

# Noisy Interpolating Sets for Low Degree Polynomials

Zeev Dvir <sup>\*</sup>      Amir Shpilka <sup>†</sup>

## Abstract

A Noisy Interpolating Set (NIS) for degree  $d$  polynomials is a set  $S \subseteq \mathbb{F}^n$ , where  $\mathbb{F}$  is a finite field, such that any degree  $d$  polynomial  $q \in \mathbb{F}[x_1, \dots, x_n]$  can be efficiently interpolated from its values on  $S$ , even if an adversary corrupts a constant fraction of the values. In this paper we construct explicit NIS for every prime field  $\mathbb{F}_p$  and any degree  $d$ . Our sets are of size  $O(n^d)$  and have efficient interpolation algorithms that can recover  $q$  from a fraction  $\exp(-O(d))$  of errors.

Our construction is based on a theorem which roughly states that if  $S$  is a NIS for degree 1 polynomials then  $d \cdot S = \{a_1 + \dots + a_d \mid a_i \in S\}$  is a NIS for degree  $d$  polynomials. Furthermore, given an efficient interpolation algorithm for  $S$ , we show how to use it in a black-box manner to build an efficient interpolation algorithm for  $d \cdot S$ .

As a corollary we get an explicit family of punctured Reed-Muller codes that is a family of good codes that have an efficient decoding algorithm from a constant fraction of errors. To the best of our knowledge no such construction was known previously.

## 1 Introduction

An interpolating set for degree  $d$  polynomials is a set of point  $S \subseteq \mathbb{F}^n$  such that if  $q(x_1, \dots, x_n)$  is a degree  $d$  polynomial over  $\mathbb{F}$  and we are given the set of values  $(q(s))_{s \in S}$  then we can reconstruct  $q$ . That is, we can find all the coefficients of  $q$ . A set  $S$  is a *noisy interpolating set* for degree  $d$  polynomials if we can reconstruct  $q$  even if an adversary corrupts an  $\epsilon$  fraction of the values in  $(q(s))_{s \in S}$ . It is not difficult to prove that a random set of size  $O(n^d)$  is a noisy interpolating set. In this paper we study the problem of giving an explicit construction of a small noisy interpolating set for degree  $d$  polynomials, that has an efficient interpolation algorithm from a constant fraction of errors. Beside being a very natural problem, this question is related to two other interesting problems: Giving explicit constructions of punctured Reed-Muller codes that are good codes and the problem of constructing pseudo-random generators against low degree polynomials. Before stating our results we describe the connection to these two problems.

---

<sup>\*</sup>Department of Computer Science, Weizmann institute of science, Rehovot, Israel. zeev.dvir@weizmann.ac.il. Research supported by Binational Science Foundation (BSF) grant, by Israel Science Foundation (ISF) grant and by Minerva Foundation grant.

<sup>†</sup>Faculty of Computer Science, The Technion, Haifa, Israel. shpilka@cs.technion.ac.il. Research supported by the Israel Science Foundation (grant number 439/06).

Reed-Muller codes are an important family of error correcting codes. They are obtained by evaluating low degree polynomials on all possible inputs taken from a finite field  $\mathbb{F}$ . More precisely, in the code  $RM(n, d)$  the messages correspond to polynomials  $q(x_1, \dots, x_n)$  over  $\mathbb{F}$  of total degree  $d$ , and the encoding is the vector of evaluations  $(q(\bar{\alpha}))_{\bar{\alpha} \in \mathbb{F}^n}$ . The well known Hadamard code, obtained from evaluating linear functions, is the code  $RM(n, 1)$  (for more information on Reed-Muller codes see [MS77]). Although Reed-Muller codes are not *good-codes* (they do not have constant rate) they play an important role both in practice and in theory. For example Reed-Muller codes are used in the early proofs of the PCP theorem [AS98, ALM<sup>+</sup>98], in constructions of pseudo-random generators [BFNW93, IW97, STV01] and extractors [TSZS01, SU05], and more. It is thus an important goal to better understand these codes and their properties.

A *puncturing* of a code is the operation of throwing away some of the coordinates of every code word. For example, every linear code (without repeated coordinates) is a punctured Hadamard code. In particular, a punctured Reed-Muller code is a mapping that sends a degree  $d$  polynomial  $q$  to its vector of values on some subset  $S \subseteq \mathbb{F}^n$ . As Reed-Muller codes are not good codes themselves, it is an interesting question to find a puncturing that yields good codes. As before, it is not difficult to see that a random subset of  $\mathbb{F}^n$  of size  $O(n^d)$  gives a good punctured Reed-Muller code. An even more interesting question is to find a puncturing that yields good codes that have efficient encoding and decoding algorithms (for a random  $S$  we don't have an efficient decoding algorithm). It is not hard to see that  $S$  is a noisy interpolating set (that has an efficient interpolation algorithm) if and only if the mapping  $q \rightarrow (q(\bar{\alpha}))_{\bar{\alpha} \in S}$  is a code of linear minimal distance (that has an efficient decoding algorithm). Our noisy interpolating sets are of size  $O(n^d)$  and therefore they give a construction of good punctured Reed-Muller codes that can be decoded efficiently. To the best of our knowledge no such construction was known prior to our result.

Another problem that is related to our results is the problem of constructing *pseudo-random generators* for low degree polynomials. A mapping  $G : \mathbb{F}^s \rightarrow \mathbb{F}^n$  is  $\epsilon$ -prg against degree  $d$  polynomials if it fools every degree  $d$  polynomial in  $n$  variables. That is, the two distributions  $q(U_n)$ , where  $U_n$  is chosen uniformly from  $\mathbb{F}^n$ , and  $q(G(U_s))$ , where  $U_s$  is chosen uniformly from  $\mathbb{F}^s$ , have statistical distance at most  $\epsilon$ , for every degree  $d$  polynomial  $q$ . Pseudo-random generators and low degree polynomials have many application in theoretical computer science, and so it is an important question to give explicit constructions of prg's for low degree polynomials, that have good parameters (i.e.  $s$  as small as possible). For the case of degree 1 polynomials, it is not hard to see that any noisy interpolating set implies a prg and viceversa. That is, if  $G : \mathbb{F}^s \rightarrow \mathbb{F}^n$  is a prg for polynomials of degree 1 then the image of  $G$  is a noisy interpolating set. Similarly if  $S \subseteq \mathbb{F}^n$  is a noisy interpolating set for degree 1 polynomials then any one to one mapping whose image is  $S$  is a prg for such polynomials. For larger values of  $d$  there is no such strong correspondence between prg's and noisy interpolating sets. As before, given a prg we get a noisy interpolating set (without an obvious efficient interpolation algorithm), however it is not clear how to get a prg from a noisy interpolating set. Constructions of prg's against low degree polynomials over very large fields were given in [Bog05] using tools from algebraic geometry. Very recently, independently from our work, Viola [Vio07], building on the works of [BV07, Lov07], showed (over any field) that  $d \cdot S$  is prg for degree  $d$  polynomials whenever  $S$  is prg for degree 1

polynomials. In particular this result implies that  $d \cdot S$  is a noisy interpolating set. However [Vio07] did not think of the problem of giving a noisy interpolating set and in particular did not give an efficient interpolation algorithm.

In the next section (“Our result”) we consider two types of noisy interpolating sets. The first one allows for repetition of points. That is, we allow  $S$  to be a multiset. This corresponds to repeating coordinates in an error correcting code. In the second variant (*proper* NIS) we do not allow such repetitions, and require  $S$  to be a simple set (and not a multiset). This is what allows us to get punctured Reed-Muller codes. For each one of these types we prove a composition theorem, saying that the sumset of a NIS for degree 1 polynomials is a NIS for higher degree polynomials (the non-multiset version of this theorem requires an additional condition on the initial NIS). We then combine these theorems with known constructions of error correcting codes to obtain our final results.

## 1.1 Our Results

In the following  $\mathbb{F}$  will always denote a finite field of prime cardinality. We denote by  $\mathbb{F}[x_1, \dots, x_n]$  the space of all polynomials in the variables  $x_1, \dots, x_n$  with coefficients in  $\mathbb{F}$ . We also denote by  $\mathbb{F}_d[x_1, \dots, x_n]$  the set of all degree  $\leq d$  polynomials in  $\mathbb{F}[x_1, \dots, x_n]$ . Since we are interested in polynomials only as *functions* over  $\mathbb{F}^n$  we will assume in the following that the individual degree of each variable is smaller than  $p = |\mathbb{F}|$ . This is w.l.o.g since for every  $a \in \mathbb{F}$  we have  $a^p = a$ . For example, over  $\mathbb{F}_2$ , the field of two elements, we will only consider multilinear polynomials.

**Definition 1.1 (Noisy Interpolating Set (NIS)).** *Let  $S = (a_1, \dots, a_m) \in (\mathbb{F}^n)^m$  be a list of points (not necessarily distinct) in  $\mathbb{F}^n$ . We say that  $S$  is an  $(n, d, \epsilon)$ -Noisy Interpolating Set (NIS) if there exists an algorithm  $A_S$  such that for every  $q \in \mathbb{F}_d[x_1, \dots, x_n]$  and for every vector  $e = (e_1, \dots, e_m) \in \mathbb{F}^m$  such that  $|\{i \in [m] \mid e_i \neq 0\}| \leq \epsilon \cdot m$ , the algorithm  $A_S$ , when given as input the list of values  $(q(a_1) + e_1, \dots, q(a_m) + e_m)$ , outputs the polynomial  $q$  (as a list of its coefficients). We say that  $S$  is a *proper* NIS if the points  $a_1, \dots, a_m$  are distinct. If  $S$  is a *proper* NIS we can treat it as a subset  $S \subseteq \mathbb{F}^n$ .*

*Let  $\mathcal{S} = (S^{(n)})_{n \in \mathbb{N}}$  be a sequence such that for every  $n \in \mathbb{N}$  we have that  $S^{(n)}$  is an  $(n, d, \epsilon)$ -NIS ( $d$  and  $\epsilon$  are fixed for all  $n$ ). We say that  $\mathcal{S}$  has an efficient interpolation algorithm if there exists a polynomial time algorithm  $M(n, L)$  that takes as input an integer  $n$  and a list  $L$  of values in  $\mathbb{F}$  such that the restriction  $M(n, \cdot)$  has the same behavior as the algorithm  $A_{S^{(n)}}$  described above (in places where there is no risk of confusion we will omit the superscript  $(n)$  and simply say that  $S$  has an efficient interpolation algorithm).*

### 1.1.1 Multiset NIS

Our first theorem shows how to use a NIS for degree 1 polynomials in order to create a NIS for higher degree polynomials. In order to state the theorem we need the following notation. For two lists<sup>1</sup> of points  $S = (a_1, \dots, a_m) \in (\mathbb{F}^n)^m$  and  $T = (b_1, \dots, b_\ell) \in (\mathbb{F}^n)^\ell$  we define the sum  $S \boxplus T$  to be the list  $(a_i + b_j)_{i \in [m], j \in [\ell]} \in (\mathbb{F}^n)^{m \cdot \ell}$ . The reason for the notation is to make

<sup>1</sup>Instead of a list one can have a multi-set in mind, however it is slightly easier to describe and prove our results using the notion of lists.

a clear distinction between sets and multi-sets. In particular we shall use the notation  $S + T$  to denote set addition. That is, for two sets  $S$  and  $T$  we define  $S + T = \{s + t \mid s \in S, t \in T\}$ .

**Theorem 1.** *Let  $0 < \epsilon_1 < 1/2$  be a real number. Let  $S_1$  be an  $(n, 1, \epsilon_1)$ -NIS and for each  $d > 1$  let  $S_d = S_{d-1} \boxplus S_1$ . Then for every  $d > 1$  the set  $S_d$  is an  $(n, d, \epsilon_d)$ -NIS with  $\epsilon_d = (\epsilon_1/2)^d$ . Moreover, if  $S_1$  has an efficient interpolation algorithm, then so does  $S_d$ .*

Combining Theorem 1 with known constructions of error correcting codes (in order to define  $S_1$ ) we get the following corollary:

**Corollary 1.2.** *For every prime field  $\mathbb{F}$  and for every  $d > 0$ , there exists an  $\epsilon > 0$  and a family  $\mathcal{S} = (S^{(n)})_{n \in \mathbb{N}}$  such that for every  $n \in \mathbb{N}$ ,  $S^{(n)}$  is an  $(n, d, \epsilon)$ -NIS, and such that  $\mathcal{S}$  has an efficient interpolation algorithm. Moreover, for each  $n \in \mathbb{N}$  we have  $|S^{(n)}| = O(n^d)$  and it is possible to generate the set  $S^{(n)}$  in time  $\text{poly}(n)$ .*

### 1.1.2 Proper NIS

We are able to prove an analog of Theorem 1 for proper NIS. We would like to show that if  $S_1$  is a proper  $(n, 1, \epsilon_1)$ -NIS then the sets  $S_d = S_{d-1} + S_1$  (this time we use the operation of set addition) are proper  $(n, d, \epsilon_d)$ -NIS's. However, in order to prove the theorem, we will need the initial set  $S_1$  to satisfy a certain natural condition on the intersections of different 'shifts' of the sets  $S_{d-1}$ . Roughly speaking, this condition will guarantee that there is no small set of errors in  $S_d = S_{d-1} + S_1$  that has a large intersection with many shifts of  $S_{d-1}$  (one can see from the proof of Theorem 1 why this is useful). The operation of set addition is defined as follows: Let  $A$  and  $B$  be two subsets of  $\mathbb{F}^n$ . We define  $A + B = \{a + b \mid a \in A, b \in B\}$ . For an element  $c \in \mathbb{F}^n$  and for a subset  $A \subseteq \mathbb{F}^n$  we denote  $A + c = \{a + c \mid a \in A\}$ .

**Definition 1.3 (Condition  $\star_k$ ).** *Let  $S \subseteq \mathbb{F}^n$ . Let  $S_0 = \{0\}$  and for each  $d \geq 1$  let  $S_d = S_{d-1} + S$ . Let  $k > 0$  be an integer. For  $x \in S_d$  we let  $N_d(x) = |\{b \in S \mid x \in S_{d-1} + b\}|$ . We say that  $S$  satisfies condition  $\star_k$  if for every  $0 < d \leq k$  we have*

$$|\{x \in S_d \mid N_d(x) > d\}| \leq |S_{d-2}|$$

We are now ready to state the analog of Theorem 1 for proper NIS's.

**Theorem 2.** *Let  $0 < \epsilon_1 < 1/2$  be a real number and let  $k > 0$  an integer. There exists a constant  $C_0$ , depending only on  $\epsilon$  and  $k$ , such that for all  $n > C_0$  the following holds: Let  $S_1$  be a proper  $(n, 1, \epsilon_1)$ -NIS and for each  $d > 1$  let  $S_d = S_{d-1} + S_1$ . Suppose  $S_1$  satisfies condition  $\star_k$ . Then, for every  $1 < d \leq k$  the set  $S_d$  is a (proper)  $(n, d, \epsilon_d)$ -NIS with*

$$\epsilon_d = \frac{1}{d!} \cdot \left(\frac{\epsilon_1}{3}\right)^d.$$

Moreover, if  $S_1$  has an efficient interpolation algorithm, then so does  $S_d$ .

As before, combining Theorem 2 with known constructions of error correcting codes gives the following corollary:

**Corollary 1.4.** *For every prime field  $\mathbb{F}$  and for every  $d > 0$ , there exists an  $\epsilon > 0$  and a family  $\mathcal{S} = (S^{(n)})_{n \in \mathbb{N}}$  such that for every  $n \in \mathbb{N}$ ,  $S^{(n)}$  is a proper  $(n, d, \epsilon)$ -NIS with an efficient interpolation algorithm. Moreover, for each  $n \in \mathbb{N}$  we have  $|S^{(n)}| = O(n^d)$  and it is possible to generate the set  $S^{(n)}$  in time  $\text{poly}(n)$ .*

It is easy to see that any  $(n, d, \epsilon)$ -proper NIS induces a puncturing of the Reed-Muller code  $RM(n, d)$  that has an efficient decoding from a fraction  $\epsilon$  of errors (we keep the coordinates of the RM-code corresponding to the points in the NIS). The following corollary is therefore immediate. (See Section 2.2 for the definition of a good family of codes).

**Corollary 1.5.** *For every prime field  $\mathbb{F}$  and for every  $d > 0$ , there exists an  $\epsilon > 0$  and a family  $\{C_n\}$  of good codes such that  $C_n$  is a punctured  $RM(n, d)$ -code. Furthermore, the family  $\{C_n\}$  has an efficient encoding and decoding algorithms from a constant fraction of errors.*

## 1.2 Organization

In Section 2 we give preliminary results that we will use in the rest of the paper. In Section 3 we prove Theorem 1 and Corollary 1.2. In Section 4, we prove Theorem 2 and Corollary 1.4. Appendix A describes a construction of a family of error correcting codes with certain special properties that is used in Section 4.

## 2 Preliminaries

Let  $q \in \mathbb{F}[x_1, \dots, x_n]$  be a polynomial. We denote by  $\deg(q)$  the total degree of  $q$ . The  $i$ -th homogeneous part of  $q$  is the sum of all monomials in  $q$  of degree  $i$ , and is denoted with  $q_i$ . We denote  $q_{\leq i} = \sum_{j=0}^i q_j$ . We now give some useful facts and definitions regarding partial derivatives of polynomials and error correcting codes.

### 2.1 Partial and directional derivatives

In this section we define partial and directional derivatives of polynomials over finite fields and discuss some of their properties.

We start by defining the partial derivatives of a monomial. Let  $M(x) = x_1^{c_1} \cdot \dots \cdot x_n^{c_n}$  be a monomial in  $n$  variables so that  $c_i \geq 0$  for all  $i$ . The partial derivative of  $M(x)$  with respect to  $x_i$  is zero if  $c_i = 0$  and

$$\frac{\partial M}{\partial x_i}(x) = c_i \cdot x_1^{c_1} \cdot \dots \cdot x_i^{c_i-1} \cdot \dots \cdot x_n^{c_n}$$

if  $c_i > 0$ . The definition extends to general polynomials by linearity. It is easy to verify that if  $\deg(q) = d$  then all of the partial derivatives of  $q$  are of degree at most  $d - 1$ . Also, since we assume that the individual degrees are smaller than the characteristic of the field we have that  $\frac{\partial q}{\partial x_i}(x) \equiv 0$  iff  $x_i$  does not appear in  $q(x)$ . We denote by

$$\Delta_q(x) = \left( \frac{\partial q}{\partial x_1}(x), \dots, \frac{\partial q}{\partial x_n}(x) \right)$$

the partial derivative vector of  $q$ . The directional derivative of a polynomial  $q \in \mathbb{F}[x_1, \dots, x_n]$  in direction  $a \in \mathbb{F}^n$  is denoted by  $\partial_q(x, a)$  and is defined as

$$\partial_q(x, a) = \sum_{i=1}^n a_i \cdot \frac{\partial q}{\partial x_i}(x).$$

That is,  $\partial_q(x, a)$  is the inner product of  $a$  and  $\Delta_q(x)$ .

The following lemma gives a more ‘geometric’ view of the partial derivatives of  $q$  by connecting them to difference of  $q$  along certain directions.

**Lemma 2.1.** *Let  $q \in \mathbb{F}_d[x_1, \dots, x_n]$  and let  $q_d$  be its homogenous part of degree  $d$ . Let  $a, b \in \mathbb{F}^n$ . Then*

$$q(x+a) - q(x+b) = \partial_{q_d}(x, a-b) + E(x),$$

where  $\deg(E) \leq d-2$  (we use the convention that  $\deg(0) = -\infty$ ). That is, the directional derivative of  $q_d$  in direction  $a-b$  is given by the homogenous part of degree  $d-1$  in the difference  $q(x+a) - q(x+b)$ .

*Proof.* It is enough to prove the lemma for the case that  $q$  is a monomial of degree  $d$ . The result will then follow from linearity and from the fact that the derivatives of all monomials in  $q$  of degree smaller than  $d$  are of degree at most  $d-2$ . Let  $M(x) = x_1^{c_1} \cdot \dots \cdot x_n^{c_n}$  be a monomial of degree  $d$ . We observe that the expression  $M(x+a)$  can be written as

$$\begin{aligned} M(x+a) &= \prod_{i=1}^n (x_i + a_i)^{c_i} \\ &= M(x) + \sum_{i=1}^n a_i \cdot \frac{\partial M}{\partial x_i}(x) + E_1(x), \end{aligned}$$

where  $\deg(E_1) \leq d-2$ . Now, subtracting  $M(x+b)$  from  $M(x+a)$ , the term  $M(x)$  cancels out and we are left with the desired expression.  $\square$

The next easy lemma shows that a polynomial  $q$  can be reconstructed from his partial derivatives.

**Lemma 2.2.** *Let  $q \in \mathbb{F}_d[x_1, \dots, x_n]$ . Given the vector of partial derivatives  $\Delta_q(x)$ , it is possible to reconstruct  $q$  in polynomial time.*

*Proof.* We go over all monomials of degree  $\leq d$  and find out the coefficient they have in  $q$  using the following method. For a monomial  $M$  let  $i$  be the first index such that  $x_i$  appears in the  $M$  with positive degree. Consider  $\frac{\partial q}{\partial x_i}(x)$  and check whether the coefficient of the derivative of that monomial is zero or not. For example, if we want to ‘decode’ the monomial  $x_1^5 \cdot x_2^3$  we will look at the coefficient of  $x_1^4 \cdot x_2^3$  in  $\frac{\partial q}{\partial x_1}(x)$ . To get the coefficient in  $q$  we have to divide it by the degree of  $x_i$  in the monomial, that is by 5 (remember that the individual degrees are smaller than  $p = |\mathbb{F}| = \text{char}(\mathbb{F})$ ).  $\square$

## 2.2 Error correcting codes

We start with the basic definitions of error correcting codes. Let  $\mathbb{F}$  be a field and let  $E : \mathbb{F}^n \rightarrow \mathbb{F}^m$ . Denote by  $C = E(\mathbb{F}^n)$  the image of  $E$ . Then  $C$  is called an  $[m, n, k]$ -code if for any two codewords  $E(v), E(u) \in C$ , where  $u \neq v$ , we have that  $\text{dist}(E(u), E(v)) \geq k$  ('dist' denotes the hamming distance). We refer to  $m$  as the *block length* of  $C$  and to  $k$  as the *minimal distance* of  $C$ . We denote by  $R = n/m$  the *relative rate* of  $C$  and with  $\delta = k/m$  the *relative minimal distance* of  $C$ , and say that  $C$  is an  $[R, \delta]$ -code. When  $E$  is a linear mapping we say that  $C$  is a linear code and call the  $m \times n$  matrix computing  $E$  the *generating matrix* of  $C$ . A map  $D : \mathbb{F}^m \rightarrow \mathbb{F}^n$  can correct  $t$  errors (w.r.t.  $E$ ) if for any  $v \in \mathbb{F}^n$  and any  $w \in \mathbb{F}^m$  such that  $\text{dist}(E(v), w) \leq t$  we have that  $D(w) = v$ . Such a  $D$  is called a *decoding algorithm* for  $C$ .

A family of codes  $\{C_i\}$ , where  $C_i$  is an  $[R_i, \delta_i]$ -code of block length  $m_i$ , has *constant rate* if there exists a constant  $R > 0$  such that for all codes in the family it holds that  $R_i \geq R$ . The family has a *linear distance* if there exists a constant  $\delta > 0$  such that for all codes in the family we have  $\delta_i \geq \delta$ . In such a case we say that the family is a family of  $[R, \delta]$  codes. If a family of codes as above has  $\lim_{i \rightarrow \infty} m_i = \infty$ , a constant rate and a linear minimal distance then we say that the family is a family of *good codes* and that the codes in the family are good. Similarly, we say that the family of codes has a decoding algorithm for a fraction  $\tau$  of errors if for each  $C_i$  there is a decoding algorithm  $D_i$  that can decode from  $\tau \cdot m_i$  errors. When  $C$  is a linear code we define the dual of  $C$  in the following way:

$$C^\perp \triangleq \{y \in \mathbb{F}^m : \forall x \in C \langle y, x \rangle = 0\}.$$

**Lemma 2.3.** *Let  $C$  be an  $[m, n, k]$ -code over  $\mathbb{F}$  such that  $C$  has an efficient decoding algorithm that can correct an  $\alpha$ -fraction of errors. For  $i \in [m]$  let  $a_i \in \mathbb{F}^n$  be the  $i$ 'th row of the generating matrix of  $C$ . Then,*

1. *Let  $S^0 = (a_1, \dots, a_m, \bar{0}, \dots, \bar{0}) \in (\mathbb{F}^n)^{2m}$ . Then  $S^0$  is an  $(n, 1, \alpha/2)$ -NIS with an efficient interpolation algorithm.*
2. *Let  $S = (a_1, \dots, a_m) \in (\mathbb{F}^n)^m$  and suppose that the maximal hamming weight of a codeword in  $C$  is smaller than  $(1 - 2\alpha) \cdot m$ . Then  $S$  is an  $(n, 1, \alpha)$ -NIS with an efficient interpolation algorithm.*

*Proof. 1.* Let  $q$  be a degree 1 polynomial. The interpolation algorithm for  $S^0$  will work as follows: we first look at the values of  $q(x)$  on the last  $m$  points (the zeros). The majority of these values will be  $q(0)$  which will give us the constant term in  $q$ . Let  $q_1(x) = q(x) - q(0)$  be the linear part of  $q$ . Subtracting  $q(0)$  from the values in the first  $m$  coordinates we reduce ourselves to the problem of recovering the homogenous linear function  $q_1$  from its values on  $(a_1, \dots, a_m)$  with at most  $\alpha \cdot m$  errors. This task is achieved using the decoding algorithm for  $C$ , since the vector  $(q_1(a_1), \dots, q_1(a_m))$  is just the encoding of the vector of coefficients of  $q_1$  with the code  $C$ .

**2.** Here we cannot decipher the constant term of  $q(x)$  so easily. Let  $(v_1, \dots, v_m) \in \mathbb{F}$  be the list of input values given to us ( $v_i = q(a_i)$  for a  $1 - \alpha$  fraction of the  $i$ 's). What we will do is we will go over all  $p = |\mathbb{F}|$  possible choices for  $q(0)$  and for each 'guess'  $c \in \mathbb{F}$  do the

following: Subtract  $c$  from the values  $(v_1, \dots, v_m)$  and then use the decoding algorithm of  $C$  to decode the vector  $V_c = (v_1 - c, \dots, v_m - c)$ . It is clear that for  $c = q(0)$ , this procedure will give us the list of coefficients of  $q(x)$  (without the constant term). Our task is then to figure out which invocation of the decoding algorithm is the correct one.

Say that the decoding algorithm, on input  $V_c$  returned a linear polynomial  $q_c(x)$  (there is no constant term). We can then check to see whether  $q_c(a_i) + c$  is indeed equal to  $v_i$  for a  $1 - \alpha$  fraction of the  $i$ 's. If we could show that this test will succeed only for a single  $c \in \mathbb{F}$  then we are done and the lemma will follow. Suppose in contradiction that there are two linear polynomial  $q_c(x)$  and  $q_{c'}(x)$  such that both agree with a fraction  $1 - \alpha$  of the input values. This means that there exist two codewords  $W_c, W_{c'} \in \mathbb{F}^m$  (the encodings of  $q_c$  and  $q_{c'}$ ) in  $C$  such that  $\text{dist}(V_c, W_c) \leq \alpha \cdot m$  and  $\text{dist}(V_{c'}, W_{c'}) \leq \alpha \cdot m$ . Therefore

$$\text{dist}(V_c - V_{c'}, W_c - W_{c'}) \leq 2\alpha \cdot m.$$

The vector  $V_c - V_{c'}$  has the value  $c' - c \neq 0$  in all of its coordinates and so we get that the hamming weight of the codeword  $W_c - W_{c'}$  is at least  $(1 - 2\alpha) \cdot m$ , contradicting the properties of  $C$ .  $\square$

**Lemma 2.4.** *Let  $C$  be an  $[m, n, k]$ -code over  $\mathbb{F}$  such that the dual  $C^\perp$  has minimal distance  $> r$ . Let  $S \subseteq \mathbb{F}^n$  be the set of points corresponding to the rows of the generating matrix of  $C$ . Then  $S$  satisfies condition  $\star_s$  for  $s \geq r/2$ .*

*Proof.* Let  $0 < d \leq r/2$  be an integer and let  $S_d = S + \dots + S$ ,  $d$  times. We will prove the lemma by showing that

$$\{x \in S_d \mid N_d(x) > d\} \subseteq S_{d-2}.$$

Let  $x \in S_d$ . We can thus write

$$x = a_1 + \dots + a_d,$$

with  $a_1, \dots, a_d \in S$ . Suppose now that  $N_d(x) > d$ . This means that there exists a way to write  $x$  as a sum of elements in  $S$  as follows

$$x = b_1 + \dots + b_{d-1} + c,$$

with  $c \notin \{a_1, \dots, a_d\}$ . Combining the two ways to write  $x$  we get

$$a_1 + \dots + a_d - b_1 - \dots - b_{d-1} - c = 0.$$

Since the distance of the dual is larger than  $2d$  the above equality must be a trivial one. That is, all the vectors should cancel in pairs (or have a coefficient that is a multiple of  $|\mathbb{F}|$ ). Since  $c \notin \{a_1, \dots, a_d\}$  we get that  $c$  must be cancelled using the  $b_i$ 's only. This means that  $x$  is in  $S_{d-2}$ .  $\square$

### 3 Constructing a NIS

In this section we prove Theorem 1 and Corollary 1.2. We start with the proof of Corollary 1.2, which is a simple application of known results in coding theory.



*Proof of Corollary 1.2.* To prove the corollary, all we need to do is to construct, for all  $n \in \mathbb{N}$ , an  $(n, 1, \epsilon)$ -NIS  $S_1^{(n)}$  with an efficient interpolation algorithm and  $\epsilon$  which does not depend on  $n$ . The corollary will then follow by applying Theorem 1.

To construct  $S_1$  we take a good family of linear codes  $\{C_n\}$ , where  $C_n$  is an  $[m_n, n, k_n]$ -code over  $\mathbb{F}$  (see Section 2.2) that has an efficient decoding algorithm that can decode a constant fraction of errors and such that the generating matrix of  $C_n$  can be found in polynomial time (such families of codes are known to exist, see [MS77]). Let  $a_1, \dots, a_{m_n} \in \mathbb{F}^n$  be the rows of its generating matrix. We define  $S_1^{(n)}$  to be the list of points  $(a_1, \dots, a_{m_n}, b_1, \dots, b_{m_n})$ , where for each  $j \in [m_n]$  we set  $b_j = 0$ . That is,  $S_1^{(n)}$  contains the rows of the generating matrix of a good code, together with the points 0, taken with multiplicity  $m_n$ . Lemma 2.3 now shows that  $S_1^{(n)}$  satisfies all the required conditions.  $\square$

### 3.1 Proof of Theorem 1

Let  $S_1 = (a_1, \dots, a_m) \in (\mathbb{F}^n)^m$  be an  $(n, 1, \epsilon_1)$ -NIS of size  $|S_1| = m$ . Let  $S_{d-1} = (b_1, \dots, b_r) \in (\mathbb{F}^n)^r$  be an  $(n, d-1, \epsilon_{d-1})$ -NIS of size  $|S_{d-1}| = r$ . Let  $A_{d-1}$  and  $A_1$  be interpolation algorithms for  $S_{d-1}$  and  $S_1$  respectively, as described in Definition 1.1. Let  $S_d = S_{d-1} \boxplus S_1$ . We will prove the theorem by showing that  $S_d$  has an interpolation algorithm that makes at most a polynomial number of calls to  $A_{d-1}$  and to  $A_1$  and can ‘correct’ a fraction

$$\epsilon_d = \frac{\epsilon_1 \cdot \epsilon_{d-1}}{2}$$

of errors. We will describe the algorithm  $A_d$  and prove its correctness simultaneously (the number of calls to  $A_{d-1}$  and  $A_1$  will clearly be polynomial and so we will not count them explicitly).

Fix  $q \in \mathbb{F}_d[x_1, \dots, x_n]$  to be some degree  $d$  polynomial and let  $q_d$  be its homogenous part of degree  $d$ . Let us denote  $S_d = (c_1, \dots, c_{mr})$ , where each  $c_i$  is in  $\mathbb{F}^n$ . We also denote by  $e = (e_1, \dots, e_{mr}) \in \mathbb{F}^{mr}$  the list of ‘errors’, so that  $|\{i \in [mr] \mid e_i \neq 0\}| \leq \epsilon_d \cdot mr$ . The list  $S_d$  can be partitioned in a natural way into all the ‘shifts’ of the list  $S_{d-1}$ . Let us define for each  $i \in [m]$  the list  $T_i = (b_1 + a_i, \dots, b_r + a_i) \in (\mathbb{F}^n)^r$ . We thus have that  $S_d$  is the concatenation of  $T_1, \dots, T_m$  (the order does not matter here). We can also partition the list of errors in a similar way into  $m$  lists  $e^{(1)}, \dots, e^{(m)}$ , each of length  $r$ , such that  $e^{(i)}$  is the list of errors corresponding to the points in  $T_i$ .

We say that an index  $i \in [m]$  is *good* if the fraction of errors in  $T_i$  is at most  $\epsilon_{d-1}/2$ , otherwise we say that  $i$  is *bad*. That is,  $i$  is good if

$$|\{j \in [r] \mid e_j^{(i)} \neq 0\}| \leq (\epsilon_{d-1}/2) \cdot |T_i| = (\epsilon_{d-1}/2) \cdot r.$$

Let  $E = \{i \in [m] \mid i \text{ is bad}\}$ . From the bound on the total number of errors we get that

$$|E| \leq \epsilon_1 \cdot m. \tag{1}$$

**Overview:** The algorithm is divided into three steps. In the first step we look at all pairs  $(T_i, T_j)$  and from each one attempt to reconstruct, using  $A_{d-1}$ , the directional derivative  $\partial_{q_d}(x, a_i - a_j)$ . We will claim that this step gives the correct output for most pairs  $(T_i, T_j)$

(namely, for all pairs in which both  $i$  and  $j$  are good). In the second step we take all the directional derivatives obtained in the previous step (some of them erroneous) and from them reconstruct, using  $A_1$ , the vector of derivatives  $\Delta_{q_d}(x)$  and so also recover  $q_d(x)$ . In the last step of the algorithm we recover the polynomial  $q_{\leq d-1}(x) = q(x) - q_d(x)$ , again using  $A_{d-1}$ . This will give us  $q(x) = q_{\leq d-1}(x) + q_d(x)$ .

**Step I:** Let  $i \neq j \in [m]$  be two good indices as defined above. We will show how to reconstruct  $\partial_{q_d}(x, a_i - a_j)$  from the values in  $T_i, T_j$ . Recall that we have at our disposal two lists of values

$$L_i = \left( q(b_1 + a_i) + e_1^{(i)}, \dots, q(b_r + a_i) + e_r^{(i)} \right)$$

and

$$L_j = \left( q(b_1 + a_j) + e_1^{(j)}, \dots, q(b_r + a_j) + e_r^{(j)} \right).$$

Taking the difference of the two lists we obtain the list

$$L_{ij} = \left( q(b_1 + a_i) - q(b_1 + a_j) + e_1^{(i)} - e_1^{(j)}, \dots, q(b_r + a_i) - q(b_r + a_j) + e_r^{(i)} - e_r^{(j)} \right).$$

Observe that since  $i$  and  $j$  are both good we have that the term  $e_\ell^{(i)} - e_\ell^{(j)}$  is non zero for at most  $\epsilon_{d-1} \cdot r$  values of  $\ell \in [r]$ . Therefore, we can use algorithm  $A_{d-1}$  to recover the degree  $d-1$  polynomial  $Q_{ij}(x) = q(x+a_i) - q(x+a_j)$  from the list  $L_{ij}$ . From Lemma 2.1 we see that throwing away all monomials of degree less than  $d-1$  in  $Q_{ij}$  leaves us with  $\partial_{q_d}(x, a_i - a_j)$ .

We carry out this first step for *all* pairs  $(T_i, T_j)$  and obtain  $\binom{m}{2}$  homogenous polynomials of degree  $d-1$ , let us denote them by  $R_{ij}(x)$ . We know that if  $i$  and  $j$  are both good then

$$R_{ij}(x) = \partial_{q_d}(x, a_i - a_j).$$

If either  $i$  or  $j$  are bad, we do not know anything about  $R_{ij}(x)$  (we can assume w.l.o.g that it is homogenous of degree  $d-1$  since if it is not then clearly the pair is bad and we can modify it to any polynomial we wish, without increasing the total number of errors).

**Step II:** In this step we take the polynomials  $R_{ij}$  obtained in the first step and recover from them the polynomial  $\Delta_{q_d}(x)$  (then, using Lemma 2.2, we will also have  $q_d(x)$ ). We start by giving some notations: The set of degree  $d-1$  monomials is indexed by the set

$$I_{d-1} = \{(\alpha_1, \dots, \alpha_n) \mid \alpha_i \geq 0, \alpha_1 + \dots + \alpha_n = d-1\}.$$

We denote by  $x^\alpha = \prod_i x_i^{\alpha_i}$  and also denote by  $\mathbf{coef}(x^\alpha, h)$  the coefficient of the monomial  $x^\alpha$  in a polynomial  $h(x)$ . Let  $\alpha \in I_{d-1}$  and define the degree 1 polynomial

$$U_\alpha(y_1, \dots, y_n) = \sum_{\ell=1}^n \mathbf{coef} \left( x^\alpha, \frac{\partial q_d}{\partial x_\ell} \right) \cdot y_\ell.$$

Observe that

$$\begin{aligned} \partial_{q_d}(x, a_i - a_j) &= \sum_{\ell=1}^n (a_i - a_j)_\ell \cdot \frac{\partial q_d}{\partial x_\ell}(x) \\ &= \sum_{\alpha \in I_{d-1}} x^\alpha \cdot U_\alpha(a_i - a_j). \end{aligned}$$

Therefore, for each pair  $i, j$  such that both  $i$  and  $j$  are good we can get the (correct) values  $U_\alpha(a_i - a_j)$  for all  $\alpha \in I_{d-1}$  by observing the coefficients of  $R_{ij}$ .

Fix some  $\alpha \in I_{d-1}$ . Using the procedure implied above for all pairs  $i \neq j \in [m]$ , we get  $\binom{m}{2}$  values  $u_{ij} \in \mathbb{F}$  such that if  $i$  and  $j$  are good then

$$u_{ij} = U_\alpha(a_i - a_j).$$

We now show how to recover  $U_\alpha$  from the  $u_{ij}$ 's. Repeating this procedure for all  $\alpha \in I_{d-1}$  will give us  $\Delta_{q_d}(x)$ .

Since we fixed  $\alpha$  let us denote by  $U(y) = U_\alpha(y)$ . We have at our disposal a list of values  $(u_{ij})_{i,j \in [m]}$  such that there exists a set  $E = \{i \in [m] \mid i \text{ is bad}\}$  of size  $|E| \leq \epsilon_1 \cdot m$  such that if  $i$  and  $j$  are not in  $E$  then  $u_{ij} = U(a_i - a_j)$ . Let us partition the list  $(u_{ij})$  according to the index  $j$  into  $m$  disjoint lists:  $B_j = (u_{1j}, \dots, u_{mj})$ . If  $j \notin E$  then the list  $B_j$  contains the values of the degree 1 polynomial  $U_j(y) = U(y - a_j)$  on the set  $S_1$  with at most  $\epsilon_1 \cdot m$  errors (the errors will correspond to indices  $i \in E$ ). Therefore, we can use  $A_1$  in order to reconstruct  $U_j$ , and from it  $U$ . The ‘problem’ is that we do not know which  $j$ 's are good. This problem can be easily solved by applying the above procedure for all  $j \in [m]$  and then taking the majority vote. Since all good  $j$ 's will return the correct  $U(y)$  we will have a clear majority of at least a  $1 - \epsilon_1$  fraction. Combining all of the above gives us the polynomial  $q_d(x)$  and so the second step is complete.

**Step III:** Having recovered  $q_d$  in the last step we now subtract the value  $q_d(c_i)$  from the input list of values (these are the values of  $q(x)$  on  $S_d$ , with  $\leq \epsilon_d$  fraction of errors). This reduces us to the problem of recovering the degree  $d - 1$  polynomial  $q_{\leq d-1}(x) = q(x) - q_d(x)$  from its values in  $S_d$  with a fraction  $\epsilon_d$  of errors. Solving this problem is easy to do by using the algorithm  $A_{d-1}$  on the values in each list  $T_j$  (defined in the beginning of this section) and then taking the majority. Since for good  $j$ 's  $T_j$  contains at most  $\epsilon_{d-1} \cdot r$  errors, and since there are more than half good  $j$ 's, we will get a clear majority and so be able to recover  $q_{\leq d-1}$ .

## 4 Constructing a proper NIS

In this section we prove Theorem 2 and Corollary 1.4. As before we start with the proof of Corollary 1.4 and then go on to prove the theorem.

*Proof of Corollary 1.4.* This will be similar to the proof of Corollary 1.2. All we need to do is find, for each  $n$ , a proper  $(n, 1, \epsilon)$ -NIS that satisfies condition  $\star_d$ . In order to do so we will need a good family of codes  $\{C_n\}$ , such that each  $C_n$  is an  $[m_n, n, k_n]$ -code, which satisfies the following three conditions:

1.  $\{C_n\}$  has an efficient decoding algorithm that can decode a constant fraction  $\alpha = \Omega(1)$  of errors.
2. The minimal distance of the dual  $C_n^\perp$  is larger than  $2 \cdot d$ .
3. Each codeword in  $C_n$  has hamming weight at most  $(1 - 2\alpha) \cdot m_n$ .

Such a family of codes can be constructed explicitly using standard techniques (see Appendix A).

The points in  $S_1$  are given by the rows of the generating matrix of the code  $C_n$ . The rows are all distinct since otherwise we would have a codeword of weight 2 in the dual. From Lemma 2.4 we get that  $S_1$  satisfies condition  $\star_d$  and from item 2. of Lemma 2.3 we get that  $S_1$  is an  $(n, 1, \alpha)$ -proper NIS with an efficient interpolation algorithm.  $\square$

## 4.1 Proof of Theorem 2

The proof of Theorem 2 uses the same arguments as the proof of Theorem 1. The only difference being that the ‘shifts’ of  $S_{d-1}$  (denoted by the  $T_i$ ’s) might intersect. However, using the fact that  $S_1$  satisfies condition  $\star_k$  for  $k \geq d$ , we will be able control the effect of these intersections. To be more precise, the only place where the proof will differ from the proof of Theorem 1 is where we claim that there are not too many ‘bad’ indices  $i \in [m]$  (see below for details). The rest of the proof will be identical and so we will not repeat the parts that we do not have to.

We start by setting the notations for the proof. Let  $S_1 \subseteq \mathbb{F}^n$  be a proper  $(n, 1, \epsilon_1)$ -NIS such that  $|S_1| = m$  and let  $S_{d-1} \subseteq \mathbb{F}^n$  be a proper  $(n, d-1, \epsilon_{d-1})$ -NIS obtained by summing  $S_1$  with itself  $d-1$  times. Let  $A_{d-1}$  and  $A_1$  be interpolation algorithms for  $S_{d-1}$  and  $S_1$  respectively. Let  $S_d = S_{d-1} + S_1$ . As before, the theorem will follow by giving an interpolation algorithm for  $S_d$  that makes at most a polynomial number of calls to  $A_{d-1}$  and to  $A_1$  and can ‘correct’ a fraction

$$\epsilon_d = \frac{\epsilon_{d-1} \cdot \epsilon_1}{3d} \quad (2)$$

of errors (the expression for  $\epsilon_d$  appearing in the theorem can be deduced from this equation by a simple induction).

Fix a degree  $d$  polynomial  $q \in \mathbb{F}_d[x_1, \dots, x_n]$  and let  $q_d$  be its homogenous part of degree  $d$ . Write  $S_1 = \{a_1, \dots, a_m\}$  (the  $a_i$ ’s are distinct). The set  $S_d$  can be divided into  $m$  (possibly overlapping) sets  $T_1, \dots, T_m$  such that

$$T_i = S_{d-1} + a_i.$$

and

$$S_d = \bigcup_{i \in [m]} T_i.$$

Our algorithm is given a list of values of  $q(x)$  on the set  $S_d$  with at most  $\epsilon_d \cdot |S_d|$  errors. Let  $B \subseteq S_d$  be the set of points in  $S_d$  such that the value of  $q(x)$  on these points contains an error. We therefore have

$$|B| \leq \epsilon_d \cdot |S_d| \quad (3)$$

We say that an index  $i \in [m]$  is *bad* if

$$|T_i \cap B| > (\epsilon_{d-1}/2) \cdot |T_i| = (\epsilon_{d-1}/2) \cdot |S_{d-1}|.$$

The next claim bounds the number of bad  $i$ ’s.

**Claim 4.1.** *Let  $E = \{i \in [m] \mid i \text{ is bad}\}$ . Then*

$$|E| \leq \epsilon_1 \cdot m.$$

If we can prove Claim 4.1 then the rest of the proof of Theorem 2 will be identical to the proof of Theorem 1 (starting from the part entitled ‘Step I’), as the reader can easily verify. Indeed Claim 4.1 is the analog of Equation (1) for the case of proper sets. We therefore end this section with a proof of Claim 4.1.

#### 4.1.1 Proof of Claim 4.1:

In order to prove Claim 4.1 we will require the following auxiliary claim on the growth rate of the sumsets of  $S$ .

**Claim 4.2.** *Let  $S \subseteq \mathbb{F}^n$  be of size  $m$  such that  $S$  satisfies condition  $\star_k$  and let the sets  $S_0, S_1, \dots, S_k$  be as in Def 1.3 (the sumsets of  $S$ ). Then there exists a constant  $C_1$  depending only on  $k$  such that if  $|S| > C_1$  then for every  $d \leq k$  we have*

$$|S_d| \geq \frac{1}{2d} \cdot |S_{d-1}| \cdot |S|. \quad (4)$$

*Proof.* The proof is by induction on  $d = 1, 2, \dots, k$ . For  $d = 1$  the bound in (4) is trivial. Suppose the bound holds for  $d - 1$  and let us prove it for  $d$ . Let us denote  $S = \{a_1, \dots, a_m\}$  and  $T_i = S_{d-1} + a_i$ . Clearly  $S_d = \bigcup_{i=1}^m T_i$ . We will lower bound the number of points in  $S_d$  in the following way: For each  $i$  we will take all points  $x \in T_i$  such that  $N_d(x) \leq d$  and then divide the number of points we got by  $d$ . This will give the bound

$$\begin{aligned} |S_d| &\geq \frac{1}{d} \cdot \sum_{i=1}^m (|T_i| - |S_{d-2}|) \\ &= \frac{1}{d} \cdot |S| \cdot |S_{d-1}| - \frac{1}{d} \cdot |S| \cdot |S_{d-2}| \end{aligned}$$

Using the inductive hypothesis on the second term yields

$$\begin{aligned} |S_d| &\geq \frac{1}{d} \cdot |S| \cdot |S_{d-1}| - \frac{2d-2}{d} \cdot |S_{d-1}| \\ &\geq \frac{1}{2d} \cdot |S| \cdot |S_{d-1}|, \end{aligned}$$

where the last inequality holds for sufficiently large  $|S|$  (as a function of  $d$ ).  $\square$

*Proof of Claim 4.1.* Let  $t = |E|$  and suppose in contradiction that  $t > \epsilon_1 \cdot m = \epsilon_1 \cdot |S|$ . This means that there are at least  $t$  sets  $T_{i_1}, \dots, T_{i_t}$  such that  $|B \cap T_{i_j}| \geq (\epsilon_{d-1}/2) \cdot |S_{d-1}|$ . From condition  $\star_d$  we know that, apart from a set of size at most  $|S_{d-2}|$ , each ‘bad’ element appears in at most  $d$  sets  $T_{i_j}$ . Therefore

$$\begin{aligned} \epsilon_d \cdot |S_d| \geq |B| &\geq \frac{1}{d} \cdot \epsilon_1 \cdot |S| \cdot \left( \frac{\epsilon_{d-1}}{2} \cdot |S_{d-1}| - |S_{d-2}| \right) \\ &= \frac{\epsilon_{d-1} \cdot \epsilon_1}{2d} \cdot |S_{d-1}| \cdot |S| - \frac{\epsilon_1}{d} \cdot |S| \cdot |S_{d-2}| \\ &\geq \frac{\epsilon_{d-1} \cdot \epsilon_1}{2d} \cdot |S_d| - \frac{\epsilon_1}{d} \cdot |S| \cdot |S_{d-2}|. \end{aligned}$$

Using Claim 4.2 twice we get that

$$|S| \cdot |S_{d-2}| \leq 2(d-1) \cdot |S_{d-1}| \leq 4d(d-1) \cdot \frac{|S_d|}{|S|}.$$

Plugging this inequality into the previous one we get that

$$\epsilon_d \cdot |S_d| \geq \frac{\epsilon_{d-1} \cdot \epsilon_1}{2d} \cdot |S_d| - \epsilon_1 \cdot 4 \cdot (d-1) \cdot \frac{|S_d|}{|S|},$$

which after division by  $|S_d|$  gives

$$\begin{aligned} \epsilon_d &\geq \frac{\epsilon_{d-1} \cdot \epsilon_1}{2d} - \frac{\epsilon_1 \cdot 4 \cdot (d-1)}{|S|} \\ &> \frac{\epsilon_{d-1} \cdot \epsilon_1}{3d}, \end{aligned}$$

(when  $m = |S|$  is sufficiently large, as a function of  $\epsilon_1$  and  $d$ ) contradicting Eq. 3.  $\square$

## Acknowledgement

We would like to thank Parikshit Gopalan for bringing the problem to our attention, and Emanuele Viola for sharing [Vio07] with us. We also thank Shachar Lovett and Chris Umans for helpful discussions.

## References

- [ALM<sup>+</sup>98] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *JACM*, 45(3):501–555, 1998.
- [AS98] S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of NP. *JACM*, 45(1):70–122, 1998.
- [BFNW93] L. Babai, L. Fortnow, N. Nisan, and A. Wigderson. Bpp has subexponential time simulations unless exptime has publishable proofs. *Complexity Theory*, 3:307–318, 1993.
- [Bog05] A. Bogdanov. Pseudorandom generators for low degree polynomials. In *Proceedings of the 37th annual STOC*, pages 21–30, 2005.
- [BV07] A. Bogdanov and E. Viola. Pseudorandom bits for polynomials. In *Proceedings of the 48th Annual FOCS*, pages 41–51, 2007.
- [HB98] W. C. Huffman and Richard A. Brualdi. *Handbook of Coding Theory*. Elsevier Science Inc., New York, NY, USA, 1998.

- [IW97] R. Impagliazzo and A. Wigderson. P=BPP unless E has subexponential circuits: derandomizing the xor lemma. In *Proceedings of the 29th STOC*, pages 220–229, 1997.
- [Jus72] J. Justesen. A class of constructive asymptotically good algebraic codes. *IEEE Transactions on Information Theory*, 18:652–656, 1972.
- [Lov07] S. Lovett. Unconditional pseudorandom generators for low degree polynomials. *ECCC Report TR07-075*, 2007.
- [MS77] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes, Part II*. North-Holland, 1977.
- [STV01] M. Sudan, L. Trevisan, and S. Vadhan. Pseudorandom generators without the XOR lemma. *J. of Computer and System Sciences*, 62, 2001.
- [SU05] R. Shaltiel and C. Umans. Simple extractors for all min-entropies and a new pseudorandom generator. *JACM*, 52(2):172–216, 2005.
- [TSZS01] A. Ta-Shma, D. Zuckerman, and S. Safra. Extractors from Reed-Muller Codes. In *42nd Annual FOCS*, pages 638–647, 2001.
- [Vio07] E. Viola. The sum of  $d$  small-bias generators fools polynomials of degree  $d$ . Manuscript, 2007.

## Appendix

### A A family of codes with special properties

In the proof of Corollary 1.4 we used a family of good linear error correcting codes  $\{C_n\}$ , over a prime field  $\mathbb{F}$  of size  $p$ , such that  $C_n$  has message length  $n$  and such that the following three properties are satisfied

1. The family  $\{C_n\}$  has an efficient encoding/decoding algorithms from a constant fraction of errors.
2. The dual distance of  $C_n$  is  $\omega(1)$ .
3. If the block length of  $C_n$  is  $m_n$  then every codeword in  $C_n$  has hamming weight at most  $(1 - \delta) \cdot m_n$  for some constant  $\delta > 0$ .

Efficient codes with good dual distance are known to exist over any finite field [HB98]. It is possible that some of these codes already possess property (3) above, however, we did not find a proof for that. Since we only require the dual distance to be super-constant we can give a simple self-contained construction satisfying all three conditions above. We sketch the argument below without giving a complete proof.

Say we wish to construct a code as above for messages of length  $n$ . We pick an integer  $\ell$  such that  $\ell \cdot p^\ell = \theta(n)$  (more accurately, we pick  $\ell$  such that, say,  $\ell \cdot p^\ell$  is at most  $0.9n$  and

at least  $0.8n/p$ . Notice that  $\ell < \log(n)$ . We construct a field  $\mathbb{E}$  of size  $p^\ell$  (this can be done deterministically in time  $\text{poly}(n)$ ). We now take a Reed Solomon code  $R$  over  $\mathbb{E}$ , of block length  $p^\ell$ , whose messages are univariate polynomials of degree  $\alpha \cdot p^\ell$ , for some constant  $\alpha$ . Notice that this code has minimal distance  $(1 - \alpha) \cdot p^\ell$  (over  $\mathbb{E}$ ), relative rate  $\alpha$ , and its dual distance is at least  $\alpha \cdot p^\ell$ . Let  $\varphi : \mathbb{E} \rightarrow \mathbb{F}^\ell$  be the natural representation of  $\mathbb{E}$  as  $\ell$ -tuples over  $\mathbb{F}$ . We can extend  $\varphi$  in the natural way to be a mapping from  $\mathbb{E}^{p^\ell}$  to  $(\mathbb{F}^\ell)^{p^\ell}$ . In particular we can think of  $\varphi(R)$  as a code over  $\mathbb{F}$  of the same relative rate and of at least the same minimal distance. It is not hard to see that there exists an invertible linear transformation  $A : \mathbb{F}^\ell \rightarrow \mathbb{F}^\ell$ , such that  $v = (v_1, \dots, v_{p^\ell}) \in R^\perp$  if and only if<sup>2</sup>  $(A\varphi(v_1), \dots, A\varphi(v_{p^\ell})) \in \varphi(R)^\perp$  (there is a slight abuse of notations here, but the meaning is clear). In particular we get that the minimal distance of  $\varphi(R)^\perp$  is the same as the minimal distance of  $R^\perp$ .

Now, for every  $1 \leq i \leq p^\ell$  we pick a code  $\tilde{C}_i : \mathbb{F}^\ell \rightarrow \mathbb{F}^{m_i}$ , such that  $m_i = O(\ell)$  and such that  $\sum_{i=1}^{p^\ell} m_i = n$ . Moreover, we would like the codes  $\tilde{C}_i$  to satisfy the three properties stated above. It is not hard to play a little with the parameters to get that such  $\tilde{C}_i$ 's and  $m_i$ 's exist. We also note that the  $\tilde{C}_i$ 's can be constructed recursively. We now “concatenate” the code<sup>3</sup>  $\varphi(R)$  with the  $\tilde{C}_i$ 's, such that  $C_i$  is applied on the  $i$ 'th block, of length  $\ell$ , of  $\varphi(R)$ . Denote the resulting code with  $C_n$ . It is now easy to verify that indeed the block length of  $C_n$  is  $n$ , that it has constant rate and linear minimal distance, and that  $C_n^\perp$  has minimal distance  $\omega(1)$ .

---

<sup>2</sup>We would like to have  $\varphi(R)^\perp = \varphi(R^\perp)$ , however this is not true, so we have  $A$  that “fixes” this.

<sup>3</sup>This is not the usual definition of code concatenation, however it is similar in spirit to the construction of Justesen codes [Jus72].