Electronic Colloquium on Computational Complexity, Report No. 310 (2026)

Deterministic factorization of constant depth circuits from *whitebox* PIT

Somnath Bhattacharjee * Mrinal Kumar † Shanthanu S. Rai†
Varun Ramanathan ‡ Ramprasad Saptharishi† Shubhangi Saraf§

Abstract

We show that over the field of rational numbers, there is a deterministic algorithm for factorization of constant depth circuits that makes calls to a (whitebox or blackbox) PIT algorithm for such circuits. The time complexity of the algorithm (and hence the number of calls to the PIT oracle) is polynomially bounded in the size and degree of the input circuit. Our reduction to PIT is fairly simple and general and applies as it is to other natural classes of algebraic circuits like formulas and algebraic branching programs.

Prior to this work, the known deterministic factorization algorithms for constant depth circuits crucially relied on *blackbox* PIT [Bhattacharjee et al.]. Our main technical ingredient is an efficient, deterministic algorithm that, when given access to sufficiently high degree truncations of *all* the power series roots of a constant depth circuit, uses oracle calls to any (whitebox or blackbox) PIT algorithm for these circuits, and partitions the roots into the sets of conjugates; thereby identifying the irreducible factors.

For our proof, we rely on a root recombination criterion using logarithmic derivatives [Sasaki et al., Bostan et al., Lecerf] and show that this criterion can be efficiently and deterministically implemented for power series roots of constant depth circuits using access to any PIT oracle. On the way, we discuss an algorithm that may be of independent interest: it takes as input a matrix whose entries are multivariate constant depth circuits over a field $\mathbb{F}$, requires oracle access to any PIT algorithm for these circuits and outputs a basis for the $\mathbb{F}$-kernel of this matrix in polynomial time.

1 Introduction

The complexity of polynomial factorization as an algorithmic task and the relationship between the circuit complexities of a polynomial and that of its factors are questions of fundamental interest in computational algebra and algebraic complexity. Both these families of questions saw substantial advances in the 1980s in a sequence of results of Kaltofen (e.g. see [Kal89, KT90, vzGK85, Bür04]) who showed both a *closure* result for algebraic circuits under factorization, and gave efficient randomized algorithms for this task (see, for instance, the survey of Forbes and Shpilka [FS15] for a

*University of Toronto, Canada. Email: somnath.bhattacharjee@mail.utoronto.ca. Research partially supported by an NSERC Discovery Grant

†Tata Institute of Fundamental Research, Mumbai, India. Email: {mrinal, shanthanu.rai, ramprasad}@tifr.res.in. Research supported by the Department of Atomic Energy, Government of India, under project number RTI400112, and in part by research grants from Google, SERB and Premji Invest.

‡University of Copenhagen, Denmark. Email: vara@di.ku.dk. Supported by the European Research Council (ERC) under grant agreement no. 101125652 (ALBA).

§University of Toronto, Canada. Email: shubhangi.saraf@utoronto.ca. Research partially supported by the McLean Award and an NSERC Discovery Grant.

ISSN 1433-8092

brief discussion of the literature). More formally, the closure result showed that if an n variate degree d polynomial f over the field of rational numbers[1] is computable by an algebraic circuit of size s, then all its irreducible factors can be computed by circuits of size $\text{poly}(s, d)$. The algorithmic results consisted of efficient randomized algorithms that compute these irreducible factors in randomized $\text{poly}(s, d)$ time, when given the circuit for f as input. Remarkably, these algorithms work in both the *blackbox* and the *whitebox* settings: in the blackbox setting the algorithm is just given evaluation access to a blackbox that contains f and it outputs blackboxes corresponding to the irreducible factors of f, and in the whitebox setting the algorithm has access to the internal wirings of a circuit for f, and it outputs explicitly, the circuits for its irreducible factors.

Subsequent to these results, perhaps the two most important research directions in this area have been that of extending the closure result to the natural structured subclasses of algebraic circuits, e.g formulas/branching programs/constant depth circuits, and that of designing efficient deterministic algorithms for polynomial factorization. There has been substantial progress on the question of closure results in recent years, and we now know that algebraic branching programs [ST21], formulas & constant depth circuits [BKR+26a] are all *closed* under factorization, and so is the complexity class VNP [CKS19].

The progress on the question of designing deterministic factorization algorithms has been comparatively subdued. A work of Kopparty, Saraf and Shpilka [KSS15] showed that derandomizing polynomial identity testing (PIT) is both necessary and sufficient for derandomizing polynomial factorization for general algebraic circuits. While the reduction from PIT to factorization is relatively simple, the other direction required a clean algebraic understanding of the necessity and use of randomness in Kaltofen's factorization algorithms. This equivalence is also variant preserving in the sense that blackbox PIT for general circuits is equivalent to blackbox polynomial factorization, and whitebox PIT is equivalent to factorization in the whitebox model.

An intriguing aspect of the equivalence result of Kopparty et al. is that it does not appear to *scale down*: while it is easy to see that PIT reduces to factorization for models like formulas and constant depth circuits, it was not clear if derandomizing PIT for these models would imply efficient deterministic factorization algorithms for them. This research direction was explored in a sequence of recent results [KRS24, KRSV26, BKR+25], and led to the results of Bhattacharjee et al. [BKR+25] that gave a subexponential time deterministic algorithm for factorization of constant depth algebraic circuits. In a follow up work [BKR+26a], Bhattacharjee et al. established closure results for constant depth circuits (and other classes). They also used this closure result to simplify many of the technical details of the algorithm in [BKR+25], and obtained a clean statement connecting polynomial factorization to deterministic blackbox PIT for constant depth circuits. When combined with the known deterministic subexponential time algorithms for blackbox PIT for constant depth circuits,[2] this yielded another deterministic, subexponential time algorithm for factorization of these circuits.

Very curiously, the deterministic factorization algorithms in [BKR+25] and [BKR+26a] crucially relied on the fact that the known subexponential time PIT algorithms for constant depth circuits were in the *blackbox* model. In particular, it was not known if a similar deterministic factorization algorithm can be obtained from a whitebox PIT algorithm for constant depth circuits. In this work, we answer this question in affirmative. We state the theorem informally here, and refer to Theorem 4.1 for a formal statement.

Theorem 1.1 (Informal)**.** *There is a deterministic algorithm that takes as input a size s, degree d*

[1] Slightly weaker results of a similar flavor also hold over finite fields.

[2] That were proved via a combination of superpolynomial lower bounds for these circuits from [LST25] with some partial closure results for such circuits in [CKS19].

multivariate algebraic circuit of constant depth over the rational numbers,[3] *requires oracle access to a PIT algorithm for constant depth circuits in either blackbox or whitebox setting, and outputs constant depth circuits for the irreducible factors of the input polynomial, together with their multiplicities, in* $\mathrm{poly}(s, d)$ *time.*

Furthermore, a similar reduction from factorization to deterministic PIT holds for algebraic formulas as well.

For algebraic branching programs a similar reduction can be seen in the work of Sinhababu & Thierauf [ST21]. Thus, Theorem 1.1, together with the reduction in [BKR+26a] give a scaled down version of the equivalence between PIT and polynomial factorization for general circuits shown by Kopparty, Saraf and Shpilka [KSS15].

As of now, we do not have deterministic PIT algorithms for constant depth algebraic circuits in the whitebox setting that are faster than those in the blackbox setting. However, it is typically the case in derandomization literature that whitebox algorithms are often discovered before blackbox PIT algorithms for many classes of algebraic circuits and they tend to be faster (see [SY10, Sections 4.6 and 4.8] for a general overview). Theorem 1.1 shows that any improvement in the state of art for PIT for constant depth circuits, even in the whitebox setting would result in an improvement in the state of art for polynomial factorization for these circuits. We remark that the various structural results for constant depth circuits in the work of Andrews & Wigderson [AW26] and those in [BKR+26a, BKR+26b] play a crucial role in the proof of Theorem 1.1. However, given these structural results, the final factorization algorithm for constant depth circuits obtained in this work appears to be technically simpler in many of its details than those known from prior work [BKR+25, BKR+26a].

On the way to the proof of Theorem 1.1, we show the following result that may be of independent interest.

Theorem 1.2 (Kernel basis computation over base field)**.** *There is a deterministic algorithm that takes as input a matrix* $M(\mathbf{X})$ *of dimension* m *whose entries are multivariate polynomials computable by size* s*, degree* d *algebraic circuits of constant depth over a field* $\mathbb{F}$ *(satisfying* $|\mathbb{F}| > d$*), requires oracle access to a PIT algorithm for constant depth circuits over* $\mathbb{F}$ *in either blackbox or whitebox setting, and outputs a basis for the* $\mathbb{F}$*-kernel of* $M(\mathbf{X})$ *in time* $\mathrm{poly}(s, m, d)$*.*

Here, by an $\mathbb{F}$*-kernel, we mean the* $\mathbb{F}$*-linear space of* m*-dimensional vectors over* $\mathbb{F}$ *that are in the kernel of* $M(\mathbf{X})$*.*

The proof of Theorem 1.2 was discovered with AI assistance (ChatGPT Astra).

Yet another independently interesting application of Theorem 1.2 is to the question of linear independence testing of polynomials. In particular, we have the following immediate corollary.

Corollary 1.3. *There is a deterministic algorithm that takes as input polynomials* $f_1, f_2, \ldots, f_m$ *given by circuits of size at most* s *and degree* d*, requires oracle access to a (whitebox or blackbox) PIT algorithm for circuits of the form* $(\sum_i \alpha_i f_i)$ *for* $\alpha_i \in \mathbb{F}$*, and decides if* $f_1, \ldots, f_m$ *are linearly independent over the field* $\mathbb{F}$ *in time* $\mathrm{poly}(s, d, m)$*.*

While this problem has been studied in the literature (see, for instance, [Kay10, CKW11, BSV26]), it is unclear if this specific reduction to *whitebox* PIT was known to exist. The corollary can be instantiated with additional guarantees on the structure of the circuits for $f_1, f_2, \ldots, f_m$ to obtain

[3] Throughout this paper, we work over the field of rational numbers. Assuming the correctness of the recent efficient deterministic univariate factorization algorithms over finite fields [Ope26], the results also extend to finite fields of sufficiently large characteristic.

deterministic algorithms for testing the linear independence of polynomials in interesting settings. For instance, if the input polynomials are all sparse, then using the known deterministic PIT for sparse polynomials, this can be done in polynomial time even in the blackbox setting. Similarly, we can obtain a polynomial time deterministic algorithm for testing linear independence of ROABPs (in the same order) in the whitebox setting using the PIT algorithm of Raz & Shpilka [RS05]. While neither of these two applications appear to be new, the algorithms obtained using Corollary 1.3 appear to be much more modular and essentially remain the same at a high level, even as the circuit classes containing the input change.

1.1 An overview of the proofs

Let $f(\mathbf{X}, Y)$ be the input polynomial given as a small constant depth circuit with rational coefficients. Then, the main steps in the factorization algorithm on input f in various proofs in the literature, and in particular in [BKR+25, BKR+26a] are as follows.

1. **Preprocessing:** In this step, by an invertible linear transformation of the form $X_i \to X_i + a_i Y$, for $a_i \in \mathbb{F}$, it is ensured that f is monic in Y. At this point, using standard techniques, f can be assumed to be square free. Now, by shifting the $\mathbf{X}$-variables further if necessary, we reduce to the setting that $f(\mathbf{0}, Y)$ is square free and splits into distinct linear factors[4] over $\mathbb{F}$. We now pick a new formal variable T, and substitute every variable X_i by $X_i T$. The new polynomial, which we refer to as $F(T, Y)$ here is viewed as a bivariate in T, Y, with its coefficients in $\mathbb{F}(\mathbf{X})$.

2. **Computing power series roots:** Using the squarefreeness of $f(\mathbf{0}, Y)$, and hence $F(0, Y)$, we get from Theorem 2.11 that the Y-roots of F are power series in T, with coefficients in $\mathbb{F}(\mathbf{X})$. Furthermore, sufficiently high degree truncations of these roots are computable by small constant depth circuits, and these circuits can be efficiently and deterministically constructed from a circuit for F and the roots of $F(0, Y)$.

3. **Obtaining irreducible factors:** The final step of the factoring algorithms is to identify the power series roots of F that come from the same irreducible factor of F. Once this partition of the set of roots is obtained, the irreducible factors of F (and hence f) can be constructed efficiently, using
$$f_i(T, Y) = \prod_{i \in S} (Y - \phi_i(T)) \bmod T^{\deg_T(f)+1},$$
where, $S \subseteq [\deg_Y(F)]$ is the set of indices of all power series roots of the irreducible factor f_i.

The power series root computation step above is completely deterministic, and randomness is used only for preprocessing and for the root combination steps. Of these, it is not difficult to see (and is discussed in detail in the proofs in [KSS15, BKR+25, BKR+26a]) that randomness for preprocessing can be removed with access to PIT algorithms. For the setting of constant depth circuits, we need the additional amazing fact proved by Andrews & Wigderson [AW26] (also see [BKR+26b]) that Resultants and Discriminants are computable by constant depth circuits. This allows us to show that for constant depth circuits, the preprocessing phase of the factorization algorithms can be derandomized using an access to a PIT oracle, in either the blackbox or the whitebox setting. A crucial (and simple) part of this argument is that one can efficiently and deterministically solve the search version of PIT, where the goal is to find a point where the input polynomial evaluates to a non-zero value, when given access to a PIT oracle (see Lemma 2.2).

[4] We get rid of this assumption in Section 3 by using Hensel Lifting.

The main technical difficulty that needed to be addressed for the proofs in this paper, and in [BKR+25, BKR+26a] was to derandomize the root combination step to identify roots that correspond to each irreducible factor of F. More concretely, we have the following problem: we have constant depth circuit computing a polynomial $F(T, Y) \in \mathbb{F}(\mathbf{X})[T, Y]$ with $\deg_Y(F) \leq d, \deg_T(F) \leq D$, and also have constant depth circuits for polynomials $B_1, B_2, \ldots, B_d \in \mathbb{F}(\mathbf{X})[T, Y]$ such that for every i,

$$F(T, B_i) \equiv 0 \bmod T^m,$$

for some $m = \text{poly}(d, D)$ that is large enough. Our goal is to partition the $B_i's$ into sets corresponding to the irreducible factors of F. In [BKR+25, BKR+26a], this was done by using blackbox PIT for constant depth circuits (given via a hitting set generator): the hitting set is used to set the $\mathbf{X}$ variables to field constants in $\mathbb{F}$. This gives us many bivariate polynomials $g(T, Y)$ in T, Y with coefficients over the base field $\mathbb{F}$, that can be factorized deterministically in polynomial time using the algorithm outlined at the beginning of this section. The proofs in [BKR+25, BKR+26a] proceed by using the properties of the hitting set generator to guarantee that the correct partition of the power series roots of most of the bivariates g into buckets corresponding to their irreducible factors is in one-one correspondence with the correct partitioning of the power series roots of $F(T, Y) \in \mathbb{F}(\mathbf{X})[T, Y]$. This part of the proof uses the crucial fact that a hitting set generator for constant depth circuits will maintain the non-zeroness of an exponentially large set of constant depth circuits at once. This argument is only used in the proof of correctness of the algorithm, and is not implemented in the algorithm itself. Thus, this argument crucially relies on the fact that we have access to a hitting set generator (and not just a whitebox algorithm) for PIT of constant depth circuits.

For our proof, we give an alternative way to correctly combine the power series roots into irreducible factors. The proof that we present in Section 3 is based on the so-called *logarithmic derivative* idea that is very well studied in the factorization literature (see [SSKS91, SS93, BLS+04, Lec06, CL07, Lec10] and the references therein), albeit in the bivariate or the dense representation setting. For this overview, we try to provide some intuition into the method.

Since $F(0, Y)$ is square free, all of its roots, that we assume are present in the base field $\mathbb{F}$, must be distinct. Since $B_1, \ldots, B_d$ are power series roots of $F(T, Y)$ obtained by *lifting* the roots of $F(0, Y)$, they must have distinct constant terms. Suppose $H(T, Y)$ is an irreducible factor of $F(T, Y)$. Then, there is a subset $S \subseteq [d]$, with $|S| = \deg_Y(H)$ such that $\{B_i : i \in S\}$ is the set of truncations of all power series Y-roots of H.

Since H is a factor of F, we have that its T-degree is at most D. Thus, the elementary symmetric polynomials of the power series roots of H must be polynomials in $\mathbb{F}(\mathbf{X})[T]$ and must have T-degree at most D. Thus, in the expressions for the elementary symmetric polynomials of the power series roots of H, there must be significant cancellations, leading to these expressions yielding a low degree polynomials in T on simplification. The root recombination method is based on the fact that the converse of this observation is also essentially true: if all elementary symmetric polynomials of a subset $\hat{A}$ of power series roots of F are low degree polynomials in T, then $\hat{A}$ corresponds to a (not necessarily irreducible) factor of F.

While mathematically insightful, it is not immediately clear if we can *efficiently* search for subsets $\hat{A}$ of the power series roots of F that have this cancellation property, since trying all subsets is not within the budget. This approach seems more amenable to efficient algorithms if we work with power sum symmetric polynomials instead. Due to Newton-Girard identities that let us express power sums as bounded degree polynomials of elementary symmetric polynomials (and vice-versa), and the above discussion, we get that for any subset $\hat{A}$ of the power series roots of F, the power sum symmetric polynomials of degree[5] at most d on $\hat{A}$ are all low degree polynomials if and only if $\hat{A}$

[5]Recall that unlike the elementary symmetric polynomials that are zero when the degree exceeds the number of

corresponds to a factor of F. The advantage is that the search for subsets $\hat{A}$ now looks a little more tractable: consider formal variables $u_1, u_2, \ldots, u_d$ and the linear constraints

$$\forall \ell \leq d, \qquad \deg_T \left(\sum_i u_i \phi_i^\ell \right) \leq \hat{D}_\ell,$$

where ϕ_i's are the true (untruncated) power series roots of F, and each $\hat{D}_l \in \mathbb{N}$ is an appropriately chosen parameter. Once we have such a bound for $\ell \leq d$, a similar (ℓ-dependent) bound follows for *all* ℓ via the recurrence $p_\ell = -a_1 p_{\ell-1} - \cdots - a_d p_{\ell-d}$ where each p_ℓ denotes $\sum_{i \in [d]} u_i \phi_i^\ell$, and $F = Y^d + a_1 Y^{d-1} + \cdots + a_d$. A succinct equivalent representation of this system obtained by tying together the powers of ϕ_i into a generating function $(1/(Y - \phi_i))$ is as follows:

$$\deg_T \left([Y^{-\ell-1}] \sum_i u_i \cdot \frac{1}{(Y - \phi_i(T))} \right) \leq \hat{D}_\ell.$$

Each $1/(Y - \phi_i(T))$ is the "logarithmic derivative" of $Y - \phi_i(T)$ and similarly, $\sum_{i \in [d]} \frac{1}{Y - \phi_i(T)}$ is the logarithmic derivative of F; see [Wei] for a general definition. Clearing the denominator with $F(T, Y) = \prod_{i \in [d]} (Y - \phi_i(T))$ give us the system:

$$\deg_T \left([Y^{-\ell-1}] \left(\sum_i u_i \cdot \prod_{j \neq i} (Y - \phi_j(T)) \right) \right) \leq \tilde{D}_\ell$$

for some appropriately scaled $\tilde{D}_\ell$. But this is a system that uses power-series roots. Instead, we work with the following very closely related linear system described in [SSKS91, SS93, BLS+04, Lec06, CL07, Lec10]:

$$\deg_T \left(\left(\sum_i u_i \cdot \prod_{j \neq i} (Y - B_j(T)) \right) \bmod T^m \right) \leq D.$$

Note that this only relies on the truncations of the power series roots, and $\tilde{D}_\ell$ are all set to D.

This is clearly a homogeneous linear system in $u_1, \ldots, u_d$. If $F_1, F_2, \ldots, F_r$ are all the irreducible factors of F and $w_1, w_2, \ldots, w_r \in \{0,1\}^d$ are vectors defined by setting the jth coordinate of w_i to 1 if and only if B_j is a truncated power series root of F_i, then we observe that $w_1, w_2, \ldots, w_r$ are indeed solutions of this linear system: setting the u-variables according to w_i reduces the expression

$$\sum_i u_i \cdot \prod_{j \neq i} (Y - B_j(T)) \bmod T^m$$

to $\partial_Y(F_i) \frac{F}{F_i}$, and for $m \gg D$, this clearly has T-degree at most D.

A key technical observation for the proof is that $w_1, w_2, \ldots, w_r$ are in-fact a basis of the $\mathbb{F}$-linear space of solutions of this linear system in $\mathbb{F}^d$! Furthermore, they are the unique basis of this solution space (in $\mathbb{F}^d$) in the *reduced row echelon* form! Thus, if we could somehow find the unique basis of the solution space in the reduced row echelon form, we would be done. The challenge is to do this deterministically, perhaps using a PIT oracle, assuming that F has a small constant depth circuit.

From the results in [BKR+26a], we know that each B_j has a small constant depth circuit. This implies that the linear system in $u_1, \ldots, u_d$ reduces to finding a vector in $\mathbb{F}^d$ in the kernel of a matrix

variables, power sum polynomials do not vanish at high degrees.

M. The entries of the matrix M are n-variate polynomials in $\mathbb{F}[\mathbf{X}]$ that are computable by small constant depth circuits. An algorithm for this problem, using access to a whitebox PIT oracle for constant depth circuit is the final technical ingredient of our proof. Several variants of polynomial factorization algorithms in literature (e.g [BKR+26a, BKR+26a]) involve a linear system solving step. However, these systems are typically over the field $\mathbb{F}(\mathbf{X})$. Our algorithm for finding a basis for the $\mathbb{F}$-kernel of M crucially relies on the fact that we are looking for an $\mathbb{F}$-kernel and not an $\mathbb{F}(\mathbf{X})$-kernel (whose computation we believe is harder to derandomize). This algorithm, presented in Section 3 was discovered with AI assistance and happens to be surprisingly clean and elegant in hindsight. Indeed we are optimistic that such an algorithm might be of independent interest. Given such a basis for the kernel of M, we compute the unique basis of this space in the reduced row echelon form, and this gives us $w_1, \ldots, w_r$, thereby completing the recombination step.

AI disclosure

The proof of Theorem 1.2 was discovered using ChatGPT Astra. The rest of the paper including all the technical ideas and the writing are due to its human authors.

2 Preliminaries

2.1 Notation

- Throughout this paper we will be working over a characteristic zero field (such as $\mathbb{Q}$).
- $[n]$ denotes the set $\{1, 2, \ldots, n\}$.
- For any vector $v \in \mathbb{F}^n$, support of v is the set $\{i \in [n] : v_i \neq 0\}$.
- We will use $\mathbf{X}$ to represent the set of variables $\{X_1, \ldots, X_n\}$. We use $\mathbb{F}[T]$ and $\mathbb{F}[\![T]\!]$ to refer to the ring of polynomials and formal power series in T respectively.
- For any power series or polynomial $P(Z)$, $[Z^k]\{P\}$ will denote the coefficient of Z^k in $P(Z)$.
- For a polynomial $P(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$, $\deg(P)$ denotes the total degree of P, and for an $X_i \in \mathbf{X}$, $\deg_{X_i}(P)$ denotes the degree of P in X_i.
- Let $\{v_1, \ldots, v_r\}$ be a set of vectors over a field $\mathbb{K}$. Then for any subfield $\mathbb{F} \subseteq \mathbb{K}$, we denote $\mathrm{span}_{\mathbb{F}}(v_1, \ldots, v_r)$ to be the space of $\mathbb{F}$-linear combinations of v_is.
- Let $M \in \mathbb{F}^{m \times n}$ be a $m \times n$ matrix. Define $\ker_{\mathbb{F}}(M)$ to denote the kernel of M.
- For any polynomial (or formal power series) $P(T)$, $P(T) \bmod T^{D+1}$ will denote the polynomial obtained by discarding all terms of degree at least $D+1$.
- We will use $\mathsf{Esym}_r\{v_1, \ldots, v_m\}$ to denote the r-th elementary symmetric polynomial of the set $\{v_1, \ldots, v_m\}$, i.e. $\mathsf{Esym}_r\{v_1, \ldots, v_m\} = \sum_{S \subseteq [m], |S|=r} \prod_{i \in S} v_i$.

2.2 Algebraic Circuits

We will define the standard model of computation, namely algebraic circuits, and record some useful and well-known facts pertaining them which will be useful later own.

Algebraic Circuits: An algebraic circuit C is a DAG with a designated output node, wherein the leaf nodes are labeled by variables and field constants and the internal nodes are labeled by $+$ and $\times$. The circuit C computes the polynomial $P(\mathbf{X})$ at the output gate, by inductively computing it at each node from leaf nodes to the output node. The size of the circuit C is the number of wires in C, and the depth is the length of the longest path from the leaf node to the output node.

Michael Ben-Or observed that algebraic circuits are "closed" under polynomial interpolation. This means given a circuit C for a polynomial $f(\mathbf{X}, Y)$, there exist circuits for coefficients of $f(\mathbf{X}, Y)$ (in Y) which are not much larger in size than C. Moreover, there is an efficient algorithm to compute these circuits from C.

Lemma 2.1 (Interpolation)**.** *Let $\mathbb{F}$ be a sufficiently large field and $f(\mathbf{X}, Y) \in \mathbb{F}[\mathbf{X}, Y]$ be a $(n+1)$-variate polynomial with $\deg_Y(f) = d$. Suppose $f(\mathbf{X}, Y)$ is computable by a circuit of size s and depth Δ. Then, there is an algorithm that outputs a multi-output circuit of size $\mathrm{poly}(s, n, d)$ and depth $\Delta + O(1)$ and runs in time $\mathrm{poly}(n, s, d)$. The multi-output circuit outputs $[Y^m]f(\mathbf{X}, Y)$ for all $m \in \{0, 1, \ldots, d\}$.*

White-box PIT: One of the fundamental problems in algebraic complexity is the Polynomial Identity Testing (PIT) problem: Given a circuit C (or any well-defined algebraic model), does it compute the 0 polynomial? If the internal wiring in the circuits are revealed to us, then this is the question of PIT in the white-box model (succinctly referred to as the white-box PIT problem). Coming up with efficient PIT algorithms has important connections to lower bounds in different algebraic models, polynomial factorization, etc [KI04, SY10, KSS15, KS19]. Using an algorithm for the decision version of the problem, that is determining if a circuit C (say with n inputs) is non-zero, one can find a point $\alpha \in \mathbb{F}^n$ such that C is non-zero at α (namely, the search version).

Lemma 2.2 (Decision to Search PIT)**.** *Let $\mathbb{F}$ be a field such that $|\mathbb{F}| > d$ and $f(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$ be an n-variate, degree d non-zero polynomial from a circuit class $\mathcal{C}$. Then there is an $\mathrm{poly}(n, d)$ algorithm to compute an $\alpha \in \mathbb{F}^n$ such that $f(\alpha) \neq 0$ that uses oracle queries to a white-box PIT algorithm for $\mathcal{C}$.*

2.3 Resultant and the Poisson Formula

The resultant of $f(Y)$ and $g(Y)$ is an indicator for the GCD of f and g being non-trivial. In this section, we will cover some basic and well-known facts about the resultant. The reader can refer to [vzGG13] for a more detailed account about resultants. The resultant of f and g is defined as follows.

Definition 2.3 (Resultant)**.** Let $f(Y), g(Y) \in \mathbb{F}[Y]$ be nonzero polynomials of degrees d_f and d_g, respectively. Write

$$f(Y) = \sum_{k=0}^{d_f} a_k Y^k, \qquad g(Y) = \sum_{k=0}^{d_g} b_k Y^k.$$

The *Sylvester matrix* of f and g is the $(d_f + d_g) \times (d_f + d_g)$ matrix M

$$\begin{bmatrix} a_{d_f} & a_{d_f-1} & \cdots & a_0 & & \\ & \ddots & & & \ddots & \\ & & a_{d_f} & a_{d_f-1} & \cdots & a_0 \\ b_{d_g} & b_{d_g-1} & \cdots & b_0 & & \\ & \ddots & & & \ddots & \\ & & b_{d_g} & b_{d_g-1} & \cdots & b_0 \end{bmatrix}$$

The *resultant* of f and g, denoted by $\mathrm{Res}_Y(f,g)$, is defined as the determinant of the above matrix.

$$\mathrm{Res}_Y(f,g) := \det(M).$$

□

As mentioned previously, $\mathrm{Res}_Y(f,g)$ is zero exactly when the GCD of f and g is non-trivial. This fact is clearer from following equivalent characterization of the resultant.

Lemma 2.4 (Poisson Formula)**.** *Let $f(Y), g(Y) \in \mathbb{F}[Y]$ be nonzero polynomials of degrees d_f and d_g and suppose $\mathcal{B} = \{\beta_1, \beta_2, \ldots, \beta_{d_g}\}$ are the multiset of roots of g. Then,*

$$\mathrm{Res}_Y(f,g) = (-1)^{d_f d_g} \,\mathrm{lc}(g)^{d_f} \prod_{i \in [d_g]} f(\beta_i)$$

where $\mathrm{lc}(g)$ denotes the leading coefficient of g.

For bivariate polynomials $f, g \in \mathbb{F}[T,Y]$, $\mathrm{Res}_Y(f,g)$ is the resultant when f and g are viewed as polynomials in $\mathbb{F}[T][Y]$. In this case, $\mathrm{Res}_Y(f,g)$ would be a polynomial in T. From the definition of the resultant, we can bound the degree of $\mathrm{Res}_Y(f,g)$.

Lemma 2.5. *Let $f, g \in \mathbb{F}[T,Y]$ be polynomials with Y-degrees d_f and d_g respectively. Suppose the T-degree of both the polynomials is at most D. Then, $\deg(\mathrm{Res}_Y(f,g))$ is at most $D(d_f + d_g)$.*

2.4 Deterministic Univariate Factorization

Over rational numbers, univariate factorization can be performed in deterministic polynomial time using the well-known LLL algorithm of Lenstra, Lenstra, and Lovász [LLL82]. We define a convenient notation for the runtime of univariate factorization below, and then state the LLL theorem.

Definition 2.6. Let $f \in \mathbb{F}[X]$ be an univariate polynomial of degree d. We use $\mathcal{T}(\mathbb{F}, d)$ to denote the time taken to factor f over $\mathbb{F}$. □

Theorem 2.7 (Univariate factorization over rationals [LLL82])**.** *There is a deterministic algorithm that, given a polynomial $f \in \mathbb{Q}[x]$ of degree d and coefficient bit complexity at most B, computes the irreducible factorization of f over $\mathbb{Q}$ in time $\mathrm{poly}(d, B)$. Thus, over $\mathbb{Q}$, $\mathcal{T}(\mathbb{Q}, d)$ is polynomial in d and in the bit complexity of coefficients of the input.*

2.5 Polynomial factorisation preliminaries

The following is a standard pre-processing step that is performed in almost all factorisation algorithms (for instance, see [BKR+26a] for more details).

Lemma 2.8 (Factorisation pre-processing)**.** *Let $F(\mathbf{X}, Y) \in \mathbb{F}[\mathbf{X}, Y]$ be an $(n+1)$-variate polynomial with $\deg_Y F \leq d$ and $\deg F \leq D$, such that F is computable by a circuit C of size s and depth Δ. Then from the circuit C for F, and access to a PIT oracle for $\mathrm{poly}(s,n,d)$-sized depth $\Delta + O(1)$ circuits, we can efficiently (in time $\mathrm{poly}(s,d,n)$) compute a circuit $\tilde{C}$ of size $\mathrm{poly}(s,n,d)$ and depth $\Delta + O(1)$ for a polynomial $\tilde{F}(\mathbf{X}, T, Y) \in \mathbb{F}[\mathbf{X}][T,Y]$ with the following properties:*

- *$\tilde{F}$ is monic in Y of Y-degree d and total degree at most D.*
- *every monomial in $\tilde{F}$ that is divisible by any $\mathbf{X}$-variable is also divisible by T and thus $\tilde{F}(\mathbf{X}, 0, Y) = \tilde{F}(\mathbf{0}, 0, Y)$,*

- *$\tilde{F}$ is square-free, and the polynomial $\tilde{F}(\mathbf{0}, 0, Y) = \tilde{F}(X, 0, Y)$ is also square-free.*

Furthermore, the circuits for the factors of $F(\mathbf{X}, Y)$ can be efficiently recovered from circuits for the factors of $\tilde{F}(\mathbf{X}, T, Y)$.

Thus, from this point on, our focus is to factorize a polynomial $f(T, Y) \in \mathbb{F}[\mathbf{X}][T, Y]$ (where $|\mathbf{X}| = n$) such that it is: (i) square-free (ii) monic in Y, with Y-degree at most d (iii) $f(0, Y)$ is square-free (iv) computable by a circuit of size $\text{poly}(s, n, d)$ and depth Δ. We first state Gauss' lemma, which essentially states that it is equivalent to study the factorization of $f(T, Y)$ over $\mathbb{F}[\mathbf{X}][T, Y]$ and $\mathbb{F}(\mathbf{X})[T, Y]$. The version we state below suffices for our setting, for a more general version refer [vzGG13, Corollary 6.10]

Lemma 2.9 (Gauss' lemma)**.** *Let $f \in \mathbb{F}[\mathbf{X}][T, Y]$ be monic in Y. Then f is irreducible in $\mathbb{F}(\mathbf{X})[T, Y]$ if and only if f is irreducible in $\mathbb{F}[\mathbf{X}][T, Y]$.*

Since $f(T, Y)$ is monic in Y, the irreducible factors of f over $\mathbb{F}[\mathbf{X}]$ are also monic in Y. Hence, these irreducible factors are also irreducible over $\mathbb{F}(\mathbf{X})$. Thus, the factorization of f over $\mathbb{F}[\mathbf{X}]$ and $\mathbb{F}(\mathbf{X})$ are the same. Hence, we will study the factorization $f(T, Y)$ over the field $\mathbb{F}(\mathbf{X})$ (instead of the ring $\mathbb{F}[\mathbf{X}]$). Suppose $f(T, Y)$ factorizes as follows,

$$f(T, Y) = f_1(T, Y) \cdot f_2(T, Y) \cdots f_k(T, Y),$$

where $f_1, \ldots, f_k$ are irreducible polynomials in $\mathbb{F}(\mathbf{X})[T, Y]$. Also, suppose $f(0, Y)$ factorizes as,

$$f(0, Y) = (Y - \beta_1) \cdot (Y - \beta_2) \cdots (Y - \beta_d),$$

where $\beta_1, \ldots, \beta_d \in \overline{\mathbb{F}}$. We recall the below standard lemma (known as Hensel's lemma in the literature) that $f(T, Y)$ completely factorizes into linear factors in power series roots in Y.

Lemma 2.10 (Hensel's Lemma)**.** *Let $f(T, Y) \in \mathbb{F}[X][T, Y]$ be a polynomial which is monic in variable Y such that $f(0, Y)$ is square-free. Suppose $f(0, Y)$ factorizes into,*

$$f(0, Y) = (Y - \beta_1) \cdot (Y - \beta_2) \cdots (Y - \beta_d),$$

where $\beta_1, \ldots, \beta_d$ are distinct elements in $\overline{\mathbb{F}}$. Then, $f(T, Y)$ factorizes uniquely into,

$$f(T, Y) = (Y - \varphi_1(T)) \cdot (Y - \varphi_2(T)) \cdots (Y - \varphi_d(T)),$$

where for each $i \in [d]$, $\varphi_i(T) \in \mathbb{F}(\beta_i)(\mathbf{X})[\![T]\!]$ and $\varphi_i(0) = \beta_i$.

[BKR+26a, Theorem 4.1] showed how to obtain polynomial size and constant depth for the (truncations of) above power series roots $\varphi_i(T)$. We record this a theorem below for our setting, and note that this can be obtained by essentially invoking [BKR+26a, Theorem 4.1] for the shifted polynomial $f(T, Y + \beta_i)$.

Theorem 2.11 ([BKR+26a])**.** *Let $f(T, Y) \in \mathbb{F}[X][T, Y]$ be a polynomial as in the setting of Lemma 2.10. Suppose f is computable by a circuit of size s and depth Δ. Then for each $i \in [d]$,*

$$\varphi_i(T) \bmod T^{\ell},$$

is computable by a circuit C of size $\text{poly}(s, n, d, \ell)$ and depth $\Delta + O(1)$ over the field $\mathbb{F}(\mathbf{X})(\beta_i)$. Moreover, there is an algorithm to compute C with at most $\text{poly}(s, n, d, \ell)$ arithmetic operations.

The number of arithmetic operations specified above can be obtained by inspecting the proof of [BKR+26a, Theorem 4.1]. Using the above theorem, we can also obtain constant depth circuits for "hensel lifts" of the polynomial $f(T, Y)$. Suppose $f(0, Y)$ factorizes as,

$$f(0, Y) = \tilde{g}_1(Y) \cdot \tilde{g}_2(Y) \cdots \tilde{g}_r(Y)$$

where for each $i \in [r]$, $\tilde{g}_1, \tilde{g}_2, \ldots, \tilde{g}_r \in \mathbb{F}[Y]$ are irreducible polynomials. The monic (in Y) Hensel-lift from $\tilde{g}_1, \tilde{g}_2, \ldots, \tilde{g}_r \in \mathbb{F}[Y]$ is the following factorization,

$$f(T, Y) = g_1(T, Y) \cdots g_r(T, Y)$$

where $g_i(T, Y) \in \overline{\mathbb{F}}[\mathbf{X}][\![T]\!][Y]$ is given by

$$g_i(T, Y) = \prod_{j \in [d] : \tilde{g}_i(\beta_j) = 0} (Y - \varphi_j(T)).$$

Although each $\varphi_i(T)$ is a power series that may use coefficients from $\overline{\mathbb{F}}$, all coefficients of g_i are symmetric functions in the roots of $\tilde{g}_i(Y)$ and thus $g_i(T, Y) \in \mathbb{F}[\mathbf{X}][\![T]\!][Y]$.

Using Theorem 2.11, we can construct size $\mathrm{poly}(n, s, d, B)$ size and $\Delta + O(1)$ depth circuits C_j for $\varphi_j(T) \bmod T^B$ over the field $\mathbb{F}(\beta_j)$. Using these circuits, we can build $\mathrm{poly}(n, s, d, B)$ size and depth $\Delta + O(1)$ circuits[6] for $g_i(T, Y)$ over the field $\mathbb{F}(\beta_j : \tilde{g}_i(\beta_j) = 0)$. In order to obtain constant depth circuits for $g_i(T, Y)$ over the base field $\mathbb{F}$, we recall the following theorem from [BKR+26b].

Theorem 2.12 (Complexity of symmetric polynomials [BKR+26b])**.** *Let $n, d, s \in \mathbb{N}$, and suppose that $\mathbb{F}$ is a field satisfying $|\mathbb{F}| \geq (nds)^c$ for an absolute constant $c > 0$. Suppose that $P \in \mathbb{F}[x_1, \ldots, x_n]$ is symmetric of degree d and that a circuit of size s and depth Δ computes P.*

Write $\mathsf{Esym}_i(\mathbf{X})$ for the ith elementary symmetric polynomial in $\mathbf{X} = (X_1, \ldots, X_n)$. Let $Q \in \mathbb{F}[Z_1, \ldots, Z_n]$ be the unique polynomial satisfying

$$P(\mathbf{X}) = Q\big(\mathsf{Esym}_1(\mathbf{X}), \ldots, \mathsf{Esym}_n(\mathbf{X})\big).$$

Then $\deg Q \leq d$, and there is a circuit over $\mathbb{F}$ computing Q with size $\mathrm{poly}(s, d, n)$ and depth $\Delta + O(1)$.

Remark 2.13. In fact, by inspecting the proof of Theorem 2.12 in [BKR+26b, Theorem 16], given a size s and depth Δ for P, the above theorem gives us an algorithm to compute the size $\mathrm{poly}(s, d, n)$ and depth $\Delta + O(1)$ circuit for Q in $\mathrm{poly}(s, d, n)$ time.

Suppose $a_{i,1}, \ldots, a_{i,l_i} \in \mathbb{F}$ are the coefficients of $\tilde{g}_i$. Observe that g_i is symmetric over the roots of $\tilde{g}_i$ and the coefficients $a_{i,1}, \ldots, a_{i,l_i} \in \mathbb{F}$ are elementary symmetric polynomials (upto signs) in the roots of $\tilde{g}_i$. Thus, we invoke Theorem 2.12 to obtain a circuit of $\mathrm{poly}(n, s, d, B)$ size and $\Delta + O(1)$ depth computing g_i which uses constants from $\mathbb{F}(a_{i,1}, \ldots, a_{i,l_i}) = \mathbb{F}$. From Theorem 2.11 and Theorem 2.12, all the above circuit constructions can be performed with at most $\mathrm{poly}(n, s, d, B)$ many arithmetic operations.

Theorem 2.14 (Computing Hensel-lifts efficiently)**.** *Let $f(T, Y) \in \mathbb{F}[X][T, Y]$ be a polynomial which is monic in variable Y such that $f(0, Y)$ is square-free with the irreducible factorization $f(0, Y) = \tilde{g}_1(Y) \ldots \tilde{g}_r(Y)$. Suppose f is computable by a circuit of size s and depth Δ. Then, for any $B \geq 0$, we can compute (in time $\mathrm{poly}(s, n, d, B)$) circuits of size $\mathrm{poly}(s, d, n, B)$ and depth $\Delta + O(1)$ for polynomials $g_1(T, Y), \cdots, g_r(T, Y) \in \mathbb{F}[\mathbf{X}][T, Y]$, each of which are monic in Y, such that for every $i \in [r]$, $g_i(0, Y) = \tilde{g}_i(Y)$, and*

$$f(T, Y) = g_1(T, Y) \cdots g_r(T, Y) \bmod T^B.$$

[6] Technically, we will get rational functions over $\{\beta_j : \tilde{g}_i(\beta_j) = 0\}$, where the β_js can be thought of as formal variables, and both the numerator and the denominator have constant-depth circuits. But this can be handled to eventually get constant-depth division-free circuits for the Hensel lifts, based on the ideas discussed in [BKR+26a, Section 5].

2.6 Standard algorithms in linear algebra

We summarize some well-known and useful facts from Linear Algebra below. The reader can refer [Sho09, Sections 14.4–14.5] for a more thorough treatment of the topic.

Lemma 2.15. *Given a matrix $M \in \mathbb{F}^{m_1 \times m_2}$ as input, there is an algorithm that computes a basis for $\ker(M)$ and performs atmost $\mathrm{poly}(m_1, m_2)$ field operations.*

Proof Sketch. We can compute $\ker(M)$ by Gaussian Elimination. □

Lemma 2.16. *Let $V, W \subseteq \mathbb{F}^m$ be two subspaces specified by their bases $\mathcal{B}_1$ and $\mathcal{B}_2$ respectively, then there is an algorithm to compute a basis $\mathcal{B}$ for the space $V \cap W$. Moreover, the algorithm performs at most $\mathrm{poly}(m, |\mathcal{B}_1|, |\mathcal{B}_2|)$ many field operations.*

Proof Sketch. Find a basis for the kernel of the matrix $[\, A \;\; -B \,]$. Suppose $(x_1, y_1), \ldots, (x_t, y_t)$ is the kernel of the matrix obtained from Lemma 2.15. Then, $Ax_1, \ldots, Ax_t$ is the required basis of $V \cap W$. □

Lemma 2.17. *Let $V \subseteq \mathbb{F}^m$ be a vector space with basis $\mathcal{B}$. Suppose V has a basis $\mathcal{B}'$ such that $\mathcal{B}' \subseteq \{0,1\}^m$ and every element in $\mathcal{B}'$ has disjoint support. Then such a basis $\mathcal{B}'$ is unique. Moreover, given $\mathcal{B}$ there is an algorithm which does at most $\mathrm{poly}(m)$ field operations and outputs $\mathcal{B}'$.*

Proof Sketch. Let A, B be matrices whose rows are $\mathcal{B}$ and $\mathcal{B}'$ respectively. Suppose the rows in B are ordered such that for rows i, j (with $i < j$), the first 1 in row j does not appear before the first 1 in row i. The lemma follows by noticing that B is the reduced row echelon form (rref) of A and rref of a matrix can be computed efficiently. □

3 Combining power-series into actual factors

For this section, using Lemma 2.8 if required, we may assume that $f(T, Y) \in \mathbb{F}[\mathbf{X}][T, Y]$ ($|\mathbf{X}| = n$) such that, (i) $\deg_Y(f) \leq d$ (ii) $\deg(f) \leq D$ (iii) f is monic in Y and square-free (iv) $f(0, Y) \in \mathbb{F}[Y]$ is square-free.

Let $f(T, Y) = f_1(T, Y) \cdots f_k(T, Y)$ be the irreducible factorisation (over $\mathbb{F}$) of f that we are trying to compute. Let the power-series Y-roots of f be $\varphi_1(T), \ldots, \varphi_d(T) \in \overline{\mathbb{F}}[\mathbf{X}][\![T]\!]$. Let the irreducible factorisation of $f(0, Y)$ over $\mathbb{F}$ be

$$f(0, Y) = \tilde{g}_1(Y) \cdot \cdots \cdot \tilde{g}_r(Y)$$

which is also equal to $\prod_{i \in [d]} (Y - \varphi_i(0))$ over $\overline{\mathbb{F}}$. Using the $\tilde{g}_i(Y)$s, we compute the Hensel lifts (using Theorem 2.14) $g_1(T, Y), \ldots, g_r(T, Y)$ satisfying

$$f(T, Y) = g_1(T, Y) \cdots g_r(T, Y) \bmod T^B.$$

where $B > 2D^2$. Each irreducible factor f_i corresponds to a product of some block of g_j's above. Thus, $[r] = S_1 \sqcup \cdots \sqcup S_k$ such that $f_i = \prod_{j \in S_i} g_j \bmod T^B$. Our goal is to try and find the blocks $S_1 \sqcup \cdots \sqcup S_k$. The following is a homogeneous linear system that would allow us to find these blocks.

Consider the following homogeneous linear system over indeterminates $\mu_1, \cdots, \mu_r \in \mathbb{F}(\mathbf{X})$

$$\mu_1 \cdot (\partial_Y g_1) \cdot g_2 \cdots g_r + \cdots + \mu_r \cdot g_1 \cdots g_{r-1} \cdot (\partial_Y g_r) = Q(T, Y) \bmod T^B$$

for some polynomial $Q(T,Y)$ with $\deg_Y Q < \deg_Y f = d$ and $\deg Q \le \deg f = D$. This is equivalent to ensuring that the coefficient of all $T^a Y^b$ of $\mu_1 \cdot (\partial_Y g_1) \cdot g_2 \cdots g_r + \cdots + \mu_r \cdot g_1 \cdots g_{r-1} \cdot (\partial_Y g_r)$ is zero whenever $a + b > D$ or $b \ge d$, which are homogeneous linear equations in the indeterminates $\mu_1, \cdots, \mu_r$.

Theorem 3.1 ([SSKS91, SS93, BLS+04, Lec06, CL07, Lec10]). *Suppose $f(T,Y) \in \mathbb{F}[\mathbf{X}][T,Y]$ satisfies the conditions above and suppose $f(T,Y) = g_1(T,Y) \cdots g_r(T,Y) \bmod T^B$ with $B > 2D^2$ where $\deg f \le D$. Let $\mathcal{L}_B$ denote the solution space to the above system $(\mu_1, \cdots, \mu_r) \in \mathbb{F}(\mathbf{X})^r$. Then,*

$$\mathcal{L}_B = \operatorname{span}\{\mathbf{1}_{S_1}, \ldots, \mathbf{1}_{S_k}\}.$$

We present a short proof here for completeness.

Proof. We first prove the easier direction of showing that each $\mathbf{1}_{S_i}$ is a solution to the above system for $i \in [k]$. Indeed, since $\prod_{j \in S_i} g_j = f_i \bmod T^B$, we have

$$\sum_{j \in S_i} (\partial_Y g_j) \prod_{k \ne j} g_k = (\partial_Y f_i) \cdot \prod_{k \ne i} f_k \bmod T^B$$

and the RHS is clearly a polynomial of Y-degree at most $d-1$ and total degree at most D.

Suppose $\mu = (\mu_1, \cdots, \mu_r) \in \mathbb{F}(\mathbf{X})^r$ be an arbitrary nonzero solution to the system

$$\mu_1 \cdot (\partial_Y g_1) g_2 \cdots g_r + \cdots + \mu_r \cdot g_1 \cdots g_{r-1} (\partial_Y g_r) = Q(T,Y) \bmod T^B$$

with $\deg_Y Q < d$ and $\deg Q \le D$. We will show that μ is constant on coordinates in S_i. For simplicity, let us assume $1, 2 \in S_1$ so that $g_1 g_2 \mid f_1$, and we shall show that $\mu_1 = \mu_2$. Pick any root $\varphi_\ell(T)$ of $g_1(T,Y)$. Then,

$$\begin{aligned} Q(T, \varphi_\ell) &= \mu_1 \cdot (\partial_Y g_1)(T, \varphi_\ell) \cdot (g_2 \cdots g_r)(T, \varphi_\ell) \bmod T^B \\ &= \mu_1 \cdot (\partial_Y f_1)(T, \varphi_\ell) \cdot (f_2 \cdots f_k)(T, \varphi_\ell) \end{aligned}$$

Therefore, for $G_1(T,Y) := Q(T,Y) - \mu_1 \cdot (\partial_Y f_1) f_2 \cdots f_k$, we have $G_1(T, \varphi_\ell) = 0 \bmod T^B$. Using Poisson Formula (Lemma 2.4), this implies that,

$$\operatorname{Res}_Y(G_1(T,Y), f_1(T,Y)) = (-1)^{\deg_Y(G_1)\deg_Y(f_1)} \operatorname{lc}_Y(f_1)^{\deg_Y(G_1)} \prod_{j : f_1(T,\varphi_j) = 0} G_1(T, \varphi_j) = 0 \bmod T^B.$$

On the other hand, by Lemma 2.5, $\operatorname{Res}_Y(G_1, f_1)$ is a polynomial in T of degree at most $2D^2$. Hence,

$$\operatorname{Res}_Y(G_1(T,Y), f_1(T,Y)) = 0 \implies f_1(T,Y) \mid G_1(T,Y).$$

Along exactly the same lines, we can also obtain that $G_2(T,Y) := Q(T,Y) - \mu_2 \cdot (\partial_Y f_1) f_2 \cdots f_k$ is divisible by f_1 as well. This implies that f_1 must divide $(\mu_1 - \mu_2) \cdot (\partial_Y f_1) f_2 \cdots f_k$ and this is possible only if $\mu_1 = \mu_2$. ∎

In other words, $\mathcal{L}_B$ is precisely the space that is spanned by the indicator vectors corresponding to the blocks corresponding to irreducible factors of f.

3.1 Computing the kernel of the system using whitebox PIT

In the previous section, we saw that the main *recombination step* in factorization is reduced to computing the kernel of a certain matrix of polynomials. Moreover the kernel is known to be spanned by vectors in the base field.

In this section, we will see a general algorithm (by reduction to white-box PIT) for computing the $\mathbb{F}$-kernel of any matrix $M(\mathbf{X}) \in \mathbb{F}(\mathbf{X})^{m_1 \times m_2}$ (where $|\mathbf{X}| = n$) whose entries consist of rational functions represented by some structured circuits. The $\mathbb{F}$-kernel of $M(\mathbf{X})$, denoted by $\ker_{\mathbb{F}}(M(\mathbf{X}))$, is the subspace of all vectors $v \in \mathbb{F}^{m_2}$ such that $M(\mathbf{X})v = 0$.

Theorem 3.2. *Suppose we have an oracle for PIT for arbitrary linear combinations of entries of the matrix $M(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]^{m_1 \times m_2}$. Then, the $\mathbb{F}$-kernel of $M(\mathbf{X})$ can be computed with at most $\mathrm{poly}(n, d, m_1, m_2)$ many operations (including calls to the PIT oracle), where d is an upper bound on the degree of the polynomial entries in M.*

Proof. The proof of the theorem is the following algorithm. Let us begin with $V = \mathbb{F}^{m_2}$ and we will keep pruning V down to $\ker_{\mathbb{F}}(M(\mathbf{X}))$. In the general case, let $V \leq \mathbb{F}^{m_2}$ be spanned by $\{u_1, \cdots, u_\ell\} \subseteq \mathbb{F}^{m_2}$, and we will always maintain the invariant that $\ker_{\mathbb{F}}(M(\mathbf{X})) \subseteq V$.

For each $i \in [\ell]$, we wish to check if $u_i \in \ker_{\mathbb{F}}(M(\mathbf{X}))$ or not. This amounts to checking if $M(\mathbf{X})u_i = 0$ or not and this amounts to m_1-many PITs of $\mathbb{F}$-linear combinations of the entries of $M(\mathbf{X})$. If one of these PITs is nonzero, we use the search-to-decision reduction for PIT (Lemma 2.2) to compute, in time $\mathrm{poly}(n, d)$, a point $\mathbf{a}$ such that $M(\mathbf{a})u_i \neq 0$, and prune V to $V' = V \cap \ker(M(\mathbf{a}))$ (by using Lemma 2.15 and Lemma 2.16). Note that V' continues to contain $\ker_{\mathbb{F}}(M(\mathbf{X}))$ but is now a smaller dimension subspace of V. However, if it happens that $M(\mathbf{X})u_i = 0$ for all $i \in [\ell]$, then every u_i is infact in $\ker_{\mathbb{F}}(M(\mathbf{X}))$ and thus we can safely return $\{u_1, \cdots, u_\ell\}$ as a $\mathbb{F}$-basis for $\ker_{\mathbb{F}}(M(\mathbf{X}))$.

This immediately yields a correct and efficient algorithm for computing $\ker_{\mathbb{F}}(M(\mathbf{X}))$, provided we have access to a PIT oracle for $\mathbb{F}$-linear combination of entries of $M(\mathbf{X})$. ■

We summarise the above proof into the following algorithm.

Algorithm 1: Computing the $\mathbb{F}$-kernel of a matrix of arithmetic circuits

Input: $M(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]^{m_1 \times m_2}$ (where $|X| = n$), where each matrix entry is from a structured class of circuits.
Output: Basis for $\ker_{\mathbb{F}}(M(\mathbf{X}))$
Oracle access: PIT algorithm for $\mathbb{F}$-linear combinations of entries of $M(\mathbf{X})$.

```
1 Initialise V = F^{m_2} given by basis B = {e_1, ..., e_{m_2}} ⊆ F^{m_2}.
2 for u_i ∈ B do
3 |   Run m_1 oracle calls (one for each row) to check if M(X)u_i = 0.
4 |   if M(X)u_i ≠ 0 then
5 |   |   Use the PIT oracle to find a point a ∈ F^{|X|} such that M(a)u_i ≠ 0.
6 |   |   Replace V with V ∩ ker(M(a)), and B with a basis of V. Go to line 2.
7 if M(X)u_i = 0 for all u_i ∈ B then
8 |   return the B as the basis for ker_F M(X).
```

4 Factorization using White-box PIT Algorithm

In this section, we will design the final factorization algorithm for constant-depth circuits which makes oracle calls to a white-box PIT algorithm. As we saw in the proof overview (Section 1.1) and Section 3, the logarithmic derivative method gives us a linear system, which specifies the appropriate truncated power series roots to combine together to obtain the corresponding irreducible factors.

Theorem 4.1. *Let $n, D, s, \Delta \in \mathbb{N}$, $\mathbb{F}$ be the field of rational numbers.*

There is an algorithm that takes as input an n-variate degree D polynomial $F \in \mathbb{F}[\mathbf{X}]$ computed by a circuit of size s and depth Δ, requires oracle calls to a whitebox PIT algorithm for $\mathrm{poly}(s, n, D)$-size and $\Delta + O(1)$-depth circuits, runs in $\mathrm{poly}(n, s, d, D)$ time, and outputs constant depth circuits for all the irreducible factors of F together with their multiplicities.

Remark 4.2. While stated and proved here for constant depth circuits, we note that Theorem 4.1 also extends to other natural algebraic models like algebraic formulas, algebraic branching programs and algebraic circuits, with very minimal changes to the proof.

Remark 4.3. One can also keep track of the bit complexity of the rational numbers that show up in the computation in the following proof. Since all our computations involve "constant-depth computations" over input coefficients, the bit-complexities for any numbers generated in the algorithm will remain polynomially bounded in the input parameters. We skip this bookkeeping for clarity of exposition.

Proof of Theorem 4.1. For ease of exposition, we will work with an $(n+1)$-variate polynomial $F(\mathbf{X}, Y)$, where the last variable is specially denoted by Y. Suppose F is computable by a circuit of size s and depth Δ. By the discussion in Lemma 2.8, we can reduce factoring $F(\mathbf{X}, Y)$ to factoring a polynomial $f(T, Y) \in \mathbb{F}[\mathbf{X}][T, Y]$ ($|\mathbf{X}| = n$) such that, (i) $\deg_Y(f) \leq d$ (ii) $\deg(f) \leq D$ (iii) f is monic in Y and square-free (iv) $f(0, Y) \in \mathbb{F}[Y]$ is square-free (v) $f(T, Y)$ is computable by a circuit of size $\mathrm{poly}(s, n, D)$ and depth $\Delta + O(1)$. So henceforth, we will aim to factor $f(T, Y)$.

Suppose $f(T, Y)$ factorizes as follows,

$$f(T, Y) = f_1(T, Y) \cdot f_2(T, Y) \cdots f_k(T, Y). \tag{4.1}$$

where (i) for all $i \in [k]$, f_i is an irreducible polynomial in $\mathbb{F}(X)[T, Y]$. And, suppose $f(0, Y)$ factorizes as,

$$f(0, Y) = \tilde{g}_1(Y) \cdot \tilde{g}_2(Y) \cdots \tilde{g}_r(Y)$$

where for each $i \in [r]$, $\tilde{g}_1, \tilde{g}_2, \ldots, \tilde{g}_r \in \mathbb{F}[Y]$ are irreducible polynomials.

We can obtain this factorization of $f(0, Y)$ in $\mathrm{poly}(n, s, D)$ time as follows: Obtain coefficients of $f(0, Y)$ from C using the interpolation lemma (Lemma 2.1) in $\mathrm{poly}(n, s, D)$ time. Then from the coefficients of $f(0, Y)$, we invoke the efficient deterministic univariate factorization algorithm over rationals in [LLL82] to factor $f(0, Y)$ over $\mathbb{F}$ (Theorem 2.7).

Then, we Hensel lift from $\tilde{g}_1, \tilde{g}_2, \ldots, \tilde{g}_r \in \mathbb{F}[Y]$ to obtain the following factorization,

$$f(T, Y) = g_1(T, Y) \cdots g_r(T, Y) \bmod T^B$$

where $B = 2D^2 + 1$ and $g_i(T, Y) \in \mathbb{F}(X)[T, Y]$. By Theorem 2.14, we can build circuits of size $\mathrm{poly}(s, n, D)$ and depth $\Delta + O(1)$ for $g_i(T, Y)$ over the base field. Moreover, this construction requires at most $\mathrm{poly}(s, n, D)$ arithmetic operations.

As observed in Section 3, we would like to find a partition $[r] = S_1 \sqcup \cdots \sqcup S_k$ such that

$$f_i := \left(\prod_{j \in S_i} g_j \right) \bmod T^{D+1}.$$

By Theorem 3.1, using the circuits for $g_i(T, Y)$, we can build a matrix $M \in \mathbb{F}[\mathbf{X}]^{\mathrm{poly}(d,D) \times \mathrm{poly}(d,D)}$ whose $\mathbb{F}$-kernel is $\mathrm{span}_{\mathbb{F}} \{\mathbf{1}_{S_1}, \ldots, \mathbf{1}_{S_k}\}$. From the description of the linear system in Theorem 3.1, it is clear that this matrix can be constructed with at most $\mathrm{poly}(n, s, D)$ many arithmetic operations.

By Theorem 3.2, we can find a basis $\mathcal{B}$ for the $\mathbb{F}$-kernel of M with at most $\mathrm{poly}(n, s, D)$ many arithmetic operations (with oracle calls to white-box PIT algorithms for $\mathrm{poly}(s, d, n)$ size and $\Delta + O(1)$ circuits). By Lemma 2.17, we can obtain $\mathbf{1}_{S_1}, \ldots, \mathbf{1}_{S_k}$ from the $\mathcal{B}$ in $\mathrm{poly}(d)$ operations. Thus, we can obtain circuits of size $\mathrm{poly}(n, s, D)$ and depth $\Delta + O(1)$ for

$$f_i := \left(\prod_{j \in S_i} g_j \right) \bmod T^{D+1}$$

from the circuits of g_j, and clearly this can be done with at most $\mathrm{poly}(n, s, D)$ many arithmetic operations.

To recover the multiplicity of an irreducible factor f_i, we can perform divisibility tests to compute the highest power of f_i which divides f. Previous works (see [For15, BKR+26a]) discuss the reduction of divisibility testing of constant-depth circuits to PIT for constant-depth circuits, and it is easy to observe that this reduction also works in the whitebox setting.

This completes the proof of the theorem. ■

References

[AW26] Robert Andrews and Avi Wigderson. Constant-depth arithmetic circuits for linear algebra problems. *SIAM Journal on Computing*, pages FOCS24–236–FOCS24–295, 2026.

[BKR+25] Somnath Bhattacharjee, Mrinal Kumar, Varun Ramanathan, Ramprasad Saptharishi, and Shubhangi Saraf. Deterministic factorization of constant-depth algebraic circuits in subexponential time. In *2025 IEEE 66th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 684–709, 2025.

[BKR+26a] Somnath Bhattacharjee, Mrinal Kumar, Shanthanu S. Rai, Varun Ramanathan, Ramprasad Saptharishi, and Shubhangi Saraf. Closure under factorization from a result of furstenberg. In *Proceedings of the 58th Annual ACM Symposium on Theory of Computing*, STOC '26, page 174–185, New York, NY, USA, 2026. Association for Computing Machinery.

[BKR+26b] Somnath Bhattacharjee, Mrinal Kumar, Shanthanu S. Rai, Varun Ramanathan, Ramprasad Saptharishi, and Shubhangi Saraf. Constant-Depth Circuits for Polynomial GCD over Any Characteristic. In Dana Moshkovitz, editor, *41st Computational Complexity Conference (CCC 2026)*, volume 383 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 16:1–16:21, Dagstuhl, Germany, 2026. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[BLS+04] A. Bostan, G. Lecerf, B. Salvy, É. Schost, and B. Wiebelt. Complexity issues in bivariate polynomial factorization. In *Proceedings of the 2004 International Symposium on Symbolic and Algebraic Computation*, ISSAC '04, page 42–49, New York, NY, USA, 2004. Association for Computing Machinery.

[BSV26] Vishwas Bhargava, Shubhangi Saraf, and Ilya Volkovich. Linear independence, alternants and applications. *SIAM Journal on Computing*, 55(1):28–64, 2026.

[Bür04] Peter Bürgisser. The complexity of factors of multivariate polynomials. *Foundations of Computational Mathematics*, 4(4):369–396, 2004. Corrected version available on arXiv, 2018.

[CKS19] Chi-Ning Chou, Mrinal Kumar, and Noam Solomon. Closure results for polynomial factorization. *Theory of Computing*, 15(13):1–34, 2019.

[CKW11] Xi Chen, Neeraj Kayal, and Avi Wigderson. Partial derivatives in arithmetic complexity and beyond. *Foundations and Trends in Theoretical Computer Science*, 6(1–2):1–138, 2011.

[CL07] Guillaume Chèze and Grégoire Lecerf. Lifting and recombination techniques for absolute factorization. *Journal of Complexity*, 23(3):380–420, 2007.

[For15] Michael A. Forbes. Deterministic divisibility testing via shifted partial derivatives. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 451–465, 2015.

[FS15] Michael A. Forbes and Amir Shpilka. Complexity theory column 88: Challenges in polynomial factorization. *ACM SIGACT News*, 46(4):32–49, 2015.

[Kal89] Erich Kaltofen. Factorization of Polynomials Given by Straight-Line Programs. In *Randomness and Computation*, pages 375–412. JAI Press, 1989.

[Kay10] Neeraj Kayal. Algorithms for arithmetic circuits. Technical Report TR10-073, Electronic Colloquium on Computational Complexity, 2010.

[KI04] Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004.

[KRS24] Mrinal Kumar, Varun Ramanathan, and Ramprasad Saptharishi. *Deterministic Algorithms for Low Degree Factors of Constant Depth Circuits*, pages 3901–3918. SIAM, 2024.

[KRSV26] Mrinal Kumar, Varun Ramanathan, Ramprasad Saptharishi, and Ben Lee Volk. Towards deterministic algorithms for constant-depth factors of constant-depth circuits. *ACM Trans. Comput. Theory*, 18(1), February 2026.

[KS19] Mrinal Kumar and Ramprasad Saptharishi. Hardness-randomness tradeoffs for algebraic computation. *Bulletin of EATCS*, 3(129), 2019.

[KSS15] Swastik Kopparty, Shubhangi Saraf, and Amir Shpilka. Equivalence of polynomial identity testing and polynomial factorization. *computational complexity*, 24(2):295–331, 2015.

[KT90] Erich Kaltofen and Barry M. Trager. Computing with polynomials given by black boxes for their evaluations: Greatest common divisors, factorization, separation of numerators and denominators. *Journal of Symbolic Computation*, 9(3):301–320, 1990. Computational algebraic complexity editorial.

[Lec06] Grégoire Lecerf. Sharp precision in Hensel lifting for bivariate polynomial factorization. *Mathematics of Computation*, 75(254):921–933, 2006.

[Lec10] Grégoire Lecerf. New recombination algorithms for bivariate polynomial factorization based on Hensel lifting. *Applicable Algebra in Engineering, Communication and Computing*, 21(2):151–176, 2010.

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

[LST25] Nutan Limaye, Srikanth Srinivasan, and Sébastien Tavenas. Superpolynomial lower bounds against low-depth algebraic circuits. *Journal of the ACM*, 72(4):1–35, 2025.

[Ope26] OpenAI. Deterministic polynomial factorization over prime fields. OpenAI Math Release preprint, October 2026. October 4, 2026.

[RS05] Ran Raz and Amir Shpilka. Deterministic polynomial identity testing in non-commutative models. *Comput. Complex.*, 14(1):1–19, 2005.

[Sho09] Victor Shoup. *A Computational Introduction to Number Theory and Algebra*. Cambridge University Press, 2 edition, 2009.

[SS93] Tateaki Sasaki and Mutsuko Sasaki. A unified method for multivariate polynomial factorizations. *Japan Journal of Industrial and Applied Mathematics*, 10(1):21–39, 1993.

[SSKS91] Tateaki Sasaki, Masayuki Suzuki, Miroslav Kolář, and Mutsuko Sasaki. Approximate factorization of multivariate polynomials and absolute irreducibility testing. *Japan Journal of Industrial and Applied Mathematics*, 8(3):357–375, 1991.

[ST21] Amit Sinhababu and Thomas Thierauf. Factorization of polynomials given by arithmetic branching programs. *Computational Complexity*, 30(2):15:1–15:47, 2021.

[SY10] Amir Shpilka and Amir Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5:207–388, 3 2010.

[vzGG13] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, 3 edition, 2013.

[vzGK85] Joachim von zur Gathen and Erich Kaltofen. Factoring sparse multivariate polynomials. *Journal of Computer and System Sciences*, 31(2):265–287, 1985.

[Wei] Eric W. Weisstein. Logarithmic derivative. MathWorld—A Wolfram Resource.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il