One-Sided Error Testing of Monomials and Affine Subspaces[*]

Oded Goldreich Dana Ron

May 9, 2020

Abstract

We consider the query complexity of three versions of the problem of testing monomials and affine (and linear) subspaces with one-sided error, and obtain the following results:

1. The general problem, in which the arity of the monomial (resp., co-dimension of the subspace) is not specified, has query complexity $\widetilde{O}(1/\epsilon)$, where ϵ denotes the proximity parameter.
2. The bounded problem, in which the arity of the monomial (resp., co-dimension of the subspace) is upper bounded by a fixed parameter, has query complexity $\widetilde{O}(1/\epsilon)$.
3. The exact problem, in which the arity of the monomial (resp., co-dimension of the subspace) is required to equal a fixed parameter (e.g., equals 2), has query complexity $\widetilde{\Omega}(\log n)$, where n denotes the length of the argument for the tested function.

The running time of the testers in the positive results is linear in their query complexity.

[*]This research was partially supported by the Israel Science Foundation (grant No. 1146/18).

Contents

1 Introduction

Property Testing is the study of super-fast (randomized) algorithms for approximate decision making. These algorithms are given direct access to items of a huge data set, and determine whether this data set has some predetermined (global) property or is far from having this property, while accessing a small portion of the data set. Thus, property testing is a relaxation of decision problems and it focuses on algorithms, called *testers*, that only read parts of the input and make decisions which carry a small probability of error.

Consequently, testers are modeled as (randomized) oracle machines and the inputs are modeled as functions to which the tester has an oracle access. In addition, these testers obtain explicit inputs that specify the domain of the input function and a proximity parameter, denoted ϵ. (The tester is required to accept inputs that have the property and reject inputs that are ϵ-far from the property, where both requirements are probabilistic.)

We focus on two main distinctions within the context of property testing. The first is the distinction between size-oblivious testers and more general testers, where size-oblivious means that the complexity of the tester depends only on the proximity parameter. The second distinction is between testers that have one-sided error and testers that have two-sided error, where testers of one-sided error are required to always accept functions that have the property, while rejecting with probability at least 2/3 any function that is ϵ-far from the property (see Definition 2.1).

We consider several testing problems for which size-oblivious (two-sided error) testers are known [10], and study the complexity of one-sided error testers for them. The problems we consider are version of the problems of testing monomials and testing affine spaces.

1.1 Testing monomials

We say that a function $f : \{0,1\}^n \to \{0,1\}$ is a (monotone) k-monomial if there exists a k-set $I \subseteq [n]$ such that $f(x) = \bigwedge_{i \in I} x_i$. In this case, we say that f has arity k. We consider three version of the problem of testing monomials (equiv., sets of monomials).

1. *The general (unbounded) version*: Here we consider the set of monimials of unspecified arity.
2. *The bounded version*: Here, for a fixed parameter k, we consider the set of all momomilas having arity at most k.
3. *The exact version*: Here, for a fixed parameter k, we consider the set of all k-momomilas (i.e., the set of monomials of arity exactly k).

Note that when considering two-sided error testers, the three versions are essentially equivalent. This is the case because, when given oracle access to a function $f : \{0,1\}^n \to \{0,1\}$, we can estimate the density of $f^{-1}(1)$, whereas the density of this set determines the arity of f in case f is a monomial.[1] Such an estimate cannot be obtained without error probability, which means that the foregoing equivalence does not necessary hold in the context of size-oblivious one-sided error testers. This raises the question of whether two-sided error is inherent to size-oblivious testers for the various versions of the problem (cf. [6, Prob. 5.8]). Our first result is that two-sided error is inherent in the case of the exact version.

[1] The exact details are irrelevant at this point, and will become clear later on.

Theorem 1.1 (one-sided error testing of k-monomials cannot be size-oblivious): *For every $k \in [2, (1-o(1)) \cdot n]$, one-sided error testing whether $f : \{0,1\}^n \to \{0,1\}$ is a monomial that depends on exactly k variables requires query complexity $\widetilde{\Omega}(\log n)$.*

Following the initial posting of this work, Nader Bshouty (private comm., May 2020) proved a lower bound of $\Omega(k \log(n/k))$. His result is tight, since exact learning k-monomials can be performed using $k \log n$ queries. We stress that Theorem 1.1 refers to testing whether f is a monomial that depends on *exactly* k variables. In fact, the proof shows that the lower bound applies also to the task of distinguishing k-monomials from $(k-1)$-monomials, when requiring that k-monomials are always accepted (i.e., one-sided error). In contrast, testing whether a function is a monomial that depends on *at most* k variables does admit a size-oblivious one-sided error tester.

Theorem 1.2 (one-sided error testing monomials of size at most k): *For every $k \in [n]$, there exists a $\widetilde{O}(1/\epsilon)$-time one-sided error tester for the set of all functions $f : \{0,1\}^n \to \{0,1\}$ that are monomials that depend on at most k variables, where the time bound refers to a model in which basic operations on n-bit strings can be performed in unit time.*

A simple approach that yields a (typically) weaker complexity bound generalizes the (two-step) dictatorship tester of [10] by first testing that the function is a polynomial of degree at most k, and then applying the conjunction check to a self-corrected version of the function (see details in Section 1.3). This approach yields a complexity bound of $O(1/\epsilon) + \widetilde{O}(2^{3k})$, and relies on the work of [1], which post-dated [10].

Our main approach, which establishes Theorem 1.2, follows the original (two-step) strategy of [10], as implemented in [7]. Specifically, we first test whether the function f describes an $(n-k)$-dimensional affine subspace, and then check whether this affine subspace is of the right form (i.e., is a translation by 1^n of a linear subspace spanned by axis-parallel vectors). This approach also allows us to prove the following result.

Theorem 1.3 (one-sided error testing monomials of unbounded arity): *There exists a $\widetilde{O}(1/\epsilon)$-time one-sided error tester for the set of all functions $f : \{0,1\}^n \to \{0,1\}$ that are monomials, where the time bound refers to a model in which basic operations on n-bit strings can be performed in unit time.*

This result improves over a recent $\exp(1/\epsilon)$-time (one-sided error) tester that was presented by Filmus *et al.* [4] (see [4, Sec. 8]).[2]

1.2 Testing affine and linear subspaces

The aforementioned approach to testing monomials is based on the observation that $f : \{0,1\}^n \to \{0,1\}$ is a k-monomial if and only if $f^{-1}(1)$ is a translation by 1^n of an axis-parallel $(n-k)$-dimensional linear subspace; that is, if $f^{-1}(1) = \{yG + 1^n : y \in \{0,1\}^{n-k}\}$ for some $(n-k)$-by-n full-rank matrix G that has k all-zero columns. Hence, the first step is testing whether $h(x) = f(x) + 1^n$ describes an $(n-k)$-dimensional linear subspace (i.e., $h^{-1}(1)$ is an $(n-k)$-dimensional linear subspace), and the second step is to test whether this linear subspace is axis-parallel. As for monomials, we consider all three versions of this problem; that is, the case that the

[2]We mention that Filmus *et al.* [4] analyze a specific tester (i.e., the "and"-test, which is called conjunction testing in [2, 10]).

co-dimension must equal k, the case that it is at most k, and the case in which it is unspecified (or unbounded).

We first mention that the proof of Theorem 1.1 implies that the lower bound holds for the task of testing whether a function describes an $(n-k)$-dimensional linear subspace (i.e., the exact version). Hence, we turn to the other two versions.

Theorem 1.4 (one-sided error testing of linear subspaces): *There exist $\widetilde{O}(1/\epsilon)$-time one-sided error testers for the following two properties.*

1. *The set of functions that describe linear subspaces.*
2. *For any k, the set of functions that describe linear subspaces of co-dimension at most k.*

As above, the time bound refers to a model in which basic operations on n-bit strings can be performed in unit time.

Theorem 1.4 builds on the randomized reduction of testing $(n-k)$-dimensional linear subspaces to testing the linearity of related functions that range over 2^k+1 possible values. While the original reduction, as presented in [7], incurs a two-sided error, we adapt it to a one-sided error reduction for the cases in Theorem 1.4. In addition, we use this related function when testing that the linear subspace is axis-parallel (i.e., the second step). See details in Section 1.3. (We mention that a version of Theorem 1.4 holds also for testing affine subspaces.)

1.3 Techniques

When allowing two-sided error, all versions of both problems reduce to the exact version problem. Specifically, testing monomials with proximity parameter ϵ reduces to testing monomials that depend on at most $\log_2(1/\epsilon)$ variables with proximity parameter ϵ, and the latter reduces to testing k-monomials for $k \in [\log_2(1/\epsilon)]$. The same holds for testing linear subspaces. The observation underlying all these claims is that a k-monomial evaluates to 1 on a 2^{-k} fraction of the domain (resp., a $(n-k)$-dimensional linear subspace of $\mathcal{F}^n$ has density $|\mathcal{F}|^{-k}$, where $\mathcal{F}$ is any finite field). Needless to say, we can approximate the density of 1-values (up to an additive error of ϵ) by a two-sided error algorithm (that makes $O(1/\epsilon)$ queries). In contrast, this is not possible when using a one-sided error algorithm (since it is required to always answer correctly on inputs of a specified density).

1.3.1 Testing monomials of bounded arity via low degree testing

As hinted above, for the case of $k=1$ (a.k.a dictatorships) Parnas, Ron, and Samorodnitsky [10] presented a one-sided error tester of query-complexity $O(1/\epsilon)$. It operates by first testing whether the function is linear, and then checking whether its self-corrected version depends on a single variable by performing a so-called conjunction test. Here we generalize this approach by first testing whether the function is a polynomial of degree k, and then checking whether its self-corrected version depends on at most k variable by performing the very same conjunction test. It seems that this route was not taken in [10] since such low-degree testers (for functions over GF(2)), let alone self-correctors, were not known at the time.

We obtain a self-corrector of low-degree polynomials over GF(2) by relying on the self-corrector that is implicit in the analysis of the tester of Alon *et al.* [1]. We also use their tester for the aforementioned first step, while setting its proximity parameter to $\epsilon' = \min(\epsilon, 2^{-k-3})$. Under this setting of parameters, their tester's complexity is $O(1/\epsilon') + \widetilde{O}(2^{2k})$.[3] The conjunction test is repeated $O(2^{2k})$ times, while using self-correction of complexity $\widetilde{O}(2^k)$, which works for functions that are 2^{-k-3}-close to having degree k. This explains our setting of ϵ'. Hence, the resulting tester has complexity $O(1/\epsilon) + \widetilde{O}(2^{3k})$. A tester of complexity $\widetilde{O}(1/\epsilon)$ is derived via the connection to linear subspaces as descibed next.

1.3.2 Testing monomials and linear subspaces

We focus on the "unbounded" version of the problems; that is, the versions in which we test monomials (which depends on an arbitrary number of variables) and linear subspaces (with an arbitrary number of dimensions). This refers to Theorem 1.3 and Part 1 of Theorem 1.4, respectively.

We follow the original (two-step) strategy of [10], but build on the implementation of [7]. Specifically, we first test whether the function describes an $(n-k)$-dimensional affine subspace, and then check whether this affine subspace is of the right form (i.e., is a translation by 1^n of a linear subspace spanned by axis-parallel vectors).

Testing affine subspaces. Following [7], we test whether a function $h : \mathcal{F}^n \to \{0,1\}$ describes an $(n-k)$-dimensional linear subspace by defining a related function $g : \mathcal{F}^n \to \mathcal{F}^k \cup \{\perp\}$ and testing whether it is linear, where the symbol $\perp \notin \mathcal{F}^k$ is used only when h does not describe an $(n-k)$-dimensional linear subspace.

Intuitively, when H is an $(n-k)$-dimensional linear subspace and V is a basis of the linear subspace that complements H, the value of $g_{H,V}(x)$ identifies the unique coset in which x resides; that is, $g_{H,V}(x) = c$ if $x + cV \in H$. This definition is applicable also when H is arbitrary provided that $cV \notin H$ for every $c \in \mathcal{F}^k \setminus \{0^k\}$, except that in general the set $\{c \in \mathcal{F}^k : x_c V \in H\}$ may not be a singleton; in such a case, we define $g_{H,V}(x) = \perp$. It turns out that if H is an $(n-k)$-dimensional linear subspace, then $g = g_{H,V}$ is linear (for any V as above); whereas if H is ϵ-far from any $(n-k)$-dimensional linear subspace, then $g = g_{H,V}$ is ϵ-far from linear [7, Clm. 3.4]). This allows to reduce testing $(n-k)$-dimensional linear subspaces to testing linearity of functions, provided we can find a suitable V, which is the main source of the two-sided error in the tester of [7].

Furthermore, the tester of [7] is given the parameter k (i.e., it tests linear subspaces of a specific co-dimension k), whereas our goal is testing linear subspaces of arbitrary dimension. Intuitively, we perform this test in iterations, starting with $k = 1$ and ending at $k = t \stackrel{\text{def}}{=} \log_{|\mathcal{F}|}(2/\epsilon)$, where in each iteration we test whether h describes an $(n-k)$-dimensional linear subspace. A negative answer may leads us to either reject or augment the current basis V by one row, where we reject only when finding evidence that $H = h^{-1}(1)$ is not a linear subspace. We can afford stopping at the end of iteration $k = t$ by relying on the following dichotomy. If h is ϵ-far from any $(n-(t+1))$-dimensional linear subspace and V is a $(t+1)$-by-n full-rank matrix, then we can easily find an x such that $|\{c \in \mathcal{F}^{t+1} : x + cV \in H\}| > 1$ (see Claim 4.5), whereas no such x exists if H is a linear subspace (see Claim 4.4). Hence, if it iteration $k = t$ we obtain a $(t+1)$-by-n matrix V, then we try to find such an x, and reject if and only if it is found.

[3]The more sophisticated analysis of [9], asserting a bound of $O(2^k/\epsilon')$ is not significantly better (if at all) under this setting of parameters.

Testing axis-parallel linear subspace. The function $g = g_{H,V}$ plays a major role also in our testing that $H = h^{-1}(1)$ is an axis-parallel subspace, assuming that H is a linear subspace. Specifically, g is used in order to self-correct h, which is needed when h is only close to describing a linear subspace.

The key observation here that an $(n-k)$-dimensional linear subspace described by h is axis-parallel if and only if there are k pairwise disjoint vectors on which h evaluates to 0, where two vectors $u, w \in \{0,1\}^n$ are called disjoint if $\{i \in [n] : u_i = 1\} \cap \{i \in [n] : w_i = 1\} = \emptyset$. Furthermore, if $H = h^{-1}(1)$ is axis-parallel, then any non-zero linear combination of the foregoing k vectors resides outside of H (see Claim 5.1), and so these vectors constitute a basis, denoted V', of the subspace that complements H.

Hence, we test whether H is axis-parallel by trying to find as many disjoint vectors outside H as possible, check whether all their non-zero linear combinations are outside H, and finally check that the function $g_{H,V'}$ defined by the corresponding basis V' is linear. Note that violation of the first condition yields a witness that H is not axis-parallel, whereas violation of the second condition implies that H is not a linear subspace (although we already verified that it is close to one).

1.3.3 A lower bound on the complexity of testing k-monomials

In contrast to the foregoing results, which refer to testing monomials of unbounded or bounded arity (resp., linear subspaces of unbounded or bounded co-dimension), we show that, for every $k \geq 2$, testing k-monomials (resp., linear subspaces of co-dimension exactly k), has higher complexity, where all results refer to one-sided error testing. The lower bound is proved by showing that no algorithm of query complexity $q = o(\log n / \log\log n)$ can always accept any 2-monomial, and yet reject a random 1-monomial with probability at least 1/2.

We prove the foregoing claim by presenting a random process that interacts with the potential q-query algorithm, and selects a 1-monomial on-the-fly in response to these queries. Specifically, the process proceeds in q iterations, such that in each iteration it answers one query, while limiting the set of consistent 1-monomials (see Construction 6.1.1). This is done such that the outcome of the process (i.e., a uniformly distributed 1-monomial) is independent of the strategy employed by the algorithm (see Claim 6.1.3), The key observation is that as long as this set contains more than one element, the answers provided so far are consistent with some 2-monomial. We show that the latter event occurs with high probability (see Claim 6.1.4), and in this case the algorithm must accept (per the one-sided error requirement).

1.4 More on related work

As mentioned above, the problem of testing whether a Boolean function is a (monotone) k-monomial was first studied by Parnas, Ron, and Samorodnitsky [10]. They provided an $O(1/\epsilon)$-time two-sided tester for this property, and en route a similar result for testing affine and linear subspaces. The tester that they presented generalizes their tester of dictatorship (i.e., the case $k = 1$), and does so by following the same two-step strategy and using similar arguments at each step.

An alternative approach is taken in [7], where the first step of [10] is replaced by a reduction of testing $(n-k)$-dimensional linear subspaces to the testing the linearity of a related function, and the second step is replaced by testing that this function depends on k variables. This yielded a $\widetilde{O}(1/\epsilon)$-time two-sided tester for both properties.

In contrast to the foregoing works that yield two-sided error testers, a $\exp(1/\epsilon)$-time one-sided error tester for monomials was recently provided by Filmus *et al.* [4]. They focus on analyzing the conjuction test (which picks two strings and compares the value of the function on them to the value of the function on the bit-by-bit conjunction of the two strings), and show that is is a proximity oblivious test with rejection probability that is independent of the size of the function (cf. [6, Def. 1.7]): Indeed, while any monomial passes this test with probability 1, any function that is δ-far from any monomial is rejected with probability at least $\exp(-1/\delta)$.

2 Preliminaries

We assume that the reader is familiar with the basic definition of property testing (see, e.g., [6]), but for sake of good order we reproduce it here. The basic definition refers to functions with domain D and range R.

Definition 2.1 (a tester for property Π): *Let Π be a set of functions of the form $f : D \to R$. A* tester for Π *is a probabilistic oracle machine, denoted T, that, on input a* proximity parameter ϵ *and oracle access to a function $f : D \to R$, outputs a binary verdict that satisfies the following two conditions.*

1. T accepts inputs in Π: *For every $\epsilon > 0$, and for every $f \in \Pi$, it holds that $\mathbf{Pr}[T^f(\epsilon)=1] \geq 2/3$.*
2. T rejects inputs that are ϵ-far from Π: *For every $\epsilon > 0$, and for every function $f : D \to R$ that is ϵ-far from Π it holds that $\mathbf{Pr}[T^f(\epsilon)=0] \geq 2/3$, where f is* ϵ-far from Π *if for every $g \in \Pi$ it holds that $|\{x \in D : f(x) \neq g(x)\}| > \epsilon \cdot |D|$.*

If the first condition holds with probability 1 (i.e., $\mathbf{Pr}[T^f(\epsilon) = 1] = 1$)*, then we say that T has* one-sided error*; otherwise, we say that T has* two-sided error.

We focus on the query complexity of such testers, while viewing $|D|$ as an additional parameter. We seek testers of query complexity that is independent of $|D|$, which means that the complexity will be a function of the proximity parameter ϵ only.

The properties we shall consider refer to functions over the domain $\mathcal{F}^n$, where $\mathcal{F}$ is a finite field.

Definition 2.2 (affine subspaces): *For fixed $k, n \in \mathbb{N}$ and a finite field $\mathcal{F}$, we say that the function $f : \mathcal{F}^n \to \{0,1\}$* describes an $(n-k)$-dimensional affine subspace *if $f^{-1}(1) = \{x \in \mathcal{F}^n : f(x)=1\}$ is an $(n-k)$-dimensional affine subspace; that is, $f^{-1}(1) = \{yG + s : y \in \mathcal{F}^{n-k}\}$, where $G \in \mathcal{F}^{(n-k)\times n}$ is an $(n-k)$-by-n full-rank matrix and $s \in \mathcal{F}^n$. When $s = 0^n$, the* described subspace is linear.

We mention that, for $\mathcal{F} = \mathrm{GF}(2)$, the set of k-monomials (see Definition 2.4 below) coincides with the set of functions that describe $(n-k)$-dimensional affine subspaces that are spanned by unit vectors (i.e., the rows of the matrix G are unit vectors).

Definition 2.3 (linear functions): *For fixed $k, n \in \mathbb{N}$ and a finite field $\mathcal{F}$, we say that $g : \mathcal{F}^n \to \mathcal{F}^k$ is* linear *if $g(x+y) = g(x) + g(y)$ for all $x, y \in \mathcal{F}^n$. Equivalently, $g(z) = zT$ for a n-by-k matrix T over $\mathcal{F}$. We say that f is* affine *if $f(z) = f'(z) + s$ for a linear function f' and some $s \in \mathcal{F}^k$.*

When $k = 1$ and $\mathcal{F} = \mathrm{GF}(2) \equiv \{0,1\}$, it holds that $f : \mathcal{F}^n \to \{0,1\}$ describes an $(n-k)$-dimensional affine subspace (resp., linear subspace) if and only if f is a non-constant affine function (resp., $f+1$ is a non-constant linear function). However, in all other cases, this does not hold; in particular, for other fields a non-constant affine function must range over $\mathcal{F}$ rather than over $\{0,1\}$, whereas for $\mathcal{F} = \mathrm{GF}(2)$ and $k > 1$ the densities do not match (i.e., an $(n-k)$-dimensional affine subspace over GF(2) has density 2^{-k}, but $f^{-1}(1)$ has density $1/2$ for any non-constant affine function $f : \mathrm{GF}(2)^n \to \mathrm{GF}(2)$).

Definition 2.4 (monomial and monotone monomial): *A Boolean function $f : \{0,1\}^n \to \{0,1\}$ is called a k-monomial if for some k-subset $I \subseteq [n]$ and $\sigma = \sigma_1 \cdots \sigma_n \in \{0,1\}^n$ it holds $f(x) = \wedge_{i \in I}(x_i \oplus \sigma_i)$. It is called a monotone k-monomial if $\sigma = 0^n$.*

Indeed, the definitions of (general and monotone) 1-monomials coincide with the notions of (general and monotone) dictatorships. We focus on the task of testing monotone k-monomials, while recalling that the task of testing k-monomials is reducible to it (see [10] or [6, Sec. 5.2.2.1]).[4] (A similar reduction holds when k is not specified.)

Conventions. When writing $\mathbf{Pr}_x[\mathrm{event}(x)]$ we refer to the case that x is selected uniformly in a set that is clear from the context; we sometimes spell out this set by writing $\mathbf{Pr}_{x \in S}[\mathrm{event}(x)]$. For sake of simplicity, we often use the phrase "with high probability" (abbrev., "w.h.p."), which mean that we can obtain arbitrary high constant probability smaller 1 (e.g., 0.99). The image of a function $f : D \to R$ is the set $\{f(e) : e \in D\} \subseteq R$. The symbol $\perp$ denotes a special symbol that is not in $\mathcal{F}^k$.

We view $\mathcal{F}$ and n as parameters, and when using O-notation we refer to universal constants that are independent of $\mathcal{F}$ and n. However, when stating time-complexity bounds, we shall assume that basic operations on elements of $\mathcal{F}^n$ (e.g., addition, selection of a random element, etc) can be performed at unit cost. For simplicity, we present all algorithms for fixed $\mathcal{F}$ and n, although all algorithms can actually take these parameters as inputs.

3 Testing Monomials via Low Degree Testing

In this section we describe a one-sided error testing algorithm for the property of being a (monotone) monomial of arity *at most* k, where k is a given integer parameter. This algorithm generalizes the algorithm presented in [10] for the case of $k = 1$ (singelton/dictatorship functions). We start with a short high-level description, and then give the precise details.

For fixed n, we consider Boolean fuinctions over n variable. Let $\mathcal{M}_{\le k}$ denote the class of monotone monomials with at most k variables, and $\mathcal{P}_{\le k}$ denote the class of polynomials of degree at most k over GF(2); indeed, $\mathcal{M}_{\le k} \subset \mathcal{P}_{\le k}$, since we associate $\{0,1\}$ with GF(2) (and $\wedge$ with multipolication). Given $f : \{0,1\}^n \to \{0,1\}$, we shall test whether $f \in \mathcal{M}_{\le k}$ in two steps. First, we test whether $f \in \mathcal{P}_{\le k}$, by using the one-sided error tester of [1], and reject if this tester rejects. Otherwise, we may assume that f is very close to a degree k polynomial, and in that case we can self-correct f by using the self-correction procedure that is implicit in the analysis of the tester of [1]. In this case, we invoke the conjunction test of [10]. That is, denoting the self-corrected

[4]The reduction in [6, Sec. 5.2.2.1] has two-sided error, since it rejects in case it does not find an element in $f^{-1}(1)$. However, accepting in such an unlikely case, we derive a one-sided error reduction.

version of f by g, the second step essentially consists of selecting uniformly $x, y \in \{0,1\}^n$ and checking whether $g(x) \wedge g(y) = g(x \wedge y)$, where $z = x \wedge y$ is the bit-by-bit conjunction of x and y (i.e., $z_i = x_i \wedge y_i$ for each $i \in [n]$).

3.1 Background

In light of the foregoing, we start by citing two results of [1].

Theorem 3.1 ([1, Thm. 1]): *For every k, there exists a one-sided error tester for $\mathcal{P}_{\leq k}$ of time and query complexity $O(1/\epsilon + k2^{2k})$.*

Lemma 3.2 (implicit in the proof of [1, Thm. 1]): *For every k, there exists a procedure* SelfCorrectPoly *that given $x \in \{0,1\}^n$ and oracle access to a function $f : \{0,1\}^n \to \{0,1\}$ makes $O(2^k)$ queries and returns a value* $\mathsf{SelfCorrectPoly}^f(x)$ *that satisfies the following conditions.*

1. *If f is 2^{-k-3}-close to $g \in \mathcal{P}_{\leq k}$, then $\mathsf{SelfCorrectPoly}^f(x) = g(x)$ with probability at least $2/3$.*
2. *If $f \in \mathcal{P}_{\leq k}$, then $\mathsf{SelfCorrectPoly}^f(x) = f(x)$ always holds.*

A proof of Lemma 3.2 that only relies on explicit results of [1] can be found in Appendix A.2. We shall also make use of the following fact.

Claim 3.3 (analysis of the conjunction test): *Let $g : \{0,1\}^n \to \{0,1\}$ be a function that belongs to $(\mathcal{P}_{\leq k} \setminus \mathcal{M}_{\leq k})$ for some $k \geq 1$. Then, $\mathbf{Pr}[g(x \wedge y) = g(x) \wedge g(y)] \leq 1 - 2^{-2k}$.*

We comment that the inequality in Claim 3.3 is tight; specifically, the function $g(z) = z_1 \cdots z_k + z_2 \cdots z_{k+1}$ is accepted with probability at least $1 - 2^{-2k+2}$.

Proof: It is instructive to move to the arithmetics of GF(2), and consider the equation

$$g(x_1y_1, ..., x_ny_n) = g(x_1, ..., x_n) \cdot g(y_1, ..., y_n),$$

where $x_1, ..., x_n, y_1, ..., y_n \in \mathrm{GF}(2)$. In this case g has the form $\sum_{i \in [t]} M_i$ such that $t \geq 2$ and the M_i's are products of at most t variables. Hence, the equation refers to the equality of two distinct polynomials that are each of degree $2k$, and the claim follows by the Schwartz-Zippel Lemma for binary fields (see, e.g., [6, Exer. 5.1]). ■

3.2 The actual tester

On input parameters k and ϵ, and query access to a function f, the testing algorithm, denoted TestMon, proceeds as follows:

1. Run the tester of Theorem 3.1 on f with parameters k and $\epsilon' = \min(\epsilon, 2^{-k-3})$. If this tester rejects, then TestMon rejects (and terminates).
2. Repeat the following conjunction test (at most) $O(2^{2k})$ times.
   (a) Select $x, y \in \{0,1\}^n$, uniformly, independently at random.

(b) Let $b_x \leftarrow \textsf{SelfCorrectPoly}^f_{O(k)}(x)$, $b_y \leftarrow \textsf{SelfCorrectPoly}^f_{O(k)}(y)$ and $b_{x\wedge y} \leftarrow \textsf{SelfCorrectPoly}^f_{O(k)}(x\wedge y)$, where $\textsf{SelfCorrectPoly}^f_t(z)$ denotes invoking $\textsf{SelfCorrectPoly}^f(z)$ for t times and taking a majority vote.

(c) If $b_{x\wedge y} \neq b_x \wedge b_y$, then output $\textsf{reject}$.

3. Output $\textsf{accept}$.

The query complexity of $\textsf{TestMon}$ is $O(1/\epsilon' + k2^k) + O(2^{2k}) \cdot O(k \cdot 2^k) = O(1/\epsilon + k2^{3k})$.

Theorem 3.4 *For any given integer parameter k, $\textsf{TestMon}$ is a one-sided error tester for $\mathcal{M}_{\leq k}$.*

Proof: Consider first the case that $f \in \mathcal{M}_{\leq k}$. Since $\mathcal{M}_{\leq k} \subset \mathcal{P}_{\leq k}$, and the tester of Theorem 3.1 has one-sided error, f passes the first step of the algorithm with probability 1. By Part 2 of Lemma 3.2, f also passes the second step with probability 1, and is hence accepted.

We thus turn to the case that f is ϵ-far from $\mathcal{M}_{\leq k}$. If f is ϵ'-far from $\mathcal{P}_{\leq k}$, then it is rejected with probability at least 2/3 in Step 1. Hence, assume from this point on that f is ϵ'-close to $\mathcal{P}_{\leq k}$, and so it satisfies the hypothesis of Part 1 of Lemma 3.2; that is, letting $g \in \mathcal{P}_{\leq k}$ be ϵ'-close to f, it follows that $\mathbf{Pr}[\textsf{SelfCorrectPoly}^f(x) = g(x)] \geq 2/3$ for every $x \in \{0,1\}^n$. Since f is ϵ-far from $\mathcal{M}_{\leq k}$ and $\epsilon \geq \epsilon'$, it follows that $g \in (\mathcal{P}_{\leq k} \setminus \mathcal{M}_{\leq k})$. Combining Part 1 of Lemma 3.2 with Claim 3.3, we infer that in each invocation of the conjunction test, with probability at least $2^{-2k} - 3 \cdot 2^{-2k-3} > 2^{-2k+1}$, over the choice of x, y and the coin tosses of $\textsf{SelfCorrectPoly}_{O(k)}$, all of the following events occur

1. $g(x) \wedge g(y) \neq g(x \wedge t)$,
2. $b_x = g^f(x)$, $b_y = g^f(y)$, and $b_{x\wedge y} = g^f(x \wedge y)$.

Observing that in such a case $\textsf{TestMon}$ rejects, and that this test is invoked $O(2^{2k})$ times, the claim follows. ■

4 Testing Affine and Linear Subspaces

We start by restating the problem. For a finite field $\mathcal{F}$ and a natural number n, we are given access to a function $h : \mathcal{F}^n \to \{0,1\}$ and wish to test whether $h^{-1}(1)$ is an affine subspace. We do so by reducing this problem to testing linearity (of a function that is related to h). Actually, we present two reductions: The first (and simpler) reduction increases the complexities by a factor of $O(1/\epsilon)$, whereas the second reduction only incurs an overhead of $\widetilde{O}(\log(1/\epsilon))$. The first reduction (presented in Section 4.1) will be used as a subroutine in the second reduction (presented in Section 4.2).

Both reductions are obtained by modifications of the reductions in [7, Sec. 3]. Recall that, unlike in [7], here we are not given the claimed dimension of the subspace nor do we need to verify it. On the other hand, we are required to always accept yes-instances (i.e., have one-sided error).

As in [7], we simplify the presentation by first reducing the problem of testing affine subspaces to testing affine subspaces. The following reduction is different from the one presented in [7, Clm. 3.2], which has a two-sided error proabability.

Claim 4.1 (reducing to the linear case): *Testing whether a function $h : \mathcal{F}^n \to \{0,1\}$ describes an affine subspace can be randomly reduced to testing whether a function $h' : \mathcal{F}^n \to \{0,1\}$ describes a linear subspace, where the reduction introduces an additive overhead of $O(1/\epsilon)$ queries.*

Proof: On input parameter $\epsilon > 0$ and oracle access to h, we proceed as follows.

1. We select uniformly a sample of $O(1/\epsilon)$ points in $\mathcal{F}^n$. If h evaluates to 0 on all these points, then we accept. Otherwise, let u be a point in this sample such that $h(u) = 1$.

2. We invoke the tester for linear subspaces on input parameter ϵ and oracle access to h' defined by $h'(x) \stackrel{\rm def}{=} h(x+u)$, and output its verdict. That is, each query x to h' is emulated by making the query $x+u$ to h.

The overhead of the reduction is due to Step 1, whereas in Step 2 we just invoke the tester for the special case.

If h describes an affine subspace, then either Step 1 finds $u \in h^{-1}(1)$ or it accepts. In the former case, the function h' (i.e., $h'(x) = h(x+u)$) describes the linear subspace $H' \stackrel{\rm def}{=} h^{-1}(1) - u$. To see that H' is a linear subspace, note that $h^{-1}(1) - u = h^{-1}(1) - h^{-1}(1)$ (i.e., $\{x - u : x \in h^{-1}(1)\} = \{x - y : x, y \in h^{-1}(1)\}$) and that $H - H$ is a linear subspace if H is an affine subspace.[5]

On the other hand, if h is ϵ-far from being an affine subspace, then it must be that $|h^{-1}(1)| > \epsilon \cdot |\mathcal{F}|^n$ (or else h is ϵ-close to describing the empty linear subspace). Hence, with high probability, Step 1 finds some $u \in h^{-1}(1)$. But in this case h' (which was defined by $h'(x) \stackrel{\rm def}{=} h(x+u)$) must be ϵ-far from describing a linear subspace.[6] Hence, in this case (i.e., h' is ϵ-far from describing a linear subspace), Step 2 rejects with high probability. ■

4.1 The first reduction

We follow the strategy of [7, Sec. 3.2], except that we refrain from rejection based on statistical evidence. Needless to say, this complicates the algorithms and their analysis.

4.1.1 Elements taken from [7]

The pivotal step in the reduction is the definition of a function $g : \mathcal{F}^n \to \mathcal{F}^k \cup \{\perp\}$ such that if $H \stackrel{\rm def}{=} h^{-1}(1)$ is an $(n-k)$-dimensional linear subspace, then g is linear (with image $\mathcal{F}^k$) and $g^{-1}(0^k) = H$. Furthermore, in this case, $g(x)$ indicates one of the $|\mathcal{F}|^k$ translations of H in which x resides; that is, if $v^{(1)}, ..., v^{(k)} \in \mathcal{F}^n$ form a basis for the k-dimensional subspace that complements H, then $g(x)$ represents coefficients $(c_1, ..., c_k) \in \mathcal{F}^k$ such that $x \in H - \sum_{i\in[k]} c_i v^{(i)}$.

Indeed, the definition of g is based on any fixed sequence of linearly independent vectors $v^{(1)}, ..., v^{(k)} \in \mathcal{F}^n$ such that for every non-zero sequence of coefficients $(c_1, ..., c_k) \in \mathcal{F}^k$ it holds that $\sum_{i\in[k]} c_i v^{(i)} \notin H$. Such sequences of vectors exist if H is an $(n-k)$-dimensional linear subspace. (The issue of finding such sequences of vectors will be dealt with later.)

Fixing such a sequence of $v^{(i)}$'s, we define $g : \mathcal{F}^n \to \mathcal{F}^k \cup \{\perp\}$ as follows. For each $x \in \mathcal{F}^n$, if $(c_1, ..., c_k) \in \mathcal{F}^k$ is the *unique* sequence that satisfies $x + \sum_{i\in[k]} c_i v^{(i)} \in H$ then $g(x) = (c_1, ..., c_k)$ and $g(x) = \perp \notin \mathcal{F}^k$ otherwise. Indeed, a unique sequence $(c_1, ..., c_k) \in \mathcal{F}^k$ exists for each $x \in \mathcal{F}^n$

[5]Indeed, if $H = \{yG+s : y\in\mathcal{F}^{n-k}\}$ is an affine subspace, then $H - H = \{x - x' : x, x' \in \mathcal{F}^n\} = \{yG+s-(y'G+s) : y, y' \in \mathcal{F}^{n-k}\} = \{(y-y')G : y \in \mathcal{F}^{n-k}\}$ is a linear subspace. We note that the opposite direction holds as well: if $H' = \{yG : y\in\mathcal{F}^{n-k}\}$ is a linear subspace, then $H' + u = \{yG + u : y\in\mathcal{F}\}$ is an affine subspace.

[6]This is so because if h' is ϵ-close to g' that describes an $(n-k)$-dimensional linear subspace (i.e., g' describes the linear subspace $\{yG : y\in\mathcal{F}^{n-k}\}$), then $g(x) \stackrel{\rm def}{=} g'(x-u)$ (equiv., $g(x+u) = g'(x)$) describes an affine subspace (i.e., g describes the affine subspace $\{yG + u : y\in\mathcal{F}^{n-k}\}$), whereas h is ϵ-close to g (since $h(x) = h'(x-u)$).

if H is an $(n-k)$-dimensional linear subspace, and in that case $g(x) \in \mathcal{F}^k$ for every $x \in \mathcal{F}^n$. But when H is not an $(n-k)$-dimensional linear subspace, it may happen that for some (or even all) x's there is no sequence $(c_1, ..., c_k) \in \mathcal{F}^k$ such that $x + \sum_{i\in[k]} c_i v^{(i)} \in H$; similarly, it may happen that there are several different sequences $(c_1, ..., c_k) \in \mathcal{F}^k$ that satisfy $x + \sum_{i\in[k]} c_i v^{(i)} \in H$. In these cases, g assumes the value $\perp$ on some inputs, and consequently it is not linear. In any case, using matrix notation, we restate the foregoing definition next (where the $v^{(i)}$'s are the rows of the matrix V).

Definition 4.2 (the function $g = g_{H,V}$): *Let $H \subseteq \mathcal{F}^n$, and V be a k-by-n full-rank matrix over $\mathcal{F}$ such that $cV \notin H$ for every $c \in \mathcal{F}^k \setminus \{0^k\}$. Then, $g_{H,V} : \mathcal{F}^n \to \mathcal{F}^k \cup \{\perp\}$ is defined such that $g_{H,V}(x) = c$ if $c \in \mathcal{F}^k$ is the unique vector that satisfies $x+cV \in H$, and $g_{H,V}(x) = \perp$ if the number of such k-long vectors is not one.*

Note that $g(x) = c$ implies $x + cV \in H$. Hence, in particular, $g_{H,V}(x) = 0^k$ implies $x \in H$; that is, $g_{H,V}^{-1}(0^k) \subseteq H$. The following result of [7] is pivotal to reducing testing linear subspace to tresting the linearity of functions.

Claim 4.3 (H versus $g_{V,H}$, [7, Clm. 3.4]): *Let H, V and $g = g_{H,V}$ be as in Definition 4.2. Then, H is an $(n-k)$-dimensional linear subspace if and only if g is a linear function with image $\mathcal{F}^k$.*

Given the importance of this result for the current work, we reproduce its proof in Appendix A.1. We stress that whenever we say that g is linear, it holds, in particular, that it never assumes the value $\perp$. (Indeed, when emulating g for the linearity tester, we shall suspend the execution if we ever encounter the value $\perp$.)[7] Note that if g is ϵ-close to being a linear function with image $|\mathcal{F}|^k$, then $g^{-1}(0^k)$ is ϵ-close to being an $(n-k)$-dimensional linear subspace (i.e., the indicator functions of these sets are ϵ-close).[8]

4.1.2 Additional observations

Recall that V is a full-rank k-by-n matrix such that $cV \notin H$ for every $c \in \mathcal{F}^k \setminus \{0^k\}$. While Claim 4.3 suffices for the analysis of the tester of $(n-k)$-dimensional linear subspaces presented in [7], it does not suffice for the current context. Specifically, Claim 4.3 does not tell us what happens when H is an $(n-k')$-dimensional linear subspace for $k' \neq k$. Note that $k' < k$ is not possible, because in that case the linear subspace that complements H cannot contain $k > k'$ independent vectors (as posulated in the definition of V). Yet, $k' > k$ is possible, and in this case $H \cup V$ spans a linear subspace of dimension $(n-k') + k < n$. We observe that in this case (i.e., when H is a linear subspace of dimension smaller than $n-k$), the function $g_{H,V}$ assumes the value $\perp$ on all vectors that are not spanned by the vectors in $H \cup V$, but otherwise $g_{H,V}$ satisfies the linearity condition.

Claim 4.4 (more on $g_{V,H}$ when H is a linear subspace): *Let H, V and $g = g_{H,V}$ be as in Definition 4.2. Suppose that H is a linear subspace. Then:*

[7]Unlike in [7], we shall distinguish between the case that $g(x) = \perp$ due to the non-existence of $c \in \mathcal{F}^k$ such that $x + cV \in H$ and the existence of several distinct c's. In the latter case, we shall reject, but in the former case we shall augment V by x and try again.

[8]To see this, consider a linear g' (with image $\mathcal{F}^k$) that is ϵ-close to g, and note that the $(n-k)$-dimensional linear space $H' = \{x \in \mathcal{F}^n : g'(x) = 0^k\}$ is ϵ-close to $g^{-1}(0^k)$, since $g'(x) \neq g(x)$ holds for any x that resides in the symmetric difference of these sets.

1. *For every $x \in \mathcal{F}^n$, if $g(x) = \perp$, then, for every $c \in \mathcal{F}^k$, it holds that $x + cV \notin H$.*

2. *For every $x, y \in \mathcal{F}^n$, if $g(x), g(y)$ and $g(x+y)$ are all in $\mathcal{F}^k$, then $g(x+y) = g(x) + g(y)$.*

Hence, an x such that $x + cV \in H$ for several distinct c's constitute a witness that H is not a linear subspace. Likewise, a pair (x, y) such that $g(x), g(y), g(x+y) \in \mathcal{F}^k$ and $g(x+y) \neq g(x) + g(y)$ constitutes a witness that H is not a linear subspace.

Proof: For Part 1, suppose that for $c, c' \in \mathcal{F}^k$ it holds that $x + cV \in H$ and $x + c'V \in H$. Then, using the linearity of H, it follows that $(x + cV) - (x + c'V) \in H$, which implies that $(c - c')V \in H$. But then, by the hypothesis regarding V, it must be that $c = c'$.

Turning to Part 2, suppose that $g(x), g(y)$ and $g(x+y)$ are all in $\mathcal{F}^k$. Then, $x + g(x)V$, $y + g(y)V$ and $(x+y) + g(x+y)V$ are all in H. Using the linearity of H, we have $(x + g(x)V) + (y + g(y)V) \in H$, which implies that $(x+y) + (g(x) + g(y))V \in H$. Using the fact that $g(x+y) \neq \perp$, which implies the uniqueness of c such that $(x+y) + cV \in H$, it must be the case that $g(x) + g(y) = g(x+y)$. ■

Claim 4.5 (more on $g_{V,H}$ when H is not a linear subspace): *Let H, V and $g = g_{H,V}$ be as in Definition 4.2. Suppose that $k > \log_{|\mathcal{F}|}(1/\rho)$, where $\rho \stackrel{\rm def}{=} |H|/|\mathcal{F}|^n$. Then, for at least a $\rho - |\mathcal{F}|^{-k}$ fraction of the points $x \in \mathcal{F}^n$ there exist several distinct $c \in \mathcal{F}^k$ such that $x + cV \in H$. Furthermore, letting $m \stackrel{\rm def}{=} \lceil k \cdot \log_2 |\mathcal{F}| \rceil$, there exists $i \in [m]$ such that for at least a $2^{m-i-1} \cdot (\rho - |\mathcal{F}|^{-k})/m$ fraction of the points $x \in \mathcal{F}^n$, there are at least $2^{i-1} + 1$ distinct $c \in \mathcal{F}^k$ such that $x + cV \in H$.*

The title of Claim 4.5 is justified by the fact that the premise (i.e., $k > \log_{|\mathcal{F}|}(1/\rho)$) cannot hold when H is a linear subspace (since if H is a linear subspace of dimension at most $n - k$, then its density must be at most $|\mathcal{F}|^{-k}$). Furthermore, Claim 4.5 suggests a procedure for detecting that H is not a linear subspace (in the case that $k > \log_{|\mathcal{F}|}(|\mathcal{F}|^n/|H|)$).

Proof: Consider a partition of $\mathcal{F}^n$ into $|\mathcal{F}|^{n-k}$ equivalence classes such that x and y are in the same class if and only if $x - y$ is spanned by the rows of V; that is, x resides in the class $\{x + cV : c \in \mathcal{F}^k\}$. Letting q denote the number of classes that contain more than one element of H, it follows that $|H| \leq q \cdot |\mathcal{F}|^k + (|\mathcal{F}|^{n-k} - q) \cdot 1$. Hence, $\frac{q}{|\mathcal{F}|^{n-k}} \geq \rho - |\mathcal{F}|^{-k}$, which establishes the main claim.[9]

To prove the furthermore claim, let q_i denote the number of classes that contain between $2^{i-1}+1$ and 2^i elements of H. Then, $|H| \leq |\mathcal{F}|^{n-k} + \sum_{i \in [m]} q_i \cdot 2^i$, since the size of each class is at most $|\mathcal{F}|^k \leq 2^m$. It follows that there exists $i \in [m]$ such that

$$\begin{aligned} \frac{q_i}{|\mathcal{F}|^{n-k}} &\geq \frac{|H| - |\mathcal{F}|^{n-k})}{m \cdot |\mathcal{F}|^{n-k} \cdot 2^i} \\ &= |\mathcal{F}|^k \cdot \frac{\rho - |\mathcal{F}|^{-k})}{m \cdot \cdot 2^i} \\ &\geq 2^{m-i-1} \frac{\rho - |\mathcal{F}|^{-k})}{m} \end{aligned}$$

[9]This is because $q \geq (|H| - |\mathcal{F}|^{n-k})/|\mathcal{F}|^k$ implies

$$\frac{q}{|\mathcal{F}|^{n-k}} \geq \frac{|H| - |\mathcal{F}|^{n-k}}{|\mathcal{F}|^n} = \rho - |\mathcal{F}|^{-k}.$$

which establishes the claim. ■

The analysis is simplified by using the following claim, which has also appeared in a recent version of [7] (see [7, Clm. 3.6]).[10]

Claim 4.6 (on the linear function closest to $g_{V,H}$): *Let H, V and $g = g_{H,V}$ be as in Definition 4.2. If g is 0.499-close to a linear function, then this linear function has image $\mathcal{F}^k$.*

The constant 0.499 can be replaced by any quantity that is smaller than $1 - |\mathcal{F}|^{-1} \geq 1/2$.

Proof: As in the proof of Claim 4.5, we consider a partition of $\mathcal{F}^n$ into equivalence classes such that $x \in \mathcal{F}^n$ resides in the class $C_x \stackrel{\rm def}{=} \{x + cV : c \in \mathcal{F}^k\}$, where, indeed, $C_x = C_{x+c'V}$ for every $c' \in \mathcal{F}^k$. A class is considered good if it contains a single element of H, which happens if and only if $g(x) \in \mathcal{F}^k$. The key observation is that if C_x is good (equiv., $g(x) \in \mathcal{F}^k$), then, for every $c \in \mathcal{F}^k$, it holds that $g(x + cV) = g(x) - c$, since $C_x \cap H = \{x + g(x)V\} = \{(x + cV) + (g(x) - c)V\}$. Now, let $f : \mathcal{F}^n \to \mathcal{F}^k$ be an arbitrary linear function that has an image that is partial to $\mathcal{F}^k$, and note that this image has size at most $|\mathcal{F}|^{k-1}$ (since the image of f must be a linear subspace). Noting that $g(x) = f(x)$ implies $g(x) \in \mathcal{F}^k$, we get

$$\begin{aligned}
\mathbf{Pr}_{x\in\mathcal{F}^n}[g(x)=f(x)] &= \mathbf{Pr}_{x\in\mathcal{F}^n,c\in\mathcal{F}^k}[g(x+cV)\in\mathcal{F}^k \ \&\ g(x+cV)=f(x+cV)] \\
&= \mathbf{Pr}_{x\in\mathcal{F}^n,c\in\mathcal{F}^k}[g(x)\in\mathcal{F}^k \ \&\ g(x+cV)=f(x+cV)] \\
&\leq \max_{x\in\mathcal{F}^n:g(x)\in\mathcal{F}^k} \{\mathbf{Pr}_{c\in\mathcal{F}^k}[g(x+cV)=f(x+cV)]\} \\
&\leq |\mathcal{F}|^{-1}.
\end{aligned}$$

To verify the last inequality note that for any $x \in \mathcal{F}^n$ such that $g(x) \in \mathcal{F}^k$ and a uniformly distributed $c \in \mathcal{F}^k$ it holds that $g(x+cV) = g(x) - c$ is uniformly distributed over $\mathcal{F}^k$, whereas the image of f contains at most $|\mathcal{F}|^{k-1}$ elements. It follows that g is at distance at least $1-|\mathcal{F}|^{-1} \geq 1/2$ from any linear function that has image that is partial to $\mathcal{F}^k$. ■

4.1.3 The main procedure

The main procedure will be invoked iteratively with increasing values of k, starting with $k = 1$, and ending with $k = \log_{|\mathcal{F}|}(1/\epsilon) + O(1)$ if not earlier. In each invocation, we provide the procedure with a k-by-n full-rank matrix V such that $cV \notin H$ for all $c \in \mathcal{F}^k \setminus \{0^k\}$. Each invocation yields one of the following three results.

1. *The invocation accepts.* This always happens when H is is an $(n - k)$-dimensional linear subspace, but may also happen otherwise.
2. *The invocation rejects.* This is always based on a witness for non-linearity of H.
3. The invocation returns a full-rank $(k + 1)$-by-n matrix V' that augments V with a single row such that $c'V' \notin H$ for all $c' \in \mathcal{F}^{k+1} \setminus \{0^{k+1}\}$.

[10]This claim was overlooked in earlier versions of [7], leading to an unnecessary complication of the algorithm and its analysis.

The following Algorithm 4.7 is titled "quasi-tester for $(n-k)$-dimensional linear subspaces" because it is not guaranteed to reject no-instances with high probability. We shall only show (see Proposition 4.8) that, with high probability, no-instances are either rejected or cause the algorithm to return a matrix that extends V.

We shall assume, without loss of generality that $0^n \in H$, since we can check this condition and reject if it is not satisfied. (In Section 4.1.4 we set the parameter ϵ' to ϵ, but in Section 4.2 we shall let ϵ' be a small positive constant.)

Algorithm 4.7 (quasi-testing whether H is an $(n-k)$-dimensional linear subspace): *On input a full-rank k-by-n matrix V, a proximity parameter $\epsilon' \in (0, 0.1]$, and oracle access to $h : \mathcal{F}^n \to \{0,1\}$, specifying $H = h^{-1}(1)$, we test whether the function $g = g_{H,V}$ is linear by invoking a linearity tester and emulating g by making queries to h. Specifically, we invoke a linearity test with proximity parameter ϵ', while providing it with oracle access to the function $g = g_{H,V}$, where $g_{H,V}$ is as in Definition 4.2. When the linearity tester queries g at x, we query h on $x + cV$ for all $c \in \mathcal{F}^k$, and answer accordingly; that is, the answer is c if c is the unique vector satisfying $h(x + cV) = 1$, and otherwise* (i.e., $g(x) = \perp$) *the execution is suspended. In the latter case, we distinguish two cases regarding $G(x) \stackrel{\text{def}}{=} \{c \in \mathcal{F}^k : x + cV \in H\}$.*

1. *If $|G(x)| \geq 2$, then we reject.*

   (Recall that by Part 1 of Claim 4.4 distinct $c_1, c_2 \in \mathcal{F}^k$ such that $x + c_iV \in H$ for both $i \in \{1,2\}$ constitute a witness that H is not a linear subspace.)

2. *If $G(x) = \emptyset$, then we check whether $G(x') = \emptyset$ for every x' that is a non-zero multiple of x. If this condition holds, then we return the $(k+1)$-by-n matrix that contains V as its first k rows and x as its $k+1^{\text{st}}$ row. Otherwise we reject.*

   (As shown in the beginning of the proof of Proposition 4.8, this $(k+1)$-by-n matrix V' is full-rank and satisfies $c'V' \notin H$ for every $c' \in \mathcal{F}^{k+1} \setminus \{0^{k+1}\}$.)

Assuming that the emulation was not suspended, we accept if the linearity tester accepts, and reject otherwise.

Recalling that linearity testing with proximity parameter ϵ' has complexity $O(1/\epsilon')$, the complexity of Algorithm 4.7 is $O(1/\epsilon') \cdot |\mathcal{F}|^k$, since each query to g is emulated using $|\mathcal{F}|^k$ queries to h.

Proposition 4.8 (analysis of Algorithm 4.7): *Suppose that $\epsilon' \in (0, 0.1]$ and V is a k-by-n full-rank matrix such that $cV \notin H$ for all $c \in \mathcal{F}^k \setminus \{0^k\}$. Then, the following holds.*

1. *If h describes an $(n-k)$-dimensional linear subspace, then Algorithm 4.7 always accepts.*

2. *If h describes a linear subspace, then Algorithm 4.7 always either accepts or returns a $(k+1)$-by-n matrix V'.*

3. *If h is ϵ'-far from describing a linear subspace, then, with high probability, Algorithm 4.7 either rejects h or returns a $(k+1)$-by-n matrix V'.*

Furthermore, whenever Algorithm 4.7 returns a matrix V' it holds that V' is full-rank and $c'V' \notin H$ holds for all $c' \in \mathcal{F}^{k+1} \setminus \{0^{k+1}\}$.

Proof: We first show that whenever Algorithm 4.7 returns a $(k+1)$-by-n matrix V', which augments V by a row x, it holds that V' is full-rank and $c'V' \notin H$ holds for all $c' \in \mathcal{F}^{k+1} \setminus \{0^{k+1}\}$. First observe that this event occurs only when for every x' that is a non-zero mnultiple of x it holds that $G(x') = \emptyset$. But in this case, if $c'V' \in H$ for some $c' = (c, b) \in \mathcal{F}^k \times \mathcal{F}$, then $b = 0$ (since otherwise $cV + bx \in H$ contradicting $G(bx) = \emptyset$), which implies $cV \in H$, which in turn implies $c = 0^k$ (by the corresponding feature of V), and so $c' = 0^{k+1}$. Lastly, note that V' is full-rank, since V is full-rank and x cannot be in the span of the rows of V, because $x = cV$ implies $x - cV = 0^n \in H$, which implies $G(x) \neq \emptyset$.

Next, suppose that H is an $(n-k)$-dimensional linear subspace. Then, the algorithm will always accept, since (by Claim 4.3) the function $g_{H,V}$ is linear. This establishes Part 1. As for Part 2, using Claim 4.4, it follows that in this case either the execution of the linearity test is suspended or the test accepts. Furthermore, in the former case, a string x was encountred such that $G(x) = \emptyset$, and $G(x') = \emptyset$ holds for any non-zero multiple of x (by linearity of H), which means that the algorithm returns an augmented matrix V'.

Turning to Part 3, we consider a function h that is ϵ'-far from describing an $(n-k)$-dimensional linear subspace, and recall that (by the premise of the proposition) also in this case V is a k-by-n full-rank matrix such that $cV \notin H$ for all $c \in \mathcal{F}^k \setminus \{0^k\}$. We shall show (below) that $g_{H,V}$ is ϵ'-far from the set of linear functions, and so (w.h.p.) the algorithm will either reject it or return an augmented matrix V'. (Indeed, given that g is far from linear, with high probability, an execution of the linearity test will either lead to rejection or to encounting a query x such that $g(x) = \perp$, which in turn leads either to rejection or to returning an augmented matrix V'.)

Assume, contrary to the foregoing claim, that $g = g_{H,V}$ is ϵ'-close to a linear function g', and recall that by Claim 4.6 it must be the case that the image of g' equals $\mathcal{F}^k$. Then, $H' = \{x \in \mathcal{F}^n : g'(x) = 0^k\}$ is an $(n-k)$-dimensional linear subspace, since $x, x' \in H'$ implies $x + x' \in H'$ (because $g'(x+x') = g'(x) + g'(x') = 0^k + 0^k = 0^k$), and $|H'| = |\mathcal{F}|^n / |\mathcal{F}|^k$ (because each image of g' has the same number of preimages). Next, letting $h' : \mathcal{F}^n \to \{0,1\}$ describe H' (i.e., $h'(x) = 1$ iff $x \in H'$), we show that $g'(x) = g(x)$ implies $h'(x) = h(x)$. This is because $g'(x) = g(x) = 0^k$ implies $h'(x) = 1$ and $h(x) = 1$ (since $g^{-1}(0^k) \subseteq h^{-1}(1)$), whereas $g'(x) = g(x) \notin \{0^n, \perp\}$ implies $h'(x) = 0$ and $h(x) = 0$ (since $h(x) = 1$ implies $g(x) \in \{0^n, \perp\}$). It follows that h is ϵ'-close to h', which contradicts our hypothesis that h is ϵ'-far from describing an $(n-k)$-dimensional linear subspace. ■

4.1.4 The actual tester

Using Algorithm 4.7, we obtain a tester for functions describing linear subspaces by using the fact if $f^{-1}(1)$ has density ρ then f is ρ-close to the all-zero function. Hence, on proximity parameter ϵ, we can afford to accept any function that seems to have density at most ϵ. On the other hand, if $h : \mathcal{F}^n \to \{0,1\}$ has density $\rho > 2 \cdot |\mathcal{F}|^{-k}$, where $\rho > \epsilon$, and we have a full-rank k-by-n matrix V such that $h(cV) \neq 1$ for all $c \in \mathcal{F}^k \setminus \{0^k\}$, then we can find a witness to the non-linearity of $h^{-1}(1)$ (i.e., x such that $|\{c : h(x + cV) = 1\}| > 1$; see Claim 4.5). This means that we may iteratively invoke Algorithm 4.7 till we get a k-by-n matrix V as above, which means that reaching $k > \log_{|\mathcal{F}|}(2/\epsilon) > \log_{|\mathcal{F}|}(2/\rho)$ suffices. A straightforward implementation of the foregoing idea follows, where we assume for simplicity that $h(0^n) = 1$.[11]

[11]Indeed, this assumption can be eliminated by checking whether $h(0^n) = 1$ and rejecting if this does not hold, while relying on the fact that 0^n reside in any linear subspace.

Algorithm 4.9 (testing linear subspaces, naive version): *On input $\epsilon > 0$ and oracle access to $h : \mathcal{F}^n \to \{0,1\}$ such that $h(0^n) = 1$, we proceed as follows.*

1. (Attempts to find an element outside H): *We take $O(1/\epsilon)$ random samples of $\mathcal{F}^n$, and accept if all the sample points reside in $H = h^{-1}(1)$. Otherwise, we pick an arbitrary sample point $x \in \mathcal{F}^n \setminus H$, and define V to be the 1-by-n matrix that contains x. Letting $t = \lfloor \log_{|\mathcal{F}|}(2/\epsilon) \rfloor$, we set $k = 1$, and proceed iteratively as follows.*

2. (Iterative invocations of the quasi-tester): *If $k \leq t$, then we invoke Algorithm 4.7 on input a k-by-n matrix V and $\epsilon' = \epsilon$, for at most $O(t+1-k)$ times, and handle the outcomes as follows.*

   (a) *If any of these invocations rejected, then we reject.*

   (b) *If any of these invocations returned a $(k+1)$-by-n matrix V', then we proceed to the next iteration* (of Step 2) *using $V \leftarrow V'$ and $k \leftarrow k+1$.*

   (c) *Otherwise* (i.e., if all invocations accepted)*, then we halt and accept.*[12]

   If $k > t$, then we proceed to the next step.

   (Note that we may proceed to Step 3 both in case H is a linear subspace of dimension smaller than $n-t$ and in case h is ϵ-far from describing a linear subspace.)

3. (Additional attempts to find witnesses for the non-linearity of H): *We select a random sample S of $O(1/\epsilon)$ elements of $\mathcal{F}^n$, and reject if for any $x \in S$ it holds that $|\{c \in \mathcal{F}^k : h(x+cV) = 1\}| \geq 2$. Otherwise, we accept.*

Whenever we halt while accepting, we also output the current matrix V, where V is fictitiously defined as empty in the case of halting in Step 1.

The complexity of the foregoing Algorithm 4.9 is

$$O(1/\epsilon) + \sum_{k\in[t]} O(t+1-k) \cdot O(|\mathcal{F}|^k/\epsilon) + O(|\mathcal{F}|^{t+1}/\epsilon) = O(|\mathcal{F}|/\epsilon^2),$$

where the second and third terms account for the complexity of Steps 2 and 3 (and $|\mathcal{F}|^t < 2/\epsilon$ by the definition of t).

Proposition 4.10 (analysis of Algorithm 4.9): *Algorithm 4.9 constitutes a one-sided error tester for linear subspaces of $\mathcal{F}^n$. Furthermore:*

1. *If the tested function h describes an $(n-k^*)$-dimensional subspace and $\epsilon \leq |\mathcal{F}|^{-k^*} < 1$, then, with high probability, the algorithm also outputs a k^*-by-n full-rank matrix V such that $h(cV) = 1$ if and only if $c = 0^{k^*}$.*

2. *Whenever Algorithm 4.10 outputs a matrix V, for some $k \in [t+1]$, it holds that V is a k-by-n full-rank matrix and $h(cV) = 0$ for every $c \in \mathcal{F}^k \setminus \{0^k\}$.*

[12]Indeed, we risk a small probability of accepting a function that is far from describing a linear subspace, but this could have happened also if we were to proceed to the next step (i.e., Step 3).

Proof: Suppose that $h : F^n \to \{0,1\}$ describes an $(n-k^*)$-dimensional linear subspace H. Since Step 1 never rejects (actually, it may even accept (especially, when $\frac{|H|}{|\mathcal{F}|^n} = |\mathcal{F}|^{-k^*} < 0.8 \cdot \epsilon$)), we consider Steps 2 and 3. Step 2 features invocations of Algorithm 4.7, and by (Parts 1 and 2 of) Proposition 4.8, each of these invocations either accepts or returns an augmented matrix. Hence, we either accept in Step 2 or proceed to Step 3. Furthermore, in the latter case, $k = t+1$, and the current matrix V is a full-rank k-by-n matrix such that $cV \notin H$ for every $c \in \mathcal{F}^k \setminus \{0^k\}$, which implies that $k \le k^*$. Anyhow, by Part 1 of Claim 4.4, $|\{c \in \mathcal{F}^k : x+cV \in H\}| \le 1$ for every $x \in \mathcal{F}^n$, and it follows that Step 3 accepts.

We now turn to the case that $h : F^n \to \{0,1\}$ is ϵ-far from describing a linear subspace. Note that $H = h^{-1}(1)$ has density between ϵ and $1-\epsilon$, since otherwise h is ϵ-close to a constant function. Using the upper bound on $|H|$, it follows that, with high probability, we proceed to Step 2 (rather than accept in Step 1). Now, by Part 3 of Proposition 4.8, with high probability, each invocation of Algorithm 4.7 either rejects or returns an augmented matrix. Hence, the probability that we accept in iteration k is upper-bounded by $\exp(-O(t+1-k))$, which means that the probability of accepting in Step 2 is at most $\sum_{k\in[t]} \exp(-O(t+1-k))$. Hence, with high probability, we reach Step 3, whereas $t+1 > \log_{|\mathcal{F}|}(2/\epsilon)$. Using (the main part of) Claim 4.5, it follows that Step 3 accepts with probability at most $(1-(\epsilon-|\mathcal{F}|^{-(t+1)}))^{O(1/\epsilon)} < (1-(\epsilon/2))^{O(1/\epsilon)}$. Hence, Step 3 rejects with high probability.

Turning to the furthermore clause, we first consider the case that h describes an $(n-k^*)$-dimensional linear subspace H, observe that if $\epsilon \le |\mathcal{F}|^{-k^*}$ and $k^* > 0$, then (w.h.p.) Algorithm 4.9 does not terminate at Step 1, but rather proceeds to Step 2 with some 1-by-n matrix consisting of a row not in H. Similarly, (w.h.p.) all invocations of Algorithm 4.7 with $k < k^*$ (which take place in Step 2) return an augmented matrix. In this case, at termination, which occurs when $k = k^*$, Algorithm 4.9 outputs a matrix as claimed. This establishes Part 1 of the furthermore claim. Lastly, Part 2 follows by inspection (and the furthermore clause of Proposition 4.8). ■

Reducing the complexity of Step 3 of Algorithm 4.9. Using the furthermore clause of Claim 4.5, rather than its main part, we can reduce the complexity of Step 3 by searching for a witness (for non-linearity) in a more economical manner (cf., [6, Sec. 8.2.4]). Specifically, letting $k = t+1$ and $m = k \log|\mathcal{F}|$, for every $i \in [m]$ we take a sample of $O(m\epsilon^{-1}/2^{m-i})$ points in $\mathcal{F}^n$ and for each such x we try to find two distinct elements in $\{c \in \mathcal{F}^k : h(x+cV)=1\}$ by taking a sample of $O(2^{m-i})$ elements in $\mathcal{F}^k$. That is:

Algorithm 4.11 (revised version of Step 3 of Algorithm 4.9): *On input parameter ϵ, a k-by-n full-rank-matrix V, and oracle h, letting $m = k\log|\mathcal{F}|$, we proceed as follows. For every $i \in [m]$, select a random sample S_i of $O(m\epsilon^{-1}/2^{m-i})$ elements of $\mathcal{F}^n$, and a random sample T_i of of $O(2^{m-i})$ elements in $\mathcal{F}^k$. If for some $i \in [m]$ and $x \in S_i$ it holds that $|\{c \in T_i : h(x+cV)=1\}| \ge 2$, then reject. Otherwise, accept.*

The complexity of the Algorithm 4.11 is $O(m^2/\epsilon) = O((k\log|\mathcal{F}|)^2/\epsilon)$, and replacing Step 3 of Algorithm 4.9 by it yields a tester for linear subspaces. The latter claim follows by modifying the proof of Proposition 4.10. Specifically, when reaching Step 3 we use the furthermore clause of Claim 4.5, which guarantees the existence of $i \in [m]$ such that for at least a $2^{m-i-1} \cdot (\epsilon - |\mathcal{F}|^{-k})/m = \Omega(\epsilon 2^{m-i}/m)$ fraction of the points $x \in \mathcal{F}^n$ satisfy $|\{c \in \mathcal{F}^k : h(x+cV)=1\}| > 2^{i-1}$. For this i, with high probability, the sample S_i contains at least one point x such that $|\{c \in \mathcal{F}^k : h(x+cV) =$

$1\}| > 2^{i-1}$, and with high probability T_i contains two distinct elements in that set. For sake of clarity and future reference, we summarize the latter argument as follows. (This argument used $\epsilon > 2 \cdot |\mathcal{F}|^{-k}$, but it extends to $\epsilon - |\mathcal{F}|^{-k} = \Omega(\epsilon)$.)

Claim 4.12 (analysis of Algorithm 4.11): *Let H, V and $g = g_{H,V}$ be as in Definition 4.2, and recall that V is a k-by-n full-rank matrix. If H has density at least ϵ and $\epsilon > 1.5 \cdot |\mathcal{F}|^{-k}$, then Algorithm 4.11 rejects with high probability.*

4.2 The second reduction

Given the revised version of Step 3 of Algorithm 4.9, presented in Algorithm 4.11, the complexity of the revised algorithm is dominated by Step 2, in which Algorithm 4.7 is invoked with its proximity parameter, denoted ϵ', set to ϵ. Recall that when invoked with a k-by-n matrix and proximity parameter ϵ', Algorithm 4.7 makes $O(|\mathcal{F}|^k/\epsilon')$ queries.

We improve the complexity of Step 2 of Algorithm 4.9 by replacing each invocation of Algorithm 4.7 (in Step 2) with two sub-steps. First (in Step 2A) we invoke Algorithm 4.7 while setting its proximity parameter to a small constant (e.g., $\epsilon' = 0.1$), and then (in Step 2B) we check whether the function h' defined based on the linear function that is closest to $g = g_{H,V}$ is ϵ-close to h.

Our starting point is the fact that, for every $\epsilon' < 1/4$, if g is ϵ'-close to being a linear function, then it is ϵ'-close to a unique linear function g', which can be computed by self-correction of g (where each invocation of the self-corrector makes two queries to g and is correct with probability at least $1 - 2\epsilon'$). Furthermore, the corresponding Boolean function h' (i.e., $h'(x) = 1$ iff $g'(x) = 0^k$) describes an $(n-k)$-dimensional linear subspace, whereas if h describes an $(n-k)$-dimensional linear subspace then $g' = g$ and $h' = h$.

The key observation is that if h is ϵ-far from describing an $(n-k)$-dimensional linear subspace, then, with high probability, either Step 2A rejects or it yields a function $g = g_{H,V}$ that is ϵ'-close to a linear function g' with image $\mathcal{F}^k$. In the latter case, the corresponding h', which describes an $(n-k)$-dimensional linear subspace (i.e., $\{x \in \mathcal{F}^n : h'(x) = 1\} = \{x \in \mathcal{F}^n : g'(x) = 0^k\}$), must be ϵ-far from h (by the foregoing hypothesis). (On the other hand, if h describes an $(n-k)$-dimensional linear subspace, then $h' = h$.)

As hinted above, Step 2B consists of testing whether h' equals h, when this testing task is performed with respect to proximity parameter ϵ. This testing is performed by using a sample of $O(1/\epsilon)$ points. For each sample point, the value of h is obtained by querying h, whereas the value of h' on all sample points is obtained by obtaining the values of g' on these points (since $h'(x) = 1$ iff $g'(x) = 0^k$), where the values of g' on these points are computed via self-correction of g.

The problem with the foregoing description is that each query to g is implemented by making $|\mathcal{F}|^k$ queries to h. Hence, a straightforward implementation of the foregoing procedure will result in making $O(|\mathcal{F}|^k/\epsilon)$ queries to h, which is no better than Algorithm 4.9. Instead, we shall use a sample of $O(1/\epsilon)$ *pairwise-independent* points such that their g'-values can be determined by the value of g' at $O(\log(1/\epsilon))$ points, which in turn are computed by self-correction of g that uses $|\mathcal{F}|^k$ queries to h per each point. The details are given in Algorithm 4.13.

Note that if h describes an $(n-k)$-dimensional linear subspace, then $g = g'$, and Step 2B accepts once reached (which always happens). On the other hand, if h is ϵ-far from this property and Step 2B is reached (which implies that g, g' and h' are well-defined), then h is ϵ-far from h', and a sample of $O(1/\epsilon)$ pairwise-independent points will contain a point of disagreement (w.h.p.).

As in Step 2 of Algorithm 4.9, such a point will yield either a witness against linearity of $h^{-1}(1)$ or an augmentation of the current matrix.

The pairwise independent sample points. The key observation here is that Step 2B can be implemented within complexity $\widetilde{O}(1/\epsilon)$ by taking a sample of $m = O(1/\epsilon)$ *pairwise independent points* in $\mathcal{F}^n$ such that evaluating g' on these m points only requires time $O(m + |\mathcal{F}|^k \cdot \widetilde{O}(\log m))$ rather than $O(|\mathcal{F}|^k \cdot m)$ time. This is done as follows.

For $t' = \lceil \log_{|\mathcal{F}|}(m+1) \rceil$, select uniformly at random $s^{(1)}, ..., s^{(t')} \in \mathcal{F}^n$, compute each $g'(s^{(j)})$ via self-correcting g, with error probability $0.01/t'$, and use the sample points $r^{(L)} = L(s^{(1)}, ..., s^{(t')})$ for m arbitrary distinct non-zero linear functions $L : \mathcal{F}^{t'} \to \mathcal{F}$. The key observations are that (1) the $r^{(L)}$'s are pairwise independent, and (2) the values of g' at all $r^{(L)}$'s can be determined based on the values of g' on the $s^{(j)}$'s. This determination is based on the fact that $g'(r^{(L)}) = L(g'(s^{(1)}), ..., g'(s^{(t')}))$, by linearity of g'. Hence, the values of g' on t' random points (i.e., the $s^{(j)}$'s) determines the value of g' on $m \leq |\mathcal{F}|^{t'} - 1$ pairwise independent points (i.e., the $r^{(L)}$'s).[13] This yields the following —

Algorithm 4.13 (implementing Step 2B): *On input proximity parameter $\epsilon \in (0, 0.1]$, a full-rank k-by-n matrix V, and oracle access to $h : \mathcal{F}^n \to \{0,1\}$, letting $m = O(1/\epsilon)$ and $t' = \lceil \log_{|\mathcal{F}|}(m+1) \rceil$, we set $t'' = O(\log_2 t')$, and proceed as follows.*

1. *Select uniformly $s^{(1)}, ..., s^{(t')} \in \mathcal{F}^n$.*

2. Determining the value of g' at the $s^{(j)}$'s: *For every $j \in [t']$, compute the value of $g'(s^{(j)})$ by using self correction on $g = g_{h^{-1}(1),V}$, which in turn queries h on $|\mathcal{F}|^k$ points per each query to g. The self-correction procedure is invoked t'' times so that the correct value is obtained with probability* $1 - 0.01/t'$.

   Specifically, select uniformly $w^{(1)}, ..., w^{(t'')} \in \mathcal{F}^n$, and set $\sigma^{(j)}$ to equal the majority vote among the values $g(s^{(j)} + w^{(1)}) - g(w^{(1)}), ..., g(s^{(j)} + w^{(t'')}) - g(w^{(t'')})$, where the values of g at each point x is determined according to the value of h at the points $\{x + cV : c \in \mathcal{F}^k\}$. (Actually, we can afford to reject if the values in $\mathcal{F}^k$ are not in consensus, since this yields a witness to the non-linearity of g.)

   Recall that $g(x) = c$ if c is the unique point in $\mathcal{F}^k$ such that $h(x + cV) = 1$, and $g(x) = \perp$ otherwise. If the value of g at any point is set to $\perp$, then we suspend the execution and either reject or return an augmented matrix (as in Algorithm 4.7).[14]

   (Indeed, $\sigma^{(j)}$ is the self-corrected value of $g'(s^{(j)})$, and it is correct with probability $1 - \exp(-\Omega(t'')) > 1 - 0.01/t'$).

3. Determining the value of g' at the $r^{(L)}$ and checking them against the values at h: *For each of m non-zero linear functions $L : \mathcal{F}^{t'} \to \mathcal{F}$, let $r^{(L)} = L(s^{(1)}, ..., s^{(t')})$ and check whether $h(r^{(L)})$ equals our guess for $h'(r^{(L)})$, where the latter value is set to 1 if and only if $L(\sigma^{(1)}, ..., \sigma^{(t')}) = 0^k$. Accept if all checks were successful* (i.e., equality holds in all). *Otherwise* (i.e., a point of disagreement if found)*, reject.*

[13] This procedure is inspired by [8] (as presented in [5, Sec. 7.1.3] for $\mathcal{F} = \mathrm{GF}(2)$).

[14] Specifically, letting $G(x) \stackrel{\text{def}}{=} \{c \in \mathcal{F}^k : h(x) = c\}$, if $|G(x)| \geq 2$, then we reject, and otherwise (i.e., when $G(x) = \emptyset$) we augment the matrix with x if $G(x') = \emptyset$ for any x' that is a non-zero multiple of x (and reject if $G(x') \neq \emptyset$).

(Recall that $g'(r^{(L)}) = g'(L(s^{(1)},...,s^{(t')})) = L(g'(s^{(1)}),...,g'(s^{(t')}))$. Hence, $L(\sigma^{(1)},...,\sigma^{(t')})$ is our educated guess for $g'(r^{(L)})$, and this guess is correct if all guesses for the $g'(s^{(j)})$'s are correct, which happens with probability 0.99).

The time complexity of Algorithm 4.13 is $O(t' \cdot t'' \cdot |\mathcal{F}|^k + m) = \widetilde{O}(\log_{|\mathcal{F}|}(1/\epsilon)) \cdot |\mathcal{F}|^k + O(1/\epsilon)$, which is $\widetilde{O}(1/\epsilon)$ when $\epsilon = O(|\mathcal{F}|^{-k})$. Hence, the time complexity of Step 2B dominates the time complexity of Step 2A, which is $O(|\mathcal{F}|^k)$. For sake of clarity, we spell out the final resulting algorithm.

Algorithm 4.14 (testing linear subspaces, improved version): *On input $\epsilon > 0$ and oracle access to $h : \mathcal{F}^n \to \{0,1\}$ such that $h(0^n) = 1$, we let $t = \lfloor \log_{|\mathcal{F}|}(2/\epsilon) \rfloor$, and proceed as follows.*

1. (Attempts to find an element outside H): *As Step 1 of Algorithm 4.9.*
2. (Iterative invocations of the quasi-tester): *We replace the invocations of Algorithm 4.7 performed in Step 2 of Algorithm 4.9 by Steps 2A and 2B as described above, where Step 2A invokes Algorithm 4.7 with proximity parameter $\epsilon' = 0.1$ and Step 2B is as detailed in Algorithm 4.13. This means that for any $k \leq t$, we invoke the combined Steps 2A and 2B for at most $O(t + 1 - k)$ times, while providing them with a full-rank k-by-n matrix V and oracle access to H, and handle the outcomes as follows.*
   - (a) *If any of these Step 2A+2B invocations rejected, then we reject.*
   - (b) *If any of these invocations returned a $(k+1)$-by-n matrix V', then we proceed to the next iteration using $V \leftarrow V'$ and $k \leftarrow k + 1$.*
   - (c) *Otherwise* (i.e., if all invocations accepted)*, then we halt and accept.*

   If $k > t$, then we proceed to the next step.
3. (Additional attempts to find witnesses for the non-linearity of H): *We replace Step 3 of Algorithm 4.9 by Algorithm 4.11.*

As in Algorithm 4.9, whenever the algorithm halts while accepting, it outputs the current matrix V, which in the case of Step 1 is fictitiously defined as empty.

The complexity of the foregoing algorithm is $\widetilde{O}(1/\epsilon) + (O(|\mathcal{F}|^t) + \widetilde{O}(1/\epsilon)) + O((t \log |\mathcal{F}|)^2/\epsilon) = \widetilde{O}(1/\epsilon)$, where the three terms correspond to the complexities of the three steps, and the inequality uses $t \leq \log_{|\mathcal{F}|}(2/\epsilon)$.

Proposition 4.15 (analysis of the foregoing algorithm): *Algorithm 4.14 constitutes a one-sided error tester for linear subspaces, and the furthermore claim of Proposition 4.10 holds too.*

Proof Sketch: We consider a single invocation of Steps 2A and 2B, where V is a full-rank k-by-n matrix such that $h(cV) = 0$ if and only if $c = 0^k$. If h describes an $(n-k)$-dimensional linear subspace, then the execution always reaches Step 2B, which always accepts, since in this case $h' = h$. If h describes an arbitrary linear subspace, then each of these steps never rejects (i.e., it either accepts or returns an augmented matrix), where the crucial observation is that rejection (in either steps) is always backed by a witness of non-linearity of $h^{-1}(1)$. (In particular, note that a point of disagreement, $r^{(L)}$, found in Step 2B.3 implies that $g(r^{(L)}) \neq L(\sigma^{(1)},...,\sigma^{(t')})$, which in turn yields a violation of the linearity of g.) On the other hand, if h is ϵ-far from describing a linear subspace, then we consider two cases.

1. If h is 0.1-far from describing a linear subspace, then (w.h.p.) Step 2A either rejects or returns an augmented matrix. Furthermore, the latter happens also if $g = g_{H,V}$ is 0.1-far from being linear.

2. Otherwise, assuming that Step 2B is reached, we consider the corresponding functions $g = g_{H,V}$ and g'. Recall g' is a linear function that is 0.1-close to g, since otherwise the foregoing case holds, and that the image of g' equals $\mathcal{F}^k$ (see Claim 4.6). Hence, h must be ϵ-far from h', since in this case h' describes an $(n-k)$-dimensional linear subspace.

   In this case, with high probability, in Step 2 of Algorithm 4.13 (which implements Step 2B), the algorithm either rejects or returns an augmented matrix or obtains the correct values of g' at all $s^{(j)}$'s, where the first two cases are due to encountering $\perp$. In the latter case, these values (i.e., the $g'(s^{(j)})$'s) correctly determine the values of g' at all the $r^{(L)}$'s. Since these $r^{(L)}$ are uniformly distributed in $\mathcal{F}^n$ in a pairwise independent manner, with probability at least $1 - \frac{m\epsilon}{(m\epsilon)^2}$, the sample contains a point on which h and h' disagree. In this case Step 3 of Algorithm 4.13 rejects.

In conclusion, if h is ϵ-far from the being a linear subspace, then (w.h.p.) an execution of Steps 2A and 2B either rejects or returns an augmented matrix. Combining this with the rest of the proof of Proposition 4.10, the current claims follows. ■

Conclusions. Proposition 4.15 implies that there exists a one-sided error tester of complexity $\widetilde{O}(1/\epsilon)$ for linear subspaces. This establishes Part 1 of Theorem 1.4. Furthermore, a closer look at its proof reveals that in case of accepting, the tester can indicate an upper bound on the dimension of the linear subspace; that is, in this case it can always output $k \in [0, K]$ whenever h describes an $(n-K)$-dimensional subspace (and does so with high probability when h is close to describing such a subspace). Hence, we also obtain a tester for the class of linear spaces of dimension at least $n-K$. This establishes Part 2 of Theorem 1.4.

5 Testing Monomials via Testing Affine Subspaces

As stated in Section 2, the function $f : \{0,1\}^n \to \{0,1\}$ is a monotone k-monomial if and only if f describes an $(n-k)$-dimensional affine subspace (over $\mathrm{GF}(2)^n$) that is a translation by 1^n of an $(n-k)$-dimensional axis-parallel linear subspace; that is, if $f^{-1}(1)$ has the form $\{yG + 1^n : y \in \{0,1\}^{n-k}\}$, where G is a full-rank $(n-k)$-by-n Boolean matrix that contains k all-zero columns. Hence, we may focus on testing that the function $h : \{0,1\}^n \to \{0,1\}$ defined by $h(x) \stackrel{\mathrm{def}}{=} f(x + 1^n)$ describes an $(n-k)$-dimensional *axis-parallel* linear subspace. (Indeed, the reduction used in the proof of Claim 4.1 is instantiated here by mandating that $u = 1^n$.)

Actually, analogously to the previous section, we aim at presenting one-sided error testers for the case that the co-dimension is not specified (or is only upper-bounded). Furthermore, we consider the more general task of testing that a function $h : \mathcal{F}^n \to \{0,1\}$ describes an axis-parallel linear subspace, where $\mathcal{F}$ is an arbitrary finite field (and $\mathcal{F} = \mathrm{GF}(2)$ is merely the special case used for testing monomials). As in the previous section, we denoted the tested function by h and let $H = h^{-1}(1)$.

Overview. The basic strategy is to first apply a tester of linear subspaces such as Algorithm 4.9 (or Algorithm 4.14). Ignoring the case that this tester halts in Step 1 (while ruling that $H = \mathcal{F}^n$), recall that whenever this tester does not reject, it provides a k-by-n matrix V that yields a function $g = g_{H,V}$ as in Definition 4.2, and that $g : \mathcal{F}^n \to \mathcal{F}^k \cup \{\perp\}$ is close to a linear function with image $\mathcal{F}^k$.

The next step is trying to find $k+1$ disjoint subsets of $[n]$ that influence h (i.e., each subset contains a variable that influences h) along with Boolean vectors that lie outside H and are supported by these subsets (see definition below), where this search is performed by using self-correction on g. Note that if H is an $(n-k)$-dimensional axis-parallel linear subspace, then this attempt is bound to fail (since in that case h is influenced by k locations in $[n]$). Indeed, in case of failure we halt accepting. On the other hand, we show that if H is (close to) an $(n-k)$-dimensional linear subspace that is not axis-parallel, then such $k+1$ vectors are found (w.h.p.).

Now, suppose that we found such vectors, denoted $v^{(1)}, ..., v^{(k+1)}$; that is, $v^{(i)} \notin H$ and the non-zero entries of the $v^{(i)}$'s reside in disjoint locations. Then, we check whether $\sum_{i\in[k+1]} c_i v^{(i)} \notin H$ for all $(c_1, ..., c_{k+1}) \in \mathcal{F}^{k+1} \setminus \{0^{k+1}\}$, and reject if this condition does not hold. As shown below (see Claim 5.1), this condition holds when H is an axis-parallel linear subspace. Hence, assuming that the condition does hold and letting V' be the corresponding $(k+1)$-by-n matrix, we invoke Step 3 of Algorithm 4.9 in a final attempt to find evidence that H is not an axis-parallel linear subspace. If this attempt fails, then we halt accepting. Note that if H is an $(n-k)$-dimensional subspace that is not axis-parallel, then such $k+1$ vectors must be dependent and so $\{c'\mathcal{F}^{k+1} : x + c'V \in H'\}| \geq 2$ for every x, and Step 3 rejects. We shall show that the same happens also when H is close to being an $(n-k)$-dimensional subspace that is not axis-parallel.

Notation and a key observation. The support of a vector $v \in \mathcal{F}^n$, denoted $\mathsf{supp}(v)$, is defined as the set of locations in which v holds non-zero entries; that is, $i \in \mathsf{supp}(v)$ if and only if the i^{th} element of v is not zero. Hence, the second step in our algorithm is searching for distinct vectors $v^{(1)}, ..., v^{(k+1)}$ such that $\mathsf{supp}(v^{(i)}) \cap \mathsf{supp}(v^{(j)}) = \emptyset$ for every $i \neq j \in [k+1]$. We claim that if H is an axis-parallel linear subspace, then $\sum_{i\in[k+1]} c_i v^{(i)} \notin H$ for all $(c_1, ..., c_{k+1}) \in \mathcal{F}^{k+1} \setminus \{0^{k+1}\}$. This follows from the case of two vectors formulated next.[15]

Claim 5.1 (on axis-parallel linear subspaces): *Suppose that $H \subseteq \mathcal{F}^n$ is an axis-parallel linear subspace; that is, for some $J \subseteq [n]$ it holds that $H = \{x \in \mathcal{F}^n : x_J = 0^{|J|}\}$. Then, for any $u, w \in \mathcal{F}^n \setminus H$ such that $\mathsf{supp}(u) \cap \mathsf{supp}(w) = \emptyset$, and every $\alpha, \beta \in \mathcal{F} \setminus \{0\}$, it holds that $\alpha u + \beta w \notin H$.*

Proof: Let H and $J \subseteq [n]$ be as in the hypothesis. Then, $v \in H$ holds if and only if $\mathsf{supp}(v) \cap J = \emptyset$. Hence, $u \notin H$ implies that $\mathsf{supp}(u) \cap J \neq \emptyset$, and ditto for w. Assuming that $\mathsf{supp}(u) \cap \mathsf{supp}(w) = \emptyset$, we have $\mathsf{supp}(\alpha u + \beta w) = \mathsf{supp}(u) \cup \mathsf{supp}(w)$ for every $\alpha, \beta \in \mathcal{F} \setminus \{0\}$. Hence,

$$\begin{aligned}
\mathsf{supp}(\alpha u + \beta w) \cap J &= (\mathsf{supp}(u) \cup \mathsf{supp}(w)) \cap J \\
&= (\mathsf{supp}(u) \cap J) \cup (\mathsf{supp}(w) \cap J) \\
&\neq \emptyset
\end{aligned}$$

which implies that $\alpha u + \beta w \notin H$. ■

[15] That is, we prove by induction on $k' = 2, ..., k+1$ that $\sum_{i\in[k']} c_i v^{(i)} \notin H$ for all $(c_1, ..., c_{k'}) \in \mathcal{F}^{k'} \setminus \{0^{k'}\}$.

The actual algorithm. In order to obtain complexity $\widetilde{O}(1/\epsilon)$ we perform the first step (of testing linear subspace) by using Algorithm 4.14, although using Algorithm 4.9 (or any other tester that also provides a basis for the complementing subspace) will alsoi yield a tester. Again, we assume for simplicity that $h(0^n) = 1$.

Algorithm 5.2 (testing axis-parallel linear subspaces): *On input $\epsilon > 0$ and oracle access to $h : \{0,1\}^n \to \{0,1\}$ such that $h(0^n) = 1$, we proceed as follows, while letting $t = \lfloor \log_{|\mathcal{F}|}(2/\epsilon) \rfloor$.*

1. (Testing that $H = h^{-1}(1)$ is a linear subspace by invoking Algorithm 4.14): *We invoke Algorithm 4.14 with proximity parameter set to $\epsilon/10$, and reject if it rejects. If the said invocation accepted in its first step* (i.e., Step 1 of Algorithm 4.14)*, then we accept. Otherwise* (i.e., the invocation accepted at a later step)*, we let V denote the k-by-n matrix output by the invocation, where $k \in \mathbb{N}$, and proceed to the next step using V.*

2. (Searching for influential sets and corresponding vectors): *If $k > t$, then we halt and accept. Otherwise, we select a random partition of $[n]$ into $t' = O(t^2)$ sets, denoted $S_1, ..., S_{t'}$, and proceed as follows.*

   (a) *For every $i \in [t']$, we test whether S_i influences h by selecting random assignments to the variables in S_i and checking whether h evaluates to 0 under any of these assignments. Specifically, we perform $t'' = O(\log t)$ such trials, where in each trial we select a random vector uniformly among all vectors that have non-zero entries only in locations that reside in S_i. We check whether such a vector $v \in \mathcal{F}^n$* (with $\mathsf{supp}(v) \subseteq S_i$) *is in H by selecting a random $r \in \mathcal{F}^n$ and checking whether $g_{H,V}(v + r) = g_{H,V}(r)$; that is, if $g_{H,V}(v + r) - g_{H,V}(r) \neq 0^k$, then it should be the case that $v \notin H$, and we may reject if $h(v) = 1$. Otherwise* (i.e., $g_{H,V}(v + r) \neq g_{H,V}(r)$ and $h(v) = 0$)*, we let $v^{(i)} \leftarrow v$, and say that S_i* influences *h.*
   (This term is justified by the fact that $h(0^n) = 1$.)

   (b) *Let $I \subseteq [t']$ be the set of indices of the influential sets. If $|I| \leq k$, then we accept. Otherwise* (i.e., $|I| > k$)*, if $|I| > t$, then let I be an arbitrary $(t + 1)$-subset of I.*
   For simplicity, suppose $I = [|I|]$. If there exists $(c_1, ..., c_{|I|}) \in \mathcal{F}^{|I|} \setminus \{0^{|I|}\}$ such that $\sum_{i \in I} c_i v^{(i)} \in H$, then we reject. Otherwise (i.e., $\sum_{i \in I} c_i v^{(i)} \notin H$ for all $(c_1, ..., c_{|I|}) \in \mathcal{F}^{|I|} \setminus \{0^{|I|}\}$)*, we let $k \leftarrow |I|$ and V be a $|I|$-by-n matrix whose rows are the vectors in $\{v^{(i)}\}_{i \in I}$.*

3. (Last attempt to find witness for the non-linearity of H): *Using the foregoing V, perform Step 3 of Algorithm 4.14* (equiv., Algorithm 4.11)*, with proximity parameter set to* $1.5|\mathcal{F}|^{-|I|}$*, and output its verdict.*

The complexity of this algorithm is $\widetilde{O}(1/\epsilon) + \widetilde{O}(t^2) \cdot |\mathcal{F}|^t = \widetilde{O}(1/\epsilon)$, where the first term accounts for Steps 1 and 3, the second term accounts for Step 2, and the inequality uses $t = \lfloor \log_{|\mathcal{F}|}(2/\epsilon) \rfloor$.

Proposition 5.3 (analysis of the algorithm): *Algorithm 5.2 constitutes a one-sided error tester for axis-parallel linear subspaces.*

Proof: If h describes an axis-parallel linear subspace of co-dimension $k^* > 0$, then Algorithm 4.14 always accepts, while outputting a matrix of dimension $k \leq k^*$.[16] If $k > t$, then Algorithm 5.2

[16]Needless to say, if $h \equiv 1$, then Algorithm 4.14 always accepts in Step 1, and in that case Algorithm 5.2 always accepts.

accepts at the beginning of Step 2. Otherwise (i.e., $k \leq t$), then it is always the case that the set I found in Step 2 has size at most k^*, and if $|I| \leq k$, then we accept in Step 2. Hence, whenever we proceed to Step 3 it holds that $k < |I| \leq k^*$ and the corresponding matrix satisfies the condition of Definition 4.2, which implies that Step 3 (if reached) always accepts (see Claim 4.4). Lastly, by Claim 5.1, every (non-zero) linear combination of the vectors $\{v^{(i)}\}_{i \in I}$ is not in $h^{-1}(1)$, which means that Step 2 never rejects.

We now turn to the case that h is ϵ-far from describing an axis-parallel linear subspace. If h is $\epsilon/10$-far from any linear subspace, then Algorithm 4.14 (invoked in Step 1) rejects with high probability. Hence, we focus on the case that h is $\epsilon/10$-close to describing some linear subspace (which is not axis-parallel). Note that the co-dimension of this subspace, denoted k^*, is at most t, because otherwise h is ϵ-close to describing the linear subspace $\{0^n\}$ (since $0.1\epsilon + |\mathcal{F}|^{-(t+1)} + |\mathcal{F}|^{-n} < \epsilon$). Furthermore, for every $k < k^*$ it holds that h is 0.3ϵ-far from describing any $(n-k)$-dimensional linear subspace, since the distance between linear subspaces of co-dimensions $k^* \in [t]$ and $k \in [k^*-1]$ is at least $|\mathcal{F}|^{-k} - |\mathcal{F}|^{-k^*} \geq |\mathcal{F}|^{-t} \leq \epsilon/2$. It follows that, with high probability, Algorithm 4.14 either rejects or accepts while outputting a matrix V of rank $k \geq k^*$. Furthermore, in the latter case, the corresponding function $g = g_{H,V}$ is 0.1ϵ-close to a linear function $g' : \{0,1\}^n \to \{0,1\}^k$.

Note that g' must be influenced by more than k variables, since otherwise the subspace described by the corresponding h' (i.e., $h'(x) = 1$ iff $g'(x)$ is the all-zero vector) is axis-parallel.[17] Hence, with high probability, Step 2 finds at least $k+1$ influential subsets and the execution proceeds to Step 3 with a matrix V of dimension $|I| \in [k+1, t+1]$. But in this case the density of $h^{-1}(1)$ is at least $|\mathcal{F}|^{-k^*} - 0.1\epsilon \geq 0.8|\mathcal{F}|^{-k} > 1.5|\mathcal{F}|^{-|I|}$, since $k^* \leq k \leq t \leq \log_{|\mathcal{F}|}(2/\epsilon)$ and $|I| > k$. Using Claim 4.12, we conclude that Step 3 rejects with high probability. ■

Conclusions. Proposition 5.3 implies that there exists a one-sided error tester of complexity $\widetilde{O}(1/\epsilon)$ for axis-parallel linear subsoaces, and thus for testing monomials. This establishes Theorem 1.3. Furthermore, a closer look at the proof reveals that in case of accepting the tester may indicate an upper bound on the dimension of the axis-parallel linear subspace; that is, it may always output $k \in [0, k^*]$ whenever h describes an $(n-k^*)$-dimensional subspace (and does so with high probability when h is close to describing such a subspace). Hence, we also obtain a tester for the class of linear spaces of dimension at least $n - k^*$. This establishes Theorem 1.2.

6 Lower Bounds for the Exact Versions

Here, we say that $f : \{0,1\}^n \to \{0,1\}$ is a k-monomial if f is the product of exactly k variables; that is, there exists a k-subset $I \subseteq [n]$ such that $f(x) = \prod_{i \in I} x_i$.

Theorem 6.1 (on the complexity of one-sided error testing k-monomials): *For every $k \geq 2$, every one-sided error tester for k-monomials has query complexity $\widetilde{\Omega}(\log(n-k))$, provided that the proximity parameter is smaller than 2^{-k}. Furthermore, this holds even if it is promised that the tested function is either a k-monomial or a $(k-1)$-monomial.*

Equivalently, the lower bound holds for testing $(n-k)$-dimensional axis-parallel linear subspaces (over GF(2)), even when guaranteed that the tested function describes an $(n-k')$-dimensional

[17]Recalling that g' is a linear function with range $\mathcal{F}^k$, it follows that if it depends on at most k variables, then it assumes the value 0 iff all these variables are set to 0.

axis-parallel linear subspace for $k' \in \{k-1, k\}$. Indeed, this establishes Theorem 1.1.

Proof: We prove the furthermore claim for $k = 2$, and it follows for general k by considering monomials that always contain the last $k-2$ variables. Moreover, we shall prove that $\widetilde{\Omega}(\log n)$ queries are required from any algorithm that satisfies the following two conditions.

1. The algorithm always accepts any 2-monomial.
2. When given oracle access to a random 1-monomial, the algorithm rejects with probability at least 1/2, where the probability is taken over the choice of the monomial.

Indeed, it suffices to consider deterministic algorithms, since one may consider the best possible random-choices of a randomized algorithm. To prove this claim, we fix an arbitrary (deterministic) algorithm that makes q queries and consider a random iterative process that selects a random 1-monomial on-the-fly, in response to the queries of the algorithm. We shall show that, with probability exceeding 1/2, after q queries the function is still undetermined and the answers received are also consistent with some 2-monomial. Since in such a case the algorithm must accept (per the one-sided error condition), it follows that it accepts a random 1-monomial with probability exceeding 1/2. This proves that the query complexity of an algorithm that satisfies the foregoing two conditions must exceed q.

In light of the above, we focus on describing and analyzing the aforementioned random iterative process. The process maintain a set of indices (of variables), denoted S, such that the corresponding 1-monomials are each consistent with the answers provided so far. Initially, $S = [n]$, and after q iterations (i.e., queries) we select an index uniformly in the current set S and consider the corresponding 1-monomial. We shall also show that if $|S| \geq 2$, then any 2-monomial with variables indices that reside in S is consistent with the answers obtained so far.

Construction 6.1.1 (an iteration of the process): *Suppose that the current set of indices equals S, and that the algorithm makes the query x. Let X be the set of 1-coordinates in x; that is, $X = \{i \in [n] : x_i = 1\}$. Then, with probability $\frac{|X \cap S|}{|S|}$, the process answers 1 and resets $S \leftarrow S \cap X$, and otherwise it answers 0 and resets $S \leftarrow S \setminus X$* (equiv., $S \leftarrow S \cap \{i \in [n] : x_i = 0\}$).

Evidently, the process always stops (after q iterations) with a non-empty set S. We call this set the output of the process. We now analyze this output.

Claim 6.1.2 (monomials consistent with the output of the process): *If the process outputs the set S, then any monomial that has all its variables in S is consistent with the answers provided by the process; that is, for any $I \subseteq S$, the monomial f_I defined by $f_I(x) = \prod_{i \in I} x_i$ is consistent with all answers.*

Proof: Fixing the random choices of the process, let S be its output and x a query made to it by the algorithm. If x was answered 1 (resp., answered 0), then $\{i \in [n] : x_i = 1\} \supseteq S$ (resp., $\{i \in [n] : x_i = 0\} \supseteq S$), and so $f_I(x) = 1$ (resp., $f_I(x) = 0$) for every $I \subseteq S$. ■

Claim 6.1.3 (the distribution of a random element in the set output by the process): *Suppose that the process outputs the set S and then we select i uniformly in S. Then, i is distributed uniformly in $[n]$.*

Proof: The execution of the process can be visualized as traversing a tree of depth q in which the paths represent possible sequences of answers, and the vertices represent the corresponding sets. The root corresponds to the set $[n]$, and if a vertex corresponds to the set S then its children correspond to the sets $S \cap X$ and $S \setminus X$, where X corresponds to the query made at this point. Now, observe that the vertices at each level of the tree correspond to a partition of $[n]$, and that a vertex that corresponds to the set S is reached with probability $|S|/n$. ■

Claim 6.1.4 (the size of the set output by the process): *For some $q = \Omega\left(\frac{\log n}{\log\log n}\right)$, with probability at least 2/3, the output of the process has size at least 2.*

Proof: Consider the sequence of sets $S_1, ..., S_q$ selected in the q steps of the process, where $S_0 = [n]$, and let $X(S_1, ..., S_{i-1})$ denote the i^{th} query made by the algorithm (when given the corresponding sequence of answers). Letting, $X = X(S_1, ..., S_{i-1})$, it holds that $S_i = S_{i-1} \cap X$ with probability $\frac{|S_{i-1} \cap X|}{|S_{i-1}|}$, and $S_i = S_{i-1} \setminus X$ otherwise. Our aim is to upper-bound the probability that S_q is a singleton. Hence, without loss of generality, we may assume that $X(S_1, ..., S_{i-1})$ only depends on S_{i-1} and that it can be replaced by a function of $|S_{i-1}|$ that merely specifies the relative sizes of $S_{i-1} \cap X$ and $S_{i-1} \setminus X$; that is, $p_i(|S_{i-1}|) = |S_{i-1} \cap X|/|S_{-i}|$.

In light of the above, it suffices to consider the sequence of random variables $\zeta_1, ..., \zeta_q$ that represent the sizes of the S_i's such that $\zeta_i = p_i(\zeta_{i-1}) \cdot \zeta_{i-1}$ with probability $p_i(\zeta_{i-1})$ and $\zeta_i = (1 - p_i(\zeta_{i-1})) \cdot \zeta_{i-1}$ otherwise. Letting $\zeta_0 = n$, our aim is to prove that $\mathbf{Pr}[\zeta_q = 1] \leq 1/3$, which we write as

$$\mathbf{Pr}\left[\prod_{i \in [q]} \frac{\zeta_{i-1}}{\zeta_i} \geq n\right] \leq 1/3. \tag{1}$$

Letting $\xi_i = \log_2(\zeta_{i-1}/\zeta_i) \geq 0$, we have $\xi_i = \log_2(1/p_i)$ with probability p_i and $\xi_i = \log_2(1/(1-p_i))$ otherwise, where we can think of p_i as adversarially chosen (given ξ_{i-1}). Clearly, $\mathbb{E}[\xi_i|\xi_1, ..., \xi_{i-1}] \leq 1$, and our aim is to show that

$$\mathbf{Pr}\left[\sum_{i \in [q]} \xi_i \geq n'\right] \leq 1/3, \tag{2}$$

where $n' = \log_2 n$. Note that the ξ_i's essentially satisfy the Martingale condition, so Eq. (2) would have held if the ξ's were bounded, alas they are not. We address this problem by using the fact that $\mathbf{Pr}[x_i > \log_2(4q)] < 1/4q$, defining $\xi'_i = \xi_i$ if $\xi_i \leq \log_2(4q)$ and $\xi'_i = 0$ otherwise, and using

$$\mathbf{Pr}\left[\sum_{i \in [q]} \xi_i \geq n'\right] \leq \mathbf{Pr}\left[\sum_{i \in [q]} \xi'_i \geq n'\right] + \frac{1}{4} \tag{3}$$

where the point is that $\xi'_i \in [0, \log_2(4q)]$. Lastly, recalling that $\mathbb{E}[\xi'_i] \leq 1$, assuming that $q = \Omega(n'/\log n')$ is small enough, and using the Martingale Tail Inequality, we upper-bound $\mathbf{Pr}\left[\sum_{i \in [q]} \xi'_i \geq n'\right]$ by $\exp(-\Omega((n'-q)/\log(4q))) = o(1)$, and the claim follows. ■

Conclusion: Combining the foregoing claims, we conclude that for some $q = \Omega((\log n)/\log\log n)$, no algorithm that makes q queries can always accept any 2-monomial, while rejecting a random 1-monomial with probability at least 1/2. The theorem follows. ■

References

[1] N. Alon, T. Kaufman, M. Krivelevich, S. Litsyn, and D. Ron. Testing Reed-Muller codes. *IEEE Transactions on Information Theory*, Vol. 51 (11), pages 4032–4039, 2005.

[2] M. Bellare, O. Goldreich and M. Sudan. Free Bits, PCPs and Non-Approximability – Towards Tight Results. *SIAM Journal on Computing*, Vol. 27, No. 3, pages 804–915, 1998. Extended abstract in *36th FOCS*, 1995.

[3] M. Blum, M. Luby and R. Rubinfeld. Self-Testing/Correcting with Applications to Numerical Problems. *Journal of Computer and System Science*, Vol. 47, No. 3, pages 549–595, 1993. Extended abstract in *22nd STOC*, 1990.

[4] Y. Filmus, N. Lifshitz, D. Minzer, and E. Mossel. AND Testing and Robust Judgement Aggregation. *arXiv:1911.00159* [cs.DM], 2019.

[5] O. Goldreich. *Computational Complexity: A Conceptual Perspective.* Cambridge University Press, 2008.

[6] O. Goldreich. *Introduction to Property Testing.* Cambridge University Press, 2017.

[7] O. Goldreich. Reducing Testing Affine Spaces to Testing Linearity of Functions. In *Computational Complexity and Property Testing* (O. Goldreich, Ed.), Springer, Lecture Notes in Computer Science (Vol. 12050), to appear. Preliminary version in *ECCC*, TR16-080, 2016.

[8] O. Goldreich and L.A. Levin. A Hard-Core Predicate for all One-Way Functions. In the proceedings of *21st ACM Symposium on the Theory of Computing*, pages 25–32, 1989.

[9] E. Haramaty, A. Shpilka, and M. Sudan. Optimal Testing of Multivariate Polynomials over Small Prime Fields. *SIAM Journal on Computing*, Vol. 42, No. 2, pages 536–562, 2013.

[10] M. Parnas, D. Ron, and A. Samorodnitsky. Testing Basic Boolean Formulae. *SIAM Journal on Disc. Math. and Alg.*, Vol. 16 (1), pages 20–46, 2002.

Appendices

A.1 Proof of Claim 4.3

Recall that Claim 4.3 asserts that *for H, V and $g = g_{H,V}$ as in Definition 4.2, it holds that H is an $(n-k)$-dimensional linear subspace if and only if g is a linear function with image $\mathcal{F}^k$.*

Proof: Recall that $g^{-1}(0^k) \subseteq H$ always holds. Furthermore, equality (i.e., $g^{-1}(0^k) = H$) holds if g never assumes the value $\perp$, since in this case $x + cV \in H$ implies that $g(x) = c$ (and so $x \in H$ implies $g(x) = 0^n$).

Now, on the one hand, if g is a linear function with image $\mathcal{F}^k$ (i.e., $g(x) = xT$ for some full-rank n-by-k matrix T), then $H = g^{-1}(0^k)$ (i.e., $H = \{x \in \mathcal{F}^n : xT = 0^k\}$), which implies that H is an $(n-k)$-dimensional linear subspace (since $H = \{yG : y \in \mathcal{F}^{n-k}\}$ for any G that is a basis of the subspace orthogonal to $T^\top$).[18]

On the other hand, if H is an $(n-k)$-dimensional linear subspace, then, for some full-rank $(n-k)$-by-n matrix G, it holds that $H = \{yG : y \in \mathcal{F}^{n-k}\}$. In this case, for every $x \in \mathcal{F}^n$ there exists a *unique* representation of x as $yG - cV$, since V is a basis for a k-dimensional linear subspace that complements the $(n-k)$-dimensional linear subspace H, Hence, for every $x \in \mathcal{F}^n$, there exists a unique $(c, y) \in \mathcal{F}^k \times \mathcal{F}^{n-k}$ such that $x + cV = yG \in H$, and $g(x) = c$ follows. We now observe that the image of g equals $\mathcal{F}^k$, since $g(0^n - cV) = c$ for every $c \in \mathcal{F}^k$, and that g is linear, since for every $x = yG - cV$ and $x' = y'G - c'V$ in $\mathcal{F}^n$, it holds that $g(x) + g(x') = c + c'$ and $c + c' = g(y''G - (c + c')V)$ holds for every $y'' \in \mathcal{F}^{n-k}$ (and in particular for $y'' = y + y'$, which implies that $c + c' = g(x + x')$). ■

A.2 Proof of Lemma 3.2

The following self-correction procedure, denoted $\mathsf{SelfCorrectPoly}^f(x)$, is presented in the analysis of tester underlying [1, Thm. 1]. On input $x \in \{0,1\}^n$ and access to a function $f : \{0,1\}^n \to \{0,1\}$, the procedure selects uniformly at random vectors $y_1, \ldots, y_k \in \{0,1\}^n$ and outputs the sum of all vectors in the span of $x, y_1, ..., y_k$ except for x and the empty linear combination; that is, the output is

$$\sum_{\emptyset \neq I \subseteq \{0,1,\ldots,n\} : I \neq \{0\}} f\left(\sum_{i \in I} y_i\right) \tag{4}$$

where $y_0 = x$. Note that each of the $2^{k+1} - 2$ inputs to f (i.e., $\sum_{i \in I} y_i$ for each non-empty $I \neq \{0\}$) is uniformly distributed in $\{0,1\}^n$.

Claim A.1 (Lemma 1 in [1]): *If $f \in \mathcal{P}_{\leq k}$, then* Eq. (4) *equals $f(y_0)$ for every $y_0, y_1, \ldots, y_k \in \{0,1\}^n$.*

This establishes Part 2 of Lemma 3.2. We establish Part 1 by observing that is f is δ-close to f' then for every x it holds that $\mathsf{SelfCorrectPoly}^f(x)$ is $(2^{k+1} - 2) \cdot \delta$-close to $\mathsf{SelfCorrectPoly}^{f'}(x)$. It

[18] Alternatively, if $g(x + x') = g(x) + g(x')$ for every $x, x' \in \mathcal{F}^n$, then $x, x' \in H$ implies $x + x' \in H$ (for every $x, x' \in \mathcal{F}^n$), since $g(x) = g(x') = 0^k$ implies $g(x + x') = 0^k$. Hence, $H = g^{-1}(0^k)$ is a linear subspace. Lastly, we note that this subspace has dimension $n - k$, since the image of g equals $\mathcal{F}^k$ and $|g^{-1}(0^k)| = |g^{-1}(c)|$ holds for every $c \in \mathcal{F}^k$.

follows that if f is 2^{-k-3}-close to $g \in \mathcal{P}_{\leq k}$, then for every x we have

$$\begin{aligned}
\mathbf{Pr}[\mathsf{SelfCorrectPoly}^f(x) = g(x)] &\geq \mathbf{Pr}[\mathsf{SelfCorrectPoly}^g(x) = g(x)] - (2^{k+1} - 2) \cdot 2^{-k-3} \\
&= 1 - (2^{k+1} - 2) \cdot 2^{-k-3}.
\end{aligned}$$

This establishes Part 1 of Lemma 3.2.