac{1}{d}$. Then it won't fail on any error pattern with some positive probability, thus some choice of randomness will always give a good approximation.

For any $w \in \Sigma_2^m$, we say that V succeeds on w if $\frac{\Delta(w,C)k}{2m} \leq \sum_{j\in[k]} V(j,w) \leq \frac{2\Delta(w,C)k}{m}$, and V fails on w otherwise. We say that V fails on corruption $z \in \{0,1\}^m$ if there is any $x \in \Sigma_1^n$ and $w \in \mathsf{Diff}(C(x), z)$ that V fails on w.

So choose a weight $d' \leq d$ and take any $z \in \{0,1\}^m$ with $wt(z) = d'$. All we need to show is that it is unlikely more than $\frac{d'k}{m2}$ of the symbols will be decoded incorrectly. If this is true, than the number of tests that fail is at least $\frac{d'k}{m2}$, and at most $\frac{3d'k}{m2}$. To be more formal, for $j \in [k]$ let F_j be the failure event that

$$\exists x \in \Sigma_1^n, w \in \mathsf{Diff}(C(x), z) : D^w(i_j, r_j) \neq C(x)_{i_j}$$

Then we want to show that

$$\Pr\left[\sum_{j\in[k]} F_j \geq \frac{d'k}{2m}\right] < \frac{1}{m}.$$

To do this, we use Chernoff bounds. See that the expectation of $\sum_{j\in[k]} F_j$ is at most

$$\mu \leq \frac{k}{10m}.$$

Then by a Chernoff bound, we have that

$$\begin{aligned}\Pr\left[\sum_{j\in[k]} F_j \geq \frac{d'k}{2m}\right] &\leq e^{-\frac{4}{2+4}\frac{4d'k}{10m}} \\ &\leq e^{-\frac{160d'm\log(m)}{60m}} \\ &\leq m^{-2.5d'} \\ &< m^{-d'}/m.\end{aligned}$$

Now if $\sum_{j\in[k]} F_j < \frac{d'k}{2m}$, then for any $x \in \Sigma_1^n$ and $w \in \mathsf{Diff}(C(x), z)$, we have that $\sum_{j\in[k]} V^w(j) \leq \frac{2d'k}{m}$ and $\sum_{j\in[k]} [V^w(j)] \geq \frac{d'k}{2m}$. Thus the probability that V fails on z is less then $m^{-d'}/m$.

So by a union bound, the probability that V fails on any corruption $z \in \{0,1\}^m$ with $wt(z) = d'$ is less than $\frac{1}{m}$. That is, the probability that V fails on any w with $\Delta(w, C) = d'$ is less then $\frac{1}{m}$. Then by a union bound, the probability that V fails on any w with $\Delta(w, C) \leq d$ is less then 1. Thus there is a V that does not fail on any w with $\Delta(w, C) \leq d$. That V is a vicinity local tester for C with $q + 1$ queries, randomness $\log(k) = \log(m) + \log(\log(m)) + \log(10)$, and approximation factor 2. □

So we have shown that message oblivious soundness allows us to perform vicinity local testing more randomness efficiently. One may wonder whether MOSLCCs are reasonable to expect. Many LCCs already have the MOS property and their proofs show soundness by showing that corruption is seen rarely. Here we prove that every typical LCC is a MOSLCC.

The idea is that a typical LCC with q queries that is $2q$ smooth on an input with at most $\frac{m}{10q}$ corruptions won't even see the corruption most of the time when correcting. Then since a typical LCC has perfect completeness, if no corruption is seen, it must correct correctly.

Lemma 3.16 (Typical LCCs are MOSLCCs)**.** *Suppose $C : \Sigma_1^n \to \Sigma_2^m$ is a typical LCC with q queries and smoothness β. Then for any s, there is a MOS corrector for C with MOS correcting radius $d = \frac{sm}{\beta}$, soundness s, and q queries. Alternatively, for any s, there is a MOS corrector for C with MOS correcting radius $d = \frac{m}{3\beta}$, soundness s, and $O(q \log(1/s))$ queries.*

Proof. The idea is that if there are only $d = \frac{sm}{\beta}$ corruptions, the probability that any corruption is seen is at most s. By perfect completeness, if no corruption is seen, it must correct correctly.

Let D be the corrector of C and assume D uses randomness R. Choose any $z \in \{0,1\}^m$ with $wt(z) \leq d$ and any $i \in [m]$. Now choose any $j \in [m]$ where $z_j = 1$. For notation, let $y \in \Sigma_2^m$ be an arbitrary string. Then

$$\Pr_{r \in \{0,1\}^R}[D^y(i, r) \text{ queries symbol } j \text{ of } y] \leq \frac{\beta}{m}.$$

So the total probability that any index that is one in z is queried is at most

$$\frac{\beta}{m} d = s.$$

Since D is non-adaptive, if on a given choice of randomness $r \in \{0,1\}^R$ and index $i \in [m]$ we have that $D^y(i, r)$ only queries indexes that are 0 in z, then for any $x \in \{0,1\}$. and $w \in \mathsf{Diff}(C(x), z)$ we must have $D^w(i, r)$ only queries symbols that agree with $C(x)$. Since D has perfect completeness, for such i and r we must have that $D^w(i, r) = C(x)_i$. Thus for any $i \in [m]$ the probability over $r \in \{0,1\}^R$ that any message $x \in \Sigma_1^n$ and corrupted codeword $w \in \mathsf{Diff}(C(x), z)$ has $D^w(i, r) \neq C(x)_i$ is at most s.

Therefore D is also a MOS corrector with soundness s and MOS correcting radius d.

Alternatively, D is a MOS corrector with MOS correcting radius $\frac{m}{3\beta}$ and soundness $\frac{1}{3}$. Then by using the amplification of Lemma 3.11, C also has a MOS corrector with correcting radius $\frac{m}{3\beta}$, soundness s, and $O(q \log(1/s))$ queries. □

Unfortunately we are only able to show typical LCCs have a MOS corrector with MOS correcting radius $O(\frac{m}{q})$, which can be small if the LCC uses many queries. So the fact that typical LCCs are MOSLCCs doesn't immediately give typical LCCs all the nice properties of the MOSLCC with the same correcting radius. This is an issue if we want to correct $\Omega(m)$ errors. We will show how to overcome this later.

4 Deterministic Decoder Construction

In this section we explain our deterministic decoder in more detail. Our decoder centers around codeword improvers and improving sets. In this section we define codeword improvers and improving sets and show how to use them to deterministically decode. In Section 5 we show how to find the necessary codeword improvers and improving sets, and provide proofs for the claims in this section.

4.1 Codeword Improvers

To decode an input w, our approach will be to iteratively decrease the number of errors in a codeword until no errors are remaining. Our first goal will be to find a function, $I : \Sigma_2^m \to \Sigma_2^m$, that will decrease the number of errors in an input from $\Omega(m)$ to $O(m^{1-\epsilon})$. Since this is a less ambitious goal than full correction, it will be easier to find randomness for a local corrector that can do this.

We say that a function I is a codeword improver for a corrupted codeword w if $I(w)$ outputs a string closer to that codeword. Specifically:

Definition 4.1 (Codeword Improver). *Let there be a code $C : \Sigma_1^n \to \Sigma_2^m$ with distance greater than $2d_1$. Let $x \in \Sigma_1^n$ and $w \in \Sigma_2^m$ with $\Delta(w, C(x)) \leq d_1$. Then say that a function $I : \Sigma_2^m \to \Sigma_2^m$ improves w to distance $d_2 \leq d_1$ with respect to C if $\Delta(I(w), C(x)) \leq d_2$.*

Our codeword improvers will just be local correctors with appropriately chosen randomness strings hard wired. We will show how to choose our codeword improvers in Section 5.1. Importantly, our codeword improvers will use few queries, since they are just calls to a local corrector.

Definition 4.2 (A Few Query Function). *For any function $f : \Sigma_1^n \to \Sigma_2^m$, we say that f is q query if for any $i \in [m]$ and $w \in \Sigma_1^n$ we have that $f(w)_i$ can be computed with only q queries to w.*

Then our goal is to start with an input $w \in \Sigma_2^m$ with $x \in \Sigma_1^n$ such that $\Delta(w, C(x)) \leq d$. Then we will find a q query codeword improver, I, that will improve w to distance ηd. Think of η as $d^{-1/\ell}$ for a large constant ℓ. Then we can set $w' = I(w)$ and find a new q codeword improver that will improve w' to distance $\eta^2 d$. Then if we repeat this process ℓ times, then we can correct w all the way to its nearest codeword. If each codeword improver only needs q queries, the final codeword corrector only needs q^ℓ queries. See Fig. 1 for a figure with the number of errors at each stage versus the number of queries to that stage.

To make finding codeword improvers efficient, we need a small list of candidate codeword improvers to search through. So we define an improving set. An improving set is a short list of candidate codeword improvers such that for any input that is not too corrupted, in expectation, the candidate codeword improvers will improve it.

Definition 4.3 (Improving Set). *Let there be a code $C : \Sigma_1^n \to \Sigma_2^m$ with distance greater than $2d_1$. Let $\mathcal{I}$ be a set of functions from Σ_2^m to Σ_2^m. Then we say that $\mathcal{I}$ is a d_1 to d_2 improving set for C if for all $w \in \Sigma_2^m$ such that for some $x \in \Sigma_1^n$ we have that $\Delta(w, C(x)) = d_1$, then*

$$\mathop{\mathbb{E}}_{I \in \mathcal{I}}[\Delta(I(w), C(x))] \leq d_2.$$

If every $I \in \mathcal{I}$ is a q query function, we say that $\mathcal{I}$ is a q query, d_1 to d_2 improving set for C. We say that $\mathcal{I}$ is time T space S uniform if for each $I \in \mathcal{I}$ and each $i \in [m]$, the function computing $I(w)_i$ is time T space S uniform.

We say that $\mathcal{I}$ is a below d_1, factor η improving set for C if for all $d \leq d_1$ we have that $\mathcal{I}$ is a d to ηd improving set for C.

A straightforward consequence of having a d_1 to d_2 improving set for a code is that for any input w with distance d_1 from a code, one of the functions $I \in \mathcal{I}$ improves w to distance d_2 with respect to C.

Lemma 4.4 (Improving Sets Contain Codeword Improvers). *Let $C : \Sigma_1^n \to \Sigma_2^m$ be a code and $\mathcal{I}$ be a d_1 to d_2 improving set for C. Then for any $w \in \Sigma_2^m$ with $\Delta(w, C) = d_1$, there is an $I \in \mathcal{I}$ such that I improves w to distance d_2 with respect to C.*

Similarly, if $\mathcal{I}$ is a below d, factor η improving set for C, then for any $w \in \Sigma_2^m$ with $\Delta(w, C) \leq d$ there is an $I \in \mathcal{I}$ such that I improves w to distance $\eta\Delta(w, C)$ with respect to C.

Proof. Let $x \in \Sigma_1^n$ be such that $\Delta(w, C(x)) = d_1$. Then suppose that for all $I \in \mathcal{I}$ we have that $\Delta(I(w), C(x)) > d_2$, then we would have that $\mathbb{E}_{I \in \mathcal{I}}[\Delta(I(w), C(x))] > d_2$. But this contradicts the definition of improving set, so some $I \in \mathcal{I}$ must have that $\Delta(I(w), C(x)) \leq d_2$.

A similar argument holds for a below d, factor η improving set. □

4.2 Finding Codeword Improvers With VLTCs

In this section we describe the deterministic decoder for a code assuming it has a small improving set and a randomness efficient vicinity local tester.

If our code both has an improving set and is locally testable, we can find codeword improvers for a given input. All we have to do is iterate through each candidate improver in an improving set, then use the local tests to see if it is a good enough improver. This works because our local tests don't just tell us if there is corruption, it gives a close *estimate* of the amount of corruption. Thus if we can show that there is a short list of candidate codeword improvers, we can efficiently search through them to find an actual codeword improver.

Lemma 4.5 (VLTCs Can Select From Improving Sets)**.** *Let $C : \Sigma_1^n \to \Sigma_2^m$ be a code that has a set of functions $\mathcal{I}$ that is a q_c query, below d_1, factor η improving set for C. Let C also be a VLTC with q_t testing queries, R testing randomness, approximation factor α and vicinity d_0. Suppose that $3d_1 \leq d_0$.*

Then there is a non-uniform algorithm that runs in time $O\left((m + 2^R q_t) q_c |\mathcal{I}| \log(|\Sigma_2|)\right)$ and space $O(R + (q_t + q_c)\log(|\Sigma_2|) + \log(m) + \log(|\mathcal{I}|))$ and takes as input a $w \in \Sigma_2^m$ with $\Delta(w, C) \leq d_1$ and outputs a $O(\log(|\mathcal{I}|))$ bit index of some $I \in \mathcal{I}$ that improves w to distance $\alpha^2 \eta \Delta(w, C)$ with respect to C.

If the improving set is uniform and computable in time T_c and space S_c, and the tester is uniform and computable in time T_t and space S_t, then the algorithm is uniform and runs in time $O\left((2^R T_t + m T_c + 2^R q_t T_c)|\mathcal{I}|\right)$ and space $O(R + S_t + S_c + \log(m) + \log(|\mathcal{I}|))$.

So if we have a code that is both a randomness efficient VLTC and has a small, below d, factor η improving set, then by applying Lemma 4.5 once we get can get an improver, I_1, that improves a d corruption input to distance ηd. Then applying it again we get another improver I_2 such that $I_2 \circ I_1$ improves the input to distance $\eta^2 d$. After applying this many times, we get a series of improvers such that when they are all composed together they completely correct the codeword.

Lemma 4.6 (Deterministic Correctors from Improving Sets and VLTC)**.** *Let $C : \Sigma_1^n \to \Sigma_2^m$ be a code that has a set of functions $\mathcal{I}$ that are q_c query, below d_1, factor η improving sets for C. Let C also be a VLTC with q_t testing queries, R testing randomness, approximation factor α and vicinity d_0. Suppose that $3d_1 \leq d_0$ and $\eta\alpha^2 < 1$ and define $\ell = \lceil \frac{\log(d_1+1)}{\log(1/(\eta\alpha^2))} \rceil$.*

Then the code C has a deterministic non-uniform corrector with correcting radius d_1 running in time

$$O(\ell(2^R q_t + m)|\mathcal{I}| q_c^\ell \log(|\Sigma_2|))$$

and space

$$O(\ell \log(|\mathcal{I}|) + (\ell q_c + q_t)\log(|\Sigma_2|) + \log(m) + R).$$

If the improving set and the tester are time T space S uniform, then the corrector is uniform and runs in time at most $O(\ell(2^R q_t + m) T |\mathcal{I}| q_c^\ell)$ and the space at most $O(\ell \min\{S, q_c \log(|\Sigma_2|)\} + \ell \log(|\mathcal{I}|) + \log(m) + R + S)$. More generally, C has a uniform, deterministic algorithm, g, which runs in time at most $O(\ell(2^R q_t + m) T |\mathcal{I}| q_c^\ell)$ and the space at most $O(\ell \min\{S, q_c \log(|\Sigma_2|)\} + \ell \log(|\mathcal{I}|) + \log(m) + R + S)$ and outputs an $O(\ell \log(|\mathcal{I}|))$ bit description of a deterministic, q_c^ℓ query function f such that if for some $x \in \Sigma_1^n$ and $y \in \Sigma_2^m$ we have that $\Delta(y, C(x)) \leq d_1$, then we have that $g(x)$ outputs an f such that $f(y) = C(x)$. Further, for each $i \in [m]$, $f(y)_i$ is computable in time at most $O(T q_c^\ell)$ and the space at most $O(\ell \min\{S, q_c \log(|\Sigma_2|)\} + S)$.

Many codes are randomness efficient VLTCs, in particular MOSLCCs are (see Theorem 3.15). So the main challenge is to find an improving set. We do this in Section 5.

4.3 Deterministic Decoder Pseudocode

Now we give pseudocode for our time and space efficient deterministic decoder. We start with the algorithm in the simple case where we are given a MOSLCC and VLTC to start with. When extending this to the the more general case, the algorithm only changes by constructing the local tester from the local corrector. When extending this to the uniform case, we only need to make the improving sets and local testers explicit.

4.3.1 Decoder for MOSLCCs and VLTCs

Let $C : \Sigma_1^n \to \Sigma_2^m$ be a code with distance $2d_0$ and let C be a MOSLCC with local corrector D with MOS correcting radius $d_1 < \frac{d_0}{3}$, q_c queries, soundness $\frac{1}{11m}$ and R_c bits of randomness. If D does not already have the soundness, we may first need to amplify it with Lemma 3.11. Let C also be a VLTC with local tester V, vicinity d_0, randomness R_t, approximation factor α, and number of queries q_t. Let ℓ be some integer for the number of iterations we are willing to run.

Since our decoder is non-uniform, it will have a preprocessing step that will hardwire our advice for our decoder, and then we will run that decoder to decode an input. We will think of the preprocessing step as outputting a list of candidate codeword improvers. For some $k = O(\alpha^2 m^{1/\ell} \log(m))$, it outputs the improving set, $\mathcal{I} = I_1, \ldots, I_k : \Sigma_2^m \to \Sigma_2^m$. For each of these codeword improvers, they can compute any output symbol using only q queries to their input.

Preprocessing: For every $j \in [k]$ and $i \in [m]$, choose a random string, $r_{i,j} \in \{0,1\}^{R_c}$, and define $I_j : \Sigma_2^m \to \Sigma_2^m$ by

$$(I_j(w))_i = D^w(i, r_{i,j}).$$

That is, each I_j just corrects every symbol with independent uniform randomness. Such a set of candidate codeword improvers $I_1, \ldots, I_k$ for sufficiently large $k = O(\alpha^2 m^{1/\ell} \log(m))$ will be an improving set with high probability.

We emphasize that the choices of randomness $r_{i,j} \in \{0,1\}^{R_c}$ will be hard coded into the decoder advice, so it does not need to be in the state of the decoder.

Decoder: Now our final corrector consists of 2 subroutines and a loop. The first subroutine takes a list of candidate codeword improvers from an improving set and evaluates their composition efficiently. This is what our final corrector is, several carefully chosen candidate codeword improvers composed together. The second is a tester that takes composed candidate codeword improvers and checks how much they improve an input. Finally the loop finds candidate codeword improvers to compose together.

Query Improvers This function takes a list of candidate codeword improvers, described by their indexes $(j_1, \ldots, j_a) \in [k]^a$, and an input $w \in \Sigma^m$ and returns individual symbols of $I_{j_a}(\ldots(I_{j_1}(w)))$. This algorithm needs to compute the corrector, D, recursively. We don't know much about D except once the index i and randomness r is fixed, $D^w(i, r)$ can be computed by a time q space $|\Sigma_2|^q$ decoder.

So in the psuedocode, we interpet D to be the program that takes as input a program f, an index i and randomness r and outputs $D^f(i, r)$ where $D^f(i, r)$ runs $D(i, r)$ with the oracle queries answered by some program f. We will use the convention that a program with a small number of the arguments filled is a program that takes as input the remaining arguments. See Algorithm 1 for pseudocode.

Algorithm 1 Query Improvers (D is the local corrector)

```
procedure QueryImprov((j_1, ..., j_a), w, i)          ▷ Computes I_{j_a}(...(I_{j_1}(w)))_i
    if a = 0 then            ▷ If we are not given any candidate codeword improvers,
        return w_i                                        ▷ return w unmodified.
    else
        return D^{QueryImprov((j_1,...,j_{a-1}),w)}(i, r_{i,j})   ▷ Run D, but every time it queries the input,
    end if                                 ▷ answer it using a call to EvalImprov.
end procedure
```

Test Current Output This subroutine takes in a list of candidate codeword improvers, described by their indexes $(j_1, \ldots, j_a) \in [k]^a$, and a starting word $w \in \Sigma_2^m$ such that for some $x \in \Sigma_1^n$ we have that $\Delta(w, C(x)) \leq d_1$. Then for $y = I_{j_a}(\ldots(I_{j_1}(w)))$ we want to output an α approximation of $\Delta(y, C(x))$.

This is done in two steps. First, y is compared to w. If they are too different, then we know that y cannot be close to $C(x)$, so we just output a large value. Otherwise we run V with every choice of randomness. Similar to D, define $V^f(r)$ to be the program that takes as input a program f and a choice of randomness r and runs V with oracle queries computed by f. See Algorithm 2 for pseudocode.

Algorithm 2 Approximate Corruption (V is the vicinity local tester)

```
procedure ApxCor(J = (j_1, ..., j_a), w)      ▷ Approximates Δ(y, C(x)) if Δ(y, C(x)) ≤ d_1.
                                              ▷ Otherwise either outputs m or approximates Δ(y, C(x)).
    b ← 0
    for all i ∈ [m] do
        if QueryImprov(J, w, i) ≠ w_i then                                   ▷ Compute Δ(y, w)
            b ← b + 1
        end if
    end for
    if b ≥ 2d_1 then                                         ▷ If y is too far from w, return m
        return m
    end if
    b ← 0
    for all r ∈ {0,1}^{R_t} do                          ▷ Run V with every choice of randomness
        if V^{QueryImprov(J,w)}(r) = 1 then
            b ← b + 1                                         ▷ Count number of failed tests.
        end if
    end for
    return αbm / 2^{R_t}                                ▷ Return approximation of distance
end procedure
```

Main Loop Finally our main algorithm will take as input a $w \in \Sigma_2^m$ such that for some $x \in \Sigma_1^n$ we have that $\Delta(w, C(x)) \leq d_1$ and prints x. This algorithm builds a list of candidate codeword improvers, each of which improves the last by a large a factor of $m^{1/\ell}$. In the end we will have a list of candidate codeword improvers that together correct w. Finally, we will output the symbols in w that correspond to symbols in x. For simplicity, we assume the first n symbols of $C(x)$ are the symbols of x. See Algorithm 3 for pseudocode.

Algorithm 3 Time and Space Efficient Deterministic Decoding

```
procedure Decode(w)                                          ▷ Decodes w with Δ(w, C) ≤ d_1.
    J ← ()                                              ▷ Start with no codeword improvement.
    for all i = ℓ − 1, ..., 0 do                                           ▷ Improve ℓ times.
        for all j ∈ [k] do                                     ▷ Try all k candidate improvers.
            J′ ← J ∘ j                 ▷ Add candidate improver to a temporary list of improvers.
            error ← ApxCor(J′, w)                            ▷ Check how much it improved w.
            if error < m^{i/ℓ}/2 then                               ▷ If it improved w enough.
                J ← J′                  ▷ Keep j on the list of improvers and continue to next i.
                break
            end if
        end for
    end for
    for all i ∈ [n] do                                     ▷ Print x using the improvers J.
        Print QueryImprov(J, w, i)
    end for
end procedure
```

4.3.2 Decoders for Typical LCCs

The algorithm for a typical LCC works in the same way. The only necessary change to the pseudocode is that the vicinity local tester, V, is not provided to us. However, a randomness efficient vicinity local tester always exists for a MOSLCC (see Theorem 3.15). This VLTC just corrects random symbols and compares them to the input. Finding the appropriate random strings for the correction will take more preprocessing time, but this can be hard coded into the decoder just like the candidate codeword improvers were.

For a typical LCC, the decoder is exactly the same as a MOSLCC, the only difference is the analysis. So to be explicit, our only change is to add the following to the following preprocessing step.

Preprocessing: For some sufficiently large $k' = O(m\log(m) + m^{1+1/\ell})$ every $j \in [k']$, choose a random string, $r'_j \in \{0,1\}^{R_c}$, and define $V^w(j)$ by

$$V^w(j) = \begin{cases} 1 & D^w(\lceil jm/k' \rceil, r'_j) \neq w_{\lceil jm/k' \rceil} \\ 0 & \text{otherwise} \end{cases}.$$

That is, each $V^w(j)$ just corrects a symbol with independent randomness and checks if it matches w.

We emphasize that the choices of randomness $r'_j \in \{0,1\}^{R_c}$ will be hard coded into the decoder itself, thus don't need to be in the state of the algorithm to be used.

Remark (Differences Between Our Pseudocode and Our Proof)**.** *We also note that the extra factor of $m^{1/\ell}$ in k' is only required to handle inputs with very large corruption: higher than $\Omega(m/q)$. That is, we only need to use the full k' tests in the first iteration of the decode procedure (Algorithm 3). After that, only $O(m\log(m))$ local tests are necessary.*

The protocol used in our proof uses this slightly modified algorithm because the analysis is simpler, but both work.

4.3.3 Uniform Decoder for Reed-Muller Codes

The uniform decoder for Reed-Muller codes uses the same decoding strategy except that the improving set and local tester is given by a curve sampler. Specifically, we can test how close a function is to a low degree polynomial by counting how many curves in the sampler there are such that the function restricted to that curve is a low degree polynomial. Each candidate codeword improver takes a curve in the sampler and runs Reed-Solomon decoding on that curve composed with the function. The deterministic decoding algorithm is the same, except that $\mathcal{I}$ and V are explicit and don't need to be precomputed or given as advice. The decoder for lifted Reed-Solomon codes is similar.

5 Deterministic Decoder Analysis

In this section we present the analysis of our deterministic decoders. We start by constructing improving sets for LCCs and MOSLCCs using the probabilistic method. Then we will prove that improving sets and VLTCs give time and space efficient deterministic decoders. Then we give deterministic decoders for MOSLCCs.

Next we show that typical LCCs have deterministic decoders. We note that this does not immediately follow from Lemma 3.16, which shows that typical LCCs are MOSLCCs, because the MOS correcting radius may be much smaller than the original correcting radius. More details are in Section 5.4.

Finally, we show that any code with good enough improving sets have efficient deterministic decoders. This shows that to make our construction explicit it suffices to only make the necessary improving sets.

5.1 Improving Sets For LCCs and MOSLCCs

Now we give improving sets for LCCs and MOSLCCs. The size of the improving set is important for the time of the decoder. In this section, we will show why improving sets that don't completely eliminate the corruption are smaller. This is why an iterative decoding approach makes it faster to find a good codeword improver. The idea is that we need less randomness to reduce the number of errors by a factor $m^{1/\ell}$ fraction versus reducing it by a factor of m.

Specifically, suppose we have a local corrector which only fails to correct a symbol with probability $1/m$. Then if we run the corrector m times for each of the m symbols, we only expect it to fail to correct m times. That is, for $j \in [m]$ and $i \in [m]$, if we choose an independent randomness $r_{i,j}$, we expect only m of the $r_{i,j}$ to fail to correct symbol i. Actually, from a Chernoff bound, the probability we fail to correct more than $100m$ times is less than 2^{-m}. Since there are only 2^m possible inputs, some choice of randomness must not fail more than $100m$ times over *every* input.

Now we want to find a $j \in [m]$ such that $r_{i,j}$ fails to correct symbol i for at most ηd choices of $i \in [m]$. If we want to remove all the errors right now, so $\eta = 0$, then it could be for some $w \in \Sigma_2^m$ that each of the first $100m$ choices of j, one of the $r_{i,j}$ fails to correct symbol i. Thus we would need to check $100m$ choices of j, and each j could take time m to check, so the time would be $\Omega(m^2)$. However, if we only wish to get the number of corruptions down to ηm for $\eta = m^{-1/\ell}$, then there can only be $\frac{100m}{\eta m} = 100m^{1/\ell}$ choices of j that have more than ηm corruptions. So only at most $m^{1/\ell}$ choices of j need to be checked before we find one that improves a given input.

Then our first codeword improver would be the local corrector with the randomness from one of these $100m^{1/\ell}$ first choices of j hard coded into it. So we only need $k = O(m^{1/\ell})$ choices of randomness to reduce an input string with $O(m)$ corruption to a string with $O(m^{1-1/\ell})$ corruption.

Lemma 5.1 (LCCs Give Codeword Improving Sets)**.** *Let $C : \Sigma_1^n \to \Sigma_2^m$ be an LCC and D be a local corrector for C with correcting radius d_1, randomness R, soundness $\frac{1}{2m}$ and q queries. Let d_2 be some distance with $d_2 \leq d_1$.*

Then for some $k = O\left(\frac{m}{d_2}\right)$ there exists a k element set of functions $\mathcal{I}$ such that for all $d \leq d_1$ we have that $\mathcal{I}$ is a q query d to d_2 improving set for C.

Proof. The idea is to just use D with random seeds for the improving set $\mathcal{I}$. With high probability the decoders fail extremely rarely, so rarely that for each not too corrupted codeword, in expectation the functions in the improving set must fail to correct at most d_2 times.

Let $k = \frac{10m}{d_2}$. Then for each $j \in [k]$ and $i \in [m]$, we choose random $r_{i,j} \in \{0,1\}^R$ and define I_j by $(I_j(w))_i = D^w(i, r_{i,j})$. Then $\mathcal{I}$ will be the set $\{I_j : j \in [m]\}$. We want to show that with positive probability, this construction works. Specifically, we want to show that with positive probability for any $x \in \Sigma_1^n$ and $w \in \Sigma_2^m$ with $\Delta(w, C(x)) \leq d_1$ we have that

$$\sum_{j \in [k]} \Delta(I_j(w), C(x)) \leq 10m.$$

We do this with a Chernoff bound. By the definition of an LCC, see that the expectation of the sum is

$$\mu \leq \frac{mk}{2m} = k/2.$$

So by the Chernoff bound, the probability that sum is more than $9m$ if $k \leq 10m$ is at most

$$e^{-0.8^2(5m/(2+0.8)} \leq e^{-m} < 2^{-m}.$$

This is more than the total number of potential $w \in \Sigma_2^m$. So in particular for any $k \leq 10m$ there exists some choice of $I_1, \ldots, I_k$ such that for all $x \in \Sigma_1^n$ and $w \in \Sigma_2^m$ with $\Delta(w, C(x)) \leq d_1$ we have that

$$\sum_{j \in [k]} \Delta(I_j(w), C(x)) \leq 9m.$$

Now we choose such $\mathcal{I}$.

For every $j \in [k]$, by definition of each I_j, for any $i \in [m]$, it only runs D once to calculate $(I_j(w))_i$, thus only every queries w at q places. Finally by choice of $\mathcal{I}$ for any $w \in \Sigma_2^m$ and $x \in \Sigma_1^n$ with $\Delta(w, C(x)) \leq d_1$ we have that

$$\mathop{\mathbb{E}}_{j \in [k]}[\Delta(I_j(w), C(x))] \leq \frac{9m}{k} = (9/10)d_2 < d_2.$$

□

The issue with this result is that it only can take an input with $O(m)$ corruptions down to ηm corruptions with $O(1/\eta)$ candidate codeword improvers. We need to keep going to get ηm corruptions down to $\eta^2 m$ corruptions with only $O(1/\eta)$ candidate codeword improvers in the improving set, but the prior argument would require $O(1/\eta^2)$ candidate codeword improvers in the improving set.

To improve this result, we need the LCC to have a stronger property we call "message oblivious soundness" (MOS) (see Definition 3.10). LCCs with MOS have the following stronger result. For any distances d_1 and d_2 within the correcting radius, there is a list of $k = O\left(\frac{d_1 \log(m)}{d_2}\right)$ candidate codeword improvers such that for any input w with $\Delta(w, C) \le d_1$ one of the candidate codeword improvers improves w to distance d_2. The analysis is similar, but instead of saying some randomness must work for every input, we say some randomness must work for every pattern of corruption.

Lemma 5.2 (MOSLCCs Give Codeword Improvers)**.** *Let $C : \Sigma_1^n \to \Sigma_2^m$ be a MOSLCC and D be a local MOS corrector for C with MOS correcting radius d_1, randomness R, soundness $\frac{1}{2m}$, and q queries. Let $\eta \in (0,1)$ be some function of n.*

Then for some $k = O\left(\frac{\log(m)}{\eta}\right)$ there exists a k element set of functions $\mathcal{I}$ that is a q query, below d_1, factor η improving set for C.

Proof. There are only $O(m^d)$ error patterns $z \in \{0,1\}^m$ with $wt(z) \le d$. The idea is to just use D with random seeds for the candidate improvers $I_1, \dots, I_k$ in the improving set $\mathcal{I}$. With high probability the decoders fail extremely rarely, so rarely that for every error pattern, one candidate codeword improver must fail less than ηd times. We do this argument over the space of error patterns, not corrupted codewords, since for small d, this space is much smaller. Thus we need fewer candidate codeword improvers to cover it.

Let $k = \frac{20\log(m)}{\eta}$. Then for each $j \in [k]$ and $i \in [m]$, we choose random $r_{i,j} \in \{0,1\}^R$ and define I_j by $(I_j(w))_i = D^w(i, r_{i,j})$. We want to show that with positive probability, this construction works. Specifically, we want to show that with positive probability for every $w \in \Sigma_2^m$ and $x \in \Sigma_1^n$ with $\Delta(w, C(x)) \le d_1$ we have that

$$\sum_{i\in[m], j\in[k]} 1_{I_j(w)_i \ne C(x)_i} \le 20\log(m)\Delta(w, C(x)).$$

To do this, we choose some $d \le d_1$, and then we want to show that with probability greater than $1 - \frac{1}{m}$ we have that for every error pattern $z \in \{0,1\}^m$ with $wt(z) = d$ we have that

$$\forall x \in \Sigma_1^n, w \in \mathsf{Diff}(C(x), z) : \sum_{i\in[m], j\in[k]} 1_{I_j(w)_i \ne C(x)_i} \le 20 d\log(m).$$

See that

$$\begin{aligned} \max_{x\in\Sigma_2^m, w\in\mathsf{Diff}(C(x),z)} \sum_{j\in[k]} \Delta(I_j(w), C(x)) &= \max_{x\in\Sigma_2^m, w\in\mathsf{Diff}(C(x),z)} \sum_{j\in[k], i\in[m]} 1_{D^w(i,r_{i,j})\ne C(x)_i} \\ &\le \sum_{j\in[k], i\in[m]} \max_{x\in\Sigma_2^m, w\in\mathsf{Diff}(C(x),z)} 1_{D^w(i,r_{i,j})\ne C(x)} \\ &= \sum_{i\in[m], j\in[k]} 1_{\exists x\in\Sigma_2^m, w\in\mathsf{Diff}(C(x),z): D^w(i,r_{i,j})\ne C(x)_i}. \end{aligned}$$

So we will actually show that with good probability, for every error pattern $z \in \{0,1\}^m$ with $wt(z) = d$ we have that

$$\sum_{i\in[m], j\in[k]} 1_{\exists x\in\Sigma_2^m, w\in\mathsf{Diff}(C(x),z): D^w(i,r_{i,j})\ne C(x)_i} = O(d\log(m)).$$

We do this with a Chernoff bound. By the definition of an MOSLCC, see that the expectation of the sum is

$$\mu \le \frac{mk}{2m} = k/2.$$

So by the Chernoff bound, the probability that sum is more than $19d\log(m)$ if $k \le 20d\log(m)$ is at most

$$e^{-0.9^2(10d\log(m))/(2+0.9)} \le e^{-2d\log(m)} < m^{-d-1}.$$

There are at most m^d many $z \in \{0,1\}^m$ with $wt(z) = d$, so in particular for all but less than $\frac{1}{m}$ of the choices of $I_1, \dots, I_k$ we have that for all of the corruptions $z \in \{0,1\}^m$ with $wt(z) = d$ we have that

$$\sum_{i \in [m], j \in [k]} 1_{\exists x \in \Sigma_2^m, w \in \mathsf{Diff}(C(x), z) : D^w(i, r_{i,j}) \neq C(x)_i} \leq 19d \log(m) < \eta k d.$$

That is, with probability more than $1 - \frac{1}{m}$, we have that $\mathcal{I}$ is a d to ηd improving set for C. Then by a union bound, with positive probability, for all $d \leq d_1$, we have that I is a d to ηd improving set for C. So let $\mathcal{I}$ be such a set.

For every $j \in [k]$, by definition of each I_j, for any $i \in [m]$, it only runs D once to calculate $(I_j(w))_i$, thus only ever queries w at q places. So $\mathcal{I}$ is a q query, below d_1, factor η improving set for C. ☐

Thus with this improvement, for any distance d, there is some choice of $k = O(\log(m)/\eta)$ candidate codeword improvers that will improve any corruption d codeword to a corruption at most ηd codeword. This gives us the small set of randomness strings we need to make the decoder deterministic. Combining this with Lemma 4.6 gives a time and space efficient decoder for any systematic MOSLCC. In the next section we prove Lemma 4.6.

5.2 Decoders From Improving Sets and VLTC Analysis

Now we know that MOSLCCs have small improving sets. But we still need to check the codeword improvers to make sure we select one that actually improves the codeword. To do this, we need our code to be checkable. We now show that if a code both has a small improving set and a vicinity local tester, then it we can find a codeword improver efficiently.

Lemma 4.5 (VLTCs Can Select From Improving Sets)**.** *Let $C : \Sigma_1^n \to \Sigma_2^m$ be a code that has a set of functions $\mathcal{I}$ that is a q_c query, below d_1, factor η improving set for C. Let C also be a VLTC with q_t testing queries, R testing randomness, approximation factor α and vicinity d_0. Suppose that $3d_1 \leq d_0$.*

Then there is a non-uniform algorithm that runs in time $O\left((m + 2^R q_t) q_c |\mathcal{I}| \log(|\Sigma_2|)\right)$ and space $O(R + (q_t + q_c)\log(|\Sigma_2|) + \log(m) + \log(|\mathcal{I}|))$ and takes as input a $w \in \Sigma_2^m$ with $\Delta(w, C) \leq d_1$ and outputs a $O(\log(|\mathcal{I}|))$ bit index of some $I \in \mathcal{I}$ that improves w to distance $\alpha^2 \eta \Delta(w, C)$ with respect to C.

If the improving set is uniform and computable in time T_c and space S_c, and the tester is uniform and computable in time T_t and space S_t, then the algorithm is uniform and runs in time $O\left((2^R T_t + m T_c + 2^R q_t T_c)|\mathcal{I}|\right)$ and space $O(R + S_t + S_c + \log(m) + \log(|\mathcal{I}|))$.

Proof. All we need to do is iterate through all the choices of $I \in \mathcal{I}$ and use the VLTC property to check if I is an improver for w. There will be two tests. If I passes both tests, the algorithm outputs the index of I in $\mathcal{I}$.

1. If $\Delta(I(w), w) > 2d_1$, then reject.
2. Run every test in the VLTC on $I(w)$. If the probability of a test failing is more than $\frac{\alpha\eta\Delta(w,C)}{m}$, then reject.

See that the total amount of space is just the space to hold a candidate codeword improver plus the space to run the codeword improver plus the space to run the LTC plus the space to hold which LTC test we are on. This is space

$$\log(|\mathcal{I}|) + (q_c + q_t)\log(|\Sigma_2|) + O(R + \log(m)).$$

The total time is just the time to decode every symbol plus the time to run every local test times the time to decode every symbol of that test all times the size of the improving set. This is time

$$O\left((m q_c + 2^R q_t q_c)|\mathcal{I}| \log(|\Sigma_2|)\right) = O\left((m + 2^R q_t) q_c |\mathcal{I}| \log(|\Sigma_2|)\right).$$

The time and space in the uniform case follows in a similar way.

Now we show that the protocol never outputs an invalid I. For $w \in \Sigma_2^m$ with $\Delta(w, C) \leq d_1$, there is some $x \in \Sigma_1^n$ with $\Delta(w, C(x)) \leq d_1$. Then suppose that for some $I \in \mathcal{I}$ that $\Delta(I(w), C(x)) > 3d_1$. Then by the triangle inequality, we must have that $\Delta(I(w), w) > 2d_1$, so the first test rejects I. Otherwise, we have that $\Delta(I(w), C(x)) \leq 3d_1 \leq d_0$. If $\Delta(I(w), C(x)) > \alpha^2\eta\Delta(w, C)$, then the probability a local tests fails is at least

$$\Pr_{r\in\{0,1\}^R}[V^{I(w)}(r)] \geq \frac{\Delta(I(w), C)}{m\alpha} > \frac{\alpha\eta\Delta(w, C)}{m}$$

so the second test rejects I.

Finally, we show that there is some $I \in \mathcal{I}$ that is good enough the protocol outputs it. By Lemma 4.4, there is an $I \in \mathcal{I}$ such that I improves w to distance $\eta\Delta(w, C)$ with respect to C. That is, if for some $x \in \Sigma_1^n$ we have that $\Delta(w, C(x)) \leq d_1$, then there is some $I \in \mathcal{I}$ such that

$$\Delta(I(w), C(x)) \leq \eta\Delta(w, C(x)) \leq d_1.$$

By the triangle inequality, $\Delta(I_j(w), w) \leq 2d_1$, so the first test passes. By the property of the VLTC, the probability the local test fails is at most

$$\Pr_{r\in\{0,1\}^R}[V^{I(w)}(r)] \leq \frac{\alpha\Delta(I(w), C(x))}{m} \leq \frac{\alpha\eta\Delta(w, C(x))}{m}.$$

So the second test passes. So the test outputs the index of some $I \in \mathcal{I}$, and any $I \in \mathcal{I}$ it outputs improves w to distance $\alpha^2\eta d$. □

So now we have that VLTCs can efficiently find a codeword improver from an improving set. Now we want to apply several codeword improvers together until we get a fully corrected codeword.

Lemma 4.6 (Deterministic Correctors from Improving Sets and VLTC)**.** *Let $C : \Sigma_1^n \to \Sigma_2^m$ be a code that has a set of functions $\mathcal{I}$ that are q_c query, below d_1, factor η improving sets for C. Let C also be a VLTC with q_t testing queries, R testing randomness, approximation factor α and vicinity d_0. Suppose that $3d_1 \leq d_0$ and $\eta\alpha^2 < 1$ and define $\ell = \lceil \frac{\log(d_1+1)}{\log(1/(\eta\alpha^2))} \rceil$.*

Then the code C has a deterministic non-uniform corrector with correcting radius d_1 running in time

$$O(\ell(2^R q_t + m)|\mathcal{I}|q_c^\ell \log(|\Sigma_2|))$$

and space

$$O(\ell \log(|\mathcal{I}|) + (\ell q_c + q_t) \log(|\Sigma_2|) + \log(m) + R).$$

If the improving set and the tester are time T space S uniform, then the corrector is uniform and runs in time at most $O(\ell(2^R q_t + m)T|\mathcal{I}|q_c^\ell)$ and the space at most $O(\ell \min\{S, q_c \log(|\Sigma_2|)\} + \ell \log(|\mathcal{I}|) + \log(m) + R + S)$. More generally, C has a uniform, deterministic algorithm, g, which runs in time at most $O(\ell(2^R q_t + m)T|\mathcal{I}|q_c^\ell)$ and the space at most $O(\ell \min\{S, q_c \log(|\Sigma_2|)\} + \ell \log(|\mathcal{I}|) + \log(m) + R + S)$ and outputs an $O(\ell \log(|\mathcal{I}|))$ bit description of a deterministic, q_c^ℓ query function f such that if for some $x \in \Sigma_1^n$ and $y \in \Sigma_2^m$ we have that $\Delta(y, C(x)) \leq d_1$, then we have that $g(x)$ outputs an f such that $f(y) = C(x)$. Further, for each $i \in [m]$, $f(y)_i$ is computable in time at most $O(Tq_c^\ell)$ and the space at most $O(\ell \min\{S, q_c \log(|\Sigma_2|)\} + S)$.

Proof. We first find suitable codeword improvers so that when they are all composed, they entirely correct the corrupted codeword. Then we run this codeword improver to decode the code. We proceed by induction.

First, by Lemma 4.5, there is an algorithm, B_1', that takes any $w_1 \in \Sigma_2^m$ that has an $x \in \Sigma_1^n$ with $\Delta(w_1, C(x)) \leq (\eta\alpha^2)^{\ell-1} d_1$ and outputs an $O(\log(|\mathcal{I}|))$ bit description of a q_c query code improver, I_1, that improves w_1 to distance $(\eta\alpha^2)^\ell d_1 \leq \frac{d_1}{d_1+1} < 1$. That is, $I_1(w_1) = C(x)$. Further B_1' runs in time $O\left((m + 2^R q_t)q_c|\mathcal{I}| \log(|\Sigma_2|)\right)$ and space $O(R + (q_t + q_c) \log(|\Sigma_2|) + \log(m) + \log(|\mathcal{I}|))$. Then by running B_1' and then running I_1 to correct all the codeword symbols, we get an algorithm, B_1, that prints $C(x)$ in time $O\left((m + 2^R q_t)q_c|\mathcal{I}| \log(|\Sigma_2|)\right)$ and space $O(R + (q_t + q_c) \log(|\Sigma_2|) + \log(m) + \log(|\mathcal{I}|))$.

Suppose that for some $i \in [\ell-1]$ we have an algorithm B_i that takes any $w_i \in \Sigma_2^m$ that has an $x \in \Sigma_1^n$ with $\Delta(w_i, C(x)) \leq (\eta\alpha^2)^{\ell-i} d_1$ and prints $C(x)$. Further suppose B_i runs in time $O(i(m + 2^R q_t)q_c^i|\mathcal{I}| \log(|\Sigma_2|))$ and space $O(i \log(|\mathcal{I}|) + (iq_c + q_t) \log(|\Sigma_2|) + \log(m) + R)$.

Then by Lemma 5.2, there is an algorithm, B_{i+1}' that takes any $w_{i+1} \in \Sigma_2^m$ that has an $x \in \Sigma_1^n$ with $\Delta(w_{i+1}, C(x)) \leq (\eta\alpha^2)^{\ell-i+1} d_1$ and outputs an $O(\log(|\mathcal{I}|))$ bit description of a q_c query code improver, I_{i+1}, that improves w_{i+1} to distance $(\eta\alpha^2)^{\ell-i}d_1$. Further, B'_{i+1} runs in time $O\left((m+2^R q_t)q_c|\mathcal{I}|\log(|\Sigma_2|)\right)$ and space $O(R+(q_t+q_c)\log(|\Sigma_2|)+\log(m)+\log(|\mathcal{I}|))$.

Now to construct B_{i+1}, one first runs B'_{i+1} on w_{i+1} to get improver I_{i+1}. Then one runs B_i on $w_i = I_{i+1}(w_{i+1})$ to print $C(x)$. By assumption $w_i = I_{i+1}(w_{i+1})$ has distance $(\eta\alpha^2)^{\ell-i}d_1$, thus B_i must print $C(x)$.

See that the time to run B_{i+1} is the time to run B'_{i+1} plus the time to simulate B_i. The time to simulate B_i is only the time to run B_i times q_c since every query to the input of B_i needs to actually query q_c elements of w. Thus simulating B_i takes time

$$O(i(m+2^R q_t)q_c^{i+1}|\mathcal{I}|\log(|\Sigma_2|)).$$

Adding the time to run B'_{i+1} still has B_{i+1} running in time

$$O((i+1)(m+2^R q_t)q_c^{i+1}|\mathcal{I}|\log(|\Sigma_2|)).$$

For space, we only need the max of the space to run B'_{i+1} and the space to simulate running B_i. The space required to simulate running B_i is just the space to hold I_{i+1} plus the space to run B_i plus the space to run I_{i+1}. This only requires space

$$\begin{aligned}
&\max\{O(R+(q_t+q_c)\log(|\Sigma_2|)+\log(m)+\log(|\mathcal{I}|)),\\
&\quad O(i\log(|\mathcal{I}|)+(iq_c+q_t)\log(|\Sigma_2|)+\log(m)+R)+q_c\log(|\Sigma_2|)\}+\log(|\mathcal{I}|)\\
=&O(((i+1)q_c+q_t)\log(|\Sigma_2|)+(i+1)\log(|\mathcal{I}|)+\log(m)+R).
\end{aligned}$$

The time and space in the uniform case follows from the uniform case of Lemma 4.5 and a similar argument. The only difference to output f instead of $C(x)$ is that instead of B_{i+1} printing the codeword, it instead just outputs the codeword improver from B_i composed with I_{i+1}. This is exactly the same thing B_ℓ does, except that B_ℓ actually evaluates the codeword improver instead of outputting it. □

5.3 Decoders for MOSLCCs Analysis

From Theorem 3.15 we know that MOSLCCs are also VLTCs, and from Lemma 5.2 we know that MOSLCCs have improving sets, so we can combine these with Lemma 4.6 to get a deterministic decoder for any systematic MOSLCC.

Theorem 5.3 (MOSLCCs have Efficient Deterministic Correctors)**.** *Suppose some systematic code $C:\Sigma_1^n \to \Sigma_2^m$ is a MOSLCC with MOS correcting radius d, soundness $\frac{1}{11m}$ and q correcting queries.*

Then for any integer $\ell \geq 1$, the code C has a deterministic non-uniform corrector with correcting radius $d/3$ running in time

$$O(\ell m d^{1/\ell}\log(m)^2 q^{\ell+1}\log(|\Sigma_2|))$$

and space

$$O(\ell(q\log(|\Sigma_2|)+\log(m))).$$

Proof. From Theorem 3.15 we know that C is also an VLTC with vicinity d, $q+1$ queries, randomness $\log(m)+\log(\log(m))+O(1)$ and approximation factor $\alpha=2$.

Let $\eta = \frac{1}{\alpha^2}(d/3+1)^{-1/\ell}$. From Lemma 5.2, for some $k = O\left(\frac{\log(m)}{\eta}\right)$ there exists a k element set of functions $\mathcal{I}$ that is a q query, below $d/3$, factor η improving set for C.

Now see that

$$\begin{aligned}
\frac{\log(d/3+1)}{\log(1/(\eta\alpha^2))} =&\frac{\log(d/3+1)}{\log(1/(d/3+1)^{-1/\ell})}\\
=&\ell.
\end{aligned}$$

Then by Lemma 4.6, the code C has a deterministic corrector with correcting radius $d/3$ running in time

$$\begin{aligned}
O(\ell(2^R q_t+m)|\mathcal{I}|q_c^\ell\log(|\Sigma_2|)) =&O(\ell(m\log(m)(q+1)+m)d^{1/\ell}\log(m)q^\ell\log(|\Sigma_2|))\\
=&O(\ell m d^{1/\ell}\log(m)^2 q^{\ell+1}\log(|\Sigma_2|))
\end{aligned}$$

and space

$$O(\ell\log(|\mathcal{I}|)+(\ell q_c+q_t)\log(|\Sigma_2|)+\log(m)+R) = O(\ell q\log(|\Sigma_2|)+\log(m)+\ell\log(\log(m))).$$

□

5.4 Decoders for Typical LCCs Analysis

Up till now, we have described the deterministic corrector for a MOSLCC, now we discuss correctors for typical LCCs. You may recall that typical LCCs are also MOSLCCs from Lemma 3.16, but in performing this transformation, the correcting distance decreases significantly, to $O(\frac{m}{q})$. So while the reduction from typical LCCs to MOSLCCs does give you an efficient deterministic decoder for typical LCCs, the decoding radius is no longer $\Omega(m)$.

The solution to this is to start by correcting as a regular LCC, but instead of correcting down to zero errors, we *only* correct into the MOS correcting radius. Since LCCs have randomness efficient tests for testing a large fraction of errors from Lemma 3.14 and have small improving sets for improving to large distances from Lemma 5.1, an LCC can efficiently correct into the MOS correction radius. Then we can use the decoder for MOSLCCs to decode the message.

The following is a more specific formulation of Theorem 1.2.

Theorem 5.4 (Typical LCCs have Efficient Deterministic Correctors)**.** *Suppose code $C : \Sigma_1^n \to \Sigma_2^m$ is a typical LCC with smoothness β, and q queries as well as an LCC with correcting radius d_1 and q queries.*

Then for any integer $\ell \geq 1$, the code C has a deterministic non-uniform corrector and decoder with correcting and decoding radius $d = d_1/3$ running in time

$$O(mq^2(\beta + \ell(m/\beta)^{1/\ell}q^\ell \log(m)^{\ell+3}2^{O(\ell)})\log(|\Sigma_2|))$$

and space

$$O(q(\log(\beta) + \ell\log(m))\log(|\Sigma_2|)).$$

Proof. The idea is to use one codeword improver to get within the MOS correcting radius, then use our efficient decoder for MOSLCCs. Specifically, we first use Lemma 5.1 to get a short list of candidate codeword improvers that will improve our input to within distance $d_2 = \frac{m}{10\beta}$, and find such a codeword improver for our input using Lemma 3.14. Then using Lemma 3.16, our code is a MOSLCC and by Theorem 5.3 our input is within the decoding radius of a time and space efficient deterministic decoder.

By Lemma 3.11, we can get a $O(q\log(m))$ query corrector for C that has soundness $\frac{1}{10m}$. Then by Lemma 5.1, for some $k = O\left(\frac{m}{d_2+1}\right) = O(\beta)$ there exists a k element set of functions $\mathcal{I}$ such that for all $d \leq d_1$ we have that $\mathcal{I}$ is a q query d to d_2 improving set for C. Denote the elements of $\mathcal{I}$ as $I_1, \ldots, I_j$.

Now consider an input $w \in \Sigma_2^m$ such that there is some $x \in \Sigma_1^n$ with $\Delta(w, C(x)) \leq d$. Our algorithm first checks all $O(\beta)$ of these candidate codeword improvers and finds one I_j such that

1. $\Delta(w, I_j(w)) \leq 2d$ and
2. for the test, V, from Lemma 3.14 we require that

   $$\Pr_{r\in\{0,1\}^R}[V^{I_j(w)}r) = 1] \leq \frac{3d_2}{4m}$$

   where $R = \log(\frac{m^2}{d_2}) + O(1) = \log(m\beta) + O(1)$.

See that such a test can be run in time

$$O((2^R q\log(m)q + m(q+1))\log(|\Sigma_2|)) = O(m\beta q^2\log(m)\log(|\Sigma_2|))$$

and space

$$O(R + q\log(m)\log(|\Sigma_2|) + \log(m)) = O(q\log(m\beta)\log(|\Sigma_2|)).$$

We claim that any I_j passing these tests must improve w to distance d_2. Suppose that $\Delta(I_j(w), C(x)) > d_2$. If $\Delta(I_j(w), C(x)) > 3d$, then $\Delta(w, I_j(w)) > 2d$, so I_j wouldn't pass. If $\Delta(I_j(w), C(x)) \leq 3d = d_1$, then by the soundness of Lemma 3.14

$$\Pr_{r\in\{0,1\}^R}[V^{I_j(w)}(r) = 1] > \frac{3d_2}{4m},$$

so the test fails. So any I_j passing the test must have $\Delta(I_j(w), C(x)) \leq d_2$.

Further see that some I_j will pass the test, specifically the I_j that improves w to distance $d_2/2$. This is by the soundness of Lemma 3.14 and the fact that $d_2 \leq d$ (if $d_2 > d$, we use the identity function as our codeword improver).

Now by Lemma 3.16, we also have that C is a MOSLCC with MOS correcting radius $d' = \frac{m}{3\beta}$, soundness $s = \frac{1}{11m}$, and $O(q\log(m))$ queries. So by Theorem 5.3, we have that for any integer $\ell \geq 1$, the code C has a deterministic corrector with correcting radius $d'/3 > \frac{m}{10\beta} = d_2$ running in time

$$O(\ell m d'^{1/\ell} \log(m)^2 (O(q\log(m)))^{\ell+1} \log(|\Sigma_2|))$$

and space

$$O((\ell q \log(m) + \ell \log(m)) \log(|\Sigma_2|)) = O(\ell q \log(m) \log(|\Sigma_2|)).$$

Since $\Delta(I_j(w), C(x)) \leq d'/3$, we have that this decoder correctly corrects $I_j(w)$. Then simulating this only requires an extra q factor time overhead and an extra q additive space overhead.

So the final time of the overall corrector is

$$\begin{aligned} &O(m\beta q^2 \log(m) \log(|\Sigma_2|)) + O(\ell m d'^{1/\ell} \log(m)^2 (O(q\log(m)))^{\ell+1} q \log(|\Sigma_2|)) \\ =&O(mq^2(\beta \log(m) + \ell (m/\beta)^{1/\ell} q^\ell \log(m)^{\ell+3} 2^{O(\ell)}) \log(|\Sigma_2|)) \end{aligned}$$

and the final space of the overall corrector is

$$O(q\log(m\beta)\log(|\Sigma_2|)) + O(\ell q \log(m) \log(|\Sigma_2|) + q\log(|\Sigma_2|)) = O(q(\log(\beta) + \ell \log(m)) \log(|\Sigma_2|)).$$

Since a typical LCC is systematic, the corrector immediately implies a decoder by just outputting the codeword bits that are message bits. □

The local tests and local corrections for correcting the input into the MOS correcting radius are the same as those for correcting the MOSLCC (except that we may need slightly more local tests). So the algorithm for the typical LCC case is the same as that for MOSLLC, the only difference is the analysis.

5.5 Improving Set is All You Need

We have shown that VLTCs and improving sets together give a time and space efficient deterministic corrector. We showed that all MOSLCCs have both of these properties. In this section, we show that improving sets directly give VLTCs. This is because, on average, the functions in an improving set give a good approximation of the closest codeword. So comparing the output of the improving set to the input gives a close estimate of the corruption.

So to construct time and space efficient decoders for a code, one only needs to find improving sets for that code. So to find *uniform* time and space efficient deterministic decoders for a code it suffices to just make the improving sets uniform.

Lemma 5.5 (Improving Sets Give VLTCs). *Let $C : \Sigma_1^n \to \Sigma_2^m$ be a code with a set of functions $\mathcal{I}$ that is a q query, below d, factor η improving set for C. Then C has a local tester that uses $q+1$ testing queries, has* $\log(|\mathcal{I}|) + \log(m)$ *testing randomness, approximation factor $\frac{1}{1-\eta}$ and vicinity d.*

If each function in $\mathcal{I}$ is uniform and can compute any single output symbol in time $T > \log(|\Sigma_2|) + \log(m)$, then the local tester is uniform and can be run in time $O(T)$.

Proof. The VLTC just uses runs every function $I \in \mathcal{I}$ on the input $w \in \Sigma_2^m$ and compares $I(w)$ to w. Since $\mathcal{I}$ is an improving set, in expectation $I \in \mathcal{I}$ will be a good approximation of $C(x)$ where $C(x)$ is the closest codeword to w. Formally, define for $(j, i) \in [|\mathcal{I}|] \times [m]$ define V by

$$V^w((j,i)) = \begin{cases} 1 & (I_j(w))_i \neq w_i \\ 0 & (I_j(w))_i = w_i \end{cases}$$

where I_j is the jth element of $\mathcal{I}$ by some canonical ordering.

So take $w \in \Sigma_2^m$ such that for some $x \in \Sigma_1^n$ we have that $\Delta(w, C(x)) \leq d$. Then

$$\begin{aligned}\Pr_{I\in\mathcal{I}, i\in[m]}[I(w)_i \neq w_i] \leq& \Pr_{i\in[m]}[w_i \neq C(x)_i] + \Pr_{I\in\mathcal{I}, i\in[m]}[I(w)_i \neq C(x)_i]\\ \leq& \frac{\Delta(w, C(x))}{m} + \eta\frac{\Delta(w, C(x))}{m}\\ \leq&(1+\eta)\frac{\Delta(w, C(x))}{m}\\ \leq& \frac{1}{1-\eta}\frac{\Delta(w, C(x))}{m}.\end{aligned}$$

And similarly

$$\begin{aligned}\Pr_{I\in\mathcal{I}, i\in[m]}[I(w)_i \neq w_i] \geq& \Pr_{i\in[m]}[w_i \neq C(x)_i] - \Pr_{I\in\mathcal{I}, i\in[m]}[I(w)_i \neq C(x)_i]\\ \leq& \frac{\Delta(w, C(x))}{m} - \eta\frac{\Delta(w, C(x))}{m}\\ \leq&(1-\eta)\frac{\Delta(w, C(x))}{m}.\end{aligned}$$

So V is a strong local tester with vicinity d, $q+1$ queries, randomness $\log(|\mathcal{I}|m)$, and approximation factor $\frac{1}{1-\eta}$. And since V only ever runs I and then queries a single bit of the input, if each function in $\mathcal{I}$ is efficient to compute, then so is V. □

Now that we know that all uniform improving sets also give uniform VLTCs, we can combine this with Lemma 4.6 to show that just a uniform improving set is all we need for uniform deterministic correcting. Further if the code is systematic, correcting implies decoding.

Lemma 5.6 (Improvers Give Correctors)**.** *Let $C : \Sigma_1^n \to \Sigma_2^m$ be a code with a set of functions $\mathcal{I}$ that is a q query, below d, factor η improving set for C. Suppose that $\eta < (1-\eta)^2$ and define $\ell = \lceil \frac{\log(d/3+1)}{\log((1-\eta)^2/\eta)} \rceil$*

Then the code C has a deterministic corrector with correcting radius $d/3$ running in time

$$O(m\ell|\mathcal{I}|^2 q^{\ell+1}\log(|\Sigma_2|))$$

and space

$$O(\ell q\log(|\Sigma_2|) + \ell\log(|\mathcal{I}|) + \log(m)).$$

If the improving set and the tester are time T space S uniform, then the running time is at most $O(m\ell|\mathcal{I}|^2q^{\ell+1}T)$ and the space is at most $O(\ell\min\{S, q\log(|\Sigma_2|)\} + \ell\log(|\mathcal{I}|) + \log(m) + S)$. More generally, C has a uniform, deterministic algorithm, g, which runs in time at most $O(m\ell|\mathcal{I}|^2q^{\ell+1}T)$ and the space is at most $O(\ell\min\{S, q\log(|\Sigma_2|)\} + \ell\log(|\mathcal{I}|) + \log(m) + S)$ and outputs an $O(\ell\log(|\mathcal{I}|))$ bit description of a deterministic, q_c^ℓ query function f such that if for some $x \in \Sigma_1^n$ and $y \in \Sigma_2^m$ we have that $\Delta(y, C(x)) \leq d_1$, then we have that $g(x)$ outputs an f such that $f(y) = C(x)$. Further, for each $i \in [m]$, $f(y)_i$ is computable in time at most $O(Tq_c^\ell)$ and the space at most $O(\ell\min\{S, q\log(|\Sigma_2|)\} + S)$.

Proof. By Lemma 5.5, we know C has a local tester that uses $q+1$ testing queries, has $\log(|\mathcal{I}|) + \log(m)$ testing randomness, approximation factor $\frac{1}{1-\eta}$ and vicinity d.

By Lemma 4.6, we have that the code C has a deterministic corrector with correcting radius $d/3$ running in time

$$O(\ell(2^R q_t + m)|\mathcal{I}|q_c^\ell\log(|\Sigma_2|)) = O(\ell(m|\mathcal{I}|q + m)|\mathcal{I}|q^\ell\log(|\Sigma_2|)) = O(\ell m|\mathcal{I}|^2q^{\ell+1}\log(|\Sigma_2|))$$

and space

$$O(\ell\log(|\mathcal{I}|) + (\ell q_c + q_t)\log(|\Sigma_2|) + \log(m) + R) = O(\ell q\log(|\Sigma_2|) + \ell\log(|\mathcal{I}|) + \log(m)).$$

And the uniform case follows similarly. □

6 Proof of Non-Uniform Version of Theorem 1.1

We have proven that MOSLCCs, typical LCCs, and any code with small improving sets all have efficient deterministic decoders. Now we want to show that some good code has an almost linear time, subpolynomial space decoder. We start with the LCCs of Kopparty, Meir, Ron-Zewi, and Saraf [Kop+17]. To do this, we only need to show that their LCC is typical.

Theorem 6.1. *The codes of [Kop+17, Theorem 1.1], denoted* $C : \{0,1\}^n \to \{0,1\}^m$ *where* $m = O(n)$*, has a deterministic non-uniform algorithm* B *computing a function* $D : \{0,1\}^m \to \{0,1\}^n$ *such that:*

Efficient: B *runs in time* $n2^{O\left(\log(n)^{3/4}\sqrt{\log(\log(n))}\right)}$ *and space* $2^{O\left(\sqrt{\log(n)\log(\log(n))}\right)}$.

Decodes: *For some* $d = \Omega(m)$*, for any* $x \in \{0,1\}^n$ *and* $w \in \{0,1\}^m$ *with* $\Delta(w, C(x)) \le d$ *we have that*

$$D(w) = x.$$

Corrects: *There is also a function* $D' : \{0,1\}^m \to \{0,1\}^m$ *and a deterministic non-uniform algorithm computing* D' *that runs in the same time and space as* B *such that for any* $x \in \{0,1\}^n$ *and* $w \in \{0,1\}^m$ *with* $\Delta(w, C(x)) \le d$ *we have that*

$$D'(w) = C(x).$$

Proof. From [Kop+17, Theorem 1.1], C is an LCC with correcting radius $\Omega(m)$ and query complexity $q = 2^{O(\sqrt{\log(n)\log(\log(n))})}$. Now we only need to show that C is also a typical LCC.

First the code is linear, so we can assume it is systematic. Now all we need to show is that it is non-adaptive, smooth, and has perfect completeness. The local corrector as described in [Kop+17, Lemma 3.4] makes many non-adaptive calls to the local corrector of a multiplicity code with suitable parameters. As long as the multiplicity code's corrector is non-adaptive, smooth, and has perfect completeness, then so does theirs.

A codeword of the multiplicity code can be seen as a function $f : \mathbb{F}_p^a \to \Sigma_2$. For our analysis, it does not matter what Σ_2 is, we only need to know the behaviour of the local corrector. The local corrector for the multiplicity code is a variation of the corrector in [Kop13, Section 4.1.2], which chooses a set of lines through the symbol we wish to correct and queries each symbol in those lines. This local correction has perfect completeness and is non-adaptive.

A random one of those lines is also uniformly randomly chosen from the set of lines through the symbol to correct. The local corrector is *almost* smooth, except that it always queries the symbol to be corrected. But if one modifies this corrector to simply not query this point, the corrector still has perfect completeness and is now $\beta \le 2q$ smooth.

Now setting $\ell = (\log(n))^{1/4}$ we can apply Theorem 5.4 to get that C has a deterministic corrector and decoder with correcting and decoding radius $d = \Omega(m)$ running in time

$$\begin{aligned}
&O(mq^2(\beta + \ell(m/\beta)^{1/\ell}q^\ell \log(m)^{\ell+3})) \\
=&O(mq^2(q + \ell(O(m))^{1/\ell}q^\ell \log(m)^{\ell+3})) \\
=&O(m\ell(O(m))^{1/\ell}q^{2+\ell}\log(m)^{\ell+3}) \\
=&O(m\ell 2^{(\log(m))^{3/4}}2^{O\left(\sqrt{\log(m)\log(\log(m))}\log(m)^{1/4}\right)}2^{O(\log(m)^{1/4}\log(\log(m)))}) \\
=&n2^{O\left(\log(n)^{3/4}\sqrt{\log(\log(n))}\right)}.
\end{aligned}$$

and space

$$O(q\log(m\beta) + \ell q \log(m)) = 2^{O(\sqrt{\log(n)\log(\log(n))})}.$$

To decode, we just use that the code is systematic. □

Remark (Exact Efficiency of Decoding)**.** *We note that the decoders for the multiplicity based codes in [Kop+17] are more efficient than those based on lifted Reed-Solomon codes. Even randomized local correctors for lifted Reed-Solomon based codes don't achieve time near* $2^{\log(n)^{3/4}}$*. However, our technique cannot give decoders running in time* $o(m2^{\sqrt{\log(m)}})$ *. We leave finding better deterministic decoders as open problem, Item 3.*

7 Uniform Decoding of Reed-Muller Codes

In this section we give an efficient, uniform decoder for Reed-Muller codes. As described in Section 5.5, all we need is an explicit, small improving set for that code. For Reed-Muller codes, an improving set can be constructed from a family of curves that is a good sampler. While there were known constructions of explicit curve samplers [TSU06; Guo13], they do not have good enough parameters for us. In this section we construct better curve samplers that may be of independent interest.

7.1 Definitions

Now we define a sampler.

Definition 7.1 (Sampler)**.** *For any set P, set of randomness strings $\mathcal{C}$, and sample size q, we say that a function $\mathsf{samp} : \mathcal{C} \to P^q$ is a sampler for P with sample size q, accuracy error ϵ and confidence error δ if for any $A \subseteq P$ with $\mu = \frac{|A|}{|P|}$ we have that*

$$\Pr_{c \in \mathcal{C}}[\Pr_{i \in [q]}[\mathsf{samp}(c)_i \in A] \geq \mu + \epsilon] \leq \delta.$$

On randomness $c \in \mathcal{C}$, we call $\mathsf{samp}(c)$ a sample.
We call samp unbiased if for every $u \in P$ we have that

$$\Pr_{c \in \mathcal{C}, i \in [q]}[\mathsf{samp}(c)_i = u] = \frac{1}{|P|}.$$

The size of samp is $|\mathcal{C}|$ and the randomness of samp is $\log(|\mathcal{C}|)$.

We need a stronger property than standard confidence error. Namely, we need the confidence error to be proportional to $\mu = \frac{|A|}{|P|}$, and in particular decrease as μ decreases. We call this strengthening of confidence error "strong confidence error".

Definition 7.2 (Sampler With Strong Confidence Error)**.** *For any set of points P, set $\mathcal{C}$, and sample size q, let $\mathsf{samp} : \mathcal{C} \to P^q$ be a sampler with accuracy error ϵ for P. We say that samp has strong confidence error δ if for any $A \subseteq P$ with $\mu = \frac{|A|}{|P|}$*

$$\Pr_{c \in \mathcal{C}}[\Pr_{i \in [q]}[\mathsf{samp}(c)_i \in A] \geq \mu + \epsilon] \leq \mu\delta.$$

We will consider low degree samplers, e.g., curve samplers and subspace samplers.

Definition 7.3 (Affine Subspace)**.** *Let dim and a be integers and $\mathbb{F}$ be a field. Then the function $s : \mathbb{F}^a \to \mathbb{F}^{\mathsf{dim}}$ is an a-dimensional affine subspace of $\mathbb{F}^{\mathsf{dim}}$ if for every $1 \leq i \leq \mathsf{dim}$ we have that s_i is a degree at most 1 function.*

A subspace sampler is a sampler whose samples are affine subspaces.

Definition 7.4 (Subspace Sampler)**.** *Let dim be an integer and $\mathbb{F}$ be a field. Let $\mathcal{S}$ be a set of a-dimensional affine subspaces of $\mathbb{F}^{\mathsf{dim}}$. Then we say that $\mathcal{S}$ is an a-dimensional subspace sampler for $\mathbb{F}^{\mathsf{dim}}$ if the function that takes $s \in \mathcal{S}$ and outputs $(s(i))_{i \in \mathbb{F}^a}$ is a sampler.*

For any $u \in \mathbb{F}^{\mathsf{dim}}$ we define the multiset $\mathcal{S}^u$ such that $s \in \mathcal{S}$ is in $\mathcal{S}^u$ with multiplicity k if there are k elements $i \in \mathbb{F}^a$ such that $s(i) = u$. We say that $\mathcal{S}$ is time T space S uniform if, for some ordering of $\mathcal{S}^u$, denoted $\mathcal{S}^u_1, \ldots, \mathcal{S}^u_{|\mathcal{S}^u|}$, given $i \in \mathbb{F}^a$ and $j \in [|\mathcal{S}^u|]$ we have that $\mathcal{S}^u_j(i)$ can be evaluated in time T and space S.

If $a = 1$, we call the subspace sampler a line sampler.

While we will use subspace samplers in our construction, ultimately we will build a curve sampler.

Definition 7.5 (Curves)**.** *Let dim be an integer and $\mathbb{F}$ be a field. Then the function $c : \mathbb{F} \to \mathbb{F}^{\mathsf{dim}}$ is a degree t curve if for every $1 \leq i \leq \mathsf{dim}$ we have that $c(\cdot)_i$ is a degree t polynomial.*

Similar to subspace samplers, curve samplers are samplers whose samples are curves. Curve samplers are useful because they can have better confidence error with fewer queries.

Definition 7.6 (Curve Sampler)**.** *Let* dim *be an integer and* $\mathbb{F}$ *be a field. Let* $\mathcal{C}$ *be a set of degree* t *curves through* $\mathbb{F}^{\mathsf{dim}}$*. Then we say that* $\mathcal{C}$ *is a curve sampler if the function that takes* $c \in \mathcal{C}$ *and outputs* $(c(i))_{i\in\mathbb{F}}$ *is a sampler.*

For any $u \in \mathbb{F}^{\mathsf{dim}}$ *we define the multiset* $\mathcal{C}^u$ *such that* $c \in \mathcal{C}$ *is in* $\mathcal{C}^u$ *with multiplicity* k *if there are* k *field elements* $i \in \mathbb{F}$ *such that* $c(i) = u$*. We say that* $\mathcal{C}$ *is time* T *space* S *uniform if, for any ordering of* $\mathcal{C}^u$*, denoted* $\mathcal{C}^u_1, \ldots, \mathcal{C}^u_{|\mathcal{C}^u|}$*, given* $i \in \mathbb{F}$ *and* $j \in [|\mathcal{C}^u|]$ *we have that* $\mathcal{C}^u_j(i)$ *can be evaluated in time* T *and space* S*.*

To actually construct our subspace samplers for $\mathbb{F}_p^{b\mathsf{dim}}$, we will use ε-biased sets over $\mathbb{F}_{p^b}^{\mathsf{dim}}$. An ε-biased set is a set that looks close to unbiased for any test that is a character.

Definition 7.7 (Character)**.** *For any field* $\mathbb{F}$ *and dimension* dim*, a function* $\chi : \mathbb{F}^{\mathsf{dim}} \to \mathbb{C}$ *is a character if for all* $x, y \in \mathbb{F}^m$ *we have that* $\chi(x + y) = \chi(x)\chi(y)$*.*

A set of points is an ε-biased set if the expectation of any character χ on that set has magnitude at most ε.

Definition 7.8 (ε-Biased Set)**.** *For any field* $\mathbb{F}$ *and dimension* dim*, a set* $\mathcal{S} \subseteq \mathbb{F}^{\mathsf{dim}}$ *is called an* ε*-biased set if for every character* χ *we have that*

$$|\mathop{\mathbb{E}}_{s\in\mathcal{S}}[\chi(s)]| \le \epsilon.$$

Finally, any ε-biased set for $\mathbb{F}^{\mathsf{dim}}$ implies a straightforward line sampler construction.

Definition 7.9 (Lines Through a Point in a Direction)**.** *For any field* $\mathbb{F}$*, dimension* dim*, and points* $u, v \in \mathbb{F}^{\mathsf{dim}}$*, we define* $\ell^{\mathbb{F}}_{u,v} : \mathbb{F} \to \mathbb{F}^{\mathsf{dim}}$ *by*

$$\ell^{\mathbb{F}}_{u,v}(z) = u + z \cdot v.$$

We call any function that can be described in this way a line.

Definition 7.10 (Lines in Directions)**.** *For any field* $\mathbb{F}$*, dimension* dim*, and set of directions* $D \subseteq \mathbb{F}^{\mathsf{dim}}$ *define* $\ell^{\mathbb{F}}_D$ *by*

$$\ell^{\mathbb{F}}_D = \{\ell_{u,v} : u \in \mathbb{F}^{\mathsf{dim}}, y \in D\}.$$

We include the field in the definition of ℓ because if $\mathbb{F}_{p^b}$ is of order p^b for prime p and constant b, then ℓ can also be viewed as an affine subspace of $\mathbb{F}_p^{b\mathsf{dim}}$.

Lemma 7.11 (Lines as Subspaces)**.** *For any prime power* p*, natural numbers* b *and* dim*, and* $D \subseteq \mathbb{F}_{p^b}^{\mathsf{dim}}$*, if* $\ell^{\mathbb{F}_{p^b}}_D$ *is a line sampler, then there is a* b*-dimensional subspace sampler* $s^{\mathbb{F}_p}_D$ *for* $\mathbb{F}_p^{b\mathsf{dim}}$ *that is unbiased with the same size, accuracy error, confidence error, and uniformity.*

Proof. This comes from identifying the elements in $\mathbb{F}_{p^b}$ with $\mathbb{F}_p^b$ and then viewing a line $\ell^{\mathbb{F}_{p^b}}_{u,v} : \mathbb{F}_{p^b} \to \mathbb{F}_{p^b}^{\mathsf{dim}}$ as a function from $\mathbb{F}_p^b$ to $\mathbb{F}_p^{b\mathsf{dim}}$.

One can identify elements $\mathbb{F}_{p^b}$ with formal polynomials over $\mathbb{F}_p$ of degree $b-1$, modulo some irreducible degree b polynomial, ψ. This suggests an additive group isomorphism $\phi : \mathbb{F}_p^b \to \mathbb{F}_{p^b}$ that identifies the elements of $\mathbb{F}_p^b$ with the coefficients of $\mathbb{F}_{p^b}$. That is, for $a_1, \ldots, a_b \in \mathbb{F}_p^b$, we have that

$$\phi(a_1, \ldots, a_b) = \sum_{i\in[b]} a_i x^{i-1}$$

where the right hand side is a formal polynomial.

Then one can show that that $\ell^{\mathbb{F}_{p^b}}_{u,v}(a) = u + av$ is a degree one function when a is viewed as a polynomial with coefficients in $\mathbb{F}_p$. The proof involves writing out the polynomial and observing that the coefficients in a are never multiplied by each other, only by constants depending on v and added together. □

7.2 Prior Curve Samplers

The simplest curve sampler is the set of all degree t curves. This curve sampler is good, but it has too many curves. If the space has n points, there are n^{t+1} such curves. We will use this curve sampler later, but restricted to a smaller subspace so there are not too many curves. The following is from [Mos17][Proposition 4.3].

Lemma 7.12 (Naive Curve Samplers)**.** *Let $\mathbb{F}$ be a field and dim be a number of dimensions so that $m = |\mathbb{F}|^{\mathsf{dim}}$.*

For any $\epsilon > 0$ and integer t, the set of all degree t-curves through $\mathbb{F}^{\mathsf{dim}}$ is a sampler with accuracy error ϵ and strong *confidence error*

$$\left(\frac{t}{\epsilon\sqrt{|\mathbb{F}|}}\right)^t (t+1).$$

We emphasize that there are only $|\mathbb{F}|^{(t+1)\mathsf{dim}}$ degree t curves through $\mathbb{F}^{\mathsf{dim}}$, this sampler is unbiased and time $t\mathsf{dim}\,polylog(|\mathbb{F}|)$ uniform.

For comparison, Ta-Shma and Umans constructed a more randomness efficient curve sampler [TSU06]. For confidence error δ, dimension dim, sufficiently large $|\mathbb{F}|$, and $n = |\mathbb{F}|^{\mathsf{dim}}$ their curve sampler has degree $poly(\log(\mathsf{dim}/\delta))^{\log(\mathsf{dim})})$ and size $poly(n\mathsf{dim}^{\log(1/\delta)})$.

Guo gives an improved curve sampler [Guo13] that has degree $poly\left(\mathsf{dim}\frac{\log(1/\delta)}{\log(|\mathbb{F}|)}\right)$ and size $poly(n/\delta)$. While Guo improves on the prior construction, the size of both curve samplers are $\Omega(n^4)$. We cannot even afford size n^2. Further, neither result proves the samplers have strong confidence error, they only prove regular confidence error.

In contrast, our curve sampler (see Theorem 7.17) has degree $O\left(\frac{\log(1/\delta)}{\log(|\mathbb{F}|)}\right)$, size $n\,poly(\mathsf{dim})(1/\delta)^{O\left(\frac{\log(1/\delta)}{\log(|\mathbb{F}|)}\right)}$, and has strong confidence error. Our curve sampler is better in three ways.

1. The dependence of our curve sampler's size on n : the size is close to n and not n^4. Our dependence on δ is worse for small δ, but it is still polynomial in $1/\delta$ as long as $\delta = \frac{1}{|\mathbb{F}|^k}$ for some constant k. This is the regime of parameters used in our results.
2. Our curve sampler has strong confidence error. It is possible that the prior curve samplers had strong confidence error, but it was not shown.
3. Our curve sampler's degree is lower: it is independent of dim and only linear in $\frac{\log(1/\delta)}{\log(|\mathbb{F}|)}$ instead of polynomial.

7.3 ε-Biased Sets To Subspace Samplers

We will now construct explicit subspace samplers from explicit ε-biased sets. For the ε-biased set, we use the construction of Ta-Shma [TS17], generalized by Jalan and Moshkovitz [JM21, Theorem 1.1]. Here we state their result for the special case of an ε-biased set for $\mathbb{F}^{\mathsf{dim}}$.

Lemma 7.13 (Small ε-Biased Sets Exist)**.** *There is a deterministic algorithm which takes as input the order of a field $\mathbb{F}$, an integer $\mathsf{dim} \geq 1$ and $\lambda > 0$, runs in time poly $\left(\frac{\mathsf{dim}\log(|\mathbb{F}|)}{\lambda}\right)$ and outputs a λ biased set $D \subseteq \mathbb{F}^{\mathsf{dim}}$ where $|D| = O\left(\frac{\mathsf{dim}\log(|\mathbb{F}|)^{O(1)}}{\lambda^{2+o(1)}}\right)$.*

In particular, if $n = |\mathbb{F}|^{\mathsf{dim}}$ and $\lambda = \frac{1}{|\mathbb{F}|}$, then $|D| = poly(|\mathbb{F}|\mathsf{dim})$.

As was noted by Ben-Sasson, Sudan, Vadhan, and Wigderson [BS+03, Lemma 4.3], any ε-biased set also gives a sampler.

Lemma 7.14 (Lines In ε-Biased Directions are Samplers)**.** *Suppose $D \subseteq \mathbb{F}^{\mathsf{dim}}$ is λ biased. Then for any set $A \subseteq \mathbb{F}^{\mathsf{dim}}$ of density $\mu = \frac{|A|}{|\mathbb{F}|^{\mathsf{dim}}}$ and any $\epsilon > 0$, we have that the set of functions $\ell_D^{\mathbb{F}}$ is a line sampler for $\mathbb{F}^{\mathsf{dim}}$ with accuracy error ϵ and strong confidence error*

$$\left(\frac{1}{|\mathbb{F}|} + \lambda\right)\frac{1}{\epsilon^2}.$$

As a corollary of these two lemmas, we have that small, efficiently computable strong line samplers exist for any field and dimension.

Corollary 7.15 (Efficient Line Samplers Exist)**.** *There is a deterministic algorithm which takes as input the order of a field $\mathbb{F}$, and an integer $\mathsf{dim} \geq 1$, that runs in time poly$(\mathsf{dim}|\mathbb{F}|)$ and outputs a set $D \subseteq \mathbb{F}^{\mathsf{dim}}$ of size poly$(|\mathbb{F}|\mathsf{dim})$ such that for any $\epsilon > 0$, we have that $\ell_D^{\mathbb{F}}$ is a line sampler for $\mathbb{F}^{\mathsf{dim}}$ with accuracy error ϵ and strong confidence error $\frac{2}{|\mathbb{F}|\epsilon^2}$.*

Further if $\mathbb{F}$ has order p^b for prime power p and integer b we have that $s_D^{\mathbb{F}_p}$ (from Lemma 7.11) is also an unbiased b-dimensional subspace sampler for $\mathbb{F}_p^{b\mathsf{dim}}$ with size $p^{b(\mathsf{dim}+O(1))}$ $poly(\mathsf{dim})$, with accuracy error ϵ, and strong confidence error $\frac{2}{p^b\epsilon^2}$. Further $s_D^{\mathbb{F}}$ is time and space poly$(p^b\mathsf{dim})$ uniform.

Proof. The algorithm just outputs the ε-biased set from Lemma 7.13 and by Lemma 7.14 we have that $\ell_D^{\mathbb{F}}$ is a sampler with the desired parameters. By Lemma 7.11 we have that $s_D^{\mathbb{F}_p^b}$ is a subspace sampler.

Notice that the size of $s_D^{\mathbb{F}}$, is just the size of the space being sampled, $p^{b\mathsf{dim}}$, times $|D|$, which is $poly(p^b\mathsf{dim})$, giving a total size of $p^{b(\mathsf{dim}+O(1))}$ $poly(\mathsf{dim})$. □

7.4 Curve Samplers

Unfortunately, subspace samplers alone have too large of a sample size. Instead, we use curve samplers. To construct our curve samplers, we will first find a subspace sampler and choose a curve through that subspace. To do this, we start by showing that we can compose subspace samplers and curve samplers to get a new sampler.

Lemma 7.16 (Composing Subspace Sampler and Curve Sampler)**.** *Suppose that $\mathcal{S}$ is a time T_1, space S_1 uniform, unbiased, a-dimensional subspace sampler for $\mathbb{F}^{\mathsf{dim}}$ with accuracy error ϵ_1 and strong confidence error δ_1. Suppose that $\mathcal{C}'$ is a time T_2 space S_2 uniform an unbiased degree t curve sampler for $\mathbb{F}^a$ with accuracy error ϵ_1 and strong confidence error δ_2.*

Then let $\mathcal{C}$ be the set of functions

$$\mathcal{C} = \{s \circ c : c \in \mathcal{C}', s \in\}.$$

Then $\mathcal{C}$ is a degree t curve sampler for $\mathbb{F}^{\mathsf{dim}}$ with accuracy error $\epsilon_1 + \epsilon_2$ and strong *confidence error $\delta_1 + \delta_2$. Further the size of $\mathcal{C}$ is $|\mathcal{S}||\mathcal{C}'|$ and $\mathcal{C}$ is time $T_1 + T_2$ and space* $\max\{S_1, S_2\} + O(|\mathbb{F}|^a)$ *uniform.*

Proof. To show this, we just need to show for any set A with density μ that the probability a random curve in $\mathcal{C}$ oversamples A by more than $\epsilon_1 + \epsilon_2$ is at most $\delta_1 + \delta_2$. To show this, we will first exclude the subspaces in $\mathcal{S}$ that oversample A by more than ϵ_1. This only excludes $\delta_1\mu$ fraction of elements in $\mathcal{C}$. Then the remaining subspaces, on average, only intersect A on at most μ fraction of points. Then since $\mathcal{C}$ has strong confidence, on average the probability the $\mathcal{C}'$ further oversamples another ϵ_2 fraction of points is at most $\mu\delta_2$.

To make this more formal, take $A \subseteq \mathbb{F}^{\mathsf{dim}}$ such that $\mu = \frac{|A|}{|\mathbb{F}|^{\mathsf{dim}}}$. First define $\mathcal{S}'$ to be the set of subspaces that don't oversample A by more than an ϵ_1 fraction. That is,

$$\mathcal{S}' = \{s \in \mathcal{S} : \Pr_{i \in \mathbb{F}^a}[s(i) \in A] \leq \mu + \epsilon_1\}.$$

Now we can rewrite the confidence error of $\mathcal{C}$ in terms of $\mathcal{S}'$. See that

$$\begin{aligned}\Pr_{c' \in \mathcal{C}}[\Pr_{i \in \mathbb{F}}[c'(i) \in A] \geq \mu + \epsilon_1 + \epsilon_2] &= \Pr_{c \in \mathcal{C}', s \in \mathcal{S}}[\Pr_{i \in \mathbb{F}}[s(c(i)) \in A] \geq \mu + \epsilon_1 + \epsilon_2] \\ &\leq \Pr_{s \in \mathcal{S}}[s \notin \mathcal{S}'] + \Pr_{s \in \mathcal{S}', c \in \mathcal{C}'}[\Pr_{i \in \mathbb{F}^a}[s(c(i)) \in A] \geq \mu + \epsilon_1 + \epsilon_2].\end{aligned}$$

See that by the strong soundness error of $\mathcal{S}$ that $\Pr_{s \in \mathcal{S}}[s \notin \mathcal{S}'] \leq \mu\delta_1$.

Now we want to show that $\mathcal{S}'$ on average intersects A on at most μ fraction of places. See that since $\mathcal{S}$

is unbiased, we have that

$$\begin{aligned}\mu =&\frac{|A|}{|\mathbb{F}|^{\mathsf{dim}}}\\ =&\Pr_{s\in\mathcal{S},i\in\mathbb{F}^{\mathsf{dim}}}[s(i)\in A]\\ =&\Pr_{s\in\mathcal{S}}[s\in\mathcal{S}']\Pr_{s\in\mathcal{S}',i\in\mathbb{F}^{\mathsf{dim}}}[s(i)\in A]+\Pr_{s\in\mathcal{S}}[s\notin\mathcal{S}']\Pr_{s\in\mathcal{S}\setminus\mathcal{S}',i\in\mathbb{F}^{\mathsf{dim}}}[s(i)\in A]\\ =&\frac{|\mathcal{S}'|}{|\mathcal{S}|}\Pr_{s\in\mathcal{S}',i\in\mathbb{F}^{\mathsf{dim}}}[s(i)\in A]+\left(1-\frac{|\mathcal{S}'|}{|\mathcal{S}|}\right)\Pr_{s\in\mathcal{S}\setminus\mathcal{S}',i\in\mathbb{F}^{\mathsf{dim}}}[s(i)\in A]\\ \geq&\frac{|\mathcal{S}'|}{|\mathcal{S}|}\Pr_{s\in\mathcal{S}',i\in\mathbb{F}^{\mathsf{dim}}}[s(i)\in A]+\left(1-\frac{|\mathcal{S}'|}{|\mathcal{S}|}\right)(\mu+\epsilon_1).\end{aligned}$$

Now subtracting from both sides, we get that:

$$\begin{aligned}\mu-\left(1-\frac{|\mathcal{S}'|}{|\mathcal{S}|}\right)(\mu+\epsilon_1)\geq&\frac{|\mathcal{S}'|}{|\mathcal{S}|}\Pr_{s\in\mathcal{S}',i\in\mathbb{F}^{\mathsf{dim}}}[s(i)\in A]\\ \frac{|\mathcal{S}'|}{|\mathcal{S}|}\mu-\left(1-\frac{|\mathcal{S}'|}{|\mathcal{S}|}\right)\epsilon_1\geq&\frac{|\mathcal{S}'|}{|\mathcal{S}|}\Pr_{s\in\mathcal{S}',i\in\mathbb{F}^{\mathsf{dim}}}[s(i)\in A]\\ \mu\geq&\Pr_{s\in\mathcal{S}',i\in\mathbb{F}^{\mathsf{dim}}}[s(i)\in A].\end{aligned}$$

Now we can bound the probability that curves through subspaces in $\mathcal{S}'$ oversample A too much. By the strong confidence error of $\mathcal{C}$, we have that for any $s\in\mathcal{S}'$

$$\begin{aligned}\Pr_{c\in\mathcal{C}'}[\Pr_{i\in\mathbb{F}}[s(c(i))\in A]\geq\mu+\epsilon_1+\epsilon_2]\leq\Pr_{c\in\mathcal{C}'}[\Pr_{i\in\mathbb{F}}[s(c(i))\in A]\geq\Pr_{i\in\mathbb{F}^a}[s(i)\in A]+\epsilon_2]\\ \leq&\delta_2\Pr_{i\in\mathbb{F}^a}[s(i)\in A].\end{aligned}$$

Now we can show that

$$\begin{aligned}\Pr_{s\in\mathcal{S}',c\in\mathcal{C}'}[\Pr_{i\in\mathbb{F}^a}[s(c(i))\in A]\geq\mu+\epsilon_1+\epsilon_2]\leq&\delta_2\Pr_{s\in\mathcal{S}',i\in\mathbb{F}^a}[s(i)\in A]\\ \leq&\delta_2\mu.\end{aligned}$$

Thus we conclude that

$$\begin{aligned}\Pr_{c'\in\mathcal{C}}[\Pr_{i\in\mathbb{F}}[c'(i)\in A]\geq\mu+\epsilon_1+\epsilon_2]\leq&\mu\delta_1+\mu\delta_2\\ =&(\delta_1+\delta_2)\mu.\end{aligned}$$

See that functions in $\mathcal{C}$ are degree t curves since it is just the composition of degree one and a degree t polynomial. See that the time to compute an element of $\mathcal{C}$ is just the time to evaluate a curve in $\mathcal{C}'$ plus the time to evaluate an element of $\mathcal{S}$. Similarly the space is just the space to evaluate both functions, and this space can be reused except for the space to hold the output of the curve. □

Now we can compose together the efficient subspace sampler based on ε-biased sets, Corollary 7.15, with the naive curve sampler of all curves, Lemma 7.12, we can get an efficient curve sampler. Now we prove a generalization of Theorem 1.3.

Theorem 7.17 (Efficient Curve Sampler). *For any prime power p, integer $b\geq 2$, and integer $\mathsf{dim}\geq 1$, there is a unbiased, degree b-curve sampler $\mathcal{C}$ for $\mathbb{F}_p^{b\mathsf{dim}}$ such that for every $\epsilon>0$, it has accuracy error ϵ and strong confidence error*

$$2b\left(\frac{2b}{\epsilon\sqrt{p}}\right)^b.$$

Further $\mathcal{C}$ has size $p^{b(\mathsf{dim}+b+O(1))}\,poly(\mathsf{dim})$ and $\mathcal{C}$ is time and space $poly(p^b\mathsf{dim})$ uniform.

Proof. By Corollary 7.15, there is an unbiased b-dimensional subspace sampler, $\mathcal{S}$, for $\mathbb{F}_p^{b\mathsf{dim}}$ with size $p^{b(\mathsf{dim}+O(1))}\, poly(\mathsf{dim})$, with accuracy error $\epsilon/2$, and strong confidence error $\frac{8}{p^b\epsilon^2}$. Further $\mathcal{S}$ is time and space $poly(p^b\mathsf{dim})$ uniform.

By Lemma 7.12 there is an unbiased, degree b-curve sampler, $\mathcal{C}'$, for $\mathbb{F}^b$ with accuracy error $\epsilon/2$ and *strong* confidence error

$$\left(\frac{2b}{\epsilon\sqrt{|\mathbb{F}|}}\right)^b (b+1).$$

Further $\mathcal{C}'$ has size $p^{(b+1)b}$ and is time and space $b^2\, polylog(p)$ uniform.

Then by Lemma 7.16, for

$$\mathcal{C} = \{s \circ c : c \in \mathcal{C}', s \in \mathcal{S}\}$$

we have that $\mathcal{C}$ is a degree b curve sampler for $\mathbb{F}_p^{b\mathsf{dim}}$ with accuracy error ϵ and *strong* confidence error

$$\begin{aligned}
\left(\frac{2b}{\epsilon\sqrt{p}}\right)^b (b+1) + \frac{8}{p^b\epsilon^2} &\leq \left(\frac{2b}{\epsilon\sqrt{p}}\right)^b (b+1) + 8\left(\frac{1}{\sqrt{p}}\right)^{2b}\left(\frac{1}{\epsilon}\right)^b \\
&\leq \left(\frac{2b}{\epsilon\sqrt{p}}\right)^b (b+1) + \left(\frac{2b}{\sqrt{p}\epsilon}\right)^b \\
&\leq 2b\left(\frac{2b}{\epsilon\sqrt{p}}\right)^b.
\end{aligned}$$

Further the size of $\mathcal{C}$ is

$$p^{b(\mathsf{dim}+O(1))}\, poly(\mathsf{dim}) p^{(b+1)b} = p^{b(\mathsf{dim}+b+O(1))}\, poly(\mathsf{dim})$$

and $\mathcal{C}$ is time and space

$$poly(p^b\mathsf{dim}) + b^2\, polylog(p) = poly(p^b\mathsf{dim})$$

uniform. □

7.5 Explicit Time and Space Efficient Decoders For Reed-Muller Codes

Curve samplers give a randomness efficient way to correct Reed-Muller codes. The codewords of Reed-Muller codes are low degree polynomials, and low degree curves composed with low degree polynomials are also low degree polynomials. This suggests a local decoder. This decoder chooses a random curve through the point we want to decode. If the curve has little corruption along it, we can correct the low degree polynomial through that curve correctly. By choosing a curve from a curve sampler , we can correct most points correctly with high probability.

Now we show that if we have an appropriate curve sampler and an appropriate Reed-Muller code, then that curve sampler gives codeword improvers for the Reed-Muller code.

Lemma 7.18 (Curve Samplers Give Codeword Improvers)**.** *Let $\mathbb{F}$ be a field and dim be a number of dimensions. Suppose $\mathcal{C}$ is a unbiased, degree t curve sampler for $\mathbb{F}^{\mathsf{dim}}$ with accuracy error ϵ and* strong *confidence error δ.*

Then for any relative corruption μ, degree $\mathsf{deg} \leq \frac{|\mathbb{F}|}{t}(1-2(\epsilon+\mu))$, we have a $k = \frac{|\mathcal{C}|}{|\mathbb{F}|^{\mathsf{dim}-1}}$ element set of functions $\mathcal{I}$ that is an $|\mathbb{F}|$ query, below $d = \mu|\mathbb{F}|^{\mathsf{dim}}$, factor δ improving set for the Reed-Muller code $RM_{|\mathbb{F}|}(\mathsf{deg}, \mathsf{dim})$ (see Definition 2.9).

If $\mathcal{C}$ is time T space S uniform, then each $I \in \mathcal{I}$ is time $|\mathbb{F}|(T + polylog(|\mathbb{F}|))$, space $S + O(|\mathbb{F}|\log(|\mathbb{F}|))$ uniform.

Proof. We start by describing the improving set. Let our input be $p' : \mathbb{F}^{\mathsf{dim}} \to \mathbb{F}$ such that for some degree deg polynomial $p : \mathbb{F}^{\mathsf{dim}} \to \mathbb{F}$ we have that $\Delta(p, p') \leq d = \mu|\mathbb{F}|^{\mathsf{dim}}$. For $j \in [k]$, we will define the jth element of $\mathcal{I}$, which we will call I_j, as the output of the following algorithm.

1. On input $u \in |\mathbb{F}|^{\mathsf{dim}}$, define the multiset $C^u = \{c \in \mathcal{C} : \exists i \in \mathbb{F}, c(i) = u\}$ where the multiplicity is the number of $i \in \mathbb{F}$ such that $c(i) = u$, as in Definition 7.6. Since $\mathcal{C}$ is a unbiased curve sampler, we have that $|\mathcal{C}^u| = \frac{|\mathcal{C}||\mathbb{F}|}{|\mathbb{F}|^{\mathsf{dim}}} = k$. Similarly, we order $\mathcal{C}^u$ so that the jth element of $\mathcal{C}^u$, denoted as $\mathcal{C}_j^u$, can be computed in time T and space S.

   Let $g' : \mathbb{F} \to \mathbb{F}$ be the function defined by $g' = p' \circ \mathcal{C}_j^u$. Similarly denote $g : \mathbb{F} \to \mathbb{F}$ to be the polynomial $p \circ \mathcal{C}_j^u$. See that g is a degree $\mathsf{deg} \cdot t$ polynomial.

   We query p' at every point in the range of $\mathcal{C}_j^u$ to get g'.

2. Then we find the degree $\mathsf{deg} \cdot t$ polynomial $g^* : \mathbb{F} \to \mathbb{F}$ closest to g'. If $\mathcal{C}_j^u$ doesn't sample too much corruption, g^* will be g.

3. Finally, for whatever $i \in \mathbb{F}$ we have that $\mathcal{C}_j^u(i) = u$, we return $g^*(i)$ as the value for $I_j(p')_u$. If $g^* = g$, then $I_j(p')_u = g^*(i) = p(\mathcal{C}_j^u(i)) = p(u)$.

All we need to show now is that with probability at least $1 - \delta\mu$, we have that $g^* = g$. This is because $I_j(p')_u$ can only be different from $p(u)$ if $g^* \neq g$.

Since degree $\mathsf{deg} \cdot t$ polynomials over $\mathbb{F}$ are a code with distance $|\mathbb{F}| - \mathsf{deg} \cdot t$, we can only have $g^* \neq g$ if g' has distance at least $\frac{1}{2}(|\mathbb{F}| - \mathsf{deg} \cdot t)$ from g. Let A be the set of points that p and p' differ. See that g' will differ from g on at least $\frac{1}{2}(|\mathbb{F}| - \mathsf{deg} \cdot t)$ locations only if $\mathcal{C}_j^u$ samples A more then $\frac{1}{2}(|\mathbb{F}| - \mathsf{deg} \cdot t)$ times. Finally, see that a uniform $\mathcal{C}_j^u$ is also a uniform element of $\mathcal{C}$ since $\mathcal{C}$ is unbiased. Thus the probability that $g \neq g^*$ is at most

$$\begin{aligned}
\Pr_{u \in \mathbb{F}^{\mathsf{dim}}, j \in [k]}\left[\sum_{i \in \mathbb{F}} 1_{\mathcal{C}_j^u(i) \in A} \geq \frac{1}{2}(|\mathbb{F}| - \mathsf{deg} \cdot t)\right] &= \Pr_{c \in \mathcal{C}}\left[\Pr_{i \in \mathbb{F}}[c(i) \in A] \geq \frac{1}{2}\left(1 - \frac{\mathsf{deg} \cdot t}{|\mathbb{F}|}\right)\right] \\
&\leq \Pr_{c \in \mathcal{C}}\left[\Pr_{i \in \mathbb{F}}[c(i) \in A] \geq \frac{1}{2}(1 - 1 + 2(\epsilon + \mu))\right] \\
&\leq \Pr_{c \in \mathcal{C}}\left[\Pr_{i \in \mathbb{F}}[c(i) \in A] \geq \mu + \epsilon\right] \\
&\leq \delta\mu.
\end{aligned}$$

See the only queries made to p' are the points in $\mathcal{C}_j^u$, which is only $|\mathbb{F}|$ points. Similarly, the time is just the time to do decoding of the Reed-Solomon code, which is $|\mathbb{F}|\, polylog(|\mathbb{F}|)$, plus the time to evaluate $\mathcal{C}_j^u$ at every point, which is $|\mathbb{F}|T$. Similarly for space. ☐

We established in Lemma 5.6 that just finding a uniform improving set is enough to give a deterministic decoder. Now that we know explicit curve samplers give us uniform improving sets for Reed-Muller codes, and we have improving sets, we show that Reed-Muller codes have explicit decoders.

Theorem 7.19 (Deterministic Correctors For Reed-Muller From Curve Samplers)**.** *Take any prime power p, integer $b \geq 2$, and integer $\mathsf{dim} \geq 1$. Let $m = p^{b\mathsf{dim}}$.*

Then for any accuracy error ϵ, relative corruption μ, and degree $\mathsf{deg} \leq \frac{p}{b}(1-2(\epsilon+\mu))$ where $\epsilon p^{1/4} \geq 2b$, the code $RM_p(\mathsf{deg}, b\mathsf{dim})$ has a uniform, deterministic corrector with correcting radius $d/3 = \mu p^{b\mathsf{dim}}/3$ running in time

$$m m^{\frac{8}{b}}\, poly(p^{b^2}\mathsf{dim})$$

and space

$$poly(p^b\mathsf{dim}).$$

Proof. From Theorem 7.17, there is an unbiased, degree b-curve sampler $\mathcal{C}$ for $\mathbb{F}_p^{b\mathsf{dim}}$ with accuracy error ϵ and strong confidence error

$$\delta = 2b\left(\frac{2b}{\sqrt{p}\epsilon}\right)^b.$$

Further $\mathcal{C}$ has size $p^{b(\mathsf{dim}+b+O(1))}\, poly(\mathsf{dim})$ and $\mathcal{C}$ is time and space $poly(p^b\mathsf{dim})$ uniform.

Now we need to simplify the expression for δ. See that

$$\begin{aligned}\delta =&2b\left(\frac{2b}{\sqrt{p}\epsilon}\right)^b\\ \le& p^{1/4}\frac{1}{p^{b/4}}\\ \le& p^{-b/8}.\end{aligned}$$

Also, since $b > 2$, we also have that $p^{1/4} > 4$, so $p^{-b/8} < \frac{1}{4}$ and thus $\delta \le \frac{1}{4}$.

From Lemma 7.18, we have a $k = \frac{p^{b(\mathsf{dim}+b+O(1))}\,poly(\mathsf{dim})}{p^{b\mathsf{dim}-1}} = p^{b(b+O(1))}\,poly(\mathsf{dim})$ element set of functions $\mathcal{I}$ that is a p query, below $d = \mu p^{b\mathsf{dim}}$, factor $\eta = \delta$ improving set for $RM_p(\mathsf{deg}, b\mathsf{dim})$ that are all time and space $T = S = poly(p^b\mathsf{dim})$ uniform.

Set $\ell = \lceil \frac{\log(d/3+1)}{\log((1-\eta)^2/\eta)} \rceil$. From Lemma 5.6 the code C has a deterministic corrector with correcting radius $d/3$ running in time

$$\begin{aligned}O(m\ell|\mathcal{I}|^2 q^{\ell+1}T) =&O\left(mT\frac{\log(d/3+1)}{\log((1-\eta)^2/\eta)}\left(p^{b(b+O(1))}\,poly(\mathsf{dim})\right)^2 p^{\frac{\log(d/3+1)}{\log((1-\eta)^2/\eta)}+2}\right)\\ \le& O\left(m\,poly(p^{b^2}\mathsf{dim})\log(m)p^{\frac{\log(m)}{\log(1/\delta)-2\log(1-\delta)}+2}\right)\\ \le& O\left(m\,poly(p^{b^2}\mathsf{dim})m^{\frac{\log(p)}{\log(1/\delta)}}\right)\\ \le& O\left(m\,poly(p^{b^2}\mathsf{dim})m^{\frac{\log(p)}{\log(p^{b/8})}}\right)\\ =& mm^{\frac{8}{b}}\,poly(p^{b^2}\mathsf{dim})\end{aligned}$$

and space

$$\begin{aligned}O(\ell\min\{S, q\log(|\mathbb{F}|)\} + \ell\log(|\mathcal{I}|) + \log(m) + S) =&O\left(S + \frac{\log(d/3+1)}{\log((1-\eta)^2/\eta)}(p + \log(|\mathcal{C}|)) + \log(m)\right)\\ \le& O\left(poly(p^b\,\mathrm{dim}) + \log(m)(p + b(\mathrm{dim} + b)\log(p\,poly(\mathrm{dim})))\right).\\ \le& poly(p^b\,\mathrm{dim}).\end{aligned}$$

□

Now to turn this into a decoder, we just observe that the Reed-Muller code is linear, thus can also be made systematic. So our decoder just runs the corrector and only outputs the symbols that are equal to the message. Now we apply Theorem 7.19 to special cases of the Reed-Muller codes.

Theorem 7.20 (*Uniform* Time And Space Efficient Decoders For Reed-Muller)**.** *For any constant $\gamma > 0$, there exists an infinite family of codes $C : \{0,1\}^n \to \{0,1\}^m$ where $m = O_\gamma(n)$ and a deterministic uniform algorithm B that computes a function $D : \{0,1\}^m \to \{0,1\}^n$ such that:*

Efficient: *B runs in time $n^{1+\gamma}$ and space n^γ.*

Decodes: *For some decoding radius $d = \Omega_\gamma(n)$, for any $x \in \{0,1\}^n$ and $w \in \{0,1\}^m$ with $\Delta(w, C(x)) \le d$ we have that*

$$D(w) = x.$$

Proof. This code is just a Reed-Muller code with carefully chosen parameters so that when we apply Theorem 7.19 we get the desired results. For any large enough integer deg we will construct one code in this family. The degree will be a small polynomial in the message length.

Specifically, for some constant c, we will set $b = \frac{c}{\gamma}$. The constant c depends on the constants in the polynomials of Theorem 7.19. We set $\mathsf{dim} = b^2$, as well as $\epsilon = \mu = \frac{1}{4}$ and prime p to some prime between $2b\mathsf{deg}$ and $4b\mathsf{deg}$. Then see that for large enough deg that $2b \le \epsilon p^{1/4}$ and $\mathsf{deg} \le \frac{p}{2b} = \frac{p}{b}(1 - 2(\epsilon + \mu))$.

Now we show that $RM_p(\mathsf{deg}, b\mathsf{dim})$ has constant rate. See that the size of a codeword in this code is

$$m = p^{b\mathsf{dim}} \leq (4b)^{b\mathsf{dim}} \mathsf{deg}^{b\mathsf{dim}}$$

and the size of a message in this code (which is the number of monomials of degree at most deg) is at least

$$n \geq \left(\frac{\mathsf{deg}}{b\mathsf{dim}}\right)^{b\mathsf{dim}} = \mathsf{deg}^{b\mathsf{dim}} (b\mathsf{dim})^{-b\mathsf{dim}}.$$

Since b and dim are both constants, the gap between the message length and codeword length is only a constant factor. So the rate is constant.

We note that

$$\begin{aligned} m \geq & n \geq \left(\frac{\mathsf{deg}}{b\mathsf{dim}}\right)^{b\mathsf{dim}} \\ \mathsf{deg} \leq & b\mathsf{dim} m^{1/(b\mathsf{dim})} \\ \leq & b\mathsf{dim} m^{1/b^3} \\ = & O(m^{\gamma/(b^2 c)}). \end{aligned}$$

Then by Theorem 7.19, the code $RM_p(\mathsf{deg}, b\mathsf{dim})$ has a uniform, deterministic corrector with relative correcting radius $\mu/3$ running in time

$$\begin{aligned} m m^{\frac{8}{b}} \, poly(p^{b^2} \mathsf{dim}) = & O(m^{1+\frac{8\gamma}{c}} \, poly((4b)^{b^2} \mathsf{deg}^{b^2})) \\ = & O(m^{1+\frac{8\gamma}{c}} \, poly(m^{\gamma/c})) \\ = & O(m \, poly(m^{\gamma/c})) \end{aligned}$$

and space

$$\begin{aligned} poly(p^b \mathsf{dim}) = & O(poly((4b)^b \mathsf{deg}^b)) \\ = & O(poly(m^{\gamma/(bc)})). \end{aligned}$$

As long as c is larger then the exponent of the polynomial, our claimed result holds. Just let c be such a constant so that b is also an integer.

Then since the code has constant relative decoding radius, it has constant relative distance. Since it also has constant rate, it is an asymptotically good code. Since n increases as deg increases, it is an infinite family of good codes.

We also note that this code could be made binary by concatenating it with a small, efficiently encodable and decodable asymptotically good binary code. If we need a code for some arbitrary message length n, this can be achieved by choosing an appropriate degree. Since the number of bits in a message for a given degree is

$$n_{\mathsf{deg}} = \Theta(\mathsf{deg}^{b\mathsf{dim}} \log(p)) = \Theta(\mathsf{deg}^{b\mathsf{dim}} \log(\mathsf{deg}))$$

we have that every n has a degree deg so that n_{deg} is greater than n and within a constant factor of n (for instance, some power of 2 degree would achieve this).

To turn correcting into decoding, we can make the encoding systematic so the message symbols are exactly some specific set of codeword symbols. More explicitly, view the encoding function as first fixing appropriate codeword symbols and performing polynomial interpolation. These symbols of the codeword $f : \mathbb{F}^{b\mathsf{dim}} \to \mathbb{F}$ are just the evaluations of f on $x_1, x_2, \ldots, x_{b\,\mathrm{dim}}$ such that $\sum_{i \in [b\,\mathrm{dim}]} x_i \leq \mathsf{deg}$.

□

8 Uniform Decoding For Lifted Reed-Solomon Codes

Finally, we give asymptotically good codes with deterministic, uniform correctors that run in $n^{o(1)}$ space and $n^{1+o(1)}$ time. These codes are based on lifted Reed-Solomon codes [GKS13].

Definition 8.1 (Lifted Reed-Solomon Codes)**.** *For a field $\mathbb{F}$, dimension dim, and degree deg, the lifted Reed-Solomon code is the code whose codewords are the set of functions $f : \mathbb{F}^{\mathsf{dim}} \to \mathbb{F}$ such that for every line $l : \mathbb{F} \to \mathbb{F}^{\mathsf{dim}}$ we have that $f \circ l$ is a polynomial of degree at most deg.*

This is a code with alphabet $\mathbb{F}$ and codeword length $|\mathbb{F}^{\mathsf{dim}}|$ if we view f as a truth table.

For prime order fields or low degree polynomials, lifted Reed-Solomon codes are equivalent to Reed-Muller codes, but for fields whose order is a prime power and for very high degree, lifted Reed-Solomon codes can have much higher rate, even approaching one. The following is from the proof of [GKS13, Theorem 1.5].

Lemma 8.2 (Lifted Reed-Solomon Codes have High Rate)**.** *For any field $\mathbb{F}$ of characteristic 2 and dimension dim for some $c = O(\mathsf{dim}^{2\mathsf{dim}})$ and $\mathsf{deg} = (1 - 2^{-c})|\mathbb{F}|$ we have that the lifted Reed-Solomon code of dimension dim and degree deg over field $\mathbb{F}$ has rate at least $1/2$.*

Lifted Reed-Solomon codes are also explicit systematic codes. In particular, we can efficiently compute which symbols of the lifted Reed-Solomon codes are message bits. The following is a corollary of [GK16, Theorem A.1].

Lemma 8.3 (Lifted Reed-Solomon Codes Are Uniformly Systematic)**.** *For a field $\mathbb{F}$, dimension dim, and degree deg, let C be the lifted Reed-Solomon code of degree deg for $\mathbb{F}^{\mathsf{dim}}$. Let n be the message length of C. Then there is an encoding function for C, which we will denote as $C : \mathbb{F}^n \to (\mathbb{F}^{\mathsf{dim}} \to \mathbb{F})$, such that there is a time and space poly$(\log(|\mathbb{F}|)\mathsf{dim})$ computable function $g : [n] \to \mathbb{F}^{\mathsf{dim}}$ such that for any $i \in [n]$ and $x \in \mathbb{F}^n$ we have that $C(x)(g(i)) = x_i$.*

8.1 Correcting Lines Set

Unfortunately, we cannot use curves to decode lifted Reed-Solomon codes, we have to use lines. But line samplers whose samples are one line do not have low enough confidence error to give us good enough local correction. Since individual lines are not enough, our local corrector chooses a small number of lines, decodes along each of them, and then outputs the most often decoded value.

We define correcting lines for a point x as any set of lines that start at x. Our goal is to have a set of lines such that most do not over sample any set.

Definition 8.4 (Correcting Lines for x)**.** *Let dim, c be natural numbers, take $\epsilon > 0$, take $\mathbb{F}$ to be a field, take $x \in \mathbb{F}^{\mathsf{dim}}$. $\mathcal{C}_x$ forms c* correcting lines *for x if $\mathcal{C}_x = (\ell_1, \dots, \ell_c)$, where $\ell_1, \dots, \ell_c$ are lines in $\mathbb{F}^{\mathsf{dim}}$ such that for all $i \in [c]$ we have that $\ell_i(0) = x$.*

For any set $A \subseteq \mathbb{F}^{\mathsf{dim}}$, we say that $\mathcal{C}_x$ is ϵ-sampling for A if

$$\Pr_{i \in [c]}\left[\Pr_{\lambda \in \mathbb{F}}[\ell_i(\lambda) \in A] \geq \frac{|A|}{|\mathbb{F}^{\mathsf{dim}}|} + \epsilon\right] < 1/2.$$

To actually correct our code, we will need many sets of correcting lines for each x such that for any small enough set A, most correcting lines for most x are ϵ-sampling for A.

Definition 8.5 (Correcting Lines Set)**.** *Let dim, c, k be natural numbers and take $\mathbb{F}$ to be a field. A* correcting lines set *is a family $\mathcal{C} = \{\mathcal{C}_{x,i} : x \in \mathbb{F}^{\mathsf{dim}}, i \in [k]\}$, where each $\mathcal{C}_{x,i}$ forms c correcting lines for x. We say $\mathcal{C}$ forms below d correcting lines set with strong confidence error δ, and accuracy error ϵ if for all $A \subseteq \mathbb{F}^{\mathsf{dim}}$ such that $|A| \leq d$, we have that*

$$\Pr_{x \in \mathbb{F}^{\mathsf{dim}}, i \in [k]}[\mathcal{C}_{x,i} \text{ is not } \epsilon\text{-sampling for } A] \leq \delta \frac{|A|}{|\mathbb{F}^{\mathsf{dim}}|}.$$

We say $\mathcal{C}$ is time T and space S uniform if for each $x \in \mathbb{F}^{\mathsf{dim}}$ and $i \in [k]$ we have that every $\ell \in \mathcal{C}_{x,i}$ can be computed in time T and space S. We call c the line count of $\mathcal{C}$.

A correcting lines set gives a straightforward way to decode lifted Reed-Solomon codes. Specifically, it gives an efficient codeword improving set.

Lemma 8.6 (Improving Set from a Correcting Lines Set)**.** *Suppose that for some dimension* dim *and field* $\mathbb{F}$ *we have* $\mathcal{C}$ *that is a time* T *space* S *uniform, below* d *correcting lines set for* $\mathbb{F}^{\mathsf{dim}}$ *with size* $|\mathcal{C}|$*, strong confidence error* δ*, accuracy error* ϵ*, and line count* c*.*

Let LRS *be the lifted Reed-Solomon code for* $\mathbb{F}^{\mathsf{dim}}$ *with degree* $\mathsf{deg} = \left(1 - 2\left(\frac{d}{|\mathbb{F}^{\mathsf{dim}}|} + \epsilon\right)\right)|\mathbb{F}|$*. Then there is a* $c(|\mathbb{F}|-1)+1$ *query, time* $c(T + \tilde{O}(\mathsf{dim}|\mathbb{F}|))$ *space* $S + O((\mathsf{dim}+c)\log(|\mathbb{F}|)) + \tilde{O}(|\mathbb{F}|))$ *below* d*, factor* δ *improving set for* LRS *with size* $|\mathcal{C}|/|\mathbb{F}^{\mathsf{dim}}|$*.*

Proof. Recall from Definition 8.5 that $\mathcal{C} = \{\mathcal{C}_{x,i} : x \in \mathbb{F}^{\mathsf{dim}}, i \in [k]\}$ where $k = |\mathcal{C}|/|\mathbb{F}^{\mathsf{dim}}|$. Then for $i \in [k]$, we define the function $\mathcal{I}_i$ as the function such that for any $x \in \mathbb{F}^{\mathsf{dim}}$ and $f : \mathbb{F}^{\mathsf{dim}} \to \mathbb{F}$ we do the following to compute $\mathcal{I}_i(f)(x)$. For each line $\ell \in \mathcal{C}_{x,i}$, first query f at every point in the line ℓ so we have $f \circ \ell$. Then decode $f \circ \ell$ to the nearest degree deg polynomial, denote that degree deg polynomial as f'_ℓ. Finally, output whatever symbol $\lambda \in \mathbb{F}$ maximizes $\Pr_{\ell \in \mathcal{C}_{x,i}}[f'_\ell(0) = \lambda]$. That is, we decode along each line in $\mathcal{C}_{x,i}$ and take the majority vote of what the corrected $f(\ell(0)) = f(x)$ should be.

See that by definition, $\mathcal{I} = \{\mathcal{I}_i : i \in [k]\}$ has the correct size, and uses $c(|\mathbb{F}|-1)+1$ queries. Further since there is fast polynomial interpolation that we only need to run c times, we have the desired time to calculate $\mathcal{I}$. For space, see we can reuse the space every time we need to retrieve a new line, and besides that we only need enough space to hold the current line ℓ, for polynomial interpolation, and the space to store the current votes of what $f(x)$ should be.

To show that $\mathcal{I}$ is a below d, factor δ improving set for LRS, we first choose a function $f' : \mathbb{F}^{\mathsf{dim}} \to \mathbb{F}$ that is a codeword of LRS and a function $f : \mathbb{F}^{\mathsf{dim}} \to \mathbb{F}$ such that $\Delta(f, f') \leq d$. Then we need to show that $\mathbb{E}_{I \in \mathcal{I}}[\Delta(I(f), f')] \leq \delta\Delta(f, f')$.

Let A be the set of $y \in \mathbb{F}^{\mathsf{dim}}$ such that $f(y) \neq f'(y)$. See that $|A| = \Delta(f, f') \leq d$. Since $\mathcal{C}$ is a below d correcting lines set for $\mathbb{F}^{\mathsf{dim}}$ with strong confidence error δ and accuracy error ϵ, we have that

$$\Pr_{x \in \mathbb{F}^{\mathsf{dim}}, i \in [k]}[\mathcal{C}_{x,i} \text{ is not } \epsilon\text{-sampling for } A] \leq \delta \frac{|A|}{|\mathbb{F}^{\mathsf{dim}}|}.$$

Recall that $\mathcal{C}_{x,i}$ is ϵ-sampling for A if and only if

$$\Pr_{\ell \in \mathcal{C}_{x,i}}\left[\Pr_{\lambda \in \mathbb{F}}[\ell(\lambda) \in A] \geq \frac{|A|}{|\mathbb{F}|^{\mathsf{dim}}} + \epsilon\right] < 1/2.$$

In particular, if $\mathcal{C}_{x,i}$ is ϵ-sampling, for at least half of the $\ell \in \mathcal{C}_{x,i}$ we have that less than $\frac{|A|}{|\mathbb{F}|^{\mathsf{dim}}} + \epsilon < \frac{d}{|\mathbb{F}^{\mathsf{dim}}|} + \epsilon$ fraction of the range of ℓ lies in A. Take such an ℓ. See that $\Delta(f' \circ \ell, f \circ \ell) < \left(\frac{d}{|\mathbb{F}^{\mathsf{dim}}|} + \epsilon\right)|\mathbb{F}|$. Since f' restricted to ℓ has degree at most $\mathsf{deg} = \left(1 - 2\left(\frac{d}{|\mathbb{F}^{\mathsf{dim}}|} + \epsilon\right)\right)|\mathbb{F}|$, and the distance between any two degree deg polynomials over $|\mathbb{F}|$ is at least $2\left(\frac{d}{|\mathbb{F}^{\mathsf{dim}}|} + \epsilon\right)|\mathbb{F}|$, we have that $f'_\ell = f' \circ \ell$. Thus since the majority of $\ell \in \mathcal{C}_{x,i}$ output the correct value of f', we have that $\mathcal{I}_i(f)(x) = f'(x)$. Thus if $\mathcal{C}_{x,i}$ is good up to accuracy error ϵ with respect to A, then $\mathcal{I}_i(f)(x) = f'(x)$.

This implies that

$$\begin{aligned}\Pr_{x \in \mathbb{F}^{\mathsf{dim}}, i \in [k]}[\mathcal{I}_i(f)(x) \neq f'(x)] \leq &\Pr_{x \in \mathbb{F}^{\mathsf{dim}}, i \in [k]}[\mathcal{C}_{x,i} \text{ is not } \epsilon\text{-sampling for } A] \\ \leq &\delta \frac{|A|}{|\mathbb{F}^{\mathsf{dim}}|} \\ \leq &\delta \frac{\Delta(f, f')}{|\mathbb{F}^{\mathsf{dim}}|}.\end{aligned}$$

Rearranging, we get

$$\begin{aligned}\mathop{\mathbb{E}}_{i \in [k]}\left[|\mathbb{F}^{\mathsf{dim}}| \Pr_{x \in \mathbb{F}^{\mathsf{dim}}}[\mathcal{I}_i(f)(x) \neq f'(x)]\right] \leq &\delta\Delta(f, f') \\ \mathop{\mathbb{E}}_{i \in [k]}[\Delta(\mathcal{I}_i(f), f')] \leq &\delta\Delta(f, f').\end{aligned}$$

Therefore $\mathcal{I}$ is a below d, factor δ improving set for LRS. ☐

While less elegant than curve samplers, correcting lines sets give another way to correct Reed-Muller codes with a better query-distance trade off than subspace samplers. More importantly, they give an efficient way to correct lifted Reed-Solomon codes. Similar to curve samplers, we can compose correcting lines sets with subspace samplers to get even better correcting lines sets.

Lemma 8.7 (Subspace Sampler Correcting Lines Set Composition)**.** *Suppose that $\mathcal{S}$ is a time T_1, space S_1 uniform, unbiased, a-dimensional subspace sampler for $\mathbb{F}^{dim}$ with accuracy error ϵ_1 and strong confidence error δ_1. Suppose for some integer d that $\mathcal{C}'$ is a time T_2 space S_2 uniform, below $\left(\frac{d}{|\mathbb{F}^{dim}|}+\epsilon_1\right)|\mathbb{F}^a|$ correcting lines set for $\mathbb{F}^a$ with accuracy error ϵ_2, strong confidence error δ_2 with line count c.*

Then there is a below d correcting lines set, $\mathcal{C}$, for $\mathbb{F}^{dim}$ with accuracy error $\epsilon_1+\epsilon_2$, strong *confidence error $\delta_1+\delta_2$, and line count c. Further the size of the final correcting lines set is $|\mathcal{C}|=|\mathcal{S}||\mathcal{C}'|$ and $\mathcal{C}$ is time $O(T_1|\mathbb{F}^a|+T_2)$ and space* $\max\{S_1,S_2\}+O(a\log(|\mathbb{F}|))$ *uniform.*

Proof. For $u\in\mathbb{F}^{\mathsf{dim}}$, define $\mathcal{S}^u=\{s\in\mathcal{S}:\exists v\in\mathbb{F}^a:s(v)=u\}$ as we did in Definition 7.4. Since $\mathcal{S}$ is unbiased $k_1=\frac{|\mathcal{S}||\mathbb{F}^a|}{|\mathbb{F}^{\mathsf{dim}}|}=|\mathcal{S}^u|$. Since $\mathcal{S}$ is uniform, there is some ordering of $\mathcal{S}$ such that for all $j\in[k_1]$ we have that $\mathcal{S}^u_j$ can be evaluated in time T_1 and space S_1. Similarly, for $k_2=\frac{|\mathcal{C}'|}{|\mathbb{F}^a|}$ we have that $\mathcal{C}'$ can be indexed by $v\in\mathbb{F}^a$ and $i\in[k_2]$ as $\mathcal{C}'_{v,i}$.

Our new correcting lines set will be indexed by $u\in\mathbb{F}^{\mathsf{dim}}$ and $i\in[k_1k_2]$. Interpret i as a pair $i_1\in[k_1]$ and $i_2\in[k_2]$. Then we construct $\mathcal{C}$ as

$$\mathcal{C}_{u,i}=\{\mathcal{S}^u_{i_1}\circ\ell:\ell\in\mathcal{C}'_{(\mathcal{S}^u_{i_1})^{-1}(u),i_2}\}.$$

By definition, $\mathcal{C}'$ has the appropriate size, and line count. To compute this, one needs to just evaluate $\mathcal{S}^u_{i_1}$ for $|\mathbb{F}^a|$ many times to calculate $(\mathcal{S}^u_{i_1})^{-1}(u)$, then evaluate $\mathcal{C}'$ and and $\mathcal{S}$ once more. This requires time $O(T_1|\mathbb{F}^a|+T_2)$. This only requires the space to calculate either $\mathcal{C}'$ or $\mathcal{S}$ plus some intermediate space to store a constant number of elements of $|\mathbb{F}|^a$, which is space $\max\{S_1,S_2\}+O(a\log(|\mathbb{F}|))$.

Notice that we can also index the correcting lines in $\mathcal{C}$ in an alternative way. First choose the subspace $s\in\mathcal{S}$ and then choose correcting lines for that subspace $s'\in\mathcal{C}'$ to get the new correcting lines $\mathcal{C}^*_{s,s'}=\{s\circ\ell:\ell\in s'\}$. This gives the same set of correcting lines as $\mathcal{C}$.

To show that $\mathcal{C}$ is a correcting lines set, take any set $A\subseteq\mathbb{F}^{\mathsf{dim}}$ with $|A|\le d$ and set $\mu=\frac{|A|}{|\mathbb{F}^{\mathsf{dim}}|}$. Now we want to restrict ourselves to the subspaces that don't oversample A. Define

$$\mathcal{S}_g=\{s\in\mathcal{S}:\Pr_{v\in\mathbb{F}^a}[s(v)\in A]<\mu+\epsilon_1\}.$$

Then since $\mathcal{S}$ is a sampler with accuracy error ϵ_1 and strong confidence error δ_1, we have that

$$\Pr_{s\in\mathcal{S}}[s\notin\mathcal{S}_g]\le\delta_1\mu.$$

For any $s\in\mathcal{S}_g$ define $A_s=\{v\in\mathbb{F}^a:s(v)\in A\}$ and $\mu_s=\frac{|A_s|}{|\mathbb{F}^a|}$. See that $|A_s|\le|\mathbb{F}^a|(\mu+\epsilon_1)\le|\mathbb{F}_a|\left(\frac{d}{|\mathbb{F}^{\mathsf{dim}}|}+\epsilon_1\right)$. Since $\mathcal{C}'$ is a below $\left(\frac{d}{|\mathbb{F}^{\mathsf{dim}}|}+\epsilon_1\right)|\mathbb{F}^a|$ correcting lines set for $\mathbb{F}^a$, we have that

$$\Pr_{s'\in\mathcal{C}'}[s' \text{ is not } \epsilon_2\text{-sampling for } A_s]\le\delta_2\mu_s.$$

In particular, see that

$$\begin{aligned}&\Pr_{s'\in\mathcal{C}'}[s' \text{ is not } \epsilon_2\text{-sampling for } A_s]\\&=\Pr_{s'\in\mathcal{C}'}\left[\Pr_{\ell\in s'}\left[\Pr_{\lambda\in\mathbb{F}}[\ell(\lambda)\in A_s]>\mu'+\epsilon_2\right]\ge 1/2\right]\\&\ge\Pr_{s'\in\mathcal{C}'}\left[\Pr_{\ell\in s'}\left[\Pr_{\lambda\in\mathbb{F}}[\ell(\lambda)\in A_s]>\mu+\epsilon_1+\epsilon_2\right]\ge 1/2\right].\end{aligned}$$

Thus

$$\Pr_{s'\in\mathcal{C}'}\left[\Pr_{\ell\in s'}\left[\Pr_{\lambda\in\mathbb{F}}[\ell(\lambda)\in A_s]>\mu+\epsilon_1+\epsilon_2\right]\ge 1/2\right]\le\delta_2\mu_s.$$

Finally, see that

$$\begin{aligned}
&\Pr_{u\in\mathbb{F}^{\mathsf{dim}},i\in[k]}[\mathcal{C}_{x,i} \text{ is not } (\epsilon_1+\epsilon_2)\text{-sampling for } A]\\
=&\Pr_{s\in\mathcal{S},s'\in\mathcal{C}}[\mathcal{C}^*_{s,s'} \text{ is not } (\epsilon_1+\epsilon_2)\text{-sampling for } A]\\
\leq&\Pr_{s\in\mathcal{S}}[s\notin\mathcal{S}_g]+\\
&\Pr_{s\in\mathcal{S},s\in\mathcal{C}'}\left[s\in\mathcal{S}_g\wedge\Pr_{\ell\in s'}\left[\Pr_{\lambda\in\mathbb{F}}[\ell(\lambda)\in A_s]>\mu+\epsilon_1+\epsilon_2\right]\geq 1/2\right]\\
\leq&\delta_1\mu+\\
&\mathop{\mathbb{E}}_{s\in S}[\delta_2\mu_s]\\
=&(\delta_1+\delta_2)\mu.
\end{aligned}$$

□

8.2 Correcting Lines Set From Curve Samplers

We can get a correcting lines set using any good enough sampler. The sampler will sample the set of lines that go through a point. For any set A, most lines will not oversample A, so a sampler for these lines will rarely choose too many lines that oversample A.

Lemma 8.8 (Correcting Lines Set From Sampler)**.** *Take any dimension a and field $\mathbb{F}$. Let $n=\frac{|\mathbb{F}^a|-1}{|\mathbb{F}|-1}$. Suppose that for some $\epsilon',\delta>0$ there is a time T and space S uniform sampler, $\mathsf{samp}:[m]\to[n]^q$, for $[n]$ with size m, sample size q, accuracy error ϵ', and strong confidence error δ.*

Then there is a set $\mathcal{C}$ such that for any ϵ we have that $\mathcal{C}$ is a time $T+\tilde{O}(a\log(|\mathbb{F}|))$ and space $S+\tilde{O}(a\log(|\mathbb{F}|))$ uniform, below $\epsilon\frac{1-2\epsilon'}{1+2\epsilon'}|\mathbb{F}^a|$ correcting lines set for $\mathbb{F}^a$ with line count q, accuracy error ϵ and strong confidence error δ/ϵ. Further $\mathcal{C}$ has size $|\mathcal{C}|=m|\mathbb{F}^a|$.

Proof. Associate each $j\in[n]$ with some direction and let $\mathsf{dir}:[n]\to\mathbb{F}^a$ be the function that converts $j\in[n]$ to a corresponding direction. Then for any $x\in\mathbb{F}^a$ and $i\in[m]$ we calculate $\mathcal{C}_{x,i}$ by running samp on i to get a set of directions to take the lines through x. Specifically,

$$\mathcal{C}_{x,i}=\left(\ell_{x,\mathsf{dir}(\mathsf{samp}(i)_j)}\right)_{j\in[q]}$$

where $\ell_{x,v}(\lambda)=x+v\lambda$ as in Definition 7.9. Let $\mathcal{C}=\{\mathcal{C}_{x,i}:x\in\mathbb{F}^a,i\in[m]\}$.

Now to show that $\mathcal{C}$ is a correcting lines set, we choose a set $A\subseteq\mathbb{F}^a$ such that $|A|\leq\epsilon\frac{1-2\epsilon'}{1+2\epsilon'}|\mathbb{F}^a|$. Define $\mu=\frac{|A|}{|\mathbb{F}^a|}\leq\epsilon\frac{1-2\epsilon'}{1+2\epsilon'}$. Now we will show that for any $x\in\mathbb{F}^a$ the probability over $i\in[m]$ that $\mathcal{C}_{x,i}$ is not ϵ-sampling for A is at most $\mu\delta/\epsilon$. So choose any $x\in\mathbb{F}^a$. Let A'_x be the set of lines starting at x such that they intersect A on more than a $\mu+\epsilon$ fraction of points. See that if $\mathsf{samp}(i)$ samples less than $1/2$ fraction of lines from A'_x than $\mathcal{C}_{x,i}$ is ϵ-sampling for A. By a Markov inequality, the fraction of lines in A'_x is at most $\mu'=\frac{\mu}{\mu+\epsilon}$. See that

$$\begin{aligned}
\frac{\mu}{\mu+\epsilon}+\epsilon'&\leq\frac{\epsilon\frac{1-2\epsilon'}{1+2\epsilon'}}{\epsilon\frac{1-2\epsilon'}{1+2\epsilon'}+\epsilon}+\epsilon'\\
&=\frac{1-2\epsilon'}{1-2\epsilon'+1+2\epsilon'}+\epsilon'\\
&=\frac{1}{2}-\epsilon'+\epsilon'\\
&=\frac{1}{2}.
\end{aligned}$$

Since samp is a sampler, the probability that the samp samples at least $1/2$ fraction of lines from A'_x is at most $\delta\frac{\mu}{\mu+\epsilon}\leq\mu\delta/\epsilon$. Thus $\mathcal{C}$ is a correcting lines set with accuracy error ϵ and strong confidence error δ/ϵ. □

In particular, we do have a good sampler with small query complexity and strong confidence error: our curve samplers. The only issue with our curve samplers is that they sample a space of size that is a prime power, but n in Lemma 8.8 is not a prime power. This can be solved through a padding argument.

Lemma 8.9 (Padding Samplers)**.** *Suppose we have a sampler,* $\mathsf{samp}' : [m] \to [n']^q$*, with accuracy error* ϵ_1 *and strong confidence error* δ_1*. Then choose any integer* k *and constant* $1 > \eta \geq 0$ *such that* $n = \frac{n'}{k+\eta}$ *is an integer. Then define* $\mathsf{samp} : [m] \to [n]^q$ *such that for* $i \in [q]$ *we have that* $\mathsf{samp}(x)_i = \mathsf{samp}'(x)_i \mod n$*. Then* samp' *is a sampler with accuracy error* $\epsilon_1 + \eta/k$ *and strong confidence error* $\delta(1 + 1/k)$*.*

Proof. Take any $A \subseteq [n]$ and set $\mu = \frac{|A|}{n}$. Now consider the set $A' \subseteq [n']$ of integers $i \in [n']$ such that $i \mod n \in A$. See that $n' = n(k + \eta)$, so $(1 - \eta)$ fraction of $i \in [n]$ have k elements in $[n']$ that map to them, but η fraction of elements have $k + 1$ elements map to them. So $|A'| \leq k|A| + \min\{\eta n, |A|\}$. Let $\mu' = \frac{|A'|}{n'}$. Then we have that

$$\begin{aligned}\mu' =& \frac{|A'|}{n'} \\ \leq& \frac{k|A| + \eta n}{n(k+\eta)} \\ \leq& \frac{k|A| + \eta n}{nk} \\ =& \mu + \frac{\eta}{k},\end{aligned}$$

and that

$$\begin{aligned}\mu' \leq& \frac{(k+1)|A|}{n(k+\eta)} \\ =& \mu\frac{k+1}{k+\eta} \\ \leq& \mu(1 + 1/k).\end{aligned}$$

Now the probability that samp' samples more than a $\mu' + \epsilon_1 \leq \mu + \epsilon_1 + \eta/k$ fraction of elements in A' is at most $\mu'\delta \leq \mu(1 + 1/k)\delta$. See that by definition of A', samp' must sample A' for samp to sample A. Thus the probability that samp samples more than $\mu + \epsilon_1 + \eta/k$ fraction of elements of A is at most $\mu\delta(1 + 1/k)$. Thus samp is a sampler with the stated parameters. ☐

Now we can use our curve samplers to get our first correcting lines set.

Lemma 8.10 (Correcting Lines Sets From Curve Samplers)**.** *Take any field* $\mathbb{F}$ *with* $|\mathbb{F}| \geq 101$ *and natural numbers* dim *and* t*. Then there is a set* $\mathcal{C}$ *such that for any* $\epsilon > 0$ *we have that* $\mathcal{C}$ *is a below* $\epsilon|\mathbb{F}^{\mathsf{dim}}|/2$ *correcting lines set for* $\mathbb{F}^{\mathsf{dim}}$ *with line count* $|\mathbb{F}|$*, accuracy error* ϵ *and strong confidence error*

$$1.01\left(\frac{10t}{\sqrt{|\mathbb{F}|}}\right)^t \frac{t+1}{\epsilon}$$

that is time $t\mathsf{dim}\,\mathrm{polylog}(|\mathbb{F}|)$ *and space* $\tilde{O}(\log(|\mathbb{F}|)(t + \mathsf{dim}))$ *uniform. Further* $\mathcal{C}$ *has size* $|\mathcal{C}| = |\mathbb{F}^{\mathsf{dim}}|^{t+2}$*.*

Proof. Set $n' = |\mathbb{F}^{\mathsf{dim}}|$ and $n = \frac{|\mathbb{F}^{\mathsf{dim}}|-1}{|\mathbb{F}|-1}$.

Then by Lemma 7.12, we have that for any $\epsilon_1 > 0$ the set of all degree t-curves through $\mathbb{F}^{\mathsf{dim}}$ is a sampler, denoted samp', with accuracy error ϵ_1 and *strong* confidence error

$$\delta_1 = \left(\frac{t}{\epsilon_1\sqrt{|\mathbb{F}|}}\right)^t (t+1).$$

Further, see that samp' can be viewed as a function $\mathsf{samp}' : [m] \to [n']^{|\mathbb{F}|}$ where $m = n'^{t+1}$ and that samp' can be evaluated in time $t\mathsf{dim}\, polylog(|\mathbb{F}|)$ and space $\tilde{O}(\log(|\mathbb{F}|)(t + \mathsf{dim}))$. Set $\epsilon_1 = \frac{1}{10}$ so that samp has accuracy error $\frac{1}{10}$ and strong confidence error

$$\delta_1 = \left(\frac{10t}{\sqrt{|\mathbb{F}|}}\right)^t (t+1).$$

Now we need to convert samp' to a sampler for a space of size n. Set $\eta = 1/n$ and $k = |\mathbb{F}| - 1$. Then see that $n = \frac{n'-1}{k}$, so $n' = kn + 1 = n(k+\eta)$ and thus $n = \frac{n'}{k+\eta}$. Then we can apply Lemma 8.9 to show that $\mathsf{samp} : [m] \to [n]^{|\mathbb{F}|}$ defined by $\mathsf{samp}(x)_i = \mathsf{samp}'(x)_i \mod n$ is a sampler for $[n]$ with accuracy error

$$\begin{aligned}\epsilon' =& \epsilon_1 + \frac{\eta}{k} \\ \leq& 1/10 + \frac{1}{n'-1} \\ \leq& 11/100.\end{aligned}$$

and strong confidence error $\delta_1(1 + 1/k) \leq 1.01\delta_1$.

Finally, we can use Lemma 8.8 with samp to get a set $\mathcal{C}$ such that for any ϵ we have that $\mathcal{C}$ is a below $\epsilon\frac{1-2\epsilon'}{1+2\epsilon'}|\mathbb{F}^{\mathsf{dim}}| \leq \epsilon|\mathbb{F}^{\mathsf{dim}}|/2$ correcting lines set for $\mathbb{F}^{\mathsf{dim}}$ with line count $|\mathbb{F}|$, accuracy error ϵ and strong confidence error

$$\delta_1(1+1/k)/\epsilon \leq 1.01\left(\frac{10t}{\sqrt{|\mathbb{F}|}}\right)^t (t+1)\frac{1}{\epsilon}$$

that is time $t\mathsf{dim}\, polylog(|\mathbb{F}|)$ and space $\tilde{O}(\log(|\mathbb{F}|)(t + \mathsf{dim}))$ uniform. Further $\mathcal{C}$ has size $|\mathcal{C}| = m|\mathbb{F}^{\mathsf{dim}}| = |\mathbb{F}^{\mathsf{dim}}|^{t+2}$. □

8.3 Uniform Deterministic Decoders for Lifted Reed-Solomon Codes

Now we can combine our efficient subspace samplers from Corollary 7.15 with the correcting lines set from Lemma 8.10 to get an efficient correcting lines set. Finally, we can use the correcting lines set to get a codeword improving set for lifted Reed-Solomon codes.

Lemma 8.11 (Efficient Improving Set For Lifted Reed-Solomon)**.** *Take any field $\mathbb{F}$ of prime power order $p \geq 101$, any dimensions dim and a, and $\epsilon > 0$. Let C be the set of lifted Reed-Solomon codes for $\mathbb{F}^{a\mathsf{dim}}$ with degree $\mathsf{deg} = (1-2\epsilon)p$. Then there is a $p(p-1)$ query, time and space $poly(p^a\mathsf{dim})$ uniform, below $\epsilon p^{a\mathsf{dim}}/12$, factor $O\left(\frac{a^{2a+1}400^a}{p^a\epsilon^2}\right)$ improving set for C with size $p^{a(2a+O(1))}poly(\mathsf{dim})$.*

Proof. By Corollary 7.15, there is an unbiased a-dimensional subspace sampler, $\mathcal{S}$, for $\mathbb{F}_p^{a\mathsf{dim}}$ with size $p^{a(\mathsf{dim}+O(1))}poly(\mathsf{dim})$, with accuracy error $\epsilon/4$, and strong confidence error $\delta_1 = \frac{32}{p^a\epsilon^2}$. Further $\mathcal{S}$ is time and space $poly(p^a\mathsf{dim})$ uniform.

Setting $t = 2a$, by Lemma 8.10, there is $\mathcal{C}'$ which is a below $\epsilon|\mathbb{F}^a|/3$ correcting lines set for $\mathbb{F}^a$ with line count p, accuracy error $2\epsilon/3$ and strong confidence error

$$\delta_2 = 1.52\left(\frac{20a}{\sqrt{p}}\right)^{2a}\frac{2a+1}{\epsilon} = O\left(\frac{a^{2a+1}400^a}{p^a\epsilon}\right)$$

that is time $\tilde{O}(a^2p)$ and space $\tilde{O}(\log(p)a)$ uniform. Further $\mathcal{C}'$ has size $|\mathcal{C}'| = p^{2a(a+1)}$.

Then by Lemma 8.7, there exists $\mathcal{C}$ that is a below $(\epsilon/3 - \epsilon/4)|\mathbb{F}^{a\mathsf{dim}}| = \epsilon p^{a\mathsf{dim}}/12$ correcting lines set for $\mathbb{F}^{a\mathsf{dim}}$ with accuracy error $\epsilon/4 + 2\epsilon/3 = 11\epsilon/12$, *strong* confidence error

$$\delta_1 + \delta_2 = O\left(\frac{a^{2a+1}400^a}{p^a\epsilon^2}\right),$$

and line count p. Further the size of the final correcting lines set is

$$|\mathcal{C}| = p^{a(\mathsf{dim}+O(1))} poly(\mathsf{dim}) p^{2a(a+1)} = p^{a(\mathsf{dim}+2a+O(1))} poly(\mathsf{dim})$$

and $\mathcal{C}$ is time and space $poly(p^a \mathsf{dim})$ uniform.

See that C is the set of lifted Reed-Solomon codes for $\mathbb{F}^{a\mathsf{dim}}$ with degree $\mathsf{deg} = (1-2(\epsilon/12+11\epsilon/12))|\mathbb{F}| = (1-2\epsilon)p$. By Lemma 8.6 there is a $p(p-1)$ query, time and space $poly(p^a\mathsf{dim})$ uniform, below $\epsilon p^{a\mathsf{dim}}/12$, factor $O\left(\frac{a^{2a+1}400^a}{p^a\epsilon^2}\right)$ improving set for C with size $p^{a(\mathsf{dim}+2a+O(1))} poly(\mathsf{dim})/p^{a\mathsf{dim}} = p^{a(2a+O(1))}$. □

All that is left is to set ϵ, p, a, and dim appropriately to give us an efficient decoding algorithm. Intuitively, we just want to set p to be large, dim to be small (but not constant), a to be even smaller (but not constant), and ϵ so that the lifted Reed-Solomon code has high rate. The code has high rate as long as $\epsilon \ll 2^{-O((a\mathsf{dim})^{a2\mathsf{dim}})}$, which can be accomplished if $a\mathsf{dim}$ is small enough.

Lemma 8.12 (Explicit Asympotically Good Codes With Uniform Sub-polynomial Space, Almost Linear Time Correctors)**.** *There exists an explicit infinite family of codes and a uniform correcting algorithm such that, for some alphabet Σ with $|\Sigma| = O(\log(n))$ for each code $C_n : \Sigma^n \to \Sigma^m$ in the family, C_n has constant rate and has a deterministic function, g, running in time $n^{1+o(1)}$ and space $n^{o(1)}$ that outputs an $n^{o(1)}$ bit description of a deterministic, $n^{o(1)}$ query function I such that for some correcting radius $d = n^{1-o(1)}$ if for some $x \in \Sigma^n$ and $y \in \Sigma^m$ we have that $\Delta(y, C(x)) \leq d$, then we have that $g(x)$ outputs an I such that $I(y) = C(x)$. Further, for each $i \in [m]$, $I(y)_i$ is computable in time and space at most $n^{o(1)}$.*

Remark. *In this proof, we are explicit that our correction algorithm actually outputs a low query function that corrects the codeword. This is important if we want to get an efficient corrector for this code combined with another code. If one only cares about decoding, then we can more simply say it corrects with correcting radius $d = n^{1-o(1)}$ in time $n^{1+o(1)}$ and space $n^{o(1)}$.*

Proof. For p that is a power of 2, set $\mathsf{dim} = \log(\log(\log(p)))$, set $a = \log(\mathsf{dim})$, and $\epsilon = 2^{-\sqrt{\log(p)}}$. Further, take such a p so that dim and a are both integers.

Then let $m = p^{a\mathsf{dim}}$ and $\mathbb{F}$ be the field of order p. Let LRS be the set of lifted Reed-Solomon codes for $\mathbb{F}^{a\mathsf{dim}}$ with degree $\mathsf{deg} = (1-2\epsilon)p$. Finally, let n be such that p^n is the number of codewords in LRS. Define $C_n = \mathsf{LRS}$. See that by definition, C_n is a code with message length n and codeword length m and there are infinitely many such codes.

By Lemma 8.2, for some

$$c = O((a\mathsf{dim})^{2a\mathsf{dim}}) \leq 2^{poly(\mathsf{dim})} = 2^{poly(\log(\log(\log(p))))} = 2^{o(\log(\log(p)))} \leq \log(p)^{o(1)} \leq o\left(\sqrt{\log(p)}\right)$$

and $\mathsf{deg} = (1-2^{-c})|\mathbb{F}| = (1-2^{-o(\sqrt{\log(p)})})|\mathbb{F}|$ we have that the lifted Reed-Solomon code of dimension $a\mathsf{dim}$ and degree deg over field $\mathbb{F}$ has rate 1/2. For large enough p we have that $\mathsf{deg} \leq (1-2\epsilon)|\mathbb{F}|$. Thus LRS has rate at least 1/2 and $m = O(n)$.

By Lemma 8.11, there is a set of functions $\mathcal{I}$ that is a $q = p(p-1)$ query, time and space $T = S = poly(p^a\mathsf{dim}) = n^{o(1)}$ uniform, below $d = \epsilon p^{a\mathsf{dim}}/12 = n\epsilon/12 = n^{1-o(1)}$, factor $\delta = O\left(\frac{a^{2a+1}400^a}{p^a\epsilon^2}\right)$ improving set for C with size $|\mathcal{I}| = p^{a(2a+O(1))} poly(\mathsf{dim}) = n^{o(1)}$.

See that

$$\begin{aligned}\delta =& O\left(\frac{a^{2a+1}400^a}{p^a\epsilon^2}\right)\\ =& 2^{\log(a)(2a+1)+\log(400)a+2\sqrt{\log(p)}-\log(p)a}\\ =& p^{-a+o(1)}.\end{aligned}$$

So for enough p we have that $\delta < 0.001 < (1-\delta)^2$. Set

$$
\begin{aligned}
\ell =& \lceil \frac{\log(d/3+1)}{\log((1-\delta)^2/\delta)} \rceil \\
\leq& \frac{\log(n\epsilon/36+1)}{\log(p^{a-o(1)})-1} \\
\leq& \frac{\log(p^{a\mathsf{dim}} 2^{-\sqrt{\log(p)}}) + O(1)}{(a-o(1))\log(p)-1} \\
\leq& \frac{a\mathsf{dim}\log(p) - \sqrt{\log(p)+O(1)} + O(1)}{(a-o(1))\log(p)-1} \\
\leq& \frac{a\mathsf{dim}\log(p)(1-o(1))}{(1-o(1))a\log(p)} \\
\leq& \mathsf{dim}(1+o(1)).
\end{aligned}
$$

By Lemma 5.6 LRS has a function g running in time

$$
\begin{aligned}
O(m\ell|\mathcal{I}|^2 q^{\ell+1} T) =& O(n\mathsf{dim}(1+o(1))(n^{o(1)})^2(p(p-1))^{\mathsf{dim}(1+o(1))+1} n^{o(1)}) \\
\leq& n n^{o(1)} n^{o(1)} n^{2(1+o(1))/a} n^{o(1)} \\
\leq& n^{1+o(1)}
\end{aligned}
$$

and space

$$
\begin{aligned}
& O(\ell \min\{S, q\log(|\mathbb{F}|)\} + \ell\log(|\mathcal{I}|) + \log(m) + S) \\
=& O(\mathsf{dim}(1+o(1))p(p-1)\log(p) + \mathsf{dim}(1+o(1))\log(n^{o(1)}) + \log(O(n)) + n^{o(1)}) \\
=& n^{o(1)}
\end{aligned}
$$

and outputs an $O(\ell\log(|\mathcal{I}|)) = n^{o(1)}$ bit description of a deterministic, $q^\ell = n^{o(1)}$ query function I such that if for some $x \in \{0,1\}^n$ and $y \in \{0,1\}^m$ we have that $\Delta(y, C(x)) \leq d = n^{1-o(1)}$, then we have that $g(x)$ outputs an I such that $I(y) = C(x)$. Further, for each $i \in [m]$, $I(y)_i$ is computable in time at most $O(Tq^\ell) = n^{o(1)}$ and the space at most $O(\ell \min\{S, q\log(p)\} + S) = n^{o(1)}$. □

8.4 Distance Amplification

To get a locally correctable code with $n^{o(1)}$ query complexity, we use a lifted Reed-Solomon code with $\mathsf{dim} = \omega(1)$. This gives us a degree that approaches the field size, so the lifted Reed-Solomon code does not have constant correcting radius. To get around this, we use the observation from [Kop+17] that expander based gap amplification [AEL95; AL96] preserves locally correctability. Specifically, expander based gap amplification gives a code that has a low query deterministic decoder that reduces the fraction of errors.

The following is implicit in the proof of [Kop+17, Lemma 3.2 and 3.5], and more clearly stated in [MMO24, Theorem 1.1] in terms of an approximate locally decodable code.

Lemma 8.13 (Explicit Deterministic Approximate Locally Decodable Codes)**.** *For any alphabet Σ, any integers n, d, and $\delta > 0$, for some $m = O(n\log(|\Sigma|) + d)$, there is an explicit code $C : \Sigma^n \to \{0,1\}^m$ and uniform decoder $D : \{0,1\}^m \to \Sigma^n$ such that for any $x \in \Sigma^n$ and any $y \in \{0,1\}^m$ with $\Delta(D(x), y) \leq d$, we have that $\Delta(D(y), x) \leq \delta n$.*

Further, every output bit of both C and D can be computed deterministically with $q = poly(\log(|\Sigma|)/\delta)$ non-adaptive queries in time $poly(q\log(n))$.

Using Lemma 8.13, we can take any code that has a decoder decoding from any distance δn to one that can be decoded from distance $\Omega(n)$ with only a constant factor increase in the codeword length and a decoding overhead that is only $poly(1/\delta)$ in time, space, and query complexity. In particular,

Lemma 8.14 (Efficient Decoding Radius Amplification)**.** *Suppose for some natural numbers n and n' there is a code $C : \Sigma_1^n \to \Sigma_2^{n'}$ with a decoder running in time T and space S with decoding radius $\delta n'$. Then for some $m = O(\log(|\Sigma_2|)n')$ there is a code $C' : \Sigma_1^n \to \{0,1\}^m$ with a decoder running in time $T\,poly(\log(|\Sigma_2|)\log(n')/\delta)$ and space $S + poly(\log(|\Sigma_2|)\log(n')/\delta)$ with decoding radius $\Omega(n)$. Further, if the decoder for C is deterministic, so is the decoder for C'.*

Proof. Use Lemma 8.13 with $d = n'\log(|\Sigma_2|)$ to get a code $C^* : \Sigma_2^{n'} \to \{0,1\}^m$ for some $m = O(n'\log(|\Sigma_2|) + d) = O(n'\log(|\Sigma|_2))$ along with decoder D^*. Then our final code is $C' = C^* \circ C$. To decode C' we run the decoder for C on the output of D^*. See that for any $x \in \Sigma_2^n$ and $y \in \{0,1\}^m$ with $\Delta(f(x), y) \le d$, we have that $\Delta(D^*(y), x) \le \delta n'$. Thus the output of D^* is in the decoding radius of C, so its decoder succeeds. Since D^* only requires time $poly(\log(|\Sigma_2|)\log(n')/\delta)$ to compute a single symbol of output, calling D^* only incurs a factor $poly(\log(|\Sigma_2|)\log(n')/\delta)$ overhead in time and space, so we get our decoding efficiency. □

Finally, one can get *Theorem* 1.1 as a corollary of Lemma 8.12, Lemma 8.3, and Lemma 8.14. More generally, we can also get code correction in a similar manner, except that instead of running a decoder for the inner code, we run a corrector and then encode with the outer code again.

Theorem 8.15 (Codes With *Uniform* Time And Space Efficient Correctors)**.** *There exists an infinite family of codes $C : \{0,1\}^n \to \{0,1\}^m$ where $m = O(n)$ and a deterministic uniform algorithm B that computes a function $D : \{0,1\}^m \to \{0,1\}^m$ such that:*

Efficient: *B runs in time $n^{1+o(1)}$ and space $n^{o(1)}$.*

Corrects: *For some decoding radius $d = \Omega(n)$, for any $x \in \{0,1\}^n$ and $w \in \{0,1\}^m$ with $\Delta(w, C(x)) \le d$ we have that*

$$D(w) = C(x).$$

Proof. Let $C' : \Sigma^n \to \Sigma^m$ be the code from Lemma 8.12 with $|\Sigma| = O(\log(n)) = n^{o(1)}$ and let $d = n^{1-o(1)}$ be its correcting radius. Let $\delta = d/m = n^{-o(1)}$. Then by Lemma 8.13, for some $d' = \Omega(n)$ and $m = O(n)$ there is a code $C'' : \Sigma^m \to \{0,1\}^{m'}$ and uniform decoder $D : \{0,1\}^{m'} \to \Sigma^m$ such that for any $x \in \Sigma^m$ and any $y \in \{0,1\}^{m'}$ with $\Delta(C''(x), y) \le d'$, we have that $\Delta(D(y), x) \le \delta m = d$. Further, every output bit of both C'' and D can be computed deterministically with $q = poly(\log(|\Sigma|)/\delta) = n^{o(1)}$ non-adaptive queries in time $poly(q\log(n)) = n^{o(1)}$.

Set $C^* = C'' \circ C'$. Now we show that C^* has an efficient corrector. On input $y \in \{0,1\}^{m'}$, this corrector first runs g from Lemma 8.12 on $D(y)$ to get local corrector I. Then the corrector runs $C''(I(D(y)))$ to compute $C^*(x)$. Running D and I only adds a factor of $n^{o(1)}$ to the time of the correction, so our corrector is efficient. To see that it corrects, see that if $\Delta(y, C''(C'(x))) \le d'$, we have that $\Delta(D(y), C'(x)) \le d$, thus the corrector from Lemma 8.12 outputs I such that $I(D(y)) = C'(x)$, and thus $C''(I(D(y))) = C''(C'(x))$.

To convert C^* to our final code with a binary message, C, we can just use $\lfloor \log(|\Sigma|) \rfloor$ bits to encode a symbol of Σ. This may decrease the rate by some $o(1)$ amount, but the rate is still constant. To get a decoder, we just use Lemma 8.3 to efficiently find the message symbols in the lifted Reed-Solomon codeword, $y_1, \ldots, y_n$, and efficiently compute the corresponding message symbols with $I(D(y_1)), \ldots, I(D(y_n))$. □

9 Open Problems

In this work, we showed that there are good codes with time and space efficient deterministic decoders, but there are still many open problems.

1. Find codes that can both be:
   - encoded in $n^{1+o(1)}$ time and $n^{o(1)}$ space.
   - decoded in $n^{1+o(1)}$ time and $n^{o(1)}$ space.

   There exist codes that can be encoded deterministically in almost $n^{1+o(1)}$ time and $n^{o(1)}$ space (the condenser codes of [CM24]), and there exist different codes that can be decoded deterministically in

$n^{1+o(1)}$ time and $n^{o(1)}$ space (any typical LCC). But these are *different* codes. We want codes that achieve both simultaneously.

We expect that by taking an appropriately chosen tensor code of a condenser code and a typical LCC one can get a tradeoff between the encoder space and the decoder space. For example, we suspect that there is a tensor code constructed in such a manner which has an encoder that runs in $n^{1+o(1)}$ time and $n^{1/2+o(1)}$ space and has a decoder that runs in $n^{1+o(1)}$ time and $n^{1/2+o(1)}$ space. Can one do better than this tensor construction?

2. Find a problem that can be solved by a randomized algorithm more time and space efficiently than a deterministic algorithm.

   For this we need a lower bound for deterministic algorithms, which is notoriously difficult to prove. However, once we constrain both the time and the space it becomes easier to prove lower bounds.

   Specifically, for some constants $\alpha' > \alpha > 0$ and $\beta' > \beta > 0$ find a function computable by a randomized algorithm in time $n^{1+\alpha}$ and space n^{β}, but not computable by a deterministic algorithm in time $n^{1+\alpha'}$ and space $n^{\beta'}$.

   We hoped that decoding locally correctable codes would be such a problem, but our results show that locally correctable codes can be deterministically decoded time and space efficiently.

3. Find codes with deterministic decoders running in quasilinear time and polylogarithmic space.

   We don't even know of codes with randomized decoders that run in quasilinear time and polylogarithmic space. If polylogarithmic query LCCs were discovered, this would imply a randomized decoder that runs in quasilinear time and polylogarithmic space. But our derandomization technique would not give a deterministic decoder that runs in quasilinear time and polylogarithmic space for such an LCC. A fundamental limitation in our decoding strategy is that it can only ever give decoders running in time $\Omega(n2^{\sqrt{\log(n)}})$, even if there are typical LCCs with polylogarithmic queries. This is because the time of the distance $\Omega(n)$ decoder is at least $n^{1+1/\ell}2^{\ell}$ for some ℓ (see for example Theorem 1.2 and Theorem 5.3), which is minimized for $\ell = \sqrt{\log(n)}$.

   Ideally one could hope for quasilinear time and polylogarithmic space decoders, or perhaps even linear time and log space. But our techniques cannot achieve this.

Acknowledgments

Thanks to Geoffrey Mon for discussions on locally correctable codes, Zeyu Guo for discussions about extractors and curve samplers, and Swastik Kopparty for his suggestion to make explicit decoders for lifted Reed-Solomon codes and a reference on how to convert correctors for lifted Reed-Solomon codes to decoders.

References

[AEL95] N. Alon, J. Edmonds, and M. Luby. "Linear time erasure codes with nearly optimal recovery". In: *Proceedings of IEEE 36th Annual Foundations of Computer Science.* 1995, pp. 512–519. DOI: 10.1109/SFCS.1995.492581.

[AL96] N. Alon and M. Luby. "A linear time erasure-resilient code with nearly optimal recovery". In: *IEEE Transactions on Information Theory* 42.6 (1996), pp. 1732–1736. DOI: 10.1109/18.556669.

[AS98] Sanjeev Arora and Shmuel Safra. "Probabilistic Checking of Proofs: A New Characterization of NP". In: *J. ACM* 45.1 (Jan. 1998), 70–122. ISSN: 0004-5411. DOI: 10.1145/273865.273901. URL: https://doi.org/10.1145/273865.273901.

[Adl78] Leonard Adleman. "Two theorems on random polynomial time". In: *19th Annual Symposium on Foundations of Computer Science (FOCS 1978).* 1978, pp. 75–83. DOI: 10.1109/SFCS.1978.37.

[Aro+98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. "Proof Verification and the Hardness of Approximation Problems". In: *J. ACM* 45.3 (May 1998), 501–555. ISSN: 0004-5411. DOI: 10.1145/278298.278306. URL: https://doi.org/10.1145/278298.278306.

[BS+03] Eli Ben-Sasson, Madhu Sudan, Salil Vadhan, and Avi Wigderson. "Randomness-efficient low degree tests and short PCPs via epsilon-biased sets". In: *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing*. STOC '03. San Diego, CA, USA: Association for Computing Machinery, 2003, 612–621. ISBN: 1581136749. DOI: 10.1145/780542.780631. URL: https://doi.org/10.1145/780542.780631.

[BS+13] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, and Eran Tromer. "On the Concrete Efficiency of Probabilistically-Checkable Proofs". In: *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*. STOC '13. Palo Alto, California, USA: Association for Computing Machinery, 2013, 585–594. ISBN: 9781450320290. DOI: 10.1145/2488608.2488681. URL: https://doi.org/10.1145/2488608.2488681.

[BW86] Elwyn R. Berlekamp and Lloyd R. Welch. *Error Correction of Algebraic Block Codes*. US Patent, Number 4,633,470, 1986. 1986.

[Bab+91] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. "Checking Computations in Polylogarithmic Time". In: *Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing*. STOC '91. New Orleans, Louisiana, USA: Association for Computing Machinery, 1991, 21–32. ISBN: 0897913973. DOI: 10.1145/103418.103428. URL: https://doi.org/10.1145/103418.103428.

[CM23] Joshua Cook and Dana Moshkovitz. "Tighter MA/1 Circuit Lower Bounds from Verifier Efficient PCPs for PSPACE". In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023)*. Ed. by Nicole Megow and Adam Smith. Vol. 275. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023, 55:1–55:22. ISBN: 978-3-95977-296-9. DOI: 10.4230/LIPIcs.APPROX/RANDOM.2023.55. URL: https://drops.dagstuhl.de/opus/volltexte/2023/18880.

[CM24] Joshua Cook and Dana Moshkovitz. "Explicit Time and Space Efficient Encoders Exist Only With Random Access". In: (2024). URL: https://eccc.weizmann.ac.il/report/2024/032/.

[CY23] Gil Cohen and Tal Yankovitz. "Asymptotically-Good RLCCs with $(log n)^{2+o(1)}$ Queries". In: 2023. URL: https://api.semanticscholar.org/CorpusID:262038079.

[Din+22] Irit Dinur, Shai Evra, Ron Livne, Alexander Lubotzky, and Shahar Mozes. "Locally testable codes with constant rate, distance, and locality". In: *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*. STOC 2022. Rome, Italy: Association for Computing Machinery, 2022, 357–374. ISBN: 9781450392648. DOI: 10.1145/3519935.3520024. URL: https://doi.org/10.1145/3519935.3520024.

[Fei+91] Uriel Feige, Shafi Goldwasser, László Lovász, Shmuel Safra, and Mario Szegedy. "Approximating Clique is Almost NP-Complete (Preliminary Version)". In: *32nd Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 1-4 October 1991*. IEEE Computer Society, 1991, pp. 2–12. DOI: 10.1109/SFCS.1991.185341. URL: https://doi.org/10.1109/SFCS.1991.185341.

[GK16] Alan Guo and Swastik Kopparty. "List-Decoding Algorithms for Lifted Codes". In: *IEEE Transactions on Information Theory* 62.5 (2016), pp. 2719–2725. DOI: 10.1109/TIT.2016.2538766.

[GKR15] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. "Delegating Computation: Interactive Proofs for Muggles". In: *J. ACM* 62.4 (Sept. 2015). ISSN: 0004-5411. DOI: 10.1145/2699436. URL: https://doi.org/10.1145/2699436.

[GKS13] Alan Guo, Swastik Kopparty, and Madhu Sudan. "New affine-invariant codes from lifting". In: *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*. ITCS '13. Berkeley, California, USA: Association for Computing Machinery, 2013, 529–540. ISBN: 9781450318594. DOI: 10.1145/2422436.2422494. URL: https://doi.org/10.1145/2422436.2422494.

[GM20] Ofer Grossman and Dana Moshkovitz. "Amplification and Derandomization without Slowdown". In: *SIAM Journal on Computing* 49.5 (2020), pp. 959–998. DOI: 10.1137/17M1110596. eprint: https://doi.org/10.1137/17M1110596. URL: https://doi.org/10.1137/17M1110596.

[Gol] Oded Goldreich. *A Primer on Pseudorandom Generators*. University lecture series. American Mathematical Soc. ISBN: 9780821883112. URL: https://books.google.com/books?id=9k6Lw2U2XCkC.

[Gro06] André Gronemeier. "A Note on the Decoding Complexity of Error-Correcting Codes". In: *Inf. Process. Lett.* 100.3 (2006), 116–119. ISSN: 0020-0190. DOI: 10.1016/j.ipl.2006.06.006. URL: https://doi.org/10.1016/j.ipl.2006.06.006.

[Guo13] Zeyu Guo. "Randomness-Efficient Curve Samplers". In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 16th International Workshop, APPROX 2013, and 17th International Workshop, RANDOM 2013, Berkeley, CA, USA, August 21-23, 2013. Proceedings*. Ed. by Prasad Raghavendra, Sofya Raskhodnikova, Klaus Jansen, and José D. P. Rolim. Vol. 8096. Lecture Notes in Computer Science. Springer, 2013, pp. 575–590. DOI: 10.1007/978-3-642-40328-6_40. URL: https://doi.org/10.1007/978-3-642-40328-6_40.

[HOW13] Brett Hemenway, Rafail Ostrovsky, and Mary Wootters. "Local Correctability of Expander Codes". In: *Automata, Languages, and Programming*. Ed. by Fedor V. Fomin, Rūsiņš Freivalds, Marta Kwiatkowska, and David Peleg. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 540–551. ISBN: 978-3-642-39206-1.

[HR18] Justin Holmgren and Ron Rothblum. "Delegating Computations with (Almost) Minimal Time and Space Overhead". In: *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*. 2018, pp. 124–135. DOI: 10.1109/FOCS.2018.00021.

[JM21] Akhil Jalan and Dana Moshkovitz. "Near-Optimal Cayley Expanders for Abelian Groups". In: *41st IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2021)*. Ed. by Mikołaj Bojańczyk and Chandra Chekuri. Vol. 213. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021, 24:1–24:23. ISBN: 978-3-95977-215-0. DOI: 10.4230/LIPIcs.FSTTCS.2021.24. URL: https://drops-dev.dagstuhl.de/entities/document/10.4230/LIPIcs.FSTTCS.2021.24.

[KM23] Vinayak Kumar and Geoffrey Mon. "Relaxed Local Correctability from Local Testing". In: *ArXiv* abs/2306.17035 (2023). URL: https://api.semanticscholar.org/CorpusID:259287351.

[KRR21] Yael Tauman Kalai, Ran Raz, and Ron D. Rothblum. "How to Delegate Computations: The Power of No-Signaling Proofs". In: *J. ACM* 69.1 (2021). ISSN: 0004-5411. DOI: 10.1145/3456867. URL: https://doi.org/10.1145/3456867.

[KSY14] Swastik Kopparty, Shubhangi Saraf, and Sergey Yekhanin. "High-rate codes with sublinear-time decoding". In: *J. ACM* 61.5 (2014). ISSN: 0004-5411. DOI: 10.1145/2629416. URL: https://doi.org/10.1145/2629416.

[KU06] Shankar Kalyanaraman and Christopher Umans. "On obtaining pseudorandomness from error-correcting codes". In: FSTTCS'06. Kolkata, India: Springer-Verlag, 2006, 105–116. ISBN: 3540499946. DOI: 10.1007/11944836_12. URL: https://doi.org/10.1007/11944836_12.

[Kop13] Swastik Kopparty. "Some remarks on multiplicity codes". In: *Discrete Geometry and Algebraic Combinatorics*. Ed. by Alexander Barg and Oleg R. Musin. Vol. 625. Contemporary Mathematics. American Mathematical Society, 2013. URL: http://www.ams.org/books/conm/625/12497.

[Kop+17] Swastik Kopparty, Or Meir, Noga Ron-Zewi, and Shubhangi Saraf. "High-Rate Locally Correctable and Locally Testable Codes with Sub-Polynomial Query Complexity". In: *J. ACM* 64.2 (2017). ISSN: 0004-5411. DOI: 10.1145/3051093. URL: https://doi.org/10.1145/3051093.

[Lun+90] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. "Algebraic methods for interactive proof systems". In: *Proceedings [1990] 31st Annual Symposium on Foundations of Computer Science*. 1990, 2–10 vol.1. DOI: 10.1109/FSCS.1990.89518.

[MMO24] Geoffrey Mon, Dana Moshkovitz, and Justin Oh. "Approximate Locally Decodable Codes with Constant Query Complexity and Nearly Optimal Rate". In: *2024 IEEE International Symposium on Information Theory (ISIT)*. 2024, pp. 2838–2843. DOI: 10.1109/ISIT57864.2024.10619326.

[Mos17] Dana Moshkovitz. "Low-degree test with polynomially small error". In: *Comput. Complex.* 26.3 (2017), 531–582. ISSN: 1016-3328. DOI: 10.1007/s00037-016-0149-4. URL: https://doi.org/10.1007/s00037-016-0149-4.

[Nis90] Noam Nisan. "Pseudorandom Generators for Space-Bounded Computations". In: *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing*. STOC '90. Baltimore, Maryland, USA: Association for Computing Machinery, 1990, 204–212. ISBN: 0897913612. DOI: 10.1145/100216.100242. URL: https://doi.org/10.1145/100216.100242.

[RRR16] Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. "Constant-Round Interactive Proofs for Delegating Computation". In: *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*. STOC '16. Cambridge, MA, USA: Association for Computing Machinery, 2016, 49–62. ISBN: 9781450341325. DOI: 10.1145/2897518.2897652. URL: https://doi.org/10.1145/2897518.2897652.

[SS96] Michael Sipser and Daniel A. Spielman. "Expander codes". In: *IEEE Transactions on Information Theory* 42.6 (1996), pp. 1710–1722.

[STV01] Madhu Sudan, Luca Trevisan, and Salil Vadhan. "Pseudorandom Generators without the XOR Lemma". In: *J. Comput. Syst. Sci.* 62.2 (2001), 236–266. ISSN: 0022-0000. DOI: 10.1006/jcss.2000.1730. URL: https://doi.org/10.1006/jcss.2000.1730.

[Spi95] Daniel A. Spielman. "Linear-Time Encodable and Decodable Error-Correcting Codes". In: *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing*. STOC '95. Las Vegas, Nevada, USA: Association for Computing Machinery, 1995, 388–397. ISBN: 0897917189. DOI: 10.1145/225058.225165. URL: https://doi.org/10.1145/225058.225165.

[Sud97] Madhu Sudan. "Decoding of Reed Solomon codes beyond the error-correction bound". In: *Journal of Complexity* 13 (1997), pp. 180–193. ISSN: 0885–064X.

[TS17] Amnon Ta-Shma. "Explicit, almost optimal, epsilon-balanced codes". In: *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*. STOC 2017. Montreal, Canada: Association for Computing Machinery, 2017, 238–251. ISBN: 9781450345286. DOI: 10.1145/3055399.3055408. URL: https://doi.org/10.1145/3055399.3055408.

[TSU06] Amnon Ta-Shma and Christopher Umans. "Better lossless condensers through derandomized curve samplers". In: *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*. 2006, pp. 177–186. DOI: 10.1109/FOCS.2006.18.

[Uma03] Christopher Umans. "Pseudo-random generators for all hardnesses". In: *J. Comput. Syst. Sci.* 67.2 (2003), 419–440. ISSN: 0022-0000. DOI: 10.1016/S0022-0000(03)00046-1. URL: https://doi.org/10.1016/S0022-0000(03)00046-1.

[Wil16] R. Ryan Williams. "Strong ETH breaks with Merlin and Arthur: short non-interactive proofs of batch evaluation". In: *Proceedings of the 31st Conference on Computational Complexity*. CCC '16. Tokyo, Japan: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016. ISBN: 9783959770088.

[Yek12] Sergey Yekhanin. *Locally Decodable Codes*. Hanover, MA, USA: Now Publishers Inc., 2012. ISBN: 1601985444.

ECCC ISSN 1433-8092
https://eccc.weizmann.ac.il