

Deterministic Shift Equivalence Testing for Sparse Polynomials

Abstract

We study the shift equivalence testing problem for high-degree sparse multivariate polynomials. Given sparse representations of two polynomials $f, g \in \mathbb{Q}[x_1, \dots, x_n]$, with exponents encoded in binary, sparsities s and t , and degree at most d , the task is to decide whether there is a vector $\mathbf{a} \in \mathbb{Q}^n$ such that

$$f(\mathbf{x} + \mathbf{a}) \equiv g(\mathbf{x}) .$$

Furthermore, one needs to output such a vector if it exists. Dvir, Oliveira, and Shpilka (ICALP 2014) studied shift equivalence for low-degree arithmetic circuits. Their framework does not apply directly to sparse polynomials, which are not closed under shifts, and they left open a deterministic algorithm polynomial in the binary sparse input size.

We answer this question positively over $\mathbb{Q}$. If the input coefficients have bit-size at most τ , we show that a rational shift of polynomial bit-size exists whenever the polynomials are shift equivalent. Moreover, our algorithm outputs such a shift and has a running time of $\text{poly}(n, s, t, \tau, \log d)$ in the bit model. The arithmetic and structural parts of our result also hold over any field of characteristic zero or characteristic larger than d .

1 Introduction

Shift Equivalence Testing (SET) asks the following seemingly simple question: Given two polynomials $f, g \in \mathbb{Q}[x_1, \dots, x_n]$, decide whether there is a vector $\mathbf{a} = (a_1, \dots, a_n) \in \mathbb{Q}^n$ such that

$$f(\mathbf{x} + \mathbf{a}) \equiv g(\mathbf{x}),$$

and output such a vector if it exists. The symbol $\equiv$ emphasizes that the equality is an identity of formal polynomials, not merely an equality of functions. Over $\mathbb{Q}$ these two notions coincide. The same is true, and the Taylor expansions used below remain valid, over any field of characteristic zero or characteristic larger than the degree under consideration. Note that shift equivalence is indeed an equivalence relation. It is reflexive, since f can be transformed into itself by the zero shift. When $f(\mathbf{x} + \mathbf{a}) = g(\mathbf{x})$, then also $f(\mathbf{x}) = g(\mathbf{x} - \mathbf{a})$, so it is symmetric. And if shifting by $\mathbf{a}$ transforms f into g and shifting by $\mathbf{b}$ transforms g into h , then $\mathbf{a} + \mathbf{b}$ transforms f into h , giving transitivity.

The algorithmic problem of Shift Equivalence Testing first appeared in computer algebra and symbolic computation. Grigoriev [Gri97] designed the first polynomial-time algorithms for this problem. However, he assumes that the polynomials are given in the so-called dense representation, that is, as a list of *all* coefficients including the ones that are zero. For a polynomial with n variables of individual degree d , this means that the input has size $(d + 1)^n$. His deterministic characteristic-zero algorithm takes $n^{O(d)}$ time, so zero-padding a sparse input does not give a running time polynomial in the sparse parameters. Grigoriev also gives a randomized algorithm for large enough finite characteristic and a quantum algorithm for characteristic two.

While the running time of Grigoriev’s algorithms is exponential in d , it is polynomial in the input size, because the input size is large. This is unsatisfying if the polynomial has only very few nonzero coefficients, since the dense representation “buys” running time for the algorithm. In a seminal paper, Dvir, Oliveira, and Shpilka [DOS14] address this question; they assume that the polynomials are given compactly by arithmetic circuits. Arithmetic circuits are succinct representations of polynomials. For instance, while the dense representation of the determinant has length $O(2^{n^2})$ (as a multilinear polynomial in n^2 variables), it can be represented as an arithmetic circuit of polynomial size. Dvir et al. design a randomized algorithm for SET that has polynomial running time in the size of the circuit and the degree d of the polynomials. In fact, their algorithm also works when having black-box access to the polynomials f and g and a bound on their degree and circuit size. The algorithm is designed by a reduction to the so-called polynomial identity testing.

The polynomial identity testing problem (PIT) is a famous problem in complexity theory. Given a polynomial $f \in \mathbb{Q}[x]$ as an arithmetic circuit C , is f identically zero? There is a simple randomized polynomial-time algorithm for the problem based on the so-called Schwartz–Zippel lemma [Sch80, Zip79], which essentially inserts a random point from a large enough set and checks whether it evaluates to zero. Derandomizing PIT is a big open problem in complexity theory. In particular, it implies non-uniform circuit lower bounds [KI04], another major open problem in the field. Note that the algorithm by Dvir et al. shows that SET reduces to PIT under Turing reductions. But also the converse is true. The polynomial f is identically zero iff it is shift-equivalent to the zero polynomial. Therefore, a full derandomization of SET is out of reach of current methods.

Since finding a deterministic algorithm for general PIT is a hard problem, researchers have developed deterministic identity tests for restricted circuit classes; see, e.g., [DG24]. For read-once oblivious algebraic branching programs (ROABPs), white-box PIT is deterministic polynomial time [RS05], while deterministic black-box PIT is known in quasipolynomial time [FS13, FSS14].

The framework of Dvir, Oliveira, and Shpilka [DOS14] therefore gives the corresponding white-box and black-box SET algorithms for ROABPs.

Their deterministic reduction requires closure properties. Sparse polynomials are not closed under shifts, so it does not apply directly and DOS'14 states no deterministic sparse-class algorithm. Indirectly, however, representing both explicit sparse inputs as ROABPs in one common variable order and using white-box ROABP PIT gives deterministic SET with $\text{poly}(n, s, t, d)$ arithmetic complexity. This is not polynomial in the binary sparse input size when d is exponential.

We will answer this question affirmatively. Before doing so, we look at a subtlety of the sparse representation. The input polynomials are given as an explicit list of nonzero terms

$$c x_1^{\alpha_1} \cdots x_n^{\alpha_n}, \quad c \in \mathbb{Q},$$

where the coefficient is given explicitly in binary representation. Furthermore, the exponents are given in binary representation, too. We call this *sparse representation*. Thus, the natural input parameters are the number of variables n , the number of coefficients $s = \text{sp}(f)$ and $t = \text{sp}(g)$ (also called sparsity), the height, equivalently bit-size, of the coefficients of f, g , and $\log d$, where d is the degree of f and g . (Sometimes, the maximum bit-size of the coefficients is called logarithmic height, but we use the term height for brevity.) While low degree is conventional in algebraic complexity, high degree is natural for sparse or lacunary polynomials: an exponent bounded by d uses only $O(\log d)$ bits. Classical sparse interpolation likewise targets this regime: Ben-Or–Tiwari has algebraic-RAM complexity polynomial in $\log d$ [BOT88], and Garg–Schost obtains such dependence under straight-line-program access [GS09]. The running time of the randomized algorithm by Dvir, Oliveira, and Shpilka is polynomial in the size of the circuit and in the degree d . If we derandomize it for sparse polynomials, then we would get an algorithm whose running time is polynomial in the size of the sparse representation and in d , but not in the size of the sparse representation alone, which is the input size.¹ We will also remove the dependence on d in the running time in our deterministic algorithm, thereby answering the question in the strongest possible way.

More formally, let **Sparse SET** denote the following problem: given two polynomials $f, g \in \mathbb{Q}[x_1, \dots, x_n]$ in sparse representation, decide whether there exists a shift $\mathbf{a} \in \mathbb{Q}^n$ such that

$$f(\mathbf{x} + \mathbf{a}) \equiv g(\mathbf{x}).$$

As mentioned above, PIT for sparse polynomials is trivial in the sparse representation: one simply compares the monomial list with the zero list. The issue is now that the sparsity is not preserved under shifts. A single monomial can become very large: if we define $\text{supp}(\mathbf{a}) := \{i : a_i \neq 0\}$, then

$$(\mathbf{x} + \mathbf{a})^\gamma = \prod_{i=1}^n (x_i + a_i)^{\gamma_i}$$

has exactly

$$\prod_{i \in \text{supp}(\mathbf{a})} (\gamma_i + 1)$$

nonzero monomials. This number can be exponential in the sparse input length, already for a single monomial and even when the exponents are given in unary. Thus, even if f is sparse, the shifted polynomial $f(\mathbf{x} + \mathbf{a})$ may be dense. The closure obstruction identified by Dvir, Oliveira, and Shpilka [DOS14, Section 6] means that sparse descriptions cannot simply be propagated

¹For this reason, sometimes sparse representations had the exponents written in unary. [KK05] tried to coin the term lacunary or supersparse for the representation when the exponents are given in binary; but here we use the term sparse representation and assume that the degrees are always given in binary.

through their descent. Although the low-degree workaround above is polynomial in d , it does not give an algorithm polynomial in the sparse input length when d is exponential. This leads to the central question addressed in this paper.

Problem 1.1. *Can Sparse SET over $\mathbb{Q}$ be solved using $\text{poly}(n, s, t, \log d)$ arithmetic operations and bit complexity $\text{poly}(n, s, t, \log d, \tau)$?*

In this paper, we answer the question affirmatively. The first key ingredient is a new structural lemma showing that, in yes instances, the witnessing shift is very special, which forces the relevant intermediate polynomials to remain sparse. This addresses the problem that sparse polynomials are not closed under shifts.

The second problem stems from the fact that the algorithm in [DOS14] has running time polynomial in d , so there is no easy way of adapting this algorithm to a running time of $\text{poly}(\log d)$. The algorithm of [DOS14] iteratively solves a system of equations yielding certain intermediate polynomials. We modify this approach in a nontrivial manner by showing that it is enough to store and compare all intermediate polynomials in the sparse representation without expanding arbitrary shifts.

There are also further obstacles. Even in yes instances over $\mathbb{Q}$, one must first show that there exists a witnessing shift $\mathbf{a}$ in $\mathbb{Q}^n$, rather than only over an algebraic extension². One must then prove that such a shift has polynomially bounded bit complexity in the input size, and for the search version, compute and output it within the same bounds.

1.1 Our results

Our main theorem is a deterministic algorithm for shift equivalence of sparse multivariate polynomials over the rationals, with running time polynomial in the sparse input parameters. We also prove polynomial bit complexity and output a shift of polynomial bit-size.

For an extension field $\mathbb{K} \supseteq \mathbb{Q}$, define

$$S_{f,g}(\mathbb{K}) := \{\mathbf{a} \in \mathbb{K}^n : f(\mathbf{x} + \mathbf{a}) \equiv g(\mathbf{x})\}.$$

In particular, $S_{f,g}(\overline{\mathbb{Q}})$ denotes the set of shifts over the algebraic closure. The rationality result below shows that if $S_{f,g}(\overline{\mathbb{Q}})$ is nonempty, then it contains a rational point. Thus, for the rational decision problem, we may write

$$S_{f,g} := S_{f,g}(\mathbb{Q})$$

without changing the decision problem.

We state the main results over $\mathbb{Q}$ to keep the notation simple. The structural statements and arithmetic algorithms are purely algebraic and remain valid over any field $\mathbb{K}$ of characteristic zero or characteristic larger than d , after replacing $\mathbb{Q}$ by $\mathbb{K}$ and omitting the height and bit-complexity assertions.

Theorem 1.1 (Main theorem). *Let $f, g \in \mathbb{Q}[x_1, \dots, x_n]$ be given in sparse representation, with sparsities s and t and degree at most d . There is a deterministic algorithm using $\text{poly}(n, s, t, \log d)$ arithmetic operations that decides whether $S_{f,g}$ is nonempty. If $S_{f,g} \neq \emptyset$, the algorithm outputs a shift $\mathbf{a} \in S_{f,g}$.*

If the input coefficients have height at most τ , then one can choose the output shift to have height $\text{poly}(n, s, t, \tau, \log d)$, and the algorithm can be implemented using $\text{poly}(n, s, t, \tau, \log d)$ bit operations.

²Consider $f(\mathbf{x} + \mathbf{z}) - g(\mathbf{x}) = \sum_{\alpha} F_{\alpha}(\mathbf{z})\mathbf{x}^{\alpha}$. Thus, one needs to find a common solution of $F_{\alpha}(\mathbf{z})$, and the weak Nullstellensatz gives a common solution $\mathbf{a} \in \overline{\mathbb{Q}}^n$; for more details, see [theorem 1.3](#)

The arithmetic algorithm is proved in [theorem 4.1](#). The bit-complexity analysis over $\mathbb{Q}$ is given in [proposition 4.1](#), and the polynomial-height output guarantee uses the small rational witness theorem, [theorem 1.3](#). This exponentially improves on the indirect ROABP consequence of the DOS'14 framework described above, whose running time has polynomial dependence on d , rather than on $\log d$. In addition, that framework does not by itself address the existence of a rational shift of polynomial bit-size. Thus both the $\text{poly}(n, s, t, \log d)$ arithmetic bound and the bit-complexity guarantee require additional ideas.

Black-box access. Given separate evaluation oracles for f, g promised to be sparse, Ben-Or–Tiwari interpolation [[BOT88](#)] recovers their explicit representations using $\text{poly}(n, s, t, \log d)$ queries and arithmetic operations; [theorem 1.1](#) then gives the same black-box arithmetic bound. This does not imply the same bit bound: evaluating x^d at 2 already returns a $(d + 1)$ -bit integer. For integer coefficients of magnitude at most H , the modular oracle of Bläser–Jindal [[BJ14](#)] yields deterministic $\text{poly}(n, s, t, \log d, \log H)$ bit-complexity interpolation. Division-free integer circuits simulate this oracle. For division-free $\mathbb{Q}$ -circuits, one skips primes dividing an input-constant denominator; after Chinese remaindering, rational reconstruction recovers each coefficient once the combined modulus exceeds $2H^2$, where H bounds its numerator and denominator [[KLW90](#)].

The first key technical ingredient is the following sparsity bound. It rules out the possibility that, in a yes instance, the sparse output is obtained only by cancellations among large shifted monomial expansions.

Theorem 1.2 (Structural sparsity bound). *Let $f, g \in \mathbb{Q}[x_1, \dots, x_n]$ have sparsities s and t , respectively, and degree at most d . If $\mathbf{a} \in S_{f,g}$, then for every monomial $\mathbf{x}^\gamma$ appearing in f or in g ,*

$$\prod_{i \in \text{supp}(\mathbf{a})} (\gamma_i + 1) \leq st.$$

In fact, the theorem we prove is much stronger than the statement above. It applies already when $g(\mathbf{x})$ and $f(\mathbf{x} + \mathbf{a})$ agree only in degrees at least some threshold r . In that case, for every monomial of f of degree at least r , the part of its shifted expansion that can contribute to degrees at least r has sparsity at most st . This truncated form is what is used during the degree-by-degree descent in the proof of [theorem 1.1](#); see [lemma 3.2](#). The stronger version is needed for our analysis to work.

Remark 1.1. The inequality in [theorem 1.2](#) does not bound the total degree of f or g . It only bounds the exponents in the coordinates in which the shift is nonzero. This is the relevant restriction for sparse representation: variables outside $\text{supp}(\mathbf{a})$ do not create binomial expansions, while a monomial $\mathbf{x}^\gamma$ expands into

$$\prod_{i \in \text{supp}(\mathbf{a})} (\gamma_i + 1)$$

terms in the shifted coordinates. Thus, the theorem does not say anything about how large the exponents in variables outside the support of the shift could be; it rules out a large binomial expansion in the shifted coordinates.

The second ingredient, needed for bit complexity over $\mathbb{Q}$, is that a small rational witness always exists. This is not automatic from the existence of an arbitrary shift: a priori, the solution set could contain only shifts over an algebraic extension, or contain rational shifts only with very large coordinates. We prove that neither obstruction occurs for sparse SET.

Theorem 1.3 (Small rational witness). *Let $f, g \in \mathbb{Q}[x_1, \dots, x_n]$ have sparsities s and t , degree at most d , and coefficient height at most τ . If $S_{f,g}(\overline{\mathbb{Q}}) \neq \emptyset$, then $S_{f,g}(\mathbb{Q})$ contains a shift whose coordinates have height*

$$\text{poly}(n, s, t, \tau, \log d).$$

Moreover, the polynomial bound is explicit.

The proof of [theorem 1.3](#) is deferred to [section A](#). It uses the structural sparsity bound again. Once a shift exists, the structural theorem restricts the derivatives of f in the shifted coordinates to a polynomial-dimensional space. Inside this space, the translation from f to g can be represented by a rational differential operator of polynomial height. Taking a finite logarithm of this operator and solving one linear system gives a rational shift of polynomial bit-size.

As a further corollary of the structural bound, we obtain a sparse-time white-box PIT algorithm for shifted-sparse expressions. Here the shifts are given explicitly.

Corollary 1.1 (White-box PIT for shifted-sparse polynomials). *Let $f, g \in \mathbb{Q}[x_1, \dots, x_n]$ be sparse polynomials of sparsities s and t and degree at most d , and let $\mathbf{a}, \mathbf{b} \in \mathbb{Q}^n$ be given. There is a deterministic algorithm using $\text{poly}(n, s, t, \log d)$ arithmetic operations that decides whether*

$$f(\mathbf{x} + \mathbf{a}) \equiv g(\mathbf{x} + \mathbf{b}).$$

Further, if the input coefficients as well as the vectors $\mathbf{a}, \mathbf{b}$ have height at most τ , then the algorithm can be implemented using $\text{poly}(n, s, t, \tau, \log d)$ bit operations.

The proof is given after [lemma 3.2](#) in [section 3.2](#).

Remark 1.2. Without the structural theorem, the immediate deterministic approach gives only a bound that depends polynomially on d . Indeed, $f(\mathbf{x} + \mathbf{a}) - g(\mathbf{x} + \mathbf{b})$ can be represented as a sum of products of univariate polynomials, and hence as a read-once oblivious algebraic branching program. Applying the white-box PIT algorithm of Raz and Shpilka [[RS05](#)] gives a deterministic verification algorithm using $\text{poly}(n, s, t, d)$ arithmetic operations. The corollary above improves this to $\text{poly}(n, s, t, \log d)$ by using the structural bound to keep the computation in sparse representation.

1.2 Proof Overview

We describe the main ideas behind the algorithm and the analysis. The algorithm is based on the homogeneous descent of [[DOS14](#)]. Their framework gives efficient algorithms for low-degree arithmetic circuits, and in particular, its running time is polynomial in the degree bound. For sparse polynomials, however, the degree can be exponentially larger than its binary encoding, so polynomial dependence on d is not sufficient.

Two additional issues have to be handled in the sparse setting. First, a sparse polynomial need not remain sparse after a shift: even $(x + a)^d$ has $d + 1$ monomials, which is exponential in $\log d$. In several variables the blow-up can be $(d + 1)^n$. Thus we cannot simply expand shifted monomials during the descent. Second, over $\mathbb{Q}$, an arithmetic algorithm is *not enough*: one must prove that a polynomial-height rational shift exists and that the algorithm can output such a small witness.

The [[DOS14](#)] descent. Write $f_i := H^i(f)$ and $g_i := H^i(g)$ for the homogeneous components of degree i . For a vector $\mathbf{v} \in \mathbb{Q}^n$, let

$$D_{\mathbf{v}} := \sum_{\ell=1}^n v_{\ell} \frac{\partial}{\partial x_{\ell}}, \quad f_i^{(r)}(\mathbf{v}, \mathbf{x}) := D_{\mathbf{v}}^r f_i(\mathbf{x}).$$

So, $D_{\mathbf{v}}$ denotes the directional derivative in direction $\mathbf{v}$. Moreover, $f_i^{(1)}(\mathbf{v}, \mathbf{x})$ is the first-order contribution of the degree- i part of f under a shift in direction $\mathbf{v}$, while $f_i^{(r)}(\mathbf{v}, \mathbf{x})$ is the corresponding r th-order contribution. Suppose first that $g(\mathbf{x}) = f(\mathbf{x} + \mathbf{c})$ for some shift $\mathbf{c}$. The homogeneous Taylor expansion is *triangular* with respect to degree: the degree- i part of g receives contributions only from the degree- i and higher homogeneous parts of f . More explicitly,

$$g_i = f_i + f_{i+1}^{(1)}(\mathbf{c}, \mathbf{x}) + \sum_{j=i+2}^d \frac{1}{(j-i)!} f_j^{(j-i)}(\mathbf{c}, \mathbf{x}).$$

The term f_i is known, and the term $f_{i+1}^{(1)}(\mathbf{c}, \mathbf{x})$ is linear in the unknown shift $\mathbf{c}$. The remaining sum is a higher-order tail coming from degrees strictly larger than $i + 1$. This suggests a *top-down descent*. At degree $d - 1$, there is no tail, so

$$g_{d-1} = f_{d-1} + f_d^{(1)}(\mathbf{c}, \mathbf{x})$$

is just a linear system in the coordinates of $\mathbf{c}$. At degree $d - 2$, the equation contains the extra term $\frac{1}{2}f_d^{(2)}(\mathbf{c}, \mathbf{x})$, which looks *quadratic* in $\mathbf{c}$. The key observation of [DOS14] is that this term is already determined once the first derivative of f_d has been matched. Indeed, if two directions $\mathbf{a}$ and $\mathbf{c}$ satisfy

$$f_d^{(1)}(\mathbf{a}, \mathbf{x}) = f_d^{(1)}(\mathbf{c}, \mathbf{x}),$$

then $D_{\mathbf{a}-\mathbf{c}}f_d = 0$. Since constant-coefficient differential operators commute, this implies that all higher directional derivatives of f_d in the directions $\mathbf{a}$ and $\mathbf{c}$ also agree. Thus the degree- $(d - 2)$ equation again has only one new unknown contribution, namely $f_{d-1}^{(1)}(\mathbf{c}, \mathbf{x})$, which is *linear* in $\mathbf{c}$. More generally, at every stage of the descent, all higher-order terms coming from previously processed homogeneous components are already fixed, and the only new condition to impose is a linear equation for the first derivative of the next homogeneous component.

Modification in the sparse setting. We apply this descent only at degrees where a new first-order term can appear. Let

$$d = \delta_1 > \delta_2 > \dots > \delta_m > 0$$

be the nonzero positive homogeneous degrees of f . Since f has sparsity s , we have $m \leq s$. At stage q , the algorithm has already processed $\delta_1, \dots, \delta_{q-1}$ and has a candidate shift $\mathbf{a}_{q-1}$ satisfying

$$f_{\delta_p}^{(1)}(\mathbf{a}_{q-1}, \mathbf{x}) = f_{\delta_p}^{(1)}(\mathbf{c}, \mathbf{x}) \quad \text{for all } p < q.$$

By the derivative observation above, the higher-order contributions of these already processed components are the same for $\mathbf{a}_{q-1}$ and for the true shift $\mathbf{c}$. Therefore the *tail*

$$P_q(\mathbf{x}) := H^{\delta_q-1} \left(H^{\geq \delta_q+1}(f)(\mathbf{x} + \mathbf{a}_{q-1}) \right)$$

is equal to the tail one would obtain from $\mathbf{c}$:

$$P_q(\mathbf{x}) = H^{\delta_q-1} \left(H^{\geq \delta_q+1}(f)(\mathbf{x} + \mathbf{c}) \right).$$

After subtracting this tail, the next condition is again linear:

$$g_{\delta_q-1} = f_{\delta_q-1} + f_{\delta_q}^{(1)}(\mathbf{b}, \mathbf{x}) + P_q(\mathbf{x}).$$

The algorithm adds this linear constraint to the previous ones and chooses a solution $\mathbf{a}_q$. Thus the nonlinear shift-equivalence problem is reduced, one homogeneous degree at a time, to simple linear algebra.

Why this is not yet an efficient algorithm. The discussion above explains why the shift problem can be reduced to a sequence of linear systems. It does not yet explain why these systems can be constructed and eventually solved from the sparse input in polynomial time. The difficulty lies in the computation of the tail P_q . A single monomial $\mathbf{x}^\gamma$, after the shift $\mathbf{x} \mapsto \mathbf{x} + \mathbf{a}$, produces $\prod_{i \in \text{supp}(\mathbf{a})} (\gamma_i + 1)$ many monomials, which may be exponential in the sparse input length. Thus, explicitly forming shifted monomials is too expensive.

One could keep shifted monomials in a circuit representation, but then the algorithm has left the sparse representation. The linear systems in the descent would have entries represented by circuits (of high degree), and constructing or comparing those entries would require PIT for a larger circuit class. The general framework of [DOS14] then gives dependence polynomial in the degree d , or in the relevant circuit/PIT parameters, rather than a polynomial in $\log d$.

There is a second issue over $\mathbb{Q}$. A naive bit-complexity analysis would bound the tail P_q in terms of the height of the current candidate $\mathbf{a}_{q-1}$, since P_q is computed using $\mathbf{a}_{q-1}$. The right-hand side of the next linear system would then have height depending on $\text{Ht}(\mathbf{a}_{q-1})$, and standard bounds for basic rational solutions (defined in [section A, definition A.1](#)) would give a recursive estimate for $\text{Ht}(\mathbf{a}_q)$ in terms of $\text{Ht}(\mathbf{a}_{q-1})$. Iterating such an estimate over up to s stages need not yield the desired polynomial bound in the sparse input parameters. Thus, the tails must be bounded globally, using a fixed small witnessing shift, rather than recursively through the candidates produced by the algorithm.

The structural lemma and proof sketch of [theorem 1.2](#). The main new structural input is that the bad shifted expansions cannot occur along a yes instance. The truncated form of the lemma says the following. Suppose

$$H^{\geq r}(g) = H^{\geq r}(f(\mathbf{x} + \mathbf{a})).$$

Then for every monomial $\mathbf{x}^\gamma$ of $H^{\geq r}(f)$,

$$\text{sp}(H^{\geq r}((\mathbf{x} + \mathbf{a})^\gamma)) \leq st.$$

Thus, once the shifted polynomial agrees with g in degrees at least r , every shifted monomial that contributes to those degrees is itself sparse *after truncation*. In particular, large shifted expansions cannot be hidden by cancellations in the final sparse polynomial.

The proof isolates one shifted monomial from the identity $g(\mathbf{x}) = f(\mathbf{x} + \mathbf{a})$. Let $f(\mathbf{x}) = \sum_{j=1}^s c_j \mathbf{x}^{\alpha_j}$, and relabel the target monomial so that $\gamma = \alpha_1$. We use Euler-type differential operators, defined as follows:

$$E_i = (x_i + a_i) \partial_{x_i}.$$

These operators are diagonal on the shifted monomial basis: $E_i(\mathbf{x} + \mathbf{a})^{\mathbf{e}} = e_i(\mathbf{x} + \mathbf{a})^{\mathbf{e}}$. Hence, for a multi-index $\mathbf{v}$, the following holds: $\mathbf{E}^{\mathbf{v}}(\mathbf{x} + \mathbf{a})^{\alpha_j} = \alpha_j^{\mathbf{v}}(\mathbf{x} + \mathbf{a})^{\alpha_j}$. Thus the operators $\mathbf{E}^{\mathbf{v}}$ act on shifted monomials exactly as evaluation functions on the exponent vectors α_j .

Using interpolation on the finite set of exponent vectors $\{\alpha_1, \dots, \alpha_s\}$, we choose a set T of s multi-indices with the following *cone-closed* property: if $\mathbf{v} \in T$ and $\mathbf{u} \preceq \mathbf{v}$ coordinatewise, then $\mathbf{u} \in T$. The concept of cone-closed basis was introduced in [FGS18]. We also choose coefficients $r_{\mathbf{v}}$ such that

$$\sum_{\mathbf{v} \in T} r_{\mathbf{v}} \alpha_j^{\mathbf{v}} = \begin{cases} 1 & j = 1, \\ 0 & j \neq 1. \end{cases}$$

Now define the isolating operator

$$\mathcal{I}_\gamma := \sum_{\mathbf{v} \in T} r_{\mathbf{v}} \mathbf{E}^{\mathbf{v}}.$$

By construction, $\mathcal{I}_\gamma$ kills all shifted monomials except the one with exponent γ . Therefore $\mathcal{I}_\gamma(g) = c_1(\mathbf{x} + \mathbf{a})^\gamma$. The same operator $\mathcal{I}_\gamma$ can also be viewed as acting on the ordinary sparse expansion of g . This is exactly where *sparsity enters*. Since $E_i = x_i \partial_{x_i} + a_i \partial_{x_i}$, each application of E_i to an ordinary monomial either preserves its exponent or decreases it by one in coordinate i . More generally, applying $E^\mathbf{v}$ to a monomial $\mathbf{x}^\beta$ can only produce monomials of the form $\mathbf{x}^{\beta - \mathbf{u}}$ with $\mathbf{u} \preceq \mathbf{v}$.

Because T is cone-closed, every such possible exponent drop $\mathbf{u}$ is again contained in T . Hence, if $g(\mathbf{x}) = \sum_{k=1}^t d_k \mathbf{x}^{\beta_k}$, then the support of $\mathcal{I}_\gamma(g)$ is contained in $\{\beta_k - \mathbf{u} : k \in [t], \mathbf{u} \in T\}$. This set has size at most $t|T| = st$. Since $\mathcal{I}_\gamma(g)$ is a nonzero scalar multiple of $(\mathbf{x} + \mathbf{a})^\gamma$, the shifted monomial $(\mathbf{x} + \mathbf{a})^\gamma$ itself has support at most st .

The truncated version needed by the algorithm follows from the same argument. Indeed, the operators E_i never increase total degree, so the isolation argument can be applied directly to the high-degree identity

$$H^{\geq r}(g) = H^{\geq r}(f(\mathbf{x} + \mathbf{a})).$$

This gives the same support bound for the truncated shifted monomials $H^{\geq r}((\mathbf{x} + \mathbf{a})^\gamma)$.

This is the point where our proof differs from the general framework of [DOS14]. Sparse polynomials are not closed under arbitrary shifts. The structural lemma saves us here: although an arbitrary shift of a sparse polynomial may be dense, the shifted pieces that arise along a yes instance of the descent are forced to remain sparse. For details, see lemma 3.2.

Computing the tails without full expansion. At the beginning of stage q , the maintained invariant gives

$$H^{\geq \delta_q}(f(\mathbf{x} + \mathbf{a}_{q-1})) = H^{\geq \delta_q}(g).$$

Therefore the structural lemma applies with $r = \delta_q$ and $\mathbf{a} = \mathbf{a}_{q-1}$. Thus, in particular, for every monomial $c_\gamma \mathbf{x}^\gamma$ in $H^{\geq \delta_q}(f)$, the high-degree truncation

$$H^{\geq \delta_q}(c_\gamma(\mathbf{x} + \mathbf{a}_{q-1})^\gamma)$$

has at most st monomials in a yes instance.

The obvious way to use this statement would be to expand $c_\gamma(\mathbf{x} + \mathbf{a}_{q-1})^\gamma$ and then keep only the terms of degree at least δ_q . Unfortunately, this is *not efficient*: the shifted monomial may be dense as a whole, even though its high-degree truncation is sparse. Nor can one simply invoke sparse interpolation as a black box, since the polynomial naturally available for evaluation is the entire shifted monomial, whose lower-degree part may be dense. The sparse object guaranteed by the structural lemma is **only** the high-degree truncation.

Instead, the algorithm enumerates only the exact degree- δ_q layer, which is part of the structurally bounded high-degree truncation. Using a truncated dynamic-programming enumeration, the algorithm either lists all drops that produce degree δ_q , if there are at most st , or detects that there are more than st and rejects. Thus it never enumerates the potentially dense lower-degree part; for details see lemma 4.1.

The desired tail P_q , however, lies one degree lower, in degree $\delta_q - 1$. The algorithm does not enumerate this lower layer directly. Let $j = |\gamma|$ and let

$$C_\gamma(\mathbf{x}) := H^{\delta_q}(c_\gamma(\mathbf{x} + \mathbf{a}_{q-1})^\gamma).$$

Then the degree- $(\delta_q - 1)$ contribution of $c_\gamma(\mathbf{x} + \mathbf{a}_{q-1})^\gamma$ is obtained from the controlled degree δ_q part by

$$H^{\delta_q-1}(c_\gamma(\mathbf{x} + \mathbf{a}_{q-1})^\gamma) = \frac{1}{j - \delta_q + 1} D_{\mathbf{a}_{q-1}} C_\gamma(\mathbf{x}).$$

Thus the algorithm enumerates only this structurally bounded layer and then obtains the one lower layer needed for the descent by a single directional derivative. Summing these contributions over the monomials of $H^{>\delta_q}(f)$ and collecting like terms gives P_q in sparse representation.

Bit complexity assuming a small rational shift. We first explain the bit-complexity analysis under the following assumption: there is a true shift $\mathbf{c} \in S_{f,g}$ whose height is polynomially bounded in the input parameters. Later, we will explain why such a shift always exists over $\mathbb{Q}$ whenever $S_{f,g}$ is nonempty.

Under this assumption, the key point is that the tails are not recursively bounded. As mentioned before, if one does it recursively, then iterating over the stages of the descent would give an exponential bound on the output height. The way around this is to prove that the tails are controlled by a single global witness. Although P_q is computed syntactically using $\mathbf{a}_{q-1}$, the descent invariant gives

$$P_q(\mathbf{x}) = H^{\delta_q-1} \left(H^{\geq \delta_q+1}(f)(\mathbf{x} + \mathbf{c}) \right)$$

for one fixed true shift $\mathbf{c}$. Hence, after collecting terms, the height of P_q is controlled in terms of the input and the height of this same $\mathbf{c}$, not in terms of a sequence of previously computed candidates. Once the tails have polynomial height, the linear systems have polynomial-height coefficients, and choosing basic rational solutions gives polynomial-height candidates; the definition and the height bound are in [section A](#), [definition A.1](#) and [lemma A.1](#). The algorithm also imposes global support checks and coefficient-height checks on the computed tails throughout the descent. These checks are harmless in a yes instance, by the structural lemma and the small-witness bound. On a no instance, the algorithm may reject because one of these checks fails or because one of the linear systems is infeasible. If neither happens, the algorithm obtains a candidate shift $\mathbf{a}$. It then performs a final verification of the identity $f(\mathbf{x} + \mathbf{a}) \equiv g(\mathbf{x})$.

This final check can again be done very efficiently: by the structural lemma, if the identity holds, then every shifted monomial of $f(\mathbf{x} + \mathbf{a})$ has support at most st after truncation with $r = 0$, and hence the whole expansion has support at most s^2t . The algorithm expands these bounded pieces, collects like terms, and compares the resulting sparse polynomial with g . Thus, the algorithm never accepts a false candidate. For the detailed analysis, we refer to [proposition 4.1](#).

Existence of small rational shift. It remains to justify that a small rational shift exists whenever any shift exists. First, if a shift exists over an algebraic extension, then a rational shift exists over $\mathbb{Q}$: the complex shift set $S_{f,g}(\mathbb{C}) = \{\mathbf{a} \in \mathbb{C}^n : f(\mathbf{x} + \mathbf{a}) = g(\mathbf{x})\}$ is an affine subspace, and averaging an algebraic point over its Galois conjugates gives a rational point in the same affine subspace.

The height bound uses the structural lemma once more. Let $\mathbf{c}$ be any rational shift and let $S = \text{supp}(\mathbf{c})$. Then every monomial $\mathbf{x}^\gamma$ of f satisfies $\prod_{i \in S} (\gamma_i + 1) \leq st$. Consequently, the derivatives of f in the variables of S span only a polynomial-dimensional space. Inside this space, the translation $f(\mathbf{x}) \mapsto f(\mathbf{x} + \mathbf{c}) = g(\mathbf{x})$ can be represented by a rational differential operator of polynomial height; see the proof of [theorem 1.3](#) in [section A](#), in particular [equation \(12\)](#). This operator is *unipotent* on the relevant derivative space, so its logarithm is a finite sum. The logarithm recovers the first-order translation direction on f , namely a polynomial of the form $D_{\mathbf{c}}f$.

Solving the linear system $D_{\mathbf{b}}f = D_{\mathbf{c}}f$ and taking a basic rational solution by the [lemma A.1](#) gives a vector $\mathbf{b}$ of polynomial height. By the derivative characterization of shifts from [\[DOS14\]](#), this $\mathbf{b}$ is again a shift from f to g . Thus, whenever Sparse SET is a yes instance over $\mathbb{Q}$, there is a witness of polynomial bit-size, and the algorithm outputs one. For details, refer to [theorem 1.3](#); its proof is deferred to [section A](#).

1.3 Related work

Algorithms related to sparse polynomials. Several basic algorithmic problems have been studied for sparse polynomials, including identity testing, interpolation, factorization, and changes of origin. Sparse identity testing and interpolation go back to the works of Zippel and Ben-Or–Tiwari, and have been followed by several deterministic and randomized algorithms for sparse polynomials and straight-line programs [Zip79, BOT88, KS01, BHLV09, GS09, AGR13]. Sparse factorization has also been extensively studied, beginning with algorithms for bounded-degree factors of sparse polynomials and followed by more recent structural and algorithmic results for factors of sparse polynomials [Jr.99, AKS06, BSV20, CS26, BKRS26]. For a survey-style introduction to the algorithmic landscape for sparse polynomials, including arithmetic, interpolation, and factorization, see Roche [Roc18].

Sparse shifts and sparsification. The algorithmic study of shifts of sparse polynomials goes back to Grigoriev and Karpinski, Lakshman and Saunders, and Grigoriev and Lakshman [GK93, LS94, GL95]. Their work considered questions such as identity testing, interpolation, and finding changes of origin under which a polynomial has a smaller sparsity. Subsequent work studied algorithms for computing sparsest shifted representations and for interpolation in shifted-sparse models [GKsL03, GR10].

Target-free sparse shifts and sparsification. A related but different problem is the *target-free sparse-shift* problem: given a polynomial f , decide whether there exists a shift $\mathbf{a}$ such that $\text{sp}(f(\mathbf{x} + \mathbf{a}))$ is small, or smaller than $\text{sp}(f)$. Recent work shows that this sparsification problem is computationally hard [CGS23, BDJ26]. This should be contrasted with Sparse SET: here both endpoints f and g are part of the input, and the question is whether $g(\mathbf{x}) = f(\mathbf{x} + \mathbf{a})$ for some shift $\mathbf{a}$. Thus, our problem is target-given rather than target-free.

Polynomial equivalence and orbit testing. There is also a broader polynomial-equivalence literature in which one allows general linear or affine changes of variables. In this setting one asks whether there are $A \in \text{GL}_n(\mathbb{Q})$ and $\mathbf{b} \in \mathbb{Q}^n$ such that $f(A\mathbf{x} + \mathbf{b}) \equiv g(\mathbf{x})$, or whether the orbit of a given polynomial intersects a prescribed structured class. Polynomial equivalence is often viewed as an algebraic analogue of graph isomorphism: already, the equivalence of homogeneous cubic forms captures graph isomorphism [AS06, Kay11]. Kayal studied efficient algorithms for special cases of polynomial equivalence and later affine projections of polynomials [Kay11, Kay12].

More recently, there has been work on equivalence testing for structured polynomial families, including read-once arithmetic formulas, design polynomials, and the Hamiltonian cycle polynomial [GST23, BDS24, Dew26]. Baraskar, Dewan, Saha, and Sinha also proved NP-hardness of testing equivalence to sparse polynomials and to constant-support polynomials under affine transformations [BDSS24]. Sparse SET is the translation-only, target-given special case, namely the case $A = I$ with both sparse endpoints f and g given as input.

Shift equivalence testing. Grigoriev studied SET itself as a decision and search problem for densely represented polynomials and gave deterministic, randomized, and quantum algorithms in different characteristics [Gri97]. In characteristic zero, the deterministic algorithm runs in time polynomial in the dense representation. For n -variate polynomials of total degree at most d , the dense representation has size $\binom{n+d}{d}$ and the running time is $n^{O(d)}$. Padding a sparse input with

zero coefficients therefore does not give an efficient algorithm when the exponents are encoded in binary.

[DOS14] result and implication for sparse SET. Dvir, Oliveira, and Shpilka [DOS14] studied SET for arithmetic circuits, in both the white-box and black-box models [DOS14]. They gave a randomized algorithm running in time polynomial in the circuit size, the number of variables, and the degree, and showed that the randomness in their algorithm is used only for polynomial identity testing. Consequently, derandomizing PIT would derandomize SET in their framework. They also obtained a general deterministic reduction for restricted circuit classes. In one form, their result applies to classes $\mathcal{M}_1, \mathcal{M}_2$ such that $\mathcal{M}_1$ is closed under taking homogeneous components and first-order directional derivatives, while $\mathcal{M}_2$ is closed under shifts. The number of arithmetic operations is polynomial in n, d , and in the size of a hitting set, or of a white-box PIT algorithm, for the class of differences $\mathcal{M}_1 - \mathcal{M}_2$. They also pointed out an analogy with algorithms for black-box groups, such as Sims' algorithm: their procedure can be viewed as finding the affine space of all shifts from f to g , rather than merely deciding whether this space is empty [Ser03].

Although Section 6 of [DOS14] observes that the sparse class is not closed under shifts and gives no direct sparse-class algorithm, a low-degree bound follows indirectly via ROABPs. A sparse polynomial

$$f(\mathbf{x}) = \sum_{i=1}^s c_i \prod_{j=1}^n x_j^{\alpha_{i,j}}$$

is a sum of products of univariate polynomials, and hence has a read-once oblivious algebraic branching program of width at most s in any variable order. Choose the same variable order for both inputs. After a shift,

$$f(\mathbf{x} + \mathbf{a}) = \sum_{i=1}^s c_i \prod_{j=1}^n (x_j + a_j)^{\alpha_{i,j}},$$

it still has such a representation of size $\text{poly}(n, s, d)$. Therefore, the general white-box theorem of [DOS14], together with white-box ROABP PIT [RS05], yields a deterministic algorithm for explicit sparse inputs using $\text{poly}(n, s, t, d)$ arithmetic operations. This is an indirect consequence rather than a sparse-case result stated in DOS'14. It is efficient in the ordinary low-degree regime, but it is not efficient in the high-degree sparsity setting: the dependence is polynomial in d , whereas the sparse input length is polynomial in $\log d$. Moreover, this framework does not by itself address bit complexity over $\mathbb{Q}$, the existence of a rational shift of polynomial height, or the problem of outputting such a small witness.

2 Preliminaries

Notation.

1. **Monomials.** For vectors $\mathbf{z} = (z_1, \dots, z_n)$ and $\mathbf{v} \in \mathbb{N}^n$, we write

$$\mathbf{z}^{\mathbf{v}} = z_1^{v_1} \cdots z_n^{v_n}.$$

When $\mathbf{z}$ is a vector of variables, then $\mathbf{z}^{\mathbf{v}}$ is the monomial with exponent vector given by $\mathbf{v}$. When $\mathbf{z}$ is a vector of field elements, then it is the monomial evaluated at the given point.

2. **Degree and derivatives.** We also define $|\mathbf{v}| := v_1 + \cdots + v_n$, $\mathbf{v}! := v_1! \cdots v_n!$, and $\partial^{\mathbf{v}} = \partial_{x_1}^{v_1} \cdots \partial_{x_n}^{v_n}$, where ∂_{x_i} is the partial derivative taken with respect to the variable x_i .

3. **Partial order.** We use the coordinate-wise partial order $\preceq$ on $\mathbb{N}^n$: for $\mathbf{v}, \mathbf{u} \in \mathbb{N}^n$, we write $\mathbf{v} \preceq \mathbf{u}$ if and only if $v_i \leq u_i$ for every $i \in [n]$. A set $T \subseteq \mathbb{N}^n$ is called *cone-closed* if, whenever $\mathbf{u} \in T$ and $\mathbf{v} \preceq \mathbf{u}$, we also have $\mathbf{v} \in T$.
4. **Polynomial notation.** For a polynomial f , $H^\ell(f)$ denotes the homogeneous part of degree ℓ of f , that is, the sum of all monomials of f that have degree ℓ . For two polynomials $f, g \in \mathbb{K}[\mathbf{x}]$, $S_{f,g}(\mathbb{K})$ denotes the set of all shift vectors $\mathbf{a} \in \mathbb{K}^n$ such that $f(\mathbf{x} + \mathbf{a}) = g(\mathbf{x})$. The set of all shifts $\mathbf{a} \in \mathbb{K}^n$ that leave f invariant, that is, $f(\mathbf{x} + \mathbf{a}) = f(\mathbf{x})$, is denoted by $S_f(\mathbb{K})$. Over $\mathbb{Q}$, we write $S_{f,g}$ for $S_{f,g}(\mathbb{Q})$. For a polynomial $f(\mathbf{x})$, $\text{sp}(f)$ denotes the sparsity of f , that is, the number of nonzero monomials in f . For a polynomial p , we also write

$$H^{\geq r}(p) := \sum_{\ell \geq r} H^\ell(p).$$

5. **Height.** A rational number r is said to have height at most τ if it can be written as $r = p/q$, where $p, q \in \mathbb{Z}$, $q \neq 0$, and $|p|, |q| \leq 2^\tau$. This is denoted by $\text{Ht}(r) \leq \tau$. The height of a sparse polynomial, vector, or matrix is the maximum height of its coefficients or entries, and is denoted by $\text{Ht}(\cdot)$.

Remark 2.1. In parts of the literature, this quantity is called logarithmic height. In this paper we simply call it height. Up to constant factors it is the bit-size of the numerators and denominators, so we use height and bit-size interchangeably in polynomial bounds.

Definition 2.1 (Directional derivative). For $r \geq 1$, the order- r directional derivative of $f(\mathbf{x}) \in \mathbb{Q}[\mathbf{x}]$ in direction $\mathbf{a} \in \mathbb{Q}^n$ is

$$f^{(r)}(\mathbf{a}, \mathbf{x}) = \sum_{\mathbf{e} \in [n]^r} \left(\prod_{i=1}^r a_{e_i} \right) \frac{\partial^r f}{\partial x_{e_1} \cdots \partial x_{e_r}}.$$

We also define $f^{(0)}(\mathbf{a}, \mathbf{x}) = f(\mathbf{x})$. For each homogeneous component $H^\ell(f)$ of f , define

$$f_\ell^{(r)}(\mathbf{a}, \mathbf{x}) = \sum_{\mathbf{e} \in [n]^r} \left(\prod_{i=1}^r a_{e_i} \right) \frac{\partial^r H^\ell(f)}{\partial x_{e_1} \cdots \partial x_{e_r}}.$$

The following lemma, due to Dvir, Oliveira and Shpilka [DOS14], states that if the first directional derivatives are identical for two directions $\mathbf{a}$ and $\mathbf{b}$, then all higher order directional derivatives are identical, too.

Lemma 2.1 (Derivative equality under shifts, [DOS14]). *Let $f \in \mathbb{Q}[\mathbf{x}]$ and $\mathbf{a}, \mathbf{b} \in \mathbb{Q}^n$. If $f^{(1)}(\mathbf{a}, \mathbf{x}) = f^{(1)}(\mathbf{b}, \mathbf{x})$, then $f^{(r)}(\mathbf{a}, \mathbf{x}) = f^{(r)}(\mathbf{b}, \mathbf{x})$ for all $r \geq 1$.*

The next lemma allows us to write the homogeneous parts of a shifted polynomial in terms of their directional derivatives.

Lemma 2.2 (Homogeneous Taylor expansion, [DOS14]). *Let $\mathbf{a} \in \mathbb{Q}^n$ and let $f \in \mathbb{Q}[\mathbf{x}]$ have degree d . Then for $0 \leq i \leq d$,*

$$H^i(f(\mathbf{x} + \mathbf{a})) = \sum_{j=i}^d \frac{1}{(j-i)!} f_j^{(j-i)}(\mathbf{a}, \mathbf{x}).$$

Finally, if $\mathbf{a}$ is a shift that transforms f into g , then $\mathbf{b}$ transforms f into g , too, if and only if the first directional derivatives are identical.

Lemma 2.3 (Shift space via first derivatives, [DOS14]). *Let $f, g \in \mathbb{Q}[\mathbf{x}]$ and $\mathbf{a}, \mathbf{b} \in \mathbb{Q}^n$. If $\mathbf{a} \in S_{f,g}$, then $\mathbf{b} \in S_{f,g}$ if and only if $f_i^{(1)}(\mathbf{b}, \mathbf{x}) = f_i^{(1)}(\mathbf{a}, \mathbf{x})$ for $0 \leq i \leq d$.*

For a polynomial $f(\mathbf{x})$ of degree d , a shift $\mathbf{a} \in \mathbb{Q}^n$, and $0 \leq i \leq d$, we define:

$$P_{i,\mathbf{a}}(f) := H^i \left(H^{\geq i+2}(f)(\mathbf{x} + \mathbf{a}) \right).$$

Lemma 2.4. *For any polynomial $f \in \mathbb{Q}[\mathbf{x}]$, we have*

$$P_{i,\mathbf{a}}(f) = H^i \left(H^{\geq i+2}(f)(\mathbf{x} + \mathbf{a}) \right) = \sum_{j=i+2}^d \frac{1}{(j-i)!} f_j^{(j-i)}(\mathbf{a}, \mathbf{x}).$$

Proof. We have

$$\begin{aligned} H^i \left(H^{\geq i+2}(f)(\mathbf{x} + \mathbf{a}) \right) &= H^i \left(\sum_{j=i+2}^d H^j(f)(\mathbf{x} + \mathbf{a}) \right) \\ &= \sum_{j=i+2}^d H^i \left(H^j(f)(\mathbf{x} + \mathbf{a}) \right) \end{aligned}$$

Applying [lemma 2.2](#) on each $H^j(f)$ gives an expansion of $H^i \left(H^j(f)(\mathbf{x} + \mathbf{a}) \right)$ in terms of the directional derivatives of the homogeneous components of $H^j(f)$. Since $H^j(f) = f_j$ has only one homogeneous component, we get a single term in the expansion:

$$H^i \left(H^j(f)(\mathbf{x} + \mathbf{a}) \right) = \frac{1}{(j-i)!} f_j^{(j-i)}(\mathbf{a}, \mathbf{x}).$$

Plugging this into the previous equation proves the lemma. □

2.1 Properties of $S_{f,g}$

Recall that for $f, g \in \mathbb{Q}[\mathbf{x}]$,

$$S_{f,g}(\mathbb{C}) = \{\mathbf{a} \in \mathbb{C}^n : f(\mathbf{x} + \mathbf{a}) = g(\mathbf{x})\}$$

where the equality is understood as an identity in $\mathbb{C}[\mathbf{x}]$. We first write two elementary facts about this shift set and then use them for the rational bit-complexity analysis.

Lemma 2.5 ([\[DOS14, Observation 4.2, Lemma 4.5\]](#)). *Let $f \in \mathbb{C}[\mathbf{x}]$, and define*

$$S_f(\mathbb{C}) := \{\mathbf{u} \in \mathbb{C}^n : f(\mathbf{x} + \mathbf{u}) = f(\mathbf{x})\}.$$

Then $S_f(\mathbb{C})$ is a linear subspace of $\mathbb{C}^n$, given by $S_f(\mathbb{C}) = \{\mathbf{a} \in \mathbb{C}^n \mid \sum_{i=1}^n a_i \frac{\partial f}{\partial x_i} = 0\}$.

Lemma 2.6 ([\[DOS14, Lemma 4.4\]](#)). *For $f, g \in \mathbb{C}[\mathbf{x}]$, the set $S_{f,g}(\mathbb{C})$ is either empty or an affine subspace of $\mathbb{C}^n$. More precisely, if $\mathbf{a}_0 \in S_{f,g}(\mathbb{C})$, then*

$$S_{f,g}(\mathbb{C}) = \mathbf{a}_0 + S_f(\mathbb{C}).$$

Proposition 2.1 (Existence of shift in $\mathbb{Q}^n$). *Let $f, g \in \mathbb{Q}[\mathbf{x}]$. If*

$$S_{f,g}(\mathbb{C}) := \{\mathbf{a} \in \mathbb{C}^n : f(\mathbf{x} + \mathbf{a}) = g(\mathbf{x})\}$$

is nonempty, then $S_{f,g}(\mathbb{Q})$ contains a rational point. Equivalently, there exists $\mathbf{q} \in \mathbb{Q}^n$ such that

$$f(\mathbf{x} + \mathbf{q}) = g(\mathbf{x}).$$

The proof is deferred to [section A](#).

3 Structural Sparsity Bounds

We first prove the structural statement that drives the algorithm. The proof needs to isolate one chosen shifted monomial from $g(\mathbf{x}) = f(\mathbf{x} + \mathbf{a})$ while keeping control over what happens when the same operator is applied to the sparse monomial expansion of g . The next lemma provides exactly the indexing set needed for this. It gives an interpolation basis for the exponent vectors of the monomials of f , but with the additional cone-closed property: if a multi-index is selected, then every smaller multi-index is selected as well. This closure is what later prevents the differential operator from creating too many possible output monomials.

3.1 Cone-closed basis

Lemma 3.1 (Cone-closed interpolation basis). *Let*

$$B = \{\mathbf{b}_1, \dots, \mathbf{b}_s\} \subseteq \mathbb{N}^n$$

be a finite set of exponent vectors. Assume that the elements of B remain distinct as points of $\mathbb{Q}^n$. For every $\mathbf{v} \in \mathbb{N}^n$, define $M_{\mathbf{v}} \in \mathbb{Q}^s$ by

$$(M_{\mathbf{v}})_j := \mathbf{b}_j^{\mathbf{v}} \quad (j \in [s]).$$

Then there is a cone-closed set $T \subseteq \mathbb{N}^n$ such that $\{M_{\mathbf{v}} : \mathbf{v} \in T\}$ is a basis of $\mathbb{Q}^s$. In particular, $|T| = s$.

Proof. Let $\text{ev}_B : \mathbb{Q}[z_1, \dots, z_n] \rightarrow \mathbb{Q}^s$ be the evaluation map

$$P \mapsto (P(\mathbf{b}_1), \dots, P(\mathbf{b}_s)).$$

We first check that this map is surjective. For each $m \in [s]$, let

$$\mathfrak{m}_m = (z_1 - b_{m,1}, \dots, z_n - b_{m,n})$$

be the ideal of polynomials vanishing at $\mathbf{b}_m$. Note that reducing a polynomial P modulo $\mathfrak{m}_m$ is nothing but evaluating P at the point $\mathbf{b}_m$.

The ideals $\mathfrak{m}_m$ are maximal, that is, they are not contained in any other proper ideal. Since the points $\mathbf{b}_1, \dots, \mathbf{b}_s$ are pairwise distinct, the ideals are pairwise coprime. By the Chinese remainder theorem, the quotient map

$$\mathbb{Q}[z_1, \dots, z_n] \rightarrow \prod_{m=1}^s \mathbb{Q}[z_1, \dots, z_n] / \mathfrak{m}_m \cong \mathbb{Q}^s$$

is surjective, and this quotient map is exactly ev_B . The image of the monomial $\mathbf{z}^{\mathbf{v}}$ under this map is exactly $M_{\mathbf{v}}$, so the family $\{M_{\mathbf{v}} : \mathbf{v} \in \mathbb{N}^n\}$ spans $\mathbb{Q}^s$.

Now fix the graded lexicographic order $\leq_{\text{grlex}}$ on $\mathbb{N}^n$: vectors are ordered first by total degree $|\mathbf{v}|$ and then ties are broken lexicographically. This order is a well-order and refines $\preceq$, meaning that $\mathbf{v} \preceq \mathbf{u}$ implies $\mathbf{v} \leq_{\text{grlex}} \mathbf{u}$. Moreover, it is compatible with addition: if $\mathbf{v} <_{\text{grlex}} \mathbf{u}$, then $\mathbf{v} + \mathbf{w} <_{\text{grlex}} \mathbf{u} + \mathbf{w}$ for every $\mathbf{w} \in \mathbb{N}^n$.

We construct T by choosing only dimension-increasing vectors. Start with $T_0 = \emptyset$. If the current span is not all of $\mathbb{Q}^s$, then choose $\mathbf{u}_{m+1}$ to be the least vector, in graded lexicographic order, such that

$$M_{\mathbf{u}_{m+1}} \notin \text{span}\{M_{\mathbf{v}} : \mathbf{v} \in T_m\},$$

and set $T_{m+1} = T_m \cup \{\mathbf{u}_{m+1}\}$. Such a vector exists whenever the current span is not all of $\mathbb{Q}^s$, because the full family $\{M_{\mathbf{v}} : \mathbf{v} \in \mathbb{N}^n\}$ spans $\mathbb{Q}^s$. Each step increases the dimension of the span by one, so after exactly s steps the chosen vectors form a basis of $\mathbb{Q}^s$. Let $T = T_s$.

When $\mathbf{u}_{m+1}$ is chosen, minimality implies that every $M_{\mathbf{q}}$ with $\mathbf{q} <_{\text{grlex}} \mathbf{u}_{m+1}$ already lies in $\text{span}\{M_{\mathbf{v}} : \mathbf{v} \in T_m\}$. After adding $\mathbf{u}_{m+1}$, the current span therefore contains every $M_{\mathbf{q}}$ with $\mathbf{q} \leq_{\text{grlex}} \mathbf{u}_{m+1}$. In particular, no smaller vector can become rank-increasing later, so the selected vectors are increasing in graded lexicographic order.

We now show that T is cone-closed. Suppose $\mathbf{u} \in T$, $\mathbf{v} \preceq \mathbf{u}$, and $\mathbf{v} \notin T$. Since grlex refines $\preceq$, we have $\mathbf{v} <_{\text{grlex}} \mathbf{u}$. Write $\mathbf{u} = \mathbf{u}_{m+1}$ (so $\mathbf{u}$ is chosen at step $m+1$). Because $\mathbf{v}$ occurs earlier in the grlex order and was not chosen when it was encountered, we have a dependence

$$M_{\mathbf{v}} = \sum_{\mathbf{w} \in T_m, \mathbf{w} <_{\text{grlex}} \mathbf{v}} \lambda_{\mathbf{w}} M_{\mathbf{w}}.$$

Since $M_{\mathbf{r}} \odot M_{\mathbf{q}} = M_{\mathbf{r}+\mathbf{q}}$ for the Hadamard product $\odot$, which is defined as the pointwise product of the vectors, taking Hadamard product with $M_{\mathbf{u}-\mathbf{v}}$ gives

$$M_{\mathbf{u}} = M_{\mathbf{u}-\mathbf{v}} \odot M_{\mathbf{v}} = \sum_{\mathbf{w} \in T_m, \mathbf{w} <_{\text{grlex}} \mathbf{v}} \lambda_{\mathbf{w}} M_{\mathbf{w}+\mathbf{u}-\mathbf{v}}.$$

Because graded lexicographic order is compatible with addition and $\mathbf{w} <_{\text{grlex}} \mathbf{v}$, each index $\mathbf{w} + \mathbf{u} - \mathbf{v}$ satisfies $\mathbf{w} + \mathbf{u} - \mathbf{v} <_{\text{grlex}} \mathbf{u}$. By minimality of $\mathbf{u}_{m+1}$, every such $M_{\mathbf{w}+\mathbf{u}-\mathbf{v}}$ already lies in $\text{span}\{M_{\mathbf{q}} : \mathbf{q} \in T_m\}$. Hence $M_{\mathbf{u}} \in \text{span}\{M_{\mathbf{q}} : \mathbf{q} \in T_m\}$, contradicting the choice of $\mathbf{u}_{m+1}$ as rank-increasing. Therefore no such $\mathbf{v}$ exists, and T is cone-closed. $\square$

3.2 Structural lemma

Lemma 3.2. *Let $f, g \in \mathbb{Q}[\mathbf{x}]$ be polynomials of degree d with sparsities s and t , respectively. Let $\mathbf{a} \in \mathbb{Q}^n$, and suppose that*

$$H^{\geq r}(g(\mathbf{x})) = H^{\geq r}(f(\mathbf{x} + \mathbf{a})).$$

Then, for every monomial $\mathbf{x}^{\gamma}$ appearing in $H^{\geq r}(f)$,

$$\text{sp}(H^{\geq r}((\mathbf{x} + \mathbf{a})^{\gamma})) \leq st.$$

Proof. Write

$$f(\mathbf{x}) = \sum_{j=1}^s d_j \mathbf{x}^{\alpha_j} \quad \text{and} \quad g(\mathbf{x}) = \sum_{k=1}^t c_k \mathbf{x}^{\beta_k},$$

where all displayed coefficients are nonzero. Fix a monomial $\mathbf{x}^{\gamma}$ appearing in $H^{\geq r}(f)$, and relabel the monomials of f so that $\alpha_1 = \gamma$.

Since the exponent vectors are distinct integer vectors, the exponent vectors of the monomials of f remain distinct as points of $\mathbb{Q}^n$. Apply [lemma 3.1](#) to these exponent vectors. We get a cone-closed set $T \subseteq \mathbb{N}^n$ with $|T| = s$ such that the vectors $M_{\mathbf{v}}$, $\mathbf{v} \in T$, defined by

$$(M_{\mathbf{v}})_j = \alpha_j^{\mathbf{v}} \quad (j \in [s], \mathbf{v} \in T)$$

form a basis of $\mathbb{Q}^s$. Hence there are coefficients $r_{\mathbf{v}}$, $\mathbf{v} \in T$, isolating the first coordinate:

$$\sum_{\mathbf{v} \in T} r_{\mathbf{v}} \alpha_j^{\mathbf{v}} = \begin{cases} 1 & \text{if } j = 1, \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

Define the operators

$$E_i = (x_i + a_i) \frac{\partial}{\partial x_i} \quad \text{and} \quad \mathbf{E}^{\mathbf{v}} = \prod_{i \in [n]} E_i^{v_i}.$$

Since $E_i(\mathbf{x} + \mathbf{a})^{\mathbf{e}} = e_i(\mathbf{x} + \mathbf{a})^{\mathbf{e}}$, we have

$$\mathbf{E}^{\mathbf{v}}(\mathbf{x} + \mathbf{a})^{\mathbf{e}} = \mathbf{e}^{\mathbf{v}}(\mathbf{x} + \mathbf{a})^{\mathbf{e}}.$$

Define

$$\mathcal{I}_{\gamma} = \sum_{\mathbf{v} \in T} r_{\mathbf{v}} \mathbf{E}^{\mathbf{v}}.$$

The operator $\mathcal{I}_{\gamma}$ never increases total degree: each $E_i = x_i \partial_{x_i} + a_i \partial_{x_i}$ preserves the degree or decreases it by one. Thus, for every polynomial h ,

$$H^{\geq r}(\mathcal{I}_{\gamma}(h)) = H^{\geq r}(\mathcal{I}_{\gamma}(H^{\geq r}(h))).$$

Using the assumed equality $H^{\geq r}(g) = H^{\geq r}(f(\mathbf{x} + \mathbf{a}))$, we get

$$H^{\geq r}(\mathcal{I}_{\gamma}(g)) = H^{\geq r}(\mathcal{I}_{\gamma}(f(\mathbf{x} + \mathbf{a}))).$$

On shifted monomials, $\mathcal{I}_{\gamma}$ acts diagonally. Indeed, $f(\mathbf{x} + \mathbf{a}) = \sum_{j=1}^s d_j (\mathbf{x} + \mathbf{a})^{\alpha_j}$, so

$$\begin{aligned} \mathcal{I}_{\gamma}(f(\mathbf{x} + \mathbf{a})) &= \sum_{j=1}^s d_j \left(\sum_{\mathbf{v} \in T} r_{\mathbf{v}} \alpha_j^{\mathbf{v}} \right) (\mathbf{x} + \mathbf{a})^{\alpha_j} \\ &= d_1 (\mathbf{x} + \mathbf{a})^{\gamma}, \end{aligned}$$

where the last equality follows from [equation \(1\)](#). Since $d_1 \neq 0$, multiplication by d_1 does not change sparsity. Therefore,

$$\text{sp}(H^{\geq r}(\mathcal{I}_{\gamma}(g))) = \text{sp}(H^{\geq r}((\mathbf{x} + \mathbf{a})^{\gamma})).$$

It remains to bound $\text{sp}(H^{\geq r}(\mathcal{I}_{\gamma}(g)))$. For $\mathbf{v} \in T$ and a monomial $\mathbf{x}^{\beta_k}$ of g , applying $\mathbf{E}^{\mathbf{v}}$ can only change the exponent to $\beta_k - \mathbf{u}$ for some $\mathbf{u} \preceq \mathbf{v}$, so its support is contained in monomials of the form

$$\mathbf{x}^{\beta_k - \mathbf{u}} \quad \text{with} \quad \mathbf{u} \preceq \mathbf{v}, \mathbf{u} \preceq \beta_k.$$

Because T is cone-closed, every such $\mathbf{u}$ lies in T . Hence $H^{\geq r}(\mathcal{I}_{\gamma}(g))$ is supported on monomials of the form

$$\mathbf{x}^{\beta_k - \mathbf{u}} \quad \text{where} \quad k \in [t], \mathbf{u} \in T.$$

There are at most $t|T| = st$ such monomials. Therefore

$$\text{sp}(H^{\geq r}((\mathbf{x} + \mathbf{a})^{\gamma})) \leq st.$$

□

Taking $r = 0$ in [lemma 3.2](#) proves the f -monomial part of [theorem 1.2](#). The same statement for monomials of g follows by applying the lemma to the identity $f(\mathbf{x}) = g(\mathbf{x} - \mathbf{a})$.

Corollary 1.1 (White-box PIT for shifted-sparse polynomials). *Let $f, g \in \mathbb{Q}[x_1, \dots, x_n]$ be sparse polynomials of sparsities s and t and degree at most d , and let $\mathbf{a}, \mathbf{b} \in \mathbb{Q}^n$ be given. There is a deterministic algorithm using $\text{poly}(n, s, t, \log d)$ arithmetic operations that decides whether*

$$f(\mathbf{x} + \mathbf{a}) \equiv g(\mathbf{x} + \mathbf{b}).$$

Further, if the input coefficients as well as the vectors $\mathbf{a}, \mathbf{b}$ have height at most τ , then the algorithm can be implemented using $\text{poly}(n, s, t, \tau, \log d)$ bit operations.

Proof of the shifted-sparse PIT corollary, [corollary 1.1](#). The identity

$$f(\mathbf{x} + \mathbf{a}) \equiv g(\mathbf{x} + \mathbf{b})$$

is equivalent, after the change of variables $\mathbf{y} = \mathbf{x} + \mathbf{b}$, to

$$f(\mathbf{y} + \mathbf{c}) \equiv g(\mathbf{y}), \quad \text{where } \mathbf{c} = \mathbf{a} - \mathbf{b}.$$

It is therefore enough to verify identities of the form $f(\mathbf{x} + \mathbf{c}) \equiv g(\mathbf{x})$.

For each monomial $c_\gamma \mathbf{x}^\gamma$ of f , compute

$$N_\gamma = \prod_{i \in \text{supp}(\mathbf{c})} (\gamma_i + 1),$$

but stop the multiplication once the value exceeds st . If $N_\gamma > st$ for some monomial, reject. This is sound: if the identity were true, then [theorem 1.2](#) would imply $N_\gamma \leq st$ for every monomial of f .

Otherwise every shifted monomial has at most st terms. Enumerate all vectors $\mathbf{v} \preceq \gamma$ with $\text{supp}(\mathbf{v}) \subseteq \text{supp}(\mathbf{c})$, form the terms

$$c_\gamma \prod_{i=1}^n \binom{\gamma_i}{v_i} (\mathbf{c}_i)^{v_i} \mathbf{x}^{\gamma - \mathbf{v}},$$

collect like terms over all monomials of f , and compare the resulting sparse polynomial with the sparse representation of g . The enumeration size is at most $s \cdot st$, and all exponent arithmetic uses numbers of bit-length $O(\log d)$, so the arithmetic operation count is $\text{poly}(n, s, t, \log d)$. Over $\mathbb{Q}$, if the input coefficients and the shift coordinates have polynomial bit-size, the same computation has polynomial bit complexity, since every nonzero enumerated term has $|\mathbf{v}| \leq st - 1$. $\square$

3.3 Height bound for computations in the algorithm

We now fix the single height cap used by the sparse tail computation and by the main algorithm. In the height and bit-complexity statements below, every polynomial bound denoted by $\text{poly}(\cdot)$ is effective. Whenever the algorithm uses such a bound as a height cap, one can replace the notation by a fixed explicit polynomial in the displayed parameters, chosen large enough to dominate the determinant bounds and sparse-expansion bounds proved below. Fix once and for all a sufficiently large absolute constant C and set

$$\mathcal{B}_{\text{glob}} = (nst(\tau + 1) \log(d + 2))^C.$$

We choose C large enough so that this single cap dominates all height bounds used below, including the sparse-expansion bounds for the intermediate tails and the determinant bounds for the basic rational solutions in the yes-instance analysis. Thus every explicit height check in the algorithm can be implemented by comparing against $\mathcal{B}_{\text{glob}}$.

Proposition 3.1. *With C chosen sufficiently large in the definition of $\mathcal{B}_{\text{glob}}$, the following holds. Suppose $f, g \in \mathbb{Q}[\mathbf{x}]$ are polynomials of degree d with $\text{Ht}(f), \text{Ht}(g) \leq \tau$ and $\text{sp}(f) = s$, $\text{sp}(g) = t$. Suppose $g(\mathbf{x}) = f(\mathbf{x} + \mathbf{c})$ for a shift $\mathbf{c} \in \mathbb{Q}^n$ whose coordinates satisfy the polynomial height bound from [theorem 1.3](#). Then, for every $0 \leq i \leq d - 1$,*

$$\text{sp}(P_{i,\mathbf{c}}(f)) \leq s^2 t, \quad (2)$$

$$\text{Ht}(P_{i,\mathbf{c}}(f)) \leq \mathcal{B}_{\text{glob}}. \quad (3)$$

Proof. Write

$$f(\mathbf{x}) = \sum_{\ell=1}^s \lambda_{\ell} \mathbf{x}^{\gamma_{\ell}}, \quad \lambda_{\ell} \neq 0.$$

By [lemma 2.4](#),

$$P_{i,\mathbf{c}}(f) = H^i \left(H^{\geq i+2}(f)(\mathbf{x} + \mathbf{c}) \right) = H^i \left(\sum_{\substack{\ell \in [s] \\ |\gamma_{\ell}| \geq i+2}} \lambda_{\ell} (\mathbf{x} + \mathbf{c})^{\gamma_{\ell}} \right).$$

Since $g(\mathbf{x}) = f(\mathbf{x} + \mathbf{c})$, [lemma 3.2](#) with $r = 0$ and $\mathbf{a} = \mathbf{c}$ implies that

$$\text{sp}((\mathbf{x} + \mathbf{c})^{\gamma_{\ell}}) \leq st \quad \text{for every } \ell \in [s].$$

Taking a homogeneous part cannot increase support. Therefore each input monomial contributes at most st monomials to $P_{i,\mathbf{c}}$, and after collecting like terms

$$\text{sp}(P_{i,\mathbf{c}}(f)) \leq s \cdot st = s^2 t.$$

It remains to bound coefficient heights. A nonzero term of $(\mathbf{x} + \mathbf{c})^{\gamma_{\ell}}$ is indexed by a vector $\mathbf{v} \preceq \gamma_{\ell}$: it is obtained by replacing v_r copies of x_r by c_r for each $r \in [n]$, and has coefficient

$$\lambda_{\ell} \prod_{r=1}^n \binom{\gamma_{\ell,r}}{v_r} c_r^{v_r}.$$

If this term is nonzero, then $\text{supp}(\mathbf{v}) \subseteq \text{supp}(\mathbf{c})$. For every $\mathbf{u} \preceq \mathbf{v}$ along a chain from $\mathbf{0}$ to $\mathbf{v}$, the corresponding term in $(\mathbf{x} + \mathbf{c})^{\gamma_{\ell}}$ is also nonzero and distinct. Hence the full expansion has at least $|\mathbf{v}| + 1$ nonzero terms. Since its sparsity is at most st as noted above, we get

$$|\mathbf{v}| \leq st - 1.$$

Thus only binomial coefficients $\binom{\gamma_{\ell,r}}{v_r}$ with $|\mathbf{v}| \leq st - 1$ can occur in a nonzero term contributing to $P_{i,\mathbf{c}}(f)$. Their product has height $O(st \log d)$, and $\prod_r c_r^{v_r}$ has polynomial height by [theorem 1.3](#). Multiplying by the input coefficient λ_{ℓ} gives height

$$\text{poly}(n, s, t, \tau, \log d)$$

for each generated term. A coefficient of $P_{i,\mathbf{c}}(f)$ is a sum of at most s such rational numbers, so its height is at most $\text{poly}(n, s, t, \tau, \log d)$. We can choose C large enough so that this height is bounded by $\mathcal{B}_{\text{glob}}$. $\square$

4 Main Algorithm and Proof of Correctness

Before running the descent, handle the trivial cases as follows. If both polynomials are constant, return 0 when they are equal and return FAIL otherwise. If exactly one polynomial is constant, or if the two nonconstant polynomials have different total degrees, return FAIL, because translations preserve total degree.

Algorithm 1 Main Algorithm for Sparse Shift Equivalence

Require: Sparse representations of nonconstant polynomials $f, g \in \mathbb{Q}[x_1, \dots, x_n]$ with $\deg(f) = \deg(g) = d$, $\text{sp}(f) = s$, $\text{sp}(g) = t$, and height at most τ .

Ensure: A shift in $S_{f,g}$, if one exists; otherwise FAIL

- 1: Set the global height cap $\mathcal{B}_{\text{glob}} = (nst(\tau + 1) \log(d + 2))^C$, for the fixed constant C chosen in [section 3.3](#).
- 2: Let $H^{\delta_1}(f), \dots, H^{\delta_m}(f)$ be the nonzero homogeneous components of f of positive degree, where $d = \delta_1 > \dots > \delta_m > 0$. The degree-zero component, if present, is not included.
- 3: Set $\mathbf{a}_0 \leftarrow \mathbf{0}$.
- 4: Set $\mathcal{L}_0$ to be the empty linear system in the variables $\mathbf{b} = (b_1, \dots, b_n)$.
- 5: **for** $q = 1, \dots, m$ **do**
- 6: Compute the following polynomial using [lemma 4.1](#) and the global cap $\mathcal{B}_{\text{glob}}$; if the procedure returns FAIL, return FAIL:

$$P_q(\mathbf{x}) \leftarrow P_{\delta_q-1, \mathbf{a}_{q-1}}(f)(\mathbf{x}) = \sum_{j=\delta_q+1}^d \frac{1}{(j - \delta_q + 1)!} f_j^{(j-\delta_q+1)}(\mathbf{a}_{q-1}, \mathbf{x}).$$

The subroutine returns FAIL if the collected P_q has more than $s^2 t$ monomials or a coefficient of height larger than $\mathcal{B}_{\text{glob}}$.

- 7: Let $\mathcal{L}_q$ be $\mathcal{L}_{q-1}$ together with the coefficient equations of

$$H^{\delta_q-1}(g(\mathbf{x})) = H^{\delta_q-1}(f(\mathbf{x})) + f_{\delta_q}^{(1)}(\mathbf{b}, \mathbf{x}) + P_q(\mathbf{x}).$$

- 8: Solve the linear system $\mathcal{L}_q$ for $\mathbf{b}$.
 - 9: **if** the above system has no solution **then**
 - 10: **return** FAIL
 - 11: **else**
 - 12: Let $\mathbf{a}_q \leftarrow \mathbf{b}$, where $\mathbf{b}$ is a basic rational solution of the system.
 - 13: **end if**
 - 14: **end for**
 - 15: Verify $f(\mathbf{x}) - g(\mathbf{x} - \mathbf{a}_m) \equiv 0$ using the sparse verifier from the proof of [corollary 1.1](#).
 - 16: **if** $f(\mathbf{x}) - g(\mathbf{x} - \mathbf{a}_m) \equiv 0$ **then**
 - 17: **return** $\mathbf{a}_m$
 - 18: **else**
 - 19: **return** FAIL
 - 20: **end if**
-

4.1 Computing the sparse representation of $P_{i,\mathbf{a}}(f)$

We now explain how to compute P_q in [algorithm 1](#).

Lemma 4.1. Given $\mathbf{a}_{q-1}$, the step that computes P_q in [algorithm 1](#) either returns FAIL or computes the sparse representation of

$$P_q(\mathbf{x}) = P_{\delta_{q-1}, \mathbf{a}_{q-1}}(f)(\mathbf{x})$$

using $\text{poly}(n, s, t, \log d)$ arithmetic operations. In the rational bit-complexity model, the same procedure uses

$$\text{poly}(n, s, t, \tau, \log d, \text{Ht}(\mathbf{a}_{q-1}))$$

bit operations. If it succeeds, then

$$\text{sp}(P_q) \leq s^2 t \quad \text{and, over } \mathbb{Q}, \quad \text{Ht}(P_q) \leq \mathcal{B}_{\text{glob}}.$$

Moreover, if

$$H^{\geq \delta_q}(f(\mathbf{x} + \mathbf{a}_{q-1})) = H^{\geq \delta_q}(g(\mathbf{x})),$$

then the enumeration checks in the procedure do not fail.

Proof. For every monomial $c_\gamma \mathbf{x}^\gamma$ of $H^{> \delta_q}(f)$, the terms of $c_\gamma(\mathbf{x} + \mathbf{a}_{q-1})^\gamma$ of degree exactly δ_q are indexed by vectors $\mathbf{v} \preceq \gamma$ satisfying

$$\text{supp}(\mathbf{v}) \subseteq \text{supp}(\mathbf{a}_{q-1}), \quad |\mathbf{v}| = |\gamma| - \delta_q. \quad (4)$$

Here v_ℓ means that, from the factor $(x_\ell + a_{q-1, \ell})^{\gamma_\ell}$, we choose v_ℓ copies of $a_{q-1, \ell}$ and the remaining $\gamma_\ell - v_\ell$ copies of x_ℓ . Their coefficients are

$$c_\gamma \prod_{\ell=1}^n \binom{\gamma_\ell}{v_\ell} (a_{q-1, \ell})^{v_\ell},$$

and the corresponding monomial is $\mathbf{x}^{\gamma - \mathbf{v}}$.

The only point at which we use the additional hypothesis in the lemma is to show that these enumeration checks cannot reject. Under that hypothesis, [lemma 3.2](#) applies with $\mathbf{a} = \mathbf{a}_{q-1}$ and $r = \delta_q$, and gives

$$\text{sp}\left(H^{\geq \delta_q}((\mathbf{x} + \mathbf{a}_{q-1})^\gamma)\right) \leq st.$$

The degree- δ_q terms are part of this polynomial. The vectors satisfying (4) are exactly the choices that produce those degree- δ_q monomials, and different vectors give different monomials. Hence the displayed bound implies that there are at most st such vectors.

We now give a procedure to enumerate all vectors satisfying (4). The procedure either returns a list of all such vectors or returns FAIL if there are more than st such vectors. The procedure is polynomial-time in the input size, which is polynomial in $n, s, t, \tau, \log d, \text{Ht}(\mathbf{a}_{q-1})$. We first compute:

$$S := \sum_{\ell: a_{q-1, \ell} \neq 0} \gamma_\ell.$$

If $|\gamma| - \delta_q > S$, then no vector satisfying (4) exists, because the sum of the coordinates of any such vector is $|\gamma| - \delta_q$, while the sum of the coordinates where $\mathbf{a}_{q-1}$ is nonzero is only S . In this case the degree- δ_q part for this monomial is empty. The degree- $(\delta_q - 1)$ part is empty as well, since it would require one additional copy of some coordinate of $\mathbf{a}_{q-1}$. Thus this monomial contributes nothing to P_q .

It remains to handle the case $|\gamma| - \delta_q \leq S$. If $|\gamma| - \delta_q \geq st$, the procedure returns FAIL. Under the lemma's additional hypothesis, this case cannot occur. Indeed, since $|\gamma| - \delta_q \leq S$, choose any

$\mathbf{v} \preceq \gamma$ supported on $\text{supp}(\mathbf{a}_{q-1})$ with $|\mathbf{v}| = |\gamma| - \delta_q$. For every $\mathbf{u} \preceq \mathbf{v}$, the corresponding term in $c_\gamma(\mathbf{x} + \mathbf{a}_{q-1})^\gamma$ is nonzero, distinct, and has degree at least δ_q . Therefore

$$\prod_{\ell: v_\ell > 0} (v_\ell + 1) \geq |\mathbf{v}| + 1 = |\gamma| - \delta_q + 1 \geq st + 1,$$

contradicting the displayed sparsity bound above.

It remains to handle the case $|\gamma| - \delta_q \leq st - 1$ and $|\gamma| - \delta_q \leq S$. Let $\ell_1, \dots, \ell_m$ be the indices for which $a_{q-1, \ell} \neq 0$. We build a dynamic-programming table whose entries are short lists of vectors. The entry $\mathcal{V}_i(h)$ records, with cap $st + 1$, all ways to obtain total drop h using only the coordinates $\ell_i, \ell_{i+1}, \dots, \ell_m$. For $i = m + 1, m, \dots, 1$ and $h = 0, \dots, |\gamma| - \delta_q$, let $\mathcal{V}_i(h)$ be a list of vectors $(v_{\ell_i}, \dots, v_{\ell_m})$ with

$$0 \leq v_{\ell_r} \leq \gamma_{\ell_r} \quad \text{for } r = i, \dots, m, \quad v_{\ell_i} + \dots + v_{\ell_m} = h.$$

If there are more than st such vectors, we keep only $st + 1$ of them.

The table is filled from the end. We start with

$$\mathcal{V}_{m+1}(0) = \{()\}, \quad \mathcal{V}_{m+1}(h) = \emptyset \quad (h > 0),$$

where $()$ is the empty vector. For $i \leq m$, the list $\mathcal{V}_i(h)$ is obtained by trying $e = 0, \dots, \min\{\gamma_{\ell_i}, h\}$ and appending e in front of every vector in $\mathcal{V}_{i+1}(h - e)$:

$$(e, \mathbf{w}) \quad \text{where} \quad \mathbf{w} \in \mathcal{V}_{i+1}(h - e).$$

We stop adding to an entry once it contains $st + 1$ vectors. By induction on i , each entry either contains all vectors satisfying its condition, or contains $st + 1$ such vectors. Hence $\mathcal{V}_1(|\gamma| - \delta_q)$ directly gives the vectors satisfying (4), after adding zero coordinates outside $\{\ell_1, \dots, \ell_m\}$.

If $\mathcal{V}_1(|\gamma| - \delta_q)$ has size $st + 1$, the procedure returns FAIL. Otherwise it is the exact list of all degree- δ_q vectors. The table has $O(nst)$ entries, each entry tries at most st values of e and stores at most $st + 1$ vectors, so this takes polynomial time. Each listed vector gives the monomial and coefficient above.

The listed terms are exactly the degree- δ_q part

$$C_\gamma(\mathbf{x}) = H^{\delta_q} (c_\gamma(\mathbf{x} + \mathbf{a}_{q-1})^\gamma).$$

We now recover the degree- $(\delta_q - 1)$ contribution from C_γ by a direct coefficient calculation. Write

$$D_{\mathbf{a}_{q-1}} = \sum_{\ell=1}^n a_{q-1, \ell} \frac{\partial}{\partial x_\ell}.$$

Let $\mathbf{e}_\ell$ denote the ℓ th standard basis vector. Fix $\mathbf{w} \preceq \gamma$ with $\text{supp}(\mathbf{w}) \subseteq \text{supp}(\mathbf{a}_{q-1})$ and $|\mathbf{w}| = |\gamma| - \delta_q + 1$. The monomial $\mathbf{x}^{\gamma - \mathbf{w}}$ can arise in $D_{\mathbf{a}_{q-1}}(C_\gamma)$ only by differentiating, for some ℓ with $w_\ell > 0$, the degree- δ_q monomial corresponding to $\mathbf{w} - \mathbf{e}_\ell$. Thus its coefficient in $D_{\mathbf{a}_{q-1}}(C_\gamma)$ is

$$c_\gamma \mathbf{a}_{q-1}^{\mathbf{w}} \sum_{\ell: w_\ell > 0} \binom{\gamma}{\mathbf{w} - \mathbf{e}_\ell} (\gamma_\ell - w_\ell + 1).$$

For every ℓ with $w_\ell > 0$,

$$\binom{\gamma}{\mathbf{w} - \mathbf{e}_\ell} (\gamma_\ell - w_\ell + 1) = w_\ell \binom{\gamma}{\mathbf{w}}.$$

Hence the coefficient above equals

$$(|\gamma| - \delta_q + 1) c_\gamma \binom{\gamma}{\mathbf{w}} \mathbf{a}_{q-1}^{\mathbf{w}},$$

which is $(|\gamma| - \delta_q + 1)$ times the coefficient of $\mathbf{x}^{\gamma-\mathbf{w}}$ in $H^{\delta_q-1}(c_\gamma(\mathbf{x} + \mathbf{a}_{q-1})^\gamma)$. Therefore the contribution of $c_\gamma \mathbf{x}^\gamma$ to P_q is

$$\frac{1}{|\gamma| - \delta_q + 1} D_{\mathbf{a}_{q-1}}(C_\gamma(\mathbf{x})).$$

Summing these contributions over all monomials of $H^{>\delta_q}(f)$ and collecting like terms gives exactly $P_{\delta_q-1, \mathbf{a}_{q-1}}(f)$.

After the terms have been collected, we make one final size check. If the result has more than s^2t monomials, or if one of its coefficients has height larger than the supplied height cap, then we return FAIL. In [algorithm 1](#), this cap is $\mathcal{B}_{\text{glob}}$. Otherwise we output the collected sparse representation.

The vector-list dynamic program, the binomial coefficients with $|\mathbf{v}| \leq st - 1$, the one additional multiplication by a coordinate of $\mathbf{a}_{q-1}$ and by an exponent in the directional derivative, the collection of like terms, and the final height comparisons all use

$$\text{poly}(n, s, t, \tau, \log d, \text{Ht}(\mathbf{a}_{q-1}))$$

bit operations. □

Remark 4.1. The computation of P_q uses the current vector $\mathbf{a}_{q-1}$, but on a yes instance its value can be bounded through the fixed witnessing shift $\mathbf{c}$. Indeed, the tail equality proved in [theorem 4.1](#) gives $P_q = P_{\delta_q-1, \mathbf{c}}(f)$ on every yes instance. Thus the sparsity and height of the computed tail are controlled by [proposition 3.1](#), using the single fixed shift $\mathbf{c}$ rather than a bound that grows recursively with q .

Theorem 4.1. *Algorithm 1 correctly determines whether there exists a shift in $S_{f,g}$.*

Proof. First suppose $S_{f,g} = \emptyset$. Then for every vector $\mathbf{a}$, the polynomial $f(\mathbf{x}) - g(\mathbf{x} - \mathbf{a})$ is nonzero. Thus no candidate can pass the final verification check. The algorithm may return FAIL earlier because an enumeration exceeds the support cap, because a computed coefficient exceeds $\mathcal{B}_{\text{glob}}$, or because a linear system is infeasible. If none of these early checks fails, then all intermediate sparse objects are bounded by the explicit caps in the algorithm and the candidate coordinates are bounded by the basic-solution determinant bound, so the final sparse verifier from [corollary 1.1](#) runs within the stated polynomial time and rejects. Hence the algorithm can only return FAIL on no instances.

It remains to consider a yes instance. By [theorem 1.3](#), fix $\mathbf{c} \in S_{f,g}$ of height $\text{poly}(n, s, t, \tau, \log d)$. Then $g(\mathbf{x}) \equiv f(\mathbf{x} + \mathbf{c})$. By [lemma 2.2](#), for every $0 \leq i \leq d-1$,

$$H^i(g(\mathbf{x})) = H^i(f(\mathbf{x})) + f_{i+1}^{(1)}(\mathbf{c}, \mathbf{x}) + P_{i,\mathbf{c}}(f)(\mathbf{x}). \quad (5)$$

We prove the following invariant at the beginning of iteration q :

- (i) Both $\mathbf{c}$ and the current vector $\mathbf{a}_{q-1}$ satisfy $\mathcal{L}_{q-1}$.
- (ii) Every solution $\mathbf{b}$ of $\mathcal{L}_{q-1}$ satisfies

$$f_{\delta_p}^{(1)}(\mathbf{b}, \mathbf{x}) \equiv f_{\delta_p}^{(1)}(\mathbf{c}, \mathbf{x}) \quad \text{for all } 1 \leq p < q. \quad (6)$$

(iii) The current vector $\mathbf{a}_{q-1}$ satisfies

$$H^{\geq \delta_q}(f(\mathbf{x} + \mathbf{a}_{q-1})) = H^{\geq \delta_q}(g(\mathbf{x})). \quad (7)$$

For $q = 1$, the first item holds because $\mathcal{L}_0$ is empty, the second item is vacuous, and the third holds because $\mathbf{a}_0 = \mathbf{0}$, $\delta_1 = d$, and translations do not change the top homogeneous component:

$$H^d(f) = H^d(f(\mathbf{x} + \mathbf{c})) = H^d(g).$$

Now assume the invariant holds at the beginning of iteration q . The high-truncation equality (7) is exactly the hypothesis used in [lemma 4.1](#) to apply [lemma 3.2](#) with $r = \delta_q$. Hence the enumeration checks in the computation of P_q do not fail.

We next compare the tail being computed with the corresponding tail at the true shift $\mathbf{c}$. If $j > \delta_q$ and $H^j(f) \neq 0$, then $j = \delta_p$ for some $p < q$. Applying the induction invariant to the algorithm's current vector $\mathbf{a}_{q-1}$ gives equality of the first directional derivatives in the directions $\mathbf{a}_{q-1}$ and $\mathbf{c}$. By [lemma 2.1](#),

$$f_j^{(r)}(\mathbf{a}_{q-1}, \mathbf{x}) \equiv f_j^{(r)}(\mathbf{c}, \mathbf{x}) \quad \text{for every } r \geq 1 \quad \text{and } j > \delta_q. \quad (8)$$

Therefore

$$P_q(\mathbf{x}) = P_{\delta_q-1, \mathbf{a}_{q-1}}(f)(\mathbf{x}) = P_{\delta_q-1, \mathbf{c}}(f)(\mathbf{x}). \quad (9)$$

By [equation \(9\)](#), the polynomial being checked is $P_{\delta_q-1, \mathbf{c}}(f)$, which depends only on the fixed witnessing shift $\mathbf{c}$. The structural lemma gives the required sparsity bound, so the size checks in [lemma 4.1](#) pass. In the rational bit-complexity version, [proposition 3.1](#) also applies: $P_{\delta_q-1, \mathbf{c}}(f)$ has height at most $\mathcal{B}_{\text{glob}}$. Hence the height checks pass as well, so the computation of P_q does not return FAIL.

Using [equations \(5\) and \(9\)](#) with $i = \delta_q - 1$, we get

$$H^{\delta_q-1}(g(\mathbf{x})) = H^{\delta_q-1}(f(\mathbf{x})) + f_{\delta_q}^{(1)}(\mathbf{c}, \mathbf{x}) + P_q(\mathbf{x}). \quad (10)$$

Thus $\mathbf{c}$ satisfies the new equations added in iteration q , and hence satisfies $\mathcal{L}_q$.

Let $\mathbf{b}$ be any solution of $\mathcal{L}_q$. Since $\mathcal{L}_{q-1} \subseteq \mathcal{L}_q$, the induction invariant gives [equation \(6\)](#) for all $p < q$. Subtracting [equation \(10\)](#) from the new equation in $\mathcal{L}_q$ gives

$$f_{\delta_q}^{(1)}(\mathbf{b}, \mathbf{x}) \equiv f_{\delta_q}^{(1)}(\mathbf{c}, \mathbf{x}). \quad (11)$$

Therefore every solution of $\mathcal{L}_q$ satisfies the derivative equalities for all $p \leq q$.

The algorithm chooses $\mathbf{a}_q$ to be a solution of $\mathcal{L}_q$. In the rational bit-complexity version it chooses a basic rational solution; by the appendix [lemma A.1](#) and the choice of $\mathcal{B}_{\text{glob}}$, this solution has height at most $\mathcal{B}_{\text{glob}}$. If $q < m$, we prove the high-truncation invariant for the next iteration. By the previous paragraph and [lemma 2.1](#), all directional derivative contributions from components $H^{\delta_p}(f)$ with $p \leq q$ are the same under $\mathbf{a}_q$ and under $\mathbf{c}$. For degrees at least δ_{q+1} , every contribution either comes from one of these already processed higher components, or from the top part of the component of the same degree, which is independent of the shift. Hence

$$H^{\geq \delta_{q+1}}(f(\mathbf{x} + \mathbf{a}_q)) = H^{\geq \delta_{q+1}}(f(\mathbf{x} + \mathbf{c})) = H^{\geq \delta_{q+1}}(g(\mathbf{x})).$$

This proves the invariant for the next iteration.

After the final iteration, the returned vector $\mathbf{a}_m$ satisfies

$$f_{\delta_p}^{(1)}(\mathbf{a}_m, \mathbf{x}) \equiv f_{\delta_p}^{(1)}(\mathbf{c}, \mathbf{x}) \quad \text{for every } 1 \leq p \leq m.$$

If $H^j(f) = 0$ or $j = 0$, then $f_j^{(1)}(\mathbf{a}_m, \mathbf{x}) = f_j^{(1)}(\mathbf{c}, \mathbf{x}) = 0$. Therefore

$$f_j^{(1)}(\mathbf{a}_m, \mathbf{x}) \equiv f_j^{(1)}(\mathbf{c}, \mathbf{x}) \quad \text{for all } 0 \leq j \leq d.$$

Since $\mathbf{c} \in S_{f,g}$, [lemma 2.3](#) implies $\mathbf{a}_m \in S_{f,g}$. The final verification succeeds, and the algorithm returns a valid shift. $\square$

Proposition 4.1 (Runtime and height bounds). *In the rational bit-complexity version, every basic rational solution $\mathbf{a}_q$ computed by [Algorithm 1](#) has height*

$$\text{poly}(n, s, t, \tau, \log d).$$

Moreover, the algorithm can be implemented using $\text{poly}(n, s, t, \tau, \log d)$ bit operations.

Proof. There are at most s nonzero positive homogeneous components of f , so the algorithm has at most s iterations. In iteration q , only the degree $\delta_q - 1$ equations are newly added; all earlier equations are kept in $\mathcal{L}_{q-1}$.

We first record the size and height of the equations added in one stage. Suppose that the algorithm reaches the linear-system step in iteration q . Then the computation of P_q has been accepted by the global checks. Hence

$$\text{sp}(P_q) \leq s^2 t, \quad \text{Ht}(P_q) \leq \mathcal{B}_{\text{glob}}.$$

Therefore the right-hand side $H^{\delta_q-1}(g) - H^{\delta_q-1}(f) - P_q$ has support at most $t + s + s^2 t$ and height bounded by $\text{poly}(n, s, t, \tau, \log d)$. Here we use the fact that $\mathcal{B}_{\text{glob}}$ was chosen to be a fixed polynomial bound large enough to dominate all height bounds used by the algorithm. The newly added equations are obtained by comparing coefficients in

$$f_{\delta_q}^{(1)}(\mathbf{b}, \mathbf{x}) = H^{\delta_q-1}(g) - H^{\delta_q-1}(f) - P_q.$$

The right-hand side has support at most $t + s + s^2 t$. On the other hand,

$$f_{\delta_q}^{(1)}(\mathbf{b}, \mathbf{x}) = \sum_{\ell=1}^n b_\ell \partial_\ell H^{\delta_q}(f).$$

Thus the support of the left-hand side is contained in the union of the supports of the n partial derivatives of $H^{\delta_q}(f)$. Since $\text{sp}(H^{\delta_q}(f)) \leq s$, this union has size at most ns . Consequently, the coefficient comparison in iteration q introduces at most $ns + t + s + s^2 t = \text{poly}(n, s, t)$ scalar linear equations.

The coefficients of these equations come from the partial derivatives $\partial_\ell H^{\delta_q}(f)$. Differentiating a monomial only multiplies its coefficient by an exponent bounded by d , so the entries of the coefficient matrix have height $O(\tau + \log d)$. The right-hand side has height $\text{poly}(n, s, t, \tau, \log d)$, as shown above. Therefore, after any iteration q , the accumulated system $\mathcal{L}_q$ has only $\text{poly}(n, s, t)$ scalar equations in the n variables $b_1, \dots, b_n$, and all entries of its coefficient matrix and right-hand side have height $\text{poly}(n, s, t, \tau, \log d)$.

We now apply the appendix [lemma A.1](#). If the accumulated system $\mathcal{L}_q$ is feasible, then a basic rational solution $\mathbf{a}_q$ has height $\text{poly}(n, s, t, \tau, \log d)$, and such a solution can be computed using $\text{poly}(n, s, t, \tau, \log d)$ bit operations. On yes instances, the choice of $\mathcal{B}_{\text{glob}}$ is large enough to dominate this polynomial height bound. On no instances, this polynomial bound is sufficient for the bit-complexity analysis; no separate height cap is imposed on $\mathbf{a}_q$.

It remains to justify that the whole algorithm has polynomial bit complexity. We argue inductively only for the cost of computing the successive tails and solutions; the height of P_q itself is controlled by the global check above, not by a recursive height bound. The initial vector is $\mathbf{a}_0 = \mathbf{0}$, so it has height 0. Suppose that the algorithm has reached iteration q and that $\mathbf{a}_{q-1}$ has height

$$\text{poly}(n, s, t, \tau, \log d).$$

Then [lemma 4.1](#) computes P_q , or returns FAIL, using $\text{poly}(n, s, t, \tau, \log d)$ bit operations. If the computation succeeds, the global support and height checks give

$$\text{sp}(P_q) \leq s^2 t, \quad \text{Ht}(P_q) \leq \mathcal{B}_{\text{glob}},$$

so the newly added equations have the polynomial size and height bounds proved above. Hence exact linear algebra, together with the appendix [lemma A.1](#), computes the next basic rational solution $\mathbf{a}_q$ using

$$\text{poly}(n, s, t, \tau, \log d)$$

bit operations, and this solution again has height

$$\text{poly}(n, s, t, \tau, \log d).$$

This completes the induction for every candidate vector produced by the algorithm.

Since there are at most s iterations, the total cost of all computations of the polynomials P_q and all linear-system solves is $\text{poly}(n, s, t, \tau, \log d)$. Finally, if the algorithm reaches the last verification step, the candidate shift has polynomial height. Therefore the final sparse verification can be performed in $\text{poly}(n, s, t, \tau, \log d)$ bit operations by the proof of [corollary 1.1](#). Hence the whole rational bit-complexity version of the algorithm runs in $\text{poly}(n, s, t, \tau, \log d)$ bit operations, and every basic rational solution $\mathbf{a}_q$ computed by the algorithm has height $\text{poly}(n, s, t, \tau, \log d)$. $\square$

5 Conclusion and Open Problems

We gave the first deterministic polynomial-time algorithm, in the sparse input size, for target-given shift equivalence of sparse multivariate polynomials over $\mathbb{Q}$. For $f, g \in \mathbb{Q}[\mathbf{x}]$ of sparsities s, t and degree at most d , the algorithm decides whether there is $\mathbf{a} \in \mathbb{Q}^n$ such that

$$f(\mathbf{x} + \mathbf{a}) \equiv g(\mathbf{x})$$

using polynomially many arithmetic operations in $n, s, t, \log d$. It outputs a rational shift of polynomial height and runs in polynomial bit complexity, with polynomial bounds in $n, s, t, \tau, \log d$. As noted in the introduction, the structural and arithmetic parts extend to any field of characteristic zero or larger than d . This exponentially improves on the indirect ROABP-based consequence of the framework of Dvir, Oliveira, and Shpilka described above [[DOS14](#)], whose dependence on the degree is polynomial in d rather than polynomial in $\log d$.

The key point is that, although sparse polynomials are not closed under shifts, our structural lemma shows that the shifted expansions arising in a yes instance remain sparse after the relevant

truncations. This allows the homogeneous descent to be carried out entirely in sparse representation. Over $\mathbb{Q}$, the same structural control also keeps the bit-complexity analysis from becoming recursive in the heights of the intermediate candidates.

Separate black-box access to two promised-sparse inputs is handled by sparse interpolation, as discussed above. This differs from receiving evaluation access only to the composed expression $f(\mathbf{x} + \mathbf{a}) - g(\mathbf{x} + \mathbf{b})$, for which our structural verifier cannot inspect f and g . Black-box PIT for any-order ROABPs [GKS17, GG20] gives polynomial dependence on d and quasipolynomial-type dependence overall; obtaining complexity polynomial in $n, s, t, \log d$ for this composed-oracle problem remains open.

Another direction is to understand the small-characteristic regime. Our algorithm uses Taylor expansion, directional derivatives, and factorial denominators, and therefore assumes that the characteristic is zero or larger than the degree. In small characteristic, several derivative-based characterizations no longer apply in the same form. It would be interesting to determine whether sparse shift equivalence still admits an efficient deterministic algorithm for formal polynomial identities in small characteristic.

Acknowledgements. The authors thank OpenAI’s ChatGPT Pro for assistance during the preparation of this manuscript. ChatGPT Pro also suggested the argument leading to the small rational shift existence proof of [theorem 1.3](#). The authors have independently checked the arguments and take full responsibility for the correctness of all results and proofs.

References

- [AGR13] Andrew Arnold, Mark Giesbrecht, and Daniel S. Roche. Faster sparse interpolation of straight-line programs. In *Computer Algebra in Scientific Computing - 15th International Workshop, CASC 2013*, volume 8136 of *Lecture Notes in Computer Science*, pages 61–74. Springer, 2013.
- [AKS06] Martín Avendaño, Teresa Krick, and Martín Sombra. Factoring bivariate sparse (lacunary) polynomials. In *Proceedings of the 2006 International Symposium on Symbolic and Algebraic Computation (ISSAC)*, pages 193–200. ACM, 2006.
- [AS06] Manindra Agrawal and Nitin Saxena. Equivalence of $\mathbb{F}$ -algebras and cubic forms. In *Proceedings of the 23rd Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 3884 of *Lecture Notes in Computer Science*, pages 115–126. Springer, 2006.
- [BDJ26] Markus Bläser, Sagnik Dutta, and Gorav Jindal. Problems from optimization and computational algebra equivalent to hilbert’s nullstellensatz. In Kasper Green Larsen and Barna Saha, editors, *Proceedings of the 2026 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2026, Vancouver, BC, Canada, January 11-14, 2026*, pages 6308–6331. SIAM, 2026.
- [BDS24] Omkar Baraskar, Agrim Dewan, and Chandan Saha. Testing equivalence to design polynomials. In *41st International Symposium on Theoretical Aspects of Computer Science, STACS 2024*, volume 289 of *Leibniz International Proceedings in Informatics*, pages 9:1–9:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024.
- [BDSS24] Omkar Baraskar, Agrim Dewan, Chandan Saha, and Pulkit Sinha. NP-hardness of testing equivalence to sparse polynomials and to constant-support polynomials. In Karl

- Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson, editors, *51st International Colloquium on Automata, Languages, and Programming, ICALP 2024, Tallinn, Estonia, July 8-12, 2024*, volume 297 of *LIPICs*, pages 16:1–16:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024.
- [BHLV09] Markus Bläser, Moritz Hardt, Richard J. Lipton, and Nisheeth K. Vishnoi. Deterministically testing sparse polynomial identities of unbounded degree. *Information Processing Letters*, 109(3):187–192, 2009.
- [BJ14] Markus Bläser and Gorav Jindal. A new deterministic algorithm for sparse multivariate polynomial interpolation. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation*, pages 51–58. ACM, 2014.
- [BKRS26] Somnath Bhattacharjee, Rishabh Kothary, Shanthanu S. Rai, and Shubhangi Saraf. Deterministic algorithms for low individual degree factors of sparse polynomials. *CoRR*, abs/2606.27293, 2026.
- [BOT88] Michael Ben-Or and Prasoona Tiwari. A deterministic algorithm for sparse multivariate polynomial interpolation. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 301–309. ACM, 1988.
- [BSV20] Vishwas Bhargava, Shubhangi Saraf, and Ilya Volkovich. Deterministic factorization of sparse polynomials with bounded individual degree. *Journal of the ACM*, 67(2):1–28, 2020.
- [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, Hamburg, Germany, March 7-9, 2023*, volume 254 of *LIPICs*, pages 22:1–22:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.
- [CS26] Aminadav Chuyoon and Amir Shpilka. On factorization of sparse polynomials of bounded individual degree. *Electron. Colloquium Comput. Complex.*, TR26, 2026.
- [Dew26] Agrim Dewan. Testing equivalence to the hamiltonian cycle polynomial. *CoRR*, abs/2606.26653, 2026.
- [DG24] Pranjal Dutta and Sumanta Ghosh. Advances in polynomial identity testing. *SIGACT News*, 55(2):38–86, 2024.
- [DOS14] Zeev Dvir, Rafael Oliveira, and Amir Shpilka. Testing equivalence of polynomials under shifts. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, volume 8572 of *Lecture Notes in Computer Science*, pages 417–428. Springer, 2014.
- [FGS18] Michael A Forbes, Sumanta Ghosh, and Nitin Saxena. Towards blackbox identity testing of log-variate circuits. In *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*, pages 54–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2018.

- [FS13] Michael A. Forbes and Amir Shpilka. Quasipolynomial-time identity testing of non-commutative and read-once oblivious algebraic branching programs. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013*, pages 243–252. IEEE Computer Society, 2013.
- [FSS14] Michael A. Forbes, Ramprasad Satharishi, and Amir Shpilka. Hitting sets for multilinear read-once algebraic branching programs, in any order. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing, STOC 2014*, pages 867–875. ACM, 2014.
- [GG20] Zeyu Guo and Rohit Gurjar. Improved explicit hitting-sets for roabps. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2020)*, pages 4–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020.
- [GK93] Dima Grigoriev and Marek Karpinski. A zero-test and an interpolation algorithm for the shifted sparse polynomials. In Gérard D. Cohen, Teo Mora, and Oscar Moreno, editors, *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, 10th International Symposium, AAECC-10, San Juan de Puerto Rico, Puerto Rico, May 10-14, 1993, Proceedings*, volume 673 of *Lecture Notes in Computer Science*, pages 162–169. Springer, 1993.
- [GKS17] Rohit Gurjar, Arpita Korwar, and Nitin Saxena. Identity testing for constant-width, and any-order, read-once oblivious arithmetic branching programs. *Theory Comput.*, 13(1):1–21, 2017.
- [GKsL03] Mark Giesbrecht, Erich L. Kaltofen, and Wen shin Lee. Algorithms for computing sparsest shifts of polynomials in power, chebyshev, and pochhammer bases. *Journal of Symbolic Computation*, 36(3–4):401–424, 2003.
- [GL95] Dima Grigoriev and Yagati N. Lakshman. Algorithms for computing sparse shifts for multivariate polynomials. In A. H. M. Levelt, editor, *Proceedings of the 1995 International Symposium on Symbolic and Algebraic Computation, ISSAC ’95, Montreal, Canada, July 10-12, 1995*, pages 96–103. ACM, 1995.
- [GR10] Mark Giesbrecht and Daniel S. Roche. Interpolation of shifted-lacunary polynomials. *computational complexity*, 19(3):333–354, 2010.
- [Gri97] Dima Grigoriev. Testing shift-equivalence of polynomials by deterministic, probabilistic and quantum machines. *Theor. Comput. Sci.*, 180(1-2):217–228, 1997.
- [GS09] Sanchit Garg and Éric Schost. Interpolation of polynomials given by straight-line programs. *Theoretical Computer Science*, 410(27-29):2659–2662, 2009.
- [GST23] Nikhil Gupta, Chandan Saha, and Bhargav Thankey. Equivalence test for read-once arithmetic formulas. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 4205–4272. SIAM, 2023.
- [Jr.99] Hendrik W. Lenstra Jr. Finding small degree factors of lacunary polynomials. *Number Theory in Progress*, 1:267–276, 1999.
- [Kay11] Neeraj Kayal. Efficient algorithms for some special cases of the polynomial equivalence problem. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1409–1421. SIAM, 2011.

- [Kay12] Neeraj Kayal. Affine projections of polynomials. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing, STOC 2012*, pages 643–662. ACM, 2012.
- [KI04] Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004.
- [KK05] Erich L. Kaltofen and Pascal Koiran. On the complexity of factoring bivariate super-sparse (lacunary) polynomials. In *Proceedings of the 2005 International Symposium on Symbolic and Algebraic Computation (ISSAC 2005)*, pages 208–215. ACM, 2005.
- [KLW90] Erich Kaltofen, Yagati N. Lakshman, and John-Michael Wiley. Modular rational sparse multivariate polynomial interpolation. In *Proceedings of the 1990 International Symposium on Symbolic and Algebraic Computation*, pages 135–139. ACM, 1990.
- [KS01] Adam R. Klivans and Daniel A. Spielman. Randomness efficient identity testing of multivariate polynomials. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, pages 216–223. ACM, 2001.
- [LS94] Yagati N. Lakshman and B. David Saunders. On computing sparse shifts for univariate polynomials. In Malcolm A. H. MacCallum, editor, *Proceedings of the International Symposium on Symbolic and Algebraic Computation, ISSAC '94, Oxford, UK, July 20-22, 1994*, pages 108–113. ACM, 1994.
- [Roc18] Daniel S. Roche. What can (and can't) we do with sparse polynomials? *CoRR*, abs/1807.08289, 2018.
- [RS05] Ran Raz and Amir Shpilka. Deterministic polynomial identity testing in non-commutative models. *Computational Complexity*, 14(1):1–19, 2005.
- [Sch80] Jacob T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, 27(4):701–717, 1980.
- [Ser03] Ákos Seress. *Permutation Group Algorithms*, volume 152 of *Cambridge Tracts in Mathematics*. Cambridge University Press, 2003.
- [Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In *Symbolic and Algebraic Computation, EUROSAM 1979*, volume 72 of *Lecture Notes in Computer Science*, pages 216–226. Springer, 1979.

A Missing Proofs

Observation A.1. Let $A \subseteq \mathbb{C}^n$ be an affine subspace. If $\mathbf{a}_1, \dots, \mathbf{a}_m \in A$ and $\lambda_1, \dots, \lambda_m \in \mathbb{C}$ satisfy

$$\sum_{i=1}^m \lambda_i = 1,$$

then

$$\sum_{i=1}^m \lambda_i \mathbf{a}_i \in A.$$

In particular, affine subspaces are closed under affine averages.

Proof. Write $A = \mathbf{a}_0 + V$, where $V \subseteq \mathbb{C}^n$ is a linear subspace. For each i , write $\mathbf{a}_i = \mathbf{a}_0 + \mathbf{v}_i$ with $\mathbf{v}_i \in V$. Then

$$\sum_{i=1}^m \lambda_i \mathbf{a}_i = \mathbf{a}_0 + \sum_{i=1}^m \lambda_i \mathbf{v}_i \in \mathbf{a}_0 + V = A,$$

since $\sum_i \lambda_i = 1$ and V is closed under linear combinations. $\square$

Proof of proposition 2.1. Write

$$f(\mathbf{x} + \mathbf{z}) - g(\mathbf{x}) = \sum_{\alpha} F_{\alpha}(\mathbf{z}) \mathbf{x}^{\alpha},$$

where each $F_{\alpha} \in \mathbb{Q}[\mathbf{z}]$. Thus $S_{f,g}(\mathbb{C})$ is the common zero set of finitely many polynomials with rational coefficients:

$$S_{f,g}(\mathbb{C}) = \{\mathbf{a} \in \mathbb{C}^n : F_{\alpha}(\mathbf{a}) = 0 \text{ for all } \alpha\}.$$

Since $S_{f,g}(\mathbb{C})$ is nonempty, choose $\mathbf{c} \in S_{f,g}(\mathbb{C})$. Then $F_{\alpha}(\mathbf{c}) = 0$ for all α . Now, consider the ideal

$$I := \langle F_{\alpha} \rangle_{\alpha} \subseteq \overline{\mathbb{Q}}[\mathbf{z}].$$

We claim that I is a proper ideal. Indeed, if $1 \in I$, then we could write

$$1 = \sum_{\alpha} H_{\alpha}(\mathbf{z}) F_{\alpha}(\mathbf{z})$$

with $H_{\alpha} \in \overline{\mathbb{Q}}[\mathbf{z}]$, and evaluating at $\mathbf{z} = \mathbf{c}$ would give $1 = 0$, a contradiction. Thus I is proper. Since $\overline{\mathbb{Q}}$ is algebraically closed, the weak Nullstellensatz gives a point

$$\mathbf{a} \in \overline{\mathbb{Q}}^n$$

such that $F_{\alpha}(\mathbf{a}) = 0$ for all α . Equivalently, $f(\mathbf{x} + \mathbf{a}) = g(\mathbf{x})$.

Let $L/\mathbb{Q}$ be a finite Galois extension containing all coordinates of $\mathbf{a}$, and let

$$G := \text{Gal}(L/\mathbb{Q}).$$

For every $\sigma \in G$, applying σ to the identity $f(\mathbf{x} + \mathbf{a}) = g(\mathbf{x})$ gives

$$f(\mathbf{x} + \sigma(\mathbf{a})) = g(\mathbf{x}),$$

because the coefficients of f and g lie in $\mathbb{Q}$. Hence $\sigma(\mathbf{a}) \in S_{f,g}(\mathbb{C})$ for every $\sigma \in G$.

Since $S_{f,g}(\mathbb{C})$ is an affine subspace, it is closed under affine averages; see [observation A.1](#). Therefore

$$\bar{\mathbf{a}} := \frac{1}{|G|} \sum_{\sigma \in G} \sigma(\mathbf{a}) \in S_{f,g}(\mathbb{C}).$$

On the other hand, $\bar{\mathbf{a}}$ is fixed by every element of G . Formally, for any $\tau \in G$, the sum is just a reordering

$$\sum_{\sigma \in G} \tau(\sigma(\mathbf{a})) = \sum_{\sigma \in G} \sigma(\mathbf{a}).$$

Thus each coordinate of $\bar{\mathbf{a}}$ lies in $L^G = \mathbb{Q}$.³ Therefore

$$\bar{\mathbf{a}} \in \mathbb{Q}^n.$$

Consequently there exists a rational vector $\mathbf{q} := \bar{\mathbf{a}} \in \mathbb{Q}^n$ such that

$$f(\mathbf{x} + \mathbf{q}) = g(\mathbf{x}),$$

as desired. $\square$

³Here $L^G := \{z \in L : \sigma(z) = z \text{ for all } \sigma \in G\}$ is the fixed field of G . Since $L/\mathbb{Q}$ is Galois and $G = \text{Gal}(L/\mathbb{Q})$, we have $L^G = \mathbb{Q}$.

Proof of [theorem 1.3](#). Suppose $S_{f,g}(\overline{\mathbb{Q}}) \neq \emptyset$. By [proposition 2.1](#), choose a rational shift $\mathbf{c} \in S_{f,g}(\mathbb{Q})$.

The proof has three steps. First, the structural bound confines all relevant derivatives of f to a polynomial-dimensional space W . Second, on W we represent the translation from f to g by a small-height unipotent operator and take its finite logarithm to recover $D_{\mathbf{c}}f$. Third, we solve $D_{\mathbf{b}}f = D_{\mathbf{c}}f$ for a small basic rational solution $\mathbf{b}$; the difference $\mathbf{b} - \mathbf{c}$ stabilizes f , so $\mathbf{b}$ is again a valid shift.

If f is constant, then $g = f$, and the zero shift is a rational witness of height 0. Thus we may assume that f is nonconstant.

Let $S = \text{supp}(\mathbf{c})$ and

$$W = \langle \partial^{\mathbf{v}} f \mid \mathbf{v} \in \mathbb{N}^n, \text{supp}(\mathbf{v}) \subseteq S \rangle.$$

By [lemma 3.2](#) with $r = 0$, every monomial $\mathbf{x}^\gamma$ of f satisfies

$$\prod_{i \in S} (\gamma_i + 1) \leq st.$$

For a fixed monomial $\mathbf{x}^\gamma$ of f , a derivative $\partial^{\mathbf{v}}(\mathbf{x}^\gamma)$ with $\text{supp}(\mathbf{v}) \subseteq S$ is nonzero exactly when $0 \leq v_i \leq \gamma_i$ for every $i \in S$. Therefore,

Fact A.1. *The number of nonzero derivatives*

$$\partial^{\mathbf{v}}(\mathbf{x}^\gamma), \quad \text{supp}(\mathbf{v}) \subseteq S,$$

is at most

$$\prod_{i \in S} (\gamma_i + 1) \leq st.$$

Fact A.2. *Each such nonzero derivative satisfies*

$$|\mathbf{v}| = \sum_{i \in S} v_i \leq \sum_{i \in S} \gamma_i \leq \prod_{i \in S} (\gamma_i + 1) - 1 \leq st - 1.$$

Since f has at most s monomials, [fact A.1](#) implies

$$\dim W \leq s^2 t.$$

Let

$$m = \dim W \leq s^2 t.$$

Choose a basis

$$B = \{w_1 = f, w_2, \dots, w_m\}$$

of W such that each

$$w_j = \partial^{\mathbf{v}_j} f$$

for some $\mathbf{v}_j \in \mathbb{N}^n$ with $\text{supp}(\mathbf{v}_j) \subseteq S$. Since we take $w_1 = f$, we have $\mathbf{v}_1 = 0$.

Now solve the linear system

$$\sum_{j=2}^m y_j w_j = g - f,$$

in variables $y_2, \dots, y_m$, where the linear equations of the system are obtained by comparing coefficients of monomials on both sides.

We claim that the system is satisfiable over $\mathbb{Q}$. This is because

$$g(\mathbf{x}) - f(\mathbf{x}) = f(\mathbf{x} + \mathbf{c}) - f(\mathbf{x}) = \sum_{\substack{\mathbf{v} \neq 0 \\ \text{supp}(\mathbf{v}) \subseteq S}} \frac{\mathbf{c}^{\mathbf{v}}}{\mathbf{v}!} \partial^{\mathbf{v}} f \in \text{span}\{w_2, \dots, w_m\},$$

where the inclusion follows because $w_1 = f$ is the only polynomial of degree d in the basis B , while the displayed sum has degree at most $d - 1$ and hence lies in the span of $B \setminus \{w_1\}$.

By [fact A.2](#), every coefficient of the derivatives $w_j = \partial^{\mathbf{v}_j} f$ has height at most

$$O(\tau + st \log d),$$

because differentiating multiplies the input coefficients by falling factorials of exponents and each exponent is bounded by d . Therefore, the linear system has coefficient height $O(\tau + st \log d)$ and it has $m - 1 \leq s^2 t$ variables. Then, by [lemma A.1](#), it has a rational solution $(y_2, \dots, y_m)$ with height

$$\text{poly}(s, t, \log d, \tau).$$

Define the differential operator:

$$Y = 1 + N \quad \text{with} \quad N = \sum_{j=2}^m y_j \partial^{\mathbf{v}_j}. \quad (12)$$

Then we have

$$Yf = f + \sum_{j=2}^m y_j w_j = g.$$

[Fact A.2](#) says that any derivative of f with respect to S -variables and total order $\geq st$ is zero. Therefore,

$$N^{st}h = 0 \quad \text{for every } h \in W.$$

Thus the logarithm of Y is finite on W :

$$\log Y = \sum_{r=1}^{st-1} \frac{(-1)^{r+1}}{r} N^r.$$

Put

$$q = (\log Y)f.$$

The polynomial q has coefficient height $\text{poly}(n, s, t, \log d, \tau)$. Indeed, work in the basis B of W . Choose m monomial coordinates in which the coefficient matrix of $w_1, \dots, w_m$ is nonsingular; this matrix has height $O(\tau + st \log d)$. For each basis element w_j , the polynomial Nw_j is a polynomial-height linear combination of derivatives of f of total order at most $st - 1$. Expressing Nw_j back in B by [lemma A.1](#) gives polynomial-height coordinates. Thus the matrix A of the transformation $N|_W$ has polynomial height. Since $A^{st} = 0$,

$$q = \left(\sum_{r=1}^{st-1} \frac{(-1)^{r+1}}{r} A^r \right) e_1$$

in B -coordinates, and repeated matrix multiplication and addition for st steps keeps the bit-height polynomial.

Now compare Y with the shift operator

$$T_{\mathbf{c}} = \exp \left(\sum_{i \in S} c_i \partial_i \right).$$

We have

$$Yf = g = T_{\mathbf{c}}f.$$

Since Y and $T_{\mathbf{c}}$ are differential operators, they commute with other differential operators. Therefore, for every $\mathbf{v}$ with $\text{supp}(\mathbf{v}) \subseteq S$,

$$(Y - T_{\mathbf{c}})\partial^{\mathbf{v}}f = \partial^{\mathbf{v}}(Y - T_{\mathbf{c}})f = 0.$$

Thus Y and $T_{\mathbf{c}}$ act identically on W . Their logarithms also act identically on W , because the logarithm is a finite polynomial in the nilpotent operator $Y - 1 = T_{\mathbf{c}} - 1$ on W . Hence

$$q = (\log Y)f = (\log T_{\mathbf{c}})f = \left(\sum_{i \in S} c_i \partial_i \right) f.$$

Finally, solve the linear system in variables $b_i, i \in S$:

$$\sum_{i \in S} b_i \partial_i f = q.$$

It is consistent, since $b_i = c_i$ is a solution. The coefficient matrix has height $O(\tau + \log d)$, and the right-hand side q has polynomial height. Thus [lemma A.1](#) gives a rational solution $(b_i)_{i \in S}$ with

$$\text{Ht}(b_i) \leq \text{poly}(n, s, t, \log d, \tau).$$

Set $b_i = 0$ for $i \notin S$. Then

$$\sum_{i=1}^n b_i \partial_i f = \sum_{i=1}^n c_i \partial_i f,$$

so

$$\sum_{i=1}^n (b_i - c_i) \partial_i f = 0$$

Hence, by [lemma 2.5](#), we have $\mathbf{b} - \mathbf{c} \in S_f(\mathbb{C})$. It follows that $\mathbf{b} \in S_{f,g} \cap \mathbb{Q}^n$ and has the desired height bound. $\square$

Definition A.1 (Basic rational solution). Let $M\mathbf{y} = \mathbf{r}$ be a feasible linear system over $\mathbb{Q}$, with $M \in \mathbb{Q}^{m \times n}$ and $\rho = \text{rank}(M)$. Choose ρ linearly independent columns of M whose span contains $\mathbf{r}$, set all other variables to zero, and solve for the variables in the chosen columns. The resulting vector is called a *basic rational solution*.

Lemma A.1 (Basic rational solution: height and bit complexity). Let $M \in \mathbb{Q}^{m \times n}$ and $\mathbf{r} \in \mathbb{Q}^m$ be a linear system $M\mathbf{y} = \mathbf{r}$ such that every entry of M and $\mathbf{r}$ has height at most H . If the system is feasible, then:

- (i) there exists a basic rational solution $\mathbf{y}^* \in \mathbb{Q}^n$, and
- (ii) every coordinate of $\mathbf{y}^*$ has height $\text{poly}(m, n, H)$.

Moreover, such a basic rational solution can be computed deterministically in $\text{poly}(m, n, H)$ bit operations.

Proof. Let $\rho = \text{rank}(M)$. Since the system is feasible, $\mathbf{r}$ lies in the column span of M . Choose ρ independent columns B that form a basis of this column span. Then $\mathbf{r} \in \text{span}(M_B)$, so the reduced system $M_B \mathbf{y}_B = \mathbf{r}$ is feasible. Choose ρ independent rows R such that the square matrix $C = (M_B)_R$ is nonsingular, and let $\mathbf{d} = \mathbf{r}_R$. Solving

$$C\mathbf{z} = \mathbf{d}$$

determines the unique vector $\mathbf{z}$ with $M_B \mathbf{z} = \mathbf{r}$; the remaining rows are satisfied because they are linear combinations of the rows in R on the column space spanned by M_B . Setting all variables outside B to zero gives a basic rational solution of the original system.

For height, write each coordinate of $\mathbf{z}$ by Cramer's rule as $\det(C_i) / \det(C)$, where C_i denotes the matrix obtained from C by replacing its i th column with the right-hand side vector $\mathbf{d}$. By Hadamard's bound, determinants of $r \times r$ matrices with entry height $\leq H$ have height $\text{poly}(r, H)$, hence each coordinate has height $\text{poly}(m, n, H)$.

For computation, use fraction-free Gaussian elimination (or equivalent exact linear algebra over $\mathbb{Q}$) to find ρ , a nonsingular basis matrix C , and the corresponding solution exactly; all intermediate integers and arithmetic cost are bounded by $\text{poly}(m, n, H)$. $\square$