Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 32 (2018)

Explicit Binary Tree Codes with Polylogarithmic Size Alphabet

Gil Cohen* Bernhard Haeupler† Leonard J. Schulman‡

Abstract

This paper makes progress on the problem of explicitly constructing a binary tree code with constant distance and constant alphabet size.

For every constant $\delta < 1$ we give an explicit binary tree code with distance δ and alphabet size $\mathrm{poly}(\log n)$, where n is the depth of the tree. This is the first improvement over a two-decade-old construction that has an exponentially larger alphabet of size $\mathrm{poly}(n)$.

As part of the analysis, we prove a bound on the number of positive integer roots a real polynomial can have in terms of its *sparsity* with respect to the Newton basis—a result of independent interest.

An extended abstract of this work appeared in The 50th ACM Symposium on Theory of Computing (STOC 2018).

*Department of Computer Science, Princeton University, Princeton, USA. Part of this work was done while the author was a CMI postdoctoral fellow at The California Institute of Technology, Pasadena, USA. Email: gilc@princeton.edu.

†Department of Computer Science, Carnegie Mellon University, Pittsburgh, USA. Supported in part by NSF grants CCF-1527110, CCF-1618280 and NSF CAREER award CCF-1750808. Email: haeupler@cs.cmu.edu.

‡E&AS Division, California Institute of Technology, Pasadena, USA. Part of this work was done while in residence at the Israel Institute for Advanced Studies; financial support is due to United States NSF grant 1618795 and to a EURIAS Senior Fellowship co-funded by the Marie Skłodowska-Curie Actions under the 7th Framework Programme. Email: schulman@caltech.edu.

ISSN 1433-8092

1 Introduction

This paper makes progress on the problem of explicitly constructing a binary tree code with constant distance and constant alphabet size.

Tree codes are a powerful but so-far elusive combinatorial structure, defined and proven to exist in [50, 52] in order to serve as a key ingredient for achieving a constant rate interactive coding scheme. Tree codes are the central object for encoding information in the interactive coding theory which developed from the initial papers. They remain a crucial building block in almost all interactive coding schemes [46, 11, 10, 22, 5, 7, 4, 6, 25, 26, 38, 1, 23, 9, 33, 53]. The absence of an explicit construction that is also efficiently decodable is the only reason why most of these schemes are computationally inefficient, requiring exponential-time computations. Other works have invested significant effort in avoiding the use of (large) tree codes, often at a considerable loss in the fraction or generality of errors that can be tolerated [23, 5, 7, 8, 36, 28, 32, 6, 25]. We refer to the excellent survey by Gelles [21] for an in-depth account of the role tree codes hold in the area of interactive coding theory.

In addition, tree codes have important uses as streaming codes for both Hamming errors [18] and synchronization errors [29, 31, 10, 30]. In control theory, although mostly unknown to the computer science community, tree codes are closely connected to anytime reliable codes that are necessary to controlling and stabilizing systems over unreliable channels [47, 48, 49, 57, 58, 27, 35]; there, too, the absence of explicit constructions of tree codes was the motivation for an elaborate work-around for certain control applications [42]. Tree codes have also found surprising application in metric embeddings [37] and complexity theory [15, 14].

Let us define tree codes and explain why one should think of them as an online version of a regular error correcting block code. A tree code consists of a complete rooted binary tree (either infinite or of finite depth n) in which each edge is labeled by a symbol from an alphabet Σ. There is a natural one-to-one mapping assigning each binary string s to a path starting at the root, where s simply indicates which child is taken in each of the steps. For a tree code, such a path naturally maps to a string over the alphabet Σ, which is formed by concatenating the symbols along the path. This way a tree code T encodes any binary string s into an equally long string $T(s)$ over Σ. This encoding has the online property because the encoding of any prefix does not depend on later symbols: any two distinct strings that agree in their first k symbols also have encodings that agree in their first k symbols. A tree code is said to achieve distance $\delta \geq 0$ if the encodings of any two strings differ in at least a δ-fraction of the positions after the strings first disagree. The rate of a tree code is $\frac{1}{\log_2 |\Sigma|}$. (More generally, a tree code may be a d-ary tree, with s in the above definition ranging over an alphabet of cardinality d; in this case for finite d the rate is $\frac{1}{\log_d |\Sigma|}$; we will use such larger-degree trees as a stepping stone but ultimately will construct a binary tree code.) A tree code is said to be asymptotically good if it achieves both constant distance $\delta > 0$ and a constant rate, namely, the alphabet size $|\Sigma|$ is $O(1)$ in n.

Three different proofs were provided in [50, 52], showing that for any $\delta < 1$ there exists a binary tree code with a constant-size alphabet achieving distance δ. All of these proofs, as well as a later quantitative improvement by Peczarski [43], rely on the probabilistic method. Interestingly however, in contrast to conventional error correcting block codes, a constant-size-alphabet random tree code is not asymptotically good and has a distance of zero, with high probability.

The problem of giving an explicit construction of asymptotically good tree codes has drawn substantial attention, but has endured as a difficult challenge.[1] Technically, for a depth n tree code to be explicit, we require that there exist a deterministic algorithm running in time $\mathrm{poly}(n)$ which on input $s \in \{0,1\}^{n'}$ ($n' \le n$), outputs the label of the last edge on the path s. We remark that one of the existence proofs is based on the Lovász local lemma [17] but the algorithmic LLL [41, 12] does not yield explicit tree codes as it must construct the entire $\exp(n)$-size tree. In fact, for explicit constructions not much has been known beyond a construction of Evans, Klugerman and Schulman [51] (dating to 1994) that provides a tree code with alphabet size $\mathrm{poly}(n)$. Pudlák [45] studies sufficient and necessary structural results for linear (MDS) tree codes and provides a construction with large arity. Moore and Schulman gave a candidate construction [40], but its distance property relies on an open conjecture about certain exponential sums. A tree code construction which reduces the brute-force time of encoding and decoding from exponential, i.e., $\exp(n)$, to sub-exponential, i.e., $\exp(n^{\varepsilon})$, at the cost of an alphabet size $\exp(1/\varepsilon)$, was given by Braverman [8].

1.1 Our Results

In this work we obtain the first proven improvement over the two-decade-old construction of [51] by giving an explicit binary tree code with constant distance and an exponentially smaller, i.e., polylogarithmic, alphabet size.

Theorem 1.1. *For every constant $\delta < 1$ and integer $n \ge 1$ there exists an explicit binary tree code $\mathsf{TC}\colon \{0,1\}^n \to \Sigma^n$ with distance δ and $|\Sigma| = (\log n)^{O(1)}$.*

Put differently, Theorem 1.1 gives a binary tree code with rate $\Omega(1/\log\log n)$ and distance δ. We point out that our techniques readily yield a depth n tree code that can be constructed in time $\exp(n^{\varepsilon})$ with alphabet size $\mathrm{poly}(1/\varepsilon)$. This improves on the alphabet size $\exp(1/\varepsilon)$ that was obtained by Braverman [8] under the same running-time restriction.

We prove Theorem 1.1 in two steps. First, we construct a tree code over the integers as given in Theorem 1.2 below. This tree code has the advantage of being of infinite depth. We then reduce the input alphabet to Boolean.

[1] Wigderson in his new book "Mathematics and Computation" calls it *"the most elegant open problem of [the] theory [of interactive coding]"* [61, page 202, Open Problem 15.34].

Theorem 1.2. *For every constant $\delta < 1$ there exists an explicit tree code*[2] $\mathsf{TC}_{\mathbb{Z}} \colon \mathbb{Z}^{\mathbb{N}} \to \mathbb{Z}^{\mathbb{N}}$ *with distance δ. Further, for every $z = (z_t)_{t\in\mathbb{N}} \in \mathbb{Z}^{\mathbb{N}}$ and $t \in \mathbb{N}$,*

$$|\mathsf{TC}_{\mathbb{Z}}(z)_t| \le 2^{O(t^2)} \cdot \left(\max\left(|z_0|, \ldots, |z_t|\right)\right)^{O(1)}.$$

Our construction is at its cleanest when $\delta = 1/2$. In this case, the dependence on t is also better. Throughout this section we focus on this tree code whose parameters are given in the following theorem.

Theorem 1.3. *There exists an explicit tree code $\mathsf{TC}_{\mathbb{Z}} \colon \mathbb{Z}^{\mathbb{N}} \to \mathbb{Z}^{\mathbb{N}}$ with distance $1/2$. Further, for every $z = (z_t)_{t\in\mathbb{N}} \in \mathbb{Z}^{\mathbb{N}}$ and $t \in \mathbb{N}$, $|\mathsf{TC}_{\mathbb{Z}}(z)_t| \le 2^t \cdot \max\left((z_0 \log z_0)^2, \ldots, (z_t \log z_t)^2\right)$.* [3]

We wish to give some remarks regarding the bound on $|\mathsf{TC}_{\mathbb{Z}}(z)_t|$ that is guaranteed by Theorem 1.3. Assume that $|z_t| \le m$ for all t. Theorem 1.3 gives a bound of $2^t(m \log m)^2 \le 2^t m^3$ on the t'th output symbol. This should be compared with the trivial bound of m^t and with the bound $m^{O(\log t)}$ that is obtained by adapting the technique of [51] to tree codes over the integers. Although our bound has an exponential dependence on t, the two parameters m and t are decoupled and so one can take t super-constant while keeping the bound polynomial in m. In the second step of our construction, we show that this property suffices to obtain the improved binary tree code claimed by Theorem 1.1 with distance $1/2$. The same argument shows how Theorem 1.2 implies Theorem 1.1 for any constant distance $\delta < 1$.

The proof of Theorem 1.3 is obtained by adapting the Reed Solomon polynomial interpolation framework to the online setting. We give an overview of the proof in Section 2 and the formal proof is the content of Section 5. To analyze the distance, we prove a bound on the number of distinct integral roots a real polynomial can have in terms of its *sparsity* in a certain basis–a result of independent interest on which we elaborate on in Section 1.1.1 below. The alphabet reduction technique we use to deduce Theorem 1.1 is covered in Section 6. Finally, in Section 7, we prove Theorem 1.2.

1.1.1 A Bound on the Number of Integral Roots via Sparsity

The fundamental theorem of algebra asserts that a degree $d > 0$ polynomial with complex coefficients has exactly d complex roots when counted with multiplicities. More generally, over any field $\mathbb{F}$, a degree $d > 0$ polynomial $f \in \mathbb{F}[x]$ has at most d roots in $\mathbb{F}$ (and exactly d roots in the algebraic closure of $\mathbb{F}$).

The sparsity of a polynomial, however, cannot be used to bound the number of its distinct roots. There are natural examples of sparse polynomials with many roots even in the base

[2] Throughout the paper, it will be convenient to use the notation $\mathbb{N} = \{0, 1, 2, \ldots\}$.

[3] If one has in advance a bound on the $|z_t|$'s in the form of a function $b \colon \mathbb{N} \to \mathbb{N}$ with $|z_t| \le b(t)$ then the log-factors in the statement of the theorem can be avoided. This is the case in our application of the theorem for the proof of Theorem 1.1. Nevertheless, we state the result in this more general form.

field, e.g., $x^p - x$ in $\mathbb{F}_p[x]$. Nevertheless, for the analysis of our tree code construction, we provide a meaningful bound on the number of positive *integer* roots (that is, roots in $\mathbb{N}$) a real polynomial can have in terms of its sparsity.

Unlike the notion of degree, sparsity is, of course, basis dependent. The basis for which our bound holds is not the standard basis $\{1, x, x^2, \ldots\}$ but rather the Newton basis which consists of polynomials of the form $\binom{x}{k} \in \mathbb{R}[x]$ for $k \in \mathbb{N}$, where

$$\binom{x}{k} = \frac{x(x-1)\cdots(x-(k-1))}{k!}.$$

It is easy to verify that for every $d \in \mathbb{N}$, the set $\{\binom{x}{k} \mid k = 0, 1, \ldots, d\}$ forms a basis for the space of univariate real polynomials of degree at most d.

Of course, with respect to this basis, the sparsity cannot be taken as a bound on the number of distinct roots a polynomial can have. Indeed, for any $d \in \mathbb{N}$, consider the degree d polynomial $\binom{x}{d}$ which has sparsity $s = 1$ in the Newton basis. Evidently, $\binom{x}{d}$ has d distinct roots at $x = 0, 1, \ldots, d-1$. Thus, one cannot hope to prove a general bound on the number of roots in terms of sparsity even when restricting to integral roots and not accounting for multiplicities.

Consider a polynomial with sparsity $s = 2$ in the Newton basis. Such a polynomial has the form

$$f(x) = \gamma\binom{x}{c} + \delta\binom{x}{d},$$

where $0 \le c < d$ are integers and γ, δ are nonzero real numbers. Clearly, $0, 1, \ldots, c-1$ are all roots of f, and c can be taken much larger than 2 – the sparsity of f. More generally, if f is a polynomial with sparsity s and $c = c(f)$ is the least integer such that $\binom{x}{c}$ appears in the expansion of f in the Newton basis then f will surely have $0, 1, \ldots, c-1$ as its roots. Again, it may be the case that $c \gg s$.

We prove that but for these c "trivial" integral roots, f has at most $s - 1$ roots in $\mathbb{N}$. This holds regardless of the degree of f. More precisely, we prove the following lemma which can be interpreted as an *additive uncertainty principle* for the Newton basis.

Lemma 1.4. *Let $f \in \mathbb{R}[x]$ be a nonzero polynomial of sparsity $s \ge 1$ in the Newton basis. Let $c \ge 0$ be the least integer such that $f(c) \neq 0$. Then, f has at most $s-1$ distinct roots in $[c, \infty) \cap \mathbb{Z}$.*

Observe that the restriction to integral roots is necessary, that is, one cannot strengthen the result by arguing about non-integral roots in $[c, \infty)$. To see this, take any integer $d > 1$ and consider the polynomial with integral coefficients $f_C(x) = (-1)^{d-1}\binom{x}{1} + C\binom{x}{d}$, where $C \in \mathbb{N}$ is chosen sufficiently large. The polynomial f_C has sparsity 2, degree d and, with the notation above, $c(f_C) = 1$. However, for C large enough, $f_C \approx C\binom{x}{d}$ (away from 0) and has $d - 1$ distinct roots in $[1, \infty)$. This is as compared with our bound of at most one root in $\{1, 2, \ldots\}$.

An equivalent statement of the lemma considers $f \in \mathbb{R}[[x]]$ as a formal power series in the Newton basis: $f(x) = \sum_{k \geq 0} f_k \binom{x}{k}$. This is well defined at nonnegative integers (regardless of whether there is an open set in which the series converges). The lemma states that if the nonzero coefficients f_k are $f_{\kappa_1}, f_{\kappa_2}, \ldots$ for some (finite or infinite) series $\kappa_1 < \kappa_2 < \ldots$, then for all s, f has at most $s-1$ distinct roots in $[\kappa_1, \kappa_s] \cap \mathbb{Z}$.

We prove Lemma 1.4 in Section 4. The proof makes use of the beautiful Gessel-Viennot Lemma (see Lemma 3.9). Given the usefulness of the degree bound on the number of roots, we believe that Lemma 1.4 should find further applications. The only existing result that it resembles, to our knowledge, is Chebotarëv's oft-rediscovered theorem (answering a question of Ostrovskiĭ) that every minor of the Fourier transform over $\mathbb{Z}/n$, n prime, is nonsingular [55]. We comment incidentally on the relationship of the latter property with maximum-distance-separability (MDS). A $k \times n$ matrix is the parity-check matrix of an MDS code if every $k \times k$ minor is nonsingular. In this sense, any collection of rows of the Fourier matrix qualifies as the parity check matrix of an MDS code, but this is not what is normally useful in coding theory since the underlying field is large (at least the n'th cyclotomic field).

2 Overview of the Construction

The polynomial interpolation framework is at the heart of several important constructions of error correcting block codes such as the Reed Solomon code. Our construction is based on identifying a suitable adjustment of the polynomial interpolation framework to the online setting. To motivate our construction, we start by highlighting the difficulties in pursuing such an approach. To this end, we recall the definition of the Reed Solomon code. Let n be an integer. Assume, for simplicity, that n is prime, and let $\mathbb{F}$ be the field of n elements. For an integer $k \leq n$, the Reed Solomon code $\mathsf{RS} \colon \mathbb{F}^k \to \mathbb{F}^n$ is defined as follows. Define the polynomial $f_m(x) = \sum_{i=0}^{k-1} m_i x^i \in \mathbb{F}[x]$. The encoding of m is defined by $\mathsf{RS}(m) = (f_m(0), f_m(1), \ldots, f_m(n-1))$.

As f_m is linear in m, the analysis of the distance of RS proceeds by proving an upper bound on the number of zero entries of a codeword that corresponds to a nonzero message. By construction, these entries correspond to the number of distinct roots of f_m in $\mathbb{F}$. Here is where the degree bound on the number of roots of f_m is invoked.

An obvious difficulty in adapting the above idea to the construction of tree codes arises from the latter's online nature. As we do not have the entire message available to us up until the very end, there is no clear sense as to which polynomial we should work with. However, the challenge is more significant. Even given the entire message, one still needs to gain nonzero output symbols starting from the index of the first nonzero entry of the message. That is, not only that one has to work with partial information, a tree code must also gain distance as soon as a disagreement, or "split," occurs and to keep the distance above a certain threshold from that point on. Restricting to the Reed Solomon construction, this means that even given the message m, the polynomial f_m defined above should somehow be evaluated on a carefully chosen sequence of points in the field. We do not know how to implement such an approach or even if it is possible in principle. Anyhow, one does not have the message in its entirety.

Having these difficulties in mind motivates our construction which we present next. Although our construction is based on polynomial interpolation, and so it is inherently algebraic, it can be motivated both using a combinatorial reasoning and from an algebraic perspective. We start by presenting the combinatorial point of view in Section 2.1. We then discuss the algebraic perspective in Section 2.2. The formal proof, given in Section 5, is presented and analyzed only via the algebraic perspective, nevertheless, we believe that the combinatorial point of view gives a natural motivation for our construction.

2.1 The Combinatorial Perspective

In this section we motivate and describe the tree code construction $\mathsf{TC}_{\mathbb{Z}} \colon \mathbb{Z}^{\mathbb{N}} \to \mathbb{Z}^{\mathbb{N}}$ from Theorem 1.3. Let $z = (z_t)_{t \in \mathbb{N}} \in \mathbb{Z}^{\mathbb{N}}$ be a message that we want to encode. For $t \in \mathbb{N}$, define $f_t \in \mathbb{R}[x]$ to be the polynomial of least degree such that $f_t(i) = z_i$ for all $i \in \{0, 1, \dots, t\}$. Note that f_t is fully determined by $z_0, \dots, z_t$. Further, observe that f_t is linear in z. Therefore, so long as we define $\mathsf{TC}_{\mathbb{Z}}(z)_t$ as a linear combination of the evaluation of the polynomials $f_0, \dots, f_t$ on fixed points, it will follow that $\mathsf{TC}_{\mathbb{Z}}(z)_t = \mathsf{TC}_{\mathbb{Z}}(z')_t$ if and only if $\mathsf{TC}_{\mathbb{Z}}(z - z')_t = \mathsf{TC}_{\mathbb{Z}}(\bar{0})_t = 0$; i.e., for purposes of distance analysis, it suffices to compare every nonzero message z against the all-zeros message.

To recap, while the Reed Solomon code interprets the message as a polynomial, in our construction (which has not yet been presented) every prefix of the message is interpreted as a polynomial, and so z, chosen by the "adversary," induces an infinite sequence of polynomials $f_0, f_1, f_2, \dots$.

Consider a scenario in which the adversary makes it so that $f_t = f_{t+1} = \cdots = f_{t+\ell}$. Intuitively, such a scenario is favorable for us. Indeed, one can imagine how useful it would be if the adversary is committed to a single polynomial f for a long interval of time while (just as in Reed Solomon) outputting evaluations of f in the interval. In some sense, it is as if the tree code is only required to work against an off-line input on that interval; the fundamental theorem of algebra can come into play and prevent having many 0s in the output during this interval. Thus, a natural idea is to penalize the adversary when switching to a new polynomial from time $t-1$ to time t. This can be done by outputting, at time t, the value

$$\delta_t = f_t(t) - f_{t-1}(t).$$

In control theory, one would call δ_t the "innovation." A complexity-theoretic point of view would interpret δ_t as a consistency checking procedure. Indeed, unless the adversary sticks to his polynomial f_{t-1} at time t, he pays in distance as then $\delta_t \neq 0$. This consistency checking symbol at time t is concatenated with $f_t(t)$, i.e., z_t. To summarize, we define $\mathsf{TC}_{\mathsf{comb}}\colon \mathbb{Z}^{\mathbb{N}} \to (\mathbb{Z}^2)^{\mathbb{N}}$ by [4]

$$\mathsf{TC}_{\mathsf{comb}}(z)_t = (z_t, \delta_t).$$

It is not immediately clear from this representation why δ_t should be an integer but as it turns out, that is the case and $|\delta_t|$ can be bounded as we discuss at the end of Section 2.2.

How large is the distance of $\mathsf{TC}_{\mathsf{comb}}$? It seems that an adversary that sticks to a polynomial for not-too-long intervals may pay very little as we do not gather sufficient amount of information during a short interval. On the other hand, it is intuitive that something is gained by this approach. How would an adversary work against $\mathsf{TC}_{\mathsf{comb}}$?

A potential attack. One attack might work as follows. Fix some $d \geq 1$. The adversary will choose $z_0, \ldots, z_d$ such that the degree d polynomial f_d has roots at $d+1, \ldots, 2d$. Now by choosing just one new nonzero value z_{2d+1}, the adversary obtains a new polynomial f_{2d+1} of degree $2d+1$ which shares the already-recorded d roots but also has $d+1$ new roots, potentially at $2d+2, \ldots, 3d+2$; the adversary then uses $z_{2d+2} = \ldots z_{3d+2} = 0$ and in this case we have $\delta_{2d+2} = \ldots = \delta_{3d+2} = 0$. The adversary can again use a nonzero z_{3d+3}, obtaining a new polynomial f_{3d+3} with $d+2$ new roots, potentially at $3d+4, \ldots, 4d+5$, and the adversary then makes the choice $z_{3d+4} = \ldots = z_{4d+5} = 0$. If this process can be repeated in this manner (i.e., if the described polynomials exist), the result will be a branch of the tree code which at depth ℓ has weight only $O(\sqrt{\ell})$.

Even if there are no polynomials that perfectly interpolate as required by the above attack, it is not obvious that one can rule out a quantitatively-relaxed version of such an attack. What one can see however, is that the adversary cannot beat $\Omega(\sqrt{\ell})$: after the s'th nonzero value, say it is z_t, there cannot be a run of zeros $0 = f_t(t+1) = f_t(t+2) = \ldots = f_t(t+s)$ as this would contradict Lemma 3.9, which establishes that the relevant minor in the linear transformation is nonsingular.

Though certainly a nontrivial bound, an $\Omega(1/\sqrt{\ell})$ distance is far from the constant distance that we are shooting for. Interestingly, $\mathsf{TC}_{\mathsf{comb}}$ has, in fact, distance $1/2$! In particular, the above attack is far from feasible. To prove that, we consider an algebraic point of view on the construction of $\mathsf{TC}_{\mathsf{comb}}$. Taking the algebraic perspective, we can prove that the adversary has a budget of roots that is bounded by the sparsity (with respect to a certain basis) of the polynomials rather than by their degree.

2.2 The Algebraic Perspective

So far, when working with the polynomials $(f_t)_t$, we did not pay attention to the basis in which the polynomials are represented. Generally, the polynomial interpolation framework

[4] Note that the output symbols are pairs of integers rather than integers as stated in Theorem 1.3. This, of course, is a non-issue and is only meant for a cleaner presentation.

works over any basis as the degree, which is typically used in the analysis, is basis invariant. However, for our purpose, the standard basis $\{1, x, x^2, \ldots\}$ has the following drawback. Let $y_0, \ldots, y_n \in \mathbb{R}$. Let $f(x) = \sum_{i=0}^{n} a_i x^i$ be the least degree polynomial that interpolates on the points $(0, y_0), (1, y_1), \ldots, (n, y_n)$. Then generally, given a new point $(n+1, y_{n+1})$, the least degree polynomial, $g(x) = \sum_{i=0}^{n+1} b_i x^i$, that interpolates on $(0, y_0), \ldots, (n+1, y_{n+1})$ will have a completely different sequence of coefficients (i.e., $a_i \neq b_i$).

By contrast, using the Newton basis, the coefficients that were already "recorded" stay intact given the new point $(n+1, y_{n+1})$. More precisely, if $f(x) = \sum_{i=0}^{n} \gamma_i \binom{x}{i}$ then $g(x) = f(x) + \gamma_{n+1}\binom{x}{n+1}$ for some $\gamma_{n+1} \in \mathbb{R}$. Classically, this fact makes the Newton basis attractive for numerical stability and was used for obtaining structural results for polynomials [20, 54, 13]. For constructing tree codes, this property is attractive as it means that for every t, the coefficient γ_t is determined by $y_0, y_1, \ldots, y_t$.

The above discussion suggests a second construction of tree codes over $\mathbb{Z}$. Let γ_t be the coefficient of $\binom{x}{t}$ in the expansion of f_t, as defined in Section 2.1. We define the tree code $\mathsf{TC}_{\mathsf{alg}} \colon \mathbb{Z}^{\mathbb{N}} \to (\mathbb{Z}^2)^{\mathbb{N}}$ by

$$\mathsf{TC}_{\mathsf{alg}}(z) = (z_t, \gamma_t),$$

where we postpone in this informal discussion the technical (and simple) aspect involved in encoding the two integers to a single binary string in a decodable way.

Interestingly, one can show that $\gamma_t = \delta_t$ for every t and so $\mathsf{TC}_{\mathsf{comb}}$ and $\mathsf{TC}_{\mathsf{alg}}$ are one and the same! The algebraic point of view on the tree code will allow us to prove a bound of $1/2$ on the distance as we now explain.

Let c be the least integer such that $z_c \neq 0$. Let $\ell \geq 1$ and set $t = c + \ell - 1$. Observe that the number of non-zeros in the sequence $\gamma_0, \gamma_1, \ldots, \gamma_t$ is precisely the sparsity of f_t in the Newton basis. This, together with the fact that for every $i \leq t$, $z_i = f_t(i)$, implies that to "break" the construction $\mathsf{TC}_{\mathsf{alg}}$, the adversary must come up with a sparse polynomial f_t that has many roots in $I = \{c, c+1, \ldots, t\}$. Indeed, if f_t is not sparse, then many of the γ-entries of $(\mathsf{TC}_{\mathsf{alg}}(z)_i)_{i \in I}$ will be nonzero. On the other hand, if f_t has only few roots in I then many of the z-entries are nonzero.

To give the quantitative bound we invoke Lemma 1.4 which implies that if the sparsity of f_t is s then there can be at most $s-1$ zeros among the z-entries of $\{\mathsf{TC}_{\mathsf{alg}}(z)_i\}_{i \in I}$. So the combined number of nonzero integers among the 2ℓ integers in $(\mathsf{TC}_{\mathsf{alg}}(z)_i)_{i \in I}$ is at least $\ell + 1$. Thus, at least half of the pairs are nonzero pairs, establishing a distance of $1/2$.

Another issue that can be handled via the algebraic perspective is related to the integrality of the output symbols. The symbol z_t is clearly an integer. However, it is not a priori clear that $\gamma_t = \delta_t$ is an integer. The Newton basis has another useful property we use—if $z_0, \ldots, z_t$ are all integers, so are the coefficients $\gamma_0, \ldots, \gamma_t$. Note that this property does not hold for the standard basis. Moreover, there is a closed formula for γ_t as a function of $z_0, \ldots, z_t$ (see Lemma 3.8) which allows us to prove the desired bound on $|\gamma_t|$.

2.3 Tree Codes for any Distance $\delta < 1$

It is fairly straightforward to adapt the ideas described above to obtain any distance $\delta < 1$. A natural strategy is to use more evaluation points. This idea, however, should be executed with some care. In this section we sketch how to obtain distance $\delta = 2/3$. The idea can be easily generalized to yield any distance $\delta < 1$ (Section 7).

Let us suggestively denote the input message by $z = (z_0, z_2, z_4, \ldots) \in \mathbb{Z}^{2\mathbb{N}}$. As before, we define a sequence of real polynomials $f_0, f_1, f_2, \ldots$. However, to obtain the improved distance, we also *define* a sequence of integers $z_1, z_3, z_5, \ldots$ inductively on $t \in \mathbb{N}$, as follows. For even t we define f_t, as before, to be the least degree real polynomial such that $\forall i \in \{0, 1, \ldots, t\}$, $f_t(i) = z_i$. We then define $f_{t+1} = f_t$ and compute $z_{t+1} = f_{t+1}(t+1)$.

For $t \in \mathbb{N}$, let γ_t be the coefficient of $\binom{x}{t}$ in the expansion of f_t. We define $\mathsf{TC}^{2/3}_{\mathsf{alg}} : \mathbb{Z}^{2\mathbb{N}} \to (\mathbb{Z}^3)^{\mathbb{N}}$ by

$$\mathsf{TC}^{2/3}_{\mathsf{alg}}(z_0, z_2, z_4, \ldots)_t = (\gamma_{2t}, z_{2t}, z_{2t+1}).$$

One can show that $\mathsf{TC}^{2/3}_{\mathsf{alg}}$ is a linear online function. To argue about the distance, let c be the least integer such that $z_{2c} \neq 0$. Let $\ell \geq 1$. Set $t = c + \ell - 1$ and denote $I = \{c, c+1, \ldots, t\}$. Observe that $\gamma_i = 0$ for every odd i, and so the number of non-zeros in the sequence $\gamma_0, \gamma_2, \ldots, \gamma_{2t}$ is the sparsity of f_{2t} in the Newton basis. By Lemma 1.4, among the evaluation points $\{2c, 2c+1, \ldots, 2t, 2t+1\}$, at most $s - 1$ are roots of f_{2t}. Thus, the number of nonzero triplets among $(\mathsf{TC}^{2/3}_{\mathsf{alg}}(z)_i)_{i \in I}$ is at least

$$\frac{s + 2\ell - (s-1)}{3} \geq \frac{2}{3}\ell,$$

proving that the distance is $2/3$.

One concern that must be addressed is the bound on the γ symbols. Unlike the $1/2$ distance construction, now γ_t depends on the *computed* value z_{t-1} which, in turn, depends on γ_{t-2}. Thus, potentially, the γ symbols can grow much faster and, indeed, the bound we give for distance $2/3$ is weaker than the corresponding bound for distance $1/2$. Nevertheless, as it turns out, for deducing Theorem 1.1, the weaker bound suffices. Further, the bound does not degrade substantially when considering any constant distance $2/3 < \delta < 1$.

3 Preliminaries

Let $n \geq 1$ be an integer and Σ some (finite or infinite) set. For a string $x = (x_1, \ldots, x_n) \in \Sigma^n$ and integers $1 \leq a \leq b \leq n$, we let $x_{[a,b]}$ denote the substring $(x_a, \ldots, x_b)$. If $\sigma \in \Sigma$ then σ^n denotes the string $(\sigma, \ldots, \sigma) \in \Sigma^n$. Given $x, y \in \Sigma^n$, we write $\mathrm{dist}(x, y)$ for their Hamming distance.

For an integer $n \geq 1$ write $[n]$ for $\{1, 2, \ldots, n\}$. We use the conventions that the natural numbers are $\mathbb{N} = \{0, 1, 2, \ldots\}$, and that $\binom{a}{b} = 0$ for integers $0 \leq a < b$.

3.1 Error Correcting Block Codes

Definition 3.1. *A function* $\mathsf{ECC}\colon \Sigma_{\mathsf{in}}^k \to \Sigma_{\mathsf{out}}^n$ *is an* error correcting block code *with distance* δ *if for every distinct* $x, y \in \Sigma_{\mathsf{in}}^k$, $\mathrm{dist}(\mathsf{ECC}(x), \mathsf{ECC}(y)) \geq \delta n$. *The rate of* ECC *is given by* $(k \log_2 |\Sigma_{\mathsf{in}}|)/(n \log_2 |\Sigma_{\mathsf{out}}|)$.

For the proof of Theorem 1.1, it is convenient to consider error correcting block codes whose output length is shorter than their input length and with output alphabet consisting of binary strings of a certain length. We make use of the following construction of error correcting block codes. The construction and its proof are given in Appendix A.

Lemma 3.2. *For every constant* $0 < \delta < 1$ *and constant integer* $t \geq 1$ *there exists an integer* $c = c(t, \delta)$ *such that for every large enough integer* n *there exists an explicit error correcting block code* $\mathsf{ECC}\colon \{0,1\}^n \to (\{0,1\}^c)^{n/t}$ *with distance* δ.

3.2 Tree Codes

Tree codes, as their name suggest, are trees with certain distance properties. However, in this paper, we use an equivalent definition of tree codes that more directly specifies their online characteristic compared to the one given in the original papers [50, 52] and, in particular, does not involve trees. This will be more convenient for presenting our construction.

Definition 3.3. *A function* $f\colon \Sigma_{\mathsf{in}}^n \to \Sigma_{\mathsf{out}}^n$ *is said to be* online *if for every* $i \in [n]$ *and* $x \in \Sigma_{\mathsf{in}}^n$, $f(x)_i$ *is determined by* $x_1, \ldots, x_i$.

Definition 3.4. *For a pair of distinct* $x, y \in \Sigma^n$, *we define* $\mathsf{split}(x, y)$ *as the least integer* $s \in [n]$ *such that* $x_s \neq y_s$.

Definition 3.5 ([50, 52])**.** *An online function* $\mathsf{TC}\colon \Sigma_{\mathsf{in}}^n \to \Sigma_{\mathsf{out}}^n$ *is a* tree code with distance δ *if for every distinct* $x, y \in \Sigma_{\mathsf{in}}^n$, *with* $s = \mathsf{split}(x, y)$, *and every* $\ell \in \{0, 1, \ldots, n-s\}$,

$$\mathrm{dist}\left(\mathsf{TC}(x)_{[s,s+\ell]}, \mathsf{TC}(y)_{[s,s+\ell]}\right) \geq \delta(\ell+1).$$

We refer to n *as the* depth *of* TC. *We refer to* $\Sigma_{\mathsf{in}}, \Sigma_{\mathsf{out}}$ *as the input alphabet and output alphabet, respectively.*

We remark that the terms depth and split are coming from the original point of view of tree codes as trees with certain distance properties. The depth is simply the depth of the tree and the split is the level at which the pair of paths diverge. We borrow this terminology even though we do not explicitly view tree codes as trees in this work. We are interested in some further properties of tree codes.

Definition 3.6. *Let* $\mathsf{TC}\colon \Sigma_{\mathsf{in}}^n \to \Sigma_{\mathsf{out}}^n$ *be a tree code.*

- *We say that* TC *is a* binary tree code *if* $\Sigma_{\mathsf{in}} = \{0,1\}$.
- *Assume* $\Sigma_{\mathsf{in}}, \Sigma_{\mathsf{out}}$ *are rings.* TC *is said to be* linear *if for every* $t \in [n]$, $\mathsf{TC}(x)_t$ *is a linear function of* x.
- *We say that* TC *is* explicit *if it can be evaluated on every input* $m \in \Sigma_{\mathsf{in}}^n$ *in polynomial time in the bit complexity of* m.

We also consider the stronger notion of infinite tree codes, as was done in the original papers [50, 52]. For a set Σ, we denote the set of all sequences $(x_i)_{i\in\mathbb{N}}$, where $x_i \in \Sigma$, by $\Sigma^{\mathbb{N}}$. One can extend the notion of a split and of online functions to functions of the form $f\colon \Sigma_{\mathsf{in}}^{\mathbb{N}} \to \Sigma_{\mathsf{out}}^{\mathbb{N}}$ in the natural way.

Definition 3.7 ([50, 52])**.** *An online function* $\mathsf{TC}\colon \Sigma_{\mathsf{in}}^{\mathbb{N}} \to \Sigma_{\mathsf{out}}^{\mathbb{N}}$ *is a* tree code with distance δ *if for every distinct* $x, y \in \Sigma_{\mathsf{in}}^{\mathbb{N}}$, *with* $s = \mathsf{split}(x,y)$, *and every integer* $\ell \ge 0$,

$$\mathrm{dist}\left(\mathsf{TC}(x)_{[s,s+\ell]}, \mathsf{TC}(y)_{[s,s+\ell]}\right) \ge \delta(\ell+1).$$

Note that an infinite tree code with distance δ yields, for every integer $n \ge 1$, a tree code of depth n with distance δ. We extend, in the natural way, the property of linearity for infinite tree codes. We say that an infinite tree code is explicit if for every $t \in \mathbb{N}$, the restriction of TC to its first t coordinates is explicit as a finite tree code. Such a restriction is well-defined as TC is an online function.

3.3 The Newton Basis

For $k \in \mathbb{N}$, the Newton polynomial $\binom{x}{k} \in \mathbb{R}[x]$ is defined by

$$\binom{x}{k} = \frac{x(x-1)\cdots(x-(k-1))}{k!}.$$

As mentioned, $\{\binom{x}{k}\}_{k=0}^{n}$ is a basis for the space of polynomials of degree at most n, over $\mathbb{R}$. In fact any function $f\colon \mathbb{N} \to \mathbb{R}$ can be expanded as a pointwise-converging power series over this basis. The following lemma gives a formula for the coefficients of the expansion.

Lemma 3.8. *Let* $f : \mathbb{N} \to \mathbb{R}$. *Then, for* $x \in \mathbb{N}$,

$$f(x) = \sum_{k=0}^{x} \gamma_k \binom{x}{k} = \sum_{k\ge 0} \gamma_k \binom{x}{k} \tag{3.1}$$

where

$$\gamma_k = \sum_{i=0}^{k} (-1)^{k-i} \binom{k}{i} f(i) = \sum_{i\ge 0} (-1)^{k-i} \binom{k}{i} f(i).$$

Furthermore, if $f \in \mathbb{R}[x]$ *is a polynomial of degree* n *then it equals the sum of the first* $n+1$ *terms of expansion* (3.1).

This is simply the inversion formula for a triangular matrix; for a proof see, e.g., Appendix A in [13].

3.4 The Gessel-Viennot Lemma

Gessel and Viennot (1985) proved a result with the following corollary, which is generally associated with their names:

Lemma 3.9 ([24], Corollary 2)**.** *Let* $0 \le a_1 < a_2 < \cdots < a_n$ *and* $0 \le b_1 < b_2 < \cdots < b_n$ *be integers. Define the* $n \times n$ *matrix* $M = M(a,b)$ *by* $M_{i,j} = \binom{a_i}{b_j}$*. Then* $a_i \ge b_i$ *for each* $i \in [n]$ *iff* $\det M \neq 0$.

(For more recent treatments see [2], Chapter 5.4 or [3], Chapter 25.)

As it happens, this lemma is much older. It appears explictly in Zia-uddin (1933) [62], and can be easily derived from a result of Pólya (1931) [44]. In Appendix C we explain this and reconnect a few articles in the literature.

4 A Bound on the Number of Integral Roots via Sparsity

Proof of Lemma 1.4. The proof is by contradiction. Let $s \ge 1$ be least integer such that there is a counterexample: a polynomial $f \in \mathbb{R}[x]$ with sparsity s, specified by integers $0 \le c_1 < \cdots < c_s$ and non-zero real numbers $\gamma_1, \ldots, \gamma_s$ such that

$$f(x) = \sum_{i=1}^{s} \gamma_i \binom{x}{c_i}$$

(note that $c = c_1$), and such that there exist integers t_i with $c \le t_1 < \cdots < t_s$ such that all $f(t_i) = 0$. Necessarily $s > 1$ as the case $s = 1$ merely reflects that $\binom{x}{c} \neq 0$ for $x \ge c$.

Now we claim that all $t_j > c_j$; this clearly holds for $j = 1$ as $f(c_1) = \gamma_1 \neq 0$. By way of contradiction, let $j \ge 2$ be a counterexample. Then, $t_{j-1} < t_j \le c_j$. By Lemma 3.8, the polynomial $\sum_{i=1}^{j-1} \gamma_i \binom{x}{c_i}$ agrees with f on $\{0, \ldots, c_j - 1\}$, so it has distinct roots $t_1, \ldots, t_{j-1}$ (all $> c_1$ since the claim holds for $j = 1$) and sparsity $j - 1$; since $j - 1 \le s - 1 < s$, this contradicts the minimality of f.

Finally, consider the $s \times s$ matrix A with entries $A_{i,j} = \binom{t_i}{c_j}$ for $i, j \in [s]$. Let $\gamma \in \mathbb{R}^s$ be the vector with entries γ_j. Then $f(t_i) = (A\gamma)_i$ so, all t_i being roots, $A\gamma = \bar{0}$. However, since $c_1 < \cdots < c_s$, $t_1 < \cdots < t_s$, and all $c_j < t_j$, the hypothesis of Lemma 3.9 is met for A and so $\det A \neq 0$, a contradiction to $\gamma \neq \bar{0}$. □

5 Infinite Tree Codes over the Integers

In this section we prove Theorem 1.3. We start by defining the construction of $\mathsf{TC}_{\mathbb{Z}}$.

The construction of $\mathsf{TC}_{\mathbb{Z}}$. Define the function $\mathsf{TC}_{\mathbb{Z}} \colon \mathbb{Z}^{\mathbb{N}} \to (\mathbb{Z}^2)^{\mathbb{N}}$ as follows. For $z = (z_t)_{t\in\mathbb{N}} \in \mathbb{Z}^{\mathbb{N}}$ let $f \colon \mathbb{N} \to \mathbb{N}$ be such that $f(t) = z_t$ for all $t \in \mathbb{N}$. By Lemma 3.8, one can expand f in the Newton basis

$$f(x) = \sum_{t\in\mathbb{N}} \gamma_t \binom{x}{t}.$$

For $t \in \mathbb{N}$, define $\mathsf{TC}_{\mathbb{Z}}(z)_t = (z_t, \gamma_t)$.

Analysis. By Lemma 3.8, for all $t \in \mathbb{N}$, γ_t is a $\mathbb{Z}$-linear combination of $z_0, \ldots, z_t$ and so $\mathsf{TC}_{\mathbb{Z}}$ is a linear online function with the asserted range. For $t \in \mathbb{N}$, define $m_t = \max(|z_i| \colon i \in \{0, 1, \ldots, t\})$. By Lemma 3.8,

$$|\gamma_t| \le \sum_{i=0}^{t} \binom{t}{i} |z_i| \le 2^t m_t.$$

Finally, since in the Theorem statement the output alphabet is $\mathbb{Z}$, one must reencode the pair (z_t, γ_t) as a single, not too large integer. In the lower-order bits, encode z_t using, say, Elias's prefix-free γ-encoding of the integers [16]; this requires no more than $1 + 2\lg\lfloor 2|z_t| + 1\rfloor$ bits. Then write γ_t in the higher-order bits. The total number bits required is therefore bounded by $t + 2\lg|z_t| + c$ for some constant c. This proves the asserted bound on the output symbols.

We turn to analyze the distance of $\mathsf{TC}_{\mathbb{Z}}$. As $\mathsf{TC}_{\mathbb{Z}}$ is linear, it suffices to consider a nonzero sequence $z = (z_t)_{t\in\mathbb{N}}$. Assume that $c \in \mathbb{N}$ is the least integer such that $z_c \neq 0$. Let $\ell \ge 0$. Set $t = c + \ell$ and define $I = \{c, c+1, \ldots, t\}$. Let $f_t \in \mathbb{R}[x]$ be the polynomial

$$f_t(x) = \sum_{i=0}^{t} \gamma_i \binom{x}{i}.$$

Observe that for every $i \in I$, the first entry of the pair $\mathsf{TC}_{\mathbb{Z}}(z)_i$ equals $z_i = f(i) = f_t(i)$. Let s be the sparsity of f_t in the Newton basis. Note that precisely s of the pairs $(\mathsf{TC}_{\mathbb{Z}}(z)_i)_{i\in I}$ have a nonzero second entry. On the other hand, by Lemma 1.4, f_t has at most $s - 1$ roots in I. Hence, the number of indices $i \in I$ for which the first entry of $\mathsf{TC}_{\mathbb{Z}}(z)_i$ is 0 is bounded above by $s - 1$. Thus, the number of indices $i \in I$ for which $\mathsf{TC}_{\mathbb{Z}}(z)_i$ is nonzero (as a pair) is bounded below by

$$\max(s, \ell + 1 - (s-1)) \ge \frac{\ell+1}{2}.$$

This completes the proof of Theorem 1.3.

We remark that obtaining a construction with output symbols that are bounded by $\mathrm{poly}(t)$ rather than the exponential dependence that was obtained above, would yield asymptotically good binary tree code.

6 Binary Tree Codes with Polylogarithmic Size Alphabet

In this section we prove Theorem 1.1. We do so for distance $\delta = 1/3$ based on Theorem 1.3. In Section 7, we explain how to achieve any distance $\delta < 1$ based on Theorem 1.2. We start by deducing the following corollary from Theorem 1.3.

Corollary 6.1. *For every integer $\ell \geq 1$ there exists an explicit tree code*

$$\mathsf{TC}_\ell \colon \left(\{0,1\}^\ell\right)^\ell \to \left(\{0,1\}^{3\ell}\right)^\ell$$

with distance $1/2$.

We remark that by using Pudlák's construction ([45], Lemma 6.1), one can obtain an explicit tree code $\mathsf{TC}\colon (\{0,1\}^{\ell^3})^\ell \to (\{0,1\}^{O(\ell^3)})^\ell$ with constant distance. This construction too would have sufficed in place of Gessel-Viennot as a starting point for the proof of Theorem 1.1, albeit with weaker parameters.

Proof of Corollary 6.1. The tree code TC_ℓ is obtained by restricting $\mathsf{TC}_\mathbb{Z}$, defined in Section 5, to its first ℓ coordinates where, for an integer b, we identify $\{0,1\}^b$ with $\{0,1,\dots,2^b-1\}$.

It will be more convenient to start the index set of TC_ℓ from 1 rather than 0 as was done in $\mathsf{TC}_\mathbb{Z}$. Note that the input symbols, when represented as integers, are bounded by $2^\ell - 1$. Therefore, by Theorem 1.3 for every $t \in [\ell]$, the t'th output symbol is bounded in absolute value by

$$2^{t-1}(2^\ell - 1)^2 \leq 2^{\ell-1}(2^\ell-1)^2 \leq 2^{3\ell-1} - 1,$$

and so, using the fact that we have a bound on the magnitude of the input integers in advance, 3ℓ bits suffice to represent the output symbols of $\mathsf{TC}_\mathbb{Z}$, including the sign of γ_t. Thus TC_ℓ inherits the distance $1/2$ bound from $\mathsf{TC}_\mathbb{Z}$. □

For the proof of Theorem 1.1, it will be convenient to introduce a relaxed notion of tree codes which we call *lagged tree codes*. Roughly, these are tree codes that are only required to gain distance after some lag from the split.

Definition 6.2. *An online function $\mathsf{TCLag}\colon \Sigma_{\mathsf{in}}^n \to \Sigma_{\mathsf{out}}^n$ is a* lagged tree code *with distance δ and lag L if for every distinct $x, y \in \Sigma_{\mathsf{in}}^n$, with $s = \mathsf{split}(x,y)$, and every integer $L \leq \ell \leq n-s$,*

$$\mathrm{dist}\left(\mathsf{TCLag}(x)_{[s,s+\ell]}, \mathsf{TCLag}(y)_{[s,s+\ell]}\right) \geq \delta(\ell+1).$$

We borrow the terminology used for tree codes for lagged tree codes. That is, we refer to n as the depth of TCLag and to $\Sigma_{\mathsf{in}}, \Sigma_{\mathsf{out}}$ as the input alphabet and output alphabet, respectively. We also extend Definition 3.6 to lagged tree codes in the natural way. Note that a tree code is a lagged tree code with lag $L = 0$. At the other extreme, using error correcting block codes, it is not hard to obtain explicit binary lagged tree codes with a trivial lag of ℓ (that is, the guarantee on the distance holds only after reading the entire codeword), constant distance, and $|\Sigma_{\mathsf{out}}| = O(1)$. In the following claim, we obtain explicit binary lagged tree codes with a constant distance, $|\Sigma_{\mathsf{out}}| = O(1)$, and depth that is quadratic in the lag.

Claim 6.3. *There exists a constant $c_{\mathsf{lag}} \geq 1$ such that for every integer $\ell \geq 1$ there exists an explicit lagged tree code $\mathsf{TCLag}_\ell \colon \{0,1\}^\ell \to (\{0,1\}^{c_{\mathsf{lag}}})^\ell$ with distance $1/3$ and lag $L = 16\sqrt{\ell}$.* [5]

Proof. First, note that we may assume $\ell \geq \ell_0$ for any desired constant ℓ_0. Indeed, based on the probabilistic proof for the existence of tree codes [50] one can efficiently find, via a brute force search, a tree code of any constant size and distance $1/3$ thus proving the claim for ℓ_0. From here on we assume $\ell \geq \ell_0$ for a large enough constant ℓ_0 and so we may ignore immaterial technical issues such as rounding.

For the construction of TCLag_ℓ we make use of the following building blocks:

- Let $\mathsf{TC}_{\sqrt{\ell}} \colon \left(\{0,1\}^{\sqrt{\ell}}\right)^{\sqrt{\ell}} \to \left(\{0,1\}^{3\sqrt{\ell}}\right)^{\sqrt{\ell}}$ be the tree code from Corollary 6.1. Recall that $\mathsf{TC}_{\sqrt{\ell}}$ has distance $1/2$.

- Let $\mathsf{ECC} \colon \{0,1\}^{3\sqrt{\ell}} \to (\{0,1\}^{c_{\mathsf{lag}}})^{\sqrt{\ell}}$ be the error correcting block code from Lemma 3.2 set with distance $5/6$. By Lemma 3.2, c_{lag} is a constant.

Let $m \in \{0,1\}^\ell$. Partition m to $\sqrt{\ell}$ consecutive blocks each consisting of $\sqrt{\ell}$ bits, namely, $m = (m_1, \ldots, m_{\sqrt{\ell}})$ where $m_i \in \{0,1\}^{\sqrt{\ell}}$. Similarly, for $t \in [\sqrt{\ell}]$, we write $\mathsf{TCLag}_\ell(m)_t$ for $\mathsf{TCLag}_\ell(m)$ projected to the t'th block, where each block consists of $\sqrt{\ell}$ elements of $\{0,1\}^{c_{\mathsf{lag}}}$. Formally, $\mathsf{TCLag}_\ell(m)_t = \mathsf{TCLag}_\ell(m)_{[(t-1)\sqrt{\ell}+1, t\sqrt{\ell}]} \in (\{0,1\}^{c_{\mathsf{lag}}})^{\sqrt{\ell}}$. We define $\mathsf{TCLag}_\ell(m)_1 = (0^{c_{\mathsf{lag}}})^{\sqrt{\ell}}$. For $t \in \{2, \ldots, \sqrt{\ell}\}$, define

$$\mathsf{TCLag}_\ell(m)_t = \mathsf{ECC}\left(\mathsf{TC}_{\sqrt{\ell}}(m)_{t-1}\right),$$

where we interpret m as an element of $(\{0,1\}^{\sqrt{\ell}})^{\sqrt{\ell}}$ when passing it to $\mathsf{TC}_{\sqrt{\ell}}$.

Observe that TCLag_ℓ is online. We turn to show that TCLag_ℓ has distance $1/3$ and lag $16\sqrt{\ell}$. Let $x, y \in \{0,1\}^\ell$ be distinct strings with $s = \mathsf{split}(x,y)$. Let $d \in [16\sqrt{\ell}, \ell - s]$. Let $i_1 \in [\sqrt{\ell}]$ be the index for which the blocks $x_{i_1}, y_{i_1} \in \{0,1\}^{\sqrt{\ell}}$ contain the split s. That is, i_1 is the split of x, y when interpreted as elements of $(\{0,1\}^{\sqrt{\ell}})^{\sqrt{\ell}}$. Note that $i_1 = \lfloor (s-1)/\sqrt{\ell} \rfloor + 1$. Set $b = \lfloor d/\sqrt{\ell} \rfloor - 1$ and observe that block numbers $i_1+1, \ldots, i_1+b-1$ are all fully contained in $[s, s+d]$. Hence, by construction, the codeword $\mathsf{TCLag}_\ell(x)$ projected to $[s, s+d]$ contains

$$(\mathsf{TCLag}_\ell(x)_{i_1+1}, \ldots, \mathsf{TCLag}_\ell(x)_{i_1+b-1}) = \left(\mathsf{ECC}\left(\mathsf{TC}_{\sqrt{\ell}}(x)_{i_1}\right), \ldots, \mathsf{ECC}\left(\mathsf{TC}_{\sqrt{\ell}}(x)_{i_1+b-2}\right)\right)$$

[5] One can achieve distance $1/2 - \varepsilon$ for any constant $\varepsilon > 0$. This will effect the value of the constant c_{lag} and the constant multiplying $\sqrt{\ell}$ in the lag L.

as a substring. Similarly, the codeword $\mathsf{TCLag}_\ell(y)$ projected to $[s, s+d]$ contains

$$(\mathsf{TCLag}_\ell(y)_{i_1+1}, \ldots, \mathsf{TCLag}_\ell(y)_{i_1+b-1}) = \left(\mathsf{ECC}\left(\mathsf{TC}_{\sqrt{\ell}}(y)_{i_1}\right), \ldots, \mathsf{ECC}\left(\mathsf{TC}_{\sqrt{\ell}}(y)_{i_1+b-2}\right)\right)$$

as a substring in the corresponding indices.

As $\mathsf{TC}_{\sqrt{\ell}}$ is a tree code with distance $1/2$ and i_1 is the split of x, y when considered as elements of $(\{0,1\}^{\sqrt{\ell}})^{\sqrt{\ell}}$, at least $(b-1)/2$ of the indices $i_1, \ldots, i_1+b-2$ are such that $\mathsf{TC}_{\sqrt{\ell}}(x)_i \neq \mathsf{TC}_{\sqrt{\ell}}(y)_i$. As ECC has distance $5/6$, each such index i contributes $5\sqrt{\ell}/6$ to the total distance. Thus, the number of disagreements between $\mathsf{TCLag}_\ell(x)$, $\mathsf{TCLag}_\ell(y)$ projected to $[s, s+d]$ is bounded below by

$$\frac{b-1}{2} \cdot \frac{5}{6}\sqrt{\ell} \geq \frac{d}{3},$$

where the last inequality follows as $d \geq 16\sqrt{\ell}$. □

Let $\ell \leq n$ be integers. Let c_{lag} be the constant from Claim 6.3. Define the function $\mathsf{TCLag}_\ell^n \colon \{0,1\}^n \to (\{0,1\}^{2c_{\mathsf{lag}}})^n$ as follows. Let $m \in \{0,1\}^n$. Write[6] $m = (m_1, \ldots, m_{2n/\ell})$ where each $m_i \in \{0,1\}^{\ell/2}$. For $i = 1, \ldots, n/\ell$ define

$$\begin{aligned} o_i &= \mathsf{TCLag}_\ell(m_{2i-1}, m_{2i}), \\ e_i &= \mathsf{TCLag}_\ell(m_{2i}, m_{\min(2i+1, 2n/\ell)}). \end{aligned}$$

(In the second equation the minimum with $2n/\ell$ is taken only to make sure that we do not go out of the index set of the message.)

Note that each of o_i, e_i is an element of $(\{0,1\}^{c_{\mathsf{lag}}})^\ell$. Let $c_i = (o_i, e_i)$ of which we think of as an element of $(\{0,1\}^{2c_{\mathsf{lag}}})^\ell$. Define

$$\mathsf{TCLag}_\ell^n(m) = (c_1, c_2, \ldots, c_{n/\ell}).$$

Claim 6.4. *Let x, y be distinct n-bit strings and let $s = \mathsf{split}(x, y)$. Assume that $s \leq n - \ell/2$. Then, for every integer $16\sqrt{\ell} \leq d \leq \ell/2$,*

$$\mathrm{dist}\left(\mathsf{TCLag}_\ell^n(x)_{[s,s+d]}, \mathsf{TCLag}_\ell^n(y)_{[s,s+d]}\right) \geq d/3.$$

Proof. Let $i \in [2n/\ell]$ be the index of blocks x_i, y_i that contain the split s. Assume i is odd and let $t = s - (i-1)\ell/2$ be the index within block i of the split. Then $\mathsf{TCLag}_\ell^n(x)_{[s,s+d]}$ contains, as a substring, $\mathsf{TCLag}_\ell(x_i, x_{i+1})_{[t,t+d]}$, where we have used the fact that $t + d \leq \ell$ as implied by the hypothesis $d \leq \ell/2$ and since, by construction, $t \leq \ell/2$. Similarly, $\mathsf{TCLag}_\ell^n(y)_{[s,s+d]}$ contains $\mathsf{TCLag}_\ell(y_i, y_{i+1})_{[t,t+d]}$ in the corresponding indices. The proof follows as TCLag_ℓ is a lagged tree code with distance $1/3$ and lag $16\sqrt{\ell}$. A similar argument holds for even i's. □

We are now ready to prove Theorem 1.1 (for distance $\delta = 1/3$).

[6] For the sake of clarity, we ignore issues of divisibility that can be easily handled.

Proof of Theorem 1.1. First, observe that it suffices to construct a lagged tree code with distance $1/3$ and lag $L = O(1)$. Such a lagged tree code can be efficiently converted to a tree code with distance $1/3$ and only a constant overhead in alphabet size by including the last L inputs in the encoding at any point of time. Set $j = \log_2 \log_2 n$ and define the sequence of integers $\ell_1, \ldots, \ell_j$ recursively as follows: $\ell_1 = 2^{20}$ and for $i \geq 1$, $\ell_{i+1} = \ell_i^2/2^{10}$. One can verify that $\ell_i = 2^{10(2^{i-1}+1)}$ and so for every integer $2^{14} \leq d \leq n$ there exists $i \in [j]$ such that $16\sqrt{\ell_i} \leq d \leq \ell_i/2$.

We define $\mathsf{TC}\colon \{0,1\}^n \to \Sigma^n$ as follows. Let $m \in \{0,1\}^n$. For $i \in [j]$ define $t_i = \mathsf{TCLag}^n_{\ell_i}(m) \in (\{0,1\}^{2c_{\mathsf{lag}}})^n$. Define

$$\mathsf{TC}(m) = (t_1, t_2, \ldots, t_j) \in (\{0,1\}^{2c_{\mathsf{lag}} j})^n.$$

Note that $\Sigma = \{0,1\}^{2c_{\mathsf{lag}} j}$ is an alphabet of size $(\log n)^{O(1)}$. TC is an online function as each of the functions $\mathsf{TCLag}^n_{\ell_1}, \ldots, \mathsf{TCLag}^n_{\ell_j}$ is online. We turn to show that TC is a lagged tree code with lag 2^{14} and distance $1/3$. Let $x, y \in \{0,1\}^n$ be distinct with $s = \mathsf{split}(x,y)$. Let $2^{14} \leq d \leq n - s$. By the above, there exists $i \in [j]$ such that $16\sqrt{\ell_i} \leq d \leq \ell_i/2$. Hence, $\mathsf{TCLag}^n_{\ell_i}$ guarantees that the fraction of disagreements between $\mathsf{TC}(x)$ and $\mathsf{TC}(y)$ projected to $[s, s+d]$ is at least $1/3$. □

7 Tree Codes for any Distance $\delta < 1$

In this section we prove the following theorem which readily implies Theorem 1.2.

Theorem 7.1. *For every integer $r \geq 1$ there exists an explicit tree code $\mathsf{TC}^r_{\mathbb{Z}}\colon \mathbb{Z}^{\mathbb{N}} \to (\mathbb{Z}^{r+1})^{\mathbb{N}}$ with distance $1 - 1/(r+1)$. Further, for every $z = (z_t)_{t \in \mathbb{N}} \in \mathbb{Z}^{\mathbb{N}}$ and $t \in \mathbb{N}$, each of the $r+1$ integers in $\mathsf{TC}^r_{\mathbb{Z}}(z)_t$ is bounded, in absolute value, by $2^{O(t^2 r)} \cdot \max(|z_0|, \ldots, |z_t|)$.*

Proof. Let $r\mathbb{N} = \{0, r, 2r, \ldots\}$ be the set of all natural numbers that are divisible by r, and let $\bar{r}\mathbb{N} = \mathbb{N} \setminus r\mathbb{N}$. Formally, the tree code $\mathsf{TC}^r_{\mathbb{Z}}$ defined next has domain $\mathbb{Z}^{r\mathbb{N}}$ but, for ease of readability, we identify $\mathbb{Z}^{r\mathbb{N}}$ with $\mathbb{Z}^{\mathbb{N}}$ in the natural way.

The construction of $\mathsf{TC}^r_{\mathbb{Z}}$. Define the function $\mathsf{TC}^r_{\mathbb{Z}}\colon \mathbb{Z}^{\mathbb{N}} \to (\mathbb{Z}^{r+1})^{\mathbb{N}}$ as follows. Given $z = (z_t)_{t \in r\mathbb{N}}$, we define a sequence of real polynomials $(f_t)_{t \in \mathbb{N}}$ and a sequence of integers $(z_t)_{t \in \bar{r}\mathbb{N}}$ inductively with respect to $t \in \mathbb{N}$ as follows:

1. Define f_{tr} to be the least degree real polynomial such that $\forall i \in \{0, 1, \ldots, tr\}$, $f_{tr}(i) = z_i$.
2. Define the real polynomials $f_{tr+1}, \cdots, f_{tr+r-1}$ to equal f_{tr}.
3. For $i = 1, \ldots, r-1$, set $z_{tr+i} = f_{tr+i}(tr+i)$.

By Lemma 3.8, there exists a sequence of integers $(\gamma_i)_{i\in\mathbb{N}}$ such that for every $t \in \mathbb{N}$, $f_t(x) = \sum_{i=0}^{t} \gamma_i \binom{x}{i}$. For $t \in \mathbb{N}$, define

$$\mathsf{TC}^r_{\mathbb{Z}}(z)_t = (\gamma_{tr}, z_{tr}, z_{tr+1}, z_{tr+r-1}).$$

Analysis. Observe that the γ_t's as well as z_t for $t \in \bar{r}\mathbb{N}$ are all $\mathbb{Z}$-linear combination of the input sequence $(z_t)_{t\in r\mathbb{N}}$, and so $\mathsf{TC}^r_{\mathbb{Z}}$ is a linear function with range $\mathbb{Z}^{r+1}$. Further, $\mathsf{TC}^r_{\mathbb{Z}}$ is online.

We turn to analyze the distance of $\mathsf{TC}^r_{\mathbb{Z}}$. As $\mathsf{TC}^r_{\mathbb{Z}}$ is linear, it suffices to consider a nonzero sequence z. Assume that $c \in \mathbb{N}$ is the least integer such that $z_{cr} \neq 0$. Let $\ell \geq 0$. Set $t = c+\ell$ and define $I = \{c, c+1, \ldots, t\}$. Recall that $f_{tr}(x) = \sum_{i=0}^{tr} \gamma_i \binom{x}{i}$. By (2), $\gamma_i = 0$ for every $i \in \bar{r}\mathbb{N}$ and so

$$f_{tr}(x) = \sum_{i=0}^{t} \gamma_{ir} \binom{x}{ir}.$$

Denote the sparsity of f_{tr} by s. Note that s of the γ-entries in $(\mathsf{TC}^r_{\mathbb{Z}}(z)_i)_{i\in I}$ are nonzero. By Lemma 1.4, f_{tr} has at most $s-1$ roots in $[cr, \infty) \cap \mathbb{Z}$ and, in particular, at most $s-1$ roots among $\{cr, cr+1, \ldots, tr+r-1\}$. As the evaluation of f_{tr} on these $(\ell+1)r$ points appear as entries in $(\mathsf{TC}^r_{\mathbb{Z}}(z)_i)_{i\in I}$, we have that at least

$$s + (\ell+1)r - (s-1) = (\ell+1)r + 1$$

of the $(\ell+1)(r+1)$ integers in $((\mathsf{TC}^r_{\mathbb{Z}})_i)_{i\in I}$ are nonzero. Thus, at least

$$\frac{(\ell+1)r+1}{r+1} > \left(1 - \frac{1}{r+1}\right)(\ell+1)$$

of the indices $i \in I$ are such that $\mathsf{TC}^r_{\mathbb{Z}}(z)_i$ is nonzero as an $(r+1)$-tuple, establishing the desired bound on the distance.

We turn to bound the output symbols. We start by bounding the γ symbols. Recall that $\gamma_i = 0$ for every i not divisible by r. For $t \in \mathbb{N}$, let $\Gamma_t = \max(|\gamma_i| \colon i \in \{0, 1, \ldots, t\})$ and define $m_{tr} = \max(|z_i| \colon i \in r\mathbb{N} \cap [0, tr])$. By Lemma 3.8, for every $t \geq 1$ we have that

$$\begin{aligned}
\gamma_{tr} &= \sum_{i=0}^{tr} (-1)^{tr-i} \binom{tr}{i} z_i \\
&= z_{tr} + \sum_{i=0}^{tr-1} (-1)^{tr-i} \binom{tr}{i} \sum_{j=0}^{i} \gamma_j \binom{i}{j},
\end{aligned}$$

and so

$$\begin{aligned}
|\gamma_{tr}| &\le |z_{tr}| + \sum_{i=0}^{tr-1} \binom{tr}{i} \sum_{j=0}^{i} |\gamma_j| \binom{i}{j} \\
&\le |z_{tr}| + \sum_{i=0}^{tr-1} \binom{tr}{i} \Gamma_i 2^i \\
&\le |z_{tr}| + \Gamma_{tr-1} \sum_{i=0}^{tr} \binom{tr}{i} 2^i \\
&= |z_{tr}| + 3^{tr}\Gamma_{tr-1}.
\end{aligned}$$

As $\Gamma_{tr-1} = \Gamma_{(t-1)r}$ and since, being an input symbol, $|z_{tr}| \le m_{tr}$, we have that $\Gamma_{tr} \le 3^{tr}\Gamma_{(t-1)r} + m_{tr}$, which implies $\Gamma_{tr} \le m_{tr} \cdot 3^{t^2 r}$.

As for the computed z values, by (2) and (3), for $t \in \mathbb{N}$ and $k \in \{1, 2, \ldots, r-1\}$,

$$\begin{aligned}
z_{tr+k} &= f_{tr+k}(tr+k) \\
&= f_{tr}(tr+k) \\
&= \sum_{i=0}^{t} \gamma_{ir} \binom{tr+k}{ir} \\
&\le m_{tr} \cdot 3^{t^2 r} \cdot 2^{tr+k},
\end{aligned}$$

which completes the proof of Theorem 7.1. □

Corollary 7.2. *There exists a universal constant $c \ge 1$ such that for every $\zeta > 0$ and every integer $\ell \ge 1$ there exists an explicit tree code*

$$\mathsf{TC}_\ell \colon \left(\{0,1\}^{\ell^2}\right)^\ell \to \left(\{0,1\}^{\frac{c}{\zeta^2}\cdot \ell^2}\right)^\ell$$

with distance $\delta = 1 - \zeta$.

Proof. The tree code TC_ℓ is obtained by restricting $\mathsf{TC}_{\mathbb{Z}}^r$, set with $r = \lceil 1/\zeta \rceil$, to its first ℓ coordinates where, as in the proof of Corollary 6.1, for an integer b, we identify $\{0,1\}^b$ with $\{0, 1, \ldots, 2^b - 1\}$. Note that the input symbols, when represented as integers, are bounded by 2^{ℓ^2}. Therefore, by Theorem 7.1, there exists a universal constant $c \ge 1$ such that every output symbol is bounded in absolute value by $2^{c(1/\zeta)^2 \ell^2}$ and so $c(1/\zeta)^2\ell^2$ bits suffice to represent the output symbols of $\mathsf{TC}_{\mathbb{Z}}^r$. Clearly, TC_ℓ inherits the $1 - 1/(r+1) \ge 1 - \zeta$ distance bound from $\mathsf{TC}_{\mathbb{Z}}^r$. □

The reduction in Section 6, instantiated with Corollary 7.2 instead of Corollary 6.1, and when executed with a suitable choice of parameters, yields a binary tree code with distance $\delta = 1 - \zeta$ and alphabet size $(\log n)^{O(1/\zeta^2)}$.

Acknowledgments

Thanks to David Zuckerman and Klim Efremenko for illuminating Pólya's work; and thanks to Noga Ron-Zewi and Shachar Lovett for many helpful discussions.

References

[1] S. Agrawal, R. Gelles, and A. Sahai. Adaptive protocols for interactive communication. In *Proc. IEEE International Symposium on Information Theory (ISIT)*, pages 595–599, 2016.

[2] M. Aigner. *A course in enumeration.* Springer, 2007.

[3] M. Aigner and G. Ziegler. *Proofs from the Book.* Springer, 4th edition, 2010.

[4] N. Alon, M. Braverman, K. Efremenko, R. Gelles, and B. Haeupler. Reliable communication over highly connected noisy networks. In *Proc. ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*, pages 165–173, 2016.

[5] Z. Brakerski and Y. T. Kalai. Efficient interactive coding against adversarial noise. In *Proc. IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 160–166, 2012.

[6] Z. Brakerski, Y. T. Kalai, and M. Naor. Fast interactive coding against adversarial noise. *Journal of the ACM (JACM)*, 61(6):35:1–30, 2014.

[7] Z. Brakerski and M. Naor. Fast algorithms for interactive coding. In *Proc. ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 443–456, 2013.

[8] M. Braverman. Towards deterministic tree code constructions. In *Proc. ACM-SIGACT Innovations in Theoretical Computer Science Conference (ITCS)*, pages 161–167, 2012.

[9] M. Braverman and K. Efremenko. List and unique coding for interactive communication in the presence of adversarial noise. In *Proc. IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 236–245, 2014.

[10] M. Braverman, R. Gelles, J. Mao, and R. Ostrovsky. Coding for interactive communication correcting insertions and deletions. *IEEE Transactions on Information Theory*, 63(10):6256–6270, 2017.

[11] M. Braverman and A. Rao. Toward coding for maximum errors in interactive communication. *IEEE Transactions on Information Theory*, 60(11):7248–7255, 2014.

[12] K. Chandrasekaran, N. Goyal, and B. Haeupler. Deterministic algorithms for the lovász local lemma. *SIAM Journal on Computing*, 42(6):2132–2155, 2013.

[13] G. Cohen, A. Shpilka, and A. Tal. On the degree of univariate polynomials over the integers. In *Electronic Colloquium on Computational Complexity (ECCC)*, 2011. TR11-002.

[14] Y. Dodis and A. B. Lewko. Interactive coding for interactive proofs. In *Proc. International Theory of Cryptography Conference (TCC)*, pages 352–366, 2016.

[15] A. Drucker. Efficient probabilistically checkable debates. In *Proc. International Workshop on Randomization and Computation (RANDOM)*, pages 519–529, 2011.

[16] P. Elias. Universal codeword sets and representations of the integers. *IEEE Transactions on Information Theory*, 21(2):194–203, March 1975.

[17] P. Erdös and L. Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. In A. Hajnal, L. Lovász, and V. T. Sós, editors, *Infinite and Finite Sets*. North Holland, 1975.

[18] M. Franklin, R. Gelles, R. Ostrovsky, and L. J. Schulman. Optimal coding for streaming authentication and interactive communication. *IEEE Transactions on Information Theory*, 61(1):133–145, 2015.

[19] A. Garcia and H. Stichtenoth. A tower of Artin-Schreier extensions of function fields attaining the Drinfeld-Vladut bound. *Inventiones Mathematicae*, 121(1):211–222, 1995.

[20] J. von zur Gathen and J. R. Roche. Polynomials with two values. *Combinatorica*, 17(3):345–362, 1997.

[21] R. Gelles. Coding for interactive communication: A survey. *Foundations and Trends in Theoretical Computer Science*, 13(1-2):1–157, 2017.

[22] R. Gelles, B. Haeupler, G. Kol, N. Ron-Zewi, and A. Wigderson. Towards optimal deterministic coding for interactive communication. In *Proc. ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1922–1936, 2016.

[23] R. Gelles, A. Moitra, and A. Sahai. Efficient coding for interactive communication. *IEEE Transactions on Information Theory*, 60(3):1899–1913, 2014.

[24] I. Gessel and G. Viennot. Binomial determinants, paths, and hook length formulae. *Advances in Mathematics*, 58(3):300–321, 1985.

[25] M. Ghaffari and B. Haeupler. Optimal Error Rates for Interactive Coding II: Efficiency and List Decoding. In *Proc. IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 394–403, 2014.

[26] M. Ghaffari, B. Haeupler, and M. Sudan. Optimal error rates for interactive coding I: Adaptivity and other settings. In *Proc. ACM Symposium on Theory of Computing (STOC)*, pages 794–803, 2014.

[27] L. Grosjean. *Practical anytime codes.* PhD thesis, KTH Royal Institute of Technology, 2016.

[28] B. Haeupler. Interactive Channel Capacity Revisited. In *Proc. IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 226–235, 2014.

[29] B. Haeupler and A. Shahrasbi. Synchronization strings: Codes for insertions and deletions approaching the singleton bound. In *Proc. ACM Symposium on Theory of Computing (STOC)*, 2017.

[30] B. Haeupler and A. Shahrasbi. Synchronization strings: Explicit constructions, local decoding, and applications. In *Proc. ACM Symposium on Theory of Computing (STOC)*, 2018.

[31] B. Haeupler, A. Shahrasbi, and E. Vitercik. Synchronization strings: Channel simulations and interactive coding for insertions and deletions. In *Proc. International Colloquium on Automata, Languages, and Programming (ICALP)*, 2018. arXiv:1707.04233.

[32] B. Haeupler and A. Velingker. Bridging the capacity gap between interactive and one-way communication. In *Proc. ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2123–2142, 2017.

[33] A. Jain, Y. T. Kalai, and A. B. Lewko. Interactive coding for multiparty protocols. In *Proc. ACM Innovations in Theoretical Computer Science Conference (ITCS)*, pages 1–10, 2015.

[34] S. N. Kersey. Invertibility of submatrices of the Pascal matrix and Birkhoff interpolation. *Journal of Mathematical Sciences: Advances and Applications*, 41:45–56, 2016. ArXiv:1303.6159v2.

[35] A. Khina, W. Halbawi, and B. Hassibi. (Almost) practical tree codes. In *Proc. IEEE International Symposium on Information Theory (ISIT)*, pages 2404–2408, 2016.

[36] G. Kol and R. Raz. Interactive channel capacity. In *Proc. ACM Symposium on Theory of Computing (STOC)*, pages 715–724, 2013.

[37] J. R. Lee, A. de Mesmay, and M. Moharrami. Dimension reduction for finite trees in L_1. In *Proc. ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 43–50, 2012.

[38] A. Lewko and E. Vitercik. Balancing communication for multi-party interactive coding. *CoRR*, abs/1503.06381, 2015.

[39] B. Lindström. On the vector representations of induced matroids. *Bulletin of the London Mathematical Society*, 5(1):85–90, 1973.

[40] C. Moore and L. J. Schulman. Tree codes and a conjecture on exponential sums. In *Proc. ACM-SIGACT Innovations in Theoretical Computer Science Conference (ITCS)*, pages 145–154, 2014.

[41] R. A. Moser and G. Tardos. A constructive proof of the general Lovász local lemma. *Journal of the ACM (JACM)*, 57(2):11, 2010.

[42] R. Ostrovsky, Y. Rabani, and L. J. Schulman. Error-correcting codes for automatic control. *IEEE Trans. Information Theory*, 55(7):2931–2941, 2009.

[43] M. Peczarski. An improvement of the tree code construction. *Information Processing Letters (IPL)*, 99(3):92–95, 2006.

[44] G. Pólya. Bemerkungen zur interpolation and zur näherungstheorie der balkenbiegung. *Z. Angew. Math. Mech.*, 11:445–449, 1931.

[45] P. Pudlák. Linear tree codes and the problem of explicit constructions. *Linear Algebra and its Applications*, 490:124–144, 2016.

[46] S. Rajagopalan and L. J. Schulman. A coding theorem for distributed computation. In *Proc. ACM Symposium on Theory of Computing (STOC)*, pages 790–799, 1994.

[47] A. Sahai. *Anytime information theory*. PhD thesis, Massachusetts Institute of Technology, 2001.

[48] A. Sahai. The necessity and sufficiency of anytime capacity for control over a noisy communication link. In *Proc. 43rd IEEE Conference on Decision and Control*, pages 1896–1901, 2004.

[49] A. Sahai and S. Mitter. The necessity and sufficiency of anytime capacity for stabilization of a linear system over a noisy communication link. *IEEE Transactions on Information Theory*, 52(8):3369–3395, 2006.

[50] L. J. Schulman. Deterministic coding for interactive communication. In *Proc. ACM Symposium on Theory of Computing (STOC)*, pages 747–756, 1993.

[51] L. J. Schulman. Postscript of 21 September 2003 to Coding for Interactive Communication.
http://users.cms.caltech.edu/∼schulman/Papers/intercodingpostscript.txt, 1994.

[52] L. J. Schulman. Coding for interactive communication. *IEEE Transactions on Information Theory*, 42(6):1745–1756, 1996.

[53] A. A. Sherstov and P. Wu. Optimal interactive coding for insertions, deletions, and substitutions. In *Proc. IEEE Symposium on Foundations of Computer Science (FOCS)*, 2017.

[54] A. Shpilka and A. Tal. On the minimal Fourier degree of symmetric boolean functions. In *Proc. 26th Annual Conference on Computational Complexity (CCC)*, pages 200–209. IEEE, 2011.

[55] P. Stevenhagen and H. W. Lenstra, Jr. Chebotarëv and his density theorem. *The Mathematical Intelligencer*, 18(2), 1996.

[56] H. Stichtenoth. *Algebraic function fields and codes*, volume 254. Springer Science & Business Media, 2009.

[57] R. T. Sukhavasi and B. Hassibi. Anytime reliable codes for stabilizing plants over erasure channels. In *Proc. IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*, pages 5254–5259, 2011.

[58] R. T. Sukhavasi and B. Hassibi. Linear error correcting codes with anytime reliability. In *Proc. IEEE International Symposium on Information Theory (ISIT)*, pages 1748–1752, 2011.

[59] J. M. Whittaker. On Lidstone's series and two-point expansions of analytic functions. *Proc. London Math. Soc.*, s2-36:451–469, 1934.

[60] J. M. Whittaker. *Interpolatory Function Theory*. Cambridge U. Press, 1935.

[61] A. Wigderson. *Mathematics and Computation*. Princeton U. Press, 2017. Book draft.

[62] Zia-uddin. Note on an "alternant" with factorial elements. *Proc. Edinburgh Math. Soc.*, 3:296–299, 1933.

A Proof of Lemma 3.2

For the proof of Lemma 3.2, we make use of algebraic-geometric codes.

Theorem A.1 ([19]. See also [56])**.** *Let p be a prime number and $m \in \mathbb{N}$ even. Set $q = p^m$. For every $0 < \rho < 1$ and a large enough integer n, there exists an explicit rate ρ linear error correcting block code $\mathsf{ECC}\colon \mathbb{F}_q^n \to \mathbb{F}_q^{n/\rho}$ with distance*

$$\delta \geq 1 - \rho - \frac{1}{\sqrt{q} - 1}.$$

Proof of Lemma 3.2. Let ℓ be the least integer such that $2^\ell \geq 2/\varepsilon + 1$ and let $q = 2^{2\ell}$. Let m be the least integer such that $1/m \leq \varepsilon/2$. By Theorem A.1, there exists an explicit error correcting block code $\mathsf{ECC}'\colon \mathbb{F}_q^n \to \mathbb{F}_q^{mn}$ with distance $1 - \varepsilon$. Identify $\mathbb{F}_q = \mathbb{F}_{2^{2\ell}}$ with $\{0,1\}^{2\ell}$ by fixing a representation for $\mathbb{F}_q$. Note that $q = O(1/\varepsilon^2) = O(1)$ and so such a representation can be computed in constant time. Set $c = 2\ell t m$. Define, $\mathsf{ECC}\colon \{0,1\}^n \to (\{0,1\}^c)^{n/t}$ as follows. Given $x \in \{0,1\}^n$, identify x with an element of $\mathbb{F}_2^n \subseteq \mathbb{F}_q^n$. For $i \in [n/t]$, define

$$\mathsf{ECC}(x)_i = \left(\mathsf{ECC}'(x)_{(i-1)mt+1},\, \mathsf{ECC}'(x)_{(i-1)mt+2},\, \ldots,\, \mathsf{ECC}'(x)_{imt}\right).$$

Note that $c = O((t/\varepsilon)\log(1/\varepsilon))$.

Consider distinct $x, y \in \{0,1\}^n$ and, as before, identify $\{0,1\}^n$ with $\mathbb{F}_2^n \subseteq \mathbb{F}_q^n$. Then, the codewords $\mathsf{ECC}'(x)$, $\mathsf{ECC}'(y)$ agree on at most ε-fraction of the coordinates. This readily implies that $\mathsf{ECC}(x)$, $\mathsf{ECC}(y)$ agree on at most ε-fraction of the coordinates. $\square$

B Strongly Palindrome-Avoiding Labelings

Here is a variation on the tree code construction problem. For $w = w_1 \ldots w_n$, $w_i \in \Sigma$ a word, let $\overline{w} = w_n \ldots w_1$ be its reversal.

Definition B.1. *$DPal(w) = \frac{1}{n} d_{Hamming}(w, \overline{w})$.*

Consider a mapping α from the edges of the infinite (rooted) 3-ary tree into Σ. For vertices u, v let $\alpha(u, v)$ be the word one reads along the simple path from u to v. For $\delta < 1$, say α is $(1-\delta)$-*palindrome avoiding* if for all u, v separated by distance greater than 1, $\mathrm{DPal}(w) \geq 1 - \delta$.

It is a consequence of the Lovász local lemma that for any $\delta > 0$ there is a $|\Sigma| < \infty$ so that such labelings α exist. One would like to construct one, in the sense that the label of any edge at distance n from the root can be computed in time polynomial in n. This is essentially a generalization of the tree code problem, in that one now also cares about pairs of vertices x, y that are not at the same level of the tree.

(The construction problem is easy for the weaker requirement that every $\alpha(u, v)$ of length at least 2 not be a palindrome.)

C Lemma 3.9 and Pólya's Two-Point Interpolation

C.1 Background

The proofs by Zia-uddin and by Gessel and Viennot (using an idea of Lindström [39]) of Lemma 3.9 are entirely combinatorial.

Pólya, on the other hand, gave an analytic proof—which we thank David Zuckerman for explaining to us—of the following result in interpolation theory:

Theorem C.1 (Pólya [44]). *For g a real univariate polynomial of degree at most n, let $g^{(k)}$ denote its k'th derivative. Let x and y be any two distinct reals, and let $S, T \subseteq \{0, \ldots, n\}$. Then g is determined by the pair of sequences $(g^{(k)}(x))_{k \in S}$, $(g^{(k)}(y))_{k \in T}$ if and only if for every $0 \leq n' \leq n$, $|S \cap \{0, \ldots, n'\}| + |T \cap \{0, \ldots, n'\}| \geq n' + 1$.*

(The "only if" is immediate; the "if" is not, and fails for instance for the analogous statement for three interpolation points.)

As noted by Kersey [34], Theorem C.1 implies Lemma 3.9; we explain this below.

Gessel and Viennot were unaware of Zia-uddin's work or that their corollary could be obtained from Pólya's work; Kersey was unaware of the work of Gessel-Viennot or Zia-uddin; and Whittaker, who used Zia-uddin's result to obtain consequences for two-point interpolation [59], was unaware (even in the monograph [60]) of Pólya's result on two-point interpolation which implies the result of Zia-uddin. We are confident that we are unaware of very much more.

C.2 Proof of Lemma 3.9 from Theorem C.1

Let $x = 0$, $y = 1$, and let a_i be the index of the ith "missing evaluation" at 0, that is, $a_i =$ the least n' such that $|S \cap \{0, \ldots, n'\}| = n' + 1 - i$, up until there is no such $n' \leq n$. Let b_i be the index of the ith "present evaluation" at 1, that is, $b_i =$ the least n' such that $|T \cap \{0, \ldots, n'\}| = i$.

Let $\bar{a}_1 \leq \ldots$ be the complement of S in $\{0, \ldots, n'\}$. The map $g \to ((g^{(k)}(0))_{k \in S}, (g^{(k)}(1))_{k \in T})$ is linear; in order to write it down explicitly we specify g by the values $g^{(k)}(0)$. Let us permute these values into the order $(\bar{a}_1, \ldots)$ followed by $(a_1, \ldots)$, that is, into the vector $(g^{(\bar{a}_1)}(0), \ldots, g^{(a_1)}(0), \ldots)$. The map carries this to the vector $(g^{(\bar{a}_1)}(0), \ldots, g^{(b_1)}(1), \ldots)$. In matrix form (acting on row vectors) the map is

$$\left(\begin{array}{c|c} I & M(\bar{a}, b) \\ \hline 0 & M(a, b) \end{array}\right)$$

This matrix is nonsingular if and only if $M(a, b)$ is nonsingular. It remains to verify that the condition "$\forall i\ a_i \geq b_i$" of Lemma 3.9 is the same as the condition "$\forall n' \ldots \geq n' + 1$" of Theorem C.1. For the $\Leftarrow$ direction, suppose to the contrary there is an i s.t. $a_i < b_i$. Then set

$n' = a_i$ and note that $|S \cap \{0, \ldots, n'\}| = n'+1-i$ while $|T \cap \{0, \ldots, n'\}| < |T \cap \{0, \ldots, b_i\}| = i$, so $|S \cap \{0, \ldots, n'\}| + |T \cap \{0, \ldots, n'\}| \leq n'$. For the $\Rightarrow$ direction, suppose to the contrary there is an n' s.t. $|S \cap \{0, \ldots, n'\}| + |T \cap \{0, \ldots, n'\}| \leq n'$. Then in particular $|S \cap \{0, \ldots, n'\}| \leq n'$, so $a_1 \leq n'$; let i be greatest s.t. $a_i \leq n'$. Then $|S \cap \{0, \ldots, n'\}| = n' + 1 - i$, so $|T \cap \{0, \ldots, n'\}| \leq i - 1$, and consequently $b_i > n'$. Therefore $b_i > a_i$.

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