CHENNAI MATHEMATICAL INSTITUTE

MASTERS THESIS

Discovering the roots: Unifying and extending results on multivariate polynomial factoring in algebraic complexity

---

Supervisor:
Dr. Nitin SAXENA

Supervisor:
Dr. Partha MUKHOPADHYAY

Author:
Pranjal DUTTA

A thesis submitted in fulfillment of the requirements
for the degree of Master of Science

in the

Department of Computer Science
Chennai Mathematical Institute

Declaration of Authorship

I, Pranjal DUTTA, declare that this thesis titled, "Discovering the roots: Unifying and extending results on multivariate polynomial factoring in algebraic complexity" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a masters degree (from CMI) at IIT Kanpur.
- The thesis has been prepared without resorting to plagiarism.
- Where I have consulted the published work of others, this is always clearly attributed.
- I have acknowledged all main sources of help.
- The thesis has not been submitted elsewhere for a degree.

Signed: Pranjal Dutta

Date: 10 June, 2018

"We cannot solve our problems with the same thinking we used when we created them."

-Albert Einstein, Genius (2017-)

Abstract

Multivariate polynomial factoring is one of the most important problems in theoretical computer science with many important applications in various regimes including decoding of Reed-Solomon, Reed-Muller codes [GS98, Sud97], integer factoring [LLMP90], algebra isomorphism [KS06, IKRS12] and primary decomposition of polynomial ideals [GTZ88]. Algebraic circuit is a natural model to represent a multivariate polynomial compactly. Here, an algebraic complexity class contains (multivariate) polynomial families instead of languages. It is a natural question whether an algebraic complexity class is closed under factoring.

Famously, Kaltofen [Kal89] showed that VP (Valiant's P), the class of low degree polynomials with small circuit representation, is uniformly closed under factoring, i.e. for a given d degree n variate polynomial f of circuit size s, there exists a randomized $\text{poly}(snd)$-time algorithm that outputs its factor as a circuit whose size is bounded by $\text{poly}(snd)$. This fundamental result has several applications such as 'hardness versus randomness' in algebraic complexity [KI03, AV08, DSY09, AGS17], derandomization of Noether Normalization Lemma [Mul17], in the problem of circuit reconstruction [KS09, Sin16], and polynomial equivalence testing [Kay11]. It is still open whether low degree factor of high degree circuits of low complexity has low complexity or not, famously known as *Factor Conjecture* (due to Peter Bürgisser).

In this paper, we show that for an algebraic circuit $f(x_1, \ldots, x_n)$ of size s we prove that each factor has size at most a polynomial in: s and the degree of the squarefree part of f. Consequently, if f_1 is a $2^{\Omega(n)}$-hard polynomial then any nonzero multiple $\prod_i f_i^{e_i}$ is equally hard for *arbitrary* positive e_i's, assuming that $\sum_i \deg(f_i)$ is at most $2^{O(n)}$. This result also establishes the fact that when square free degree of the given polynomial is small, factor conjecture is true.

It is an old open question whether the class of $\text{poly}(n)$-sized formulas (resp. algebraic branching programs) is closed under factoring. We show that given a polynomial f of degree $\text{poly}(n)$ and formula (resp. ABP) size $n^{O(\log n)}$ we can find a similar size formula (resp. ABP) factor in randomized $\text{poly}(n^{O(\log n)})$-time. Consequently, if determinant requires $n^{\Omega(\log n)}$ size formula, then the same can be said about any of its nonzero multiples. We also use the same technique to derive the algebraic complexity of gcd of two polynomials.

As part of our proofs, we present a general frame work that shows that under random linear transformation, any generic multivariate polynomial can be factored into a power series ring. Hence, we conclude that it is enough to approximate those power series roots. Moreover, the factorization adapts well to circuit complexity analysis. We generalize it to a matrix recurrence (allRootsNI) that approximates all the roots simultaneously. These techniques together help us to make progress towards the old open problems;

supplementing the large body of classical results and concepts in algebraic circuit factorization (eg. Zassenhaus, J.NT 1969; Kaltofen, STOC 1985-7 & Bürgisser, FOCS 2001).

Lastly we discuss very briefly about sparsity bound on factors and discuss interesting consequences. Most of the content of the thesis can be found in the paper "Discovering the roots: Uniform closure results for algebraic classes under factoring" (accepted at STOC'2018), a joint work with my advisor Dr. Nitin Saxena and my colleague Amit Sinhababu.

Acknowledgements

Earning a degree is always a journey and I have been fortunate to have had the support and guidance of many throughout my journey at Chennai Mathematical Institute (CMI) and Indian Institute of Technology, Kanpur (IIT K). First and foremost, I would like to thank my advisor, Nitin Saxena. The past two years working with Nitin has been a deep learning curve for me. Words would not suffice to express my gratitude towards him, from hosting me at IIT Kanpur during my first year of masters to the constant doubt-clearing session of hours. In the coming years (during my PhD : I am thankful to him for agreeing to advise me for my PhD) I hope I can inculcate in me at least a part of the enthusiasm and dedication he has towards his work and research.

I'm deeply indebted to CMI, IMSC and IIT Madras for offering truly wonderful undergraduate and post graduate courses. Thanks to Jayalal Sarma (IIT Madras) for introducing Algebraic Complexity Theory. Thanks to Samir Datta (CMI), Jayalal Sarma (IIT Madras), Sourav Chakraborty (CMI), Rajat Mittal (IIT K) and Partha Mukhopadhyay (CMI) for many enlightening discussions which motivated me to continue research. I would like to thanks Prof. Madhavan Mukund and Prof. K. V. Subrahmanyam (CMI) for allowing me to spend MSc 1st yeat at IIT K, arranging all the necessary documents and to Partha for giving me the advise to work with Nitin during masters. Special thanks to Prof Emeritus Somenath Biswas for many helpful and insightful discussions, in general which kept me motivated and on edge during my stay at IIT K.

Besides excellent professors, CMI is endowed with a very efficient and affable office staff as well. I'm very grateful to them, specially Rajeswari for a completely hassle free stay both at CMI and IIT Kanpur.

I made quite a few friends in CMI and IIT Kanpur during my bachelors and masters. Life, both in CMI and IIT K was far more enjoyable because of them. Specially, I would like to thank Sumanta da, for engaging discussions on complexity, algebraic complexity and clearing all my doubts (and spiritual ideas :p) whenever I had one. Thanks to Amit da, my collaborator who has been extremely supportive (academically and administratively and for the innumerable treats :p) and patient during my journey at IIT K; your constant questions and doubts did encourage me to dig further into specific topics specially on factoring. Thanks to Vishwas Bhargav for making the IIT K journey more enjoyable in many ways (which can not be discussed openly, xD). Thanks to Rajendra, Mahesh and Seetha Ram for making a non-boring yet motivating and enjoyable environment all the time.

The CMI list is very lengthy; it would be very difficult to make the list exhaustive, let me just thank everyone that need be thanked. It was really a wonderful span of 5 years

(well technically 4) journey at CMI and without you guys, it could have been simply boring and non-enjoyable. You guys will be missed.

Lastly, I would also like to thank my parents, my sister and my friends who have been very supportive during my ups and downs in my career. I would have been nothing without the constant support that I got from my family; I owe a lot more than you think.

Dedicated to

My family, for their unconditional love, support and guidance and simple yet inspiring take on life.

Contents

Chapter 1

Introduction

In half-century of existence, *Computational Complexity Theory* has developed into a rich, deep and broad theory with remarkable achievements and formidable challenges. It has important practical impact on computer science and industry and has forged strong connections with a diverse set of mathematical fields. The interplay between mathematics and computer science demands algorithmic approaches to various algebraic constructions. Computational Algebra precisely addresses this issue. Polynomials, being so basic and so useful, are studied in a variety of mathematical areas.

A key part of understanding any given polynomial is to understand its *factorization*, for multivariate polynomials can be uniquely factored into *irreducible polynomials* in direct analogue with the *prime factorization* of integers. While integer factorization is a seemingly hard problem for (classical) computation, since the 1960's there has been significant progress on algorithms for the polynomial factorization problem of computing the irreducible factors of a given polynomial [LLL82, Ber70, CZ81]. This intense study has yielded efficient randomized algorithms for this problem, which has important applications in complexity theory, coding theory, and cryptography.

The most basic yet still nontrivial setting for this problem is to consider *univariate polynomials*, where the natural input size of a degree d univariate polynomial is $d+1$ (the number of possible monomials). It is important to note that d is given is *unary* [1]. For n-variate degree-d polynomials, the corresponding size would be $\binom{n+d}{d}$ (the dense representation). We are interested in a concise representation. The most obvious such representation is the sparse representation where the input is the non-zero coefficients with appropriate indexing.

[1] One can consider a much more concise input representation where polynomials are given by the list of nonzero coefficients, where coefficients and exponents are given in binary. It was shown in [Pla77b] and subsequent work that algorithms for this regime are harder, and some algorithmic tasks are NP-hard

From the view of algebraic complexity theory, it is most natural to consider polynomials as being given by *arithmetic circuits*. It is natural to *desire* a classification of polynomials based on their "simplicity". The algorithmic approach suggests that the simple polynomials are those that can be computed easily. In some sense, one should be able to measure the ease of computation. Their complexity is measure by the number of arithmetic operations required to compute natural ones i.e. the elementary symmetric polynomials, determinant, permanent etc. we will formally define subsequent objects in section . The sparse representation is a special case of this (depth-2 formulas) where depth means the length of the longest input-output path. One may consider more powerful models such as (general or bounded-depth) formulas, algebraic branching programs (ABPs) etc. In all settings we are interested in producing all irreducible factors of the given polynomial, where each such factor is given a succinct representation (such as via arithmetic circuits). Without further delay, we formally define arithmetic circuits, formula, ABPs and algebraic complexity classes.

1.1 Arithmetic Circuits, Formulas and ABPs

We shall fix an underlying field $\mathbb{F}$.

Definition 1 (Arithmetic Circuits and Formulas)**.** *An arithmetic circuit is a* ***directed acyclic graph*** *with one sink (which is called the output gate). Each of the source vertices (which are called input nodes) are either labeled by a variable x_i or an element from $\mathbb{F}$. Each of the internal nodes are labeled either by $+$ or $\times$ to indicate if it is an addition or multiplication gate respectively. Sometimes edges may carry weights that are elements from $\mathbb{F}$.*
Such a circuit naturally computes a multivariate polynomial at every node. The circuit is said to compute a polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ if the output node computes f.
An arithmetic circuit is a ***formula*** *if every internal node has out-degree* 0*.*

Here is an example of circuit and formula computing $(x_1 + x_2)^2$:

Without loss of generality, the circuit or formula is assumed to be layered, with edges only between successive layers. There are some important parameters of an arithmetic circuit are the following :

- SIZE: the number of nodes and edges in the circuit or formula
- DEPTH: the longest path from a leaf gate to the output gate
- DEGREE: the *syntactic* degree of the polynomial computed at the output gate. This is computed recursively at every gate in the most natural way. One can take max of the

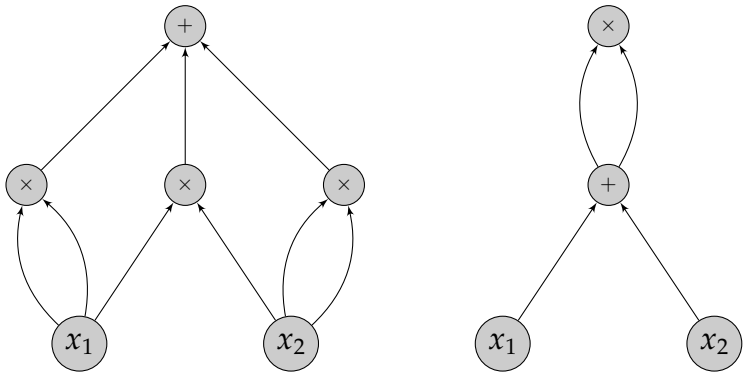


FIGURE 1.1: Formula and Circuit computing $(x_1 + x_2)^2$

degrees of children at an addition gate and the sum of the degrees at a multiplication gate. Observe that this **need not be** the degree of the polynomial computed at the output gate due to possible cancellations, but this is certainly an upper bound.

To understand *non-commutative formulas*, Nisan defined the notion of an arithmetic branching programs (ABPs).

Definition 2 (ABP)**.** *An* ABP *is a layered graph with $d+1$ layers as follows. The layers are labeled by $0, 1, \ldots, d$. The edges of the graph go from layer i to layer $i+1$. Every edge e has polynomials as their weights which is linear of the form $\ell_e = \sum_{j\in[n]} c_{e,j}x_j$ or constants. Layer 0 has only one vertex s called the **source** and layer d has only one vertex t called the **sink**. For every directed path from the source to the sink $\gamma = (e_1, e_2, \ldots, e_d)$ define the polynomial associated to , denoted $f[\gamma]$ as $f[\gamma] = \ell_{e_1} \ldots \ell_{e_d}$. The polynomial computed by the ABP is*

$$C(x) = \sum_{\gamma \in path(s,t)} f[\gamma]$$

It is well-known that sum over all paths in a layered graph can be represented by an iterated matrix multiplication.

1.2 Algebraic Complexity Classes

Arithmetic circuits provide a way, alternate to *Turing machines*. Similarly, the algebraic complexity classes contain multivariate polynomial families instead of languages. Strictly speaking, this definition is for infinite family of polynomials over n variables, one for each n, as otherwise every polynomial can be computed using $O(1)$ operations rendering the whole exercise meaningless.

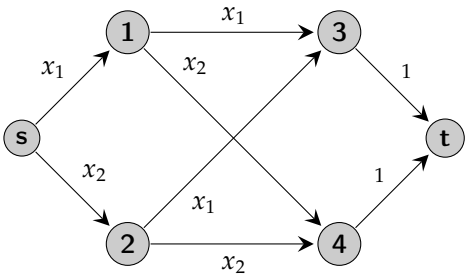


FIGURE 1.2: ABP computing $(x_1 + x_2)^2$

Valiant's paper [Val79] transformed arithmetic complexity into a complexity theory where he provides the analogs of all basic foundations of Boolean computational complexity :

- It introduces a mathematically elegant notion of efficient reducibility between polynomials, *projection*
- Arithmetic analogs of P and NP was defined which are now called respectively VP and VNP and it was established that these classes has natural complete polynomials under such reductions: permanent is complete for VNP and determinant is (nearly) complete for VP.

We use size(f) to denote the minimal size of an arithmetic circuit computing a polynomial f. The class VP, in complete analogy to P/poly, is simply all polynomials computable by polynomial size arithmetic circuits. Here it should be noted that all polynomials discussed, the degree is polynomially bounded by the number of variables. Thus e.g. the symmetric polynomials, Determinant polynomials are all in VP.

Definition 3 (The class VP)**.** *We say that $f = \{f_n\}$ is in* VP *if* size$(f_n) \leq$ poly(n) *and deg*$(f_n) \leq$ poly(n).

Defining the analog VNP of NP is a bit trickier nevertheless a natural one. In NP an *existential quantifier* is used, which can be viewed a Boolean disjunction over all possible Boolean values to possible "witnesses" in a polynomial size Boolean circuit. Similarly, in VNP this disjunction is replaced by a summation over possible "witnesses" in a polynomial size arithmetic circuit.

Definition 4 (The class VNP)**.** *We says $\{f_n\}_n$ is in* VNP *if there exist polynomials $t(n), s(n)$ and a family $\{g_n\}_n$ in* VP *such that for every n, $f_n(\overline{x}) = \sum_{w \in \{0,1\}^{t(n)}} g_n(\overline{x}, w_1, \ldots, w_{t(n)})$. Here,* witness *size is $t(n)$ and* verifier *circuit g_n has size $s(n)$.*

We clearly have VP $\subseteq$ VNP and the major problem of arithmetic complexity theory is proving that the containment is strict.

Conjecture 1 [Valiant's Hypothesis]

$\mathsf{VP} \neq \mathsf{VNP}$ *over any field* $\mathbb{F}$.

There are other restrictive classes defined below :

- VF which contains the families of n-variate polynomials of degree $n^{O(1)}$ over computed by $n^{O(1)}$-sized formulas
- The class VBP contains the families of polynomials computed by $n^{O(1)}$-sized ABP*s*
- VQP (resp. VQF) which contains the families of n-variate polynomials of degree $n^{O(1)}$, computed by $2^{\mathsf{poly}(\log n)}$-sized circuits (resp. formulas)

In general, we define $\mathsf{VF}(s), \mathsf{VBP}(s), \mathsf{VNP}(s)$ as follows. Let $s : \mathbb{N} \longrightarrow \mathbb{N}$ be a function. Define the class $\mathsf{VF}(s)$ that contain families $\{f_n\}_n$ such that n-variate f_n can be computed by an algebraic formula of size $\mathsf{poly}(s(n))$ and has degree $\mathsf{poly}(n)$. Similarly, $\mathsf{VBP}(s)$ contains families $\{f_n\}_n$ such that f_n can be computed by an ABP of size $\mathsf{poly}(s(n))$ and has degree $\mathsf{poly}(n)$. Finally, $\mathsf{VNP}(s)$ denotes the class of families $\{f_n\}_n$ such that f_n has witness size $\mathsf{poly}(s(n))$, verifier circuit size $\mathsf{poly}(s(n))$, and has degree $\mathsf{poly}(n)$.

From our definition, it is clear that $\mathsf{VQF} \subseteq \mathsf{VF}(n^{\log n})$. We have the easy containments: $\mathsf{VF} \subseteq \mathsf{VBP} \subseteq \mathsf{VP} \subseteq \mathsf{VQP} = \mathsf{VQF}$, follows from [BOC92, VSBR83].

If we relax the condition on the degree in the definition of VP, by allowing the degree to be possibly exponential, then we define the class VP_{nb}. Such circuits can compute constants of exponential bit-size (unlike VP). There are border complexity classes which we will define later. In the next section, we will focus on the closure results of complexity classes.

1.3 Previously known closure results

As mentioned earlier, the complexity classes contain multivariate polynomial families instead of languages. It is a natural question to ask whether a complexity class is closed under certain operations, such as p-projection, substitution etc. VP and VNP are closed under these operations mentioned above. A natural operation regarding polynomial is *factoring*. We study the following two questions related to multivariate polynomial factorization.

1. Let $\{f_n(x_1, \ldots, x_n)\}_n$ be a polynomial family in an algebraic complexity class $\mathcal{C}$ (egs. VP, VF, VBP, VNP etc). Let g_n be an arbitrary factor of f_n. Can we say that $\{g_n\}_n \in \mathcal{C}$? Equivalently, is the class $\mathcal{C}$ *closed under factoring*?

2. Can we design an *efficient*, i.e. deterministic or randomized poly(n)-time, algorithm to output the factor g_n with a representation in $\mathcal{C}$? (*Uniformity*)

The general formulation of the factorization problem where we are given such a (restricted) circuit is the *white-box model*. Some algorithms can even work in the more restrictive *black-box model*, where access to the given circuit is restricted to evaluating the computed polynomial at any desired point [2]. In all settings we are interested in producing all irreducible factors of the given polynomial, where each such factor is given a succinct representation (such as via algebraic circuits). Famously, Kaltofen [Kal85, Kal86, Kal87, Kal89] showed that VP is *uniformly* closed under factoring, i.e. for a given d degree n variate polynomial f of circuit size s, there exists a randomized poly(snd)-time algorithm that outputs its factor as a circuit whose size is bounded by poly(snd). This fundamental result has several applications such as 'hardness versus randomness' in algebraic complexity [KI03, AV08, DSY09, AGS18], derandomization of Noether Normalization Lemma [Mul17], in the problem of circuit reconstruction [KS09, Sin16], and polynomial equivalence testing [Kay11]. In general, multivariate polynomial factoring has several applications including decoding of Reed-Solomon, Reed-Muller codes [GS98, Sud97], integer factoring [LLMP90], primary decomposition of polynomial ideals [GTZ88] and algebra isomorphism [KS06, IKRS12].

It is natural to ask whether Kaltofen's VP factoring result can be extended to VP_{nb} which allows degree of the polynomials to be exponentially high. It is known that *not every* factor of a high degree polynomial has a small sized circuit. For example, the polynomial $x^{2^s}-1$ can be computed in size s, but it has factors over $\mathbb{C}$ that require circuit size $\Omega\left(2^{s/2}/\sqrt{s}\right)$ [LS78, Sch77]. It is conjectured that *low* degree factors of high degree small-sized circuits have *small* circuits.

Conjecture 2 [[Bür13],Conj.8.3]

Let g be a factor of a polynomial f in n variables over a field $\mathbb{F}$ of char $= 0$. Then we have with a universal constant $c > 0$ that

$$size(g) \leq (size(f) + \deg(g) + n)^c$$

Partial results towards it are known. It was shown in [Kal87] that if polynomial f given by a circuit of size s factors as $g^e h$, where g and h are coprime, then g can be computed by a circuit of size poly$(e, \deg(g), s)$. The question left open is to remove the dependency on e. In the special case where $f = g^e$, it was established that g has circuit size poly$(\deg(g), \text{size}(f))$.

[2] In some cases, the white-box and black-box models are equivalent such as sparse polynomials [BOT88, KS01]. In such cases, one can be efficiently and deterministically about the circuit from the black-box access.

On the other hand, several algorithmic problems are NP-hard, eg. computing the degree of the squarefree part, gcd, or lcm; even in the case of supersparse univariate polynomials [Pla77b].

Now, we discuss the closure results for classes more restrictive than VP(such as VF, VBP etc.). Unfortunately, Kaltofen's technique [Kal89] for VF will give a superpolynomial-sized factor formula; as it heavily *reuses* intermediate computations while working with linear algebra and Euclid gcd. The same holds for the class VBP. In contrast, extending the idea of [DSY09], Oliveira [Oli16] showed that an n-variate polynomial with *bounded individual degree* and computed by a formula of size s, has factors of formula size $\mathsf{poly}(n, s)$. Furthermore, it was established that for a given n-variate individual-degree-r polynomial, computed by a circuit (resp. formula) of size s and depth Δ, there exists a $\mathsf{poly}(n^r, s)$-time randomized algorithm that outputs any factor of f computed by a circuit (resp. formula) of depth $\Delta + 5$ and size $\mathsf{poly}(n^r, s)$. We are not aware of any work specifically on VBP factoring, except a special case in [KK08]—it dealt with the elimination of a *single* division gate from skew circuits [Jan11] that was weakened later owing to proof errors.

Going beyond VP we can ask about the closure of VNP.

Conjecture 3 [[Bür13],Conj.2.1]

VNP *is closed under factoring.*

This has been very recently shown to be true by Mrinal Kumar et al in [CKS18] where they used the technique from [DSY09].

Why do we care about closure results?

We conclude by stating a few reasons why closure results under factoring are interesting and non-trivial. First, there are classes that are *not* closed under factors. For example, the class of sparse polynomials; as a factor's sparsity may blowup super-polynomially [vzGK85]. Closure under factoring indicates the robustness of an algebraic complexity class, as, it proves that all nonzero multiples of a *hard* polynomial remain hard. For this reason, closure results are also important for proving lower bounds on the power of some algebraic proof systems [FSTW16].

The problem of checking if a circuit computes the zero polynomial is called polynomial identity testing (PIT). It turns out that this problem is easy to solve algorithmically. There are efficient randomized algorithms known. Moreover, in a surprising connection, it has been found that if there is a deterministic polynomial time algorithm for solving PIT, then certain explicit polynomials are hard to compute [KI03, Agr05]. Therefore, the solution to PIT problem has a key role in our attempt to computationally classify polynomials.

Interestingly, there are certain connections between PIT and factoring as shown in [KSS15] establishing equivalence between derandomizing PIT and factoring. Factoring is the key reason why PIT, for VP, can be reduced to very special cases, and gets tightly related to circuit lower bound questions (like VP$\neq$VNP?). See [KI03, Thm.4.1] for whitebox PIT connection and [AGS18] for blackbox PIT. One of the central reasons is:
Suppose a polynomial $f(\overline{y})$ is such that for a nonzero size-s circuit C, $C(f(\overline{y})) = 0$. Then, using factoring results for low degree C, one deduces that f also has circuit size $\mathsf{poly}(s)$. This gives us the connection: *If we picked a "hard" polynomial f then $f(\overline{y})$ would be a hitting-set generator (hsg)* [3] *for C* [KI03, Thm.7.7].
Our work is strongly motivated by the open question of proving such a result for size-s circuits C that have high degree (i.e. $s^{\omega(1)}$). Our first factoring result (Theorem 2) implies such a 'hardness to hitting-set' connection for arbitrarily high degree circuits C assuming that: the squarefree part C_{sqfree} of C has low degree. In such a case we only have to find a hitting-set for C_{sqfree} which, as our result proves, has low algebraic circuit complexity.

For general overview of factoring and connections to algebraic complexity theory, see [FS15]. In the next section, we discuss about sparsity as a *complexity measure* of polynomials and connections to algebraic complexity theory.

1.4 Sparsity and Algebraic Complexity Theory

The sparsity of f denoted as $||f||$, is the number of monomials (with non zero coefficients) appearing in f. The sparsity of f is another natural complexity measure one can use for polynomials and was studied in various contexts [vzGK85, KS01, SSS13]. Assuming a single field element fits in a single location, one can store a polynomial with s monomials on a computer using $O(s)$ memory locations and then one would like to perform basic operations on polynomials such as evaluation, multiplication, composition, etc. efficiently in the size of this representation.

Suppose we are interested in factoring a polynomial $f(x) = g(x) \cdot h(x)$ where the input is given as sparse representation. If we are to factor f and store g and h, then we must first have an upper bound on the sparsity of these factors. This problem was raised in the seminal paper of von zür Gathen and Kaltofen [vzGK85] who studied the problem of efficient polynomial factorization in the sparse representation and gave a factoring algorithm whose running time is polynomial in the size of the input and in the size of the output. But, apparently no good upper bound is known.We have the following conjecture regarding sparsity upper bound. We denote $\deg_{x_i}(f)$ to be highest degree of x_i in f.

[3] Let $\mathcal{C} \subset \mathbb{F}[x_1, \ldots, x_n]$ be a set of polynomials. A polynomial $\mathcal{G} : \mathbb{F}^s \longrightarrow \mathbb{F}^n$ is a hitting set generator for $\mathcal{C}$ with seed length s if for all $f \in \mathcal{C}$, $f \equiv 0 \iff f(\mathcal{G}) \equiv 0$

Conjecture 4 [Sparsity Conjecture]

Suppose $f \in \mathbb{C}[x_1, \ldots, x_n]$ *with* $deg_{x_i}(f) \leq r$. *Then,* $g \mid f \implies ||g|| \leq ||f||^{\log r}$.

When $r \leq 2$, Volkovich in [Vol15] showed that the above bound is true. If, in general the answer is positive, the bound will be tight. This is because consider

$$f = \prod_{i=1}^{n}(x_i^r - 1) \qquad g = \prod_{i=1}^{n}(x_i^{r-1} + x_i^{r-2} + \ldots + 1)$$

Of course, $||f|| = 2^n$ whereas $||g|| = r^n = (2^n)^{\log r}$. Observe that if conjecture 4 is true, then we have some interesting applications :

1. Using [vzGK85], one can give a *quasi-polynomial time randomized algorithm* for factoring sparse polynomials.

2. We can give a quasi-polynomial time *deterministic* algorithm for *sparse divisibility testing* [4] as shown in [DdO14] combined with deterministic sparse interpolation due to [KS01].

Very recently, Bhargav et al gave a non trivial bound on the sparsity of the factors in [BSV18] where they showed that for an n variate polynomial f such that $||f|| = s, \deg_{x_i}(f) \leq d$ and $g \mid f \implies ||g|| \leq s^{d^2 \log n}$.

1.5 Contribution of the thesis

Our main results are of two types. Some are related to closure results and size of factors. The others are related to the sparsity bound for special class of polynomials.

Before stating the results, we describe some of the assumptions and notations used throughout the paper. Set $[n]$ refers to $\{1, 2, \ldots, n\}$. Logarithms are wrt base 2.

Field. We denote the underlying field as $\mathbb{F}$ and assume that it is of characteristic 0 and algebraically closed. For eg. complex $\mathbb{C}$, algebraic numbers $\overline{\mathbb{Q}}$ or algebraic p-adics $\overline{\mathbb{Q}}_p$. All the results partially hold for other fields (such as $\mathbb{R}, \mathbb{Q}, \mathbb{Q}_p$ or finite fields of characteristic $>$ degree of the input polynomial). For a brief discussion on this issue, see Section **??**.

Ideal. We denote the variables $(x_1, \ldots, x_n)$ as $\overline{x}$. The *ideal* $I := \langle \overline{x} \rangle$ of the polynomial ring will be of special interest, and its power ideal I^d, whose generators are all degree d

[4] Given two sparse polynomials f, g so that g divides f and is asked to output a polynomial h so that $h \cdot g = f$

monomials in n variables. Often we will reduce the polynomial ring modulo I^d (inspired from *Taylor series of an analytic function around* $\bar{0}$ [Tay15]).

Radical. For a polynomial $f = \prod_i f_i^{e_i}$, with f_i's coprime irreducible nonconstant polynomials and multiplicity $e_i > 0$, we define the squarefree part as the *radical*

$$\text{rad}(f) := \prod_i f_i$$

1.5.1 Factorization over Power Series Ring

We show that under random linear transformation, any polynomial gets factored into linear power series factors. This is a structural results which will be used time and again to prove the other theorems. Of course, from proposition 1, we have $\mathbb{F}[[\bar{x}]][y]$ is UFD and hence one can talk about *factorization* over power series domain.

Theorem 1 [Power Series Complete Split]

Let $f \in \mathbb{F}[\bar{x}]$ with $deg(rad(f)) =: d_0 > 0$. Consider $\alpha_i, \beta_i \in_r \mathbb{F}$ and the map $\tau : x_i \mapsto \alpha_i y + x_i + \beta_i$, $i \in [n]$, where y is a new variable.

Then, over $\mathbb{F}[[\bar{x}]]$, $f(\tau\bar{x}) = k \cdot \prod_{i\in[d_0]}(y - g_i)^{\gamma_i}$, where $k \in \mathbb{F}^*$, $\gamma_i > 0$, and $g_i(\bar{0}) := \mu_i$. Moreover, μ_i's are distinct nonzero field elements.

We will present two proofs of the theorem 1, see chapter 3

1.5.2 Closure Results

What can we say about these f_i's if f has a circuit of size s? Our main result gives a good circuit size bound when $\text{rad}(f)$ has small degree. A more general formulation (with u_0) is:

Theorem 2

If $f = u_0 u_1$ is a nonzero product in the polynomial ring $\mathbb{F}[\bar{x}]$, with $size(f) + size(u_0) \leq s$, then every *factor of u_1 has a circuit of size* $\mathsf{poly}(s + deg(rad(u_1)))$.

Note that Kaltofen's proof technique in the VP factoring paper [Kal89] does not extend to the *exponential* degree regime (even when degree of $\text{rad}(f)$ is small) because it requires solving equations with $\deg_{x_i}(f)$ many unknowns for some x_i, where $\deg_{x_i}(f)$ denotes *individual degree* of x_i in f, which can be very high. Also, basic operations like 'determining the coefficient of a univariate monomial' become #P-hard in the exponential-degree regime [Val82]. The proof technique in Kaltofen's single factor Hensel lifting paper [Kal87, Thm.2]

works only in the perfect-power case of $f = g^e$. It can be seen that $\mathrm{rad}(f)$ "almost" equals $f/\gcd(f, \partial_{x_i}(f))$, but the gcd itself can be of exponential-degree and so one cannot hope to use [Kal87, Thm.4] to compute the gcd either. Univariate high-degree gcd computation is NP-hard [Pla77a, Pla77b].

Interestingly, our result when combined with [Kal87, Thm.3] implies that every factor g of f has a circuit of size polynomial in: $\mathrm{size}(f)$, $\deg(g)$ and $\min\{\deg(\mathrm{rad}(f)), \mathrm{size}(\mathrm{rad}(f))\}$. We leave it as an open question whether the latter expression is polynomially related to $\mathrm{size}(f)$.

Theorem 2 shows an interesting way to create *hard* polynomials. In the theorem statement let the size concluded be $(s + \deg(\mathrm{rad}(u_1)))^e$, for some constant e. If one has a polynomial $f_1(x_1, \ldots, x_n)$ that is 2^{cn}-hard, then any nonzero $f := \prod_i f_i^{e_i}$ is also $2^{\Omega(n)}$-hard for *arbitrary* positive e_i's, as long as

$$\sum_i \deg(f_i) \leq 2^{\frac{cn}{e}-1}$$

In general, for a high degree circuit f, $\mathrm{rad}(f)$ can be of high degree (exponential in size of the circuit). Ideally, we would like to show that every degree d factor of f has $\mathsf{poly}(\mathrm{size}(f), d)$-size circuit.

The next theorem reduces the above question to a special kind of modular division, where the denominator polynomial may *not* be invertible but the quotient is well-defined (eg. $x^2/x \bmod x$). All that remains is to somehow eliminate this kind of *non-unit division* operator (which we leave as an open question). Consider 'random' elements $\alpha_i, \beta_i \in_r \mathbb{F}$ and the corresponding random linear map $\tau : x_i \mapsto \alpha_i y + x_i + \beta_i$, $i \in [n]$, where y is a new variable apart from $x_1, \ldots, x_n$.

Theorem 3

If nonzero $f \in \mathbb{F}[\overline{x}]$ can be computed by a circuit of size s, then any degree d factor of $f(\tau\overline{x})$ is of the form $A/B \bmod \langle \overline{x} \rangle^{d+1}$ where polynomials A, B have circuits of size $\mathsf{poly}(sd)$.

Note that in Theorem 3, B may be non-invertible in $\mathbb{F}[\overline{x}]/\langle \overline{x} \rangle^{d+1}$ and may have a high degree (eg. 2^s). So, we cannot use the famous trick of Strassen to do division elimination here [Str73].

We prove uniform closure results, under factoring, for some algebraic complexity classes.

Theorem 4

The classes $\mathsf{VF}(n^{\log n}), \mathsf{VBP}(n^{\log n}), \mathsf{VNP}(n^{\log n})$ *are all closed under factoring. Moreover, there exists a randomized* $\mathsf{poly}(n^{\log n})$*-time algorithm that: for a given* $n^{O(\log n)}$ *sized formula (resp.* ABP*)* f *of* $\mathsf{poly}(n)$*-degree, outputs* $n^{O(\log n)}$ *sized formula (resp.* ABP*) of a nontrivial factor of* f *(if one exists).*

Remark. The "time-complexity" in the algorithmic part makes sense only in certain cases. For example, when $\mathbb{F} \in \{\mathbb{Q}, \mathbb{Q}_p, \mathbb{F}_q\}$, or when one allows computation in the BSS-model [BSS89]. In the former case our algorithm takes $\mathsf{poly}(n^{\log n})$ bit operations (assuming that the characteristic is zero or larger than the degree; see Theorem 36 in Section 8.1).

It is important to note that Theorem 4 does not follow by invoking Kaltofen circuit factoring [Kal89] and VSBR transformation [VSBR83] from circuit to log-depth formula. Formally, if we are given a formula (resp. ABP) of size $n^{O(\log n)}$ and degree $\mathsf{poly}(n)$, then it has factors which can be computed by a circuit of size $n^{O(\log n)}$ and depth $O(\log n)$. If one converts the factor circuit to a formula (resp. ABP), one would get the size upper bound of the factor formula to be a much larger $(n^{O(\log n)})^{\log n} = n^{O(\log^2 n)}$. Moreover, Kaltofen's methods crucially rely on the circuit representation to do linear algebra, division with remainder, and Euclid gcd in an efficient way; a nice overview of the implementation level details to keep in mind is [KSS15, Sec.3].

Our proof methods extend to the approximative versions $\mathcal{C}(n^{\log n})$ for $\mathcal{C} \in \{\overline{\mathsf{VF}}, \overline{\mathsf{VBP}}, \overline{\mathsf{VNP}}\}$ [5] as well (Theorem 35).

As before, Theorem 4 has an interesting lower bound consequence: If f has VF (resp. VBP resp. VNP) complexity $n^{\omega(\log n)}$ then any nonzero fg has similar hardness (for $\deg(g) \leq \mathsf{poly}(n)$).

In fact, the method of Theorem 4 yields a formula factor of size $s^e d^{2\log d}$ for a given degree-d size-s formula (e is a constant). This means— If determinant $\det_n$ requires $n^{a \log n}$ size formula, for $a > 2$, then *any* nonzero degree-$O(n)$ multiple of $\det_n$ requires $n^{\Omega(\log n)}$ size formula.

Similarly, if we conjecture that a VP-complete polynomial f_n (say the homomorphism polynomial in [DMM+14, Thm.19]) has $n^{a \log n}$ ABP complexity, for $a > 4$, then *any* nonzero degree-$O(n)$ multiple of f_n has $n^{\Omega(\log n)}$ ABP complexity.

GCD is an important operation for polynomials and one could also ask similar questions about complexity of gcd. We separately discuss some interesting results about complexity of gcd in chapter 4. Originally Kaltofen proved that low degree gcd of *low* complexity *arbitrary degree* circuits has small size circuit [Kal87, Theorem 3]. But the complexity was

[5] For definitions, see chapter 7

dependent on the number of input polynomials. We essentially remove the dependency on the number of input polynomials.

Theorem 5

Let $f_i \in \mathbb{F}[\bar{x}]$ for $i \in [m]$ and let $g = gcd(f_1, \ldots, f_m)$ such that $size(f_i) \leq s$. Then, $size(g) \leq \mathsf{poly}(s, d)$ where $d := deg(g)$

For details, see chapter 4

1.5.3 Sparsity Bound

Let us consider the set of polynomials where $\deg_{x_i}(f) = r$ or 0. We will prove that the conjecture is true for these set of polynomials. For notational purpose, let us define

$$S_{n,r} = \{f \mid f \in \mathbb{C}[x_1, \ldots, x_n] \text{ such that } \deg_{x_i}(f) = r \text{ or } 0\}$$

Theorem 6

If $f \in S_{n,r}$, then, we have $g \mid f \iff ||g|| \leq ||f||^{\log r}$.

Theorem 6 is important in the sense that example 1.4 is possibly the only example where such quasi polynomial blow up is known. Observe that $f = \prod_{i \in [n]} (x_i^r - 1) \in S_{n,r}$. Hence, we have established that possibly for the corner extreme cases, we are fine with the bound.

This is still interesting in the sense that Bhargav etal's bound in [BSV18] would give dependency of r instead of $\log r$ in the exponent.

1.6 Organization of the thesis

Chapter 1 is just for the introduction. Chapter 2 is building some back ground and tools that will be used through out the thesis. Chapter 3 is the main building block where the idea of using Newton Iteration to factorize polynomial has been described. Chapter 4 talks about complexity of gcd of two polynomials in algebraic models. Chapter 5 talks about the closure of restricted classes such as $\mathsf{VF}(n^{O(\log n)})$ and $\mathsf{VBP}(n^{O(\log n)})$ and design algorithm to output factors in the respected classes. Chapter 6 talks about the dependency of the squarefreeness on the complexity of the factor which *partially* solves the famous *factor conjecture* as well. Chapter 7 deals with the approximative complexity classes such as $\overline{\mathsf{VP}}, \overline{\mathsf{VF}}(n^{O(\log n)})$ and so on and show that they are also closed under factoring. Chapter 8

deals with the factoring of polynomials when the base field is not *good enough*. Chapter 9 talks about sparsity bound on specific class of factors. Finally, we conclude in chapter 10 by asking the immediate open questions.

Most of the contents of Chapter 3, 5, 6, 7, 8 can be found in the paper "Discovering the roots: Uniform closure results for algebraic classes under factoring" (accepted at STOC'2018), a joint work with my advisor Dr. Nitin Saxena and my colleague Amit Sinhababu.

Chapter 2

Preliminaries

This chapter is dedicated to building up the tools and lemmas that will be required in proving the theorems mentioned in the previous chapter.

2.1 Formal Power Series

In mathematics, a **formal power series** is a generalization of a polynomial, where the number of terms is allowed to be infinite; this implies giving up the possibility of replacing the variable in the polynomial with an arbitrary number. Thus a formal power series differs from a polynomial in that it may have infinitely many terms, and differs from a power series, whose variables can take on numerical values.

one may think of a formal power series as a power series in which we ignore questions of *convergence* by not assuming that the variable X denotes any numerical value (not even an unknown value). For example, consider the series

$$1 - 3X + 5X^2 - 7X^3 + 9X^4 - \dots$$

If we studied this as a power series, its properties would include, for example, that its *radius of convergence* is 1. However, as a formal power series, we may ignore this completely; all that is relevant is the sequence of coefficients $[1, -3, 5, -7, 9, \dots]$. *Arithmetic* i.e. addition and multiplication on formal power series is carried out by simply pretending that the series are polynomials. Once we have defined **multiplication** for formal power series, we can define multiplicative inverses as follows. The multiplicative inverse of a formal power series A is a formal power series C such that $AC = 1$, provided that such a formal power series exists. It turns out that if A has a multiplicative inverse, it is unique, and we denote it by A^{-1}. For example,

$$\frac{1}{1-X} = \sum_{i \geq 0} X^i$$

The set of all formal power series in X with coefficients in a commutative ring R form another ring that is written $R[[X]]$, and called the ring of formal power series in the variable X over R. In fact one can generalize for multivariate power series ring $R[[\overline{X}]]$.

Instead of looking into the factorization over $\mathbb{F}[\overline{x}]$, we look into the more analytic factorization pattern of a polynomial over $\mathbb{F}[[x_1, \ldots, x_n]]$, namely, formal power series of n-variables over field $\mathbb{F}$. To talk about factorization, we need the notion of *uniqueness* which the following proposition ensures.

Proposition 1. *[ZS75, Chap.VII] Power series ring $\mathbb{F}[[x_1, \ldots, x_n]]$ is a unique factorization domain (UFD), and so is $\mathbb{F}[[\overline{x}]][y]$.*

2.2 Randomized algorithm for linear algebra using PIT

The following lemma from [KSS15] discusses how to perform linear algebra when the coefficients of vectors are given as formula (resp. ABP). This will be crucially used in Theorem 4 when we would give an algorithm to output the factors.

Lemma 5. *(Linear algebra using PIT [KSS15, Lem.2.6])*

Let $M = (M_{i,j})_{k \times n}$ be a matrix (where k is $n^{O(1)}$) with each entry being a degree $\leq n^{O(1)}$ polynomial in $\mathbb{F}[\overline{x}]$. Suppose, we have algebraic formula (resp. ABP) of size $\leq n^{O(\log n)}$ computing each entry. Then, there is a randomized poly($n^{\log n}$)-time algorithm that either:

- *finds a formula (resp. ABP) of size poly($n^{\log n}$) computing a non-zero $u \in (\mathbb{F}[\overline{x}])^n$ such that $Mu = 0$, or*
- *outputs 0 which declares that $u = 0$ is the only solution.*

Proof. This was proved in [KSS15, Lem.2.6] for the circuit model. Since we are using a different model we repeat the details. The idea is the following. Iteratively, for every $r = 1, \ldots, n$ we shall find an $r \times r$ minor contained in the first r columns that is full rank. While continuing this process, we either reach $r = n$ in which case it means that the matrix has full column rank, hence, $u = 0$ is the only solution, or we get stuck at some value say $r = r_0$. We use the fact that r_0 is rank and using this minor we construct the required non-zero vector u.

We explain the process in a bit more detail. Using a randomized algorithm, we look for some non-zero entry in the first column. If no such entry is found we can simply take $u = (1, 0, \ldots, 0)$. So assume that such a non-zero entry is found. After permuting the rows we can assume wlog that this is $M_{1,1}$. Thus, we have found a 1×1 minor satisfying the requirements. Assume that we have found an $r \times r$ full rank minor that is composed of the

first r rows and columns (we can always rearrange and hence it can be assumed wlog that they correspond to first r rows and columns). Denote this minor by M_r.

Now for every $(r+1)\times(r+1)$ submatrix of M contained in the first $r+1$ columns and containing M_r, we check whether the determinant is 0 by randomized algorithm. If any of these submatrices have nonzero determinant, then we pick one of them and call it M_{r+1}. Otherwise, we have found that first $r+1$ columns of M are linearly dependent. As M_r is full rank, there is $v \in \mathbb{F}(\overline{x})^r$ such that $M_r v = (M_{1,r+1},\ldots,M_{r,r+1})^T$. This can be solved by applying Cramer's rule. The i-th entry of v is of the form $\det(M_r^{(i)})/\det(M_r)$, where $M_r^{(i)}$ is obtained by replacing i-th column of M_r with $(M_{1,r+1},\ldots,M_{r,r+1})^T$. Observe that $\det(M_r)$, as well as $\det(M_r^{(i)})$, are both in $\mathbb{F}[\overline{x}]$.

Then it is immediate that $u := (\det(M_r^{(1)}),\ldots,\det(M_r^{(r)}),-\det(M_r),0,\ldots,0)^T$ is the desired vector.

To find M_r, each time we have to calculate the determinant and decide whether it is 0 or not. This is simply PIT for a determinant polynomial with entries of algebraic complexity $n^{O(\log n)}$ and degree $n^{O(1)}$. So, we have a comparable randomized algorithm for this. Determinant of a symbolic $n\times n$ matrix has $n^{O(\log n)}$ size formula (resp. $\mathrm{poly}(n)$ size ABP) [MV97]. When the entries of the matrix have $n^{O(\log n)}$ size formula (resp. ABP), altogether, the determinant polynomial has the same algebraic complexity. There are $< n^2$ PIT invocations to test zeroness of the determinant. Altogether, we have a $\mathrm{poly}(n^{\log n})$-time randomized algorithm for this [Sch80]. □

2.3 Basic operations on formula, ABP and circuit

We use the following standard results on size bounds for performing some basic operations (like taking derivative) of circuits, formulas, ABPs. We denote $\mathbb{F}[[\overline{x}]]$ to be power series ring over $\mathbb{F}$.

Lemma 6. *(Eliminate single division [Str73], [SY10, Thm.2.1])*

Let f and g be two degree-D polynomials, each computed by a circuit (resp. ABP resp. formula) of size-s with $g(\overline{0}) \neq 0$. Then $f/g \bmod \langle\overline{x}\rangle^{d+1}$ can be computed by $O((s+d)d^3)$ (resp. $O(sd^2D)$ resp. $O(sd^2D^2)$) size circuit (resp. ABP resp. formula).

Proof. Assume wlog that $g(\overline{0}) = 1$; we can ensure this by appropriate normalization. So, we have the following power series identity in $\mathbb{F}[[\overline{x}]]$:

$$f/g = f/(1-(1-g)) = f + f(1-g) + f(1-g)^2 + f(1-g)^3 + \cdots.$$

Note that this is a valid identity as $1-g$ is constant free. For all $d \geq 0$, LHS = RHS $\bmod \langle\overline{x}\rangle^{d+1}$.

If we want to compute $f/g \bmod \langle \overline{x} \rangle^{d+1}$, we can take the RHS of the above identity up to the term $f(1-g)^d$ and discard the remaining terms of degree greater than d. The degree$> d$ monomials can be truncated, using Strassen's *homogenization* trick, in the case of circuits and ABPs (see [Sap16, Lem.5.2]), and an *interpolation* trick in the case of formulas (which also works for ABPs and low degree circuits, [Sap16, Lem.5.4]). A careful analysis shows that the size blow up is at most $O((s+d)d^2 \cdot d)$ (resp. $O(sd \cdot D \cdot d)$ resp. $O(sd \cdot D^2 \cdot d)$) for circuits (resp. ABP resp. formula).

Using the above result, it is easy to see, that we get $\mathrm{poly}(s,d)$ size circuit (resp. ABP resp. formula) for computing $f/g \bmod \langle \overline{x} \rangle^{d+1}$. □

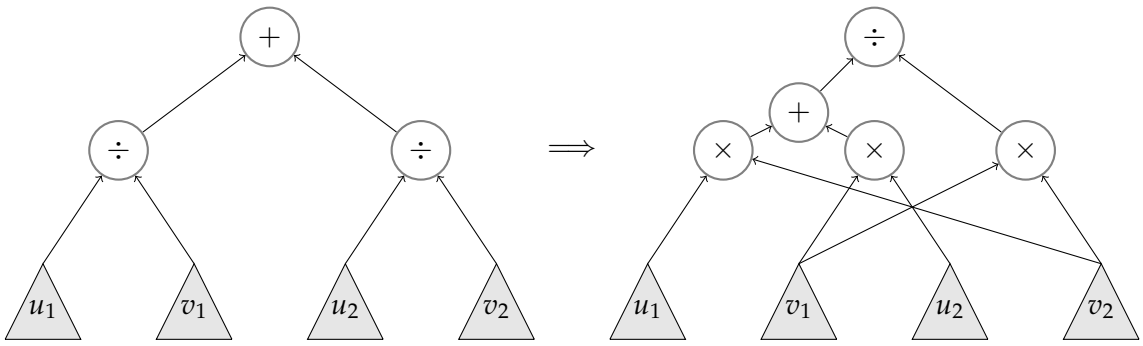
Remark. Note that it may happen that $g(\overline{0}) = 0$, thus $1/g$ does not exist in $\mathbb{F}[[\overline{x}]]$, yet f/g may be a polynomial of degree d. In such a case, we need to discuss a modified *normalization* that works. We can shift the polynomials f, g by some random $\overline{\alpha} \in \mathbb{F}^n$. The constant term of the shifted polynomial is non-zero with high probability [Sch80]. Now, we compute $f(\overline{x}+\overline{\alpha})/g(\overline{x}+\overline{\alpha})$ using the method described above. Finally, we recover the polynomial f/g by applying the reverse shift $\overline{x} \mapsto \overline{x} - \overline{\alpha}$.

What if our model has several division gates?

Lemma 7. *(Div. gates elimination [SY10, Thm.2.12]) Let f be a polynomial computed by a circuit (resp. formula), using division gates, of size s. Then, $f \bmod \langle \overline{x} \rangle^{d+1}$ can be computed by* $\mathrm{poly}(sd)$ *size circuit (resp. formula).*

Proof idea. We preprocess the circuit (resp. formula) so that the only division gate used in the modified circuit (resp. formula) is at the top. Now to remove the single division gate at the top, we use the above power series trick.

The idea of the pre-processing is the following. We can separately keep track of numerator and denominator computed at each gate and simulate addition, multiplication and division gates in the original circuit. For $+$ gate, use the identity $\frac{u_1}{v_1} + \frac{u_2}{v_2} = \frac{u_1 v_2 + u_2 v_1}{v_1 v_2}$. Similarly do for $\times$ gate. This pre-processing incurs only poly(sd) blow up in the case of circuits. In the case of formulas one has to ensure that in any path from the leaf to the root, there are only $O(\log sd)$ division gates.



□

2.3.1 Upper bound on derivative computation

Suppose $f(\overline{x})$ is a multivariate polynomial of degree d computed by a circuit(resp. formula resp. ABP) of size s. What can we say about size of its derivative with respect to one variable? What about higher order derivative? The next lemma exactly seeks the desired answer.

Lemma 8 (Derivative computation)**.** *If a polynomial $f(\overline{x}, y)$ can be computed by a circuit (resp. formula resp. ABP) of size s and degree d. Then, any $\frac{\partial^k f}{\partial y^k}$ can be computed by circuit (resp. formula resp. ABP) of size $\mathsf{poly}(sk)$.*

Proof. The idea is simply to use the homogenization and interpolation properties [Sap16, Sec.5.1-2].

Let $f(\overline{x}, y) = c_0 + c_1 y + c_2 y^2 + \ldots + c_\delta y^\delta$, where $c_0, c_1, \ldots, c_\delta \in \mathbb{F}[\overline{x}]$. Given the circuit (resp. formula resp. ABP) computing polynomial $f(\overline{x}, y)$, we can get the circuits (resp. formula resp. ABP) computing $c_0, \ldots, c_\delta$ using homogenization and interpolation as discussed before. Given $c_0, \ldots, c_\delta$, computing $\frac{\partial^k f}{\partial y^k}$ in size $\mathsf{poly}(sd)$ is trivial. We use this approach of computing derivative when the polynomial is of degree $d \leq \mathsf{poly}(s)$.

In the case of high degree circuits, we cannot use the above approach. [Kal87, Thm.1] shows that $\frac{\partial^k f}{\partial y^k}$ can be computed by a circuit of size $O(k^2 s)$, i.e. the degree of the circuit does not matter. The main idea is to inductively use the Leibniz product rule of k-th order derivative and store at each gate upto k-th derivative i.e. for a gate computing u, we store $(u, u^{(1)}, \ldots, u^{(k)})$ which we compute bottom-to-top as follows :

1. For a $+$ gate which computes w where $w = u + v$ (children gates compute u and v), as we have computed $u^{(i)}$ and $v^{(i)}$, we use the identity $w^{(i)} = u^{(i)} + v^{(i)}$ to compute i-th derivative at that gate

2. For a $\times$ gate computing w where $w = u \cdot v$, we use the following identity

$$w^{(i)} = \sum_{j=0}^{i} \binom{i}{j} u^{(i-j)} v^{(j)}$$

   to compute i-th derivative.

Overall it is very easy to compute that the final size of the circuit will be $O(k^2 s)$. □

2.3.2 Lower bound on derivative computation

One can similarly ask about lower bound on the size of higher order derivative of the given polynomial. The next lemma "almost" positively answers it.

Lemma 9 (Lower bound [Val82]). *Suppose a polynomial $f(\overline{x}, y)$ can be computed by a circuit of size s. Then, $\frac{\partial^k f}{\partial y^k}$ can be computed by circuit of size* $\mathsf{poly}(s, \log k) \implies \mathsf{VP} = \mathsf{VNP}$.

Proof. Consider the following polynomial of $n^2 + n$ variables

$$g(y_1, \ldots, y_n, z_{1,1}, \ldots, z_{n,n}) = \prod_{i \in [n]} \left(\sum_{j \in [n]} y_j z_{i,j} \right)$$

Observe that coefficient of $y_1 \ldots y_n$ in g is nothing but $\mathrm{perm}(z_{1,1}, \ldots, z_{n,n})$. Consider a new polynomial f by substituting $y_i = x^{(n+1)^{i-1}}$ (kronecker substitution). In particular, let

$$f(x, z_{1,1}, \ldots, z_{n,n}) := g(x, x^{n+1}, x^{(n+1)^2}, \ldots, x^{(n+1)^{n-1}}, z_{1,1}, \ldots, z_{n,n})$$

As kronecker substitution gives different weights to different monomials, coefficient $c_k(z_{1,1}, \ldots, z_{n,n})$ of x^k in f is actually $\mathrm{perm}(z_{1,1}, \ldots, z_{n,n})$ where $k = 1 + (n+1) + \ldots + (n+1)^{n-1}$. Therefore, it implies the following identity

$$\left. \frac{\partial^k f(x, z_{1,1}, \ldots, z_{n,n})}{\partial x^k} \right|_{x=0} = k! \mathrm{perm}(z_{1,1}, \ldots, z_{n,n})$$

Observe that $\mathrm{size}(g)$ as well as $\mathrm{size}(f)$ is $\mathsf{poly}(n)$. Hence, assuming the hypothesis we would get that $\mathrm{perm}(z_{1,1}, \ldots, z_{n,n})$ has $\mathsf{poly}(n)$ size circuit i.e. $\mathsf{VP} = \mathsf{VNP}$. □

2.4 Sylvester matrix & resultant

First, let us look at the notion of resultant of two univariate polynomials. Let $p(x), q(x) \in \mathbb{F}[x]$ be of degree a, b respectively. From Euclid's extended algorithm, it can be shown that there exist two polynomials $u(x), v(x) \in \mathbb{F}[x]$ such that

$$u(x)p(x) + v(x)q(x) = \gcd(p(x), q(x))$$

This is known as Bezout's identity. If $\gcd(p(x), q(x)) = 1$, then (u, v) with $\deg(u) \leq b$ and $\deg(v) \leq a$ is unique. Let us take

$$u(x) = u_0 + u_1 x + u_2 x^2 + \ldots + u_b x^b$$
$$v(x) = v_0 + v_1 x + v_2 x^2 + \ldots + v_a x^a$$

Now, if we use the equation $u(x)p(x) + v(x)q(x) = \gcd(p(x), q(x))$ and compare the coefficients of x^i, for $0 \leq i \leq a + b$, we get a system of linear equations in the $a + b + 2$ many unknowns (u_i's and v_i's). The system of linear equations can be represented in the

matrix form as $Mx = y$, where x consists of the unknowns. Resultant of f, g is defined as the determinant of the matrix M. It is easy to see that M is invertible if and only if the polynomials are coprime.

Now, the notion of resultant can be extended to multivariate, by defining resultant of polynomials $f(\overline{x}, y)$ and $g(\overline{x}, y)$ wrt some variable y. The idea is same as before, now we take gcd wrt the variable y and get a system of linear equations from Bezout's identity. The matrix can be explicitly written with entries being polynomial coefficients (or they could be from $\mathbb{F}[[\overline{x}]]$). This is known as Sylvester matrix, which we define next.

Definition 10. *Let $f(\overline{x}, y) = \sum_{i=0}^{l} f_i(\overline{x}) y^i$ and $g(\overline{x}, y) = \sum_{i=0}^{m} g_i(\overline{x}) y^i$. Define* Sylvester *matrix of f and g wrt y as the following $(m + l + 1) \times (m + l + 1)$ matrix:*

$$Syl_y(f, g) := \begin{bmatrix} f_l & 0 & 0 & \dots & 0 & g_m & 0 & 0 & 0 \\ f_{l-1} & f_l & 0 & \dots & 0 & g_{m-1} & g_m & 0 & 0 \\ f_{l-2} & f_{l-1} & f_l & \dots & 0 & g_{m-2} & g_{m-1} & g_l & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ f_0 & f_1 & \dots & \dots & f_l & g_0 & g_1 & \dots & g_m \\ 0 & f_0 & \dots & \dots & \dots & 0 & g_0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & \dots & f_0 & 0 & \dots & \dots & g_0 \end{bmatrix}$$

So, resultant can be formally defined as follows (for more details and alternate definitions, see [LN97, Chap.1]).

Definition 11. *Given two polynomials $f(\overline{x}, y)$ and $g(\overline{x}, y)$, define the resultant of f and g wrt y as determinant of the Sylvester matrix,*

$$Res_y(f, g) := det(Syl_y(f, g)) .$$

From the definition, it can be seen that $\text{Res}_y(f, g)$ is a polynomial in $\mathbb{F}[x]$ with degree bounded by $2\text{deg}(f)\text{deg}(g)$. Now, we state the following fundamental property of the Resultant, which is crucially used.

Proposition 2 (Res vs gcd). *1. Let $f, g \in \mathbb{F}[\overline{x}, y]$ be polynomials with positive degree in y. Then, $Res_y(f, g) = 0 \iff$ f and g have a common factor in $\mathbb{F}[\overline{x}, y]$ which has positive degree in y.*

2. There exists $u, v \in \mathbb{F}[\overline{x}]$ such that $uf + vg = Res_y(f, g)$.

The proof of this standard proposition can be found in many standard books on algebra including [vzGG13, Sec.6].

Lemma 12 (Squarefree-ness). *Let $f \in \mathbb{F}(\overline{x})[y]$ be a polynomial with $deg_y(f) \geq 1$. f is square free iff $f, f' := \partial_y f$ are coprime wrt y.*

Proof. The main idea is to show that there does not exist $g \in \mathbb{F}(\overline{x})[y]$ with positive degree in y such that $g \mid \gcd_y(f(\overline{x}, y), f'(\overline{x}, y))$. This is true because– suppose g is an irreducible polynomial with positive degree in y that divides both $f(\overline{x}, y)$ and $f'(\overline{x}, y)$. So,

$$f(\overline{x}, y) = gh \implies f'(\overline{x}, y) = gh' + g'h \implies g \mid g'h\,.$$

As g is irreducible and $\deg_y(g') < \deg_y(g)$ we deduce that $g \mid h$. Hence, $g^2 \mid f$. This contradicts the hypothesis that f is square free. □

Now, we state another standard lemma, which is useful to us and which is proved using the property of Resultant.

Lemma 13 (Coprimality). *Let $f, g \in \mathbb{F}(\overline{x})[y]$ be coprime polynomials wrt y (& nontrivial in y). Then, for $\overline{\beta} \in_r \mathbb{F}^n$, $f(\overline{\beta}, y)$ and $g(\overline{\beta}, y)$ are coprime (& nontrivial in y).*

Proof. Consider $f = \sum_{i=1}^{d} f_i y^i$ and $g = \sum_{i=1}^{e} g_i y^i$. Choose a random $\overline{\beta} \in_r \mathbb{F}^n$. Then, by Proposition 2 & [Sch80], $f_d \cdot g_e \cdot \mathrm{Res}_y(f, g)$ at $\overline{x} = \overline{\beta}$ is nonzero. This in particular implies that

$$\mathrm{Res}_y(f(\overline{\beta}, y), g(\overline{\beta}, y)) \neq 0$$

Hence, by Proposition 2, $f(\overline{\beta}, y)$ and $g(\overline{\beta}, y)$ are coprime. □

2.5 Monic Transformation

Sometimes it is easy to work with *monic* polynomial (monic wrt one variable) i.e. polynomials whose leading co-efficient wrt a variable is a field element. Of course, initially we may not be given a monic polynomial. Hence, we should find a way to make it monic so that the factorization pattern [1] does not change.

Lemma 14 (Transform to monic). *For a polynomial $f(\overline{x})$ of total degree $d \geq 0$ and random $\alpha_i \in_r \mathbb{F}$, the transformed polynomial $(\hat{f})(\overline{x}, y) := f(\overline{\alpha} y + \overline{x})$ has a nonzero constant as coefficient of y^d, and degree wrt y is d. Moreover, $\hat{f}$ has same factoring pattern as f.*

Proof. Suppose the transformation is $x_i \mapsto x_i + \alpha_i y$ where $i = 1(1)n$, any monomial $\overline{x}^{\overline{\beta}} = x_1^{\beta_1} \ldots x_n^{\beta_n}$ will contribute monomials of the form

$$x_1^{\beta'_1} y^{\beta_1 - \beta'_1} \ldots x_n^{\beta'_n} y^{\beta_n - \beta'_n} = y^{\beta_1 - \beta'_1 + \beta_2 - \beta'_2 + \ldots + \beta_n - \beta'_n} x_1^{\beta'_1} \ldots x_n^{\beta'_n}.$$

[1] What we meant is if $f = \prod f_i^{e_i}$ where f_i's are irreducible and suppose $f(\tau\overline{x})$ is monic after the transformation τ, then $f(\tau\overline{x}) = \prod f_i(\tau\overline{x})^{e_i}$ where $f_i(\tau\overline{x})$ are irreducible.

Hence degree remains same as of the initial monomial and so $\deg(\hat{f}) \leq \deg(f) = d$ (as highest degree coefficients might get cancelled). We show that for random α_i's, $\deg(\hat{f}) = \deg(f) = \deg_y(\hat{f}) = d$. Let us assume

$$S = \{\overline{\beta} \mid |\overline{\beta}|_1 = d \text{ and } c_{\overline{\beta}} \neq 0\}$$

where $f = \sum c_{\overline{\beta}} \overline{x}^{\overline{\beta}} +$ lower degree terms . Coefficient of degree d of y in $\hat{f}$ is

$$\sum_{\overline{\beta} \in S} c_{\overline{\beta}} \alpha_1^{\beta_1} \dots \alpha_n^{\beta_n}$$

We want to show that for random α_i's, $\sum_{\overline{\beta} \in S} c_{\overline{\beta}} \alpha_1^{\beta_1} \dots \alpha_n^{\beta_n} \neq 0$ so that deg remains d. It is easy to see that as

$$\sum_{\overline{\beta} \in S} c_{\overline{\beta}} t_1^{\beta_1} \dots t_n^{\beta_n} = 0$$

in variable $t_1, .., t_n$ is a non-zero polynomial, from *Schwartz-Zippel* lemma, *random points* will not be a zero of the above polynomial. Thus this makes sure that $\deg_y(\hat{f}) = d$ and as $d = \deg_y(\hat{f}) \leq \deg(\hat{f}) \leq d$, we have $\deg_y(\hat{f}) = \deg(f) = d$.

For the second part, it is enough to show that if f is irreducible, then $\hat{f}$ is irreducible too [2]. suppose $\hat{f}(\overline{x}, y)$ is not irreducible. As it is monic, assume that a non-zero polynomial(i.e. with positive degree in y)

$$g(\overline{x}, y) \mid \hat{f}(\overline{x}, y) \implies \hat{g}(\overline{x}, y) = g(\overline{x} - \overline{\alpha} y) \mid f(\overline{x})$$

As $\alpha_i \in_r \mathbb{F}$, $\hat{g}$ is a non-zero polynomial contradicting the fact that f is irreducible. □

2.6 Closure properties for VNP

VNP-*size parameter* (w, v) of F refers to w being the witness size and v being the size of the verifier circuit f.

Let $F(\overline{x}, y), G(\overline{x}, y), H(\overline{x})$ have verifier polynomials f, g and h with the VNP size parameters (w_f, v_f),(w_g, v_g), (w_h, v_h) respectively. Let the degree of F wrt y be d. Then, the following closure properties can be shown ([BCS13] or [Bür13, Thm.2.19]):

1. Add (resp. Multiply): $F + G$ (resp. FG) has VNP-size parameter $(w_f + w_g, v_f + v_g + 3)$.
2. Coefficient: $F_i(\overline{x})$ has VNP-size parameter $(w_f, (d+1)(v_f + 1))$, where $F(\overline{x}, y) =: \sum_{i=0}^{d} F_i(\overline{x}) y^i$.

[2] This is true because if reducibility and degree are both preserved, then factorization pattern must remain the same

3. Compose: $F(\overline{x}, H(\overline{x}))$ has VNP-size parameter $((d+1)(w_f + dw_h), (d+1)^2(v_f + v_h + 1))$.

Proof. All the above statements are easy to prove using the definition of VNP.

1.

$$\begin{aligned}(FG)(\overline{x}, y) &= \left(\sum_{u \in \{0,1\}^{w_f}} f(\overline{x}, u_1, \ldots, u_{w_f})\right) \cdot \left(\sum_{u \in \{0,1\}^{w_g}} g(\overline{x}, u_1, \ldots, u_{w_g})\right) \\ &= \sum_{u \in \{0,1\}^{w_f + w_g}} A(\overline{x}, u_1, \ldots, u_{w_f + w_g})\end{aligned}$$

where,

$$A(\overline{x}, u_1, \ldots, u_{w_f+w_g}) = f(\overline{x}, u_1, \ldots, u_{w_f}) \cdot g(\overline{x}, u_{w_f+1}, \ldots, u_{w_f+w_g})$$

Trivially, A has size $v_f + v_g + 3$ (extra: one node, two edges) and witness size is $w_f + w_g$.

Similarly, with $F + G$.

2. Interpolation gives, $f_i(\overline{x}) = \sum_{j=0}^{d} \alpha_j F(\overline{x}, \beta_j)$, for some distinct arguments $\beta_j \in \mathbb{F}$. Clearly, $F(\overline{x}, \beta_j)$ has VNP-size parameter (w_f, v_f). Using the previous addition property we get that the verifier circuit has size $(d+1)(v_f+1)$. Witness size remains w_f as we can reuse the witness string of F.

3. Write $F(\overline{x}, y) =: \sum_{i=0}^{d} F_i(\overline{x}) y^i$. We know that F_i has VNP-size parameter $(w_f, (d+1)(v_f+1))$. For $0 \leq i \leq d$, H^i has VNP-size parameter $(iw_h, (i+1)v_h)$ using i-fold product (Item 1). Substituting $y = H$ in F, we can calculate the VNP-size parameter.

   Suppose F_i and H^i have corresponding verifier circuits A_i and B_i respectively. Then,

$$\begin{aligned}F(\overline{x}, H(\overline{x})) &= \sum_{i=0}^{d} F_i(\overline{x}) H^i(\overline{x}) \\ &= \sum_{i=0}^{d} \left(\sum_{u \in \{0,1\}^{w_f}} A_i(\overline{x}, u)\right) \cdot \left(\sum_{u \in \{0,1\}^{iw_h}} B_i(\overline{x}, u)\right)\end{aligned}$$

   Thus, the witness size is $< (d+1)(w_f + dw_h)$. The corresponding verifier circuit size is $< (d+1)^2(v_f + v_h + 1)$.

□

2.7 Matrix and Series Inverse

Lemma 15 (Matrix inverse). *Let $\mu_i, i \in [d]$, be distinct nonzero elements in $\mathbb{F}$. Define a $d \times d$ matrix A with the (i,j)-th entry $1/(y_i - \mu_j)^2$. Its entries are in the function field $\mathbb{F}(\overline{y})$. Then, $det(A) \neq 0$.*

Proof. The idea is to consider the power series of the function $1/(y_i - \mu_j)^2$ and show that a monomial appears nontrivially in that of $\det(A)$.

We first need a claim about the coefficient operator on the determinant.

Claim 16. *Let $f_j = \sum_{i \geq 0} \beta_{j,i} x^i$ be a power series in $\mathbb{F}[[x]]$, for $j \in [d]$. Then, $Coeff_{\overline{x}^{\overline{\alpha}}} \circ det\left(f_j(x_i)\right) = det\left(\beta_{j,\alpha_i}\right)$.*

Proof of Claim 16. Observe that the rows of the matrix have disjoint variables. Thus, $x_i^{\alpha_i}$ could be produced only from the i-th row. This proves: $\text{Coeff}_{\overline{x}^{\overline{\alpha}}} \circ \det\left(f_j(x_i)\right) = \det\left(\text{Coeff}_{x_i^{\alpha_i}} \circ f_j(x_i)\right) = \det\left(\beta_{j,\alpha_i}\right)$. □

By Taylor expansion we have

$$\frac{1}{(x-\mu)^2} = \frac{1}{\mu^2} \sum_{j \geq 1} j \left(\frac{x}{\mu}\right)^{j-1} .$$

Hence, the coefficient of y_i^{i-1} in $A(i,j)$ is

$$\frac{1}{\mu_j^2} \frac{i}{\mu_j^{i-1}} = \frac{i}{\mu_j^{i+1}} .$$

By the above claim, the coefficient of $\prod_{i \in [d]} y_i^{i-1}$ in $\det(A)$ is: $\det\left(\left(\frac{i}{\mu_j^{i+1}}\right)\right)$. By cancelling i (from each row) and $1/\mu_j^2$ (from each column), we simplify it to the Vandermonde determinant:

$$\det \begin{bmatrix} \frac{1}{\mu_1^0} & \frac{1}{\mu_2^0} & \cdots & \frac{1}{\mu_d^0} \\ \frac{1}{\mu_1^1} & \frac{1}{\mu_2^1} & \cdots & \frac{1}{\mu_d^1} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{1}{\mu_1^{d-1}} & \frac{1}{\mu_2^{d-1}} & \cdots & \frac{1}{\mu_d^{d-1}} \end{bmatrix} = \prod_{i<j \in [d]} \left(\frac{1}{\mu_i} - \frac{1}{\mu_j}\right) \neq 0 .$$

Hence, the determinant of A is non-zero. □

Remark. If the characteristic of $\mathbb{F}$ is a prime $p \geq 2$ then the above proof needs a slight modification. One should consider the coefficient of $\prod_{i \in [d]} y_i^{s_i - 1}$ in $\det(A)$ for a set

$S = \{s_1, \ldots, s_d\}$ of distinct non-negative integers that are not divisible by p. Moreover, one has to consider 'random' μ_i's to deduce $\det(A) \neq 0$.

Lemma 17 (Series inverse). *Let $\delta \geq 1$. Assume that A is a polynomial of degree $< \delta$ and B is a homogeneous polynomial of degree δ, such that $A(\overline{0}) =: \mu \neq 0$. Then, we have the following identity in $\mathbb{F}[[\overline{x}]](y) \cap \mathbb{F}[[\overline{x}]][[y]]$:*

$$\frac{1}{y-(A+B)} \equiv \frac{1}{y-A} + \frac{B}{(y-\mu)^2} \mod \langle \overline{x} \rangle^{\delta+1}$$

Proof. We will use the notation $A^{[1,\delta-1]}$ to refer to the sum of the homogeneous parts of A of degrees between 1 and $\delta - 1$ (equivalently, it is $A^{<\delta} - \mu$). Note that $B \cdot A^{[1,\delta-1]}$ vanishes mod $\langle \overline{x} \rangle^{\delta+1}$. Now, in $\mathbb{F}[[\overline{x}]][[y]]$,

$$\begin{aligned}
\frac{1}{y-(A+B)} &\equiv \frac{1}{y-\mu-\left(A^{[1,\delta-1]}+B\right)} \mod \langle \overline{x} \rangle^{\delta+1} \\
&\equiv \frac{1}{y-\mu}\left(\frac{1}{1-\frac{A^{[1,\delta-1]}+B}{y-\mu}}\right) \mod \langle \overline{x} \rangle^{\delta+1} \\
&\equiv \frac{1}{y-\mu}\left(1+\left(\frac{A^{[1,\delta-1]}+B}{y-\mu}\right)+\left(\frac{A^{[1,\delta-1]}+B}{y-\mu}\right)^2+\ldots\ldots\right) \mod \langle \overline{x} \rangle^{\delta+1} \\
&\equiv \frac{1}{y-\mu}\left(1+\left(\frac{A^{[1,\delta-1]}+B}{y-\mu}\right)+\left(\frac{A^{[1,\delta-1]}}{y-\mu}\right)^2+\left(\frac{A^{[1,\delta-1]}}{y-\mu}\right)^3+\ldots\ldots\right) \mod \langle \overline{x} \rangle^{\delta+1} \\
&\equiv \frac{1}{y-\mu}\left(1+\left(\frac{A^{[1,\delta-1]}}{y-\mu}\right)+\left(\frac{A^{[1,\delta-1]}}{y-\mu}\right)^2+\ldots\ldots\right)+\frac{B}{(y-\mu)^2} \mod \langle \overline{x} \rangle^{\delta+1} \\
&\equiv \frac{1}{y-\mu}\left(\frac{1}{1-\frac{A^{[1,\delta-1]}}{y-\mu}}\right)+\frac{B}{(y-\mu)^2} \mod \langle \overline{x} \rangle^{\delta+1} \\
&\equiv \frac{1}{y-A}+\frac{B}{(y-\mu)^2} \mod \langle \overline{x} \rangle^{\delta+1} \quad .
\end{aligned}$$

□

2.8 Holomorphic function and Order of zero

In mathematics, a **holomorphic** function is a complex-valued function of one or more complex variables that is complex differentiable in a neighborhood of every point in its domain. The existence of a complex derivative in a neighborhood is a very strong condition, for it implies that any holomorphic function is actually *infinitely differentiable* and equal to

its own *Taylor series* (analytic). Holomorphic functions are the central objects of study in complex analysis.

In complex analysis, a *zero* (sometimes called a root) of a holomorphic function f is a complex number a such that $f(a) = 0$. Generally, the *multiplicity* of the zero of f at a is the positive integer n for which there is a holomorphic function g such that

$$f(z) = (z-a)^n \cdot g(z) \text{ and } g(a) \neq 0$$

We denote this by $\text{ord}_a(f) = n$. In this thesis, we are only concerned about polynomials (in general power series) which is analytic by definition. We have the following lemma :

Lemma 18 (Order Lemma)**.** *Let f and g be two analytic functions. Then,*

$$ord_{z_0}\left(\frac{f}{g}\right) = ord_{z_0}(f) - ord_{z_0}(g)$$

Proof. Let us assume that $\text{ord}_{z_0}(f) = m$ and $\text{ord}_{z_0}(g) = n$. Then, $f(z) = (z-z_0)^m h_1(z)$ and $g(z) = (z-z_0)^n h_2(z)$ where h_1 and h_2 are both analytic with $h_1(z_0) \neq 0$ and $h_2(z_0) \neq 0$. Then,

$$\frac{f}{g} = (z-z_0)^{m-n} h(z)$$

where $h(z) = \frac{h_1(z)}{h_2(z)}$. Ofcourse $h(z_0) \neq 0$ and hence $h(z)$ is holomorphic in the neighborhood of z_0 and hence the lemma holds. □

Lemma 19 (Order of Derivative)**.** *Let $f(\overline{x}, y) \in \mathbb{F}((\overline{x}))[[y]]$ such that $ord_g(f) = e$ for some $g \in \mathbb{F}((\overline{x}))$ where $e \geq 1$. Then $ord_g(\partial_y f) = e - 1$*

Proof. This simply follows from following

$$f = (y-g)^e \cdot A \implies \partial_y f = (y-g)^{e-1}\left(A + (y-g)A'\right)$$

But as $A(g) \neq 0 \implies y - g \nmid (A + (y-g)A') \implies (A + (y-g)A')$ is analytic and non-zero at $y = g$. So, the lemma follows. □

It is easy to show that f is analytic implies $f^{1/e}$ is also analytic where $e \in \mathbb{Q}$.

2.9 Newton-Puiseux Series

In mathematics, the Laurent series of a complex function $f(z)$ is a representation of that function as a power series which includes terms of negative degree. The Laurent series for

a complex function $f(z)$ about a point c is given by:

$$f(z) = \sum_{n=-\infty}^{\infty} a_n (z-c)^n$$

where a_n and c are constants. *Puiseux series* differ from Laurent series in that they allow for fractional exponents of the indeterminate, as long as these fractional exponents have bounded denominator. Formally, a Pisuex series is of the form

$$f(T) = \sum_{k \geq k_0} c_k T^{k/n}$$

for some $n \in \mathbb{N}, k_0 \in \mathbb{Z}$ and c_k are constants from the field.

Puiseux's theorem, sometimes also called the **Newton–Puiseux theorem**, asserts that, given a polynomial equation $P(x, y) = 0$, its solutions in y, viewed as functions of x, may be expanded as Puiseux series that are convergent in some neighbourhood of the origin (0 excluded, in the case of a solution that tends to infinity at the origin). In other words, every branch of an algebraic curve may be locally (in terms of x) described by a Puiseux series. In fact, the theorem tells stronger result.

Theorem 20 (Newton-Pisuex Theorem)**.** *if* $\mathbb{F}$ *is an algebraically closed field of characteristic zero, then the field of Puiseux series over* $\mathbb{F}$ *is the algebraic closure of the field of formal Laurent series over* $\mathbb{F}$*.*

Chapter 3

Newton Iteration and Factoring Polynomials

Newton iteration based numerical methods are very popular in engineering [OR00, GMS+86, BSR+05]. We will establish an interesting connection between finding the roots and complexity of factors which largely depends on the very idea of Newton Iteration. Let us first demonstrate how NI (Newton Iteration) came into the picture of factoring polynomials. For the time being, we only worry about finding the root and not the complexity. We start off by asking the simplest questions.

Question : Suppose $f(\overline{x}, y) = (y - g(\overline{x})) \cdot u(\overline{x}, y)$ where $y - g \nmid u$. Can we find g?

This is a root finding problem as $f(\overline{x}, g) = 0$. Recall that *Classical Newton Iteration* is one of the famous root finding algorithms. Suppose we want to find "good" approximation of α such that $f(\alpha) = 0$ where f is *continuously differentiable* and $f'(\alpha) \neq 0$ i.e. α is a simple root. The algorithm does the following :

1. guess a *starting* point x_0
2. calculate $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$

It can be shown that there exists a neighborhood of α such that for all starting values x_0 in that neighborhood, the sequence $\{x_n\}$ will converge to α i.e. $\lim_{n\to\infty} x_n = \alpha$.

It is then very natural to ask whether we can do similar thing to find g. If yes, what is the notion of *approximation* ? What is the *starting point*? Let us start by answering the notion of approximation in power series ring.

Let R be an integral Domain. Let us take $0 \neq A(\overline{x}) = \sum a_{\overline{\alpha}} \overline{x}^{\overline{\alpha}} \in R[[\overline{x}]]$. Define

$$\text{val}(A(\overline{x})) = \min\{|m|_1 \mid a_m \neq 0\}$$

where $|m|_1$ denotes the ℓ_1 norm [1]. Also define $\text{val}(\overline{0}) = \infty$. This is a *valuation* on ring $R[[\overline{x}]]$ as it satisfies two conditions:

1. $\text{val}(A(\overline{x}) + B(\overline{x})) \geq \min(\text{val}(A(\overline{x})), \text{val}(B(\overline{x})))$
2. $\text{val}(A(\overline{x})B(\overline{x})) = \text{val}(A(\overline{x})) + \text{val}(B(\overline{x}))$.

For any real $\epsilon \in (0,1)$, define norm$||.||_\epsilon$ on $R[[x]]$ by

$$||A(\overline{x})||_\epsilon = \epsilon^{\text{val}(A(\overline{x}))} \text{ with } ||0||_\epsilon = 0$$

One can easily show that by using the two properties that it gives a metric space structure on $R[[\overline{x}]]$ with metric

$$d_\epsilon(A(\overline{x}), B(\overline{x})) = ||A(\overline{x}) - B(\overline{x})||_\epsilon$$

We therefore obtain a notion of limit as follows:

$$\lim_{m\to\infty} A_m(\overline{x}) = A(\overline{x}) \iff \lim_{m\to\infty} d_\epsilon(A_m(\overline{x}), A(\overline{x})) = 0$$

One can show that $R[[\overline{x}]]$ is a *complete metric space*; it is the completion of $R[\overline{x}]$ under the metric defined above. We have the following proposition.

Proposition 3. *Let $A_m(\overline{x}) \in R[[\overline{x}]]$ sequence of elements. Then $\lim_{m\to\infty} A_m(\overline{x}) = A(\overline{x})$ iff for every $\overline{\alpha}$, there exists $N \geq 0$ such that for all $M \geq N$, $coeff_{\overline{x}^{\overline{\alpha}}} A_M(\overline{x}) = coeff_{\overline{x}^{\overline{\alpha}}} A(\overline{x})$.*

Now we come back to the finding g such that $f(g) = 0$ and $f'(g) \neq 0$. One can think $f(y)$ as a polynomial over $\mathbb{F}[\overline{x}][y]$. Let us try to imitate the classical NI:

- Initial starting point $y_0 = \mu$ where $\mu := g(\overline{0})$
- Define $y_{t+1} = y_t - \frac{f(\overline{x},y_t)}{f'(\overline{x},y_t)}$. Can we say that y_t is an approximation of g?
- If $f'(\overline{x}, y_t)$ is invertible, then one can show that $y_t \equiv g \bmod \langle\overline{x}\rangle^{2^t} \implies y_{t+1} \equiv g \bmod \langle\overline{x}\rangle^{2^{t+1}}$. One can use taylor expansion to prove the above.

$$\begin{aligned} f(\overline{x}, y_{t+1}) &= f\left(\overline{x},\, y_t - \frac{f(\overline{x},y_t)}{f'(\overline{x},y_t)}\right) \\ &= f(\overline{x},y_t) - f'(\overline{x},y_t)\frac{f(\overline{x},y_t)}{f'(\overline{x},y_t)} + \frac{f''(\overline{x},y_t)}{2!}\left(\frac{f(\overline{x},y_t)}{f'(\overline{x},y_t)}\right)^2 - \dots \\ &= 0 \bmod \langle\overline{x}\rangle^{2^{t+1}}. \end{aligned}$$

[1] For $x = (x_1, \dots, x_n)^T$, we define ℓ_1 norm of x denoted by $||x||_1$ to be as $||x||_1 = \sum_{i\in[n]} |x_i|$

It is also clear that as $f'(\overline{x}, y_t)$ is invertible, we have

$$y_{t+1} \equiv y_t \bmod \langle \overline{x} \rangle^{2^t} \implies y_{t+1} \equiv g \bmod \langle \overline{x} \rangle^{2^t}$$

To prove that y_{t+1} is in fact correct upto higher degree, we combine the above :

$$\begin{aligned} f(\overline{x}, g) = 0 &\implies f(\overline{x}, g) \equiv 0 \bmod \langle \overline{x} \rangle^{2^{t+1}} \\ &\implies f(\overline{x}, y_{t+1} + (g - y_{t+1})) \equiv 0 \bmod \langle \overline{x} \rangle^{2^{t+1}} \\ &\implies f(\overline{x}, y_{t+1}) + (g - y_{t+1}) f'(\overline{x}, y_{t+1}) \equiv 0 \bmod \langle \overline{x} \rangle^{2^{t+1}} \end{aligned} \tag{3.1}$$

The last line follows by using taylor series and the fact that

$$(g - y_{t+1})^i \equiv 0 \bmod \langle \overline{x} \rangle^{2^{t+1}} \text{ for } i \geq 2$$

We also have $f'(\overline{x}, y_t)$ is invertible $\iff f'(\overline{x}, y_t)\Big|_{\overline{x}=\overline{0}} \neq 0 \iff f'(\overline{0}, \mu) \neq 0$. This combined with $y_{t+1} \equiv y_t \bmod \langle \overline{x} \rangle^{2^t}$ we have

$$f'(\overline{x}, y_{t+1})\Big|_{\overline{x}=\overline{0}} \neq 0$$

i.e. $f'(\overline{x}, y_{t+1})$ is invertible. As, $f(\overline{x}, y_{t+1}) \equiv 0 \bmod \langle \overline{x} \rangle^{2^{t+1}}$, equation 3.1 ensures the fact that

$$y_{t+1} \equiv g \bmod \langle \overline{x} \rangle^{2^{t+1}}$$

- As $f(0, y_0) = 0$ and $f'(0, y_0) \neq 0$, inductively as above we have existence of y_t such that $y_t \equiv g \bmod \langle \overline{x} \rangle^{2^t}$.
- By the definition of limit, we have

  $$\lim_{t \to \infty} y_t = g$$

  So, this process works even if g is a power series. If $\deg(g) = d$ i.e. a polynomial, one can find g by calculating $y_{\log(d+1)} \bmod \langle \overline{x} \rangle^{d+1}$ as above.

Let us complicate these situations and ask whether we can do similar thing as above.

1. What if $f = (y - g_1)(y - g_2)$ but $g_1(\overline{0}) = g_2(\overline{0})$?
   (a) Pick $\overline{\alpha} \in \mathbb{F}^n$ such that $g_1(\alpha) \neq g_2(\overline{\alpha})$
   (b) $f(\overline{x} + \overline{\alpha}, y) = (y - g_1(\overline{x} + \overline{\alpha}))\,(y - g_2(\overline{x} + \overline{\alpha}))$
   (c) If we put $\overline{x} = \overline{0}$, we get $(y - g_1(\alpha))\,(y - g_2(\overline{\alpha}))$

(d) apply Newton Iteration (NI)

2. What if $f = (y-g)^e \cdot u$? We can differentiate $e-1$ times and apply NI on $f^{(e-1)}$.
3. What about $f(\overline{x}, y) = \left(y^k + c_{k-1}(\overline{x})y^{k-1} + \ldots + c_0(\overline{x})\right) \cdot u$ where $k > 1$?

We would like to relate non-linear factors to somehow root finding so that we can apply NI. Of course, one should not expect that a polynomial always has a factor which is linear in one variable as in the above. But, if one works with an algebraically closed field, then a univariate polynomial completely splits into linear factors (also see the *fundamental theorem of algebra* [CRS96, S2.5.4]). So, if we go to the algebraic closure of $\mathbb{F}(x_1, \ldots, x_{n-1})$, any multivariate polynomial which is monic in x_n will split into factors all linear in x_n. A representation of the elements of $\overline{\mathbb{F}(x_1, \ldots, x_{n-1})}$ as a finite circuit is impossible (eg. $\sqrt{x_1}$). This motivated us to prove something which shows that *all* the roots (wrt a new variable y) are actually elements from $\mathbb{F}[[x_1, \ldots, x_n]]$, after a *random* linear transformation on the variables, $\tau : \overline{x} \mapsto \overline{x} + \overline{\alpha}y + \overline{\beta}$, is applied (Theorem 1). Note– By a random choice $\alpha \in_r \mathbb{F}$ we will mean that choose randomly from a fixed finite set $S \subseteq \mathbb{F}$ of appropriate size (namely $> \deg(f)$). This will be in the spirit of [Sch80]. Once we have power series roots, one can try to approximate the roots in the same manner as above (though remember, we are not talking anything about complexity of such operations. We will talk about that later). In the next section talks about power series roots of a generic polynomial.

3.1 Power series factorization of polynomials

As discussed before, we need to first apply a random linear map, that will make sure that the resulting polynomial splits completely over the ring $\mathbb{F}[[\overline{x}]]$ (Recall: $\mathbb{F}$ is algebraically closed.). Before stating and proving the split theorem, we need to discuss about the criteria which ensures existence of a power series root. In some sense it is the reverse of the newton iteration where if the initial conditions are satisfied, newton iteration will ensure existence of a power series root.

Lemma 21. *(Power series root [BCS13, Thm.2.31]) Let $P(\overline{x}, y) \in \mathbb{F}(\overline{x})[y]$, $P'(\overline{x}, y) = \frac{\partial P(\overline{x}, y)}{\partial y}$ and $\mu \in \mathbb{F}$ be such that $P(\overline{0}, \mu) = 0$ but $P'(\overline{0}, \mu) \neq 0$. Then, there is a unique power series S such that $S(\overline{0}) = \mu$ and $P(\overline{x}, S) = 0$ i.e.*

$$y - S(\overline{x}) \mid P(\overline{x}, y)\,.$$

Moreover, there exists a rational function y_t, $\forall t \geq 0$, such that

$$y_{t+1} = y_t - \frac{P(\overline{x}, y_t)}{P'(\overline{x}, y_t)} \text{ and } S \equiv y_t \bmod \langle \overline{x} \rangle^{2^t} \text{ with } y_0 = \mu\,.$$

Proof. We give an inductive proof of existence and uniqueness together. Suppose $P = \sum_{i=0}^{d} c_i y^i$. We show that there is y_t, a rational function $\frac{A_t}{B_t}$ such that $y_t \in \mathbb{F}[[\overline{x}]]$, For all $t \geq 0$, $P(\overline{x}, y_t) \equiv 0 \bmod \langle \overline{x} \rangle^{2^t}$ and for all $t \geq 1$, $y_t \equiv y_{t-1} \bmod \langle \overline{x} \rangle^{2^{t-1}}$. The proof is by induction. Let $y_0 := \mu$. Thus, base case is true. Now suppose such y_t exists. Define $y_{t+1} := y_t - \frac{P(\overline{x}, y_t)}{P'(\overline{x}, y_t)}$.

Now, $y_t \equiv y_{t-1} \bmod \langle \overline{x} \rangle^{2^{t-1}} \implies y_t(\overline{0}) = \mu$. Hence $P'(\overline{x}, y_t)|_{\overline{x}=\overline{0}} = P'(\overline{0}, \mu) \neq 0$ and so $P'(\overline{x}, y_t)$ is a unit in the power series ring. So, $y_{t+1} \in \mathbb{F}[[\overline{x}]]$. Let us verify that it is an improved root of P; we use Taylor expansion.

$$\begin{aligned} P(\overline{x}, y_{t+1}) &= P\left(\overline{x},\, y_t - \frac{P(\overline{x}, y_t)}{P'(\overline{x}, y_t)}\right) \\ &= P(\overline{x}, y_t) - P'(\overline{x}, y_t)\frac{P(\overline{x}, y_t)}{P'(\overline{x}, y_t)} + \frac{P''(\overline{x}, y_t)}{2!}\left(\frac{P(\overline{x}, y_t)}{P'(\overline{x}, y_t)}\right)^2 - \dots \\ &= 0 \bmod \langle \overline{x} \rangle^{2^{t+1}}. \end{aligned}$$

Thus, $P(\overline{x}, y_{t+1}) \equiv 0 \bmod \langle \overline{x} \rangle^{2^{t+1}}$ and $y_{t+1} \equiv y_t \bmod \langle \overline{x} \rangle^{2^t}$. This completes the induction step.

Moreover, using the notion of limit, we have $\lim_{t \to \infty} y_t = S$, a formal power series. It is unique as μ is a non-repeated root of $P(\overline{0}, y)$. In particular, we get that for all $t \geq 0$, $P(\overline{x}, S) = 0$ or $y - S \mid P$. □

Now, we present the split theorem (theorem 1) which talks about power series roots for a generic polynomial under a random transformation. We will prove this in two ways. The first proof is via the existence of power series roots which is *constructive*, as it also gives an algorithm to find approximation of the roots up to any precision, using formal power series version of the Newton iteration method (see [BCS13, Thm.2.31]). The second proof is directly using Newton-Pisuex series and hence not fully self-contained (as we do not discuss the proof of Newton-Pisuex series).

We try to explain the idea using the following example. Consider $f = (y^2 - x^3) \in \mathbb{F}[x, y]$. Does it have a factor of the form $y - g$ where $g \in \mathbb{F}[[x]]$? The answer is clearly 'no' as $x^{\frac{3}{2}}$ does not have any power series representation in $\mathbb{F}[[x]]$. But, what if we shift x randomly? For example, if we use the shift $y \mapsto y, x \mapsto x + 1$. Then, by Taylor series around 1, we see that $(x+1)^{\frac{3}{2}}$ has a power series expansion, namely

$$(x+1)^{\frac{3}{2}} = 1 + \frac{3}{2}x + \frac{3/2 \times 1/2}{2!}x^2 + \dots$$

Does this work for a generic polynomial? Formally, we show that a random linear transformation would suffice.

First proof of Theorem 1. Let the irreducible factorization of f be $\prod_{i\in[m]} f_i^{e_i}$. We apply a random τ so that f, thus all its factors, become monic in y (Lemma 14). The monic factors $\tilde{f}_i := f_i(\tau\overline{x})$ remain irreducible ($\because \tau$ is invertible). Also, $\tilde{f}_i(\overline{0}, y) = f_i(\overline{\alpha}y + \overline{\beta})$ and $\partial_y \tilde{f}_i(\overline{0}, y)$ remain coprime ($\because \overline{\beta}$ is random, apply Lemma 13). In other words, $\tilde{f}_i(\overline{0}, y)$ is square free (Lemma 12).

In particular, one can write $\tilde{f}_1(\overline{0}, y)$ as $\prod_{i=1}^{\deg(f_1)}(y - \mu_{1,i})$ for distinct nonzero field elements $\mu_{1,i}$ (ignoring the constant which is the coefficient of the highest degree of y in $\tilde{f}_1$). Using classical Newton Iteration (see Lemma 21 or [BCS13, Thm.2.31]), one can write $\tilde{f}_1(\overline{x}, y)$ as a product of power series $\prod_{i=1}^{\deg(f_1)}(y - g_{1,i})$, with $g_{1,i}(\overline{0}) := \mu_{1,i}$. Thus, each $f_i(\tau\overline{x})$ can be factored into linear factors in $\mathbb{F}[[\overline{x}]][y]$.

As f_i's are irreducible coprime polynomials, by Lemma 13, it is clear that $\tilde{f}_i(\overline{0}, y)$, $i \in [m]$, are mutually coprime. In other words, $\mu_{j,i}$ are distinct and they are $\sum_i \deg(f_i) = d_0$ many. Hence, $f(\tau\overline{x})$ can be completely factored as $\prod_{i\in[m]} f_i(\tau\overline{x})^{e_i} = \prod_{i\in[d_0]}(y - g_i)^{\gamma_i}$, with $\gamma_i > 0$ and the field constants $g_i(\overline{0})$ being distinct. □

Second proof of Theorem 1. Let us recall Newton-Pisuex series (see section 2.9). Consider $f(\overline{x}, y, z) \in \mathbb{F}[\overline{x}][y, z]$. We will actually work over the field $\mathbb{F}(\overline{x})$ i.e. think of the polynomial $f \in \mathbb{F}(\overline{x})[y, z]$. Newton-Pisuex series tells us that $z - \phi(\overline{x}, y) \mid f(\overline{x}, y, z) \implies$

$$\phi(\overline{x}, y) = \sum_{k\geq k_0} a_k(\overline{x}) y^{k/n}$$

Now $a_k \in \mathbb{F}(\overline{x})$ implies if we shift $x_i \mapsto x_i + \alpha_i$, then shifted a_k's are actually elements in $\mathbb{F}[[\overline{x}]]$. Also, if one shifts $y \mapsto y + \beta$, then $(y + \beta)^{k/n} \in \mathbb{F}[[y]]$. Hence the shifted ϕ is actually an element in $\mathbb{F}[[\overline{x}, y]]$ which is what essentially we proved.

To actually wrap up as whole, it is enough to show that for $f(\overline{x}, y) \in \mathbb{F}[\overline{x}, y]$ irreducible,

$$f(\tau(\overline{x}, y)) = \prod_i (z - g_i(\overline{x}, y))$$

where $\tau : x_i \mapsto z + a_i \cdot x_i + b_i$ and $\tau : y \mapsto z + a \cdot y + b$ with $a_i, b_i, a, b \in_r \mathbb{F}$ and $g_i(\overline{x}, y) \in \mathbb{F}[[\overline{x}, y]]$. It is easy to argue that each irreducible factor of $f(\tau\overline{x})$ can be factored as $\prod_i (z - \phi_i(\overline{x}, y))$ where ϕ_i actually lies in algebraic closure of $\mathbb{F}[x, y]$. The orders of $z - \phi_i$'s will be 1 i.e. $z - \phi_i(\overline{x}, y) \mid\mid f$ otherwise f can not be irreducible. By the above argument, $\phi_i(\overline{x} + \overline{\alpha}, y + \beta) \in \mathbb{F}[[\overline{x}, y]]$ where $\alpha_i, \beta \in_r \mathbb{F}$. Hence the theorem follows. □

What can we say about the factors of the original polynomial? Using that fact that **1** $\mathbb{F}[[\overline{x}]]$ is a UFD and each irreducible polynomials get factored into linear factors over power series ring as shown in the above proof, we have the following corollary:

Corollary 22. *Suppose g is a polynomial factor of f. As before let $f(\tau\overline{x}) = \prod_{i\in[m]} f_i(\tau\overline{x})^{e_i} = k \cdot \prod_{i\in[d_0]}(y - g_i)^{\gamma_i}$. As $g(\tau x) \mid f(\tau\overline{x})$ we deduce that $g(\tau x) = k' \prod (y - g_i)^{c_i}$ with $0 \leq c_i \leq \gamma_i$.*

Moreover, we can get back g by applying τ^{-1} on the resulting polynomial $g(\tau\overline{x})$.

Using the same technique as above one can actually prove another version of power series split theorem.

Theorem 23 (Power Series Split theorem, version 2)**.** *Suppose $f(\overline{x}) = \prod_{i=1}^{k} f_i^{e_i} \in \mathbb{F}[x_1, \ldots, x_n]$ where f_i's are irreducible and mutually coprime polynomials with $deg_{x_i}(rad(f)) = d_i$. Then for $\overline{\alpha} \in_r \mathbb{F}^n$, $\exists c \in \mathbb{F}$ and $g_{i,j} \in \mathbb{F}[[x_1, \ldots, x_{j-1}]]$ for $j = [1, \ldots, n]$ and $i = [1, \ldots, d_j]$ with order $x_1 < \ldots < x_n$ such that*

$$f(\overline{x} + \overline{\alpha}) = c \prod_{j=1}^{n} \prod_{i=1}^{d_j} (x_j - g_{i,j})^{\gamma_{i,j}} \text{ with } g_{i,j}(\overline{0}) \neq 0 \neq g_{i',j}(\overline{0}) \text{ and } \gamma_{i,j} \geq 0$$

We will mainly use the version 1 of the theorem 1.

3.2 Factoring reduces to approximating power series roots

Using the split Theorem 1, we show that multivariate polynomial factoring reduces to power series root finding up to certain precision. Following the above notation f splits as $f(\tau\overline{x}) = \prod_{i=1}^{d_0}(y - g_i)^{\gamma_i}$. For all $t \geq 0$, it is easy to see that

$$f(\tau\overline{x}) \equiv \prod_{i=1}^{d_0}(y - g_i^{\leq t})^{\gamma_i} \mod I^{t+1}$$

where $I := \langle x_1, \ldots, x_n \rangle$. Note that there is a one-one correspondence, induced by τ, between the polynomial factors of f and $f(\tau\overline{x})$ ($\because \tau$ is invertible and f is y-free). We remark that the leading-coefficient of $f(\tau\overline{x})$ wrt y is a nonzero element in $\mathbb{F}$; so, we call it *monic* (Lemma 14). Next, we show case by case how to find a *polynomial* factor of $f(\tau\overline{x})$ from the approximate power series roots.

Case 1 : **Computing a linear factor of the form** $y - g(\overline{x})$: If the degree of the input polynomial is d, all the non-trivial factors have degree $\leq (d-1)$. So, if we compute the approximations of all the power series roots (wrt y) up to precision of degree $t = d-1$, then we can recover all the factors of the form $y - g(x_1, \ldots, x_n)$. Technically, this is supported by the uniqueness of the power series factorization (Proposition 1).

Case 2 : **Computing a monic non-linear factor**: Assume that a factor g of total degree t is of the form $y^k + c_{k-1}y^{k-1} + ... + c_1y + c_0$, where for all i, $c_i \in \mathbb{F}[\overline{x}]$. Now this factor

g also splits into linear (in y) factors above $\mathbb{F}[[\overline{x}]]$ and obviously these linear factors are also linear factors of the original polynomial $f(\tau\overline{x})$. So we have to take the right combination of some k power series roots, with their approximations (up to the degree t wrt $\overline{x}$), and take the product mod I^{t+1}. Note that if we only want to give an existential proof of the size bound of the factors, we need not find the combination of the power series roots forming a factor algorithmically. Doing it through brute-force search takes exponential time ($\binom{d}{k}$ choices). Interestingly, using a classical (linear algebra) idea due to Kaltofen, it can be done in randomized polynomial time. We will spell out the ideas later, while discussing the algorithm part of Theorem 4.

3.3 Approximating Roots

Once we are convinced that looking at approximate (power series) roots is enough, we need to investigate methods to compute them. We will now sketch two methods. The first one approximates all the roots *simultaneously* up to precision δ. The next ones approximate the roots *one at a time*. In the latter, multiplicity of the root plays an important role.

3.3.1 Recursive root finding via matrices (allRootsNI)

We *simultaneously* find the approximations of all the power series roots g_i of $f(\tau\overline{x})$. At each recursive step we get a better precision wrt degree. We show that knowing approximations $g_i^{<\delta}$, of g_i up to degree $\delta-1$, is enough to (simultaneously for all i) calculate approximations of g_i up to degree δ. This new technique, of finding approximations of the power series roots, is at the core of Theorem 2.

First, let us introduce a nice identity. For notational easiness, we assume that $f(\overline{x}, y) = \prod_i (y-g_i)^{\gamma_i}$ (i.e. relabel $f(\tau\overline{x})$). By applying the derivative operator ∂_y, we get a classic identity (which we call *logarithmic derivative identity*):

$$\frac{\partial_y f}{f} = \sum_i \frac{\gamma_i}{y-g_i}$$

Reduce the above identity modulo $I^{\delta+1}$ and define $\mu_i := g_i(\overline{0}) \equiv g_i \mod I$. This gives us (see Claim 31):

$$\frac{\partial_y f}{f} = \sum_{i=1}^{d_0} \frac{\gamma_i}{y-g_i} \equiv \sum_{i=1}^{d_0} \frac{\gamma_i}{y-g_i^{<\delta}} + \sum_{i=1}^{d_0} \frac{\gamma_i \cdot g_i^{=\delta}}{(y-\mu_i)^2} \mod I^{\delta+1}.$$

In terms of the d_0 unknowns $g_i^{=\delta}$, the above is a linear equation. (Note- We treat γ_i, μ_i's as known.) As y is a free variable above, we can fix it to d_0 "random" elements c_i in $\mathbb{F}$,

$i \in [d_0]$. One would expect these fixings to give a linear system with a unique solution for the unknowns. We can express the system of linear equations succinctly in the following matrix representation:

$$M \cdot v_\delta \;=\; W_\delta \bmod I^{\delta+1}$$

Here M is a $d_0 \times d_0$ matrix; each entry is denoted by $M(i,j) := \frac{\gamma_i}{(c_i-\mu_j)^2}$. Vector v_δ resp. W_δ is a $d_0 \times 1$ matrix where each entry is denoted by

$$v_\delta(i) := g_i^{=\delta}, \;\; W_\delta(i) := \frac{\partial_y f}{f}\Big|_{y=c_i} - G_{i,\delta}$$

where $G_{i,\delta} := \sum_{k=1}^{d_0} \frac{\gamma_k}{c_i - g_k^{<\delta}}$. We ensure that $\{c_i, \mu_i \mid i \in [d_0]\}$ are distinct, and show that the determinant of M is non-zero (Lemma 15). So, by knowing approximations up to $\delta-1$, we can recover δ-th part by solving the above system as $v_\delta \;=\; M^{-1}W_\delta \bmod I^{\delta+1}$. An important point is that the random c_i's will ensure: all the reciprocals involved in the calculation above do exist mod $I^{\delta+1}$.

Self-correction property: Does the above recursive step need an exact $g_i^{<\delta}$? We show the self correcting behavior of this process of root finding, i.e. in this iterative process there is no need to filter out the "garbage" terms of degree $\geq \delta$ in each step. If one has recovered g_i correct up to degree $\delta-1$, i.e. say we have calculated $g'_{i,\delta-1} \in \mathbb{F}(\overline{x})$ such that

$$g'_{i,\delta-1} \equiv g_i^{<\delta} \bmod I^\delta$$

and say we solve $M\tilde{v}_\delta = \widetilde{W}_\delta$ exactly, where

$$\widetilde{W}_\delta(i) := \frac{\partial_y f}{f}\Big|_{y=c_i} - \widetilde{G}_{i,\delta} \text{ and } \widetilde{G}_{i,\delta} := \sum_{k=1}^{d_0} \frac{\gamma_k}{c_i - g'_{k,\delta-1}}$$

Still, we can show that $g'_{i,\delta} := g'_{i,\delta-1} + \tilde{v}_\delta(i) \equiv g_i^{\leq\delta} \bmod I^{\delta+1}$ (Claim 32). So, we made progress in terms of the precision (wrt degree).

3.3.2 Rapid Newton Iteration with multiplicity

We show that from allRootsNI, we can derive a formula that finds $g_1^{<2^{t+1}}$ using *only* $g_1^{<2^t}$, i.e. the process has quadratic convergence and it does not involve roots other than g_1. Rewrite

$$\frac{\partial_y f}{f} = \sum_{i=1}^{d_0} \frac{\gamma_i}{y - g_i} = (1 + L_1) \cdot \frac{\gamma_1}{y - g_1}$$

where $L_1 := \sum_{1<i\leq d_0} \frac{\gamma_i}{y-g_i} \cdot \frac{y-g_1}{\gamma_1}$. This implies $\frac{f}{\partial_y f} = (1+L_1)^{-1} \cdot \frac{y-g_1}{\gamma_1}$. Now, if we put $y = y_t := g_1^{<2^t}$, then $y_t - g_i = g_1^{<2^t} - g_i$ is a unit in $\mathbb{F}[[\overline{x}]]$ for $i \neq 1$ ($\because$ it is a nonzero

constant mod I). Also,

$$y_t - g_1 = g_1^{<2^t} - g_1 \equiv 0 \mod I^{2^t} \implies L_1|_{y=y_t} \equiv 0 \mod I^{2^t}$$

Thus, $(L_1 \cdot (y - g_1))\,|_{y=y_t} \equiv 0 \mod I^{2^{t+1}}$. Hence,

$$\frac{f}{\partial_y f}|_{y=y_t} = \frac{y_t - g_1}{\gamma_1} \mod I^{2^{t+1}}$$

This shows the following :

Lemma 24 (Generalized Newton Iteration)**.** *if $f(\overline{x}, y) = (y-g)^e h$, where $h|_{y=g} \neq 0 \mod I$ and $e > 0$, then the power series for g can be approximated by the recurrence:*

$$y_{t+1} := y_t - e \cdot \frac{f}{\partial_y f}\bigg|_{y=y_t} \tag{3.2}$$

where $y_t \equiv g \mod I^{2^t}$.

This we call a *generalized Newton Iteration* formula, as it works with any multiplicity $e > 0$.

In fact, when $e = 1$, g is called a *simple root* of f; the above is an alternate proof of the classical Newton Iteration (NI) [New69] that finds a simple root in a recursive way (see Lemma 21). When *all* the roots are simple there are numerical methods to simultaneously approximate them [Dur60, Ker66, Ehr67, Abe73]. However, it is well known that NI *fails* to approximate the roots that repeat (see [Lec02]). In that case either NI is used on the function $f/\partial_y f$ or, though less frequently, the generalized NI is used in numerical methods (see [DB08, Eqn.6.3.13]).

At this point it might seem that generalized NI is stronger than classical NI but in the next two paragraphs we will establish that this is not the case; one can deduce the generalized version from the previous one in more than one way.

1) Another thing we can do is to look at $f^{1/e}$ and apply classical iteration to get the formula. See $f^{1/e}$ has order 1 at $y = g$. So, applying classical, one gets

$$y_{t+1} \equiv y_t - \frac{f^{1/e}}{\frac{1}{e} \cdot f^{1/e-1} \cdot \partial_y f} = y_t - e \cdot \frac{f}{\partial_y f}$$

2) Consider, $G = \frac{f}{f'}$. Using lemma 18 and 19, we have $\mathrm{ord}_g\left(\frac{f}{\partial_y f}\right) = 1$. Now, one can apply classical NI formula on this. We will have

$$y_{t+1} \equiv y_t - \frac{G}{G'}\bigg|_{y=y_t}$$

What we wanted to show was $y_t \equiv g \bmod I^{2^t}$. Hence, it suffices to prove (by inductive argument) that

$$e \cdot \frac{f}{f'}|_{y=y_t} \equiv \frac{G}{G'}\Big|_{y=y_t} \bmod I^{2^{t+1}}$$

From definition of G, it is clear that we need to prove that $G(G' - \frac{1}{e}) \equiv 0 \bmod I^{2^{t+1}}$. Observe that

$$\mathrm{ord}_g(G) = 1 \implies y - g \mid G \implies G(y_t) \equiv 0 \bmod I^{2^t}$$

Also observe that

$$\begin{aligned} G = \frac{1}{\frac{e}{y-g} + \frac{\partial_y h}{h}} &= \frac{y-g}{e} \cdot \frac{1}{1 + \frac{\partial_y h}{h} \cdot \frac{y-g}{e}} \\ &= \frac{y-g}{e} \cdot \left(1 - \left(\frac{\partial_y h}{h} \cdot \frac{y-g}{e}\right) + \left(\frac{\partial_y h}{h} \cdot \frac{y-g}{e}\right)^2 - \dots\right) \\ &= \frac{y-g}{e} + \text{rest} \end{aligned}$$

It is very clear that the "rest" is a power series which is of the form $\left(\frac{y-g}{e}\right)^2 \cdot A$ where A is a power series. Hence,

$$G' = \frac{1}{e} + \text{rest}$$

where rest is $\frac{y-g}{e} \cdot B$ for some power series B. Plugging $y = y_t$ in G', we have

$$G'(y_t) = \frac{1}{e} \bmod I^{2^t}$$

Hence, we have $G(G' - \frac{1}{e}) \equiv 0 \bmod I^{2^{t+1}}$ what we wanted to prove.

There is a technical point about our Eqn.3.2 when $e \geq 2$. The denominator $\partial_y f|_{y=y_t}$ is zero mod I, thus, its reciprocal does not exist! However, the ratio $(f/\partial_y f)\big|_{y=y_t}$ does exist in $\mathbb{F}[[\overline{x}]]$. On the other hand, if $e = 1$ then the denominator $\partial_y f|_{y=y_t}$ is nonzero mod I, thus, it is invertible in $\mathbb{F}[[\overline{x}]]$ and that is necessary for fast algebraic circuit computation (esp. division elimination).

We can compare the NI formula with the recurrence formula (which we call *slow* Newton Iteration) used in [DSY09, Eqn.5], [Oli16, Lem.4.1] for root finding. The slow NI formula is $Y_{t+1} = Y_t - \frac{f(\overline{x}, Y_t)}{\partial_y f(\overline{0}, Y_1)}$, where $Y_t \equiv g \bmod I^t$. The rate of convergence of this iteration is linear, as it takes δ many steps (instead of $\log \delta$) to get precision up to degree δ. One can also compare NI with other widespread processes like multifactor Hensel lifting [vzGG13, Sec.15.5], [Zas69] and the implicit function theorem paradigm [KP12, Sec.1.3], [KS16, PSS16]; however, we would not like to digress too much here as the latter concept

covers a whole lot of ground in mathematics.

We end up this chapter by discussing what is called accelerated Newton Iteration.

3.4 Algebraizing Accelerated Newton Iteration

This is inspired from the newton iteration observed in [LV16]. We try to see the algebraic version of the formula.

Suppose consider the power series factorization of f in $\mathbb{F}[[\overline{x}]][y]$ after the desired transformation $\tau : x_i \mapsto \alpha_i y + \beta_i x_i + \gamma_i$ for $\alpha_i, \beta_i, \gamma_i \in_r \mathbb{F}$,

$$f(\tau\overline{x}) = \prod_{i\in[m]} (y - g_i)^{e_i}$$

with $g_i(\overline{0}) = \mu_i$ and μ_i's are distinct. The goal is to find approximate g_i's.

Define

$$G_k(y) = \sum_{i\in[m]} \frac{e_i}{(y - g_i)^k}$$

Suppose we would like to approximate g_1. Recall the NI formula with multiplicity e_1. One can show that

$$y_{t+1} = y_t - e_1 \cdot \frac{f}{f'}\bigg|_{y=y_t} \tag{3.3}$$

works fine. Consider the following accelerated iteration formula (for $k \geq 1$):

Modified Formula 1

Define $y_0 := \mu_1$ *and*

$$y_{t+1} := y_t - \frac{G_k}{G_{k+1}}\bigg|_{y=y_t}$$

This formula 1 is interesting as there is no contribution of e_1 unlike for the NI with multiplicity e_1. We prove the following:

Lemma 25. *For* $k \geq 1$, $y_t \equiv g_1 \bmod I^{2^t} \implies y_{t+1} \equiv g_1 \bmod I^{2^{t+1}}$

Proof. We have

$$\left.\frac{G_k}{G_{k+1}}\right|_{y=y_t} = \frac{\sum_{i\in[m]} \frac{e_i}{(y_t-g_i)^k}}{\sum_{i\in[m]} \frac{e_i}{(y_t-g_i)^{k+1}}} = \frac{\left(\prod_{i\in[m]}(y_t-g_i)\right)\cdot\left(\sum_{i\in[m]}\left(\prod_{j\neq i}(y_t-g_j)^k\right)\cdot e_i\right)}{\sum_{i\in[m]}\left(\prod_{j\neq i}(y_t-g_j)^{k+1}\right)\cdot e_i}$$

$$= \frac{(y_t-g_1)\cdot\prod_{i\neq 1}(y_t-g_i)\left(\sum_{i\in[m]}\left(\prod_{j\neq i}(y_t-g_j)^k\right)\cdot e_i\right)}{\prod_{j\neq 1}(y_t-g_j)^{k+1}\cdot e_1}$$

The denominator contributes $\prod_{i\neq 1}(y_t-g_i)^{k+1}\cdot e_1$ which is when $i=1$. Observe that all other terms for $i\neq 1$ has $(y_t-g_1)^{k+1}$ which is in $I^{2^{t+1}}$ for $k\geq 1$. This in particular implies that one can write the denominator as $\prod_{i\neq 1}(y_t-g_i)^{k+1}\cdot e_1\left(1+I^{2^{t+1}}\right)$ as $\prod_{i\neq 1}(y_t-g_i)^{k+1}$ is invertible in $I^{2^{t+1}}$ because all the μ_j's are distinct. Hence,

$$\left.\frac{G_k}{G_{k+1}}\right|_{y=y_t} = \frac{(y_t-g_1)\cdot\left(\sum_{i\in[m]}\left(\prod_{j\neq i}(y_t-g_j)^k\right)\cdot e_i\right)}{\prod_{j\neq 1}(y_t-g_j)^k\cdot e_1}$$

Now look at the numerator. When $i\neq 1$, then each term has $(y_t-g_1)^k$ contributing. Already we have (y_t-g_1) outside multiplied. Hence those terms will have $(y_t-g_1)^{k+1}$ and it is 0 mod $I^{2^{t+1}}$. The only terms that remains in the sum is when $i=1$. But for $i=1$, this is exactly the denominator! Hence, they cancel out. Hence, we have

$$\left.\frac{G_k}{G_{k+1}}\right|_{y=y_t} \equiv y_t-g_1 \bmod I^{2^{t+1}} \implies y_{t+1}\equiv g_1 \bmod I^{2^{t+1}}$$

But, for this method to work for $k=0$, we should work with the following formula:

Modified Formula 2

Define $y_0 := \mu_1$ *and*

$$y_{t+1} := y_t - \frac{e_1}{\sum_{i\in[m]} e_i}\cdot\left.\frac{G_0}{G_1}\right|_{y=y_t}$$

One can prove similar as lemma 25. The reason being, $G_0 = \sum_{i \in [m]} e_i$ and $G_1 = \frac{f'}{f}$ and that is the NI with multiplicity formula. This above formula is also interesting as it involves all e_i's which seems to be not the case for NI with multiplicity as seen in equation 3.3. □

Chapter 4

GCD in Algebraic Complexity

This chapter is dedicated towards computing (or showing existence of) gcd with low complexity. In fact, for "practical" fields like $\mathbb{Q}, \mathbb{Q}_p$, or $\mathbb{F}_q$ for prime-power q. We use the notation $g \mid\mid f$ to denote that g divides f but g^2 does not divide f. Again, we denote $I := \langle x_1, \ldots, x_n \rangle$.

4.1 Computing GCD for bounded degree complexity classes

We will discuss a new method for computing gcd of two polynomials, which not only fits well in the algorithm but is also of independent interest. Of course, when computation is involved, we can only talk about polynomials with bounded degree as determining whether non-trivial gcd exists for a set of univariate polynomials with integer coefficient but high degree is NP-hard [Pla84].

Claim 26 (Computing gcd)**.** *Given two polynomials $f, g \in \mathbb{F}[\overline{x}]$ of degree d and computed by a circuit (resp. formula resp. ABP) of size s. One can compute a circuit (resp. formula resp. ABP) for $gcd(f, g)$, of size $poly(s, d)$ (resp. $\mathsf{poly}(s, d^{\log d})$), in randomized $\mathsf{poly}(s, d)$ (resp. $\mathsf{poly}(s, d^{\log d})$) time.*

Proof of Claim 26. The idea is the following. Suppose, $\gcd(f, g) =: h$ is of degree $d > 0$, then we will compute $h(\tau\overline{x})$ for a random map τ as in Theorem 1. We know wlog that

$$\tilde{f} := f(\tau\overline{x}) = \prod_i (y - A_i)^{a_i} \text{ and } \tilde{g} := g(\tau\overline{x}) = \prod_i (y - B_i)^{b_i}$$

where $A_i, B_i \in \mathbb{F}[[\overline{x}]]$. Since $\mathbb{F}[\overline{x}] \subset \mathbb{F}[[\overline{x}]]$ are UFDs (Proposition 1), we could say wlog that

$$h(\tau\overline{x}) = \prod_{i \in S} (y - A_i)^{\min(a_i, b_i)}$$

where $S = \{i \mid A_i = B_i\}$ after possible *rearrangement*. Now, as τ is a random invertible map, we can assume that, for $i \neq j$, $A_i \neq B_j$ and that $A_i(\overline{0}) \neq B_j(\overline{0})$ (Lemma 13). So, it

is enough to compute $A_i^{\overline{=}d}$ and $B_j^{\overline{=}d}$ and compare them using evaluation at $\overline{0}$. If indeed $A_i = B_i$, then $A_i^{\overline{=}d} = B_i^{\overline{=}d}$. If they are not, they mismatch at the constant term itself! Hence, we know the set S and so we are done once we have the power series roots with repetition.

Using univariate factoring, wrt y, we get all the multiplicities, of the roots, a_i and b_i's, additionally we get the corresponding starting points of classical Newton iteration, i.e. $A_i(\overline{0})$ and $B_i(\overline{0})$'s. Using NI, one can compute $A_i^{\overline{=}d}$ and $B_i^{\overline{=}d}$, for all i. Suppose, after rearrangement of A_i and B_i's (if necessary), we have

$$A_i = B_i \text{ for } i \in [s] =: S \text{ and } A_i \neq B_j \text{ for } i \in [s+1, d], j \in [s+1, d]$$

Lemma 13 can be used to deduce that $A_i(\overline{0}) \neq B_j(\overline{0})$ for $i, j \in [1, d] - S$. So, we have in fact

$$\gcd(\tilde{f}, \tilde{g}) = \prod_{i \in S} (y - A_i)^{\min(a_i, b_i)}$$

the index set S, the exponents and $A_i(\overline{0})$'s computed.

Size analysis: We compute $A_i^{\overline{=}d}$ and $B_i^{\overline{=}d}$ by NI, (possibly) after making the corresponding multiplicity one by differentiation. It is clear that at each NI step there will be a multiplicative d^2 blow up (due to interpolation, division and truncation). There are $\log d$ iterations in NI. Altogether the truncated roots have $\text{poly}(s, d^{\log d})$ size formula (resp. ABP). This directly implies that $\gcd(\tilde{f}, \tilde{g})$ has $\text{poly}(s, d^{\log d})$ size formula (resp. ABP). By taking the product of the linear factors, truncating to degree d, and applying τ^{-1}, we can compute the polynomial $\gcd(f, g)$. For circuit, all the blow ups are additional (not multiplicative as in the case for formula or abp). For $\log d$ steps, additional blow up happens. Hence, the size remains $\text{poly}(s, d)$.

Randomization is needed for τ and possibly for the univariate factoring over $\mathbb{F}$. Also, it is important to note that $\mathbb{F}$ may not be algebraically closed. Then one has to go to an extension, do the algebraic operations and return back to $\mathbb{F}$. For details, see Section 8.1. □

4.2 Complexity of Low Degree GCD

As mentioned above, we can not really talk about *computing* gcd in high degree regime. But still one can ask the following question:

Question 1

Does gcd (which is of low degree) of a set of arbitrary degree low complexity polynomials have low complexity?

Kaltofen in [Kal87, Theorem 4] answered this question positively. Surprisingly, the Euclidean algorithm does not enter in its proof, instead it is based on the so-called *EZ-GCD method* [MY73].

Theorem 7

Let $f_i \in \mathbb{F}[\overline{x}]$ for $i \in [m]$ and let $g = gcd(f_1, \ldots, f_m)$ such that $size(f_i) \leq s$. Then, $size(g) \leq \mathsf{poly}(s, d, m)$ where $d := deg(g)$

One problem with this theorem is the complexity of g depends on m. Hence, if m is super-$\mathsf{poly}(s)$, then $\mathsf{size}(g)$ is super-$\mathsf{poly}(s)$. Intuitively, the complexity should not really depend on m. In fact, If one assumes Factor conjecture 2, this implies theorem 7 should not depend on m. We in fact show that indeed complexity of g does not depend on m as stated in Chapter 1 theorem 5.

We give an alternative proof (based on NI technique that we built on) of an important theorem, already observed in [Bür04, Theorem 1.2] and using that, we prove theorem 5. We also present the original proof idea of kaltofen used to prove theorem 7. Before going any further, we state and prove the important theorem as mentioned above.

Theorem 8

Let $f \in \mathbb{F}[\overline{x}]$ such that $size(f) \leq s$. Let $g \in \mathbb{F}[\overline{x}]$ such that $f = g^e \cdot h$ with $gcd(g, h) = 1$. Then

$$size(g) \leq \mathsf{poly}(s, d, e)$$

where $deg(g) = d$.

Sketch of Proof. From theorem 1, we know that over $\mathbb{F}[[\overline{x}]]$,

$$f(\tau\overline{x}) = k \cdot \prod_{i \in [d_0]} (y - g_i)^{\gamma_i}$$

where $k \in \mathbb{F}^*$, $\gamma_i > 0$, and $g_i(\overline{0}) := \mu_i$. One can assume that

$$g = \prod_{j \in S} (y - g_j)$$

where $S \subset [d_0]$ such that $|S| = d$. One can argue that $\gamma_j = e$ whenever $j \in S$. One can differentiate $e-1$ times and apply NI to find $g_j^{\leq d}$ for all $j \in S$. It's not hard to show that $\text{size}(g) \leq \text{poly}(s, d, e)$. □

Before proving theorem 7, we present an important lemma which is in the heart of the proof of the theorem.

Lemma 27 ([Wan80]). *Suppose $f_i \in \mathbb{F}[\overline{x}]$ for $i \in [m]$ such that $g = gcd(f_1, \ldots, f_m)$. Then there exists $a_i \in \mathbb{F}$ such that*

$$gcd\left(g, \sum_{i\in[m]} a_i \cdot \frac{f_i}{g}\right) = 1$$

Proof. Consider the transformation σ which sends x_1 to x_1 and other x_i to $y_i + z_i x_1$ where y_i and z_i's are formal variables. Consider,

$$\text{Res}_{x_1}\left(g(\sigma\overline{x}), \sum_{i\in[m]} t_i \frac{f_i(\sigma\overline{x})}{g(\sigma\overline{x})}\right) := A(y_2, \ldots, y_n, z_2, \ldots, z_n, t_1, \ldots, t_m)$$

where t_i's are new variable. Of course, A is a non zero polynomial. We prove by contradiction. Suppose A is zero. This implies $g(\sigma\overline{x})$ and $\sum t_i \frac{f_i(\sigma\overline{x})}{g(\sigma\overline{x})}$ have non-trivial gcd say h which follows from proposition 2. Then, $h(\overline{y}, \overline{z}) \in \mathbb{F}[y_2, \ldots, y_n, z_2, \ldots, z_n]$ as $h \mid g(\sigma\overline{x})$. Now, σ is an isomorphism between $\mathbb{F}[\overline{x}]$ and $\mathbb{F}[x_1, y_2, \ldots, y_n, z_2, \ldots, z_n]$. Hence, $h(\sigma^{-1}(\overline{y}, \overline{z}))$ is a non-trivial gcd of g and $\sum t_i \frac{f_i}{g} \implies$

$$\text{gcd}(f_1, \ldots, f_m) = g \cdot h(\sigma^{-1}(\overline{y}, \overline{z}))$$

a contradiction!

Hence, there are $t_i = a_i$ such that A remains non-zero. As mentioned, σ is an isomorphism, so applying σ^{-1}, we will have $A(\sigma^{-1}(\overline{y}, \overline{z}))$ is a non-zero polynomial in $\mathbb{F}[\overline{x}]$. Hence,

$$\text{gcd}\left(g, \sum_{i\in[m]} a_i \cdot \frac{f_i}{g}\right) = 1$$

□

Proof of Theorem 7. Lemma 27 shows that there are a_i's such that

$$\text{gcd}\left(g, \sum_{i\in[m]} a_i \cdot \frac{f_i}{g}\right) = 1$$

Apply theorem 8 with $f = \sum a_i f_i$ and $e = 1$, this theorem clearly follows as $\text{size}(f) \leq s \cdot m$. □

Kaltofen in [Kal87, Theorem 3] proved the case of $e = 1$ of the theorem 8 and proved theorem 7. Now, we will see how the stronger theorem 8 will give us a stronger result, namely theorem 5 which removes dependency on m as desired.

Proof of theorem 5. Suppose $g = \gcd(f_1, \ldots, f_m)$ where $\text{size}(f_i) \leq s$ and $\deg(g) = d$. Suppose $g = g_1^{e_1} \ldots g_t^{e_t}$. Here is an important observation.

Observation 1. *There exists i such that $g_1^{e_1} || f_i$.*

Proof. Suppose not. Suppose $g_1^{e_1+1} | f_i$ for all i implies

$$g_1^{e_1+1} | \gcd(f_1, \ldots, f_m) = g$$

a contradiction. □

Hence, $f_i = g_1^{e_1} \cdot h_i$ with $\gcd(g_1, h_i) = 1$. We know that g_1 has $\text{poly}(s, \deg(g_1), e_1)$ size circuit from theorem 8. But of course $\deg(g_1)$ and $e_1 \leq d$. So, $\text{size}(g_1) \leq \text{poly}(s, d)$. This is true for each g_i. Hence $\text{size}(g) \leq \text{poly}(s, d)$. □

4.3 Strassen's Problem on computing Numerator and Denominator

Here is an interesting question regarding computing denominator and numerator from a division circuit:

Question 2

Consider an algebraic circuit C with division gates allowed and suppose it computes a rational polynomial, let us say $\frac{f}{g}$ where $f, g \in \mathbb{F}[\overline{x}]$ with $gcd(f, g) = 1$. Can we output f and g as algebraic circuits with no division gate allowed?

One can think of pushing the division gate at top by computing (u, v) at each gate (bottom to top) as done in lemma 7. One would have $f \cdot h$ and $g \cdot h$ calculated at the top. But the issue is how to calculate $\frac{f \cdot h}{h}$ as degree of h can be arbitrary.

Kaltofen in [Kal86] showed that using *pade approximation*, we can output f and g with high probability of size $\text{poly}(s, d + e)$ where $d = \deg(f)$ and $e = \deg(g)$. Interesting, he gave an alternative approach in [Kal87] which although asymptotically bigger size than the one in [Kal86], but serves the same importance nevertheless. We present the result which also uses theorem 7 and lemma 27.

Answer of Question 2. It is very easy to see that $f \cdot h$ and $g \cdot h$ have $\mathsf{poly}(s)$ size circuits. From lemma 27, we know that there exists $a_1, a_2, b_1, b_2 \in \mathbb{F}$ with $a_1 \cdot b_2 - a_2 \cdot b_1 \neq 0$ such that

$$\gcd(h, a_1 \cdot f + b_1 \cdot g) = 1$$

and

$$\gcd(h, a_2 \cdot f + b_2 \cdot g) = 1$$

Using theorem 7, one can deduce that $a_1 \cdot f + b_1 \cdot g$ and $a_2 \cdot f + b_2 \cdot g$ both have circuits of size $\mathsf{poly}(s, \max(d, e))$. From linear combinations, one can find f and g which trivially will have circuits of size $\mathsf{poly}(s, d, e)$. □

Chapter 5

Closure of restricted complexity classes

This chapter is dedicated towards proving closure results for certain algebraic complexity classes. As mentioned, for the time being we will assume that the field is algebraically closed. We give efficient randomized algorithm to output the complete factorization of polynomials belonging to that class (stated as Theorem 36).

Proof of Theorem 4. There are essentially two parts in the proof. The first part talks only about the existential closure results. In the second part, we discuss the algorithm.

Proof of closure: Given f of degree d, we randomly shift by $\tau : x_i \mapsto x_i + y\alpha_i + \beta_i$. From Theorem 1 we have that $\tilde{f}(\overline{x}, y) := f(\tau\overline{x})$ splits like $\tilde{f} = \prod_{i=1}^{d_0}(y - g_i)^{\gamma_i}$, with $g_i(\overline{0}) =: \mu_i$ being distinct. Here is the detailed size analysis of the factors of polynomials represented by various models of our interest.

Size analysis for formula: Suppose f has a formula of size $n^{O(\log n)}$. To show size bound for all the factors, it is enough to show that the approximations of the power series roots, i.e. $g_i^{\leq d}$ has size $n^{O(\log n)}$ size formula. This follows from the reduction of factoring to approximations of power series roots.

We differentiate $\tilde{f}$ wrt y, $(\gamma_i - 1)$ many times, so that the multiplicity of the root we want to recover becomes exactly one. The differentiation would keep the size $\text{poly}(n^{\log n})$ (Lemma 8). Now, we have $(y - g_i) \mid\mid \tilde{f}^{(\gamma_i - 1)}$ by repeated use of lemma 19 and we can apply classical Newton iteration formula. For all $0 \leq t \leq \log d + 1$, we compute A_t and B_t such that $A_t / B_t \equiv g_i \bmod I^{2^t}$. Moreover, B_t is invertible in $\mathbb{F}[[\overline{x}]]$ ($\because g_i$ is a simple root of $\tilde{f}^{(\gamma_i - 1)}$).

To implement this iteration using the formula model, each time there would be a blow up of d^2. Note that in a formula, there can be many copies of the same variable in the leaf nodes and if we want to feed something in that variable, we have to make equally many copies. That means we may need to make s ($= \text{size}(f)$) many copies at each step. We claim that it can be reduced to only d^2 many copies.

We can pre-compute (with blow up at most $\mathsf{poly}(sd)$) all the coefficients $C_0, \ldots, C_d$ wrt y, given the formula of $\tilde{f} =: C_0 + C_1 y + \ldots + C_d y^d$ using interpolation. We can do the same for the derivative formula. For details on this interpolation trick, see [Sap16, Lem.5.3]. Using interpolation, we can convert the formula of $\tilde{f}$ and its derivative to the form $C_0 + C_1 y + \ldots + C_d y^d$. In this modified formula, there are $O(d^2)$ many leaves labelled as y. So in the modified formula of the polynomial $\tilde{f}$ and in its derivative, we are computing and plugging in (for y) d^2 copies of $g_i^{<2^t}$ to get $g_i^{<2^{t+1}}$. This leads to d^2 blow up at each step of the iteration.

As B_t's are invertible, we can keep track of the division gates across iterations and, in the end, eliminate them causing a one-time size blow up of $\mathsf{poly}(sd)$ (Lemma 7).

Now, assume that $\mathrm{size}(A_t, B_t) \leq S_t$. Then we have $S_{t+1} \leq O(d^2 S_t) + \mathsf{poly}(sd)$. Finally, we have $S_{\log d+1} = \mathsf{poly}(sd) \cdot d^{2\log d} = \mathsf{poly}(n^{\log n})$.

Hence, $g_i^{\leq d} \equiv A_{\log d+1}/B_{\log d+1} \bmod I^{d+1}$ has $\mathsf{poly}(n^{\log n})$ size formula, and so does every polynomial factor of f after applying τ^{-1}.

Size analysis for ABP: This analysis is similar to that of the formula model, as the size blow up in each NI iteration for differentiation, division, and truncation (to degree $\leq d$) is the same as that for formulas. A noteworthy difference is that we need to eliminate division in *every* iteration (Lemma 6) and we cannot postpone it. This leads to a blow up of d^4 in each step. Hence, $S_{\lg d+1} = \mathsf{poly}(sd) \cdot d^{4\log d} = \mathsf{poly}(n^{\log n})$.

Size analysis for VNP: Suppose f can be computed by a verifier circuit of size, and witness size, $n^{O(\log n)}$. We call both the verifier circuit size and witness size as size parameter. Now, our given polynomial $\tilde{f}$ has $n^{O(\log n)}$ size parameters. As before, it is enough to show that $g_i^{\leq d}$ has $n^{O(\log n)}$ size parameters.

For the preprocessing (taking $\gamma_i - 1$-th derivative of $\tilde{f}$ wrt y), the blow up in the size parameters is only $\mathsf{poly}(n^{\log n})$. Now we analyze the blow up due to classical Newton iteration. We compute A_t and B_t such that $A_t/B_t \equiv g_i \bmod I^{2^t}$. Using the closure properties of VNP(discussed in Section 2.6), we see that each time there is a blow up of d^4. The main reason for this blow up is due to the *composition* operation, as we are feeding a polynomial into another polynomial.

Assume that the verifier circuit $\mathrm{size}(A_t, B_t) \leq S_t$ and witness size $\leq W_t$. Then we have $S_{t+1} \leq O(d^4 S_t) + \mathsf{poly}(n^{\log n})$. So, finally we have $S_{\log d+1} = \mathsf{poly}(sd) \cdot d^{4\log d} = \mathsf{poly}(n^{\log n})$. It is clear that $g_i^{\leq d} \equiv A_{\log d+1}/B_{\log d+1} \bmod I^{d+1}$ has $\mathrm{poly}(n^{\log n})$ size verifer circuit. Same analysis works for W_t and witness size remains $n^{O(\log n)}$. Moreover, we get the corresponding bounds for every polynomial factor of f after applying τ^{-1}.

Randomized Algorithm. We give the broad steps of our algorithm below. We are given $f \in \mathbb{F}[\bar{x}]$, of degree $d > 0$, as input.

1. Choose $\overline{\alpha}, \overline{\beta} \in_r \mathbb{F}^n$ and apply $\tau : x_i \to x_i + \alpha_i y + \beta_i$. Denote the transformed polynomial $f(\tau\overline{x})$ by $\tilde{f}(\overline{x}, y)$. Wlog, from Theorem 1, $\tilde{f}$ has factorization of the form $\prod_{i=1}^{d_0}(y - g_i)^{\gamma_i}$, where $\mu_i := g_i(\overline{0})$ are distinct.

2. Factorize $\tilde{f}(\overline{0}, y)$ over $\mathbb{F}[y]$. This will give γ_i and μ_i's.

3. Fix $i = i_0$. Differentiate $\tilde{f}$, wrt y, $(\gamma_{i_0} - 1)$ many times to make g_{i_0} a simple root.

4. Apply Newton iteration (NI), on the differentiated polynomial, for $k := \lceil \log(2d^2 + 1) \rceil$ iterations; starting with the approximation $\mu_{i_0} \pmod I$. We get $g_{i_0}^{<2^k}$ at the end of the process (mod I^{2^k}).

5. Apply the transformation $x_i \mapsto Tx_i$ (T acts as a degree-counter). Consider $\tilde{g}_{i_0} := g_{i_0}^{<2^k}(T\overline{x})$. Solve the following homogeneous linear system of equations, over $\mathbb{F}[\overline{x}]$, in the unknowns u_{ij} and v_{ij}'s,

$$\sum_{0 \le i+j < d} u_{ij} \cdot y^i T^j \;=\; (y - \tilde{g}_{i_0}) \cdot \sum_{\substack{0 \le i < d \\ 0 \le j < 2^k}} v_{ij} \cdot y^i T^j \mod T^{2^k}.$$

   Solve this system, using Lemma 5, to get a nonzero polynomial (if one exists) $u := \sum_{0 \le i+j < d} u_{ij} \cdot y^i T^j$.

6. If there is no solution, return "f is irreducible".

7. Otherwise, find the minimal solution wrt $\deg_y(u)$ by brute force (try all possible degrees wrt y; it is in $[d-1]$).

8. Compute $G(\overline{x}, y, T) := \gcd_y(u(\overline{x}, y, T), \tilde{f}(T\overline{x}, y))$ using Claim 26.

9. Compute $G(\overline{x}, y, 1)$ and transform it by $\tau^{-1} : x_i \mapsto x_i - \alpha_i y - \beta_i$, $i \in [n]$, and $y \mapsto y$. Output this as an irreducible polynomial factor of f.

Claim 28 (Existence)**.** *If f is reducible, then the linear system (Step 5) has a non-trivial solution.*

Proof of Claim 28. If f is reducible, then let $f = \prod f_i^{e_i}$ be its prime factorization. Assume wlog that $y - g_{i_0} \mid \tilde{f}_1 := f_1(\tau\overline{x})$. Of course $0 < \deg_y(\tilde{f}_1) = \deg(f_1) < d$.

Observe that we are done by picking u to be $\tilde{f}_1(T\overline{x}, y)$. For, total degree of f_1 is $< d$, and so that of $\tilde{f}_1(T\overline{x}, y)$ wrt the variables y, T is $< d$.

Moreover, $y - g_{i_0} \mid \tilde{f}_1 \implies \tilde{f}_1 = (y - g_{i_0})v$, for some $v \in \mathbb{F}[[\overline{x}]][y]$ with $\deg_y < d$. Hence, $\tilde{f}_1 \equiv (y - g_{i_0}^{<2^k}) \cdot v \mod I^{2^k} \implies u \equiv (y - \tilde{g}_{i_0}) \cdot v(T\overline{x}, y) \mod T^{2^k}$. This shows the existence of a nontrivial solution of the linear system (Step 5). □

Now, we show that if the linear system has a solution, then the solution corresponds to a non-trivial polynomial factor of f.

Claim 29 (Step 8's success)**.** *If the linear system (Step 5) has a non-trivial solution, then* $0 < \deg_y G \leq \deg_y u < d$.

Proof of Claim 29. Suppose (u, v) is the solution provided by the algorithm in Lemma 5 (u being in the unknown LHS and v being the unknown RHS). Consider $G = \gcd_y(u, \tilde{f}(Tx, y))$. We know that there are polynomials a and b such that $au + b\tilde{f}(Tx, y) = \mathrm{Res}_y(u, \tilde{f}(Tx, y))$ (Section 2.4). Consider $\deg_T(\mathrm{Res}_y(u, \tilde{f}(Tx, y))$. As degree of T in u and $\tilde{f}(Tx, y)$ can be at most d, hence degree of T in Resultant can be atmost $2d^2$ (Section 2.4). Clearly, $\deg_y G \leq \deg_y u < d$. If $\deg_y G = 0$ then the resultant of $u, \tilde{f}(T\overline{x}, y)$ wrt y will be nonzero (Proposition 2). Suppose the latter happens.

Now, we have $u = (y - \tilde{g}_{i_0})v \bmod T^{2^k}$. Since $y - g_{i_0} \mid \tilde{f}$ we get that $y - g_{i_0}(T\overline{x}) \mid \tilde{f}(T\overline{x}, y)$. Assume that $\tilde{f}(Tx, y) =: (y - g_{i_0}(T\overline{x})) \cdot w$.

Thus, we can rewrite the previous equation as: $au + b\tilde{f}(T\overline{x}, y) \equiv (y - \tilde{g}_{i_0})(av + bw) \equiv \mathrm{Res}_y(u, \tilde{f}(Tx, y)) \bmod T^{2^k}$. Note that the latter is nonzero mod T^{2^k} because the resultant is a nonzero polynomial of $\deg_T < 2^k$. Putting $y = \tilde{g}_{i_0}$ the LHS vanishes, but RHS does not ($\because$ it is independent of y). This gives a contradiction.

Thus, $\mathrm{Res}_y(u, \tilde{f}(Tx, y) = 0$. This implies that $0 < \deg_y G < d$. □

Next we show that if one takes the minimal solution u (wrt degree of y), then it will correspond to an irreducible factor of f. We will use the same notation as above.

Claim 30 (Irred. factor)**.** *Suppose* $y - g_{i_0} \mid \tilde{f}_1$ *and* f_1 *is an irreducible factor of* f*. Then,* $G = c \cdot \tilde{f}_1(Tx, y)$*, for* $c \in \mathbb{F}^*$*, and* $\deg_y(G) = \deg_y(u) = \deg_y(f_1)$ *in Step 8.*

Proof of Claim 30. Suppose f is reducible, hence as shown above, G is a non-trivial factor of $\tilde{f}(T\overline{x}, y)$. Recall that $\tilde{f}(T\overline{x}, y) = \prod_i (y - g_i(T\overline{x}))^{\gamma_i}$ is a factorization over $\mathbb{F}[[\overline{x}, T]]$. We have that $y - \tilde{g}_{i_0} \mid G \bmod T^{2^k}$. Thus, $y - g_{i_0}(T\overline{x}) \mid G$ absolutely ($\because$ the power series ring is a UFD and use Theorem 1). So, $y - g_{i_0}(T\overline{x}) \mid \gcd_y(G, \tilde{f}_1(T\overline{x}, y))$ over the power series ring. Since, $\tilde{f}_1(T\overline{x}, y)$ is an irreducible polynomial, we can deduce that $\tilde{f}_1(T\overline{x}, y) \mid G$ in the polynomial ring. So, $\deg_y(f_1) \leq \deg_y(G)$.

We have $\deg_y(\tilde{f}_1(T\overline{x}, y)) = \deg(f_1) =: d_1$. By the above discussion, the linear system in Step 7 will not have a solution of $\deg_y(u)$ below d_1. Let us consider the linear system in Step 7 that wants to find u of $\deg_y = d_1$. This system has a solution, namely the one with $u := \tilde{f}_1(T\overline{x}, y) \bmod T^{2^k}$. Then, by the above claim, we will get the G as well in the subsequent Step 8. This gives $\deg_y(G) \leq \deg_y(u) = d_1$. With the previous inequality we get $\deg_y(G) = \deg_y(u) = \deg_y(f_1)$. In particular, G and $\tilde{f}_1(Tx, y)$ are the same up to a nonzero constant multiple. □

Alternative to Claim 26: The above proof (Claim 30) suggests that the gcd question of Step 8 is rather special: One can just write u as $\sum_{0 \leq i \leq d_1} c_i(\overline{x}, T) y^i$ and then compute

the polynomial $G = \sum_{0 \leq i \leq d_1} (c_i / c_{d_1}) \cdot y^i$ as a formula (resp. ABP), by eliminating division (Lemma 6).

Once we have the polynomial G we can fix $T = 1$ and apply τ^{-1} to get back the irreducible polynomial factor f_1 (with power series root g_{i_0}).

The running time analysis of the algorithm is by now routine. If we start with an f computed by a formula (resp. ABP) of size $n^{O(\log n)}$, then as observed before, one can compute $\tilde{g}_{i_0}$ which has $n^{O(\log n)}$ size formula (resp. ABP). This takes care of Steps 1-4.

Now, solve the linear system in Steps 5-7 of the algorithm. Each entry of the matrix is a formula (resp. ABP) size $n^{O(\log n)}$. The time complexity is similar by invoking Lemma 5.

Steps 8 is to compute gcd of two $n^{O(\log n)}$ size formulas (resp. ABPs) which again can be done in $n^{O(\log n)}$ time giving a size $n^{O(\log n)}$ formula (resp. ABP) as discussed above.

This completes the randomized $\mathsf{poly}(n^{\log n})$-time algorithm that outputs $n^{O(\log n)}$ sized factors.

□

Remarks.

1. The above results hold true for the classes $VBP(s), VF(s), \mathsf{VNP}(s)$ for any size function $s = n^{\Omega(\log n)}$. The above analysis actually tells us that factors of a polynomial f computed by a formula (resp. ABP) of size s and degree d can be *computed* by a $\mathsf{poly}(s, d^{O(\log d)})$ size formula (resp. ABP); more over there is a randomized $\mathsf{poly}(s, d^{O(\log d)})$ time algorithm that can output any factor. Same holds for VNP (only the size).

2. The above analysis implies that VP is uniform closed under factoring. One can view this as an alternative proof to the famous result in [Kal89].

3. By the above remark, our result can be extended to prove closure result for polynomials in VNPwith *constant* individual degree. There are very interesting polynomials in this class, namely Permanent.

5.1 PIT is equivalent to factoring

In [KSS15], Kopparty et al showed that the problem of *deterministically factoring multivariate polynomials* reduces to the problem of *deterministic polynomial identity testing* (PIT). Specifically, given an arithmetic circuit (either explicitly or via black-box access) that computes a polynomial $f(\bar{x})$, the task of computing arithmetic circuits for the factors of f can be solved deterministically, given a deterministic algorithm for the PIT. For the blackbox model, one can use **effective and efficient Hilbert irreducibility** and [Kal90] together to deduce the

equivalence. For details, see [KSS15, section 1.2.1].

For the white box model, of course it is easy to observe that factoring implies PIT. To show the converse implication, we use the algorithm described above. The randomization in the algorithm comes for the following reasons :

1. choosing the "random points" for linear transformation
2. solving linear system of equation (see, section 5)

Observe that random point choosing was nothing but selecting points where a certain polynomial won't vanish; this in particular implies that if we have a deterministic algorithm for PIT, randomization in the transformation step can be removed. Solving the linear system can also be derandomized using PIT as the randomization came in determining whether a symbolic determinant was 0 or not. In totality, one could deduce that factoring can be derandomized assuming deterministic algorithm for PIT.

5.2 Factors of constant individual degree polynomials have small complexity

This section is dedicated to showing complexity bound on the factors of polynomials which has constant individual degree and computed by a size s formula (resp. circuit). Let us denote $\mathsf{VF}_{\text{const}}$ to be a set of family of polynomials so that $(f_n) \in \mathsf{VF}_{\text{const}} \implies \deg_{x_i}(f_n) \leq r$ for some constant r and every $n \in \mathbb{N}$ where f_n can be computed by formula of size $\mathsf{poly}(n)$. Similarly, one can define $\mathsf{VP}_{\text{const}}, \mathsf{VBP}_{\text{const}}, \mathsf{VNP}_{\text{const}}$ for circuits, ABP's and VNP polynomials. One could show the following.

Theorem 9

$\mathsf{VP}_{const}, \mathsf{VBP}_{const}, \mathsf{VF}_{const}, \mathsf{VNP}_{const}$ *are closed under factoring.*

This was originally shown for circuits and formulas in [Oli16] where he additionally showed that the depth of the factors also get increased by a constant. Here, we will not be talking about small depth factors as we are interested in closure results. Using the version 2 of split theorem (see theorem 23) and the **reversal** technique as used in [Oli16] one could establish the closure result. For details, see [Oli16].

Remarks. The same result and technique also holds for polynomial class in VNP as mentioned above which is interesting in the sense that there are very interesting polynomials in this class, namely Permanent. Similar technique can be used to establish that VNP is closed under factoring in [CKS18].

Chapter 6

Complexity of factor and square-freeness

This chapter proves theorem 2 and theorem 3. The proofs are self contained and we assume for the sake of simplicity that the underlying field $\mathbb{F}$ is algebraically closed and has characteristic 0. When this is not the case, we discuss the corresponding theorems in chapter 8. As mentioned earlier, whole point of these efforts it towards proving the factor conjecture 2. Let us first ask a simpler case for the high degree case.

6.1 Special case $f = g^e$

Suppose, we are given f size that $\text{size}(f) \leq s$. Suppose $f = g^e$ (i.e. as in the factor conjecture 2, $h = 1$). Does the factor conjecture hold? i.e. can we show that $\text{size}(g) \leq \text{poly}(s, d)$ where $d = \deg(g)$? Kaltofen in [Kal87] positively answered the question.

Theorem 10

Suppose $f = g^e \in \mathbb{F}[\overline{x}]$ is a polynomial such that $\text{size}(f) \leq s$. Then, $\text{size}(g) \leq \text{poly}(s, d)$ where $d = \deg(g)$.

We will show a slightly different approach (different from the original proof due to kaltofen) to solve this problem although fundamentally they are equivalent. The above theorem requires the field size to be large enough (for prime characteristic p) and e to be non-multiple of p.

Proof of theorem 10. Without loss of generality, one can assume that $f(\overline{0}) = 1$ by shifting and scaling. This implies that $g(\overline{0}) = 1$. Observe that,

$$g = f^{1/e} = (1 + (f-1))^{1/e} = 1 + \frac{1}{e} \cdot (f-1) + \binom{1/e}{2} \cdot (f-1)^2 + \ldots \binom{1/e}{d} (f-1)^d + \ldots$$

Observe that $f-1 \equiv 0 \bmod \langle \overline{x} \rangle$. Hence $(f-1)^i$ where $i \geq d+1$ contributes terms bigger degree d. So, one can rightly deduce that

$$g = 1 + \frac{1}{e} \cdot (f-1) + \binom{1/e}{2} \cdot (f-1)^2 + \ldots \binom{1/e}{d} (f-1)^d \bmod \langle \overline{x} \rangle^{d+1}$$

Using lemma 7, we certainly have $\mathsf{size}(g) \leq \mathsf{poly}(s,d)$. □

6.2 Complexity of factors polynomially related to degree of radical: Proof of Theorem 2

As seen in the previous section, we could ask similar question that if $f = g_1^{e_1} g_2^{e_2}$ can be computed be a size s circuit and $\deg(g_1)$ and $\deg(g_2)$ are both bounded by d, what can we say about size of g_1 and g_2? One would understand that it is not possible to generalize the above proof for two different factors. In this section, we use Theorem 1 and allRootsNI to partially solve the case of circuits with exponential degree (stated in [Kal86] and studied in [Kal87, Bür04]).

Proof of Theorem 2. From the hypothesis $f = u_0 u_1$. Define $\deg(f) =: d$. Suppose $u_1 = h_1^{e_1} \ldots h_m^{e_m}$, where h_i's are coprime irreducible polynomials. Let d_0 be the degree of $\mathrm{rad}(u_1) = \prod_i h_i$. Note that $\deg(h_i), m \leq d_0$ and the multiplicity $e_i \leq d \leq s^{O(s)}$, where s is the size bound of the input circuit. Thus, to get the size bound of any factor of u_1, it is enough to show that for each i, h_i has a circuit of size $\mathsf{poly}(sd_0)$.

Using Theorem 1, we have $\tilde{f}(\overline{x}, y) := f(\tau \overline{x}) = k \cdot u_0(\tau \overline{x}) \cdot \prod_{i \in [d_0]} (y - g_i)^{\gamma_i}$, with $g_i(\overline{0}) := \mu_i$ being distinct. From Corollary 22 we deduce that

$$h_i(\tau \overline{x}) = k_i \prod_{i \in [d_0]} (y - g_i^{\leq d_0})^{\delta_i} \bmod I^{d_0+1}$$

with ideal $I := \langle x_1, \ldots, x_n \rangle$, exponent $\delta_i \in \{0,1\}$ and nonzero $k_i \in \mathbb{F}$. We can get h_i by applying τ^{-1}. Hence, it is enough to bound the size of $g_i^{\leq d_0}$.

Let $\tilde{u_0} := u_0(\tau \overline{x})$. From the repeated applications of Leibniz rule of the derivative ∂_y, we deduce, by recalling : $\partial_y(FG) = (\partial_y F)G + F(\partial_y G)$

$$\frac{\partial_y \tilde{f}}{\tilde{f}} = \frac{\partial_y \tilde{u_0}}{\tilde{u_0}} + \sum_{i=1}^{d_0} \frac{\gamma_i}{y - g_i}$$

At this point we move to the formal power series, so that the reciprocals can be approximated as polynomials. Note that $y - g_i$ is invertible in $\mathbb{F}[[\overline{x}]]$ when y is assigned any value $c_i \in \mathbb{F}$ which is not equal to μ_i. We intend to find $g_i \bmod I^\delta$ inductively, for

all $\delta \geq 1$. We assume that μ_i's and γ_i's are known. Suppose, we have recovered up to $g_i \bmod I^\delta$ and we want to recover $g_i \bmod I^{\delta+1}$. The relevant recurrence, for $\delta \geq 1$, is:

Claim 31 (Recurrence). $\sum_{i=1}^{d_0} \gamma_i \cdot \frac{g_i^{=\delta}}{(y-\mu_i)^2} \equiv \frac{\partial_y \tilde{f}}{\tilde{f}} - \frac{\partial_y \tilde{u_0}}{\tilde{u_0}} - \sum_i \frac{\gamma_i}{(y - g_i^{<\delta})} \bmod I^{\delta+1}$.

Proof of Claim 31. Using a power series calculation (Lemma 17), we have

$$\frac{1}{y-g_i} \equiv \frac{1}{y-\left(g_i^{<\delta}+g_i^{=\delta}\right)} \equiv \frac{1}{y-g_i^{<\delta}} + \frac{g_i^{=\delta}}{(y-\mu_i)^2} \bmod I^{\delta+1}$$

Multiplying by γ_i and summing over $i \in [d_0]$, the claim follows. □

By knowing approximation up to the $\delta-1$ homogeneous parts of g_i, we want to find the δ-th part by solving a linear system. For concreteness, assume that we have a rational function $g'_{i,\delta-1} := C_{i,\delta-1}/D_{i,\delta-1}$ such that $g'_{i,\delta-1} \equiv g_i^{<\delta} \bmod I^\delta$. Next, we show how to compute $g_i^{\leq\delta}$.

We recall the process as outlined in allRootsNI (Section 3.3.1). In the free variable y, we plug-in d_0 random field value c_i's and get the following system of linear equations: $M \cdot v_\delta = W_\delta$, where M is a $d_0 \times d_0$ matrix with (i,j)-th entry, $M(i,j) := \gamma_j/(c_i-\mu_j)^2$. Column v_δ resp. W_δ is a $d_0 \times 1$ matrix whose i-th entry is denoted $v_\delta(i)$ resp. $(\partial_y \tilde{f}/\tilde{f} - \partial_y \tilde{u_0}/\tilde{u_0})|_{y=c_i} - \tilde{G}_{i,\delta}$, where $\tilde{G}_{i,\delta} := \sum_{j=1}^{d_0} \gamma_j/(c_i - g'_{j,\delta-1})$. Think of the solution v_δ as being both in $\mathbb{F}(\overline{x})^{d_0}$ and in $\mathbb{F}[[\overline{x}]]^{d_0}$; both the views help.

Now we will prove two interesting facts. First, M is invertible (Lemma 15). Second, define $g'_{i,0} := \mu_i$ and, for $\delta \geq 1$, $g'_{i,\delta} := g'_{i,\delta-1} + v_\delta(i)$. Then, $g'_{i,\delta}$ approximates g_i well:

Claim 32 (Self-correction). *Let $i \in [d_0]$ and $\delta \geq 0$. Then, $g'_{i,\delta} \equiv g_i^{\leq\delta} \bmod I^{\delta+1}$.*

Proof of Claim 32. We prove this by induction on δ. It is true for $\delta = 0$ by definition. Suppose it is true for $\delta-1$. This means we have $g'_{i,\delta-1} \equiv g_i^{<\delta} \bmod I^\delta$ for all i. Let us write $g'_{i,\delta-1} =: g_i^{<\delta} + A_{i,\delta} + A'_{i,\delta}$, where $A'_{i,\delta} \equiv 0 \bmod I^{\delta+1}$ and $A_{i,\delta}$ is homogeneous of degree δ. Hence, for $i \in [d_0]$, the linear constraint is: $\sum_{j=1}^{d_0} \gamma_j \cdot v_\delta(j)/(c_i-\mu_j)^2 \equiv \partial_y \tilde{f}/\tilde{f} - \partial_y \tilde{u_0}/\tilde{u_0} - \sum_j \gamma_j/(c_i - g'_{j,\delta-1}) \bmod I^{\delta+1}$.

The "garbage" term $A_{j,\delta}$ in RHS can be isolated using Lemma 17 as: $1/(c_i - g'_{j,\delta-1}) \equiv \frac{1}{c_i - \left(g_j^{<\delta} + A_{j,\delta}\right)} \equiv 1/(c_i - g_j^{<\delta}) + A_{j,\delta}/(c_i-\mu_j)^2 \bmod I^{\delta+1}$. So, we get:

$$\sum_{j=1}^{d_0} \frac{\gamma_j \cdot v_\delta(j)}{(c_i-\mu_j)^2} \equiv \frac{\partial_y \tilde{f}}{\tilde{f}} - \frac{\partial_y \tilde{u_0}}{\tilde{u_0}} - \sum_{j=1}^{d_0} \frac{\gamma_j}{c_i - g_j^{<\delta}} - \sum_{j=1}^{d_0} \frac{\gamma_j \cdot A_{j,\delta}}{(c_i-\mu_j)^2} \bmod I^{\delta+1}.$$

Rewriting this, using Claim 31, we get:

$$\sum_{j=1}^{d_0} \frac{\gamma_j}{(c_i-\mu_j)^2}\left(v_\delta(j)+A_{j,\delta}\right) \equiv \sum_{j=1}^{d_0} \frac{\gamma_j}{(c_i-\mu_j)^2}\cdot g_j^{=\delta} \bmod I^{\delta+1}.$$

Thus,

$$\sum_{j=1}^{d_0} \gamma_j \cdot \frac{(v_\delta(j)+A_{j,\delta}-g_j^{=\delta})}{(c_i-\mu_j)^2} \equiv 0 \bmod I^{\delta+1}$$

As we vary $i \in [d_0]$ we deduce, by Lemma 15, that $v_\delta(j)+A_{j,\delta}-g_j^{=\delta} \equiv 0 \bmod I^{\delta+1}$. This implies

$$g'_{j,\delta} = g'_{j,\delta-1} + v_\delta(j) \equiv (g_j^{<\delta}+A_{j,\delta})+(g_j^{=\delta}-A_{j,\delta}) = g_j^{\leq\delta} \bmod I^{\delta+1}$$

This proves it for all $j \in [d_0]$. □

Size analysis: Here we give the overall process of finding factors using allRootsNI technique and analyze the circuit size needed at each step to establish the size bound of the factors. As discussed before, we need to analyze only the power series root approximation $g_i^{\leq\delta}$ or $g'_{i,\delta}$.

At the $(\delta-1)$-th step of allRootsNI process, we have a multi-output circuit (with division gates) computing $g'_{i,\delta-1}$ as a rational function, for all $i \in [d_0]$. Specifically, let us assume that $g'_{i,\delta-1} =: C_{i,\delta-1}/D_{i,\delta-1}$, where $D_{i,\delta-1}$ is invertible in $\mathbb{F}[[\overline{x}]]$. So, the circuit computing $g'_{i,\delta-1}$ has a division gate at the top that outputs $C_{i,\delta-1}/D_{i,\delta-1}$. We would eliminate this division gate only in the end (see the standard Lemma 7). Now we show how to construct the circuit for $g'_{i,\delta}$, given the circuits for $g'_{i,\delta-1}$.

From $v_\delta = M^{-1}W_\delta$, it is clear that there exist field elements β_{ij} such that

$$v_\delta(i) = \sum_{j=1}^{d_0} \beta_{ij} W_\delta(j) = \sum_{j=1}^{d_0} \beta_{ij}\left((\frac{\partial_y \tilde{f}}{\tilde{f}} - \frac{\partial_y \imath\tilde{u}_0}{\imath\tilde{u}_0})|_{y=c_j} - \tilde{G}_{j,\delta}\right)$$

Initially we precompute, for all $j \in [d_0]$, $(\partial_y\tilde{f}/\tilde{f} - \partial_y\imath\tilde{u}_0/\imath\tilde{u}_0)|_{y=c_j}$: Note that $\partial_y\tilde{f}$ has $\mathsf{poly}(s)$ size circuit (high degree of the circuit does not matter, see Lemma 8). Invertibility of $\tilde{f}|_{y=c_j}$ and $\imath\tilde{u}_0|_{y=c_j}$ follows from the fact that we chose c_j's randomly. In particular, $\tilde{f}(\overline{0},y)$, and so $\imath\tilde{u}_0(\overline{0},y)$, have roots in $\mathbb{F}$ which are distinct from c_j, $j \in [d_0]$. Thus, $\tilde{f}(\overline{x},c_j)$ and $\imath\tilde{u}_0(\overline{x},c_j)$ have non-zero constants and so are invertible in $\mathbb{F}[[\overline{x}]]$. Similarly, $\gamma_\ell/(c_j-g'_{\ell,\delta-1})$ exists in $\mathbb{F}[[\overline{x}]]$.

Thus, the matrix recurrence allows us to calculate the polynomials $C_{i,\delta}$ and $D_{i,\delta}$, given their $\delta-1$ analogues, by adding $\mathsf{poly}(d_0)$ many wires and nodes. The precomputations costed us size $\mathsf{poly}(s,\delta)$. Hence, both $C_{i,\delta}$ and $D_{i,\delta}$ has $\mathsf{poly}(s,\delta,d_0)$ sized circuit.

We can assume we have only one division gate at the top, as for each gate G we can keep track of numerator and denominator of the rational function computed at G, and simulate all the algebraic operations easily in this representation. When we reach precision $\delta = d_0$, we can eliminate the division gate at the top. As D_{i,d_0} is a unit, we can compute its inverse using the power series inverse formula and approximate only up to degree d_0 (Lemma 6). Finally, the circuit for the polynomial $g_i^{\leq d_0} \equiv C_{i,d_0}/D_{i,d_0} \bmod I^{d_0+1}$, for all $i \in [d_0]$, has size $\mathsf{poly}(s, d_0)$.

Altogether, it implies that any factor of u_1 has a circuit of size $\mathsf{poly}(s, d_0)$. □

Remark: It is worth observing that from theorem 1, to solve *factor conjecture* it is enough to show that if $f = (y-g)^e \cdot h$ for some $g, h \in \mathbb{F}[[\overline{x}]]$, then $\mathrm{size}(g^{\leq d}) \leq \mathsf{poly}(\mathrm{size}(f), d)$. In order to do that one could argue that there might be way to compute $\frac{\delta^{e-1} f}{\delta y}$ in $\mathsf{poly}(\log e, \mathrm{size}(f))$ circuit. That would solve *factor conjecture* as then one could simply do the Newton iteration. As e can be at most doubly exponential in n, we would be fine. But one can show that there is no hope to doing differentiation (however clever you are unless $\mathsf{VP} = \mathsf{VNP}$) using lemma 9.

6.3 Low degree factors of general circuits: Proof of Theorem 3

Here, we introduce an approach to handle the general case when $\mathrm{rad}(f)$ has exponential degree. We show that allowing a special kind of modular division gate gives a small circuit for any low degree factor of f.

The *modular division* problem is to show that if f/g has a representative in $\mathbb{F}[[\overline{x}]]$, where polynomials f and g can be computed by a circuit of size s, then $f/g \bmod \langle \overline{x}^d \rangle$ can be computed by a circuit of size $\mathsf{poly}(sd)$. Note that if g is invertible in $\mathbb{F}[[\overline{x}]]$, then the question of modular division can be solved using Strassen's trick of division elimination [Str73]. But, in our case g is not invertible in $\mathbb{F}[[\overline{x}]]$ (though f/g is well-defined).

Proof of Theorem 3. As discussed before, to show size bound for an arbitrary factor (with low degree) of f, it is enough to show the size bound for the approximations of power series roots. From Theorem 1, $\tilde{f}(\overline{x}, y) = f(\tau\overline{x}) = k \cdot \prod_{i=1}^{d_0} (y - g_i)^{\gamma_i}$, with $g_i(\overline{0}) := \mu_i$ being distinct.

Fix an i from now on. To calculate $g_i^{\leq \delta}$, we iteratively use Newton iteration with multiplicity for $\log \delta + 1$ many times. We know that there are rational functions $\hat{g}_{i,t}$ such that $\hat{g}_{i,t+1} := \hat{g}_{i,t} - \gamma_i \cdot \frac{\tilde{f}}{\partial_y \tilde{f}}\big|_{y=\hat{g}_{i,t}}$ and $\hat{g}_{i,t} \equiv g_i \bmod \langle \overline{x} \rangle^{2^t}$. We compute $\hat{g}_{i,t}$'s incrementally, $0 \leq t \leq \log \delta + 1$, by a circuit with division gates. As before, $\tilde{f}$ and $\partial_y \tilde{f}$ have $\mathsf{poly}(s)$ size circuits.

If $\hat{g}_{i,t}$ has S_t size circuit with division, then $S_{t+1} = S_t + O(1)$. Hence, $\hat{g}_{i,\lg\delta+1}$ has $\mathsf{poly}(s, \log\delta)$ size circuit with division.

By keeping track of numerator and denominator of the rational function computed at each gate, we can assume that the only division gate is at the top. As the size of $\hat{g}_{i,\log\delta+1}$ was initially $\mathsf{poly}(s, \log\delta)$ with intermediate division gates, it is easy to see that when division gates are pushed at the top, it computes A/B with size of both A and B still $\mathsf{poly}(s, \log\delta)$.

Finally, a degree δ polynomial factor $h|f$ will require us to estimate $g_i^{\leq\delta}$ for that many i's. Thus, such a factor has $\mathsf{poly}(s\delta)$ size circuit, using a single modular division. □

Chapter 7

Closure of Approximative Complexity classes

In this chapter, we show that all our closure results, under factoring, can be naturally generalized to corresponding approximative algebraic complexity classes.

In complexity theory it has proven useful to study *"approximative algorithms"*, which use arithmetic with infinite precision and nevertheless only give us an approximation of the solution to be computed, however with any precision required. This systematically emerged in early works on matrix multiplication (the notion of border rank, see [BCS13]). It is also an important concept in the geometric complexity theory program (see [GMQ16]). The notion of approximative complexity can be motivated through two ways, *topological* and *algebraic* and both the perspectives are known to be equivalent. Both allow us to talk about the *convergence* $\epsilon \to 0$.

In what follows, we can see ϵ as a formal variable and $\mathbb{F}(\epsilon)$ as the function field. For an algebraic complexity class C, the approximation is defined as follows [BIZ17, Defn.2.1].

Definition 33 (Approximative closure of a class [BIZ17])**.** *Let C be an algebraic complexity class over field $\mathbb{F}$. A family (f_n) of polynomials from $\mathbb{F}[\overline{x}]$ is in the* class $\overline{C}(\mathbb{F})$ *if there are polynomials $f_{n;i}$ and a function $t : \mathbb{N} \mapsto \mathbb{N}$ such that g_n is in the class C over the field $\mathbb{F}(\epsilon)$ with $g_n(\overline{x}) = f_n(\overline{x}) + \epsilon f_{n;1}(\overline{x}) + \epsilon^2 f_{n;2}(x) + \ldots + \epsilon^{t(n)} f_{n;t(n)}(\overline{x})$.*

The above definition can be used to define closures of classes like VF, VBP, VP, VNPwhich are denoted as $\overline{\mathsf{VF}}$, $\overline{\mathsf{VBP}}$, $\overline{\mathsf{VP}}$, $\overline{\mathsf{VNP}}$ respectively. In these cases one can assume wlog that the degrees of g_n and $f_{n;i}$ are $\mathsf{poly}(n)$.

Following Bürgisser [Bür01]:- Let $K := \mathbb{F}(\epsilon)$ be the rational function field in variable ϵ over the field $\mathbb{F}$. Let R denote the subring of K that consists of rational functions defined in $\epsilon = 0$. Eg. $1/\epsilon \notin R$ but $1/(1+\epsilon) \in R$.

Definition 34. *[Bür01, Defn.3.1] Let $f \in \mathbb{F}[x_1, \ldots, x_n]$. The* approximative complexity $\overline{size}(f)$ *is the smallest number r, such that there exists F in $R[x_1, \ldots, x_n]$ satisfying $F|_{\epsilon=0} = f$ and circuit size of F over* constants *K is $\leq r$.*

Note that the circuit of F may be using division by ϵ implicitly in an intermediate step. So, we cannot simply assign $\epsilon = 0$ and get a circuit free of ϵ. Also, the degree involved can be arbitrarily large wrt ϵ. Thus, potentially $\overline{\text{size}}(f)$ can be smaller than $\text{size}(f)$.

Using this new notion of size one can define the analogous class $\overline{\mathsf{VP}}$. It is known to be closed under factors [Bür01, Thm.4.1]. The idea is to work over $\mathbb{F}(\epsilon)$, instead of working over $\mathbb{F}$, and use Newton iteration to approximate power series roots. Note that in the case of $\overline{\mathsf{VF}}$, $\overline{\mathsf{VBP}}$, $\overline{\mathsf{VP}}$ and $\overline{\mathsf{VNP}}$ the polynomials have $\text{poly}(n)$ degree. So, by using repeated differentiation, we can assume the power series root (of $\tilde{f} := f(\tau\bar{x})$) to be simple (i.e. multiplicity$= 1$) and apply classical NI. We need to carefully analyze the implementation of this idea.

Root finding using NI over K**.** For degree-d $f \in \mathbb{F}[\bar{x}]$ if $\overline{\text{size}}(f) = s$ then: $\exists F \in R[\bar{x}]$ with a size s circuit satisfying $F|_{\epsilon=0} = f$. The degree of F wrt $\bar{x}$ may be greater than d. In that case we can extract the part up to degree d and truncate the rest [Bür04, Prop.3.1]. So wlog $\deg_{\bar{x}}(F) = \deg(f)$.

By applying a random τ (using constants $\mathbb{F}$) we can assume that $\tilde{F} := F(\tau\bar{x}) \in R[\bar{x}, y]$ is *monic* (i.e. leading-coefficient, wrt y in $\tilde{F}$, is invertible in R). Otherwise, $\deg_y(\tilde{F}) = \deg_y(\tilde{f}) = \deg_{\bar{x}}(f)$ will decrease on substituting $\epsilon = 0$ contradicting $F|_{\epsilon=0} = f$. Wlog, we can assume that the leading-coefficient of $\tilde{F}$ wrt y is 1 and the y-monomial's degree is d. From now on we have $\tilde{F}|_{\epsilon=0} = \tilde{f}$ and both have their leading-coefficients 1 wrt y.

Let μ be a root of $\tilde{f}(\bar{0}, y)$ of multiplicity one (as discussed before). Since $\tilde{F}(\bar{0}, y) \equiv \tilde{f}(\bar{0}, y) \bmod \epsilon$, we can build a power series root $\mu(\epsilon) \in \mathbb{F}[[\epsilon]]$ of $\tilde{F}(\bar{0}, y)$ using NI, with μ as the starting point. But $\mu(\epsilon)$ may not converge in K. To overcome this obstruction [Bür01] devised a clever trick.

Define $\hat{F} := \tilde{F}(\bar{x}, y + \mu + \epsilon) - \tilde{F}(\bar{0}, \mu + \epsilon)$. Note that $(\bar{0}, 0)$ is a simple root of $\hat{F}(\bar{x}, y)$ [Bür04, Eqn.5]. So, a power series root y_∞ of $\hat{F}$ can be built iteratively by classic NI (Lemma 21):

$$y_{t+1} := y_t - \left.\frac{\hat{F}}{\partial_y \hat{F}}\right|_{y=y_t}.$$

Where, $y_\infty \equiv y_t \bmod \langle \bar{x} \rangle^{2^t}$. One can easily prove that y_t is defined over the coefficient field K, using induction on t.

Note that $\hat{F}|_{\epsilon=0} = \tilde{f}(\bar{x}, y + \mu) - \tilde{f}(\bar{0}, \mu) = \tilde{f}(\bar{x}, y + \mu)$. So, y_∞ is associated with a root of $\tilde{f}$ as well. This implies that by using several such roots y_∞, we can get an appropriate product $\hat{G} \in R[\bar{x}, y]$, such that an actual polynomial factor of $\tilde{f}$ (over field $\mathbb{F}$) equals $\hat{G}|_{\epsilon=0}$.

The above process, when combined with the first part of the proof of Theorem 4, does imply:

Theorem 35 (Approximative factors)**.** *The approximative complexity classes $\overline{VF}(n^{\log n})$, $\overline{VBP}(n^{\log n})$ and $\overline{\mathsf{VNP}}(n^{\log n})$ are closed under factors.*

The same question for the classes $\overline{\mathsf{VF}}$, $\overline{\mathsf{VBP}}$ we leave as an open question. (Though, for the respective bounded individual-degree polynomials we have the result as before.) Also one can use the reversal technique (used in [Oli16]) to show that $\overline{\mathsf{VNP}}$ is closed under factoring as well!

Remarks.

1. $\overline{\mathsf{VP}}$ is closed under factoring. This was already observed in [Bür04]. The above sketch is sufficient to prove this result.

2. Let $\mathbb{F}$ be a field of characteristic zero and assume that g is an irreducible factor of degree d and multiplicity e of a polynomial $f \in \mathbb{F}[\overline{x}]$. Then, one can show that

$$\overline{\mathsf{size}}(g) = \mathsf{poly}(\overline{\mathsf{size}}(f), \deg(g))$$

   This in particular implies that factor conjecture 2 is true with respect to approximative complexity. For proof details, see [Bür04, Theorem 4.1].

Chapter 8

Factoring in Field Extension

Upto chapter 7, we used the fact that $\mathbb{F}$ is algebraically closed and characteristic 0. This was partly because for split theorem, we wanted to have the univariate polynomials to factor completely into linear ones. What happens when the field is not that "nice enough"? Namely, what if $\mathbb{F}$ is not algebraically closed or it is of characteristic non-zero? The next two sections are dedicated towards these issues and factoring results on these fields.

8.1 When field $\mathbb{F}$ is not algebraically closed

We show that all our results "partially" hold true for fields $\mathbb{F}$ which are not algebraically closed. The common technique used in all the proofs is the structural result (Theorem 1) which talks about power series roots with respect to y. Recall that we use a random linear map $\tau : x_i \mapsto x_i + \alpha_i y + \beta_i$, where $\alpha_i, \beta_i \in_r \mathbb{F}$, to make the input polynomial f monic in y and the individual degree of y equal to $d := \deg(f)$. If we set all the variables to zero except y, we get a univariate polynomial $\tilde{f}(\overline{0}, y)$ whose roots we are interested in finding explicitly.

The other common technique in our proofs is the classical NI, which starts with just one field root, say μ_1 of $\tilde{f}(\overline{0}, y)$, and builds the full power series on it. Let $E \subsetneq \overline{\mathbb{F}}$ be the smallest field where a root μ_1 can be found. Say, $g | \tilde{f}_1(\overline{0}, y)$ is the minimal polynomial for μ_1. The degree of the extension $E := \mathbb{F}[z]/(g(z))$ is at most d. So, computations over E can be done efficiently. The key idea is to view $E/\mathbb{F}$ as a vector space and simulate the arithmetic operations over E by operations over $\mathbb{F}$. The details of this kind of simulation can be seen in [vzGG13]. In circuits it means that we make $\deg(E/\mathbb{F})$ copies of each gate and simulate the algebraic operations on these 'tuples' following the $\mathbb{F}$-module structure of $E[\overline{x}]$.

Once we have found all the power series roots of $\tilde{f}(\overline{x}, y)$ over $E[[\overline{x}]]$, say starting from each of the conjugates $\mu_1, \ldots, \mu_i \in E$, it is easy to get a polynomial factor in $E[\overline{x}, y]$. This factor will not be in $\mathbb{F}[\overline{x}, y]$, unless E is a splitting field of $\tilde{f}_1(\overline{0}, y)$. A more practical method is: While solving the linear system over E in Steps 5-7 (Algorithm in Theorem 4) we can

demand an $\mathbb{F}$-solution u. Basically, at the level of algorithm in Lemma 5, we can rewrite the linear system $Mw = (\sum_{0\le i\le d} M_i z^i)\cdot w = 0$ as $M_i w = 0$ ($i \in [0,d]$), where the entries of the matrix M_i are given as formulas (resp. ABP) computing a $\mathsf{poly}(n)$ degree polynomial in $\mathbb{F}[\overline{x}]$. This way we get the desired $\mathbb{F}$-solution u. Then, Steps 8-9 will yield an irreducible polynomial factor of f in $\mathbb{F}[\overline{x}, y]$. This sketches the following more practical version of Theorem 4.

Theorem 36. *For $\mathbb{F}$ a number field, a local field, or a finite field (with characteristic $>\deg(f)$), there exists a randomized $\mathsf{poly}(sn^{\log n})$-time algorithm that: for a given $n^{O(\log n)}$ size formula (resp. ABP) f of $\mathsf{poly}(n)$-degree and bitsize s, outputs $n^{O(\log n)}$ sized formulas (resp. ABPs) corresponding to each of the nontrivial factors of f.*

Note that over these fields there are famous randomized algorithms to factor univariate polynomials in the base case, see [vzGG13, Part III] & [Pau01].

The allRootsNI method in Theorem 2 seems to require all the roots $\mu_i, i \in [d_0]$, to begin with. Let $\tilde{u}_1 := \mathrm{rad}(u_1(\tau\overline{x}))$. Since μ_i's are in the *splitting field* $E \subset \overline{\mathbb{F}}$ of $\mathrm{rad}(\tilde{u}_1(\overline{0}, y))$, we do indeed get the size bound of the power series roots $g_i^{\le d_0}$ of $\tilde{u}_1$ assuming the constants from E. As seen in the proof, any irreducible polynomial factor $\tilde{h}_i := h_i(\tau\overline{x})$ of $\mathrm{rad}(\tilde{u}_1)$ is some product of these $(y - g_i^{\le d_0})$'s mod I^{d_0+1}. So, for the polynomial $\tilde{h}_i$ in $\mathbb{F}[\overline{x}, y]$ we get a size upper bound over constants E. We leave it as an open question to transfer it over constants $\mathbb{F}$ (note: $E/\mathbb{F}$ can be of exponential degree).

8.2 Multiplicity issue in prime characteristic

The main obstruction in prime characteristic is when the multiplicity of a factor is a p-multiple, where $p \ge 2$ is the characteristic of $\mathbb{F}$. In this case, all versions of Newton iteration fail. This is because the derivative of a p-powered polynomial vanishes. When p is greater than the degree of the input polynomial, these problems do not occur, so all our theorems hold (also see Section 8.1).

When p is smaller than the degree of the input polynomial in Theorem 4, adapting an idea from [KSS15, Sec.3.1], we claim that we can give $n^{O(\lambda \log n)}$-sized formula (resp. ABP) for the p^{e_i}-th power of f_i, where f_i is a factor of f whose multiplicity is divisible exactly by p^{e_i}, and λ is the number of distinct p-powers that appear.

Note that presently it is an open question to show that: If a circuit (resp. formula resp. ABP) of size s computes f^p, then f has a $\mathrm{poly}(sp)$-sized circuit (resp. formula resp. ABP).

Theorem 4 can be extended to all characteristic as follows.

Theorem 37. *Let $\mathbb{F}$ be of characteristic $p \ge 2$. Suppose the $\mathsf{poly}(n)$-degree polynomial given by a $n^{O(\log n)}$ size formula (resp. ABP) factors into irreducibles as $f(\overline{x}) = \prod_i f_i^{p^{e_i} j_i}$, where $p \nmid j_i$. Let $\lambda := \#\{e_i | i\}$.*

Then, there is a poly$(n^{\lambda \log n})$-size formula (resp. ABP) computing $f_i^{p^{e_i}}$ over $\overline{\mathbb{F}}_p$.

Proof sketch. Note that $\lambda = O(\log_p n)$.

Let the transformed polynomial of degree d split into power series roots as follows: $\tilde{f} := f(\tau\overline{x}, y) = \prod_{i=1}^{d_0}(y - g_i)^{\gamma_i}$.

$p \nmid \gamma_i$: If g_i is such that $p \nmid \gamma_i$, then we can find the corresponding power series roots using Newton iteration and recover all such factors. After recovering all such irreducible polynomial factors, we can divide $\tilde{f}$ by their product. Let $G := \tilde{f} / \prod_{p \nmid \gamma_i}(y - g_i)^{\gamma_i}$. Clearly, G is now a p-power polynomial.

$p \mid \gamma_i$: Computing the highest power of p that divides the exponent of G (given by a formula resp. ABP) is easy. First, write the polynomial as $G = c_0 + c_1 y + ... + c_d y^d$ using interpolation. Note that it is a p^e-th power iff: $c_i = 0$ whenever $p^e \nmid i$, and p^{e+1} does not have this property. After computing the right value of p^e, we can reduce factoring to the case of a non-p-power.

Rewrite G as $\hat{G} := \sum_{p^e | i} c_i(\overline{x}) \cdot y^{i/p^e}$, i.e. replacing y^{p^e} by y. Clearly, g is an irreducible factor of G iff $\hat{g}$ is an irreducible factor of $\hat{G}$.

We can now apply NI to find the roots of $\tilde{G}$, that have multiplicity coprime to p. Divide by their product and then repeat the above.

Size analysis. If G can be computed by a size s formula (resp. ABP), $\hat{G}$ can be computed by a size $O(d^2 s)$ formula (resp. ABP). Similarly, a single division gate leads to a blow up by a factor of $O(d^2)$. The number of times we need to eliminate division is at most $\lambda \log d$. So the overall size is $n^{O(\lambda \log n)}$.

However, the splitting field E where we get all the roots of $\tilde{f}(\overline{0}, y)$ may be of degree $\Omega(d!)$. So, we leave the efficiency aspects of the algorithm as an open question. □

High degree case. Note that the above idea cannot be implemented efficiently in the case of high degree circuits. Still we can extend our Theorem 2 using allRootsNI. The key observation is that the allRootsNI formula still holds but the summands that appear are exactly the ones corresponding to g_i with $\gamma_i \neq 0 \bmod p$.

This motivates the definition of a partial radical: $\text{rad}_p(f) := \prod_{p \nmid e_i} f_i$, if the prime factorization of f is $\prod_i f_i^{e_i}$.

Theorem 38. *Let $\mathbb{F}$ be of characteristic $p \geq 2$. Let $f = u_0 u_1$ such that size(f)+size(u_0) $\leq s$. Any factor of $rad_p(u_1)$ has size poly($s + \deg(rad_p(u_1))$) over $\overline{\mathbb{F}}$.*

Proof idea: Observe that the roots with multiplicity divisible by p do not contribute to the allRootsNI process. So, the process works with $\text{rad}_p(u_1)$ and the linear algebra complexity involved is polynomial in its degree.

Chapter 9

Sparsity bound of factors

9.1 General Upper Bound

This section deals with sparsity upper bounds on factors.

Theorem 39. *If g is a multi-linear polynomial which divides $f \in \mathbb{F}[\overline{x}]$, then, $||g|| \leq ||f||$.*

Proof. Proof goes via simple induction on the number of variables n. Of course, base case $n = 1$ is trivial. Suppose it is true for $n \leq k$. We prove that it is true for $n = k + 1$. Suppose $g(x_1, \ldots, x_k, y) := Ay + B \mid \sum_{i=0}^{r} C_i y^i := f(x_1, \ldots, x_k, y)$ where A and B are both multi-linear polynomials. Then $A \mid C_r$ and $B \mid C_0$. Hence $||A|| \leq ||C_r||$ and $||B|| \leq ||C_0||$. Hence $||g|| = ||Ay + B|| = ||A|| + ||B|| \leq ||C_r|| + ||C_0|| \leq ||f||$. □

Theorem 40. *If g is a multi-quadratic polynomial which divides another multi-quadratic polynomial $f \in \mathbb{F}[\overline{x}]$, then, $||g|| \leq ||f||$.*

Proof. Proof goes via simple induction on the number of variables n. Of course, base case $n = 1$ is trivial. Suppose it is true for $n \leq k$. We prove that it is true for $n = k + 1$. Suppose $g(x_1, \ldots, x_k, y) := b_2 y^2 + b_1 y + b_0 \mid \sum_{i=0}^{r} a_2 y^2 + a_1 y + a_0 := f(x_1, \ldots, x_k, y)$ where a_i and b_i's are multi-quadratic polynomials. If $b_2 \neq 0$, then it must be the case that

$$\frac{a_2}{b_2} = \frac{a_1}{b_1} = \frac{a_0}{b_0} = G$$

where G is a polynomial. This implies, by induction hypothesis that,

$$||b_i|| \leq ||a_i|| \implies ||g|| = \sum ||b_i|| \leq \sum ||a_i|| = ||f||$$

If $b_2 = 0$, then $b_1 \mid a_2$ and $b_0 \mid a_0$ and similarly the theorem follows. □

The immediate question is whether we can extend this to constant individual degree. For the time being, it seems that there is no technique known to handle this type of situation (very recently Bhargav etal in [BSV18] have used Newton polytopes and an approximate

version of the classical Caratheodory's Theorem to show non-trivial upper bound on the sparsity of the factor which would give quasi polynomial i.e. $s^{O(\log n)}$ bound on the sparsity of factor of n variate, constant individual degree s sparsity polynomial).

9.2 Sparsity bound for special class of Polynomials

Recall, conjecture 4. We also showed that the conjecture is optimal in the parameter sense $|logr$ for the following example:

$$f = \prod_{i=1}^{n}(x_i^r - 1) \qquad g = \prod_{i=1}^{n}(x_i^{r-1} + x_i^{r-2} + \ldots + 1)$$

In fact the belief is that these are the bad cases. So, let us consider the set of polynomials where

$$\deg_{x_i}(f) = r \text{ or } 0$$

We will prove that the conjecture is true for these set of polynomials. For notational purpose let us define

$$S_{n,r} = \{f | f \in \mathbb{C}[x_1, \ldots, x_n] \text{ with } |\text{var}(f)| = n \text{ such that degree}_{x_i}(f) = r \text{ or } 0\}$$

We prove that for $f \in S_{n,r}$, the conjecture is actually true. Before proceeding , we need some observations and lemmas and an important theorem.

Observation 2. *If $f \in S_{n,r}$, then there exists $P \in S_{n,1}$ such that $f(x_1, \ldots, x_n) = P(x_1^r, \ldots, x_n^r)$*

Now we invoke [Kar89, Theorem 1.6] which implies the following theorem:

Theorem 41. *Let $\mathbb{F}$ be an arbitary field such that $\sqrt{-1} \in \mathbb{F}$. Let $n \geq 1$ and let $a \in \mathbb{F}$. Then $X^n - a$ is reducible over $\mathbb{F}$ if and only if there exists a prime $p \mid n$ such that $a \in \mathbb{F}^p$.*

Proof. $\Longleftarrow$ is obvious . For $\Longrightarrow$, suppose $X^n - a$ is reducible but $a \notin \mathbb{F}^p$ for any $p \mid n$. In particular $a \notin \mathbb{F}^p$ for any odd prime $p \mid n$ and $a \notin \mathbb{F}^2$. Then certainly, $a \notin -4\mathbb{F}^4$. So that implies $X^n - a$ is irreducible a contradiction ! □

Lemma 42. *Suppose $f = Ax_n^r - B$ where $A, B \in \mathbb{C}[x_1, \ldots, x_{n-1}]$ and A and B has no non-trivial gcd (i.e. if $h|A, B$ then h must be a unit) , is reducible . Then there exists two polynomials $u, v \in \mathbb{C}[x_1, \ldots, x_{n-1}]$ and a prime $p|r$ such that $A = u^p$ and $B = v^p$.*

Proof. Consider the field of fraction $\mathbb{C}(x_1, \ldots, x_{n-1})$. We observe the following :

Observation 3. *$f = Ax_n^r - B$ is reducible over $\mathbb{C}[x_1, \ldots, x_{n-1}] \Longleftrightarrow x_n^r - \frac{B}{A}$ is reducible over $\mathbb{C}(x_1, \ldots, x_{n-1})$.*

Using observation 3, we have that $x_n^r - \frac{B}{A}$ is reducible over $\mathbb{C}(x_1, \ldots, x_{n-1})$. As $\sqrt{-1} \in \mathbb{C}(x_1, \ldots, x_{n-1})$, using theorem 41, we can say that $\exists \lambda \in \mathbb{C}(x_1, \ldots, x_{n-1})$ and $p \mid r$ such that

$$\frac{B}{A} = \lambda^p$$

Suppose $\lambda = \frac{v'}{u'}$ where $u', v' \in \mathbb{C}[x_1, \ldots, x_{n-1}]$. Then we have

$$\frac{B}{A} = \frac{v'^p}{u'^p}$$

As A and B has no non-trivial gcd , hence we must have $A = cu'^p = u^p$ and $B = cv'^p = v^p$ for some $c \in \mathbb{C}$ and $u, v \in \mathbb{C}[x_1, \ldots, x_{n-1}]$. □

Observation 4. *For a polynomial f over any field $\mathbb{F}$, $f(x_1^m, \ldots, x_n^m) = 0 \iff f(x_1, \ldots, x_n) = 0$.*

Lemma 43. *If $\exists u \in \mathbb{C}[x_1, \ldots, x_n]$ such that $u^2 \mid f(x_1^m, \ldots, x_n^m)$ for some integers $m, n \geq 1$ and $f \in S_{n,1}$, then $\|u\| = 1$.*

Proof. If $u \in \mathbb{C}$, then we are obviously done. Otherwise , we prove this by inducting on n .

Base Case ($n = 1$) : Suppose $f(x_1) = Ax_1 + B$ where $A, B \in \mathbb{C}$. If

$$u^2 \mid f(x_1^m) = Ax_1^m + B$$

All roots of $Ax_1^m + B = 0$ are precisely

$$\left(-\frac{B}{A}\right)^{\frac{1}{m}} \zeta_m^i \qquad \forall 1 \leq i \leq m$$

where ζ_m is m-th primitive root of unity . They are all distinct if $B \neq 0$ and hence it can not have repeated roots i.e. $u^2 \nmid Ax_1^m + B$ unless $B = 0$. But $B = 0 \implies u = ax_1^i$ for some $0 \leq i \leq \frac{m}{2}$ and $a \in \mathbb{C}$. Hence $\|u\| = 1$.

Induction Hypothesis : Suppose this is true for $\leq n - 1$ variate polynomials.

Induction Step : Suppose

$$f(x_1, \ldots, x_n) = A(x_2, \ldots, x_n)x_1 + B(x_2, \ldots, x_n)$$

where $A, B \in S_{\leq n-1,1}$. Hence

$$f(x_1^m, \ldots, x_n^m) = A(x_2^m, \ldots, x_n^m)x_1^m + B(x_2^m, \ldots, x_n^m) = A(x_2^m, \ldots, x_n^m)\left(x_1^m + \frac{B(x_2^m, \ldots, x_n^m)}{A(x_2^m, \ldots, x_n^m)}\right)$$

Clearly $A(x_2^m, \ldots, x_n^m)$ and $x_1^m + \frac{B(x_2^m, \ldots, x_n^m)}{A(x_2^m, \ldots, x_n^m)}$ has no non-trivial gcd in $C(x_2, \ldots, x_n)[x_1]$. Now consider this field extension

$$C' = C(x_2, \ldots, x_n)\left(\left(\frac{B(x_2^m, \ldots, x_n^m)}{A(x_2^m, \ldots, x_n^m)}\right)^{\frac{1}{m}}\right)$$

As in C' , roots of $x_1^m + \frac{B(x_2^m, \ldots, x_n^m)}{A(x_2^m, \ldots, x_n^m)} = 0$ are precisely

$$\left(-\frac{B(x_2^m, \ldots, x_n^m)}{A(x_2^m, \ldots, x_n^m)}\right)^{\frac{1}{m}} \zeta_m^i \forall 1 \leq i \leq m$$

which are distinct if $B(x_2^m, \ldots, x_n^m) \neq 0$. In that case we must have, from induction hypothesis that

$$u^2 \mid A(x_2^m, \ldots, x_n^m) \implies \|u\| = 1$$

If $B(x_2^m, \ldots, x_n^m) = 0$, then using observation 4 , we have $B(x_2, \ldots, x_n) = 0$ i.e.

$$f(x_1, \ldots, x_n) = A(x_2, \ldots, x_n)x_1$$

In that case ,

$$u^2 \mid f(x_1^m, \ldots, x_n^m) = A(x_2^m, \ldots, x_n^m)x_1^m \implies u = ax_1^i$$

for some $1 \leq 2i \leq m$ and $a \in \mathbb{C}[x_2, \ldots, x_n]$. That means

$$a^2 \mid A(x_2^m, \ldots, x_n^m) \underset{\text{induction hypothesis}}{\implies} \|a\| = 1 \implies \|u\| = 1$$

□

Note 1. *From lemma 43 it follows that $u^2 \mid f$ where $f \in S_{n,r}$ implies $\|u\| = 1$.*

Lemma 44. *Suppose $u, v \in \mathbb{C}[\overline{y}]$ such that $\|u\| = \|v\| = 1$ and they are disjoint variable monomials and suppose $h \mid u^p x^r - v^p$ for some prime $p \mid r$ where $h = \sum_{i=0}^{k} a_i x^i$ where $a_i \in \mathbb{C}[\overline{y}]$. Then $\|a_i\| = 1$ for each $0 \leq i \leq k$.*

Proof. Suppose $\frac{r}{p} = m$. Consider the following field extension $C' = \mathbb{C}(\overline{y})\left(\left(\frac{v}{u}\right)^{\frac{1}{m}}\right)$. Now all roots of

$$u^p x^r - v^p = 0$$

in C' are actually $\left(\frac{v}{u}\right)^{\frac{1}{m}} \zeta_r^i$ for all $0 \leq i \leq r-1$. If $h \mid u^p x^r - v^p$, then roots of h must be

$$\alpha_j = \left(\frac{v}{u}\right)^{\frac{1}{m}} \zeta_r^{i_j}$$

where $1 \leq j \leq k$ and i_j's are distinct numbers in the range $[0, r-1]$. That implies

$$\frac{a_l}{a_0} = (-1)^l \sum_{1 \leq i_1 < \ldots < i_l \leq k} \alpha_{i_1} \ldots \alpha_{i_l} = \left(\frac{cv}{u}\right)^{\frac{l}{m}}$$

for any $0 \leq l \leq k$ for some $c \in \mathbb{C}$. That means

$$a_l = p_l (cv)^{\frac{l}{m}} \text{ and } a_0 = p_l u^{\frac{l}{m}}$$

where $p_l \in \mathbb{C}[\overline{y}]$ given the fact that $u^{\frac{l}{m}}, v^{\frac{l}{m}}$ are actually polynomials. As $a_0 \mid v^p \implies \|a_0\| = 1$. Hence $\|p_l\| = 1 \implies \|a_l\| = 1$ forall $0 \leq l \leq k$.

□

Now we are ready to prove the following theorem 6.

Proof of theorem 6. For $r \leq 2$, it follows directly from [Vol15] . So assume $r \geq 3$. We will induct on the number of variables.

Base case ($n = 1$) : We have

$$f = a x_1^r + b$$

where $a, b \in \mathbb{C}$. If $b = 0$ then , $\|f\| = 1$ as well as $\|g\|$. If $b \neq 0$, then $g \mid f$ would have sparsity $\leq r$ as $\deg(g) \leq r-1$. Hence

$$\|g\| \leq r = 2^{\log r} = \|f\|^{\log r}$$

Induction Hypothesis : Suppose this is true for $\leq n-1$ variable polynomials i.e whenever $f \in S_{\leq n-1, r}$.

Induction Step : Suppose

$$f = s_r x_1^r - s_0$$

where $s_r, s_0 \in S_{\leq n-1, r}$. If $s_0 = 0$ then $f = s_r x_1^r$ and $g \mid f$ implies

$$g = a x_1^i$$

where $a \mid s_r$ and $0 \leq 2i \leq r$. So, from induction hypothesis we have

$$\|g\| = \|a\| \leq \|s_r\|^{\log r} = \|f\|^{\log r}$$

Hence we may assume that $s_0 \neq 0$. Suppose gcd $(s_r, s_0) = G$ (this means upto constant) . Also assume that

$$s_r = GA, s_0 = GB \text{ and } \gcd(A, B) = 1$$

Then

$$f = G(Ax_1^r - B)$$

If $(Ax_1^r - B)$ is irreducible , then

$$g \mid f \implies g = g_1 \text{ or } g = g_1(Ax_1^r - B)$$

where $g_1 \mid G$. Now $g_1 \mid G \mid s_r$, so from induction hypothesis , we have

$$\|g_1\| \leq \|s_r\|^{\log r} \leq (\|s_r\| + \|s_0\|)^{\log r} = \|f\|^{\log r}$$

So If $g = g_1$, we are done. If $g = g_1(Ax_1^r - B)$, then

$$g_1 A \mid GA = s_r \text{ and } g_1 B \mid GB = s_0$$

Hence again by induction hypothesis ,

$$\|g\| = \|g_1 A\| + \|g_1 B\| \leq \|s_r\|^{\log r} + \|s_0\|^{\log r} \leq (\|s_r\| + \|s_0\|)^{\log r} = \|f\|^{\log r}$$

Hence assume $Ax_1^r - B$ is reducible , then from lemma 42 , we have existence of u and v such that

$$A = u^p \text{ and } B = v^p$$

for some $p \mid r$. Hence we have

$$f = G(u^p x_1^r - v^p)$$

As $s_r, s_0 \in S_{\leq n-1,r}$, from note 1 , we have $\|u\| = \|v\| = 1$. Now as A, B has no non-trivial gcd , it implies that u and v are disjoint variable monomials. Now, suppose $g \mid f$. Then , $g = g_1 g_2$ where

$$g_1 \mid G \text{ and } g_2 \mid u^p x_1^r - v^p$$

From lemma 44,

$$g_2 = \sum_{i=0}^{k} c_i x_1^i$$

where $k \leq r-1$ and $\|c_i\| = 1$ .So, $\|g_1 c_i\| = 1$ forall $0 \leq i \leq k$. Hence

$$g = \sum_{i=0}^{k} g_1 c_i x_1^i \implies \|g\| = \sum_{i=0}^{k} \|g_1 c_i\| = \sum_{i=0}^{k} \|g_1\| = (k+1)\|g_1\|$$

Now as $g_1 \mid G \mid s_r, s_0$ and both $s_r, s_0 \neq 0$, from induction hypothesis we have

$$\|g_1\| \leq \min(\|s_r\|^{\log r}, \|s_0\|^{\log r})$$

Hence

$$\begin{aligned}
\|g\| = (k+1)\|g_1\| &\leq r\|g_1\| \\
&\leq r\left(\min(\|s_r\|, \|s_0\|)\right)^{\log r} \\
&= \left(2 \min\left(\|s_r\|, \|s_0\|\right)\right)^{\log r} \\
&\leq \left(\|s_r\| + \|s_0\|\right)^{\log r} \\
&= \|f\|^{\log r}
\end{aligned}$$

□

▷ Now we focus on degree ≤ 3 case. We can always show that quadratic irreducible polynomials are of our interest as when we induct , we can easily get sparsity bound for linear terms and so , if there is one degree 3 or 1 variable , we are done ! So , a factor where all the degree $= 2$ and is irreducible is of our prime interest and if one can show some significant bound on them, we would be done with the constant individual degree ≤ 3 case.

Chapter 10

Conclusion and Open Problems

The old *Factors conjecture* states that for a nonzero polynomial f: $g \mid f \implies \text{size}(g) \leq \text{poly}(\text{size}(f), \deg(g))$ (see Conjecture 2). Motivated by Theorem 2, we would like to strengthen it to:

Conjecture 1 (radical)**.** *For a nonzero f: $min\{deg(rad(f)), size(rad(f))\} \leq \text{poly}(size(f))$.*

Is the Radical conjecture true if we replace size by $\overline{\text{size}}$?

In low degree regime also there are many open questions. Can we identify a class "below" VPthat is closed under factoring? We conclude with some interesting questions.

1. Are VF, VBP closed under factoring? We might consider Theorem 4 as a positive evidence. Additionally, note that these classes are already closed under e-th root taking. This is easy to see using the classic Taylor series of $(1+f)^{1/e}$, where $f \in \langle \overline{x} \rangle$.

   In fact, what about the classes which are contained in $VF(n^{\log n})$ but larger than VF. For example, is $\text{VF}(n^{\log\log n})$ closed under factoring?

2. Can we find a suitable analog of Strassen's (non-unit) division elimination for high degree circuits? This, by Theorem 3, will resolve Factors conjecture. In fact it is left as an open question whether division elimination can be done in $\text{poly}(s, \log d)$ size circuit (for reference see https://www.cs.tau.ac.il/~shpilka/wact2016/concreteOpenProblems/openprobs.pdf). For example, one can show that $\frac{x^n-1}{x-1} = \sum_{i=0}^{n-1} x^i$ has circuit of size $O(\log^2 n)$ and thus it would be interesting to come up with an counter example as well!

3. Our results weaken when $\mathbb{F}$ is not algebraically closed or has a small prime characteristic (Sections 8.1, 8.2). Can we strengthen the methods to work for all $\mathbb{F}$?

4. Is sparsity Conjecture 4 true? In particular can one improve the sparsity bound proved in [BSV18]?

Bibliography

[Abe73] Oliver Aberth. Iteration methods for finding all zeros of a polynomial simultaneously. *Mathematics of computation*, 27(122):339–344, 1973. 38

[Agr05] Manindra Agrawal. Proving lower bounds via pseudo-random generators. In *International Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 92–105. Springer, 2005. 7

[AGS18] Manindra Agrawal, Sumanta Ghosh, and Nitin Saxena. Bootstrapping variables in algebraic circuits. In *50th Annual ACM Symposium on the Theory of Computing, STOC'18, June 25-29, 2018, Los Angeles, CA*, 2018. 6, 8

[AV08] Manindra Agrawal and V Vinay. Arithmetic circuits: A chasm at depth four. In *Foundations of Computer Science, 2008. FOCS'08. IEEE 49th Annual IEEE Symposium on*, pages 67–75. IEEE, 2008. 6

[BCS13] Peter Bürgisser, Michael Clausen, and Amin Shokrollahi. *Algebraic complexity theory*, volume 315. Springer Science & Business Media, 2013. 23, 32, 33, 34, 61

[Ber70] Elwyn R Berlekamp. Factoring polynomials over large finite fields. *Mathematics of computation*, 24(111):713–735, 1970. 1

[BIZ17] Karl Bringmann, Christian Ikenmeyer, and Jeroen Zuiddam. On algebraic branching programs of small width. In *32nd Computational Complexity Conference, CCC 2017, July 6-9, 2017, Riga, Latvia*, pages 20:1–20:31, 2017. 61

[BOC92] Michael Ben-Or and Richard Cleve. Computing algebraic formulas using a constant number of registers. *SIAM Journal on Computing*, 21(1):54–58, 1992. 5

[BOT88] Michael Ben-Or and Prasoon Tiwari. A deterministic algorithm for sparse multivariate polynomial interpolation. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 301–309. ACM, 1988. 6

[BSR+05] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96. ACM, 2005. 29

[BSS89] Lenore Blum, Mike Shub, and Steve Smale. On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. *Bulletin (New Series) of the American Mathematical Society*, 21(1):1–46, 1989. 12

[BSV18] Vishwas Bhargava, Shubhangi Saraf, and Ilya Volkovich. Deterministic factorization of sparse polynomials with bounded individual degree. Private Communication, 2018. 9, 13, 69, 77

[Bür01] Peter Bürgisser. The complexity of factors of multivariate polynomials. In *In Proc. 42th IEEE Symp. on Foundations of Comp. Science*, 2001. 61, 62

[Bür04] Peter Bürgisser. The complexity of factors of multivariate polynomials. *Foundations of Computational Mathematics*, 4(4):369–396, 2004. (Preliminary version in FOCS 2001). 45, 56, 62, 63

[Bür13] Peter Bürgisser. *Completeness and reduction in algebraic complexity theory*, volume 7. Springer Science & Business Media, 2013. 6, 7, 23

[CKS18] Chi-Ning Chou, Mrinal Kumar, and Noam Solomon. Some closure results for polynomial factorization and applications. 2018. 7, 54

[CRS96] Richard Courant, Herbert Robbins, and Ian Stewart. *What is Mathematics?: an elementary approach to ideas and methods*. Oxford University Press, USA, 1996. 32

[CZ81] David G Cantor and Hans Zassenhaus. A new algorithm for factoring polynomials over finite fields. *Mathematics of Computation*, pages 587–592, 1981. 1

[DB08] Germund Dahlquist and Åke Björck. Numerical methods in scientific computing, volume I. *Society for Industrial and Applied Mathematics*, 2008. 38

[DdO14] Zeev Dvir and Rafael Mendes de Oliveira. Factors of sparse polynomials are sparse. *arXiv preprint arXiv:1404.4834*, 2014. 9

[DMM+14] Arnaud Durand, Meena Mahajan, Guillaume Malod, Nicolas de Rugy-Altherre, and Nitin Saurabh. Homomorphism polynomials complete for VP. In *34th International Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS*, pages 493–504, 2014. 12

[DSY09] Zeev Dvir, Amir Shpilka, and Amir Yehudayoff. Hardness-randomness tradeoffs for bounded depth arithmetic circuits. *SIAM Journal on Computing*, 39(4):1279–1293, 2009. (Preliminary version in STOC'08). 6, 7, 39

[Dur60] Émile Durand. Solutions numériques des équations algébriques. Tome I, Équations du type F(x)= 0, racines d'un polynôme. 1960. 38

[Ehr67] Louis W Ehrlich. A modified Newton method for polynomials. *Communications of the ACM*, 10(2):107–108, 1967. 38

[FS15] Michael A. Forbes and Amir Shpilka. Complexity theory column 88: Challenges in polynomial factorization1. *SIGACT News*, 46(4):32–49, 2015. 8

[FSTW16] Michael A Forbes, Amir Shpilka, Iddo Tzameret, and Avi Wigderson. Proof complexity lower bounds from algebraic circuit complexity. In *Proceedings of the 31st Conference on Computational Complexity*, page 32. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016. 7

[GMQ16] Joshua A. Grochow, Ketan D. Mulmuley, and Youming Qiao. Boundaries of VP and VNP. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*, volume 55, pages 34:1–34:14, 2016. 61

[GMS+86] Philip E Gill, Walter Murray, Michael A Saunders, John A Tomlin, and Margaret H Wright. On projected Newton barrier methods for linear programming and an equivalence to Karmarkar's projective method. *Mathematical programming*, 36(2):183–209, 1986. 29

[GS98] Venkatesan Guruswami and Madhu Sudan. Improved decoding of reed-solomon and algebraic-geometric codes. In *Foundations of Computer Science, 1998. Proceedings. 39th Annual Symposium on*, pages 28–37. IEEE, 1998. 6

[GTZ88] Patrizia Gianni, Barry Trager, and Gail Zacharias. Gröbner bases and primary decomposition of polynomial ideals. *Journal of Symbolic Computation*, 6(2):149–167, 1988. 6

[IKRS12] Gábor Ivanyos, Marek Karpinski, Lajos Rónyai, and Nitin Saxena. Trading grh for algebra: algorithms for factoring polynomials and related structures. *Mathematics of Computation*, 81(277):493–531, 2012. 6

[Jan11] Maurice J Jansen. Extracting roots of arithmetic circuits by adapting numerical methods. In *2nd Symposium on Innovations in Computer Science (ICS 2011)*, pages 87–100, 2011. 7

[Kal85] Erich Kaltofen. Computing with polynomials given by straight-line programs I: greatest common divisors. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, pages 131–142, 1985. 6

[Kal86] Erich Kaltofen. Uniform closure properties of p-computable functions. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 330–337, 1986. 6, 47, 56

[Kal87] Erich Kaltofen. Single-factor hensel lifting and its application to the straight-line complexity of certain polynomials. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 443–452. ACM, 1987. 6, 10, 11, 12, 19, 45, 47, 55, 56

[Kal89] Erich Kaltofen. Factorization of polynomials given by straight-line programs. *Randomness and Computation*, 5:375–412, 1989. 6, 7, 10, 12, 53

[Kal90] Erich Kaltofen. Polynomial factorization 1982-1986. *Dept. of Comp. Sci. Report*, pages 86–19, 1990. 53

[Kar89] Gregory Karpilovsky. *Topics in field theory*, volume 155. Elsevier, 1989. 70

[Kay11] Neeraj Kayal. Efficient algorithms for some special cases of the polynomial equivalence problem. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 1409–1421. Society for Industrial and Applied Mathematics, 2011. 6

[Ker66] Immo O Kerner. Ein Gesamtschrittverfahren zur Berechnung der Nullstellen von Polynomen. *Numerische Mathematik*, 8(3):290–294, 1966. 38

[KI03] Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 355–364. ACM, 2003. 6, 7, 8

[KK08] Erich Kaltofen and Pascal Koiran. Expressing a fraction of two determinants as a determinant. In *Proceedings of the twenty-first international symposium on Symbolic and algebraic computation*, pages 141–146. ACM, 2008. 7

[KP12] Steven G Krantz and Harold R Parks. *The implicit function theorem: history, theory, and applications*. Springer Science & Business Media, 2012. 39

[KS01] Adam R Klivans and Daniel Spielman. Randomness efficient identity testing of multivariate polynomials. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 216–223. ACM, 2001. 6, 8, 9

[KS06] Neeraj Kayal and Nitin Saxena. Complexity of ring morphism problems. *computational complexity*, 15(4):342–390, 2006. 6

[KS09] Zohar S Karnin and Amir Shpilka. Reconstruction of generalized depth-3 arithmetic circuits with bounded top fan-in. In *Computational Complexity, 2009. CCC'09. 24th Annual IEEE Conference on*, pages 274–285. IEEE, 2009. 6

[KS16] Mrinal Kumar and Shubhangi Saraf. Arithmetic circuits with locally low algebraic rank. In *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, pages 34:1–34:27, 2016. 39

[KSS15] Swastik Kopparty, Shubhangi Saraf, and Amir Shpilka. Equivalence of polynomial identity testing and polynomial factorization. *computational complexity*, 24(2):295–331, 2015. 8, 12, 16, 53, 54, 66

[Lec02] Grégoire Lecerf. Quadratic newton iteration for systems with multiplicity. *Foundations of Computational Mathematics*, 2(3):247–293, 2002. 38

[LLL82] Arjen Klaas Lenstra, Hendrik Willem Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982. 1

[LLMP90] Arjen K Lenstra, Hendrik W Lenstra, Mark S Manasse, and John M Pollard. The number field sieve. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 564–572. ACM, 1990. 6

[LN97] Rudolph Lidl and Harald Niederreiter. *Finite Fields*. Cambridge University Press, Cambridge, UK, 1997. 21

[LS78] Richard J Lipton and Larry J Stockmeyer. Evaluation of polynomials with super-preconditioning. *Journal of Computer and System Sciences*, 16(2):124–139, 1978. 6

[LV16] Anand Louis and Santosh Srinivas Vempala. Accelerated newton iteration for roots of black box polynomials. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 732–740, 2016. 40

[Mul17] Ketan Mulmuley. Geometric complexity theory V: Efficient algorithms for Noether normalization. *Journal of the American Mathematical Society*, 30(1):225–309, 2017. 6

[MV97] Meena Mahajan and V Vinay. A combinatorial algorithm for the determinant. In *SODA*, pages 730–738, 1997. 17

[MY73] Joel Moses and David Y. Y. Yun. The ez gcd algorithm. In *Proceedings of the ACM Annual Conference*, ACM '73, pages 159–166, New York, NY, USA, 1973. ACM. 45

[New69] Isaac Newton. De analysi per aequationes numero terminorum infinitas [on analysis by infinite series] (in latin). 1669. (published in 1711 by William Jones). 38

[Oli16] Rafael Oliveira. Factors of low individual degree polynomials. *Computational Complexity*, 2(25):507–561, 2016. (Preliminary version in CCC'15). 7, 39, 54, 63

[OR00] James M Ortega and Werner C Rheinboldt. *Iterative solution of nonlinear equations in several variables*. SIAM, 2000. 29

[Pau01] Sebastian Pauli. Factoring polynomials over local fields. *Journal of Symbolic Computation*, 32(5):533–547, 2001. 66

[Pla77a] David Alan Plaisted. New NP-hard and NP-complete polynomial and integer divisibility problems. In *Foundations of Computer Science, 18th Annual Symposium on*, pages 241–253. IEEE, 1977. 11

[Pla77b] David Alan Plaisted. Sparse complex polynomials and polynomial reducibility. *Journal of Computer and System Sciences*, 14(2):210–221, 1977. 1, 7, 11

[Pla84] David A Plaisted. New np-hard and np-complete polynomial and integer divisibility problems. *Theoretical Computer Science*, 31(1-2):125–138, 1984. 43

[PSS16] Anurag Pandey, Nitin Saxena, and Amit Sinhababu. Algebraic independence over positive characteristic: New criterion and applications to locally low algebraic rank circuits. In *41st International Symposium on Mathematical Foundations of Computer Science, MFCS 2016, August 22-26, 2016 - Kraków, Poland*, pages 74:1–74:15, 2016. 39

[Sap16] Ramprasad Saptharishi. A survey of lower bounds in arithmetic circuit complexity. *URL https://github. com/dasarpmar/lowerbounds-survey/releases. Version*, 3(0), 2016. 18, 19, 50

[Sch77] Claus-Peter Schnorr. Improved lower bounds on the number of multiplications/divisions which are necessary to evaluate polynomials. In *International Symposium on Mathematical Foundations of Computer Science*, pages 135–147. Springer, 1977. 6

[Sch80] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, October 1980. 17, 18, 22, 32

[Sin16] Gaurav Sinha. Reconstruction of real depth-3 circuits with top fan-in 2. In *31st Conference on Computational Complexity*, 2016. 6

[SSS13] Chandan Saha, Ramprasad Saptharishi, and Nitin Saxena. A case of depth-3 identity testing, sparse factorization and duality. *Computational Complexity*, 22(1):39–69, 2013. 8

[Str73] Volker Strassen. Vermeidung von divisionen. *Journal für die reine und angewandte Mathematik*, 264:184–202, 1973. 11, 17, 59

[Sud97] Madhu Sudan. Decoding of reed solomon codes beyond the error-correction bound. *Journal of complexity*, 13(1):180–193, 1997. 6

[SY10] Amir Shpilka and Amir Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends® in Theoretical Computer Science*, 5(3–4):207–388, 2010. 17, 18

[Tay15] Brook Taylor. Methodus incrementorum directa et inversa [direct and reverse methods of incrementation] (in latin). 1715. (Translated into English in Struik, D. J. (1969). A Source Book in Mathematics 1200–1800. Cambridge, Massachusetts: Harvard University Press. pp. 329–332.). 10

[Val79] L. G. Valiant. Completeness classes in algebra. In *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing*, STOC '79, pages 249–261, New York, NY, USA, 1979. ACM. 4

[Val82] L Valiant. Reducibility by algebraic projections in: Logic and algorithmic. In *Symposium in honour of Ernst Specker*, pages 365–380, 1982. 10, 20

[Vol15] Ilya Volkovich. Computations beyond exponentiation gates and applications. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 22, page 42, 2015. 9, 73

[VSBR83] Leslie G. Valiant, Sven Skyum, Stuart Berkowitz, and Charles Rackoff. Fast parallel computation of polynomials using few processors. *SIAM Journal on Computing*, 12(4):641–644, 1983. 5, 12

[vzGG13] Joachim von zur Gathen and Jürgen Gerhard. *Modern computer algebra*. Cambridge university press, 2013. 21, 39, 65, 66

[vzGK85] Joachim von zur Gathen and Erich Kaltofen. Factoring sparse multivariate polynomialsa. *Journal of Computer and System Sciences*, 31(2):265–287, 1985. 7, 8, 9

[Wan80] Paul S. Wang. The eez-gcd algorithm. *SIGSAM Bull.*, 14(2):50–60, May 1980. 46

[Zas69] Hans Zassenhaus. On Hensel factorization, I. *Journal of Number Theory*, 1(3):291–311, 1969. 39

[ZS75] Oscar Zariski and Pierre Samuel. *Commutative algebra. II. Reprint of the 1960 edition*, volume 29. Graduate Texts in Mathematics, 1975. 16