

# A primer on the closure of algebraic complexity classes under factoring

C.S. Bhargav<sup>✉</sup> and Prateek Dwivedi<sup>✉</sup> and Nitin Saxena<sup>✉</sup>

**Abstract** Polynomial factorisation is a fundamental problem in computational algebra. Over the past half century, a variety of algorithmic techniques have been developed to tackle different variants of this problem. In parallel, algebraic complexity theory classifies polynomials into complexity classes based on their computational hardness. This raises a natural question: Are these complexity classes closed under factorisation?

In this survey, we revisit pivotal techniques in polynomial factorisation: Hensel lifting, Newton iteration, and Lagrange inversion. These techniques have played an essential role in resolving key factoring questions in algebraic complexity for more than half a century. We examine and organise the known results through the lens of these techniques, discussing their underlying mathematical equivalence while reflecting on how their applications vary depending on the problem context.

We focus on prominent algebraic complexity classes, including VP (circuits of polynomial size and degree), its closure  $\overline{VP}$ , the class VNP (verifier circuits of polynomial size and degree), VBP (polynomial-size branching programs), VF (polynomial-size formulas), and  $VP_{nb}$  (circuits of polynomial size and exponential degree). We also discuss bounded-depth circuits and sparse polynomials. Along the way, we highlight several unresolved open problems.

---

C.S. Bhargav

University of Regensburg, Germany, and IIT Kanpur, India, e-mail: [bhargav.cs@ur.de](mailto:bhargav.cs@ur.de)

Prateek Dwivedi

ITU Copenhagen, Denmark, e-mail: [prdw@itu.dk](mailto:prdw@itu.dk)

Nitin Saxena

IIT Kanpur, India, e-mail: [nitin@cse.iitk.ac.in](mailto:nitin@cse.iitk.ac.in)

## Contents

1	Introduction .....	3
1.1	Univariate factoring .....	3
1.2	Fine-grained developments .....	8
2	Model of computation .....	8
2.1	Structural results .....	10
3	Applications of factoring .....	11
3.1	Equivalence to identity testing .....	11
3.2	Hardness vs Randomness .....	13
3.3	Other applications .....	16
4	Factoring via Hensel lifting .....	17
4.1	Polynomials in the dense representation .....	18
4.2	Polynomials over integers .....	22
4.3	Polynomials over $p$ -adic numbers .....	23
4.4	Small algebraic circuits .....	25
4.5	Algebraic branching programs .....	30
4.6	Explicit polynomials .....	33
5	Factoring via Newton iteration .....	36
5.1	Low depth circuits .....	39
5.2	High degree circuits .....	43
5.3	Algebraic approximation .....	45
6	Newton iteration to Hensel lifting and back .....	49
7	Factoring via Lagrange Inversion .....	50
7.1	Small algebraic formulas & Shallow circuits .....	53
8	Factoring via Convex Geometry .....	57
8.1	Sparse polynomials .....	57
	Acknowledgements .....	61

## 1 Introduction

The problem of finding a nontrivial factor of a polynomial is a classical and fundamental problem, with a rich history spanning centuries [Gat06]. Kaltofen [Kal82; Kal90; Kal92] gave a thorough treatment of several foundational results, while von zur Gathen and Panario [GP01] focused on factorisation over finite fields. These developments are also covered in greater depth in standard textbooks such as [GG13] and [Sho09].

In this survey, we focus on the core techniques and ideas that drive factorisation results within algebraic circuit complexity. Our aim is to illustrate how these tools have contributed to resolving long-standing questions in the field. Notably, Forbes and Shpilka [FS15] have provided an accessible exposition of the high-level ideas behind some factoring algorithms, emphasising their relevance to other problems in algebraic complexity. Building on this, our survey aims to delve into the technical intricacies of these results by organising them according to the underlying methods. More than algorithms, we will be concerned with whether the factors of a ‘structured’ polynomial are also structured. We begin with a gentle introduction to each technique, followed by a discussion of the key results that it enables. The simplest case, perhaps, is when the polynomial is of a single variable.

### 1.1 Univariate factoring

The polynomial ring  $\mathbb{F}[x]$  over a field  $\mathbb{F}$  is a well-known unique factorisation domain. This ensures that any non-constant polynomial  $f \in \mathbb{F}[x]$  can be uniquely decomposed into the product of a constant and a collection of irreducible polynomials. Consequently, computing this decomposition presents a very natural computational challenge. The quest to factor polynomials traces its roots back to ancient Babylonian times, and it has consistently been considered a problem of profound importance due to its extensive theoretical and practical applications.

#### ? Problem (1.1.1)

Given a univariate polynomial  $f(x)$  over a field  $\mathbb{F}$ , compute pairwise distinct irreducible polynomials  $f_1, \dots, f_r \in \mathbb{F}[x]$  and a tuple  $(e_1, \dots, e_r) \in \mathbb{N}^r$  such that  $f = f_1^{e_1} \cdots f_r^{e_r}$ .

---

Considered over the rational numbers  $\mathbb{Q}$ , the polynomial  $x^2 - 2$  is irreducible, whereas it factors as  $(x - 3)(x - 4) \pmod{7}$ . Evidently, the problem critically depends on the field  $\mathbb{F}$ . We begin by considering factorisation over a *finite field*  $\mathbb{F}_q$  of order  $q = p^a$ , for some prime  $p$ . Despite its simplicity, this case involves several non-trivial ideas that serve as the foundation for more general algorithms, including those over the rationals and algebraic number fields.

The input polynomial  $f(x) = \sum_{i=0}^d c_i x^i \in \mathbb{F}_q[x]$  is given in the *dense representation* as a list of (coefficient, exponent) pairs  $(c_i, i)$  for all  $0 \leq i \leq d$ . Let  $t \in \mathbb{F}_p[y]$  be some irreducible polynomial of degree  $a$ , such that  $\mathbb{F}_q \cong \mathbb{F}_p[y]/\langle t(y) \rangle$ . Given such a  $t$ , arithmetic operations in  $\mathbb{F}_q$  can be performed in time  $\text{poly}(\log q)$ . Hence, we assume that the irreducible polynomial  $t$  is part of the input to the algorithm. A thorough exposition of computational issues in finite fields is available in [GS13] and [AB09, Section A.4]. The goal is then to solve univariate factoring over  $\mathbb{F}_q$  in time polynomial in  $d$  and  $\log q$ . We will be describing two classical algorithms, each offering a different perspective on the problem.

An important subroutine that underlies most of the factoring algorithms is computing the greatest common divisor (GCD) of two polynomials. The classical Euclidean algorithm can be used to efficiently compute the gcd [GG13, Section 3]. Throughout this text, we use the notation  $\text{poly}(n)$  to denote any function of the form  $O(n^c)$  for some constant  $c$ . In particular, when we say that an algorithm takes  $\text{poly}(n)$  operations over a field  $\mathbb{F}$ , we refer to a count of field-level arithmetic operations performed within  $\mathbb{F}$ . This complexity measure abstracts away the bit-level details of the field operations, which depend on the representation of  $\mathbb{F}$ .

**Lemma 1.1 (GCD)** *Let  $\mathbb{F}$  be a field and  $f, g \in \mathbb{F}[x]$  be polynomials of degree at most  $d$ . Then,  $\text{gcd}(f, g)$  can be computed in  $\text{poly}(d)$  operations over  $\mathbb{F}$ . Moreover, the algorithm returns  $a, b \in \mathbb{F}[x]$  such that  $a \cdot f + b \cdot g = \text{gcd}(f, g)$ .*

Most factoring algorithms over finite fields, and beyond, follow a standard three-step structure (see Figure 1):

1. **Square-free Factorisation:** Note that each  $e_i \geq 1$  in Problem 1.1.1. The *square-free* part of  $f$ , denoted by  $\text{rad}(f) = f_1 \cdots f_r$ , is called the *radical* of  $f$ . This can be efficiently obtained by dividing  $f$  by  $\text{gcd}(f, \partial_x f)$ . If  $\partial_x f = 0$  then  $f$  is a perfect  $p$ -th power. We can apply the Frobenius automorphism to extract its  $p$ -th root, effectively computing  $f^{1/p}$ .
2. **Distinct Degree Factorisation:** The goal is to decompose  $\text{rad}(f)$  into a product  $\tilde{f}_1 \cdots \tilde{f}_d$ , where each  $\tilde{f}_i$  is the product of all irreducible degree- $i$  factors of  $f$ . A

classical result from field theory shows that

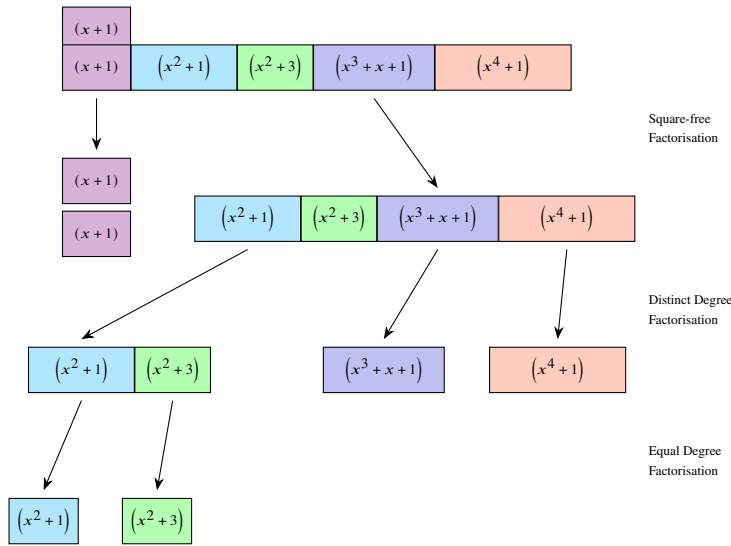
$$\tilde{f}_i = \gcd(x^{q^i} - x, f),$$

which can be computed efficiently using repeated squaring and [Lemma 1.1](#). The efficiency arises from two observations: (1) computing  $\gcd(x^{q^i} - x, f)$  is equivalent to  $\gcd(f, (x^{q^i} - x) \bmod f)$ , and (2) the powers  $x^{q^i} \bmod f$  can be computed iteratively via

$$x^{q^i} \equiv (x^{q^{i-1}})^q \pmod{f}.$$

In practice, for uniformly random polynomials of high degree, this step often dominates the running time of the overall factoring process.

3. **Equal Degree Factorisation:** After the above preprocessing, we are left with the task of factoring a polynomial into irreducibles of the same degree. Two classical algorithms for this purpose are Cantor–Zassenhaus and Berlekamp’s algorithm. While Berlekamp’s method is deterministic, Cantor–Zassenhaus is randomised and tends to scale better for large finite fields. Both approaches fundamentally exploit the structure given by the Chinese Remainder Theorem.



**Fig. 1** Polynomial factorisation steps. The input polynomial has two degree-1 factors, two degree-2 factors, one degree-3 factor, and one degree-4 factor. Adapted from [GS92].

In the remainder of this section, we focus on algorithms for *Equal Degree Factorisation*. After performing the two preprocessing steps, we are left with factoring a polynomial of the form  $f = f_1 \cdot f_2 \cdot \dots \cdot f_r$ , where each  $f_i$  is an irreducible polynomial of the same degree.

### 1.1.1 Berlekamp's algorithm

At the core of Berlekamp's algorithm lies a fundamental number-theoretic observation: suppose there exists a polynomial  $h \in \mathbb{F}_q[x]$  such that

$$h^p - h \equiv 0 \pmod{f},$$

and  $1 \leq \deg(h) < \deg(f)$ . Then, by the identity  $h^p - h = \prod_{\alpha \in \mathbb{F}_q} (h - \alpha)$ , the polynomial  $f$  can be factored by computing  $\gcd(f, h - \alpha)$  for all  $\alpha \in \mathbb{F}_q$ .

To see that such a non-constant  $h$  exists, consider the Chinese Remainder Theorem isomorphism  $\mathbb{F}_q[x]/\langle f \rangle \cong \bigoplus_{i=1}^r \mathbb{F}_q[x]/\langle f_i \rangle$ , where the  $f_i$  are the distinct irreducible factors of the square-free  $f$  ( $r \geq 2$ ). Any polynomial  $h$  mapping to  $(\alpha_1, \dots, \alpha_r)$  under this isomorphism, with each  $\alpha_i \in \mathbb{F}_p$ , satisfies  $h^p \equiv h \pmod{f}$ . Such an  $h$  is non-constant precisely when not all  $\alpha_i$  are equal.

To find such an  $h$ , the algorithm considers the  $\mathbb{F}_p$ -vector space

$$V := \{h(x) : \deg(h) \leq d, h^p - h = 0 \pmod{f}\}. \quad (1)$$

Recall  $q = p^a$ . This is a subspace of  $\mathbb{F}_q[x]/\langle f \rangle$  of dimension at most  $\deg(f) \cdot a$ , and a basis for  $V$  can be efficiently computed using standard linear algebra over  $\mathbb{F}_p$ . Any non-trivial (i.e., non-constant) basis element  $h \in V$  yields a non-trivial factor of  $f$  via GCD computations.

Notably,  $q$ -many GCD computations would be inefficient. To avoid that, elements of  $\mathbb{F}_q$  are represented as vectors over  $\mathbb{F}_p$ . Notably, Berlekamp's algorithm does not require the input polynomial to have irreducible factors of the same degree; thus, the distinct-degree factorisation step can be bypassed. With careful implementation, the total runtime is bounded by  $\text{poly}(p, d, \log q)$ , making it a deterministic polynomial-time algorithm in small characteristic, e.g., when  $p = (da)^{O(1)}$ . For an accessible exposition, see the lecture notes by Kopparty [Kop14, Lecture 10].

### 1.1.2 Cantor–Zassenhaus Algorithm

As described earlier, Berlekamp’s algorithm is an efficient deterministic algorithm for fields of small characteristic. By contrast, the Cantor–Zassenhaus algorithm leverages randomness to offer a more efficient alternative for factoring over large-characteristic fields, such as when  $p = (da)^{\omega(1)}$ . The key idea is to choose a polynomial  $h \in \mathbb{F}_q[x]$  of degree at most  $d - 1$  uniformly at random, and check whether the following gcd yields a non-trivial factor:

$$\gcd\left(f, h^{\frac{q^d-1}{2}} - 1 \pmod{f}\right).$$

To see why this works, recall that  $f = f_1 \cdot f_2 \cdots f_r$ , obtained after the preprocessing steps. Since each  $f_i$  is irreducible of degree at most  $d$ , the Chinese Remainder Theorem yields an isomorphism:

$$\frac{\mathbb{F}_q[x]}{\langle f \rangle} \cong \frac{\mathbb{F}_q[x]}{\langle f_1 \rangle} \times \cdots \times \frac{\mathbb{F}_q[x]}{\langle f_r \rangle}.$$

Consider a random polynomial  $h \in \mathbb{F}_q[x]/\langle f \rangle$  that maps to  $(h_1, \dots, h_r)$  under this isomorphism, with the condition that  $\gcd(f, h) = 1$  (i.e., none of the  $h_i$  are zero, which would already yield a factor of  $f$ ). Then,  $h^{q^d} - h = 0$ . Assuming  $q$  is odd, it follows that

$$h \cdot \left(h^{\frac{q^d-1}{2}} - 1\right) \cdot \left(h^{\frac{q^d-1}{2}} + 1\right) = 0.$$

The identity  $h^{q^d} - h = \prod_{\alpha \in \mathbb{F}_{q^d}} (h - \alpha)$  implies that, excluding 0, exactly half the elements of  $\mathbb{F}_{q^d}$  satisfy  $h^{\frac{q^d-1}{2}} = 1$  and the other half satisfy  $h^{\frac{q^d-1}{2}} = -1$ . Under the CRT map, the components of  $h' := h^{\frac{q^d-1}{2}} - 1 \pmod{f}$  are therefore distributed such that each  $h'_i \in \{0, -2\}$ .

A non-trivial factor is obtained if  $h'$  contains a mix of both 0 and  $-2$  entries. If all  $h'_i = 0$ , then  $h' \equiv 0 \pmod{f}$  and  $\gcd(f, h')$  yields the trivial factor  $f$ . Similarly, if all  $h'_i = -2$ , then  $h'$  is coprime to  $f$  and the gcd yields 1. Since the components are chosen independently and at random, the probability of obtaining such a mixed vector is  $1 - (1/2)^{r-1} \geq 1/2$  (for  $r \geq 2$ ). Thus, computing  $\gcd(f, h^{\frac{q^d-1}{2}} - 1)$  yields a non-trivial factor of  $f$  with high probability.

When  $q$  is even, we instead compute  $h' = h + h^2 + h^{2^2} + \cdots + h^{2^{r-1}}$  and check whether  $\gcd(f, h')$  is non-trivial. This approach works because one can show that  $h'(h' + 1) = h^{2^r} + h$ , enabling a similar probabilistic argument. The algorithm relies primarily on computing gcds and repeated squaring. The probabilistic bounds imply

that, in expectation, only a constant number of repetitions (at most two) are required. Thus, the overall complexity of the Cantor–Zassenhaus algorithm is  $\text{poly}(d, \log q)$ .

### ? Problem (1.1.2)

Devise a deterministic algorithm that, given a univariate polynomial  $f \in \mathbb{F}_q[x]$  of degree  $d$ , computes its complete factorisation in time polynomial in  $d$  and  $\log q$ .

---

*Remark 1.2* It is unknown whether computing square roots modulo a prime  $p$  (i.e., determining  $x$  such that  $x^2 \equiv a \pmod{p}$ ) is achievable in deterministic time polynomial in  $\log p$ . Readers are referred to [Iva+12] for a detailed survey on this line of work.

## 1.2 Fine-grained developments

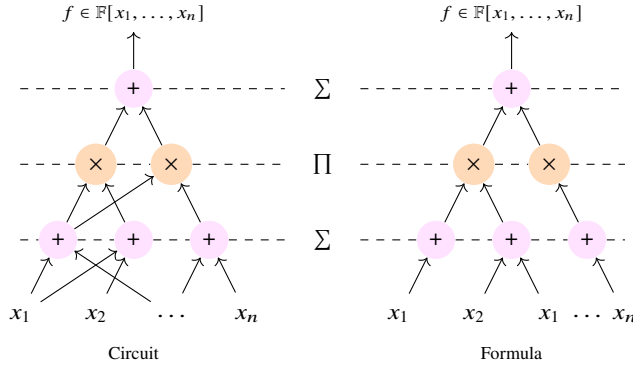
The complexity of univariate factoring via the Cantor–Zassenhaus algorithm is primarily dominated by modular exponentiation, requiring  $O(d^{2+o(1)} \log q)$  operations in  $\mathbb{F}_q$ . By optimising the Equal Degree Factorisation step, von zur Gathen and Shoup reduced this cost to  $O(d^{2+o(1)} + d^{1+o(1)} \log q)$  operations [GS92]. Kaltofen and Lobo subsequently matched this complexity by adapting Berlekamp’s algorithm to a black-box linear algebra model [KL94]. The quadratic barrier was breached by Kaltofen and Shoup, who utilised fast matrix multiplication to achieve a complexity of  $O(d^{1.815} \log q)$  [KS98]. More recently, Kedlaya and Umans leveraged a novel approach for modular composition to further reduce the complexity to  $O(d^{1.5+o(1)} + d^{1+o(1)} \log q)$  operations in  $\mathbb{F}_q$  [KU11].

## 2 Model of computation

We will mostly focus on *arithmetic/algebraic circuits*, a model very natural for computing multivariate polynomials in the variables  $\mathbf{x} := (x_1, \dots, x_n)$ <sup>1</sup> over a field  $\mathbb{F}$ . Introduced by Valiant [Val79a] to develop the algebraic analogue of NP-completeness [Val82], it has led to the development of a rich and varied theory of algebraic complexity (see [Bür24]).

---

<sup>1</sup> We will use bold letters to denote tuples of variables.



**Fig. 2** An algebraic circuit and a formula of depth 3.

**Definition 2.1 (Algebraic Circuits and Formulas)** An *algebraic circuit*, defined over a field  $\mathbb{F}$ , is a *layered directed acyclic graph* with alternating layers of ‘+’ and ‘x’ gates, and a single root, called the ‘output’ gate. The ‘input’ *leaf* gates are labelled by either a variable from  $x_1, \dots, x_n$  or a constant from  $\mathbb{F}$ . If the graph is a *tree*, then we call it a *formula*. See Figure 2 for an illustration.

A circuit computes a polynomial  $f \in \mathbb{F}[x]$  in the natural way: a ‘+’ gate sums up the polynomials from its children, whereas a ‘x’ gate computes their product, with the root finally computing  $f$ . The *size* of a circuit is the total number of vertices in the graph. The *depth* of the circuit is the number of layers in the circuit, or equivalently, the length of the longest path from the root to a leaf.

A *complexity class* in the algebraic setting comprises families (or sequences) of polynomials, where a family  $(f_n)_{n \in \mathbb{N}}$ , contains multivariate polynomials  $f_n$  over a field  $\mathbb{F}$ , and the number of variables in  $f_n$  grows polynomially with  $n$ .

In Boolean complexity, the notion of efficient computation is captured by the class P of problems solvable in polynomial time. The algebraic analogue over a field  $\mathbb{F}$  is  $VP_{\mathbb{F}}$  (for Valiant’s P, called  $p$ -computable by Valiant) and consists of all polynomial families where  $f_n$  has degree  $\text{poly}(n)$  and the smallest circuit (over  $\mathbb{F}$ ) computing  $f_n$  has size  $\text{poly}(n)$  (Definition 4.8). We will usually drop the field from the notation when the context makes it clear. Note that the notion of computation is *non-uniform* — the circuits of  $f_n$  for different  $n$  need not be related to one another. A prime example of a polynomial family in VP is  $(\text{Det}_n)$ , defined by the *determinant* of the  $n \times n$  symbolic matrix  $(x_{ij})_{1 \leq i, j \leq n}$ :

$$\text{Det}_n = \sum_{\sigma \in \mathcal{S}_n} \left( \text{sgn}(\sigma) \prod_{i=1}^n x_{i, \sigma(i)} \right).$$

The algebraic analogue of the class NP is called VNP. Informally<sup>2</sup>, it consists of polynomial families which are ‘explicit’, in the sense that given a monomial of  $f_n$ , we can compute the corresponding coefficient efficiently, say in polynomial time. It is not hard to show that  $\text{VP} \subseteq \text{VNP}$ , and the long-standing conjecture of Valiant [Val79a] is that there are explicit polynomial families that cannot be computed efficiently, i.e.,  $\text{VP} \subsetneq \text{VNP}$ . A prominent ‘explicit’ candidate for this separation is the family of *permanents*,

$$\text{Per}_n = \sum_{\sigma \in \mathcal{S}_n} \left( \prod_{i=1}^n x_{i, \sigma(i)} \right).$$

The determinant and permanent families essentially characterise the classes VP and VNP, respectively. Hence, Valiant’s conjecture is also sometimes called the Permanent versus Determinant problem [Agr06]. It is the algebraic version of Cook’s hypothesis [Coo71], the famous P vs. NP problem (see [AB09] for more details). There is a formal sense in which the VP vs. VNP problem is a ‘stepping stone’ towards the P vs. NP problem [Bür00b]. For details on the connection between Valiant’s and Cook’s hypotheses, and the progress on Valiant’s conjecture, we encourage readers to consult [BCS97; Bür99; Bür00a; SY10; CKW10; Mah14; Sap21].

## 2.1 Structural results

Algebraic circuits impose a combinatorial structure on the polynomials being computed. We will now list (without proofs) some structural properties of algebraic circuits that showcase the robustness of the model and will be useful for us in the future. For proofs, see [SY10; Sap21]. To begin with, we can extract coefficients (with respect to a single variable) of a polynomial computed by a small circuit efficiently.

**Lemma 2.2 (Interpolation)** *Let  $\mathbb{F}$  be a field with  $|\mathbb{F}| > k$ , and  $f \in \mathbb{F}[\mathbf{x}, y]$  be a polynomial with  $\deg_y(f) = k$ . Suppose that  $f(\mathbf{x}, y) = \sum_{j=0}^k f_j(\mathbf{x})y^j$  where  $f_j \in \mathbb{F}[\mathbf{x}]$  for all  $j \in \{0, 1, \dots, k\}$ .*

*If  $f(\mathbf{x}, y)$  can be computed by a circuit of size  $s$  and depth  $\Delta$ , then for all  $j \in \{0, 1, \dots, k\}$ ,  $f_j(\mathbf{x})$  can be computed by a circuit of size  $O(sk)$  and depth  $\Delta + 1$ .*

<sup>2</sup> For the formal version, see Definition 4.14.

As we are concerned only with multivariate *polynomials*, we can always formally define the partial derivative of a polynomial with respect to a variable (and by extension, multiple variables) over any field  $\mathbb{F}$ . Computing the partial derivatives of a circuit with respect to a variable of bounded individual degree is an efficient operation.

**Lemma 2.3 (Partial Derivatives)** *Let  $\mathbb{F}$  be a field with  $|\mathbb{F}| > r$ , and  $f \in \mathbb{F}[\mathbf{x}, y]$  be a polynomial with  $\deg_y(f) = r$ . If  $f(\mathbf{x}, y)$  can be computed by a circuit of size  $s$  and depth  $\Delta$ , then for all  $0 \leq j \leq r$ , the partial derivative  $\partial_y^j f(\mathbf{x}, y)$  can be computed by a circuit of size  $O(sr^3)$  and depth  $\Delta$ .*

The observant reader will have noticed the conspicuous absence of divisions in our definition of algebraic circuits. It turns out that our notions of complexity do not depend on this exclusion, for the most part. Division can be eliminated efficiently over any field [Str73; HY11].

**Lemma 2.4 (Division Elimination)** *Let  $f \in \mathbb{F}[\mathbf{x}]$  be a polynomial computed by a circuit (with division gates) of size  $s$ . Then for every  $d \in \mathbb{N}$ , there exists a circuit without division gates of size  $\text{poly}(s, d)$  that computes  $f \bmod \langle \mathbf{x} \rangle^{d+1}$ .*

In the univariate factoring algorithm, we saw that gcd computation was an essential ingredient. The following lemma shows that it can be computed efficiently in the algebraic circuit model. Refer to [KSS15, Lemma 2.9] for the complete proof.

**Lemma 2.5 (GCD in circuits)** *Let  $f, g \in \mathbb{F}[\mathbf{x}]$  be two  $n$ -variate polynomials of degree at most  $d$  computed by circuits of size  $s$ . Then, the  $\text{gcd}(f, g)$  can be computed by a circuit of size  $\text{poly}(s, d, n)$ .*

### 3 Applications of factoring

Polynomial factoring is not only a great mathematical problem. The techniques developed for its solution have a wide range of applications in various areas of computer science. We briefly describe some of them.

#### 3.1 Equivalence to identity testing

Consider a class of polynomials  $C$  that can be computed by algebraic circuits. The *polynomial identity testing* (PIT) problem for  $C$  asks whether a given polynomial  $f \in C$

is identically zero. Due to the PIT lemma [Ore22; DL78; Zip79; Sch80], a simple randomised algorithm for this problem has been known for decades. However, designing a polynomial-time deterministic algorithm for PIT for the class VP remains a long-standing open problem. Nevertheless, several efficient deterministic algorithms have been developed for restricted circuit classes. The problem is studied in two settings: the *black-box* setting, where only polynomial evaluation is permissible, and the *white-box* setting, where the internals of the circuit are accessible.

The importance of this fundamental problem stems from its applications to equivalence checking, perfect matching, primality testing, and more. The assumption of efficient identity testing algorithms has led to a variety of derandomisation results. Most notably, PIT is tightly connected to proving strong lower bounds for algebraic circuits (more in Section 3.2). For a comprehensive treatment, we refer the reader to classical surveys such as [SY10; Sax09; Sax14], as well as the recent exposition in [DG24] and the references therein.

PIT was first linked to factorisation when Shpilka and Volkovich [SV10] observed that the polynomial  $f(\mathbf{x}) + yz$  has two irreducible factors over disjoint sets of variables if and only if  $f(\mathbf{x})$  is identically zero. This observation implies that a deterministic algorithm for multivariate polynomial factorisation would suffice to derandomise PIT.

The connection between the two problems was further solidified by the work of Kopparty, Saraf, and Shpilka [KSS15], who showed that a derandomised PIT algorithm also leads to a deterministic multivariate factoring algorithm. Together, these results establish the equivalence between derandomising PIT and polynomial factorisation in both black-box and white-box settings.

A natural and simpler question related to polynomial factoring is that of *divisibility testing*: given two polynomials  $f$  and  $g$ , determine whether  $g$  divides  $f$ . One could factor both  $f$  and  $g$  followed by comparing the irreducible factors using PIT to solve this problem using randomisation. However, since non-trivial deterministic PIT algorithms are known in several restricted settings, it is natural to ask whether divisibility testing is easier in such settings.

This question was first studied by Saha, Saptharishi, and Saxena [SSS13], who reduced the problem of testing divisibility of sparse polynomials by a linear polynomial to PIT for expressions of the form  $\sum_i \mathbf{x}^{a_i} \cdot f_i(\mathbf{x})^{d_i}$ , where each  $\mathbf{x}^{a_i}$  is a monomial and  $\deg f_i \leq 1$ . Known PIT algorithms for such structured polynomials were applied to obtain efficient and deterministic divisibility testing algorithms (see [RS05; FS13]). Later, Forbes [For15] extended this line of work to test divisibility of sparse polynomials by constant-degree polynomials. The problem in this case was reduced to PIT for almost similar polynomials of the form  $\sum_i \mathbf{x}^{a_i} \cdot f_i(\mathbf{x})^{d_i}$  with  $\deg f_i \leq 2$ , and gave

a quasi-polynomial time algorithm for this case. For certain classes of polynomials, Forbes also showed a general reduction to PIT [For15, Section 7]. For a more comprehensive discussion on these equivalences, we refer the reader to the survey by Shpilka and Forbes [FS15, Section 5].

### 3.2 Hardness vs Randomness

In the previous section, we hinted at the tight connection between the derandomisation of PIT and strong lower bounds for algebraic circuits. Informally, this connection implies that PIT can be efficiently derandomised if and only if there exist explicit polynomials of high circuit complexity. Interestingly, factoring results on the algebraic circuit classes play a surprising yet pivotal role in establishing this connection.

Heintz and Schnorr [HS80], and subsequently Agrawal [Agr05], proved that efficient black-box identity testing implies strong circuit lower bounds. Specifically, they showed that a deterministic polynomial-time black-box PIT algorithm for circuits of size  $s$  guarantees the existence of a polynomial with coefficients in PSPACE that requires algebraic circuits of size exponential in  $s$ . This yields an exponential lower bound from a polynomial-time PIT algorithm. A central open question is whether the complexity of computing the coefficients can be reduced to #P/poly, as this would imply a separation of VP from VNP [SY10, Open Problem 17]. Kabanets and Impagliazzo [KI04] further strengthened this direction of the connection by proving that derandomising PIT, even in the weaker white-box setting, would imply either  $VP \neq VNP$  or  $NEXP \not\subseteq P/\text{poly}$ .

Kabanets and Impagliazzo strengthened this connection by proving the reverse direction, drawing inspiration from Nisan and Wigderson’s results in the Boolean setting [NW94]. To describe this, we introduce an alternative notion of black-box identity testing. Consider a polynomial map  $\mathcal{G} := (g_1, \dots, g_n) : \mathbb{F}^r \rightarrow \mathbb{F}^n$  with seed length  $r < n$ . The map  $\mathcal{G}$  is a *hitting-set generator* for a class of polynomials  $C$  if, for every non-zero  $f \in C$ , the composed polynomial  $f \circ \mathcal{G} = f(g_1(\mathbf{y}), \dots, g_n(\mathbf{y}))$  is also non-zero. For a detailed discussion of the connections between hitting-set generators and PIT, see [For14, Section 3.2.2]. The high-level idea of Kabanets and Impagliazzo is as follows: to test whether an  $n$ -variate polynomial  $f \in C$  is identically zero, construct a hitting-set generator  $\mathcal{G}$  of seed length  $r = O(1)$ , where each  $g_i$  has degree  $\text{poly}(n)$ . Identity testing of  $f$  then reduces to checking whether the constant-variate (degree- $r$ ) polynomial  $f \circ \mathcal{G}$  is identically zero—a problem efficiently solvable via the PIT lemma.

To illustrate the idea of a hitting-set generator in action, consider a generator with seed length  $(n-1)$  defined as  $\mathcal{G}(\mathbf{y}) = (y_1, \dots, y_{n-1}, g(\mathbf{y}))$ , where  $g$  is a hard polynomial. This generator extends  $r = n - 1$  variables by one additional coordinate using a hard polynomial. Now, suppose  $f(\mathbf{y}, x)$  is a non-zero polynomial in  $C$ . For the sake of contradiction, assume that  $f \circ \mathcal{G} = f(y_1, \dots, y_{n-1}, g(\mathbf{y})) = 0$ . This implies that  $(x - g)$  divides  $f$ . As we will discuss later (Section 4.4), if  $f$  is computable by a small algebraic circuit, then so are all of its factors. This phenomenon is known as *closure under factoring*. Therefore,  $(x - g)$  must also be computable by a small circuit, contradicting our assumption that  $g$  has large circuit complexity. Hence,  $\mathcal{G}$  is as a valid hitting-set generator for  $C$ .

Moving beyond the single-variable stretch, Kabanets and Impagliazzo [KI04] adapted the Nisan-Wigderson design [NW94] to the algebraic setting. Their construction yields a generator  $\mathcal{G}_{\text{KI}}$  that stretches a seed of length  $r = \text{poly}(\log n)$  to  $n$  variables, using a polynomial  $g$  that is assumed to be hard for arithmetic circuits. Consequently, this generator leads to a black-box PIT algorithm. Crucially, the correctness of this derandomisation relies fundamentally on the closure of algebraic circuits under factoring, which will be explored in detail in the upcoming parts of this survey.

**Theorem 3.1 (Combinatorial Hardness Implies PIT)** *Suppose there exists a family  $\{f_n\}_{n \in \mathbb{N}}$  of  $n$ -variate multilinear polynomials such that each  $f_n$  requires algebraic circuits of size  $2^{\Omega(n)}$ . Then, there exists a black-box PIT algorithm running in time  $2^{\text{poly}(\log n)}$  for the class of  $n$ -variate polynomials of degree  $\text{poly}(n)$  computable by circuits of size  $\text{poly}(n)$ .*

The core argument in [KI04] proceeds by contrapositive: if a non-zero polynomial  $f$  vanishes on the image of the generator (i.e.,  $f \circ \mathcal{G}_{\text{KI}} = 0$ ), then the hard polynomial  $g$  used to construct  $\mathcal{G}_{\text{KI}}$  must essentially be computable by a small circuit, contradicting the hardness assumption.

However, over fields of characteristic  $p > 0$ , this factorisation argument encounters a significant obstacle. The original proof could only deduce that some power  $g^{p^k}$  (where  $p^k \leq \deg(f)$ ) has a small circuit, rather than  $g$  itself. This gap remained open until Andrews [And20] showed that  $p$ -th roots of circuits can be computed efficiently, provided the number of variables is small. Andrews' result thus allows for the efficient recovery of  $g$  from  $g^{p^k}$ , extending the hardness-to-randomness tradeoffs to fields of low characteristic.

Theorem 3.1 was subsequently improved with better parameters by the authors of [Guo+22]. They showed that a constant-variate hard polynomial can imply a polynomial-time black-box PIT algorithm. A key contribution in their work is the use of an *algebraic*

generator, in contrast to the *combinatorial design*-based generator used in [KI04]. The combinatorial design inherently suffers from limitations—most notably, it cannot yield better than a quasi-polynomial time PIT algorithm.

**Theorem 3.2 (Non-combinatorial Hardness-to-PIT)** *Let  $k := O(1)$ . Suppose  $\{f_d\}_{d \in \mathbb{N}}$  is a family of  $k$ -variate polynomials of degree  $d$ , and each  $f_d$  requires algebraic circuits of size at least  $d^{0.1}$ . Then, there exists a black-box PIT algorithm running in time  $s^{O(k^2)}$  for the class of  $s$ -variate, degree- $s$  polynomials computable by algebraic circuits of size  $s$ .*

The main limitation of the above results in extending them to restricted models such as formulas and constant-depth circuits was the lack of closure under factoring for these models. Only weaker statements were known earlier [DSY09; CKS19b; And20]. The recent resolution of these closure questions for restricted models by [Bha+25a] proved that the hardness-versus-randomness connection holds for these models as well.

**Theorem 3.3 ([Bha+25a, Theorem 6.1])** *Let  $\Delta > 0$ . Suppose  $\{f_d\}_{d \in \mathbb{N}}$  is a family of  $n$ -variate polynomials of degree  $d$ , and each  $f_d$  requires depth- $\Delta$  algebraic circuits of size at least  $2^{\Omega(n)}$ . Then, there exists a black-box PIT algorithm running in time  $n^{O(\log n)}$  for the class of circuits of depth  $\Delta - O(1)$ .*

We conclude this section by highlighting an important application of the hardness versus randomness paradigm—namely, the phenomenon of *bootstrapping* [AGS19; KST23; Guo+22]. This idea relies on recursively leveraging the connection between hardness and polynomial identity testing (PIT). Remarkably, it was proved in [Guo+22, Theorem 1.6] that a complete derandomisation of PIT can be achieved from even a mildly non-trivial derandomisation of PIT.

**Theorem 3.4 (Bootstrapping)** *Suppose there exists a black-box PIT algorithm that runs in time  $(s^k - 1)$  for the class of  $k$ -variate polynomials of individual degree  $s$ , computable by algebraic circuits of size  $s^{0.1}$ . If  $s$  is sufficiently large, then there exists a black-box PIT algorithm running in time  $s^{O(k^2)}$  for  $s$ -variate polynomials computable by algebraic circuits of size  $s$ .*

To prove this theorem, one first uses the assumed mildly non-trivial black-box PIT to derive an explicit hard polynomial, via the hardness-from-derandomisation results of Heintz-Schnorr [HS80] and Agrawal [Agr05]. One then feeds this hardness into the combinatorial hardness-to-PIT Theorem 3.1 to obtain the stronger PIT algorithm claimed. A technical point is that this argument requires carefully matching parameters

across several steps which we omit here. For a more detailed exposition of this connection between factoring and its implications for the hardness versus randomness frontier, we refer the reader to the survey by Kumar and Saptharishi [KS19].

### 3.3 Other applications

The field of *coding theory* [GRS23] deals with developing *error-correcting codes* — ways of adding (minimal) redundancy to data such that even if parts of it get corrupted during transmission, one can recover the original information. Reed–Solomon codes are particularly ubiquitous and also ‘optimal’, in a sense. They treat the original message as a univariate polynomial and the encoding is the evaluation of this polynomial at various points over some finite field. When the number of errors is too large, we cannot decode a corrupted message uniquely, but we can produce a small list of potential decodings (also known as *list decoding*). The list decoding algorithm of Sudan [Sud97] and the later improvement by Guruswami and Sudan [GS99] crucially use polynomial factorisation. We point the reader to the survey of Forbes and Shpilka [FS15, Section 3.1] for more details.

The problem of learning algebraic circuits is called *reconstruction* [SY10, Chapter 5]. We are given black-box access to a polynomial computed by a circuit  $C$  from some nice family of circuits  $\mathcal{C}$ , and we need to ‘learn’ an arithmetic circuit computing the same polynomial as  $C$ . Efficient polynomial factorisation plays an important role in many reconstruction algorithms [KS09; Sin16; Sin22; SS25]. Polynomial factorisation is also helpful in algebraic property testing [AS03] and the construction of pseudorandom generators for low-degree polynomials [Bog05; DGV24].

In *proof complexity*, a central problem is to prove that certain propositional tautologies need extremely lengthy proofs, even in very powerful proof systems. An important work of Cook and Reckhow [CR79] showed that such proof complexity lower bounds, provided we are able to show them for every propositional proof system, would separate the complexity classes NP and coNP, and in turn, also P from NP. Closure of a class under factoring is another way of saying that multiples of hard polynomials from the class remain hard (more generally, one can study the complexity of ideals [Gro20]). Using such hard multiples, Forbes, Shpilka, Tzameret, and Wigderson [For+21] showed lower bounds against certain algebraic proof systems. For more about proof complexity, see [Kra95; Juk12; Kra19; AF22; HLT24; Beh+25; LST26].

Polynomial factorisation also has applications to various other problems in mathematics, such as derandomising Noether's Normalisation Lemma [Mul17], the primary decomposition of polynomial ideals [GTZ88], and isomorphism of algebras [KS06; Iva+12]. In cryptography, polynomial factorisation is often used as a subroutine, for example, in index calculus algorithms and public-key encryption [MOV97, Chapter 3], and in factoring integers [Bre00]. Polynomial factorisation algorithms, and the tools developed alongside them, have proven invaluable in the cryptanalysis of lattice-based schemes [NV10] as well as post-quantum cryptosystems [DPS20].

## 4 Factoring via Hensel lifting

Hensel lifting was first introduced by Kurt Hensel in a series of papers [Hen97; Hen04; Hen08; Hen18], although an earlier form of it seems to have been known to Gauß [Fre07]. For an element  $p$  in a ring  $\mathcal{R}$ , Hensel lifting gives a method to compute factorisation modulo  $p^\ell$  (for any  $\ell > 0$ ) from the factorisation modulo  $p$ . The term *lifting* refers to the process of improving the *precision* of the factor, by iteratively refining a coarse factorisation modulo  $p$  to modulo higher powers  $p^\ell$ .

**Theorem 4.1 (Hensel lifting)** *Let  $\mathcal{R}$  be a ring and  $\mathcal{I}$  be an ideal in  $\mathcal{R}$ . Consider elements  $f, g, h \in \mathcal{R}$  such that  $f \equiv g \cdot h \pmod{\mathcal{I}}$  and there exist  $u, v \in \mathcal{R}$  such that  $u \cdot g + v \cdot h \equiv 1 \pmod{\mathcal{I}}$ . Then, we have:*

1. **Existence.** *There exist  $g', h' \in \mathcal{R}$  such that  $f \equiv g' \cdot h' \pmod{\mathcal{I}^2}$  and*

$$g' \equiv g \pmod{\mathcal{I}} \quad \text{and} \quad h' \equiv h \pmod{\mathcal{I}}$$

2. **Pseudo-Coprimality.** *For some  $u' \equiv u \pmod{\mathcal{I}}$  and  $v' \equiv v \pmod{\mathcal{I}}$ , we have  $u' \cdot g' + v' \cdot h' \equiv 1 \pmod{\mathcal{I}^2}$ .*
3. **Uniqueness.** *If any other  $\tilde{g}, \tilde{h}$  satisfy the above conditions, then there must be a  $\mu \in \mathcal{I}$  such that  $\tilde{g} \equiv g'(1 + \mu) \pmod{\mathcal{I}^2}$  and  $\tilde{h} \equiv h'(1 - \mu) \pmod{\mathcal{I}^2}$ .*

The polynomials  $g'$  and  $h'$  defined in the Hensel lifting theorem above are called the *lifts* of  $g$  and  $h$ , respectively. The theorem is not merely existential; it is constructive and provides an explicit procedure to compute the lifts. Let  $e := f - gh$ . Then the lifts can be obtained as follows:

$$g' := g + e \cdot v \quad \text{and} \quad h' := h + e \cdot u. \tag{2}$$

To apply Hensel lifting for polynomial factoring, we typically set the ring  $\mathcal{R} = \mathbb{F}[x, y]$  and choose the ideal  $\mathcal{I} = \langle y^k \rangle$  for some  $k \geq 1$ . In the case of a multivariate polynomial, we transform it to a bivariate polynomial. We will discuss this in more detail in the upcoming sections. When working over a ring of polynomials, a monic version of the theorem further guarantees the uniqueness of the monic lifts.

**Theorem 4.2 (Monic Hensel lifting)** *Let  $\mathbb{F}[x, y]$  be a ring of polynomials. Consider monic polynomials  $f, g, h \in \mathbb{F}[x, y]$  such that  $f \equiv g \cdot h \pmod{y}$  and there exist  $u, v \in \mathbb{F}[x, y]$  such that  $u \cdot g + v \cdot h \equiv 1 \pmod{y}$ . Then, we have:*

1. **Existence.** *There exist monic polynomials  $g', h' \in \mathbb{F}[x, y]$  such that  $f \equiv g' \cdot h' \pmod{y^2}$  and*

$$g' \equiv g \pmod{y} \quad \text{and} \quad h' \equiv h \pmod{y}$$

2. **Pseudo-Coprimality.** *For some  $u' \equiv u \pmod{y}$  and  $v' \equiv v \pmod{y}$ , we have  $u' \cdot g' + v' \cdot h' \equiv 1 \pmod{y^2}$ .*
3. **Uniqueness.** *If any other monic  $\tilde{g}, \tilde{h}$  satisfy the above conditions, then  $\tilde{g} \equiv g' \pmod{y^2}$  and  $\tilde{h} \equiv h' \pmod{y^2}$ .*

Once again let  $e := f - gh$ , and define  $\hat{g}, \hat{h}$  as in Equation (2). Compute the following expressions using the division with remainder algorithm:

$$\frac{\hat{g} - g}{y} = q \cdot g + r.$$

Then the unique monic lifts can be computed as follows:

$$g' := g + y \cdot r \quad \text{and} \quad h' := \hat{h} \cdot (1 + q \cdot y) \tag{3}$$

An expository proof of Hensel lifting can be found in [KSS15, Lemma 3.4].

#### 4.1 Polynomials in the dense representation

To demonstrate the use of Hensel lifting in factoring, consider the problem of factoring a bivariate polynomial  $f \in \mathbb{F}_q[x, y]$  of degree at most  $d$ , where  $q = p^a$ . The input is given in the dense representation, as in Section 1.1. In fact, the univariate factoring algorithms will be used as subroutines in the bivariate setting. For convenience, we view  $f$  as an element of  $(\mathbb{F}_q[x])[y]$ , effectively shifting  $x$  into the base ring and treating it as a constant.



### Bivariate Factoring.

The high-level idea of bivariate factoring is to first obtain factorisation of  $f$  modulo  $y$ , and then lift them using Hensel lifting. But before we can do that, we need to ensure that  $f$  satisfies certain properties, as described below.

1.  $f(x, 0)$  is **square-free**: For  $a \in \mathbb{F}_q$ , the univariate polynomial  $f(x, a)$  is square-free if and only if

$$\gcd\left(f(x, a), \partial_x f(x, a)\right) \neq 1.$$

From [Lemma 4.4](#), this is equivalent to  $\text{Res}_x(f, \partial_x f)|_{y=a} \neq 0$ . Since the degree of the resultant in  $y$  is at most  $2d^2$ , it suffices to test  $O(2d^2)$  values of  $a \in \mathbb{F}_q$  (or a suitable extension) to find one for which  $f(x, a)$  is square-free. A simple linear transformation will ensure that the resultant is non-zero at  $a = 0$ . Further, this would also imply that  $f$  is square-free.

2.  $f$  is **monic in  $x$** : Let  $f_d(y)$  be the leading coefficient of  $f$  of degree at most  $d$ . Then,

$$\tilde{f} = f_d^{d-1} \cdot f\left(x/f_d, y\right)$$

is a monic polynomial in  $x$ .

For simplicity, let  $f$  denote the monic and square-free polynomial obtained from the transformations above. Ensuring that  $f(x, 0)$  is square-free is crucial, as it allows us to apply univariate factoring algorithms ([Section 1.1](#)) to find  $g_0, h_0 \in \mathbb{F}[x]$  such that

$$f \equiv g_0 \cdot h_0 \pmod{y},$$

where  $g_0$  is a monic irreducible factor in  $x$ . We then apply Hensel lifting ([Theorem 4.2](#)) for  $t$  iterations to obtain  $g_t$  and  $h_t$  satisfying

$$f \equiv g_t \cdot h_t \pmod{y^{2^t}}.$$

To understand the nature of  $g_0$ , consider the irreducible factorisation  $f = f_1 \cdot \dots \cdot f_r$ . When restricted to  $y = 0$ , a bivariate factor  $f_i$  may become reducible. Consequently,  $g_0$  corresponds to an irreducible factor of the restriction of some  $f_i(x, 0)$ . Further, the lifting process continues until  $2^t > 2d^2$ , a bound chosen to exploit the properties of resultants, as we address in the proof of the following claim.

**Proposition 4.5** *If the input polynomial  $f$  is reducible, then there exists a non-trivial factor  $f_1$  of  $f$  such that  $f_1 \equiv g_t \cdot \ell \pmod{y^{2^t}}$  for some polynomial  $\ell$ . Furthermore,  $\deg_x(f_1) \leq \deg_x(f)$  and  $\deg_y(f_1) \leq \deg_y(f)$ .*

*Proof.* First let us prove that such a factor exists. Since  $g_0$  is an irreducible factor of  $f \pmod{y}$ , we know that  $g_0$  must divide some irreducible factor of  $f \pmod{y}$ , say  $f_1$ . Let  $f = f_1 \cdot h$ . Then, for some  $\ell_0$ ,

$$f_1 \equiv g_0 \cdot \ell_0 \pmod{y}.$$

By [Theorem 4.2](#) we can lift  $g_0$  as a factor of  $f_1$  and get  $f_1 \equiv g'_t \cdot \ell_t \pmod{y^{2^t}}$ . Multiplying both sides by  $h$  gives

$$f \equiv g'_t \cdot h'_t \pmod{y^{2^t}}.$$

By the uniqueness property of [Theorem 4.2](#), it follows that  $g'_t \equiv g_t \pmod{y^{2^t}}$ , which implies the existence of  $f_1 \equiv g_t \cdot \ell_t \pmod{y^{2^t}}$ .

To see that  $f_1$  is non-trivial, let us assume for the sake of contradiction that  $\gcd_x(f, f_1) = 1$ . Then there exist  $u, v$  such that

$$u \cdot f + v \cdot f_1 = \text{Res}_x(f, f_1).$$

Substituting the lifted expressions, we get

$$g_t \cdot (u h_t + v \ell_t) \equiv \text{Res}_x(f, f_1) \pmod{y^{2^t}}.$$

Since  $g_t$  is monic, and  $\text{Res}_x(f, f_1) \in \mathbb{F}_q[y]$ , it follows that

$$u h_t + v \ell_t \equiv 0 \pmod{y^{2^t}}.$$

Since  $2^t$  is greater than the degree of the resultant  $2d^2$ , it follows that  $\text{Res}_x(f, f_1) = 0$ . This yields a contradiction, and thus  $\gcd_x(f, f_1) \neq 1$ .  $\square$

The above proposition suggests that a candidate polynomial  $\tilde{g}$  can be obtained by solving the following linear system:

$$\tilde{g} \equiv g_t \cdot \ell \pmod{y^{2^t}}, \tag{4}$$

where the degree bounds for  $\tilde{g}$  are taken to be the same as those guaranteed by the previous claim. Finally, we compute  $\gcd(f, \tilde{g})$  to obtain the required non-trivial factor  $g$  of  $f$ .

Finally, we recall that the algorithm factors a polynomial that results from a sequence of preprocessing steps. Hence, it is necessary to undo these steps—particularly the monic transformation—in order to recover a factor of the original input polynomial. By applying the Cantor–Zassenhaus algorithm for univariate factorisation and analysing the

cost of lifting factors via Hensel lifting, it can be observed that the overall complexity of bivariate factoring is  $\text{poly}(d, \log q)$  operations over  $\mathbb{F}_q$ .

Naturally, the bivariate factoring algorithm extends to the multivariate setting. However, as the number of variables increases, the runtime of the algorithm degrades rapidly. Later, we will explore how to address this challenge while still reusing insights from the factoring algorithms discussed thus far.

## 4.2 Polynomials over integers

The finite field factoring algorithm can be extended to factoring polynomials over integers and thereby over rationals as well. Although the details are outside the scope of this survey, we will briefly discuss the high-level idea of the algorithm. Consider a degree  $d$  input polynomial  $f \in \mathbb{Z}[x]$  with coefficients of bit-length at most  $\ell$ . Therefore, the coefficients of  $f$  are between  $-2^\ell$  and  $2^\ell$ , and the goal is to obtain a non-trivial factor of  $f$  in  $\text{poly}(d, \ell)$  time.

The algorithm for factoring polynomials over the integers closely follows the same template as the bivariate factoring algorithm. It begins by choosing a sufficiently large prime  $p$  such that  $f \bmod p$  is square-free. The prime  $p$  plays a role analogous to the variable  $y$  in the bivariate setting. The polynomial  $f$  is then factored modulo  $p$  using a univariate factoring algorithm (see [Section 1.1](#)). The resulting factors are lifted to a factor modulo  $p^{2^t}$  via Hensel lifting. As in the bivariate case, it can be shown that  $\log(n^3 \ell)$  iterations of the lifting step suffice.

The most challenging part of the algorithm lies in solving the linear system similar to [Equation \(4\)](#), under the additional constraint that the coefficients of the factor  $g$  are small. Although this is not immediately obvious, the problem reduces to finding a shortest vector in a lattice—a task that is known to be NP-hard in general.

Nevertheless, the celebrated algorithm of Lenstra, Lenstra, and Lovász [[LLL82](#)] provides an efficient *weak-approximation* algorithm for this problem, which turns out to be sufficient for the purposes of polynomial factoring. For a detailed exposition, see [[Sap17](#), Chapter 13] and [[GG13](#), Section 16.5].

### 4.3 Polynomials over $p$ -adic numbers

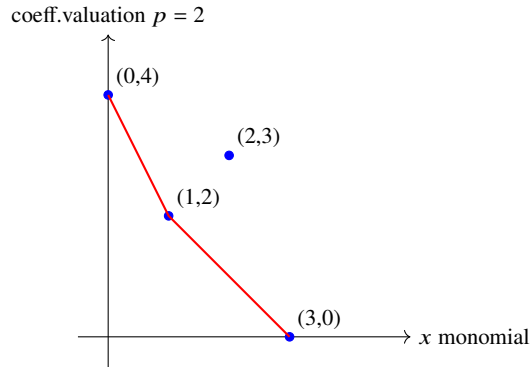
There is no *topology*, nor a geometric interpretation, inherent in a finite field. Classically, this motivated mathematicians to ‘extend’ a finite field  $\mathbb{F}_q$  ( $q$  is a power of a prime  $p$ ) to a characteristic *zero* field. The latter is called an *unramified  $p$ -adic field* construction.

**$p$ -adics.** For simplicity, consider the object  $\mathbb{Z}_p$ , the ring of  *$p$ -adic integers*. Its elements are infinite series of the type  $(a_0 + a_1 \cdot p + a_2 \cdot p^2 + \dots)$  with the *digits*  $a_i$ ’s in  $[0 \dots p - 1]$ . The notion of *convergence* here requires an *ultra-metric*, defined via prime-powers,  $|p^i|_u := p^{-i}$ . (Exercise: Check that it defines a *metric*, i.e. a map from  $\mathbb{Z}_p$  to  $\mathbb{Q}$ .) Also, we sometimes call  $i$  the  *$p$ -adic valuation of  $p^i$* . This forces an *unreal* comparison of real numbers:  $1 > p > p^2 > \dots > p^\infty = 0$  ! The convergence allows certain integers to become invertible or *units*, e.g.  $1/(1 - p) = 1 + p + p^2 + \dots \in \mathbb{Z}_p$ . It is easy to show: (1)  $\mathbb{Z}_p$  is an integral domain, so it has a field of fractions called  *$p$ -adic rationals*  $\mathbb{Q}_p$ , which has non-integer elements like  $1/p$  and  $1/(p^2 + 2p^3)$ , and (2) The characteristic of  $\mathbb{Z}_p$  and  $\mathbb{Q}_p$  is 0.

Given a polynomial  $f(x) \in \mathbb{Q}_p[x]$  as a binary string of size  $n$ , can we factor it, over  $\mathbb{Q}_p$ , in time  $\text{poly}(n)$ ? A randomised poly-time algorithm was provided by [Chi87; CG00]. The two papers are written in very different styles. We will sketch the basic ideas common to them, by working through an example.

Consider the 2-adic polynomial  $f(x) = (x - 2)^2(x - 4) + 32 \in \mathbb{Q}_2[x]$ . Note that  $f \equiv x^3 \pmod{2}$  has no coprime factors over  $\mathbb{F}_2$ , so the algorithm of Section 1.1.2 cannot be applied to get factors, or check irreducibility, ‘lifted’ to  $\mathbb{Z}/4\mathbb{Z}$ . This forces us to work modulo higher 2-powers and to perform some new operations. Seeing broadly, there are two major algebraic themes in the algorithm, which we oversimplify for the sake of presentation as follows.

**Theme 1 (Newton-Hensel).** Consider the *Newton diagram* of the example polynomial above,  $f = x^3 - 8x^2 + 20x + 16$ , by plotting the monomial-exponents in the X-axis (namely,  $\{0, 1, 2, 3\}$ ) and the 2-adic valuation of the integral-coefficients in the Y-axis (respectively,  $\{4, 2, 3, 0\}$ ). The lower-boundary of this diagram has two edges of slopes 1 and 2 respectively, suggesting that the  $\overline{\mathbb{Q}}_2$ -roots of  $f(x)$  are (exactly) divisible by  $2^1$  and  $2^2$  respectively. This gives us the ‘transformation’  $f(2x)$  to study, and to find the first factor by a version of *Hensel lifting*. Thus,  $f(2x)/8 = (x - 2)(x - 1)^2 + 4 = (x - 2 + 2^2 + 2^5 + \dots) \cdot (x^2 - (2 + 2^2 + 2^5 + \dots)x + 1 - 2^4)$ . The key property we use here is the coprimality of the two factors  $(x - 2), (x - 1)^2 \pmod{2}$ ; so, the algorithm of Section 1.1.2 can be applied over the finite field  $\mathbb{F}_2$ .



**Fig. 3** The Newton Diagram for  $f(x)$ .

An inverse transformation ( $x \mapsto x/2$ ) gives us the two factors of  $f$  as,  $g_1 := (x - 2^2 + 2^3 + 2^6 + \dots)$  and  $g_2 := (x^2 - (2^2 + 2^3 + 2^6 + \dots)x + (2^2 - 2^6 + \dots))$ , respectively. Clearly,  $g_1$  is an *irreducible* 2-adic factor of  $f$ . What about  $g_2$ ?

**Theme 2 ( $p$ -adic ramification).** Again from the Newton diagram of  $g_2$  we learn that it has two distinct  $\overline{\mathbb{Q}}_2$ -roots, both (exactly) divisible by  $2^1$ . So, we study the new transformation  $g_2(2x)/4 = (x^2 - (2 + 2^2 + 2^5 + \dots)x + 1 - 2^4) =: T(x)$ . Clearly,  $T \equiv (x - 1)^2 \pmod{4}$  with no coprime factors mod 2. In this case both the tricks of Newton and Hensel fail.

We have  $T \equiv x^2 - 6x + 1 \equiv (x - 3)^2 - 8 \pmod{16}$ . From here we learn two properties: (1)  $T$  is irreducible (yielding a *certificate* of irreducibility for  $g_2$ ), as  $\sqrt{8}$  does not exist in  $\mathbb{Q}_2$ ; and (2)  $T$  suggests a more intricate transformation to progress to a deeper root of  $f$ , namely,  $(x - 3) \mapsto (x - 3)\sqrt{2}$  over the (*ramified*) field  $\mathbb{Q}_2(\sqrt{2})$ .

In general, if a repeated number of these steps achieve a ramification degree equal to that of the degree of  $f$ , then we have a certificate of irreducibility of  $f$  over  $\mathbb{Q}_p$ .

The above two algebraic themes give a randomised poly-time algorithm to factor  $f(x) \in \mathbb{Q}_p[x]$ . The time-complexity is based on analysing the (Galois) symmetries of both the ramified and unramified field extensions of  $\mathbb{Q}_p$  that the algorithm constructs.

### ? Open Problem (4.3.1)

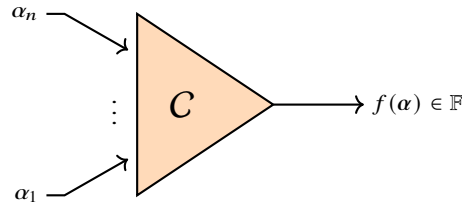
Given an integral polynomial  $f(x) \in \mathbb{Z}[x]$  of degree  $d$  and a prime-power  $p^k$ , can we factor  $f \pmod{p^k}$ , in randomised poly( $dk \log p$ ) time?

In this case, there is no unique factorisation property, and we cannot use division by  $p$ , as it is now a zero-divisor. See [DMS21; CDS24] for a detailed survey.

#### 4.4 Small algebraic circuits

The factoring algorithms discussed in earlier sections assume that the input polynomial is given in the dense representation, that is, as an explicit list of its coefficients. However, algebraic circuits introduced in Section 2 provide a much more succinct way to represent polynomials. Consequently, the number of non-zero coefficients of a polynomial computed by a small circuit can be exponential in the circuit size, making the conversion to dense representation inefficient. Therefore, we require algorithms that can operate directly on the circuit model. We begin by formally describing the model of computation for the algorithm.

Let  $f \in \mathbb{F}[\mathbf{x}]$  be an  $n$ -variate polynomial computed by an algebraic circuit  $C$  of size  $s$ . In the black-box model, access to  $C$  is restricted to evaluation queries only (see Figure 4). Given such a circuit  $C$  and the degree bound  $d = \deg(f)$ , the goal is to compute the irreducible factors of  $f$ .



**Fig. 4** Black-box access to polynomial via circuit  $C$

From now on, we will work over fields of characteristic zero unless otherwise specified. The results also hold if the characteristic is sufficiently large compared to the degree of the polynomial under consideration.

A special case of multivariate factorisation arises when the polynomial is a perfect power, i.e.,  $f = g^e$  where  $g$  is an irreducible polynomial and  $e \geq 1$ . For fields of large characteristic, the generalised binomial theorem gives a simple way to compute  $g$  from the circuit computing  $f$ .

**Theorem 4.6 (Perfect Power Factorisation)** *Let  $\mathbb{F}$  be a field of characteristic zero and  $f \in \mathbb{F}[\mathbf{x}]$  be an  $n$ -variate polynomial of degree  $d$  computable by a circuit of size  $s$ . If*

$f = g^e$  for some irreducible polynomial  $g$  and  $e \geq 1$ , then  $g$  can be computed by a circuit of size  $\text{poly}(s, n, d)$ .

*Proof Sketch.* We begin by performing a linear transformation on  $f$  to ensure that  $f(0, \dots, 0) = 1$ . Then using the generalised binomial theorem, we have

$$\begin{aligned} g &= f^{1/e} = \left(1 + (f - 1)\right)^{1/e} \\ &= \sum_{i=0}^d \binom{1/e}{i} \cdot (f - 1)^i \bmod \langle \mathbf{x} \rangle^{d+1}. \end{aligned} \quad (5)$$

The last equality holds because  $(f - 1)^i$  contributes terms of degree strictly larger than  $d$  for all  $i > d$ . Algebraic circuits can be efficiently added and multiplied, and further, the division required to compute the binomial coefficients in Equation (5) can be eliminated using Lemma 2.4. Overall, given only black-box access to  $C$  computing  $f$ , we can obtain the circuit for  $g$  of size  $\text{poly}(s, n, d)$ .  $\square$

It is worth highlighting that the multiplicity  $e$  can be exponential in the size  $s$  of the circuit computing  $f$  (e.g., constructed via repeated squaring). Consequently, Theorem 4.6 implies factor closure not just for special polynomials of VP, but also for high-degree polynomials of the form  $f = g^e$  computable by small circuits. We will revisit this special case in Theorem 5.13, where we discuss an alternative proof. In the upcoming section, we will study the general factorisation problem, where the known results are not as strong as in this special case.

### Black-box multivariate factoring.

Remarkably, Kaltofen and Trager [KT90] showed that in the black-box setting, efficient circuit factorisation is achievable. To describe it, we need to state Hilbert's Irreducibility Theorem, which guarantees that irreducible factors of  $f$  maintain their irreducibility profile when restricted to the random subspace.

**Theorem 4.7 (Effective Hilbert Irreducibility Theorem)** *Let  $S \subseteq \mathbb{F}$  be a large enough finite subset and  $g \in \mathbb{F}[\mathbf{x}, y]$  be a monic polynomial of degree  $d$  such that  $\partial_y g$  is non-zero. If  $|S|$  is at least  $d^6$  and  $g$  is irreducible, then for most of the  $\bar{\alpha}, \bar{\beta} \in S^n$ , the polynomial  $g(\alpha_1 x + \beta_1, \dots, \alpha_n x + \beta_n, y)$  is irreducible.*

The effective version of Hilbert’s Irreducibility Theorem was first proved by Kaltofen [Kal95, Section 3]. For a clear and accessible exposition, see Sudan’s lecture notes [Sud98, Lecture 9].

Consider the polynomial  $f$  restricted to a suitable subspace so that the theorem above holds:

$$f_{\alpha,\beta} := f(y, \alpha_1 x + \beta_1, \dots, \alpha_n x + \beta_n).$$

Let  $\ell$  be the number of irreducible factors of  $f_{\alpha,\beta}$  obtained using the bivariate factoring algorithm (see Section 4.1). Assuming  $f$  is monic in  $y$  and  $\partial_y f$  is non-zero, one can prove using Theorem 4.7 that  $f$  has  $\ell$  irreducible factors with high probability. Note that, since the bivariate factoring algorithm requires the input polynomial to be in dense representation, the coefficients of  $f_{\alpha,\beta}$  are computed via interpolation (ref Lemma 2.2).

Given a point  $(a, b_1, \dots, b_n) \in \mathbb{F}^{n+1}$  and an index  $i$ , we want to compute the  $i$ -th factor  $f_i(a, b_1, \dots, b_n)$ . For this, define a trivariate polynomial which captures both the projection of  $f$  on  $\bar{\alpha}, \beta$  and the evaluation point  $(a, b_1, \dots, b_n)$  as follows:

$$\hat{f}(y, z_1, z_2) := f(y, \alpha_1 z_1 + \beta_1 + (b_1 - \beta_1)z_2, \dots, \alpha_n z_1 + \beta_n + (b_n - \beta_n)z_2),$$

where each  $x_i \mapsto \alpha_i z_1 + \beta_i + (b_i - \beta_i)z_2$ . Note that  $\hat{f}(a, 0, 1) = f(a, b_1, \dots, b_n)$  and  $\hat{f}(y, x, 0) = f_{\alpha,\beta}(y, x)$ . Using interpolation described in Lemma 2.2, we can obtain the coefficients of  $\hat{f}$ . Kaltofen [Kal85] proved that factoring a constant-variate polynomial, in particular the trivariate polynomial  $\hat{f}$ , is efficiently doable using univariate and bivariate factorisations. Let  $\{\hat{f}_i(y, x_1, x_2) : i \in [\ell']\}$  be the set of irreducible factors of  $\hat{f}$ . Find an index  $j$  such that  $\hat{f}_j(y, x, 0) = \tilde{f}_i(y, x)$  and output  $\hat{f}_j(a, 0, 1)$ .

### Circuit factoring

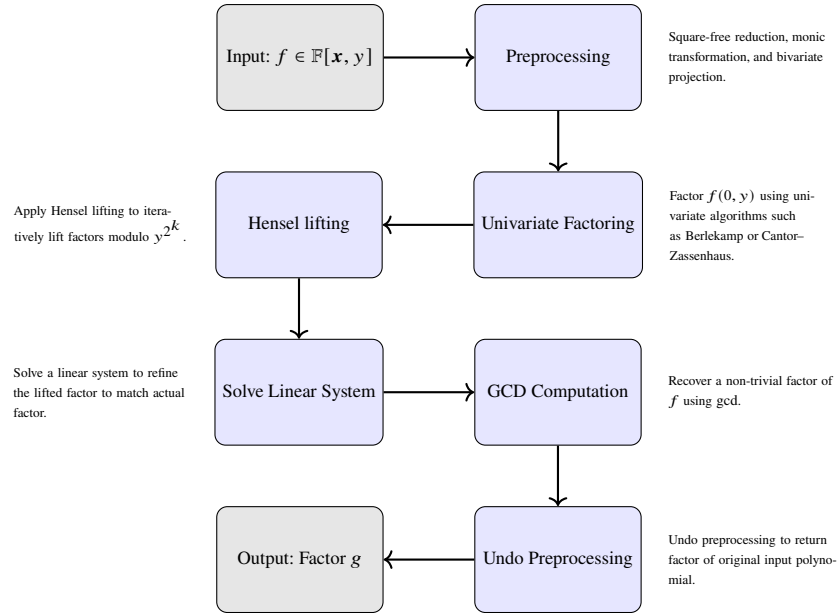
Black-box factoring is a strong notion of factorisation, wherein the algorithm is given oracle access to a polynomial and is required to construct oracle access to its factors—without relying on or even knowing the underlying representation or model of computation. Remarkably, this idea can be adapted to show that if a polynomial is computed by a small algebraic circuit, then all its irreducible factors can be computed by small circuits [Kal89]. The complexity class VP contains all polynomial families that can be computed by ‘small’ circuits.

**Definition 4.8** (VP) A polynomial family  $f = (f_n)$  is in the class VP over the field  $\mathbb{F}$  if both the number of variables and degree of  $f_n$  are bounded by  $\text{poly}(n)$  and moreover,

the size of the smallest circuit over  $\mathbb{F}$  computing  $f_n$ , denoted  $\text{size}_{\mathbb{F}}(f_n)$  is bounded by  $\text{poly}(n)$ .

Note that polynomials of degree  $\exp(n)$  can be computed by circuits of size  $\text{poly}(n)$  by repeated squaring. Such families are not in VP due to the degree restriction in its definition. There are good reasons for imposing this restriction [Val79a; Gro13]. As mentioned earlier, we will also discuss small circuits of high degree in Section 5.2. The main result of this section is the closure of VP under taking factors.

Over characteristic zero, we can without loss of generality assume that  $f = g^e \cdot h$ , where  $g$  and  $h$  are coprime and  $e \geq 1$ . The special case of  $h = 1$  can be handled as before using Equation (5).



**Fig. 5** Overview of Multivariate Factoring using Hensel lifting.

**Theorem 4.9 (VP factor closure)** *Let  $\mathbb{F}$  be a field of characteristic zero, and  $f \in \mathbb{F}[x, y]$  be an  $(n+1)$ -variate, degree  $d$  polynomial computable by a circuit of size  $s$ . If there are coprime factors  $g$  and  $h$  such that  $f = g^e \cdot h$ , where  $e \geq 1$ , then  $g$  is also computable by a circuit of size  $\text{poly}(s, n, e)$ .*

*Proof Sketch.* As in the univariate case, we first perform a square-free reduction to ensure  $f$  has no repeated factors, by dividing  $f$  by  $\text{gcd}(f, \partial_y f)$ . The efficiency of this

step follows from the structural results on algebraic circuits discussed in [Section 2.1](#). It is important to note that the multiplicity parameter  $e$  appearing in the final size bound for the factor originates from this square-free reduction step. Next, as before we apply a linear transformation to make  $f$  monic in  $x$ . To reduce the problem to bivariate factoring, we invoke the effective Hilbert Irreducibility Theorem ([Theorem 4.7](#)), effectively projecting the variables  $x, y$  onto a two-dimensional plane spanned by  $x$  and  $y$ . This projection preserves the irreducibility profile of the factors. After this sequence of transformations, we work with the polynomial  $f = g \cdot h \in \mathbb{F}[x, y]$ , where  $g$  and  $h$  remain coprime and  $f$  is monic in  $y$ . Furthermore, we can ensure that the restriction to  $y = 0$  preserves coprimality, meaning  $g(x, 0)$  and  $h(x, 0)$  are coprime. These conditions satisfy the prerequisites for Hensel lifting. Crucially, all these preprocessing steps are reversible and can be efficiently implemented using algebraic circuits.

We iteratively apply Hensel lifting to compute polynomials  $g_k$  and  $h_k$  such that  $f \equiv g_k \cdot h_k \pmod{y^{2^k}}$ . The lifting proceeds until  $2^k > 2d^2$  (refer to [Proposition 4.5](#)). Each iteration consists of basic algebraic operations outlined in [Equation \(3\)](#), all of which can be performed efficiently by leveraging the structural results of algebraic circuits. Since Hensel lifting yields only an *approximate* factorisation modulo  $y^{2^k}$ , the actual factor  $g$  of  $f$  is recovered by solving the linear system  $g' \equiv g_k \cdot \ell \pmod{y^{2^k}}$ .

By computing the gcd of  $f$  with the solution  $g'$  of the linear system, we recover a non-trivial factor  $g$  of  $f$ . Finally, we reverse all preprocessing steps to obtain an algebraic circuit computing  $g$ .  $\square$

In [Section 7](#), we discuss a non-iterative proof of factor closure for VP based on a classical identity due to Lagrange. Extending this closure result to fields of small characteristic remains an open problem.

#### **? Open Problem (4.4.1)**

Is the class VP closed under taking factors over fields of positive characteristic?

---

As mentioned earlier, Kaltofen [[Kal87](#), Theorem 2] proved a special case of factor closure for polynomials in VP of the form  $g^e$  (i.e.,  $h = 1$  in [Theorem 4.9](#)) over fields of characteristic zero. Andrews [[And20](#)] removed the dependence on characteristic in this result for the log-variate regime, thereby making progress toward resolving [Problem 4.4.1](#). Later, we will discuss progress on this question in a more general setting, where factors of polynomials in VP over finite fields are shown to be *explicit*.

There are other natural classes for which a factor closure result, similar to [Theorem 4.9](#), does hold. Notably, these include classes that contain VP, and also classes that are contained in VP. We discuss these next.

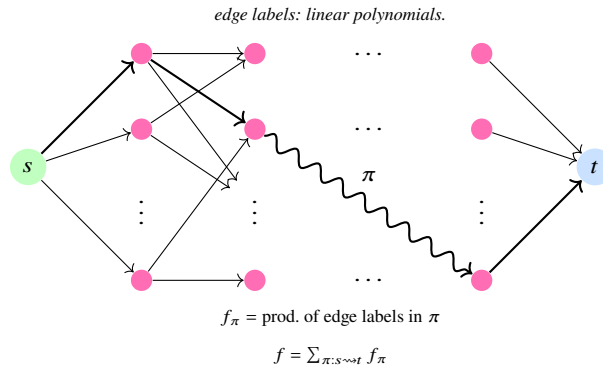
## 4.5 Algebraic branching programs

Another polynomial computation model that is often studied is algebraic branching programs.

**Definition 4.10 (Algebraic Branching Program)** An *Algebraic Branching Program* (ABP) is a directed acyclic graph where the edges are labelled by linear polynomials over a field. The graph has a unique source node  $s$  and a sink node  $t$  (refer to [Figure 6](#)). For every path  $\pi$  from  $s$  to  $t$ , let  $f_\pi$  denote the product of the labels on the edges of  $\pi$ . The polynomial computed by the ABP is the sum of polynomials computed along all the different  $s - t$  paths:

$$f = \sum_{\pi: s \rightsquigarrow t} f_\pi.$$

The *size* of the ABP refers to the total number of vertices in the graph, while the *length* of the ABP is the length of the longest path from  $s$  to  $t$ .



**Fig. 6** Algebraic Branching Program (ABP)

Although not immediately evident, ABPs are equivalent to restricted algebraic circuits known as *skew circuits*. In a skew circuit, each multiplication gate is allowed to have at most one child that is not an input gate. For a proof of equivalence between

ABPs and skew circuits, see [Mah14]. Similar to general circuits, ABPs are closed under addition and multiplication with only an additive increase in size. Strassen’s classical technique for division elimination, originally developed for algebraic circuits, can be adapted to ABPs as well (see Lemma 2.4). We state a few non-trivial structural results about ABPs that will be instrumental in the upcoming factor closure result.

**Lemma 4.11 (ABP Closure Properties)** *Let  $f \in \mathbb{F}[\mathbf{x}, y]$  be an  $(n + 1)$ -variate polynomial of degree  $d$  computed by an ABP of size  $s$ , such that  $f = \sum_{i=1}^d f_i(\mathbf{x})y^i$ . Then:*

1. **Coefficient Extraction.** *For any  $i \in [d]$ ,  $f_i$  can be computed by an ABP of size  $\text{poly}(s, n, d)$ .*
2. **GCD.** *For any polynomial  $g$  computable by an ABP of size  $s$ ,  $\text{gcd}(f, g)$  can be computed by an ABP of size  $\text{poly}(s, n, d)$ .*
3. **Composition.** *Let  $g_1, \dots, g_n$  be polynomials computable by ABPs of size  $s$ . Then the composition  $f(g_1, \dots, g_n)$  can be computed by an ABP of size  $\text{poly}(s, n)$ .*

Refer to [ST21, Section 2] for detailed discussion on these closure properties. Similar to the class of small degree polynomials computable by small circuits Definition 4.8, we can define the class of polynomials computable by small ABPs.

**Definition 4.12 (VBP)** A polynomial family  $f = (f_n)$  is said to be in the class VBP if the number of variables in  $f_n$  is bounded by  $\text{poly}(n)$  and furthermore,  $f_n$  can be computed by an algebraic branching program of size at most  $\text{poly}(n)$ .

A polynomial computed by a polynomial-size algebraic branching program necessarily has a polynomially bounded degree. Therefore, it is not hard to show that  $\text{VBP} \subseteq \text{VP}$ . In the previous section, we discussed the closure of the class VP under factorisation. A natural question is whether a similar closure property holds for VBP. Kaltofen and Koiran [KK08] made progress in this direction by showing that if  $f, g \in \text{VBP}$  and  $f = g \cdot h$ , then  $h$  also belongs to VBP. Their approach relies on interpreting algebraic branching programs (ABPs) as a restricted class of circuits known as skew circuits discussed earlier. Using this division closure, Jansen [Jan11] leveraged the connection between ABPs and the determinant to adapt the power iteration algorithm for computing eigenvalues, to prove factor closure for polynomials of the form  $f = f_1 \cdot (y - f_2) \cdots (y - f_n)$ , where the  $f_i$  are pairwise distinct. More recently, Sinhababu and Thierauf [ST21] revisited the Hensel lifting method to completely solve the open problem by proving factor closure of VBP.

It is important to observe that the multivariate factoring algorithm from Theorem 4.9 does not directly extend to Algebraic Branching Programs (ABPs). The main challenge

lies in the monic variant of Hensel lifting, which requires division with remainder at each step—a costly operation for ABPs when repeated frequently (see Equation (3)). Although composition is technically feasible in ABPs (Lemma 4.11), the multiplicative blow-up renders repeated use impractical. This limitation also rules out other factoring strategies that rely on efficient composition, which we shall explore in later sections.

Nevertheless, Sinhababu and Thierauf [ST21, Theorem 4.1] circumvented the obstacles by employing the classical version of Hensel lifting (Theorem 4.1), which avoids division altogether.

**Theorem 4.13 (VBP Closure)** *Let  $f \in \mathbb{F}[\mathbf{x}]$  be an  $n$ -variate polynomial of degree  $d$  computed by an ABP of size  $s$ . Then, every factor  $g$  of  $f$  can be computed by an ABP of size  $\text{poly}(s, n, d)$ .*

*Proof Sketch.* The special case  $f = g^e$  is handled via the generalised binomial expansion (analogous to Equation (5)), while leveraging the structural properties of ABPs discussed in Lemma 4.11. For the general case  $f = g^e \cdot h$  (where  $e \geq 1$ ), we follow the same sequence of preprocessing steps as before: computing a derivative for square-free reduction, applying a random shift to ensure that  $f$  is monic and that  $g$  and  $h$  remain coprime, and finally reducing the polynomial to a bivariate form. These steps prepare the polynomial for the application of Hensel lifting. As before, we begin the lifting process by applying a univariate factoring algorithm to  $f(x, 0)$  (see Section 1.1). However, instead of using the monic variant of Hensel lifting—as in the previous setting—which involves division at each step, we employ the classical version of Hensel lifting (see Theorem 4.1). The process is repeated as before for  $t$  iterations to obtain polynomials  $g_t$  and  $h_t$  such that

$$f \equiv g_t \cdot h_t \pmod{y^{2^t}},$$

where  $t = O(\log d)$  suffices. Finally all that remains is to solve the linear system arising as before from

$$\tilde{g} \equiv g_t \cdot h'_t \pmod{y^{2^t}}.$$

Solving this linear system can be performed efficiently by an ABP, as the determinant of a matrix with polynomial entries is known to be in VBP (see [MV97]; for an exposition, see [Sap17, Section 3.3.3]). Since a guaranteed solution requires the linear system to be homogeneous, the polynomial  $\tilde{g}$  is not monic. Sinhababu and Thierauf observed that the monic irreducible factor  $g$  can be recovered by dividing  $\tilde{g}$  by its leading coefficient (see [ST21, Lemma 4.19]). It is then easy to observe that all the steps, including the preprocessing steps, can be performed efficiently on ABPs using the structural results discussed earlier.  $\square$

As for VP, it is an open question to extend the above result to fields of small characteristic.

**? Open Problem (4.5.1)**

Is the class VBP closed under taking factors over fields of positive characteristic?

---

## 4.6 Explicit polynomials

A compelling class of polynomials was defined by Valiant [Val79a] as a non-deterministic analogue of VP. This class is denoted by VNP and is defined as follows.

**Definition 4.14 (VNP)** A family of polynomials  $f = (f_n)$  is said to be in VNP over the field  $\mathbb{F}$  if there exist functions  $k, \ell, m : \mathbb{N} \rightarrow \mathbb{N}$  all polynomially bounded, and a polynomial family  $g = (g_n) \in \text{VP}$  with  $g_n \in \mathbb{F}[x_1, \dots, x_{k(n)}, y_1, \dots, y_{m(n)}]$  such that for all  $n$ ,

$$f_n(x_1, \dots, x_{k(n)}) = \sum_{w \in \{0,1\}^{m(n)}} g_{\ell(n)}(x_1, \dots, x_{k(n)}, w_1, \dots, w_{m(n)}).$$

The polynomial family  $g$  is often referred to as the family of *verifier* polynomials, and the variables  $y_1, \dots, y_{m(n)}$  are called *witness* variables. This is analogous to the Boolean class NP which contains functions  $f(x_1, \dots, x_n)$  that can be written as a logical OR of a verifier function  $g(x_1, \dots, x_n, y_1, \dots, y_m)$ ,  $m = \text{poly}(n)$ , over all Boolean assignments to the witness variables  $y_1, \dots, y_m$ . The OR in NP is replaced by a summation in the definition of VNP. This structural shift aligns VNP naturally with #P, the complexity class that counts the number of accepting paths of a nondeterministic Turing machine, rather than merely asking if one exists. This analogy is made rigorous by Valiant's seminal results, which established that permanent polynomial is complete for VNP [Val79a], and is also #P-complete [Val79b] in the Boolean setting.

Clearly,  $\text{VP} \subseteq \text{VNP}$ . Motivated by Kaltofen's results on the factors of VP, Bürgisser [Bür00a, Conjecture 2.1] conjectured that VNP is closed under factorisation as well. Chou, Kumar, and Solomon [CKS19b, Theorem 2.9] showed that Bürgisser's conjecture is in fact true when the field has characteristic 0. The key ideas were based on a related but different factoring tool that we are going to discuss in the upcoming section.

Further, Bhargav, Dwivedi, and Saxena [BDS24, Theorem 1.6] proved that the factor closure holds over finite fields as well, using Hensel lifting in combination with Valiant's

criterion (and its converse) for *low-degree* polynomials to be in VNP (cf. [Bür00a, Proposition 2.20]). We denote by  $\langle \gamma \rangle$  the Boolean encoding of any mathematical object  $\gamma$ .

**Proposition 4.15 (Valiant’s Criterion)** *Consider a polynomial family  $f = (f_n)$  with the number of variables and degree of  $f_n$  bounded by  $\text{poly}(n)$ . Suppose  $f_n = \sum_e c_e \mathbf{x}^e$ . If for every  $n$ , there exists a function  $\phi_n \in \#\text{P}/\text{poly}$  such that given any exponent vector  $e$ , we have  $\phi_n(\langle e \rangle) = \langle c_e \rangle$ , then  $f \in \text{VNP}$ .*

Over finite fields, a weaker variant of Valiant’s criterion suffices, wherein the coefficient function  $\phi$  lies in  $\#\text{P}/\text{poly}$  (see [Bür00a, Section 4.3]). We omit this subtlety when it is clear from the context.

Finally, it is once again important to state that VNP is closed under several standard operations, such as addition, multiplication, and composition [Val82, Section 4]. However, the closure under composition is particularly subtle. A naive substitution of variables may not work, and a more refined approach is needed [CKS19b, Claim 8.4]. In particular, an alternative characterisation of VNP proves useful—namely, that VNP polynomials can be expressed as hypercube sums of polynomial-size formulas (see [MP08, Theorem 2] for a proof). We refer the reader to the full version of [BDS24] for detailed proofs.

**Theorem 4.16 (Closure of VNP)** *Let  $f(\mathbf{x})$  be an  $n$ -variate polynomial of degree  $d$  such that*

$$f(\mathbf{x}) = \sum_{\mathbf{w} \in \{0,1\}^m} q(\mathbf{x}, \mathbf{w}),$$

where  $q(\mathbf{x}, \mathbf{y})$  can be computed by a circuit of size  $s$ . If  $g(\mathbf{x})$  is any factor of  $f$  of degree  $r \leq d$ , then it can be written as

$$g(\mathbf{x}) = \sum_{\mathbf{e} \in \{0,1\}^{m'}} u(\mathbf{x}, \mathbf{e}),$$

where  $m'$  and  $\text{size}(u)$  are both bounded by  $\text{poly}(n, s, r, d, m)$ .

*Proof Sketch.* We sketch a unified proof covering both characteristic zero and finite fields of prime characteristic  $p$ . In doing so, we highlight the novel ideas from [CKS19b], which proved the result for characteristic zero, and [BDS24], which extended it to the finite field setting. For the small characteristic case assume  $\mathbb{F} = \mathbb{F}_q$ , where  $q = p^a$ .

We aim to show that if  $f \in \text{VNP}$  and  $f = g^e \cdot h$  with  $\text{gcd}(g, h) = 1$ , then  $g \in \text{VNP}$ . Recall that in earlier proofs, the special case  $h = 1$  was handled separately. For large

fields, an identity of the form [Equation \(5\)](#) works in VNP as well. However, over finite fields, extra care is needed.

Let  $h = 1$  and  $e = p^i \cdot \hat{e}$ , where  $\gcd(\hat{e}, p) = 1$ . We first extract the  $\hat{e}$ -th root. Define the following auxiliary polynomial with  $g_1 := g^{p^i}$ :

$$\begin{aligned} \hat{F}(z) &:= z^{\hat{e}} - f = z^{\hat{e}} - (g^{p^i})^{\hat{e}} \\ &= (z - g_1) \cdot (z^{\hat{e}-1} + z^{\hat{e}-2}g_1 + \cdots + g_1^{\hat{e}-1}) \\ &= (z - g_1) \cdot Q. \end{aligned}$$

Since  $p \nmid \hat{e}$ , it is easy to prove that  $z - g_1$  is coprime to  $Q$ . We can therefore apply Hensel lifting to  $\hat{F}(z)$ . As observed in [[BDS24](#), Lemma 5.9], the lifting process yields a small circuit  $B$  and computes the factor  $(z - g_1)$ . Explicitly, this can be written as a composition:

$$(z - g_1) = B(\hat{f}_0(\mathbf{x}), \dots, \hat{f}_d(\mathbf{x})),$$

where the inputs  $\hat{f}_i(\mathbf{x})$  are the coefficients of the polynomial obtained after applying the standard factoring preprocessing steps to  $\hat{F}$ . Crucially, each of these polynomials  $\hat{f}_i(\mathbf{x})$  remains in VNP. Finally, using the closure of VNP under composition with polynomial-size circuits, we conclude that the factor  $(z - g_1)$ , and consequently  $g_1$ , belongs to VNP.

Therefore, the only remaining case is when  $\hat{e} = 1$  and  $f = g^{p^i}$ . Note that this case requires separate attention only over small characteristic fields. Since the characteristic of the field is  $p$ , we can associate the coefficients of  $f$  to those of  $g$  via the standard Frobenius map. In particular, if the coefficient of  $\mathbf{x}^k$  in  $g$  is  $c_g$ , and the coefficient of  $\mathbf{x}^{p^i \cdot k}$  in  $f$  is  $c_f$ , then  $c_g = (c_f)^{1/p^i}$ .

In [[BDS24](#)], the authors observed that the *converse* of Valiant's criterion holds over finite fields. In particular, all the coefficients of  $f$  can be computed by a #P/poly function. Together with the Frobenius correspondence between the coefficients, this yields a #P/poly function for computing the coefficients of the factor  $g$ . Then, using [Proposition 4.15](#), we can conclude that  $g$  can be written as a hypercube sum of small-sized circuits.  $\square$

*Remark 4.17* [Theorem 4.16](#) also partially answers [Problem 4.4.1](#) and [Problem 4.5.1](#), proving that factors of polynomials in VP (and VBP) are in VNP over finite fields. Robert Andrews communicated to us an (as yet) unpublished proof (based on the ideas of Malod and Portier [[MP08](#)] and Andrews [[And20](#)]) that extends the closure of VNP to all *perfect fields*. A field  $\mathbb{F}$  is perfect if either it is of characteristic zero, or, if the

characteristic is a prime  $p$ , then all elements of  $\mathbb{F}$  are  $p$ -th powers. All finite fields are perfect. Thus, it only remains to settle the closure of VNP over infinite non-perfect fields (e.g. the fraction field  $\mathbb{F}_p(t)$  for some indeterminate  $t$ ).

### ? Open Problem (4.6.1)

Is the class VNP closed under taking factors over infinite non-perfect fields of positive characteristic?

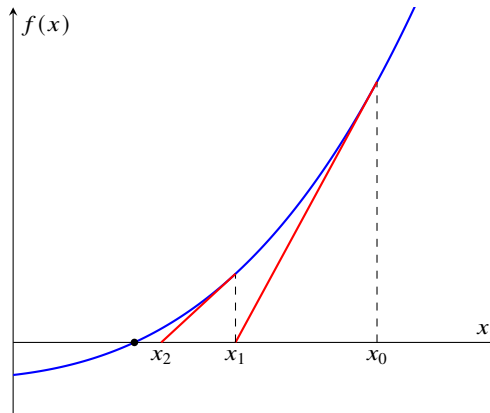
---

## 5 Factoring via Newton iteration

In 1669, Isaac Newton described a method to approximate roots of *real-valued* functions. Also known as the Newton-Raphson method, the idea is to start with an initial approximate root  $x_0$  of the equation  $f(x) = 0$  and successively produce better approximations using the rule

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)},$$

provided  $f'(x_i) \neq 0$ . The value  $x_{i+1}$  is the  $x$ -intercept of the tangent  $y = f(x_i) + f'(x_i)(x - x_i)$  to the curve of  $f(x)$  at the point  $x_i$ . The slope  $f'(x_i)$  of the tangent keeps changing with the approximation (Figure 7). Alternatively, this process can be viewed as finding a root of the Taylor approximation of  $f(x)$  (up to linear terms) at the point  $x_i$ . It is intuitive that  $x_{i+1}$  is closer than  $x_i$  to a root of  $f$ .

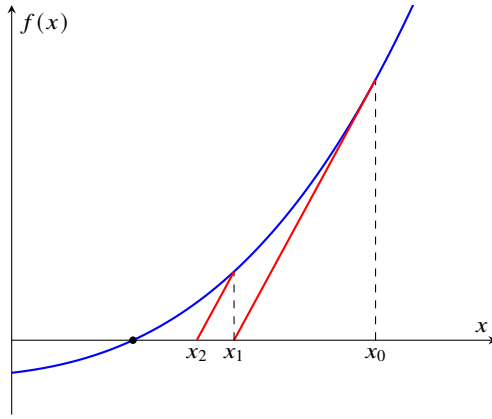


**Fig. 7** Fast Newton iteration (dynamic slope)

Assuming the derivative  $f'(x_0)$  at the initial guess is non-zero, a simpler rule

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_0)}$$

also works. Unlike the previous case, the slope  $f'(x_0)$  of the line  $y = f(x_i) + f'(x_0)(x - x_i)$  (which is no longer the tangent) remains unchanged, and convergence to the root is slower (Figure 8).



**Fig. 8** Slow Newton iteration (fixed slope)

For simplicity of exposition, we will work over an algebraically closed field  $\mathbb{F}$  of characteristic zero in this section, and remark about other fields when appropriate. Consider a multivariate polynomial  $f(\mathbf{x}, y)$  over  $\mathbb{F}$  whose root  $g(\mathbf{x})$  with respect to  $y$  (i.e.,  $f(\mathbf{x}, g(\mathbf{x})) = 0$ ) we want to find. Newton's method can be adapted to this case, provided the derivative is invertible and there is a meaningful notion of convergence. We view  $f$  as a univariate polynomial  $f \in \mathbb{F}(\mathbf{x})[y]$  over the field of functions  $\mathbb{F}(\mathbf{x})$ , and work over the formal power series ring  $\mathbb{F}[[\mathbf{x}]]$ . The ring of formal power series  $\mathbb{F}[[\mathbf{x}]]$  can be viewed as the *completion* of the polynomial ring  $\mathbb{F}[\mathbf{x}]$  with respect to the maximal ideal  $\langle \mathbf{x} \rangle$ , which endows it with the  $\langle \mathbf{x} \rangle$ -adic topology. Similar to the notion of  $p$ -adic numbers (Section 4.3), two multivariate polynomials (or power series)  $p(\mathbf{x})$  and  $q(\mathbf{x})$  are considered *close* up to degree  $t$  if they only differ in terms of degree greater than  $t$ , i.e.,  $p(\mathbf{x}) = q(\mathbf{x}) \pmod{\langle \mathbf{x} \rangle^{t+1}}$ . We first state the slower version of Newton's method that is useful in factoring applications.

**Lemma 5.1** ([CKS19b, Lemma 5.1]) *Let  $f(\mathbf{x}, y) \in \mathbb{F}(\mathbf{x})[y]$  be a polynomial over  $\mathbb{F}(\mathbf{x})$  and  $\mu \in \mathbb{F}$  be such that  $f(\mathbf{0}, \mu) = 0$  but  $\delta := \partial_y f(\mathbf{0}, \mu) \neq 0$ .*

Then, there is a unique power series  $\varphi \in \mathbb{F}[[\mathbf{x}]]$  with  $\varphi(\mathbf{0}) = \mu$  such that  $f(\mathbf{x}, \varphi) = 0$ . Set  $\varphi_0 = \varphi(\mathbf{0})$ , and for all  $t \geq 0$ , define

$$\varphi_{t+1} := \varphi_t - \frac{f(\mathbf{x}, \varphi_t)}{\delta}.$$

The rate of convergence of the sequence of polynomials  $\varphi_t \in \mathbb{F}[\mathbf{x}]$  to the root  $\varphi$  is linear:

$$\varphi = \varphi_t \bmod \langle \mathbf{x} \rangle^{t+1} \text{ for all } t \geq 0.$$

Suppose that the polynomial  $f(\mathbf{x}, y) \in \mathbb{F}[\mathbf{x}, y]$  has a degree- $r$  polynomial  $g(\mathbf{x}) \in \mathbb{F}[\mathbf{x}]$  as a root (w.r.t.  $y$ ). After  $r$  steps of Newton iteration, we get a polynomial  $\varphi_r$  (of degree possibly greater than  $r$ ) that agrees with  $g$  on all terms up to degree  $r$ . We obtain  $g$  by truncating  $\varphi_r$  up to terms of degree at most  $r$ . Seen differently, Newton iteration lets us find a *linear* factor  $(y - g(\mathbf{x}))$  of  $f(\mathbf{x}, y)$  by iteratively approximating the root  $g(\mathbf{x})$ , provided the root is of multiplicity one, also called a *simple* root. If the root is not simple, i.e.,  $y - g(\mathbf{x})$  occurs with multiplicity  $e > 1$  in  $f$ , the derivative at the root  $\partial_y f(\mathbf{x}, g(\mathbf{x})) = 0$ , so we consider instead the  $(e - 1)$ -th partial derivative  $\partial_{y^{e-1}} f(\mathbf{x}, y)$ , which has  $g(\mathbf{x})$  as a simple root.

*Remark 5.2* If the field  $\mathbb{F}$  has characteristic  $p$ , taking derivatives to make the root simple would kill the polynomial, unless the multiplicity  $e$  is coprime to  $p$ . In case  $e = p^\ell e'$  where  $e'$  is coprime to  $p$ , we can replace  $y$  with  $z^{1/p^\ell}$ , making  $g(\mathbf{x})^{p^\ell}$  a root (w.r.t.  $z$ ) of multiplicity  $e'$ , that *can* be made simple by taking derivatives. Hence, we can only approximate the  $p^\ell$ -th power of  $g$ , and not  $g$  itself.

A ‘random’ invertible shift of the variables  $\mathbf{x} \mapsto \mathbf{x} + \alpha y + \beta$ , where  $\alpha$  and  $\beta$  are random elements from  $\mathbb{F}^n$ , ensures that  $f$  is monic in  $y$  and thus, so are its factors (see, e.g., [DSS22, Lemma 14]). However, in general, the factors of  $f(\mathbf{x}, y)$  may not be linear. Nevertheless, over the algebraically closed field  $\overline{\mathbb{F}(\mathbf{x})}$ ,  $f$  uniquely factors as  $\prod_i (y - \varphi_i(\mathbf{x}))^{e_i}$  where for all  $i$ ,  $e_i > 0$  and  $\varphi_i(\mathbf{x}) \in \overline{\mathbb{F}(\mathbf{x})}$ . In fact, it can be shown that after the variable shift,  $f$  splits over the power series ring i.e.,  $\varphi_i(\mathbf{x}) \in \mathbb{F}[[\mathbf{x}]]$  for all  $i$  (see [DSS22, Theorem 17] for a proof). Moreover, the constant terms of the roots  $\mu_i := \varphi_i(\mathbf{0})$  are all *distinct* non-zero field elements. Using Newton iteration, we can approximate these power series roots.

*Remark 5.3* If the field  $\mathbb{F}$  is not algebraically closed, we might only find all the roots  $\mu_i$  in an extension  $\mathbb{K} \supseteq \mathbb{F}$  of degree  $k!$  (at the most), where  $k$  is the degree of  $f$  with respect to  $y$ . We will then have that the power series roots  $\varphi_i(\mathbf{x}) \in \mathbb{K}[[\mathbf{x}]]$ .

We now state a version of Newton iteration with *quadratic* rate of convergence.

**Lemma 5.4 ([DSS22, Lemma 15])** *Let  $f(\mathbf{x}, y) \in \mathbb{F}(\mathbf{x})[y]$  be a polynomial over  $\mathbb{F}(\mathbf{x})$  and  $\mu \in \mathbb{F}$  be such that  $f(\mathbf{0}, \mu) = 0$  but  $\delta := \partial_y f(\mathbf{0}, \mu) \neq 0$ .*

*Then, there is a unique power series  $\varphi \in \mathbb{F}[[\mathbf{x}]]$  with  $\varphi(\mathbf{0}) = \mu$  such that  $f(\mathbf{x}, \varphi) = 0$ . Set  $\varphi_0 = \varphi(\mathbf{0})$ , and for all  $t \geq 0$  define*

$$\varphi_{t+1} := \varphi_t - \frac{f(\mathbf{x}, \varphi_t)}{\partial_y f(\mathbf{x}, \varphi_t)}.$$

*Since  $\varphi_t(\mathbf{0}) = \mu$  for all  $t \geq 0$  and since  $\delta \neq 0$ , the denominator above is invertible in the power series ring. The sequence of rational functions  $\varphi_t(\mathbf{x}) \in \mathbb{F}(\mathbf{x})$  converges to  $\varphi$  at a quadratic rate:*

$$\varphi = \varphi_t \bmod \langle \mathbf{x} \rangle^{2^t} \text{ for all } t \geq 0.$$

The above Newton-Iteration lemmas can also be viewed as power series versions of the *Implicit Function Theorem* (see, e.g. [KP13, Chapter 1] and [Art22, Theorem 9.2.1]). Returning to the problem of factoring  $f(\mathbf{x}, y)$ , suppose it has a non-trivial factor  $g(\mathbf{x}, y)$  of degree  $r$ . As discussed earlier, we can assume  $g$  is a factor of multiplicity one by considering appropriate partial derivatives. Since  $f$  is monic in  $y$ , so is  $g$ , and splits as  $g = \prod_{i=1}^r (y - \varphi_i)$  for some power series roots  $\varphi_i(\mathbf{x}) \in \mathbb{F}[[\mathbf{x}]]$ . Crucially, these  $\varphi_i$ 's are a subset of the roots of  $f$ . Given this set (via the *distinct* degree-0 terms of the  $\varphi_i$ 's), we can build the approximate power series roots up to degree  $r$  using Newton iteration on  $f$ . Finally, in order to obtain  $g$ , it is enough to multiply the approximate linear factors corresponding to  $g$ , truncate the result up to terms of degree  $r$ , and invert the variable shift.

*Remark 5.5* The closure of a particular algebraic model under factoring does not require an efficient algorithm to *find* the roots (factors), but only the *structural* ability to perform Newton iteration (and other pre- and post-processing steps) within the model. Whether we use the faster or slower version of Newton's method will also depend on the convenience afforded by the model. We will have a chance to briefly comment on algorithms for factoring once we discuss the factor closure of a model.

## 5.1 Low depth circuits

In this section, we consider factors of polynomials computable by small-size circuits that are also of *bounded-depth*. These circuits form a fascinating subclass of arithmetic circuits since general circuits can be “efficiently” depth-reduced, unlike Boolean circuits. An influential body of work [Val+83; AV08; Koi12; Tav15; Gup+16] shows that any

circuit of size  $s$  computing a polynomial of degree  $d$  can be equivalently written as a *homogeneous* depth-4 circuit of size  $s^{O(\sqrt{d})}$ , or even a depth-3 circuit of size  $s^{O(\sqrt{d})}$  (over fields of characteristic zero), if one foregoes homogeneity.

Dvir, Shpilka, and Yehudayoff [DSY09] studied factors of low-depth circuits in order to extend the Hardness-Randomness paradigm of Kabanets and Impagliazzo [KI04] (see Section 3.2) to this model. Of particular interest to them were factors of the form  $y - g(\mathbf{x})$  of a bounded-depth circuit  $f(\mathbf{x}, y)$ . They showed that if the individual degree of  $f$  with respect to  $y$  is bounded, then the root  $g(\mathbf{x})$  has a small bounded-depth circuit.

**Theorem 5.6 ([DSY09, Theorem 4])** *Let  $f \in \mathbb{F}[\mathbf{x}, y]$  be an  $(n+1)$ -variate polynomial with  $\deg(f) \leq d$  and  $\deg_y(f) \leq k$ , computed by a depth- $\Delta$  circuit of size  $s$ . Then, its factors of the form  $y - g(\mathbf{x})$  with  $\deg(g) = r$  can be computed by circuits of size  $\text{poly}(s, r^k, d, n)$  and depth  $\Delta + O(1)$ .*

*Proof Sketch.* The obvious approach would be to use Lemma 5.1 to successively approximate the root  $g(\mathbf{x})$  via Newton iteration. Let us assume the preconditions for using the lemma are met. The crucial observation [DSY09, Lemma 3.1] is that the  $t$ -th approximate  $\varphi_t$  can be expressed as a  $(k+1)$ -variate, degree- $t$  polynomial  $Q_t$  in the coefficients of  $f$ , i.e.,  $\varphi_t = Q_t(f_0, \dots, f_k)$ , where  $f(\mathbf{x}, y) = \sum_{j=0}^k f_j(\mathbf{x})y^j$ , with  $f_j(\mathbf{x}) \in \mathbb{F}[\mathbf{x}]$ . After  $r$  iterations, we get the polynomial  $\varphi_r = Q_r(f_0, \dots, f_k)$  of degree at most  $dr$  such that  $g = \varphi_r \bmod \langle \mathbf{x} \rangle^{r+1}$ .

The sparsity of  $Q_r$  is at most  $\binom{r+k}{k} = O(r^k)$ . Composing the trivial depth-2 circuit for  $Q_r$  (of size  $O(r^k)$ ) with the depth- $\Delta$  circuits for the  $f_j$ 's (of size  $O(sk)$ ) obtained via interpolation (Lemma 2.2) on  $f$ , we have a circuit for  $\varphi_r$  of depth  $\Delta + 2$  and size  $\text{poly}(r^k, s)$ . To extract  $g$ , we again use interpolation by first mapping  $x_i \rightarrow zx_i$  and using Lemma 2.2 to obtain all the components of  $\varphi_r$  of degree at most  $r$  (w.r.t.  $z$ ) causing a size blow-up of  $\text{poly}(r^k, s, d)$ . The depth does not change.

By repeatedly taking partial derivatives of  $f$  using Lemma 2.3 (incurring a size blow-up of at most  $\text{poly}(s, k)$ ) and a translation of the coordinates (resulting in a depth increase by 1), we can ensure that  $g$  is a simple root of  $f$  (and thus, Lemma 5.1 is applicable). So we have a depth  $\Delta + 3$  circuit for  $y - g(x)$  of size  $\text{poly}(r^k, s, d, n)$ .  $\square$

*Remark 5.7* In light of Remark 5.2, if the field  $\mathbb{F}$  is of characteristic  $p$ , then Theorem 5.6 only applies to  $p$ -th powers of linear factors. For a factoring *algorithm*, we work over the field of rational numbers  $\mathbb{Q}$ , or a finite field. We note that the preprocessing steps to ensure applicability of Newton iteration can be performed efficiently, with access to randomness. The initial value  $\varphi(\mathbf{0}) = \mu$  (which is guaranteed to be in the field if a root

$g$  exists) is found by factoring  $f(\mathbf{0}, y)$  using a univariate factoring algorithm, and each distinct  $\mu$  lifts to a different root of  $f$ . Therefore, the arguments also yield a *randomised* polynomial time algorithm to *find* the linear factors of  $f$  [DSY09, Theorem 4].

Rafael Oliveira [Oli16] generalised this result to show that polynomials of bounded individual degrees and computable by small bounded-depth circuits are closed under factorisation. A little thought reveals that if a polynomial has its individual degree bounded, then so will its factors.

**Theorem 5.8 ([Oli16, Theorem 1.2])** *Let  $f \in \mathbb{F}[\mathbf{x}, y]$  be an  $(n+1)$ -variate polynomial with  $\deg(f) \leq d$ , and individual degree bounded by  $k$ . Suppose  $f$  can be computed by a depth- $\Delta$  circuit of size  $s$ . Then, any factor  $g(\mathbf{x}, y)$  of degree  $r$  that divides  $f$  can be computed by a circuit of size  $\text{poly}(r^k, k, s, d, n)$  and depth  $\Delta + O(1)$ .*

*Proof Sketch.* The main idea is to use the argument at the beginning of Section 5 and essentially lift the above result of Dvir, Shpilka and Yehudayoff to all factors. Let us begin by writing  $f(\mathbf{x}, y) = \sum_{j=0}^k f_j(\mathbf{x})y^j$  and  $g(\mathbf{x}, y) = \sum_{j=0}^{\ell} g_j(\mathbf{x})y^j$ , where  $\ell \leq k$ . A shift of the variables  $\mathbf{x} \mapsto \mathbf{x} + \alpha y + \beta$  to ensure that  $f$  is monic in  $y$ , however, would make its individual degree  $kn$ . Therefore, in order to reduce to the monic case, we rewrite

$$f(\mathbf{x}, y) = f_k(\mathbf{x}) \left( y^k + \sum_{j=0}^{k-1} f_j(\mathbf{x})/f_k(\mathbf{x})y^j \right) \text{ and}$$

$$g(\mathbf{x}, y) = g_{\ell}(\mathbf{x}) \left( y^{\ell} + \sum_{j=0}^{\ell-1} g_j(\mathbf{x})/g_{\ell}(\mathbf{x})y^j \right).$$

The idea is that, since  $g$  divides  $f$ , the leading term  $g_{\ell}$  divides  $f_k$ . If we are able to recover the factor  $g_{\ell}$  from  $f_k$  by an induction on the number of variables (note the one fewer variable), and we somehow obtain a circuit for  $g/g_{\ell}$  using Newton iteration for monic polynomial factoring, then multiplying the circuits for  $g_{\ell}$  and  $g/g_{\ell}$  would give us  $g$ .

Although we can extract  $f_k$  from  $f$  via interpolation, we cannot afford the resulting size blow-up (by a factor of  $k$ ) in the induction argument. To avoid this, we work with the polynomial  $\tilde{f}(\mathbf{x}, y) := y^k f(\mathbf{x}, 1/y)$ , which is the *reversal* of (the coefficients of) the polynomial  $f$ . The leading coefficient of  $\tilde{f}$  is now  $f_0(\mathbf{x}) = f(\mathbf{x}, 0)$  which has a smaller circuit than  $f$ . Moreover, the factors of the reversal of a polynomial are just the reversals of the original factors.

Once we find (by induction) a circuit of depth  $\Delta + O(1)$  and size  $\text{poly}(r^k, s, d)$  for  $g_0(\mathbf{x})$  (the leading coefficient of  $\tilde{g}(\mathbf{x}, y)$ ), we can use Theorem 5.6 to approximate the

roots  $\varphi_i(\mathbf{x}) \in \mathbb{F}[\mathbf{x}]$  (up to degree  $d$ ) of the monic rational function  $\tilde{g}/g_0 = \prod_{i=1}^{\ell} (y - \varphi_i)$ . Using Newton iteration on  $\tilde{f}$ , whose degree in  $y$  is bounded by  $k$ , we can approximate these roots by circuits of depth  $\Delta + O(1)$  and size  $\text{poly}(r^k, s, d)$ . We further obtain a circuit of size  $\text{poly}(r^k, s, k, d)$  for  $\tilde{g}/g_0$  by multiplying the approximations for the circuits and truncating using interpolation, resulting in a depth increase by at most a constant. Finally, we get  $g(\mathbf{x}, y)$  by multiplying  $g_0$  and  $\tilde{g}/g_0$  and computing the reversal of  $\tilde{g}$ .  $\square$

*Remark 5.9* If the field  $\mathbb{F}$  has characteristic  $p$ , then we only obtain factors up to a power of  $p$  (see [Remark 5.2](#)). Over fields that are not algebraically closed, [Remark 5.3](#) implies that the circuit we obtain for the factors is over an extension field  $\mathbb{K} \supseteq \mathbb{F}$  of degree at most  $k!$ . This circuit can be simulated over  $\mathbb{F}$  with a  $\text{poly}(k!)$  size blow up, and depth increase by a factor of  $O(\log k!)$  [[HY11](#), Theorem 4.2], which is efficient when  $k$  is bounded.

The above result can be used to obtain a *randomised* algorithm running in time  $\text{poly}(s, (nk)^k)$  to compute all the factors of the polynomial  $f$  [[Oli16](#), Remark 1.3]. In addition to the ideas in [Remark 5.7](#), we need to determine which power series roots correspond to a factor  $g$  over  $\mathbb{F}$ , and this can be done by testing (deterministically) the  $2^k$  possible subsets of the roots.

Chou, Kumar, and Solomon [[CKS19b](#)] removed the restriction on individual degrees at the expense of requiring that the *total degree* of the factor be small.

**Theorem 5.10** ([\[CKS19b, Theorem 2.1\]](#)) *Let  $f \in \mathbb{F}[\mathbf{x}, y]$  be an  $(n + 1)$ -variate polynomial of degree  $d$ . Suppose  $f$  can be computed by a depth- $\Delta$  circuit of size  $s$ . Then, any factor  $g(\mathbf{x}, y)$  of degree  $r$  that divides  $f$  can be computed by a circuit of size  $\text{poly}(r^{\sqrt{r}}, n, d, s)$  and depth  $\Delta + O(1)$ .*

*Proof Sketch.* The broad outline is still the reduction of factoring to approximating roots, same as in the proof of [Theorem 5.8](#). Consider the simple case when  $g$  is a root of  $f$ . If  $\deg_y(f) \geq n$ , the earlier observation by Dvir, Shpilka and Yehudayoff that there exists a  $(\deg_y(f) + 1)$ -variate polynomial  $\varphi_r$  that approximates  $g$  up to degree  $r$  becomes trivial since  $g$  is already an  $n$ -variate polynomial.

However, irrespective of  $\deg_y(f)$ , Chou, Kumar, and Solomon show that  $\varphi_r$  can in fact be written as an  $(r + 1)$ -variate,  $\text{poly}(r)$ -sized, degree- $r$  polynomial  $A_r$  in the (at most)  $r$ -th order partial derivatives of  $f$  [[CKS19b](#), Lemma 5.3]<sup>3</sup>. Using the depth-

<sup>3</sup> A proof of closure of VNP under factoring also follows from here using closure properties of VNP under coefficient extraction and composition (see [[CKS19b](#), Claim 8.3 and Claim 8.4]).

reduction results mentioned at the beginning of this section, we can further squash this to a depth-3 circuit for  $A_r$  of size  $r^{O(\sqrt{r})}$  when the field has characteristic zero. Composition with the depth- $\Delta$ ,  $\text{poly}(s, d, r)$ -sized circuits for the partial derivatives results in a depth- $(\Delta + 3)$  circuit for  $\varphi_r$ , from which we can extract  $g$  using interpolation, as before.

The rest of the argument to lift this to a general factor remains almost the same as in [Theorem 5.8](#). The argument does not need special handling of the leading coefficient, as we do not need to worry about maintaining individual degree after the coordinate shift.  $\square$

*Remark 5.11* If the degree  $r$  of the factor is at most  $O(\log^2 n / \log \log n)$ , then the size bound for  $g$  in [Theorem 5.10](#) above is  $\text{poly}(n)$ .

Recently, there has been a flurry of activity on *derandomising* the factorisation algorithms for constant depth circuits [[KRS24](#); [Kum+24](#); [DST24](#)]. There has also been progress on computing GCD, resultants, and various other problems in linear algebra using circuits of constant depth [[AW24](#); [Bha+25b](#)]. Building on this, Bhattacharjee, Kumar, Ramanathan, Saptharishi, and Saraf [[Bha+25c](#)] gave the first subexponential time *deterministic* algorithm for constant depth circuit factorisation. Soon after, the same authors along with Rai [[Bha+25a](#)] showed that if a polynomial can be computed by a depth  $\Delta$  circuit of size  $s$ , then any of its factors can be computed by a circuit of size  $\text{poly}(s)$  and depth  $\Delta + O(1)$ , thereby positively answering the factorisation question for bounded depth circuits. We will sketch this breakthrough technique later in [Section 7.1](#). We note that the positive characteristic case still remains open.

### ? Open Problem (5.1.1)

Let  $f \in \mathbb{F}_q[\mathbf{x}]$  be a multivariate polynomial over a finite field  $\mathbb{F}_q$  of prime characteristic  $p$ . If  $f$  can be computed by a size  $s$ , depth  $\Delta$  circuit, is it the case that any factor of  $f$  can be computed by a circuit of depth  $O(\Delta)$  and size  $\text{poly}(s)$ ?

---

## 5.2 High degree circuits

The families of polynomials we considered till now had ‘low’ degree, bounded by a polynomial in the number of variables. Malod [[Mal03](#); [Mal07](#)] considered the case when the degrees are not bounded.

**Definition 5.12** A family of polynomials  $f = (f_n)$  is said to be in  $\text{VP}_{nb}$  over the field  $\mathbb{F}$  if the number of variables and the size of the smallest circuit for  $f_n$  (over  $\mathbb{F}$ ) are bounded by  $\text{poly}(n)$ .

Note that the degree of a polynomial family in  $\text{VP}_{nb}$  can be *exponential* in the size of the circuit. Kaltofen [Kal87] showed (Theorem 4.9) that for an  $n$ -variate polynomial  $f = g^e h$  where  $g$  and  $h$  are coprime and  $g$  has multiplicity  $e$ ,

$$\text{size}(g) = \text{poly}(\text{size}(f), \deg(g), e, n). \quad (6)$$

Hence, the best size upper bound one can deduce for (exponential-degree) factors of polynomials in  $\text{VP}_{nb}$  from Kaltofen's result is exponential. This is unavoidable in general: the polynomial  $x^{2^n} - 1 = \prod_{i=1}^{2^n} (x - \xi^i)$  where  $\xi$  is a  $2^n$ -th root of unity, can be computed by a circuit of size  $O(n)$ . Lipton and Stockmeyer [LS78] showed that a random exponential-degree factor  $\prod_{i \in S} (x - \omega^i)$  where  $S \subset [2^n]$  and  $|S| = \exp(n)$ , requires  $\exp(n)$ -size circuits. So  $\text{VP}_{nb}$  is *not* closed under taking factors<sup>4</sup>. Nevertheless, Bürgisser's *Factor Conjecture* [Bür00a, Conj. 8.3] (stated below) is that the size of the factor  $g$  in Equation (6) should be independent of its multiplicity  $e$ . In particular,  $\text{poly}(n)$ -degree factors of a polynomial family in  $\text{VP}_{nb}$  should be in  $\text{VP}$ .

### ? Open Problem (5.2.1)

Show that if  $g(\mathbf{x})$  is a factor of an  $n$ -variate polynomial  $f(\mathbf{x})$  then,

$$\text{size}(g) = \text{poly}(\text{size}(f), \deg(g), n).$$

---

See [Bür04, Section 4] for some applications of the above conjecture to decision complexity. Over the years, there has been partial progress towards a resolution. In the case when  $f$  is a power of  $g$ , Kaltofen [Kal87] (cf. [Bür04, Proposition 6.1]) already showed the conjecture to be true (see Theorem 4.6 for an alternative proof).

**Theorem 5.13 ([Kal87, Theorem 2])** *Suppose  $f(\mathbf{x}) = g(\mathbf{x})^e \in \mathbb{F}[\mathbf{x}]$  is an  $n$ -variate polynomial. Then,*

$$\text{size}(g) = \text{poly}(\text{size}(f), \deg(g), n).$$

*Proof Sketch.* We cannot directly approximate the roots of  $f$  (and hence  $g$ ) as outlined in Section 5 since we cannot reduce to the multiplicity one case by taking  $(e - 1)$  partial

<sup>4</sup> In fact, the same argument also shows the non-closure of  $\text{VNP}_{nb}$ .

derivatives - the size blow up will be  $\text{poly}(\text{size}(f), e)$ . Instead, we find the root of the equation  $y^e - f(\mathbf{x})$  by Newton iteration (Lemma 5.4).

By shifting variables we can ensure that all the  $O(\log(\deg(g)))$  steps can be performed. Crucially, we only need the homogeneous components of  $f$  up to  $\deg(g)$  for the intermediate steps, and any involved exponentiations can be performed using repeated squaring at most  $O(\log e)$  times. Straightforward analysis will show that

$$\text{size}(g) = \text{poly}(\text{size}(f), \deg(g), n, \log e),$$

whereby noting that  $e \leq \exp(\text{size}(f))$  gives the result.  $\square$

As another special case of the Factor Conjecture, Dutta, Saxena, and Sinhababu [DSS22] showed that the conjecture also holds if the *square-free* part of  $f$  is of low degree. Suppose  $f = \prod_{i=1}^m f_i^{e_i}$  is the complete factorisation of  $f$  with  $f_i$ 's irreducible. Recall that the square-free part of  $f$  (also called the *radical*) is the polynomial  $\text{rad}(f) := \prod_{i=1}^m f_i$ .

**Theorem 5.14** *Let  $f(\mathbf{x}, y)$  be an  $n$ -variate polynomial and let  $g(\mathbf{x}, y)$  be a factor of  $f$ . We then have*

$$\text{size}(g) = \text{poly}(\text{size}(f), \deg(\text{rad}(f)), n).$$

*Proof Sketch.* We follow the exposition of Sinhababu [Sin19]. Consider the derivative

$$\partial_y f(\mathbf{x}, y) = \partial_y \left( \prod_{i=1}^m f_i^{e_i} \right) = \sum_{i=1}^m e_i f_i^{e_i-1} \partial_y f_i \left( \prod_{j \neq i} f_j^{e_j} \right).$$

Rewrite it as  $\partial_y f = \prod_{i=1}^m f_i^{e_i-1} u$  with  $u = \sum_{i=1}^m e_i \partial_y f_i \left( \prod_{j \neq i} f_j \right)$ . The auxiliary polynomial  $F := f + z \partial_y f$  factors as  $\prod_{i=1}^m f_i^{e_i-1} \left( \prod_{i=1}^m f_i + z \cdot u \right)$ , where  $z$  is a fresh variable. The factor  $G := \text{rad}(f) + z \cdot u$  is coprime to  $\prod_{i=1}^m f_i^{e_i-1}$ , has the same degree as  $\text{rad}(f)$  and is of multiplicity one. Using Kaltofen's result (Theorem 4.9), we get  $\text{size}(G) = \text{poly}(\text{size}(f), \deg(\text{rad}(f)), n)$ . One can obtain  $\text{rad}(f)$  by setting  $z = 0$  in  $G$ , and any irreducible factor  $f_i$  in size  $\text{poly}(\text{size}(f), \deg(\text{rad}(f)), n)$  by further factoring  $\text{rad}(f)$ . The result follows by suitably combining powers of  $f_i$  (via repeated squaring) to form  $g$ .  $\square$

Shortly after proposing the factor conjecture, Bürgisser [Bür04] showed its plausibility: low-degree factors can be *approximated* by small circuits! We will discuss this next.

### 5.3 Algebraic approximation

A natural notion of approximation in Valiant's framework was studied by Bürgisser [Bür04] (refer [BCS97, Ch. 15, 20] for historical motivation). A polynomial  $f \in \mathbb{F}[\mathbf{x}]$  is said to be approximated by another polynomial  $F \in \mathbb{F}[\varepsilon][\mathbf{x}]$  to an *order of approximation*  $M$  if

$$F(\mathbf{x}, \varepsilon) = \varepsilon^M f(\mathbf{x}) + \varepsilon^{M+1} Q(\mathbf{x}, \varepsilon), \quad (7)$$

for some polynomial  $Q(\mathbf{x}, \varepsilon) \in \mathbb{F}[\mathbf{x}, \varepsilon]$ . The *approximate/border size* of  $f$ , denoted  $\overline{\text{size}}(f)$ , is defined as the size of the smallest circuit *over the ring of constants*  $\mathbb{F}[\varepsilon]$  computing a polynomial  $F$  that approximates  $f$ .

Over fields like  $\mathbb{R}$  or  $\mathbb{C}$ , one can think of the above as an approximation in the sense  $\lim_{\varepsilon \rightarrow 0} \varepsilon^{-M} F = f$ . Another way of formulating the notion of approximation is to consider  $F_\varepsilon \in \mathbb{F}(\varepsilon)[\mathbf{x}]$  over the rational function field  $\mathbb{F}(\varepsilon)$  instead, and require an approximation of the form

$$F_\varepsilon(\mathbf{x}) = f(\mathbf{x}) + \varepsilon Q_\varepsilon(\mathbf{x}),$$

where  $Q_\varepsilon \in \mathbb{F}[\varepsilon][\mathbf{x}]$  is a polynomial. The border complexity of  $f$  is defined as before, but with the circuit size now calculated over  $\mathbb{F}(\varepsilon)$ . In this case, although  $F_{\varepsilon=0} = f$  is defined at  $\varepsilon = 0$ , the intermediate computations in the circuit for  $F$  (over  $\mathbb{F}(\varepsilon)$ ) might not be. Scaling arguments show that these two notions of approximation are equivalent [Bür04, Lemma 5.6]. For a detailed discussion on other natural definitions of approximation (both topological and algebraic), and their equivalence, we point the reader to [Bür04, Section 5], [BCS97, Appendix 20.6] and [BIZ18, Section 2]. The notion of border complexity naturally suggests an approximation version of the class VP.

**Definition 5.15** ( $\overline{\text{VP}}$ ) A polynomial family  $f = (f_n)$  is in the class  $\overline{\text{VP}}$  if the number of variables of  $f_n$ , its degree, and the size of the smallest circuit over  $\mathbb{F}[\varepsilon]$  approximating  $f_n$  are bounded by  $\text{poly}(n)$ .

It is clear that  $\text{VP} \subseteq \overline{\text{VP}}$ . In an attempt to utilise sophisticated tools from algebraic geometry and representation theory to study Valiant's conjecture, Mulmuley and Sohoni [MS01; MS08] proposed  $\overline{\text{VP}} \not\subseteq \text{VNP}$  as the mathematically nicer (but possibly harder) conjecture. Further details on the *Geometric Complexity Theory* (GCT) program for proving lower bounds can be found in [Reg02; Mul11; Bür+11; Mul12; Gro12; Lan17; BI25].

Returning to the factorisation problem, Bürgisser showed that  $\text{poly}(n)$ -degree factors of families in  $\text{VP}_{nb}$  are in  $\overline{\text{VP}}$ .

**Theorem 5.16 ([Bür04, Theorem 1.3])** *Let  $\mathbb{F}$  be a field of characteristic 0 and let  $f(\mathbf{x}, y)$  be an  $(n + 1)$ -variate polynomial over  $\mathbb{F}$ . Suppose  $g(\mathbf{x}, y)$  is a factor of  $f$ . Then,*

$$\overline{\text{size}}(g) = \text{poly}(\text{size}(f), \deg(g), n).$$

*Proof Sketch.* It suffices to show the result for irreducible  $g$ . Let  $f = g^e h$  where  $g$  and  $h$  are coprime. As earlier, we would like to approximate the roots of  $g$ . Due to the possibly exponential multiplicity  $e$ , we cannot reduce to the case of a simple root by taking derivatives of  $f$ . We can, however, find a point  $p$  (equal to  $(\mathbf{0}, 0)$  after shifting coordinates), such that  $g$  vanishes at  $p$  but both  $h$  and  $\partial_y g$  do not [Bür04, Lemma 3.3]. We can then try to build the corresponding (unique) power series root  $\varphi \in \mathbb{F}[[\mathbf{x}]]$  of  $g$  (which it shares with  $f$ ) using Newton iteration (Lemma 5.4) on  $f$ . This seems quite straightforward, except that  $\partial_y f = e g^{e-1} h \partial_y g + g^e \partial_y h$  always vanishes at  $p$  when  $e > 1$ . The main idea is to consider the *perturbed* polynomial

$$F_\varepsilon(\mathbf{x}, y) := f(\mathbf{x}, y + \varepsilon) - f(\mathbf{0}, \varepsilon).$$

Note that  $F_{\varepsilon=0} = f$  and  $F_\varepsilon(\mathbf{0}, 0) = 0$ . We encourage the reader to check that the derivative  $\partial_y F_\varepsilon(\mathbf{0}, 0)$  does not vanish anymore, and hence we can use Lemma 5.4 to construct the root  $\psi_\varepsilon$  of  $F_\varepsilon$ , which in fact is an algebraic approximation of the root  $\varphi$  of  $f$  (and also  $g$ ), i.e.  $\psi_{\varepsilon=0} = \varphi$  (see [Bür04, Proposition 3.4]). This way, we can efficiently *approximate* the factor  $g$ .  $\square$

Recently, an alternative proof of the above theorem was given in [Bha+25a], replacing the use of Lemma 5.4 with a classical power series root identity, which provided a conceptually simpler analysis of the size of the approximating circuit. The approximate circuit constructed for the low-degree factor in Theorem 5.16 has additional structure. Note that, a priori, the order of approximation  $M$  for a polynomial  $f$  can be arbitrarily large. However, Bürgisser [Bür04; Bür20, Theorem 5.7] showed that over algebraically closed fields,  $M = \exp(\overline{\text{size}}(f))$ . Therefore, the free constants from  $\mathbb{F}[\varepsilon]$  used for approximation can be assumed to be ‘only’ exponential in degree and, hence, size. Since the circuit for the factor  $g$  above is constructed via Newton iteration, the constants in  $\mathbb{F}[\varepsilon]$  are in fact circuits of size  $\text{poly}(\text{size}(f))$  (but exponential in degree). Hence, imposing this restriction on the size of the univariate polynomials in  $\varepsilon$  is quite natural.

**Definition 5.17** ( $\overline{\text{VP}}_\varepsilon$ ) A polynomial family  $f = (f_n)$  over the field  $\mathbb{F}$  is in the class  $\overline{\text{VP}}_\varepsilon$  if the number of variables of  $f_n$ , its degree, and the size of the smallest circuit over  $\mathbb{F}$  approximating  $f_n$  are bounded by  $\text{poly}(n)$ .

Note that the circuit size of the approximating polynomial is over the base field  $\mathbb{F}$ , thus also incorporating the size of the constants in  $\mathbb{F}[\varepsilon]$ . We call such circuits ‘presentable’, and  $\overline{\text{VP}}_\varepsilon$  as *presentable*  $\overline{\text{VP}}$ . Over finite fields, it can be shown that  $\overline{\text{VP}}_\varepsilon$  is contained in VNP, a fact not known about  $\overline{\text{VP}}$ .

**Theorem 5.18** (**[BDS24, Theorem 1]**) *Let  $\mathbb{F}$  be a finite field, and  $f \in \mathbb{F}[\mathbf{x}]$  be an  $n$ -variate polynomial of degree  $d$  such that the size of the smallest circuit (over  $\mathbb{F}$ ) approximating  $f$  is of size  $s$ . Then,  $f$  can be written as*

$$f(\mathbf{x}) = \sum_{\mathbf{a} \in \{0,1\}^m} g(\mathbf{x}, \mathbf{a}),$$

where  $m$ , and  $\text{size}_{\mathbb{F}}(g(\mathbf{x}, \mathbf{y}))$  are bounded by  $\text{poly}(s, n, d)$ .

*Proof Sketch.* The main idea is to use Valiant’s criterion ([Proposition 4.15](#)) to show that the polynomial  $f = \sum_{\mathbf{e}} c_{\mathbf{e}} \mathbf{x}^{\mathbf{e}}$  has coefficients  $c_{\mathbf{e}}$  that are “easy to describe”. Notice that the polynomial  $F$  approximating  $f$  has a small circuit but exponential degree, where the high degree is only due to  $\varepsilon$ .

In the approximating expression for  $f$  ([Equation \(7\)](#)), we interpolate  $F$  on *all* the variables (including  $\varepsilon$ ) using appropriate powers of unity. Consequently, each coefficient  $c_{\mathbf{e}}$  of  $f$  can be written as a hypercube sum over a small-size high-degree circuit. When  $\mathbb{F}$  is a finite field, we can *simulate* this algebraic circuit using a small Boolean circuit. As a result, we further obtain that the coefficient function of  $f$  is in #P/poly, whence we can apply Valiant’s criterion.  $\square$

As a corollary of [Theorem 5.18](#), and the observation that the approximate circuit in [Theorem 5.16](#) is presentable, we can conclude that the families of low-degree factors of high-degree circuits are in VNP, making partial progress towards the Factor Conjecture ([Problem 5.2.1](#)) which postulates that low-degree factors are in VP.

**Corollary 5.19** *Let  $f \in \mathbb{F}_q[\mathbf{x}]$  be an  $n$ -variate polynomial over a finite field  $\mathbb{F}_q$  of characteristic  $p$ . Suppose  $g$  is a  $\text{poly}(n)$ -degree irreducible factor of  $f$  with multiplicity coprime to the characteristic  $p$ . If  $\text{size}(f) = s$ , then  $g$  can be written as*

$$g(\mathbf{x}) = \sum_{\mathbf{a} \in \{0,1\}^m} u(\mathbf{x}, \mathbf{a}),$$

where  $m$ , and  $\text{size}(u(\mathbf{x}, \mathbf{y}))$  are bounded by  $\text{poly}(s, n, d)$ .

The use of interpolation and Valiant's criterion as in [Theorem 5.18](#) does not seem helpful over infinite fields, and it is an open problem to extend these results to all fields.

**? Open Problem (5.3.1)**

Can we show that  $\overline{\text{VP}}_\varepsilon \subseteq \text{VNP}$  over *any* field  $\mathbb{F}$ ?

---

## 6 Newton iteration to Hensel lifting and back

Hensel lifting ([Section 4](#)) and Newton iteration ([Section 5](#)), though different when viewed from afar, are quite similar upon closer inspection.

Let  $f(\mathbf{x}, y)$  be a monic (with respect to  $y$ ) polynomial of degree  $d$  with a root  $g(\mathbf{x})$  (with respect to  $y$ ) of multiplicity one, i.e.  $f(\mathbf{x}, g(\mathbf{x})) = 0$ , but  $\partial_y f(\mathbf{x}, g(\mathbf{x})) \neq 0$ . When we wanted to find iteratively better approximations to  $g$ , we started with an approximation  $g_0$  that was consistent with  $g$  modulo the ideal  $\langle \mathbf{x} \rangle$  and used Newton's method to obtain a better approximation  $g_1$  that was consistent with  $g$  modulo  $\langle \mathbf{x} \rangle^2$ . The polynomial  $g$  being a root of  $f$  implies that  $f$  can be factored as  $f(\mathbf{x}, y) = (y - g(\mathbf{x}))h(\mathbf{x}, y)$  for some polynomial  $h$ . So, we can instead view our Newton iteration process as a Hensel lift of the factorisation  $f = (y - g_0)h_0 \pmod{\langle \mathbf{x} \rangle}$  to the factorisation  $f = (y - g_1)h_1 \pmod{\langle \mathbf{x} \rangle^2}$ . Since  $g$  was a root of multiplicity one, the factors  $(y - g(\mathbf{x}))$  and  $h(\mathbf{x}, y)$  were coprime. In this sense, Newton iteration is a special case of Hensel lifting.

In the other direction, we can also view Hensel lifting as a special case of (multivariate) Newton iteration. Suppose that we have the factorisation  $f(\mathbf{x}, y) = g(\mathbf{x}, y)h(\mathbf{x}, y)$ . Let us write  $f = \sum_{i=0}^d f_i y^i$ , and similarly  $g = \sum_{i=0}^{d_1} g_i y^i$ ,  $h = \sum_{i=0}^{d_2} h_i y^i$  where  $f_i, g_i, h_i \in \mathbb{F}[\mathbf{x}]$  for all  $i$ , and  $d_1 + d_2 = d$ . Noting that  $f_d = 1$ , we can view this factorisation as a system of equations:

$$\begin{aligned} f_0 &= g_0 h_0 \\ f_1 &= g_0 h_1 + g_1 h_0 \\ &\vdots \\ f_{d-1} &= g_{d_1-1} h_{d_2} + g_{d_1} h_{d_2-1} = g_{d_1-1} h_{d_2-1}. \end{aligned}$$

For a new set of  $d$  variables  $\{\mathbf{u}, \mathbf{w}\} = \{u_0, \dots, u_{d_1-1}, w_0, \dots, w_{d_2-1}\}$ , consider the  $d$  equations

$$\begin{aligned}
\varphi_0 &:= f_0 - u_0 w_0 \\
\varphi_1 &:= f_1 - u_0 w_1 - u_1 w_0 \\
&\vdots \\
\varphi_{d-1} &:= f_{d-1} - u_{d_1-1} w_{d_2-1}.
\end{aligned}$$

The coefficients  $(g_0, \dots, g_{d_1-1}, h_0, \dots, h_{d_2-1})$  are a common zero of the equations  $\varphi = (\varphi_0, \dots, \varphi_{d-1})$ . Given an approximate root  $(\mathbf{a}, \mathbf{b}) \in \mathbb{F}[\mathbf{x}]^d$  modulo the ideal  $\langle \mathbf{x} \rangle$ , a *multivariate generalisation* of Newton iteration [GG13] gives a better approximation

$$(\mathbf{a}^*, \mathbf{b}^*) = (\mathbf{a}, \mathbf{b}) - J^{-1} \varphi(\mathbf{a}, \mathbf{b})$$

modulo the ideal  $\langle \mathbf{x} \rangle^2$ , where the matrix of polynomials  $J = (\partial_t \varphi_i)_{0 \leq i \leq d-1, t \in \{u, w\}}$  is the *Jacobian* of  $\varphi$ .<sup>5</sup> We can now view a Hensel lift of the factorisation  $f = g^{(0)} h^{(0)} \bmod \langle \mathbf{x} \rangle$  to  $f = g^{(1)} h^{(1)} \bmod \langle \mathbf{x} \rangle^2$  as a *multivariate* Newton iteration step of improving an approximate root to the above set of equations modulo  $\langle \mathbf{x} \rangle$ , to modulo  $\langle \mathbf{x} \rangle^2$ . The Jacobian and the Sylvester Matrix (Definition 4.3) are the same up to permutation of rows and columns [CKS19a, Lemma 3.2]. Hence, the invertibility of the Jacobian is equivalent to the coprimality of  $g$  and  $h$  [GG13, Exercise 15.21]. Thus, Newton iteration generalises Hensel lifting.

More generally, the two methods can be derived from one another over valuation rings. This is what von zur Gathen [Gat84] has to say:

*“Note that while Yun [Yun76] motivates the Hensel method as a special form of the Newton method (“Hensel meets Newton”), here the Newton method is a corollary of the Hensel method (“Hensel beats Newton”).”*

This folklore connection between Hensel lifting and Newton iteration has also appeared explicitly in a few places [Zip81; Gat84; Art22]. Chou, Kumar, and Solomon [CKS19a] used this connection to give a simplified proof of Kaltofen’s VP closure result using the multivariate Newton iteration. For more restricted models, the method that is convenient while factoring depends on the efficiency of the factoring primitive that the model affords.

---

<sup>5</sup> If we want to approximate a root with respect to a single variable  $y$  of a single equation  $\varphi(\mathbf{x}, y)$ , the Jacobian is a  $(1 \times 1)$  matrix  $[\partial_y \varphi]$ , and this is just the usual Newton iteration.

## 7 Factoring via Lagrange Inversion

Given a power series  $z = f(y)$  it is natural to ask if the inverse function  $f^{-1}(z)$  has a power series as well. Lagrange was the first to attempt this *reversion of series* in the late 18th century to study planetary motion, which was generalised by Hans Heinrich Bürmann.

**Theorem 7.1 (Lagrange inversion formula)** *Let  $\mathbb{F}$  be a field of characteristic zero. Let  $f(y) \in \mathbb{F}[[y]]$  be a formal power series with a simple root at  $y = 0$ , meaning  $f(0) = 0$  and  $\partial_y f(0) \neq 0$ . Then, the unique compositional inverse  $g(z) \in \mathbb{F}[[z]]$  satisfying  $f(g(z)) = z$  is given by*

$$g(z) = \sum_{n \geq 1} c_n z^n, \quad (8)$$

where

$$c_n = \frac{1}{n} \operatorname{coef}_{y^{n-1}} \left( \frac{y}{f(y)} \right)^n.$$

There are many proofs of the Lagrange inversion formula (see e.g., [Sta24, Section 5.4]). We sketch the key insights of a straightforward derivation of the above theorem using complex analysis and contour integration [WW21].

*Proof Sketch by Complex Analysis.* Let  $C$  be a small enough circle around 0; let  $u$  refer to a point in  $C$ . Without loss of generality,  $f$  is injective on  $C$ . Let  $\zeta := f(u)$  refer to a point in  $f(C)$ . It follows:  $d\zeta = f'(u) du$  and  $u = f^{-1}(\zeta) = g(\zeta)$ .

$$g(z) = \frac{1}{2\pi i} \oint_{f(C)} \frac{g(\zeta)}{\zeta - z} d\zeta = \frac{1}{2\pi i} \oint_C \frac{u f'(u)}{f(u) - z} du.$$

We can write this as a formal power series in  $z$ ,

$$g(z) = \sum_{n \geq 0} z^n \left( \frac{1}{2\pi i} \oint_C \frac{u f'(u)}{f(u)^{n+1}} du \right).$$

Thus,

$$c_n = \frac{1}{2\pi i} \oint_C \frac{u f'(u)}{f(u)^{n+1}} du.$$

By *integration by parts*, it simplifies to

$$c_n = \frac{1}{2\pi i} \oint_C \frac{1}{n f(u)^n} du.$$

By the *Residue Theorem*, we get

$$c_n = \frac{1}{n} \operatorname{coef}_{y^{-1}} \left( \frac{1}{f(y)^n} \right) = \frac{1}{n} \operatorname{coef}_{y^{n-1}} \left( \frac{y}{f(y)} \right)^n.$$

□

The expression for the coefficient above is purely algebraic and holds over any field of characteristic zero, and not just the complex numbers. To that end, we now discuss a more elementary proof by Stanley [Sta24, First proof of Theorem 5.4.2].

*Proof Sketch by Algebra.* Let the compositional inverse be written as  $g(z) = \sum_{m \geq 1} c_m \cdot z^m$ . Since  $g(z)$  is the compositional inverse of  $f(y)$ , substituting  $z = f(y)$  gives  $g(f(y)) = y$ . Substituting this into the series for  $g$  yields:

$$y = \sum_{m \geq 1} c_m \cdot f(y)^m$$

Differentiating both sides with respect to  $y$  gives:

$$1 = \sum_{m \geq 1} m \cdot c_m \cdot f(y)^{m-1} \cdot \partial_y f(y)$$

For any  $n \in \mathbb{N}$ , to isolate the  $n$ -th coefficient, we multiply both sides by  $(y/f(y))^n$ :

$$\left( \frac{y}{f(y)} \right)^n = \sum_{m \geq 1} m \cdot c_m \cdot y^n \cdot f(y)^{m-n-1} \cdot \partial_y f(y).$$

We can split the sum on the right-hand side into two parts: the terms where  $m \neq n$ , and the single term where  $m = n$ . Notice that when  $m \neq n$ , the expression  $f(y)^{m-n-1} \partial_y f(y)$  is the derivative of  $\frac{1}{m-n} f(y)^{m-n}$ .

$$\left( \frac{y}{f(y)} \right)^n = \left( \sum_{m \neq n} \frac{m \cdot c_m}{m-n} \cdot y^n \cdot \partial_y (f(y)^{m-n}) \right) + n \cdot c_n \cdot y^n \cdot \frac{\partial_y f(y)}{f(y)}$$

Next, we take the coefficient of  $y^{n-1}$  on both sides. A fundamental property of formal Laurent series is that the derivative of any series never contains a  $y^{-1}$  term. Therefore, the entire sum of derivatives vanishes to give:

$$\operatorname{coef}_{y^{n-1}} \left( \frac{y}{f(y)} \right)^n = n \cdot c_n \cdot \operatorname{coef}_{y^{-1}} \frac{\partial_y f(y)}{f(y)}$$

To evaluate the remaining residue on the right, recall that  $f(y)$  has a simple root at  $y = 0$ . We can write  $f(y) = y \cdot h(y)$  where  $h(0) \neq 0$ . Using the product rule:

$$\frac{\partial_y f(y)}{f(y)} = \frac{h(y) + y \partial_y h(y)}{yh(y)} = \frac{1}{y} + \frac{\partial_y h(y)}{h(y)}$$

Since  $h(0) \neq 0$ , the term  $\frac{\partial_y h(y)}{h(y)}$  is a standard power series with no negative powers. Thus, the coefficient of  $y^{-1}$  is exactly 1. Therefore,

$$n \cdot c_n = \text{coef}_{y^{n-1}} \left( \frac{y}{f(y)} \right)^n$$

□

Lagrange Inversion can be used as the basis of a *unifying* factoring technique for all the models of computation we have seen previously. As an important example, we consider algebraic *formulas*, a model slightly weaker than circuits (Section 4.4) and branching programs (Section 4.5), but nonetheless extremely natural.

## 7.1 Small algebraic formulas & Shallow circuits

The class VF is a natural restriction of VP where the graph underlying the circuit is a tree (Figure 2). Since a formula of size  $s$  can never compute a polynomial of degree greater than  $O(s)$ , we do not need to impose any additional degree restriction.

**Definition 7.2 (VF)** A family of polynomials  $f = (f_n)$  is said to be in VF over the field  $\mathbb{F}$  if the minimum size of the *formula* (over  $\mathbb{F}$ ) computing  $f_n$  is bounded by a polynomial function in the number of variables  $n$ .

*Remark 7.3* It is not hard to see that  $\text{VF} \subseteq \text{VBP} \subseteq \text{VP}$ . None of the containments are known to be strict.

The factors obtained via monic Hensel lifting (Section 4.4) turn out to require polynomial divisions with remainder (essentially circuits), even if we start with a formula. We do not know how to convert circuits to formulas without a quasi-polynomial blow up in the formula size. The best result we knew in this direction, until recently, was the factor closure of quasi-polynomial sized formulas, first shown by Dutta, Saxena, and Sinhababu [DSS22, Theorem 3]. We state here the version from the work of Chou, Kumar, and Solomon [CKS19b] which gives an alternative proof of the same result using the ideas in Section 5.1.

**Theorem 7.4 ([CKS19b, Theorem 9.1])** *Let  $f(\mathbf{x})$  be a polynomial of degree  $d$  in  $n$  variables which can be computed by a formula of size  $s$ . If  $g(\mathbf{x})$  is a degree- $r$  factor of  $f$ , then it can be computed by a formula of size  $\text{poly}(s, n, d, r^{O(\log r)})$ .*

To avoid divisions, a reasonable attempt would be to extend the techniques of Sinhababu and Thierauf for factoring branching programs (Section 4.5). All but the last step of solving a linear system (equivalently, computing a determinant) can indeed be done using small formulas. Unfortunately, we do not know of any poly( $n$ )-sized formulas for the  $n \times n$  symbolic determinant. Thus the question of factor closure for formulas remained open, till a breakthrough appeared in 2025 [Bha+25a]. This paper eliminated the need for iterations, unlike Newton or Hensel. It is based on a fundamental, analytic result on the power series root due to Furstenberg [Fur67, Prop.2]. As the authors also note [Bha+25a, Section 1.2.1], we can alternatively use Lagrange inversion (Theorem 7.1), whose surprising connection to root-finding we now present.

**Theorem 7.5 ([Bha+25a, Corollary 1.6])** *Let  $\mathbb{F}$  be a characteristic zero field and  $F(x, y) \in \mathbb{F}[x, y]$ . We are interested in a simple root  $\varphi(x) \in \mathbb{F}[[x]]$  such that  $F(x, y) = (y - \varphi) \cdot Q(x, y)$  with  $\partial_y F(0, 0) = 1$  and  $\varphi(0) = 0$ . Then,*

$$\varphi(x) = \sum_{m \geq 1} \text{coef}_{y, m-1} \left[ \frac{(y - F(x, y))^m}{m} \right]. \quad (9)$$

*Proof Sketch.* The proof relies on identifying the root  $\varphi(x)$  as a specific evaluation of a formal power series derived via the Lagrange inversion formula. Define the auxiliary function:

$$z = \frac{y}{y - F(x, y)} =: f(y).$$

Rearranging this, we get  $y = z \cdot (y - F(x, y))$ . Observe that if we substitute  $z = 1$ , we obtain  $y = y - F(x, y)$ , which implies  $F(x, y) = 0$ . Thus, evaluating the inverse function  $y(z) := f^{-1}(z)$  at  $z = 1$  should yield the well-defined root  $\varphi(x) \in \mathbb{F}[[x]]$ .

We apply Theorem 7.1 to express  $y(z)$  as a power series in  $z$ . Note that  $y/f(y) = y - F(x, y)$ . Substituting this into the inversion formula (Equation (8)), we obtain:

$$\begin{aligned} y(z) &= \sum_{m \geq 1} \frac{z^m}{m} \cdot \text{coef}_{y, m-1} \left[ \left( \frac{y}{f(y)} \right)^m \right] \\ &= \sum_{m \geq 1} \frac{z^m}{m} \cdot \text{coef}_{y, m-1} [(y - F(x, y))^m] \in \mathbb{F}[x][[z]]. \end{aligned} \quad (10)$$

Now we try to ‘pass’ from the power series convergence notion in  $\mathbb{F}[x][[z]]$  to that in  $\mathbb{F}[[x]]$ . To recover  $\varphi(x)$ , we must justify setting  $z = 1$  in Equation (10). This operation is valid in the ring  $\mathbb{F}[[x]]$  if the series converges.

Recall the hypothesis  $\partial_y F(0, 0) = 1$  and  $F(0, 0) = 0$ . This implies that the linear term of  $F(x, y)$  in  $y$  is exactly  $y$ . Consequently, the difference  $y - F(x, y)$  contains no

constant term and no linear term in  $y$  (when  $x = 0$ ). We can therefore write:

$$y - F(x, y) \in \langle x, y^2 \rangle.$$

Now, consider the term  $(y - F(x, y))^m$ . It is a sum of monomials of the form  $x^a (y^2)^b$  where  $a + b = m$ . We are interested in the coefficient of  $y^{m-1}$ . For a term to contribute to  $y^{m-1}$ , we must have  $2b \leq m - 1$ . Therefore, the  $x$ -degree  $a$  must satisfy  $a \geq (m + 1)/2$ .

Since the degree of  $x$  grows with  $m$ , the series in Equation (10) converges well in  $\mathbb{F}[[x]]$  when  $z = 1$ . Setting  $z = 1$  yields precisely the formula in Equation (9), and by construction, this value satisfies  $F(x, \varphi(x)) = 0$ .  $\square$

With familiarity of the factoring techniques discussed in Section 5 and the structural results from Section 2.1, we can now prove Theorem 4.9 in a single step rather than using iterative techniques. In addition to providing a simpler proof of known results, this *magical* identity, in fact, resolves long-standing factor-closure questions. In what follows, we prove that bounded depth circuits are closed under factoring. We will first sketch the proof for algebraically closed fields of characteristic zero, where the argument is relatively straightforward, and then discuss how to extend it to arbitrary fields.

**Theorem 7.6 ([Bha+25a, Theorem 4.3]) (Shallow circuit)** *Let  $\mathbb{F}$  be an algebraically closed field of characteristic zero, and  $f \in \mathbb{F}[\mathbf{x}, y]$  be an  $(n + 1)$ -variate, degree  $d$  polynomial computable by a circuit of size  $s$  and depth  $\Delta$ . Then, any factor  $g(\mathbf{x}, y)$  of  $f$  is computable by a circuit of size  $\text{poly}(s, n, d)$  and depth  $O(\Delta)$ .*

*Proof Sketch.* The main structure of the proof follows the template from Section 5. We begin by preprocessing  $f$  so that all the roots  $\varphi_i \in \mathbb{F}[[\mathbf{x}]]$  in the factorisation  $f(\mathbf{x}, y) = \prod_{i=1}^d (y - \varphi_i(\mathbf{x}))$  are simple. We can assume  $g$  shares the first  $r$  roots with  $f$ :

$$g(\mathbf{x}, y) = \prod_{i=1}^r (y - \varphi_i(\mathbf{x})). \quad (11)$$

In particular,  $g(\mathbf{0}, y) = \prod_{i=1}^r (y - \varphi_i(\mathbf{0}))$  where  $\mu_i := \varphi_i(\mathbf{0}) \in \mathbb{F}$  since  $\mathbb{F}$  is closed. In order to lift  $\mu_i$  to  $\varphi_i$ , we would like to use Theorem 7.5. By scaling  $f$ , we can ensure that  $\partial_y f(\mathbf{0}, 0) = 1$ . Notice that for any  $\mu_i$ ,  $\tilde{\varphi}_i := \varphi_i - \mu_i$  is a simple root of  $\tilde{f}_i(\mathbf{x}, y) := f(\mathbf{x}, y + \mu_i)$  with constant term  $\tilde{\varphi}_i(\mathbf{0}) = 0$ .

We can now apply Theorem 7.5 to get a power series expression for each of the  $\tilde{\varphi}_i$ 's and hence, also the  $\varphi_i$ 's. From the discussion in the latter part of the proof of Theorem 7.5, it can be seen that terms with index  $m > 2r$  contribute only to degrees strictly greater than  $r$  in the  $\mathbf{x}$  variables. Hence, we can truncate  $\varphi_i$ 's to index at most

2r. Given that  $f$  (and thus,  $\tilde{f}$ ) has a size  $s$  circuit of depth  $\Delta$ , all these operations can be performed in size  $\text{poly}(s, n, d)$  and depth  $\Delta + O(1)$  with the help of interpolation (Lemma 2.2). Plugging in these circuits for the roots into Equation (11) and extracting the components of degree at most  $r$  gives us the result.  $\square$

As seen earlier (Remark 5.3), if the field  $\mathbb{F}$  is not algebraically closed, the roots  $\{\mu_i\}_{i \in [r]}$  of  $g(\mathbf{0}, y)$  come from a very high degree splitting field  $\mathbb{K} \supseteq \mathbb{F}$ . Notice that the circuit for the power series roots  $\{\varphi_i\}$  (and hence, also  $g$ ) just uses constants from  $\mathbb{F}$  if we treat the roots  $\{\mu_i\}$  as input variables. Moreover, the coefficient of  $y^j$  in Equation (11) is the  $j$ -th elementary symmetric polynomial  $e_j(\varphi_1, \dots, \varphi_r) = \sum_{S \subseteq [r], |S|=j} \prod_{i \in S} \varphi_i$  in the power series roots. One can now use a result by Andrews and Wigderson [AW24, Theorem I.8] on efficiently computing elementary symmetric polynomials to obtain a circuit for these coefficients (and hence  $g$ ) over  $\mathbb{F}$ . This is what the authors of [Bha+25a] do. We describe a more direct approach based on a related recent result by the same authors [Bha+25b].

We view  $g(\mathbf{x}, y)$  (which is truly a polynomial over  $\mathbb{F}$ ) as a polynomial (over  $\mathbb{F}[\mathbf{x}, y]$ ) in the *root variables*  $\boldsymbol{\mu} := (\mu_1, \dots, \mu_r)$ . Note that  $g$  is *symmetric* with respect to the roots  $\{\mu_i\}$ : any permutation of the roots merely permutes the factors of the product in Equation (11). The fundamental theorem of symmetric polynomials (e.g., see [Lan02, Theorem IV.6.1] and [BC17]) states that there is a polynomial  $G$  (over  $\mathbb{F}[\mathbf{x}, y]$ ) such that  $g(\boldsymbol{\mu}) = G(e_1(\boldsymbol{\mu}), \dots, e_r(\boldsymbol{\mu}))$ , where  $e_j(\boldsymbol{\mu})$  is the  $j$ -th elementary symmetric polynomial. Crucially, the  $e_j(\boldsymbol{\mu})$ 's are the (signed) coefficients of  $g(\mathbf{0}, y)$ . Therefore, we can instead write  $g$  as a polynomial in these coefficients, which are in the base field  $\mathbb{F}$ . We know from the proof of Theorem 7.6 that  $g$  has a circuit of size  $\text{poly}(s, n, d)$  and depth  $O(\Delta)$ , and as it turns out, so does  $G$ . This is the bounded-depth analogue of a similar result by Bläser and Jindal [BJ19, Theorem 4] for general circuits.

**Theorem 7.7 ([Bha+25b, Theorem 1.2])** *Let  $g \in \mathbb{F}[\mathbf{x}, y][\boldsymbol{\mu}]$  be a polynomial of degree  $d$  which is symmetric in  $\boldsymbol{\mu}$  variables and computable by a circuit of size  $s$  and depth  $\Delta$ . Let  $G \in \mathbb{F}[\mathbf{x}, y][\mathbf{z}]$  be the unique degree  $d$  polynomial such that  $f = G(e_1(\boldsymbol{\mu}), \dots, e_r(\boldsymbol{\mu}))$ , where  $e_1, \dots, e_r$  are the elementary symmetric polynomials. Then  $G$  is computable by a circuit of size  $\text{poly}(s, r, d)$  and depth  $\Delta + O(1)$ .*

*Remark 7.8* We note that [Bha+25b, Theorem 1.2] assumes the symmetric polynomial is an element of  $\mathbb{F}[\boldsymbol{\mu}]$ . The slight generalisation presented here follows directly from their proof and is more convenient for our application.

The above theorem shows that  $G$  also has a circuit of size  $\text{poly}(s, n, d)$  and depth  $\Delta + O(1)$  over  $\mathbb{F}[x, y]$ . All that remains is to undo the preprocessing steps to obtain the factors of  $f$  over the field  $\mathbb{F}$ .

*Remark 7.9* There is a version of Lagrange’s inversion formula for fields of positive characteristic as well (see e.g., Theorem 1 and Remark 2 in [Hu16]). If the field  $\mathbb{F}$  is polynomially large in  $snd$ , Theorem 7.7 also holds. But due to inseparability issues (Remark 5.2), if the field  $\mathbb{F}$  is of characteristic  $p$ , then the closure only holds up to  $p$ -th powers [Bha+25b, Theorem 5.1].

Inspecting the proof of Theorem 7.6, one can see that it also works for more general models like formulas, branching programs, and circuits. In particular, we have the closure of VF under factorisation.

**Theorem 7.10 ([Bha+25a, Theorem 1.1])** *Let  $\mathbb{F}$  be a field of characteristic zero, and  $f$  be an  $n$ -variate, degree  $d$  polynomial over  $\mathbb{F}$  computed by an algebraic formula of size  $s$ . Then, any factor  $g$  of  $f$  can be computed by a formula of size  $\text{poly}(s, n, d)$  over  $\mathbb{F}$ .*

### ? Open Problem (7.1.1)

Is the class VF closed under taking factors over fields of positive characteristic?

---

## 8 Factoring via Convex Geometry

We will now study a model that is *not* closed under factoring in general, but has been a natural and elegant crucible of factoring ideas for decades.

### 8.1 Sparse polynomials

*Sparse polynomials* are perhaps the simplest class that one can study. The sparsity of a polynomial  $f$ , denoted by  $\|f\|$ , is the number of monomials in it. Alternatively, one can think of the sparsity as the size of a  $\Sigma\Pi$  circuit computing  $f$ . The work of von zur Gathen and Kaltofen [GK85], which initiated the study of factoring sparse polynomials, gave a *randomised* algorithm that outputs a factor in time polynomial in the sparsity of

the factor. Naturally, it makes sense to ask if factors of sparse polynomials are sparse. Unfortunately, as they showed, this is *not* true in general. The sparsity of a factor can be superpolynomial in the sparsity of the original polynomial.

*Example 8.1 ([GK85])* The polynomial  $f = \prod_{i=1}^n (x_i^d - 1)$  has sparsity  $\|f\| = 2^n$ , but one of its factors  $g = \prod_{i=1}^n (1 + x_i + \dots + x_i^{d-1})$  has sparsity  $\|g\| = d^n = \|f\|^{\log d}$ .

If the individual degree is comparable to the number of variables, then the blow-up can be exponential.

*Example 8.2 ([BSV20])* Over the field  $\mathbb{F}_p$ , the polynomial  $f = \sum_{i=1}^n x_i^p$  has sparsity  $\|f\| = n$ , but the factor  $g = (\sum_{i=1}^n x_i)^d$ , for  $0 < d < p$  has sparsity  $\|g\| = \binom{n+d-1}{d} \approx \|f\|^d$ .

The above examples are essentially the “worst” that we have, and it is natural to wonder whether they are indeed the worst *possible*. One might then still hope that factors of a polynomial with *bounded* individual degree are sparse.

### ? Open Problem (8.1.1)

Let  $f = g \cdot h$  be a polynomial with bounded (constant) individual degree. Then, does the following hold:  $\|g\| = \text{poly}(\|f\|)$ ?

Tools from convex geometry have been useful in recent progress in studying this question. For a polynomial  $f = \sum_{\mathbf{e}} c_{\mathbf{e}} \mathbf{x}^{\mathbf{e}}$ , consider the set of exponent vectors in its support,

$$\text{sup}(f) := \{\mathbf{e} : c_{\mathbf{e}} \neq 0\} \subseteq \mathbb{Z}^n.$$

The convex hull of the points in the support denoted  $\text{Conv}(\text{sup}(f))$  is called the *Newton Polytope* corresponding to  $f$ :

$$P_f := \text{Conv}(\text{sup}(f)) = \left\{ \sum_{\mathbf{e}} \alpha_{\mathbf{e}} \mathbf{e} : \sum_{\mathbf{e}} \alpha_{\mathbf{e}} = 1, 0 \leq \alpha_{\mathbf{e}} \in \mathbb{R}, \mathbf{e} \in \text{sup}(f) \right\} \subseteq \mathbb{R}^n.$$

A *vertex* of  $P_f$  is a point in the polytope that *cannot* be written as a *non-trivial* convex combination (i.e.,  $\alpha_{\mathbf{e}} < 1$  for all  $\mathbf{e}$ ) of points in  $P_f$ . We will denote the vertex set of a polytope  $P$  by  $V(P)$ . Ostrowski [Ost99] observed that for polynomials  $g$  and  $h$ ,

$$P_{gh} = P_g + P_h,$$

where the addition is a *Minkowski sum*, consisting of all points  $\mathbf{a} + \mathbf{b}$ , such that  $\mathbf{a} \in P_g$  and  $\mathbf{b} \in P_h$ . It can be shown that

$$\max\{|V(P_g)|, |V(P_h)|\} \leq |V(P_g + P_h)| \leq |V(P_g)| \cdot |V(P_h)|. \quad (12)$$

The upper bound on  $|V(P_g + P_h)|$  is straightforward. For the lower bound, see [BSV20, Proposition 3.2] and Appendix K in the book of Schinzel [Sch00]. This suggests a way to prove sparsity bounds. For a polynomial  $f = gh$ , if we can show  $g$  is ‘dense’ by showing a lower bound on  $|V(P_g)|$ , then by the above inequality, we also get a lower bound on  $|V(P_g + P_h)|$ , and in turn the sparsity of  $f$ . In other words, if  $f$  is sparse, so is  $g$ .

Consider the case when  $f = gh$  and  $g$  is *multilinear*, i.e., the individual degree of  $g$  is at most 1. A moment’s thought shows that every monomial of  $g$  corresponds to a vertex, i.e.,  $\text{sup}(g) = V(P_g)$ . Combining this with the lower bound in Equation (12), we get

$$\|f\| \geq |V(P_f)| = |V(P_g + P_h)| \geq |V(P_g)| = \|g\|. \quad (13)$$

Therefore, if  $f = gh$  is a multilinear polynomial, then  $g$  is also multilinear, and we get  $\|g\| \leq \|f\|$  from above, showing that sparse multilinear polynomials are closed under factoring. Shpilka and Volkovich [SV10] gave an efficient *deterministic* algorithm for factoring sparse multilinear polynomials. Volkovich [Vol15] used the sparsity bound in Equation (13) to first extend their result to sparse polynomials that split into multilinear factors. In a later work [Vol17], he proved that factors of *multiquadratic* polynomials are also sparse, and gave an efficient deterministic algorithm to factor such polynomials.

However, in general, the size of the vertex set  $|V(P_g)|$  could be much smaller than the sparsity  $\|g\|$  of the polynomial. The polynomial  $g = (\sum_{i=1}^n x_i)^d$  in Example 8.2 has only  $n$  vertices in its Newton Polytope (corresponding to  $x_1^d, \dots, x_n^d$ ), but has  $O(n^d)$  monomials. If the individual degree of a polynomial is bounded, Bhargava, Saraf, and Volkovich [BSV20] showed that the vertex set is not too small either.

**Theorem 8.3** *Let  $\mathbb{F}$  be an arbitrary field, and let  $g$  be a polynomial over  $\mathbb{F}$  in  $n$  variables of individual degree  $d$ . Then,*

$$|V(P_g)| \geq \|g\|^{1/O(d^2 \log n)}.$$

The bound is tight with regard to dependence on  $n$  (see [BSV20, Claim 4.4]). As an immediate corollary of the above theorem, we deduce a sparsity bound for the factors of polynomials with bounded individual degrees.

**Corollary 8.4** *Let  $f = gh$  be a polynomial in  $n$  variables of individual degree  $d$  and sparsity  $\|f\| = s$ . Then, the sparsity of  $g$  is  $\|g\| \leq s^{O(d^2 \log n)}$ .*

*Proof.* Note that the individual degree of  $g$  is bounded by  $d$  as well. Using Equation (12) and Theorem 8.3, we get

$$\|f\| \geq |V(P_f)| = |V(P_g + P_h)| \geq |V(P_g)| \geq \|g\|^{1/O(d^2 \log n)},$$

and thus, the required bound.  $\square$

Although the sparsity bound of Corollary 8.4 is worse than Example 8.2, it can be viewed as affirmative evidence towards Problem 8.1.1. Using the above sparsity bound, one can also obtain a deterministic *quasi-polynomial* time algorithm for sparse polynomials of bounded individual degree [BSV20, Theorem 2] (also see [HG23]). We now give a brief overview of the ideas in the proof of Theorem 8.3.

*Proof Sketch.* [Theorem 8.3] The classical Carathéodory's theorem of convex geometry states that any point  $\mathbf{x} \in \text{Conv}(X)$  in the convex hull of a set of points  $X \subseteq \mathbb{R}^n$  can be written as a convex combination of at most  $n + 1$  points in  $X$ . In fact, the convex combination only needs to use the vertices of  $\text{Conv}(X)$ . We need much fewer than  $n + 1$  points if we are okay with *approximating*  $\mathbf{x}$  by a convex combination.

In the approximate version of Carathéodory's theorem [BSV20, Theorem 3.6], we have a set  $X$  with bounded  $\ell_\infty$  norm (i.e.,  $\max_{\mathbf{y} \in X} \|\mathbf{y}\|_\infty \leq 1$ ). Given an  $\varepsilon > 0$ , any point  $\mathbf{x} \in \text{Conv}(X)$  that lies in the convex hull of  $X$  can be  $\varepsilon$ -approximated by a point  $\mathbf{x}'$  (i.e.,  $\|\mathbf{x} - \mathbf{x}'\|_\infty \leq \varepsilon$ ), and moreover,  $\mathbf{x}' = \sum_{\mathbf{y}} \alpha_{\mathbf{y}} \mathbf{y}$  is a *uniform* convex combination ( $\alpha_{\mathbf{y}} = 1/k$  whenever  $\alpha_{\mathbf{y}} \neq 0$ ) of at most  $k = O(\log n / \varepsilon^2)$  vertex points from  $V(\text{Conv}(X))$ . The bound on  $k$  follows from sampling points  $\mathbf{y} \in V(\text{Conv}(X))$  with probability  $\alpha_{\mathbf{y}}$  where  $\mathbf{x} = \sum_{\mathbf{y}} \alpha_{\mathbf{y}} \mathbf{y}$ , and then using Chernoff's inequality to show that sampling  $O(\log n / \varepsilon^2)$  points and taking their uniform convex combination  $\varepsilon$ -approximates  $\mathbf{x}$ . For us,  $X$  will be the scaled-down version of  $\text{sup}(g)$ , i.e.,

$$X = \{1/d \cdot \mathbf{e} : \mathbf{e} \in \text{sup}(g)\}.$$

Note that  $|X| = \|g\|$  and  $|V(\text{Conv}(X))| = |V(P_g)|$ . Moreover, for distinct  $\mathbf{x} \neq \mathbf{y} \in X$ , we have  $\|\mathbf{x} - \mathbf{y}\|_\infty \geq 1/d$ . Hence, if we choose  $\varepsilon$  to be something slightly smaller than  $1/2d$ , by the triangle inequality, a point  $\mathbf{x}' \in \text{Conv}(X)$  can  $\varepsilon$ -approximate only one of  $\mathbf{x}$  or  $\mathbf{y}$ , not both. So, every point  $\mathbf{x} \in X$  is guaranteed a distinct point  $\mathbf{x}' \in \text{Conv}(X)$  that approximates it *and* is a uniform convex combination of at most  $k = O(\log n / \varepsilon^2)$  points of  $V(\text{Conv}(X))$ . The number of such  $\mathbf{x}'$  is at most  $|V(\text{Conv}(X))|^k$ . Hence,

$|X| \leq |V(\text{Conv}(X))|^k$ , and we can choose  $\varepsilon$  sufficiently close to (but smaller than)  $1/2d$  to get  $k = O(d^2 \log n)$ , and thus our desired bound.  $\square$

Note that the polynomials in [Example 8.1](#) and [Example 8.2](#) with dense factors were *symmetric*. Bisht and Saxena [[BS25](#)] answered [Problem 8.1.1](#) in the affirmative in this case.

**Theorem 8.5** ([\[BS25, Lemma 4.12\]](#)) *Let  $f = gh$  be a factorisation with  $g$  being a symmetric polynomial in  $n$  variables of individual degree  $d$  and sparsity  $\|f\| = s$ . Then,  $\|g\| \leq s^{O(d^2 \log d)}$ .*

Furthermore, they used the above result to give a deterministic polynomial-time factoring algorithm [[BS25, Theorem 1.2](#)] in this case. For general sparse polynomials, the aforementioned result of Bhattacharjee, Kumar, Ramanathan, Saptharishi, and Saraf [[Bha+25c](#)] gives a subexponential deterministic algorithm as a special case when the depth is two.

## Acknowledgements

The authors thank the conducive atmosphere of the *Workshop on Algebraic Complexity Theory 2023* in University of Warwick to initiate new ideas; and the *Workshop on Recent Trends in Computer Algebra 2023* in Institut Henri Poincaré, Paris for giving N.S. the opportunity to chalk out the details of [[Sax23](#)]. C.S.B. and N.S. thank the organisers and the participants in Ruhr University Bochum for the interesting discussions in the *8th Workshop on Algebraic Complexity Theory (WACT 2025)*. We also thank Amit Sinhababu and Varun Ramanathan for insightful discussions on Newton iteration and for pointing us to some relevant references.

C.S.B. thanks the European Union (ERC, CountHom, 101077083) for funding part of this work.

N.S. thanks the DST-SERB agencies for funding support through the Core Research Grant (CRG/2020/000045) and the J.C. Bose National Fellowship (JCB/2022/57), as well as the N. Rama Rao Chair (2019–) of the Department of CSE, IIT Kanpur.

P.D. thanks the *Independent Research Fund Denmark* for funding support (FLows 10.46540/3103-00116B), and also acknowledges the support of Basic Algorithms Research Copenhagen (BARC) through the Villum Investigator Grant 54451.

We thank the anonymous reviewers for providing useful feedback that greatly helped in improving the presentation and correctness of the survey.

## References

- [AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity*. Cambridge University Press, Cambridge, 2009.
- [AF22] Robert Andrews and Michael A. Forbes. “Ideals, Determinants, and Straightening: Proving and Using”. In: *STOC ’22—Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*. ACM, New York, 2022, pp. 389–402.
- [Agr05] Manindra Agrawal. “Proving Lower Bounds via Pseudo-Random Generators”. In: *FSTTCS 2005: Foundations of Software Technology and Theoretical Computer Science*. Vol. 3821. Lecture Notes in Comput. Sci. Springer, Berlin, 2005, pp. 92–105.
- [Agr06] Manindra Agrawal. “Determinant versus Permanent”. In: *International Congress of Mathematicians. Vol. III*. Eur. Math. Soc., Zürich, 2006, pp. 985–997.
- [AGS19] Manindra Agrawal, Sumanta Ghosh, and Nitin Saxena. “Bootstrapping Variables in Algebraic Circuits”. In: *Proc. Natl. Acad. Sci. USA* 116.17 (2019), pp. 8107–8118.
- [And20] Robert Andrews. “Algebraic Hardness versus Randomness in Low Characteristic”. In: *35th Computational Complexity Conference*. Vol. 169. LIPIcs. Leibniz Int. Proc. Inform. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2020, Art. No. 37, 32.
- [Art22] Michael Artin. *Algebraic Geometry—Notes on a Course*. Vol. 222. Graduate Studies in Mathematics. American Mathematical Society, Providence, RI, 2022.
- [AS03] Sanjeev Arora and Madhu Sudan. “Improved Low-Degree Testing and Its Applications”. In: *Combinatorica* 23.3 (2003), pp. 365–426.
- [AV08] Manindra Agrawal and V. Vinay. “Arithmetic Circuits: A Chasm at Depth Four”. In: *49th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. IEEE Computer Society, 2008, pp. 67–75.
- [AW24] Robert Andrews and Avi Wigderson. “Constant-Depth Arithmetic Circuits for Linear Algebra Problems”. In: *2024 IEEE 65th Annual Symposium on Foundations of Computer Science—FOCS 2024*. IEEE Computer Society, 2024, pp. 2367–2386.
- [BC17] Ben Blum-Smith and Samuel Coskey. “The Fundamental Theorem on Symmetric Polynomials: History”. In: *College Mathematics Journal* 48.1 (2017), pp. 18–29. arXiv: 1301.7116 [math.HO].
- [BCS97] Peter Bürgisser, Michael Clausen, and M. Amin Shokrollahi. *Algebraic Complexity Theory*. Vol. 315. Grundlehren Der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]. Springer-Verlag, Berlin, 1997.

- [BDS24] C. S. Bhargav, Sagnik Dutta, and Nitin Saxena. “Improved Lower Bound, and Proof Barrier, for Constant Depth Algebraic Circuits”. In: *ACM Trans. Comput. Theory* 16.4 (2024), 23:1–23:22.
- [Beh+25] Amik Raj Behera, Nutan Limaye, Varun Ramanathan, and Srikanth Srinivasan. “New Bounds for the Ideal Proof System in Positive Characteristic”. In: *52nd International Colloquium on Automata, Languages, and Programming, ICALP 2025, Aarhus, Denmark, July 8-11, 2025*. Vol. 334. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025, 22:1–22:20.
- [Bha+25a] Somnath Bhattacharjee, Mrinal Kumar, Shanthanu Rai, Varun Ramanathan, Ramprasad Saptharishi, and Shubhangi Saraf. *Closure under factorization from a result of Furstenberg*. 2025. Electronic Colloquium on Computational Complexity: TR25-084. To appear in STOC, 2026.
- [Bha+25b] Somnath Bhattacharjee, Mrinal Kumar, Shanthanu Rai, Varun Ramanathan, Ramprasad Saptharishi, and Shubhangi Saraf. *Constant-depth circuits for polynomial GCD over any characteristic*. 2025. Electronic Colloquium on Computational Complexity: TR25-085.
- [Bha+25c] Somnath Bhattacharjee, Mrinal Kumar, Varun Ramanathan, Ramprasad Saptharishi, and Shubhangi Saraf. “Deterministic Factorization of Constant-Depth Algebraic Circuits in Subexponential Time”. In: *66th Annual IEEE Symposium on Foundations of Computer Science—FOCS 2025*. IEEE, 2025, pp. 684–709.
- [BI25] Markus Bläser and Christian Ikenmeyer. “Introduction to Geometric Complexity Theory”. In: *Theory Comput.* Graduate Surveys 10 (2025), p. 166.
- [BIZ18] Karl Bringmann, Christian Ikenmeyer, and Jeroen Zuiddam. “On Algebraic Branching Programs of Small Width”. In: *J. ACM* 65.5 (2018), Art. 32, 29.
- [BJ19] Markus Bläser and Gorav Jindal. “On the Complexity of Symmetric Polynomials”. In: *10th Innovations in Theoretical Computer Science Conference (ITCS 2019)*. Vol. 124. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019, 47:1–47:14.
- [Bog05] Andrej Bogdanov. “Pseudorandom Generators for Low Degree Polynomials”. In: *STOC’05: Proceedings of the 37th Annual ACM Symposium on Theory of Computing*. ACM, New York, 2005, pp. 21–30.
- [Bre00] Richard P. Brent. “Recent Progress and Prospects for Integer Factorisation Algorithms”. In: *Computing and Combinatorics (Sydney, 2000)*. Vol. 1858. Lecture Notes in Comput. Sci. Springer, Berlin, 2000, pp. 3–22.
- [BS25] Pranav Bisht and Nitin Saxena. “Derandomization via Symmetric Polytopes: Poly-time Factorization of Certain Sparse Polynomials”. In: *ACM Trans. Comput. Theory* 17.2 (2025), 12:1–12:20.
- [BSV20] Vishwas Bhargava, Shubhangi Saraf, and Ilya Volkovich. “Deterministic Factorization of Sparse Polynomials with Applications to Algebraic Complexity”. In: *J. ACM* 67.2 (2020), Art. 8, 28.

- [Bür+11] Peter Bürgisser, J. M. Landsberg, Laurent Manivel, and Jerzy Weyman. “An Overview of Mathematical Issues Arising in the Geometric Complexity Theory Approach to VP”. In: *SIAM J. Comput.* 40.4 (2011), pp. 1179–1209.
- [Bür00a] Peter Bürgisser. *Completeness and Reduction in Algebraic Complexity Theory*. Vol. 7. Algorithms and Computation in Mathematics. Springer-Verlag, Berlin, 2000.
- [Bür00b] Peter Bürgisser. “Cook’s versus Valiant’s Hypothesis”. In: *Theoret. Comput. Sci.* 235 (2000), pp. 71–88.
- [Bür04] Peter Bürgisser. “The Complexity of Factors of Multivariate Polynomials”. In: *Found. Comput. Math.* 4.4 (2004), pp. 369–396.
- [Bür20] Peter Bürgisser. “Correction to: The Complexity of Factors of Multivariate Polynomials”. In: *Found. Comput. Math.* 20.6 (2020), pp. 1667–1668.
- [Bür24] Peter Bürgisser. *Completeness Classes in Algebraic Complexity Theory*. 2024. arXiv: [2406.06217](https://arxiv.org/abs/2406.06217) [cs].
- [Bür99] Peter Bürgisser. “On the Structure of Valiant’s Complexity Classes”. In: *Discrete Math. Theor. Comput. Sci.* 3.3 (1999), pp. 73–94.
- [CDS24] Sayak Chakrabarti, Ashish Dwivedi, and Nitin Saxena. “Solving Polynomial Systems over Non-Fields”. In: *J. Symbolic Comput.* 125 (2024), Paper No. 102314, 29.
- [CG00] David G. Cantor and Daniel M. Gordon. “Factoring Polynomials over  $p$ -adic Fields”. In: *Algorithmic Number Theory (Leiden, 2000)*. Vol. 1838. Lecture Notes in Comput. Sci. Springer, Berlin, 2000, pp. 185–208.
- [Chi87] A. L. Chistov. “Efficient Factorization of Polynomials over Local Fields”. In: *Doklady Akademii Nauk SSSR* 293.5 (1987), pp. 1073–1077.
- [CKS19a] Chi-Ning Chou, Mrinal Kumar, and Noam Solomon. *Closure of VP under Taking Factors: A Short and Sweet Proof*. 2019. arXiv: [1903.02366](https://arxiv.org/abs/1903.02366) [cs].
- [CKS19b] Chi-Ning Chou, Mrinal Kumar, and Noam Solomon. “Closure Results for Polynomial Factorization”. In: *Theory Comput.* 15 (2019), Paper No. 13, 34.
- [CKW10] Xi Chen, Neeraj Kayal, and Avi Wigderson. “Partial Derivatives in Arithmetic Complexity and Beyond”. In: *Found. Trends Theor. Comput. Sci.* 6.1-2 (2010), front matter, 1–138.
- [Coo71] Stephen A. Cook. “The Complexity of Theorem-Proving Procedures”. In: *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing (STOC)*. ACM, 1971, pp. 151–158.
- [CR79] Stephen A. Cook and Robert A. Reckhow. “The Relative Efficiency of Propositional Proof Systems”. In: *J. Symbolic Logic* 44.1 (1979), pp. 36–50.
- [DG24] Pranjal Dutta and Sumanta Ghosh. “Complexity Theory Column 121: Advances in Polynomial Identity Testing”. In: *ACM SIGACT News* 55.2 (2024), pp. 53–88.

- [DGV24] Ashish Dwivedi, Zeyu Guo, and Ben Lee Volk. “Optimal Pseudorandom Generators for Low-Degree Polynomials over Finite Fields”. In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Vol. 317. LIPIcs. Leibniz Int. Proc. Inform. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2024, Art. No. 44, 19.
- [DL78] Richard A. DeMillo and Richard J. Lipton. “A Probabilistic Remark on Algebraic Program Testing”. In: *Inf. Process. Lett.* 7.4 (1978), pp. 193–195.
- [DMS21] Ashish Dwivedi, Rajat Mittal, and Nitin Saxena. “Efficiently Factoring Polynomials modulo  $p^4$ ”. In: *J. Symbolic Comput.* 104 (2021), pp. 805–823.
- [DPS20] Jintai Ding, Albrecht Petzoldt, and Dieter S. Schmidt. *Multivariate Public Key Cryptosystems*. Vol. 80. Advances in Information Security. Springer, New York, 2020.
- [DSS22] Pranjal Dutta, Nitin Saxena, and Amit Sinhababu. “Discovering the Roots: Uniform Closure Results for Algebraic Closure”. In: *J. ACM* 69.3 (2022), Art. 18, 39.
- [DST24] Pranjal Dutta, Amit Sinhababu, and Thomas Thierauf. “Derandomizing Multivariate Polynomial Factoring for Low-Degree Polynomials”. In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Vol. 317. LIPIcs. Leibniz Int. Proc. Inform. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2024, Art. No. 75, 20.
- [DSY09] Zeev Dvir, Amir Shpilka, and Amir Yehudayoff. “Hardness-Randomness Tradeoffs for Bounded Depth Arithmetic Circuits”. In: *SIAM J. Comput.* 39.4 (2009), pp. 1279–1293.
- [For+21] Michael A. Forbes, Amir Shpilka, Iddo Tzameret, and Avi Wigderson. “Proof Complexity Lower Bounds from Algebraic Circuit Complexity”. In: *Theory Comput.* 17 (2021), Paper No. 10, 88.
- [For14] Michael Andrew Forbes. “Polynomial Identity Testing of Read-Once Oblivious Algebraic Branching Programs”. PhD thesis. Massachusetts Institute of Technology, 2014.
- [For15] Michael A. Forbes. “Deterministic Divisibility Testing via Shifted Partial Derivatives”. In: *2015 IEEE 56th Annual Symposium on Foundations of Computer Science—FOCS 2015*. IEEE Computer Soc., Los Alamitos, CA, 2015, pp. 451–465.
- [Fre07] Günther Frei. “The Unpublished Section Eight: On the Way to Function Fields over a Finite Field”. In: *The Shaping of Arithmetic after C. F. Gauss’s It Disquisitiones Arithmeticae*. Springer, Berlin, 2007, pp. 159–198.
- [FS13] Michael A. Forbes and Amir Shpilka. “Quasipolynomial-Time Identity Testing of Non-Commutative and Read-Once Oblivious Algebraic Branching Programs”. In: *2013 IEEE 54th Annual Symposium on Foundations of Computer Science—FOCS 2013*. IEEE Computer Soc., Los Alamitos, CA, 2013, pp. 243–252.
- [FS15] Michael A. Forbes and Amir Shpilka. “Complexity Theory Column 88: Challenges in Polynomial Factorization”. In: *ACM SIGACT News* 46.4 (2015), pp. 32–49.

- [Fur67] Harry Furstenberg. “Algebraic Functions over Finite Fields”. In: *Journal of Algebra* 7 (1967), pp. 271–277.
- [Gat06] Joachim von zur Gathen. “Who Was Who in Polynomial Factorization: 1”. In: *Proceedings of the 2006 International Symposium on Symbolic and Algebraic Computation*. ISSAC '06. Association for Computing Machinery, 2006, p. 2.
- [Gat84] Joachim von zur Gathen. “Hensel and Newton Methods in Valuation Rings”. In: *Math. Comp.* 42.166 (1984), pp. 637–661.
- [GG13] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. 3rd ed. Cambridge University Press, Cambridge, 2013.
- [GK85] Joachim von zur Gathen and Erich Kaltofen. “Factoring Sparse Multivariate Polynomials”. In: *J. Comput. System Sci.* 31 (1985), pp. 265–287.
- [GP01] Joachim von zur Gathen and Daniel Panario. “Factoring Polynomials over Finite Fields: A Survey”. In: *J. Symbolic Comput.* 31.1 (2001), pp. 3–17.
- [Gro12] Joshua A. Grochow. “Symmetry and Equivalence Relations in Classical and Geometric Complexity Theory”. PhD thesis. University of Chicago, 2012.
- [Gro13] Joshua Grochow. *Degree Restriction for Polynomials in VP*. Theoretical Computer Science Stack Exchange. 2013.
- [Gro20] Joshua A. Grochow. “Complexity in Ideals of Polynomials: Questions on Algebraic Complexity of Circuits”. In: *Bull. Eur. Assoc. Theor. Comput. Sci. EATCS* 130 (2020), pp. 40–62.
- [GRS23] Venkat Guruswami, Atri Rudra, and Madhu Sudan. “Essential Coding Theory”. Book Draft. 2023.
- [GS13] S. B. Gashkov and I. S. Sergeev. “Complexity of Computation in Finite Fields”. In: *J. Math. Sci.* 191.5 (2013), pp. 661–685.
- [GS92] Joachim von zur Gathen and Victor Shoup. “Computing Frobenius Maps and Factoring Polynomials”. In: *Comput. Complexity* 2.3 (1992), pp. 187–224.
- [GS99] Venkatesan Guruswami and Madhu Sudan. “Improved Decoding of Reed-Solomon and Algebraic-Geometric Codes”. In: *IEEE Trans. Inform. Theory* 45.6 (1999), pp. 1757–1767.
- [GTZ88] Patrizia Gianni, Barry Trager, and Gail Zacharias. “Gröbner Bases and Primary Decomposition of Polynomials”. In: *J. Symbolic Comput.* 6.2-3 (1988), pp. 149–167.
- [Guo+22] Zeyu Guo, Mrinal Kumar, Ramprasad Saptharishi, and Noam Solomon. “Derandomization from Algebraic Hardness”. In: *SIAM J. Comput.* 51.2 (2022), pp. 315–335.
- [Gup+16] Ankit Gupta, Pritish Kamath, Neeraj Kayal, and Ramprasad Saptharishi. “Arithmetic Circuits: A Chasm at Depth 3”. In: *SIAM J. Comput.* 45.3 (2016), pp. 1064–1079.

- [Hen04] Kurt Hensel. “*Neue Grundlagen der Arithmetik*”. In: *J. Reine Angew. Math.* 127 (1904), pp. 51–84.
- [Hen08] Kurt Hensel. *Theorie der Algebraischen Zahlen*. Vol. 1. BG Teubner, 1908.
- [Hen18] Kurt Hensel. “*Eine neue Theorie der Algebraischen Zahlen*”. In: *Math. Z.* 2.3-4 (1918), pp. 433–452.
- [Hen97] Kurt Hensel. “*Über eine neue Begründung der Theorie der Algebraischen Zahlen*.” In: *Jahresbericht der Deutschen Mathematiker-Vereinigung* 6 (1897), pp. 83–88.
- [HG23] Qiao-Long Huang and Xiao-Shan Gao. “*New Sparse Multivariate Polynomial Factorization Algorithms over Integers*”. In: *Proceedings of the International Symposium on Symbolic & Algebraic Computation (ISSAC 2023)*. ACM, New York, 2023, pp. 315–324.
- [HLT24] Tuomas Hakoniemi, Nutan Limaye, and Iddo Tzameret. “*Functional Lower Bounds in Algebraic Proofs: Symmetry*”. In: *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*. ACM, 2024, pp. 1396–1404.
- [HS80] Joos Heintz and Claus-Peter Schnorr. “*Testing Polynomials Which Are Easy to Compute (Extended Abstract)*”. In: *Proceedings of the 12th Annual ACM Symposium on Theory of Computing, April 28-30, 1980, Los Angeles, California, USA*. ACM, 1980, pp. 262–272.
- [Hu16] Yining Hu. *Explicit Implicit Function Theorem for All Fields*. Nov. 2016. arXiv: [1611.01415](https://arxiv.org/abs/1611.01415) [math].
- [HY11] Pavel Hrubeš and Amir Yehudayoff. “*Arithmetic Complexity in Ring Extensions*”. In: *Theory of Computing* 7.8 (2011), pp. 119–129.
- [Iva+12] Gábor Ivanyos, Marek Karpinski, Lajos Rónyai, and Nitin Saxena. “*Trading GRH for Algebra: Algorithms for Factoring*”. In: *Math. Comp.* 81.277 (2012), pp. 493–531.
- [Jan11] Maurice J. Jansen. “*Extracting Roots of Arithmetic Circuits by Adapting Numerical Methods*”. In: *Innovations in Computer Science - ICS 2011*. Tsinghua University Press, 2011, pp. 87–100.
- [Juk12] Stasys Jukna. *Boolean Function Complexity*. Vol. 27. Algorithms and Combinatorics. Springer, Heidelberg, 2012.
- [Kal82] Erich Kaltofen. “*Factorization of Polynomials*”. In: *Computer Algebra: Symbolic and Algebraic Computation*. Vienna: Springer, 1982, pp. 95–113.
- [Kal85] Erich Kaltofen. “*Polynomial-Time Reductions from Multivariate to Bi- and Univariate Integral Polynomial Factorization*”. In: *SIAM J. Comput.* 14.2 (1985), pp. 469–489.

- [Kal87] Erich Kaltofen. “Single-Factor Hensel Lifting and Its Application to the Straight-Line Complexity of”. In: *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*. STOC '87. Association for Computing Machinery, 1987.
- [Kal89] Erich Kaltofen. “Factorization of Polynomials given by Straight-Line Programs”. In: *Adv. Comput. Res.* 5 (1989), pp. 375–412.
- [Kal90] Erich Kaltofen. “Polynomial Factorization 1982–1986”. In: *Computers in mathematics*. CRC Press, 1990, pp. 285–309.
- [Kal92] Erich Kaltofen. “Polynomial Factorization 1987–1991”. In: *LATIN '92 (São Paulo, 1992)*. Vol. 583. Lecture Notes in Comput. Sci. Springer, Berlin, 1992, pp. 294–313.
- [Kal95] Erich Kaltofen. “Effective Noether Irreducibility Forms and Applications”. In: *J. Comput. System Sci.* 50.2 (1995), pp. 274–295.
- [KI04] Valentine Kabanets and Russell Impagliazzo. “Derandomizing Polynomial Identity Tests Means Proving P = NP”. In: *Comput. Complexity* 13.1-2 (2004), pp. 1–46.
- [KK08] Erich Kaltofen and Pascal Koiran. “Expressing a Fraction of Two Determinants as a Determinant”. In: *ISSAC 2008*. ACM, New York, 2008, pp. 141–146.
- [KL94] Erich L. Kaltofen and Austin Lobo. “Factoring High-Degree Polynomials by the Black Box Berlekamp”. In: *Proceedings of the International Symposium on Symbolic and Algebraic Computation, ISSAC '94*. ACM, 1994, pp. 90–98.
- [Koi12] Pascal Koiran. “Arithmetic Circuits: The Chasm at Depth Four Gets Wider”. In: *Theoret. Comput. Sci.* 448 (2012), pp. 56–65.
- [Kop14] Swastik Kopparty. *Algorithmic Number Theory*. 2014. Lecture notes, Rutgers University.
- [KP13] Steven G. Krantz and Harold R. Parks. *The Implicit Function Theorem*. Modern Birkhäuser Classics. Birkhäuser/Springer, New York, 2013.
- [Kra19] Jan Krajíček. *Proof Complexity*. Vol. 170. Encyclopedia of Mathematics and Its Applications. Cambridge University Press, Cambridge, 2019.
- [Kra95] Jan Krajíček. *Bounded Arithmetic, Propositional Logic, and Complexity Theory*. Vol. 60. Encyclopedia of Mathematics and Its Applications. Cambridge University Press, Cambridge, 1995.
- [KRS24] Mrinal Kumar, Varun Ramanathan, and Ramprasad Satharishi. “Deterministic Algorithms for Low Degree Polynomials”. In: *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, Philadelphia, PA, 2024, pp. 3901–3918.
- [KS06] Neeraj Kayal and Nitin Saxena. “Complexity of Ring Morphism Problems”. In: *Comput. Complexity* 15.4 (2006), pp. 342–390.

- [KS09] Zohar S. Karnin and Amir Shpilka. “Reconstruction of Generalized Depth-3 Arithmetic Circuits with Bounded Top”. In: *24th Annual IEEE Conference on Computational Complexity*. IEEE Computer Soc., Los Alamitos, CA, 2009, pp. 274–285.
- [KS19] Mrinal Kumar and Ramprasad Saptharishi. “Hardness-Randomness Tradeoffs for Algebraic Computation”. In: *Bull. Eur. Assoc. Theor. Comput. Sci. EATCS* 129 (2019), pp. 56–87.
- [KS98] Erich Kaltofen and Victor Shoup. “Subquadratic-Time Factoring of Polynomials over Finite Fields”. In: *Math. Comp.* 67.223 (1998), pp. 1179–1197.
- [KSS15] Swastik Kopparty, Shubhangi Saraf, and Amir Shpilka. “Equivalence of Polynomial Identity Testing and Polynomial Identity Testing”. In: *Comput. Complexity* 24.2 (2015), pp. 295–331.
- [KST23] Mrinal Kumar, Ramprasad Saptharishi, and Anamay Tengse. “Near-Optimal Bootstrapping of Hitting Sets for Algebraic Circuits”. In: *Theory Comput.* 19 (2023), Paper No. 12, 30.
- [KT90] Erich Kaltofen and Barry M. Trager. “Computing with Polynomials given by Black Boxes for Their Evaluations: GCD”. In: *J. Symbolic Comput.* 9.3 (1990), pp. 301–320.
- [KU11] Kiran S. Kedlaya and Christopher Umans. “Fast Polynomial Factorization and Modular Composition”. In: *SIAM J. Comput.* 40.6 (2011), pp. 1767–1802.
- [Kum+24] Mrinal Kumar, Varun Ramanathan, Ramprasad Saptharishi, and Ben Lee Volk. *Towards Deterministic Algorithms for Constant-Depth Factors of Constant-Depth Circuits*. 2024. arXiv: [2403.01965](https://arxiv.org/abs/2403.01965) [cs].
- [Lan02] Serge Lang. *Algebra*. Revised 3rd. Vol. 211. Graduate Texts in Mathematics. Springer, New York, NY, 2002.
- [Lan17] J. M. Landsberg. *Geometry and Complexity Theory*. Vol. 169. Cambridge Studies in Advanced Mathematics. Cambridge University Press, Cambridge, 2017.
- [LLL82] A. K. Lenstra, H. W. Lenstra Jr., and L. Lovász. “Factoring Polynomials with Rational Coefficients”. In: *Math. Ann.* 261.4 (1982), pp. 515–534.
- [LS78] Richard J. Lipton and Larry J. Stockmeyer. “Evaluation of Polynomials with Super-Preconditioning”. In: *J. Comput. System Sci.* 16.2 (1978), pp. 124–139.
- [LST26] Jiaqi Lu, Rahul Santhanam, and Iddo Tzameret. “AC<sup>0</sup>[p]-Frege Cannot Efficiently Prove That Constant-Depth Algebraic Circuits Are Not Universal”. In: *17th Innovations in Theoretical Computer Science Conference, ITCS2026, Bocconi University, Milan, Italy, January 27-30, 2026*. Vol. 362. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2026, 99:1–99:25.
- [Mah14] Meena Mahajan. “Algebraic Complexity Classes”. In: *Perspectives in Computational Complexity*. Vol. 26. Progr. Comput. Sci. Appl. Logic. Birkhäuser/Springer, Cham, 2014, pp. 51–75.
- [Mal03] Guillaume Malod. “Polynômes et coefficients”. PhD thesis. Université Claude Bernard - Lyon, 2003.

- [Mal07] Guillaume Malod. “The Complexity of Polynomials and Their Coefficient Functions”. In: *Twenty-Second Annual IEEE Conference on Computational Complexity (CCC’07)*. 2007, pp. 193–204.
- [MOV97] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press Series on Discrete Mathematics and Its Applications. CRC Press, Boca Raton, FL, 1997.
- [MP08] Guillaume Malod and Natacha Portier. “Characterizing Valiant’s Algebraic Complexity Classes”. In: *J. Complexity* 24.1 (2008), pp. 16–38.
- [MS01] Ketan D. Mulmuley and Milind Sohoni. “Geometric Complexity Theory. I. An Approach to the P vs. NP Problem”. In: *SIAM J. Comput.* 31.2 (2001), pp. 496–526.
- [MS08] Ketan D. Mulmuley and Milind Sohoni. “Geometric Complexity Theory. II. Towards Explicit Obstructions”. In: *SIAM J. Comput.* 38.3 (2008), pp. 1175–1206.
- [Mul11] Ketan D. Mulmuley. “On P vs. NP and Geometric Complexity Theory”. In: *J. ACM* 58.2 (2011), Art. 5, 26.
- [Mul12] Ketan D. Mulmuley. “The GCT Program toward the P vs. NP Problem”. In: *Commun. ACM* 55.6 (2012), pp. 98–107.
- [Mul17] Ketan D. Mulmuley. “Geometric Complexity Theory V: Efficient Algorithms for Noether Normalization”. In: *J. Amer. Math. Soc.* 30.1 (2017), pp. 225–309.
- [MV97] Meena Mahajan and V. Vinay. “A Combinatorial Algorithm for the Determinant”. In: *Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 1997)*. ACM-SIAM, 1997, pp. 730–738.
- [NV10] Phong Q. Nguyen and Brigitte Vallée, eds. *The LLL Algorithm: Survey and Applications*. Information Security and Cryptography. Berlin, Heidelberg: Springer, 2010.
- [NW94] Noam Nisan and Avi Wigderson. “Hardness vs. Randomness”. In: *J. Comput. System Sci.* 49.2 (1994), pp. 149–167.
- [Oli16] Rafael Oliveira. “Factors of Low Individual Degree Polynomials”. In: *Comput. Complexity* 25.2 (2016), pp. 507–561.
- [Ore22] Øystein Ore. “Über höhere kongruenzen”. In: *Norsk Mat. Forenings Skrifter* (1922).
- [Ost99] Alexander Ostrowski. “On the Significance of the Theory of Convex Polyhedra for Formal Algebra”. In: *SIGSAM Bull.* 33.1 (1999), p. 5.
- [Reg02] Kenneth W. Regan. “Understanding the Mulmuley-Sohoni Approach to P vs. NP”. In: *Bull. Eur. Assoc. Theor. Comput. Sci. EATCS* 78 (2002), pp. 86–99.
- [RS05] Ran Raz and Amir Shpilka. “Deterministic Polynomial Identity Testing in Non-Commutative Models”. In: *Comput. Complexity* 14.1 (2005), pp. 1–19.

- [Sap17] Ramprasad Saptharishi. “Algebra and Computation”. Lecture Notes. TIFR, Mumbai, 2017.
- [Sap21] Ramprasad Saptharishi. “A Survey of Lower Bounds in Arithmetic Circuit Complexity”. Github Survey, 2021.
- [Sax09] Nitin Saxena. “Progress on Polynomial Identity Testing”. In: *Bull. Eur. Assoc. Theor. Comput. Sci. EATCS 99* (2009), pp. 49–79.
- [Sax14] Nitin Saxena. “Progress on Polynomial Identity Testing-II”. In: *Perspectives in Computational Complexity*. Vol. 26. Progr. Comput. Sci. Appl. Logic. Birkhäuser/Springer, Cham, 2014, pp. 131–146.
- [Sax23] Nitin Saxena. *Closure of Algebraic Classes Under Factoring*. 2023. Talk at Recent Trends in Computer Algebra (2023) in Institut Henri Poincaré, Paris.
- [Sch00] Andrzej Schinzel. *Polynomials with Special Regard to Reducibility*. Vol. 77. Encyclopedia of Mathematics and Its Applications. Cambridge University Press, Cambridge, 2000.
- [Sch80] J. T. Schwartz. “Fast Probabilistic Algorithms for Verification of Polynomial Identities”. In: *J. ACM* 27.4 (1980), pp. 701–717.
- [Sho09] Victor Shoup. *A Computational Introduction to Number Theory and Algebra*. 2nd ed. Cambridge University Press, Cambridge, 2009.
- [Sin16] Gaurav Sinha. “Reconstruction of Real Depth-3 Circuits with Top Fan-in 2”. In: *31st Conference on Computational Complexity*. Vol. 50. LIPIcs. Leibniz Int. Proc. Inform. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2016, Art. No. 31, 53.
- [Sin19] Amit Sinhababu. “Power series in complexity: Algebraic Dependence, Factor Conjecture and Hitting Set for Closure”. PhD thesis. Indian Institute of Technology Kanpur, 2019.
- [Sin22] Gaurav Sinha. “Efficient Reconstruction of Depth Three Arithmetic Circuits with Top Fan-in Two”. In: *13th Innovations in Theoretical Computer Science Conference (ITCS 2022)*. Vol. 215. Leibniz International Proceedings in Informatics (Lipics). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022, 118:1–118:33.
- [SS25] Shubhangi Saraf and Devansh Shringi. “Reconstruction of Depth 3 Arithmetic Circuits with Top Fan-in 3”. In: *40th Computational Complexity Conference*. Vol. 339. LIPIcs. Leibniz Int. Proc. Inform. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2025, Art. No. 21, 22.
- [SSS13] Chandan Saha, Ramprasad Saptharishi, and Nitin Saxena. “A Case of Depth-3 Identity Testing, Sparse Factorization”. In: *Comput. Complexity* 22.1 (2013), pp. 39–69.

- [ST21] Amit Sinhababu and Thomas Thierauf. “Factorization of Polynomials given by Arithmetic Branching”. In: *Comput. Complexity* 30.2 (2021), Paper No. 15, 47.
- [Sta24] Richard P. Stanley. *Enumerative Combinatorics. Volume 2*. 2nd ed. Vol. 208. Cambridge Studies in Advanced Mathematics. Cambridge University Press, Cambridge, 2024.
- [Str73] Volker Strassen. “Vermeidung von Divisionen”. In: *J. Reine Angew. Math.* 264 (1973), pp. 184–202.
- [Sud97] Madhu Sudan. “Decoding of Reed Solomon Codes beyond the Error-Correction Bound”. In: *J. Complexity* 13.1 (1997), pp. 180–193.
- [Sud98] Madhu Sudan. “Algebra and Computation”. Lecture Notes. Massachusetts Institute of Technology, 1998.
- [SV10] Amir Shpilka and Ilya Volkovich. “On the Relation between Polynomial Identity Testing and Finding”. In: *Automata, Languages and Programming. (ICALP 2010)*. Vol. 6198. Lecture Notes in Comput. Sci. Springer, Berlin, 2010, pp. 408–419.
- [SY10] Amir Shpilka and Amir Yehudayoff. “Arithmetic Circuits: A Survey of Recent Results and Open Questions”. In: *Found. Trends Theor. Comput. Sci.* 5.3-4 (2010), 207–388 (2010).
- [Tav15] Sébastien Tavenas. “Improved Bounds for Reduction to Depth 4 and Depth 3”. In: *Inform. and Comput.* 240 (2015), pp. 2–11.
- [Val+83] L. G. Valiant, S. Skyum, S. Berkowitz, and C. Rackoff. “Fast Parallel Computation of Polynomials Using Permutation Networks”. In: *SIAM J. Comput.* 12.4 (1983), pp. 641–644.
- [Val79a] L. G. Valiant. “Completeness Classes in Algebra”. In: *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing (STOC)*. ACM, New York, 1979, pp. 249–261.
- [Val79b] Leslie G. Valiant. “The Complexity of Computing the Permanent”. In: *Theoretical Computer Science* 8.2 (1979), pp. 189–201.
- [Val82] L. G. Valiant. “Reducibility by Algebraic Projections”. In: *Logic and Algorithmic (Zurich, 1980)*. Vol. 30. Monogr. Enseign. Math. Univ. Genève, Geneva, 1982, pp. 365–380.
- [Vol15] Ilya Volkovich. “Deterministically Factoring Sparse Polynomials into Multilinear Factors and Sums of Products”. In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Vol. 40. LIPIcs. Leibniz Int. Proc. Inform. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2015, pp. 943–958.
- [Vol17] Ilya Volkovich. “On Some Computations on Sparse Polynomials”. In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Vol. 81. LIPIcs. Leibniz Int. Proc. Inform. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2017, Art. No. 48, 21.

- [WW21] E. T. Whittaker and G. N. Watson. *A Course of Modern Analysis—an Introduction to the General Theory of Infinite* 5th ed. Cambridge University Press, Cambridge, 2021.
- [Yun76] David Y. Y. Yun. “Hensel Meets Newton—Algebraic Constructions in an Analytic Setting”. In: *Analytic Computational Complexity (Proc. Sympos., Carnegie-Mellon Univ., Pittsburgh, Pa., 1975)*. Academic Press, New York-London, 1976, pp. 205–215.
- [Zip79] Richard Zippel. “Probabilistic Algorithms for Sparse Polynomials”. In: *Symbolic and Algebraic Computation (EUROSAM '79, Internat. Sympos., Marseille, 1979)*. Vol. 72. Lecture Notes in Comput. Sci. Springer, Berlin-New York, 1979, pp. 216–226.
- [Zip81] Richard Zippel. “Newton’s Iteration and the Sparse Hensel Algorithm (Extended Abstract)”. In: *Proceedings of the Fourth ACM Symposium on Symbolic and Algebraic Computation. SYMSAC '81*. New York, NY, USA: Association for Computing Machinery, 1981, pp. 68–72.