An effective description of the roots of multivariates mod p^k and the related Igusa's local zeta function

Sayak Chakrabarti [*†] Nitin Saxena [‡]

Abstract

Finding roots of a multivariate polynomial $f(\mathbf{x})$, over a prime field $\mathbb{F}_p$, is a fundamental question with a long history and several practical algorithms are now known. Effective algorithms for describing the roots modulo p^k, $k \geq 2$, for any general multivariate polynomial, were unknown until the present paper. The main obstruction is lifting the *singular* $\mathbb{F}_p$ roots to $\mathbb{Z}/p^k\mathbb{Z}$. Such roots may be numerous and behave unpredictably, i.e., they may or may not lift from $\mathbb{Z}/p^j\mathbb{Z}$ to $\mathbb{Z}/p^{j+1}\mathbb{Z}$.

We give the first algorithm to describe the roots of a bivariate polynomial over $\mathbb{Z}/p^k\mathbb{Z}$ in a practical way. Notably, when the degree of the polynomial is constant, our algorithm runs in deterministic time which is polynomial in $p + k$. This is a significant improvement over brute force, which would require exploring p^{2k} possible values. Our method also gives the first efficient algorithms for the following problems (which were also open): (1) efficiently representing all the (possibly infinitely-many) p-adic roots, and (2) computing the underlying Igusa's local zeta function. We also obtain a new, effective method to prove the rationality of this zeta function.

2012 ACM CCS concept: Theory of computation– Algebraic complexity theory; Theory of computation– Problems, reductions and completeness; Computing methodologies– Algebraic algorithms; Computing methodologies– Hybrid symbolic-numeric methods.
Keywords: polynomial, prime-power, bivariate, p-adic, root-finding, root-counting, deterministic, Igusa, zeta function, tree

Contents

[*]Computer Science, Columbia University, New York, USA Email: sayaksc@gmail.com
[†]Part of this work appears in the author's Master thesis at IIT Kanpur (e-link at [Cha22]).
[‡]Department of CSE, Indian Institute of Technology, Kanpur, India, Email: nitin@cse.iitk.ac.in

1 Introduction

Roots of polynomials in fields and rings have played an important role in mathematics and computer science for decades, with applications in a variety of topics. The roots of univariate polynomials modulo prime powers are relatively easy to find as we can find the roots in finite fields using factorization [CZ81, Ber67, Ber70, KU11, Rón87], after which we can efficiently lift the roots to modulo higher powers of p using the recent developments from [Pan95, BLQ13, DMS21]. However, even though we can factorize *multi*variate polynomials [Kal82, Kal85], their roots usually do not correspond to a factor. Eg. even the irreducible polynomial $y - x$ has numerous roots! Such polynomials pose problems in root-finding as they have exponentially many roots in the base/prime field itself, and we can not quickly guess which root will lift to mod p^2. In our paper, we resolve this issue by giving an efficient algorithm to find roots modulo p^k for any prime p and k (given in unary); assuming that the degree d of the polynomial and the number n of the variables, are both small.

Prime field $\mathbb{F}_p$ and *p-adic integers* $\mathbb{Z}_p$ are unique factorization domains, and polynomials (above them) behave in an expected way. There are some algorithms to factorize polynomials in $\mathbb{Z}_p$ [Chi87, CG00, GNP12]. However, the properties in rings of characteristic as prime powers, which can be seen as a world between $\mathbb{F}_p$ and $\mathbb{Z}_p$, are still a mystery to us. There has been extensive work in this since the famous Hensel's lifting [Hen18, Zas69, Zas78], where factors of polynomials are lifted from $\mathbb{F}_p$ to $\mathbb{Z}/p^k\mathbb{Z}$. Several variants of Hensel's lifting are available in various topics in algebra & number theory; but they fail when the polynomial does not factorize into coprime factors. This, when interpreted in terms of roots, means that it is difficult to lift *singular* roots. (Eg., $f = x_1^3 - px_2^2$, or $f = x_1^3 - p$, modulo p^2. Here, it is unclear how to even test the *existence* of roots; as the only $\mathbb{F}_p$-root here, $x_1 \equiv 0 \bmod p$, starts behaving unpredictably when 'lifted' modp^2.) There have been partially successful attempts to tackle this, and achieve factorization of *uni*variate polynomials mod p^k [DMS21, Sir17, Săl05, CL01]. Factorization mod p^k has only been solved until $k \leq 4$. Due to this, we can not use factorization, in any way, when finding roots mod p^k. Furthermore, due to the availability of 'exponentially' many factors, as well as roots, in these rings, its (data) structure has been of interest in mathematics and computer science. Eg. [DM97, Mau01, GCM21] analyze root-sets mod p^k, while [DMS19, CGRW19, KRRZ20, DS20, RRZ21] count the number of roots for a given polynomial. However, most of the works are restricted to univariate polynomials, as we did not have practical algorithms to find even *one* root of *bi*variate polynomials mod p^k. *This paper gives the first algorithm to find all the roots of an n-variate degree-d polynomial, over $\mathbb{Z}/p^k\mathbb{Z}$ resp. $\mathbb{Z}_p$, efficiently (for small n, d and varying p, k).*

Let us take a famous example as our algorithm's special-case— a *hyperelliptic curve*, given by the equation $y^2 + u(x)y + v(x) = 0$. Rational roots of (hyper)elliptic curves have been widely studied with several papers in this area [Sch95, MVZ93, LMMS94, GH00, Ked01, Sat02, MCT02]. Our algorithm can find all $(\mathbb{Z}/p^k\mathbb{Z})$-roots (resp. p-adic) of not only these, but general curves (that may have singularities). Even a *single* singular point forces us to explore its p^{k-1} possible lifts.

Root finding and counting have interesting applications in complexity theory too. We know

that finding a solution to a system of constant-degree polynomials in any ring is NP-complete. Our algorithm solves a special case: where we want a common solution of *low*-variate, *low*-degree polynomials. Furthermore, root counting is a very hard problem. [EK90] showed that counting the number of roots of a multivariate polynomial of degrees as small as 3 is #P-complete, while [GGL08] showed that modular root counting, over $\mathbb{F}_q$, is NP-hard for prime modulus other than the characteristic of the field. [ZG03] also showed that the problem of computing Igusa's local zeta function is NP-complete even for $p = d = 2$ which can be shown from the arithmetization of SAT.

There has been extensive work on computing related zeta functions. [Ked04, Lau06, Har15, CRW20] compute the zeta function which encodes the size of a variety in finite fields, in time polynomial in the characteristic p. Improving this, say to $\mathsf{poly}(\log p)$, is a central open question. Our paper focuses on the regime of $\mathsf{poly}(p)$-time too, as the case of prime-power seems harder than finite fields (for instance, it has *no* known formulation of the famous Riemann Hypothesis).

In this paper, we are interested in another zeta function called the *Igusa's local zeta function* (Igusa's LZF), used to encode the number of roots of a polynomial modulo prime powers [Igu74, Igu77]. Formally, for a polynomial $f(\mathbf{x}) \in \mathbb{Z}_p[\mathbf{x}]$, the Igusa's local zeta function is defined as

$$Z_{f,p}(s) := \int_{\mathbb{Z}_p^n} |f(\mathbf{x})|_p^s \cdot |d\mathbf{x}|, \tag{1}$$

for $s \in \mathbf{C}$, where the real part of s is positive. Igusa first proved that his LZF converges to a rational function. [Den84] gave another proof for the rationality of this function. We give an alternate, more constructive proof by computing the Poincaré series; which can be seen as a generating function for point-counting. The Poincaré series for a polynomial f and a prime p is defined as $P_{f,p}(t) := \sum_{i=0}^{\infty} N_i(f) \cdot (p^{-n}t)^i$, where $t \in \mathbb{C}$, $|t| < 1$ and $N_i(f)$ refers to the number of roots of $f \bmod p^i$. Choosing $t = p^{-s}$, we denote $P_{f,p}(p^{-s}) = \sum_{i=0}^{\infty} N_i(f) \cdot (p^{-n}p^{-s})^i$. [Igu07] showed a connection between Poincaré series and Igusa's LZF given as:

$$P_{f,p}(p^{-s}) = \frac{1 - p^{-s} \cdot Z_{f,p}(s)}{1 - p^{-s}}.$$

Notice that proving rationality of Poincaré series $P(p^{-s})$ in p^s implies the rationality of Igusa's LZF in p^s and vice versa. We use this relation to show that Poincaré series is indeed a rational function by counting roots modulo p^k.

Despite the rationality being proved, explicit computation of this zeta function has remained a challenge. Root counting helps in computing this using the Poincaré series [DS20, ZG03, DH01], but has been restricted to univariates. Giving the first algorithm, *we compute Igusa's local-zeta function for bivariates; thus, giving a new proof of its rationality as well.*

Based on this, we provide a simple extension of the algorithms to compute and count the roots n-variates as well, thereby proving the rationality of Igusa's local-zeta function for multivariates.

1.1 Previous work

There have been several works on finding roots of polynomials in rings. [Pan95] gave an approach for finding roots of univariate polynomials modulo prime-powers in randomized polynomial time, which was greatly improved by [BLQ13, NRS17]. We are aware of only one work studying roots of *bi*variate polynomials $\bmod p^k$ [RRZ21], where they count the total number of roots if the given polynomial $f(x, y)$ can be written as $f_1(x) + f_2(y)$, i.e., variable 'separated'. On the other hand, our algorithm does not require such assumptions.

Roots modulo p^k can be seen as an intermediate world between roots in $\mathbb{F}_p$ and roots in $\mathbb{Z}_p$. There have been papers to find roots of system of polynomials in certain finite fields [HW99, LPT+17, BKW19, Kay05]. However, for finding $\mathbb{Z}_p$ roots, certain upper bounds given by N have been shown in [BM67, Chi21] which state that the existence of a solution mod p^N implies a $\mathbb{Z}_p$-root. Among the improved results, [Chi21] showed $N \leq (nd)^{O(n^3 2^n)}$. [DS20] showed $N \leq d(\Delta+1)$, where Δ is the discriminant-valuation; but it is only for univariate polynomials. We prove stronger structural results for the p-adic roots of bivariate polynomials, as discussed in Section 5. Building on the work of [DS20], we give a better bound and a much more efficient algorithm to find all the roots of an n-variate in $\mathbb{Z}_p$ as well.

Clearly, the literature suggests that roots behave "nicely" in $\mathbb{F}_p$ and $\mathbb{Z}_p$; but the properties in $\mathbb{Z}/p^k\mathbb{Z}$ are quite different for 'small' $k \geq 2$. [Dwi23, Zhu20] have extensively explored the behavior of polynomials in these intermediate rings. In our paper, we essentially generalize the approach of [BLQ13] (see Algorithm 3) to non-trivially *reduce* root finding modulo p^k to the problem of solving a univariate multi-polynomial system.

Computation of Igusa's local zeta function, and its rationality via the Poincaré series, have also been an important question in computational number theory. [Igu74, Igu77] proved the rationality of the Poincaré series, while [Den84] proved the same for a system of polynomials. [DS20] gave the first algorithm to compute Igusa's local zeta function of univariates in (deterministic) poly-time. In this paper, by describing the p-adic roots of multivariates, we give the first algorithm to compute Igusa's local zeta function practically, as well as re-prove the rationality of the Poincaré series.

1.2 Our results: Find roots in $\mathbb{Z}/p^k\mathbb{Z}$, $\mathbb{Z}_p$, and compute the Poincaré series

Our results give a new constructive understanding of the roots modulo p-power of multivariate systems. We use a new data-structure in the form of a *tree*, to view these roots with increasing exponent of the modulus. This tree data-structure, essentially, performs *desingularization* of roots—segregating them until they are non-singular —and lift to the required exponent of the prime.

Furthermore, we devise a new data-structure called *representative roots* to represent the exponentially many (resp. infinitely many p-adic) roots compactly. Our main ideas come from $n = 2$.

Theorem 1 (Bivariates)**.** *Given $f \in (\mathbb{Z}/p^k\mathbb{Z})[x_1, x_2]$ of degree d, we can decide if a root of $f(x_1, x_2)$ exists, in deterministic* $\mathsf{poly}((k+d+p)^d)$ *time. If roots do exist, we can find and count all the roots (outputting them in a* compact *data structure).*

Based on Theorem 1, we give the following two corollaries, both of which were open problems. In a way, we bridge the gap between rings of the form $\mathbb{Z}/p^k\mathbb{Z}$ and $\mathbb{Z}_p$ by giving better bounds than existing works.

Corollary 2 (p-adic)**.** *Given $f \in \mathbb{Z}[x_1, x_2]$ of degree d and the absolute value of its coefficients bounded above by $M > 0$, we can find all the p-adic-roots of f (in $\mathbb{Z}_p$) in deterministic* $\mathsf{poly}((\log M + d + p)^d)$ *time (i.e., output their k digits in a compact data structure).*

Corollary 3 (Local-zeta fn.)**.** *Given $f \in \mathbb{Z}[x_1, x_2]$ of degree d and the absolute value of its coefficients bounded above by $M > 0$, we can compute the Poincaré series $P(t) =: A(t)/B(t)$ associated with f and a prime p, in deterministic* $\mathsf{poly}((\log M + d + p)^d)$ *time.*

The algorithm for computing p-adic roots, and the proof of Corollary 2 is described in Section 5. Based on this proof and simple power series computations similar to that of [DS20], we can compute the Igusa's zeta function for bivariates. This has been briefly described in Section 6.

There are major dissimilarities between roots of univariate and bivariate polynomials. However, n-variate polynomials, for 'small' $n \geq 2$, behave in a similar manner in our proof. Thus, after describing root-finding of bivariates (Theorem 1), we sketch its generalization to n-variates in Section 7. Essentially, we reduce bivariate root finding to that of a system of univariates; whereas, for n-variates ($n \geq 3$), we reduce the problem to $(n-1)$-variate root-finding. Thus, the generalization follows in an inductive fashion.

Theorem 4 (n-variates)**.** *Given a system of polynomials* $\{f_i(x_1, \ldots, x_n) \in \mathbb{Z}[\mathbf{x}] \mid \forall i \in [m]\}$ *of degrees* $\leq d$ *and the absolute value of its coefficients bounded above by* $M > 0$*. We can find all its* common $(\mathbb{Z}/p^k\mathbb{Z})$*-roots, resp. its* p*-adic-roots, resp. its Poincaré series, in deterministic* $\mathsf{poly}((m \log M + d + p)^{(2d(n-1))^{n-1}})$ *time.*

1.3 Difficulty of the problem

It is easy to lift a *non-singular* $\mathbb{F}_p$-root, i.e., root of f mod p at which some first-order derivative of f is nonzero, to any $\mathbb{Z}/p^k\mathbb{Z}$ (see Theorem 5 and Corollary 6). In contrast, this paper is a significant advancement in the case when $\mathbb{F}_p$-roots are *singular*. It can be viewed as a reduction to non-singular roots, of a 'higher' dimension variety: which gets created while using the process of *lifting* (eg. extend a root (x_1, x_2) of f mod p^j to mod p^{j+1} by perturbation). In our method, the dependence on p cannot be improved, as bivariates can have $\Omega(p)$ singular roots at each step of lifting.

Practicalities. Though our algorithm is slow for large degree *bi*variates or large primes, it is the first idea that works, for *small* degree d and prime p, much better than the brute-force algorithm. Our general algorithm is doubly-exponential in n (= number of variables), but, unsurprisingly the complexity is expected to be 'bad' in n as the problem of counting $\mathbb{F}_2$-roots (say, for a system with $d = 2$) is already NP-hard and even #P-hard [EK90, ZG03, GGL08].

1.4 Proof overview: Algorithms 1 & 2

Let us see the high-level idea in our main theorem, Theorem 1.

If (a_1, a_2) is a root of $f(x_1, x_2)$ mod p, then we transform the polynomial to another polynomial given by $f(a_1 + px_1, a_2 + px_2)$. In order to find $\mathbb{F}_p$-roots of this polynomial in the next step, we remove the 'extra' p-powers by dividing by p^v; where $v := v_p(f(a_1 + px_1, a_2 + px_2))$ is the *val-multiplicity* and $v_p(\cdot)$ is the p-valuation. We define this step, of transforming the coordinates and subsequent division by the p-power, as the *lifting step* or *lifting of roots*. The polynomial will be modified at each step such that its $\mathbb{F}_p$-root returns a coordinate of the final (p-adic resp. $\mathbb{Z}/p^k\mathbb{Z}$) root. Notice that if (a_1, a_2) is an $\mathbb{F}_p$-root of $f(x_1, x_2)$, and after lifting, the polynomial becomes $\tilde{f}(x_1, x_2) := p^{-v} f(a_1 + px_1, a_2 + px_2)$ which has an $\mathbb{F}_p$-root (b_1, b_2), then $(a_1 + pb_1, a_2 + pb_2)$ is a root of $f(x_1, x_2)$ mod p^{v+1}. The *uni*variate case of this lifting technique was developed in [BLQ13, DMS19].

However, it might happen that root (a_1, a_2) in the lifting process does *not* lift to higher powers of p; but some other root does lift, as illustrated by the following example.

Example 1. *Consider* $f(x_1, x_2) := x_1^3 - x_2^3 + 3x_2 - 3x_1 + 5$ *and* $p := 5$*.* $(1, 1)$ *and* $(2, 2)$ *are its* $\mathbb{F}_p$*-roots. Starting with the root* $(1, 1)$*, the process of lifting given by the transformation* $(x_1, x_2) \mapsto (1 + 5x_1, 1 + 5x_2)$ *and division by* 5*, yields the polynomial* $25x_1^3 - 25x_2^3 + 15x_1^2 - 15x_2^2 + 1$ *which does not have* $\mathbb{F}_5$*-roots.*

Although, restarting with the root as $(2,2)$ *yields the polynomial* $25x_1^3-25x_2^3+30x_1^2-30x_2^2+9x_1-9x_2+1$ *after lifting.* $(1,0)$ *is now its* $\mathbb{F}_5$*-root!*

Thus, we iteratively loop over all the possible roots at each step, by fixing one variable, say x_1, with p-many possibilities, and finding the possible d-many (or p-many) values of x_2.

Val-multiplicity vs valuation. For a polynomial $f(\mathbf{x})$, we define the *effective polynomial* as $(f \bmod p)$, where the coefficients are in $\mathbb{F}_p$ (we can assume $f \bmod p$ is non-constant). Similarly, the *effective degree* d_1 of $f(\mathbf{x})$ is the degree of $(f \bmod p)$, while $d \geq d_1 \geq 1$ will be the total degree of f.

We define a *local root* of $f(\mathbf{x})$ as a root of the effective polynomial. For a local root $\mathbf{a} \in \mathbb{F}_p^2$, *local valuation* is defined as $v_p(f(\mathbf{a}))$. Recall: val-multiplicity of local root is defined as $v_p(f(\mathbf{a}+p\mathbf{x}))$, i.e., the minimum valuation of the coefficients of the polynomial thus formed; we sometimes shorten it to val-mult($\mathbf{a}$). Obviously, val-multiplicity is at most the local valuation.

Idea of Algorithm 1: Branching by val-multiplicities. As we have seen in Example 1, different $\mathbb{F}_p$ roots in steps of lifting can give rise to different val-multiplicities. Thus, we will view the algorithm as finding roots along a search *tree* (Fig.1). Note that there are at most dp local roots of f in $\mathbb{F}_p^2$.

The *nodes* of the tree contain the polynomials whose roots we are interested in finding from that point on. The *branches* arising from a node correspond to each local root $\mathbf{a} \in \mathbb{F}_p^2$. The *child*ren of this node will be the polynomials obtained from lifting by the local root in accordance to the branch, given by $f_{j+1}(\mathbf{x}) := p^{-v} f_j(\mathbf{a}+p\mathbf{x})$; where $v :=$ val-mult($\mathbf{a}$).

At depth j of the tree, the node contains the polynomial $f_j(\mathbf{x})$ that has been obtained by performing lifting j times on the original polynomial $f(\mathbf{x})$ using a contiguous sequence of local roots. Suppose we are at a node with $f_j(\mathbf{x})$ over $\mathbb{Z}/p^{k_j}\mathbb{Z}$, and consider the lifting according to root $\mathbf{a}$ with val-multiplicity $v_j := v_p(f_j(\mathbf{a}+p\mathbf{x}))$. The child of this node is $f_{j+1}(\mathbf{x}) := p^{-v_j} f_j(\mathbf{a}+p\mathbf{x})$, and we are interested in its roots over the ring $\mathbb{Z}/p^{(k_j-v_j)}\mathbb{Z}$. Thus, our target prime power reduces as we traverse down in the tree.

Theorem 5-(1) shows that the effective degree of the *newly* lifted polynomial will be at most the val-multiplicity of the local root considered, which in turn is $\leq d_1$ (= old effective degree). So, if val-multiplicity is $< d_1$, then the new effective degree reduces after lifting. Our algorithm's hard case is when a local root (a_1, a_2) has val-multiplicity d_1 (i.e., maximum possible). Then, the effective polynomial can be written as a "d_1-form", which is in the ideal $\langle x_1 - a_1, x_2 - a_2\rangle^{d_1} \subseteq \mathbb{F}_p[\mathbf{x}]$ (Lemma 7). Next, we branch into a special val-multiplicity $= d_1$ part of the search tree, which requires a more complicated version of 'lifting steps' as we shall see.

To summarize, the degree reduction case of local root is where effective degree reduces ($v < d_1$ *suffices, due to Theorem 5); while val-multiplicity* d_1 *case is where val-mult* $v = d_1$*.*

Naively, the depth of this search tree can be $\Omega(k)$, as very few lifting steps may have degree reduction. Our second big idea is a way to overcome this basic obstruction (Algorithm 2).

At any given node, we consider the 'chains' that are responsible for the contiguous val-multiplicity d_1 branches. We store them in an array (D in Algo.1). This is done in the red part of the tree in Fig.1. After this is done, the search tree branches into the easier degree-reduction cases (branches with val-mult.$< d_1$); which is denoted by the green nodes (enclosed by the left rectangle in Fig.1).

So, in the tree, the lifting can either arise simply from the degree reduction cases, or can be a more complicated one where we transform the polynomial in Algorithm 2 after which we are guaranteed to branch into a degree reduction case.

This has been schematically portrayed in Figure 1, where the red node is a more complicated transformation than the simple lifting of a local root (done in green node).

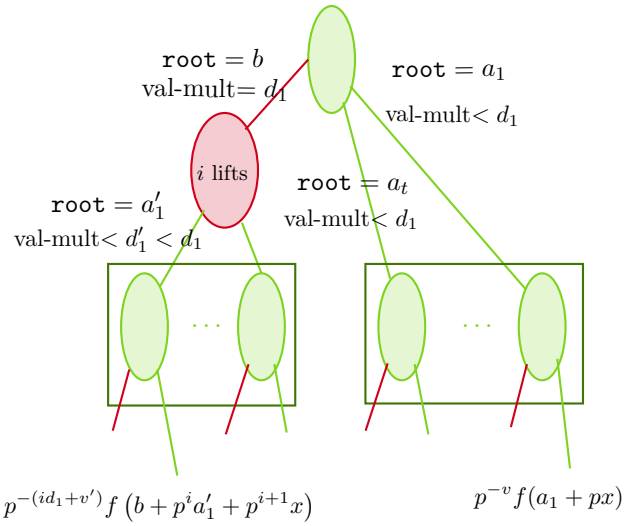

Figure 1: Branching along the search tree.

Tree depth and fanin. The crux of Algorithm 1 lies in our Theorem 5 which states that optimistically the effective degree reduces "most of the time". The algorithm ends when we either have exhausted the power of p, denoted by k, or when the effective degree becomes 1. The latter case (Theorem 5) gives a root and is essentially (p-adic) Hensel lifting on a *linear* polynomial [Hen18].

Algorithm 1 ensures that the tree-depth is $O(d)$, while the more complicated Algorithm 2 ensures that the number of branches at every node is $O(k^2dp)$.

Hensel's Lifting. Given a non-singular root $\mathbf{a}$ of polynomial $h(\mathbf{x})$, we can lift it to modulo any p-power (like in Theorem 5-(2)), using a variant of p-adic Hensel's lifting [Hen18]. Since $\mathbf{a}$ is a non-singular root, at least one of the first-order derivatives of $h(\mathbf{x})$ will not vanish. Corollary 6 implies that the val-multiplicity is then exactly 1, and in the next step of lifting, the effective polynomial will be *linear*. If this linear polynomial is of the form $m_1x_1 + m_2x_2 + m_0$, then (say) we can fix x_1 to any value in $[p]$ and find the corresponding unique value of x_2 to yield a root by simple lifting. For the next p-adic coordinate, after lifting, these m_1, m_2 (coefficients of x_1 resp. x_2) will not change; while m_0 might change. Thus, from Theorem 5-(2), we have the fact that the effective polynomial continues to stay linear, and we can fix the current-coordinate x_1 to find the corresponding x_2 every time; enabling us to lift to modulo any p-power (for arbitrary fixing of x_1 in this example).

Idea of Algorithm 2: Chain of d_1-forms. We create a process of 'removing' (or combining) contiguous val-multiplicity d_1 lifts, instead of brute-forcing over them in the search tree. This removal-process is guided by something called d_1-*forms* (Lemma 7), and will be subdivided into single and multiple val-multiplicity d_1 roots. (1) When *multiple* val-multiplicity roots exist, we show in Lemma 8 that the polynomial has a special form (namely d_1-*power*). We traverse these cases in a contiguous way. (2) Next we traverse over cases where val-multiplicity d_1 root is *unique*. At the end, we encounter lesser val-multiplicity roots and we can recursively call the root-finding algorithm. Overall, we find the lengths of these contiguous traversals, as well as the possibilities of the underlying d_1-powers resp. d_1-nonpowers. This is discussed in the latter-half of Section 3, by considering a dynamic basis-change on the variables $\mathbf{x}$, that guides the search for local roots well. In this process we *reduce* the root-finding of bivariate polynomials to that of a system of

univariate polynomials, and employ the idea of [BLQ13] to find representative-roots of a univariate polynomial system. Finally, see Step 15 (Algorithm 1) for the consolidated transformation that we call *contiguous chain of val-multiplicity d_1 lifts*.

To summarize, in this case of successive val-mult=d_1 lifts, we show that the contiguous chains are few (i.e., polynomial in k, d, p), and every branch appearing from these chains ends in a degree reduction case. This is depicted in the search tree (Fig.1) as a red node; it 'jumps' over all the val-mult=d_1 cases (Sections 3–4) before branching to the green nodes. This bounds the tree-depth to $2d$.

Stopping condition, representative roots and root-counting. The algorithm terminates when either a root gets completely specified mod p^k, or when effective degree ≤ 1 (any of its roots can be Hensel lifted all the way to our required power of p), or when no root exists. In the third case, the root-set returned is just the emptyset ϕ, while in the first case it is a singleton.

For the second case, roots will be returned in terms of representative roots. Eg. when the lifted polynomial is zero modulo p^ℓ, any value in $\mathbb{Z}/p^\ell\mathbb{Z}$ is a root, and thus we return $*_1$ resp. $*_2$ for the coordinates x_1 resp. x_2, which represent the entire $\mathbb{Z}/p^\ell\mathbb{Z}$. The roots returned will be $(*_1, *_2)$, with the number of possibilities being $p^{2\ell}$. This will be termed as our usual *representative root*.

When the effective degree is 1— as we sketched before, we can fix one variable, say x_1, as a local root and find the value of x_2 (in each p-adic coordinate one by one). Even if only one variable is present in the linear form, say x_2, the other variable x_1 will still be free; so, for any given value of x_1, denoted by $*$, we can find the corresponding values of the local roots of x_2, thus, yielding a root of the polynomial modulo p^ℓ. Let us denote this function for determining x_2 from any value of x_1 by $c(\cdot)$, which simply finds each coordinate of x_2 using Hensel's lifting. Thus, the output can be denoted as $(*, c(*))$. The number of roots represented in this expression is p^ℓ. This type of representative root will be termed as *linear-representative*. (Note: In the presence of an offset, eg. $(r_1 + p^{\ell_1}*, r_2 + p^{\ell_2}\mu(*))$, this case can contribute more roots, so a more careful calculation is done in Section 6.)

2 Evolution of effective degree during lifting steps

In this section, we analyze the effective degree at each step and look more closely as to when this decreases, or remains the same, by looking at the val-multiplicity of the local root during lifting. The proof idea is to analyze the monomials in terms of x_1 and x_2, and see how they behave after the transformation $(x_1, x_2) \mapsto (a_1 + px_1, a_2 + px_2)$ followed by division by appropriate power of p. This can be summed up by the following theorem.

Theorem 5 (Degree reduction)**.** *For a polynomial $f(x_1, x_2) \in (\mathbb{Z}/p^k\mathbb{Z})[x_1, x_2]$, given an $\mathbb{F}_p^2$-root (a_1, a_2) of $f(x_1, x_2)$, let us denote $g(x_1, x_2) := p^{-v} f(a_1 + px_1, a_2 + px_2)$, where $v := v_p(f(a_1 + px_1, a_2 + px_2))$. Let the previous effective degree be $d_1 := \deg(f(x_1, x_2) \bmod p)$ and current effective degree be $d_2 := \deg(g(x_1, x_2) \bmod p)$. Then the following holds:*

1. *If $d_1 > 1$, then $d_2 \leq v \leq d_1$. (So, $d_2 = d_1$ only if $v = d_1$.)*
2. *If $d_1 = 1$, then $d_2 = 1$.*

Proof. We have two cases.

Case 1: $\mathbf{d_1 > 1}$. We have a local root (a_1, a_2) such that $v = v_p(f(a_1 + px_1, a_2 + px_2))$. Using Taylor's expansion (Definition 17), we can write the polynomial in the form (say over $\mathbb{Z}_p$)

$$f(a_1 + px_1, a_2 + px_2) \;=\; \sum_{\ell=0}^{d} \left(\sum_{|\mathbf{i}|=\ell} \frac{\partial_{\mathbf{x}^\mathbf{i}} f(\mathbf{a})}{\mathbf{i}!} \cdot (px_1)^{i_1} (px_2)^{i_2} \right), \tag{2}$$

where $d := \deg(f)$. The terms in Equation 2 need to vanish modulo p^v for all $\ell \leq v$. In particular, $p^{v-|\mathbf{i}|} \mid \frac{\partial_{\mathbf{x}^\mathbf{i}} f(\mathbf{a})}{\mathbf{i}!}$. Suppose $v > d_1$, then by the above equation $f(a_1 + x_1, a_2 + x_2) \equiv 0 \bmod p$, implying $f(\mathbf{x}) \equiv 0 \bmod p$, which contradicts with $d_1 > 1$. Thus, $v \leq d_1$.

However, we do have a term which has valuation exactly v (= val-multiplicity of the local root), and this can be obtained only from monomials where $i_1 + i_2 \leq v$ (that too in the effective polynomial part). So, the highest degree term surviving among these (in $g \bmod p$) has degree $d_2 \leq v \leq d_1$.

Remark: The case of $d_2 = d_1$ implies that $v = d_1$. Thus, $p^{v-|\mathbf{i}|} \mid \frac{\partial_{\mathbf{x}^\mathbf{i}} f(\mathbf{a})}{\mathbf{i}!}$ for all orders $|\mathbf{i}| < d_1$; while, some order-d_1 partial-derivative at $\mathbf{a}$ has p-valuation exactly 0.

Case 2: $\mathbf{d_1 = 1}$ (Hensel's Lifting). Write $f(x_1, x_2) =: f_1(x_1, x_2) + p \cdot f_2(x_1, x_2)$. We have the effective polynomial $\deg(f_1(\mathbf{x})) = 1$, and hence it can be written as a linear polynomial $m_1 x_1 + m_2 x_2 + m_0$. Since (a_1, a_2) is a local root, we transform f to get $m_1(a_1 + px_1) + m_2(a_2 + px_2) + m_0 + p \cdot f_2(a_1 + px_1, a_2 + px_2)$. Dividing by p, and going mod$p$, we get in the next step to another linear polynomial: $m_1 x_1 + m_2 x_2 + m_0'$. So we end up with $d_2 = 1$. □

Example 2. *Let us see how the effective degree could* reduce. *Consider $f(x_1, x_2) = x_1^2 + x_2^3 \bmod p$. This has degree $d_1 = 3$. Clearly, $(0, 0)$ is its root modulo p. So, apply the transformation $(x_1, x_2) \mapsto (0 + px_1, 0 + px_2)$, to get $g(x_1, x_2) := p^{-2} f(px_1, px_2) = x_1^2 + px_2^3$, which has effective degree $d_2 = 2 = v < d_1$.*

The proof of Theorem 5 relies on analyzing the partial derivatives in Taylor's expansion. Using this technique, we get a corollary on partial derivatives of $f(\mathbf{x})$, which motivates the inclusion of the term 'multiplicity' in our new concept of 'val-multiplicity'.

Corollary 6. *Local root $\mathbf{a}$ of $f(\mathbf{x})$ has val-multiplicity $\geq v$, if and only if $p^{v-|\mathbf{i}|} \mid \frac{\partial_{\mathbf{x}^\mathbf{i}} f(\mathbf{a})}{\mathbf{i}!}$, for all orders $|\mathbf{i}| < v$.*

Using Theorem 5, we get the idea of effective degree reduction. If root $(a_1, a_2) \in \mathbb{F}_p^2$ is such that $f(a_1 + px_1, a_2 + px_2) \not\equiv 0 \bmod p^{d_1}$, then we can repeat the appropriate transformation $(x_1, x_2) \mapsto (a_1 + px_1, a_2 + px_2)$, until the effective degree reduces to 1. Once this happens, we have a *compact description* of all its roots by Hensel lifting, as we can arbitrarily fix one variable and uniquely find the p-adic value of the other variable.

However, the problem arises when the root (a_1, a_2) is such that $f(a_1 + px_1, a_2 + px_2) \equiv 0 \bmod p^{d_1}$. In this case, the degree may not reduce, and we might need to lift k/d_1 many times. This is computationally infeasible, the search-tree becomes very deep/large, and takes time exponential in k/d_1. We tackle this case next.

3 Structure of f via rank of local roots of val-mult=d_1

We need to handle the challenge of our local root $\mathbf{a}$ of f having val-multiplicity $v = d_1$. Here, the effective degree may not reduce in the next step. We first show the structure of such $f(x_1, x_2)$.

Lemma 7 (d_1-form at $\mathbf{a}$)**.** *If $\mathbf{a} \in \mathbb{F}_p^2$ is a root of $f(\mathbf{x}) \bmod p$ such that $f(a_1 + px_1, a_2 + px_2) \equiv 0 \bmod p^{d_1}$, where d_1 is the effective degree of f, then $f(\mathbf{x})$ is in the ideal $\langle x_1 - a_1, x_2 - a_2\rangle^{d_1} \subset \mathbb{F}_p[\mathbf{x}]$.*

Proof. Recall Taylor's expansion (Definition 17) and Corollary 6. Write $f(\mathbf{x})$ as $f((x_1 - a_1) + a_1, (x_2 - a_2) + a_2) = \sum_{\ell=0}^{\infty} A_\ell$, where,

$$A_\ell := \sum_{|\mathbf{i}|=\ell} \frac{\partial_{\mathbf{x}^\mathbf{i}} f(\mathbf{a})}{\mathbf{i}!} \cdot (x_1 - a_1)^{i_1} (x_2 - a_2)^{i_2} \ .$$

By Corollary 6, we know that the A_t's, for $t < d_1$, vanish modulo p as the root has val-multiplicity $v = d_1$. Furthermore, for $t > d_1$, A_t's vanish modulo p; as f has effective degree d_1 and this A_t has derivatives of order $> d_1$. Thus, all A_t's, apart from A_{d_1}, vanish modulo p. So, the polynomial f is of the form $\sum_i c_i \cdot (x_1 - a_1)^{d_1 - i}(x_2 - a_2)^i$, which is the required d_1-*form in* $\mathbf{x} - \mathbf{a}$. □

In Lemma 7's situation, if $\mathbf{a}$ is unique, then using the structure of f we can easily find the root (eg. a simple search in $\mathbb{F}_p^2$), and lift without getting into multiple val-mult=d_1 branching. A serious obstruction arises when there are *several* local roots $\mathbf{a}$ of val-multiplicity $= d_1$. We will now show the extra special structure of such an $f(x_1, x_2)$.

Without loss of generality, let $\mathbf{0}$ be a local root of val-multiplicity $= d_1$. This means that $f \in \langle x_1, x_2\rangle^{d_1}$. If another local root $\mathbf{a} \neq \mathbf{0}$ exists with val-multiplicity $= d_1$, then we also have $f(\mathbf{x}) \in \langle x_1 - a_1, x_2 - a_2\rangle^{d_1}$. So, $f \in \langle x_1, x_2\rangle^{d_1} \cap \langle x_1 - a_1, x_2 - a_2\rangle^{d_1} \subset \mathbb{F}_p[\mathbf{x}]$. Then, we show f to be a *perfect-power*!

Lemma 8 (Two val-mult=d_1 roots)**.** *For a polynomial $f \in \mathbb{F}_p[x_1, x_2]$ of degree d_1, if f is in the ideal $\langle x_1, x_2\rangle^{d_1} \cap \langle x_1 - a_1, x_2 - a_2\rangle^{d_1}$, for some $\mathbf{a} \neq \mathbf{0} \in \mathbb{F}_p^2$, then we have $f \equiv c \cdot (a_2 x_1 - a_1 x_2)^{d_1} \bmod p$, where $c \in \mathbb{F}_p^*$.*

Proof. W.l.o.g., assume that $a_1 \in \mathbb{F}_p^*$. Thus, we have

$$\begin{aligned} \langle x_1 - a_1, x_2 - a_2\rangle^{d_1} &= \langle x_1 - a_1,\ a_1 x_2 - a_1 a_2\rangle^{d_1} \\ &= \langle x_1 - a_1,\ a_1 x_2 - a_1 a_2 - a_2(x_1 - a_1)\rangle^{d_1} \\ &= \langle x_1 - a_1,\ a_1 x_2 - a_2 x_1\rangle^{d_1}. \end{aligned} \tag{3}$$

Also, $\langle x_1, x_2\rangle^{d_1} = \langle x_1, a_1 x_2 - a_2 x_1\rangle^{d_1}$ (as $a_1 \neq 0$). The intersection of these two ideals modulo the ideal $\langle a_1 x_2 - a_2 x_1\rangle$ is: $\langle a_1 x_2 - a_2 x_1\rangle + \langle x_1(x_1 - a_1)\rangle^{d_1}$ (as x_1 and $x_1 - a_1$ are coprime mod $a_1 x_2 - a_2 x_1$). Since f has effective degree less than $2d_1$, we deduce: $(a_2 x_1 - a_1 x_2) \mid f$.

The quotient $f/(a_2 x_1 - a_1 x_2) \in \langle x_1, a_2 x_1 - a_1 x_2\rangle^{d_1 - 1} \cap \langle x_1 - a_1, a_2 x_1 - a_1 x_2\rangle^{d_1 - 1}$. Clearly, degree of this quotient polynomial is $d_1 - 1$. So, we can repeat this process to show that $(a_2 x_1 - a_1 x_2)^{d_1} | f$; which makes the two equal up to a constant multiple. □

Essentially, this means f is d_1-th power of a linear polynomial iff rank of the val-mult=d_1 roots is two (i.e., multiple such roots). In the case of unique val-mult=d_1 root we will call the polynomial d_1-*nonpower-form*, while that for multiple val-mult=d_1 roots, we call the polynomial d_1-*power*.

Branching in d_1-nonpower-form. In this case, find the unique val-multiplicity d_1 root, and do the lifting step. There is no branching required.

Branching in d_1-power. Without loss of generality, the effective polynomial will be of the form $(a_2 x_1 - a_1 x_2)^{d_1}$. So, there are p roots (of val-mult=d_1), namely, $(a_1 t, a_2 t)$ for any $t \in \mathbb{F}_p$. This leads to branching, which we will avoid by inventing a different strategy.

The first observation (Lemma 9) is that d_1-nonpower-form can not lead to a d_1-power. Thus, we deduce that whenever a contiguous chain of d_1-power lifting ends, then every d_1-form in the subsequent contiguous lifting steps is a d_1-nonpower-form.

Lemma 9 (Nonpower to power?)**.** *If f is a d_1-nonpower-form having a single val-mult=d_1 root $\mathbf{a}$, then its lift $p^{-d_1} f(\mathbf{a}+p\mathbf{x})$ is not a d_1-power.*

Proof. W.l.o.g. we can assume $\mathbf{a}=\mathbf{0}$. Since the effective polynomial is a d_1-form having $(0,0)$ as the root, it is of the form

$$f(x_1,x_2) \equiv \sum_{i=0}^{d_1} c_i x_1^i x_2^{d_1-i} \bmod p\,. \tag{4}$$

After lifting given by $(x_1,x_2)\mapsto(px_1,px_2)$, followed by division by p^{d_1}, this polynomial will become

$$\sum_{i=0}^{d_1} c_i x_1^i x_2^{d_1-i} + g(x_1,x_2)\,,$$

for some polynomial g of degree $\leq d_1-1$. Suppose this lift is a d_1-power, say $(L+m_0)^{d_1} \bmod p$, where $m_0\in\mathbb{F}_p$ and L is a linear form in x_1,x_2. Now comparing the degree d_1 homogeneous parts in all these equations, we conclude that $f\equiv L^{d_1} \bmod p$. This contradicts the fact that it was a d_1-nonpower-form. Therefore, d_1-nonpower-forms can not become d_1-powers in one lifting step. □

So we mainly need to study the case: A d_1-power, say L^{d_1}, is followed by another d_1-power, say L'^{d_1}, in the next lifting step. Next, we unearth the structure that goes in the formation of L' after lifting the polynomial $L^{d_1}+\langle p\rangle$. This gives us the optimized bound on the branching of the red-nodes of the tree (Fig.1).

3.1 Structure of consecutive d_1-powers.

For a d_1-form, the effective polynomial $f(x_1,x_2) \bmod p$ will be of the form L^{d_1}, for some linear polynomial L (eg. x_1+x_2+1). Without loss of generality, assume $\{L,x_2,1\}$ to be of rank=3 (over $\mathbb{F}_p$). Let us rewrite f in the basis $\{L,x_2\}$, instead of $\{x_1,x_2\}$, denoted by $\tilde{f}(L,x_2)$ $(=f(\mathbf{x}))$. Since it is an invertible linear transformation, it now suffices to find roots of

$$\tilde{f} =: L^{d_1} + p\cdot L^{d_1-1}\cdot u_1(x_2) + p\cdot L^{d_1-2}\cdot u_2(x_2) + \cdots + p\cdot u_{d_1}(x_2)\,. \tag{5}$$

Lift d_1-power to d_1-power. Suppose that after lifting given by $p^{-d_1}\tilde{f}(pL,x_2)$, the effective polynomial is *again* a d_1-power; then it has to be the case that

$$\begin{aligned} L^{d_1} + L^{d_1-1}\cdot u_1(x_2) + L^{d_1-2}\cdot u_2(x_2)/p + \cdots + u_{d_1}(x_2)/p^{d_1-1} \\ \equiv (L+u_1(x_2)/d_1)^{d_1} \bmod p\,, \end{aligned} \tag{6}$$

for some univariate polynomials u_j's, such that Equation 6 is a perfect power of the linear polynomial $L+u_1(x_2)/d_1$. Consequently, those local roots a_2 for which the above system is satisfied, transform the previous 'root' L to $p\,(L+u_1(a_2)/d_1)$ in this lifting step.

Expanding the RHS, we also obtain equations, for $j\in[d_1]$, as

$$u_j(x_2) \equiv p^{j-1}\binom{d_1}{j}\cdot(u_1(x_2)/d_1)^j \bmod p^j\,. \tag{7}$$

Note: In the case where $p|d_1$, the above modulus can be further increased to clear away p-multiples from the denominator. In this fashion, we create a system of modular equations (in x_2) for the first step of lifting. Moving on, we consider the next lift.

Two consecutive d_1-power liftings. The effective polynomial after the first step was $(L + c(x_2))^{d_1}$. Let us look at the polynomials obtained before division by p^{d_1}. It was $(pL + pc(x_2))^{d_1} + \langle p^{d_1+1} \rangle$. Composing this with another lift of the same kind, the polynomial has to be of the form $(p(pL + pc(x_2)) + p^2\tilde{c}(x_2))^{d_1} + \langle p^{2d_1+1} \rangle$. This implies that we can directly lift $L \mapsto p^2L$, divide by p^{2d_1}, and find the value of $c(x_2) + \tilde{c}(x_2)$. So, Equation 6 can be replaced by the lift $p^{-2d_1}\tilde{f}(p^2L, x_2)$ equalling a d_1-power:

$$L^{d_1} + L^{d_1-1} \cdot u_1(x_2)/p + L^{d_1-2} \cdot u_2(x_2)/p^3 + \cdots + u_{d_1}(x_2)/p^{2d_1-1} \equiv (L + u_1(x_2)/pd_1)^{d_1} \bmod p\,. \tag{8}$$

Furthermore, we can write down the univariate modular equations like Equation 7 to find the root for x_2 that works in the lift.

In this way any i-length contiguous chain of d_1-power liftings, can be directly written as a system of univariate modular equations like Equation 7. It comes from the constraint that the lift $p^{-id_1}\tilde{f}(p^iL, x_2)$ has to equal a d_1-power mod p. Next, this system can be solved by adapting [BLQ13] to get the representative roots for the x_2 variable. Of course, on substituting this in x_2, we will know the final d_1-power L'^{d_1} that the contiguous i lifts yield.

How many consecutive d_1-powers? The length of this chain can be at most k/d_1. So, we go over all $i \leq \lfloor k/d_1 \rfloor$. Iterating over them in decreasing order, we find all the possible ways of getting d_1-powers (before moving to other cases). This ensures that we do not miss any $(\mathbb{Z}/p^k\mathbb{Z})$-root of f in the search-tree.

Example 3. *Consider the polynomial $f(x_1, x_2) = x_1^2 \bmod p^k$. The d_1-power contiguous chain will be of length $k/2$; and each time $L = x_1$. The corresponding root will be $(p^{k/2} \cdot *_1, *_2)$.*

Notation for x_2 representatives. A problem arises when the representative for x_2 is $*_2$, i.e., x_2 can take any p-adic value. Eg. if we lift $f = L^{d_1} + p^{d_1}x_2^{d_1+1}$ (with free $x_2 = *_2$) then we get $g := L^{d_1} + x_2^{d_1+1}$. The degree of the new polynomial has now *increased*, which we never want to happen in our tree branchings. In order to prevent this, we preprocess the representative root $x_2 = *_2$ by increasing the precision by one coordinate. In other words, we consider the representative as $a + p \cdot *_2$, for $a \in \{0, \ldots, p-1\}$.

The following lemma shows that the effective degree never grows in lifting steps, in our algorithm.

Lemma 10 (Degree invariant)**.** *The effective degree in each transformation described for d_1-forms is always d_1.*

Proof. Let us follow the above notation in the basis $\{L, x_2\}$ and start with effective degree d_1. We now know that every lifting step looks like the map: $L \mapsto pL$ and $x_2 \mapsto a_2 + p^{i_2}x_2$ (for some $i_2 \geq 1$ and integer a_2), followed by division by p^{d_1}. As we calculated in Theorem 5, such a lifting step yields effective degree $\leq d_1$. Since we are in the d_1-form case, this implies that the effective degree remains d_1 always. □

Summing up. The structure discovered above gives a natural pseudocode that we describe in Algorithm 2. The contiguous val-mult=d_1 chain will have some d_1-powers, say i_1 many, followed by i_3 many d_1-nonpower forms, from which we have $i_1 + i_3 \leq k/d_1$. The d_1-nonpower forms can not

lead to d_1-powers again, due to Lemma 9. Also, to get the i_1 many d_1-powers, we need to use the univariate root-finding of [BLQ13] and get representatives R_1 for x_2 (in general L_2, independent of L_1). Going over each $i_1, i_3 \leq \lfloor k/d_1 \rfloor$, and each of the representatives R_1, we compute the intermediate representative-roots R (and could continue with our recursion on the local roots with subsequent degree-reduction). This algorithm makes sure that, in the tree, we 'jump' the cases of val-mult= d_1 (effective degree) quickly, and reach the degree-reduction branchings.

4 The algorithm: Proof of Theorem 1

Using the above ideas, we prove Theorem 1 by giving the complete algorithm to find roots of a bivariate polynomial modulo p^k.

4.1 Main algorithm for root-finding

ROOT-FIND() in Algorithm 1 takes as *input*: the polynomial $f_j(x_1, x_2)$ and the number p^{k_j} ($k =: k_0$ initially). Main algorithm starts by calling ROOT-FIND$(f(x_1, x_2), p^k)$. If there are valid roots, it outputs the set of roots $R \subseteq (\mathbb{Z}/p^k\mathbb{Z})^2$, otherwise returns ϕ.

REMOVE-d_1-FORM() in Algorithm 2 eliminates intermediate lifts where effective degree does not decrease. It speeds-up the search for roots to higher precision coordinates, by jumping over contiguous cases of roots of val-multiplicity d_1. REMOVE-d_1-FORM() outputs an *array* of: A linear transformation which can be used to jump over the val-multiplicity d_1 roots, or a linear-representative root.

Algorithm 1 Root Finding of $f_j(x_1, x_2)$ mod p^{k_j}

procedure ROOT-FIND$(f_j(x_1, x_2), p^{k_j})$
  if $k_j \leq 0$ OR $f_j(x_1, x_2) \equiv 0 \bmod p^{k_j}$ **then return** $(*_1, *_2)$
  Define $d_1 := \deg(f_j \bmod p)$, $R := \phi$.
  if $d_1 = 1$ **then**
    return linear-representative $(*, c(*))$ or $(c(*), *)$, where $c(\cdot)$ is given by Hensel's Lifting.
  for $a_1 \in \{0, p-1\}$ **do**
    for a_2 such that $f_j(a_1, a_2) \equiv 0 \bmod p$ and val-mult($\mathbf{a}$)$< d_1$ **do**
      $f_{j+1}(x_1, x_2) := p^{-v} f_j(a_1 + px_1, a_2 + px_2)$, where $v := v_p(f_j(a_1 + px_1, a_2 + px_2))$.
      $S :=$ ROOT-FIND$(f_{j+1}, p^{k_j - v})$ *//aka green node in Fig.1*
      $R := R \cup (\mathbf{a} + pS)$
  if val-multiplicity= d_1 root exists **then**
    $D :=$ REMOVE-d_1-FORM(f_j, p^{k_j}) *//aka red node in Fig.1*
    for $(r_1 + p^{i_1} L_1, r_2 + p^{i_2} L_2, i_3) \in D$ **do**
      Write f_j in basis $\{L_1, L_2\}$ to get $\tilde{f}_j(L_1, L_2) := f_j(x_1, x_2)$.
      Lift it to $\tilde{f}_j(L_1, L_2) := p^{-i_3 d_1} \cdot \tilde{f}_j(r_1 + p^{i_1} L_1, r_2 + p^{i_2} L_2)$.
      if $k_j - i_3 d_1 \leq 0$ **then**
        The roots will be $(r_1 + p^{i_1} \cdot *_1, r_2 + p^{i_2} \cdot *_2)$ in (L_1, L_2) basis.
        Consider the tuple $(r_1 + p^{i_1} \cdot *_1, r_2 + p^{i_2} \cdot *_2)$ and perform the inverse linear transformation from (L_1, L_2) to (x_1, x_2) on this tuple as a whole. Store this representative root (with two independent $*$'s) in a set S.

19: $R := R \cup S$

20: **else**

21: For $\tilde{f}_j \bmod p^{k_j - i_3 d_1}$, find the val-mult $< d_1$ local roots and then recursively find all the roots; as done in Steps 6-10. Call this set $\tilde{R}$.

22: For each root $(\tilde{r}_1, \tilde{r}_2) \in \tilde{R}$ of $\tilde{f}_j \bmod p^{k_j - i_3 d_1}$: consider $(r_1 + p^{i_1}\tilde{r}_1, r_2 + p^{i_2}\tilde{r}_2)$ and perform inverse linear transformation from (L_1, L_2) to (x_1, x_2) on them. Store these final roots $(\bmod p^{k_j})$ in a set S.

23: $R := R \cup S$

24: **return** R

4.2 Remove-d_1-Form( ) subroutine: Handling contiguous d_1-forms (aka red nodes in Fig.1)

In Algorithm 1, we do not want the lifting to go on for several recursion steps; since, the time complexity is exponential in the number of steps (=tree-depth). The favorable case is when the effective-degree reduces, e.g., when the val-multiplicity of local root is $< d_1$ (from Theorem 5). As we will see, the Remove-d_1-Form() subroutine ensures inside the red nodes (whose starting local root has val-multiplicity $= d_1$) that the effective degree reduces when we go down to its child. In this section we sketch the pseudocode based on the ideas developed in Section 3.1.

Data structure returned. In order to lift the contiguous d_1-forms, we return an array of tuples of the form $(a_1 + p^{i_1}L_1, a_2 + p^{i_2}L_2, i_3)$. This gives us information on jumping over the val-mult$= d_1$ roots by first covering the d_1-powers followed by d_1-nonpowers. This is done in a basis (L_1, L_2) of variables possibly different from (x_1, x_2). As in Equation 8, we form equations in terms of L_2 and find the roots, such that after lifting according to these (representative) roots, the effective polynomial will be $L_1^{d_1}$. Note that in each lifting according to the fixed part of the representative root, the linear polynomial will change by only a constant. Therefore, $\{1, L_1, L_2\}$ will also span the same space as that of $\{1, x_1, x_2\}$. So, given a root in (L_1, L_2) basis, we can recover the root in (x_1, x_2) basis uniquely.

With information from this tuple, we can do the following sequence of liftings in 'one-shot': i_1-steps of d_1-powers at first, followed by i_3-steps of d_1-nonpower-forms.

Pseudocode. Summing up, Remove-d_1-Form( ) 'jumps' over the intermediate local roots of val-multiplicity d_1 so that Root-Find( ) can continue to degree reducing cases (in Steps 6-10 of Algo.1).

The *input* is the polynomial and the prime-power, while the *output* is a tuple of linear polynomials, denoting intermediate representative-roots (over $(\mathbb{Z}/p^k\mathbb{Z})^2$).

Algorithm 2 Finding intermediate val-mult=d_1 roots in one-shot

1: **procedure** Remove-d_1-Form$(f(x_1, x_2), p^k)$

2: Define $d_1 := \deg(f \bmod p)$, $R := \phi$.

3: **for** $i_1 \in \{\lceil k/d_1 \rceil, \ldots, 0\}$ **do**

4: $R_1 := \phi$

5: Find the linear polynomial L such that $f \equiv L^{d_1} \bmod p$. If L is x_2-multiple, then set $L_2 := x_1$, otherwise set $L_2 := x_2$.

6: Compute the (basis-change) polynomial $\tilde{f}$ such that $\tilde{f}(L, L_2) = f(x_1, x_2)$.

Write $\tilde{f}$ as in Equation 8 and form (univariate, modular) equations like Equation 7 in terms of polynomials in L_2 such that i_1-many contiguous d_1-powers exist (Section 3.1).
Find the representative-roots, in $\mathbb{Z}/p^{i_1 d_1}\mathbb{Z}$, of the system of equations formed in terms of L_2 (as in the previous step) using [BLQ13] and store them into R_1.
for each representative-root $r_2 + p^{i_2} * \in R_1$ **do**
    Find linear polynomial L_1 obtained in the end, by substituting the representative in L_2, using the method of Section 3.1. Note: $i_2 \geq 1$ and L_1 has to be of the form $L + c$ for some integer c.
    Write $\tilde{f}(L, L_2)$ in basis L_1, L_2 given by $g(L_1, L_2) := \tilde{f}(L, L_2)$.
    Lift $g(L_1, L_2) := p^{-i_1 d_1} \cdot g(p^{i_1} L_1\,,\, r_2 + p^{i_2} L_2)$
    for $i_3 \in \{\lceil k/d_1 \rceil - i_1, \ldots, 0\}$ **do**
        if $\exists \mathbf{r}' \in (\mathbb{Z}/p^{i_3}\mathbb{Z})^2$, g is d_1-nonpower-form consecutive i_3-times **then**
            In each precision $\mathbf{r}'$ is unique; so it can be searched easily in the space $\mathbb{F}_p^2$.
            $R_0 := (\, p^{i_1} r_1' + p^{i_1+i_3} L_1\,, r_2 + p^{i_2} r_2' + p^{i_2+i_3} L_2\,, i_1 + i_3\,)$
            if $R_0 \notin R$ **then**
                $R := R \cup \{R_0\}$
        else
            break
return R

Considering the degree reduction and the val-multiplicity d_1 cases, we get the final number of leaves in the search tree as $(\text{fanin})^{\text{depth}} = O((k^2 dp)^{2d})$, which gives us the following theorem.

Theorem 11 (Correctness of Algorithm 1)**.** *Given a polynomial $f \in \mathbb{Z}[x_1, x_2]$ of degree d, a prime p and an integer k. Algorithm 1 using Algorithm 2 as a subroutine, correctly returns all the roots $\mathbf{a} \in (\mathbb{Z}/p^k\mathbb{Z})^2$ of $f \bmod p^k$, in deterministic $\mathsf{poly}((k + d + p)^d)$ time.*

5 Computing p-adic roots: Proof of Corollary 2

In this section, we provide a bound for k_0 in terms of the degree d and the maximum absolute value M of the coefficients, such that finding a root modulo p^{k_0} would imply finding all representative (p-adic) $\mathbb{Z}_p$-roots of f. Let us assume we are given with the complete factorization of $f(x_1, x_2)$ given by $f(x_1, x_2) =: \prod_{i=0}^{r} g_i(x_1, x_2)^{e_i}$, where $g_i(x_1, x_2)$'s are coprime over $\mathbb{Z}_p$. Even if f has some square-full factors (some e_i's are ≥ 2), we can eliminate them efficiently, by computing its gcd with the first-order derivatives. This will result in the new polynomial being of the form $\prod_{i=0}^{r} g_i(x_1, x_2)$, which we will call the *radical polynomial* $\text{rad}(f)$. $\text{rad}(f)$ has the same set of roots over $\mathbb{Z}_p$ as that of $f(x_1, x_2)$, while its coefficients can be bounded as follows.

Lemma 12 (Bound for p-adic radical)**.** *If a polynomial f of degree d has the absolute-value of its coefficients bounded by M, then its radical polynomial $\text{rad}(f)$ has coefficients bounded by $M^{O(d)}$.*

Proof. We prove this result using Euclid's algorithm for finding the gcd of f and any one of its first-order derivative f'.

At each step of the Euclid's gcd algorithm, we have two polynomials q_i and q_{i+1}, where $\deg(q_i) \geq \deg(q_{i+1})$. There exist unique polynomials q, q_{i+2} such that

$$q_i = q \cdot q_{i+1} + q_{i+2}, \tag{9}$$

such that the remainder has $\deg(q_{r+2}) < \deg(q_{i+1})$ and $gcd(q_{i+1}, q_i) = gcd(q_{i+2}, q_{i+1})$. Using this, we proceed with the algorithm for gcd on q_{i+1} and q_{i+2}.

We show that the coefficients of q_i are bounded by M^{F_i}, where F_i is the i-th Fibonacci number. This can be proved using induction where we assume that the coefficients of q_j is bounded by M^{F_j} for all $j \leq i+1$. From equation 9, we infer that the quotient, q, will have its coefficients bounded by that of q_i. This quotient multiplied by q_{i-1} will give the bound for the remainder, which thus is bounded by the product of bounds of coefficients of q_i and q_{i+1}, which is $M^{F_i+F_{i+1}} = M^{F_{i+2}}$. Now, this procedure continues for $\log d$ steps, implying that the coefficients of the gcd of f and its derivative is bounded by $M^{F_{\log d}} = M^{O(d)}$. Dividing f by this gcd will give the bounds on the coefficient of $\mathrm{rad}(f)$, which is also $M^{O(d)}$. □

Therefore, without loss of generality we consider $f(x_1, x_2)$ to be square-free having absolute value of coefficients $\leq M^{O(d)}$, and continue with our algorithm of finding roots over $\mathbb{Z}_p$.

Representative roots over $\mathbb{Z}/p^k\mathbb{Z}$ vs roots over $\mathbb{Z}_p$. The output of the algorithm, in the base-case, is either the representative root $(*_1, *_2)$ when the exponent of p required gets achieved (Step 2 of Algorithm 1), or linear-representative root $(*, c(*))$ (Steps 4-5 of Algorithm 1).

The main objective of this section is to show that there exists a large enough k_0 such that for all $k > k_0$, if a representative root $(*_1, *_2)$ is returned $\bmod p^k$, then the fixed part of the root is already a $\mathbb{Z}_p$-root. In the other case, for linear-representative roots, we can simply use Hensel lifting to lift to $\mathbb{Z}_p$-roots (or, to as much precision as we wish).

Discriminant. Let $f' := \partial_{x_2} f(x_1, x_2)$ be the first-order derivative of f, assuming it to be non-zero without loss of generality. The resultant [CLO13, Chp. 3] of f and f' w.r.t. x_2 is denoted by $R(x_1) := \mathsf{Res}_{x_2}(f(x_1, x_2), f'(x_1, x_2))$, which is also one of the *discriminants* of f. $R(x_1)$ is not identically zero in $\mathbb{Z}_p$, as this would imply: f and its derivative have a common factor; contradicting the radical condition.

The roots of $R(x_1)$ given by $\hat{x}_1$ satisfy the condition that the univariate polynomial $f(\hat{x}_1, x_2)$ is square-full. Furthermore, given $\hat{x}_1$, we can easily find the values of x_2 (d-many), as it becomes the univariate root-finding problem over $\mathbb{Z}_p$ (solved in [BLQ13, DMS19, DS20]).

Bound to distinguish $\mathbb{Z}_p$ roots. The main idea is to find a bound for the exponent of p such that each root returned using root-finding is either a linear-representative root, or a unique lift of this root is a $\mathbb{Z}_p$ root. A similar bound was achieved for univariate polynomials by [DS20]. However, the complications of lifting multivariate roots did not arise there, as every p-adic root corresponded to a factor.

Lemma 13 (Discriminant of radical)**.** *Let $f \in \mathbb{Z}[x_1, x_2]$ be of degree d whose coefficients have absolute value bounded above by M. Let its radical polynomial be $g := \mathrm{rad}(f)$. The $\mathbb{Z}_p$-roots of $R(x_1) := \mathsf{Res}_{x_2}(g, g')$ are in one-one correspondence to the representative roots of $R(x_1) \bmod p^k$, for any $k \geq k_1 := \Theta(d^6 \log M)$.*

Proof. We have the polynomial $f(x_1, x_2)$ of degree d. Its radical polynomial $g := \mathrm{rad}(f)$, has degree $\leq d$ and coefficients bounded by $M^{O(d)}$ (Lemma 12).

The resultant polynomial $R(x_1) = \mathsf{Res}_{x_2}(g, g')$ is the determinant of a $(2d+1) \times (2d+1)$ matrix consisting of elements formed from the coefficients of g. Thus, the degree of $R(x_1)$ is $< 2d^2 + d$, and the absolute-value of the coefficients is $< (dM^{O(d)})^{2d+1}$.

Now, we need to find a bound on k_1 such that the roots of $R(x_1)$ are in one-one correspondence to those of $g(x_1, x_2) \bmod p^{k_1}$. [DS20, Thm. 20] showed that the representative roots of a univariate polynomial modulo p^k, for $k > d'(\Delta+1)$, are in one-one correspondence to the roots of that polynomial in $\mathbb{Z}_p$, where d' is the degree and Δ is the p-valuation of its discriminant. In our case of finding $\mathbb{Z}_p$-roots of $R(x_1)$, the degree is $2d^2+d$, while the discriminant is at most $((dM^{O(d)})^{2d+1})^{2d'+1}$, where $d' = 2d^2 + d$. Thus, the valuation of discriminant of R is bounded by $O(d^4 \log_p M)$. Substituting the values of d' and Δ, we have $k_1 := \Theta(d^6 \log_p M)$. □

$\mathbb{Z}_p$-roots. Consider $g = \mathrm{rad}(f)$ and $k_1 = \Theta(d^6 \log M)$. Define $g_2(x_2) := \mathrm{Res}_{x_1}(g(x_1, x_2), R(x_1))$, with $R(x_1) = \mathsf{Res}_{x_2}(g, g')$. The roots x_2 of g_2 originate from the values of the roots x_1 of R that make g square full. So, applying [DS20, Thm. 20] again on this univariate polynomial g_2, it suffices to compute its roots mod p^{k_2}, to compute its distinct p-adic roots; where k_2 is asymptotically $\log_p(p^{k_1 \cdot 2d^2 \cdot d}) = O(d^9 \log M)$.

Using the value of k_2 thus obtained, we find roots of $g(x_1, x_2)$ from $\textsc{Root-Find}(g, p^{k_2})$. Let $(\tilde{a}_1, \tilde{a}_2)$ be the fixed-part of a root obtained. If $R(\tilde{a}_1) \equiv 0 \bmod p^{k_2}$, then the above argument, that defined k_2, ensures that $(\tilde{a}_1, \tilde{a}_2)$ does lift to a $\mathbb{Z}_p$-root of R, g_2, g and finally f (in this case uniquely). However, there might also exist some roots over $\mathbb{Z}_p$ which do not necessarily arise from this case, as we shall see now.

Non-root of discriminant. The next case is that of $R(\tilde{a}_1) \neq 0 \bmod p^{k_2}$. Consider the univariate polynomial $g(\tilde{a}_1, x_2)$. We know that its $\mathbb{Z}_p$-roots are different $\bmod p^{k_2}$ and at most d many; one of which is $\tilde{a}_2$. Define $g_1(x_2) := p^{-v} \cdot g(\tilde{a}_1\,,\, \tilde{a}_2 + x_2)$, where $v \geq 0$ is the p-valuation of $g(\tilde{a}_1\,,\, \tilde{a}_2 + x_2)$ as a polynomial over $\mathbb{Z}_p$. Note that x_2 divides g_1, but x_2^2 does not divide $g_1 \pmod p$ since the resultant is non-zero. Thus, 0 is a simple-root of g_1 and we can potentially Hensel lift it to p-adics.

To implement this formally, we need to increase the precision so that the extra p-factors can be removed from g. Note that if we assume $p \nmid g(\tilde{a}_1, x_2)$ then $v \leq k_2 + (k_2 - 1)(d-1) < d \cdot k_2$. Fix $k_0 := d \cdot k_2 = \Theta(d^{10} \log M)$. Now consider $\tilde{g}(\mathbf{x}) := p^{-v} \cdot g(\tilde{a}_1 + p^{k_2} x_1\,,\, \tilde{a}_2 + p^{k_2} x_2) \bmod p^{k_0}$. By the argument above, $\tilde{g} \bmod p$ is linear in x_2 (it is easier to see by substituting $x_1 = 0$). Thus, an extension of this root has to end up in some leaf of $\textsc{Root-Find}(g, p^{k_0})$ algorithm as say $(\tilde{a}_1', \tilde{a}_2')$; which will Hensel lift to p-adic integral root(s) due to the linear x_2 term in the lift.

Since the set of p-adic roots for f and g is the same, we could as well run $\textsc{Root-Find}(f, p^{k_0})$. This proves the following lemma.

Lemma 14 (p^{k_0} is p-adic)**.** *Let $f \in \mathbb{Z}[x_1, x_2]$ be of degree d and having absolute-value of coefficients bounded by M. Each root represented in the leaves of the tree of* $\textsc{Root-Find}(f, p^{k_0})$*, for $k_0 := \Theta(d^{10} \log M)$, lifts to a $\mathbb{Z}_p$-root of $f(x_1, x_2)$.*

We further need the condition that the structure of this tree does not change with k for $k \geq k_0$. In order to show that, we prove the following lemma. Denote $R_1(x_1) := \mathsf{Res}_{x_2}(g, \partial_{x_2}(g))$ and $R_2(x_2) := \mathsf{Res}_{x_1}(g, \partial_{x_1}(g))$

Lemma 15 (Fix p-adic tree)**.** *If a leaf of the tree given by Lemma 14 returns a representative root with the fixed part (a_1, a_2), that is not linear-representative, then $R_1(a_1) = R_2(a_2) = 0 \bmod p^k$.*

Moreover, (a_1, a_2) lifts to a unique root of f over $\mathbb{Z}_p$; and their number does not change as k grows beyond k_0.

Also, the tree (Fig.1) in our algorithm does not change, and remains isomorphic, for $k \geq k_0$; except the leaf with the root $\mathbf{0}$.

Proof. As argued above, the representative roots which are not linear-representatives, must satisfy the condition $R_1(a_1) \equiv 0 \bmod p^{k_0}$. Similarly, we can show that $R_2(a_2) \equiv 0 \bmod p^{k_0}$.

Assume that for some large enough k, $k \geq k_0$, a new leaf in the tree of Figure 1 appears, with the root (r_1, r_2) such that $R_1(r_1) \equiv R_2(r_2) \equiv 0 \bmod p^k$. However, this leads to a contradiction as the branch corresponding to (r_1, r_2) should have already been present in the tree at precision k_0 in the representative root.

As argued before, this root r_j of $R_j \bmod p^{k_0}$ always leads to $\mathbb{Z}_p$ roots of R_j for $j = 1, 2$ (due to [DS20, Thm. 20]), and that of g_2, g, f. Thus, the number of representative roots can not decrease and the number of representative roots which are not linear is fixed once we reach k_0, and hence (a_1, a_2) has a unique lift to $\mathbb{Z}_p$.

Together with Hensel lifting, it is then clear that, the linear-representative roots can neither increase in number, nor reduce, as $k \geq k_0$ grows.

The only possible change is, for $k \geq k_0$, if the leaf with fixed root $\mathbf{0}$ is used to lift to $f(p^v x_1, p^v x_2) \bmod p^k$, with $k > v \geq k_0$. This may create a new subtree under the old leaf $\mathbf{0}$; as these types of branches are the only ones that were not explored in Algorithm 1 mod p^{k_0}. □

The following examples should help illustrate the p-adic machinery more clearly.

Example 4. *Consider the polynomial $f = (x_1 - 1)(x_2 - 2) \bmod p^k$. The first step of our algorithm has to be $x_1 = 1$ or $x_2 = 2$. Considering the root $\mathbf{a} := (1, 3)$, the polynomial after lifting becomes $x_1(1 + px_2)$, which is an (effective) linear form. Thus, a linear-representative root will be returned, which has x_2 as the free variable while x_1 will stay fixed to 1. This gives the leaf $\mathbf{r} := (1+p\mu(*), 3+p*)$, and a computable $\mathbb{Z}_p$-function $\mu(\cdot)$, which allows the p-adic lift of $\mathbf{a}$. In this case, $\mu = 0$.*

Example 5. *Now, consider $f(x_1, x_2) = (x_1 - px_2)(x_1 - 2px_2) \bmod p^k$. Lifting the root $\mathbf{a} := (0, 1)$ gives us $(x_1 - 1 - px_2)(x_1 - 2 - 2px_2)$, which is not effective linear yet. Choosing the next lifting-step around the root $(1, 0)$, the polynomial after lifting becomes $(x_1 - px_2)(-1 + px_1 - 2p^2x_2)$, which is an (effective) linear polynomial. A linear-representative root will be returned, corresponding to $(x_1 - px_2)$, which has x_2 as the free variable while x_1 depends on it. This gives the leaf $\mathbf{r} := (p + p^3\mu(*),\ 1 + p^2*)$, and a computable $\mathbb{Z}_p$-function $\mu(\cdot)$, which allows the p-adic lift of $\mathbf{a}$. In this case, $\mu(w) := w$.*

Blowing up the root 0. There may be $\mathbb{Z}_p$-roots which can not be 'noticed' modulo p^{k_0}, because they are indistinguishable from $\mathbf{0}$. This is seen in the following example.

Example 6. *Define the polynomial $x_1^3 + x_2^3 \bmod p^k$, for $p > 3$ and $3|k$. A subset of its linear-representative roots are $(p^j + p^{j+1}*, -p^j + p^{j+1}\mu(*))$, for any $j < k/3$ and $\mu(w) := -w$. However, $(p^{k/3}*_1, p^{k/3}*_2)$ is a non linear-representative root. It can lift to the p-adic root $\mathbf{0}$, but it can also lift to $(p^k, -p^k)$; which our algorithm could not explore due to the precision being only p^k.*

The following theorem completes the connection, of Algorithm 1, with *all* p-adic roots of f. Fundamentally, it scales up the roots by p^v-multiple, whenever possible, and creates a new data-structure for representatives in the leaves of the fixed tree modulo mod p^{k_0}, in Fig.1. It can also be seen as a way of further *blowing-up* the leaf of the fixed tree that gives the $\mathbf{0}$ root.

Theorem 16 (High val p-adic roots)**.** *We can efficiently 'expand' the leaves, of the search tree, as follows:*

(1) Define a set of representative-roots $\mathcal{H}_v$, for $v \geq k_0$, s.t. for each root $\mathbf{a} \in \mathcal{H}_v$, $p^v\mathbf{a}$ lifts to a p-adic root of f.

(2) We can compute the fixed tree for $\mathcal{H}_{k_0}$ efficiently by Algorithm 1. The other sets $\mathcal{H}_v$, for $v > k_0$, lift from the same representatives as in the leaves of $\mathcal{H}_{k_0}$; so we do not recompute them.

Let (r'_1, r'_2) be a p-adic root of f. Then, $\exists v \geq 0$, $\exists$ root $\mathbf{a} \in \mathcal{H}_v$ lifting to $\mathbf{a}'$, for which $(r'_1, r'_2) = p^v\mathbf{a}'$. In this sense, our fixed finite tree covers all (∞-many) p-adic roots of f.

Proof. Let u be the p-valuation of $\mathbf{r}'$, i.e., $p^u||(r'_1, r'_2)$. If $u = \infty$, i.e., $(r'_1, r'_2) = \mathbf{0}$, then clearly some leaf in the set $\mathcal{H}_{k_0}$ will satisfy the required statement.

If $u < k_0$, then $\mathbf{r}' \neq 0 \bmod p^{k_0}$; so it will be covered in some nonzero leaf of the tree of Lemma 15.

Assume $\infty > u \geq k_0$. Now consider the system $f(p^u \cdot \mathbf{x}) = 0$, say over $\mathbb{Z}_p$. Write $f =: \sum_{m \leq i \leq d} f_i$ into homogeneous-parts, with m being the least-degree part ($f_m \neq 0$). Thus, by homogeneity, the system becomes

$$0 \;=\; f(p^u \cdot \mathbf{x})/p^{um} \;=\; \sum_{m \leq i \leq d} p^{u(i-m)} \cdot f_i(x_1, x_2)\,. \tag{10}$$

If $f = f_m$ (i.e., f is homogeneous), then $f(p^u \cdot \mathbf{x}) = 0$ iff $f(\mathbf{x}) = 0$. Thus, $\mathcal{H}_v$, for all $v \geq 0$, is given by the fixed tree in Lemma 15 and we are done.

Assume $f \neq f_m$ (i.e., f is *in*homogeneous). Then, the above system implies: $f_m(x_1, x_2) \equiv 0 \bmod p^u$. Since f_m has bounded coefficients and $u \geq k_0$, we compute the fixed tree (Lemma 15 for $f_m \bmod p^{k_0}$) efficiently; and all its leaves (except $\mathbf{0}$) lift to p-adic roots. Each leaf, in our Algorithm 1 can be viewed as defining a nontrivial p-adic map $\mu : \mathbb{Z}_p \to \mathbb{Z}_p$ s.t. w.l.o.g., $f_m(w, \mu(w)) = 0$, where w is a variable. Check whether $f(p^u w\,, p^u\mu(w)) = 0$. Then, by repeating this argument (on f_{m+1} etc.) we can deduce: $f_i(w, \mu(w)) = 0$ for all $m \leq i \leq d$. Since, μ is a p-adic function common to these polynomials, that are all upper bounded by the parameters of f, we can learn μ by working with each f_i just $\bmod p^{k_0}$.

Algorithmically, we find this common μ by first invoking Algorithm 1 on $f_m \bmod p^{k_0}$ and then verifying it for $f(p^{k_0}w\,, p^{k_0}\mu(w)) \equiv 0 \bmod p^{k_0(d+1)}$. [Or, we could construct the tree common to the system $\{f_m, \ldots, f_d\} \bmod p^{k_0}$.]

But this shows an interesting property p-adically that $f(p^{k_0}w\,, p^{k_0}\mu(w)) = 0$ iff $f_i(w, \mu(w)) = 0$, for all $m \leq i \leq d$ iff $f(p^u w\,, p^u\mu(w)) = 0$, for all $u \geq 0$. Essentially, the high-valuation roots arise only from homogeneous polynomial system! □

Lemmas 14-15 and Theorem 16 describe the p-adic nature of the tree and the representative roots, after the threshold bound of k_0. We can continue this tree until the required precision, which finishes the proof of Corollary 2.

6 Computing Igusa's local zeta function: Proof of Corollary 3

We will show how to compute the Poincaré series, by expressing the number of roots of $f(x_1, x_2) \bmod p^k$, for every k, in a special form. In this subsection, for ease of understanding, we provide this algorithm for bivariates using Section 5. The techniques used generalize to n-variates based on the machinery of Section 7; thus proving the rationality of the Poincaré series in general.

When we consider f modulo p^k, for large enough k's, the fixed-part of the representative-roots will correspond to p-adic roots, while the remaining-part has 'free' coordinates, eg. $(*_1, *_2)$, which get fixed as we increase k. For $k = k_0$, denote $\mathcal{R}$ as the subset of representative-roots which are not linear-representative roots, while the set $\mathcal{L}$ as the set of linear-representative roots.

Recall the bound of $k_0 = \Theta(d^{10} \log M)$ (Lemma 14) to distinguish between $\mathbb{Z}/p^k\mathbb{Z}$ and $\mathbb{Z}_p$-roots: For small values of k, i.e., $k < k_0$, we can count the number of roots in deterministic time $\mathsf{poly}((d + p + \log M)^d)$. For large k's, however, we want to prove a special form to sum up the infinite Poincaré series.

Non linear-representative roots $\mathcal{R}$. Consider a root in $\mathcal{R}$; its fixed-part, say $\mathbf{r}$, will lift to $\mathbb{Z}_p$-roots of f. Its representative part appears due to the contribution of extra p-powers by the other derivatives that appear in the Taylor-series around $\mathbf{r}$. Let e be the multiplicity of $\mathbf{r}$: which can be found as the (largest) e such that $f \in \langle \mathbf{x} - \mathbf{r} \rangle^e$, but $f \notin \langle \mathbf{x} - \mathbf{r} \rangle^{e+1}$.

Now, f can be written as

$$f \;=\; \sum_{i=0}^{e} c_i(x_1, x_2) \cdot (x_1 - r_1)^i (x_2 - r_2)^{e-i} \,. \tag{11}$$

Define $v := v_p(\gcd(\{c_i(\mathbf{r}) \mid i\}))$. Since $\mathbf{r}$ is not a common root of c_i's, this value (e, v) does not change as we increase k and make $\mathbf{r}$ more precise. Consider the $\mathbb{Z}_p$-root $\mathbf{r}'$ that $\mathbf{r}$ lifts to. Let us now consider the ways in which the digits of $\mathbf{r}'$ may be perturbed, and yet it be a root (mod p^k), as we increase k arbitrarily. Let ℓ_1 denote the length up to which we want to keep r_1 equal to r'_1 and the rest to $*$ (similarly define ℓ_2). Consider

$$f(r'_1 + p^{\ell_1} x_1 \,, r'_2 + p^{\ell_2} x_2) \;=\; \sum_{i=0}^{e} c_i(\mathbf{r}' + p^{\mathbf{l}}\mathbf{x}) \cdot (r'_1 - r_1 + p^{\ell_1} x_1)^i \cdot (r'_2 - r_2 + p^{\ell_2} x_2)^{e-i} \,. \tag{12}$$

The valuation of this expression is the minimum of $v_p(c_i(\mathbf{r})) + \ell_1 i + \ell_2(e - i)$, over all i. If this is $\geq k$ then $\mathbf{x}$ can take any value in $\mathbb{Z}/p^{k-\ell_1}\mathbb{Z} \times \mathbb{Z}/p^{k-\ell_2}\mathbb{Z}$, and extend, by the above equation, to a root mod p^k. This is what should happen as we are not in the linear-representative case. We need to count these possibilities, which will give us the number of ways the root $\mathbf{r}$ could lift as k increases. For this, we should consider only those possibilities of (ℓ_1, ℓ_2) that are *minimal*, i.e., $(\ell_1 - 1, \ell_2)$ and $(\ell_1, \ell_2 - 1)$ should violate the linear-inequality system. This is a system of half-spaces in the plane, forming an open polygon $\mathcal{P}$ with either $\leq e$ vertices or just one hyperplane. It can be checked that in all cases the counting function

$$\sum_{(\ell_1, \ell_2) \in \mathcal{P}} p^{(k-\ell_1)} \cdot p^{(k-\ell_2)} \tag{13}$$

can be rewritten as a sum of $p^{u_i(k)}$, $i \leq e$, where $u_i(k)$ is a linear function in k over $\mathbb{Q}$. Let us call this sum $N_{k,\mathbf{r}}(f)$. Importantly, the number of summands here is fixed, and does not grow with k.

The following example illustrates the notion of this polytope.

Example 7. *Consider the polynomial $f(x_1, x_2) = x_1^2 x_2 \bmod p^k$. Here, for the root $(0, 1)$, the value of e is 2, and accordingly ℓ_1, the precision of x_1 required, is $= k/2$. However, the value of e for $(1, 0)$ is 1 and $\ell_2 = k$. For the root $\mathbf{0}$, both variables contribute powers of p, where they are zero with precision ℓ_1, ℓ_2 respectively. Then, we must have $2\ell_1 + \ell_2 \geq k$, which gives the hyperplane in ℓ_1, ℓ_2; summing over all these values we can calculate Eqn.13.*

Linear-representative roots $\mathcal{L}$. Consider a linear-representative root $\mathbf{r} \in \mathcal{L}$, with fixed part $\mathbf{a}$ of length (e_1, e_2) respectively. Up to linear transformations, we can claim that our algorithm defines a computable $\mathbb{Z}_p$-function $\mu(\cdot)$ s.t. for all $u \in \mathbb{Z}_p$, $f(x_1 + a_1 + p^{e_1}u\,, x_2 + a_2 + p^{e_2}\mu(u)) = 0$. Using this fact, we write Equation 11 in the ideal form, defining (e, v) as the largest integers s.t.,

$$f(x_1 + a_1\,, x_2 + a_2 + p^{e_2}\mu(0)) \;\in\; p^v \cdot \langle x_1\,, x_2\rangle^e \;+\; \langle x_1\,, x_2\rangle^{e+1}\,, \tag{14}$$

over $\mathbb{Z}_p[\mathbf{x}]$.

Note that we have defined (e, v) by fixing $u = 0$. The motivation is that if we use some other u in $\mathbb{Z}_p$, we will get the same values (e, v). To show this, say for some $0 \neq u \in \mathbb{Z}_p$,

$$f(x_1 + a_1 + p^{e_1}u\,, x_2 + a_2 + p^{e_2}\mu(u)) \;\in\; p^{v'} \cdot \langle x_1\,, x_2\rangle^{e'} \;+\; \langle x_1\,, x_2\rangle^{e'+1}\,.$$

Then, by Lemma 15, $e' = e$, because the tree remains isomorphic, even when we make the root more precise than $k \geq k_0$. In the same algorithm, lifting steps cause division by p-powers and reach the leaf $\mathbf{r}$, so v' remains v even when we make the root more precise than $k \geq k_0$.

Fix $u \in \mathbb{Z}/p^{k-e_1}\mathbb{Z}$, and consider the unique p-adic root $\mathbf{r}'$ that the leaf $\mathbf{r}$ gives above. We follow a similar process of counting as done for Equation 12. Varying u, the polytope boundary $\mathcal{P}$ does not change (similar to the argument given after Equation 14); while on the other hand, fixing $(\ell_1 - e_1)$-digits of u (resp. ℓ_1 value), fixes that many in $\mu(u)$ (resp. ℓ_2 value). Thus, we get a *partial* count (slightly different from Equation 13) as:

$$\sum_{(\ell_1,\ell_2)\,\in\,\mathcal{P}\,\cup\,\{(k-e_2)-(\ell_1-e_1)\,\leq\,(k-\ell_2)\}} p^{(\ell_1-e_1)} \cdot p^{(k-e_1)-(\ell_1-e_1)} \cdot p^{(k-e_2)-(\ell_1-e_1)}\,. \tag{15}$$

We call this sum $N'_{k,\mathbf{r}}(f)$; and is written as a sum of $p^{u_i(k)}$, $i \leq e$, where $u_i(k)$ is a *linear* function in k over $\mathbb{Q}$.

Blowing-up root 0. We are left with roots with valuation in the interval $[k_0, k-1]$, since these are zero mod p^{k_0} and does not get included in $\mathcal{L}$ or $\mathcal{R}$. To account for these, we need to resort to the set $\mathcal{H}_{k_0}$, constructed in Theorem 16 that 'blows-up' the leaf node $\mathbf{0}$ in the tree of finding roots modulo p^{k_0}.

Now, from the way Algorithm 1 works, the representatives in $\mathcal{H}_{k_0}$ generate a disjoint set of $(\mathbb{Z}/p^k\mathbb{Z})$-roots, which are in number $= \sum_{\mathbf{r}\in\mathcal{H}_{k_0}} N_{k_0,\mathbf{r}}$. This sum is easy to precompute (by Theorem 1), as it is independent of k. Each of these roots can be multiplied by p^e, for $e \in [k_0 \ldots k-1]$, to get the 'high'-valuation roots. Thus, the total number of such roots is $= (k - k_0) \cdot \sum_{\mathbf{r}\in\mathcal{H}_{k_0}} N_{k_0,\mathbf{r}}$.

Overall, the above summands account for all the $\mathbb{Z}/p^k\mathbb{Z}$-roots of f, for $k \geq k_0$.

Summing up, the number of roots modulo p^k, for $k < k_0$, can be counted by Theorem 1. Fixing $k = k_0$, we compute the data-structures related to the fixed tree; which has the linear-representative roots in $\mathcal{L}$, $\mathcal{H}_{k_0}$, and the remaining representative-roots in $\mathcal{R}$.

Then, the number of roots of f modulo p^k, $N_k(f)$, is given by

$$N_k(f) \;=\; \sum_{\mathbf{r}\in\mathcal{R}} N_{k,\mathbf{r}}(f) \;+\; \sum_{\mathbf{r}\in\mathcal{L}} N'_{k,\mathbf{r}}(f) \;+\; (k - k_0) \cdot \sum_{\mathbf{r}\in\mathcal{H}_{k_0}} N_{k_0,\mathbf{r}}\,. \tag{16}$$

Recall that the number of roots modulo p^k due to any representative root $\mathbf{r}$ (or a leaf in the fixed tree in Theorem 16) is $N_{k,\mathbf{r}}(f)$ resp. $N'_{k,\mathbf{r}}(f)$; defined separately in Equations 13 and 15.

Now, the Poincaré series is given by (eg. see [DS20])

$$P(t) \;=\; P_0(t) + \sum_{\mathbf{r}\in\mathcal{R}} P_{\mathbf{r}}(t) + \sum_{\mathbf{r}\in\mathcal{L}} Q_{\mathbf{r}}(t) + \left(\sum_{\mathbf{r}\in\mathcal{H}_{k_0}} N_{k_0,\mathbf{r}}\right)\cdot \sum_{k\geq k_0}(k-k_0)\cdot(t/p)^k\,, \tag{17}$$

where $P_0(t) = \sum_{k<k_0} N_k(f)\cdot(t/p)^k$, $P_{\mathbf{r}}(t) := \sum_{k\geq k_0} N_{k,\mathbf{r}}(f)\cdot(t/p)^k$ and $Q_{\mathbf{r}}(t) := \sum_{k\geq k_0} N'_{k,\mathbf{r}}(f)\cdot(t/p)^k$.

The expression of Equation 16 is a sum of $p^{u_i(k)}$'s and $u_i(k)$'s, $i \leq O(d)$, where $u_i(k)$'s are *linear* functions in k over $\mathbb{Q}$. Thus, Equation 17 can be easily expressed in a 'closed-form formula' by summing the geometric progression over k's, as was done in [DS20, Lem. 23]. Thus, $P_{\mathbf{r}}(t), Q_{\mathbf{r}}(t)$ are rational functions in $\mathbb{Q}(t)$. Therefore, the rational function for the Poincaré series $P(t)$ is computable as promised in Corollary 3.

7 Generalization to n-variates: Proof of Theorem 4

In this section, we generalize our approach of root-finding and counting to n-variate polynomial $f(\mathbf{x})$ using similar techniques as used in bivariates. The key idea is to reduce the problem to finding roots of $(n-1)$-variate polynomial systems. Algorithm 3 contains the simple modification of the algorithms of [BLQ13, DMS21] to univariate polynomial system-solving. For the case of multivariates, we first show a modification to Algorithm 1 for solving a system of polynomial equations $\bmod p^k$. The next subsection gives an overview of how to extend the single bivariate root-finding algorithm to that for solving a *system* of bivariates. Then, using that, we solve 3-variate systems. The proof can be straightforwardly generalized to n-variates, for any $n \geq 3$.

7.1 Solving bivariates simultaneously

Suppose we have m polynomials $f_1(x_1,x_2),\ldots,f_m(x_1,x_2)$, each of degree $\leq d$. At each step of lifting, we iterate over all the roots of each polynomial separately, but in parallel, such that the local roots of each iteration are common to every polynomial. This can be intuitively thought of as creating trees as in Figure 1 corresponding to each polynomial, whose nodes are 'isomorphic', i.e., we branch corresponding to a root if and only if the root is present in the trees of all the polynomials. Also, whichever linear transformation we apply on $\mathbf{x}$ acts on all the f_i's simultaneously. We refer to this data-structure as a *parallel* search-tree.

We continue growing these trees (and considering only those branches which correspond to a common root); with effective degree decreasing as we go down, until the stopping conditions are satisfied. When some polynomial has a representative root $(*_1,*_2)$, we proceed to finding roots of the remaining set of polynomial. However, when linear-representative roots are obtained for some polynomial, the analysis can be divided into two cases by their *rank*. When the linear forms (i.e., given by coefficients of x_1 and x_2) are of rank 1 or 2. For rank 2, we can check for a unique root by solving simple linear equations; so, they either have one common root, or none.

The difficulty arises when these linear forms are of rank 1. Again, the isomorphism of the trees corresponding to each polynomial will be used, but after reducing this problem to solving simultaneous equations over a fewer number of polynomials.

Rank= 1 linear forms. Suppose we have the polynomials in the form $ax_1+bx_2+c_i-ph_i(x_1,x_2)$, where $a,b,c_i \in \{0,\ldots,p-1\}$, for all $i\in[m]$. If all the c_i's are not the same, we obviously do not

have a solution; so we terminate this branch. Assume $c_i = c$ and write the polynomials in the basis of $L := (ax_1 + bx_2 + c)$ and x_2, w.l.o.g assuming $a \neq 0$. (Otherwise, we can use the basis L, x_1). We have the system as

$$\begin{aligned} L &\equiv pg_1(L, x_2) \bmod p^k\,; \\ L &\equiv pg_2(L, x_2) \bmod p^k\,; \\ &\vdots \\ L &\equiv pg_m(L, x_2) \bmod p^k\,, \end{aligned} \tag{18}$$

where g_i's can be obtained from h_i's by using the change of basis. So, we get the local-root here, namely L must be 0 mod p in the current step (based on any value of x_2). Hence we lift L to pL. This grows the tree further. Performing this transformation and subtracting the first equation from each of the other equations, we have :

$$\begin{aligned} L &\equiv g_1(pL, x_2) \bmod p^{k-1}\,; \\ 0 &\equiv \tilde{g}_2(pL, x_2) \bmod p^{k-1}\,; \\ &\vdots \\ 0 &\equiv \tilde{g}_m(pL, x_2) \bmod p^{k-1}\,, \end{aligned} \tag{19}$$

where $\tilde{g}_i = g_i(L, x_2) - g_1(L, x_2)$, for $i \in \{2, \ldots, m\}$.

Now, on the $(m-1)$-many $\tilde{g}_i$'s, we find the values of x_2 that satisfy the equations mod$p$. For a fixing of x_2 from the latter $m-1$ equations in Equation 19, we uniquely get the value of L from the first equation (where the effective polynomial is linear in L). Using these values, we lift both L and x_2, creating a Figure 1-like tree where the branches are such that they satisfy all the m equations. Finally, at the leaf we get the representatives for x_2 too, and halt the algorithm.

Thus, we create the m isomorphic trees for $O(d)$ many lifting steps as before (Figure 1), restrict the linear condition on the first polynomial and simultaneously solve the next $m-1$ polynomials modulo a smaller power of p and continue with our construction of $m-1$ isomorphic trees. Using this subroutine, we can find all the roots of the system of bivariate equations in time complexity same as that of Theorem 1, along with a multiplicative overhead of m for storing this array of polynomials in each node.

7.2 Solving trivariates polynomial systems

Lifting Step. Given a root $\mathbf{a} \in \mathbb{F}_p^3$ and polynomial $f_j(\mathbf{x})$ in the j-th step, we find the polynomial after lifting as $f_{j+1}(\mathbf{x}) = p^{-v} f_j(\mathbf{a} + p\mathbf{x})$, where $v = v_p(f_j(\mathbf{a} + p\mathbf{x}))$ is the val-multiplicity of the root $\mathbf{a}$. Theorem 5 can be similarly proved to show that effective degree reduces in all cases other than when $d_1 = 1$ or $v = d_1$. We again form a tree similar to Figure 1 with the invariant that effective degree must reduce along depth.

Val-multiplicity d_1 case. As in Algorithm 2, we again 'jump' over the val-multiplicity d_1 cases directly so that the recursion can continue to degree reduction cases. Lemma 7 can be proved for 3-variates to show that the polynomial must have the d_1-form $\langle x_1 - a_1, x_2 - a_2, x_3 - a_3 \rangle^{d_1}$, where $\mathbf{a}$ is a val-multiplicity d_1 root. However, when multiple val-multiplicity d_1 roots exist, without loss of generality, given by $\mathbf{0}$ and $\mathbf{a}$ (where $a_1 \neq 0$), we can modify the proof of Lemma 8 to show that the

effective polynomial is zero modulo $\langle a_1x_2 - a_2x_1, a_1x_3 - a_3x_1\rangle^{d_1}$. The rank of the nonzero roots, given by $\langle \mathbf{x} - \mathbf{a}\rangle$, can be 1 or 2 (in general, $1, 2, \ldots, n-1$ for n-variates).

In the case of rank=2 val-mult=d_1 roots, there will be only one linear polynomial, say $f \in \langle a_1x_2 - a_2x_1\rangle^{d_1} + \langle p\rangle$. So, the equations will similar Equation 6, except that the polynomials u_j's will be in two variables, say x_2 and x_3. We will form a bivariate system, and solve it using the simultaneous bivariate root-finding as discussed above; to find all the representative-roots for x_2, x_3. This new version of Algorithm 2 will take $\mathsf{poly}((k+d+p)^d)$-time; and make the tree this wider.

In the case of rank=1 val-mult=d_1 roots, we will have a multinomial expansion having two linear forms $\{a_1x_2 - a_2x_1, a_1x_3 - a_3x_1\}$ instead of a single binomial expansion as done in Equation 8. So, now, we have two linear forms L, L' instead of a single L_1. We need to find the values of the third variable, say x_3, such that the resulting effective polynomial after lifting is again in $\langle L, L'\rangle^{d_1}$. This gives constraints similar to Equation 7. From these, we can use the univariate Algorithm 3 to solve them.

Finally, the techniques of Section 3.1 can be smoothly generalized to search for the contiguous d_1-forms in $\mathsf{poly}((k+d+p)^d)$-time. In particular, we will search them in the decreasing order of the rank of underlying val-mult=d_1 roots: rank=2, rank=1 and then rank=0 in the last.

Conclusion. We can find the roots of $f(\mathbf{x})$ mod p^k, for 3-variates where the only difference from bivariates is the handling of contiguous val-multiplicity d_1 roots (due to the more possibilities of d_1-forms). The same extension can be performed for n-variate m polynomials, for any constant n. Thus, Algorithm 1 fits well in the general framework, and finds all the roots.

Time complexity. For a 3-variate polynomial f, at each step of the tree (Fig.1), there are $O(dp^3)$ branches corresponding to val-mult$< d_1$ roots. For val-mult=d_1 roots, there are three ordered possibilities in the chain: rank=2 val-mult=d_1 root, rank=1 val-mult=d_1 roots, and rank=0 (unique) val-mult=d_1 roots. Similar to Lemma 9, we can show that rank can not increase after lifting by a val-mult=d_1 root.

For rank=2, we solve a system of bivariate equations. The number of possible branches of bivariates is $O((k^2d+p^2)^{2d})$. Furthermore, the number of possibilities of these special contiguous chains is $O(k^3)$. Therefore, the total number of leaves of the tree for 3-variates will be $O(\,((k^2d+p^2)^{2d}\cdot k^3 + p^3)^d\,)$, which is bounded by $O((k+d+p)^{(4d)^2})$.

We claim that the time complexity, for any n, is bounded by $O((k+d+p)^{(2d(n-1))^{n-1}})$. We prove this using induction by assuming this to be true for $(n-1)$-variates and derive the complexity for n-variates.

For n-variates ($n \geq 3$), the val-multiplicity d_1 roots will have ranks from $n-1$ to 0. For rank=$(n-1)$, we get the maximum upper bound. We will be solving a system of $(n-1)$-variate equations, which will lead to $O((k+d+p)^{(2d(n-2))^{n-2}})$ possibilities and representative-roots. Furthermore, there are k^{n-1} possibilities for the chain. Thus, one tree size is $s_1 := O(\,(k^{n-1}(k+d+p)^{(2d(n-2))^{n-2}} + p^n)^{2d}\,)$. But we may need to repeat this $n-1$ times on each leaf, when we have a system of n-variates to solve. Thus, the time becomes $s_1^{n-1} = O((k+d+p)^{(2d(n-1))^{n-1}})$.

Using this technique for finding all the roots modulo p^k for n-variates, we can generalize the algorithms for finding $\mathbb{Z}_p$-roots and computing the Igusa local zeta function for a system of m polynomials in n-variables as well. For finding roots over $\mathbb{Z}_p$, we consider the resultant w.r.t. one variable at a time, find a bound similar to Lemma 13, and proceed with the analysis of roots which are in $\mathbb{Z}_p$ or are not roots of the discriminant. At each step, the bounds according to one variable at a time will be obtained, which get multiplied to give a bound k_0 for $f(\mathbf{x})$ such that roots of $f(\mathbf{x})$ mod p^{k_0} gives us roots which correspond to $\mathbb{Z}_p$ roots as well. This will be a generalization of

Theorem 16 to n-variates. Similarly we can count roots, where the number of possibilities due to linear-representative roots depends on the rank of the linear forms, and the sum will again be a rational form.

The complexity of finding $\mathbb{Z}_p$ points and that of computing Igusa local zeta function will remain deterministic $\mathsf{poly}((m \log M + p + d)^{(2d(n-1))^{n-1}})$-time.

8 Conclusion and future work

Root finding of bivariates, and of n-variates for any constant n, gives rise to the following questions of finding faster algorithms, over a Galois ring.

1. We leave the question of root-finding, for constant d and n, in polylog(p)-time open. Some progress has been made in [CDS24] using deeper algebraic-geometric methods.

2. Can we use *random projections to a line*: E.g. can root-finding of $f(x_1, \ldots, x_n)$ be reduced to root-finding of $f(x_1, \mathbf{a}t + \mathbf{b})$, with high probability, for $\mathbf{a}, \mathbf{b} \in (\mathbb{Z}/p^k\mathbb{Z})^{n-1}$? This would give simpler algorithms for the general n case.

   Hilbert's irreducibility test (over a Galois field) states that a polynomial $f(x_1, \ldots, x_n)$ is factorizable iff : $f(x_1, a_1t + b_1, a_2t + b_2, \ldots, a_{n-1}t + b_{n-1})$ is factorizable and $f(x_1, b_1, \ldots, b_{n-1})$ is square free with high probability, where the probability ranges over constants $\overline{a}$ and $\overline{b}$. A similar technique might be used to show that $f(x_1, \ldots, x_n)$ has roots if the bivariate polynomial $f(x_1, a_1t + b_1, a_2t + b_2, \ldots, a_{n-1}t + b_{n-1})$ has roots over a *Galois ring*. We leave this as an open problem to analyze and improve the time complexity of Section 7.

Acknowledgements: We thank Ashish Dwivedi for useful discussions on many related problems in this area. We thank the anonymous reviewers of the conference version, whose feedback improved our presentation. N.S. thanks the funding support from DST-SERB (CRG/2020/45 + JCB/2022/57) and N. Rama Rao Chair.

References

[Ber67] Elwyn R Berlekamp. Factoring polynomials over finite fields. *Bell System Technical Journal*, 46(8):1853–1859, 1967. 2

[Ber70] Elwyn R Berlekamp. Factoring polynomials over large finite fields. *Mathematics of computation*, 24(111):713–735, 1970. 2

[BKW19] Andreas Björklund, Petteri Kaski, and Ryan Williams. Solving systems of polynomial equations over gf (2) by a parity-counting self-reduction. In *46th International Colloquium on Automata, Languages, and Programming (ICALP), 2019, Patras, Greece*, volume 132 of *LIPIcs*, pages 26:1–26:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. 4

[BLQ13] Jérémy Berthomieu, Grégoire Lecerf, and Guillaume Quintin. Polynomial root finding over local rings and application to error correcting codes. *Applicable Algebra in Engineering, Communication and Computing*, 24(6):413–443, 2013. 2, 3, 4, 5, 8, 12, 13, 15, 16, 22, 30, 31

[BM67] BJ Birch and K McCann. A criterion for the p-adic solubility of diophantine equations. *The Quarterly Journal of Mathematics*, 18(1):59–63, 1967. 4

[CDS24] Sayak Chakrabarti, Ashish Dwivedi, and Nitin Saxena. Solving polynomial systems over non-fields and applications to modular polynomial factoring. *Journal of Symbolic Computations*, 125:102314, 2024. 25

[CG00] David G. Cantor and Daniel M. Gordon. Factoring polynominals over p-adic fields. In *Algorithmic Number Theory, 4th International Symposium, ANTS-IV, Leiden, Netherlands*, volume 1838 of *Lecture Notes in Computer Science*, pages 185–208. Springer, 2000. 2

[CGRW19] Qi Cheng, Shuhong Gao, J Maurice Rojas, and Daqing Wan. Counting roots for polynomials modulo prime powers. *The Open Book Series*, 2(1):191–205, 2019. 2

[Cha22] Sayak Chakrabarti. Multivariate polynomials modulo prime powers: their roots, zeta-function and applications. Master's thesis, CSE, IIT Kanpur, India, 2022. 1

[Chi87] Alexander Leonidovich Chistov. Efficient factorization of polynomials over local fields. *Doklady Akademii Nauk*, 293(5):1073–1077, 1987. 2

[Chi21] Alexander L Chistov. An effective algorithm for deciding solvability of a system of polynomial equations over p-adic integers. *Algebra i Analiz*, 33(6):162–196, 2021. 4

[CL01] Howard Cheng and George Labahn. Computing all factorizations in $z_n[x]$. In *Proceedings of the 2001 International Symposium on Symbolic and Algebraic Computation (ISSAC), Ontario, Canada*, pages 64–71. ACM, 2001. 2

[CLO13] David Cox, John Little, and Donal O'Shea. *Ideals, varieties, and algorithms: an introduction to computational algebraic geometry and commutative algebra*. Springer Science & Business Media, 2013. 16

[CRW20] Qi Cheng, J Maurice Rojas, and Daqing Wan. Computing zeta functions of large polynomial systems over finite fields. *arXiv preprint arXiv:2007.13214*, pages 1–10, 2020. 3

[CZ81] David G Cantor and Hans Zassenhaus. A new algorithm for factoring polynomials over finite fields. *Mathematics of Computation*, 36(154):587–592, 1981. 2, 30

[Den84] Jan Denef. The rationality of the poincaré series associated to the p-adic points on a variety. *Invent. math*, 77(1):1–23, 1984. 3, 4

[DH01] Jan Denef and Kathleen Hoornaert. Newton polyhedra and igusa's local zeta function. *Journal of number Theory*, 89(1):31–64, 2001. 3

[DM97] Bruce Dearden and Jerry Metzger. Roots of polynomials modulo prime powers. *European Journal of Combinatorics*, 18(6):601–606, 1997. 2

[DMS19] Ashish Dwivedi, Rajat Mittal, and Nitin Saxena. Counting Basic-Irreducible Factors Mod p^k in Deterministic Poly-Time and p-Adic Applications. In *Proceedings of 34th Computational Complexity Conference (CCC 2019)*, pages 15:1–15:29. Springer, 2019. 2, 5, 16

[DMS21] Ashish Dwivedi, Rajat Mittal, and Nitin Saxena. Efficiently factoring polynomials modulo p^4. *Journal of Symbolic Computation*, 104:805–823, 2021. 2, 22, 30

[DS20] Ashish Dwivedi and Nitin Saxena. Computing igusa's local zeta function of univariates in deterministic polynomial-time. *Open Book Series*, 4(1):197–214, 2020. 2, 3, 4, 5, 16, 17, 18, 22

[Dwi23] Ashish Dwivedi. *Polynomials over composites: Compact root representation via ideals and algorithmic consequences*. PhD thesis, CSE, IIT Kanpur, India, 2023. 4

[EK90] Andrzej Ehrenfeucht and Marek Karpinski. *The computational complexity of (xor, and)-counting problems*. International Computer Science Inst., 1990. 3, 5

[GCM21] Aditya Gulati, Sayak Chakrabarti, and Rajat Mittal. On algorithms to find p-ordering. In *Conference on Algorithms and Discrete Applied Mathematics*, pages 333–345. Springer, 2021. 2, 30

[GGL08] Parikshit Gopalan, Venkatesan Guruswami, and Richard J Lipton. Algorithms for modular counting of roots of multivariate polynomials. *Algorithmica*, 50(4):479–496, 2008. 3, 5

[GH00] Pierrick Gaudry and Robert Harley. Counting points on hyperelliptic curves over finite fields. In *International Algorithmic Number Theory Symposium*, pages 313–332. Springer, 2000. 2

[GNP12] Jordi Guàrdia, Enric Nart, and Sebastian Pauli. Single-factor lifting and factorization of polynomials over local fields. *Journal of Symbolic Computation*, 47(11):1318–1346, 2012. 2

[Gou97] Fernando Q Gouvêa. p-adic numbers. In *p-adic Numbers*. Springer, 1997. 30

[Har15] David Harvey. Computing zeta functions of arithmetic schemes. *Proceedings of the London Mathematical Society*, 111(6):1379–1401, 2015. 3

[Hen18] Kurt Hensel. Eine neue theorie der algebraischen zahlen. *Mathematische Zeitschrift*, 2(3):433–452, 1918. 2, 7

[HW99] M-D Huang and Y-C Wong. Solvability of systems of polynomial congruences modulo a large prime. *computational complexity*, 8(3):227–257, 1999. 4

[Igu74] Jun-ichi Igusa. Complex powers and asymptotic expansions. i. functions of certain types. *Journal für die reine und angewandte Mathematik*, 0268_0269:110–130, 1974. 3, 4

[Igu77] Jun-Ichi Igusa. Some observations on higher degree characters. *American Journal of Mathematics*, 99(2):393–417, 1977. 3, 4

[Igu07] Jun-ichi Igusa. *An introduction to the theory of local zeta functions*, volume 14. American Mathematical Soc., 2007. 3

[Kal82] Erich Kaltofen. A polynomial-time reduction from bivariate to univariate integral polynomial factorization. In *23rd Annual Symposium on Foundations of Computer Science (FOCS 1982)*, pages 57–64. IEEE, 1982. 2

[Kal85] Erich Kaltofen. Polynomial-time reductions from multivariate to bi-and univariate integral polynomial factorization. *SIAM Journal on Computing*, 14(2):469–489, 1985. 2

[Kay05] Neeraj Kayal. Solvability of a system of bivariate polynomial equations over a finite field. In *International Colloquium on Automata, Languages, and Programming*, pages 551–562. Springer, 2005. 4

[Ked01] Kiran S Kedlaya. Counting points on hyperelliptic curves using monsky-washnitzer cohomology. *Journal of the Ramanujan Mathematical Society*, 16(4):323–338, 2001. 2

[Ked04] Kiran S Kedlaya. Computing zeta functions via p-adic cohomology. In *International Algorithmic Number Theory Symposium*, pages 1–17. Springer, 2004. 3

[Kob12] Neal Koblitz. *p-adic Numbers, p-adic Analysis, and Zeta-Functions*, volume 58. Springer Science & Business Media, 2012. 30

[KRRZ20] Leann Kopp, Natalie Randall, Joseph Rojas, and Yuyu Zhu. Randomized polynomial-time root counting in prime power rings. *Mathematics of Computation*, 89(321):373–385, 2020. 2

[KU11] Kiran S Kedlaya and Christopher Umans. Fast polynomial factorization and modular composition. *SIAM Journal on Computing*, 40(6):1767–1802, 2011. 2

[Lau06] Alan GB Lauder. A recursive method for computing zeta functions of varieties. *LMS Journal of Computation and Mathematics*, 9:222–269, 2006. 3

[LMMS94] Frank Lehmann, Markus Maurer, Volker Müller, and Victor Shoup. Counting the number of points on elliptic curves over finite fields of characteristic greater than three. In *International Algorithmic Number Theory Symposium*, pages 60–70. Springer, 1994. 2

[LPT+17] Daniel Lokshtanov, Ramamohan Paturi, Suguru Tamaki, Ryan Williams, and Huacheng Yu. Beating brute force for systems of polynomial equations over finite fields. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2190–2202. SIAM, 2017. 4

[Mau01] Davesh Maulik. Root sets of polynomials modulo prime powers. *Journal of Combinatorial Theory, Series A*, 93(1):125–140, 2001. 2

[MCT02] Kazuto Matsuo, Jinhui Chao, and Shigeo Tsujii. An improved baby step giant step algorithm for point counting of hyperelliptic curves over finite fields. In *International Algorithmic Number Theory Symposium*, pages 461–474. Springer, 2002. 2

[MVZ93] Alfred J Menezes, Scott A Vanstone, and Robert J Zuccherato. Counting points on elliptic curves over $\mathbb{F}_{2^m}$. *Mathematics of computation*, 60(201):407–420, 1993. 2

[NRS17] Vincent Neiger, Johan Rosenkilde, and Éric Schost. Fast computation of the roots of polynomials over the ring of power series. In *Proceedings of the 2017 ACM on International Symposium on Symbolic and Algebraic Computation*, pages 349–356, 2017. 3

[Pan95] Peter N Panayi. *Computation of Leopoldt's P-adic regulator*. PhD thesis, University of East Anglia, Norwich, England, 1995. 2, 3, 30

[Rón87] Lajos Rónyai. Factoring polynomials over finite fields. In *28th Annual Symposium on Foundations of Computer Science (FOCS 1987)*, pages 132–137. IEEE, 1987. 2

[RRZ21] Caleb Robelle, J Maurice Rojas, and Yuyu Zhu. Sub-linear point counting for variable separated curves over prime power rings. *arXiv preprint arXiv:2102.01626*, page 18, 2021. 2, 3

[Săl05] Ana Sălăgean. Factoring polynomials over z4 and over certain galois rings. *Finite fields and their applications*, 11(1):56–70, 2005. 2

[Sat02] Takakazu Satoh. On p-adic point counting algorithms for elliptic curves over finite fields. In *International Algorithmic Number Theory Symposium*, pages 43–66. Springer, 2002. 2

[Sch95] René Schoof. Counting points on elliptic curves over finite fields. *Journal de théorie des nombres de Bordeaux*, 7(1):219–254, 1995. 2

[Sir17] Carlo Sircana. Factorization of polynomials over $\mathbb{Z}/(p^n)$. In *Proceedings of the 2017 ACM on International Symposium on Symbolic and Algebraic Computation*, pages 405–412, 2017. 2

[Zas69] Hans Zassenhaus. On hensel factorization, i. *Journal of Number Theory*, 1(3):291–311, 1969. 2

[Zas78] Hans Zassenhaus. A remark on the hensel factorization method. *Mathematics of Computation*, 32(141):287–292, 1978. 2

[ZG03] WA Zuniga-Galindo. Computing igusa's local zeta functions of univariate polynomials, and linear feedback shift registers. *Journal of Integer Sequences*, 6:36, 2003. 3, 5

[Zhu20] Yuyu Zhu. *Trees, point counting beyond fields, and root separation*. PhD thesis, Texas A&M University, USA, 2020. 4

A Preliminaries

We describe some useful notation and previous works.

We use $\mathbf{x}$ to denote the tuple $(x_1, x_2, \ldots, x_n)$. Operations are similarly defined as $\mathbf{a}+\mathbf{b} := (a_1+b_1, a_2+b_2, \ldots, a_n+b_n)$, and $c\cdot\mathbf{a} := (c\cdot a_1, \ldots, c\cdot a_n)$, for a scalar c. Similarly, for $\mathbf{i} = (i_1, i_2, \ldots, i_n)$, we have $\mathbf{x}^{\mathbf{i}} = x_1^{i_1} x_2^{i_2} \ldots x_n^{i_n}$ with degree $|\mathbf{i}|$, and $\mathbf{i}! := i_1! i_2! \ldots i_n!$.

Based on the Taylor's expansion of polynomials in univariates, we define multivariate Taylor's expansion.

Definition 17 (Taylor's expansion/ series)**.** *Given a polynomial $f(\mathbf{x})$ of degree d, we can write it as (over any characteristic)*

$$f(\mathbf{a}+\mathbf{x}) \;=\; \sum_{\ell=0}^{\infty} \left(\sum_{|\mathbf{i}|=\ell} \frac{\partial_{\mathbf{x}^{\mathbf{i}}} f(\mathbf{a})}{\mathbf{i}!} \cdot \prod_{j=1}^{n} x_j^{i_j} \right), \tag{20}$$

where $\partial_{\mathbf{x}^\mathbf{i}} f := \frac{\partial^{i_1+\cdots+i_n} f}{\partial x_1^{i_1} \ldots \partial x_n^{i_n}}$ *is an* order $|\mathbf{i}|$ *partial derivative.*

For a prime p, we can write any integer a as a power series $a =: a_0 + a_1 p + a_2 p^2 + \ldots$, for $a_i \in \{0, 1, \ldots, p-1\}$. We write $\tilde{a} \in \mathbb{Z}_p$, *the ring of p-adic integers*, as a tuple $(a_0, a_1, a_2, \ldots)$. The j-th coordinate corresponds to a_j, and $\tilde{a} \bmod p^k$ is defined as the projection upto the $(k-1)$-th coordinate, i.e. $a_0 + a_1 p + \ldots a_{k-1} p^{k-1}$. Similarly, we define *the field of p-adic numbers* as the fraction field of $\mathbb{Z}_p$, denoted as $\mathbb{Q}_p$. For more literature on p-adic numbers, we direct the reader to [Gou97, Kob12].

Definition 18 (Valuation)**.** *For an integer n and a prime p, we define its* valuation *w.r.t. p, denoted $v_p(n)$, as the largest integer v such that $p^v | n$.*

Definition 19. *A representative of a ring R, denoted by the symbol $*$, takes all values in the ring R. Formally, it is the set $* := \{a | a \in R\}$.*

We further define the operations:

- $b + * = \{b + a | a \in *\}$ for $b \in R$,
- $b* = \{ba | a \in *\}$ for $b \in R$.

Using this definition, for $\beta + p^\ell * \subseteq \mathbb{Z}/p^k\mathbb{Z}$ for $\beta \in \mathbb{Z}/p^\ell\mathbb{Z}$, $\ell \leq k$, we have

$$\beta + p^\ell * = \{\beta + p^\ell a | a \in \mathbb{Z}/p^{k-\ell}\mathbb{Z}\} \tag{21}$$

In a similar fashion, a *representative root* of a polynomial $f(x) \in \mathbb{Z}/p^k\mathbb{Z}$ is denoted by a set $\beta + p^\ell *$ for $\beta \in \mathbb{Z}/p^\ell\mathbb{Z}$, $\ell \leq k$ such that for any $A \in \beta + p^\ell *$, we have $f(A) \equiv 0 \mod p^k$. The *length* of this representative root is the number of precision coordinates of the fixed part β, which is ℓ.

For more properties of representative roots, we direct the reader to [Pan95, BLQ13, DMS21, GCM21].

Solving univariates simultaneously. Using this compact notation, we present the standard Algorithm 3 to find *all* the roots of a univariate polynomial $f(x) \in \mathbb{Z}/p^k\mathbb{Z}$; which is due to [Pan95, BLQ13, DMS21]. However, as required in this paper, we give a slight modification where we solve a system of univariates modulo p^k. The algorithm starts with the input array $(f_1, \ldots, f_r, p, k, \ldots, k)$. This can be seen as a slight modification of the Root-Find algorithm ([DMS21, Algorithm 1]) where instead of looping only over the roots of the polynomial, we loop over the common roots in order to find a root of *all* the polynomials in the system.

Algorithm 3 Root finding of $f_1(x), \ldots, f_r(x) \bmod p^k$

procedure ROOT-FIND-BLQ$(f_1, \ldots, f_r, p, k_1, \ldots, k_r)$
  if $r = 0$ **then return** $*$
  if $\exists i$ such that $f_i(x) \equiv 0 \bmod p^{k_i}$ or $k_i = 0$ **then**
    return ROOT-FIND-BLQ$(f_1, \ldots, f_{i-1}, f_{i+1}, \ldots, f_r, p, k_1, \ldots, k_{i-1}, k_{i+1}, \ldots, k_r)$
  $R :=$ roots of $\gcd\{f_i(x) \bmod p \mid i \in [r]\}$ [Eg. use Cantor-Zassenhaus' algorithm [CZ81]].
  if $R == \phi$ **then return** ϕ
  $S := \phi$
  for $a \in R$ **do**

9: $\tilde{f}_i(x) := p^{v_i} f_i(a + px)\ \forall i \in [r]$, where $v_i = v_p(f_i(a + px))$.
10: $R_a := \text{Root-Find-BLQ}(f_1, \ldots, f_r, p, k - v_1, \ldots, k - v_r)$
11: $S := S \cup (a + pR_a)$
12: **return** S

The correctness of Algorithm 3 directly follows from [BLQ13, Corollary 4], where they prove the correctness for a single polynomial. [BLQ13, Corollary 4] also states that the number of representative roots is at most d many when only a single polynomial is considered.

Theorem 20. *Algorithm 3 runs in randomized* $\mathsf{poly}(\max_i \deg(f_i), \log p, k)$ *time and returns at most d-many representative roots.*