Local Testing and Decoding of High-Rate Error-Correcting Codes*

Swastik Kopparty† Shubhangi Saraf‡

Abstract

We survey the state of the art in constructions of locally testable codes, locally decodable codes and locally correctable codes of high rate.

1 Introduction

This aim of this article is to survey some recent developments surrounding locally testable codes, locally decodable codes and locally correctable codes of constant rate. "Local" codes, which are error-correcting codes equipped with "local" algorithms, have a long history of interaction with complexity theory, with notable applications to interactive proofs, probabilistically-checkable proofs (PCPs), hardness amplification, private information retrieval, and algebraic complexity theory [LFKN92, Sha90, BFLS91, AS98, ALM+98, CKGS98, STV01]. In the late 1980s and early 1990s as the locality properties of some classical codes were being understood and harnessed for the first time in some important complexity theory applications, it became clear that a deeper understanding of locality in codes would be an interesting and fruitful pursuit.

Let Σ be a finite alphabet. A code of length n over the alphabet Σ is simply a subset C of Σ^n, and its elements are called codewords. A code C can be used to represent data; letting $k = \log_{|\Sigma|} |C|$, one can choose a bijection $E : \Sigma^k \to C$ (called the encoding map). Then the representation of the string $s \in \Sigma^k$ is $E(s)$. The blowup in the size of the representation is measured by the *rate* R, defined by $R = k/n$.

For our purposes, the advantage of using such a representation is to protect data from errors. We will work with the (relative) Hamming distance on Σ^n, which is defined by $\Delta(x, y) = \frac{1}{n} \cdot |\{i \in [n] \mid x_i \neq y_i\}|$. The error-correcting ability of a code can be measured by its minimum distance δ, which is defined by:

$$\delta = \min_{x,y \in C, x \neq y} \Delta(x, y).$$

Given any $r \in \Sigma^n$ which is promised to be at most ρ-close to some codeword $c \in C$ (where $\rho < \delta/2$), that c is uniquely determined. Finding that c is the algorithmic problem of decoding the code C. Closely related to this problem is the algorithmic problem of determining if any arbitrary r is within distance ρ of some codeword c.

---

*This survey appeared as a guest column in SIGACT News 2016.

†Department of Mathematics & Department of Computer Science, Rutgers University, Piscataway NJ 08854, USA. Email: swastik.kopparty@gmail.com. Research Supported in part by a Sloan Fellowship and NSF grants CCF-1253886 and CCF-1540634.

‡Department of Computer Science, Rutgers University, Piscataway NJ 08854, USA. Email: shubhangi.saraf@gmail.com. Research Supported in part by NSF grants CCF-1350572 and CCF-1540634.

In recent years, it has been of great interest to understand what algorithmic problems on codes can be solved *locally*. This brings us to the following three key definitions.

1. A code C is called *locally testable* with q queries if there is a randomized algorithm T, which when given oracle access to $r \in \Sigma^n$, makes at most q queries to r, and has the following behavior[1]:
   - If $r \in C$, then $\Pr[T \text{ accepts}] = 1$,
   - $\Pr[T \text{ rejects}] \geq \frac{1}{4} \cdot \Delta(r, C)$.
2. A code C is called *locally correctable* from ρ-fraction errors with q queries if there is a randomized algorithm D, which when given oracle access to $r \in \Sigma^n$ such that $\Delta(r, c) < \rho$ for $c \in C$, and given $i \in [n]$, makes at most q queries to r and has the following behavior:
   - $\Pr[D \text{ outputs } c_i] \geq 0.9$.
3. A code C along with an encoding map $E : \Sigma^k \to C$ is called *locally decodable* from ρ-fraction errors with q queries, if there is a randomized algorithm D, which when given oracle access to $r \in \Sigma^n$ such that $\Delta(r, E(s)) < \rho$ for $s \in \Sigma^k$, and given $j \in [k]$, makes at most q queries to r and has the following behavior:
   - $\Pr[D \text{ outputs } s_j] \geq 0.9$.

The difference between LDCs (locally decodable codes) and LCCs (locally correctable codes) is that an LDC allows recovery of a coordinate of the original data s, while an LCC allows recovery of a coordinate of the original codeword c. Informally, we will call a code which is either locally testable, locally decodable or locally correctable with $o(k)$ queries a "local" code.

The most basic example of a local code is the Hadamard code. Here $\Sigma = \mathbb{F}_2$ (the field of 2 elements) and $n = 2^k$. The n coordinates are indexed by vectors $v \in \mathbb{F}_2^k$. Given a message $s \in \mathbb{F}_2^k$, the v coordinate of the encoding $E(s)$ is given by:

$$(E(s))_i = \langle s, v \rangle.$$

Let us quickly see how to locally correct this from 0.01-fraction errors using just 2 queries. Given $r \in \Sigma^n$ which is 0.01-close to the codeword c, if we want to recover the v coordinate of c, we do the following: pick w uniformly at random from $\mathbb{F}_2^k$, and output $r_w + r_{v+w}$. We have that $\Pr_w[r_w \neq c_w] \leq 0.01$ and $\Pr_w[r_{v+w} \neq c_{v+w}] \leq 0.01$. Thus with probability at least 0.98, both those events do not occur, and in that case we have that the output of the algorithm equals:

$$r_w + r_{v+w} = c_w + c_{v+w} = \langle s, w \rangle + \langle s, v + w \rangle = \langle s, v \rangle = c_v.$$

Hadamard codes are also locally testable, via a simple 3-query test: sample $v, w \in \mathbb{F}_2^k$ uniformly at random, and check that $r_v + r_w = r_{v+w}$. The analysis of this test is quite a bit more difficult than for local decodability, and we do not prove it here.

The focus of this survey is local codes in the constant rate regime. For a long time, the only known family of local codes in this regime were constant-variable Reed-Muller codes over large

[1]The constant 1/4 in the requirement on the rejection probability was just chosen for concreteness. Given a code which is locally testable with the 1/4 replaced by ϵ, one can amplify the rejection probability by independent repetition $O(1/\epsilon)$ times and make it satisfy the definition with the constant 1/4.

fields, which are codes based on low degree polynomials. These codes could achieve any constant rate $R < 1/2$, constant distance δ, and local testability/correctability with about $O(n^{-\log(1/R)})$ queries, where n is the length of the code. Recently, a number of new constructions of local codes of constant rate were discovered, based on a variety of different ideas. These codes improved the rate-distance tradeoffs achievable by LTCs (locally testable codes) and LDCs, while also reducing the query complexity significantly. Today we know:

- LTCs which can achieve any constant rate $R < 1$, any constant distance $\delta < 1 - R$, and with query complexity $O((\log n)^{O(\log \log n)})$,
- LDCS and LCCs which can achieve any constant rate $R < 1$, any constant distance $\delta < 1-R$, and with query complexity $O(2^{\sqrt{\log n \log \log n}})$.

On the other hand, it is known that LDCs/LCCs with constant rate require query complexity at least $\Omega(\log n)$. For LTCs, there is no lower bound on query complexity known, and for all we know LTCs with constant rate, constant distance and constant query complexity could exist!

In this survey, will discuss all known local codes in the constant rate regime. Finally, we will also give one application, to the construction of PCPs for SAT with constant rate.

Remarks:

- Throughout this article, we only construct codes with large alphabets. In all cases the alphabet size can be reduced to binary by the process of code concatenation with negligible loss of parameters.
- For a code of length n and dimension k, expressing the query complexity in terms of n and k are essentially equivalent since in the high rate regime n and k are within a constant factor of each other. We use both in this article.
- We henceforth will not talk about LDCs in this article. All the LCCs we talk about are also LDCs with the same query complexity for a suitable choice of encoding map.
- All the local testing and local correction algorithms we describe can also be made to run very efficiently (in particular these are sublinear time algorithms). The running time is always polynomial in the query complexity.

We will not discuss any of the many significant advances in the subconstant rate regime. For locally testable codes and PCPs it is known that with just slightly subconstant rate one can have constant query complexity [Din07, BS08a]. For locally decodable codes, it is known that constant query complexity can be achieved for codes where n is subexponential in k [Yek08, Efr12, DGY11]. For locally correctable codes, nothing beyond the codes discussed in this survey is known.

2 Reed-Muller Codes

In this section we describe the classical Reed-Muller codes based on multivariate polynomials and why they are locally testable and locally correctable.

Let q be a prime power, let $m = O(1)$ be a fixed integer, and let $\delta \in (0, 1)$. Let $d = (1-\delta) \cdot q$. The Reed-Muller code associated with these parameters is given by the space of functions $f : \mathbb{F}_q^m \to \mathbb{F}_q$

which are computed by polynomials of total degree at most d. By a well-known bound, often called the Schwartz-Zippel lemma, on the number of zeroes of polynomials in product sets, this code has distance δ.

When $m = 1$, these codes are also called Reed-Solomon codes, and they have a number of wonderful properties. In particular, they achieve the optimal tradeoff between rate R and distance δ (satisfying $R + \delta = 1$). Increasing m worsens the tradeoff, with the relationship $R = \frac{1}{m!}(1-\delta)^m$. In particular, once $m > 1$, we must have rate $< 1/2$. Classically the advantage of increasing m was that it reduced the alphabet size (as a function of the length of the code). But from our point of view, a far more important advantage of increasing m is that it endows the code with local properties.

The emergence of locality in Reed-Muller codes is based on the following simple observation: the restriction of a degree d multivariate polynomial to a line is a degree d univariate polynomial. The results here are due to [BFLS91, RS96, FS95].

Local Correction of Reed-Muller codes: Given a received word $r : \mathbb{F}_q^m \to \mathbb{F}_q$ such that r is close in Hamming distance to the codeword corresponding to the polynomial $f(X_1, \ldots, X_m)$, the classical local correction algorithm works as follows. Given a coordinate $\mathbf{a} \in \mathbb{F}_q^m$, we want to recover the "corrected" symbol at coordinate $\mathbf{a}$, namely $f(\mathbf{a})$. The algorithm picks a random direction $\mathbf{b} \in \mathbb{F}_q^m$ and looks at the restriction of r to coordinates in the line $L = \{\mathbf{a} + \mathbf{b}t \mid t \in \mathbb{F}_q\}$. With high probability over the choice of $\mathbf{b}$, r and f will agree on many positions of L. Now $f|_L$ is simply the vector consisting of evaluations of the univariate polynomial $Q(T) = f(\mathbf{a} + \mathbf{b}T) \in \mathbb{F}_q[T]$, which is of degree $\leq d$. Thus $r|_L$ gives us q "noisy" evaluations of a polynomial $Q(T)$ of degree $\leq (1-\delta) \cdot q$; this enables us to recover $Q(T)$. Evaluating $Q(T)$ at $T = 0$ gives us $P(\mathbf{a})$, as desired. Notice that this decoding algorithm makes q queries, which is $O(n^{1/m})$.

Local testing of Reed-Muller codes: The local tester for Reed-Muller codes is quite natural given the above discussion. Given a received word $r : \mathbb{F}_q^m \to \mathbb{F}_q$, one picks a line ℓ uniformly at random in $\mathbb{F}_q^m$, and tests that the restriction $r|_\ell$ is computed by a degree d univariate polynomial.

To show that this is a local tester, we would at the very least need the following *local characterization* to hold: a function $f : \mathbb{F}_q^m \to \mathbb{F}_q$ is of degree at most d if and only if for every line ℓ in $\mathbb{F}_q^m$, the restriction $f|_\ell$ is of degree at most d. The only if direction was noted above, and holds for all d. The if direction turns out to be true, but only for $d \leq (1 - \frac{1}{p}) \cdot q$, where p is the characteristic of $\mathbb{F}_q$. We refer the reader to [FS95] for a proof. In particular, if q is a prime, then the above local characterization holds for all $d < q$. At the other extreme, if $q = 2^\ell$, then the local characterization holds only when $d < q/2$ – this seems quite unfortunate, but will be turned to our advantage in a later section.

Next we describe how how one deduces the local testability of Reed-Muller codes from the local characterization. The strategy for this proof originated in [BLR93], and it was greatly clarified by Kaufman and Sudan [KS08].

Given a function r which fails the local test with probability at most η, we want to show that r is $O(q^2 \cdot \eta)$-close to some codeword f of the Reed-Muller code. How do we find a codeword which r is close to? The idea is to define the function $f : \mathbb{F}_q^m \to \mathbb{F}_q$ to be the "locally corrected version" of r as follows. For each $x \in \mathbb{F}_q^m$, and each line L passing through x, and look at the univariate polynomial $Q_{L,x}$ of degree at most d (if any) which agrees with $r|_{L \setminus \{x\}}$, and let $y_{L,x} \in \mathbb{F}_q$ be the value of $Q_{L,x}$ at the point x (i.e., $y_{L,x}$ is the unique value for $r(x)$ (if any) that makes $r|_L$ a

univariate polynomial of degree at most d). Then $f(x)$ is defined to equal the most popular value of $y_{L,x}$ as L varies over all lines passing through x. Given this definition, one can show that f is (2η)-close to r. The rest of the proof shows that if $\eta < O(\frac{1}{q^2})$, then for every line L in $\mathbb{F}_q^m$, the univariate function $f|_L$ has degree at most d (for a very illuminating proof of this, see [KS08]). By the local characterization theorem, this means that f is a polynomial of total degree at most d, and thus r is close to some codeword of the Reed-Muller code, as desired.

Summarizing, we see that Reed-Muller codes of length n are locally testable and locally correctable with query complexity $n^{O(1/m)}$. Thus for constant m, we get constant rate $< \frac{1}{m!}$, constant distance, and query complexity n^β for any constant $\beta > 0$. In particular, we only get locality properties for rates $< 1/2$.

See also [RS97, AS03, STV01] for related results.

3 Lifted Reed-Solomon codes

In this section, we describe the Lifted Reed-Solomon codes of [GKS13]. These codes are very closely related to Reed-Muller codes, and improve on them by achieving rate arbitrarily close to 1 (while preserving their local testability and local correctability). These codes were discovered as part of a long and beautiful line of research studying affine-invariant codes, which was initiated in [KS08].

Let q be a prime power, let $m = O(1)$ be a fixed integer, and let $\delta \in (0, 1)$. Let $d = (1-\delta)\cdot q$. The Lifted Reed-Solomon code associated with these parameters is the space of all functions $f : \mathbb{F}_q^m \to \mathbb{F}_q$ such that for everly line $L = \{\mathbf{a} + \mathbf{b}t : t \in \mathbb{F}_q\}$, the restriction $f|_L$ (when viewed as a univariate function from $\mathbb{F}_q \to \mathbb{F}_q$) agrees with a polynomial of degree at most d.

An immediate observation is that any codeword f of the Reed-Muller code of m-variate polynomials of degree d over $\mathbb{F}_q$ is also a codeword of the Lifted Reed-Solomon code. Are there any other codewords? The local characterization theorem for Reed-Muller codes mentioned in the previous section shows that when $d \leq (1 - \frac{1}{p}) \cdot q$, then there are none; for such d, Lifted Reed-Solomon codes are precisely the same as Reed-Muller codes. However for $d > (1 - \frac{1}{p}) \cdot q$ (i.e., for $\delta < 1/p$) there are more codewords in the Lifted Reed-Solomon code.

By design, Lifted Reed-Solomon codes have rich local structure, and this quite easily implies their local correctability and local testability. The local correction algorithm is exactly as in the case of Reed-Muller codes: to recover the value at the point $\mathbf{a}$, one queries a random line L through $\mathbf{a}$, decodes that to the nearest univariate polynomial of degree at most d, and outputs the value of that univariate polynomial at $\mathbf{a}$. The local testing algorithm is also exactly as in the case of Reed-Muller codes: one queries a random line and tests if it agrees with a polynomial of degree at most d. The analysis of this local test again goes via a local characterization in terms of restrictions to lines. However in this case, the local characterization doesn't need to be proved: the local characterization is precisely the definition of Lifted Reed-Solomon Codes! The reduction from the local testing to the local characterization works exactly as in the case of Reed-Muller codes, and this gives the full analysis of the local tester.

What remains to understand is the rate and the distance of Lifted Reed-Solomon codes. The distance turns out to be $\delta - o(1)$: this follows from the fact that for any function f of the Lifted Reed-Solomon code with α-fraction nonzero values, with $1-o(1)$ probability over a randomly chosen line L, the restriction $f|_L$ will have $(\alpha \pm o(1))$-fraction nonzero values. However, if f is a codeword of the Lifted Reed-Solomon code, $f|_L$ is a polynomial of degree $(1-\delta)q$, and thus must have either 0-fraction or at least δ-fraction nonzero coordinates.

Finally we come to the rate, which is the most interesting. If the characteristic p is constant, then it turns out that there are significantly more codewords in the Lifted Reed-Solomon code than there are in the Reed-Muller code: so many more that the rate of the Lifted Reed-Solomon code can approach 1.

Let us demonstrate in the case $m = 2$ and when F_q is of characteristic 2. Which functions $f : \mathbb{F}_q^2 \to \mathbb{F}_q$ lie in the Lifted Reed-Solomon code? This question is given a complete answer in [GKS13] in terms of the polynomial representation $f(X,Y) = \sum_{0 \le i,j < q} a_{ij} X^i Y^j$. Here we are interested in showing that the dimension is large, and so it will suffice for us to show that there are many linearly independent elements in the code. To do this, we will study when a monomial $g(X,Y) = X^i Y^j$ lies in the code. Note that if we restrict g to a line $\ell(T) = (\alpha_1 T + \alpha_0, \beta_1 T + \beta_0)$, we get the function

$$\begin{aligned} g|_\ell(T) &= (\alpha_1 T + \alpha_0)^i (\beta_1 T + \beta_0)^j \\ &= \sum_{r \le i} \sum_{s \le j} \alpha_1^r \alpha_0^{i-r} \beta_1^s \beta_0^{j-s} \binom{i}{r} \binom{j}{s} T^{r+s}. \end{aligned}$$

This function will equal a univariate polynomial of degree at most d at all points of $\mathbb{F}_q$ if, when we reduce it mod $T^q - T$, we see no monomials of degree $> d$. Reducing the above polynomial mod $T^q - T$ amounts to replacing T^{r+s} in the above expression with $T^{r+s \pmod{q}^*}$ (where $a \pmod{q}^* = 0$ if $a = 0$ and $a \pmod{q}^* = b \in \{1, \ldots, q-1\}$ if $a \neq 0$ and $a = b \pmod{q-1}$). This will happen if i, j satisfy the following criterion: for every $r \le i, s \le j$, if[2] $\binom{i}{r} \neq 0 \mod 2$ and $\binom{j}{s} \neq 0 \mod 2$, then $r + s \pmod{q}^* \le d$. Via Lucas' theorem (which gives a characterization of when $\binom{a}{b} = 0 \mod 2$, we deduce that the monomial $X^i Y^j$ lies in the code if (i,j) lies in the set:

$$S = \{(i,j) \mid \forall r \le_2 i, j \le_2 s, r + s \pmod{q}^* \le d\},$$

where $a \le_2 b$ means that set of coordinates that equal 1 in the binary representation of a is a subset of the set of coordinates that equal 1 in the binary representation of b. Finally, an analysis of the set S shows that its size is $\ge (1 - \epsilon_\delta) \cdot q^2$, where $\epsilon_\delta \to 0$ as $\delta \to 0$. Thus the dimension of the lifted Reed-Solomon is at least $(1 - \epsilon_\delta) \cdot q^2$. Thus the rate of the Lifted Reed-Solomon code is at least $1 - \epsilon_\delta$.

A similar phenomenon happens for general m. Setting parameters suitably, we get for every $\beta > 0$, codes with arbitrarily large length n, distance δ, rate $1 - \epsilon_\delta$ that are locally testable and locally correctable with $O(n^\beta)$ queries.

See also [Guo13] and [GK14] for related results.

4 Multiplicity Codes

In this section, we describe the multiplicity codes of [KSY14] and why they are locally correctable. These codes are another family of codes very closely related to Reed-Muller codes - they are polynomial based codes and they build upon and extend the classical Reed-Muller codes. Just like the family of Lifted Reed-Solomon codes, multiplicity codes also improve upon Reed-Muller codes by achieving rate arbitrarily close to 1 and being locally correctable with $O(n^\beta)$ queries (and time) for arbitrary small constant β.

[2] This is where the characteristic is playing a role.

We first introduce the simplest example of multiplicity codes, which already achieves a better rate than than any choice of Reed-Muller codes described earlier, while being locally self-correctable with sublinear queries.

Construction of bivariate Multiplicity codes: Let q be a prime power, let $\delta > 0$ and let $d = 2(1-\delta)q$ (which is twice what it was in the Reed-Muller example). The multiplicity code of **order 2** evaluations of degree d bivariate polynomials over $\mathbb{F}_q$ is the code defined as follows. The coordinates are indexed by $\mathbb{F}_q^2$ (so $n = q^2$) and the codewords are indexed by bivariate polynomials of degree at most d over $\mathbb{F}_q$. However the alphabet will now be $\mathbb{F}_q^3$. The codeword corresponding the polynomial $P(X,Y)$ is the vector

$$C(P) = \langle (P(\mathbf{a}), \frac{\partial P}{\partial X}(\mathbf{a}), \frac{\partial P}{\partial Y}(\mathbf{a})) \rangle_{(\mathbf{a}) \in \mathbb{F}_q^2} \in (\mathbb{F}_q^3)^{q^2}.$$

In words, the $\mathbf{a}$ coordinate consists of the evaluation of P and its partial derivatives $\frac{\partial P}{\partial X}$ and $\frac{\partial P}{\partial Y}$ at $\mathbf{a}$. Because two distinct polynomials of degree at most d can agree *with multiplicity 2* on at most $d/2q$-fraction of the points in $\mathbb{F}_q^2$, this code has distance $\delta = 1 - d/2q$. Since the alphabet size is now q^3, the message length k equals the number of q^3-ary symbols required to specify a polynomial of degree at most d; this is clearly $\binom{d+1}{2}/3$. Thus the rate of this code is $(\binom{d+1}{2}/3)/q^2 \approx 2(1-\delta)^2/3$.

Summarizing the differences between this multiplicity code with the family of Reed-Muller codes described earlier: (a) instead of polynomials of degree $(1-\delta)q$, we consider polynomials of degree double of that, (b) instead of evaluating the polynomials, we take their "order-2" evaluation. This yields a code with the same distance, while the rate improved from $< 1/2$ to nearly $2/3$.

Local Self-Correction of bivariate Multiplicity codes: Given a received word $r \in (\mathbb{F}_q^3)^{q^2}$ such that r is close in Hamming distance to the codeword corresponding to $P(X,Y)$, we will show how to locally self-correct. Given a point $\mathbf{a} \in \mathbb{F}_q^2$, we want to recover the "corrected" symbol at coordinate $\mathbf{a}$, namely $(P(\mathbf{a}), \frac{\partial P}{\partial X}(\mathbf{a}), \frac{\partial P}{\partial Y}(\mathbf{a}))$. Just as in the example of Reed-Muller codes, the algorithm picks a random direction $\mathbf{b} = (b_1, b_2) \in \mathbb{F}_q^2$ and looks at the restriction of r to coordinates in the line $L = \{\mathbf{a} + \mathbf{b}t \mid t \in \mathbb{F}_q\}$. With high probability over the choice of $\mathbf{b}$, we will have that $r|_L$ and $C(P)|_L$ agree in many locations. Our intermediate goal will be to recover the univariate polynomial[3] $Q(T) = P(\mathbf{a} + \mathbf{b}T)$. The important observation is that for every $t \in \mathbb{F}_q$, the $\mathbf{a} + \mathbf{b}t$ coordinate of $C(P)$ completely determines both the value and the 1st derivative of the univariate polynomial $Q(T)$ at the point t; indeed, by the chain rule we have:

$$(Q(t), \frac{\partial Q}{\partial T}(t)) = (P(\mathbf{a}+\mathbf{b}t), b_1 \frac{\partial P}{\partial X}(\mathbf{a}+\mathbf{b}t) + b_2 \frac{\partial P}{\partial Y}(\mathbf{a}+\mathbf{b}t)).$$

Thus our knowledge of $r|_L$ gives us access to q "noisy" evaluations of the polynomial $Q(T)$ and its derivative $\frac{\partial Q}{\partial T}(T)$, where $Q(T)$ is of degree $\leq 2(1-\delta)q$. It turns out that this is enough to recover the polynomial $Q(T)$. Evaluating $Q(T)$ at $T = 0$ gives us $P(\mathbf{a})$. Evaluating the derivative $\frac{\partial Q}{\partial T}(T)$ at $T = 0$ gives us the directional derivative of P at $\mathbf{a}$ in the direction $\mathbf{b}$ (which equals $b_1 \frac{\partial P}{\partial X}(\mathbf{a}) + b_2 \frac{\partial P}{\partial Y}(\mathbf{a})$). We have clearly progressed towards our goal of computing the tuple $(P(\mathbf{a}), \frac{\partial P}{\partial X}(\mathbf{a}), \frac{\partial P}{\partial Y}(\mathbf{a}))$, but we are not yet there. The final observation is that if we pick another direction $\mathbf{b}'$, and repeat the

[3] Unlike in the Reed-Muller case, here there is a distinction between recovering $Q(T)$ and recovering $C(P)|_L$. It turns out that recovering $C(P)|_L$ given only $r|_L$ is impossible.

above process to recover the directional derivative of P at $\mathbf{a}$ in direction $\mathbf{b}'$, then the two directional derivatives of P at $\mathbf{a}$ in directions $\mathbf{b}, \mathbf{b}'$ together suffice to recover $\frac{\partial P}{\partial X}(\mathbf{a})$ and $\frac{\partial P}{\partial Y}(\mathbf{a})$, as desired. This algorithm makes $2q$ queries, which is $O(k^{1/2})$.

General Multiplicity codes: The basic example of a multiplicity code above already achieves rate $R > 1/2$ while allowing local decoding with sublinear query complexity. To get codes of rate approaching 1, we modify the above example by considering evaluations of all derivatives of P up to an even higher order. In order to locally recover the higher-order derivatives of P at a point $\mathbf{a}$, the decoding algorithm will pick many random lines passing through $\mathbf{a}$, try to recover the restriction of P to those lines, and combine all these recovered univariate polynomials in a certain way. To reduce the query complexity to $O(k^\beta)$ for small β, we modify the above example by considering multivariate polynomials in a larger number of variables m. The local decoding algorithm for this case, in order to locally recover at a point $\mathbf{a} \in \mathbb{F}_q^m$, decodes by picking random lines passing through $\mathbf{a}$; the reduced query complexity occurs because lines (with only q points) are now much smaller relative to a higher dimensional space $\mathbb{F}_q^m$. Increasing both the maximum order of derivative taken and the number of variables simultaneously yields multiplicity codes with the arbitrarily large rate < 1, constant distance, which are locally correctable with query complexity $O(n^\beta)$ for arbitrary constant $\beta > 0$.

See also [Kop15, Kop14] for related results.

5 Tensor codes

The tensor product is a general operation on codes which produces long codes out of short codes (with some loss in the rate and distance). It turns out that the tensored codes also admit non-trivial local testers. The remarkable aspect here is that one can produce locally testable codes by starting with *any code* and tensoring it. This phenomenon was first discovered by Ben-Sasson and Sudan [BSS06], and further investigated by a number of works, culminating with an analysis by Viderman [Vid15] which was flexible enough to produce LTCs of arbitrarily high rate and query complexity $O(n^\beta)$ for every constant $\beta > 0$.

Let $\mathbb{F}$ be a field, and let D be a finite set. Let C be a linear code contained in $\mathbb{F}^D$ (thus each codeword has length $|D|$, and each codeword can be viewed as a function from D to $\mathbb{F}$).

We define the tensor power of C, denoted C^2 to be the set of all functions $f : D \times D \to \mathbb{F}$ such that: (1) for each $x \in D$, $f(x, \cdot) \in C$, and (2) for each $y \in D$, $f(\cdot, x) \in C$. Informally, the codewords of C^2 are those two-dimensional arrays such that each row-restriction and each column restriction lies in C. One defines the m'th tensor power similarly, in terms of m-dimensional arrays.

The following properties of tensor powering are easy to check. Let R denote the rate of C and let δ denote the distance of C. (1) the rate of C^m equals R^m, (2) the distance of C^m equals δ^m.

There is a natural local test for the tensor code C^2, which arises from the definition of the code. Given a received word $r : D^2 \to \mathbb{F}$, one picks a uniformly random row or column and checks that r restricted to that row or column is a codeword of C. Clearly every codeword of C^2 passes this test with probability 1. If this test passes with high probability, then many rows and many columns of r are codewords of C. One can then show (using simple linear algebra) that there is a codeword of C^2 which is consistent with these rows and columns (and thus close to r). This shows that the above test indeed is a local test for C^2.

For larger m, such a simple analysis does not seem to work. Instead, we first analyze a different "planes-based" test for the case $m = 3$, which will have a key "robustness" property. We will then reduce the case of general m to the case of $m = 3$ by recursion (the feasibility of this recursion relies on the robustness of the test).

Now fix $m = 3$. The following test is very natural. Given a received word $r : D^3 \to \mathbb{F}$, one picks a uniformly random 2-dimensional plane P in D^3 (i.e., either $\{x\} \times D \times D$, or $D \times \{y\} \times D$, or $D \times D \times \{z\}$), and checks that $r|_P$ is a codeword of C^2. Again, every codeword of C^3 passes this test with probability 1. The key claim for us is that this is a robust local test, namely:

$$\Delta(r, C^3) \leq O(\delta^3) \cdot \mathbb{E}_P[\Delta(r|_P, C^2)].$$

In other words, if r is far from C^3, then for many planes P, $r|_P$ is far from C^2. This lets us reduce the question of testing proximity to C^3 to the question of testing proximity to C^2, and lends itself very nicely to a recursion. So for example local testing of $C^9 = (C^3)^3$ reduces to local testing of $(C^3)^2 = C^6 = (C^2)^3$, which reduces to local testing of $(C^2)^2$, which can be done with $|D|^4$ queries. More generally, C^{3^i} (which has length $|D|^{3^i}$) can be tested with with $|D|^{2^i}$ queries. If we started with a code C with distance $\Omega(1) < \delta \ll \frac{1}{3^i}$ and rate $\gg 1 - \frac{1}{3^i}$, then for sufficiently large constants i the code C^{3^i} can have arbitrarily high rate, constant distance, block-length n and can be locally tested with n^β queries for constant $\beta > 0$. This gives the desired high-rate locally testable codes.

Viderman's analysis [Vid15] of the robust local test goes via an insightful analysis of the kinds of errors in a received word. For a given r, the point (x, y, z) is called *fixable* if there is a value $\alpha \in P$ such that for every plane P containing (x, y, z) (there are exactly three such planes), the codeword of C^2 closest to $r|_P$ takes the same value value α at (x, y, z). Otherwise the point (x, y, z) is called unfixable; this is indicative of some large scale dischord and deep-rooted resentment within the received word - two huge planes have different opinions about what should be the true value at (x, y, z). The crux is then to argue that the unfixable points are all contained in very few planes. This uses the crucial property that if we have two planes P, P', and two codewords of C^2, g defined on P and h defined on P', and if $g|_P$ and $h|_{P'}$ disagree at one point, then they must disagree on $\delta|D|$ points. This is because the $g|_{P \cap P'}$ and $h|_{P \cap P'}$ are both codewords of C. For details, see [Vid15].

6 Constant rate PCPs for SAT

One of the early motivations for the study of local testing and local decoding was the strong connection between these notions and the notion of probabilistically checkable proofs. In this section we describe an application from [BSKK+13] of the previous results on testing of tensor codes back to the theory of PCPs. The main result is a construction of *constant rate* PCPs for SAT.

A PCP is a format for writing witnesses for NP languages so that the validity of the witness can be checked locally. The famous PCP theorem [AS98, ALM+98] states that for any NP language L, there is a randomized polynomial time verifier $V(x, \pi)$ such that:

- the proof length (i.e., the length of π) is polynomial in the length of x,
- $V(x, \pi)$ makes only constantly many queries into the proof π,
- if $x \in L$, then there exists a proof π such that $V(x, \pi)$ always accepts,

- if $x \notin L$, then for every candidate proof π',

$$\Pr[V(x, \pi') \text{ accepts}] < 0.5$$

This theorem is nontrivial even if we allow the proof length to be arbitrary, and we bound the number of queries made by V into π by any sublinear function of the length of x! We will focus on making the proof length linear (or near-linear) in the length of x, while making the query complexity sublinear.

Below we consider a particular NP language, Grid-CSP, and show how to give a PCP with short proof length for this language.

Definition 6.1 (Grid-CSP)**.** *Let a be an integer and let $n = a^2$. We have a collection of n variables $(v_{i,j})_{1 \le i,j \le a}$, which we view as arranged on an $a \times a$ grid. For each $(i,j) \in [a]$, we have a 5-ary predicate $C_{i,j} : \{0,1\}^5 \to \{0,1\}$.*

Our goal is to find an assignment to the variables so that for all $(i,j) \in [a] \times [a]$,

$$C_{i,j}(v_{i,j}, v_{i-1,j}, v_{i+1,j}, v_{i,j-1}, v_{i,j+1}) = 1.$$

(For the edge cases $i = 1$ or a, or $j = 1$ or a, we assume that $C_{i,j}$ is always 1).

The classical NP witness for Grid-CSP is simply a satisfying assignment for the $v_{i,j}$. We will begin by seeing how classical polynomial-based codes can give a $\tilde{O}(\sqrt{n})$-query probabilistically checkable witness of size $O(n \log n)$ for this problem. We will then describe some ideas that go into the construction of PCPs for general SAT instances of size n where the size of the witness is $O(n)$, thus achieving constant rate.

We remark that a construction of a $\mathsf{polylog}(n)$-query PCP (of polynomial rate) for arbitrary NP languages can be given along these lines. There are two changes: one needs to work with higher dimensional grids (dimension nearly $\log n$), and one needs to consider a slight variation of the Grid-CSP which is more amenable to efficient reductions from general NP complete problems.[4]

A key component in the construction of the PCP for grid-CSP is the notion of a low-degree extension. Let p be a prime of size approximately $10a$, and let us view $[a]$ as a subset of $\mathbb{F}_p$. Given an assignment in $\{0,1\}^n$ to the variables $(v_{i,j})_{i,j \in [a]}$, we consider the polynomial $f(X,Y) \in \mathbb{F}_p[X,Y]$ of degree $\le a-1$ in each variable such that $f(i,j) = v_{i,j}$. Then the entire evaluation table of f on $\mathbb{F}_p \times \mathbb{F}_p$ is called the low-degree extension of v.

The first part of the PCP will be the low-degree extension of a satisfying assignment v. We will augment this with some further information that will allow us to quickly check that the first part really is the low-degree extension of a satisfying assignment v.

Let $r : \mathbb{F}_p \times \mathbb{F}_p \to \mathbb{F}_p$ be a given function. It is supposed to be the low-degree extension of a satisfying assignment. We want to perform some low-query test on r, such that if the test passes with high probability, it must mean that there exists a satisfying assignment to the given Grid-CSP. Our test will actually have a stronger property: if the test passes on r with high probability, it means that r is close to the low-degree extension of a satisfying assignment.

The verifier's test will have three subtests.

[4] With the Grid-CSP problem as currently defined, a SAT formula of size m can be reduced to a Grid-CSP problem of size m^2, and the PCP we construct shows that it can be tested using $\tilde{O}(m)$ queries, which is trivial for the original SAT formula. Nevertheless, there is a slight variation of Grid-CSP which does allow for more size-efficient reductions from SAT, and it is this variation that gets used in the constant rate PCP constructions.

1. In r passes the first subtest with high probability, it will mean that that r is close to some polynomial f with individual degrees at most $a-1$.

2. If r passes the second subtest with high probability, it will mean that f is $\{0,1\}$-valued on $[a] \times [a]$. Note that we are performing queries on r, but we are deducing properties of f.

3. If r passes the third subtest with high probability, it will mean that $f|_{[a]\times[a]}$ is a satisfying assignment to the given Grid-CSP with high probability.

We now describe how to perform the different subtests.

The first subtest is exactly the local testing problem for the code of evaluations of polynomials with individual degree at most $a-1$ in each variable. This code is a tensor code: it is the 2-wise tensor product of Reed-Solomon codes. By the testability of tensor codes, this code is locally testable with a queries, and (by definition of local testing) this local test achieves what the first subtest is supposed to achieve.

The second and third subtests will follow a similar framework. To check that the values of f on $[a] \times [a]$ are 0 or 1, consider the polynomial $g(X,Y) = f(X,Y)^2 - f(X,Y)$. We wish to check that g vanishes on $[a] \times [a]$. To check that the constraints are satisfied, we first interpolate a polynomial $Q(X, Y, B_1, B_2, B_3, B_4, B_5)$ with individual degrees at most $(a-1, a-1, 1, 1, 1, 1, 1)$ such that for all $i, j \in [a]$ and all $b_1, b_2, b_3, b_4, b_5 \in \{0,1\}$, we have:

$$Q(i, j, b_1, b_2, b_3, b_4, b_5) = 1 - C_{i,j}(b_1, b_2, b_3, b_4, b_5).$$

Then consider the polynomial $h(X,Y) = Q(X, Y, f(X,Y), f(X-1,Y), f(X+1,Y), f(X,Y-1), f(X,Y+1))$. We wish to check that h vanishes on $[a] \times [a]$.

Summarizing, we want to check that a certain polynomial vanishes on a set of the form $S \times S$. In our setting we do not have oracle access to the polynomial, but as we see below we do have access to a noisy version of it. Suppose r is δ-close to f. Then the function $\tilde{g} : \mathbb{F}_p \times \mathbb{F}_p \to \mathbb{F}_p$ defined by:

$$\tilde{g}(x,y) = r(x,y)^2 - r(x,y)$$

is δ-close to g. Similarly, the function $\tilde{h} : \mathbb{F}_p \times \mathbb{F}_p \to \mathbb{F}_p$ defined by:

$$\tilde{h}(x,y) = Q(x, y, r(x,y), f(x-1,y), f(x+1,y), f(x,y-1), f(x,y+1)),$$

is 5δ close to h. Note that we can access the value of $\tilde{g}$ and $\tilde{h}$ at any point in $\mathbb{F}_p \times \mathbb{F}_p$.

The test for vanishing of a polynomial $P(X,Y)$ on $S \times S$ given access to a noisy version $\tilde{P} : \mathbb{F}_p \times \mathbb{F}_p \to \mathbb{F}_p$ can be done in two ways: one based on the sum-check protocol (following [LFKN92]), and the other based on the Combinatorial Nullstellensatz (following Ben-Sasson-Sudan [BS08a]). We briefly describe the Combinatorial Nullstellensatz approach. The underlying theorem is the following.

Theorem 6.1 (Combinatorial Nullstellensatz [Alo99])**.** *Let $P(X,Y)$ be a polynomial of degree at most d in each variable. Let S be a set. Then P vanishes on $S \times S$ if and only there exist polynomials $A(X,Y)$ and $B(X,Y)$ with degrees at most d in each variable such that*

$$P(X,Y) = A(X,Y) \cdot \prod_{s \in S}(X - s) + B(X,Y) \cdot \prod_{s \in S}(Y - s). \tag{1}$$

This motivates the following probabilistically checkable proof that P vanishes on the set $S \times S$: write down the evaluation tables of the polynomials $A(X,Y)$ and $B(X,Y)$ given by the previous theorem. The verifier can check that (1) these evaluation tables given by the prover, $\tilde{A}$ and $\tilde{B}$, are close to polynomials of degree at most d in each variable, and (2) verify that the identity:

$$\tilde{P}(X,Y) = \tilde{A}(X,Y) \cdot \prod_{s \in S}(X - s) + \tilde{B}(X,Y) \cdot \prod_{s \in S}(Y - s) \tag{2}$$

holds for a random point $(x, y) \in \mathbb{F}_p \times \mathbb{F}_p$. If these checks both pass with high probability, then we conclude that there exist polynomial A, B such that Equation (1) holds with high probability when we substitute a random $(x, y) \in \mathbb{F}_p^2$ into it. By the low-degreeness of P, A, B, we conclude that Equation (1) holds identically, and thus that P vanishes on $S \times S$.

This concludes the description of the polynomial-based PCP for Grid-CSP. The verifier accesses $O(a)$ $\mathbb{F}_p$-ary symbols from the proof, leading to a query complexity of $O(a \log p) = O(\sqrt{n} \cdot \log n)$. The length of the PCP is $O(p^2)$ $\mathbb{F}_p$-ary symbols, which is $O(n \cdot \log n)$ bits long. This was almost a constant rate PCP, but we missed for the trivial reason that the bits of our original satisfying assignment were treated as $\mathbb{F}_p$-ary symbols, thus losing a $O(\log p) = O(\log n)$ factor in the length!

This motivates the constant rate PCP construction of [BSKK+13]: the idea is to replace the large-alphabet polynomial codes with a small alphabet code. In order that boolean constraints can be captured by operations on the code, we would like the code to be amenable to algebraic operations such as pointwise addition and pointwise multiplication: this suggests that we use algebraic-geometric codes (AG codes) [Sti93]. Having decided to work with AG codes, a new issue arises. What is the analogue of the operation $x \mapsto x + 1$ in the setting of AG codes? For Reed-Solomon codes, we used a highly underrated property: if $f(X)$ is a codeword of the Reed-Solomon code, then so is $f(X + 1)$. The appropriate generalization is an *automorphism* of the AG code - a permutation σ of the coordinates so that for any codeword of the code f, $f \circ \sigma$ is also a codeword of the code f. Using AG codes with a *transitive automorphism group* in place of the Reed-Solomon codes above, one gets a constant rate PCP for a Grid-like CSP with $O(\sqrt{n})$ query complexity. Furthermore, one can reduce any SAT instance to an instance of this Grid-like CSP with only a constant factor blow-up in the instance size. This gives a PCP for SAT with constant rate and $O(\sqrt{n})$ query complexity. The query-complexity can be further reduced to $O(n^\beta)$ for constant $\beta > 0$, while reducing the rate to a smaller constant, using higher dimensional tensors. For details, see [BSKK+13].

7 Locally Correctable Expander Codes

In this section we describe the expander based construction by [HOW15] of high-rate locally correctable codes with polynomially small query complexity and decoding time. This is another very different construction of a family of codes with rate approaching 1 and with query complexity $O(k^\beta)$ for arbitrary small constant β.

7.1 Construction of locally correctable expander codes

Expander Codes: Let G be a d-regular expander graph on n-vertices that is also a Ramanujan graph. Thus the expansion parameter is λ where $\lambda \leq \frac{2\sqrt{d-1}}{d}$. Let C_{in} be an inner code of block length d. Then the expander code of length $N = nd$ arising from G and C_{in}, which we denote by

$C = C_N(C_{in}, G)$, is the following: First consider the bipartite graph H which is the double cover of G, in other words H has n left vertices and n right vertices (each side is a copy of the vertex set of G) and vertex u on the left is joined to vertex v on the right if and only if (u, v) is an edge of G. The codewords of C are all assignments of symbols to the edges of H so that for every vertex v in H, the assignment restricted to the edges incident to that vertex forms a codeword of C_{in}.

We now describe the specific choice of inner code that will be used in the construction of the high-rate expander based LCCs.

The inner code: Let $\epsilon > 0$ be given. Let C_0 be a linear code over some alphabet Σ with the following properties: The block length of C_0 is d (d will be chosen to be a sufficiently large constant). Moreover C_0 has a smooth local reconstruction procedure[5] with query complexity $q_0 = d^\epsilon$, rate $R_0 > 1 - \epsilon$ and distance δ for some constant δ such that $\delta > 10\frac{1}{\sqrt{d}}$. The multiplicity codes defined earlier satisfy these parameters. Additionally one can construct codes based on affine geometries that also satisfy these parameters (see [HOW15]).

The final code and its parameters: Now consider the code $C = C_N(C_0, G)$ which is the expander code of length $N = nd$ arising from a d-regular Ramanujan graph on n vertices G and the inner code C_0 as described above. Then as long as d is chosen to be a sufficiently large constant, C will be locally correctable with query complexity $O(N^\epsilon)$, rate $r \geq 1 - 2\epsilon$ and distance at least $\delta^2/2$, and the local correction algorithm can tolerate ρ fraction of errors, where ρ is a small positive constant (independent of n).

7.2 Algorithm for local correction and sketch of analysis

On input a codeword edge (u, v) and given query access to a ρ-corrupted codeword c, the algorithm for local correcting of c to recover (u, v) will proceed in two steps. In the first step the algorithm identifies about $q' \approx N^{\epsilon/2}$ edges of the graph such that the true value at those edges would determine the value of the codeword at (u, v), and such that the distribution of each of these edges is extremely close to uniform (and in particular independent of the choice of edge (u, v)). In the second step the algorithm decodes the true value of the codeword at each of those q' locations with very high probability, so that almost surely it gets the value of all of them correctly and hence can decode the true value at i with high probability. We elaborate on the two steps below.

(Step 1:) To decode edge (u, v), the algorithm invokes the smooth decoder for the inner code C_0 at the vertex u. The decoder identifies q_0 edges emanating from the vertex u such that they are a decoding set for (u, v), and each edge is distributed as a uniformly random edge out of u. Say those edges are $(u, v_1), (u, v_2), \ldots, (u, v_{q_0})$. For each such (u, v_i), the algorithm uses the smooth decoder for C_0 at the vertex v_i to identify q_0 edges emanating from the vertex v_i that are a decoding set for (u, v_i) and such that each of the edges is distributed as a uniformly random edge out of v_i. Now for each of the q_0^2 edges identified so far, we repeat the above process and keep going until eventually we identify a q_0-ary tree of depth L (for some suitable parameter L). Observe that the distribution of each leaf is the same as that of the end point of a random walk of length L starting at u. The parameter L is chosen suitably large so that this distribution is extremely close to uniform and so

[5] i.e., each coordinate of a codeword can be recovered by making few uniformly random queries to the remaining coordinates of the codeword.

that the number of leaves, q_0^L, is about $N^{\epsilon/2}$. (By the choice of parameters of the code, this can be done.) Note also that by the way the tree was generated, the true values of codeword symbols at the leaves determines the root (u, v).

(Step 2:) For each leaf (which is distributed almost uniformly), the algorithm decodes each of these with very high probability. It does this as follows. Fix a leaf (u', v'). First the algorithm identifies a q_0-ary tree $\mathcal{T}$ of depth L rooted at (u', v') just as in step 1. The true values at the leaves of this tree determine (u', v'), but now we are in the setting where ρ fraction of the edges are corrupted and we want to show that we can still decode the true value at (u', v') with high probability. There are two main observations.

1. One can prove a Chernoff-like bound on the number of corruptions on any path from root to leaf in the tree $\mathcal{T}$. One first shows that for every path, the total fraction of errors one can encounter on any path is at most some small constant $\gamma < 1/2$, where γ is not much larger than ρ, and this holds with very high probability. By a union bound this must hold for every path with high probability. Thus the true codeword and the ρ-corrupted codeword are "pretty close" on every path of $\mathcal{T}$. This motivates the following distance function. For any two labelings T_1 and T_2 of $\mathcal{T}$, define $\delta(T_1, T_2)$ to be the maximum over all paths p from root to a leaf in the tree of $\delta(p_1, p_2)$ where p_1 is the labeling of path p in T_1 and p_2 is the labeling of path p in T_2, and $\delta(p_1, p_2)$ is just the fractional Hamming distance between these strings. Thus if T_1 is the labeling of $\mathcal{T}$ arising from a true codeword and T_2 is the labeling arising from a ρ-corrupted codeword, then with high probability $\delta(T_1, T_2) \leq \gamma$, where γ is some small constant less than $1/2$.

2. For any two distinct codewords c_1, c_2 which have different values assigned at (u', v') (the root of the tree), there will be a entire path from root to leaf in $\mathcal{T}$ where the codewords differ at every location. Thus if T_1 and T_2 are the labelings of the trees cooresponding to these codewords then $\delta(T_1, T_2) = 1$. This fact is actually quite easy to prove : since the roots differ, and since the children of the root determine the root, one of the children must differ. Continuing in this manner, it is easy to see that there is an entire path on which the labelings differ. By the previous item, and a simple triangle inequality, the labelings for ρ-corrupted versions of c_1 and c_2 would also have very large distance.

Once we have the above two items, it is easy to see that information theoretically the decoding problem can be solved - the value at (u', v') must be the unique $\alpha \in \Sigma$ for which there is a codeword that assigns α to (u', v') and such that the codeword and the given labeling have distance at most γ in the tree metric defined. It can be shown that the running time can be made efficient as well.

8 Locally Correctable Codes with Subpolynomial Query Complexity

In this section we describe the construction by [KMRS16] of high-rate locally correctable codes with *subpolynomially small* query complexity and decoding time. More precisely these codes have rate approaching 1 that are locally correctable from a constant fraction of errors with query complexity $\exp(\sqrt{\log n \cdot \log \log n})$.

8.1 Construction of the codes

There are two main components in the construction of these codes. In the first component one constructs high-rate locally correctable codes with subpolynomially small query complexity, but with subconstant fractional distance and hence tolerating only a subconstant fraction of errors. In the second component one shows how to take such a code and amplify its distance using a technique of Alon and Luby [AL96] to make it a constant, without hurting the rate and query complexity by too much. These codes are first constructed over a very large alphabet (of size roughly $\exp\exp(\sqrt{\log n \cdot \log\log n})$).

High-rate LCCs with subpoly query complexity and subconstant distance: To construct such codes we use multiplicity codes [KSY14] in the non-standard regime of super-constant number of variables. By choosing the number of variables of the underlying polynomials to be $m = \sqrt{\frac{\log n}{\log\log n}}$, we can suitably also set the remaining parameters (such as degree of polynomials and order of derivatives) so that one obtains a code C_M of length n_M such that its relative distance is $\delta_M \geq \Omega\left(\sqrt{\frac{\log\log n}{\log^3 n}}\right)$, rate is $r_M \geq 1 - \frac{1}{\log n}$, alphabet size is $\exp\left(\exp\left(\sqrt{\log n \cdot \log\log n}\right)\right)$, and such that C_M is locally correctable from $\tau_M \geq \frac{1}{10} \cdot \delta_M$ errors with query complexity $2^{O(\sqrt{\log n \cdot \log\log n})}$.

Distance Amplification: Let $\tau > 0$ and $\epsilon > 0$ be any small constants. We will now transform the code C_M into a new code C of distance $\delta \geq 2\tau$ and rate $\approx 1 - (2\tau + \epsilon)$ for any small constant ϵ. The alphabet size will be $\exp\left(\exp O\left(\sqrt{\log n \cdot \log\log n}\right)\right)$, and C will be locally correctable from τ fraction errors with query complexity $2^{O(\sqrt{\log n \cdot \log\log n})}$.

We describe now the transformation. One crucial ingredient will be the existence of combinatorial objects called *samplers* which we now define. Let $G = (U \cup V, E)$ be a bipartite d-regular graph with $|U| = |V| = n$. We say that G is an (α, γ)-sampler if the following holds for every $T \subseteq V$: For at least $1 - \alpha$ fraction of the vertices $s \in U$ it holds that

$$\frac{|E(s,T)|}{d} - \frac{|T|}{n} \leq \gamma.$$

The following lemma follows from the expander mixing lemma.

Lemma 8.1. *For every $\alpha, \gamma > 0$ and every sufficiently large $n \in \mathbb{N}$ there exists a bipartite d-regular graph $G_{n,\alpha,\gamma} = (U \cup V, E)$ with $|U| = |V| = n$ and $d = \text{poly}\left(\frac{1}{\alpha \cdot \gamma}\right)$ such that $G_{n,\alpha,\gamma}$ is an (α, γ)-sampler.*

Such samplers can also be very efficiently constructed.

In our code construction we will use a $(\tau_M, \epsilon/2)$ sampler (for parameter $\epsilon > 0$ as specified above and τ_M being the error tolerance of the family of multiplicity codes) of degree $d = \text{poly}\left(\frac{1}{\tau_M \cdot \epsilon/2}\right)$ on a bipartite graph with n left and right vertices. Let $G_{n,\tau_M,\epsilon/2}$ be such a sampler.

We start with a codeword $c_M$of length n_M of the multiplicity code as described above. Pick a parameter $b = d(1 - (2\tau + \epsilon))$. We partition each codeword into $n = n_M/b$ blocks of length b (let us call these blocks $D_1, D_2, \ldots, D_n$) and encode each of them by a Reed-Solomon code of distance $2\tau + \epsilon$. Thus each codeword of the Reed-Solomon code will have length d. Let the resulting string be c' and let $B_1, \ldots, B_n$ be the resulting codewords of the Reed-Solomon code. We then apply a pseudorandom permutation to the symbols of c' as follows.

Let $G_{n,\alpha,\gamma}$ be the graph as defined above and let $U = \{u_1, \ldots, u_n\}$ and $V = \{v_1, \ldots, v_n\}$ be the left and right vertices of $G_{n,\alpha,\gamma}$ respectively. For each $i \in [n]$ and $j \in [d]$, we write the j-th symbol of B_i on the j-th edge of u_i. Then, we construct new blocks $S_1, \ldots, S_n \in \Sigma = \Sigma_M^d$, by setting the j-th symbol of S_i to be the symbol written on the j-th edge of v_i.

Finally, we define the codeword c of $C \subseteq \Sigma^n$ as follows: the i-th coordinate c_i is the block S_i, reinterpreted as a symbol of the alphabet $\Sigma \stackrel{\text{def}}{=} \Sigma_M^d$.

8.2 Sketch of the decoding algorithm and its analysis

We will first describe an algorithm A for correcting symbols of the outer multiplicity codeword c_M from a ρ-corrupted codeword c. Recall that each codeword symbol c_i is actually a block S_i, and thus we are given as input a codeword with at most ρ fraction of the blocks being corrupted. It is not difficult to see that the algorithm A can be modified to correct symbols of the final codeword c by only a modest blow up in query complexity.

On input coordinate i, the algorithm A uses the local corrector A_M for multiplicity codes as a subroutine. Each time A_M queries a coordinate j, A will try to compute it and give it as input to A_M. If it can succeed in doing this with probability at least τ_M (where τ_M is the error tolerance for the underlying multiplicity code) for a random j, then A_M will succeed in outputting the correct value of the ith coordinate with high probability.

In order to compute the value of the jth coordinate, A does the following. It will try to recover the entire block D_r which contains the jth multiplicity codeword symbol. To do this, it suffices to recover the symbols of B_r with at most $\tau + \epsilon/2$ fraction errors, since then A can decode the corrupted Reed-Solomon codeword to the unique true underlying message which corresponds to D_r. Now if only $\tau + \epsilon/2$ fraction of the neighbors of B_r are corrupted, then by querying the neighbors of B_r, A can indeed find B_r with at most $\tau + \epsilon/2$ fraction errors. Thus it suffices to show that with high probability (at least $1 - \tau_M$), this is indeed the case. This follows immediately from the choice of samplers.

9 High-rate locally testable codes with quasi-polylogarithmic query complexity

In this section we describe the construction by [KMRS16] of high-rate locally testable codes with quasi-polylogarithmic query complexity and testing time. More precisely we show how to construct codes of rate approaching 1, of constant distance, that are locally testable with query complexity $(\log n)^{O(\log \log n)}$.

9.1 Construction of the codes

Here too (just as in the previous section where we described the construction of high-rate LCCs with sub polynomial query complexity), there will be two main steps in the construction. In the first step we construct high-rate LTCs but with only subconstant distance, and in the second step we amplify the distance of the codes using a technique of Alon and Luby [AL96], while preserving their local testability. In particular, starting with an LTC with relative distance δ, we can amplify its relative distance to any $0 < \delta' < 1$ while increasing the query complexity by a factor of $\text{poly}(1/\delta)$ and decreasing the rate by a factor of only $\approx 1 - \delta'$. This technique is useful, since it is easier to

construct LTCs with small relative distance, and this technique allows us to transform such LTCs into ones with better relative distance.

The second step is very similar to that described earlier for LCCs, and for details see [KMRS16]. The main novelty is in the first step where we give an improved construction of LTCs in the sub-constant relative distance regime. More specifically, we show how to construct LTCs of high rate, with relative distance $1/\mathsf{polylog}(n)$, and query complexity $(\log n)^{O(\log\log n)}$. Using the Alon-Luby distance-amplification, this gives LTCs of high rate with constant relative distance and query complexity $(\log n)^{O(\log\log n)}$.

The construction of LTCs in the sub-constant relative distance regime uses an iterative strategy, along the lines of the construction of Meir [Mei09] (which is itself inspired by the PCP and LTC construction of [BS08b],[6] and similar in spirit to several other classical, iterative, brave, yet moderate, algorithms [Gol11]: the zig-zag product [RVW02], undirected connectivity in log-space [Rei08] and the PCP theorem via gap amplification [Din07]). The main new ingredient in the iterative strategy is again the Alon-Luby distance-amplification technique (but in a different setting of parameters).

9.1.1 The iterative construction

We now describe the iterative construction of high-rate LTCs with relative distance $1/\mathsf{polylog}(n)$ and query complexity $(\log n)^{O(\log\log n)}$. The construction starts with a code of very small block length, which can be tested simply by reading the entire received word. Then, the block length is increased iteratively, while the rate, relative distance, and query complexity are not harmed by too much.

More specifically, suppose we want to construct a code with block length n. We start with a code of block length $\operatorname{poly}\log n$, rate $1-\frac{1}{\operatorname{poly}\log n}$, and relative distance $\frac{1}{\operatorname{poly}\log n}$. Then, in each iteration, the block length and rate are (roughly) squared, the relative distance is maintained, and the query complexity is increased by a factor of $\operatorname{poly}\log n$. Thus, after $\approx \log\log n$ iterations, we obtain an LTC with block length n, constant rate, relative distance $\frac{1}{\operatorname{poly}\log n}$, and query complexity $(\log n)^{O(\log\log n)}$, as required.

A single iteration. We now describe the structure of a single iteration. Suppose that, at the beginning of the iteration, the code C has block length n, rate r, relative distance δ, and query complexity q. We apply to C the following two operations:

- **Tensor product:** We replace C with its tensor product C^2. The tensor product C^2 is the code that contains all $n\times n$ matrices M such that that all rows of M and all columns of M are codewords of C.

  The code C^2 has block length n^2, rate r^2, relative distance δ^2, and query complexity $q\cdot\operatorname{poly}(1/\delta)$.

- **Distance amplification:** We apply (a variant of) the Alon-Luby distance-amplification to the code C^2, and amplify the relative distance from δ^2 to δ. The resulting code has block length $O(n^2)$, rate $(1-\delta)\cdot r^2$, relative distance δ, and query complexity $q\cdot\operatorname{poly}(1/\delta)$.

[6] In the constant query regime the LTC construction of Meir [Mei09] achieves the same parameters as that of [BS08b]. However, it seems that the construction of [BS08b] inherently leads to a sub-constant rate and therefore is not suitable for the current purpose.

We will not state the precise variation of the distance amplification here but we give a description in the next section after motivating the need for the variation.

The iteration ends after the distance amplification.

9.2 High level sketch of local testability

We need to show that the two operations of tensor product and distance amplification preserve the property of local testability. The main challenge will be to ensure this for the tensor product operation. The testability of the distance amplification operation can be shown by decomposing this operation into simpler block-wise operations, and showing that each of these operations preserves local testability.

The local testability of the tensor product. We need the tensor product to preserve the local testability, i.e., to increase the query complexity by a factor of at most $\text{poly}(1/\delta)$. However, this does not necessarily hold if C is an arbitrary code. In fact, there is a long line of research that attempts to understand the conditions under which tensor product preserves the local testability [BS06, Val05, CR05, DSW06, GM12, BV09, Vid15].

In order to resolve this issue, we use the following idea of [Mei09]. We say that a code C_0 has *property* $(\diamond)$ if there exists a code D such that C_0 is the tensor product D^2. It follows from [BS06, Vid15] that the tensor product operation roughly preserves the local testability of codes that have property $(\diamond)$. Specifically, if C_0 has property $(\diamond)$ and is locally testable with query complexity q and has relative distance δ, then C_0^2 is locally testable with query complexity $q \cdot \text{poly}(1/\delta)$.

Hence, in order to make our construction go through, we maintain the invariant that our code C has property $(\diamond)$ throughout the iterations. This requires us to show that a single iteration preserves the property $(\diamond)$ of the code. To this end, first observe that the tensor product operation clearly preserves the property $(\diamond)$. The more challenging part is to make sure that the distance amplification preserves the property $(\diamond)$. In order to do so, we define a new operation, which we called $(\diamond)$-distance amplification, which amplifies the distance while preserving the property $(\diamond)$ and the local testability.

The $(\diamond)$-distance amplification. We conclude by describing how the $(\diamond)$-distance amplification works: Suppose that we are given a code C_0 that has the property $(\diamond)$, and we wish to amplify its relative distance to δ. By definition, there exists some code D such that $C_0 = D^2$. We apply the Alon-Luby distance-amplification to D to obtain a new code D' with relative distance $\sqrt{\delta}$. We now define the code $C_0' = (D')^2$ to be the result of applying the $(\diamond)$-distance amplification to C_0. Observe that C_0' indeed has relative distance δ, and that it has the property $(\diamond)$, as required.

It remains to prove that the $(\diamond)$-distance amplification preserves the local testability, i.e., that this operation increases the query complexity by a factor of at most $\text{poly}(1/\delta)$. The testability of the distance amplification operation can be shown by decomposing this operation into simpler block-wise operations, and showing that each of these operations preserves local testability.

Acknowledgements

We would like to thank Eli Ben-Sasson, Alan Guo, Yohay Kaplan, Or Meir, Noga Ron-Zewi, Madhu Sudan, Michael Viderman, Avi Wigderson and Sergey Yekhanin for many enjoyable discussions and

collaborations on these topics over the years.

References

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

[ALM+98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and intractability of approximation problems. *Journal of ACM*, 45(3):501–555, 1998.

[Alo99] Noga Alon. Combinatorial nullstellensatz. *Combinatorics Probability and Computing*, 8(1):7–30, 1999.

[AS98] Sanjeev Arora and Shmuel Safra. Probabilistic checkable proofs: A new characterization of NP. *Journal of ACM*, 45(1):70–122, 1998.

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

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

[BLR93] Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. *J. Comput. Syst. Sci.*, 47(3):549–595, 1993.

[BS06] Eli Ben-Sasson and Madhu Sudan. Robust locally testable codes and products of codes. *Random Struct. Algorithms*, 28(4):387–402, 2006.

[BS08a] Eli Ben-Sasson and Madhu Sudan. Short pcps with polylog query complexity. *SIAM J. Comput.*, 38(2):551–607, 2008.

[BS08b] Eli Ben-Sasson and Madhu Sudan. Short PCPs with polylog query complexity. *SIAM J. Comput.*, 38(2):551–607, 2008.

[BSKK+13] Eli Ben-Sasson, Yohay Kaplan, Swastik Kopparty, Or Meir, and Henning Stichtenoth. Constant rate PCPs for circuit-SAT with sublinear query complexity. In *FOCS*, pages 320–329. IEEE Computer Society, 2013.

[BSS06] Eli Ben-Sasson and Madhu Sudan. Robust locally testable codes and products of codes. *Random Struct. Algorithms*, 28(4):387–402, 2006.

[BV09] Eli Ben-Sasson and Michael Viderman. Tensor products of weakly smooth codes are robust. *Theory of Computing*, 5(1):239–255, 2009.

[CKGS98] Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. Private information retrieval. *J. ACM*, 45(6):965–981, 1998.

[CR05] Don Coppersmith and Atri Rudra. On the robust testability of tensor products of codes. *Electronic Colloquium on Computational Complexity (ECCC)*, (104), 2005.

[DGY11] Zeev Dvir, Parikshit Gopalan, and Sergey Yekhanin. Matching vector codes. *SIAM J. Comput*, 40(4):1154–1178, 2011.

[Din07] Irit Dinur. The PCP theorem by gap amplification. *Journal of ACM*, 54(3):241–250, 2007.

[DSW06] Irit Dinur, Madhu Sudan, and Avi Wigderson. Robust local testability of tensor products of LDPC codes. In *proceedings of the 9th International Workshop on Randomization and Computation (RANDOM)*, pages 304–315. Springer, 2006.

[Efr12] Klim Efremenko. 3-query locally decodable codes of subexponential length. *SIAM J. Comput.*, 41(6):1694–1703, 2012.

[FS95] Katalin Friedl and Madhu Sudan. Some improvements to total degree tests. In *proceedings of the 3rd Israel Symposium on the Theory of Computing and Systems (ISTCS)*, pages 190–198. IEEE Computer Society, 1995.

[GK14] Alan Guo and Swastik Kopparty. List-decoding algorithms for lifted codes. *CoRR*, abs/1412.0305, 2014.

[GKS13] Alan Guo, Swastik Kopparty, and Madhu Sudan. New affine-invariant codes from lifting. In *proceedings of the Innovations in Theoretical Computer Science Conference (ITCS)*, pages 529–540. ACM Press, 2013.

[GM12] Oded Goldreich and Or Meir. The tensor product of two good codes is not necessarily locally testable. *Inf. Proces. Lett.*, 112(8-9):351–355, 2012.

[Gol11] Oded Goldreich. Bravely, moderately: A common theme in four recent works. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, pages 373–389. Springer, 2011.

[Guo13] Alan Guo. High rate locally correctable codes via lifting. *Electronic Colloquium on Computational Complexity (ECCC)*, 20:53, 2013.

[HOW15] Brett Hemenway, Rafail Ostrovsky, and Mary Wootters. Local correctability of expander codes. *Inf. Comput*, 243:178–190, 2015.

[KMRS16] Swastik Kopparty, Or Meir, Noga Ron-Zewi, and Shubhangi Saraf. High-rate locally-correctable and locally-testable codes with sub-polynomial query complexity. In Daniel Wichs and Yishay Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 202–215. ACM, 2016.

[Kop14] Swastik Kopparty. Some remarks on multiplicity codes. In *Proceedings of the AMS Special Session on Discrete Geometry and Algebraic Combinatorics*, Contemporary Mathematics, 2014.

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

[KS08] Tali Kaufman and Madhu Sudan. Algebraic property testing: the role of invariance. In Cynthia Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 403–412. ACM, 2008.

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

[LFKN92] Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *J. ACM*, 39(4):859–868, 1992.

[Mei09] Or Meir. Combinatorial construction of locally testable codes. *SIAM J. Comput.*, 39(2):491–544, 2009.

[Rei08] Omer Reingold. Undirected connectivity in log-space. *J. ACM*, 55(4), 2008.

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

[RS97] Ran Raz and Shmuel Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *STOC*, pages 475–484, 1997.

[RVW02] Omer Reingold, Salil Vadhan, and Avi Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders. *Annals of Mathematics*, 155(1):157–187, 2002.

[Sha90] Adi Shamir. IP=PSPACE. In *FOCS*, pages 11–15, 1990.

[Sti93] Henning Stichtenoth. *Algebraic function fields and codes*. Universitext. Springer, 1993.

[STV01] Madhu Sudan, Luca Trevisan, and Salil P. Vadhan. Pseudorandom generators without the XOR lemma. *J. Comput. Syst. Sci.*, 62(2):236–266, 2001.

[Val05] Paul Valiant. The tensor product of two codes is not necessarily robustly testable. In *Proceedings of the 9th International Workshop on Randomization and Computation (RANDOM)*, pages 472–481. Springer, 2005.

[Vid15] Michael Viderman. A combination of testability and decodability by tensor products. *Random Struct. Algorithms*, 46(3):572–598, 2015.

[Yek08] Sergey Yekhanin. Towards 3-query locally decodable codes of subexponential length. *J. ACM*, 55(1), 2008.