Towards Deterministic Algorithms for Constant-Depth Factors of Constant-Depth Circuits

Mrinal Kumar *   Varun Ramanathan*   Ramprasad Saptharishi*   Ben Lee Volk†

Abstract

We design a deterministic subexponential time algorithm that takes as input a multivariate polynomial f computed by a constant-depth circuit over rational numbers, and outputs a list L of circuits (of unbounded depth and possibly with division gates) that contains all irreducible factors of f computable by constant-depth circuits. This list L might also include circuits that are spurious: they either do not correspond to factors of f or are not even well-defined, e.g. the input to a division gate is a sub-circuit that computes the identically zero polynomial.

The key technical ingredient of our algorithm is a notion of the *pseudo-resultant* of f and a factor g, which serves as a proxy for the resultant of g and f/g, with the advantage that the circuit complexity of the pseudo-resultant is comparable to that of the circuit complexity of f and g. This notion, which might be of independent interest, together with the recent results of Limaye, Srinivasan and Tavenas [LST21] helps us derandomize one key step of multivariate polynomial factorization algorithms — that of deterministically finding a good starting point for Newton Iteration for the case when the input polynomial as well as the irreducible factor of interest have small constant-depth circuits.

1   Introduction

Algorithms for polynomial factorization is an area in which the field of computer algebra has been remarkably successful. Unlike the analogous and notoriously hard problem of *integer* factorization, a sequence of works in the last few decades provided clever and efficient algorithms for factorizing *polynomials*. These algorithms work in many different settings: finite and infinite fields, univariate and multivariate, white- vs. black-box, and so on [FS15, vzGG13].

One of the first questions in the design of factorization algorithms is how is the input represented (and, for that matter, what are the requirements regarding the representation of the output).

An n-variate polynomial of degree d has potentially $\binom{n+d}{d}$ monomials, so representing it as a list of coefficients under some ordering of the monomials (the so-called *dense* representation) would cause the input to be of roughly that size. Even in this model, where the running time is allowed to be polynomial in $\binom{n+d}{d}$, factorization is not a trivial task. Some polynomials, however, have more efficient representations, the most natural of which being an *algebraic circuit*. Such a circuit is a directed acyclic graph, whose inputs are labeled by variables $x_1, \ldots, x_n$ and field elements, and internal nodes labeled by the arithmetic operations $+, -, \times, \div$. Such a circuit naturally represents (or computes) the polynomial(s) computed in its output gate(s).

A series of influential works culminated in efficient randomized algorithms by Kaltofen [Kal89] and Kaltofen and Trager [KT88] for factorization of polynomials given by algebraic circuits. These algorithms have found numerous applications in various fields (e.g., [KI04, DSY09, Sud97, GS99], to name a few, and see [FS15] for more background) They also established the highly non-obvious, and perhaps surprising fact, that factors of polynomials computed by small circuits themselves have small circuits.[1]. Faced with this fact, it is natural to wonder how far this connection extends: that is, suppose $\mathcal{C}$ is a class of algebraic circuits. Can we argue that factors of polynomials that have circuits in $\mathcal{C}$ themselves have circuits in $\mathcal{C}$? Such connections are known when $\mathcal{C}$ is a class that is powerful enough to support the arguments used in Kaltofen's proof, such as the class of polynomial size algebraic circuits or polynomial size algebraic branching programs [ST21]. But much less is known when $\mathcal{C}$ is a weaker class, even for classes that we understand pretty well. For example, despite considerable effort [BSV20] such a statement isn't known to hold when $\mathcal{C}$ is the class of depth-2 circuits of polynomial size (that is, circuits that compute sparse polynomials).

Further, algorithms of Kaltofen [Kal89] and Kaltofen and Trager [KT88] are randomized, and from a theoretical viewpoint this raises the intriguing problem of derandomizing them. However, as observed by Shpilka and Volkovich [SV10], obtaining a deterministic algorithm for factorization implies an efficient algorithm for the famous Polynomial Identity Testing (PIT) problem, which in turn implies circuit lower bounds [KI04]. Indeed, $f(x_1, \ldots, x_n)$ is non-zero if and only if the polynomial $g(x_1, \ldots, x_n, y, z) = f(x_1, \ldots, x_n) + yz$ is irreducible, and given a circuit for f one can easily obtain a circuit for g of nearly the same size, and further, for almost any non-trivial class $\mathcal{C}$, a $\mathcal{C}$-circuit for f implies a $\mathcal{C}$-circuit for g. Thus, we see that factorization of circuits from restricted classes $\mathcal{C}$ is at least as hard as polynomial identity testing for circuits from the same class.

This motivates studying the factorization problem for restricted classes $\mathcal{C}$ for which we can obtain non-trivial PIT algorithms. One can be even further encouraged by the fact that Kopparty, Saraf and Shpilka [KSS14] proved an equivalence between derandomization of PIT and factorization: namely, they proved that a polynomial time deterministic algorithm for PIT would im-

---

[1]A more accurate statement would be that if f is an n-variate polynomials of degree poly(n) that has a circuit of size poly(n), then any factor g of f has a circuit of size poly(n). In general, however, n-variate polynomials with circuits of size poly(n) can have exponential degree. For such polynomials, the questions of whether their factors are efficiently computable is still open and known as the *factor conjecture* [Bür00, Conjecture 8.3]

ply a polynomial time algorithm for factorization, by showing how to use PIT to derandomize Kaltofen's algorithm. However, as before (and for the same reason), their argument breaks down when specialized to circuits from a restricted class $\mathcal{C}$: even when trying to factorize polynomials from a weak class $\mathcal{C}$, Kaltofen's algorithm (and consequentially, its conditional derandomization by Kopparty, Saraf and Shpilka [KSS15]) is forced to solve PIT instances for strong classes, such as the class of polynomial size algebraic branching program. This leaves open the interesting question of deterministically factoring polynomials computed by limited classes of circuits.

In fact, even for simpler questions such as irreducibility testing of polynomials, or checking whether one polynomial divides another, it is hard to obtain deterministic algorithms, even under restricting assumptions on the complexity of the polynomials involved. A step in this direction was taken by Forbes [For15], who showed how to deterministically test whether a quadratic polynomial divides a sparse polynomial.

More recently, in [KRS23] the first three authors of this paper obtained a deterministic quasipolynomial time algorithm that outputs all constant-degree factors of a sparse polynomial. Even more generally, they showed how to use recent PIT results for constant-depth circuits [LST21] in order to compute all constant-degree factors of constant-depth circuits, albeit in subexponential time.

Polynomials of constant degree are in particular sparse and hence as a natural possible generalization of this result, one would like to be to deterministically output all sparse factors of constant-depth circuits, and even more generally, all constant-depth factors of constant-depth circuits. In this work, we make a step in this direction.

1.1 Our Results

Our main result in this work is the following theorem.

Theorem 1.1 (Subexponential list containing constant-depth factors of constant-depth circuits)**.** *Let $\mathbb{Q}$ be the field of rational numbers and $\varepsilon > 0, \Delta_1, \Delta_2 \in \mathbb{N}$ be arbitrary constants.*

Then, there is a deterministic algorithm that takes as input an algebraic circuit $C \in (\Sigma\Pi)^{(\Delta_1)}$ of size s, bit-complexity t and degree D, along with a size parameter m, and outputs a list of circuits $L = \{C_1, \ldots, C_r\}$ with the following properties.

1. *Each $C_i \in L$ is an arithmetic circuit of size $\mathrm{poly}(s, m, D)$ and bit-complexity $\mathrm{poly}(t, s, D, m)$, with division gates.*

2. *For every irreducible factor g of C with a size m $(\Sigma\Pi)^{(\Delta_2)}$-circuit computing it, there exists a $C_i \in L$ such that C_i computes g.*

The size of the list L as well as the running time of the algorithm are bounded above by $O(smDt)^{O((smDt)^{\varepsilon})}$.

We remark that indeed Theorem 1.1 doesn't completely solve the problem of finding sparse (or more generally, low depth) factors of low depth circuits. It merely outputs a "short" (subexponential) list L of candidate polynomials such that every irreducible factor of the input is contained

in L. As explained in Section 2, for technical reasons, we currently do not know how to deterministically prune the list L to obtain the true factors.

Yet another technicality is that our algorithm needs a size bound on the size of a circuit for the factors we care about because no structural guarantees are known for the factors of constant-depth circuits. For instance, we do not know if the factors of a constant-depth circuit also have small constant-depth circuits.

A slightly more general statement. Our techniques in the proof of Theorem 1.1 give a slightly more general statement connecting the questions of deterministic polynomial factorization and deterministic polynomial identity testing. Let $\mathcal{C}$ be a class of circuits that are somewhat rich in the sense that if a circuit C is in $\mathcal{C}$, then the circuit C' obtained from C by a change of basis continues to be in $\mathcal{C}$, and for circuits $C_1, C_2, \ldots, C_n$ in $\mathcal{C}$, and a circuit $B(y_1, y_2, \ldots, y_n)$ of constant depth, the circuit $B(C_1, C_2, \ldots, C_n)$ continues to be in $\mathcal{C}$. For instance, the class of polynomials computable by small constant-depth circuits, or by small formulas satisfy the above property. Our techniques in the proof of Theorem 1.1 generalize to such classes and give a deterministic reduction from the question of computing irreducible factors of a polynomial $f \in \mathcal{C}$ that are also computable in $\mathcal{C}$ to the question of polynomial identity testing of polynomials (of slightly larger complexity) in $\mathcal{C}$; the major caveat being that the algorithm will only output a list containing all such irreducible factors of interest, but might also contain some spurious circuits. Moreover, these circuits in the output list are potentially of unbounded depth, can contain division gates and some of these spurious circuits in the list might not even be well defined, i.e., they involve division by an identically zero polynomial. In contrast to this, the prior known connections between PIT and polynomial factorization, most notably in the work of Kopparty, Saraf and Shpilka [KSS15] gives a truly deterministic polynomial factorization algorithm, given access to an oracle for PIT. However, even if the input to the polynomial factorization algorithm is very structured (for instance, is a sparse polynomial), the PIT instances generated in the process appear to be very general.

In spite of this slight generality, for a clean and complete presentation of the results, we only work with constant-depth circuits in the rest of the paper.

Field dependence of our result. For our results, we work over the field of rational numbers due to various technical reasons. Our proofs rely on derivative-based techniques like working with the Taylor expansion of a polynomial, and working over fields of characteristic zero ensures that derivatives do not inadvertently vanish. We also need a deterministic algorithm for univariate factorization over the underlying field as a subroutine for our algorithms. Moreover, we also need a non-trivial deterministic algorithm for polynomial identity testing of constant-depth circuits over the underlying field [LST21]. The field of rational numbers satisfies all these requirements.

1.2 Open Problems

We conclude this section with some open problems.

- Perhaps the most natural question here would be to prune the list output in Theorem 1.1 so that it has no circuits that do not correspond to true factors. One possible approach to do this would be to understand the complexity of the lifting process in polynomial factorization algorithms better, and show improved upper bounds on the complexity of the roots and factors and then do deterministic divisibility testing. At the moment, it is unclear to us if this approach is feasible.

- One special case of Theorem 1.1 that is already very interesting is that of sparse polynomials. It would be interesting to see if Theorem 1.1 can be improved for this case in any way. For instance, can the time complexity of the algorithm be reduced to quasipolynomial time, or can spurious factors in the output list be eliminated deterministically when the input is sparse?

- In general, the question of what is the complexity of factors of simple polynomials (e.g. sparse polynomials, constant-depth circuits) is of great interest, yet poorly understood. We do not know whether such classes are closed under taking factors, and evidence in any direction, either towards or against such closure results, would be very interesting.

Organization

The rest of the paper is organized as follows. We discuss a high level overview of the proof in Section 2 and introduce and discuss the notion of pseudo-resultant in Section 4. In Section 5, we build upon this notion to describe and present our final algorithms.

As with many of the papers on polynomial factorization, we have a somewhat elaborate preliminaries section with many of the standard facts and results. To keep this paper self-contained, we include this section in the paper; however, to avoid obstructing the flow of the paper, we have moved most of the section to Appendix A and Appendix B.

2 Proof Overview

In this section, we give a brief overview of the main ideas in our proof. But first, we set up some notation: f denotes the input polynomial of degree d that is given via a small algebraic circuit of size s and depth Δ computing it. Furthermore, we assume that f factors as $f = g^m \cdot h$, where g is an irreducible polynomial and can be computed by a small constant-depth circuit, and g and h do not have a non-trivial GCD. Throughout the underlying field is assumed to be the field $\mathbb{Q}$ of rational numbers. We note that a priori, it is not clear if h can also be computed by a small

constant-depth circuit. However, this follows immediately from an observation in a recent prior work of Kumar, Saptharishi and Ramanathan [KRS23].

We now outline the sketch of our ideas. Our algorithm follows the general outline of a typical factorization algorithms in the literature (e.g., [Kal89]). However, in their original formulations, some of these steps rely on randomness, and we elaborate how we get around this and implement the step in a deterministic way.

Making f monic: As the first step of the algorithm, we make the polynomial f monic in one of the variables, which we denote by y here by applying an invertible linear transformation to the variables. This is typically done by choosing field constants $\mathbf{a} = (a_1, a_2, \ldots, a_n) \in \mathbb{F}^n$ at random, and replacing the variable x_i in the circuit for f by the linear form $x_i + a_i y$. Since the underlying field is large enough, via the Schwartz-Zippel lemma, we get that the coefficient of y^d in $f(\mathbf{x} + \mathbf{a} \cdot y)$ is a non-zero field constant with high probability, and up to a scaling by this field constant, we obtain a polynomial that is monic in y. For the ease of notation, we continue to denote this new polynomial by f. To derandomize this step, we note that the coefficient of y^d in $f(\mathbf{x} + \mathbf{a} \cdot y)$ is just the evaluation of the homogeneous component of $f(\mathbf{x})$ of degree d on the input $\mathbf{a}$. But since f has a size s, depth Δ circuit, its degree d homogeneous component has a circuit of size $s' = \mathrm{poly}(s, d)$ and depth $\Delta' = \Delta + O(1)$ (see Lemma A.2). Thus, if we try all possible inputs $\mathbf{a}$ from a hitting set for size s', depth Δ' circuits, one of the vectors will have the desired property. A deterministic subexponential time construction of such a hitting set for constant-depth circuits was shown by Limaye, Srinivasan and Tavenas [LST21].

For the rest of the algorithm, we view f as a polynomial in $\mathbb{Q}[\mathbf{x}][y]$. We also note that since f is monic in y, without loss of generality, g and h can be assumed to be monic in y (Lemma B.4).

Reducing the multiplicity of g in f: The remaining steps of the algorithm assume that the multiplicity of g in f, denoted by the parameter m here, is one. This can be assumed without loss of generality. Indeed, to reduce to this case, we run our algorithm on all the partial derivatives of f with respect to y, i.e. the polynomials $\{\frac{\partial^i f}{\partial y^i} : i \in \{0, 1, \ldots, d-1\}\}$; the observation being that g is an irreducible factor of multiplicity one of at least one of these polynomials. Moreover, all these polynomials can be computed by a circuit of size $\mathrm{poly}(s, d)$ and depth Δ (Lemma A.3). So, it suffices to focus on recovering factors of multiplicity one of f. This observation was also used in [KRS23].

Finding a good starting point for Newton Iteration: The goal now is to view f as a univariate in y with coefficients from $\mathbb{Q}[\mathbf{x}]$ and obtain a root $\Phi(\mathbf{x})$ of f in the ring $\mathbb{Q}[[\mathbf{x}]]$ of power series in $\mathbf{x}$. Moreover, in order to recover the factor g of f using this root, it must be the case that this is a root of g (and hence a root of f). In fact, by standard techniques, it suffices to recover an approximation

of Φ with high enough accuracy, i.e., to recover $\Phi_k(\mathbf{x}) := \Phi(\mathbf{x}) \mod \langle \mathbf{x} \rangle^k$ for some $k > 2d^2$. This is done via Newton iteration, where we start from the constant term (i.e. $\Phi_0 := \Phi(\mathbf{x}) \mod \langle \mathbf{x} \rangle$) of such a Φ, and *lift* it to obtain Φ_k. While the lifting process is itself deterministic (see Lemma 3.1), a crucial issue with respect to the starting point of the lifting process is the following: Φ_0 must be a root of $f(\mathbf{x}, y) \mod \langle \mathbf{x} \rangle$ of multiplicity one, and in fact must come from $g \mod \langle \mathbf{x} \rangle$. This is typically done by translating the $\mathbf{x}$ variables to $\mathbf{x} + \mathbf{a}$ such that the univariate polynomials $g(\mathbf{a}, y)$ and $h(\mathbf{a}, y)$ do not share a non-trivial GCD and $g(\mathbf{a}, y)$ is square-free. This ensures that every root of $f(\mathbf{a}, y) = g(\mathbf{a}, y)h(\mathbf{a}, y)$ that comes from $g(\mathbf{a}, y)$ is a root of multiplicity one of $f(\mathbf{a}, y)$ and hence is a good starting point for lifting via Newton Iteration.

In typical factorization algorithms, such a point $\mathbf{a}$ is found by first preprocessing the input f such that it is square-free, and then setting $\mathbf{a}$ to be a non-zero of the discriminant of f. Since the discriminant is an efficiently computable low-degree polynomial (assuming f is), by the Schwartz-Zippel lemma $\mathbf{a}$ can be chosen at random. This guarantees that $f(\mathbf{a}, y) = g(\mathbf{a}, y)h(\mathbf{a}, y)$ is square-free and hence $g(\mathbf{a}, y)$ and $h(\mathbf{a}, y)$ don't have a non-trivial GCD. In our setting, we do not have the guarantee that $f(\mathbf{x}, y)$ is square-free since h might not be square-free, but we know that g is a multiplicity one irreducible factor of f. Moreover, we would like to find such a point $\mathbf{a}$ deterministically. A serious quantitative issue with respect to derandomizing the application of the Schwartz-Zippel lemma here is that even though f has a small constant-depth circuit, its discriminant is the determinant of a matrix of dimension roughly $2d$ with polynomial entries, a powerful model for which no non-trivial deterministic PIT algorithms are known. This is also the issue if we try to look at the resultant of g and h instead.

One intriguing possibility here would be to show that the discriminant or the resultant of polynomials with small constant-depth circuits have relatively small constant-depth circuits. However, we do not know how to show this. Instead, we come up with an alternative polynomial $R_{f,g}$, referred to as the *pseudo-resultant* of f and g, and rely on its properties to arrive at our good starting point for the lifting process. In particular, we note that the complexity of $R_{f,g}$ is close to the complexity of f and g. We then show that when g is an irreducible factor of f with multiplicity one, $R_{f,g}(\mathbf{x}, y) \not\equiv 0$. Moreover, using ideas from a divisibility testing algorithm of Forbes [For15], we can show that for any $\mathbf{a} \in \mathbb{Q}^n$ such that $R_{f,g}(\mathbf{a}, y) \not\equiv 0$, there is a root of multiplicity one of $f(\mathbf{a}, y)$ that is a root of $g(\mathbf{a}, y)$ but not $h(\mathbf{a}, y)$. This root becomes a good starting point for the lifting process. Furthermore, since $R_{f,g}$ is computable by a small constant-depth circuit, such a point $\mathbf{a}$ can be obtained deterministically in subexponential time using the results in [LST21].

Intuitively, while the randomized algorithm for finding a good starting point for lifting uses the resultant to find an $\mathbf{a}$ such that $g(\mathbf{a}, y)$ and $h(\mathbf{a}, y)$ do not have a non-trivial GCD, we observe that this is a bit of an overkill, and it suffices to find an $\mathbf{a}$ such that the GCD of $g(\mathbf{a}, y)$ and $h(\mathbf{a}, y)$ is not equal to $g(\mathbf{a}, y)$. Equivalently, it suffices to find an $\mathbf{a}$ such that $g(\mathbf{a}, y)$ does not divide $h(\mathbf{a}, y)$. This happens to be a computationally easier condition to achieve deterministically via the pseudo-

resultant. This is essentially the main technical idea in this work, and is formally discussed in Section 4.

One undesirable quantitative issue here is that the circuits obtained for Φ via Newton iteration, while being of polynomially bounded size, can have unbounded depth. This is one of the reasons that the output list of our algorithm can have spurious circuits not corresponding to factors of f.

Solving a linear system to obtain g: Once we have obtained an approximate root $\Phi_k(\mathbf{x})$ of $f(\mathbf{x}, y)$, we would like to recover the real factor $g(\mathbf{x}, y)$. If the y-degree of g is known to be d', then this is done via thinking of g as $g(\mathbf{x}, y) = y^{d'} + \sum_{i=0}^{d'-1} g_i(\mathbf{x}) y^i$, where g_i is a formal variable, and imposing the constraint that

$$
g(\mathbf{x}, \Phi_k(\mathbf{x})) = \varphi_k(\mathbf{x})^{d'} + \sum_{i=0}^{d'-1} g_i(\mathbf{x}) \Phi_k(\mathbf{x})^i \equiv 0 \mod \langle \mathbf{x} \rangle^k .
$$

Furthermore, each g_i can be written as $g_i(\mathbf{x}) = \sum_{j=0}^{d} g_{i,j}(\mathbf{x})$, where $g_{i,j}$ is the homogeneous component of g_i of degree j. The constraint above translates naturally into a linear system in the variables $g_{i,j}$, and the factor g does satisfy this system of equations. Furthermore, it can be shown that if d' equals the y-degree of g and $k > 2d^2$, then g is the unique solution to this system of linear equations. Moreover, if we set a similar system with the y-degree of the factor of interest being set to something less than d', then the resulting linear system has no solution. We rely on these properties of the linear system to obtain an algebraic circuit for g. Essentially, the idea is that if we know d' and the resulting linear system is given by a square matrix, then uniqueness of solution of degree d' guarantees that this matrix is invertible, and the solution can then be expressed as an algebraic circuit given by Cramer's rule. However, the constraint matrix of the system might not be square in general, and in this case, we first make the system square (deterministically) before applying Cramer's rule. The details can be found in Lemma 3.2.

One added subtlety here is that in general, we would not know the exact y-degree of g. In this case, we try all values from 1 to $d - 1$, and for each one formally construct an algebraic circuit with divisions and unbounded depth (as noted earlier, the circuit for Φ can already have unbounded depth) and include all such circuits in our final list of solutions.

Differences with [KRS23]: Many of the high-level ideas in the algorithm outlined above are similar to those in a prior work of Kumar, Ramanathan and Saptharishi [KRS23], where deterministic subexponential time algorithms were given for computing constant-degree factors of constant-depth circuits. The similarities include the motivation for the problems as well as the high-level approach of understanding the structure of the PIT instances appearing in the randomized algorithms for multivariate factorization for these special cases carefully and derandomizing those steps. However, there are key technical differences. The main technical result of [KRS23] is an

upper bound on the complexity of the resultant of a constant-depth circuit and a constant-degree polynomial. However, we are unable to prove such an upper bound in our case. Instead, we define the pseudo-resultant and observe that its circuit complexity is comparable to that of the given input polynomial and the factors we care about. Moreover, it helps us completely avoid working with the resultant of g and h or the discriminant of f, and lets us find a good starting point for Newton iteration in deterministic subexponential time. There are also some technical differences in how we go from approximate power series roots to actual factors of f, but the main takeaway from this work is perhaps the notion of pseudo-resultant and its properties.

Inability to prune the list: Unlike the results in [KRS23], where only true low degree factors of the input polynomial are output, we end up with a list of circuits containing the circuits for all irreducible factors with small constant-depth circuits, but perhaps much more. This difference stems from the fact that in the algorithm of [KRS23], at an intermediate step a list of constant-degree polynomials (essentially expressed as a sum of monomials) is constructed, where similar to this work, the list contains all the true constant-degree factors, as well as some spurious factors. However, since in [KRS23] every polynomial in this intermediate list is a constant-degree polynomial in the monomial basis, the list is pruned to true factors by checking if a polynomial in the list divides the given input polynomial, and whether they are irreducible (which can be done deterministically since the degree of the polynomial is a constant). However, in this work, the list consists of unbounded depth algebraic circuits with division gates. For such circuits, we do not even know how to test if the circuit is well-defined, i.e., that there are no denominators that are identically zero. Hence, it is unclear to us if the list can be pruned to the true irreducible factors in a non-trivial way deterministically. Addressing this issue seems like a very interesting open problem.

3 Notations and preliminaries

We shall briefly describe our notation and some basic definitions. For a more detailed preliminaries section, please refer to Appendix A and Appendix B.

- Throughout this paper, we work over the field $\mathbb{Q}$ of rational numbers. For some of the statements that are used more generally, we use $\mathbb{F}$ to denote an underlying field.

- We use boldface lower case letters like $\mathbf{x}, \mathbf{y}, \mathbf{a}$ to denote tuples, e.g. $\mathbf{x} = (x_1, x_2, \ldots, x_n)$. The arity of the tuple is either stated or will be clear from the context.

- For a polynomial f and a non-negative integer k, $\mathrm{Hom}_k(f)$ denotes the homogeneous component of f of degree *equal* to k. $\mathrm{Hom}_{\leq k}(f)$ denotes the sum of homogeneous components

of f of degree at most k, i.e.,

$$\mathrm{Hom}_{\leq k}(f) := \sum_{i=0}^{k} \mathrm{Hom}_i(f)$$

- For a parameter $k \in \mathbb{Z}_{\geq 0}$, we will use $(\Sigma\Pi)^{(k)}$ to refer to product-depth k circuits[2] with the root gate being $+$ and the deepest layer of gates being $\times$. Since any constant-depth algebraic circuit of depth k and size s can be converted to a formula of depth k and size s^{k+1} i.e. $\mathrm{poly}(s)$, we will use the terms circuits and formulas interchangeably, without any loss in the final bounds we prove.

- Let f and g be multivariate polynomials such that $g \mid f$. Then, the *multiplicity* or *factor multiplicity* of g in f is defined to be the greatest integer a such that g^a divides f.

- The size of a circuit over $\mathbb{Q}$ is equal to the sum of the number of edges and the sum of the bit-complexity of the constants appearing in the circuit.

- In our proofs, we encounter circuits over fields of the form $\mathbb{Q}(\alpha)$ where α is some algebraic number. For such circuits, we think of their description of each field constant as an element of $\mathbb{K} = \frac{\mathbb{Q}[u]}{A(u)}$ where $A(u)$ is its minimal polynomial of α. The bit-complexity of any element of $\mathbb{Q}(\alpha)$ is defined as the bit-complexity of the unique representation as a element of $\mathbb{K}$. The bit-complexity of such a circuit is the total description size of the circuit, which includes the bit-complexity of any constants used internally.

3.1 Newton iteration

The following are some standard facts about Newton iteration and the proofs of these lemmas is present in .

Lemma 3.1 (Newton iteration). *Let $F(\mathbf{x}, y) \in \mathbb{F}[\mathbf{x}, y]$ be monic in y. Suppose $u \in \mathbb{F}$ such that*

$$F(\mathbf{0}, u) = 0,$$
$$(\partial_y F)(\mathbf{0}, u) \neq 0.$$

Then, for all $k \geq 0$, there is an approximate root *polynomial $\Phi_k \in \mathbb{F}[\mathbf{x}]$ that satisfies*

$$F(\mathbf{x}, \Phi_k) = 0 \bmod \langle \mathbf{x} \rangle^{k+1},$$
$$\Phi_k(\mathbf{0}) = u.$$

---

[2]We emphasize that this notation does *not* refer to the kth power of a polynomial computed by a $\Sigma\Pi$ circuit.

Furthermore, if we are provided F a circuit of size and bit-complexity s, and also the element $u \in \mathbb{F}$ with bit-complexity at most B, we can output an algebraic circuit for the approximate root $\Phi_k(\mathbf{x})$ of size at most $\mathrm{poly}(s, k)$ and bit-complexity $\mathrm{poly}(s, B, k)$.

Lemma 3.2 (Computing minimal polynomials of approximate roots). *Let $\Phi_k(\mathbf{x}) \in \mathbb{F}[\mathbf{x}]$ be provided by an algebraic circuit of size and bit-complexity at most s. Suppose there exists a polynomial $G(\mathbf{x}, y) = \sum_{i=0}^{d} G_i(\mathbf{x}) y^i$ that is monic in y and irreducible satisfying*

$$G(\mathbf{x}, \Phi_k(\mathbf{x})) = 0 \bmod \langle \mathbf{x} \rangle^{k+1},$$
$$\deg G \leq D,$$
$$\deg_y G = d,$$
$$k \geq 2Dd.$$

Then, given d and the circuit for Φ_k, for every $i \in [d]$, we can compute an algebraic circuit (with only $\mathbf{x}$ variables and no y), with divisions, of size and bit-complexity at most $\mathrm{poly}(s, d, D)$ for the polynomial $G_i(\mathbf{x})$. In particular, we can compute a circuit of size and bit-complexity $\mathrm{poly}(s, d, D)$ for $G(\mathbf{x}, y)$.

3.2 Pseudo-quotients for preserving indivisibility

We define the pseudo-quotient of a pair of polynomials f and g, which will act as a proxy for the quotient of f and g.

Definition 3.3 (Pseudo-quotients, [For15]). *Let $f, g \in \mathbb{Q}[\mathbf{x}]$ be non-zero polynomials with $g(\mathbf{0}) = \beta \neq 0$. The pseudo-quotient of f and g is defined as*

$$\mathrm{Hom}_{\leq D-d} \left(\left(\frac{f(\mathbf{x})}{\beta} \right) \cdot (1 + \tilde{g} + \tilde{g}^2 + \cdots + \tilde{g}^{D-d}) \right)$$

where $D = \deg(f)$, $d = \deg(g)$ and $\tilde{g} = 1 - \frac{g}{\beta}$.

More generally, if $\boldsymbol{\alpha} \in \mathbb{Q}^n$ is such that $g(\boldsymbol{\alpha}) \neq 0$, the pseudo-quotient of f and g around $\boldsymbol{\alpha}$ is defined as $\tilde{Q}(\mathbf{x} - \boldsymbol{\alpha}, y)$ where $\tilde{Q}(\mathbf{x}, y)$ is the pseudo-quotient of $f(\mathbf{x} + \boldsymbol{\alpha})$ and $g(\mathbf{x} + \boldsymbol{\alpha})$. ◇

Forbes [For15] showed that the pseudo-quotient allows one to reduce divisibility testing to a polynomial identity test.

Theorem 3.4 (Divisibility testing to PIT [For15]). *Let $f(\mathbf{x})$ and $g(\mathbf{x})$ be non-zero polynomials over a field $\mathbb{F}$ such that $g(\mathbf{0}) = \beta \neq 0$. Then, g divides f if and only if $f(\mathbf{x}) - Q_{f,g}(\mathbf{x}) \cdot g(\mathbf{x})$ is identically zero, where $Q_{f,g}$ is the pseudo-quotient of f and g.*

4 Pseudo-resultant

In this section, we will define the *pseudo-resultant* of a polynomial f and a factor g. The pseudo-resultant serves as a proxy for the resultant as it is easy to argue about its circuit complexity, and it suffices for our applications. This definition and its applications to polynomial factorization that we discuss later in the paper are essentially the main technical contributions of this paper.

For a polynomial $f(\mathbf{x}, y)$, let $S_f(\mathbf{x}, y)$ denote $(\partial_y f(\mathbf{x}, y))^2$ and $S_{f(\mathbf{a},y)}(y)$ denotes $(\partial_y f(\mathbf{a}, y))^2$.

Definition 4.1 (Pseudo-resultant). *Given $f, g \in \mathbb{F}[\mathbf{x}, y]$, let $Q(\mathbf{x}, y)$ be the pseudo-quotient of S_f and g interpreted as univariates in y (with coefficients in $\mathbb{F}(\mathbf{x})$). Then, the* pseudo-resultant *of f and g with respect to y, denoted by $R_{f,g}$, is a polynomial in $\mathbb{F}[\mathbf{x}, y]$ defined as*

$$R_{f,g} := S_f - Q \cdot g. \qquad \Diamond$$

The variable y is special in the above definition, and will remain so throughout the paper.

We make the following simple observation that summarizes a natural property of the pseudo-quotient and the pseudo-resultant under substitutions.

Observation 4.2 (Pseudo-quotients and pseudo-resultants under substitutions). *Suppose $f(\mathbf{x}, y), g(\mathbf{x}, y) \in \mathbb{F}[\mathbf{x}, y]$ are monic in y, and suppose $\mathbf{a} \in \mathbb{F}^{|\mathbf{x}|}$ such that $g(\mathbf{a}, 0) \neq 0$. Let $Q(\mathbf{x}, y)$ be the pseudo-quotient of S_f and g interpreted as univariates in y (with coefficients in $\mathbb{F}(\mathbf{x})$). Let $Q'(y)$ be the pseudo-quotient of $S_{f(\mathbf{a},y)}$ and $g(\mathbf{a}, y)$. Let $R_{f(\mathbf{a},y),g(\mathbf{a},y)} := S_{f(\mathbf{a},y)} - Q' \cdot g(\mathbf{a}, y)$ be the pseudo-resultant of $f(\mathbf{a}, y)$ and $g(\mathbf{a}, y)$. Then:*

1. *$Q(\mathbf{a}, y) = Q'(y)$.*

2. *$R_{f,g}(\mathbf{a}, y) = R_{f(\mathbf{a},y),g(\mathbf{a},y)}(y)$.*

Proof. For the first part, let $g(\mathbf{x}, y) = g_0(\mathbf{x}) + g_1(\mathbf{x})y + \cdots + y^d$. By the definition of the pseudo-quotient, we have

$$Q(\mathbf{x}, y) = \mathrm{Hom}_{\leq D-d}\left[\frac{S_f(\mathbf{x}, y)}{g_0(\mathbf{x})}\left(1 + \tilde{g}(\mathbf{x}, y) + \tilde{g}(\mathbf{x}, y)^2 + \cdots + \tilde{g}(\mathbf{x}, y)^{D-d}\right)\right],$$

$$Q'(y) = \mathrm{Hom}_{\leq D-d}\left[\frac{S_{f(a,y)}(y)}{g_0(\mathbf{a})}\left(1 + \hat{g}(y) + \hat{g}(y)^2 + \cdots + \hat{g}(y)^{D-d}\right)\right]$$

where $\tilde{g}(\mathbf{x}, y) = 1 - \frac{g(\mathbf{x},y)}{g_0(\mathbf{x})}$ and $\hat{g}(y) = 1 - \frac{g(\mathbf{a},y)}{g_0(\mathbf{a})}$. It is evident that $\hat{g}(y) = \tilde{g}(\mathbf{a}, y)$. Since the operations of taking partial derivatives with respect to y (in the definitions of S_f and $S_{f(\mathbf{a},y)}$), taking homogeneous components with respect to y and evaluating the $\mathbf{x}$ variables commute with each other, it follows that $Q(\mathbf{a}, y) = Q'(y)$ as claimed.

For the same reason, it follows that $S_f(\mathbf{x}, y) = S_{f(\mathbf{a},y)}(y)$. Combining this observation with the first part, we get $S_f(\mathbf{a}, y) - Q(\mathbf{a}, y)g(\mathbf{a}, y) = S_{f(\mathbf{a},y)}(y) - Q'(y)g(\mathbf{a}, y)$ i.e. $R_{f,g}(\mathbf{a}, y) =$

$R_{f(\mathbf{a},y),g(\mathbf{a},y)}(y)$.   □

The following lemma tells us how to use the pseudo-resultant to find a suitable starting point **a** for Newton iteration.

Lemma 4.3 (Properties of pseudo-resultant). *Suppose $\mathbb{F}$ is a field, $f \in \mathbb{F}[\mathbf{x},y]$ is monic in y, and $f = g \cdot h$, where g is irreducible and $g \nmid h$. Let $S_f(\mathbf{x},y) = (\partial_y f(\mathbf{x},y))^2$, let $R_{f,g}$ denote the pseudo-resultant of f and g with respect to y, and let $R_{f(\mathbf{a},y),g(\mathbf{a},y)}(y)$ denote the pseudo-resultant of $f(\mathbf{a},y)$ and $g(\mathbf{a},y)$ with respect to y. Then, the following properties hold.*

1. *$g \nmid S_f$ in $\mathbb{F}[\mathbf{x},y]$ and $\mathbb{F}(\mathbf{x})[y]$. Equivalently, $R_{f,g}(\mathbf{x},y) \not\equiv 0$ in $\mathbb{F}(\mathbf{x})[y]$.*

2. *For any $\mathbf{a} \in \mathbb{F}^n$ such that $R_{f,g}(\mathbf{a},y) \not\equiv 0$ in $\mathbb{F}[y]$, $\mathbf{a}$ satisfies the following property: there exists $u \in \overline{\mathbb{F}}$ such that $f(\mathbf{a},u) = 0$, $h(\mathbf{a},u) \neq 0$ and $\partial_y f(\mathbf{a},u) \neq 0$. In particular, u is a multiplicity-1 root of $f(\mathbf{a},y)$.*

Proof. Since the polynomial f is monic in y, we have that g, h and S_f are all monic in y as well. Thus, by Gauss' Lemma (Lemma B.4), we get that divisibility in $\mathbb{F}[\mathbf{x},y]$ is the same as that in $\mathbb{F}(\mathbf{x})[y]$, so we focus on divisibility in $\mathbb{F}[\mathbf{x},y]$.

From $f = g \cdot h$ and the product rule for derivatives, we get

$$\partial_y f = h \cdot \partial_y g + g \cdot \partial_y h.$$

Now, since g is irreducible, we have that g divides $(\partial_y f)^2$ if and only if g divides $\partial_y f$. This, in turn, happens if and only if g divides $h \cdot \partial_y g$. But from the hypothesis of the lemma, we know that g is irreducible, and does not divide h. Moreover, since the y-degree of $\partial_y g$ is less than the y-degree of g, we have that g does not divide $\partial_y g$ either. Thus, from the irreducibility of g, it follows that g does not divide the product $h \cdot \partial_y g$, and hence, g does not divide $(\partial_y f)^2$ (in the ring $\mathbb{F}[\mathbf{x},y]$). As mentioned earlier, it now follows from Gauss' lemma (Lemma B.4) that g does not divide $(\partial_y f)^2$ (in the ring $\mathbb{F}(\mathbf{x})[y]$) i.e. $g \nmid S_f$. Combining the definition of $R_{f,g}$ and Forbes' reduction of divisibility testing to PIT (Theorem 3.4), it follows that $R_{f,g}(\mathbf{x},y) \not\equiv 0$ in $\mathbb{F}(\mathbf{x},y)$. This completes the proof of the first item.

We now proceed with the proof of the second item. Let $\mathbf{a}$ be a setting of the $\mathbf{x}$-variables such that $R_{f,g}(\mathbf{a},y) = R_{f(\mathbf{a},y),g(\mathbf{a},y)}(y) \not\equiv 0$ (where the equality follows from Observation 4.2) or equivalently, $g(\mathbf{a},y) \nmid S_f(\mathbf{a},y)$ (by Theorem 3.4 and the fact that $S_f(\mathbf{a},y) = S_{f(\mathbf{a},y)}(y)$). Let $g(\mathbf{a},y)$ factor as $g(\mathbf{a},y) = \prod_i (y - \alpha_i)^{e_i}$ over the algebraic closure $\overline{\mathbb{F}}$ of the field $\mathbb{F}$. Let $g_1(y) = \prod_{i:e_i=1} (y - \alpha_i)$ be the product of linear factors of multiplicity 1, and $g_2(y) = \prod_{i:e_i \geq 2} (y - \alpha_i)^{e_i}$ be the product of rest of the factors. We first claim that for every root α_i of $g_2(y)$ with multiplicity $e_i \geq 2$, it holds that $(y - \alpha_i)^{e_i} \mid (\partial_y g(\mathbf{a},y))^2$. Indeed, denote by $q(y)$, the polynomial $g(\mathbf{a},y)/(y - \alpha_i)^{e_i}$. Then, by the

13

product rule, we have

$$\partial_y g(\mathbf{a}, y) = e_i (y - \alpha)^{e_i - 1} q(y) + \partial_y q(y) \cdot (y - \alpha_i)^{e_i},$$

Since $e_i \geq 2$, we have that $2(e_i - 1) \geq e_i$. Thus, $(y - \alpha_i)^{e_i} \mid (\partial_y g(\mathbf{a}, y))^2$ and more generally, since $(y - \alpha_i)^{e_i}$ and $(y - \alpha_j)^{e_j}$ are relatively prime whenever $\alpha_i \neq \alpha_j$, we get that $g_2(y) \mid (\partial_y g(\mathbf{a}, y))^2$.

Furthermore, setting $\mathbf{x}$ to $\mathbf{a}$ in the expression for $S_f(\mathbf{x}, y)$, we get that

$$S_f(\mathbf{a}, y) = (\partial_y g(\mathbf{a}, y) \cdot h(\mathbf{a}, y))^2 + (g(\mathbf{a}, y) \cdot \partial_y h(\mathbf{a}, y))^2 + 2 \cdot (\partial_y g(\mathbf{a}, y)) \cdot g(\mathbf{a}, y) \cdot h(\mathbf{a}, y) \cdot (\partial_y h(\mathbf{a}, y)).$$

We note here that all the partial derivatives in the above expression are taken with respect to the variable y, whereas all the substitutions are happening with respect to the $\mathbf{x}$-variables, and thus these two operations commute with each other. From the form of the expression above, the definition of g_2 and our earlier observation that $g_2(y) \mid (\partial_y g(\mathbf{a}, y))^2$, it immediately follows that $g_2(y) \mid S_f(\mathbf{a}, y)$.

Since $g(\mathbf{a}, y)$ does not divide $S_f(\mathbf{a}, y)$ but $g_2(y)$ does, it must be the case that $g_1(y)$ does not divide $S_f(\mathbf{a}, y)$. Thus, $g_1(y)$ has degree at least one and at least one of its roots in $\overline{\mathbb{F}}$ is *not* a root of $S_f(\mathbf{a}, y)$. We claim that this root satisfies the requirements of being the $u \in \overline{\mathbb{F}}$ from the claim. To this end, we note that

- $S_f(\mathbf{a}, u) \neq 0$ or equivalently, $\partial_y f(\mathbf{a}, u) \neq 0$

- $f(\mathbf{a}, u) = g(\mathbf{a}, u) \cdot h(\mathbf{a}, u) = g_1(u) \cdot g_2(u) \cdot h(\mathbf{a}, u) = 0$ since we chose u so that $g_1(u) = 0$

- $\partial_y f(\mathbf{a}, u) = \partial_g(\mathbf{a}, u) \cdot h(\mathbf{a}, u) + \partial_h(\mathbf{a}, u) \cdot g(\mathbf{a}, u) = \partial_y g(\mathbf{a}, u) \cdot h(\mathbf{a}, u)$ by the previous bullet. But $\partial_y f(\mathbf{a}, u) \neq 0$ and hence $h(\mathbf{a}, u) \neq 0$.

This completes the proof of the second item. $\qquad\square$

4.1 A starting point for Newton iteration

The next lemma proves that in the setting of constant-depth circuits, the complexity of the pseudo-resultant of f and g is comparable to the complexity of f and g. Hence, if we want to find a point $\mathbf{a}$ that will help us start Newton iteration using the sufficient condition from Lemma 4.3 (property 2), then it is enough to consider points in a hitting set for constant-depth circuits.

Lemma 4.4. *Suppose $f \in \mathbb{F}[\mathbf{x}, y]$ is a polynomial of degree D with the following properties.*

- *f is monic in y.*

- *$f = g \cdot h$, where $g = \sum_i g_i(\mathbf{x}) y^i$ is a monic irreducible polynomial of degree d such that $g \nmid h$, and $g_0(\mathbf{x}) \neq 0$.*

- *f can be computed by a circuit in $(\Sigma\Pi)^{(\Delta_1)}$ of size at most s_1 and g can be computed by a size s_2 circuit in $(\Sigma\Pi)^{(\Delta_2)}$ for some constants $\Delta_1, \Delta_2 \in \mathbb{N}$.*

Let $R_{f,g}(\mathbf{x}, y) \in \mathbb{F}(\mathbf{x})[y]$ be the pseudo-resultant of f and g with respect to y. For $\Delta = \max(\Delta_1, \Delta_2)$, let $\mathcal{H}$ be a hitting set for $(n+1)$-variate $(\Sigma\Pi)^{(\Delta+2)}$-circuits of size at most $11s_1s_2D^5$ and degree at most $5D^2$. Then, there exists a point $\tilde{\mathbf{a}} = (a_1, \dots, a_n, a_{n+1}) = (\mathbf{a}, a_{n+1}) \in \mathcal{H}$ such that $R_{f,g}(\mathbf{a}, y) \not\equiv 0$.

Proof. Let $Q(\mathbf{x}, y)$ be the pseudo-quotient of S_f and g as univariates in y. Let the numerator and the denominator of $R_{f,g}(\mathbf{x}, y) = S_f(\mathbf{x}, y) - Q(\mathbf{x}, y) \cdot g(\mathbf{x}, y)$ be $P_{\text{num}}(\mathbf{x}, y)$ and $P_{\text{den}}(\mathbf{x})$ respectively. The proof proceeds by bounding the complexity of the numerator and the denominator, and observing that the given hitting set suffices to find a non-zero for both of them.

Bounding the complexity of the denominator $P_{\text{den}}(\mathbf{x})$: $S_f(\mathbf{x}, y)$ and $g(\mathbf{x}, y)$ do not contribute to the denominator. The only contributions to $P_{\text{den}}(\mathbf{x})$ (from $Q(\mathbf{x}, y)$) are $g_0(\mathbf{x})$ by the $\frac{S_f(\mathbf{x}, y)}{g_0(\mathbf{x})}$ term, and $g_0(\mathbf{x})^{D-d}$ from the powers of $\tilde{g}$. Thus, $P_{\text{den}}(\mathbf{x}) = g_0(\mathbf{x})^{D-d+1}$. Since $g(\mathbf{x}, y)$ has a $(\Sigma\Pi)^{(\Delta_2)}$-circuit of size s_2, so does $g_0(\mathbf{x})$ (obtained by just setting y to zero). Thus, $P_{\text{den}}(\mathbf{x})$ has a $(\Sigma\Pi)^{(\Delta_2+1)}$-circuit of size at most $s_2 + D \le s_2D$. The degree of P_{den} is clearly upper bounded by D^2.

Bounding the complexity of the numerator $P_{\text{num}}(\mathbf{x}, y)$: The numerator of $Q(\mathbf{x}, y)$, denoted by $Q_{\text{num}}(\mathbf{x}, y)$, is

$$Q_{\text{num}}(\mathbf{x}, y) = \text{Hom}_{D-d} \left[S_f(\mathbf{x}, y) \left(\sum_{i=0}^{D-d} g_0^{D-d-i}(g_0 - g)^i \right) \right].$$

From the definition of S_f and standard computation of partial derivatives (see Lemma A.3), $S_f(\mathbf{x}, y)$ has a $(\Sigma\Pi)^{(\Delta_1)}$-circuit of size $(10s_1D^3)$. $\sum_{i=0}^{D-d} g_0^{D-d-i}(g_0 - g)^i$ has a $(\Sigma\Pi)^{(\Delta_2+1)}$-circuit of size s_2D^3. Taking its product with $S_f(\mathbf{x}, y)$ and using interpolation to get the homogeneous components of degree at most $D - d$ (Lemma A.2) gives us a $(\Sigma\Pi)^{(\Delta+1)}$-circuit of size $(10s_1D^5)$ for Q_{num}. Thus, $P_{\text{num}}(\mathbf{x}, y) = P_{\text{den}}(\mathbf{x})S_f(\mathbf{x}, y) - Q_{\text{num}}(\mathbf{x}, y)g(\mathbf{x}, y)$ has a $(\Sigma\Pi)^{(\Delta+1)}$-circuit of size $10s_1s_2D^5$. From the expression of the numerator, we have that a crude bound on its degree is at most $2D + D^2 + D^2 \le 4D^2$.

The hitting set $\mathcal{H}$ suffices : Since $P_{\text{num}}(\mathbf{x}, y)$ and $P_{\text{den}}(\mathbf{x})$ have relatively small $(\Sigma\Pi)^{(\Delta+1)}$-circuits, then their product has a $(\Sigma\Pi)^{(\Delta+2)}$-circuit of size at most $10s_1s_2D^5 + s_2D \le 11s_1s_2D^5$ and its degree is at most $D^2 + 4D^2 = 5D^2$. Thus, a hitting set $\mathcal{H}$ for this class will contain a point $\tilde{\mathbf{a}} = (a_1, \dots, a_n, a_{n+1}) = (\mathbf{a}, a_{n+1})$ such that $P_{\text{num}}(\tilde{\mathbf{a}}) \cdot P_{\text{den}}(\mathbf{a})$ is non-zero. In particular, $\frac{P_{\text{num}}(\mathbf{a}, y)}{P_{\text{den}}(\mathbf{a})} = R_{f,g}(\mathbf{a}, y) \not\equiv 0$.

$\square$

5   Generating a list of candidate factors

Algorithm 1: Computing a list of candidate irreducible factors of multiplicity-1, degree d and computable by a depth Δ' circuit of size m, when the input polynomial has depth Δ

  Input : A size parameter m in unary; degree parameter $d \in \mathbb{N}$; a $(\Sigma\Pi)^{\Delta}$-circuit of size s, bit-complexity t, total degree D, computing a polynomial $f(\mathbf{x}, y) \in \mathbb{Q}[x_1, \ldots, x_n, y]$ that is monic in y.

  Output: A list $L = \{C_1, \ldots, C_r\}$ of algebraic circuits (with division gates) such that for every irreducible factor $g(\mathbf{x}, y)$ of $f(\mathbf{x})$ s.t. $g^2 \nmid f$ and g has a size m circuit of depth Δ', y-degree equal to d, there exists $i \in [r]$ such that $C_i \equiv g(\mathbf{x})$.

1 Set the list $L' = \varnothing$.

2 Compute hitting-set H_1 for $(n+1)$-variate $(\Sigma\Pi)^{(\max(\Delta,\Delta')+2)}$-circuits of size $(11smD^5)$ and degree $(5D^2)$ using Theorem A.4.

3 Let H_2 be the projection of the points in H_1 on the first n coordinates.

4 **for** $\mathbf{a} = (a_1, \ldots, a_n) \in H_2$ **do**

5 $\quad F(\mathbf{x}, y) := f(x_1 + a_1, \ldots, x_n + a_n, y)$

6 $\quad$ Factorise the polynomial $F(\mathbf{0}, y) \in \mathbb{Q}[y]$ into irreducible factors as

$$F(\mathbf{0}, y) = \sigma \cdot F_1(y)^{e_1} \cdots F_l(y)^{e_l}.$$

  $\quad$ where $0 \neq \sigma \in \mathbb{Q}$ and each $F_i(y)$ is monic.

7 $\quad$ **forall** F_i s.t. $e_i = 1$ **do**

8 $\quad\quad$ Set $\mathbb{K} := \frac{\mathbb{Q}[u]}{\langle F_i(u)\rangle}$.

9 $\quad\quad$ **if** $\partial_y F(\mathbf{0}, u) = 0$ **then**

10 $\quad\quad\quad$ Skip to the next F_i.

11 $\quad\quad$ Use Lemma 3.1 to obtain a circuit for $\Phi_k(\mathbf{x})$ such that $F(\mathbf{x}, \Phi_k) = 0 \bmod \langle \mathbf{x}\rangle^{k+1}$ for $k = 2D^2$.

12 $\quad\quad$ Use Lemma 3.2 to obtain a circuit (with divisions) C for the minimal polynomial $G(\mathbf{x}, y)$ for Φ_k modulo the ideal $\langle \mathbf{x}\rangle^{k+1}$.

13 $\quad\quad$ Apply the transformation $x_i \to x_i - a_i$ on C to get a circuit C'.

14 $\quad\quad$ Update $L' := L' \cup \{C'\}$.

15 **return** L'

Remark (Obtaining circuits over the base field $\mathbb{Q}$). *Although the above algorithm outputs circuits over algebraic extensions of $\mathbb{Q}$ of the form $\mathbb{K} = \frac{\mathbb{Q}[u]}{A(u)}$, they can be transformed syntactically to circuits over the base field $\mathbb{Q}$, with only a polynomially large blow-up in size, via standard techniques (Lemma B.8).* ◇

For simplicity, Algorithm 1 deals with the special case when the factors we care about have multiplicity one and a fixed degree d. Later, we will describe the final algorithm that will iterate over the values of the degree and multiplicity parameters, and run Algorithm 1 as a subroutine.

We now bound the running time of the above algorithm, and prove its correctness. In the following subsection, we use this algorithm to prove the main theorem Theorem 1.1.

5.1 Proof of correctness of Algorithm 1

Theorem 5.1 (Correctness of Algorithm 1). *Let $\mathbb{Q}$ be the field of rational numbers and $\varepsilon > 0$, $\Delta, \Delta' \in \mathbb{N}$ be arbitrary constants. Let $f(\mathbf{x}, y) \in \mathbb{Q}[\mathbf{x}, y]$ be any polynomial that is monic in y, is computed by a $(\Sigma\Pi)^{(\Delta)}$-circuit of size s, bit-complexity B, degree D and let g be an irreducible factor of multiplicity one of f with y-degree exactly d such that g is computable by a $(\Sigma\Pi)^{(\Delta')}$-circuit of size m.*

If Algorithm 1 is invoked on input m, d, a circuit for f with size s, bit-complexity t, D, then the output is a list L of circuits of size $\mathrm{poly}(s, m, D)$ and bit-complexity $\mathrm{poly}(s, t, m, D)$ such that L contains at least one circuit that computes g.

Moreover, the algorithm terminates in time $(O(smD^5)^{O(\tilde{\Delta})} \cdot n)^{O_\varepsilon((smD^7)^\varepsilon)} \cdot \mathrm{poly}(s, D, B)$, where $\tilde{\Delta} = \max(\Delta, \Delta') + 2$.

Proof. We start with the proof of correctness. Suppose $f = g \cdot h$ where g is one of the irreducible factors of interest. Let $R_{f,g}$ be the pseudo-resultant of f and g with respect to y. From Lemma 4.4, we get that there exists a point $(a_1, a_2, \ldots, a_n, a_{n+1})$ in the hitting set H_1 defined in the algorithm such that $R_{f,g}(a_1, \ldots, a_n, y) \not\equiv 0$. In other words, there is an $\mathbf{a} = (a_1, a_2, \ldots, a_n) \in H_2$ such that $R_{f,g}(\mathbf{a}, y) \not\equiv 0$; let us fix such an $\mathbf{a}$.

From this and the second item of Lemma 4.3, we get that there is a $u \in \overline{\mathbb{Q}}$ such that $g(\mathbf{a}, u) = 0$, $h(\mathbf{a}, u) \neq 0$ and $\partial_y(f)(\mathbf{a}, u) \neq 0$. If $G(\mathbf{x}, y) = g(\mathbf{x} + \mathbf{a}, y)$ and $H(\mathbf{x}, y) = h(\mathbf{x} + \mathbf{a}, y)$, this u must be a root of one of the F_i's obtained in Line 6. Hence, $F(\mathbf{0}, u) = f(\mathbf{a}, u) = 0$ and $\partial_y F(\mathbf{0}, u) = \partial_y f(\mathbf{a}, u) \neq 0$. We now satisfy the hypothesis for Newton Iteration (Lemma 3.1) and we can obtain a circuit for the approximate root Φ_k for F satisfying $\Phi_k(\mathbf{0}) = u$. Since $F(\mathbf{x}, \Phi_k) = G(\mathbf{x}, \Phi_k) \cdot H(, \Phi_k) = 0 \bmod \langle\mathbf{x}\rangle^{k+1}$ and $H(\mathbf{0}, \Phi_k(\mathbf{0})) = h(\mathbf{a}, u)$ is a nonzero scalar in $\mathbb{K}$, we have that $G(\mathbf{x}, \Phi_k) = 0 \bmod \langle\mathbf{x}\rangle^{k+1}$. Therefore, Lemma 3.2 will yield a circuit (with divisions) for G. Undoing the initial translate via $x_i \to x_i - a_i$ yields a circuit for $g(\mathbf{x}, y)$.

From the description of the algorithm, we get that time complexity is at most

$$T_0 + |H_2| \cdot D \cdot (T_1 + T_2 + T_3)$$

where, T_0 is the time taken to computing an appropriate hitting set in Line 2, $T_1 \leq \mathrm{poly}(D, B)$ is the time taken to factorize the polynomial $F(\mathbf{0}, y)$ (Theorem B.1), $T_2 \leq \mathrm{poly}(s, D, B)$ is the time taken for computing the approximate root (Lemma 3.1) and computing the minimal polynomial (Lemma 3.2). The size of H_2 is at most the size of H_1, and Theorem A.4 tells us that $|H_1| \leq ((11smD^5)^{O(\tilde{\Delta})} \cdot n)^{O_\varepsilon((44smD^7)^\varepsilon)}$ (where $\tilde{\Delta} = \max(\Delta, \Delta') + 2$); moreover, Theorem A.4 tells us that T_0, the time-complexity of computing H_1 at Line 2, has the same expression as the size of H_1. Thus, the total running time is $((11smD^5)^{O(\tilde{\Delta})} \cdot n)^{O_\varepsilon((55smD^7)^\varepsilon)} \cdot \mathrm{poly}(s, D, B)$. The bit-complexity bound follows as the only nontrivial constant added to each circuit is the algebraic number u whose minimal polynomial is one of the factors of $F(\mathbf{0}, y)$ and hence has small bit-complexity as well. $\qed$

5.2 Proof of Theorem 1.1

We now describe our final algorithm, and its analysis to complete the proof of Theorem 1.1.

Algorithm 2: Computing a list of candidate irreducible factors computable by a depth Δ' circuit of size m, when the input polynomial has depth Δ

    Input : A size parameter m in unary; a $(\Sigma\Pi)^{\Delta}$-circuit of size s, bit-complexity t, degree D,
                computing a polynomial $f(\mathbf{x}) \in \mathbb{Q}[x_1, \ldots, x_n]$.
    Output: A list $L = \{C_1, \ldots, C_r\}$ of algebraic circuits (with division gates) such that for every
                irreducible factor $g(\mathbf{x})$ of $f(\mathbf{x})$ s.t. g has a size m circuit of depth Δ', there exists $i \in [r]$ such
                that $C_i \equiv g(\mathbf{x})$.

1 Set the output list $L' = \emptyset$.
2 Compute hitting-set H_1 for n-variate $(\Sigma\Pi)^{(\Delta)}$-circuits of size (sD^3) and degree D using
   Theorem A.4.
3 for $\alpha \in H_1$ **do**
4   Define $\tilde{f}(\mathbf{x}, y) := f(\mathbf{x} + \alpha \cdot y) = f(x_1 + \alpha_1 y, \ldots, x_n + \alpha_n y)$
5   **for** $i = 0 \ldots D - 1$ **do**
6   Compute a circuit C_i for the polynomial $\tilde{f}_i(\mathbf{x}, y) := \frac{\partial^i \tilde{f}}{\partial y^i}$
7   **for** $d = 1 \ldots D - 1$ **do**
8   Compute a list $L_{i,d}$ of candidate degree d irreducible factors of multiplicity one of
                $\tilde{f}_i(\mathbf{x}, y)$ using Algorithm 1
9   Set $L' := L' \cup L_{i,d}$

10 Let L be the list of circuits obtained by setting y to 0 in circuits in L'.
11 return L

Theorem 5.2 (Correctness of Algorithm 2). *Let* $\mathbb{Q}$ *be the field of rational numbers and* $\varepsilon > 0, \Delta, \Delta' \in \mathbb{N}$ *be arbitrary constants. Let* $f(\mathbf{x}) \in \mathbb{Q}[x_1, \ldots, x_n]$ *be a polynomial computed by a* $(\Sigma\Pi)^{(\Delta)}$-*circuit of size s, bit-complexity t, degree D and let g be an irreducible factor of f such that g is computable by a* $(\Sigma\Pi)^{(\Delta')}$-*circuit of size m.*

If Algorithm 2 is invoked on input m, a circuit for f with size s, bit-complexity t, D, then the output is a list L of circuits of size $\mathrm{poly}(s, m, D)$ *and bit-complexity* $\mathrm{poly}(s, t, m, D)$ *such that L contains at least one circuit that computes g. Moreover, the algorithm terminates in time* $\left(\left((O(smD^8))^{O(\tilde{\Delta})} \cdot n \right) \right)^{O_{\varepsilon}(smD^{10})^{\varepsilon}}$ · $\mathrm{poly}(s, D, T)$, *where* $\tilde{\Delta} = \max\{\Delta, \Delta'\} + 2$.

Proof. Let D be the total degree of f. If we view $f(\mathbf{x} + \alpha \cdot y)$ as a formal polynomial in y, we get that the coefficient of y^D in it is equal to the evaluation of the degree D homogeneous component of f on input (α). Moreover, from Lemma A.1, we get that this homogeneous component has a $(\Sigma\Pi)^{(\Delta)}$-circuit of size at most sD^3. Therefore, there exists an α in the hitting set H_1 for which the polynomial $\tilde{f}(\mathbf{x}, y) = f(\mathbf{x} + \alpha \cdot y)$ has y degree equal to D, and hence (up to multiplication by a non-zero field constant) is monic in y.

We will focus on an arbitrary fixed irreducible factor $g \mid f$ and prove that there exists a circuit in the final output list L that computes g. Let i^* be the multiplicity of g in f. Let the degree of g be d. Then, $\tilde{g}(\mathbf{x}, y) = g(\mathbf{x} + \boldsymbol{\alpha} \cdot y)$ will be a degree-d factor of multiplicity one of $\tilde{f}_{i^*-1}(\mathbf{x}, y) := \frac{\partial^i \tilde{f}}{\partial y^i}$. By Lemma A.3, $\tilde{f}_{i^*-1}(\mathbf{x}, y)$ has a $(\Sigma\Pi)^{(\Delta)}$-circuit of size at most sD^3. Moreover, the transformation $\mathbf{x} \mapsto \mathbf{x} + \boldsymbol{\alpha} \cdot y$ maintains the irreducibility of g by Lemma B.7. Thus, in the $(i^* - 1)^{th}$ iteration of Line 5 and the dth iteration of Line 7, Line 8 will compute a list of candidate factors that will include $\tilde{g}$ as guaranteed by Theorem 5.1. Setting $y = 0$ (which can be done because Lemma 3.2 guarantees that only the $\mathbf{x}$ variables show up in the denominators) to go from the list L' to L will give us a list that contains $\tilde{g}(\mathbf{x}, 0) = g$.

From the description of the algorithm, the running time of Algorithm 2 is at most

$$T_0 + |H_1| \cdot D^2 \cdot T_1 \cdot \text{poly}(s, D, t)$$

where T_0 is the time taken to compute the hitting set H_1, T_1 is the time taken to invoke Algorithm 1 in Line 8, H_1 is the hitting set computed in Line 2 using Theorem A.4, the D^2 factor is to account for the two loops in Line 5 and Line 7, and the $\text{poly}(s, D, t)$ factor is to account for the rest of the steps such as computing $\tilde{f}$ from f, computing circuits for derivatives, obtaining a list of circuits by setting $y = 0$, etc. T_1 is the running time of the first algorithm on $\tilde{f}_{i^*-1}$ which has a $(\Sigma\Pi)^{(\Delta)}$-circuit of size $s' = sD^3$. Thus, from Theorem 5.1, T_1 is at most $((11smD^8)^{O(\tilde{\Delta})} \cdot n)^{O_\varepsilon((55smD^{10})^\varepsilon)} \cdot \text{poly}(s, D, t)$. Applying Theorem A.4 for a $(\Sigma\Pi)^{(\Delta)}$-circuit of size sD^3 and degree D tells us that $((sD^3)^{O(\Delta)} \cdot n)^{O_\varepsilon((sD^4)^\varepsilon)}$ will be a bound on both T_0 as well as the size of H_1. Thus, the final time complexity is $\left((11smD^8)^{O(\tilde{\Delta})} \cdot n\right)^{O_\varepsilon(55smD^{10})^\varepsilon} \cdot \text{poly}(s, D, T)$. The bit-complexity bound follows from Theorem 5.1. $\square$

References

[BSV20] Vishwas Bhargava, Shubhangi Saraf, and Ilya Volkovich. Deterministic Factorization of Sparse Polynomials with Bounded Individual Degree. *J. ACM*, 67(2):8:1–8:28, 2020.

[Bür00] Peter Bürgisser. *Completeness and Reduction in Algebraic Complexity Theory*, volume 7 of *Algorithms and computation in mathematics*. Springer, 2000.

[DSY09] Zeev Dvir, Amir Shpilka, and Amir Yehudayoff. Hardness-Randomness Tradeoffs for Bounded Depth Arithmetic Circuits. *SIAM J. Comput.*, 39(4):1279–1293, 2009.

[For15] Michael A. Forbes. Deterministic Divisibility Testing via Shifted Partial Derivatives. In *Proceedings of the 2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS)*, FOCS '15, page 451–465, USA, 2015. IEEE Computer Society.

[FS15]   Michael A. Forbes and Amir Shpilka. Complexity Theory Column 88: Challenges in Polynomial Factorization1. *SIGACT News*, 46(4):32–49, dec 2015.

[GS99]   Venkatesan Guruswami and Madhu Sudan. Improved decoding of Reed-Solomon and algebraic-geometry codes. *IEEE Transactions on Information Theory*, 45(6):1757–1767, 1999.

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

[KI04]   Valentine Kabanets and Russell Impagliazzo. Derandomizing Polynomial Identity Tests Means Proving Circuit Lower Bounds. *Computational Complexity*, 13(1-2):1–46, 2004. Preliminary version in the *35th Annual ACM Symposium on Theory of Computing (STOC 2003)*.

[KRS23]   Mrinal Kumar, Varun Ramanathan, and Ramprasad Saptharishi. Deterministic Algorithms for Low Degree Factors of Constant Depth Circuits. *CoRR*, abs/2309.09701, 2023. Pre-print available at arXiv:2309.09701.

[KSS14]   Neeraj Kayal, Chandan Saha, and Ramprasad Saptharishi. A super-polynomial lower bound for regular arithmetic formulas. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC 2014)*, pages 146–153, 2014. Pre-print available at eccc:TR13-091.

[KSS15]   Swastik Kopparty, Shubhangi Saraf, and Amir Shpilka. Equivalence of Polynomial Identity Testing and Polynomial Factorization. *Computational Complexity*, 24(2):295–331, 2015. Preliminary version in the *29th Annual IEEE Conference on Computational Complexity (CCC 2014)*.

[KT88]   Erich L. Kaltofen and Barry M. Trager. Computing with Polynomials Given By Black Boxes for Their Evaluation: Greatest Common Divisors, Factorization, Separation of Numerators and Denominators. In *29th Annual Symposium on Foundations of Computer Science, White Plains, New York, USA, 24-26 October 1988*, pages 296–305. IEEE Computer Society, 1988.

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

[LST21]   Nutan Limaye, Srikanth Srinivasan, and Sébastien Tavenas. Superpolynomial Lower Bounds Against Low-Depth Algebraic Circuits. In *Proceedings of the 62nd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2021)*, pages 804–814. IEEE,

2021. Preliminary version in the Electronic Colloquium on Computational Complexity (ECCC), Technical Report TR21-081.

[Sap15]   Ramprasad Saptharishi. A survey of lower bounds in arithmetic circuit complexity. Github survey, 2015.

[ST21]   Amit Sinhababu and Thomas Thierauf. Factorization of Polynomials Given by Arithmetic Branching Programs. *Comput. Complex.*, 30(2):15, 2021.

[Sud97]   Madhu Sudan. Decoding of Reed Solomon Codes beyond the Error-Correction Bound. *J. Complexity*, 13(1):180–193, 1997.

[SV10]   Amir Shpilka and Ilya Volkovich. On the Relation between Polynomial Identity Testing and Finding Variable Disjoint Factors. In *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010, Bordeaux, France, July 6-10, 2010, Proceedings, Part I*, volume 6198 of *Lecture Notes in Computer Science*, pages 408–419. Springer, 2010.

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

A   Preliminaries

A.1   Standard preliminaries using interpolation

Lemma A.1 (Univariate interpolation (Lemma 5.3 [Sap15])). *Let $f(x) = f_0 + f_1 x + \cdots + f_d x^d$ be a univariate polynomial of degree at most d. Then, for any $0 \leq r \leq d$ and there are[3] field constants $\alpha_0, \ldots, \alpha_d$ and $\beta_{r0}, \ldots, \beta_{rd}$ such that*

$$f_r = \beta_{r0} f(\alpha_0) + \cdots + \beta_{rd} f(\alpha_d).$$

Furthermore, the bit-complexity of all field constants is bounded by $\mathrm{poly}(d)$. □

Lemma A.2 (Computing homogeneous components (Lemma 5.4 [Sap15])). *Let $f \in \mathbb{Q}[\mathbf{x}]$ be an n-variate degree d polynomial. Then, for an $0 \leq i \leq d$, there are field constants $\alpha_0, \ldots, \alpha_d$ and β_{i0}, β_{id} of bit-complexity* $\mathrm{poly}(d)$ *such that*

$$\mathrm{Hom}_i(f) = \beta_{i0} f(\alpha_0 \cdot \mathbf{x}) + \cdots + \beta_{id} f(\alpha_d \cdot \mathbf{x}).$$

In particular, if f is computable by $(\Sigma\Pi)^{(k)}$-formulas of size / bit-complexity at most s then $\mathrm{Hom}_i(f)$ is computable by $(\Sigma\Pi)^{(k)}$-formulas of size / bit-complexity at most $\mathrm{poly}(s, d)$.

---

[3]In fact, for any choice of distinct $\alpha_0, \ldots, \alpha_d$, there are appropriate $\beta_{r0}, \ldots, \beta_{rd}$ satisfying the equation. If the α_i's are chosen to have small bit-complexity, we can obtain a $\mathrm{poly}(d)$ bound on the bit-complexity of the associated β_{ri}'s.

Lemma A.3 (Computing partial derivatives in one variable). *Let $f \in \mathbb{Q}[\mathbf{x}]$ be an n-variate degree d polynomial. Then, for an $0 \leq r \leq d$, there are field elements α_i's and β_{ij}'s in $\mathbb{Q}$ of bit-complexity $\mathrm{poly}(d)$ such that*

$$\frac{\partial^r f}{\partial x_1^r} = \sum_{i=0}^{d} x_1^i \cdot (\beta_{i0} f(\alpha_0, x_2, \ldots, x_n) + \cdots + \beta_{id} f(\alpha_d, x_2, \ldots, x_n))$$

In particular, if f is computable by $(\Sigma\Pi)^{(k)}$-formulas of size / bit-complexity at most s then $\frac{\partial^r f}{\partial x_1^r}$ is computable by $(\Sigma\Pi)^{(k)}$-formulas of size at most $8sd^3$ and bit-complexity at most $\mathrm{poly}(s, d)$.

Proof. We may consider the polynomial f as a univariate in x_1, and extract each coefficient of x_1^i using Lemma A.1 and recombine them to get the appropriate partial derivative. That justifies the claimed expression.

As for the size, note that multiplying a $(\Sigma\Pi)^{(k)}$-formula of size s by x_1^i, by using distributivity of the top addition gate, results in a $(\Sigma\Pi)^{(k)}$-formula of size at most $s \cdot d$. Thus, the overall size of the above expression for the partial derivative is at most $8sd^3$. $\qquad\square$

A.2 Deterministic PIT for constant-depth circuits

Theorem A.4 (PIT for constant-depth formulas (modification of Corollary 6 [LST21])). *Let $\varepsilon > 0$ be a real number and $\mathbb{F}$ be a field of characteristic 0. Let C be an algebraic formula of size and bit-complexity $s \leq \mathrm{poly}(n)$, depth $k = o(\log\log\log n)$ computing a polynomial on n variables, then there is a deterministic algorithm that can check whether the polynomial computed by C is identically zero or not in time $(s^{O(k)} \cdot n)^{O_\varepsilon((sD)^\varepsilon)}$.*

B   Building blocks

B.1   Univariate factorization over rational numbers

The following classical theorem of Lenstra, Lenstra and Lovász gives us an efficient algorithm for factoring univariate polynomials over the field of rational numbers.

Theorem B.1 (Factorizing polynomials with rational coefficients [LLL82, vzGG13]). *Let $f \in \mathbb{Q}[x]$ be a monic polynomial of degree d. Then there is a deterministic algorithm computing all the irreducible factors of f that runs in time $\mathrm{poly}(d, t)$, where t is the maximum bit-complexity of the coefficients of f.*

B.2 Resultant

Definition B.2 (The Resultant). *Let $\mathcal{R}$ be a commutative ring. Given polynomials g and h in $\mathcal{R}[y]$, where:*

$$g(y) = g_0 + \cdots + y^d \cdot g_d$$
$$h(y) = h_0 + y \cdot h_1 + \cdots + y^D \cdot h_D$$

with g_d and $h_D \neq 0$ the Resultant *of g and h, denoted by $\mathrm{Res}_y(g,h)$, is the determinant of the $(D + d) \times (D + d)$ Sylvester matrix Γ of g and h, given by:*

$$\Gamma = \begin{bmatrix} h_0 & h_1 & \ldots & & h_D & & & \\ & \ddots & \ddots & & & \ddots & \ddots & \\ & & h_0 & h_1 & & & \ldots & h_D \\ g_0 & \ldots & & g_d & & & & \\ & g_0 & \ldots & & g_d & & & \\ & & \ddots & \ddots & & & \ddots & \\ & & & g_0 & \ldots & & & g_d \end{bmatrix}$$

$\diamond$

Lemma B.3 (Resultant and gcd (Corollary 6.20 [vzGG13])). *Let $\mathcal{R}$ be a unique factorization domain and $g, h \in \mathcal{R}[y]$ be non-zero polynomials. Then:*

$$\deg_y(\gcd(g,h)) \geq 1 \iff \mathrm{Res}_y(g,h) = 0$$

where $\gcd(g,h) \in \mathcal{R}[y]$ and $\mathrm{Res}_y(g,h) \in \mathcal{R}$.

Moreover, there exist polynomials $A, B \in \mathcal{R}[y]$ such that $\mathrm{Res}_y(g,h) = Ag + Bh$.

B.3 Irreducible polynomials in the field of fractions of a UFD

Lemma B.4 (Gauss' Lemma (Corollary 6.10 [vzGG13])). *Let R be a unique factorization domain with field of fractions K. Suppose a polynomial $f \in R[y]$ is monic in y. Then f is irreducible in $K[y]$ if and only if f is irreducible in $R[y]$.*

As a corollary, the factorization of any monic polynomial f into its irreducible factors in $R[y]$ is exactly the factorization of f into its irreducible factors in $K[y]$.

B.4 Newton iteration

Lemma 3.1 (Newton iteration). *Let $F(\mathbf{x}, y) \in \mathbb{F}[\mathbf{x}, y]$ be monic in y. Suppose $u \in \mathbb{F}$ such that*

$$F(\mathbf{0}, u) = 0,$$
$$(\partial_y F)(\mathbf{0}, u) \neq 0.$$

Then, for all $k \geq 0$, there is an approximate root *polynomial $\Phi_k \in \mathbb{F}[\mathbf{x}]$ that satisfies*

$$F(\mathbf{x}, \Phi_k) = 0 \bmod \langle \mathbf{x} \rangle^{k+1},$$
$$\Phi_k(\mathbf{0}) = u.$$

Furthermore, if we are provided F a circuit of size and bit-complexity s, and also the element $u \in \mathbb{F}$ with bit-complexity at most B, we can output an algebraic circuit for the approximate root $\Phi_k(\mathbf{x})$ of size at most poly(s, k) *and bit-complexity* poly(s, B, k).

Proof. The approximate roots are given by the following recursive definition:

$$\Phi_0 = u$$
$$\text{For all } k \geq 0, \quad \Phi_{k+1} = \Phi_k - \frac{F(\mathbf{x}, \Phi_k)}{\partial_y F(\mathbf{0}, u)}.$$

It is clear from the above definition that Φ_{k+1} is a polynomial in $\mathbf{x}$ and its circuit complexity is at most an additive $O(s)$ larger than the circuit size of Φ_k. Since the only constant introduced in the circuit is $(\partial_y F(\mathbf{0}, u))^{-1}$, the bit-complexity is bounded by an additive poly(s, B).

We now show that Φ_{k+1} satisfies the requirement by induction (the base case of Φ_0 is trivial). By Taylor expansion around $(\mathbf{x}, \Phi_k)$, if $z \in \langle \mathbf{x} \rangle^{k+1}$, we have

$$F(\mathbf{x}, \Phi_k + z) = F(\mathbf{x}, \Phi_k) + z \cdot \partial_y F(\mathbf{x}, \Phi_k)$$
$$+ \left(\frac{z^2}{2!} \right) \partial_{y^2} F(\mathbf{x}, \Phi_k) + \cdots$$
$$= F(x, \Phi_k) + z \cdot \partial_y F(\mathbf{x}, \Phi_k) \bmod \langle \mathbf{x} \rangle^{k+2} \qquad (\text{since } z^2 \in \langle \mathbf{x} \rangle^{2(k+1)} \subseteq \langle \mathbf{x} \rangle^{k+2}).$$

Furthermore, since $z \in \langle \mathbf{x} \rangle^{k+1}$, we have that

$$z \cdot \partial_y F(\mathbf{x}, \Phi_k) = z \cdot \partial_y F(\mathbf{0}, \Phi_k(\mathbf{0})) \bmod \langle \mathbf{x} \rangle^{k+2}$$

since the other terms from the second multiplicand only contribute higher degree terms in $\mathbf{x}$. Thus,

$$F(\mathbf{x}, \Phi_k + z) = F(x, \Phi_k) + z \cdot \partial_y F(\mathbf{0}, u) \bmod \langle \mathbf{x} \rangle^{k+2},$$
$$= 0 \bmod \langle \mathbf{x} \rangle^{k+2} \quad \text{when } z = -\frac{F(x, \Phi_k)}{\partial_y F(\mathbf{0}, u)},$$
$$\implies F(\mathbf{x}, \Phi_{k+1}) = 0 \bmod \langle \mathbf{x} \rangle^{k+2}. \qquad \square$$

Lemma B.5 (Mininum polynomials of approximate roots). *Let $\mathbb{F}$ be a field of characteristic zero. Let $\varphi(\mathbf{x}) \in \mathbb{F}[\mathbf{x}]$ and $G(\mathbf{x}, y) \in \mathbb{F}[\mathbf{x}, y]$ be polynomials such that G is irreducible of y-degree d_y, $\mathbf{x}$-degree d, is*

monic in y and satisfies

$$G(\mathbf{x}, \varphi) \equiv 0 \mod \langle \mathbf{x} \rangle^k,$$

for some natural number k greater than $2d_y d$. If $H(\mathbf{x}, y) \in \mathbb{F}[\mathbf{x}, y]$ is a non-zero polynomial of y-degree at most d_y, $\mathbf{x}$-degree d, is monic in y and satisfies

$$H(\mathbf{x}, \varphi) \equiv 0 \mod \langle \mathbf{x} \rangle^k,$$

then, H must be equal to G.

Proof. We consider both G and H as univariates in y with coefficients in $\mathbb{F}(\mathbf{x})$. Let R be their resultant with respect to the variable y. Clearly, from the definition of the resultant, we have that R is a polynomial in $\mathbb{F}[\mathbf{x}]$ of degree at most $2d_y d$. If R is identically zero, then we have from Lemma B.3 that G and H have a non-trivial GCD as polynomials in $\mathbb{F}(\mathbf{x}, y)$. But then, since G and H are monic in y, it follows from Lemma B.4 that they must have a non-trivial GCD in $\mathbb{F}[\mathbf{x}, y]$. Now, since G is irreducible, this can happen only if G divides H. Since the degree of H is at most d, we get that they must be a non-zero constant multiple of each other. But they are both monic in y and hence they must be equal to each other. Thus, if R is identically zero, then we are done. It now remains to show that R is indeed identically zero.

We have from Lemma B.3 that there exist polynomials $A(\mathbf{x}, y), B(\mathbf{x}, y)$ such that

$$A(\mathbf{x}, y)G(\mathbf{x}, y) + B(\mathbf{x}, y)H(\mathbf{x}, y) = R(\mathbf{x}).$$

Now, plugging in $y = \varphi(\mathbf{x})$ on both sides of the above equation, we get

$$A(\mathbf{x}, \varphi(\mathbf{x}))G(\mathbf{x}, \varphi(\mathbf{x})) + B(\mathbf{x}, \varphi(\mathbf{x}))H(\mathbf{x}, \varphi(\mathbf{x})) = R(\mathbf{x}).$$

But from the hypothesis of the lemma, we know that $G(\mathbf{x}, \varphi) \equiv 0 \mod \langle \mathbf{x} \rangle^k$ and $H(\mathbf{x}, \varphi) \equiv 0 \mod \langle \mathbf{x} \rangle^k$. Moreover, $k > 2d_y d$. Thus, the left-hand side of the identity above vanishes modulo $\langle \mathbf{x} \rangle^{2d_y d+1}$. So, the resultant R vanishes modulo $\langle \mathbf{x} \rangle^{2d_y d+1}$. But since its degree is less than $2d_y d$, this means that R is identically zero as a polynomial in $\mathbb{F}[\mathbf{x}]$. This completes the proof of the lemma. □

Lemma 3.2 (Computing minimal polynomials of approximate roots). *Let $\Phi_k(\mathbf{x}) \in \mathbb{F}[\mathbf{x}]$ be provided by an algebraic circuit of size and bit-complexity at most s. Suppose there exists a polynomial $G(\mathbf{x}, y) =$*

$\sum_{i=0}^{d} G_i(\mathbf{x}) y^i$ *that is monic in y and irreducible satisfying*

$$G(\mathbf{x}, \Phi_k(\mathbf{x})) = 0 \bmod \langle \mathbf{x} \rangle^{k+1},$$

$$\deg G \le D,$$

$$\deg_y G = d,$$

$$k \ge 2Dd.$$

Then, given d and the circuit for Φ_k, for every $i \in [d]$, we can compute an algebraic circuit (with only $\mathbf{x}$ variables and no y), with divisions, of size and bit-complexity at most $\mathrm{poly}(s, d, D)$ for the polynomial $G_i(\mathbf{x})$. In particular, we can compute a circuit of size and bit-complexity $\mathrm{poly}(s, d, D)$ for $G(\mathbf{x}, y)$.

Proof. By Lemma B.5, we know that G must be the lowest degree polynomial of degree at most D and y-degree at most d that satisfies $G(\mathbf{x}, \Phi_k) = 0 \bmod \langle \mathbf{x} \rangle^{k+1}$. We shall express this as a linear system in terms of the coefficients of G.

$$
\begin{aligned}
G(\mathbf{x}, y) &= G_0 + G_1 \cdot y + G_2 \cdot y^2 + \cdots + G_d y^d \\
&= \sum_{i=0}^{d} G_i y^i \\
&= \sum_{i=0}^{d} \left(\sum_{j=0}^{D} G_{ij} \right) \cdot y^i
\end{aligned}
$$

where each G_{ij} is the degree-j homogeneous part of the coefficient of y^i in G. We will treat each G_{ij} as an indeterminate and express the condition $G(\mathbf{x}, \Phi_k(\mathbf{x}))$ as a system of linear equations in G_{ij}'s.

For $r = 0, \ldots, k$, let $\Phi_k^r = \Phi_k^{(r,0)} + \cdots + \Phi_k^{(r,k)}$ where $\Phi_k^{(r,\ell)}$ is the degree-ℓ homogeneous part of Φ_k^r. Given a circuit of size s for Φ_k, we can obtain $\mathrm{poly}(s)$-sized circuits for each of $\Phi_k^{(r,\ell)}$.

$$
\begin{aligned}
G(\mathbf{x}, \Phi_k) &= \sum_{i=0}^{d} \sum_{j=0}^{D} G_{ij} \Phi_k^i = \sum_{i=0}^{d} \sum_{j=0}^{D} \sum_{\ell=0}^{k} G_{ij} \Phi_k^{(i,\ell)} \\
&= \sum_{m=0}^{D+k} \left(\sum_{i=0}^{d} \sum_{j=0}^{D} G_{ij} \Phi_k^{(i,m-j)} \right)
\end{aligned}
$$

where the last equality is just grouping the terms in terms of the $\mathbf{x}$-degree. Thus, the condition

that $G(\mathbf{x}, \Phi_k(\mathbf{x})) = 0 \bmod \langle \mathbf{x} \rangle^{k+1}$ can be expressed as

$$\left(\sum_{i=0}^{d} \sum_{j=0}^{D} G_{ij} \Phi_k^{(i,m-j)} \right) = 0 \quad \text{for all } m = 0, \dots, k,$$

$$G_{d,0} = 1,$$

$$G_{d,j} = 0 \quad \text{for all } j > 0.$$

From the uniqueness of solution, it follows that the column rank of the constraint matrix M of this linear system is full. Now, if the coefficient vector of G is a solution of $M \cdot \mathbf{z} = \mathbf{b}$, then it is also a solution of the linear system $M^\dagger \cdot M\mathbf{z} = M^\dagger \cdot \mathbf{b}$, where $M^\dagger$ denotes the conjugate transpose of M. Moreover, from Lemma B.6, we have that $M^\dagger \cdot M$ is a square matrix of full rank. Thus, this unique solution of $M^\dagger \cdot M\mathbf{z} = M^\dagger \cdot b$ can be written in closed form as $(M^\dagger M)^{-1}(M^\dagger \cdot \mathbf{b})$.

Now, by expressing the inverse $(M^\dagger M)^{-1} = \frac{\mathrm{adj}(M^\dagger M)}{\det(M^\dagger M)}$, where adj refers to the 'adjugate matrix', we get a circuit of size (and bit-complexity) $\mathrm{poly}(s, d, D)$ for each G_{ij}, and hence, for $G(\mathbf{x}, y) = \sum_{i=0}^{d} \left(\sum_{j=0}^{D} G_{ij} \right) \cdot y^i$. $\qquad \square$

B.5 A basic linear algebra fact

We use the following basic fact in the proof of correctness of our algorithm.

Lemma B.6. *Let M be an $r \times c$ matrix with entries in $\mathbb{C}[\mathbf{x}]$ such that $r \geq c$ and the rank of M over the field $\mathbb{C}(\mathbf{x})$ equals c.*

Then, the rank of the $c \times c$ matrix $M^\dagger M$ over the field $\mathbb{C}(\mathbf{x})$ is also equal to c. Here, $M^\dagger$ denotes the conjugate transpose of M.

Proof. Since the rank of M over $\mathbb{C}(\mathbf{x})$ equals c, there is a substitution $\mathbf{a} \in \mathbb{C}^n$ for the variables such that $N = M(\mathbf{a})$ has rank c over $\mathbb{C}$. To show that $M^\dagger M$ is full rank over $\mathbb{C}(\mathbf{x})$ it suffices to show that $N^\dagger N$ is full rank over $\mathbb{C}$. We do this by arguing that the kernel of $N^\dagger N$ does not contain a non-zero vector.

Let $u \in \mathbb{C}^n$ be an arbitrary non-zero vector. We now argue that any such u cannot be in the kernel of $N^\dagger N$. To see this, consider the inner product $\langle u, N^\dagger N u \rangle$ where $\langle u, v \rangle := u^\dagger v$. Clearly, this equals $\langle Nu, Nu \rangle$, which equals the square of the ℓ_2 norm of the vector Nu. Since N has full column rank and u us non-zero, we have that Nu is a non-zero vector in $\mathbb{C}^n$, and thus its ℓ_2 norm is non-zero. Thus, $N^\dagger Nu$ must be non-zero, and $N^\dagger N$ must be full rank. $\qquad \square$

B.6 Irreducibility under a shift of variables

Since the final step of solving a linear system after Newton iteration works only when the candidate factor is irreducible, we need to ensure that the initial shift of variables does not affect the irreducibility of the factors; this is guaranteed by the following lemma.

Lemma B.7. *Let $\mathbb{F}$ be a field and $g(\mathbf{x}) \in \mathbb{F}[\mathbf{x}]$ be an n-variate irreducible polynomial. Then, for every $\mathbf{a} \in \mathbb{F}^n$, the polynomial $G(\mathbf{x}, y) := g(\mathbf{x} + \mathbf{a} \cdot y)$ is also irreducible.*

Proof. We will interpret the polynomial $g(\mathbf{x}) \in \mathbb{F}[\mathbf{x}]$ as $g(\mathbf{x}, y) \in \mathbb{F}[\mathbf{x}, y]$ with the property that it does not depend on the variable y. The mapping $(\mathbf{x}, y) \mapsto (\mathbf{x} + \alpha \cdot y, y)$ is an invertible linear transformation M on the vector of variables $(x_1, \ldots, x_n, y) = (\mathbf{x}, y)$.

We shall prove the contrapositive of the claim: if the polynomial $G(\mathbf{x}, y) := g(M(\mathbf{x}, y))$ has a non-trivial factorization, then so does g. Formally, if $g(M(\mathbf{x}, y)) = g_1(\mathbf{x}, y) \cdot g_2(\mathbf{x}, y)$ for some non-constant polynomials g_1 and g_2, then $g(\mathbf{x}, y) = g_1(M^{-1}(\mathbf{x}, y)) \cdot g_2(M^{-1}(\mathbf{x}, y))$ such that $g_1(M^{-1}(\mathbf{x}, y))$ and $g_2(M^{-1}(\mathbf{x}, y))$ are also non-constant polynomials. Observe that the operation of taking a product of two polynomials commutes with the operation of applying a linear transformation on the variables; it is easy to see for monomials, and it follows for polynomials by linearity of M. Thus, applying M^{-1} on both sides of $g(M[\mathbf{x}, y]) = g_1(\mathbf{x}, y) \cdot g_2(\mathbf{x}, y)$ gives us that $g(\mathbf{x}, y) = g_1(M^{-1}(\mathbf{x}, y)) \cdot g_2(M^{-1}(\mathbf{x}, y))$. Here, $g_1(M^{-1}(\mathbf{x}, y))$ and $g_2(M^{-1}(\mathbf{x}, y))$ will be non-constants since M is invertible. Thus, if G has a non-trivial factorization, then so does g. $\square$

B.7 Converting a circuit over a number field to a circuit over $\mathbb{Q}$

Lemma B.8 (Constructing a circuit over the base field). *Given as input an irreducible polynomial $A(u)$ of degree r and bit-complexity B, and a C, with divisions, of size and formal-degree at most s over a field $\frac{\mathbb{Q}[u]}{A(u)}$ that computes a polynomial $g(\mathbf{x}) \in \mathbb{Q}[\mathbf{x}]$. There is a deterministic algorithm that can output another circuit C' with size $\mathrm{poly}(s, r, B)$ over the field $\mathbb{Q}$ computing the polynomial $g(\mathbf{x})$.*

Proof. For any polynomial $p(u) \in \mathbb{Q}[u]$, we will use $(p \bmod A(u))$ to denote the unique polynomial $p'(u)$ of degree less than $A(u)$ such that $A(u) \mid p(u) - p'(u)$.

We may assume without loss of generality[4] that the input circuit C only has a division gate at the root, i.e. it computes g in the form $g(\mathbf{x}) = \frac{C_{\mathrm{num}}(\mathbf{x})}{C_{\mathrm{den}}(\mathbf{x})}$ where C_{num} and C_{den} are circuits without divisions over $\mathbb{Q}[u]/A(u)$. Let s, d be bounds on the size and degree of the circuits, respectively, of C_{num} and C_{den}.

By interpreting u as a formal variable, let $C'_{\mathrm{num}}, C'_{\mathrm{den}} \in Q[\mathbf{x}]$ be the resulting polynomials (so

---

[4]The standard 'division-elimination' approach keeps tracks of numerators and denominators in gate of the original circuit. This transformation only increases the size by a polynomial factor.

that $C'_{num} \bmod A(u) = C_{num}$ and similarly for C'_{den}). We may write C'_{num}, C'_{den} as

$$C'_{num} = C'^{(0)}_{num} + C'^{(1)}_{num}u + \cdots + C'^{(d)}_{num}u^d \in \mathbb{Q}[\mathbf{x}, u]$$

$$C'_{den} = C'^{(0)}_{den} + C'^{(1)}_{den}u + \cdots + C'^{(d)}_{den}u^d \in \mathbb{Q}[\mathbf{x}, u].$$

$$C_{num} = (C'_{num} \bmod A(u)) = C^{(0)}_{num} + C^{(1)}_{num}u + \cdots + C^{(r-1)}_{num}u^{r-1}$$

$$= \sum_{i=0}^{d} C'^{(0)}_{num}(u^i \bmod A(u))$$

$$C_{den} = (C'_{den} \bmod A(u)) = C^{(0)}_{den} + C^{(1)}_{den}u + \cdots + C^{(r-1)}_{den}u^{r-1}$$

$$= \sum_{i=0}^{d} C'^{(0)}_{den}(u^i \bmod A(u))$$

Using Lemma A.1, we can obtain circuits for each $C'^{(i)}_{den}, C'^{(i)}_{den}$ of size at most $\mathrm{poly}(s, d)$, and thus we can compute circuits for $C^{(i)}_{num}$ and $C^{(i)}_{den}$ as well.

To 'invert' C_{den}, we consider indeterminates $b_0, b_1, \ldots, b_{r-1}$ such that

$$(C^{(0)}_{den} + \cdots + C^{(r-1)}_{den}u^{r-1}) \cdot (b_0 + b_1u + \cdots + b_{r-1}u^{r-1}) = 1 \bmod A(u),$$

which is once again a linear system over the indeterminate b_j's, with coefficients being linear combinations of the circuits $C^{(i)}_{den}$'s. We can therefore express the b_j's as efficient rational functions using the circuits $C^{(i)}_{den}$'s via Cramer's rule.

Finally, if $\frac{C_{num}}{C_{den}}$ was indeed a polynomial $g(\mathbf{x}) \in \mathbb{Q}[\mathbf{x}]$, we must have that

$$((C_{num} \cdot (b_0 + b_1u + \cdots b_{r-1}u^{r-1})) \bmod A(u)) = g(\mathbf{x}).$$

Thus, we can once again compute the LHS above and return the circuit for the coefficient of u^0 in the above expression. The size and bit-complexity bounds follow readily from the description. $\quad\square$