Electronic Colloquium on Computational Complexity, Report No. 4 (2024)

Testing equivalence to design polynomials

Omkar Baraskar
Indian Institute of Science
omkarb@iisc.ac.in

Agrim Dewan
Indian Institute of Science
agrimdewan@iisc.ac.in

Chandan Saha*
Indian Institute of Science
chandan@iisc.ac.in

Abstract

An n-variate polynomial g of degree d is a (n, d, t) *design polynomial* if the degree of the gcd of every pair of monomials of g is at most $t - 1$. The power symmetric polynomial $\mathsf{PSym}_{n,d} := \sum_{i=1}^{n} x_i^d$ and the sum-product polynomial $\mathsf{SP}_{s,d} := \sum_{i=1}^{s} \prod_{j=1}^{d} x_{i,j}$ are instances of design polynomials for $t = 1$. Another example is the Nisan-Wigderson design polynomial NW, which has been used extensively to prove various arithmetic circuit lower bounds. Given black-box access to an n-variate, degree-d polynomial $f(\mathbf{x}) \in \mathbb{F}[\mathbf{x}]$, how fast can we check if there exist an $A \in \mathrm{GL}(n, \mathbb{F})$ and a $\mathbf{b} \in \mathbb{F}^n$ such that $f(A\mathbf{x} + \mathbf{b})$ is a (n, d, t) design polynomial? We call this problem "testing equivalence to design polynomials", or alternatively, "equivalence testing for design polynomials".

In this work, we present a randomized algorithm that finds $(A, \mathbf{b})$ such that $f(A\mathbf{x} + \mathbf{b})$ is a (n, d, t) design polynomial, if such A and $\mathbf{b}$ exist, provided $t \leq d/3$. The algorithm runs in $(nd)^{O(t)}$ time and works over any sufficiently large $\mathbb{F}$ of characteristic 0 or $> d$. As applications of this test, we show two results – one is structural and the other is algorithmic. The structural result establishes a polynomial-time equivalence between the graph isomorphism problem and the polynomial equivalence problem for design polynomials. The algorithmic result implies that Patarin's scheme (EUROCRYPT 1996) can be broken in quasi-polynomial time if a random sparse polynomial is used in the key generation phase.

We also give an efficient learning algorithm for n-variate random affine projections of multilinear degree-d design polynomials, provided $n \geq d^4$. If one obtains an analogous result under the weaker assumption "$n \geq d^\epsilon$, for any $\epsilon > 0$", then the NW family is *not* VNP-complete unless there is a VNP-complete family whose random affine projections are learnable. It is not known if random affine projections of the permanent are learnable.

The above algorithms are obtained by using the vector space decomposition framework, introduced by Kayal and Saha (STOC 2019) and Garg, Kayal and Saha (FOCS 2020), for learning non-degenerate arithmetic circuits. A key technical difference between the analysis in the papers by Garg, Kayal and Saha (FOCS 2020) and Bhargava, Garg, Kayal and Saha (RANDOM 2022) and the analysis here is that a certain adjoint algebra, which turned out to be trivial (i.e., diagonalizable) in prior works, is non-trivial in our case. However, we show that the adjoint arising here is triangularizable which then helps in carrying out the vector space decomposition step.

*Partially supported by a MATRICS grant of the Science and Engineering Research Board, DST, India.

ISSN 1433-8092

Contents

1 Introduction

The polynomial equivalence problem (PE) is a fundamental problem in algebraic complexity theory. Given two polynomials, f and g, how fast can we check if one is in the orbit of the other? *Orbit* of a polynomial $f \in \mathbb{F}[\mathbf{x}]$ is the set $\{f(A\mathbf{x}) : A \in \mathrm{GL}(|\mathbf{x}|, \mathbb{F})\}$. In other words, PE is the problem of checking if f and g are the same function up to a change of the coordinate system. It can be regarded as the algebraic analog of the graph isomorphism (GI) problem.

Much is unknown about the exact complexity of PE. Over finite fields, PE is unlikely to be NP-complete [Thi98, Sax06], but no polynomial-time algorithm is known unless f and g are quadratic forms [Lam04, Ara11]. PE for cubic forms over $\mathbb{Q}$ is not known to be decidable. Cubic form equivalence (CFE) is polynomial-time equivalent to several other fundamental problems in algebra and linear algebra [GQ23a, GQ23b, AS06]. GI reduces to CFE in polynomial time [AS05], but the converse is not known to be true. A natural question emerges at this point:

Is GI polynomial-time equivalent to PE for some natural class of polynomials?

We provide an affirmative answer to this question in Theorem 2 by studying the problem of testing equivalence to design polynomials which we also refer to as the equivalence testing problem for the family of design polynomials (see Definitions 1.1 and 1.2).

Equivalence testing (ET) is closely related to the PE problem. ET for a polynomial family or a circuit class $\mathscr{F}$ is a problem wherein we are given a polynomial f, and we wish to check if f is in the orbit of some polynomial or circuit in $\mathscr{F}$.[1] Efficient ET algorithms are known for a variety of polynomial families and a few circuit classes, namely the permanent [Kay12], the determinant [Kay12, GGKS19], the iterated matrix multiplication polynomial family [KNST19, MNS20], the elementary and power symmetric polynomials [Kay11], the sum-product polynomial family [MS21], the continuant [MS21], and read-once formulas [GST23]. One important family that is missing from the above list is the Nisan-Wigderson design polynomial family NW (see Equation 1). The NW family has been used in many results on arithmetic circuit lower bounds in the last decade. But, unlike the other families, NW had no known ET. In fact, ET for NW over $\mathbb{Q}$ was not known to be decidable. We ask a more general question:

Is there an ET algorithm for the family of (general) design polynomials?

Our main result, given in Theorem 1, is an ET algorithm for design polynomials over *any* field of zero or sufficiently large characteristic. The algorithm reveals a structural property of invertible transformations between design polynomials that enables us to prove Theorem 2. The running time of the algorithm also helps us point out a vulnerability of Patarin's authentication scheme if a random sparse polynomial is chosen in the key generation phase.

Patarin [Pat96] proposed a zero-knowledge authentication scheme based on the presumed hardness of PE for random cubic forms (more generally, constant-degree forms). A random n-variate cubic form $f(\mathbf{x})$ is chosen in the key generation phase along with two random transformations $A_1, A_2 \in \mathrm{GL}(n, \mathbb{F})$. The polynomials $g_1 := f(A_1\mathbf{x})$ and $g_2 := f(A_2\mathbf{x})$ are then made public; the secret is the transformation $A_1^{-1}A_2$, which maps g_1 to g_2. A random cubic form has sparsity (i.e., number of monomials) $O(n^3)$. It is natural to ask: What if we choose a *random* $O(n^3)$-sparse polynomial f of a *higher degree* (see Definition 1.4) in the key generation step?

Can Patarin's scheme be broken if a random $n^{O(1)}$-sparse polynomial is chosen as the key?

[1] Note that the ET problem for $\mathscr{F}$ is not the same as the PE problem for $\mathscr{F}$. In the latter case, we are given two polynomials $f, g \in \mathscr{F}$, and we wish to check if one is in the orbit of the other. One may alternatively call the ET problem for $\mathscr{F}$ as "testing equivalence to $\mathscr{F}$" (as in the title of this article).

It turns out that a random sparse polynomial is a design polynomial and has no nontrivial permutation symmetry with high probability (see Lemma 3.7 and Proposition 3.7). These features let us invoke Theorem 1 and answer the above question in Theorem 3 via a reduction to GI.

Our final result is a learning algorithm for random affine projections of design polynomials. An equivalence test for NW is a special case of learning affine projections of NW. Consider the (qd, d, t) design polynomial $\mathsf{NW}_{q,d,t}$ as defined in Equation 1. A polynomial $f = \mathsf{NW}_{q,d,t}(A\mathbf{x} + \mathbf{b})$, where $|\mathbf{x}| = n \leq qd$, $A \in \mathbb{F}^{qd \times n}$ and $\mathbf{b} \in \mathbb{F}^{qd}$, is an n-variate affine projection of $\mathsf{NW}_{q,d,t}$. Given access to f, can we learn the unknown A and $\mathbf{b}$? If $n = qd$ and $A \in \mathrm{GL}(n, \mathbb{F})$, then the problem is the same as ET for NW. However, for arbitrary $n < qd$ and $A \in \mathbb{F}^{qd \times n}$, the problem can be rather difficult, even for $t = 1$, as every depth-3 circuit is an affine projection of $\mathsf{NW}_{q,d,1}$ for some q and d. Learning depth-3 circuits in the worst-case is a challenging problem due to known depth reduction results (see the discussion in [KS19]). But does the task of discovering A become easier if A is randomly chosen? In other words:

Can we learn random affine projections of NW *efficiently?*

Random affine projections of special design polynomials, such as the power symmetric polynomial and the sum-product polynomial, have been studied, and efficient learning algorithms have been provided in [KS19]. But it is unclear what to expect for NW. The reason is that, unlike the determinant and the permanent, we do not have a good understanding of the expressive power of affine projections of NW. The permanent is VNP-complete under p-projections[2] [Val79], but NW is not known to be so. For $d = n^{O(1)}$, no learning algorithm is known for n-variate random affine projections of the $d \times d$ permanent which has time complexity polynomial in $\binom{n+d}{n}$ – the maximum sparsity of any n-variate, degree-d polynomial. In fact, it is conjectured in [Aar08] that n-variate random affine projections of the $d \times d$ *determinant* form a pseudorandom function family when $d = n^{O(1)}$. If true, then there is no $\binom{n+d}{n}^{O(1)}$-time learning algorithm for random affine projections of the determinant. If such a conclusion holds for the determinant, which is VBP-complete, then we expect the same to hold for the permanent or any other VNP-complete family, as VBP $\subseteq$ VNP. So, an efficient learning algorithm for random affine projections of NW may indicate that NW is *not* VNP-complete.

In Theorem 4, we give an efficient learning algorithm for n-variate random affine projections of multilinear degree-d design polynomials, provided $n \geq d^4$. If we obtain an analogous result under the weaker assumption "$n \geq d^\epsilon$, for any $\epsilon > 0$", then the NW family is not VNP-complete assuming that there is no VNP-complete family whose random affine projections are efficiently learnable. On the other hand, if NW happens to be VNP-complete, then it is unlikely that we will be able to weaken the $n \geq d^4$ condition significantly without compromising the other parameters of the theorem considerably.

1.1 Our results

We now state our results formally. Assume that an efficient univariate polynomial factoring algorithm over $\mathbb{F}$ is available; this assumption is well justified over $\mathbb{Q}$ and $\mathbb{F}_q$ [LLL82, Ber70].

Definition 1.1 (Design polynomial)**.** An n-variate, degree-d polynomial $g = \sum_{i \in [s]} c_i m_i$, where m_i is a monomial and $c_i \in \mathbb{F}$, is a (n, d, s, t) design polynomial if $\forall i \neq j$, $\deg \gcd(m_i, m_j) < t$.

Some well-known polynomials that are also design polynomials are the sum-product polynomial $\mathsf{SP}_{s,d} := \sum_{i=1}^{s} \prod_{j=1}^{d} x_{i,j}$, which is a $(sd, d, s, 1)$ design polynomial, and the power

[2] If every row of A has at most one nonzero entry, then it is a p-projection.

symmetric polynomial $\mathsf{PSym}_{n,d} := \sum_{i=1}^{n} x_i^d$, which is a $(n, d, n, 1)$ design polynomial. The most relevant example is the Nisan-Wigderson design polynomial $\mathsf{NW}_{q,d,t}$, which is defined as:

$$\mathsf{NW}_{q,d,t} := \sum_{h \in \mathbb{F}_q[y],\ \deg h < t} \prod_{i=0}^{d-1} x_{i,h(i)}\,, \qquad \text{for } t \le d \le q, \text{ where } q \text{ is a prime.} \tag{1}$$

As two univariate polynomials of degree $< t$ agree at $\le t-1$ points, $\mathsf{NW}_{q,d,t}$ is a (qd, d, q^t, t) design polynomial. Whenever the parameter s is not required, we will write (n, d, t) design polynomial by omitting s. Also, for simplicity, we assume that design polynomials are *homogeneous*. Our results hold for non-homogeneous design polynomials as well.

Definition 1.2 (The ET problem for design polynomials)**.** Given black-box access to an n-variate, degree-d polynomial $f \in \mathbb{F}[\mathbf{x}]$, check if there exist an $A \in \mathrm{GL}(n, \mathbb{F})$ and a $\mathbf{b} \in \mathbb{F}^n$ such that $f(A\mathbf{x} + \mathbf{b})$ is a (n, d, t) design polynomial, and if so, recover A and $\mathbf{b}$.

Theorem 1 (ET for design polynomials)**.** *Let $n, d, s, t \in \mathbb{N}$, $d \ge 3t$, $\mathrm{char}(\mathbb{F}) = 0$ or $> d$ and $|\mathbb{F}| > \max(s^3, d^7)$. There is a randomized, $\mathrm{poly}((nd)^t)$-time[3] algorithm that takes input black-box access to an n-variate, degree-d polynomial $f \in \mathbb{F}[\mathbf{x}]$, with the promise that there exist some (n, d, s, t) design polynomial g and some $A \in \mathrm{GL}(n, \mathbb{F})$ such that $f = g(A\mathbf{x})$, and outputs, with high probability, a $B \in \mathrm{GL}(n, \mathbb{F})$ and a (n, d, s, t) design polynomial h such that $B = PSA$ and $f = h(B\mathbf{x})$, where P, S are permutation and scaling matrices, respectively.*

Remarks.

1. If g is multilinear, then the condition $d \ge 3t$ can be improved to $d > 2t$.
2. The condition $|\mathbb{F}| > \max(s^3, d^7)$ arises due to the use of the Schwartz–Zippel lemma[4] in our analysis and in the factorization algorithm of [KT90]. If f is given as a circuit, the algorithm can work with an extension field, irrespective of the size of $\mathbb{F}$, and still obtain a B with entries in $\mathbb{F}$. This feature of the algorithm is explained in Remark 3.1. A finite field extension can be constructed efficiently (see Section 14.9 in [vzGG03]).
3. The theorem gives an ET algorithm for NW (see Theorem 5). The algorithm also works over $\mathbb{Q}$, where ET for NW was not known to be decidable.
4. A random sparse polynomial is a $(n, d, s, d/3)$ design polynomial with high probability (see Lemma 3.7). Thus, we have ET for random sparse polynomials.

In Section 3, we prove Theorem 1 and elaborate on how non-homogeneous design polynomials and transforms of the form $A\mathbf{x} + \mathbf{b}$, where $A \in \mathrm{GL}(n, \mathbb{F})$ and $\mathbf{b} \in \mathbb{F}^n$, are handled.

Definition 1.3 (Symmetries of a polynomial)**.** Let $f \in \mathbb{F}[\mathbf{x}]$ be an n-variate, degree-d polynomial. The set $\mathscr{G}_f := \{A \in \mathrm{GL}(n, \mathbb{F}) : f(A\mathbf{x}) = f\}$ is the group of symmetries of f.

The following corollaries are proven in Section A. The authors of [GS19] studied the symmetries of NW by examining the Lie algebra associated with it, while these corollaries of Theorem 1 (which is proved using a different technique) hold for general design polynomials.

Corollary 1.1 (Symmetries of design polynomials)**.** Let $d \ge 3t$, f be a (n, d, t) design polynomial and $A \in \mathscr{G}_f$. Then, $A = PS$, where P is a permutation matrix and S is a scaling matrix.

Corollary 1.2 (Equivalent design polynomials)**.** Let $d \ge 3t$, $A \in \mathrm{GL}(n, \mathbb{F})$ and f, g be (n, d, t) design polynomials such that $f = g(A\mathbf{x})$. Then, $A = PS$, where P, S are as stated above.

[3] Here, "time" means number of field operations. Over $\mathbb{Q}$, the complexity is $\mathrm{poly}((nd)^t, \beta)$, where β is the bit complexity of the coefficients of f. Also, the time complexity of the algorithm is optimal (see Remark 3.2).

[4] also known as the DeMillo–Lipton–Schwartz–Zippel lemma [DL78, Zip79, Sch80].

Our second result, proven in Section 3.5, shows that GI $\equiv_p$ PE for design polynomials with all-one coefficients. Here, $\equiv_p$ denotes polynomial-time, many-one equivalence.

Theorem 2 (GI and PE). *GI $\equiv_p$ PE for $(n, 6, 2)$ design polynomials with all-one coefficients.*

Theorem 2 holds for (n, d, t) design polynomials for any $d \geq 6$ and $t \leq d/3$, and also for (n, d, t) multilinear design polynomials with $d \geq 5$ and $t < d/2$. Thus, PE for (n, d, t) design polynomials with all-one coefficients, for $d \geq 6$ and $t \leq d/3$, polynomial-time reduces to PE for $(n, 6, 2)$ design polynomials with all-one coefficients.[5]

Our third result, proven in Section 3.5, shows that if a random sparse polynomial of sufficiently large constant degree is used in Patarin's scheme for public and private key generations, then the private key can be recovered in quasi-polynomial time.

Definition 1.4 (Random sparse polynomial). An n-variate, degree-d, s-sparse polynomial $f(\mathbf{x})$ is a random s-sparse polynomial if each of the s monomials is formed by picking d variables uniformly and independently at random from $\mathbf{x}$; the coefficients are then chosen arbitrarily.

Theorem 3 (A vulnerability of Patarin's scheme). *Let $n, s, d, q \in \mathbb{N}$, $n > d^8$, $n^3 \leq s < (\frac{n}{d^2})^{d/6}$, $d \geq 25$ be a constant, and $q = n^{O(1)}$. Let f be a random s-sparse polynomial over $\mathbb{F}_q$. If f is used in Patarin's scheme for key generation, then the scheme can be broken in* quasi-poly(n) *time.*

Remarks.
1. The bound on s, stated in the theorem, is for simplicity. The precise bound is $n^2 \log(n) \leq s \leq \sqrt{\epsilon}(\frac{n}{d^2})^{d/6}$, where ϵ is the constant from Lemma 3.7. The lower bound on d can be derived from the inequality $n^2 \log(n) \leq \sqrt{\epsilon}(\frac{n}{d^2})^{d/6}$ and fixing $\epsilon = 0.01$.
2. Patarin's scheme was shown to be vulnerable in [Kay11] when using a random constant-degree *multilinear* polynomial for key generation. A random polynomial there was defined by selecting the coefficients of *all* multilinear monomials randomly and independently, while we allow for arbitrary coefficients, non-multilinear monomials, and a lower number of monomials.

Our fourth and final result, proven in Section 4, gives an algorithm to learn random affine projections of multilinear design polynomials such as the polynomials in the NW family.

Definition 1.5 (Affine projections). Let $m, n \in \mathbb{N}$, $m \geq n$. Let f and g be polynomials in n and m variables, respectively. If $f(\mathbf{x}) = g(A\mathbf{x} + \mathbf{b})$ for some $A \in \mathbb{F}^{m \times n}$ and $\mathbf{b} \in \mathbb{F}^m$, then f is an affine projection of g. An affine projection is random if $A \in_r \mathbb{F}^{m \times n}$, where $\in_r$ denotes that the elements of A are chosen randomly and independently from a sufficiently large subset of $\mathbb{F}$.

Definition 1.6 (Learning affine projections of design polynomials). Given black-box access to an n-variate $f \in \mathbb{F}[\mathbf{x}]$, which is an affine projection of an unknown (m, d, s, t) design polynomial g, recover $B \in \mathbb{F}^{m \times n}$, $\mathbf{c} \in \mathbb{F}^m$ and a (m, d, s, t) design polynomial h such that $f = h(B\mathbf{x} + \mathbf{c})$.

Theorem 4 (Learning random affine projections of multilinear design polynomials). *Let $m, n, d, s, t \in \mathbb{N}$, $m \geq n \geq d^{4+\epsilon}$, where $\epsilon > 0$, $d \geq 3t$, $s < (\frac{\sqrt{n}}{d^2})^{\frac{d}{13}}$. Let* char$(\mathbb{F}) = 0$ *or* $> d$ *and* $|\mathbb{F}| \geq$ poly$(sd)d^t$. *There is a randomized,* poly(m, s, n^t)*-time algorithm that takes input black-box access to an n-variate, degree-d polynomial $f \in \mathbb{F}[\mathbf{x}]$, with the promise that there exist some multilinear (m, d, s, t) design polynomial g and some $A \in_r \mathbb{F}^{m \times n}$ such that $f = g(A\mathbf{x})$, and outputs, with high probability, a $B \in \mathbb{F}^{m \times n}$ and a multilinear (m, d, s, t) polynomial h such that $B = PSA$ and $f = h(B\mathbf{x})$, where P, S are permutation and scaling matrices, respectively.*

[5] It is worth noting, in [AS05], the authors showed a reduction from PE for degree-d forms to PE for cubic forms over fields containing d-th roots. However, the reduction there does not seem to preserve the design condition.

Remarks. 1. For $\mathsf{NW}_{q,d,t}$ with $t \leq d/1300$, $m = qd = n^{10}$ and $n \geq d^5$, it holds that $s = q^t < (\frac{\sqrt{n}}{d^2})^{\frac{d}{13}}$. Thus, we can learn random affine projections of $\mathsf{NW}_{q,d,t}$ for $m = \text{poly}(n)$, *assuming* $n \geq d^{4+\epsilon}$. For the precise bound on $|\mathbb{F}|$ in the statement, see Remark 4.1.

2. Theorem 4 *does not* imply that either NW is not VNP-complete or there is a VNP-complete family whose random affine projections are learnable. The reason is that the algorithm assumes $n \geq d^{4+\epsilon}$, but it may be the case that a VNP polynomial family is a projection of NW in the setting $n < d^4$.
3. As mentioned before, our motivation for designing a learning algorithm for random affine projections of multilinear design polynomials originates from NW, which is also multilinear and design. We believe that a similar theorem holds for non-multilinear design polynomials as well. Theorem 1 provides evidence towards this belief since it holds for non-multilinear design polynomials as well.

In Section 4, we elaborate on how non-homogeneous multilinear design polynomials and general transforms of the form $A\mathbf{x} + \mathbf{b}$, where $A \in_r \mathbb{F}^{m \times n}$ and $\mathbf{b} \in \mathbb{F}^m$, are handled.

1.2 Proof techniques

The core underlying technique, used to prove Theorems 1 and 4, is based on the vector space decomposition framework introduced in [GKS20, KS19]. Suppose that f can be expressed as:

$$f = T_1 + T_2 + \ldots + T_s \,, \tag{2}$$

and we wish to learn the *terms* $T_1, \ldots, T_s$ that are *simple* in some sense. For example, in our setting, each T_i is a product of linear forms (see details on next page). The authors in [GKS20, KS19] reduce the task of learning the T_i's to the vector space decomposition (VSD) problem. We define the VSD problem first and then discuss the reduction.

Vector space decomposition (VSD) for $(\mathscr{L}, U, V)$: Given bases of vector spaces U, V and a set of linear maps $\mathscr{L}$ from U to V, output a (further indecomposable) decomposition of U, V as:

$$U = U_1 \oplus \cdots \oplus U_s \;\text{ and }\; V = V_1 \oplus \cdots \oplus V_s \,, \text{ such that } \langle \mathscr{L} \circ U_i \rangle \subseteq V_i \text{ for all } i \in [s].$$

Reducing the learning problem to VSD: We choose appropriate sets of operators $\mathscr{L}_1$ and $\mathscr{L}_2$ to obtain spaces $U = \langle \mathscr{L}_1 \circ f \rangle$ and $V = \langle \mathscr{L}_2 \circ U \rangle$ such that the following conditions are satisfied:

- $U = U_1 \oplus \cdots \oplus U_s$ and $V = V_1 \oplus \cdots \oplus V_s$, where $U_i = \langle \mathscr{L}_1 \circ T_i \rangle$ and $V_i = \langle \mathscr{L}_2 \circ U_i \rangle$.
- Each pair (U_i, V_i) is *indecomposable* with respect to $\mathscr{L}_2$.
- The (further indecomposable) decomposition is *unique* up to a reordering of U_i's and V_i's.
- T_i's can be recovered efficiently from the bases of the U_i's.

If such two sets of operators can be found, then learning $T_1, \ldots, T_s$ reduces to the VSD problem for $(\mathscr{L}_2, U, V)$. We need the (U_i, V_i)'s to be indecomposable and unique as otherwise a VSD algorithm might output some other decomposition, making the recovery of the T_i's hard.

Solving the VSD problem: The authors of [CIK97] gave a polynomial-time algorithm for the symmetric case of the problem when $U = V$. The algorithm works over finite fields, $\mathbb{C}, \mathbb{R}$ but not over $\mathbb{Q}$ (if we wish to output a decomposition over $\mathbb{Q}$). The authors of [GKS20] showed a

reduction of VSD to the symmetric case. The authors of [BGKS22] and [GKS20] exploited the structure of U and V (arising in their settings) to get a VSD algorithm that also works over $\mathbb{Q}$.

The above VSD framework gives rise to a meta algorithm for learning the "terms". To use the algorithm for Theorems 1 and 4, we need appropriate sets of operators $\mathscr{L}_1$ and $\mathscr{L}_2$ that satisfy the conditions mentioned above. The conditions point to four technical steps in the analysis; we state these steps first and then discuss how to execute them for Theorems 1 and 4.

Algorithm 1 Meta algorithm [GKS20]

- **Input**: $f = T_1 + T_2 + ... + T_s$, where T_i's are unknown "simple" terms.
- **Output**: $T_1', T_2' \dots T_s'$ such that $T_i' = T_{\pi(i)}$ for some permutation π on $[s]$.

1. Compute $U = \langle \mathscr{L}_1 \circ f \rangle$ and $V = \langle \mathscr{L}_2 \circ \mathscr{L}_1 \circ f \rangle$.
2. Solve VSD for $(\mathscr{L}_2, U, V)$; find the decompositions $U = U_1 \oplus \cdots \oplus U_s$, $V = V_1 \oplus \cdots \oplus V_s$.
3. Recover T_i' from U_i.

1. **Direct sum structure**: This means establishing $U = U_1 \oplus \cdots \oplus U_s$ and $V = V_1 \oplus \cdots \oplus V_s$.

2. **Uniqueness of decomposition**: Once the direct sum holds, it needs to be shown that the decomposition of U and V is indecomposable with respect to $\mathscr{L}_2$ and unique up to permutations of the U_i's and V_i's. Inspired by the Krull-Schmidt theorem [KSB], [GKS20] analyzed the *adjoint algebra*[6] associated with $(\mathscr{L}_2, U, V)$ and pointed out a sufficient condition for uniqueness to hold. The adjoint algebra for $(\mathscr{L}, U, V)$ is defined as $\text{Adj}(\mathscr{L}, U, V) := \{(D, E) \mid D : U \to U,\ E : V \to V$ are linear maps, and $\forall L \in \mathscr{L},\ LD = EL\}$. The authors of [GKS20] noted that if $\text{Adj}(\mathscr{L}, U, V)$ is block diagonalizable, i.e., $\forall (D, E) \in \text{Adj}(\mathscr{L}, U, V)$ if $D(U_i) \subseteq U_i$ and $E(V_i) \subseteq V_i$, then the decomposition is unique. So, we need to show that $\text{Adj}(\mathscr{L}_2, U, V)$ is block diagonalizable.

3. **Vector space decomposition**: With 1 and 2 satisfied, an algorithm is required to decompose U and V. As mentioned before, the vector space decomposition algorithm in [CIK97] does not quite work over $\mathbb{Q}$. But fortunately, the adjoint algebra comes to the rescue again. Suppose, $\text{Adj}(\mathscr{L}_2, U, V)$ is block diagonalizable. If it is further block *equi-triangularizable* (refer Definition 2.2) and has an element (D, E), where D has s distinct eigenvalues, then the U_i's are the generalized eigenspaces of D which can be computed efficiently[7]. Thus, if $\text{Adj}(\mathscr{L}_2, U, V)$ is block equi-triangularizable, then computing vector space decomposition reduces to computing generalized eigenspaces.

4. **Recovery of** T_i: Finally, once $U_i = \langle \mathscr{L}_1 \circ T_i \rangle$ is obtained, we need to derive T_i from it.

Connecting our problems with the learning problem given by Equation (2): Let $g = \sum_{i \in [s]} c_i m_i$ be a design polynomial and $f = g(A\mathbf{x}) = \sum_{i \in [s]} T_i$, where $T_i = c_i m_i(A\mathbf{x})$ is a product of linear forms. Then, we can learn the unknown transformation A (up to permutation and scaling) by learning and factoring the terms $T_1, \dots T_s$. We do so by implementing the above steps for Theorems 1 and 4; we discuss this next.

[6] The authors of [GKS20] attributed the use of adjoint algebra in their work to a suggestion by Youming Qiao.

[7] The existence of such an element implies that for a random $(D, E) \in \text{Adj}(\mathscr{L}_2, U, V)$, D has s distinct eigenvalues with high probability by the Schwartz-Zippel lemma.

1.2.1 Implementing the four steps for Theorem 1

We choose $\mathscr{L}_1 = \mathscr{L}_2 = \partial^t$. For a (n, d, s, t) design polynomial g with $d \geq 3t$, we show Step 1 in Lemma 3.1 by leveraging the design property, Step 2 by showing that the adjoint is block diagonalizable (Lemma 3.3), and Step 3 by showing further that the adjoint is block equi-triangularizable (Proposition 3.3). Since the input f is in the orbit of g, these properties also hold for f by Lemma 3.2. A term T_i of f is in the orbit of a monomial of g, and so, T_i is a product of linear forms. The recovery process for T_i from U_i is the same as that in [KS19]. The transform and the design polynomial are then obtained from the T_i's (Proposition 3.1).

1.2.2 Implementing the four steps for Theorem 4

We choose $\mathscr{L}_1 = \partial^k$ and $\mathscr{L}_2 = \partial^2$, where $k > t$ is appropriately chosen in Lemma 4.4. First, we show that assuming the two non-degeneracy conditions stated in Section 4.1, all the steps can be implemented. Step 1 is immediate from the first non-degeneracy condition. Lemma 4.1 shows that $\mathrm{Adj}(\partial^2, U, V)$ is block diagonalizable (in fact, block equi-triangularizable) for non-degenerate affine projections. Hence, both Steps 2 and 3 hold. The process of recovering the terms (i.e., Step 4) is the same as that for Theorem 1.

Second, we show that random affine projections of design polynomials are non-degenerate with high probability. The second non-degeneracy condition holds with high probability given the restriction on the field size. If we show that $\dim U = s\binom{d}{k}$, then the direct sum structure holds[8]. By the Schwartz-Zippel lemma, it is sufficient to show that for every design polynomial, there exists an affine projection such that $\dim U = s\binom{d}{k}$. We do this by showing that the probability that $\dim U = s\binom{d}{k}$ is non-zero if f is chosen from a specific class of affine projections as described by the *two-phase* random process in the proof of Lemma 4.4.

1.3 Comparison with previous work

As mentioned earlier, ET has been studied for various polynomial families. ET algorithms for power symmetric polynomials [Kay11] and read-once formulas [GST23] were given by analyzing the factors of the Hessian determinant. Analyzing the Lie algebra of the determinant [Kay12, GGKS19], the permanent [Kay12], and the iterated matrix multiplication [KNST19, MNS20] polynomials led to ET algorithms for these families. For the elementary symmetric polynomials, the maximal dimension of the space of second-order partials gave an ET algorithm [Kay11]. It was shown in [Gro12] that over algebraically closed fields of characteristic 0, ET for polynomials characterized by the continuous part of their symmetries is equivalent to testing matrix isomorphism to their Lie algebras, implying over such fields an efficient ET exists for such polynomials, provided matrix isomorphism to their Lie algebra can be solved efficiently. However, for design polynomials, these techniques *do not* work. For example, the $(2, d, 2, 3)$ design polynomial $x_1 x_2^{d-1} + x_1^{d-1} x_2$, where $d \geq 3$ is odd, has a trivial Lie algebra and an irreducible Hessian determinant over $\mathbb{Q}$. For $d \geq 3$, the $(2d, d, 2, 1)$ design polynomial $\prod_{i=1}^{d} x_i + \prod_{i=d+1}^{2d} x_i$ has second-order partials dimension $2\binom{d}{2}$, which is less than the maximum possible dimension $\binom{2d}{2}$. Design polynomials, particularly NW, are *not* characterized by the continuous part of their symmetries (see [GS19]); hence the ET algorithm implied from [Gro12] does not apply. Thus, a different technique must be used for design polynomials.

The VSD framework was used previously in [KS19, GKS20, BGKS22] to design learning algorithms. The authors of [MS21] showed that polynomials in the orbit of the sum-product polynomial satisfy the non-degeneracy conditions of [KS19], and so, we have an ET algorithm

[8] as $U \subseteq U_1 + \cdots + U_s$ and $\dim U_i \leq \binom{d}{k}$.

for this family. But note that a sum-product polynomial has a read-once formula; an ET algorithm can also be obtained via the Hessian determinant [GST23]. To our knowledge, our work is the first to use the VSD framework to develop an ET algorithm (in Theorem 1) for a natural family of polynomials for which none of the other three techniques work. Also, there is a notable difference between the analysis in [GKS20, BGKS22] and the analysis here. In [GKS20, BGKS22], the relevant adjoint algebra is diagonalizable, while in our case the adjoint is block equi-triangularizable. The VSD algorithms of previous works recovered the component spaces by computing eigenspaces, while we compute generalized eigenspaces to do so.

We learn affine projections of design polynomials (in Theorem 4) under certain non-degeneracy conditions similar to those of [KS19]. However, proving the non-degeneracy of random affine projections of design polynomials requires a more involved analysis than proving the non-degeneracy of random depth-3 circuits in [KS19] (see Sections 1.2.2 and C.4). The reason is, unlike the terms of the circuit models studied in [KS19, GKS20, BGKS22], the terms in our case have *shared randomness* as the same random affine form can appear in multiple terms. Theorem 4 is a significant generalization of the main result in [KS19] that handles random affine projections of the sum-product polynomial, which is a special design polynomial.

The learning algorithm in Theorem 4 is proper as it outputs an appropriate affine map. It is also an average-case algorithm for learning affine projections of design polynomials as the input is a random affine projection. This average-case, proper learning algorithm exploits the property that the space of partial derivatives of an affine projection of a design polynomial is *low* dimensional (under the technical conditions mentioned in the theorem statement) to reduce the learning problem to VSD. A natural question arises at this point: is it always possible to design an average-case, proper learning algorithm (via a reduction to VSD) for affine projections of a model satisfying such low dimensional partial derivatives space property? The answer is unclear. Section A.3 gives an example of affine projections of low width algebraic branching programs (ABPs) satisfying the technical assumptions of Theorem 4 and with low dimensional partial derivatives spaces, and for which (to the best of our knowledge) no average-case, proper learning algorithm is known. The authors of [KNS19] gave such an algorithm assuming that the widths are the same across the layers of the ABP.

2 Preliminaries

2.1 Notations and definitions

For $n \in \mathbb{N}, [n]$ is the set $\{1, 2 \dots n\}$. We use b.b.a to refer to black-box access. The set of $n \times n$ invertible matrices over $\mathbb{F}$ is denoted as $\mathrm{GL}(n, \mathbb{F})$. For two polynomials f and g, $f \sim g$ denotes that f is in the orbit of g. Variable sets are denoted as $\mathbf{x}, \mathbf{y}$ and $\mathbf{z}$. Permutation and scaling matrices are denoted as P and S, respectively. A monomial in $\mathbf{x}$ is denoted as $\mathbf{x}^{\boldsymbol{\alpha}} := x_1^{\alpha_1} x_2^{\alpha_2} \dots x_n^{\alpha_n}$, which has total degree $|\boldsymbol{\alpha}| := \sum_{i=1}^{n} \alpha_i$. The set of degree t derivatives in $\mathbf{x}$ is denoted as ∂^t while $\partial^t f$ denotes the set of degree t derivatives of the polynomial f. The vector space spanned by a set of polynomials S over $\mathbb{F}$ is denoted as $\langle S \rangle$. Typically, g denotes a (n, d, s, t) design polynomial $g = g_1 + g_2 + \cdots + g_s$ where g_i are monomials of degree d, while f denotes the input polynomial $f = g(A\mathbf{x}) = T_1 + T_2 + \cdots + T_s$, where $T_i = g_i(A\mathbf{x})$ for $A \in \mathrm{GL}(n, \mathbb{F})$. Define the spaces $U, U_i, V, V_i, U', U_i', V', V_i'$ as follows:

$$U := \langle \mathscr{L}_1 \circ f \rangle, \quad U' := \langle \mathscr{L}_1 \circ g \rangle, \quad V := \langle \mathscr{L}_2 \circ \mathscr{L}_1 \circ f \rangle, \quad V' := \langle \mathscr{L}_2 \circ \mathscr{L}_1 \circ g \rangle,$$

$$U_i := \langle \mathscr{L}_1 \circ T_i \rangle, \quad U_i' := \langle \mathscr{L}_1 \circ g_i \rangle, \quad V_i := \langle \mathscr{L}_2 \circ \mathscr{L}_1 \circ T_i \rangle, \quad V_i' := \langle \mathscr{L}_2 \circ \mathscr{L}_1 \circ g_i \rangle,$$

where $\mathscr{L}_1 = \mathscr{L}_2 = \partial^t$ (as defined in Section 1.2.1). These spaces will be used in Section 3.

Definition 2.1 (Adjoint Algebra)**.** The adjoint algebra associated with $(\mathscr{L}_2, U, V)$ is defined as $\mathrm{Adj}(\mathscr{L}_2, U, V) := \{(D, E) \mid D : U \to U,\ E : V \to V \text{ are linear maps, and } \forall L \in \mathscr{L}_2,\ LD = EL\}$.

Adjoint algebra was introduced in Section 4.3 of [Wil09] as an associative ring to study the decompositions of bilinear maps and has been used in [BMW17] for developing a fast isomorphism testing algorithm for a subclass of finite p-groups. A meta-framework for designing learning algorithms for arithmetic circuits was given in [GKS20], where the learning problem was reduced to the vector space decomposition problem, and the uniqueness of vector space decomposition was proved by analyzing a certain adjoint algebra (refer [GKS20] for details).

Definition 2.2 (Equi-triangular matrix)**.** An equi-triangular matrix is triangular with equal diagonal entries. Block equi-triangularizability and simultaneous block equi-triangularizability are defined similarly as block diagonalizability and simultaneous block diagonalizability.

2.2 Algorithmic preliminaries

Fact 2.1. Given black-box access to an n-variate degree d polynomial f, we can compute black-box access to $\frac{\partial^k f}{\partial \mathbf{x}^{\boldsymbol{\alpha}}}$ in $\mathrm{poly}(n, d^k)$ time, where $|\boldsymbol{\alpha}| = k$. (Refer [KNST19] for a proof idea.)

Fact 2.2. Given black-boxes to n-variate degree d polynomials $f_1, f_2 \ldots, f_l$, there is a randomized $\mathrm{poly}(n, l, d)$ time algorithm that computes a basis for the vector space

$$(f_1, f_2 \ldots f_l)^{\perp} := \{(\alpha_1, \alpha_2, \ldots, \alpha_l) \in \mathbb{F}^l : \sum_{i=1}^{l} \alpha_i f_i = 0\}$$

Refer [Kay11] for a proof of the above fact. A corollary of Fact 2.2 is:

Fact 2.3. Given black-box access to linearly independent polynomials $f_1, \ldots, f_l$ and an $f = \sum_{i=1}^{l} \beta_i f_i$, where $\beta_i \in \mathbb{F}$, the β_i's can be computed in randomized $\mathrm{poly}(n, l, d)$ time.

Fact 2.4. Let $d \in \mathbb{N}$, $\mathrm{char}(\mathbb{F}) = 0$ or $> d$ and $|\mathbb{F}| \geq d^6$. Given black-box access to an n-variate degree d polynomial f, black-box access to its irreducible factors can be computed in randomized $\mathrm{poly}(n, d)$ time. (Refer [KT90] for details.)

3 Equivalence testing for design polynomials

We state the ET algorithm in Algorithm 2. The precise time complexity of the algorithm is $\mathrm{poly}(\binom{n+t}{n} d^t)$, we discuss this further in Section 3.2 which analyzes the ET algorithm. Section 3.3 analyzes the adjoint algebra of a design polynomial g. Based on the structure of the adjoint, Section 3.4 develops and analyses a VSD algorithm. In Section 3.5, Theorems 2, 3 and 5 are proven. The proofs of all lemmas and propositions in this section can be found in Section B.

3.1 The algorithm

Step 1: As discussed in Section 1.2, $\mathscr{L}_1 = \partial^t$ and $\mathscr{L}_2 = \partial^t$ are used to define U and V. Black-box access to bases of U and V is computable using Facts 2.1 and 2.2.
Step 2: The vector space decomposition algorithm of Section 3.4 gives b.b.a to bases of U_i's.
Step 3: Each $\frac{\partial^t f}{\partial \mathbf{x}^{\boldsymbol{\alpha}}}$ is expressible as a sum of $u_{i\boldsymbol{\alpha}} \in U_i$'s, where each $u_{i\boldsymbol{\alpha}}$ is $\frac{\partial^t T_i}{\partial \mathbf{x}^{\boldsymbol{\alpha}}}$. Using Fact 2.3, b.b.a to $\frac{\partial^t f}{\partial \mathbf{x}^{\boldsymbol{\alpha}}}$ and U_i's, black-boxes to $u_{i\boldsymbol{\alpha}}$ can be obtained. It can be verified using Lagrange's formula that the described black-box for T_i is the correct one.
Step 4: The proof of Proposition 3.1 details the recovery process for B and h.

Proposition 3.1. Assuming b.b.a to the T_i's, B and h can be recovered in $\mathrm{poly}(n, s, d)$ time.

Algorithm 2 Equivalence testing for design polynomials

Input: Black-box access to an $f \in \mathbb{F}[\mathbf{x}]$, where $f = g(A\mathbf{x})$ for some unknown (n, d, s, t) design polynomial g and $A \in \mathrm{GL}(n, \mathbb{F})$.

Output: A matrix B and a (n, d, s, t) design polynomial h, where $B = PSA$ and $f = h(B\mathbf{x})$ for some permutation matrix P and scaling matrix S.

/* Performing VSD */

1. Compute black-box access to bases of $U = \langle \partial^t f \rangle$ and $V = \langle \partial^t U \rangle$.
2. Perform VSD for (∂^t, U, V) using Algorithm 3. Let $U = U_1 \oplus \cdots \oplus U_s$ be the decomposition returned by Algorithm 3.

/* Recovering black-box access to T_i */

3. Express $\frac{\partial^t f}{\partial \mathbf{x}^{\boldsymbol{\alpha}}} = u_{1\boldsymbol{\alpha}} + \cdots + u_{s\boldsymbol{\alpha}}$, where $u_{i\boldsymbol{\alpha}} \in U_i$ and obtain black-box access to $u_{i\boldsymbol{\alpha}}$ for all $i \in [s]$ and $\mathbf{x}^{\boldsymbol{\alpha}}$ of degree t. The black-box for T_i is given by $\frac{(d-t)!}{d!} \sum_{|\boldsymbol{\alpha}|=t} \binom{t}{\alpha_1 \ldots \alpha_n} \mathbf{x}^{\boldsymbol{\alpha}} u_{i\boldsymbol{\alpha}}(\mathbf{x})$.

/* Recovering B and h */

4. From black-box access to the T_i's, recover and return a $B \in \mathrm{GL}(n, \mathbb{F})$ and a (n, d, s, t) design polynomial h such that $f = h(B\mathbf{x})$ using Proposition 3.1.

3.2 Analysis of the algorithm

Each step is randomized with a small probability of error. For the analysis, we assume that each step executes correctly. An implicit check is made at the end to see if h is a (n, d, s, t) design polynomial, B is invertible and $f = h(B\mathbf{x})$, failing which the algorithm is repeated.

The correctness of the algorithm holds if it executes each of the four steps listed in Section 1.2 correctly. By Lemmas 3.1 and 3.2 (given below), U and V have the required direct sum structure. The correctness of the vector space decomposition follows from the correctness of Facts 2.1 and 2.2 and that of the vector space decomposition algorithm of Section 3.4. The uniqueness of decomposition follows from Proposition 3.3 and Lemmas 3.2 and 3.6. The correctness of the recovery of T_i's, B and h follows from Fact 2.3 and Proposition 3.1.

Let U', U_i', V', V_i' be as defined in Section 2.1.

Lemma 3.1. For $d \geq 3t$, $U' = U_1' \oplus U_2' \cdots \oplus U_s'$ and $V' = V_1' \oplus V_2' \cdots \oplus V_s'$.

The following proposition gives the time complexity of the algorithm.

Proposition 3.2. Algorithm 2 has a running time of $\mathrm{poly}(\binom{n+t}{n} d^t)$.

Typically $n > d$, in which case the running time is $\mathrm{poly}(n^t)$.

Remark 3.1. As stated in a remark of Theorem 1, if f is given as a circuit and $|\mathbb{F}|$ is small, we can work over an extension of $\mathbb{F}$. Fact 2.4 returns black-box access to the linear factors, up to scaling by the extended field elements. By interpolating the linear forms and appropriately rescaling them, the linear forms can be assumed to be over $\mathbb{F}$. The resulting product of linear forms is scaled by a constant c, which must belong to $\mathbb{F}$ because the product of linear forms is $T_i \in \mathbb{F}[\mathbf{x}]$. Thus, Algorithm 2 is unaffected when working over a field extension.

Remark 3.2. The complexity of Algorithm 2 is optimal, if $d \ll n$. This can be shown using a simple information-theoretic argument. For simplicity, we assume that the underlying field is $\mathbb{F}_p$; an appropriate dimension-based argument holds over other fields. Corollary 1.2 partitions the set of design polynomials into PS-equivalence classes. Let $s = (\frac{n}{d})^t$. Over $\mathbb{F}_p$, there are

$\geq p^s$ design polynomials[9], and the size of an equivalence class is $\leq n!p^n$. Thus, the number of classes $N \geq p^{s-n}/n! = \Theta(p^s)$ assuming $t \geq 2$. Each class needs $\geq \log(N)$ information bits to represent. Thus, any ET algorithm that identifies the equivalence class of f from black-box access to f requires time at least $\log(N)$ as the black-box needs to communicate $\geq \log(N)$ information bits. Note, $\log(N)^{O(1)} = \text{poly}(s, \log(p))$ matches the complexity of Algorithm 2.

3.3 Structure of the adjoint algebra

Once we fix bases of the spaces U, U', V and V', a linear operator from one of these spaces to another can be naturally viewed as a matrix. Thus, $\text{Adj}(\partial^t, U, V)$ and $\text{Adj}(\partial^t, U', V')$ are sets of tuples of matrices with respect to appropriately chosen bases. We now show that $\text{Adj}(\partial^t, U, V)$ is block equi-triangularizable, i.e., the set of matrices $\{D : (D, E) \in \text{Adj}(\partial^t, U, V)\}$ is simultaneously block equi-triangularizable. By Lemma 3.2, it suffices to show this for $\text{Adj}(\partial^t, U', V')$. Note, after fixing bases of U' and V', the operators $\partial^t : U' \to V'$ are also matrices. When we say $\text{Adj}(\partial^t, U, V)$ is equal to $\text{Adj}(\partial^t, U', V')$ in the lemma below, we mean they are equal as sets of tuples of matrices with respect to appropriately chosen bases for U, V, U' and V'.

Lemma 3.2. Let U, V, U_i, V_i, U', V', U_i' and V_i' be vector spaces as defined in Section 2.1 with $\mathscr{L}_1 = \mathscr{L}_2 = \partial^t$. Define the invertible map $\phi : \mathbb{F}[\mathbf{x}] \to \mathbb{F}[\mathbf{x}]$ as $\phi(p) := p(A\mathbf{x})$ for $p \in \mathbb{F}[\mathbf{x}]$, where $A \in \text{GL}(n, \mathbb{F})$ is as in Algorithm 2. Then,

(i) $U' \cong U$, $V' \cong V$, $U_i' \cong U_i$ and $V_i' \cong V_i$ via the map ϕ. In other words, if $\mathscr{B}$ is a basis of U', $\phi(\mathscr{B})$ is a basis of U. This holds similarly for the other spaces.

(ii) Let $\mathscr{B}_1'$ and $\mathscr{B}_2'$ be bases for U' and V' respectively, with $\phi(\mathscr{B}_1')$ and $\phi(\mathscr{B}_2')$ being basis of U and V respectively. Then, $\text{Adj}(\partial^t, U', V')$, with respect to bases $\mathscr{B}_1'$ and $\mathscr{B}_2'$, is equal to $\text{Adj}(\partial^t, U, V)$, with respect to bases $\phi(\mathscr{B}_1')$ and $\phi(\mathscr{B}_2')$.

Lemma 3.3. If $(D', E') \in \text{Adj}(\partial^t, U', V')$, then $D'(U_i') \subseteq U_i'$ and $E'(V_i') \subseteq V_i'$ for all $i \in [s]$.

The direct sum structure of U' and V' implies that every $\partial^t : U' \to V'$ is block diagonal, with respect to the monomial basis of U' and V', and by Lemma 3.3, $\text{Adj}(\partial^t, U', V')$ is block diagonal with respect to these bases. This and the adjoint condition imply that $\text{Adj}(\partial^t, U', V')$ comprises the adjoints of the g_i's. Thus, it suffices to analyze the adjoint of the g_i's. The adjoint of a monomial need not be trivial, as shown in Section B.10. We show that the adjoint algebra of a monomial is equi-triangularizable. For g_i, an arbitrary monomial of g, $\mathscr{B}_i' := \{\partial^t g_i\}$ is a basis[10] for U_i'. For $(D', E') \in \text{Adj}(\partial^t, U', V')$, let D_i' and E_i' be the restriction of D' and E' to U_i' and V_i' respectively. Represent D_i' as a $\dim(U_i') \times \dim(U_i')$ matrix with respect to $\mathscr{B}_i'$, where $D_i'[\mathbf{x}^{\boldsymbol{\alpha}}][\mathbf{x}^{\boldsymbol{\beta}}]$ is the coefficient of $\frac{\partial^t g_i}{\partial \mathbf{x}^{\boldsymbol{\alpha}}}$ in $D_i'\left(\frac{\partial^t g_i}{\partial \mathbf{x}^{\boldsymbol{\beta}}}\right)$. Lemma 3.4 shows when $D_i'[\mathbf{x}^{\boldsymbol{\alpha}}][\mathbf{x}^{\boldsymbol{\beta}}]$ is 0 and that all entries $D_i'[\mathbf{x}^{\boldsymbol{\alpha}}][\mathbf{x}^{\boldsymbol{\alpha}}]$ are equal. We use the notation $\text{mon}(c \cdot \mathbf{x}^{\boldsymbol{\alpha}}) := \mathbf{x}^{\boldsymbol{\alpha}}$, where $c \in \mathbb{F}\backslash\{0\}$; the definition is naturally extended to a set of monomials. For e.g., $\text{mon}\{2x_1^4, 5x_1x_2\} = \{x_1^4, x_1x_2\}$.

Lemma 3.4. Let $|\boldsymbol{\alpha}| = |\boldsymbol{\beta}| = t$, $\frac{\partial^t g_i}{\partial \mathbf{x}^{\boldsymbol{\alpha}}} \neq 0$ and $\frac{\partial^t g_i}{\partial \mathbf{x}^{\boldsymbol{\beta}}} \neq 0$. Let D_i' be as above. Then,

(i) $D_i'[\mathbf{x}^{\boldsymbol{\alpha}}][\mathbf{x}^{\boldsymbol{\beta}}] = 0$, if $\text{mon}\{\partial^t(\frac{\partial^t g_i}{\partial \mathbf{x}^{\boldsymbol{\alpha}}})\} \not\subseteq \text{mon}\{\partial^t(\frac{\partial^t g_i}{\partial \mathbf{x}^{\boldsymbol{\beta}}})\}$, and

(ii) $D_i'[\mathbf{x}^{\boldsymbol{\alpha}}][\mathbf{x}^{\boldsymbol{\alpha}}] = D_i'[\mathbf{x}^{\boldsymbol{\beta}}][\mathbf{x}^{\boldsymbol{\beta}}]$.

The following proposition shows that $\text{Adj}(\partial^t, U', V')$ is block equi-triangularizable.

[9] The $\text{NW}_{q,d,t}$ polynomial has $s = \binom{n}{d}^t$ monomials; varying the coefficients over $\mathbb{F}_p$ gives rise to p^s distinct design polynomials.

[10] Assume that the set $\{\partial^t g_i\}$ consists of only the nonzero derivatives.

Proposition 3.3. A basis $\mathscr{B}'$ for U' exists with respect to which any D', where $(D', E') \in \mathrm{Adj}(\partial^t, U', V')$ for some E', is block equi-triangular.

As detailed in the proof of the above proposition, reordering each $\mathscr{B}'_i$ and concatenating them gives $\mathscr{B}'$. For each $\mathscr{B}'_i$, a directed acyclic graph G_i is constructed with vertices as $\mathscr{B}'_i$. For $\frac{\partial^t g_i}{\partial \mathbf{x}^\alpha}, \frac{\partial^t g_i}{\partial \mathbf{x}^\beta} \in \mathscr{B}'_i$, if $\mathrm{mon}\{\partial^t\left(\frac{\partial^t g_i}{\partial \mathbf{x}^\alpha}\right)\} \subseteq \mathrm{mon}\{\partial^t\left(\frac{\partial^t g_i}{\partial \mathbf{x}^\beta}\right)\}$ then an edge from $\frac{\partial^t g_i}{\partial \mathbf{x}^\alpha}$ to $\frac{\partial^t g_i}{\partial \mathbf{x}^\beta}$ exists in G_i. The topological sort of G_i gives the reordering of $\mathscr{B}'_i$. By Lemma 3.4, D'_i is equi-triangular with respect to the reordered $\mathscr{B}'_i$ implying D' is block equi-triangular with respect to $\mathscr{B}'$.

3.4 Vector space decomposition

Algorithm 3 Vector space decomposition algorithm

Input: B.b.a to bases of spaces U and V.
Output: B.b.a to bases of spaces $W_1, \ldots, W_s$, where $W_i = U_{\pi(i)}$ for some permutation π.

1. Compute a basis $D^{(1)}, \ldots, D^{(b)}$ of $\mathrm{Adj}(\partial^t, U, V)_1 := \{D \mid (D, E) \in \mathrm{Adj}(\partial^t, U, V)\}$.
2. Pick $c_1, \ldots, c_b \in_r S \subseteq \mathbb{F}$, where $|S| = s^3$. Let $D := c_1 D^{(1)} + \cdots + c_b D^{(b)}$.
3. Factorize the characteristic polynomial of D and obtain its eigenvalues $\lambda_1, \cdots, \lambda_s$.
4. Compute $W_i := \mathrm{Ker}((D - \lambda_i I)^{\dim\ U})$ for all $i \in [s]$ and output b.b.a to bases of $W_1, \ldots, W_s$.

Step 1 is executed by solving the linear system arising from $LD = EL$, for $L \in \mathscr{L}_2$, with the entries of D and E as the variables. In Step 2, a random linear combination of the $D^{(i)}$'s gives a random operator D. The eigenvalues of D can be found by factorization of the characteristic polynomial.[11] Step 4 involves the computation of the generalized eigenspaces of D.

Lemma 3.5. D has s distinct eigenvalues with probability $\geq 1 - \frac{\binom{s}{2}}{|S|}$.

Lemma 3.6. Let $D \in \mathrm{Adj}(\partial^t, U, V)$ such that it has s distinct eigenvalues $\lambda_1, \ldots, \lambda_s$. Then, $\mathrm{Ker}((D - \lambda_i I)^{\dim\ U}) = U_{\pi(i)}$ for all $i \in [s]$, and for some permutation π on $[s]$.

Note, Lemma 3.6 proves the uniqueness of decomposition as well as the indecomposability of U_i and V_i with respect to ∂^t.

Proposition 3.4. Algorithm 3 has a running time of $\mathrm{poly}(\binom{n+t}{n} d^t)$.

Handling non-homogeneous polynomials and translations. For non-homogeneous (n, d, s, t) design polynomials, if the degree of *all* monomials is $\geq 3t$, Lemmas 3.1, 3.3 and Proposition 3.3 hold. When $f(\mathbf{x}) = g(A\mathbf{x} + \mathbf{b})$ for $\mathbf{b} \in \mathbb{F}^n$, then since this transform is also invertible, a lemma similar to Lemma 3.2 holds. The analysis then proceeds in the same way. Proposition 3.1 can also recover translations. For multilinear (n, d, s, t) design polynomials, $\mathscr{L}_2 = \partial^1$ with the adjoint $\mathrm{Adj}(\partial^1, U, V)$ improves the bound on t to $d \geq 2t + 1$. Lemmas 3.1, 3.3 and 3.4 hold with some minor changes, and $\mathrm{Adj}(\partial^1, U', V')$ can be shown to be trivial.

3.5 Applications of the equivalence test

3.5.1 $GI \equiv_p PE$ for design polynomials: Proof of Theorem 2

GI $\leq_p$ PE for design polynomials: Let $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ be two n-vertex simple graphs with e edges each. Let there be an arbitrary ordering on the edges of both graphs with

[11] Here, we need an efficient univariate polynomial factorization algorithm over $\mathbb{F}$.

$\boldsymbol{I}_1$ an index function mapping E_1 to $[e]$ and $\boldsymbol{I}_2$ similarly mapping E_2 to $[e]$. Introduce variables $x_1, \ldots, x_n$ and $y_1, \ldots, y_e$. Construct

$$M_1 := \{x_i x_j y^4_{\boldsymbol{I}_1(i,j)} : (i,j) \in E_1\} \text{ and } M_2 := \{x_l x_k y^4_{\boldsymbol{I}_2(l,k)} : (l,k) \in E_2\}.$$

Let $h_1(\mathbf{z}) := \sum_{m \in M_1} m$ and $h_2(\mathbf{z}) := \sum_{m \in M_2} m$, where $\mathbf{z} = \mathbf{x} \sqcup \mathbf{y}$. Clearly, h_1 and h_2 are $(n+e, 6, e, 2)$ design polynomials with all-one coefficients constructible in poly(n) time. Using Corollary 1.2, the coefficients being 1, and $\mathbf{y}$ variables having degree 4, Proposition 3.5 holds.

Proposition 3.5. For h_1, h_2 as above, $G_1 \cong G_2 \iff h_1 \sim h_2$.

PE for design polynomials $\leq_p$ GI: Let $h_1(\mathbf{x})$ and $h_2(\mathbf{x})$ be (n, d, s, t) design polynomials with all-one coefficients, satisfying $d \geq 3t$. If h_1 and h_2 are multilinear, construct hypergraphs H_1 and H_2 with $\mathbf{x}$ as the vertices and subsets of vertices corresponding to the monomials of h_1 and h_2 as the hyperedges, respectively. Observe that $h_1 \sim h_2$ iff $H_1 \cong H_2$ as, by Corollary 1.2, if $h_1 \sim h_2$, then they are equivalent via a permutation matrix. It is well-known that hypergraph isomorphism reduces to GI (refer [Mil79]). If h_1 and h_2 are non-multilinear, the argument is a bit more elaborate: Now the monomials correspond to multisets of $\mathbf{x}$, while hyperedges need to be subsets of vertices. This can be handled by examining a standard reduction from hypergraph isomorphism to GI that uses bipartite graphs (see the opening paragraph of [ADKT15]). By introducing in-between vertices to handle parallel edges, both graphs G_1, G_2 can be constructed in poly(s) time. More details are given in the proof of the following proposition.

Proposition 3.6. For graphs G_1, G_2 as above, $G_1 \cong G_2 \iff h_1 \sim h_2$.

3.5.2 Cryptanalysis of Patarin's scheme: Proof of Theorem 3

Patarin's authentication scheme [Pat96] is based on the presumed hardness of PE for random polynomials of constant degree. It is a provably perfect zero-knowledge authentication scheme; thus Alice can prove to Bob that she knows a secret without revealing *any* information about the secret. The key generation process is as follows:

1. Select an n-variate degree d polynomial $f(\mathbf{x}) \in_r \mathbb{F}_q[\mathbf{x}]$, where d is a constant.
2. Select two matrices $A_1, A_2 \in_r \mathbb{F}_q^{n \times n}$.[12]
3. Compute the public key $(f_1, f_2) := (f(A_1\mathbf{x}), f(A_2\mathbf{x}))$ and the private key $C = A_2^{-1}A_1$.

The authentication procedure is as follows:

1. Alice selects an $R \in_r \mathbb{F}_q^{n \times n}$ and computes $g := f_1(R\mathbf{x})$ and sends it to Bob.
2. Bob receives g and picks $h := f_1$ or f_2 with probability $\frac{1}{2}$, challenging Alice to show $h \sim g$.
3. Alice receives h. If $h = f_1$, she sends R. If $h = f_2$, she sends CR.

The attack. (We elaborate on each of the following three steps below.)

1. **Invoking Theorem 1:** Invoke Theorem 1 on f_1 and f_2 to obtain $h_1 \sim f_1$ and $h_2 \sim f_2$.
2. **Recovering P:** Use Theorem 2 to construct graphs G_1 and G_2 corresponding to h_1 and h_2 and use Babai's algorithm [Bab16] for GI to recover a permutation matrix P.

[12] A random matrix is invertible with high probability.

3. **Recovering S:** Solve the system of monomial equations arising from $h_2(\mathbf{x}) = h_1(PS\mathbf{x})$, with the entries of S as variables.

The attack relies on Lemma 3.7, Propositions 3.7, 3.8, Theorems 1, 2 and Corollary 1.2.

Lemma 3.7. A random s-sparse polynomial, as per Definition 1.4, is a (n, d, s, t) design polynomial with probability at least $1 - \epsilon$, if $n > d^2$ and $s \leq \sqrt{\epsilon}(\frac{n}{d^2})^{\frac{t}{2}}$ where $0 < \epsilon < 1$.

Proposition 3.7. If f is a random s-sparse polynomial as per Lemma 3.7 with $s \geq n^3$, $n > d^8$, $d \geq 25$, $t = \frac{d}{3}$ and $\epsilon = 0.01$, f has no non-trivial permutation symmetry with high probability.[13]

Proposition 3.8. Let f be as in Proposition 3.7. For $h_1(\mathbf{x}) = f(P_1S_1\mathbf{x})$ and $h_2(\mathbf{x}) = f(P_2S_2\mathbf{x})$, where P_1, P_2 are permutation matrices and S_1, S_2 are scaling matrices, there exists a unique permutation matrix P such that $h_2(\mathbf{x}) = h_1(PS\mathbf{x})$ for some scaling matrix S.

Remark 3.3. For a sparse polynomial, $s = \text{poly}(n)$. If $d < n^{1/2-\delta}$ (where $\delta \in (0, 1/2)$) is greater than a sufficiently large constant, and $\epsilon = 0.01$, then t can be chosen to be a constant $\leq d/3$ such that $s \leq \sqrt{\epsilon}(\frac{n}{d^2})^{\frac{t}{2}}$. So, by Lemma 3.7, a random sparse polynomial satisfying the above degree constraint is a (n, d, s, t) design polynomial. Theorem 1 then gives a $O(\text{poly}(n^t)) = \text{poly}(n)$ time ET algorithm for random sparse polynomials.

Let us now discuss the steps of the above-mentioned attack on Patarin's scheme.

Invoking Theorem 1. For $n > d^8$, $\epsilon = 0.01$, $t = d/3$, $d \geq 25$ and $n^3 \leq s \leq 0.1(\frac{n}{d^2})^{\frac{d}{6}}$, Lemma 3.7 implies f is, with high probability, a $(n, d, s, d/3)$ design polynomial. Thus, invoking Theorem 1 on f_1 and f_2 gives h_1, h_2, $P_1S_1A_1$ and $P_2S_2A_2$ where $f_1 = h_1(P_1S_1A_1\mathbf{x})$ and $f_2 = h_2(P_2S_2A_2\mathbf{x})$. Clearly, $h_1 \sim h_2$ by the transform $P_1S_1(P_2S_2)^{-1} = PS$ for appropriate P and S. If P and S can be recovered, then $A_1^{-1}A_2 = (P_1S_1A_1)^{-1}PS(P_2S_2A_2)$ can be recovered. As $n > d^8$ and d is a constant, this step requires $\text{poly}(n)$ time.

Recovering P. Note that P maps the monomials of h_1 to h_2 while S scales the coefficients accordingly. To recover P, treat h_1 and h_2 as design polynomials with all-one coefficients and use Theorem 2 and the GI algorithm of [Bab16]. This step can be done in quasi-$\text{poly}(s)$ time. The uniqueness of P, which holds by Propositions 3.8 and 3.7, implies the correctness of this step.

Recovering S. Let $h_1(\mathbf{x}) = \sum_{i=1}^{s} c_i m_i$ and $h_2(\mathbf{x}) = \sum_{i=1}^{s} \tilde{c}_i m_i$. Now $h_2(\mathbf{x}) = h_1(PS\mathbf{x}) = h_1(S'P\mathbf{x})$ for an appropriate scaling matrix S'. Treat the diagonal entries of S' as variables $\{z_1, z_2 \dots z_n\}$. Equating the coefficients of the monomials, we get $c_i m_i(z_1, \cdots, z_n) = \tilde{c}_j$ where $m_j = m_i(P\mathbf{x})$. If $m_i = x_1^{\alpha_{i,1}} x_2^{\alpha_{i,2}} \dots x_n^{\alpha_{i,n}}$, we get the following monomial equations:

$$z_1^{\alpha_{i,1}} z_2^{\alpha_{i,2}} \dots z_n^{\alpha_{i,n}} = \tilde{c}_j c_i^{-1} \quad \forall\, i \in [s]. \tag{3}$$

There are s such equations in n variables, which is converted to a system of linear equations by taking $\log(z_j)$ as variables and $\alpha_{i,j}$ and $\log(\tilde{c}_j c_i^{-1})$ as constants. Computing $\log(a)$ over $\mathbb{F}_q$ is finding the discrete logarithm of a with respect to a generator γ of $\mathbb{F}_q^{\times}$.[14] Since $q = O(\text{poly}(n))$, γ can be found and discrete log can be computed in $O(\text{poly}(n))$ time. We get a system of s linear equations over $\mathbb{Z}_{q-1}$ which can be solved in $\text{poly}(s, q)$ time using the Chinese Remainder Theorem, refer Chapter 5 of [vzGG03] for details.

[13] meaning, for any permutation matrix P, $f(P\mathbf{x}) = f(\mathbf{x})$ implies P is the identity matrix.
[14] that is finding a $b \in [0, q-2]$ such that $\gamma^b = a$.

3.5.3 Equivalence testing for NW

The PS-equivalence testing problem for NW is as follows: given a polynomial f, check if $f = \mathsf{NW}_{q,d,t}(PS\mathbf{x})$ for some permutation P and scaling S, and recover them if so. Theorem 5, which is proved in Section B.18, follows from Theorems 1 and 2, and the GI algorithm in [Bab16].

Theorem 5. *Let $q, d, t \in \mathbb{N}$, $q \geq d$, $t \leq d/3$, $|\mathbb{F}| \geq q^{3t}$, $\mathrm{char}(\mathbb{F}) = 0$ or $> d$. ET for $\mathsf{NW}_{q,d,t}$ reduces to PS-equivalence testing for $\mathsf{NW}_{q,d,t}$ in $\mathrm{poly}(q^t)$ time. Further, PS-equivalence testing for $\mathsf{NW}_{q,d,t}$ reduces to S-equivalence testing for $\mathsf{NW}_{q,d,t}$ in $\text{quasi-poly}(q^t)$ time.*

S-equivalence testing for NW. The S-equivalence testing problem for NW is as follows: Given a polynomial f, check if $f = \mathsf{NW}_{q,d,t}(S\mathbf{x})$ for some scaling S, and recover it if so. Over $\mathbb{R}$ and $\mathbb{F}_p$, S-equivalence testing for NW can be done by the algorithm of [BRS17] in $\mathrm{poly}(q, \beta)$ time, where β is the bit complexity of the coefficients of f, and also by an algorithm of [GS19]. Over $\mathbb{Q}$, S-equivalence testing of NW can be done in $\mathrm{poly}(q^t, \beta)$ time, assuming oracle access to integer-factoring. This combined with Theorem 5 gives a $\text{quasi-poly}(q^t)$ time algorithm for ET for NW. Here is a proof sketch of S-equivalence test for $\mathsf{NW}_{q,d,t}$ over $\mathbb{Q}$: If f is S-equivalent to $\mathsf{NW}_{q,d,t}$, then $f = \sum_{h \in \mathbb{F}_q[y],\ \deg h < t} c_h \prod_{i=0}^{d-1} x_{i,h(i)}$, where $c_h \in \mathbb{F}$. With the entries of S as $\mathbf{z}$ variables, we get the following equations:

$$z_{0,h(0)} z_{1,h(1)} \cdots z_{d-1,h(d-1)} = c_h^{-1} \quad \text{for } h \in \mathbb{F}_q[y]_{<t} \,.$$

There are q^t many equations in qd variables. Since $c_h^{-1} \in \mathbb{Q}$, $c_h^{-1} = a/b$ for some $a, b \in \mathbb{Z}$. Using the integer-factoring oracle, factor the integers a, b into primes $p_1, p_2 \ldots p_l$. Now, reduce it to solving an appropriate Diophantine linear system by taking logarithms to the base p_i, with $\log_{p_i}(z_{j,h(j)})$ as variables $w_{j,h(j),i}$. This Diophantine linear system has lqd variables and lq^t equations. Treating this system as a matrix, check its consistency and then determine the linearly independent rows (say there are k of them). A $k \times k$ submatrix with non-zero determinant m can be formed from these rows, which corresponds to expressing the Diophantine linear system as linear equations in k of the $w_{j,h(j),i}$ variables with constants as affine forms in the remaining $w_{j,h(j),i}$ variables. By Cramer's rule, a solution to such a system is a fraction with the numerator as the affine forms and the denominator as m. The problem then further reduces to solving a linear system determined by the affine forms over the ring $\mathbb{Z}_m$ since $w_{j,h(j),i}$ must be integers. This whole process can be done in $\mathrm{poly}(q^t, \beta)$ time.

Therefore, ET for NW can be solved in time quasi-polynomial in the sparsity of $\mathsf{NW}_{q,d,t}$.

4 Learning random affine projections of design polynomials

In this section, Theorem 4 is proven. Section 4.1 lists the non-degeneracy conditions imposed on affine projections of design polynomials. In Sections 4.2 and 4.4, we state and analyze the learning algorithm and the vector space decomposition algorithm, respectively. The adjoint of non-degenerate affine projections is analyzed in Section 4.3. In Section 4.5, we show random affine projections of multilinear design polynomials are non-degenerate with high probability.

4.1 Non-degeneracy conditions

Let $g(\mathbf{y}) = g_1 + \cdots + g_s$ be a multilinear (m, d, s, t) design polynomial with g_i's as monomials. Let $f(\mathbf{x}) = g(l_1, l_2 \ldots l_m) = T_1 + \cdots + T_s$ be an n-variate affine projection of g with $T_i =$

$g_i(l_1, \cdots, l_m)$, a product of d linear forms and L_i be the set of linear forms in T_i. We say f is a random affine projection if the coefficients of the l_i's are randomly chosen from $\mathbb{F}$. Define:

$$U := \langle \partial^k f \rangle, \;\; V := \langle \partial^{k+2} f \rangle, \;\; U_i := \langle \partial^k T_i \rangle, \;\; V_i := \langle \partial^{k+2} T_i \rangle,$$

where k is as in Lemma 4.4 and $d \geq 2k+2$. We say f is non-degenerate if the following holds:

1. $U = U_1 \oplus \cdots \oplus U_s$ and $V = V_1 \oplus \cdots \oplus V_s$.
2. The set L_i is $\mathbb{F}$-linearly independent for all $i \in [s]$.

4.2 The algorithm and its analysis

Algorithm 4 is similar to Algorithm 2, except $\mathscr{L}_1 = \partial^k$ and $\mathscr{L}_2 = \partial^2$ define U and V respectively and Algorithm 5 for vector space decomposition computes eigenspaces. Each step of the algorithm is randomized with a small error probability. An implicit check is run at the end of Step 4 to see if h is a multilinear (m, d, s, t) design polynomial and the linear forms per T_i are linearly independent, failing which the algorithm is repeated.

Algorithm 4 Learning random affine projections of multilinear design polynomials

Input: B.b.a to $f = g(A\mathbf{x})$, where $A \in \mathbb{F}^{m \times n}$ and g is some unknown (m, d, s, t) polynomial.
Output: A matrix B and (m, d, s, t) design polynomial h, where $B = PSA$ and $f = h(B\mathbf{x})$.

/* Performing VSD */

1. Compute black-box access to some bases of $U = \langle \partial^k f \rangle$ and $V = \langle \partial^2 U \rangle$.
2. Perform VSD for (∂^2, U, V): let $U = U_1 \oplus \cdots \oplus U_s$.

/* Recovering black-box access to T_i */

3. Express $\frac{\partial^k f}{\partial \mathbf{x}^{\boldsymbol{\alpha}}} = u_{1\boldsymbol{\alpha}} + \cdots + u_{s\boldsymbol{\alpha}}$, where $u_{i\boldsymbol{\alpha}} \in U_i$ and obtain black-box access to $u_{i\boldsymbol{\alpha}}$ for all $i \in [s]$ and $\mathbf{x}^{\boldsymbol{\alpha}}$ of degree k. The black box for T_i is $\frac{(d-k)!}{d!} \sum_{|\boldsymbol{\alpha}|=k} \binom{k}{\alpha_1 \ldots \alpha_n} \mathbf{x}^{\boldsymbol{\alpha}} u_{i\boldsymbol{\alpha}}(\mathbf{x})$.

/* Recovering B and h */

4. From black box access to the T_i's, recover and return $B \in \mathbb{F}^{m \times n}$ and (m, d, s, t) design polynomial h using Proposition 4.1.

Analysis and Time complexity: We assume that each step executes without error and f is non-degenerate, which holds for a random affine projection with high probability by Lemma 4.4. The correctness of Algorithm 4 holds if it executes each of the four steps listed in Section 1.2 correctly. Non-degeneracy condition 1 implies the direct sum structure of U and V. The correctness of VSD follows from that of Facts 2.1, 2.2 and Algorithm 5. The uniqueness of decomposition follows from Lemmas 4.1 and 4.3. The correctness of the recovery of T_i's, B and h follows from Fact 2.3 and Proposition 4.1. Proposition 4.2 gives the time complexity.

Proposition 4.1. The matrix B and polynomial h are recoverable in $\text{poly}(m, n, s, d)$ time, assuming b.b.a to the T_i's.

Proposition 4.2. Algorithm 4 has a running time of $\text{poly}(m, s, n^t)$.

Remark 4.1. The precise bound on $|\mathbb{F}|$ is $|\mathbb{F}| \geq \max(s^3, d^7, d^{t+4} s^{1+4/\epsilon})$, where ϵ is the constant from $n \geq d^{4+\epsilon}$. The use of Schwartz-Zippel lemma in Algorithm 5 imposes $|\mathbb{F}| \geq s^3$, Fact 2.4 imposes $|\mathbb{F}| \geq d^7$ and for a random affine projection to be non-degenerate, $|\mathbb{F}| \geq d^{t+4} s^{1+4/\epsilon}$ is required which follows from the proof of Lemma 4.4. Unlike the ET algorithm, we cannot work over a field extension because the input itself is random.

Remark 4.2. Unlike ET for NW, the permutation and scaling matrices are not recovered as the motivation for learning random affine projections of multilinear design polynomials is to understand the expressive power of the affine projections of NW. Theorem 4 proves that when $n \geq d^{4+\epsilon}$, the random affine projections of NW are distinguishable from random polynomials.

4.3 Structure of the adjoint algebra

Lemma 4.1 states that $\mathrm{Adj}(\partial^2, U, V)$ is trivial[15] for a non-degenerate f. The idea is to leverage the fact that each T_i is an affine projection of a multilinear monomial. By condition 2, there exists an $A_i \in \mathrm{GL}(n, \mathbb{F})$, such that in $f(A_i\mathbf{x})$, $T_i(A_i\mathbf{x})$ is a multilinear monomial. By Lemma 3.2, the adjoint of $f(A_i\mathbf{x})$ and that of $f(\mathbf{x})$ are equal as sets of matrices with respect to appropriate bases of their respective derivative spaces.[16] The operators in the adjoint of $f(A_i\mathbf{x})$ are shown to be invariant on the derivative spaces of $T_i(A_i\mathbf{x})$ by using the fact that the derivative space of $T_i(A_i\mathbf{x})$ is the space of multilinear polynomials in d variables. The block diagonality then holds. Because of the direct sum structure, the block diagonality of the adjoint and the block diagonality of ∂^2 operators, it suffices to analyze the adjoint of individual T_i's. Since $T_i(A_i\mathbf{x})$ is a multilinear monomial, its adjoint is trivial and thus so is the adjoint of T_i.

Lemma 4.1. Let g and f be as defined in Section 4.1. If f is non-degenerate, then $\mathrm{Adj}(\partial^2, U, V)$ is block-diagonal and is also trivial (thus, also block equi-triangular).

4.4 Vector space decomposition

Algorithm 5 Vector space decomposition algorithm

Input: B.b.a to bases of spaces U and V.
Output: B.b.a to bases of $W_1, \ldots, W_s$ where $W_i = U_{\pi(i)}$ for some permutation π.

1. Compute a basis $D^{(1)}, \ldots, D^{(b)}$ of $\mathrm{Adj}(\partial^2, U, V)_1 = \{D \mid (D, E) \in \mathrm{Adj}(\partial^2, U, V)\}$.
2. Select $c_1, \ldots, c_b \in_r S \subseteq \mathbb{F}$, $|S| = s^3$ and let $D = c_1 D^{(1)} + \cdots + c_b D^{(b)}$.
3. Factorize the characteristic polynomial of D to obtain eigenvalues $\lambda_1, \ldots, \lambda_s$.
4. Compute $W_i := \mathrm{Ker}(D - \lambda_i I)$ for all $i \in [s]$ and output b.b.a to the bases of $W_1, \ldots, W_s$.

Analysis and time complexity: Algorithm 5 is similar to Algorithm 3 except it uses ∂^2 operators and computes the eigenspaces of D, instead of generalized eigenspaces. Thus, the analysis for Algorithm 5 is the same as for Algorithm 3. Lemmas 4.1 and 4.2 prove that algorithm 5 works with high probability. Proposition 4.3 gives the time complexity.

Lemma 4.2. D has s distinct eigenvalues with probability $\geq 1 - \frac{\binom{s}{2}}{|S|}$.

Lemma 4.3. Let $D \in \mathrm{Adj}(\partial^k, U, V)$ such that it has s distinct eigenvalues denoted $\lambda_1, \ldots, \lambda_s$. Then $\mathrm{Ker}(D - \lambda_i I) = U_{\pi(i)}$ for all $i \in [s]$, where π is some permutation on $[s]$.

Proposition 4.3. Algorithm 5 has a running time of $\mathrm{poly}(n, s, d^t)$.

Handling non-homogeneous polynomials and translation. The non-degeneracy conditions are the same for non-homogeneous multilinear (m, d, s, t) design polynomials and when $f(\mathbf{x}) = g(A\mathbf{x} + \mathbf{b})$ for $A \in_r \mathbb{F}^{m \times n}$ and $\mathbf{b} \in \mathbb{F}^m$. For the non-homogeneous case, if the degree of *all* monomials is $\geq 2k + 2$, then Lemmas 4.1 and 4.4 hold, and the analysis proceeds in the same way. Proposition 4.1 can also recover translations.

[15] This is a special case of being block equi-triangularizable.
[16] Lemma 3.2 holds more generally for any two polynomials equivalent by invertible linear transforms.

4.5 Random affine projections are non-degenerate

Lemma 4.4 states that a random affine projection f is non-degenerate with high probability. Showing f is non-degenerate reduces to showing certain matrices, with entries as the coefficients of the l_i's, are full rank with high probability when the entries are chosen randomly. The main technical challenge is in proving condition 1 because the g_i's share variables, thus T_i's share the l_i's. A two-stage random process is used to show the existence of an affine projection which satisfies condition 1. The Schwartz-Zippel lemma then implies that a random affine projection also satisfies condition 1 with high probability.

Lemma 4.4. Let g be a polynomial as defined in Section 4.1 and f be its random affine projection. For $k = t + \left\lfloor \frac{2\log(s)}{\log\left(\frac{\sqrt{n}}{d^2}\right)} \right\rfloor + 1$, f is non-degenerate with high probability.

5 Conclusion

In this work, we design an ET algorithm for general design polynomials (Theorem 1) and a learning algorithm for random affine projections of multilinear design polynomials (Theorem 4). As an application of the ET algorithm, we show that GI is polynomial-time equivalent to PE for design polynomials with all-one coefficients (Theorem 2). As another application, we show that Patarin's authentication scheme can be broken if it uses a higher degree sparse polynomial for key generation (Theorem 3). We also give an ET algorithm for the NW design polynomial using Theorem 1 (Theorem 5). Theorem 4 is a significant generalization of the main result in [KS19] that gave a learning algorithm for random affine projections of the sum-product polynomial, which is a special multilinear design polynomial.

Both the algorithms are based on the vector space decomposition framework of [KS19, GKS20]. This work's main technical contributions include analysing a non-trivial adjoint algebra associated with design polynomials and developing a VSD algorithm based on generalized eigenspaces. We end by listing some related questions:

1. **ET for sparse polynomial:** What is the complexity of ET for the class of sparse polynomials? That is, given black-box access to a polynomial f and a parameter s, what is the complexity of testing whether f is in the orbit of some s-sparse polynomial? The authors of [CGS23] showed that the shift equivalence problem for sparse polynomials (i.e., when $f = g(\mathbf{x} + \mathbf{b})$ for some sparse polynomial g and $\mathbf{b} \in \mathbb{F}^n$) is undecidable over $\mathbb{Z}$.

   (**A recent update:** In an ongoing work [BDSS24], Pulkit Sinha and the authors of this paper have resolved this question by showing that this problem is NP-hard.)

2. **Weakening $n \geq d^{4+\epsilon}$:** Can the condition $n \geq d^{4+\epsilon}$ in Theorem 4 be changed to $n \geq d^{\delta}$ for arbitrary $\delta > 0$? Doing so would give stronger evidence that NW is not VNP-complete.

3. **Efficient ET for NW:** Theorem 5 gives an ET for NW, but it is not a polynomial-time algorithm. Is there a polynomial-time ET algorithm for NW? Our ET algorithm is for general design polynomials; it is possible that analyzing the properties of the NW polynomial may yield an efficient ET algorithm specifically for NW.

Acknowledgments

We thank the reviewers for their valuable feedback, which has helped us improve the presentation of this work. In particular, we thank one of the reviewers who pointed us to appropriate citations for the adjoint algebra.

References

[Aar08] Scott Aaronson. Arithmetic natural proofs theory is sought. http://www.scottaaronson.com/blog/?p=336, 2008. 2

[ADKT15] Vikraman Arvind, Bireswar Das, Johannes Köbler, and Seinosuke Toda. Colored Hypergraph Isomorphism is Fixed Parameter Tractable. *Algorithmica*, 71(1):120–138, 2015. Conference version appeared in the proceedings of FSTTCS 2010. 13

[Ara11] Manuel Araújo. Classification of quadratic forms. https://www.math.tecnico.ulisboa.pt/~ggranja/manuel.pdf, 2011. 1

[AS05] Manindra Agrawal and Nitin Saxena. Automorphisms of finite rings and applications to complexity of problems. In *23rd Annual Symposium on Theoretical Aspects of Computer Science, STACS 2005*, pages 1–17, 2005. 1, 4

[AS06] Manindra Agrawal and Nitin Saxena. Equivalence of $\mathbb{F}$-algebras and cubic forms. In *23rd Annual Symposium on Theoretical Aspects of Computer Science, STACS 2006*, pages 115–126, 2006. 1

[Bab16] László Babai. Graph isomorphism in quasipolynomial time [extended abstract]. In Daniel Wichs and Yishay Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 684–697. ACM, 2016. 13, 14, 15, 36

[BDSS24] Omkar Baraskar, Agrim Dewan, Chandan Saha, and Pulkit Sinha. Testing equivalence to sparse polynomials is NP-hard. Manuscript in preparation, 2024. 18

[Ber70] Elwyn R Berlekamp. Factoring polynomials over large finite fields. *Mathematics of Computation*, 24:713–735, 1970. 2

[BGKS22] Vishwas Bhargava, Ankit Garg, Neeraj Kayal, and Chandan Saha. Learning generalized depth three arithmetic circuits in the non-degenerate case. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022. 6, 7, 8

[BMW17] Peter A. Brooksbank, Joshua Maglione, and James B. Wilson. A fast isomorphism test for groups whose lie algebra has genus 2. *Journal of Algebra, Volume 473, Pages 545-590, ISSN 0021-8693*, 2017. 9

[BRS17] Markus Bläser, B. V. Raghavendra Rao, and Jayalal Sarma. Testing Polynomial Equivalence by Scaling Matrices. In *Proceedings of 21st International Symposium on Fundamentals of Computation Theory (FCT), France*, volume 10472, pages 111–122, 2017. 15

[CGS23] Suryajith Chillara, Coral Grichener, and Amir Shpilka. On hardness of testing equivalence to sparse polynomials under shifts. In Petra Berenbrink, Patricia Bouyer, Anuj Dawar, and Mamadou Moustapha Kanté, editors, *40th International Symposium on Theoretical Aspects of Computer Science, STACS 2023, March 7-9, 2023, Hamburg, Germany*, volume 254 of *LIPIcs*, pages 22:1–22:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. 18

[CIK97] Alexander Chistov, Gábor Ivanyos, and Marek Karpinski. Polynomial time algorithms for modules over finite dimensional algebras. In *Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation*, ISSAC '97, page 68–74, New York, NY, USA, 1997. Association for Computing Machinery. 5, 6

[DL78] Richard A. DeMillo and Richard J. Lipton. A probabilistic remark on algebraic program testing. *Inf. Process. Lett.*, 7(4):193–195, 1978. 3

[GGKS19] Ankit Garg, Nikhil Gupta, Neeraj Kayal, and Chandan Saha. Determinant equivalence test over finite fields and over Q. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Greece*, volume 132 of *LIPIcs*, pages 62:1–62:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. 1, 7

[GKS20] Ankit Garg, Neeraj Kayal, and Chandan Saha. Learning sums of powers of low-degree polynomials in the non-degenerate case. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 889–899. IEEE, 2020. 5, 6, 7, 8, 9, 18

[GQ23a] Joshua A. Grochow and Youming Qiao. On the complexity of isomorphism problems for tensors, groups, and polynomials I: tensor isomorphism-completeness. *SIAM J. Comput.*, 52(2):568–617, 2023. Conference version appeared in the proceedings of ITCS 2021. 1

[GQ23b] Joshua A. Grochow and Youming Qiao. On the complexity of isomorphism problems for tensors, groups, and polynomials IV: linear-length reductions and their applications. *CoRR*, abs/2306.16317, 2023. 1

[Gro12] Joshua A. Grochow. *Symmetry and equivalence relations in classical and geometric complexity theory*. PhD thesis, University of Chicago, Chicago, IL, 2012. 7

[GS19] Nikhil Gupta and Chandan Saha. On the symmetries of and equivalence test for design polynomials. In *44th International Symposium on Mathematical Foundations of Computer Science, MFCS 2019, August 26-30, 2019, Aachen, Germany*, volume 138 of *LIPIcs*, pages 53:1–53:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. 3, 7, 15

[GST23] Nikhil Gupta, Chandan Saha, and Bhargav Thankey. Equivalence test for read-once arithmetic formulas. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 4205–4272. SIAM, 2023. 1, 7, 8

[Kay11] Neeraj Kayal. Efficient algorithms for some special cases of the polynomial equivalence problem. In Dana Randall, editor, *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 1409–1421. SIAM, 2011. 1, 4, 7, 9

[Kay12] Neeraj Kayal. Affine projections of polynomials: extended abstract. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 643–662, 2012. 1, 7

[KNS19] Neeraj Kayal, Vineet Nair, and Chandan Saha. Average-case linear matrix factorization and reconstruction of low width algebraic branching programs. *Comput. Complex.*, 28(4):749–828, 2019. 8, 22

[KNST19] Neeraj Kayal, Vineet Nair, Chandan Saha, and Sébastien Tavenas. Reconstruction of full rank algebraic branching programs. *ACM Trans. Comput. Theory*, 11(1):2:1–2:56, 2019. Conference version appeared in the proceedings of CCC 2017. 1, 7, 9

[KS19] Neeraj Kayal and Chandan Saha. Reconstruction of non-degenerate homogeneous depth three circuits. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 413–424. ACM, 2019. 2, 5, 7, 8, 18

[KSB] Krull-Schmidt Theorem. https://mathstrek.blog/2015/01/17/krull-schmidt-theorem/. 6

[KT90] Erich Kaltofen and Barry M. Trager. Computing with Polynomials Given By Black Boxes for Their Evaluations: Greatest Common Divisors, Factorization, Separation of Numerators and Denominators. *J. Symb. Comput.*, 9(3):301–320, 1990. Conference version appeared in the proceedings of FOCS 1988. 3, 9

[Lam04] T. Y. Lam. *Introduction To Quadratic Forms Over Fields*. American Mathematical Society, 2004. 1

[LLL82] Arjen K Lenstra, Hendrik W Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982. 2

[Mil79] Gary L. Miller. Graph isomorphism, general remarks. *J. Comput. Syst. Sci.*, 18(2):128–142, 1979. 13

[MNS20] Janaky Murthy, Vineet Nair, and Chandan Saha. Randomized Polynomial-Time Equivalence Between Determinant and Trace-IMM Equivalence Tests. In *45th International Symposium on Mathematical Foundations of Computer Science, MFCS 2020, August 24-28, 2020, Prague, Czech Republic*, volume 170 of *LIPIcs*, pages 72:1–72:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. 1, 7

[MS21] Dori Medini and Amir Shpilka. Hitting sets and reconstruction for dense orbits in VP_e and $\Sigma\Pi\Sigma$ circuits. In *36th Computational Complexity Conference, CCC 2021, July 20-23, 2021, Toronto, Ontario, Canada (Virtual Conference)*, volume 200 of *LIPIcs*, pages 19:1–19:27. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. 1, 7

[Pat96] Jacques Patarin. Hidden fields equations (HFE) and isomorphisms of polynomials (IP): two new families of asymmetric algorithms. In *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, pages 33–48, 1996. 1, 13

[Sax06] Nitin Saxena. *Morphisms of rings and applications to complexity*. PhD thesis, Indian Institute of Technology, Kanpur, 2006. 1

[Sch80] Jacob T. Schwartz. Fast Probabilistic Algorithms for Verification of Polynomial Identities. *J. ACM*, 27(4):701–717, 1980. 3

[Thi98] Thomas Thierauf. The isomorphism problem for read-once branching programs and arithmetic circuits. *Chicago J. Theor. Comput. Sci.*, 1998, 1998. 1

[Val79] Leslie G. Valiant. Completeness Classes in Algebra. In *Proceedings of the 11h Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1979, Atlanta, Georgia, USA*, pages 249–261, 1979. 2

[vzGG03] Joachim von zur Gathen and Jürgen Gerhard. *Modern computer algebra (2. ed.)*. Cambridge University Press, 2003. 3, 14

[Wil09] James B. Wilson. Decomposing p-groups via Jordan algebras. *Journal of Algebra, Volume 322, Issue 8, Pages 2642-2679, ISSN 0021-8693,*, 2009. 9

[Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In *Symbolic and Algebraic Computation, EUROSAM '79, An International Symposiumon Symbolic and Algebraic Computation, Marseille, France, June 1979, Proceedings*, pages 216–226, 1979. 3

A Missing proofs from Section 1

A.1 Proof of Corollary 1.1

Let $f(\mathbf{x})$ be a (n, d, t) design polynomial with $d \geq 3t$. As $f = f(I\mathbf{x})$, where I is the $n \times n$ identity matrix, the algorithm in Theorem 1 outputs $B = PSI = PS$ with high probability on input f. Now suppose that $f(\mathbf{x}) = f(A\mathbf{x})$ for some $A \in \mathrm{GL}(n, \mathbb{F})$. But then again, by Theorem 1, the output is P_1S_1A with high probability. As both the statements hold with high probability, there is a choice of randomness such that the algorithm outputs $B = PS = P_1S_1A$ for some permutation matrices P, P_1 and scaling matrices S, S_1. This implies that $A = (P_1S_1)^{-1}PS = P_2S_2$ for appropriate permutation matrix P_2 and scaling matrix S_2.

A.2 Proof of Corollary 1.2

Let f and g be (n, d, t) design polynomials with $d \geq 3t$. Let $f(\mathbf{x}) = g(A\mathbf{x})$, for some $A \in \mathrm{GL}(n, \mathbb{F})$. As $f = f(I\mathbf{x})$ is a design polynomial, the algorithm in Theorem 1 outputs $B = PS$ with high probability on input f. Also, since $f(\mathbf{x}) = g(A\mathbf{x})$, the algorithm outputs P_1S_1A with high probability. As both statements hold with high probability, there is a choice of randomness such that the algorithm outputs $B = PS = P_1S_1A$ for some permutation matrices P, P_1 and scaling matrices S, S_1. This implies that $A = (P_1S_1)^{-1}PS = P_2S_2$ for appropriate permutation matrix P_2 and scaling matrix S_2.

A.3 Affine projections of low width ABPs

Let $\mathsf{IMM}_{\mathbf{w},d}$ denote the iterated matrix multiplication polynomial of degree d and width vector $\mathbf{w} = (w_1, \ldots, w_{d-1})$. If $w_i \leq w$ for all $i \in [d-1]$, then we say that $\mathsf{IMM}_{\mathbf{w},d}$ has width bounded by w. Let m be the number of variables in $\mathsf{IMM}_{\mathbf{w},d}$. As mentioned in Section 1.2, the authors of [KNS19] gave an algorithm for learning random affine projections of low width $\mathsf{IMM}_{\mathbf{w},d}$ in the *uniform* width case, i.e., when $w_1 = \ldots = w_{d-1} = w \leq \sqrt{n}/2$. Their algorithm crucially relies on uniformity of the width (i.e., $w_1 = \ldots = w_{d-1}$), and the time complexity of the algorithm is dominated by $\mathrm{poly}(d^{w^3})$. However, to our knowledge, a learning algorithm for random affine projections of *non-uniform* width $\mathsf{IMM}_{\mathbf{w},d}$ is not known, even in the low-width case.

We show that for any $d > 32$, we can choose k, $n \leq m$, and w such that the n-variate affine projections of $\mathsf{IMM}_{\mathbf{w},d}$ have low dimensional k-th order partial derivative spaces, and the technical assumption $n \geq d^4$ (as in Theorem 4) is satisfied. But, unlike Theorem 4, we do not know how to leverage this "low dimensional derivatives space" property to design a learning algorithm for random affine projections of non-uniform width $\mathsf{IMM}_{\mathbf{w},d}$, even if $w \leq \sqrt{n}/2$.

Proposition A.1. Let $d > 32$, $d^{0.6} \leq k < d$, $n \in \mathbb{N}$ such that $n \geq d^5$.[17] Let $w = (1+\epsilon)\sqrt{\frac{n}{d}}$

[17] The lower bound of $d^{0.6}$ on k has been stated for simplicity, $k > d^{0.5+\delta}$ for $\delta > 0$ also suffices.

where $\epsilon = \frac{k}{5d}$.[18] For this choice of parameters, there exists $\mathsf{IMM}_{\mathbf{w},d}$ with width bounded by w and with $m = w_1 + w_{d-1} + \sum_{i=1}^{d-2} w_i w_{i+1}$ variables, where $2w + w^2(d-2) \geq m > n$, such that the following hold:

(i) $2w + w^2(d-2) \gg n$.

(ii) The k-th order partial derivative space of an n-variate affine projection of $\mathsf{IMM}_{\mathbf{w},d}$ is of low dimension. That is,

$$w^{2k}\binom{d}{k} \ll \binom{n+k-1}{k}.$$

Proof. (i) We show that the gap between $2w + w^2(d-2)$ and n is super constant. Note that,

$$2w + w^2(d-2) > w^2(d-2) = (1+\epsilon)^2(d-2)\frac{n}{d} > (1+2\epsilon)(1-\frac{2}{d})n.$$

Further, as $d > 32$ and $\epsilon = \frac{k}{5d}$, we have

$$(1+2\epsilon)(1-\frac{2}{d})n > (1+\frac{\epsilon}{4})n = \left(1+\frac{k}{20d}\right)n.$$

Thus,

$$2w + w^2(d-2) - n > \frac{kn}{20d} \geq \frac{n}{20d^{0.4}}$$

showing a super constant gap.

(ii) Let f be an n-variate affine projection of $\mathsf{IMM}_{\mathbf{w},d}$. Using the chain rule of differentiation and the fact that $\langle \partial^k f \rangle$ is spanned by the order k derivatives of $\mathsf{IMM}_{\mathbf{w},d}$ with variables substituted by the affine forms of the affine projection map, we get that

$$\dim\langle \partial^k f \rangle \leq w^{2k}\binom{d}{k}.$$

For any n-variate degree d polynomial f, $\dim\langle \partial^k f \rangle \leq \binom{n+k-1}{k}$. As $\binom{n+k-1}{k} \geq \binom{n}{k}$ for $k > 0$,

$$w^{2k}\binom{d}{k} \ll \binom{n}{k} \implies w^{2k}\binom{d}{k} \ll \binom{n+k-1}{k}.$$

Now,

$$w^{2k}\binom{d}{k} \ll \binom{n}{k} \iff w^{2k} \ll \prod_{i=0}^{k-1}\frac{n-i}{d-i} \iff (1+\epsilon)^{2k}(\frac{n}{d})^k \ll \prod_{i=0}^{k-1}\frac{n-i}{d-i}$$

$$\iff (1+\epsilon)^{2k} \ll \prod_{i=1}^{k-1}\frac{1-i/n}{1-i/d}. \tag{4}$$

We lower bound the RHS of inequality (4) as

[18] Ceiling and floor notations have been omitted for simplicity.

$$\prod_{i=1}^{k-1} \frac{1-i/n}{1-i/d} \geq \prod_{i=1}^{k-1} \frac{e^{\frac{-2i}{n}}}{e^{\frac{-i}{d}}}.$$

If this lower bound is much larger than the LHS of inequality (4), then inequality (4) holds. Thus, we need

$$(1+\epsilon)^{2k} \ll \prod_{i=1}^{k-1} \frac{e^{\frac{-2i}{n}}}{e^{\frac{-i}{d}}} \iff (1+\epsilon)^{2k} \ll \prod_{i=1}^{k-1} e^{\frac{-2i}{n}+\frac{i}{d}} \iff (1+\epsilon)^{k} \ll e^{\frac{k(k-1)}{4}(\frac{1}{d}-\frac{2}{n})}.$$

Note $(1+\epsilon)^k = (1+\frac{k}{5d})^k < e^{\frac{k^2}{5d}}$ and $\frac{1}{d} - \frac{2}{n} > \frac{7}{8d}$ for $d > 2$ and $n \geq d^5$. Hence if

$$e^{\frac{k^2}{5d}} \ll e^{\frac{7k(k-1)}{32d}}$$

then inequality (4) holds. Dividing $e^{\frac{7k(k-1)}{32d}}$ by $e^{\frac{k^2}{5d}}$ shows,

$$e^{\frac{7k(k-1)}{32d} - \frac{k^2}{5d}} = e^{\frac{3k^2-35k}{160d}}.$$

As $k \geq d^{0.6}$, $e^{\frac{7k(k-1)}{32d}} > e^{\frac{k^2}{5d}}$ by an exponential factor in $d^{0.2}$, implying inequality (4). Thus,

$$w^{2k}\binom{d}{k} \ll \binom{n+k-1}{k}.$$

□

The condition $2w + w^2(d-2) \gg n$ for the given choice of w, n and k establishes that the gap between the maximum possible variables in $\mathsf{IMM}_{\mathbf{w},d}$ and n is large enough such that an $\mathsf{IMM}_{\mathbf{w},d}$ with non-uniform width exists. An ABP with non-uniform width can be viewed as an ABP with uniform width where nodes have been removed from the layers of the ABP. Each removal of a node results in a reduction of at most $2w$ many variables. We need to ensure that the removal of the nodes preserves the condition $m > n$, as we are considering n-variate affine projections of an m-variate polynomial.

As proven, $2w + w^2(d-2) > n$ by $\frac{kn}{20d} \geq \frac{n}{20d^{0.4}}$. Note that,

$$\frac{1}{2w}\frac{kn}{20d} \geq \frac{\sqrt{n}d^{0.1}}{40(1+\epsilon)}.$$

Thus, the gap between $2w + w^2(d-2)$ and n is larger than the maximum possible loss in variables incurred due to the removal of a node by a super-constant factor. This shows that an $\mathsf{IMM}_{\mathbf{w},d}$ with non-uniform width and with width bounded by w and having m variables, where $w^2(d-2) + 2w \geq m > n$, exists for the given choice of w, n and k. So, Proposition A.1 applies to such an $\mathsf{IMM}_{\mathbf{w},d}$.

B Missing proofs from Section 3

B.1 Proof of Lemma 3.1

The linearity of ∂^t operator implies $U' \subseteq U'_1 + U'_2 \cdots + U'_s$. To show the direct sum and equality, it suffices to show $U'_i \subseteq U'$ and $U'_1 + \cdots + U'_s = U'_1 \oplus \cdots \oplus U'_s$. A monomial basis $\mathscr{B}'_i$ for U'_i can be formed from $\{\partial^t g_i\}$, where g_i is a monomial of the design polynomial g.

Since $\deg \gcd(g_i, g_j) < t$ for $i \neq j$, for every $\mathbf{x}^{\boldsymbol{\alpha}}$ of degree t such that $\frac{\partial^t g_i}{\partial \mathbf{x}^{\boldsymbol{\alpha}}} \neq 0$, $\frac{\partial^t g_j}{\partial \mathbf{x}^{\boldsymbol{\alpha}}} = 0$ for all $j \neq i$. For every such $\mathbf{x}^{\boldsymbol{\alpha}}$,

$$\frac{\partial^t g}{\partial \mathbf{x}^{\boldsymbol{\alpha}}} = \sum_{j=1}^{s} \frac{\partial^t g_j}{\partial \mathbf{x}^{\boldsymbol{\alpha}}} = \frac{\partial^t g_i}{\partial \mathbf{x}^{\boldsymbol{\alpha}}}.$$

Thus, $\{\partial^t g_i\} \subseteq \{\partial^t g\}$ which implies $U_i' \subseteq U'$. The condition $d \geq 3t \geq 2t$ ensures that for any $i \neq j$, there do not exist $\mathbf{x}^{\boldsymbol{\alpha}}, \mathbf{x}^{\boldsymbol{\beta}}$ such that $\frac{\partial^t g_i}{\partial \mathbf{x}^{\boldsymbol{\alpha}}} \neq 0$, $\frac{\partial^t g_j}{\partial \mathbf{x}^{\boldsymbol{\beta}}} \neq 0$ and $\frac{\partial^t g_i}{\partial \mathbf{x}^{\boldsymbol{\alpha}}} = \frac{\partial^t g_j}{\partial \mathbf{x}^{\boldsymbol{\beta}}}$ (up to scalar multiples), as the contrary implies that $\deg \gcd(g_i, g_j) \geq t$. This shows that $U_1' + \cdots + U_s' = U_1' \oplus \cdots \oplus U_s'$. A similar argument shows $V' = V_1' \oplus V_2' \cdots \oplus V_s'$ as $d \geq 3t$.

B.2 Proof of Lemma 3.2

1. If $f = g(A\mathbf{x})$, then for $W_k = \langle \partial^k f \rangle$ and $W_k' = \langle \partial^k g \rangle$, we have $W_k' \cong W_k$ via the linear map ϕ. Indeed, from the chain rule for derivatives and the invertibility of A it follows that $\{p_1, \ldots, p_l\}$ is a basis of W_k' if and only if $\{\phi(p_1), \ldots, \phi(p_l)\}$ is a basis of W_k.

2. In this proof, whenever we say 'basis', we will mean 'ordered basis'. Let $\mathscr{B}_1' = (p_1, p_2 \ldots p_l)$ be a basis of U' and $\mathscr{B}_2' = (q_1, q_2 \ldots q_m)$ be a basis of V'. Since $U' \cong U$ and $V' \cong V$, $\mathscr{B}_1 := \phi(\mathscr{B}_1') = (p_1(A\mathbf{x}), p_2(A\mathbf{x}) \ldots p_l(A\mathbf{x}))$ is a basis of U and $\mathscr{B}_2 := \phi(\mathscr{B}_2') = (q_1(A\mathbf{x}), q_2(A\mathbf{x}) \ldots q_m(A\mathbf{x}))$ is a basis of V. Fixing $\mathscr{B}_1'$ and $\mathscr{B}_2'$ as bases of U' and V' respectively, the operators $\partial^t : U' \to V'$ and the elements of $\mathrm{Adj}(\partial^t, U', V')$ can be represented as matrices and tuples of matrices, respectively. Similarly, fixing $\mathscr{B}_1$ and $\mathscr{B}_2$ as bases of U and V respectively, the operators $\partial^t : U \to V$ and the elements of $\mathrm{Adj}(\partial^t, U, V)$ can be represented as matrices and tuples of matrices, respectively.

   With the bases fixed, the idea is to use the adjoint condition $LD = EL$, as stated in Section 1.2 for the adjoint, which gives rise to a system of linear equations with the entries of D and E being variables. Since $\dim(U) = \dim(U')$ and $\dim(V) = \dim(V')$, the number of variables in the system $\partial^t D = E\partial^t$, where $(D, E) \in \mathrm{Adj}(\partial^t, U, V)$, is the same as that of the system $\partial^t D' = E'\partial^t$, where $(D', E') \in \mathrm{Adj}(\partial^t, U', V')$. To infer that both these systems have the same solution space, showing that the matrix of every operator in $\partial^t : U \to V$, with respect to bases $\mathscr{B}_1$ and $\mathscr{B}_2$, is a linear combination of the matrices of the operators in $\partial^t : U' \to V'$, with respect to bases $\mathscr{B}_1'$ and $\mathscr{B}_2'$, suffices as the invertibility of A implies that the converse is also true.

   Suppose $\mathbf{x}^{\boldsymbol{\alpha}}$ is a degree t monomial and $L_{\boldsymbol{\alpha}}$ be the matrix corresponding to the operator $\frac{\partial^t}{\partial \mathbf{x}^{\boldsymbol{\alpha}}} : U \to V$ with respect to bases $\mathscr{B}_1$ and $\mathscr{B}_2$. Let

$$\frac{\partial^t p_i(A\mathbf{x})}{\partial \mathbf{x}^{\boldsymbol{\alpha}}} = \sum_{j=1}^{m} a_{j,i,\boldsymbol{\alpha}} q_j(A\mathbf{x}),$$

   where $(a_{1,i,\boldsymbol{\alpha}}, \ldots, a_{m,i,\boldsymbol{\alpha}})^T$ is the i-th column of $L_{\boldsymbol{\alpha}}$. From the chain rule of derivatives, it also holds that:

$$\frac{\partial^t p_i(A\mathbf{x})}{\partial \mathbf{x}^{\boldsymbol{\alpha}}} = \sum_{|\boldsymbol{\beta}|=t} c_{\boldsymbol{\alpha},\boldsymbol{\beta}} \frac{\partial^t p_i}{\partial \mathbf{x}^{\boldsymbol{\beta}}}(A\mathbf{x}), \quad \text{for } c_{\boldsymbol{\alpha},\boldsymbol{\beta}} \in \mathbb{F}.$$

   Further, since $\frac{\partial^t p_i}{\partial \mathbf{x}^{\boldsymbol{\beta}}} = \sum_{j=1}^{m} b_{j,i,\boldsymbol{\beta}} q_j$, where $(b_{1,i,\boldsymbol{\beta}}, \ldots, b_{m,i,\boldsymbol{\beta}})^T$ is the i-th column of $L'_{\boldsymbol{\beta}}$, the matrix of the operator $\frac{\partial^t}{\partial \mathbf{x}^{\boldsymbol{\beta}}} : U' \to V'$ with respect to the bases $\mathscr{B}_1'$ and $\mathscr{B}_2'$, we have $\frac{\partial^t p_i}{\partial \mathbf{x}^{\boldsymbol{\beta}}}(A\mathbf{x}) = \sum_{j=1}^{m} b_{j,i,\boldsymbol{\beta}} q_j(A\mathbf{x})$. Thus,

$$\frac{\partial^t p_i(A\mathbf{x})}{\partial \mathbf{x}^{\boldsymbol{\alpha}}} = \sum_{|\boldsymbol{\beta}|=t} c_{\boldsymbol{\alpha},\boldsymbol{\beta}} \sum_{j=1}^{m} b_{j,i,\boldsymbol{\beta}} q_j(A\mathbf{x}).$$

Hence, $a_{j,i,\boldsymbol{\alpha}} = \sum_{|\boldsymbol{\beta}|=t} c_{\boldsymbol{\alpha},\boldsymbol{\beta}} b_{j,i,\boldsymbol{\beta}}$. This implies $L_{\boldsymbol{\alpha}} = \sum_{|\boldsymbol{\beta}|=t} c_{\boldsymbol{\alpha},\boldsymbol{\beta}} L'_{\boldsymbol{\beta}}$.

B.3 Proof of Lemma 3.3

Let $(D', E') \in \mathrm{Adj}(\partial^t, U', V')$. Let $\mathscr{B}'_i$ be the basis of monomials for U'_i obtained from $\{\partial^t g_i\}$. Showing that $D'(b_i) \in U'_i$ for any $b_i \in \mathscr{B}'_i$ suffices to prove the lemma.

Let $D'(b_i) = u'_1 + \cdots + u'_s$ where $u'_j \in U'_j$ and is expressed in the basis $\mathscr{B}'_j$. Suppose $u'_j \neq 0$ for some $j \neq i$. By the design condition and the fact that $d \geq 3t$, $\deg \gcd(b_j, b_k) < t$ for any $b_j \in \mathscr{B}'_j$ and $b_k \in \mathscr{B}'_k$, where $j \neq k$. Thus, there exists a degree t monomial $\mathbf{x}^{\boldsymbol{\beta}}$ such that $\frac{\partial^t u'_j}{\partial \mathbf{x}^{\boldsymbol{\beta}}} \neq 0$ and $\frac{\partial^t b_i}{\partial \mathbf{x}^{\boldsymbol{\beta}}} = 0$ and $\frac{\partial^t u'_k}{\partial \mathbf{x}^{\boldsymbol{\beta}}} = 0$ for all $k \neq j$.

Using $\mathbf{x}^{\boldsymbol{\beta}}$ for the derivative map,

$$\frac{\partial^t D'(b_i)}{\partial \mathbf{x}^{\boldsymbol{\beta}}} = \frac{\partial^t u'_1}{\partial \mathbf{x}^{\boldsymbol{\beta}}} + \cdots + \frac{\partial^t u'_s}{\partial \mathbf{x}^{\boldsymbol{\beta}}}.$$

Because $\frac{\partial^t}{\partial \mathbf{x}^{\boldsymbol{\beta}}} D' = E' \frac{\partial^t}{\partial \mathbf{x}^{\boldsymbol{\beta}}}$,

$$E'\Big(\frac{\partial^t b_i}{\partial \mathbf{x}^{\boldsymbol{\beta}}}\Big) = \frac{\partial^t u'_j}{\partial \mathbf{x}^{\boldsymbol{\beta}}}$$

$$\implies E'(0) = \frac{\partial^t u'_j}{\partial \mathbf{x}^{\boldsymbol{\beta}}} \implies \frac{\partial^t u'_j}{\partial \mathbf{x}^{\boldsymbol{\beta}}} = 0$$

This contradicts $\frac{\partial^t u'_j}{\partial \mathbf{x}^{\boldsymbol{\beta}}} \neq 0$. Thus, $u'_j = 0$ for all $j \neq i$ implying $D'(b_i) \in U'_i$.

A monomial basis can be formed for V'_i from $\{\partial^{2t} g_i\}$. For any e from this basis, $e = \frac{\partial^t b_i}{\partial \mathbf{x}^{\boldsymbol{\alpha}}}$, for some $b_i \in \mathscr{B}'_i$ and $\mathbf{x}^{\boldsymbol{\alpha}}$ with $|\boldsymbol{\alpha}| = t$. Using this and the adjoint condition we get

$$E'(e) = E'\Big(\frac{\partial^t b_i}{\partial \mathbf{x}^{\boldsymbol{\alpha}}}\Big) = \frac{\partial^t D'(b_i)}{\partial \mathbf{x}^{\boldsymbol{\alpha}}}.$$

As $D'(U'_i) \subseteq U'_i$ and $\langle \partial^t U'_i \rangle = V'_i$, we have $E'(e) \in V'_i$. Repeating for all such e in the monomial basis of V'_i, $E'(V'_i) \subseteq V'_i$ holds.

B.4 Proof of Lemma 3.4

(i) Assume $\mathrm{mon}\{\partial^t(\frac{\partial^t g_i}{\partial \mathbf{x}^{\boldsymbol{\alpha}}})\} \not\subseteq \mathrm{mon}\{\partial^t(\frac{\partial^t g_i}{\partial \mathbf{x}^{\boldsymbol{\beta}}})\}$. Then there exists $\mathbf{x}^{\boldsymbol{\gamma}}$, where $|\boldsymbol{\gamma}| = t$, such that $\frac{\partial^t}{\partial \mathbf{x}^{\boldsymbol{\gamma}}}(\frac{\partial^t g_i}{\partial \mathbf{x}^{\boldsymbol{\alpha}}}) \neq 0$ and $\mathrm{mon}(\frac{\partial^t}{\partial \mathbf{x}^{\boldsymbol{\gamma}}}(\frac{\partial^t g_i}{\partial \mathbf{x}^{\boldsymbol{\alpha}}})) \notin \mathrm{mon}\{\partial^t(\frac{\partial^t g_i}{\partial \mathbf{x}^{\boldsymbol{\beta}}})\}$. Consider the following cases:

- Suppose $\frac{\partial^t}{\partial \mathbf{x}^{\boldsymbol{\gamma}}}(\frac{\partial^t g_i}{\partial \mathbf{x}^{\boldsymbol{\beta}}}) = 0$ then,

$$\frac{\partial^t}{\partial \mathbf{x}^{\boldsymbol{\gamma}}} D'_i\Big(\frac{\partial^t g_i}{\partial \mathbf{x}^{\boldsymbol{\beta}}}\Big) = E'_i\Big(\frac{\partial^t}{\partial \mathbf{x}^{\boldsymbol{\gamma}}}\Big(\frac{\partial^t g_i}{\partial \mathbf{x}^{\boldsymbol{\beta}}}\Big)\Big)$$

$$\implies \frac{\partial^t}{\partial \mathbf{x}^{\boldsymbol{\gamma}}} D'_i\Big(\frac{\partial^t g_i}{\partial \mathbf{x}^{\boldsymbol{\beta}}}\Big) = 0$$

$$\implies \frac{\partial^t}{\partial \mathbf{x}^\gamma}\Big(\sum c_\delta \frac{\partial^t g_i}{\partial \mathbf{x}^\delta}\Big) = 0,$$

where $c_\delta = D_i'[\mathbf{x}^\delta][\mathbf{x}^\beta]$. Hence, by the assumption $\frac{\partial^t}{\partial \mathbf{x}^\gamma}(\frac{\partial^t g_i}{\partial \mathbf{x}^\alpha}) \neq 0$, $D_i'[\mathbf{x}^\alpha][\mathbf{x}^\beta] = 0$.

- Suppose $\frac{\partial^t}{\partial \mathbf{x}^\gamma}\Big(\frac{\partial^t g_i}{\partial \mathbf{x}^\beta}\Big) \neq 0$. Using the adjoint condition, we have

$$\frac{\partial^t}{\partial \mathbf{x}^\gamma} D_i'\Big(\frac{\partial^t g_i}{\partial \mathbf{x}^\beta}\Big) = E_i'\Big(\frac{\partial^t}{\partial \mathbf{x}^\gamma}\Big(\frac{\partial^t g_i}{\partial \mathbf{x}^\beta}\Big)\Big) = \frac{\partial^t}{\partial \mathbf{x}^\beta} D_i'\Big(\frac{\partial^t g_i}{\partial \mathbf{x}^\gamma}\Big).$$

The coefficient of $\frac{\partial^t}{\partial \mathbf{x}^\gamma}(\frac{\partial^t g_i}{\partial \mathbf{x}^\alpha})$ in $\frac{\partial^t}{\partial \mathbf{x}^\gamma} D_i'\Big(\frac{\partial^t g_i}{\partial \mathbf{x}^\beta}\Big)$ is $D_i'[\mathbf{x}^\alpha][\mathbf{x}^\beta]$. In $\frac{\partial^t}{\partial \mathbf{x}^\beta} D_i'\Big(\frac{\partial^t g_i}{\partial \mathbf{x}^\gamma}\Big)$, the coefficient of $\mathrm{mon}(\frac{\partial^t}{\partial \mathbf{x}^\gamma}(\frac{\partial^t g_i}{\partial \mathbf{x}^\alpha}))$ must be 0 because the contrary would imply $\mathrm{mon}(\frac{\partial^t}{\partial \mathbf{x}^\gamma}(\frac{\partial^t g_i}{\partial \mathbf{x}^\alpha})) \in \mathrm{mon}\{\partial^t(\frac{\partial^t g_i}{\partial \mathbf{x}^\beta})\}$. By the equality above, $D_i'[\mathbf{x}^\alpha][\mathbf{x}^\beta] = 0$.

(ii) Let $\mathbf{x}^\delta := \mathrm{lcm}(\mathbf{x}^\alpha, \mathbf{x}^\beta)$. Since $\mathbf{x}^\alpha$ and $\mathbf{x}^\beta$ are degree t monomials, $t+1 \leq \deg(\mathbf{x}^\delta) \leq 2t$. Let $\mathbf{x}^\gamma := \mathbf{x}^\delta \mathbf{x}^{\delta'}$ such that $\deg(\mathbf{x}^\gamma) = 2t$. Clearly $\mathbf{x}^\gamma = \mathbf{x}^{\gamma_1}\mathbf{x}^\alpha = \mathbf{x}^{\gamma_2}\mathbf{x}^\beta$ for appropriate $\boldsymbol{\gamma_1}$ and $\boldsymbol{\gamma_2}$. Using the adjoint condition, we get

$$\frac{\partial^t}{\partial \mathbf{x}^{\gamma_1}} D_i'\Big(\frac{\partial^t g_i}{\partial \mathbf{x}^\alpha}\Big) = E_i'\Big(\frac{\partial^t}{\partial \mathbf{x}^{\gamma_1}}\Big(\frac{\partial^t g_i}{\partial \mathbf{x}^\alpha}\Big)\Big) = E_i'\Big(\frac{\partial^t}{\partial \mathbf{x}^{\gamma_2}}\Big(\frac{\partial^t g_i}{\partial \mathbf{x}^\beta}\Big)\Big) = \frac{\partial^t}{\partial \mathbf{x}^{\gamma_2}} D_i'\Big(\frac{\partial^t g_i}{\partial \mathbf{x}^\beta}\Big).$$

The coefficient of $\frac{\partial^t}{\partial \mathbf{x}^{\gamma_1}}(\frac{\partial^t g_i}{\partial \mathbf{x}^\alpha})$ in $\frac{\partial^t}{\partial \mathbf{x}^{\gamma_2}} D_i'\Big(\frac{\partial^t g_i}{\partial \mathbf{x}^\beta}\Big)$ is $D_i'[\mathbf{x}^\alpha][\mathbf{x}^\alpha]$ and that of $\frac{\partial^t}{\partial \mathbf{x}^{\gamma_2}}(\frac{\partial^t g_i}{\partial \mathbf{x}^\beta})$ in $\frac{\partial^t}{\partial \mathbf{x}^{\gamma_2}} D_i'\Big(\frac{\partial^t g_i}{\partial \mathbf{x}^\beta}\Big)$ is $D_i'[\mathbf{x}^\beta][\mathbf{x}^\beta]$. The equality above and $\mathbf{x}^\alpha\mathbf{x}^{\gamma_1} = \mathbf{x}^\beta\mathbf{x}^{\gamma_2}$ imply $D_i'[\mathbf{x}^\alpha][\mathbf{x}^\alpha] = D_i'[\mathbf{x}^\beta][\mathbf{x}^\beta]$.

B.5 Proof of Lemma 3.5

By the discussion in Section 3.3, there is a basis $\mathscr{B}$ with respect to which D is block equi-triangular. In this basis, since each block of D has equal diagonal entries and there are s blocks, showing that any two distinct blocks have different diagonal entries suffices. Let $\{D^{(1)}, D^{(2)} \ldots D^{(b)}\}$ be the elements of the basis computed for $\mathrm{Adj}(\partial^t, U, V)_1 := \{D \mid (D, E) \in \mathrm{Adj}(\partial^t, U, V)\}$. A random element of $\mathrm{Adj}(\partial^t, U, V)_1$ is of form $\sum_{j=1}^b c_j D^{(j)}$, where c_j's are chosen uniformly at random from $S \subseteq \mathbb{F}$. Note that:

- Let 1_{U_i} and 1_{U_i} be the projection operator to the spaces U_i and V_i, respectively. In the basis $\mathscr{B}$, the i-th block of 1_{U_i} is the identity matrix while the rest are all 0 blocks. Then, $(1_{U_i}, 1_{V_i}) \in \mathrm{Adj}(\partial^t, U, V)$ because for any $u \in U$ and any ∂^t operator, where $u = u_1 + \cdots + u_s$, $\partial^t(u) = v = v_1 + \cdots + v_s$, and $u_i \in U_i, v_i \in V_i$, we have

$$\partial^t(1_{U_i}(u)) = \partial^t(u_i) = v_i \text{ and } 1_{V_i}(\partial^t u) = 1_{V_i}(v) = v_i.$$

Therefore, $(\sum_{i=1}^s \lambda_i 1_{U_i}, \sum_{i=1}^s \lambda_i 1_{V_i})$ is an element of $\mathrm{Adj}(\partial^t, U, V)$, where $\lambda_i \neq \lambda_j$ for $i \neq j$. Clearly, $\sum_{i=1}^s \lambda_i 1_{U_i}$ is a block-diagonal matrix with each block being a scalar multiple of identity, hence it has s distinct eigenvalues.

- Let $i_1 \neq i_2$ and $D^{(j)}[i][i]$ be the diagonal entry of the i-th block of $D^{(j)}$. The difference of the diagonal entries of the i_1-th and i_2-th block is $\sum_{j=1}^b c_j(D^{(j)}[i_1][i_1] - D^{(j)}[i_2][i_2])$ which is a linear polynomial in c_j's. As 1_{U_i} is in the adjoint, this polynomial in c_j's is non-zero.

Applying the Schwartz-Zippel lemma on this polynomial, the i_1-th and i_2-th block have the same diagonal entries with probability $\leq \frac{1}{|S|}$. By applying union bound on all $\binom{s}{2}$ possible pairs of i_1 and i_2, the probability that some pair of blocks have equal diagonal entries is $\leq \frac{\binom{s}{2}}{|S|}$. Hence, D has s distinct eigenvalues with probability $\geq 1 - \frac{\binom{s}{2}}{|S|}$.

B.6 Proof of Lemma 3.6

By the discussion in Section 3.3, there exists a basis $\mathscr{B}$ where D is equi-triangular. We will prove the lemma with respect to this basis $\mathscr{B}$. With respect to $\mathscr{B}$, the diagonal entries of the i-th block (up to a permutation π on $[s]$) of $(D - \lambda_i I)$ are 0. It is easily seen that the i-th block of $(D - \lambda_i I)$ is a nilpotent[19] matrix, as all the diagonal entries are zero and the $(\dim U)$-th power of $(D - \lambda_i I)$ leads to the columns of the i-th block becoming 0. The other blocks in $(D - \lambda_i I)^{\dim U}$ have non-zero diagonal entries as $\lambda_1, \ldots, \lambda_s$ are distinct (by Lemma 3.5). Hence, $\mathrm{Ker}((D - \lambda_i I)^{\dim\ U}) = U_i$.

B.7 Proof of Lemma 3.7

Let $f = c_1 m_1 + \cdots + c_s m_s$ be formed as per Definition 1.4 where m_i are degree d monomials in $\mathbf{x}$ and c_i are arbitrary constants. For $i \neq j$, let $F_{i,j}$ be the event that $\deg \gcd(m_i, m_j) \geq t$.

For some fixed i and j, $F_{i,j}$ happens if and only if there exists a monomial m of degree t such that $m | m_i$ and $m | m_j$, where $|$ denotes monomial division. Thus, bounding the probability of the latter event bounds that of $F_{i,j}$.

By choosing d variables uniformly at random with repetition, m_i can be formed in n^d ways. Suppose, a particular $m = \prod_{i=1}^{n} x_i^{\alpha_i}$ of degree t divides m_i. Such an m_i can be formed by first selecting the variables of m, which can be done in $\binom{d}{t}\binom{t}{\alpha_1,\ldots,\alpha_n}$ and then choosing the rest of the variables of m_i in n^{d-t} ways. Thus,

$$\Pr[m | m_i] \leq \frac{\binom{d}{t}\binom{t}{\alpha_1,\ldots,\alpha_n} n^{d-t}}{n^d} = \frac{\binom{d}{t}\binom{t}{\alpha_1,\ldots,\alpha_n}}{n^t},$$

where $\binom{t}{\alpha_1,\ldots,\alpha_n} \leq t!$ is the multinomial coefficient corresponding to m. Since m_i and m_j are selected independently of one another,

$$\Pr[m | m_i \text{ and } m | m_j] \leq \frac{\binom{d}{t}^2 \binom{t}{\alpha_1,\ldots,\alpha_n}^2}{n^{2t}}.$$

Applying union bound on all $\binom{n+t-1}{t}$ degree t monomials, we have

$$\Pr[F_{i,j}] \leq \sum_{\sum \alpha_i = t} \frac{\binom{d}{t}^2 \binom{t}{\alpha_1 \cdots \alpha_n}^2}{n^{2t}}.$$

Using union bound on the $\binom{s}{2}$ possible pairs of monomials we get:

$$\begin{aligned}\Pr[\exists\, i, j \in [s] : F_{i,j}] &\leq \sum_{1 \leq i < j \leq s} \sum_{\sum \alpha_i = t} \frac{\binom{d}{t}^2 \binom{t}{\alpha_1 \cdots \alpha_n}^2}{n^{2t}} \\ &= \binom{s}{2} \sum_{\sum \alpha_i = t} \frac{\binom{d}{t}^2 \binom{t}{\alpha_1 \cdots \alpha_n}^2}{n^{2t}}\end{aligned}$$

[19] A linear operator T is nilpotent if T^k is the zero operator for some positive integer k.

$$
\begin{aligned}
&\leq s^2 \frac{\binom{d}{t}^2}{n^{2t}} \sum_{\sum \alpha_i = t} \binom{t}{\alpha_1 \cdots \alpha_n}^2 \\
&\leq s^2 \frac{\binom{d}{t}^2}{n^{2t}} \sum_{\sum \alpha_i = t} (t!)^2 = s^2 \frac{\binom{d}{t}^2 (t!)^2 \binom{n+t-1}{t}}{n^{2t}} \\
&\leq s^2 \frac{d^{2t}\binom{n+t-1}{t}}{n^{2t}} = s^2 \frac{d^{2t}}{n^{2t}} \prod_{j=1}^{t} \left(\frac{n+j-1}{j} \right).
\end{aligned}
$$

As $n \geq 1$ and $\frac{n+j}{j+1} \leq \frac{n+j-1}{j}$ for all $j \geq 1$, therefore $\frac{n+j-1}{j} \leq n$. Thus

$$
s^2 \frac{d^{2t}}{n^{2t}} \prod_{j=1}^{t} \left(\frac{n+j-1}{j} \right) \leq s^2 \frac{d^{2t} n^t}{n^{2t}} = \frac{s^2 d^{2t}}{n^t}.
$$

By assumption $s \leq \sqrt{\epsilon} \left(\frac{n}{d^2} \right)^{t/2}$. Thus,

$$
s^2 \frac{d^{2t}}{n^t} \leq \epsilon.
$$

Hence with probability $\geq 1 - \epsilon$, f is a (n, d, s, t) design polynomial.

B.8 Proof of Proposition 3.1

Assume that black-box access to the T_i's is available. Fact 2.4 gives[20] black-box access to the irreducible factors of T_i up to scaling by field elements. Since T_i is in the orbit of a monomial, it must factor into a product of linear forms. From black-box access to the factors of T_i, the linear forms can be recovered by querying the black-box at the set of points $\{e_i : i \in [n]\}$, where e_i is the point with i-th coordinate 1 and the rest as 0. Normalize the coefficients of all the linear forms across all the black-boxes of the T_i's for all $i \in [s]$. Thus, per T_i we get a product of linear forms Q_i which is multiplied by some field element due to the normalization process.

Now, assign to each linear form an $\mathbf{x}$-variable while maintaining the consistency of assignment across T_i's, to get a transform B. Since it is not known which variable is mapped to a particular linear form in the original transformation A, the recovered linear forms are assigned to a permutation of the variables. Thus, the transform B is related to A as $B = PSA$, for some permutation matrix P and scaling matrix S, where the scaling is due to the normalization.

Finally, for each T_i, form a monomial h_i by replacing each linear form of Q_i with the $\mathbf{x}$ variable corresponding to it in the transform B. This forms the polynomial $h = f(B^{-1}\mathbf{x}) = c_1 h_1 + \cdots + c_s h_s$, where the c_i are constants resulting from normalization of the linear forms. The polynomial $h = g((PS)^{-1}\mathbf{x})$ as well, and so it is a (n, d, s, t) design polynomial.

For the time complexity, Fact 2.4 is executed s times, and each invocation requires time $\text{poly}(n, d)$. The linear forms can be recovered and normalized in $\text{poly}(n, d, s)$ time. Similarly, h can be formed in $\text{poly}(n, d, s)$ time. Hence, the recovery process takes $\text{poly}(n, d, s)$ time.

When the input is $f = g(A\mathbf{x} + \mathbf{b})$, the recovery of the translation vector $\mathbf{b}$ is similar, as the returned black-box factors are affine forms and can be queried at all 0's to recover $\mathbf{b}$, again up to scaling. Thus, the recovered transform $B = PSA$ and the translation vector is $\mathbf{c} = PS\mathbf{b}$.

[20] This assumes that there is an efficient univariate polynomial factorization algorithm over $\mathbb{F}$.

B.9 Proof of Proposition 3.2

The time complexity of the algorithm is dominated by Steps 1 and 2. Step 1 uses Fact 2.1 to compute black-box access to bases of U and V. The precise time complexity of Fact 2.1 for a single invocation is $\text{poly}(n, d^l)$ time, where l is the number of variables in the monomial $\mathbf{x}^{\boldsymbol{\alpha}}$, with respect to which differentiation is done. Note $l \leq \min(t, n)$. Since there are $\binom{n+t-1}{t}$ many order t derivatives, this step needs $\text{poly}(\binom{n+t}{n} d^{\min(t,\, n)})$ time to execute. The complexity of Step 2 is $\text{poly}(\binom{n+t}{n} d^t)$ as proven in Proposition 3.4.

Using Fact 2.3, Step 3 can be executed. There are $O(sd^t)$ many linearly independent polynomials (since $\dim U \leq s\binom{d}{t} = O(sd^t)$) of degree $d - t$ and $\binom{n+t-1}{t}$ many derivatives of degree t; hence this step requires $\text{poly}(s, \binom{n+t}{t} d^t)$ time. The running time of the recovery procedure is $\text{poly}(n, d, s)$ by Proposition 3.1.

Thus, the overall complexity of the algorithm is $\text{poly}(s\binom{n+t}{n} d^t)$. Note, $s < \binom{n+t}{n}$ for (n, d, s, t) design polynomials[21]. Hence, the running time is $\text{poly}(\binom{n+t}{n} d^t)$.

B.10 Monomial with non-trivial adjoint

Proposition B.1. For $g = x_1^6 x_2^2$, $\text{Adj}(\partial^2, U', V')$ is non-trivial, where $U' = \langle \partial^2 g \rangle$ and $V' = \langle \partial^4 g \rangle$.

Proof. Consider the monomial $g = x_1^6 x_2^2$. We show for $\mathscr{L}_1 = \mathscr{L}_2 = \partial^2$, $\text{Adj}(\partial^2, U', V')$ is non-trivial by solving the linear system arising from $\partial^2 D' = E' \partial^2$, where $(D', E') \in \text{Adj}(\partial^2, U', V')$.

A basis for U' is $\{x_1^6, x_1^5 x_2, x_1^4 x_2^2\}$ and that for V' is $\{x_1^4, x_1^3 x_2, x_1^2 x_2^2\}$. Any operator $D' : U' \to U'$ can be represented as a 3×3 matrix:

$$\begin{pmatrix} d_{11} & d_{12} & d_{13} \\ d_{21} & d_{22} & d_{23} \\ d_{31} & d_{32} & d_{33} \end{pmatrix}.$$

Similarly, for $E' : V' \to V'$, the representation is:

$$\begin{pmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{pmatrix}.$$

The three possible order 2 derivative maps in x_1 and x_2 are $\frac{\partial^2}{\partial x_1^2}$, $\frac{\partial^2}{\partial x_1 x_2}$ and $\frac{\partial^2}{\partial x_2^2}$, which form a basis of $\langle \partial^2 \rangle$. The matrix representation of these maps (call them L_1, L_2 and L_3, respectively) with respect to these bases is as:

$$L_1 = \begin{pmatrix} 30 & 0 & 0 \\ 0 & 20 & 0 \\ 0 & 0 & 12 \end{pmatrix},$$

$$L_2 = \begin{pmatrix} 0 & 5 & 0 \\ 0 & 0 & 8 \\ 0 & 0 & 0 \end{pmatrix},$$

$$L_3 = \begin{pmatrix} 0 & 0 & 2 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

[21] This can be proved via a counting argument: a degree t monomial can divide at most one monomial of a (n, d, s, t) design polynomial and there are $\binom{n+t-1}{t}$ degree t monomials.

Applying the adjoint condition on these operators gives the following equations:

$$L_1 D = E L_1 \implies \begin{pmatrix} 30d_{11} & 30d_{12} & 30d_{13} \\ 20d_{21} & 20d_{22} & 20d_{23} \\ 12d_{31} & 12d_{32} & 12d_{33} \end{pmatrix} = \begin{pmatrix} 30e_{11} & 20e_{12} & 12e_{13} \\ 30e_{21} & 20e_{22} & 12e_{23} \\ 30e_{31} & 20e_{32} & 12e_{33} \end{pmatrix}$$

$$L_2 D = E L_2 \implies \begin{pmatrix} 5d_{21} & 5d_{22} & 5d_{23} \\ 8d_{31} & 8d_{32} & 8d_{33} \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 5e_{11} & 8e_{12} \\ 0 & 5e_{21} & 8e_{22} \\ 0 & 5e_{31} & 8e_{32} \end{pmatrix}$$

$$L_3 D = E L_3 \implies \begin{pmatrix} 2d_{31} & 2d_{32} & 2d_{33} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 2e_{11} \\ 0 & 0 & 2e_{21} \\ 0 & 0 & 2e_{31} \end{pmatrix}.$$

From these equations, we get that $d_{11} = d_{22} = d_{33} = e_{11} = e_{22} = e_{33}$ and d_{21}, d_{31}, d_{32}, e_{21}, e_{31} and e_{32} are all 0. The remaining equations are $5d_{23} = 8e_{12}$, $20d_{23} = 12e_{23}$, $30d_{12} = 20e_{12}$ and $30d_{13} = 12e_{13}$. It can be easily checked that the resulting adjoint operators look like:

$$D = \begin{pmatrix} d_{11} & d_{12} & d_{13} \\ 0 & d_{11} & \frac{12}{5}d_{12} \\ 0 & 0 & d_{11} \end{pmatrix} E = \begin{pmatrix} d_{11} & \frac{3}{2}d_{12} & \frac{5}{2}d_{13} \\ 0 & d_{11} & 4d_{12} \\ 0 & 0 & d_{11} \end{pmatrix}.$$

□

B.11 Proof of Proposition 3.3

To show that D' is equi-triangular, it is enough to show that D'_i is equi-triangular. We do this by re-ordering the elements of $\mathscr{B}'_i = \{\frac{\partial^t g_i}{\partial \mathbf{x}^\alpha} : |\alpha| = t \text{ and } \frac{\partial^t g_i}{\partial \mathbf{x}^\alpha} \neq 0\}$ (note that this is a basis of U'_i). Consider a directed graph $G = (V, E)$ where $V = \mathscr{B}'_i$ and $E = \{(\frac{\partial^t g_i}{\partial \mathbf{x}^\alpha}, \frac{\partial^t g_i}{\partial \mathbf{x}^\beta}) : \text{mon}\{\partial^t(\frac{\partial^t g_i}{\partial \mathbf{x}^\alpha})\} \subseteq \text{mon}\{\partial^t(\frac{\partial^t g_i}{\partial \mathbf{x}^\beta})\}\}$. The graph G is acyclic for if G had a cycle $\frac{\partial^t g_i}{\partial \mathbf{x}^{\alpha_1}}, \ldots, \frac{\partial^t g_i}{\partial \mathbf{x}^{\alpha_k}}$, then by the definition of the edge set, this is equivalent to saying that $\text{mon}\{\partial^t(\frac{\partial^t g}{\partial \mathbf{x}^{\alpha_1}})\} \subseteq \text{mon}\{\partial^t(\frac{\partial^t g}{\partial \mathbf{x}^{\alpha_k}})\} \subseteq \text{mon}\{\partial^t(\frac{\partial^t g}{\partial \mathbf{x}^{\alpha_1}})\}$ which implies $\text{mon}\{\partial^t(\frac{\partial^t g}{\partial \mathbf{x}^{\alpha_1}})\} = \text{mon}\{\partial^t(\frac{\partial^t g}{\partial \mathbf{x}^{\alpha_k}})\}$ – a contradiction as $\mathbf{x}^{\alpha_1}$ and $\mathbf{x}^{\alpha_k}$ are distinct monomials and $d \geq 3t$.

Let Top(.) denote the topological sort of a graph. Define $\text{Top}_i := \text{Top}(\mathscr{B}'_i, E_i) = (\mathbf{x}^{\alpha_1}, \ldots, \mathbf{x}^{\alpha_b})$, where $b = \dim\ U'_i$. Observe that if $j > k$ then $D'_i[\mathbf{x}^{\alpha_j}][\mathbf{x}^{\alpha_k}] = 0$ by Lemma 3.4. This is because of the fact that $D'_i[\mathbf{x}^{\alpha_j}][\mathbf{x}^{\alpha_k}] \neq 0$ would imply $\text{mon}\{\partial^t(\frac{\partial^t g_i}{\partial \mathbf{x}^{\alpha_j}})\} \subseteq \text{mon}\{\partial^t(\frac{\partial^t g_i}{\partial \mathbf{x}^{\alpha_k}})\}$ which implies that there is an edge from $\frac{\partial^t g_i}{\partial \mathbf{x}^{\alpha_j}}$ to $\frac{\partial^t g_i}{\partial \mathbf{x}^{\alpha_k}}$ when $j > k$ contradicting that Top_i is a topological sort.

A basis $\mathscr{B}'$ of U' can be formed by concatenating all the re-ordered $\mathscr{B}'_i$ bases. Note that $D'_i[\mathbf{x}^{\alpha_j}][\mathbf{x}^{\alpha_k}] = 0$ for $j > k$. Thus, D' in the basis $\mathscr{B}'$ is a block diagonal matrix with each block being upper-triangular with all diagonal entries equal. Hence, $\text{Adj}(\partial^t, U', V')$ is block equi-triangularizable.

B.12 Proof of Proposition 3.4

The dominant time complexity is that of step 1, which involves solving a system of linear equations in $\dim(U)^2 + \dim(V)^2$ variables with $\binom{n+t-1}{t} \cdot \dim(V) \cdot \dim(U)$ many equations. Since $\dim(U) \leq s\binom{d}{t} < sd^t$ and $\dim(V) \leq s\binom{d}{2t} < sd^{2t}$, there are $\binom{n+t-1}{t}\text{poly}(sd)^t$ linear equations and $\text{poly}(sd^t)$ variables in the system.[22] Thus, such a system can be solved in

[22] For a degree d multilinear monomial h, $\dim\langle \partial^t h \rangle = \binom{d}{t}$ and any non-multilinear monomial of degree d is a p-projection of a degree d multilinear monomial. This gives the upper bound on the dimensions of U and V.

poly$(s\binom{n+t-1}{t}d^t)$ time. The computation of the eigenvalues requires factoring a univariate polynomial of degree $< sd^t$, which can be done in (randomized) poly(sd^t) time, and computing the null spaces can also be done in poly(sd^t) time. Thus, the overall time complexity is poly$(s\binom{n+t-1}{t}d^t)$ = poly$(\binom{n+t-1}{t}d^t)$ as $s \leq \binom{n+t-1}{t}$ (a degree-t monomial in n variables can divide at most one monomial of an (n, d, s, t) polynomial).

B.13 Proof of Proposition 3.5

Suppose $G_1 \cong G_2$, then there is a permutation $\pi : V_1 \to V_2$ which also preserves edges. Clearly, π is also a permutation on $\mathbf{x}$, where $\pi(x_i) := x_{\pi(i)}$. Extend π to a function $\sigma : \mathbf{z} \to \mathbf{z}$ as:

1. $\sigma(x_i) := x_{\pi(i)}$.
2. $\sigma(y_{\boldsymbol{I}_1(i,j)}) := y_{\boldsymbol{I}_2(\pi(i),\pi(j))}$, where $(i, j) \in E_1$ and $(\pi(i), \pi(j)) \in E_2$.

Clearly, σ is a permutation on $\mathbf{z}$ as well as from M_1 to M_2. If P is the permutation matrix corresponding to σ, then $h_2(\mathbf{z}) = h_1(P\mathbf{z})$. Hence, $h_1 \sim h_2$ via P.

Conversely, suppose $h_1 \sim h_2$. By Corollary 1.2, $h_2(\mathbf{z}) = h_1(PS\mathbf{z})$; the corollary can be applied here as $d = 6, t = 2$, and so, the $d \geq 3t$ condition is satisfied. Since the coefficient of every monomial of h_1 and h_2 is 1, we can assume S to be the identity matrix, thus $h_2(\mathbf{z}) = h_1(P\mathbf{z})$ and P induces a permutation from M_1 to M_2. Suppose P maps some $\mathbf{x}$ variable to some $\mathbf{y}$ variable. Since the degree of a $\mathbf{y}$ variable is 4 in any monomial of both polynomials, $h_1(P\mathbf{z})$ contains a monomial where the degree of some $\mathbf{x}$ variable is 4. This contradicts $h_1(P\mathbf{z}) = h_2(\mathbf{z})$. Thus, P must permute the $\mathbf{x}$ variables amongst themselves and $\mathbf{y}$ variables amongst themselves. Hence, P induces a permutation π from V_1 to V_2. Because P also induces a permutation from M_1 to M_2, which correspond to E_1 and E_2 respectively, therefore π is a permutation from V_1 to V_2 preserving edges. Hence, $G_1 \cong G_2$ via π.

B.14 Proof of Proposition 3.6

Let M_1 and M_2 be the set of monomials of h_1 and h_2 respectively, with $|M_1| = |M_2| = s$. Construct graph $G_1(V_1, E_1)$ for h_1, where initially $V_1 := \mathbf{x} \sqcup M_1$. If $a_{i,j}$ is the exponent of $x_i \in \mathbf{x}$ in $m_j \in M_1$, then introduce $a_{i,j}$ many vertices, $y_1, y_2 \ldots y_{a_{i,j}}$ and connect x_i and m_j to all these y-vertices. Do this for all x_i and m_j. G_1 is similar to a bipartite graph where edges exist only between $\mathbf{x}$ and M_1, except for an intermediary set of y vertices to make G_1 a simple graph. Finally, attach each $x_i \in \mathbf{x}$ to a complete graph on ns vertices and similarly attach each $m_j \in M_1$ to a complete graph on $ns + 1$ vertices. Construct $G_2(V_2, E_2)$ for h_2 similarly.

Suppose $h_2 \sim h_1$. By Corollary 1.2, $h_2(\mathbf{x}) = h_1(PS\mathbf{x})$. Since all coefficients are one, we can assume S to be the identity matrix, hence $h_2(\mathbf{x}) = h_1(P\mathbf{x})$. Now, P permutes the $\mathbf{x}$ variables and is also a permutation from M_1 to M_2. Construct the function $\pi : V_1 \to V_2$ as follows:

1. $\forall x_i \in \mathbf{x}, \pi(x_i) = x_j$, where P maps x_i to x_j.
2. $\forall m_i \in M_1, \pi(m_i) = m_j$, where P maps m_i to m_j.
3. π maps the y vertices and the gadget vertices while preserving edges accordingly.

Clearly, π is a permutation from V_1 to V_2 which preserves edges. Hence $G_1 \cong G_2$.

Now, suppose $G_1 \cong G_2$ via permutation $\pi : V_1 \to V_2$. Clearly, π must map a vertex from the variable vertices of G_1 to the variable vertices of G_2 because of the attached gadgets to these vertices. Similarly, π must map a vertex from M_1 to a vertex in M_2 due to the attached gadgets. Finally, π must also preserve the edge between a variable vertex x_i and a monomial vertex m_j,

including the intermediary y vertex. Thus, π describes a permutation P on the $\mathbf{x}$ variables such that P is also a permutation from M_1 to M_2. Hence, $h_1(P\mathbf{x}) = h_2(\mathbf{x})$ implying $h_1 \sim h_2$.

B.15 Probability bound on a permutation mapping monomials to monomials

Let π be a permutation on $[n]$. For a monomial $m_i = \prod_{i=1}^{n} x_i^{\alpha_i}$, let $\pi(m_i) := \prod_{i=1}^{n} x_{\pi(i)}^{\alpha_i}$. Let $E_{i,j}^{\pi}$ denote the event $m_j = \pi(m_i)$ for random degree d monomials m_i and m_j (as per Definition 1.4). Lemma B.1 bounds the probability that a non-trivial permutation π maps monomial m_i to m_j; Lemma B.2 bounds the probability that π fixes a monomial m_i.

Lemma B.1. For $i \neq j$, $\Pr[E_{i,j}^{\pi}] < (\frac{d}{n})^d$.

Proof. Once the variables for m_i are chosen, $\pi(m_i)$ determines a unique monomial. Let $m_i = \prod_{i=1}^{n} x_i^{\alpha_i}$, where $\sum_{i=1}^{n} \alpha_i = d$, then $\pi(m_i) = \prod_{i=1}^{n} x_{\pi(i)}^{\alpha_i}$. Thus, for $m_j = \pi(m_i)$ to happen, once m_i is chosen, the choices of variables for m_j are fixed by π, with only the order of the choices varying. The number of ways to choose the variables of m_j is then given by $\binom{d}{\alpha_1 \cdots \alpha_n}$. Hence

$$\Pr[E_{i,j}^{\pi}] = \frac{\binom{d}{\alpha_1 \cdots \alpha_n}}{n^d} \leq \frac{d!}{n^d} < \left(\frac{d}{n}\right)^d.$$

□

Lemma B.2. Let τ be the permutation on $[n]$ as: $\tau(1) = 2$, $\tau(2) = 1$ and $\tau(i) = i$ for $i \geq 3$. Let π be a non-trivial permutation on $[n]$. Then, for any $i \in [s]$, $\Pr[E_{i,i}^{\pi}] \leq \Pr[E_{i,i}^{\tau}] < e^{-d/n}$.

Proof. Let $\pi = \pi_1 \cdots \pi_k$ be the decomposition of π into cycles. Since π is a non-trivial permutation on $[n]$, there exists at least one cycle of length ≥ 2. Let π_q be a cycle of length $l \geq 2$. If $\pi(m_j) = m_j$ and if variable x_p is present in m_j and p is part of the cycle π_q, then all the variables corresponding to the elements of π_q must be present in m_j with the same multiplicities.

The event $E_{j,j}^{\pi}$ happens when all the elements of π_q are chosen i times, for some $0 \leq i \leq d/l$, and then the remaining choices are made from the rest of the variables, which are at most $(n-l)^{d-li}$ many. Thus,

$$\Pr[E_{j,j}^{\pi}] \leq \sum_{i=0}^{d/l} \frac{\binom{d}{li}\binom{li}{i \cdots i}(n-l)^{d-li}}{n^d} = \frac{d!(n-l)^d}{n^d} \left(\sum_{i=0}^{d/l} \frac{1}{(i!)^l (d-li)!(n-l)^{li}} \right),$$

where $\binom{li}{i \cdots i}$ is the multinomial coefficient denoting the number of arrangements of the elements of π_q, which are li in count. For the permutation τ,

$$\Pr[E_{j,j}^{\tau}] = \sum_{i=0}^{d/2} \frac{\binom{d}{2i}\binom{2i}{i}(n-2)^{d-2i}}{n^d} = \frac{d!(n-2)^d}{n^d} \left(\sum_{i=0}^{d/2} \frac{1}{(i!)^2 (d-2i)!(n-2)^{2i}} \right).$$

The first equality holds because τ fixes all variables other than x_1, x_2 and choosing any of these variables for m_j automatically means they are fixed by τ. For $2 < l \leq d$, $(n-l)^d < (n-2)^d$. Thus, showing that

$$\frac{1}{(i!)^l (d-li)!(n-l)^{li}} < \frac{1}{(i!)^2 (d-2i)!(n-2)^{2i}}$$

for all $1 \leq i \leq d/l$ suffices, as for $i = 0$ equality holds. Now,

$$\frac{1}{(i!)^l (d-li)!(n-l)^{li}} < \frac{1}{(i!)^2 (d-2i)!(n-2)^{2i}}$$

$$\iff (i!)^l(d-li)!(n-l)^{li} > (i!)^2(d-2i)!(n-2)^{2i}$$

$$\iff (i!)^{l-2}(n-l)^{li} > \frac{(d-2i)!}{(d-li)!}(n-2)^{2i}. \tag{5}$$

As $l \leq d$ and $n > d^2$,the left hand side of inequality (5) can be lower bounded as:

$$(i!)^{l-2}(n-l)^{li} > (n-d)^{li} > \left(\frac{n}{2}\right)^{li},$$

while the right hand side can be upper bounded as:

$$\frac{(d-2i)!}{(d-li)!}(n-2)^{2i} < d^{(l-2)i}n^{2i}.$$

Thus, if

$$n^{li} > 2^{li}d^{(l-2)i}n^{2i} \iff n^{l-2} > 2d^{l-2},$$

then we are done. Since $d \geq l \geq 3$ and $n > d^2$, the last inequality and, consequentially, inequality (5) holds implying that

$$\Pr[E^{\pi}_{j,j}] \leq \frac{d!(n-l)^d}{n^d}\Big(\sum_{i=0}^{d/l}\frac{1}{(i!)^l(d-li)!(n-l)^{li}}\Big)$$

$$\leq \frac{d!(n-2)^d}{n^d}\Big(\sum_{i=0}^{d/2}\frac{1}{(i!)^2(d-2i)!(n-2)^{2i}}\Big) = \Pr[E^{\tau}_{j,j}].$$

Now as $n > d^2$ (and also assuming $d \geq 3$),

$$\Pr[E^{\tau}_{j,j}] = \frac{d!(n-2)^d}{n^d}\Big(\sum_{i=0}^{d/2}\frac{1}{(i!)^2(d-2i)!(n-2)^{2i}}\Big)$$

$$< \frac{(n-2)^d}{n^d}\Big(1+\frac{d^2(d-1)}{2(n-2)^2}\Big) < \Big(1-\frac{2}{n}\Big)^d\Big(1+\frac{d^3}{n^2}\Big)$$

$$< e^{-2d/n}e^{d^3/n^2} < e^{-2d/n+d/n} = e^{-d/n}.$$

□

B.16 Proof of Proposition 3.7

Fix an arbitrary non-trivial permutation π on $[n]$ and let F^{π} denote the event that π is a permutation symmetry of f. Let $q = e^{-d/n}$ and $p = \left(\frac{d}{n}\right)^d$, q and p are the upper bounds on $\Pr[E^{\pi}_{i,i}]$ and $\Pr[E^{\pi}_{i,j}]$ from Lemma B.2 and Lemma B.1, respectively.

Without loss of generality, let $m_1, m_2 \ldots, m_i$ be i monomials which are permuted amongst themselves in cycles of length ≥ 2 by π, thus π does not fix any of these monomials. Let there be k cycles with lengths $l_1, l_2 \ldots, l_k$. In the ith cycle, m_{i_1} maps to m_{i_2}, m_{i_2} maps to $m_{i_3}, \cdots,$ $m_{i_{l_i-1}}$ maps to $m_{i_{l_i}}$ and $m_{i_{l_i}}$ maps to m_{i_1}, which is precisely the event $\bigcap_{j=1}^{l_i-1} E^{\pi}_{i_j,i_{j+1}} \bigcap E^{\pi}_{i_{l_i},i_1}$. The probability that a cycle of length l_i occurs, with some fixed ordering of the monomials, is:

$$\Pr\Big[\bigcap_{j=1}^{l_i-1} E^{\pi}_{i_j,i_{j+1}} \bigcap E^{\pi}_{i_{l_i},i_1}\Big] \leq \Pr\Big[\bigcap_{j=1}^{l_i-1} E^{\pi}_{i_j,i_{j+1}}\Big] = \prod_{j=1}^{l_i-1}\Pr\Big[E^{\pi}_{i_j,i_{j+1}}\Big] \leq p^{l_i-1},$$

where the equality holds because the events $E^{\pi}_{i_j,i_{j+1}}$ for $j = 1$ to $l_i - 1$ are independent as each monomial is sampled independently of the others.

Suppose π is a permutation symmetry of f. In that case, π induces a permutation on the s monomials, with i of the monomials lying in cycles of length ≥ 2 and the remaining $s - i$ monomials being fixed, where $i = 0$ or $2 \leq i \leq s$. Thus,

$$\Pr[F^{\pi}] \leq q^s + \sum_{i=2}^{s} \binom{s}{i} q^{s-i} \sum_{k=1}^{i/2} \sum_{\sum_j l_j = i, l_j \geq 2} \binom{i}{l_1 \cdots l_k} \prod_{j=1}^{k} ((l_j - 1)! p^{l_j - 1})$$

$$\leq q^s + \sum_{i=2}^{s} \binom{s}{i} q^{s-i} \sum_{k=1}^{i/2} \sum_{\sum_j l_j = i, l_j \geq 2} i! p^{i-k}$$

$$\leq q^s + \sum_{i=2}^{s} s^i q^{s-i} \sum_{k=1}^{i/2} \sum_{\sum_j l_j = i, l_j \geq 2} p^{i-k}.$$

The total number of terms in the innermost sum is equal to the number of solutions to $\sum_j l_j = i, l_j \geq 2$, which is upper bounded by $\binom{i+k-1}{k-1}$. Thus,

$$q^s + \sum_{i=2}^{s} s^i q^{s-i} \sum_{k=1}^{i/2} \sum_{\sum_j l_j = i, l_j \geq 2} p^{i-k} \leq q^s + \sum_{i=2}^{s} s^i q^{s-i} \sum_{k=1}^{i/2} \binom{i+k-1}{k-1} p^{i-k}.$$

Note that for $1 \leq k \leq i/2 - 1$, $\binom{i+k-1}{k-1} p^{i-k} < \binom{i+k}{k} p^{i-k-1}$ as this condition is equivalent to $p < \frac{i+k}{k}$, which is certainly true because $\frac{i+k}{k} > 3$. Thus, the last term in the internal summation is the largest. Hence,

$$q^s + \sum_{i=2}^{s} s^i q^{s-i} \sum_{k=1}^{i/2} \binom{i+k-1}{k-1} p^{i-k} \leq q^s + \sum_{i=2}^{s} s^i q^{s-i} \sum_{k=1}^{i/2} \binom{3i/2-1}{i/2-1} p^{i/2}$$

$$\leq q^s + \sum_{i=2}^{s} s^i q^{s-i} \sum_{k=1}^{i/2} \prod_{j=1}^{i/2-1} \left(\frac{i+j}{j}\right) p^{i/2}.$$

As $\frac{i+j}{j} \leq i + 1$,

$$q^s + \sum_{i=2}^{s} s^i q^{s-i} \sum_{k=1}^{i/2} \prod_{j=1}^{i/2-1} \left(\frac{i+j}{j}\right) p^{i/2} \leq q^s + \sum_{i=2}^{s} s^i q^{s-i} \sum_{k=1}^{i/2} (i+1)^{i/2-1} p^{i/2}$$

$$= q^s + q^s \sum_{i=2}^{s} \frac{i}{2} (i+1)^{i/2-1} \left(\frac{s\sqrt{p}}{q}\right)^i \leq q^s + q^s \sum_{i=2}^{s} \left(\frac{s^2\sqrt{p}}{q}\right)^i.$$

This inequality follows because $\frac{i}{2}(i+1)^{i/2-1} < i^i \leq s^i$ for $2 \leq i \leq s$. Now, $\frac{s^2\sqrt{p}}{q} < 1$ because $s < \left(\frac{n}{d^2}\right)^{d/6}$ due to the assumption that f is a random s-sparse polynomial as per Lemma 3.7 and $\left(\frac{n}{d^2}\right)^{d/3} < e^{-d/n} \left(\frac{n}{d}\right)^{d/2}$ for $n > d^8$ and $d \geq 25$. Therefore,

$$q^s + q^s \sum_{i=2}^{s} \left(\frac{s^2\sqrt{p}}{q}\right)^i \leq sq^s.$$

Thus $\Pr[F^{\pi}] \leq sq^s$. By union bound on all non-trivial permutations π,

$$\Pr[f \text{ has a permutation symmetry}] \leq \sum_{\pi} \Pr[F^{\pi}] < n! q^s s < n^n q^s s.$$

We require that

$$n^n q^s s \ll 1 \iff n^n s \ll e^{ds/n}.$$

Since $s = \Omega(n^3)$, the last inequality holds.

B.17 Proof of Proposition 3.8

Clearly $h_1 \sim h_2$ via the transform $S_1^{-1} P_1^{-1} P_2 S_2 = PS$, where $P = P_1^{-1} P_2$ and S is an appropriate scaling. Suppose $h_2(\mathbf{x}) = h_1(P'S'\mathbf{x})$, for some permutation P' and scaling S'. Then, $f(P_2 S_2 \mathbf{x}) = h_2(\mathbf{x}) = h_1(P'S'\mathbf{x}) = f(P_1 S_1 P'S'\mathbf{x})$. This implies $f(\mathbf{x}) = f(P_1 S_1 P'S' S_2^{-1} P_2^{-1}\mathbf{x}) = f(P_1 P' P_2^{-1} \tilde{S}\mathbf{x})$, where $\tilde{S}$ is an appropriate scaling matrix. Note that $P_1 P' P_2^{-1}$ permutes the monomials of f amongst themselves, thus $P_1 P' P_2^{-1}$ is a permutation symmetry of f. By Proposition 3.7, only the trivial permutation is a permutation symmetry of f. Thus $P_1 P' P_2^{-1} = I \implies P' = P_1^{-1} P_2 = P$, proving uniqueness of P.

B.18 Proof of Theorem 5

Invoke Theorem 1 on the polynomial f to obtain a (n, d, s, t) design polynomial h. If f is in the orbit of $\mathsf{NW}_{q,d,t}$, then $h = \mathsf{NW}_{q,d,t}(PS\mathbf{x})$ for some permutation P and scaling S. Thus, the problem reduces to testing if f is PS-equivalent to $\mathsf{NW}_{q,d,t}$. Note, the permutation P maps the monomials of $\mathsf{NW}_{q,d,t}$ to that of h and there may exist multiple such permutations.

Construct graphs G_1 and G_2 for h and $\mathsf{NW}_{q,d,t}$ using the reduction of Theorem 2, ignoring the coefficients of h. Invoke the GI algorithm of [Bab16] to obtain a permutation $\tilde{P}$ such that $\tilde{P}$ maps the monomials of h to those of $\mathsf{NW}_{q,d,t}$. Now, we show that there exists a scaling matrix $\tilde{S}$ for $\tilde{P}$, such that $h(\tilde{P}\tilde{S}\mathbf{x}) = \mathsf{NW}_{q,d,t}$. We already have that

$$h(\mathbf{x}) = \mathsf{NW}_{q,d,t}(PS\mathbf{x})$$

$$\implies h(\tilde{P}\mathbf{x}) = \mathsf{NW}_{q,d,t}(PS\tilde{P}\mathbf{x}).$$

Take $\tilde{S} := \tilde{P}^{-1} S^{-1} \tilde{P}$, which is a diagonal matrix. Then

$$h(\tilde{P}\tilde{S}\mathbf{x}) = \mathsf{NW}_{q,d,t}(PS\tilde{P}\tilde{S}\mathbf{x}) = \mathsf{NW}_{q,d,t}(PS\tilde{P}(\tilde{P}^{-1}S^{-1}\tilde{P})\mathbf{x}) = \mathsf{NW}_{q,d,t}(P\tilde{P}\mathbf{x}).$$

Now, P maps the monomials of $\mathsf{NW}_{q,d,t}$ to those of h and $\tilde{P}$ maps the monomials of h to those of $\mathsf{NW}_{q,d,t}$. The transform $P\tilde{P}$ in effect maps the monomials of $\mathsf{NW}_{q,d,t}$ amongst themselves. Further, since the coefficients of all the monomials in $\mathsf{NW}_{q,d,t}$ are 1, therefore $P\tilde{P}$ is actually a permutation symmetry of $\mathsf{NW}_{q,d,t}$. Thus, $h(\tilde{P}\tilde{S}\mathbf{x}) = \mathsf{NW}_{q,d,t}(P\tilde{P}\mathbf{x}) = \mathsf{NW}_{q,d,t}(\mathbf{x})$ and the problem reduces to testing if f is S-equivalent to $\mathsf{NW}_{q,d,t}$.

For the time complexity of the reduction, Theorem 1 requires $O(\text{poly}(q^t))$, since for $\mathsf{NW}_{q,d,t}$, $n = qd > d > t$. Since the sparsity of $\mathsf{NW}_{q,d,t}$ is q^t, therefore the graph isomorphism algorithm has a running time of quasi-poly(q^t). Thus, overall the reduction requires quasi-poly(q^t) time.

C Missing proofs from Section 4

C.1 Proof of Lemma 4.1

Let f be a non-degenerate n-variate affine projection of g, where

$$f = T_1 + \cdots + T_s,$$

$T_i = \prod_{j=1}^{d} l_{i_j}$ with $l_{i_j} \in L_i$. The set L_i is linearly independent for all $i \in [s]$ by the non-degeneracy conditions. Fix an i and choose $A_i \in \mathrm{GL}(n, \mathbb{F})$, such that $T_i(A_i\mathbf{x}) = \prod_{j=1}^{d} x_{i_j}$, by extending the linear forms in L_i to a set of n linearly independent linear forms. Consider,

$$h = f(A_i\mathbf{x}) = \tilde{T}_1 + \cdots + \tilde{T}_s$$

where $\tilde{T}_j = T_j(A_i\mathbf{x})$ for all $j \in [s]$, and define

$$\tilde{U} := \langle \partial^k h(\mathbf{x}) \rangle, \;\; \tilde{U}_j := \langle \partial^k T_j(A_i\mathbf{x}) \rangle, \;\; \tilde{V} := \langle \partial^{k+2} h(\mathbf{x}) \rangle, \;\; \tilde{V}_j := \langle \partial^{k+2} T_j(A_i\mathbf{x}) \rangle.$$

By Lemma 3.2 (see footnote 16), $\tilde{U} \cong U$, $\tilde{U}_j \cong U_j$, $\tilde{V} \cong V$ and $\tilde{V}_j \cong V_j$. Further, $\mathrm{Adj}(\partial^2, U, V)$, as a set of tuples of matrices with respect to appropriate bases for U and V, is equal to $\mathrm{Adj}(\partial^2, \tilde{U}, \tilde{V})$, as a set of tuples of matrices with respect to appropriate bases for $\tilde{U}$ and $\tilde{V}$. Thus, showing $\tilde{D}(\tilde{U}_i) \subseteq \tilde{U}_i$ implies $D(U_i) \subseteq U_i$, where $\tilde{D} \in \mathrm{Adj}(\partial^2, \tilde{U}, \tilde{V})$ and $D \in \mathrm{Adj}(\partial^2, U, V)$.

As $\tilde{T}_i = \prod_{j=1}^{d} x_{i_j}$, a basis of $\tilde{U}_i$, call it $\tilde{\mathscr{B}}_i$, is the set of all degree $d-k$ multilinear monomials in $\{x_{i_1}, \cdots x_{i_d}\}$. Extend $\tilde{\mathscr{B}}_i$ to a basis of $\tilde{U}$. Let $\tilde{u} \in \tilde{U}_i$ which is a multilinear polynomial in $\{x_{i_1}, \cdots x_{i_d}\}$. Suppose

$$\tilde{D}(\tilde{u}) = \tilde{u}_1 + \tilde{u}_2 \cdots + \tilde{u}_s,$$

where $\tilde{u}_j \in \tilde{U}_j$. Choose $\mathbf{x}^{\boldsymbol{\alpha}}$ such that $\frac{\partial^2 \tilde{u}}{\partial \mathbf{x}^{\boldsymbol{\alpha}}} = 0$. Such a monomial exists because $\tilde{u}$ is a multilinear polynomial in $\{x_{i,1}, x_{i,2} \ldots x_{i,d}\}$. Then

$$\frac{\partial^2(\tilde{D}\tilde{u})}{\partial \mathbf{x}^{\boldsymbol{\alpha}}} = \frac{\partial^2 \tilde{u}_1}{\partial \mathbf{x}^{\boldsymbol{\alpha}}} + \cdots + \frac{\partial^2 \tilde{u}_s}{\partial \mathbf{x}^{\boldsymbol{\alpha}}}$$

$$\implies \tilde{E}\left(\frac{\partial^2 \tilde{u}}{\partial \mathbf{x}^{\boldsymbol{\alpha}}}\right) = \frac{\partial^2 \tilde{u}_1}{\partial \mathbf{x}^{\boldsymbol{\alpha}}} + \cdots + \frac{\partial^2 \tilde{u}_s}{\partial \mathbf{x}^{\boldsymbol{\alpha}}}$$

$$\implies \sum_{j \neq i} \frac{\partial^2 \tilde{u}_j}{\partial \mathbf{x}^{\boldsymbol{\alpha}}} = 0$$

$$\implies \frac{\partial^2 \tilde{u}_j}{\partial \mathbf{x}^{\boldsymbol{\alpha}}} = 0 \quad \forall j \neq i.$$

The last implication follows from the direct sum of the $\tilde{V}_j$ spaces. Thus, for all monomials $\mathbf{x}^{\boldsymbol{\alpha}}$ which are not multilinear monomials in $\{x_{i_1}, \cdots x_{i_d}\}$, the last equality holds. Since $\mathrm{char}(\mathbb{F}) = 0$ or $> d$, for any polynomial $p(\mathbf{x})$, $\frac{\partial^2 p}{\partial \mathbf{x}^{\boldsymbol{\alpha}}}$ is 0 for the set of monomials just mentioned iff p is a multilinear polynomial in $\{x_{i_1}, \cdots x_{i_d}\}$. Thus, $\tilde{u}_j$ is a degree $d-k$ multilinear polynomial in $\{x_{i_1}, \ldots x_{i_d}\}$. Hence, $\tilde{u}_j \in U_i$ which implies $\tilde{u}_j = 0$ by the direct sum condition. Thus, $\tilde{D}(\tilde{U}_i) \subseteq \tilde{U}_i$ implying $D(U_i) \subseteq U_i$. Repeating this argument for all $i \in [s]$, we get that $D(U_i) \subseteq U_i$ for all $i \in [s]$. It easily follows from the direct sum structure of U and V and $\partial^2 : U \to V$ being block diagonal that $E(V_i) \subseteq V_i$ for all $i \in [s]$.

Because of the direct sum structure of U and V and the fact that the U_i and V_i spaces are invariant under the adjoint operators, bases can be found for U and V where $\mathrm{Adj}(\partial^2, U, V)$ is a

set of tuples of block diagonal matrices with respect to these bases. Further, every $\partial^2 : U \to V$ is also block diagonal. Thus, $\text{Adj}(\partial^2, U, V))$ is comprised of $\text{Adj}(\partial^2, U_i, V_i)$'s which are the adjoint of the individual T_i's. As each T_i is equivalent to a multilinear monomial $\tilde{T}_i$ and $\text{Adj}(\partial^2, \tilde{U}_i, \tilde{V}_i)$ is trivial (provable by a modified Lemma 3.4), by Lemma 3.2 it follows that $\text{Adj}(\partial^2, U_i, V_i)$ is trivial and thus so is $\text{Adj}(\partial^2, U, V)$.

C.2 Proof of Lemma 4.2

The proof is similar to that of Lemma 3.5. By Lemma 4.1, there exists a basis $\mathscr{B}$ with respect to which D is a block diagonal matrix, with each block being a constant multiple of the identity matrix. Since each block of D has equal diagonal entries and there are s blocks, showing that any two distinct blocks have different diagonal entries suffices. Let $\{D^{(1)}, D^{(2)} \dots D^{(b)}\}$ be the elements of the basis computed for $\text{Adj}(\partial^2, U, V)_1 = \{D \mid (D, E) \in \text{Adj}(\partial^2, U, V)\}$. A random element of $\text{Adj}(\partial^2, U, V)_1$ is of form $\sum_{j=1}^{b} c_j D^{(j)}$, where $c_j \in_r S \subseteq \mathbb{F}$. Note that:-

- Define 1_{U_i} and 1_{V_i} as the projection operator to the spaces U_i and V_i, respectively. In the basis $\mathscr{B}$, the i-th block of 1_{U_i} is the identity matrix while the rest are all 0 blocks. Then $(1_{U_i}, 1_{V_i}) \in \text{Adj}(\partial^2, U, V)$ because for any $u \in U$ and any ∂^2 operator, where $u = u_1 + \cdots + u_s$, $v = v_1 + \cdots + v_s$, $\partial^2(u) = v$, $u_i \in U_i$ and $v_i \in V_i$, it holds that:

  $$\partial^2(1_{U_i}(u)) = \partial^2(u_i) = v_i \text{ and } 1_{V_i}(\partial^2 u) = 1_{V_i}(v) = v_i$$

  Therefore, $(\sum_{i=1}^{s} \lambda_i 1_{U_i})$ belongs to $\text{Adj}(\partial^2, U, V)$, where $\lambda_i \neq \lambda_j$ for $i \neq j$. Clearly, $\sum_{i=1}^{s} \lambda_i 1_{U_i}$ is a block-diagonal matrix with each block being a scalar multiple of identity, hence it has s distinct eigenvalues.

- Let $i_1 \neq i_2$ and $D[i_1][i_1]$ denote the diagonal entry of the i_1-th block of an operator D. The difference of the diagonal entry of the i_1-th and i_2-th block is $\sum_{j=1}^{b} c_j (D^{(j)}[i_1][i_1] - D^{(j)}[i_2][i_2])$ which is a linear polynomial in c_j's. As 1_{U_i} is in the adjoint, this polynomial in c_j's is non-zero.

Applying the Schwartz-Zippel lemma on this polynomial, the i_1-th and i_2-th block have the same diagonal entries with probability $\leq \frac{1}{|S|}$. By applying union bound on all $\binom{s}{2}$ possible pairs of i_1 and i_2, the probability that some pair of blocks have equal diagonal entries is $\leq \frac{\binom{s}{2}}{|S|}$. Hence, D has s distinct eigenvalues with probability $\geq 1 - \frac{\binom{s}{2}}{|S|}$.

C.3 Proof of Lemma 4.3

By Lemma 4.1, there exists a basis $\mathscr{B}$ where D is a block diagonal matrix where each block is a scalar multiple of the identity matrix. In $\mathscr{B}$, the i-th block of $(D - \lambda_i I)$ is the 0 matrix while the other blocks in $D - \lambda_i I$ are non-zero multiples of the identity matrix as by Lemma 4.2, D has s distinct eigenvalues. Thus, $\text{Ker}(D - \lambda_i I) = U_i$.

C.4 Proof of Lemma 4.4

Let the $\mathbf{y}$ variables be assigned the linear forms $\{l_1, \dots, l_m\}$, where $l_q = \sum_{r=1}^{n} a_{q,r} x_q$. Let the matrix $A \in \mathbb{F}^{m \times n}$ represent the linear forms. Then,

$$f(\mathbf{x}) = g(A\mathbf{x}) = T_1 + \cdots + T_s$$

is a polynomial with $T_i := \prod_{j=1}^{d} l_{i_j}$ and $l_{i_j} \in L_i$.

Note that the set L_i can be expressed as a $d \times n$ matrix A_i, with entries as $a_{q,r}$. If there exists a $d \times d$ minor of A_i with a non-zero determinant, then L_i is linearly independent. By the Schwartz-Zippel lemma, with probability $\geq 1 - \frac{d}{|S|}$, where $S \subseteq \mathbb{F}$, there exists a $d \times d$ minor in A_i which has a non-zero determinant when the $a_{q,r}$ are chosen randomly from S. By union bound on all s sets, the probability that all L_i are linearly independent for $i \in [s]$ is $\geq 1 - \frac{ds}{|S|}$.

For non-degeneracy condition 1, consider the $\binom{n+k-1}{k} \times s\binom{d}{k}$ matrix B_1, with rows indexed by degree k monomials in $\mathbf{x}$ and columns indexed by the degree k derivatives of the monomials of g with l_i's substituted. Each row of C_1 represents an element of $\{\partial^k f\}$ expressed as a linear combination of the elements of $\{\partial^k g(A\mathbf{x})\}$ with the entries as polynomials in $a_{q,r}$. Similarly, we have the $\binom{n+k+1}{k+2} \times s\binom{d}{k+2}$ matrix B_2 for the space V with entries as polynomials in $a_{q,r}$.

If there exists a minor of dimension[23] $s\binom{d}{k} \times s\binom{d}{k}$ with non-zero determinant in B_1, then $\dim U = s\binom{d}{k}$. This determinant is a polynomial of degree $ds\binom{d}{k}$ in mn variables. By the Schwartz-Zippel lemma, if this determinant is non-zero for some choice of $a_{q,r}$'s, then with probability $\geq 1 - \frac{ds\binom{d}{k}}{|S|}$, this determinant is non-zero for a random choice of $a_{q,r}$ from $S \subseteq \mathbb{F}$. Similarly, if B_2 has a minor of dimension $s\binom{d}{k+2} \times s\binom{d}{k+2}$ with non-zero determinant for some choice of $a_{q,r}$, then $\dim V = s\binom{d}{k+2}$ with probability $\geq 1 - \frac{ds\binom{d}{k+2}}{|S|}$. For $|S| \geq sd^{k+3}$, both the non-degeneracy conditions hold with high probability. We now show that there exists a choice of $a_{q,r}$'s for which B_1 and B_2 have full column rank.

Consider the following two-stage process of assigning linear forms to $\mathbf{y}$:

1. Each of the $\mathbf{y}$ variables of g are assigned $\sqrt{n}$ partition variables $\{p_1, p_2 \dots p_{\sqrt{n}}\}$ of the n variables where each p_i corresponds to a $\sqrt{n}$ size partition W_i of $\mathbf{x}$.

2. For each p_i, a linear form is chosen in the variables contained in W_i.

The two-stage random process involves assigning to each $\mathbf{y}$ variable a p_i chosen uniformly at random and then replacing p_i with a random linear form in the variables of W_i. The claim is that under this random process, $\dim U = s\binom{d}{k}$ and $\dim V = s\binom{d}{k+2}$ with non-zero probability.

Let g_i and g_j be two arbitrary monomials of g with gcd $h_{i,j}$. By the design condition, $h_{i,j}$ has $< t$ variables. Let $h_i := \frac{g_i}{h_{i,j}}$ and $h_j := \frac{g_j}{h_{i,j}}$. After assigning partition variables to g_i and g_j, let M_i and M_j denote the monomials in the partition variables respectively and similarly denote $\tilde{M}_i$ and $\tilde{M}_j$ for h_i and h_j.

Let $N = \sqrt{n}$ and $F_{i,j}$ be the event deg gcd$(M_i, M_j) \geq t + c$. Note that the common variables in g_i and g_j are already assigned the same partition, thus deg gcd(M_i, M_j) rises iff there are variables in the variable disjoint monomials $\tilde{g}_i$ and $\tilde{g}_j$ which get assigned the same partition variable. Thus,

$$\deg \gcd(\tilde{M}_i, \tilde{M}_j) \geq c \iff \deg \gcd(M_i, M_j) \geq t + c.$$

Clearly, deg gcd$(\tilde{M}_i, \tilde{M}_j) \geq c$ if and only if $\exists M$ such that deg $M = c$ and $M | \tilde{M}_i$ and $M | \tilde{M}_j$. A monomial in N variables of degree d can be formed by choosing d variables uniformly at random with repetition in N^d ways. Suppose M divides $\tilde{M}_i$ for some degree c monomial M. Such an $\tilde{M}_i$ can be formed by first forming the degree c monomial M, which can be done in $\leq \binom{d}{c}\binom{c}{\alpha_1 \dots \alpha_N}$, and then choosing the remaining variables in N^{d-c} ways. Thus

[23] Note that k will be chosen such that the number of rows is greater than the number of columns to ensure this is possible.

$$\Pr[M|\tilde{M}_i, \deg M = c] \leq \frac{\binom{d}{c}\binom{c}{\alpha_1 \dots \alpha_N} N^{d-c}}{N^d} = \frac{\binom{d}{c}\binom{c}{\alpha_1 \dots \alpha_N}}{N^c},$$

where $\binom{c}{\alpha_1 \dots \alpha_N} \leq c!$ is the multinomial coefficient corresponding to M.

Since $\tilde{M}_i$ and $\tilde{M}_i$ are selected independently of one another (as $\tilde{g_i}$ and $\tilde{g_j}$ are variable disjoint), therefore

$$\Pr[M|\tilde{M}_i, M|\tilde{M}_j] \leq \frac{\binom{d}{c}^2 \binom{c}{\alpha_1 \dots \alpha_N}^2}{N^{2c}}.$$

Using union bound on all possible degree c monomials ($\binom{N+c-1}{c}$ of them) gives

$$\Pr[F_{i,j}] \leq \sum_{\sum \alpha_i = c} \frac{\binom{d}{c}^2 \binom{c}{\alpha_1 \dots \alpha_N}^2}{N^{2c}}.$$

Finally, applying union bound on the $\binom{s}{2}$ possible pairs of monomials:

$$\begin{aligned}
\Pr[\exists\, i, j \in [s] : F_{i,j}] &\leq \sum_{1 \leq i < j \leq s} \sum_{\sum \alpha_i = c} \frac{\binom{d}{c}^2 \binom{c}{\alpha_1 \dots \alpha_N}^2}{N^{2c}} \\
&= \binom{s}{2} \sum_{\sum \alpha_i = c} \frac{\binom{d}{c}^2 \binom{c}{\alpha_1 \dots \alpha_N}^2}{N^{2c}} \\
&\leq s^2 \frac{\binom{d}{c}^2}{N^{2c}} \sum_{\sum \alpha_i = c} \binom{c}{\alpha_1 \cdots \alpha_N}^2 \\
&\leq s^2 \frac{\binom{d}{c}^2}{N^{2c}} \sum_{\sum \alpha_i = c} (c!)^2 = s^2 \frac{\binom{d}{c}^2 (c!)^2 \binom{N+c-1}{c}}{N^{2c}} \\
&\leq s^2 \frac{d^{2c} \binom{N+c-1}{c}}{N^{2c}} = s^2 \frac{d^{2c}}{N^{2c}} \prod_{j=1}^{c} \Big(\frac{N+j-1}{j} \Big).
\end{aligned}$$

As $n > d^4 \geq 1$ and $\frac{N+j}{j+1} \leq \frac{N+j-1}{j}$ for all $j \geq 1$, therefore $\frac{N+j-1}{j} \leq N$. Thus

$$s^2 \frac{d^{2c}}{N^{2c}} \prod_{j=1}^{c} \Big(\frac{N+j-1}{j} \Big) \leq s^2 \frac{d^{2c} N^c}{N^{2c}} = \frac{s^2 d^{2c}}{N^c}.$$

We need that

$$\frac{s^2 d^{2c}}{N^c} < 1 \iff s^2 < \Big(\frac{N}{d^2} \Big)^c.$$

Using $N = \sqrt{n}$ and taking $c = \left\lfloor \frac{2 \log(s)}{\log(\sqrt{n}/d^2)} \right\rfloor + 1$, the g_i's after being mapped to monomials in $\mathbf{p}$ have gcd of degree $< t + c$ with non-zero probability as:

$$\frac{s^2 d^{2c}}{N^c} = \frac{d^2}{\sqrt{n}} < 1.$$

and $n > d^4$. Since the partitions are variable disjoint, the linear forms assigned to each monomial are linearly independent. Let $k = t + c$. When the T_i's, which are product of linear

forms, are expanded, the gcd of any two monomials from different T_i's is of degree $< k$. Also, $d \geq 2k+2$. Hence, for any $\mathbf{x}^{\boldsymbol{\alpha}}$ with $|\boldsymbol{\alpha}| = k$, $\frac{\partial^k f}{\partial \mathbf{x}^{\boldsymbol{\alpha}}} = \frac{\partial^k T_i}{\partial \mathbf{x}^{\boldsymbol{\alpha}}}$ for some i. Thus, for $k = c + t$, the direct sum and equality hold for U and V by a proof similar to that of Lemma 3.1. Due to non-degeneracy condition 2 and Lemma 3.2 (see footnote 16), $\dim U_i = \binom{d}{k}$. Therefore $\dim U = s\binom{d}{k}$ and similarly $\dim V = s\binom{d}{k+2}$. Finally, the number of rows is greater than the number of columns in B_1 for this choice of k as:

$$\binom{n+k-1}{k} \geq \left(\frac{n}{k}\right)^k \text{ and } s\binom{d}{k} \leq s\left(\frac{ed}{k}\right)^k$$

Thus, if

$$s\left(\frac{ed}{k}\right)^k < \left(\frac{n}{k}\right)^k \iff s < \left(\frac{n}{ed}\right)^k$$

then we are done. Now, $k > \frac{2\log(s)}{\log(\sqrt{n}/d^2)}$, therefore

$$\left(\frac{n}{ed}\right)^k > \left(\frac{n}{ed}\right)^{\frac{2\log(s)}{\log(\sqrt{n}/d^2)}} > s.$$

Similarly, the number of rows is greater than the number of columns in B_2 as well. The constraint on $|\mathbb{F}|$ can be easily derived from $|S| \geq sd^{k+3}$, $n \geq d^{4+\epsilon}$ and the choice of k.

C.5 Proof of Proposition 4.1

The proof is similar to that of Proposition 3.1. Assume black-box access to the T_i's is available. Fact 2.4 gives black-box access to the irreducible factors of T_i up to scaling by field elements. Since T_i is in the orbit of a monomial, it must factor into a product of linear forms. From black-box access to the factors of T_i, the linear forms can be recovered by querying the black-box at the set of points $\{e_i : i \in [n]\}$, where e_i is the point with i-th coordinate 1 and the rest as 0. Normalize the coefficients of all the linear forms across all the black-boxes of the T_i's for all $i \in [s]$. Thus, per T_i we get a product of linear forms Q_i which is multiplied by some field element due to the normalization process.

Now, assign to each linear form a $\mathbf{y}$ variable while maintaining the consistency of assignment across T_i's, to get a transform $B \in \mathbb{F}^{m \times n}$. Since it is not known which variable is mapped to a particular linear form in the original transformation, the recovered linear forms are assigned to a permutation of the variables. Thus, the transform B that is returned is related to A as $B = PSA$, for some permutation matrix P and scaling matrix S, where both are $m \times m$ matrices and the scaling is due to the normalization.

Finally, for each T_i, form a monomial h_i by replacing each linear form of Q_i with the $\mathbf{y}$ variable corresponding to it in the transform B. This forms the polynomial $h(\mathbf{y}) = c_1 h_1 + \cdots + c_s h_s$, with c_i's being constants resulting from the normalization of the linear forms, such that $f = h(B\mathbf{x})$. The polynomial h is a multilinear (m, d, s, t) design polynomial as each T_i is a product of d distinct linear forms mapped to distinct $\mathbf{y}$ variables.

For the time complexity, Fact 2.4 is executed s times, and each invocation requires time $\text{poly}(n, d)$. The linear forms can be recovered and normalized in $\text{poly}(m, n, d, s)$ time. Similarly, h can be formed in $\text{poly}(m, n, d, s)$ time. Hence, the recovery process takes $\text{poly}(m, n, d, s)$ time.

When the input is $f = g(A\mathbf{x} + \mathbf{b})$, the recovery of the translation vector $\mathbf{b}$ is similar, as the returned black-box factors are affine forms and querying them at all 0's recovers $\mathbf{b}$ up to scaling. Thus, the recovered transform B is PSA, and the recovered translation vector $\mathbf{c}$ is $PS\mathbf{b}$.

C.6 Proof of Proposition 4.2

The time complexity of Algorithm 4 is dominated by Steps 1 and 2. Step 1 uses Fact 2.1 to compute black-box access to the basis. The precise time complexity of Fact 2.1 for a single invocation is $\text{poly}(n, d^l)$ where l is the number of variables in $\mathbf{x}^{\boldsymbol{\alpha}}$ with respect to which differentiation is done. Note $l \leq \min(k, n)$. Since there are $\binom{n+k-1}{k}$ many order k derivatives and $k < n$, Step 1 needs $\text{poly}(\binom{n+k}{n} d^k$ time to execute. The complexity of Step 2 is $\text{poly}(n, s, d^t)$ by Proposition 4.3.

Using Fact 2.3, Step 3 can be executed. There are $O(sd^k)$ many linearly independent polynomials (since $\dim U = s\binom{d}{k} = O(sd^k)$) of degree $d-k$ and $\binom{n+k-1}{k}$ many derivatives of degree k, hence Step 3 requires $\text{poly}(s, \binom{n+k}{k} d^k)$ time. The running time of the recovery procedure is $\text{poly}(m, n, s, d)$ by Proposition 4.1.

Thus, the overall complexity of the algorithm is $\text{poly}(m, s, \binom{n+k}{k} d^k)$. Further, since $n > d > k$, this simplifies to $\text{poly}(m, s, (nd)^k)$. For the given choice of k and because $n \geq d^{4+\epsilon}$, $(nd)^k = \text{poly}(n^t, s)$. Hence, the complexity is $\text{poly}(m, s, n^t)$.

C.7 Proof of Proposition 4.3

The dominant time complexity is that of Step 1, which involves solving a system of $\binom{n+1}{2} \cdot \dim(V) \cdot \dim(U)$ many linear equations in $\dim(U)^2 + \dim(V)^2$ variables. Since $\dim(U) = s\binom{d}{k} < sd^k$ and $\dim(V) = s\binom{d}{k+2} < sd^{k+2}$, there are $\text{poly}(nsd^k)$ linear equations and $\text{poly}(sd^k)$ variables in the system. Thus, such a system can be solved in $\text{poly}(nsd^k)$ time. The computation of the eigenvalues requires factoring a univariate polynomial of degree sd^k, which can be done in (randomized) $\text{poly}(sd^k)$ time, and computing the null spaces, which can also be done in $\text{poly}(sd^k)$ time. Thus, the overall time complexity is $\text{poly}(nsd^k)$. Now,

$$k = t + \left\lfloor \frac{2\log s}{\log(\sqrt{n}/d^2)} \right\rfloor + 1 < t + \frac{2\log s}{\log(\sqrt{n}/d^2)} + 1.$$

Thus,

$$d^k < d^{t+1} d^{\frac{2\log s}{\log(\sqrt{n}/d^2)}} = d^{t+1} s^{\frac{2\log(d)}{\log(\sqrt{n}/d^2)}} < d^{t+1} s^{4/\epsilon}.$$

The last inequality follows because $n \geq d^{4+\epsilon}$. Thus, the time complexity is $\text{poly}(n, s, d^t)$.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il