Testing Sparse Functions over the Reals

Vipul Arora*[1] Arnab Bhattacharyya[2] Philips George John[3] Sayantan Sen[3]

[1]UC Berkeley. vipularora@berkeley.edu.
[2]University of Warwick. arnab.bhattacharyya@warwick.ac.uk.
[3]CNRS-CREATE & NUS. philips.george.john@u.nus.edu.
[4]Centre for Quantum Technologies, National University of Singapore. sayantan789@gmail.com.

Abstract

Over the last three decades, function testing has been extensively studied over Boolean, finite fields, and discrete settings. However, to encode the real-world applications more succinctly, function testing over the reals (where the domain and range, both are reals) is of prime importance. Recently, there have been some works in the direction of testing for algebraic representations of such functions: the work by Fleming and Yoshida (ITCS 20), Arora, Kelman, and Meir (SOSA 25) on *linearity testing* and the work of Arora, Bhattacharyya, Fleming, Kelman, and Yoshida (SODA 23) for testing *low-degree polynomials*. Our work follows the same avenue, wherein we study three well-studied sparse representations of functions, over the reals, namely (i) *k-linearity*, (ii) *k-sparse polynomials*, and (iii) *k-junta*.

In this setting, given approximate query access to some $f : \mathbb{R}^n \to \mathbb{R}$, we want to decide if the function satisfies some property of interest, or if it is *far* from all functions that satisfy the property. Here, the distance is measured in the ℓ_1-metric, under the assumption that we are drawing samples from the Standard Gaussian distribution. We present efficient testers and $\Omega(k)$ lower bounds for testing each of these three properties.

1 Introduction

Property testing [BLR90, RS96, GGR96] is a rigorous framework of studying the global properties of large datasets by accessing only a few entries of it. In particular, given query access to an unknown "huge object", our goal is to understand some property of the object by only inspecting a few entries of it. Formally, we can define property testing by considering the data set as a function f over the underlying domain. For example, consider the problem of *linearity testing* of functions $f : \mathbb{F}^n \to \mathbb{F}$, where $\mathbb{F}$ is a finite field. Here, the f is given via a query oracle, i.e., on input $\boldsymbol{x}$, the oracle returns $f(\boldsymbol{x})$. The goal is to distinguish with high probability if f is a linear function, or f is "far" from all linear functions, by performing as few queries to the oracle as possible. The field of property testing was initiated in the work of [BLR90, BLR93], who studied the problem of *self-testing* of programs, where the goal was to understand the correctness of a program by verifying its outputs on a set of correlated input data. A fundamental problem they studied is that of testing if f is *linear* (or generally, a homomorphism of abelian groups), with the notion of ε-farness of $f : \mathbb{F}^n \to \mathbb{F}$ from function class $\mathcal{P}$ defined as:

$$\mathsf{dist}_{\mathsf{Unif}(\mathbb{F}^n),\ell_0}(f, \mathcal{P}) = \inf_{g \in \mathcal{P}} \Pr_{\boldsymbol{x} \sim \mathsf{Unif}(\mathbb{F}^n)}[f(\boldsymbol{x}) \neq g(\boldsymbol{x})] \geq \varepsilon,$$

*Supported in part by NRF-AI Fellowship R-252-100-B13-281, and the Simons Institute.

i.e., for each function g that satisfies property $\mathcal{P}$, f disagrees with g on at least an ε-fraction of the inputs $\boldsymbol{x}$ (uniformly drawn from $\mathbb{F}^n$). Hence this is an ℓ_0-distance w.r.t. the notion of exact agreement. In the context of linearity testing, $\mathcal{P}$ denotes the class of linear functions/homomorphisms (the ℓ_0-distance definition given above can be easily generalized in terms of function domain and range, input distribution etc). They showed a constant query (independent of $|\mathbb{F}|$ and n) linearity tester, eponymously known as the *BLR tester*.

Over the last three decades, property testing has been extensively studied in many settings, such as when the unknown object is a function, or a graph, or probability distributions, with many natural connections to real-world problems, e.g., in the context of probabilistically checkable proofs [ALM+98, AS98, RS97, Din07], PAC learning [GGR96], program checking [RS92, RS96], approximation algorithms [GGR96, FGL+91] and many more.

In particular, starting from the very first work that initiated the field of property testing [BLR90], over the last three decades, testing various properties of functions has been extensively studied in several settings, culminating in a wide array of tools and techniques. The study of function property testing has been extended beyond the Boolean setting, such as when the function is defined over finite fields [AKK+05, KR06, JPRZ09, FS95, BKS+10, BFH+13, GOS+11, Sam07], over hyper-grids [DGL+99, BRY14b, CS13, BCP+17], etc. One may see the books [Gol17, BY22] and the surveys [Fis04, R+08, Ron09] for detailed references.

It turns out that, in many practical settings, the functions are defined over continuous domain and are real-valued. Here, we are given query access to an unknown function $f : \mathbb{R}^n \to \mathbb{R}$ via an oracle and our goal is to distinguish if f satisfies some property $\mathcal{P}$, or is ε-far from all functions that satisfy $\mathcal{P}$. We define the notion of ε-farness in the following way: we fix a reference distribution $\mathcal{D}$ supported over $\mathbb{R}^n$, and we say f is ε-far[1] from $\mathcal{P}$ if the following holds:

$$\mathsf{dist}_{\mathcal{D},\ell_1}(f, \mathcal{P}) \triangleq \inf_{g\in\mathcal{P}} \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{D}} [|f(\boldsymbol{x}) - g(\boldsymbol{x})|] \geq \varepsilon.$$

Studying function testing in this setting often requires new techniques from the Boolean and finite field counterparts. One of the common choices of the reference distribution is the standard Gaussian distribution $\mathcal{N}(\mathbf{0}, I_n)$. Although the problem of function testing over the reals has several practical motivations, there have been comparatively few works in this setting till now compared to the Boolean and finite field counterparts such as testing surface areas [Nee14, KNOW14], testing halfspaces [MORS10a, MORS10b, MORS09], linear separators [BBBY12], high-dimensional convexity [CFSS17], linear k-juntas [DMN19] etc. Interestingly, these works mostly focus on the setting when f is Boolean valued, that is, $f : \mathbb{R}^n \to \{\pm 1\}$.

The setting where the range of f is real, has also been studied, e.g., in the work of [BRY14a], wherein they test properties of functions, defined over finite hyper-grids, with respect to L_p-distances. In [BKR23], functions over the hypercube, i.e., $f : \{0,1\}^n \to \mathbb{R}$ are studied for monotonicity. Recently, [FPJ23, FPJ24] studied L^p testing of monotonicity of Lipschitz functions $f : [0,1]^n \to \mathbb{R}$.

[FY20] studied the problem of linearity testing in full generality, i.e., for functions $f : \mathbb{R}^n \to \mathbb{R}$. Later, [ABF+23] studied the problem of testing low-degree polynomials in this model, followed by the work of [AKM25], which improved these results by achieving query-optimality with respect to the proximity parameter ε. In this work, we make significant progress in testing various fundamental notions of sparsity in this model, which have been extensively studied in other models.

When the function f to be tested is real valued, there are two different models that can be considered. In the first model, known as the *arbitrary precision arithmetic* model, or the *exact testing* model, we assume that the oracle representing f can give the exact value of f at any point of

[1]The notion of ℓ_1-distance is more useful in our setting, especially in the approximate query setting, which we define later. For example, consider the case if $g = f + \varepsilon$ for some $\varepsilon > 0$, then $\mathsf{dist}_{\mathcal{D},\ell_1}(f, g) = \varepsilon$, whereas $\mathsf{dist}_{\mathcal{D},\ell_0}(f, g) = 1$.

query $\boldsymbol{x}$. However, from an implementation viewpoint, this can't be achieved in practice. So, a *finite precision arithmetic model* is studied instead:

Definition 1.1 (η-*approximate query*)**.** The oracle, when queried for $f(\boldsymbol{x})$, outputs $\widetilde{f}(\boldsymbol{x})$ such that $|\widetilde{f}(\boldsymbol{x}) - f(\boldsymbol{x})| \leq \eta$, for some small parameter $\eta \in (0,1)$, for every query point $\boldsymbol{x}$.

It is clear that for any property $\mathcal{P}$, any tester for $\mathcal{P}$ in the approximate model will also work in the exact model. The notion of approximate testing was studied in earlier works [GLR+91, ABCG93, EKR01]. In this context, η can be thought of as the resolution limit of the computational machine, i.e., if the machine offers some α bits of precision, then $\eta = 2^{-\alpha}$. η can also be thought of the noise reliability threshold of a channel communicating reals, i.e., if some information $a \in \mathbb{R}$ is transmitted on a channel with a reliability threshold of η, then the received observable $\widetilde{a} \in \mathbb{R}$ satisfies $|a - \widetilde{a}| \leq \eta$. When $\eta = 0$, this is the exact query model. Our results require appropriate upper bounds on η, which will be stated in each context. All our upper bounds in this work are in the approximate query model. We use the exact query model for proving the lower bounds, since lower bounds for $\eta = 0$ also hold for any η-approximate query model for $\eta > 0$.

Similar to the context of access to the unknown functions, there are some variations in terms of the error profile of the testing algorithms. A tester is said to have *two-sided error* if it can err in both the cases: when $f \in \mathcal{P}$, and when f is ε-far from $\mathcal{P}$. This is in contrast with *one-sided error* testers, which always decides correctly when $f \in \mathcal{P}$, and can only err when f is ε-far from $\mathcal{P}$. Similarly, a tester is said to be *adaptive* if it performs queries based on the answers it obtained in the previous queries. On the other hand, a *non-adaptive* tester performs all its queries together in a single round.

In this work, we design testers in the approximate query model for three sparse function representations: k-linear, k-sparse low-degree polynomials, and k-juntas. Our testers for k-linearity, and k-sparse low-degree polynomials have two-sided error, whereas our k-junta tester has one-sided error. Our testers for k-linearity and k-junta are adaptive, while our tester for k-sparse low-degree polynomials is non-adaptive. Although these properties have been very well studied in the Boolean and finite fields regime, this work strives to do the same over continuous domains. We believe the new techniques we have developed to design these testers will be of independent interest.

1.1 Our results

In this work, we mainly focus on three problems: testing (i) k-linearity, (ii) k-sparse polynomials, and (iii) k-juntas. Throughout this work, we assume that our reference distribution $\mathcal{D}$ is the standard Gaussian $\mathcal{N}(\mathbf{0}, I_n)$, unless otherwise stated.

1.1.1 Testing k-linear functions

Definition 1.2 (k-linearity)**.** Let $f : \mathbb{R}^n \to \mathbb{R}$, and $k \in \mathbb{N}$ be a parameter. f is a k-*linear* function if there exists a set $S \subseteq [n] : |S| \leq k$, and there exist $\{c_i \in \mathbb{R} : i \in S\}$, such that for any $\boldsymbol{x} \in \mathbb{R}^n$,

$$f(\boldsymbol{x}) = \sum_{i \in S} c_i \boldsymbol{x}_i.$$

This problem has been extensively studied over finite domains, with exact queries. [FKR+04] designed the first tester for k-linearity with query complexity $\widetilde{O}(k^2)$ by studying the related problem of testing k-juntas. Later, [Bla09] improved the bound for testing k-juntas to $O(k \log k)$ queries. Using the famous BLR test [BLR90], along with this result gives a tester for k-linearity with $O(k \log k)$ query complexity. The first lower bounds for this problem were presented by [FKR+04], proving $\Omega(\sqrt{k})$ non-adaptive, and $\Omega(\log k)$ adaptive queries are necessary for testing k-linearity. These were first improved by [Gol10], to $\Omega(k)$ non-adaptive, and $\Omega(\sqrt{k})$ adaptive query lower bounds. This was further improved by [BBM12, BK12] who proved $\Omega(k)$ adaptive query lower bound. Interestingly, the $\Omega(k)$ adaptive query lower bound by [BBM12] was proved by showing a novel connection to communication complexity. Recently, [Bsh23] presented an optimal, two-sided error, non-adaptive algorithm for this problem that requires $O(k \log k)$ queries. Our result for testing k-linearity is summarized in the following theorem.

Theorem 1.3 (Informal, see Theorem 4.1). *Let $k \in \mathbb{N}$, $f : \mathbb{R}^n \to \mathbb{R}$ be given via an η-approximate query access, and $\varepsilon, \eta \in (0, 2/3)$ be such that $\eta < \min\left\{\varepsilon, O\left(\min_{i\in[n]:f(\boldsymbol{e}_i)\neq 0} \frac{|f(\boldsymbol{e}_i)|}{(nk)^2}\right)\right\}$, where $\boldsymbol{e}_i$ denotes the i^{th} standard unit vector. There exists an $\widetilde{O}(k \log k + 1/\varepsilon)$-query tester (Algorithm 1) that distinguishes if f is k-linear, or is ε-far from all k-linear functions, with probability at least $2/3$.*

1.1.2 Testing k-sparse low-degree polynomials

Definition 1.4 (k-sparsity). Let $f : \mathbb{R}^n \to \mathbb{R}$, and $k \in \mathbb{N}$ be a parameter. Define $\boldsymbol{x}^{\boldsymbol{\alpha}} \triangleq \prod_{i=1}^n x_i^{\alpha_i}$, for any $\boldsymbol{x} \triangleq (x_1, \ldots, x_n) \in \mathbb{R}^n$, and $\boldsymbol{\alpha} \triangleq (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n$. A polynomial $f(x_1, \ldots, x_n) = \sum_{i=1}^{\ell} a_i \boldsymbol{x}^{\boldsymbol{d}_i}$, with $a_i \neq 0$, and $\boldsymbol{d}_i \in \mathbb{N}^n$ for every $i \in [\ell]$, is said to be a *k-sparse* polynomial if $\ell \leq k$.

Grigorescu *et al.* [GJR10] studied this problem for functions on finite fields, i.e., $f : \mathbb{F}_q^n \to \mathbb{F}_q$, assuming that q is large enough. Using the machinery of Hankel matrices (Definition 3.13) associated with a polynomial, from Ben-Or and Tiwari [BOT88], they designed a *local* tester with a query complexity of $O(k)$ (independent of d), assuming f to be an individual-degree-d polynomial (note that all functions $\mathbb{F}_q^n \to \mathbb{F}_q$ are polynomials of individual degree $\leq q$).

The assumption of polynomiality can no longer be assumed for functions over continuous domains. Hence, a preliminary step is needed to weed out functions that are far from being low-degree polynomials. Recently [ABF+23] designed a *local* low-degree tester with a query complexity of $O(d^5)$, for polynomials over $\mathbb{R}^n$ with *total degree* d, which we will use in our work. Thus, we restrict our attention to polynomials of *total degree* at most d. Our result for testing k-sparse low-degree polynomials is stated below.

Theorem 1.5 (Informal, see Theorem 5.1). *Let $k \in \mathbb{N}$, $\varepsilon, \eta \in (0, 1)$ be parameters such that $\eta < \min\{\varepsilon, 1/2^{2^n}\}$, and $f : \mathbb{R}^n \to \mathbb{R}$ be given via an η-approximate query access. Then there exists an $\widetilde{O}(d^5 + d^2/\varepsilon + dk^3)$-query tester (Algorithm 4), that distinguishes if f is a k-sparse, degree-d polynomial, or is ε-far from any such polynomial, with probability at least $2/3$.*

A critical ingredient in proving this result is a probabilistic upper bound on the maximum singular value of Hankel matrices associated with sparse, low-degree polynomials (Definition 5.4). This result may be of independent interest, and is proved in Section 5.2.

Theorem 1.6 (Probabilistic Upper Bound on $\sigma_{\max}$). *Let $f : \mathbb{R}^n \to \mathbb{R}$, $f(\boldsymbol{x}) = \sum_{i=1}^k a_i M_i(\boldsymbol{x})$ be a k-sparse, degree-d polynomial, where M_i's are its non-zero monomials, and $\sigma_{\max}(H_t(f, \boldsymbol{u}))$ denote the largest singular value of the t-dimensional Hankel matrix associated with f at a point $u \in \mathbb{R}^n$, $H_t(f, \boldsymbol{u})$. Then, for any $\gamma \in (0, 1)$, with $\boldsymbol{a} \triangleq (a_1, \cdots, a_k)^\top \in \mathbb{R}^k$,*

$$\Pr_{\boldsymbol{u}\sim\mathcal{N}(\boldsymbol{0}, I_n)}\left[\sigma_{\max}(H_t(f, \boldsymbol{u})) \geq \|\boldsymbol{a}\|_2^2 \left(2^{d/2}\lceil d/2\rceil! + \sqrt{\frac{k}{\gamma}} 2^{d/2}\sqrt{d!}\right)^{2t}\right] \leq \gamma.$$

Remark 1.7. *We note that* Test-k-Sparse *(Algorithm 4) also works for testing k-linear functions (by setting the degree $d = 1$). However, in this context, we make the following remarks.*

(i) *Since the query complexity of* Test-k-Sparse *is* $\widetilde{O}(d^5 + d^2/\varepsilon + dk^3)$, *if we invoke it for k-linearity testing, the bound would be* $\widetilde{O}(k^3 + 1/\varepsilon)$ *which is worse than that of* Test-k-Linear. *On the other hand,* Test-k-Sparse *works for arbitrary d, which significantly generalizes the set of functions that can be efficiently tested.*

(ii) *The dependence on the approximation parameter η is inverse doubly exponential in n for* Test-k-Sparse, *as compared to inverse in $(nk)^2$ for* Test-k-Linear. *Thus, for the wide range of parameters when η is not too small, applying* Test-k-Linear *will have provable guarantees, compared to* Test-k-Sparse.

(iii) *Another salient difference is that* Test-k-Linear *is an adaptive tester, as opposed to* Test-k-Sparse, *which is non-adaptive.*

1.1.3 Testing k-juntas

Definition 1.8 (k-junta)**.** Let $f : \mathbb{R}^n \to \mathbb{R}$, and $k \in \mathbb{N}$ be a parameter. A coordinate $i \in [n]$ is said be *influential* with respect to f, if for some $\boldsymbol{x} \in \mathbb{R}^n$, changing the value of $\boldsymbol{x}_i$ changes the value of $f(\boldsymbol{x})$. f is said to be a *k-junta*, if there are at most k influential variables with respect to f.

Testing whether a Boolean function is a k-junta has been extensively studied in the exact query model. The first result in this context was by [PRS02], which was followed by the work of [FKR+04], who designed a $\widetilde{O}(k^2)$-query tester. Later, [DLM+07] extended it to the finite range setting. [Bla08] then gave an $\widetilde{O}(k^{3/2})$-query non-adaptive tester for this problem, while for adaptive testers, [Bla09] showed $\widetilde{O}(k \log k + k/\varepsilon)$ queries suffice. It is important to note that all these results use Fourier-analytic techniques. Notably, [BWY15] designed a new algorithm for testing k-juntas with similar optimal bounds in the context of testing partial isomorphism of functions. Interestingly, this work deviates from the common Fourier analytic approach and instead presents a combinatorial approach to this problem. This algorithm from [BWY15] will be used in designing our tester for k-juntas. We would like to note that recently [DMN19] studied the linear k-junta testing problem, where the function f is defined as $f : \mathbb{R}^n \to \{+1, -1\}$. This is different from our setting of real-valued functions. It is not clear to us if their techniques can be generalized to our setting.

In terms of lower bounds, [FKR+04] showed a lower bound of $\Omega(\sqrt{k})$ queries for non-adaptive testers, which was improved to $\widetilde{\Omega}(k^{3/2}/\varepsilon)$ by [CST+18]. For adaptive testers, [CG04] showed an $\Omega(k)$ lower bound, which was then improved to $\Omega(k \log k)$ by [Sag18].

Our result for testing k-juntas is summarized in the following theorem.

Theorem 1.9 (Informal, see Theorem 6.1)**.** *Let $k \in \mathbb{N}$, and $\varepsilon, \eta \in (0, 1)$ be parameters such that $\eta < \min\{O(\varepsilon/k^2), O(1/k^2 \log^2 k)\}$, and $f : \mathbb{R}^n \to \mathbb{R}$ be given via an η-approximate query access. There exists a one-sided error $\widetilde{O}\left(\frac{k \log k}{\varepsilon}\right)$-query tester (Algorithm 6), that distinguishes if f is a k-junta, or is ε-far from all k-juntas, with probability at least $2/3$.*

1.1.4 Lower bounds

Now we briefly mention our lower bound results, which hold even for adaptive testers. For all these three properties (k-linearity, k-sparse degree-d polynomials, and k-juntas), we prove lower bounds of $\Omega(\max\{k, 1/\varepsilon\})$ queries. Additionally, for k-sparse degree-d polynomials, we prove a lower bound of $\Omega(\max\{k, d, 1/\varepsilon\})$ queries. All our lower bounds follow from the general reduction from communication complexity, introduced by [BBM12], coupled with some folklore results. We use Set-Disjointness as the hard instance to prove our lower bounds.

Theorem 1.10. *Given exact query access to $f : \mathbb{R}^n \to \mathbb{R}$, some $k, d \in \mathbb{N}$ and a distance parameter $\varepsilon \in (0,1)$, $\Omega(\max\{k, 1/\varepsilon\})$ queries are necessary for testing the following properties with probability at least $2/3$:*

(i) k-linearity.

(ii) k-junta.

(iii) k-sparse degree-d polynomials.

Moreover, for testing k-sparse degree-d polynomial, the lower bound is improved to $\Omega(\max\{d, k, 1/\varepsilon\})$.

Discussion

In this work, we systematically study the problem of testing sparse real-valued functions given via approximate queries, over continuous domain. We study three well-studied properties: (i) k-linearity, (ii) k-sparse, low-degree polynomials, and (iii) k-juntas, and design efficient testers for them. Our work opens directions to several interesting questions.

We note that our results have constraints on the approximate query parameter η. The first open question is whether these can be improved. Furthermore, our k-sparse degree-d polynomial tester performs $\widetilde{O}(d^5 + d^2/\varepsilon + dk^3)$ queries, and our lower bound for this problem is $\Omega(\max\{d, k, 1/\varepsilon\})$. The second open question is whether the gaps in these bounds (w.r.t. k, and d) can be improved, e.g., by assuming additional structure on the underlying function, like Lipschitzness, etc.

Another interesting direction is to design tolerant testers [PRR06] for these properties. This is different from the approximate query testing notion, as in tolerant testing, the decision boundary is expanded to require that functions that are sufficiently close to the property are also accepted with high probability, whereas in the approximate query model, we accept functions $\widetilde{f}$ such that $\mathsf{dist}_{\mathcal{D},\infty}(f, \widetilde{f}) \leq \eta$ (pointwise η-close) which is a stronger constraint compared to the expected ℓ_1-distance $\mathsf{dist}_{\mathcal{D},\ell_1}(f, g)$ that we use between functions.

Finally, we use the standard Gaussian distribution as the reference distribution. It would be interesting to see if our results can be extended to other concentrated distributions as well.

A comparison of our upper bounds for the three problems, with the corresponding restrictions, is provided in Table 1.

Problem	Upper Bound	Restriction
k-linearity	$\widetilde{O}(k \log k + 1/\varepsilon)$	$\eta < \min\left\{\varepsilon, O\left(\min_{i\in[n]:f(\boldsymbol{e}_i)\neq 0} \frac{\lvert f(\boldsymbol{e}_i)\rvert}{(nk)^2}\right)\right\}$
k-sparsity	$\widetilde{O}(d^5 + d^2/\varepsilon + dk^3)$	$\eta < \min\{\varepsilon, 1/2^{2^n}\}$
k-junta	$\widetilde{O}\left(\frac{k \log k}{\varepsilon}\right)$	$\eta < \min\{O(\varepsilon/k^2), O(1/k^2 \log^2 k)\}$

Table 1: A comparison of the upper bounds, as well as the restrictions, for the three sparse representation testing problems. The upper bounds, and the restrictions in the three rows follow sequentially from Theorem 1.3, Theorem 1.5, and Theorem 1.9, respectively.

Organization of the paper

The rest of the paper is organized as follows: In Section 2, we present an overview of our results and techniques, followed by a discussion of the preliminaries required, in Section 3. In Section 4, we present our k-linearity tester, which is followed by our tester for k-sparse low-degree polynomials in Section 5, and our k-junta tester in Section 6. Finally, in Section 7, we present the lower bounds. Some associated proofs and subroutines from prior works are moved to the appendix for brevity.

2 Overview of our results

We present a brief overview of our techniques, starting with k-linearity testing.

Testing k-linearity We build upon the *self-correct and test* approach of [BLR90, HK07]. Instead of directly testing if f is k-linear, we construct a function $g_{\text{self-correct}}$ such that if f is k-linear, then $g_{\text{self-correct}}$ will also be k-linear. Moreover, we simulate queries to $g_{\text{self-correct}}$ using queries to f, and test this newly constructed function $g_{\text{self-correct}}$.

We use the Gaussian distribution as the reference distribution, since there are no uniform distributions over continuous domains with infinite support. (This deviates from the self-correction approach of [BLR90].) In particular, we evaluate f on a set of points sampled from $\mathcal{N}(\mathbf{0}, I_n)$ to construct the self-corrected function $g_{\text{self-correct}}$. To deal with the fact that different points from $\mathcal{N}(\mathbf{0}, I_n)$ have different probability masses, the idea is to radially project the sampled points from $\mathcal{N}(\mathbf{0}, I_n)$ into a small Euclidean ball $\mathrm{B}(\mathbf{0}, r)$ of a small (constant) radius ($r = 1/50$ suffices) such that the probability masses of all points sampled from that ball is roughly the same. Moreover, since we work with an approximate oracle, we use the following notion of the self-corrected function,

$$g(\boldsymbol{p}) \triangleq \kappa_{\boldsymbol{p}} \cdot \underset{\boldsymbol{x} \sim \mathcal{N}(\mathbf{0}, I_n)}{\mathsf{med}} \left[g_{\boldsymbol{x}} \left(\frac{\boldsymbol{p}}{\kappa_{\boldsymbol{p}}} \right) \right] = \kappa_{\boldsymbol{p}} \cdot \underset{\boldsymbol{x} \sim \mathcal{N}(\mathbf{0}, I_n)}{\mathsf{med}} \left[f \left(\frac{\boldsymbol{p}}{\kappa_{\boldsymbol{p}}} - \boldsymbol{x} \right) + f(\boldsymbol{x}) \right]$$

where $\kappa_{\boldsymbol{p}} : \mathbb{R}^n \to \mathbb{R}$ is defined as $\kappa_{\boldsymbol{p}} \triangleq \begin{cases} 1 & \text{, if } \|\boldsymbol{p}\|_2 \le r \\ \lceil \|\boldsymbol{p}\|_2 / r \rceil & \text{, if } \|\boldsymbol{p}\|_2 > r \end{cases}$, so that $\boldsymbol{p}/\kappa_{\boldsymbol{p}} \in \mathrm{B}(\mathbf{0}, r)$. This definition of self-correction function was used in [FY20, ABF+23, AKM25].

To test k-linearity (Algorithm 1), we first test if f is pointwise close to some additive (aka. linear) function using Approximate Additivity Tester (Algorithm 7). If it rejects f, we also reject f. However, if Algorithm 7 does not reject, then the self-corrected function g is pointwise close to some linear function. As we only have access to an approximate oracle access to f, we can't simulate g exactly. So, we use Approximate-g, the approximate query oracle for g (part of Algorithm 8). The work of [ABF+23, AKM25] proves: (i) g and Approximate-g are pointwise close in $\mathrm{B}(\mathbf{0}, r)$, and (ii) f and Approximate-g are also pointwise closeness. So, f is pointwise close to Approximate-g.

Now we partition the n-variables $[n]$ into k^2 buckets uniformly at random, and test if the number of buckets with influential variables (variables whose values determine the value of f) is at most k. As a result of the partition, if f is k-linear, the influential variables will be in different buckets, and we can then easily detect them. This idea is formalized in the subroutines FindInfBucket (Algorithm 2) and FindInfBuckets (Algorithm 3), which are used to estimate the number of influential buckets.

Testing k-sparsity Our k-sparse, degree-d polynomial tester (Algorithm 4) adopts a similar approach. We first test if f is a low-degree polynomial, using the ApproxLowDegreeTester (Algorithm 9) from [ABF+23, AKM25]. If it rejects f, we also reject f. However, if Algorithm 9 doesn't reject f, then f is pointwise close to a low-degree polynomial.

As in k-linearity testing, we use a self-corrected function g from [ABF+23, AKM25]: For points $\boldsymbol{p} \in \mathrm{B}(\mathbf{0}, r)$, $g(\boldsymbol{p})$ is the (weighted) median value of $g_{\boldsymbol{q}}(\boldsymbol{p}) \triangleq \sum_{i=1}^{d+1} (-1)^{i+1} \binom{d+1}{i} f(\boldsymbol{p} + i\boldsymbol{q})$, weighted according to the probability of $\boldsymbol{q} \sim \mathcal{N}(\mathbf{0}, I_n)$, i.e.,

$$g(\boldsymbol{p}) \triangleq \underset{\boldsymbol{q} \sim \mathcal{N}(\mathbf{0}, I_n)}{\mathsf{med}} [g_{\boldsymbol{q}}(\boldsymbol{p})].$$

Intuitively, $g_{\boldsymbol{q}}(\boldsymbol{p})$ is the value that f should take, if, when restricted to the line $\mathrm{L}_{\boldsymbol{p},\boldsymbol{q}} \triangleq \{\boldsymbol{p} + t\boldsymbol{q}, t \in \mathbb{R}\}$, f would be a degree-d univariate polynomial. Taking the weighted median over all directions

$\boldsymbol{q} \sim \mathcal{N}(\mathbf{0}, I_n)$, ensures that the self-correction proportionately respects the values of f, in a local neighborhood of $\boldsymbol{p}$. For $\boldsymbol{p} \notin \mathrm{B}(\mathbf{0}, r), g$ is defined via radial extrapolation from within $\mathrm{B}(\mathbf{0}, r)$ along the radial line $\mathrm{L}_{\mathbf{0},\boldsymbol{p}}$.

Given exact query access to f, we can simulate query access to the self-corrected function g. As we only have approximate query access to f, we use ApproxQuery-g (Algorithm 10, the approximate oracle to the self-corrected function associated with f), which was proved to be pointwise close to g. As a result, f will be pointwise close to ApproxQuery-g as well.

Once we have that f is close to a low-degree polynomial, we use a Hankel matrix (Definition 3.13) based characterization for sparse polynomials: [GJR10, BOT88] proved a polynomial is k-sparse, if and only if its associated Hankel matrix has a non-zero determinant. This can be efficiently tested with only $2k + 1$ queries to f. Unfortunately, since we only have approximate query access to f, this technique no longer works, as determinants of sums of matrices do not behave nicely. So we take a probabilistic approach, showing that the *noisy* Hankel matrix constructed from the approximate query to f is not too far from the exact Hankel matrix (proved in Observation 5.7). Finally, we show that if f is a k-sparse, low-degree polynomial, then the smallest eigenvalue of the *noisy* Hankel matrix associated with ApproxQuery-g is not too large. We prove this in Theorem 5.8 using Weyl's inequality (Theorem 3.15). Combining them all, our main result is proved in Section 5.3.

Testing k-junta Finally, we discuss our algorithm for testing k-juntas (Algorithm 6). Our approach is to first randomly partition the n-variables into k^2 buckets. If f is a k-junta, the k influential variables will be separated into distinct buckets (by birthday paradox).

Now we run $O(k/\varepsilon)$ iterations to find if there exists any influential variable in any bucket. To find the influential variables, we use the subroutine FindInfBucket (Algorithm 2). In these $O(k/\varepsilon)$ iterations, if we find more than k influential variables in f, we reject it. Otherwise, we accept f. Our analysis follows a combinatorial style similar to [BWY15]. We would like to note that although we use FindInfBucket for k-linearity testing as well, the analysis here significantly deviates from that of k-linearity testing and is presented in Claim 6.10. The main result is formally proved in Section 6.1.

3 Preliminaries

Notations Throughout this work, we use boldface letters to represent vectors of length n and normal face letters for variables. Specifically, $\boldsymbol{e}_i \triangleq (0, \ldots, 0, 1_i, 0, \ldots, 0)$ denotes the i^{th} standard unit vector. For $n \in \mathbb{N}$, let $[n]$ denote the set $\{1, \ldots, n\}$. For a matrix A, let $\|A\|_\infty$, $\|A\|_{\mathrm{op}}$, and $\|A\|_{\mathrm{F}}$ denote the supremum, operator, and Frobenious norms of A, respectively. See [HJ12] for the formal definitions. For concise expressions and readability, we use the asymptotic complexity notion of $\widetilde{O}(\cdot)$, where we hide poly-logarithmic dependencies of the parameters. For any $f : \mathbb{R}^n \to \mathbb{R}$, let $\|f\|_{\infty,C}$ denote the ℓ_∞ norm, i.e. the supremum value attained by f over some $C \subset \mathbb{R}^n$. Let Π_n be the class of functions $\mathbb{R}^n \to \mathbb{R}$ satisfying some particular first-order property and let $\Pi = \cup_{n\geq 1}\Pi_n$.

Definition 3.1 (ℓ_1-distance)**.** Let $\mathcal{D} = \{\mathcal{D}_n\}_{n\geq 1}$ is family of distributions with $\mathcal{D}_n$ being a distribution on $\mathbb{R}^n$. For two arbitrary functions $f, g : \mathbb{R}^n \to \mathbb{R}$, the ℓ_1-distance between f and g is defined as:

$$\mathsf{dist}_{\mathcal{D},\ell_1}(f, g) \triangleq \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{D}_n} [|f(\boldsymbol{x}) - g(\boldsymbol{x})|].$$

We also define the ℓ_p-distance of f to the class Π_n, and hence the class Π, by

$$\mathsf{dist}_{\mathcal{D},\ell_p}(f, \Pi) \triangleq \mathsf{dist}_{\mathcal{D},\ell_p}(f, \Pi_n) \triangleq \inf_{g\in\Pi_n} \mathsf{dist}_{\mathcal{D},\ell_p}(f, g).$$

Remark 3.2. *Since our distributions are supported over continuous spaces, the distances we consider are also defined over such continuous spaces, i.e.,* $\mathsf{dist}_{\mathcal{D},\ell_j}(\cdot,\cdot) \equiv \mathsf{dist}_{\mathcal{D},L_j}(\cdot,\cdot)$*, for all* j*.*

We have $\mathsf{dist}_{\mathcal{D},\ell_p}(f,\Pi) = 0$ if and only if there exists a function $g \in \Pi_n$ which agrees with f almost everywhere with respect to $\mathcal{D}_n$ (in the measure-theoretic sense). We will only concern ourselves with $p \in \{0,1\}$ since we are dealing with scalar-valued functions (aka functionals) and hence $\|f(\boldsymbol{x}) - g(\boldsymbol{x})\|_p = |f(\boldsymbol{x}) - g(\boldsymbol{x})|$ for all $p > 0$.

The notion of ℓ_1-distance makes more sense in the case of real-valued functions, especially if the evaluation of f is not exact. To see this, consider the case when $g = f + \varepsilon$ for some $\varepsilon > 0$, then $\mathsf{dist}_{\mathcal{D},\ell_1}(f,g) = \varepsilon$ whereas $\mathsf{dist}_{\mathcal{D},\ell_0}(f,g) = 1$.

We have the following observation that shows that over finite fields, any $(k+1)$-linear function is always far from any k-linear function in ℓ_0-distance.

Observation 3.3. *For* $1 \le k < n$*, for any* $(k+1)$*-linear function* $f : \mathbb{F}_2^n \to \mathbb{F}_2$*, we have* $\mathsf{dist}_{\ell_0}(f,g) = 1/2$ *for any* k*-linear function* $g : \mathbb{F}_2^n \to \mathbb{F}_2$*.*

Proof. Let $f(\boldsymbol{x}) = \sum_{i\in[n]} c_i x_i$, and $S \triangleq \{i \in [n] : c_i \neq 0\}$ with $|S| = k+1$, since f is $(k+1)$-linear. For any k-linear g with $g(\boldsymbol{x}) = \sum_{i\in[n]} a_i x_i$, there is at least one $i^* \in S$ for which $a_{i^*} = 0$. Then

$$\begin{aligned}
\mathsf{dist}_{\ell_0}(f,g) &= \Pr_{\boldsymbol{x}\leftarrow F_2^n}[f(\boldsymbol{x}) \neq g(\boldsymbol{x})] = \Pr_{\boldsymbol{x}\leftarrow F_2^n}[f(\boldsymbol{x}) + g(\boldsymbol{x}) = 1] \\
&= \Pr_{\boldsymbol{x}\leftarrow F_2^n}\left[\sum_{i=1}^{n}(a_i + c_i)x_i = 1\right] && \text{(Note } c_{i^*} = 1, a_{i^*} = 0) \\
&= \Pr_{\boldsymbol{x}\leftarrow \mathbb{F}_2^n}\left[x_{i^*} + \sum_{i\neq i^*}(a_i + c_i)x_i = 0\right] = \frac{1}{2}.
\end{aligned}$$

□

Unfortunately, there is no general analogous result in the real case with the ℓ_1-distance, but there is an upper bound, provided the distribution $\mathcal{D}_n$ has some concentration.

Claim 3.4. *If* $f, g : \mathbb{R}^n \to \mathbb{R}$ *are linear functions with* $f(\mathbf{0}) = g(\mathbf{0})$*, and* $L^2_{\mathcal{D}}(n) = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_n}\|\boldsymbol{x}\|_2$*, then*

$$\mathsf{dist}_{\mathcal{D},\ell_1}(f,g) \le \|f - g\|_2 \cdot L^2_{\mathcal{D}}(n).$$

Proof. Suppose $f(\boldsymbol{x}) = \sum_{i\in[n]} a_i x_i$ and $g(\boldsymbol{x}) = \sum_{i\in[n]} b_i x_i$ for some $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^n$. Let $h = f - g$, so that $h(\boldsymbol{x}) = \sum_{i\in[n]}(a_i - b_i)x_i = \langle \boldsymbol{a} - \boldsymbol{b}, \boldsymbol{x}\rangle$. We have, by Cauchy-Schwarz inequality,

$$\mathsf{dist}_{\mathcal{D},\ell_1}(f,g) \triangleq \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{D}_n}|h(\boldsymbol{x})| \le \|\boldsymbol{a} - \boldsymbol{b}\|_2 \cdot \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{D}_n}\|\boldsymbol{x}\|_2 = \|f - g\|_2 \cdot L^2_{\mathcal{D}}(n).$$

□

Our definitions hold for the general reference distributions $\mathcal{D}$ which are suitably concentrated. Later, we work with the standard Gaussian distribution $\mathcal{N}(\mathbf{0}, I_n)$, which also has this desired concentration property.

Definition 3.5 (Concentrated distribution)**.** Let $\varepsilon \in (0,1), R \ge 0$, and $\boldsymbol{c} \in \mathbb{R}^n$. A distribution $\mathcal{D}$ supported on $\mathbb{R}^n$ is $(\varepsilon, R, \boldsymbol{c})$-*concentrated* if most of its mass is contained in a ball of radius R centered at some point $\boldsymbol{c} \in \mathbb{R}^n$, i.e.,

$$\Pr_{\boldsymbol{p}\sim\mathcal{D}}[\boldsymbol{p} \in \mathrm{B}(\boldsymbol{c}, R)] \ge 1 - \varepsilon.$$

Fact 3.6 ([BHK20, Theorem 2.9])**.** The standard Gaussian distribution $\mathcal{N}(\mathbf{0}, I_n)$ is $(0.01, 2\sqrt{n}, \mathbf{0})$-concentrated.

For brevity, we may omit the third entry corresponding to the center of the ball, when the center is the origin. For example, $(0.01, 2\sqrt{n})$-concentrated means $(0.01, 2\sqrt{n}, \mathbf{0})$-concentrated.

Definition 3.7 (Property Tester)**.** Let $\mathcal{P}$ be a real function property. An algorithm is said to be a *tester* for $\mathcal{P}$ with respect to distance measure $\mathsf{dist}(\cdot,\cdot)$; with proximity parameter $\varepsilon > 0$, completeness error $c \in (0,1)$, and soundness error $s \in (0,1)$ if, given query access (either exact, or η-approximate query access) to a function $f : \mathbb{R}^n \to \mathbb{R}$, and sample access to a reference distribution $\mathcal{D}_n$ on $\mathbb{R}^n$, the algorithm performs $q(n, d, k, \varepsilon, c, s)$ queries to f and:

(i) Outputs Accept with probability $\geq 1 - c$ (over the randomness of the algorithm), if $f \in \mathcal{P}$.

(ii) Outputs Reject with probability $\geq 1 - s$, if $\mathsf{dist}(f, g) \geq \varepsilon$ for all $g \in \mathcal{P}$.

3.1 Preliminaries on Polynomials

We briefly discuss some notions, and properties of polynomials:

Definition 3.8 (Monomials)**.** Suppose $x_1, \ldots, x_n$ are indeterminates. A *monomial* in these indeterminates is a product of the form $x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}$ where $\boldsymbol{\alpha} \triangleq (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n$, which we also denote by $\boldsymbol{x}^{\boldsymbol{\alpha}}$. The total degree of monomial $\boldsymbol{x}^{\boldsymbol{\alpha}}$ is $\|\boldsymbol{\alpha}\|_1 \triangleq \alpha_1 + \cdots + \alpha_n$. The degree of $\boldsymbol{x}^{\boldsymbol{\alpha}}$ in the variable x_i is α_i. The individual degree of $\boldsymbol{x}^{\boldsymbol{\alpha}}$ is $\|\boldsymbol{\alpha}\|_\infty \triangleq \max_{i \in [n]} \alpha_i$. Over a field $\mathbb{F}$, each monomial $\boldsymbol{x}^{\boldsymbol{M}} = x_1^{M_1} \cdots x_n^{M_n}$ for $\boldsymbol{M} \in \mathbb{N}^n$ corresponds to a function $\boldsymbol{M} : \mathbb{F}^n \to \mathbb{F}$, $\boldsymbol{M}(\boldsymbol{z}) \triangleq z_1^{M_1} \cdots z_n^{M_n}$.

Definition 3.9 (Polynomials)**.** A *polynomial* in $x_1, \ldots, x_n$ (aka an n-variate polynomial) over a field $\mathbb{F}$ is a finite $\mathbb{F}$-linear combination of monomials in $x_1, \ldots, x_n$. That is, an n-variate polynomial over $\mathbb{F}$ is of the form $P(\boldsymbol{x}) = \sum_{i=1}^m a_i \boldsymbol{x}^{\boldsymbol{M}_i}$ where $m \geq 0$, $\boldsymbol{M}_1, \ldots, \boldsymbol{M}_m \in \mathbb{N}^n$ and $a_1, \ldots, a_m \in \mathbb{F} \setminus \{0\}$. The set of all such polynomials is denoted as $\mathbb{F}[x_1, \ldots, x_n]$.

Definition 3.10 (Polynomial sparsity)**.** The *sparsity* of a polynomial $P(\boldsymbol{x}) = \sum_{i=1}^m a_i \boldsymbol{x}^{\boldsymbol{M}_i}$, denoted $\|P\|_0$, is the number of non-zero coefficients a_i in its monomial-basis representation. The unique polynomial P with $\|P\|_0 = 0$ is the *zero polynomial*, denoted by $P \equiv 0$.

Definition 3.11 (Total and individual degrees)**.** The *total degree* $\deg(P)$ of a non-zero polynomial $P(\boldsymbol{x}) = \sum_{i \in [m]} a_i \boldsymbol{x}^{\boldsymbol{M}_i}$ is the maximum total degree of its monomials, i.e., $\deg(P) \triangleq \max_{i \in [m]} \|\boldsymbol{M}_i\|_1$. Similarly, the *individual degree* $\mathrm{ideg}(P)$ of P is the maximum individual degree of its monomials, i.e., $\mathrm{ideg}(P) \triangleq \max_{i \in [m]} \|\boldsymbol{M}_i\|_\infty$. The total, as well as individual degree of the zero polynomial $P \equiv 0$ is defined to be $-\infty$.

In this work, we will be primarily be working on total degree. So, we will use degree to represent total degree when it is clear from the context.

When $\mathbb{F}$ is an infinite field, each polynomial $P(x_1, \ldots, x_n) = \sum_{i \in [m]} a_i \boldsymbol{x}^{\boldsymbol{M}_i}$ over $\mathbb{F}$ corresponds to a *unique* $\mathbb{F}$-valued function $P : \mathbb{F}^n \to \mathbb{F}$ with $P(\boldsymbol{z}) = \sum_{i=1}^m a_i \boldsymbol{M}_i(\boldsymbol{z}) = \sum_{i=1}^m a_i \boldsymbol{z}_1^{M_{i,1}} \cdots \boldsymbol{z}_n^{M_{i,n}}$. Such functions are referred to as *polynomial functions*. Unlike with finite fields, the equality of two real polynomial functions implies the equality of the formal polynomials. Thus, we use the formal polynomial $P \in \mathbb{R}[x_1, \ldots, x_n]$ and the *polynomial function* $P : \mathbb{R}^n \to \mathbb{R}$ interchangeably.

Definition 3.12. Let $\mathcal{P}^{\mathrm{tot}}_{n,d,k}$ denote the class of all n-variate real polynomials with *total degree* $\leq d$ and sparsity $\leq k$.

$$\mathcal{P}^{\mathrm{tot}}_{n,d,k} \triangleq \left\{ P \in \mathbb{R}[x_1, \ldots, x_n] \mid P = \sum_{i=1}^k a_i \boldsymbol{x}^{\boldsymbol{M}_i} : \forall\, i \in [k], a_i \in \mathbb{R}, \boldsymbol{M}_i \in \mathbb{N}^n : \|\boldsymbol{M}_i\|_1 \leq d \right\}.$$

Similarly, let $\mathcal{P}^{\text{ind}}_{n,d,k}$ denote the class of all n-variate real polynomials with *individual degree* $\leq d$ in all variables and sparsity $\leq k$.

$$\mathcal{P}^{\text{ind}}_{n,d,k} \triangleq \left\{ P \in \mathbb{R}[x_1,\ldots,x_n] \mid P = \sum_{i=1}^{k} a_i \boldsymbol{x}^{\boldsymbol{M}_i} : \forall\, i \in [k], a_i \in \mathbb{R}, \boldsymbol{M}_i \in \{\{0\} \cup [d]\}^{\otimes n} \right\}.$$

3.2 Preliminaries on Linear Algebra

We will use the notions of Hankel and Vandermonde matrices in this work.

Definition 3.13 (Hankel Matrix)**.** For any $n > 0$ and nodes $x_0, \ldots, x_{2n-2} \in (R, +, \cdot)$, the $n \times n$ symmetric Hankel matrix with nodes $x_0, \ldots, x_{2n-2}$ is denoted $H_n(x_0, \ldots, x_{2n-2})$ and is given by

$$H_n(x_0,\ldots,x_{2n-2}) \triangleq \begin{pmatrix} x_0 & x_1 & \cdots & x_{n-1} \\ x_1 & x_2 & \cdots & x_n \\ \vdots & \vdots & \ddots & \vdots \\ x_{n-1} & x_n & \cdots & x_{2n-2} \end{pmatrix}.$$

Definition 3.14 (Vandermonde matrix and determinant)**.** For any $m, n > 0$ and $x_1, \ldots, x_m \in (R, +, \cdot)$, the $m \times n$ Vandermonde matrix with nodes $x_1, \ldots, x_m$ denoted by $\mathsf{V}_n(x_1, \ldots, x_m)$ is:

$$\mathsf{V}_n(x_1,\ldots,x_m) \triangleq \begin{pmatrix} 1 & x_1 & \cdots & x_1^{n-1} \\ 1 & x_2 & \cdots & x_2^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & \cdots & x_m^{n-1} \end{pmatrix}.$$

If $m = n$, $\det\left(\mathsf{V}_n(x_1, \ldots, x_n)\right) = \prod_{1 \leq i < j \leq n}(x_j - x_i)$, which is the Vandermonde determinant formula.

3.3 Results from Perturbation Theory

For analyzing our testers, we use the following result from perturbation theory.

Theorem 3.15 (Weyl's Inequality [HJ94])**.** *Let $\hat{\Sigma} = \Sigma + E$, where $\Sigma, E \in \mathbb{R}^{n \times n}$ are both symmetric matrices. Let $\lambda_i(A)$ denote the i^{th} eigenvalue of (symmetric matrix) A, sorted in non-increasing order, and let $\|A\|_{\text{op}}$ denote its ℓ_2-operator (spectral) norm. Then,*

$$\max_{i \in [n]} \{|\lambda_i(\hat{\Sigma}) - \lambda_i(\Sigma)|\} \leq \|E\|_{\text{op}} \leq \max\{|\lambda_d(E)|, |\lambda_1(E)|\}.$$

3.4 Results from Measure Theory

The following celebrated inequality of Carbery and Wright [CW01] provides anti-concentration bounds for polynomials of i.i.d Gaussian random variables (more generally, log-concave), which will be used in our proofs, particularly in the analysis of our k-linearity tester.

Theorem 3.16 ([CW01, Theorem 8])**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a polynomial of degree d, such that $\mathsf{Var}_{\mathcal{N}(\mathbf{0},I)}[f] = 1$. Then, for any $t \in \mathbb{R}$, and any $\varepsilon > 0$,*

$$\Pr_{\boldsymbol{x} \sim \mathcal{N}(\mathbf{0},I)}[|f(\boldsymbol{x}) - t| \leq \varepsilon] \leq O(d)\varepsilon^{1/d}.$$

A recent result of Glazer and Mikulincer [GM22] provided a more suitable form for applying Carbery-Wright to polynomials, where the variance is lower bounded in terms of the coefficients.

Theorem 3.17 ([GM22, Corollary 4]). *If $f : \mathbb{R}^n \to \mathbb{R}$ is a polynomial $f(\boldsymbol{x}) = \sum_{i=1}^{k} a_i \boldsymbol{x}^{\boldsymbol{M}_i}$ of degree d, then there exists an absolute constant $C > 0$ such that for any $t \in \mathbb{R}$ and $\varepsilon > 0$,*

$$\Pr_{\boldsymbol{x}\sim\mathcal{N}(\mathbf{0},I)}[|f(\boldsymbol{x}) - t| \le \varepsilon] \le Cd\left(\frac{\varepsilon}{\mathrm{coeff}_d(f)}\right)^{1/d}, \text{ where } \mathrm{coeff}_d^2(f) \triangleq \sum_{i\in[k]:\|\boldsymbol{M}_i\|_1=d} a_i^2.$$

The Paley-Zygmund anti-concentration result will be useful for analyzing our junta tester.

Theorem 3.18 ([PZ32]). *For a random variable $X \ge 0$ with finite variance, for all $\theta \in [0, 1]$,*

$$\Pr[X \ge \theta\,\mathbb{E}[X]] \ge (1-\theta)^2 \frac{\mathbb{E}^2[X]}{\mathbb{E}[X^2]}.$$

4 k-linearity Testing

In this section, we present and analyze our algorithm Test-k-Linear (Algorithm 1).

Theorem 4.1 (Generalization of Theorem 1.3). *Let $k \in \mathbb{N}$, $f : \mathbb{R}^n \to \mathbb{R}$ be given via an η-approximate query access, and $\varepsilon, \eta \in (0, 2/3)$ be such that $\eta < \min\left\{\varepsilon, O\left(\min_{i\in[n]:f(\boldsymbol{e}_i)\neq 0} \frac{|f(\boldsymbol{e}_i)|}{(nk)^2}\right)\right\}$, where $\boldsymbol{e}_i$ denotes the i^{th} standard unit vector. There exists a tester* Test-k-Linear *(Algorithm 1) that performs $\widetilde{O}(k \log k + 1/\varepsilon)$ queries and guarantees:*

- Completeness: *If f is a k-linear function, then Algorithm 1* Accepts *with probability at least $2/3$.*
- Soundness: *If f is ε-far from all k-linear functions, then Algorithm 1* Rejects *with probability at least $2/3$.*

Algorithm 1: Test-k-Linear

Inputs: η-approximate query oracle $\widetilde{f}$ for $f : \mathbb{R}^n \to \mathbb{R}, k \in \mathbb{N}$, proximity parameter $\varepsilon \in (0, 1)$.
Output: Returns Accept iff f is a k-linear function.
Run Approximate Additivity Tester$(f, \mathcal{N}(\mathbf{0}, I), \eta, \varepsilon, 2\sqrt{n})$ (Algorithm 7).
If Algorithm 7 rejects, **return** Reject.
Set $r = O(k^2)$.
Bucket each $i \in [n]$ into $B = \{B_1, \ldots, B_r\}$, uniformly at random. ▷ $[n] = \uplus_{i=1}^{r} B_i$.
$\mathsf{InfBuckets} = \mathsf{FindInfBuckets}(\text{Approximate-}g, B, [r])$. ▷ $\mathsf{FindInfBuckets}$ = Algorithm 3
if $|\mathsf{InfBuckets}| > k$ **then**
  return Reject.
return Accept.

Algorithm 1 uses two subroutines FindInfBucket (Algorithm 2) and FindInfBuckets (Algorithm 3). We will first present and analyze them in Section 4.1, and then analyze Algorithm 1 in Section 4.2.

A closely related (and *efficiently testable*) notion of *additivity* may be noted here.

Definition 4.2 (Additive function)**.** A function $f : A \to B$ is *additive*, if for all $x, y \in A$, $f(x \oplus_A y) = f(x) \oplus_B f(y)$, where $\oplus_A$ and $\oplus_B$ denote the bitwise-xor operations in A and B, respectively.

Over finite domains, additivity implies linearity. But over continuous domains, this isn't always the case. However, for continuous functions, testing additivity suffices from the following fact:

Fact 4.3 ([Kuc09, Section 5.2])**.** For continuous functions, additivity is equivalent to linearity.

Algorithm 1 uses a query oracle to the self-corrected function, Approximate-g (presented in Algorithm 10). We present a brief discussion of self-correction for additivity testing:

Self-Correction We use the definition of self-correction from [ABF+23, AKM25], who studied the problem of testing additive (linear) functions, given approximate oracle access. Let r be a sufficiently small rational; $r \triangleq 1/50$ suffices. Define the value of the self-corrected function g at a point $\boldsymbol{p} \in \mathrm{B}(\mathbf{0}, r)$ as the (weighted) median value of $g_{\boldsymbol{x}}(\boldsymbol{p}) \triangleq f(\boldsymbol{p} - \boldsymbol{x}) + f(\boldsymbol{x})$, each weighted according to its probability mass under $\boldsymbol{x} \sim \mathcal{N}(\mathbf{0}, I_n)$. For points $\boldsymbol{p} \notin \mathrm{B}(\mathbf{0}, r)$, we project them into the ball by scaling by a sufficiently large contraction factor that depends on the magnitude of $\boldsymbol{p}$.

$$g(\boldsymbol{p}) \triangleq \kappa_{\boldsymbol{p}} \cdot \underset{\boldsymbol{x} \sim \mathcal{N}(\mathbf{0}, I_n)}{\mathsf{med}} \left[g_{\boldsymbol{x}} \left(\frac{\boldsymbol{p}}{\kappa_{\boldsymbol{p}}} \right) \right] = \kappa_{\boldsymbol{p}} \cdot \underset{\boldsymbol{x} \sim \mathcal{N}(\mathbf{0}, I_n)}{\mathsf{med}} \left[f\left(\frac{\boldsymbol{p}}{\kappa_{\boldsymbol{p}}} - \boldsymbol{x} \right) + f(\boldsymbol{x}) \right], \tag{1}$$

where $\kappa_{\boldsymbol{p}} : \mathbb{R}^n \to \mathbb{R}$ is defined as $\kappa_{\boldsymbol{p}} \triangleq \begin{cases} 1 & , \text{if } \|\boldsymbol{p}\|_2 \le r \\ \lceil \|\boldsymbol{p}\|_2 / r \rceil & , \text{if } \|\boldsymbol{p}\|_2 > r \end{cases}$, so that $\boldsymbol{p}/\kappa_{\boldsymbol{p}} \in \mathrm{B}(\mathbf{0}, r)$.

We note the following results from [ABF+23, AKM25] about their approximate additivity tester.

Theorem 4.4 ([ABF+23, Theorem D.1], and [AKM25, Theorem 3.3])**.** *Let $\alpha, \varepsilon > 0$ and $\mathcal{D}$ be an unknown $(\varepsilon/4, R)$-concentrated distribution. There exists a one-sided error, $O(1/\varepsilon)$-query tester (Algorithm 7) which with probability at least $99/100$, distinguishes when f is pointwise α-close to some additive function and when, for every additive function h, $\Pr_{\boldsymbol{p} \sim \mathcal{D}}[|f(\boldsymbol{p}) - h(\boldsymbol{p})| > O(Rn^{1.5}\alpha)] > \varepsilon$.*

Lemma 4.5 ([ABF+23, Lemma D.3 and D.6])**.** *If* TestAdditivity$(f, 3\alpha)$ *(Algorithm 8) accepts with probability at least $1/3$, then g is a 42α-additive function inside the small ball $\mathrm{B}(\mathbf{0}, r)$, and furthermore, for every $\boldsymbol{p} \in \mathrm{B}(\mathbf{0}, r)$ it holds that*

$$\Pr_{\boldsymbol{x} \sim \mathcal{N}(\mathbf{0}, I_n)}[|g(\boldsymbol{p}) - g_{\boldsymbol{x}}(\boldsymbol{p})| \ge 12\alpha] < 12/125.$$

Lemma 4.6 ([ABF+23, Lemma D.4 and D.5])**.** *If* TestAdditivity*$(f, 3\alpha)$ accepts with probability at least $1/3$, then for every $\boldsymbol{p}, \boldsymbol{q} \in \mathrm{B}(\mathbf{0}, r)$ with $\|\boldsymbol{p} + \boldsymbol{q}\|_2 \le r$, it holds that*

$$|g(\boldsymbol{p} + \boldsymbol{q}) - g(\boldsymbol{p}) - g(\boldsymbol{q})| \le 42\alpha.$$

The following lemma gives us a way to scale the closeness for degree-d polynomials.

Lemma 4.7 ([ABF+23, Lemma 4.19])**.** *Let $R > r' > 0$ be any real numbers. If g is pointwise η-close to a degree-d polynomial in $\mathrm{B}(\mathbf{0}, r')$, then g is pointwise $(12R/r')^d\eta$-close to a degree-d polynomial on all points in $\mathrm{B}(\mathbf{0}, R)$.*

Using Lemma 4.5, Lemma 4.7 and the structure of Approximate-g subroutine, they bound the distance between g and Approximate-g.

Claim 4.8. *If* TESTADDITIVITY$(f, 3\alpha)$ *(Algorithm 8) accepts with probability at least* $1/3$*, then* APPROXIMATE-g *is pointwise* 6α*-close to* g*, in the ball* $\mathrm{B}(\mathbf{0}, r)$ *with high probability; i.e.,*

$$\Pr_{\boldsymbol{p}\sim\mathcal{D}}[|g(\boldsymbol{p}) - \text{APPROXIMATE-}g(\boldsymbol{p})| \leq 6\alpha(12R/r) \mid \boldsymbol{p} \in \mathrm{B}(\mathbf{0}, R)] \geq 1 - \varepsilon/4.$$

The following observation directly follows using the triangle inequality, along with the above claim and the notion of η-approximate queries.

Observation 4.9. *If* APPROXIMATE ADDITIVITY TESTER *does not reject* f *with probability at least* $2/3$*, then following the description of* APPROXIMATE ADDITIVITY TESTER*, we have*

$$\Pr_{\boldsymbol{p}\sim\mathcal{D}}[|f(\boldsymbol{p}) - \text{APPROXIMATE-}g(\boldsymbol{p})| \leq 750Rn^{1.5}\alpha \mid \boldsymbol{p} \in \mathrm{B}(\mathbf{0}, R)] \geq 1 - \frac{\varepsilon}{4}.$$

We will invoke them with $\mathcal{D} = \mathcal{N}(\mathbf{0}, I_n)$, implying $R = 2\sqrt{n}$ (from Fact 3.6).

4.1 Analyses of Subroutines

Algorithm 2: FindInfBucket(f, B, S)

Inputs: η-approximate query oracle $\widetilde{f}$ for $f : \mathbb{R}^n \to \mathbb{R}$, Random Bucketing $B = \{B_1, \ldots, B_r\}$ of $[n]$, $\emptyset \neq S \subseteq [r]$

Output: Either $\emptyset$, or an influential bucket B_j (with $j \in S$) of f. B_j is an influential bucket if x_v is an influential variable of f for some $v \in B_j$.

Let $V \triangleq \{i \in [n] : i \in B_j \text{ for some } j \in S\}$. ▷ The #variables in V will be $|V| \approx \frac{|S|n}{r}$.

Sample two independent Gaussian vectors $\boldsymbol{x}, \boldsymbol{y} \sim \mathcal{N}(\mathbf{0}, I_n)$.

if $|\widetilde{f}(\boldsymbol{x}_V \boldsymbol{y}_{\overline{V}}) - \widetilde{f}(\boldsymbol{y})| \leq 2\eta$ **then**
- **return** $\emptyset$.

else
- **if** $S = \{j\}$ *for some* j **then**
  - **return** B_j.
- **else**
  - Partition S into two parts: $S^{(L)}$ and $S^{(R)}$ with $1 \leq |S^{(R)}| \leq |S^{(L)}| \leq |S^{(R)}| + 1$.
  - $\mathsf{ret}_L \leftarrow$ FindInfBucket$(f, B, S^{(L)})$
  - **if** $\mathsf{ret}_L = \emptyset$ **then**
    - **return** FindInfBucket$(f, B, S^{(R)})$.
  - **else**
    - **return** ret_L.

The subroutine FindInfBucket is presented in Algorithm 2. To analyze it, we need:

Definition 4.10 (Influential bucket for linear functions)**.** For a function $f : \mathbb{R}^n \to \mathbb{R}$, a bucket $B \subseteq [n]$ is said to be an *influential bucket* if there exists at least one variable $\boldsymbol{x}_i, i \in B$, which is influential with respect to f (i.e., the value of $f(\boldsymbol{x})$ changes with a change $\boldsymbol{x}_i$). For a linear f, with $f(\boldsymbol{x}) = \sum_{i\in[n]} a_i \boldsymbol{x}_i$, $\boldsymbol{x}_i$ is influential w.r.t f if and only if $a_i \neq 0$, and hence B is an influential bucket iff $a_i \neq 0$ for some $i \in B$.

Claim 4.11 (Correctness of $\mathsf{FindInfBucket}$)**.** *If $f : \mathbb{R}^n \to \mathbb{R}$, with $f(\boldsymbol{x}) = \sum_{i\in[n]} a_i \boldsymbol{x}_i$ is given via an η-approximate query oracle $\widetilde{f}$, where $\eta \leq \frac{1}{100k^2} \min_{k\in[n]: a_k \neq 0} |a_k|$, $B = \{B_1, \ldots, B_r\}$ is a partition of $[n]$, and $\emptyset \neq S \subseteq [r]$, then $\mathsf{FindInfBucket}(f, B, S)$ (Algorithm 2) guarantees the following:*

1. *If none of the buckets $\{B_i, i \in S\}$ are influential, $\mathsf{FindInfBucket}(f, B, S)$ always returns $\emptyset$ and performs exactly 2 queries to f.*
2. *Otherwise, with probability at least $1 - 8\lceil \log|S| \rceil^2/10k^2$, $\mathsf{FindInfBucket}(f, B, S)$ returns B_j for some $j \in S$ which is an influential bucket, and performs $\leq 8\lceil \lg(|S|)\rceil^2$ queries to f.*

Proof. Let $\boldsymbol{x}$ and $\boldsymbol{y}$ be the Gaussian random vectors sampled by Algorithm 2, and $\boldsymbol{w} \triangleq \boldsymbol{y}_{\overline{V}} \boldsymbol{x}_V \in \mathbb{R}^n$.

(Part 1) Since no B_j is influential for $j \in S$, $a_i = 0$ for all $i \in V$ giving us

$$f(\boldsymbol{w}) - f(\boldsymbol{y}) = \sum_{j\in\overline{V}} a_j \underbrace{(\boldsymbol{w}_j - \boldsymbol{y}_j)}_{=\boldsymbol{y}_j - \boldsymbol{y}_j = 0} + \sum_{j \in V} \underbrace{a_j}_{=0} \underbrace{(\boldsymbol{w}_j - \boldsymbol{y}_j)}_{=\boldsymbol{x}_j - \boldsymbol{y}_j} = 0.$$

Hence, $|\widetilde{f}(\boldsymbol{w}) - \widetilde{f}(\boldsymbol{z})| = |\widetilde{f}(\boldsymbol{w}) - f(\boldsymbol{w}) - (\widetilde{f}(\boldsymbol{z}) - f(\boldsymbol{z}))| \leq \overbrace{|\widetilde{f}(\boldsymbol{w}) - f(\boldsymbol{w})|}^{\leq \eta} + \overbrace{|\widetilde{f}(\boldsymbol{z}) - f(\boldsymbol{z})|}^{\leq \eta} \leq 2\eta$ by the triangle inequality, ensuring the check in Line 5 (Algorithm 2) always holds. So in this case, $\mathsf{FindInfBucket}(f, B, S)$ will always return $\emptyset$, and make exactly 2 queries to f.

(Part 2) It only remains to prove the second part of the claim (when there exists some $j \in S$ with B_j being influential, i.e., there exists $k \in B_j$ such that $a_k \neq 0$). We do so by strong induction on the size of S.

When $|S| = 1$, we have $S = \{j\}$ and $V = B_j$, with $a_k \neq 0$ for some $k \in B_j$. Then $f(\boldsymbol{w}) - f(\boldsymbol{y}) = \sum_{k\in B_j} a_k(\boldsymbol{x}_k - \boldsymbol{y}_k)$. This implies $\Pr_{\boldsymbol{x},\boldsymbol{y}\sim\mathcal{N}(\mathbf{0},I_n)}[f(\boldsymbol{w}) = f(\boldsymbol{z})] = 0$, since $\sum_{k\in B_j} a_k(\boldsymbol{x}_k - \boldsymbol{y}_k) \not\equiv 0$, and any non-zero polynomial over reals vanishes only on a set (of its zeroes) of measure zero. Moreover,

$$\begin{aligned}
\Pr_{\boldsymbol{x},\boldsymbol{y}\sim\mathcal{N}(\mathbf{0},I_n)}[|\widetilde{f}(\boldsymbol{w}) - \widetilde{f}(\boldsymbol{y})| > 2\eta] &\geq \Pr_{\boldsymbol{x},\boldsymbol{y}\sim\mathcal{N}(\mathbf{0},I_n)}[|f(\boldsymbol{w}) - f(\boldsymbol{y})| > 4\eta] \\
&= 1 - \Pr_{\boldsymbol{x},\boldsymbol{y}\sim\mathcal{N}(\mathbf{0},I_n)}\left[\left|\sum_{k\in B_j} a_k(\boldsymbol{x}_k - \boldsymbol{y}_k)\right| \leq 4\eta\right] \\
&\geq 1 - \Pr_{\boldsymbol{x},\boldsymbol{y}\sim\mathcal{N}(\mathbf{0},I_n)}\left[\max_{k\in B_j} |a_k| \left|\sum_{k\in B_j} (\boldsymbol{x}_k - \boldsymbol{y}_k)\right| \leq 4\eta\right] \\
&= 1 - \Pr_{\boldsymbol{x},\boldsymbol{y}\sim\mathcal{N}(\mathbf{0},I_n)}\left[\frac{|\sum_{k\in B_j}(\boldsymbol{x}_k - \boldsymbol{y}_k)|}{\sqrt{2|B_j|}} \leq \frac{2\sqrt{2}\eta}{\sqrt{|B_j|}\max_{k\in B_j}|a_k|}\right].
\end{aligned}$$

As $\boldsymbol{x}, \boldsymbol{y} \sim \mathcal{N}(\mathbf{0}, I_n)$ are independent, we have $\sum_{k\in B_j} \frac{\boldsymbol{x}_k - \boldsymbol{y}_k}{\sqrt{2|B_j|}} \equiv \sum_{k\in B_j} \frac{x_k - y_k}{\sqrt{2|B_j|}} \in \mathbb{R}[\cup_{k\in B_j}\{x_k, y_k\}]$, $\mathsf{Cov}[x_i, x_j] = \mathsf{Cov}[y_i, y_j] = 0$, for all $i \neq j \in B_j$, and $\mathsf{Cov}[x_i, y_j] = 0$, for all $i, j \in S$, giving us

$$\underset{\boldsymbol{x},\boldsymbol{y}\sim\mathcal{N}(\mathbf{0},I_n)}{\mathsf{Var}}\left[\sum_{k\in B_j} \frac{\boldsymbol{x}_k - \boldsymbol{y}_k}{\sqrt{2|B_j|}}\right] = \frac{1}{2|B_j|} \sum_{k\in B_j} \left(\underset{x_k\sim\mathcal{N}(0,1)}{\mathsf{Var}}[x_k] + \underset{y_k\sim\mathcal{N}(0,1)}{\mathsf{Var}}[y_k]\right) = 1.$$

Now, applying Theorem 3.16 on $\sum_{k\in B_j} \frac{\boldsymbol{x}_k - \boldsymbol{y}_k}{\sqrt{2|B_j|}}$, (with $d = 1, n = 2|B_j|$, and $t = 0$) we get

$$\Pr_{\boldsymbol{x},\boldsymbol{y}\sim\mathcal{N}(\mathbf{0},I_n)}\left[\frac{|\sum_{k\in B_j}(\boldsymbol{x}_k - \boldsymbol{y}_k)|}{\sqrt{2|B_j|}} \leq \frac{2\sqrt{2}\eta}{\sqrt{|B_j|}\max_{k\in B_j}|a_k|}\right] \leq O\left(\frac{2\sqrt{2}\eta}{\sqrt{|B_j|}\max_{k\in B_j}|a_k|}\right) \ll \frac{1}{10k^2}.$$

The last inequality follows by the assumption that $\eta \le \frac{1}{100k^2}\min_{k\in[n]:a_k\neq 0}|a_k| \le \frac{\sqrt{|B_j|}}{100k^2}\max_{k\in B_j}|a_k|$. Hence, with probability at least $1-1/10k^2$, Algorithm 2 will return $S=\{j\}$ in Line 9.

Now, let $|S|=k>1$. Using a similar argument as in the base case, $\mathsf{FindInfBucket}(f,B,S)$ will return $\emptyset$ with low probability, since $f(\boldsymbol{w})-f(\boldsymbol{y})=\sum_{j\in V}a_j(\boldsymbol{x}_j-\boldsymbol{y}_j)$ will be 0 only when $\boldsymbol{x}_V-\boldsymbol{y}_V$ lies in a $(|V|-1)$-dimensional hyperplane. However,

$$\begin{aligned}
\Pr_{\boldsymbol{x},\boldsymbol{y}\sim\mathcal{N}(\mathbf{0},I_n)}[|\tilde f(\boldsymbol{w})-\tilde f(\boldsymbol{y})|\le 2\eta] &\le \Pr_{\boldsymbol{x},\boldsymbol{y}\sim\mathcal{N}(\mathbf{0},I_n)}[|f(\boldsymbol{w})-f(\boldsymbol{y})|\le 4\eta]\\
&= \Pr_{\boldsymbol{x},\boldsymbol{y}\sim\mathcal{N}(\mathbf{0},I_n)}\left[\left|\sum_{j\in V}a_j(\boldsymbol{x}_j-\boldsymbol{y}_j)\right|\le 4\eta\right]\\
&\le \Pr_{\boldsymbol{x},\boldsymbol{y}\sim\mathcal{N}(\mathbf{0},I_n)}\left[\max_{j\in V}|a_j|\left|\sum_{j\in V}(\boldsymbol{x}_j-\boldsymbol{y}_j)\right|\le 4\eta\right]\\
&\le \Pr_{\boldsymbol{x},\boldsymbol{y}\sim\mathcal{N}(\mathbf{0},I_n)}\left[\left|\sum_{j\in V}\frac{\boldsymbol{x}_j-\boldsymbol{y}_j}{\sqrt{2|V|}}\right|\le\frac{2\sqrt{2}\eta}{\sqrt{|V|}\cdot\max_{j\in V}|a_j|}\right]\triangleq(\star).
\end{aligned}$$

As earlier, we may observe: as $\boldsymbol{x},\boldsymbol{y}\sim\mathcal{N}(\mathbf{0},I_n)$ are sampled independently, we have

$$\sum_{j\in V}\frac{\boldsymbol{x}_j-\boldsymbol{y}_j}{\sqrt{2|V|}}\equiv\sum_{j\in V}\frac{x_j-y_j}{\sqrt{2|V|}}\in\mathbb{R}[\cup_{j\in V}\{x_j,y_j\}],$$

and with $\mathsf{Cov}[x_i,x_j]=\mathsf{Cov}[y_i,y_j]=0$, for all $i\neq j\in V$, and $\mathsf{Cov}[x_i,y_j]=0$, for all $i,j\in V$, we get

$$\mathop{\mathsf{Var}}_{\boldsymbol{x},\boldsymbol{y}\sim\mathcal{N}(\mathbf{0},I_n)}\left[\sum_{j\in V}\frac{x_j-y_j}{\sqrt{2|V|}}\right]=\frac{1}{2|V|}\sum_{j=1}^{|V|}\left(\mathop{\mathsf{Var}}_{x_j\sim\mathcal{N}(0,1)}[x_j]+\mathop{\mathsf{Var}}_{y_j\sim\mathcal{N}(0,1)}[y_j]\right)=1.$$

Applying Theorem 3.16 on $\sum_{j\in V}\frac{x_j-y_j}{\sqrt{2|V|}}$, (with $d=1,n=2|V|$, and $t=0$) we get

$$(\star)=\Pr_{\boldsymbol{x},\boldsymbol{y}\sim\mathcal{N}(\mathbf{0},I_n)}\left[\left|\sum_{j\in V}\frac{\boldsymbol{x}_j-\boldsymbol{y}_j}{\sqrt{2|V|}}\right|\le\frac{2\sqrt{2}\eta}{\sqrt{|V|}\cdot\max_{j\in V}|a_j|}\right]\le O\left(\frac{2\sqrt{2}\eta}{\sqrt{|V|}\cdot\max_{j\in V}|a_j|}\right)\ll\frac{1}{10k^2}.$$

The last inequality again follows by $\eta\le\frac{1}{100k^2}\min_{k\in[n]:a_k\neq0}|a_k|\le\frac{\sqrt{|V|}}{100k^2}\max_{k\in V}|a_k|$.

Thus, with probability at least $1-1/10k^2$, the condition in Line 5 will not hold, ensuring Algorithm 2 reaches the recursive step (Lines 8–13).

By construction, $S=S^{(L)}\sqcup S^{(R)}$, with $1\le|S^{(L)}|\le\lceil|S|/2\rceil$ and $1\le|S^{(R)}|\le\lfloor|S|/2\rfloor$. Either,

(i) $a_j=0$ for all $j\in S^{(L)}$, and there exists $S_R\subseteq S^{(R)}: S_R\neq\emptyset$ and $a_j\neq0$ for all $j\in S_R$, or

(ii) there exists $S_L\subseteq S^{(L)}$ such that $S_L\neq\emptyset$ and $a_j\neq0$ for all $j\in S_L$.

In Case (i), $\mathsf{FindInfBucket}(f,B,S^{(L)})$ will always return $\emptyset$ and perform 2 queries (by Part 1) and hence the return value of $\mathsf{FindInfBucket}(f,B,S)$ will be $\mathsf{FindInfBucket}(f,B,S^{(R)})$. Using the strong induction hypothesis we get, with probability at least $1-4\lceil\log|S|\rceil/10k^2$, this return value will be $\{j\}$ for some $j\in S_R$, and the total number of queries made will be $\le 2+2+4\lceil\lg(\lfloor|S|/2\rfloor)\rceil\le 4+4(\lceil\lg|S|\rceil-1)=4\lceil\lg|S|\rceil$.

In Case (ii), again using strong induction hypothesis, with probability at least $1-4\lceil\log|S|\rceil/10k^2$, $\mathsf{FindInfBucket}(f, B, S^{(L)})$ will return $\{j\}$ for some $j \in S_L$, and thus the algorithm will return $\{j\}$ (line 13). The number of queries made will be $\leq 2+4\lceil\lg(\lceil|S|/2\rceil)\rceil \leq 2+4(\lceil\lg|S|\rceil-\frac{1}{2}) = 4\lceil\lg|S|\rceil$.

We may note, if we run $\mathsf{FindInfBucket}(f, B, S)$ on some set S of bucket-indices with $|S| > 1$, each recursive step (irrespective of whether it falls in case (i) or case (ii) above) will succeed with probability $\geq 1-4\lceil\log|S|\rceil/10k^2$. For the top level call to succeed, all the recursive calls (at most $2\lceil\lg|S|\rceil$ in number) must also succeed. So, we do a union bound over all the failure events to upper bound the total failure probability by $8\lceil\log|S|\rceil^2/10k^2$, and the query complexity by $8\lceil\lg|S|\rceil^2$. ☐

Now we are ready to describe and analyze the subroutine FindInfBuckets.

Algorithm 3: $\mathsf{FindInfBuckets}(\widetilde{f}, B, X)$

```
1 Inputs: η-approximate query oracle f̃ for f : ℝ^n → ℝ, Bucketing B = {B_1, ..., B_r},
  ∅ ≠ X ⊆ [r].
2 Output: A set of influential buckets InfBuckets ⊆ {B_j : j ∈ X} with respect to f.
3 InfBuckets ← ∅
4 for j=1 to 8k do
5     RetVal = FindInfBucket(f, B, X)
6     if RetVal = B_i for some i then
7         InfBuckets ← InfBuckets ∪ {B_i}
8         X ← X \ {i}
9 return InfBuckets.
```

Claim 4.12 (Correctness of FindInfBuckets). *If $f : \mathbb{R}^n \to \mathbb{R}$, with $f(\boldsymbol{x}) = \sum_{i\in[n]} a_i\boldsymbol{x}_i$, is given via the η-approximate query oracle $\tilde{f}$, where $\eta \leq \frac{1}{100k^2}\min_{k\in[n]:a_k\neq 0}|a_k|$, and $\emptyset \neq X \subseteq [n]$, then* $\mathsf{FindInfBuckets}(f, B, X)$ *(Algorithm 3) performs at most $64k\lceil\log(|X|)\rceil^2$ queries, and guarantees:*

(i) If f is ℓ-linear function for some $\ell > 8k$, then Algorithm 3 will return a set of $8k$ influential buckets in f with probability at least $1-64\lceil\log|X|\rceil^2/10k$.

(ii) If f is ℓ-linear function for some $\ell \leq 8k$, then Algorithm 3 will return the set of all influential buckets in f with probability at least $1-64\lceil\log|X|\rceil^2/10k$.

Proof. Note, Algorithm 3 calls Algorithm 2. So, we use Claim 4.11, and proceed case-wise:

(i) Consider the case when f is ℓ-linear for some $\ell > 8k$. This implies that the set of indices X given as input to Algorithm 3 contains more than $8k$ influential variables. Thus, in every iteration of the FOR loop starting in Line 4 of Algorithm 3, Algorithm 2 in Line 5 will return an influential bucket, say B_i, with probability at least $1-8\lceil\log|X|\rceil^2/10k^2$, following Claim 4.11 (ii). Line 7 then computes $\mathsf{Val} = a_i$, followed by Line 9 updating f to $f - a_ix_i$, and X to $X \setminus \{i\}$, removing $\{i\}$ from any future iterations. Since the FOR loop in Algorithm 3 runs for $8k$ iterations, and the number of influential variables is more than $8k$, using a union bound over all the iterations, with probability at least $1-64\lceil\log|X|\rceil^2/10k$, Algorithm 3 returns a set of $8k$ influential variables.

(ii) When f is ℓ-linear for some $\ell \leq 8k$. As before, this implies that the set of indices X given as input to Algorithm 3 contains at most $8k$ influential variables. Following the same argument, as in the above case, we may claim, the $8k$ iterations of the FOR loop in Algorithm 3 returns all the $\ell \leq 8k$ influential variables in f, with probability at least $1-64\lceil\log|X|\rceil^2/10k$.

Query Complexity: Since each call of $\mathsf{FindInfBucket}(f, B, X)$ makes $\leq 8\lceil\lg(|X|)\rceil^2$ queries to f, and there are at most $8k$ such calls, the total number of queries to f is $\leq 64k\lceil\lg(|X|)\rceil^2$. $\square$

4.2 Analysis of k-linearity tester

We are now ready to prove the main theorem of this section:

Proof of Theorem 4.1. **Completeness:** Since f is a k-linear function, following Theorem 4.4, we have: Approximate Additivity Tester accepts f, and hence by Lemma 4.6, g is pointwise 42η-close to linearity in $\mathrm{B}(\mathbf{0}, r)$. Combined with Lemma 4.7, we get g is pointwise $42\eta(12R/r)$-close to linearity in $\mathrm{B}(\mathbf{0}, R)$. Now from Claim 4.8, we know that g and Approximate-g are pointwise $6\eta(12R/r)$-close in $\mathrm{B}(\mathbf{0}, R)$ with probability at least $1 - \varepsilon/4$. Using the triangle inequality, this implies that Approximate-g is pointwise $48\eta(12R/r)$-close to some linear function with probability at least $1 - \varepsilon/4$. Moreover, from Observation 4.9, we get: Approximate-g is in fact pointwise $750Rn^{1.5}\eta$-close to f, with probability at least $1 - \varepsilon/2$.

With $R = 2\sqrt{n}$, following the guarantee of Claim 4.12, which we ensure by our assumption on η:

$$O(n^2\eta) \leq \frac{1}{100k^2} \min_{k \in [n]: a_k \neq 0} |a_k|, \text{ or equivalently } \eta \leq O\left(\min_{i \in [n]: f(\boldsymbol{e}_i) \neq 0} \frac{|f(\boldsymbol{e}_i)|}{(nk)^2}\right),$$

we get that $\mathsf{FindInfBuckets}(\textsc{Approximate-}g, B, [r])$ will return at most k-influential variables of f with probability at least $1 - 128\lceil\log k\rceil^2/5k$. Thus with probability at least $1 - \varepsilon/2 - 128\lceil\log k\rceil^2/5k$, Test-$k$-Linear will Accept.

Soundness: Let us consider the case when f is ε-far from k-linearity. We will prove the contrapositive. We will show that if Test-k-Linear does not reject f with probability at least $1 - \delta (\geq 2/3)$, then f is pointwise close to some k-linear function, with non-zero probability.

Note that if Approximate Additivity Tester rejects f with probability $\geq 1 - \delta$, we are done. So, let us consider the case when Approximate Additivity Tester accepts with probability $\geq \delta$. As in the completeness proof, from Theorem 4.4, and Lemma 4.7, we know that g is pointwise $42\eta(12R/r)$-close to linearity in $\mathrm{B}(\mathbf{0}, R)$, and from Claim 4.8, we have that g and Approximate-g are pointwise $6\eta(12R/r)$-close in $\mathrm{B}(\mathbf{0}, R)$ with probability at least $1 - \varepsilon/4$. This implies that Approximate-g is pointwise $48\eta(12R/r)$-close to some linear function with probability at least $1 - \varepsilon/4$. Again, from Observation 4.9, we get: Approximate-g is pointwise $750Rn^{1.5}\eta$-close to f, with probability at least $1 - \varepsilon/4$, implying now f must be pointwise $(48(12R/r) + 750Rn^{1.5})\eta$-close to linearity, with probability at least $1 - \varepsilon/2$. Note, here $R = 2\sqrt{n}$, and $r = 1/50$.

With our assumption on η again ensuring the conditions for Claim 4.12 are met, i.e.,

$$(48(12R/r) + 750Rn^{1.5})\eta \leq \frac{1}{100k^2} \min_{k \in [n]: a_k \neq 0} |a_k|,$$

we get $\mathsf{FindInfBuckets}(\textsc{Approximate-}g, B, [r])$ will return at most $8k$-influential variables of f with probability at least $1 - 128\lceil\log k\rceil^2/5k$.

Since Test-k-Linear accepts f with probability $\geq \delta$, this implies that the total number of influential variables returned by $\mathsf{FindInfBuckets}$ is at most k, with probability $\geq \delta$. Combining the above, we can conclude that f is pointwise $(48(12R/r) + 750Rn^{1.5})\eta$-close to a k-linear function, with probability at least $\delta - \varepsilon/2 - 128\lceil\log k\rceil^2/5k$. This concludes the soundness argument.

Query complexity: From Theorem 4.4, we know that Approximate Additivity Tester performs $O(\frac{1}{\varepsilon})$ queries. Following Claim 4.12, we also know that $\mathsf{FindInfBuckets}$ performs $\widetilde{O}(k \log k)$ queries. Combining them, we have: Test-k-Linear performs $\widetilde{O}(k \log k + {}^1/_\varepsilon)$ queries in total. $\square$

5 k-Sparse Low Degree Testing

In this section, we present and analyze our sparse low degree tester (Algorithm 4).

Theorem 5.1 (Generalization of Theorem 1.5)**.** *Let $\eta < \min\{\varepsilon, 1/2^{2^n}\}$. Given η-approximate query access to $f : \mathbb{R}^n \to \mathbb{R}$ that is bounded in $\mathrm{B}(\mathbf{0}, 2d\sqrt{n})$, there exists a tester* TEST-k-SPARSE *(Algorithm 4) that performs $\widetilde{O}(d^5 + d^2/\varepsilon + dk^3)$ queries and guarantees:*

- COMPLETENESS: *If f is a k-sparse, degree-d polynomial, then Algorithm 4* ACCEPTS *with probability at least $1 - \varepsilon/4$.*

- SOUNDNESS: *If f is ε-far from all k-sparse degree-d polynomials, then Algorithm 4* REJECTS *with probability at least $2/3$.*

Algorithm 4: TEST-k-SPARSE

Inputs: η-approximate query oracle $\widetilde{f}$ for $f : \mathbb{R}^n \to \mathbb{R}$, that is bounded in $\mathrm{B}(\mathbf{0}, R)$, sparsity parameter $k \in \mathbb{N}$, proximity parameter $\varepsilon \in (0, 1)$, degree parameter $d \in \mathbb{N}$.
Output: ACCEPT iff f is k-sparse function.
Run APPROXLOWDEGREETESTER$(f, d, \mathcal{N}(\mathbf{0}, I_n), \eta, \varepsilon, R, R)$ (Algorithm 9).
if *Algorithm 9 rejects* **then**
  return REJECT.
Call APPROX-POLY-SPARSITY-TEST(APPROXQUERY-g) (Algorithm 5).
if APPROX-POLY-SPARSITY-TEST*(*APPROXQUERY-g*) accepts* **then**
  return ACCEPT. ▷ APPROXQUERY-g in Algorithm 10
else
  return REJECT.

We will prove Theorem 5.1 in Section 5.3, after developing the necessary machinery. Algorithm 4 first invokes APPROXLOWDEGREETESTER (Algorithm 9) to reject functions that are far from any low-degree polynomial. We record some useful claims about Algorithm 9, and its subroutines (Algorithm 10) from [ABF+23] and (its improvement in) [AKM25]:

Theorem 5.2 ([AKM25, Theorem 3.6])**.** *Let $d \in \mathbb{N}$, for $L > 0$, $f : \mathbb{R}^n \to \mathbb{R}$ be bounded in the ball $\mathrm{B}(\mathbf{0}, L)$, and for $\varepsilon \in (0, 1), R > 0$, let $\mathcal{D}$ be an $(\varepsilon/4, R)$-concentrated distribution. For $\alpha > 0, \beta \geq 2^{(2n)^{O(d)}}(R/L)^d\alpha$, given α-approximate query access to f, and sampling access to $\mathcal{D}$, there is an one-sided error, $O(d^5 + \frac{d^2}{\varepsilon})$-query* APPROXLOWDEGREETESTER *(Algorithm 9) which, distinguishes between the case when f is pointwise α-close to some degree-d polynomial and the case when, for every degree-d polynomial $h \colon \mathbb{R}^n \to \mathbb{R}$, $\Pr_{\boldsymbol{p}\sim\mathcal{D}}[|f(\boldsymbol{p}) - h(\boldsymbol{p})| > \beta] > \varepsilon$.*

Lemma 5.3 ([ABF+23, Lemma 4.4])**.** *Let $r = (4d)^{-6}$, $\delta = 2^{d+1}\alpha$, as set in Algorithm 10, and $R > r$. If* APPROXCHARACTERIZATIONTEST *fails with probability at most $2/3$, then g is pointwise $2^{(2n)^{45d}}(R/L)^d\delta$-close to a degree-$d$ polynomial in $\mathrm{B}(\mathbf{0}, 2dR\sqrt{n}/L)$. Furthermore, for every point $\boldsymbol{p} \in \mathrm{B}(\mathbf{0}, 2dR\sqrt{n}/L)$,* APPROXQUERY-$g(\boldsymbol{p})$ *well approximates $g(\boldsymbol{p})$ with high probability, that is,*

$$\Pr_{\boldsymbol{p}\sim\mathcal{D}}\left[|g(\boldsymbol{p}) - \text{APPROXQUERY-}g(\boldsymbol{p})| \leq \left(\frac{24dR\sqrt{n}}{Lr}\right)^d 2^{d+4}\delta\right] \geq 1 - \frac{\varepsilon}{4}.$$

To invoke these results, we assume: (i) f is bounded in $\mathrm{B}(\mathbf{0}, R)$, i.e., we set $L = R$, and (ii) $\alpha = \eta$. Additionally, since we work over standard Gaussians, we set $R = 2d\sqrt{n}$.

5.1 Testing sparsity of polynomials given exact query access

In this section, we design an algorithm for testing sparsity of *polynomial functions* $f : \mathbb{R}^n \to \mathbb{R}$, by extending the machinery developed in [GJR10] and [BOT88] to the real numbers.

Let us first define the notion of Hankel Matrices associated with polynomials.

Definition 5.4 (Hankel Matrix for polynomials [GJR10, BOT88]). Consider any $\boldsymbol{u} \triangleq (u_1, \ldots, u_n) \in \mathbb{R}^n$, and define $\boldsymbol{u}^i \triangleq (u_1^i, \ldots, u_n^i) \in \mathbb{R}^n, \forall i \in \mathbb{N}$. For a function $f : \mathbb{R}^n \to \mathbb{R}$ and an integer $t \in \mathbb{Z}_{>0}$, define the t-dimensional *Hankel matrix* associated with f at $\boldsymbol{u}$ to be the following:

$$H_t(f, \boldsymbol{u}) \triangleq \begin{pmatrix} f(\boldsymbol{u}^0) & f(\boldsymbol{u}^1) & \ldots & f(\boldsymbol{u}^{t-1}) \\ f(\boldsymbol{u}^1) & f(\boldsymbol{u}^2) & \ldots & f(\boldsymbol{u}^t) \\ \vdots & \vdots & \ddots & \vdots \\ f(\boldsymbol{u}^{t-1}) & f(\boldsymbol{u}^t) & \ldots & f(\boldsymbol{u}^{2t-2}) \end{pmatrix} \in \mathbb{R}^{t \times t}.$$

Note that if we have exact query access to f, the Hankel matrix $H_t(f, \boldsymbol{u})$ can be computed using $2t-1$ queries to f for any point $\boldsymbol{u} \in \mathbb{R}^n$.

We note the following observation about Hankel matrices. (It follows essentially the same argument as in [BOT88, GJR10], since the decomposition that they use over finite domains also works over the reals.) For completeness, a proof of this observation is provided in Appendix B.

Observation 5.5 (Generalization of [BOT88, Section 4], and [GJR10, Lemma 4]). *Let $f : \mathbb{R}^n \to \mathbb{R}$ be an exactly k-sparse polynomial over the reals, i.e., $f(\boldsymbol{x}) = \sum_{i=1}^{k} a_i M_i(\boldsymbol{x})$, where $a_1, \ldots, a_k \in \mathbb{R} \setminus \{0\}$, and $M_1, \ldots, M_k$ are the monomials of f. Then for all $\ell + 1 \leq k$,*

$$\det\left(H_{\ell+1}(f, \boldsymbol{x})\right) = \sum_{\substack{S \subseteq [k] \\ |S| = \ell+1}} \prod_{i \in S} a_i \prod_{\substack{i,j \in S \\ i<j}} (M_j(\boldsymbol{x}) - M_i(\boldsymbol{x}))^2,$$

is a non-zero polynomial of degree $\leq 2\binom{\ell+1}{2}\deg(f)$ in $\boldsymbol{x}$, while for all $\ell + 1 > k$, $\det\left(H_{\ell+1}(f, \boldsymbol{x})\right) \equiv 0$.

As a preliminary, we argue that Observation 5.5 can be used to test whether a polynomial f is $(\leq k)$-sparse, or not (with error probability 0, in fact!), given exact query access to f.

Lemma 5.6. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a polynomial function (of any degree). Given exact query access to f, there is an algorithm that makes $2k+1$ queries to f, and exactly tests whether $\|f\|_0 \leq k$ (returns* Accept*), or $\|f\|_0 > k$ (returns* Reject*) with error probability* 0.

Proof. The algorithm is as follows: Sample a single point $\boldsymbol{u} \sim \mathcal{N}(\boldsymbol{0}, I_n)$. Construct the Hankel matrix $H_{k+1}(f, \boldsymbol{u})$ as in Definition 5.4 using $2k-1$ exact queries to f. Test whether $\det\left(H_{k+1}(f, \boldsymbol{u})\right) \stackrel{?}{=} 0$ and return Accept if the determinant is 0, and Reject otherwise.

Let $Q(\boldsymbol{x}) \equiv \det(H_{k+1}(f, \boldsymbol{x})) \in \mathbb{R}[\boldsymbol{x}]$, so that the test is $Q(\boldsymbol{u}) \stackrel{?}{=} 0$. If $\|f\|_0 \leq k$, by Observation 5.5, Q is the zero polynomial and hence the algorithm will always Accept after finding that $Q(\boldsymbol{u}) = 0$. If $\|f\|_0 \geq k+1$, then Q is a non-zero polynomial, and hence $\Pr_{\boldsymbol{u} \sim \mathcal{N}(0, I_n)}[Q(\boldsymbol{u}) = 0] = 0$; this is because the Lebesgue measure, and hence the probability measure with respect to any continuous distribution, of the zero-set of any non-zero real polynomial is 0. Thus, the algorithm will Reject with probability 1 over the randomness of $\boldsymbol{u}$. $\square$

5.2 Testing sparsity of polynomials given approximate query access

Now, instead of exact query access, we have η-approximate query access to a polynomial f. We make the following observation about the Hankel matrix constructed with the queried values $\tilde{f}$, since the approximate-query oracle guarantees that $|\tilde{f}(\boldsymbol{z}) - f(\boldsymbol{z})| \leq \eta$ for all points $\boldsymbol{z} \in \mathbb{R}^n$. We provide a proof of this observation in Appendix B.

Observation 5.7. *Let $f : \mathbb{R}^n \to \mathbb{R}$ and let $\tilde{f}$ be an η-approximate query oracle to f. Then for any $t \geq 1$ and $\boldsymbol{u} \in \mathbb{R}^n$, we can express*

$$H_t(\tilde{f}, \boldsymbol{u}) = H_t(f, \boldsymbol{u}) + E_t(\boldsymbol{u}),$$

where $E_t(\boldsymbol{u})$ is a Hankel-structured, noise matrix*, such that $\|E_t(\boldsymbol{u})\|_\infty \leq \eta$, and $\|E_t(\boldsymbol{u})\|_{\mathrm{op}} \leq \eta t$.*

We also need the following probabilistic upper bound on the eigenvalues of the Hankel matrix of a polynomial, that was briefly introduced in Section 1.1.2. We restate it here for convenience.

Theorem 1.6 (Probabilistic Upper Bound on $\sigma_{\max}$)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$, $f(\boldsymbol{x}) = \sum_{i=1}^k a_i M_i(\boldsymbol{x})$ be a k-sparse, degree-d polynomial, where M_i's are its non-zero monomials, and $\sigma_{\max}(H_t(f, \boldsymbol{u}))$ denote the largest singular value of the t-dimensional Hankel matrix associated with f at a point $u \in \mathbb{R}^n$, $H_t(f, \boldsymbol{u})$. Then, for any $\gamma \in (0, 1)$, with $\boldsymbol{a} \triangleq (a_1, \cdots, a_k)^\top \in \mathbb{R}^k$,*

$$\Pr_{\boldsymbol{u} \sim \mathcal{N}(\boldsymbol{0}, I_n)} \left[\sigma_{\max}(H_t(f, \boldsymbol{u})) \geq \|\boldsymbol{a}\|_2^2 \left(2^{d/2} \lceil d/2 \rceil ! + \sqrt{\frac{k}{\gamma}} 2^{d/2} \sqrt{d!} \right)^{2t} \right] \leq \gamma.$$

Proof of Theorem 1.6. Let us denote $H_t(f, \boldsymbol{u})$ by $H_{\boldsymbol{u}}$. Then for any $\boldsymbol{z} = (z_1, \ldots, z_t)^\top \in \mathbb{R}^t$, we have

$$\begin{aligned}
|\boldsymbol{z}^\top H_{\boldsymbol{u}} \boldsymbol{z}| &= \left| \sum_{i \in [t]} \sum_{j \in [t]} [H_{\boldsymbol{u}}]_{i,j} z_i z_j \right| \leq \sum_{i \in [t]} \sum_{j \in [t]} \left| f(\boldsymbol{u}^{i+j-2}) \right| |z_i| |z_j| \qquad (\because [H_{\boldsymbol{u}}]_{i,j} = f(\boldsymbol{u}^{i+j-2})) \\
&\leq \sum_{i \in [t]} \sum_{j \in [t]} |z_i| |z_j| \left(\sum_{p \in [k]} |a_p| |M_p(\boldsymbol{u}^{i+j-2})| \right) \\
&= \overline{\boldsymbol{z}}^\top V^\top D V \overline{\boldsymbol{z}}, \text{ where } \overline{\boldsymbol{z}} = (|z_1|, \ldots, |z_t|)^\top \in \mathbb{R}^t_{\geq 0}, \\
V &= \begin{pmatrix} 1 & 1 & \ldots & 1 \\ |M_1(\boldsymbol{u})| & |M_2(\boldsymbol{u})| & \ldots & |M_k(\boldsymbol{u})| \\ |M_1(\boldsymbol{u})|^2 & |M_2(\boldsymbol{u})|^2 & \ldots & |M_k(\boldsymbol{u})|^2 \\ \vdots & \vdots & \ddots & \vdots \\ |M_1(\boldsymbol{u})|^{t-1} & |M_2(\boldsymbol{u})|^{t-1} & \ldots & |M_k(\boldsymbol{u})|^{t-1} \end{pmatrix}^\top, \text{ and } D = \begin{pmatrix} |a_1| & 0 & \ldots & 0 \\ 0 & |a_2| & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & |a_k| \end{pmatrix},
\end{aligned}$$

wherein in the last step, we use the same decomposition as in the proof of Observation 5.5, but applied to $H_t(|f|, \boldsymbol{u})$, with $|f|(\boldsymbol{u}) \triangleq \sum_{p \in [k]} |a_p| |M_p(\boldsymbol{u}^{i+j-2})|$.

Thus, if $\|\overline{\boldsymbol{z}}\|_2 = \|\boldsymbol{z}\|_2 \leq 1$, we have

$$\sigma_{\max}(H_{\boldsymbol{u}}) \leq \overline{\boldsymbol{z}} V^\top (D^{\frac{1}{2}})^\top D^{\frac{1}{2}} V \overline{\boldsymbol{z}} = \left\| D^{\frac{1}{2}} V \overline{\boldsymbol{z}} \right\|_2^2 \leq \|D^{\frac{1}{2}} V\|_{\mathrm{op}}^2 \|\overline{\boldsymbol{z}}\|_2^2 \leq \|D^{\frac{1}{2}} V\|_{\mathrm{op}}^2 \leq \|D^{\frac{1}{2}} V\|_{\mathrm{F}}^2, \text{ where}$$

$$D^{\frac{1}{2}} V = \begin{pmatrix} |a_1| & |a_1||M_1(\boldsymbol{u})| & \cdots & |a_1||M_1(\boldsymbol{u})|^{t-1} \\ |a_2| & |a_2||M_2(\boldsymbol{u})| & \cdots & |a_2||M_2(\boldsymbol{u})|^{t-1} \\ \vdots & \vdots & \ddots & \vdots \\ |a_k| & |a_k||M_k(\boldsymbol{u})| & \cdots & |a_k||M_k(\boldsymbol{u})|^{t-1} \end{pmatrix}.$$

By $U \geq \max\left\{2, \max_{p\in[k]} |M_p(\boldsymbol{u})|\right\}$, we get $\|D^{\frac{1}{2}}V\|_{\mathrm{F}}^2 \leq \sum_{r=0}^{t-1} \|\boldsymbol{a}\|_2^2 U^{2r} = \|\boldsymbol{a}\|_2^2 \left(\frac{U^{2t}-1}{U-1}\right) \leq \|\boldsymbol{a}\|_2^2 \cdot U^{2t}$.

Finally, for any $\boldsymbol{\alpha} \triangleq (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n$, we have $\mathbb{E}_{\boldsymbol{u}\sim\mathcal{N}(\mathbf{0},I_n)}[|\boldsymbol{u}^{\boldsymbol{\alpha}}|] = \prod_{i\in[n]} \mathbb{E}_{u_i\sim\mathcal{N}(0,1)}[|u_i^{\alpha_i}|]$. From [Ela61, Equations (1), (4), and (15)], for $s \in \mathbb{N}_{>0}$, we get

$$\mathop{\mathbb{E}}_{u\sim\mathcal{N}(0,1)}[|u^s|] = \begin{cases} \frac{s!}{2^{s/2}(s/2)!} \leq \frac{2^{s/2}\lfloor s/2\rfloor!}{\sqrt{\pi}} & \text{, if } s \text{ is even (and } > 0\text{), and} \\ 2^{\lfloor s/2\rfloor}\left(\lfloor \frac{s}{2}\rfloor\right)!\sqrt{\frac{2}{\pi}} = \frac{2^{s/2}\lfloor s/2\rfloor!}{\sqrt{\pi}} & \text{, if } s \text{ is odd.} \end{cases} \tag{2}$$

The inequality in the "s is even" case follows from: (i) $\frac{s!}{2^s(s/2)!} = \frac{\Gamma\left(\frac{s}{2}+\frac{1}{2}\right)}{\sqrt{\pi}}$ [Wol24], and (ii) the monotonicity of the Gamma function in $(\frac{1}{2}, \infty)$ which implies $\Gamma\left(\frac{s}{2}+\frac{1}{2}\right) \leq \Gamma\left(\frac{s}{2}+1\right) = \frac{s}{2}!$ for even s.

Since $\mathbb{E}_{u\sim\mathcal{N}(0,1)}[|u^0|] = 1$, for any $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n$ with $\|\boldsymbol{\alpha}\|_1 = \sum_{i\in[n]} \alpha_i \leq d$, we have

$$\mathop{\mathbb{E}}_{\boldsymbol{u}\sim\mathcal{N}(\mathbf{0},I_n)}[|\boldsymbol{u}^{\boldsymbol{\alpha}}|] \leq \prod_{i:\alpha_i\neq 0} \left(\frac{2^{\alpha_i/2}\lfloor \alpha_i/2\rfloor!}{\sqrt{\pi}}\right) \leq 2^{d/2}\lceil d/2\rceil!.$$

By a similar argument, $\mathrm{Var}[|\boldsymbol{u}^{\boldsymbol{\alpha}}|] \leq \mathbb{E}_{\boldsymbol{u}}[|\boldsymbol{u}^{2\boldsymbol{\alpha}}|] \leq 2^d \cdot d!$.

Thus, by Chebyshev's inequality, for any $d \geq 0$, $k \geq 1$, and $\boldsymbol{\alpha} \in \mathbb{N}^n$ with $\|\boldsymbol{\alpha}\|_1 \leq d$,

$$\Pr_{\boldsymbol{u}\sim\mathcal{N}(\mathbf{0},I_n)}\left[|\boldsymbol{u}^{\boldsymbol{\alpha}}| \geq 2^{d/2}\lceil d/2\rceil! + \sqrt{\frac{k}{\gamma}}2^{d/2}\sqrt{d!}\right] \leq \frac{\gamma}{k}.$$

Thus, from the above discussion, $M_p(\boldsymbol{u}) \leq \left(2^{d/2}\lceil d/2\rceil! + \sqrt{\frac{k}{\gamma}}2^{d/2}\sqrt{d!}\right)$ for every $p \in [k]$, w.p $\geq 1-\gamma$ (using the union bound), and hence $\sigma_{\max}(H_t(f,\boldsymbol{u})) \leq \|\boldsymbol{a}\|_2^2\left(2^{d/2}\lceil d/2\rceil! + \sqrt{\frac{k}{\gamma}}2^{d/2}\sqrt{d!}\right)^{2t}$. $\square$

We now present and analyze the correctness of Algorithm 5:

Algorithm 5: APPROX-POLY-SPARSITY-TEST: Approx-query sparsity test for polynomials

```
Input: η-approximate query oracle f̃ to a polynomial f : ℝ^n → ℝ of total degree ≤ d,
  sparsity parameter k ∈ ℕ.
Output: ACCEPT iff f is a k-sparse polynomial.
Let T ← 4 min{C, 1} · d(k+1)^2, where C is the absolute constant in Theorem 3.17.
for t ∈ [T] do
    Sample u_t = (u_{t,1}, ..., u_{t,n}) ~ N(0, I_n).
    for i = 0, ..., 2k do
        Compute u_t^i ≜ (u_{t,1}^i, ..., u_{t,n}^i), and f̃(u_t^i).
    Compute the Hankel matrix H_{k+1}(f̃, u_t) as in Definition 5.4.
    Compute the smallest singular value σ_min^(t) of H_{k+1}(f̃, u_t).
if σ_min^(t) ≤ η(k+1) for all t ∈ [T] then
    return ACCEPT ((≤ k)-sparse).
else
    return REJECT ((> k)-sparse).
```

Theorem 5.8 (Sparsity testing of polynomials with approximate queries)**.** *Given η-approximate query access $\widetilde{f}$ to a polynomial $f : \mathbb{R}^n \to \mathbb{R}$ of total degree $\leq d$, assuming $\eta \leq \frac{\left(\mathrm{coeff}_{d_Q}(Q)\right)^{\frac{1}{2}}}{(2(k+1))2^{\Theta(k^3 d)}(\sigma_{\max}(H_{\boldsymbol{u}}))^k}$, wherein $H_{\boldsymbol{u}} = H_{k+1}(f, \boldsymbol{u}), Q_{\boldsymbol{u}} = (\det(H_{\boldsymbol{u}}))^2, d_Q = \deg(Q)$, and $\mathrm{coeff}^2_{d_Q}(Q)$ is as in Theorem 3.17,* Approx-Poly-Sparsity-Test *(Algorithm 5) performs at most $O(dk^3)$ queries and guarantees:*

(i) If f has sparsity at most k, the algorithm always Accepts, *and*

(ii) If f has sparsity $> k$, the algorithm Rejects *with probability at least $\frac{2}{3}$.*

Proof. The query complexity follows directly from Algorithm 5, since the tester performs at most $(2k+1)T$ queries for $T \leq O(k^2 d)$. It remains to argue the completeness and soundness guarantees.

For any $\boldsymbol{u} \in \mathbb{R}^n$, let us denote $H_{k+1}(\tilde{f}, \boldsymbol{u})$ by $\tilde{H}_{\boldsymbol{u}}$ and $H_{k+1}(f, \boldsymbol{u})$ by $H_{\boldsymbol{u}}$. Observe, since $\tilde{H}_{\boldsymbol{u}}$ and $H_{\boldsymbol{u}}$ are symmetric matrices, their singular values are just the absolute values of their eigenvalues, i.e., $\sigma_{\min}(\tilde{H}_{\boldsymbol{u}}) = \min_{i \in [k+1]} |\lambda_i(\tilde{H}_{\boldsymbol{u}})|$ and $\sigma_{\min}(H_{\boldsymbol{u}}) = \min_{j \in [k+1]} |\lambda_j(H_{\boldsymbol{u}})|$.

Completeness: We have $\|f\|_0 \leq k$. By Observation 5.5, we get $\det(H_{k+1}(f, \boldsymbol{x})) \equiv 0$, giving us: for any $\boldsymbol{u} \in \mathbb{R}^n$, $\det(H_{\boldsymbol{u}}) = 0$, and hence, $\lambda_{i^*}(H_{\boldsymbol{u}}) = 0$ for some $i^* \in [k+1]$. From Observation 5.7, we can write $\tilde{H}_{\boldsymbol{u}} = H_{\boldsymbol{u}} + E$, for a symmetric matrix E with $\|E\|_{\mathrm{op}} \leq \eta(k+1)$. Then, by Weyl's inequality (Theorem 3.15), we have $|\lambda_{i^*}(\tilde{H}_{\boldsymbol{u}}) - \lambda_{i^*}(H_{\boldsymbol{u}})| \leq \eta(k+1)$, which implies $\sigma_{\min}(\tilde{H}_{\boldsymbol{u}}) \leq |\lambda_{i^*}(\tilde{H}_{\boldsymbol{u}})| \leq \eta(k+1)$. So, Algorithm 5 will always Accept (Lines 10–11).

Observe: For any non-singular symmetric matrix $A \in \mathbb{R}^{(k+1)\times(k+1)}$, we have

$$(\sigma_{\min}(A))^{2(k+1)} \leq (\det(A))^2 = \prod_{i=1}^{k+1} (\sigma_i(A))^2 \leq (\sigma_{\min}(A))^2 (\sigma_{\max}(A))^{2k},$$

where $\sigma_{\max}(A)$ is the largest magnitude eigenvalue of A and $\sigma_{\min}(A)$ is the smallest magnitude eigenvalue of A. Thus, $\sigma_{\min}(A) \leq \Xi$ implies $\det(A)^2 \leq (\Xi)^2 \sigma_{\max}(A)^{2k}$.

Soundness: We have $\|f\|_0 > k$. Let $Q(\boldsymbol{x}) \triangleq (\det(H_{k+1}(f, \boldsymbol{x})))^2$. By Observation 5.5, Q is a non-zero polynomial with total degree $\leq 2(k+1)^2 d$, and $(\det(H_{\boldsymbol{u}}))^2 = Q(\boldsymbol{u})$ for all $\boldsymbol{u} \in \mathbb{R}^n$.

Let $\boldsymbol{u} \sim \mathcal{N}(\boldsymbol{0}, I_n)$, and $\sigma_{\min}(\tilde{H}_{\boldsymbol{u}}) \leq \eta(k+1)$. Then, as in completeness, by Weyl's Inequality (Theorem 3.15), and Observation 5.7, we have $\sigma_{\min}(H_{\boldsymbol{u}}) \leq 2\eta(k+1)$. Thus, we must have

$$Q(\boldsymbol{u}) = (\det(H_{\boldsymbol{u}}))^2 \leq (2\eta(k+1))^2 \sigma_{\max}(H_{\boldsymbol{u}})^{2k} \triangleq \Delta,$$

as long as $\det(H_{\boldsymbol{u}}) \neq 0$ (which will happen with probability 1 over the choice of $\boldsymbol{u}$, since $H_{\boldsymbol{u}} \not\equiv 0$).

First, suppose that f is not a constant polynomial. Then, $d_Q \triangleq \deg(Q) \geq 2$ by construction, since it is the square of a non-constant polynomial. Now, we may invoke an anti-concentration result.

From Theorem 3.17, we have $\Pr_{\boldsymbol{u} \sim \mathcal{N}(\boldsymbol{0}, I_n)}[Q(\boldsymbol{u}) \leq \Delta] \leq C d_Q \left(\frac{\Delta}{\mathrm{coeff}_{d_Q}(Q)}\right)^{1/d_Q}$. Observe,

$$C d_Q \left(\frac{\Delta}{\mathrm{coeff}_{d_Q}(Q)}\right)^{1/d_Q} \leq 1 - \frac{1}{C d_Q} \iff \left(\frac{\Delta}{\mathrm{coeff}_{d_Q}(Q)}\right) \leq \left(\frac{1}{C d_Q} - \left(\frac{1}{C d_Q}\right)^2\right)^{d_Q}.$$

Assuming $C \geq 1$, so that $1/(C d_Q) \leq 1/2$ (since $d_Q \geq 2$) [2], we have

[2] We renormalize the constant from Theorem 3.17 appropriately.

$$\ln\left(\left(\frac{1}{Cd_Q}-\left(\frac{1}{Cd_Q}\right)^2\right)^{d_Q}\right) = d_Q\left[\ln\left(1-\frac{1}{Cd_Q}\right)-\ln(Cd_Q)\right] \geq d_Q\left[\frac{-1/(Cd_Q)}{1-1/(Cd_Q)}-\ln(Cd_Q)\right]$$
$$\geq -d_Q\ln(Cd_Q)-\frac{2}{Cd_Q} \geq -(d_Q\ln(Cd_Q)+1),$$

where we use the inequality $\ln(1+z) \geq \frac{z}{1+z}$ for $z > -1$, and the fact that $1/(Cd_Q) \leq 1/2$. Then

$$\left(\frac{1}{Cd_Q}-\left(\frac{1}{Cd_Q}\right)^2\right)^{d_Q} \geq \exp\left(-(d_Q\ln(Cd_Q)+1)\right) \geq \underbrace{\exp\left(-2(k+1)^2 d\ln(2C(k+1)^2 d)+1\right)}_{\triangleq \mathsf{UB}_{d,k}} \geq 2^{-\Theta(k^3 d)}.$$

Thus, if $\Delta \leq \mathsf{UB}_{d,k}\cdot \mathrm{coeff}_{d_Q}(Q)$, we will have $\Pr_{\boldsymbol{u}\sim\mathcal{N}(\mathbf{0},I_n)}[Q(\boldsymbol{u}) \leq \Delta] \leq 1-\frac{1}{Cd_Q}$. If $\eta \leq \frac{\left(\mathrm{coeff}_{d_Q}(Q)\right)^{\frac{1}{2}}}{(2(k+1))2^{\Theta(k^3 d)}(\sigma_{\max}(H_{\boldsymbol{u}}))^k} \leq \frac{\left(\mathsf{UB}_{d,k}\cdot\mathrm{coeff}_{d_Q}(Q)\right)^{\frac{1}{2}}}{(2(k+1))(\sigma_{\max}(H_{\boldsymbol{u}}))^k}$, then $\Delta = (2\eta(k+1))^2\sigma_{\max}(H_{\boldsymbol{u}})^{2k} \leq \mathsf{UB}_{d,k}\cdot\mathrm{coeff}_{d_Q}(Q)$.

With the above condition satisfied, we will have

$$\Pr_{\boldsymbol{u}}\left[\sigma_{\min}(\tilde{H}_{\boldsymbol{u}}) \leq \eta(k+1)\right] \leq \Pr_{\boldsymbol{u}}\left[\sigma_{\min}(H_{\boldsymbol{u}}) \leq 2\eta(k+1)\right] \leq \Pr_{\boldsymbol{u}}[Q(\boldsymbol{u}) \leq \Delta] \leq 1-\frac{1}{Cd_Q}.$$

Now, consider the operation of Algorithm 5. In every iteration $t \in [T]$ for $T = 4\min\{C,1\}\cdot d(k+1)^2 \geq 2Cd_Q$, we will have $\Pr_{\boldsymbol{u}_t}\left[\sigma_{\min}(\tilde{H}_{\boldsymbol{u}_t}) \leq \eta(k+1)\right] \leq 1-1/(Cd_Q)$ by the above argument. Then, over T independent rounds, the probability that this event occurs every time, i.e., f is accepted, is

$$\prod_{t\in[T]}\Pr_{\boldsymbol{u}_t}\left[\sigma_{\min}(\tilde{H}_{\boldsymbol{u}_t}) \leq \eta(k+1)\right] \leq \left(1-\frac{1}{Cd_Q}\right)^T \leq \frac{1}{e^2} < \frac{1}{3}$$

for our choice of T. Thus, the algorithm will REJECT with probability at least $\frac{2}{3}$. □

5.3 Analysis of k-Sparsity Low Degree Tester

We are now ready to analyze our sparse low degree tester (Algorithm 4).

Proof of Theorem 5.1. Let us start with completeness.

Completeness: Since f is a k-sparse, degree-d polynomial, from Theorem 5.2, we have: APPROX-LOW-DEGREE-TESTER always accepts f, and hence, from Lemma 5.3, we have that g is pointwise $2^{(2n)^{45d}}R^d 2^{d+1}\eta$-close to a degree-$d$ polynomial, say h, in $\mathrm{B}(\mathbf{0},R)$. Moreover,

$$\Pr_{\boldsymbol{p}\sim\mathcal{D}}\left[|g(\boldsymbol{p})-\textsc{ApproxQuery-}g(\boldsymbol{p})| \leq \left(\frac{24d\sqrt{n}}{r}\right)^d 2^{2d+5}\eta \mid \boldsymbol{p}\in\mathrm{B}(\mathbf{0},2d\sqrt{n})\right] \geq 1-\frac{\varepsilon}{2}.$$

$$\implies \Pr_{\boldsymbol{p}\sim\mathcal{D}}\left[|h(\boldsymbol{p})-\textsc{ApproxQuery-}g(\boldsymbol{p})| \leq \left(\left(\frac{24d\sqrt{n}}{r}\right)^d 2^{2d+5}+2^{(2n)^{45d}}R^d 2^{d+1}\right)\eta\right] \geq 1-\frac{\varepsilon}{4}. \quad (3)$$

By setting $\eta = 1/2^{2^n}$, the conditions of Theorem 5.8 are met, giving us that APPROX-POLY-SPARSITY-TEST(APPROXQUERY-g) (Algorithm 5) will ACCEPT with probability $\geq 1-\varepsilon/4$. So with probability at least $1-\varepsilon/4$, TEST-k-SPARSE (Algorithm 4) will accept f.

Soundness: Let f be ε-far from all k-sparse, degree-d polynomials. We will show that if Algorithm 4 accepts with probability at least $1/3$, then f must be ε-close to some k-sparse, degree-d polynomial. From the premise, Approx-Low-Degree-Tester (Algorithm 9) also accepts f with probability at least $1/3$. Then, as in completeness, from Lemma 5.3, we have: g is pointwise $2^{(2n)^{45d}} R^d 2^{d+1}\eta$-close to some degree-$d$ polynomial, say $h(\boldsymbol{x}) = \sum_{\|\boldsymbol{M}_i\|_1 \le d} a_i \boldsymbol{M}_i(\boldsymbol{x})$, i.e., for all $\boldsymbol{p} \in \mathrm{B}(\boldsymbol{0}, R)$, $|g(\boldsymbol{p}) - h(\boldsymbol{p})| \le 2^{(2n)^{45d}} R^d 2^{d+1}\eta$, and (3) still holds.

From the premise, Approx-Poly-Sparsity-Test(ApproxQuery-g) (Algorithm 5) does not reject with probability at least $1/3$. In this case, as long as the closeness of ApproxQuery-g and h satisfies the assumption in Theorem 5.8, h will be k-sparse. The assumption is:

$$(3 \cdot 2^{(2n)^{45d}} + 1)\eta \le \frac{\left(\mathrm{coeff}_{d_Q}(Q)\right)^{\frac{1}{2}}}{(2(k+1))2^{\Theta(k^3 d)} \left(\sigma_{\max}(H_{\boldsymbol{u}})\right)^k}, \tag{4}$$

wherein $H_{\boldsymbol{u}} = H_{k+1}(h, \boldsymbol{u}), Q(\boldsymbol{u}) = (\det(H_{\boldsymbol{u}}))^2, d_Q = \deg(Q)$, $\mathrm{coeff}^2_{d_Q}(Q) \le (\|\boldsymbol{a}\|_2^2 n^d)^{(k+1)}((k+1)!)^2$, and $\sigma_{\max}(H_{\boldsymbol{u}})$ may be bounded using Theorem 1.6 (with $\gamma = 0.01, t = k+1$), i.e.,

$$\Pr_{\boldsymbol{u} \sim \mathcal{N}(\boldsymbol{0}, I_n)} \left[\sigma_{\max}(H_{\boldsymbol{u}}) \ge \|\boldsymbol{a}\|_2^2 \left(2^{d/2} \lceil d/2 \rceil ! + \sqrt{\frac{k}{0.01}} 2^{d/2} \sqrt{d!} \right)^{2(k+1)} \right] \le 0.01.$$

So, with probability at least 0.99, $\sigma_{\max}(H_{\boldsymbol{u}}) \le \|\boldsymbol{a}\|_2^2 \left(2^{d/2}\lceil d/2 \rceil ! + 10\sqrt{k} 2^{d/2}\sqrt{d!}\right)^{2(k+1)}$. Plugging these into (4), we observe, setting $\eta \le 1/2^{2^n}$ satisfies it, implying that f is ε-close to some k-sparse, degree-d polynomial.

Query complexity: The query complexity of Algorithm 4 consists of two parts:

- query complexity of ApproxLowDegreeTester (Algorithm 9) which is $O(d^5 + d^2/\varepsilon)$ (from Theorem 5.2), and
- the query complexity of Approx-Poly-Sparsity-Test (Algorithm 5) which is at most $O(dk^3)$ (from Theorem 5.8).

Combining the above, we can say that Algorithm 4 performs $O(d^5 + d^2/\varepsilon + dk^3)$ queries in total. □

6 k-Junta Testing

In this section, we present and analyze our k-junta tester.

Theorem 6.1 (Generalization of Theorem 1.9). *Given η-approximate query access to an unknown function $f : \mathbb{R}^n \to \mathbb{R}$, a proximity parameter $\varepsilon \in (0,1)$ such that $\eta < \min\{\varepsilon/16k^2, O(1/k^2 \log^2 k)\}$, there exists a one-sided error tester (Algorithm 6) that performs $\widetilde{O}((k \log k)/\varepsilon)$ queries and:*

- **Completeness:** *If f is a k-junta, then Algorithm 6 always* Accepts.
- **Soundness:** *If f is ε-far from all k-juntas, then Algorithm 6* Rejects *with probability at least $2/3$.*

In order to prove Theorem 6.1, we first build some machinery, and then we analyze Algorithm 6 and its subroutine (Algorithm 2 for juntas) in Section 6.1. Let us start with the notion of influence.

Algorithm 6: Test-k-Junta

Inputs: η-Approximate function oracle $\widetilde{f} : \mathbb{R}^n \to \mathbb{R}$, $k \in \mathbb{N}, \varepsilon, \eta \in (0, 1)$.
Output: Output Accept iff f is a k-junta.
Choose a random partition $\mathcal{B}$ of $[n]$ into $r = O(k^2)$ parts.
Initialize $S \leftarrow [r]$, $I \leftarrow \emptyset$.
for $i = 1$ *to* $O(k/\varepsilon)$ **do**
  $B \leftarrow \mathsf{FindInfBucket}(\widetilde{f}, B, S)$ (Algorithm 2).
  if $B \neq \emptyset$ *(*$= B_j$ *for some* $j \in S$*)* **then**
    $I \leftarrow I \cup \{j\}$. $S \leftarrow S \setminus \{j\}$.
  if $|I| > k$ **then**
    return Reject.
return Accept.

Definition 6.2 (Influence)**.** Let $f : \mathbb{R}^n \to \mathbb{R}$ be a function. For any set $S \subseteq [n]$, the influence of f over S with respect to a distribution $\mathcal{D}$ over $\mathbb{R}^n$ is defined as follows:

$$\mathsf{Infl}_f(S) = \mathop{\mathbb{E}}_{\boldsymbol{x},\boldsymbol{y}\sim\mathcal{D}}\left[\left|f(\boldsymbol{y}) - f(\boldsymbol{x}_S\boldsymbol{y}_{\overline{S}})\right|\right].$$

We now prove a structural result. Let $\mathcal{J}_k$ denote the set of all k-juntas on n variables.

Theorem 6.3. *If* $\mathsf{dist}(f, \mathcal{J}_k) \geq \varepsilon$*, then for all* $S \subseteq [n] : |S| \leq k$*, with* $\overline{S} \triangleq [n] \setminus S$*,* $\mathsf{Infl}_f(\overline{S}) \geq \varepsilon$*.*

Proof. Fix some $S \subseteq [n]$, such that $|S| \leq k$. Let $\mathcal{J}_S$ denote the set of all juntas on S. Note $\mathcal{J}_S \subsetneq \mathcal{J}_k$. Define $g_S : \mathbb{R}^n \to \mathbb{R}$ as the junta on S that minimizes the distance from f:

$$g_S \triangleq \arg\min_{g\in\mathcal{J}_S}\{\mathsf{dist}(f, g)\}.$$

Observe that $\mathsf{Infl}_{g_S}(\overline{S}) = 0$. We give a method of constructing such a g_S, coset-wise. For every $\boldsymbol{x} \in \mathbb{R}^{|\overline{S}|}$, define a function $f_{\boldsymbol{x}} : \mathbb{R}^n \to \mathbb{R}$ as $f_{\boldsymbol{x}}(\boldsymbol{y}) \triangleq f(\boldsymbol{x}_{\overline{S}}\boldsymbol{y}_S)$. Observe that $f_{\boldsymbol{x}} \in \mathcal{J}_S$, and hence $f_{\boldsymbol{x}} \in \mathcal{J}_k$, for every $\boldsymbol{x} \in \mathbb{R}^{|\overline{S}|}$. So, from the premise, we have, for every $\boldsymbol{x} \in \mathbb{R}^{|\overline{S}|}$,

$$\mathsf{dist}_{\mathcal{D},\ell_1}(f, f_{\boldsymbol{x}}) = \mathop{\mathbb{E}}_{\boldsymbol{y}\sim\mathcal{D}}\left[|f(\boldsymbol{y}) - f_{\boldsymbol{x}}(\boldsymbol{y})|\right] = \mathop{\mathbb{E}}_{\boldsymbol{y}\sim\mathcal{D}}\left[\left|f(\boldsymbol{y}) - f(\boldsymbol{x}_{\overline{S}}\boldsymbol{y}_S)\right|\right] \geq \varepsilon.$$

Since, the role of $\boldsymbol{x}$ is determined only by the values in the variables in $\overline{S}$, we may as well extend it for all n variables, i.e., for all $\boldsymbol{x} \in \mathbb{R}^n$, we have

$$\mathop{\mathbb{E}}_{\boldsymbol{y}\sim\mathcal{D}}\left[\left|f(\boldsymbol{y}) - f(\boldsymbol{x}_{\overline{S}}\boldsymbol{y}_S)\right|\right] \geq \varepsilon.$$

Now we may sample $\boldsymbol{x}$ from any distribution $\mathcal{D}'$ supported on $\mathbb{R}^n$ as well, i.e.

$$\mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{D}',\boldsymbol{y}\sim\mathcal{D}}\left[\left|f(\boldsymbol{y}) - f(\boldsymbol{x}_{\overline{S}}\boldsymbol{y}_S)\right|\right] \geq \varepsilon.$$

In particular, we may set $\mathcal{D}' = \mathcal{D}$ to get

$$\mathsf{Infl}_f(\overline{S}) = \mathop{\mathbb{E}}_{\boldsymbol{x},\boldsymbol{y}\sim\mathcal{D}}\left[\left|f(\boldsymbol{y}) - f(\boldsymbol{x}_{\overline{S}}\boldsymbol{y}_S)\right|\right] \geq \varepsilon.$$ ◻

Next we prove a lemma that connects the influence of the union of two sets of variables, with the influences of individual sets.

Lemma 6.4 (Sub-additivity of Influence)**.** *For every function $f : \mathbb{R}^n \to \mathbb{R}$, and any $S, T \subseteq [n]$,*

$$\max\{\mathsf{Infl}_f(S), \mathsf{Infl}_f(T)\} \leq \mathsf{Infl}_f(S \cup T) \leq \mathsf{Infl}_f(S) + \mathsf{Infl}_f(T).$$

Proof. From Definition 6.2, we have

$$\mathsf{Infl}_f(S) = \mathop{\mathbb{E}}_{\boldsymbol{x},\boldsymbol{y}\sim\mathcal{D}}\left[\left|f(\boldsymbol{y}) - f(\boldsymbol{x}_S\boldsymbol{y}_{\overline{S}})\right|\right], \ \mathsf{Infl}_f(T) = \mathop{\mathbb{E}}_{\boldsymbol{x},\boldsymbol{y}\sim\mathcal{D}}\left[\left|f(\boldsymbol{y}) - f(\boldsymbol{x}_T\boldsymbol{y}_{\overline{T}})\right|\right], \text{ and}$$

$$\begin{aligned}
\mathsf{Infl}_f(S \cup T) &= \mathop{\mathbb{E}}_{\boldsymbol{x},\boldsymbol{y}\sim\mathcal{D}}\left[\left|f(\boldsymbol{y}) - f(\boldsymbol{x}_{S\cup T}\boldsymbol{y}_{\overline{S\cup T}})\right|\right] \\
&= \mathop{\mathbb{E}}_{\boldsymbol{x},\boldsymbol{y}\sim\mathcal{D}}\left[\left|f(\boldsymbol{y}) - f(\boldsymbol{x}_S\boldsymbol{y}_{\overline{S}}) + f(\boldsymbol{x}_S\boldsymbol{y}_{\overline{S}}) - f(\boldsymbol{x}_{S\cup T}\boldsymbol{y}_{\overline{S\cup T}})\right|\right] \\
&\leq \mathop{\mathbb{E}}_{\boldsymbol{x},\boldsymbol{y}\sim\mathcal{D}}\left[\left|f(\boldsymbol{y}) - f(\boldsymbol{x}_S\boldsymbol{y}_{\overline{S}})\right|\right] + \mathop{\mathbb{E}}_{\boldsymbol{x},\boldsymbol{y}\sim\mathcal{D}}\left[\left|f(\underbrace{\boldsymbol{x}_S\boldsymbol{y}_{\overline{S}}}_{\triangleq\boldsymbol{y}'\sim\mathcal{D}}) - f(\underbrace{\boldsymbol{x}_{S\cup T}\boldsymbol{y}_{\overline{S\cup T}}}_{=\boldsymbol{x}_T\boldsymbol{y}'_{\overline{T}}})\right|\right] \\
&= \mathop{\mathbb{E}}_{\boldsymbol{x},\boldsymbol{y}\sim\mathcal{D}}\left[\left|f(\boldsymbol{y}) - f(\boldsymbol{x}_S\boldsymbol{y}_{\overline{S}})\right|\right] + \mathop{\mathbb{E}}_{\boldsymbol{x},\boldsymbol{y}\sim\mathcal{D}}\left[\left|f(\boldsymbol{y}) - f(\boldsymbol{x}_T\boldsymbol{y}_{\overline{T}})\right|\right] \\
&= \mathsf{Infl}_f(S) + \mathsf{Infl}_f(T).
\end{aligned}$$

Non-negativity of Infl trivially completes the other part of the lemma. ☐

Next we show, if f is far from being a k-junta, the influence of the complement of S in Algorithm 6 (i.e. all buckets not yet identified as influential) is high, unless we identify more than k parts.

Lemma 6.5. *Let $f : \mathbb{R}^n \to \mathbb{R}$ and $\mathcal{B} = \{B_1, \ldots, B_r\}$ be a random partition of $[n]$, for $r = \Theta(k^2)$. If $\mathsf{dist}(f, \mathcal{J}_k) \geq \varepsilon$, with probability $\geq 99/100$ over the randomness of the partition, we have $\mathsf{Infl}_f(\overline{S}) \geq \varepsilon/4$ for any $S \subseteq [n]$ which is a union of at most k parts of $\mathcal{B}$.*

Before proving it, let us first define the notions of intersecting families, and p-biased measure.

Definition 6.6 (Intersecting family)**.** Let $\ell \in \mathbb{N}$. A family of subsets $\mathcal{C}$ of $[n]$ is *ℓ-intersecting* if for any two sets $S, T \in \mathcal{C}$, $|S \cap T| \geq \ell$.

Definition 6.7 (p-biased measure)**.** Let $p \in (0, 1)$. Construct a set $S \subseteq [n]$ by including each index $i \in [n]$ in S with probability p. Then the *p-biased measure* is defined as follows:

$$\mu_p(\mathcal{C}) = \Pr_S[S \in \mathcal{C}].$$

We will use the following result to bound the p-biased measure of intersecting families.

Theorem 6.8 ([DS05, Fri08])**.** *Let $\ell \geq 1$ be an integer and let $\mathcal{C}$ be a ℓ-intersecting family of subsets of $[n]$. For any $p < \frac{1}{\ell+1}$, the p-biased measure of $\mathcal{C}$ is bounded by $\mu_p(\mathcal{C}) \leq p^\ell$.*

Now we are ready to prove Lemma 6.5. We note that the proof is similar to that of [BWY15, Lemma 2]. We are adding it here for completeness.

Proof of Lemma 6.5. We will prove: with high probability over the random partition $\mathcal{I}$, $\mathsf{Infl}_f(\overline{S}) \geq \frac{\varepsilon}{4}$, when S is a union of $\leq k$ parts of $\mathcal{I}$.

Consider the family of all sets whose complements have influence at most $t\varepsilon$, for some $0 \leq t \leq \frac{1}{2}$:

$$\mathcal{F}_t = \{S \subseteq [n] : \mathsf{Infl}_f(\overline{S}) < t\varepsilon\}.$$

Consider any two sets $S_1, S_2 \in \mathcal{F}_{1/2}$, i.e., $\max\{\mathsf{Infl}_f(\overline{S_1}), \mathsf{Infl}_f(\overline{S_2})\} < \varepsilon/2$. By Lemma 6.4, we get

$$\mathsf{Infl}_f(\overline{S_1 \cap S_2}) = \mathsf{Infl}_f(\overline{S_1} \cup \overline{S_2}) \leq \mathsf{Infl}_f(\overline{S_1}) + \mathsf{Infl}_f(\overline{S_2}) < 2\varepsilon/2 = \varepsilon. \tag{5}$$

As $\mathsf{dist}(f, \mathcal{J}_k) \geq \varepsilon$, for every $S \subseteq [n]$ of size $|S| \leq k$, we have $\mathsf{Infl}_f(\overline{S}) \geq \varepsilon$. Comparing with (5), we may infer $|\overline{\overline{S_1} \cup \overline{S_2}}| = |S_1 \cap S_2| > k$. Since we started with two arbitrary sets S_1 and S_2 from $\mathcal{F}_{1/2}$, $\mathcal{F}_{1/2}$ must then be a $(k+1)$-intersecting family (Definition 6.6). Now, consider the two cases:

(i) **$\mathcal{F}_{1/2}$ contains only sets of size at least $2k$.**

We note that $\mathcal{F}_{1/4}$ is a $2k$-intersecting family. To see this, let us assume that it is not the case. Then there exist $T_1, T_2 \in \mathcal{F}_{1/4} : |T_1 \cap T_2| < 2k$. So, by Lemma 6.4, we get

$$\mathsf{Infl}_f(\overline{T_1 \cap T_2}) = \mathsf{Infl}_f(\overline{T_1} \cup \overline{T_2}) \leq \underbrace{\mathsf{Infl}_f(\overline{T_1})}_{<\varepsilon/4} + \underbrace{\mathsf{Infl}_f(\overline{T_2})}_{<\varepsilon/4} < 2\varepsilon/4 < \varepsilon/2.$$

This means that $T_1 \cap T_2 \in \mathcal{F}_{1/2}$ (by definition of $\mathcal{F}_{1/2}$) with $|T_1 \cap T_2| < 2k$, which contradicts our assumption, in this case, that all sets in $\mathcal{F}_{1/2}$ have cardinality $\geq 2k$.

Let $S \subseteq [n]$ be a union of k parts in $\mathcal{I}$. Since $\mathcal{I}$ is a random r-partition of $[n]$, S is a random subset obtained by including each element of $[n]$ in S with probability $\frac{k}{r} \leq \frac{1}{2k+1}$. By Theorem 6.8, we can bound the following:

$$\Pr\left[\mathsf{Infl}_f(\overline{S}) < \frac{\varepsilon}{4}\right] = \Pr[S \in \mathcal{F}_{1/4}] = \mu_{k/r}\left(\mathcal{F}_{1/4}\right) \leq \left(\frac{k}{r}\right)^{2k}.$$

Now we apply the union bound over all possible choices of such S (a union of k parts of $\mathcal{I}$). Then, the probability that at least one such S has $\mathsf{Infl}_f(\overline{S}) < \varepsilon/4$ is at most

$$\binom{r}{k}\left(\frac{k}{r}\right)^{2k} \leq \left(\frac{er}{k}\right)^{k}\left(\frac{k}{r}\right)^{2k} \leq \left(\frac{ek}{r}\right)^{k} = O(k^{-k}) \ll \frac{1}{100}.$$

(ii) **$\mathcal{F}_{1/2}$ contains at least one set of size less than $2k$**

Let $J \in \mathcal{F}_{1/2}$ be such that $|J| < 2k$. For any random partition $\mathcal{I}$ with $\Theta(k^2)$ parts, all the elements of J are in different parts of $\mathcal{I}$, with high probability ($\geq 99/100$). For any $T \in \mathcal{F}_{1/2}$, $|J \cap T| \geq k+1$ (since, $\mathcal{F}_{1/2}$ is a $(k+1)$-intersecting family), and hence T is not covered by any union of k parts of $\mathcal{I}$. That is, for any $S \subseteq [n]$ which is covered by $\leq k$ parts of $\mathcal{I}$, $S \notin \mathcal{F}_{1/2}$ which implies $\mathsf{Infl}_f(\overline{S}) \geq \varepsilon/2$. ☐

6.1 Proof of correctness of Test-k-Junta

As Algorithm 6 invokes Algorithm 2, we first prove its correctness for juntas.

Definition 6.9 (Influential variables and buckets)**.** Let $f : \mathbb{R}^n \to \mathbb{R}$, and $k \in \mathbb{N}$. A variable $\boldsymbol{x}_i, i \in [n]$ is said be *influential* with respect to f, if for some $\boldsymbol{x} \in \mathbb{R}^n$, only changing $\boldsymbol{x}_i$ changes the value of $f(\boldsymbol{x})$. A bucket $B \subseteq [n]$ is said to be influential if it contains the index of at least one influential variable, and non-influential otherwise.

Let $\mathcal{E}$ denote the event that for $\boldsymbol{x}, \boldsymbol{y} \sim \mathcal{N}(\mathbf{0}, I_n)$, $\boldsymbol{x}, \boldsymbol{y} \in \mathrm{B}(\mathbf{0}, 2\sqrt{n})$. By Fact 3.6, we have $\Pr_{\boldsymbol{x},\boldsymbol{y}\sim\mathcal{N}(\mathbf{0},I_n)}[\overline{\mathcal{E}}] \leq 0.01$. Let $\kappa \triangleq \min_{\substack{V\subseteq[n] \\ V \text{ is influential}}} \left\{\mathbb{E}_{\boldsymbol{x},\boldsymbol{y}\sim\mathcal{N}(\mathbf{0},I_n)}\left[|f(\boldsymbol{x}_V\boldsymbol{y}_{\overline{V}}) - f(\boldsymbol{y})| \mid \mathcal{E}\right]\right\}$.

As mentioned before, we use FindInfBucket (Algorithm 2) for testing k-linearity as well. However, the proof of correctness of Algorithm 2 in Section 4, presented in Claim 4.11, holds only for linear functions, which may not be the case in general. So, we prove the same for juntas as well.

Claim 6.10 (Correctness of Algorithm 2 for juntas)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be given via the η-approximate query oracle $\widetilde{f}$, where $\eta \leq \min\{\kappa/4, 1/(1000k^2 \log^2 k)\}$, $B = \{B_1, \ldots, B_r\}$ be a partition of $[n]$, and $\emptyset \neq S \subseteq [r]$. With $\kappa \geq \max\{2\|f\|_{\infty,\mathrm{B}(\mathbf{0},2\sqrt{n})}, \varepsilon/(4k^2)\}$,* FindInfBucket$(f, B, S)$ *guarantees:*

1. *If none of the B_i's are influential,* FindInfBucket(f, B, S) *always returns $\emptyset$ and performs exactly 2 queries to f.*

2. *Otherwise, with probability at least $1 - 16\eta\lceil\log|S|\rceil^2/\kappa$,* FindInfBucket$(f, B, S)$ *returns B_j for some $j \in S$ which is κ-influential and performs $\leq 8\lceil\lg(|S|)\rceil^2$ queries to f.*

Proof. Let $\boldsymbol{x}$ and $\boldsymbol{y}$ be the Gaussian random vectors sampled by Algorithm 2, and $\boldsymbol{w} \triangleq \boldsymbol{y}_{\overline{V}}\boldsymbol{x}_V \in \mathbb{R}^n$.

1. As none of the B_i's are influential, $V (= \{i \in [n] : i \in B_j \text{ for some } j \in S\})$ does not contain any influential variable. So, $f \equiv f\,|_{\overline{V}}$, where $f\,|_{\overline{V}} \colon \mathbb{R}^{[|n]\setminus V|} \to \mathbb{R}$, i.e., for every $\boldsymbol{x} \in \mathbb{R}^n$, with $\boldsymbol{x}_{\overline{V}} \triangleq (\boldsymbol{x}_i, i \notin V) \in \mathbb{R}^{[|n]\setminus V|}$, $f(\boldsymbol{x}) = f\,|_{\overline{V}}\,(\boldsymbol{x}_{\overline{V}})$. So, in Line 5 of Algorithm 2, $f(\boldsymbol{w}) = f(\boldsymbol{y})$, and hence $|\widetilde{f}(\boldsymbol{w}) - \widetilde{f}(\boldsymbol{y})| = |\widetilde{f}(\boldsymbol{w}) - f(\boldsymbol{w}) - (\widetilde{f}(\boldsymbol{y}) - f(\boldsymbol{y}))| \leq \overbrace{|\widetilde{f}(\boldsymbol{w}) - f(\boldsymbol{w})|}^{\leq\eta} + \overbrace{|\widetilde{f}(\boldsymbol{y}) - f(\boldsymbol{y})|}^{\leq\eta} \leq 2\eta$, by the triangle inequality, implying it will always return $\emptyset$, performing exactly 2 queries to f.

2. We will show: When S has some j such that B_j contains a κ-influential variable, then the test in lines 5-10 will fail with probability $1 - 8\eta\lceil\log|S|\rceil/\kappa$. Let $Z \triangleq |f(\boldsymbol{x}_V\boldsymbol{y}_{\overline{V}}) - f(\boldsymbol{y})|$.

   If $\mathbb{E}_{\boldsymbol{x},\boldsymbol{y}}[Z] \geq \kappa$, then, we argue (by anticoncentration) that $\Pr_{\boldsymbol{x},\boldsymbol{y}}[Z > 4\eta] \geq 1 - 8\eta\lceil\log|S|\rceil/\kappa$, and hence $\Pr_{\boldsymbol{x},\boldsymbol{y}}\left[|\widetilde{f}(\boldsymbol{x}_V\boldsymbol{y}_{\overline{V}}) - \widetilde{f}(\boldsymbol{y})| > 2\eta\right]$ with the same probability. To use anticoncentration on Z, we first bound its second moment, conditioned on the event $\mathcal{E}$:

$$\mathop{\mathbb{E}}_{\boldsymbol{x},\boldsymbol{y}\sim\mathcal{N}(\mathbf{0},I_n)}[Z^2 \mid \mathcal{E}] = \mathop{\mathbb{E}}_{\boldsymbol{x},\boldsymbol{y}\sim\mathcal{N}(\mathbf{0},I_n)}[f(\boldsymbol{x}_V\boldsymbol{y}_{\overline{V}})^2 + f(\boldsymbol{y})^2 - 2f(\boldsymbol{x}_V\boldsymbol{y}_{\overline{V}})f(\boldsymbol{y}) \mid \mathcal{E}] \leq 4\|f\|^2_{\infty,\mathrm{B}(\mathbf{0},2\sqrt{n})}.$$

   Then, applying Theorem 3.18 (with $\theta = \frac{4\eta}{\kappa} \in (0,1)$), conditioned on $\mathcal{E}$, we get

$$\Pr[Z \geq 4\eta \mid \mathcal{E}] \geq \Pr\left[Z \geq \frac{4\eta}{\kappa}\underbrace{\mathbb{E}[Z]}_{\geq\kappa} \mid \mathcal{E}\right] \geq \left(1 - \frac{4\eta}{\kappa}\right)^2 \frac{\mathbb{E}^2[Z \mid \mathcal{E}]}{\mathbb{E}[Z^2 \mid \mathcal{E}]} \geq \left(1 - \frac{4\eta}{\kappa}\right)^2 \frac{\kappa^2}{4\|f\|^2_{\infty,\mathrm{B}(\mathbf{0},2\sqrt{n})}}.$$

   From the premise, we have $\kappa \geq 2\|f\|_{\infty,\mathrm{B}(\mathbf{0},2\sqrt{n})}$, giving us $\Pr[Z \geq 4\eta \mid \mathcal{E}] \geq (1 - 4\eta/\kappa)^2$.

   Thus, with probability at least $1 - 8\eta/\kappa$, the condition in Line 5 will not hold, ensuring Algorithm 2 reaches the recursive step (Lines 8–13). By construction, $S = S^{(L)} \sqcup S^{(R)}$, with $1 \leq |S^{(L)}| \leq \lceil|S|/2\rceil$ and $1 \leq |S^{(R)}| \leq \lfloor|S|/2\rfloor$. We have two possibilities:

   (i) $S^{(L)}$ contains no influential buckets and there exists $\emptyset \neq S_R \subseteq S^{(R)}$ such that B_j is influential for all $j \in S_R$.

   (ii) There exists $\emptyset \neq S_L \subseteq S^{(L)}$ such that B_j is influential for all $j \in S_L$.

In Case (i), $\mathsf{FindInfBucket}(f, B, S^{(L)})$ will always return $\emptyset$ and perform 2 queries (by Part 1) and hence the return value of $\mathsf{FindInfBucket}(f, B, S)$ will be $\mathsf{FindInfBucket}(f, B, S^{(R)})$. Using the strong induction hypothesis we get, with probability at least $1 - 8\eta\lceil\log|S|\rceil/\kappa$, this return value will be $\{j\}$ for some $j \in S_R$ and the total number of queries made will be $\leq 2 + 2 + 4\lceil\lg(\lfloor|S|/2\rfloor)\rceil \leq 4 + 4(\lceil\lg|S|\rceil - 1) = 4\lceil\lg|S|\rceil$.

In Case (ii), again using the strong induction hypothesis, with probability at least $1 - 8\eta\lceil\log|S|\rceil/\kappa$, $\mathsf{FindInfBucket}(f, B, S^{(L)})$ will return $\{j\}$ for some $j \in S_L$ and thus the algorithm will return $\{j\}$ (line 13). The number of queries made will be $\leq 2 + 4\lceil\lg(\lceil|S|/2\rceil)\rceil \leq 2 + 4(\lceil\lg|S|\rceil - \frac{1}{2}) = 4\lceil\lg|S|\rceil$.

Note that, if we run $\mathsf{FindInfBucket}(f, B, S)$ for some set S of bucket-indices with $|S| > 1$, each recursive step (irrespective of whether it falls in case (i) or case (ii) above) will succeed with probability $\geq 1 - 8\eta\lceil\log|S|\rceil/\kappa$. For the top level call to succeed, all the recursive calls must succeed. The number of recursive calls made is at most $2\lceil\lg|S|\rceil$. So, we do a union bound over all the failure events to bound the failure probability of the top-level call by $16\eta\lceil\log|S|\rceil^2/\kappa$. ◻

We are now ready to prove the main theorem of this section:

Proof of Theorem 6.1. Let us start with the completeness proof.

Completeness: The argument of completeness of Algorithm 6 holds easily, as when f is a k-junta, at most k of the buckets in the random partition $\mathcal{B}$ will be influential. From Claim 6.10, we get that each execution of the For loop increments I by: either the index of an influential bucket (if it finds one), or nothing, while also removing the bucket from future loop executions. Hence after all its executions, $|I|$ is incremented at most k times, falsifying the check in line 10, making f Accept.

Soundness: Now we prove soundness. Here our goal is to prove if f is ε-far from all k-juntas, then Algorithm 6 rejects it with probability at least $2/3$. In particular, if f is ε-far from being k-junta, and S is the union of at most k-parts of $\mathcal{I}$, then Line 7 of Algorithm 6 will be satisfied with high probability.

By Lemma 6.5, we have: as long as $|I| \leq k$, $\mathsf{Infl}_f(S) \geq \varepsilon/4$ (S is the complement of I with respect to $[n]$, in the execution of Algorithm 6). At a point in the execution, when exactly k influential buckets have been identified, $\mathsf{Infl}_f(S) \geq \varepsilon/4$, $|S| = O(k^2) - k = O(k^2)$, and thus, by Lemma 6.4, there must exist a bucket $B_i, i \in S$ such that $\mathsf{Infl}_f(B_i) \geq \varepsilon/(4k^2)$. This bucket will then be output by $\mathsf{FindInfBucket}(f, B, S)$. So, until $k + 1$ influential buckets have been identified, each invocation of $\mathsf{FindInfBucket}(f, B, S)$ in an execution of its For loop, returns an influential bucket with probability at least $1 - 64\eta\lceil\log k\rceil^2/\kappa$. Treating the outcome of each invocation as a geometric random variable, we conclude: to recover at least $k + 1$ such buckets with probability $\geq 2/3$, $k/\left(1 - 64\eta\lceil\log k\rceil^2/\kappa\right) = k + k\sum_{i=1}^{\infty}(64\eta\lceil\log k\rceil^2/\kappa)^i = O(k/\varepsilon)$ iterations of the For loop suffice. Due to $|I| > k$, in line 10, the algorithm will reject with this same probability.

Query Complexity: From Claim 6.10, we know: each invocation of $\mathsf{FindInfBucket}(f, B, S)$ performs $\leq 4\log|S|$ queries to f. With $O(k/\varepsilon)$ such invocations in total, and $|S| = r = O(k^2)$, the overall query complexity thus is $O((k \log k)/\varepsilon)$. ◻

7 Lower Bounds

In this section, we will present our lower bound results, restated here for convenience:

Theorem 1.10. *Given exact query access to $f : \mathbb{R}^n \to \mathbb{R}$, some $k, d \in \mathbb{N}$ and a distance parameter $\varepsilon \in (0, 1)$, $\Omega\left(\max\{k, 1/\varepsilon\}\right)$ queries are necessary for testing the following properties with probability at least $2/3$:*

(i) k-linearity.

(ii) k-junta.

(iii) k-sparse degree-d polynomials.

Moreover, for testing k-sparse degree-d polynomial, the lower bound is improved to $\Omega\left(\max\{d, k, 1/\varepsilon\}\right)$.

For proving the lower bounds, we follow the approach of [BBM12], i.e., we show the hardness of testing these problems by reducing the Set-Disjointness problem, a canonical hard problem in the field of communication complexity, to them. Proving lower bounds in the exact query model directly gives us lower bounds in the approximate query model. For brevity, we only present the lower bound argument for k-linearity. The other results follow the same approach.

A lower bound of $\Omega(\frac{1}{\varepsilon})$ queries: For any $\varepsilon \in (0, 1)$, it is a folklore result that distinguishing if a function f is linear, or ε-far from being linear in ℓ_1-distance, requires $\Omega(1/\varepsilon)$ queries. An exposition for this is provided in [Fis24].

A lower bound of $\Omega(d)$ queries for k-sparse, degree-d polynomial testing: As before, it is another folklore result that given a proximity parameter $\varepsilon \in (0, 1)$, $\Omega(d)$ queries are necessary to distinguish if an unknown function f is a degree-d polynomial, or a degree-$(d+1)$ (or $(d-1)$) polynomial (and hence, is ε-far from all degree-d polynomials) with probability at least $2/3$.

To prove the $\Omega(k)$ lower bound, we next present a reduction from Set-Disjointness.

7.1 Communication complexity setting

We consider the two-player communication game setting, featuring Alice and Bob. Alice has a function f, and Bob has a function g, and they jointly want to evaluate another function h (which is a function of f and g). We assume that both the players have unbounded computational power, but to evaluate h, they need to communicate among themselves, and the objective is to keep this communication overhead as small as possible.

It is known that it takes linear in the size of the bits of the largest set to solve Set-Disjointness.

Theorem 7.1 ([HW07], Set-Disjointness Lower Bound)**.** *Let Alice and Bob have two sets A and B, respectively, each of size at most k from a universe of size n. In order to distinguish if $|A \cap B| = 1$, or $|A \cap B| = 0$, $\Omega(k)$ bits of communication between Alice and Bob are required.*

7.2 Connection between Communication and Query complexity

Consider two functions f and g, a property $\mathcal{P}$, $h \triangleq f - g$, and a communication problem $C_{h,\mathcal{P}}$: Alice and Bob receive f and g, respectively, and they want to decide if $h \in \mathcal{P}$, or f is ε-far from $\mathcal{P}$. [BBM12] proved that $\mathcal{R}(C_{h,\mathcal{P}})$, the randomized communication complexity of $C_{h,\mathcal{P}}$, is at most twice $Q(\mathcal{P})$, the query complexity of deciding the property $\mathcal{P}$.

Lemma 7.2 ([BBM12, Lemma 2.2])**.** *Let $\mathcal{P}$ be a property of functions, and h be a function. Then $\mathcal{R}(C_{h,\mathcal{P}}) \leq 2Q(\mathcal{P})$.*

Now, given two sets A and B, we construct two functions f and g suitably as follows.

7.3 Construction of the hard instances for the lower bound

Given $A \subseteq [n] : |A| = k$, Alice constructs a polynomial $f = \sum_{i \in A} \boldsymbol{x}_i$. Similarly, Bob constructs a polynomial $g = \sum_{i \in B} \boldsymbol{x}_i$. Note that $f, g : \mathbb{R}^n \to \mathbb{R}$. Let $h = f - g$. Consider the two cases:

(i) $|A \cap B| = 1$**:** In this case h would be a $(2k+2)$ linear function.

(ii) $|A \cap B| = 0$**:** In this case, h would be a $2k$-linear function.

Now we show that under the standard Gaussian distribution $\mathcal{N}(\mathbf{0}, I_n)$, h's corresponding to cases (i), and (ii) above, are sufficiently far in ℓ_1-distance with probability at least $2/3$.

Lemma 7.3. *Let f_1 be a $(2k+2)$-linear function, and f_2 be a $2k$-linear function with coefficients from $\{0, 1\}$. Under the standard Gaussian distribution $\mathcal{N}(\mathbf{0}, I_n)$, f_1 is $\Omega(1)$-far from f_2 in ℓ_1-distance with probability at least $2/3$.*

Proof. Consider the function $g = f_1 - f_2$. Observe that g is at least a 2-linear function, with at least the two terms from f_1 not present in f_2. Without loss of generality, let $g(\boldsymbol{x}) = \boldsymbol{x}_1 + \boldsymbol{x}_2$. We are interested in the event where $|g(\boldsymbol{x})| > O(1)$.

Using Theorem 3.17 (with $d = 1, t = 0$, $g(\boldsymbol{x}) = \boldsymbol{x}_1 + \boldsymbol{x}_2$, and $\mathrm{coeff}_1(g) = \sqrt{2}$), we get: $\forall \varepsilon > 0$,

$$\Pr_{\boldsymbol{x} \sim \mathcal{N}(\mathbf{0}, I)} \left[|g(\boldsymbol{x})| \le \varepsilon \right] \le C \left(\frac{\varepsilon}{\sqrt{2}} \right).$$

Setting $\varepsilon = \frac{1}{2C}$, we have the following:

$$\Pr_{\boldsymbol{x} \sim \mathcal{N}(\mathbf{0}, I)} \left[|g(\boldsymbol{x})| \le \frac{1}{2C} \right] \le \left(\frac{1}{2\sqrt{2}} \right) \le 1/3.$$

Thus, f_1 is at least $\frac{1}{2C}$-far from f_2 with probability at least $2/3$. □

Simulation

Let C_h be the communication problem where Alice and Bob have two sets A and B, respectively, each of size k. As discussed before, they have constructed the two functions f and g, respectively, from A and B. Equivalently, we can also say that Alice and Bob are given two functions f and g respectively, with the promise that the function $h = f - g$ is either a $2k$-linear, or a $(2k+2)$-linear function, and they should accept if and only if h is a $2k$-linear function.

From the above reduction, it is clear that $\mathcal{R}(C_h) \ge \mathcal{R}(\textsc{Set Disjointness}) = \Omega(k)$. By Lemma 7.2, we can say that any testing algorithm that can distinguish between $2k$-linear and $(2k+2)$-linear functions with probability at least $2/3$, requires $\Omega(k)$ queries. Moreover, from Lemma 7.3, we know that any $2k$-linear function is $\Omega(1)$-far from any $(2k+2)$-linear function. Thus, any $2k$-linearity tester (with $\varepsilon = o(1)$) that uses $o(k)$ queries can not distinguish $2k$-linear functions (which should be accepted) and $(2k+2)$-linear functions (which should be rejected) correctly, with probability $\ge 2/3$. This completes the proof of the lower bound of k-linearity testing.

References

[ABCG93] Sigal Ar, Manuel Blum, Bruno Codenotti, and Peter Gemmell. Checking approximate computations over the reals. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of Computing*, pages 786–795, 1993. 3

[ABF+23] Vipul Arora, Arnab Bhattacharyya, Noah Fleming, Esty Kelman, and Yuichi Yoshida. Low Degree Testing over the Reals. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 738–792. SIAM, 2023. https://epubs.siam.org/doi/epdf/10.1137/1.9781611977554.ch31. 2, 4, 7, 13, 19, 38, 39, 40

[AKK+05] Noga Alon, Tali Kaufman, Michael Krivelevich, Simon Litsyn, and Dana Ron. Testing reed-muller codes. *IEEE Transactions on Information Theory*, 51(11):4032–4039, 2005. 2

[AKM25] Vipul Arora, Esty Kelman, and Uri Meir. On Optimal Testing of Linearity. In *Proceedings of the 2025 Symposium on Simplicity in Algorithms (SOSA)*, pages 65–76. SIAM, 2025. https://arxiv.org/abs/2411.14431. 2, 7, 13, 19, 38, 39, 40

[ALM+98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, may 1998. 2

[AS98] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: a new characterization of np. *J. ACM*, 45(1):70–122, jan 1998. 2

[BBBY12] Maria-Florina Balcan, Eric Blais, Avrim Blum, and Liu Yang. Active property testing. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 21–30. IEEE, 2012. 2

[BBM12] Eric Blais, Joshua Brody, and Kevin Matulef. Property testing lower bounds via communication complexity. *computational complexity*, 21:311–358, 2012. 4, 5, 31

[BCP+17] Roksana Baleshzar, Deeparnab Chakrabarty, Ramesh Krishnan S Pallavoor, Sofya Raskhodnikova, and C Seshadhri. Optimal unateness testers for real-valued functions: Adaptivity helps. *arXiv preprint arXiv:1703.05199*, 2017. 2

[BFH+13] Arnab Bhattacharyya, Eldar Fischer, Hamed Hatami, Pooya Hatami, and Shachar Lovett. Every locally characterized affine-invariant property is testable. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 429–436, 2013. 2

[BHK20] Avrim Blum, John Hopcroft, and Ravindran Kannan. *Foundations of Data Science*. Cambridge University Press, 2020. 9

[BK12] Eric Blais and Daniel Kane. Tight bounds for testing k-linearity. In *International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 435–446. Springer, 2012. 4

[BKR23] Hadley Black, Iden Kalemaj, and Sofya Raskhodnikova. Isoperimetric Inequalities for Real-Valued Functions with Applications to Monotonicity Testing. In Kousha Etessami, Uriel Feige, and Gabriele Puppis, editors, *50th International Colloquium on Automata, Languages, and Programming (ICALP 2023)*, volume 261 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 25:1–25:20, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. 2

[BKS+10] Arnab Bhattacharyya, Swastik Kopparty, Grant Schoenebeck, Madhu Sudan, and David Zuckerman. Optimal testing of reed-muller codes. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 488–497. IEEE, 2010. 2

[Bla08] Eric Blais. Improved bounds for testing juntas. In *International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 317–330. Springer, 2008. 5

[Bla09] Eric Blais. Testing juntas nearly optimally. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 151–158, 2009. 3, 5

[BLR90] Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 73–83, 1990. 1, 2, 3, 7

[BLR93] Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. *Journal of Computer and System Sciences*, 47(3):549–595, 1993. 1

[BOT88] Michael Ben-Or and Prasoon Tiwari. A deterministic algorithm for sparse multivariate polynomial interpolation. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, page 301–309, New York, NY, USA, 1988. Association for Computing Machinery. 4, 8, 20, 39, 40

[BRY14a] Piotr Berman, Sofya Raskhodnikova, and Grigory Yaroslavtsev. L_p-Testing. In *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '14, page 164–173, New York, NY, USA, 2014. Association for Computing Machinery. 2

[BRY14b] Eric Blais, Sofya Raskhodnikova, and Grigory Yaroslavtsev. Lower bounds for testing properties of functions over hypergrid domains. In *2014 IEEE 29th Conference on Computational Complexity (CCC)*, pages 309–320. IEEE, 2014. 2

[Bsh23] Nader H Bshouty. An optimal tester for k-linear. *Theoretical Computer Science*, 950:113759, 2023. 4

[BWY15] Eric Blais, Amit Weinstein, and Yuichi Yoshida. Partially symmetric functions are efficiently isomorphism testable. *SIAM Journal on Computing*, 44(2):411–432, 2015. 5, 8, 27

[BY22] Arnab Bhattacharyya and Yuichi Yoshida. *Property Testing: Problems and Techniques*. Springer Nature, 2022. 2

[CFSS17] Xi Chen, Adam Freilich, Rocco A. Servedio, and Timothy Sun. Sample-based high-dimensional convexity testing. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, pages 37:1–37:20, 2017. 2

[CG04] Hana Chockler and Dan Gutfreund. A lower bound for testing juntas. *Information Processing Letters*, 90(6):301–305, 2004. 5

[CS13] Deeparnab Chakrabarty and Comandur Seshadhri. Optimal bounds for monotonicity and lipschitz testing over hypercubes and hypergrids. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 419–428, 2013. 2

[CST$^+$18] Xi Chen, Rocco A. Servedio, Li-Yang Tan, Erik Waingarten, and Jinyu Xie. Settling the query complexity of non-adaptive junta testing. *J. ACM*, 65(6):40:1–40:18, 2018. 5

[CW01] Anthony Carbery and James Wright. Distributional and L^q Norm Inequalities for Polynomials over Convex Bodies in $\mathbb{R}^n$. *Mathematical Research Letters*, 8:233–248, 2001. 11

[DGL+99] Yevgeniy Dodis, Oded Goldreich, Eric Lehman, Sofya Raskhodnikova, Dana Ron, and Alex Samorodnitsky. Improved testing algorithms for monotonicity. In *International Workshop on Randomization and Approximation Techniques in Computer Science*, pages 97–108. Springer, 1999. 2

[Din07] Irit Dinur. The PCP Theorem by Gap Amplification. *J. ACM*, 54(3):12–es, jun 2007. 2

[DLM+07] Ilias Diakonikolas, Homin K Lee, Kevin Matulef, Krzysztof Onak, Ronitt Rubinfeld, Rocco A Servedio, and Andrew Wan. Testing for concise representations. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 549–558. IEEE, 2007. 5

[DMN19] Anindya De, Elchanan Mossel, and Joe Neeman. Is your function low dimensional? In *Conference on Learning Theory*, pages 979–993. PMLR, 2019. 2, 5

[DS05] Irit Dinur and Samuel Safra. On the hardness of approximating minimum vertex cover. *Annals of mathematics*, pages 439–485, 2005. 27

[EKR01] Funda Ergün, S Ravi Kumar, and Ronitt Rubinfeld. Checking approximate computations of polynomials and functional equations. *SIAM Journal on Computing*, 31(2):550–576, 2001. 3

[Ela61] Regina C. Elandt. The folded normal distribution: Two methods of estimating parameters from moments. *Technometrics*, 3(4):551–562, 1961. 22

[FGL+91] U. Feige, S. Goldwasser, L. Lovasz, S. Safra, and M. Szegedy. Approximating clique is almost np-complete. In *[1991] Proceedings 32nd Annual Symposium of Foundations of Computer Science*, pages 2–12, 1991. 2

[Fis04] Eldar Fischer. The art of uninformed decisions. *Current Trends in Theoretical Computer Science: The Challenge of the New Century*, 2004. 2

[Fis24] Eldar Fischer. A basic lower bound for property testing, 2024. https://arxiv.org/abs/2403.04999. 31

[FKR+04] Eldar Fischer, Guy Kindler, Dana Ron, Shmuel Safra, and Alex Samorodnitsky. Testing juntas. *Journal of Computer and System Sciences*, 68(4):753–787, 2004. 3, 5

[FPJ23] Renato Ferreira Pinto Jr. Directed Poincaré Inequalities and L^1 Monotonicity Testing of Lipschitz Functions. In Nicole Megow and Adam Smith, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023)*, volume 275 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 61:1–61:18, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. 2

[FPJ24] Renato Ferreira Pinto Jr. Directed isoperimetry and monotonicity testing: A dynamical approach. In *2024 IEEE 65th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 2295–2305. IEEE, 2024. 2

[Fri08] Ehud Friedgut. On the measure of intersecting families, uniqueness and stability. *Combinatorica*, 28(5):503–528, 2008. 27

[FS95] Katalin Friedl and Madhu Sudan. Some improvements to total degree tests. In *Proceedings Third Israel Symposium on the Theory of Computing and Systems*, pages 190–198. IEEE, 1995. 2

[FY20] Noah Fleming and Yuichi Yoshida. Distribution-free testing of linear functions on $\mathbb{R}^n$. In *11th Innovations in Theoretical Computer Science Conference (ITCS 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020. 2, 7, 38

[GGR96] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. In *Proceedings of 37th Conference on Foundations of Computer Science*, pages 339–348, 1996. 1, 2

[GJR10] Elena Grigorescu, Kyomin Jung, and Ronitt Rubinfeld. A local decision test for sparse polynomials. *Inf. Process. Lett.*, 110(20):898–901, 2010. 4, 8, 20, 39

[GLR+91] Peter Gemmell, Richard Lipton, Ronitt Rubinfeld, Madhu Sudan, and Avi Wigderson. Self-testing/correcting for polynomials and for approximate functions. In *STOC*, volume 91, pages 32–42. Citeseer, 1991. 3

[GM22] Itay Glazer and Dan Mikulincer. Anti-concentration of polynomials: Dimension-free covariance bounds and decay of fourier coefficients. *Journal of Functional Analysis*, 283(9):109639, 2022. 12

[Gol10] Oded Goldreich. On testing computability by small width obdds. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques: 13th International Workshop, APPROX 2010, and 14th International Workshop, RANDOM 2010, Barcelona, Spain, September 1-3, 2010. Proceedings*, pages 574–587. Springer, 2010. 4

[Gol17] Oded Goldreich. *Introduction to property testing*. Cambridge University Press, 2017. 2

[GOS+11] Parikshit Gopalan, Ryan O'Donnell, Rocco A Servedio, Amir Shpilka, and Karl Wimmer. Testing fourier dimensionality and sparsity. *SIAM Journal on Computing*, 40(4):1075–1100, 2011. 2

[HJ94] Roger A Horn and Charles R Johnson. *Topics in matrix analysis*. Cambridge university press, 1994. 11

[HJ12] Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge University Press, 2012. 8

[HK07] Shirley Halevy and Eyal Kushilevitz. Distribution-free property-testing. *SIAM Journal on Computing*, 37(4):1107–1138, 2007. 7

[HW07] Johan Håstad and Avi Wigderson. The randomized communication complexity of set disjointness. *Theory of Computing*, 3(1):211–219, 2007. 31

[JPRZ09] Charanjit S Jutla, Anindya C Patthak, Atri Rudra, and David Zuckerman. Testing low-degree polynomials over prime fields. *Random Structures & Algorithms*, 35(2):163–193, 2009. 2

[KNOW14] Pravesh Kothari, Amir Nayyeri, Ryan O'Donnell, and Chenggang Wu. Testing surface area. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 1204–1214. SIAM, 2014. 2

[KR06] Tali Kaufman and Dana Ron. Testing polynomials over general fields. *SIAM Journal on Computing*, 36(3):779–802, 2006. 2

[Kuc09] Marek Kuczma. *An Introduction to the Theory of Functional Equations and Inequalities*. Birkhäuser Basel, 2009. 13

[MORS09] Kevin Matulef, Ryan O'Donnell, Ronitt Rubinfeld, and Rocco A Servedio. Testing±1-weight halfspace. In *International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 646–657. Springer, 2009. 2

[MORS10a] Kevin Matulef, Ryan O'Donnell, Ronitt Rubinfeld, and Rocco A Servedio. Testing halfspaces. *SIAM Journal on Computing*, 39(5):2004–2047, 2010. 2

[MORS10b] Kevin Matulef, Ryan O'Donnell, Ronitt Rubinfeld, and Rocco Servedio. Testing (subclasses of) halfspaces. In *Property testing: current research and surveys*, pages 334–340. Springer, 2010. 2

[Nee14] Joe Neeman. Testing surface area with arbitrary accuracy. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 393–397, 2014. 2

[PRR06] Michal Parnas, Dana Ron, and Ronitt Rubinfeld. Tolerant property testing and distance approximation. *Journal of Computer and System Sciences*, 72(6):1012–1042, 2006. 6

[PRS02] Michal Parnas, Dana Ron, and Alex Samorodnitsky. Testing basic boolean formulae. *SIAM Journal on Discrete Mathematics*, 16(1):20–46, 2002. 5

[PZ32] R. E. A. C. Paley and A. Zygmund. A note on analytic functions in the unit circle. *Mathematical Proceedings of the Cambridge Philosophical Society*, 28(3):266–272, 1932. 12

[R+08] Dana Ron et al. Property testing: A learning theory perspective. *Foundations and Trends® in Machine Learning*, 1(3):307–402, 2008. 2

[Ron09] Dana Ron. Algorithmic and analysis techniques in property testing. *Found. Trends Theor. Comput. Sci.*, 2009. 2

[RS92] Ronitt Rubinfeld and Madhu Sudan. Self-Testing Polynomial Functions Efficiently and Over Rational Domains. In *Proceedings of the 3rd Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms (SODA)*, pages 23–32, 1992. 2

[RS96] Ronitt Rubinfeld and Madhu Sudan. Robust Characterizations of Polynomials with Applications to Program Testing. *SIAM Journal on Computing*, 25(2):252–271, 1996. 1, 2

[RS97] Ran Raz and Shmuel Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability pcp characterization of np. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 475–484, 1997. 2

[Sag18] Mert Saglam. Near Log-Convexity of Measured Heat in (Discrete) Time and Consequences. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 967–978. IEEE Computer Society, 2018. 5

[Sam07] Alex Samorodnitsky. Low-degree tests at large distances. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 506–515, 2007. 2

[Wol24] WolframAlpha. Analysis of expression (alternative forms). 2024. 22

A Omitted Algorithms from the Main Body

Here we state the algorithms from [FY20], [ABF+23] and [AKM25] for completeness.

We begin with Algorithm 7, a query-optimal approximate additivity tester, with its subroutines provided in Algorithm 8. The properties of this tester were recorded earlier in Theorem 4.4.

Algorithm 7: [ABF+23, Algorithm 7] and [AKM25, Algorithm 5]: (Query-)Optimal Approximate Additivity Tester

---

Procedure APPROXIMATE ADDITIVITY TESTER$(f, \mathcal{D}, \alpha, \varepsilon, R)$
- **Given:** Query access to $f\colon \mathbb{R}^n \to \mathbb{R}$, sampling access to an unknown $(\varepsilon/4, R)$-concentrated distribution $\mathcal{D}$, a noise parameter $\alpha > 0$, and a farness parameter $\varepsilon > 0$;
- $\delta \leftarrow 3\alpha$, $r \leftarrow 1/50$;
- **return** REJECT if TESTADDITIVITY(f, δ) returns REJECT;
- **for** $N_7 \leftarrow O(1/\varepsilon)$ *times* **do**
  - Sample $\boldsymbol{p} \sim \mathcal{D}$;
  - **if** $\boldsymbol{p} \in \mathrm{B}(\mathbf{0}, R)$ **then**
    - **return** REJECT if $|f(\boldsymbol{p}) -$ APPROXIMATE-$g(\boldsymbol{p}, f, \delta)| > 5\delta n^{1.5}\kappa_{\boldsymbol{p}}$, or if APPROXIMATE-$g(\boldsymbol{p}, f, \delta)$ returns REJECT.
- **return** ACCEPT.

Algorithm 8: [ABF+23, Algorithm 8] and [AKM25, Algorithm 6] Additivity Subroutines

---

Procedure TESTADDITIVITY(f, δ)
- **Given:** Query access to $f\colon \mathbb{R}^n \to \mathbb{R}$, threshold parameter $\delta > 0$;
- **for** $N_8 \leftarrow O(1)$ *times* **do**
  - Sample $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, I_n)$;
  - **return** REJECT if $|f(-\boldsymbol{x}) + f(\boldsymbol{x})| > \delta$;
  - **return** REJECT if $|f(\boldsymbol{x} - \boldsymbol{y}) - (f(\boldsymbol{x}) - f(\boldsymbol{y}))| > \delta$;
  - **return** REJECT if $\left| f\left(\frac{\boldsymbol{x}-\boldsymbol{y}}{\sqrt{2}}\right) - \left(f\left(\frac{\boldsymbol{x}-\boldsymbol{z}}{\sqrt{2}}\right) + f\left(\frac{\boldsymbol{z}-\boldsymbol{y}}{\sqrt{2}}\right)\right)\right| > \delta$;
- **return** ACCEPT.

Procedure APPROXIMATE-$g(\boldsymbol{p}, f, \delta)$
- **Given:** $\boldsymbol{p} \in \mathbb{R}^n$, query access to $f\colon \mathbb{R}^n \to \mathbb{R}$, threshold parameter $\delta > 0$;
- Sample $\boldsymbol{x}_1 \sim \mathcal{N}(\mathbf{0}, I_n)$;
- **return** $\kappa_{\boldsymbol{p}}\left(f(\boldsymbol{p}/\kappa_{\boldsymbol{p}} - \boldsymbol{x}_1) + f(\boldsymbol{x}_1)\right)$.

This is followed by Algorithm 9, a query-optimal (w.r.t. the distance parameter ε) approximate low-degree tester, with its subroutines in Algorithm 10, and its properties recorded in Theorem 5.2.

Algorithm 9: [ABF+23, Algorithm 3] and [AKM25, Algorithm 7] Optimal Approximate Low Degree Tester

```
Procedure ApproxLowDegreeTester(f, d, D, α, ε, R, L)
    Given: Query access to f: ℝ^n → ℝ that is bounded in B(0, L) for some L > 0, a degree
           d ∈ ℕ, sampling access to an unknown (ε/4, R)-concentrated distribution D, a
           noise parameter α > 0, and a farness parameter ε > 0.
    δ ← 2^{d+1} α, r ← (4d)^{-6};
    return Reject if ApproxCharacterizationTest rejects;
    for N_9 ← O(ε^{-1}) times do
        Sample p ~ (2d√n / L) D;
        if p ∈ B(0, 2dR√n/L) then
            return Reject if |f(p) − ApproxQuery-g(p)| > 2 · 2^{(2n)^{45d}} (R/L)^d δ, or if
              ApproxQuery-g(p) rejects.
    return Accept.
```

B Omitted Proofs from Section 5

Observation 5.5 (Generalization of [BOT88, Section 4], and [GJR10, Lemma 4]). *Let $f : \mathbb{R}^n \to \mathbb{R}$ be an exactly k-sparse polynomial over the reals, i.e., $f(\boldsymbol{x}) = \sum_{i=1}^{k} a_i M_i(\boldsymbol{x})$, where $a_1, \ldots, a_k \in \mathbb{R} \setminus \{0\}$, and $M_1, \ldots, M_k$ are the monomials of f. Then for all $\ell + 1 \leq k$,*

$$\det\left(H_{\ell+1}(f, \boldsymbol{x})\right) = \sum_{\substack{S \subseteq [k] \\ |S| = \ell+1}} \prod_{i \in S} a_i \prod_{\substack{i,j \in S \\ i<j}} \left(M_j(\boldsymbol{x}) - M_i(\boldsymbol{x})\right)^2,$$

is a non-zero polynomial of degree $\leq 2\binom{\ell+1}{2} \deg(f)$ in $\boldsymbol{x}$, while for all $\ell+1 > k$, $\det\left(H_{\ell+1}(f, \boldsymbol{x})\right) \equiv 0$.

Proof. Since for all $i \in [k]$ and $\alpha, \beta \in \mathbb{N}_{\geq 0}$, $M_i(\boldsymbol{x}^\alpha) M_i(\boldsymbol{x}^\beta) = M_i(\boldsymbol{x}^{\alpha+\beta})$, we get: for all $\boldsymbol{u} \in \mathbb{R}^n$,

$$H_{\ell+1}(f, \boldsymbol{u}) = \begin{pmatrix} \sum_{i=1}^k a_i M_i(\boldsymbol{u}^0) & \sum_{i=1}^k a_i M_i(\boldsymbol{u}^1) & \ldots & \sum_{i=1}^k a_i M_i(\boldsymbol{u}^\ell) \\ \sum_{i=1}^k a_i M_i(\boldsymbol{u}^1) & \sum_{i=1}^k a_i M_i(\boldsymbol{u}^2) & \ldots & \sum_{i=1}^k a_i M_i(\boldsymbol{u}^{\ell+1}) \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^k a_i M_i(\boldsymbol{u}^\ell) & \sum_{i=1}^k a_i M_i(\boldsymbol{u}^{\ell+1}) & \ldots & \sum_{i=1}^k a_i M_i(\boldsymbol{u}^{2\ell}) \end{pmatrix}$$

$$= \begin{pmatrix} M_1(\boldsymbol{u}^0) & M_2(\boldsymbol{u}^0) & \ldots & M_k(\boldsymbol{u}^0) \\ M_1(\boldsymbol{u}^1) & M_2(\boldsymbol{u}^1) & \ldots & M_k(\boldsymbol{u}^1) \\ M_1(\boldsymbol{u}^2) & M_2(\boldsymbol{u}^2) & \ldots & M_k(\boldsymbol{u}^2) \\ \vdots & \vdots & \ddots & \vdots \\ M_1(\boldsymbol{u}^\ell) & M_2(\boldsymbol{u}^\ell) & \ldots & M_k(\boldsymbol{u}^\ell) \end{pmatrix} \begin{pmatrix} a_1 M_1(\boldsymbol{u}^0) & a_1 M_1(\boldsymbol{u}^1) & \ldots & a_1 M_1(\boldsymbol{u}^\ell) \\ a_2 M_2(\boldsymbol{u}^0) & a_2 M_2(\boldsymbol{u}^1) & \ldots & a_2 M_2(\boldsymbol{u}^\ell) \\ \vdots & \vdots & \ddots & \vdots \\ a_k M_k(\boldsymbol{u}^0) & a_k M_k(\boldsymbol{u}^1) & \ldots & a_k M_k(\boldsymbol{u}^\ell) \end{pmatrix}$$

$$= \underbrace{\begin{pmatrix} 1 & 1 & \ldots & 1 \\ M_1(\boldsymbol{u}) & M_2(\boldsymbol{u}) & \ldots & M_k(\boldsymbol{u}) \\ (M_1(\boldsymbol{u}))^2 & (M_2(\boldsymbol{u}))^2 & \ldots & (M_k(\boldsymbol{u}))^2 \\ \vdots & \vdots & \ddots & \vdots \\ (M_1(\boldsymbol{u}))^\ell & (M_2(\boldsymbol{u}))^\ell & \ldots & (M_k(\boldsymbol{u}))^\ell \end{pmatrix}}_{h_1 : \mathbb{R}^k \to \mathbb{R}^{\ell+1}} \underbrace{\begin{pmatrix} a_1 & 0 & \ldots & 0 \\ 0 & a_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & a_k \end{pmatrix}}_{h_2 : \mathbb{R}^k \to \mathbb{R}^k} \underbrace{\begin{pmatrix} 1 & M_1(\boldsymbol{u}) & \ldots & (M_1(\boldsymbol{u}))^\ell \\ 1 & M_2(\boldsymbol{u}) & \ldots & (M_2(\boldsymbol{u}))^\ell \\ \vdots & \vdots & \ddots & \vdots \\ 1 & M_k(\boldsymbol{u}) & \ldots & (M_k(\boldsymbol{u}))^\ell \end{pmatrix}}_{h_3 : \mathbb{R}^{\ell+1} \to \mathbb{R}^k}.$$

Algorithm 10: [ABF+23, Algorithm 4] and [AKM25, Algorithm 8]: Approximate Subroutines

[Recall $\alpha_i \triangleq (-1)^{i+1}\binom{d+1}{i}$ and $\delta = 2^{d+1}\alpha$.]

Procedure ApproxCharacterizationTest
- $N_{10} \leftarrow O(d^2)$;
- **for** N_{10} *times* **do**
  - **for** $j \in \{1, \ldots, d+1\}$ **do**
    - **for** $t \in \{0, \ldots, d+1\}$ **do**
      - Sample $\boldsymbol{p} \sim \mathcal{N}(\mathbf{0}, j^2(t^2+1)I), \boldsymbol{q} \sim \mathcal{N}(\mathbf{0}, I_n)$;
      - **return** Reject if $|\sum_{i=0}^{d+1} \alpha_i \cdot f(\boldsymbol{p} + i\boldsymbol{q})| > \delta$;
      - Sample $\boldsymbol{p} \sim \mathcal{N}(\mathbf{0}, j^2 I), \boldsymbol{q} \sim \mathcal{N}(\mathbf{0}, (t^2+1)I)$;
      - **return** Reject if $|\sum_{i=0}^{d+1} \alpha_i \cdot f(\boldsymbol{p} + i\boldsymbol{q})| > \delta$;
    - Sample $\boldsymbol{p}, \boldsymbol{q} \sim \mathcal{N}(\mathbf{0}, j^2 I)$;
    - **return** Reject if $|\sum_{i=0}^{d+1} \alpha_i \cdot f(\boldsymbol{p} + i\boldsymbol{q})| > \delta$;
- **return** Accept;

Procedure ApproxQuery-g($\boldsymbol{p}$)
- **if** $\boldsymbol{p} \in \mathrm{B}(\mathbf{0}, r)$ **then**
  - **return** ApproxQuery-g-InBall($\boldsymbol{p}$);
- **for** $i \in \{0, 1, \ldots, d\}$ **do**
  - $c_i \leftarrow \frac{r}{\|\boldsymbol{p}\|_2} \cos\left(\frac{\pi(i+1/2)}{d+1}\right)$;
  - $v(c_i) \leftarrow$ ApproxQuery-g-InBall($c_i\boldsymbol{p}$) ;
- Let $p_{\boldsymbol{p}} \colon \mathbb{R} \to \mathbb{R}$ be the unique degree-d polynomial such that $p_{\boldsymbol{p}}(c_i) = v(c_i)$ for all i;
- **return** $p_{\boldsymbol{p}}(1)$;

Procedure ApproxQuery-g-InBall($\boldsymbol{p}$)
- Sample $\boldsymbol{q}_1 \sim \mathcal{N}(\mathbf{0}, I_n)$;
- **return** $\sum_{i=1}^{d+1} \alpha_i \cdot f(\boldsymbol{p} + i\boldsymbol{q}_1)$;

Thus, if $\ell + 1 > k$, the mapping h_3 in the above expression is necessarily non-injective, and hence $H_{\ell+1}(f, \boldsymbol{u})$ is necessarily singular, for any $\boldsymbol{u}$. This implies that $\det(H_{\ell+1}(f, \boldsymbol{x})) \equiv 0$. If $k \geq \ell + 1$, the determinant expansion in the observation follows from the following argument in [BOT88]:

1. For any fixed $\boldsymbol{u}$, by standard determinant expansion, $\det(H_{\ell+1}(f, \boldsymbol{u}))$ is a polynomial $Q(\boldsymbol{a})$, and the total degree of each monomial (in the "variables" $a_1, \ldots, a_k$) in Q is exactly $\ell + 1$.

2. If $\operatorname{rank}(H_{\ell+1}(f, \boldsymbol{u})) \leq \|[a_1 \cdots a_k]\|_0 < \ell + 1$, we must have $\det(H_{\ell+1}(f, \boldsymbol{u})) \equiv 0$, irrespective of the values of the non-zero $\{a_1, \ldots, a_k\}$, for any $\boldsymbol{u} \in \mathbb{R}^n$. Hence each monomial (in $a_1, \ldots, a_k$) of $Q(a_1, \ldots, a_k)$ must have at least $\ell + 1$ of the variables $\{a_i\}$. Since its total degree is exactly $\ell + 1$, each monomial must be of the form $\prod_{i \in S} a_i$ for some $S \subseteq [k]$, with $|S| = \ell + 1$.

3. The coefficient of monomial $\prod_{i \in S} a_i$ in Q will be $Q(c_1, \ldots, c_k)$ with $c_i = \mathbb{1}\{i \in S\}$. But, by the decomposition above, this will be the square of the Vandermonde determinant (see Definition 3.14), $\det(V_{\ell+1}(\{x_i\}_{i \in S})) = \prod_{\substack{i,j \in S \\ i<j}} (M_j(\boldsymbol{u}) - M_i(\boldsymbol{u}))$.

Summing these terms over all $S \subseteq [k]$ with $|S| = \ell + 1$, completes the determinant expansion. $\square$

Observation 5.7. *Let $f : \mathbb{R}^n \to \mathbb{R}$ and let $\tilde{f}$ be an η-approximate query oracle to f. Then for any $t \geq 1$ and $\boldsymbol{u} \in \mathbb{R}^n$, we can express*

$$H_t(\tilde{f}, \boldsymbol{u}) = H_t(f, \boldsymbol{u}) + E_t(\boldsymbol{u}),$$

where $E_t(\boldsymbol{u})$ is a Hankel-structured, noise matrix*, such that $\|E_t(\boldsymbol{u})\|_\infty \leq \eta$, and $\|E_t(\boldsymbol{u})\|_{\mathrm{op}} \leq \eta t$.*

Proof of Observation 5.7. Let $\boldsymbol{u} \in \mathbb{R}^n$, and $\alpha_i \triangleq \tilde{f}(\boldsymbol{u}^i) - f(\boldsymbol{u}^i)$ for each $i \in \{0, 1, \ldots, 2t-2\}$, so that $|\alpha_i| \leq \eta$ by the approximation-oracle guarantee. We can rewrite $H_t(\tilde{f}, \boldsymbol{u})$ (by Definition 5.4) as:

$$\begin{aligned}
H_t(\tilde{f}, \boldsymbol{u}) &= \begin{pmatrix} \tilde{f}(\boldsymbol{u}^0) & \tilde{f}(\boldsymbol{u}^1) & \ldots & \tilde{f}(\boldsymbol{u}^{t-1}) \\ \tilde{f}(\boldsymbol{u}^1) & \tilde{f}(\boldsymbol{u}^2) & \ldots & \tilde{f}(\boldsymbol{u}^t) \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{f}(\boldsymbol{u}^{t-1}) & \tilde{f}(\boldsymbol{u}^t) & \ldots & \tilde{f}(\boldsymbol{u}^{2t-2}) \end{pmatrix} = \begin{pmatrix} f(\boldsymbol{u}^0) + \alpha_0 & f(\boldsymbol{u}^1) + \alpha_1 & \ldots & f(\boldsymbol{u}^{t-1}) + \alpha_{t-1} \\ f(\boldsymbol{u}^1) + \alpha_1 & f(\boldsymbol{u}^2) + \alpha_2 & \ldots & f(\boldsymbol{u}^t) + \alpha_t \\ \vdots & \vdots & \ddots & \vdots \\ f(\boldsymbol{u}^{t-1}) + \alpha_{t-1} & f(\boldsymbol{u}^t) + \alpha_t & \ldots & f(\boldsymbol{u}^{2t-2}) + \alpha_{2t-2} \end{pmatrix} \\
&= \underbrace{\begin{pmatrix} f(\boldsymbol{u}^0) & f(\boldsymbol{u}^1) & \ldots & f(\boldsymbol{u}^{t-1}) \\ f(\boldsymbol{u}^1) & f(\boldsymbol{u}^2) & \ldots & f(\boldsymbol{u}^t) \\ \vdots & \vdots & \ddots & \vdots \\ f(\boldsymbol{u}^{t-1}) & f(\boldsymbol{u}^t) & \ldots & f(\boldsymbol{u}^{2t-2}) \end{pmatrix}}_{=H_t(f, \boldsymbol{u})} + \underbrace{\begin{pmatrix} \alpha_0 & \alpha_1 & \ldots & \alpha_{t-1} \\ \alpha_1 & \alpha_2 & \ldots & \alpha_t \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{t-1} & \alpha_t & \ldots & \alpha_{2t-2} \end{pmatrix}}_{\triangleq E_t(\boldsymbol{u})}.
\end{aligned}$$

The Hankel structure of $E_t(\boldsymbol{u})$ is evident. $\|E_t(\boldsymbol{u})\|_\infty \leq \eta$ follows from the approximation-oracle guarantee. This in turn shows $\|E_t(\boldsymbol{u})\|_{\mathrm{F}} \leq \sqrt{\eta^2 t^2} = \eta t$, which implies the bound on $\| \mathbb{E}_t(\boldsymbol{u})\|_{\mathrm{op}}$. $\square$