

July 2014

On The Applications of Lifting Techniques

Esmail Mehrabi

The University of Western Ontario

Supervisor

Dr. Eric Schost


The University of Western Ontario

Graduate Program in Computer Science

A thesis submitted in partial fulfillment of the requirements for the degree in Doctor of Philosophy

© Esmail Mehrabi 2014

Follow this and additional works at: <https://ir.lib.uwo.ca/etd>

 Part of the [Numerical Analysis and Scientific Computing Commons](#), and the [Theory and Algorithms Commons](#)

Recommended Citation

Mehrabi, Esmail, "On The Applications of Lifting Techniques" (2014). *Electronic Thesis and Dissertation Repository*. 2126.
<https://ir.lib.uwo.ca/etd/2126>

This Dissertation/Thesis is brought to you for free and open access by Scholarship@Western. It has been accepted for inclusion in Electronic Thesis and Dissertation Repository by an authorized administrator of Scholarship@Western. For more information, please contact tadam@uwo.ca.

ON THE APPLICATIONS OF LIFTING TECHNIQUES
(Thesis format: Integrated Article)

by

Esmail Mehrabi

Graduate Program in Computer Science

A thesis submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy

The School of Graduate and Postdoctoral Studies
The University of Western Ontario
London, Ontario, Canada

© Esmail Mehrabi 2014

Abstract

Lifting techniques are some of the main tools in solving a variety of different computational problems related to the field of computer algebra. In this thesis, we will consider two fundamental problems in the fields of computational algebraic geometry and number theory, trying to find more efficient algorithms to solve such problems.

The first problem, *solving systems of polynomial equations*, is one of the most fundamental problems in the field of computational algebraic geometry. In this thesis, we discuss how to solve bivariate polynomial systems over either $k(T)$ or \mathbb{Q} using a combination of lifting and modular composition techniques. We will show that one can find an equiprojectable decomposition of a bivariate polynomial system in a better time complexity than the best known algorithms in the field, both in theory and practice.

The second problem, *polynomial factorization over number fields*, is one of the oldest problems in number theory. It has lots of applications in many other related problems and there have been lots of attempts to solve the problem efficiently, at least, in practice. Finding p -adic factors of a univariate polynomial over a number field uses lifting techniques. Improving this step can reduce the total running time of the factorization in practice. We first introduce a multivariate version of the Belabas factorization algorithm over number fields. Then we will compare the running time complexity of the factorization problem using two different representations of a number field, univariate vs multivariate, and at the end as an application, we will show the improvement gained in computing the splitting fields of a univariate polynomial over a rational field.

Keywords. Polynomial system solving, Polynomial factorization, Computer Algebra

Acknowledgments

First and foremost I would like to offer my sincerest gratitude to my supervisor, Dr. Éric Schost, who has supported me throughout my thesis with his patience and knowledge. I attribute the level of my Ph.D. degree to his encouragement and effort, and without him, this thesis would not have been completed or written.

Secondly, all my sincere thanks and appreciation go to all the members from our Ontario Research Centre for Computer Algebra (ORCCA) lab in the Department of Computer Science for their invaluable support and assistance, and all the members of my thesis examination committee.

Finally, I would like to thank all of my friends and family members for their consistent encouragement and continued support.

Contents

Abstract	iii
Acknowledgements	iv
List of Algorithms	vi
List of Tables	vii
1 Introduction	1
1.1 Solving bivariate polynomial systems	1
1.1.1 Preliminaries	2
1.1.2 Problem statement and related works	5
1.2 Univariate polynomial factorization	10
2 Solving bivariate polynomial system: the case of non-singular solutions	13
2.1 Introduction and main results	13
2.2 Preliminaries	17
2.2.1 Notation and basic results	17
2.2.2 Chinese Remainder techniques	18
2.2.3 Specialization properties	19
2.3 A direct algorithm	20
2.4 Normal form algorithms	22
2.4.1 Reduction modulo one triangular set	23
2.4.2 Reduction modulo several triangular sets	25
2.5 Proof of the main results	27
2.5.1 One lifting step	27
2.5.2 Main algorithm	29
2.6 Experimental results	30
3 Solving bivariate polynomial systems: the case of singular solutions	33

3.1	Introduction	33
3.2	Quantitative estimates	39
3.2.1	Polynomials in general position	39
3.2.2	Non-vanishing conditions	46
3.2.3	Conservation of intersection multiplicity	47
3.3	Finding zeros in a list	51
3.4	Normal forms for derivatives	53
3.4.1	Auxiliary results	55
3.4.2	Proof of Proposition 3	59
3.5	The deflation lemma	61
3.6	The σ -decomposition	65
3.6.1	Computing all m_i 's and H_i 's	67
3.6.2	Computing all J_i 's	70
3.6.3	Computing all n_i 's, a_i 's and K_i 's	73
3.7	Newton iteration	78
3.8	Main algorithm	82
3.8.1	Choosing parameters	82
3.8.2	Computations modulo p	83
3.8.3	Analysis of one lifting step	84
3.8.4	Total cost	88
4	Univariate Polynomial Factorization	89
4.1	Introduction	89
4.2	Factoring polynomials over \mathbb{Z} and \mathbb{Q}	90
4.2.1	Some basic considerations	90
4.2.2	The approach of van Hoeij	92
4.3	Factoring polynomials over number fields	110
4.3.1	Factor construction	112
4.3.2	Factor Combination	119
4.3.3	Bound on the Traces	127
4.3.4	Choosing a Denominator	129
4.3.5	Splitting Fields	134

List of Algorithms

1	Main Algorithm	29
2	zero_index(P, r)	52
3	zero_index_vectorial(P, R)	53
4	m_H_rational(F, G, x, y)	67
5	compute_m_H(F, G, P, S)	68
6	compute_J($F, G, [(C_i, T_i, m_i, H_i, n_i)]_{1 \leq i \leq s}$)	71
7	compute_n_a_K ($F, G, [(P_i, S_i, H_i, m_i)]_{1 \leq i \leq s}$)	74
8	lift_y($F, G, \Sigma \bmod N$)	86
9	lift_x_y(N, F, G, Σ_N)	87

List of Tables

2.1	Lifting vs CRT	31
4.1	Splitting Fields Computation	139

Chapter 1

Introduction

The following two problems are the main primary objectives of this research work:

- Solving bivariate polynomial systems over \mathbb{Q} or $k[t]$, where k is a field
- Univariate polynomial factorization over number fields

Both of these problems are the most fundamental and challenging subjects in the field of computer algebra, as they play an important role in many higher-level algorithms. In the following, we describe both problems in detail by providing the required background, related work done in the area, and the main contributions of our research.

1.1 Solving bivariate polynomial systems

Solving systems of polynomial equations is one of the most fundamental and most studied subjects in mathematical sciences. If the case of linear systems is theoretically well understood, the case of non-linear systems is, by essence, richer and more complex. Thus, research in this area covers theory, algorithms, implementation techniques and applications.

In this research work, we investigate the complexity of solving bivariate polynomial systems. This question is interesting in its own right, but it also plays an important role in many higher-level algorithms, such as computing the topology of plane and space curves [26, 19] or solving general polynomial systems [33].

Many recent contributions on this question discuss computing real solutions of bivariate systems with integer or rational coefficients [28, 25, 57, 11, 27], by a combination of symbolic elimination and real root isolation techniques. Our interest here is on the complexity of the symbolic component of such algorithms.

By solving a bivariate polynomial system, we mainly refer to a triangular representations of the solutions, obtained by a triangular decomposition.

Triangular decompositions are one of the main tools for solving polynomial systems. For systems of algebraic equations, they provide a convenient way to describe the complex solutions and a step toward decompositions into irreducible components. Combined with other techniques, they are used for these purposes in many computer algebra systems such as Maple or Singular.

1.1.1 Preliminaries

In this section we will talk about the two main ingredients of the following sections. The first one explains what exactly we mean by solving a bivariate system, which is called the equiprojectable decomposition of the given system. The second one introduces a kind of measurement to estimate the size of the problem, addressing the optimality issues.

Definition 1. *Let $F, G \in \mathbb{L}[X, Y]$ generates a zero dimensional ideal, where \mathbb{L} is a domain. A triangular decomposition of the ideal $\langle F, G \rangle$ over \mathbb{K} , the fraction field of \mathbb{L} , is defined as follows:*

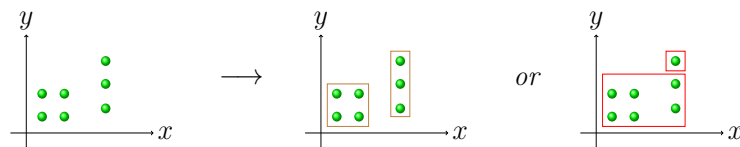
$$\sqrt{\langle F, G \rangle} = \langle U_1(X), V_1(X, Y) \rangle \cap \cdots \cap \langle U_s(X), V_s(X, Y) \rangle$$

where $\sqrt{\langle F, G \rangle}$ is the radical ideal of $\langle F, G \rangle$, and

- $U_i \in \mathbb{K}[X]$ is monic
- $V_i \in \mathbb{K}[X, Y]$ is monic in Y and reduced modulo U_i

Triangular decomposition of a given polynomial system is one way, among others, of representing the solutions of the system. But, these solutions can be represented using different triangular sets, which indeed, puts a question to the uniqueness of such a decomposition.

Example 1. *The following picture shows two different triangular decompositions of an ideal, from a geometrical point of view.*



We are not going to talk about the technical issues of the above triangular decomposition here, as we will do so in the coming chapters. But the aim of the example is to show how such a decomposition works geometrically. Indeed, we are going to pack the roots of the given system of bivariate polynomials in such a way that in each package there exists the same number of points over any x -coordinate. This sort of packaging the roots seems interesting since we can represent the given polynomial system using two new polynomials $U(x), V(x, y)$ where U is a univariate polynomial only in the variable x , and we can then extract useful information about, for instance, the properties of its roots and the dimension of the given ideal.

The other important point that we want to make in this example is that such a triangular decomposition is not necessarily unique. As we can see from the picture, the variety of the given system can be packed into two different triangular sets, one including 4 and 3 points, the other 6 and 1 points, while both satisfying the mentioned properties of triangular decomposition. This non-uniqueness property creates some difficulties when we are using a lifting technique to find such a decomposition starting from its image modulo some prime ideal in the base ring. We will talk about this problem and the proposed technique to resolve such a problem in the following. But it is worth to mentioning that the problem would be resolved using a restricted version of triangular decomposition, called equiprojectable decomposition.

One way to guarantee the uniqueness property of such a decomposition is using a canonical decomposition of a 0-dimensional ideal, which is called the equiprojectable decomposition, defined as follows.

Using the same notation as in Definition 1, let $Z \subset \bar{\mathbb{K}}^2$ be the variety of the ideal generated by F, G , where $\bar{\mathbb{K}}$ is the algebraic closure of \mathbb{K} . Let $\pi : \bar{\mathbb{K}}^2 \rightarrow \bar{\mathbb{K}}$ be the projection on the X -space given by $(x, y) \rightarrow x$. To $p = (x, y)$ in Z , we associate the positive integer $N(Z, p)$ defined as the cardinality of the fibre $\pi^{-1}(x) \cap Z$: this is the number of points in Z lying above x . We say that Z is equiprojectable if there exists a positive integer n such that $N(Z, p) = n$ for all $p \in Z$.

It is proved in [7] that Z is equiprojectable if and only if its defining ideal admits a Gröbner basis for the lexicographic order $Y > X$ that is a monic triangular set, i.e. of the form $T = (U(X), V(X, Y))$, with U and V monic in respectively X and Y and with coefficients in K .

When Z is not equiprojectable, it can be decomposed into equiprojectable sets, usually in a non-unique manner. The equiprojectable decomposition [23] is a canonical way to do so: it decomposes Z into subsets Z_{n_1}, \dots, Z_{n_s} , where for all $i \in \{1, \dots, s\}$, Z_{n_i} is the set of all $p \in Z$ for which $N(Z, p) = n_i$.

If Z is defined over \mathbb{K} , then all Z_{n_i} are defined over \mathbb{K} as well, so they can be represented by monic triangular sets

$$T_1 \left| \begin{array}{c} V_1(X, Y) \\ U_1(X) \end{array} \right. \quad \dots \quad T_s \left| \begin{array}{c} V_s(X, Y) \\ U_s(X) \end{array} \right.$$

with coefficients in \mathbb{K} . If we let $m_i = |\pi(Z_{n_i})|$, then T_i has bidegree (m_i, n_i) for all i , and $\sum_{i \leq s} m_i n_i = \delta$, where δ is the cardinal of Z .

By abuse of notation, we will call the family of monic triangular sets $T = (T_1, \dots, T_s)$ the equiprojectable decomposition of Z . Since $\sqrt{\langle F, G \rangle}$ is a radical ideal of $\mathbb{K}[X, Y]$ that remains radical in $\overline{\mathbb{K}}[X, Y]$, its zero-set Z is defined over \mathbb{K} ; then, we define the equiprojectable decomposition of $\langle F, G \rangle$ as that of Z .

Example 2. *In Example 1, the first decomposition is equiprojectable, while the other one is not. Since for the second one, by projecting the points over their x -coordinates, you see that there are 3 points having the third x -coordinate as the x -coordinate, but the other two has only two points above their x -coordinate.*

So from now on, by solving a bivariate system, we mean computing the equiprojectable decomposition of the given system.

In the complexity analysis, we are mainly interested in the following two domains associated with the defined length functions:

- $\mathbb{L} = k[T]$ and $\mathbb{K} = k(T)$, for a field k , where we use the length function $\lambda(a) = \deg(a)$, for $a \in \mathbb{L} - \{0\}$;
- $\mathbb{L} = \mathbb{Z}$ and $\mathbb{K} = \mathbb{Q}$, where we use the length function $\lambda(a) = \log(|a|)$, for $a \in \mathbb{L} - \{0\}$.

In both cases, the length of $a \in \mathbb{L}$ represents the amount of storage needed to represent it, in terms of elements of k , resp. bits. It will be useful to introduce a notion of

length for polynomials with coefficients in \mathbb{K} : if P is such a polynomial, $\lambda(P)$ denotes the maximum of the lengths $\lambda(n_i)$ and $\lambda(d_i)$, where n_i and d_i are the numerators and denominators of the coefficients of P , when written in reduced form using a common denominator. Note that when $\mathbb{L} = k[T]$, we are studying the intersection of two surfaces in a 3-dimensional space with coordinates T, X, Y ; the output describes the solution curve for generic T .

1.1.2 Problem statement and related works

Problem 1. *Let $F, G \in \mathbb{L}[X, Y]$ of degree at most d in each variable, generating a zero dimensional ideal over \mathbb{K} , the fraction field of \mathbb{L} . What we aim to do is to compute the equiprojectable decomposition of the ideal $\langle F, G \rangle$ over \mathbb{K} , that is, finding all monic triangular sets $\langle U_i, V_i \rangle$ such that*

$$\sqrt{\langle F, G \rangle} = \langle U_1(X), V_1(X, Y) \rangle \cap \cdots \cap \langle U_s(X), V_s(X, Y) \rangle$$

There has been lots of work to solve the problem as efficiently as possible. The most recent ones are mainly concerned with finding the real solutions of the given bivariate system [57, 11, 27]. These algorithms usually follow two main steps, called *symbolic elimination* and *real root isolation*.

The first step is the main objective of this research. Symbolic elimination tries to find the required decomposition by first computing the resultant of F, G in Y and then applying some techniques such as regular gcd [48] to compute the actual U_i and V_i 's. In terms of complexity, both steps have the same cost, up to logarithmic factors.

Using Chinese Remainder Theorem (CRT) or Reischert's method [55] for finding the resultant of two multivariate polynomials, the resultant of two bivariate polynomials F, G in Y can be computed in $\tilde{O}(d^3)$ operations in \mathbb{L} . It is still an open problem to compute this resultant in $O(d^2)$ operations in \mathbb{L} , since the whole problem, the inputs and the output, can be stored using $O(d^2)$ elements of \mathbb{L} . Assuming $\mathbb{L} = k[T]$ (resp. $\mathbb{L} = \mathbb{Z}$), and letting $\ell = \max(\lambda(F), \lambda(G)) \leq d$, the total complexity of the best known algorithm for the symbolic elimination part would be of order $\tilde{O}(d^5)$ operations in k (resp. bit operations).

What we aim to do in this work, is to solve Problem 1 in a more efficient way, both

in theory and practice, using lifting techniques [33, 59], which will be discussed through the next chapters. To this aim, we will consider Problem 1 in the following two cases:

- (a) the case of non-singular solutions
- (b) in general case, considering singular solutions

The proposed idea for solving Problem 1, is to use lifting techniques [33, 59] to lift the triangular sets to an appropriate precision, starting from the image of them modulo some (lucky) maximal ideal of \mathbb{L} .

But the required assumption enabling lifting possible, is the invertibility of the Jacobian matrix of F, G modulo the triangular sets. The invertibility assumption is itself equivalent to non-singularity of the given system. So as long as there is no singular point, case (a), in the variety of $\langle F, G \rangle$, we have shown in [39] that Problem 1 can be solved by applying lifting technique and using a fast (collective) modular composition in a more efficient way, both theoretically and practically. The main results of [39] is stated in the following two theorems, for the cases of $\mathbb{L} = k[T]$, and $\mathbb{L} = \mathbb{Z}$, respectively.

In what follows, we let $M : \mathbb{N} \rightarrow \mathbb{N}$ be such that over any ring, univariate polynomials of degree less than d can be multiplied in $M(d)$ ring operations, under the super-linearity conditions of [66, Ch. 8]: using FFT techniques, we can take $M(d) \in O(d \log(d) \log \log(d))$. We also let ω be such that we can multiply $n \times n$ matrices using $O(n^\omega)$ ring operations, over any ring. The best known bound is $\omega < 2.38$ [65].

Theorem 1. *Let k be a field and let F, G be in $k[T][X, Y]$, with $d = \max(\deg(F), \deg(G))$ and $\ell = \max(\lambda(F), \lambda(G))$. If k has characteristic at least $4d^2(6d^2 + 9d\ell)$, one can compute the modified equiprojectable decomposition [see 2.1 for definition] of $\langle F, G \rangle$ over $k(T)[X, Y]$ by a probabilistic algorithm with probability of success at least $1/2$, using*

$$O(M(d^2)M(d\ell + d^2)d^{(\omega-1)/2} \log(d\ell)) \subset O^{\sim}(d^{3.69}\ell + d^{4.69})$$

operations in k .

Theorem 2. *Let $\varepsilon > 0$, let F, G be in $\mathbb{Z}[X, Y]$, $d = \max(\deg(F), \deg(G))$, and $\ell = \max(\lambda(F), \lambda(G))$. One can compute the modified equiprojectable decomposition of $\langle F, G \rangle$ over $\mathbb{Q}[X, Y]$ by a probabilistic algorithm with probability of success at least $1/2$, using $O(d^{3+\varepsilon}\ell + d^{4+\varepsilon})$ bit operations.*

Note that if we let $\ell \leq d$, then the complexity of Theorem 1 (resp. Theorem 2) becomes $\tilde{O}(d^{4.69})$ (resp. $O(d^{4+\varepsilon})$) which is less than $\tilde{O}(d^5)$ for the best known algorithm. In addition, for the case of $\mathbb{L} = \mathbb{Z}$, we almost reached the optimal running time.

But if we relax the non-singularity assumption, case (b), the situation would become worse and more challenging, as we can not directly apply the lifting techniques introduced in [59] anymore. So we need a new approach of lifting to handle the singularity issues.

Lifting technique, which is basically a Newton's iterator, is one of the most popular component of polynomial system solvers, either from numeric or symbolic point of view. This iterator usually handles smooth situations only, since in the case, the Jacobian matrix of the given system is invertible.

Generalizing Newton iterator to the singular situations and the question of how to design an efficient iterator with quadratic convergence in degenerate cases is still an ongoing challenging problem. There have been lots of attempts to find such an iterator. In order to be develop Newton-type methods that converge to multiple roots, deflation techniques which consist in adding new equations in order to reduce the multiplicity have already been considered. For an overview of the work done in the area, we refer the reader to Chapter 3.

Our approach to solve Problem 1 in the case (b), is based on the Lecerf's idea introduced in [40], since we are interested in exact computation and p-adic lifting application which is a continuation of our previous work done in [39], while most of the recent work done in the area [61, 2, 1, 8, 45] are considering the problem from numerical point of view which are not of interest in our work here.

In [40], Lecerf proposed a new iterator generalizing the regular Newton iterator for multiple roots which compared to a smooth case, he proved quadratic convergence with a small overhead that grows with the square of the square of the multiplicity of the root.

Indeed, in [40], Lecerf proved that we can resolve Problem 1 in general case, not only for bivariate systems which are our primary interest here, but for multivariate polynomial systems. The main idea behind the Lecerf's approach is to replace the given multivariate polynomial system ψ with a new one, say $\tilde{\psi}$, such that for the given root x^* of ψ with

multiplicity M , x^* is still a root of the new system $\tilde{\psi}$, but with multiplicity $\tilde{M} < M$. The ideal generated by the new polynomial system $\tilde{\psi}$ is usually called *deflated* ideal. Granting this fact, we can find a new deflated system $\tilde{\psi}$ in which x^* is a non-singular root, by just repeating the same process several times.

In [40], Lecerf showed that the complexity of his algorithm grows with the square of the multiplicity M of the root, compared to the smooth case, that is, its complexity can be M^2 times the multiplicity of Newton iteration for non-singular roots, which does not seem satisfying. Note that M can be as large as the degree of the whole system, where the system ψ has only one root with multiplicity greater than one.

But we are actually interested in solving Problem 1 in bivariate case using the same amount of complexity as in the non-singular situation, aiming to gain the same improvement compared to the best known methods in the field. Following the same approach as in [39] for packaging the triangular sets together in a way that we can apply lifting technique on the whole system efficiently, which is independent of the choice of newton iterator, the only remaining problem for extending the same approach introduced in chapter one to the case of singular solutions is to generalize the Newton iterator with not only quadratic convergence, as Lecerf did, but with the same complexity as in the smooth case up to some poly-logarithmic factors.

In Chapter 3, we slightly change the definition of what we mean by solving a bivariate polynomial system. As we know, several approaches exist to describe the solutions of our system: Gröbner bases, triangular representations, or descriptions based on univariate polynomials. For instance, in [39], the authors relied on a canonical description of a zero-dimensional variety, called the equiprojectable decomposition from [23]. Although it would be natural to use this kind of description here as well, the techniques we rely on are slightly easier to apply when working in generic coordinates. Indeed, if we are in generic coordinates, the zeros of $F = G = 0$ can simply be described, called *rational univariate representation* of the ideal $\langle F, G \rangle$, by a pair of polynomials in $\mathbb{K}[X]$, of the form

$$P(X) = 0, \quad Y = S(X). \quad (1.1)$$

Let us temporarily let Z' be the set of all non-singular solutions of the polynomial system F, G . In [39], we gave with Lebreton an algorithm to compute a triangular representation of Z' ; this algorithm could be adapted to give a rational univariate representation, after putting the equations in generic coordinates.

In a nutshell, the main idea is to compute the output modulo a prime p , then *lift* this representation modulo powers of p using a suitable form of Newton iteration. Looking only at points in Z' makes it straightforward to apply such techniques, since by assumption, at such points, the Jacobian matrix of (F, G) is invertible.

In Chapter 3, we show that we can extend these ideas to find a univariate representation of $V(F, G)$, in a time close to optimal; this matches the results of [39] in the case where all solutions of $F = G = 0$ are simple. The algorithm uses a combination of the lifting techniques introduced by Lecerf in [40] and a modification of the Kedlaya-Umans modular composition algorithm [36], following the same strategy as in [39].

Our algorithm is probabilistic of the Monte Carlo kind: one can choose an arbitrary error threshold, say $1/2^P$, and the algorithm guarantees that the result is correct with probability at least $1 - 1/2^P$. Part of the randomness simply amounts to choosing an integer in a finite set. Another component is more involved, as it amounts to choosing primes. Since this is a delicate question in itself, and not the topic of this paper, we will use the following device: we assume that we are given an oracle \mathcal{O} , which takes as input an integer B , and returns a prime number in $\{B + 1, \dots, 2B\}$, uniformly distributed within this set of primes.

Theorem 3. *Let $F, G \in \mathbb{Z}[X, Y]$ with degree at most d and height at most h , that have no nontrivial factor in $\mathbb{Q}[X, Y]$.*

For any $\varepsilon > 0$, there exists an algorithm with the following characteristics. Given $P \geq 1$, the algorithm computes the Rational Univariate Representation of the system $F_t = G_t = 0$, where t is an integer of height $O(P + \log(d))$, $F_t = F(X + tY, Y)$ and $G_t = G(X + tY, Y)$. The running time is $d^{3+\varepsilon}O^\sim((h + d)P)$ bit operations, and the probability of success is at least $1 - 1/2^P$.

For fixed P , this is thus $d^{3+\varepsilon}O^\sim(h + d)$ bit operations, which almost matches the known upper bounds on the output size. We are not aware of previous result that would be comparable in terms of complexity. As reviewed in [39], previous approaches to this question had cost at least $O^\sim(d^4h + d^5)$, for Monte Carlo algorithms. For Las Vegas algorithms, the known bounds are higher yet, namely $O^\sim(d^5h + d^6)$, see [15] for latest results in this context.

For the case where our base ring is $\mathbb{A} = k[T]$ instead of \mathbb{Z} , we were not able to obtain the same result as in [39, Theorem 1], as we were not able to find a fast algorithm for one specific question of modular composition appearing in the process of evaluating the

derivatives of the given system. However, the approach given for the case $\mathbb{A} = \mathbb{Z}$ remains correct in the case of $\mathbb{A} = k[T]$ as our base ring.

1.2 Univariate polynomial factorization

The other problem that we are concerned with in this work, is the factorization of univariate polynomials over number fields.

Until 2000, the two main algorithms able to factor a polynomial P over $\mathbb{Q}[X]$ were the Berlekamp-Zassenhaus algorithm [12, 69], which starts by factoring h over $\mathbb{Q}_p[X]$ for a suitable prime number p and tries to recombine the p -adic factors, and the Lenstra-Lenstra-Lov'sz algorithm, based on their celebrated LLL lattice reduction algorithm [42]. While the latter is polynomial-time and the former exponential in the worst case, the former performs far better on average, in practice.

In 2002, van Hoeij [63] published an algorithm which, while following the Berlekamp-Zassenhaus argument, uses a new approach for lattice basis reduction to guess the correct recombination. In [10], it was shown that van Hoeij algorithm is polynomial-time, and recently in [64], the authors tried to fill the gap between theoretical and practical complexity, by expressing the complexity with respect to the number of LLL switches occurring in the LLL-reduction steps.

There are two main approaches to factor a univariate polynomial over a number field. The first approach which is based on [62], says that factoring over $F(\alpha)$ is doable if we can factor over F , where F is a field and α is in the algebraic closure of F . Since we know how to factor over \mathbb{Q} [69, 42, 63], so we can do it over any number fields, using [62].

The other approach is to apply a similar method as we do over \mathbb{Q} , which is done by Lenstra in 1982 [41]. Then Belabas in [9] combined a modification of the Lenstra method and van Hoeij's factors recombination approach to introduce a polynomial-time algorithm for factoring univariate polynomials over number fields. These methods are based on two main steps, *modular factorization* and *factors combination*. The first step is just applying the Hensel lifting on the factors of the given polynomial, say $h \in \mathbb{Q}(\alpha)[x]$, modulo some (lucky) prime ideal, say P , up to some precision. The other step deals with the problem of finding the true factors of h over $\mathbb{Q}(\alpha)$ from the P -adic factors. Now our main concern is to solve the factorization problem, following the second approach, in a

more efficient way. To make the problem more clear, we state it as below.

Problem 2. Let $h \in \mathbb{K}[x]$ of degree n , where \mathbb{K} is a number field, and $[\mathbb{K} : \mathbb{Q}] = N$. Find the irreducible polynomials h_1, \dots, h_r in $\mathbb{K}[x]$ and β_1, \dots, β_r in \mathbb{N} , such that $h = h_1^{\beta_1} \dots h_r^{\beta_r}$.

As we know, a number field \mathbb{K} is an algebraic extension of \mathbb{Q} , which can be represented in many different ways. Two of the most common representations of \mathbb{K} are:

- **Univariate representation**

\mathbb{K} is represented as the quotient $\mathbb{Q}[t]/P(t)$, where P is an irreducible univariate polynomial of degree $[\mathbb{K} : \mathbb{Q}] = N$. Indeed, \mathbb{K} includes all polynomials over \mathbb{Q} with degree less than N , and all computations in \mathbb{K} are done modulo the polynomial P .

- **Multivariate representation**

\mathbb{K} is represented as the quotient $\mathbb{Q}[t_1, \dots, t_q]/\langle H_1, \dots, H_q \rangle$, where H_1, \dots, H_q is a monic triangular set; for $1 \leq i \leq q$, H_i is a polynomial in $\mathbb{Q}[t_1, \dots, t_i]$ with degree h_i in t_i and reduced modulo H_1, \dots, H_{i-1} , and irreducible on $\frac{\mathbb{Q}[t_1, \dots, t_{i-1}]}{\langle H_1, \dots, H_{i-1} \rangle}$, and the ideal $\langle H_1, \dots, H_q \rangle$ is a radical ideal. Note that $[\mathbb{K} : \mathbb{Q}] = N = \prod_{i=1}^q h_i$.

All of the given references above are dealing with the univariate representation of \mathbb{K} , which is more commonly used in Number theory. Of course, the choice of different representations for a given number field can potentially effect the cost of the factorization algorithm, at least in practice. So now the question that we are going to consider in this work is how choosing the above representations can effect the cost of the polynomial factorization in practice.

To this aim, we first need to present a factorization algorithm which uses the multivariate representation of the number field \mathbb{K} . Our approach for factoring a univariate polynomial over a number field is inspired by Belabas [9]. Following the same approach as in [9], as we already said, we need to modify the main two steps of the algorithm, *modular factorization* and *factor combination*.

The multivariate modular factorization phase, which is basically a multivariate lifting, can be done using [59]. For the factor recombination phase, which uses the LLL approach, we made a lattice using the modular images of the polynomials H_1, \dots, H_q and the obtained modular factors. All required theoretical support for the new lattice and the correctness of the whole factorization algorithm in the new representation, have been done in Chapter 4. At the end of Chapter 4, we examine our approach against the

previous ones for one of the most important applications of polynomial factorization, that is computing *splitting fields*. We will show that computing splitting fields of a given polynomial over \mathbb{Q} can be done more efficiently if we use multivariate representation for the underlying number fields in the main core of this computation, which is the factorization algorithm.

Chapter 2

Solving bivariate polynomial system: the case of non-singular solutions

2.1 Introduction and main results

We investigate the complexity of solving bivariate polynomial systems. This question is interesting in its own right, but it also plays an important role in many higher-level algorithms, such as computing the topology of plane and space curves [26, 19] or solving general polynomial systems [33].

Many recent contributions on this question discuss computing real solutions of bivariate systems with integer or rational coefficients [28, 25, 57, 11, 27], by a combination of symbolic elimination and real root isolation techniques. Our interest here is on complexity of the “symbolic” component of such algorithms. One of our main results says that we can solve bivariate systems with integer coefficients in essentially optimal time, at least for non-singular solutions.

Geometric description. Let \mathbb{A} be a domain, let \mathbb{K} be its field of fractions and let $\overline{\mathbb{K}}$ be an algebraic closure of \mathbb{K} .

Let X, Y be the coordinates and let $Z \subset \overline{\mathbb{K}}^2$ be a finite set defined over \mathbb{K} and of cardinality δ (so the defining ideal $I \subset \overline{\mathbb{K}}[X, Y]$ of Z is generated by polynomials in $\mathbb{K}[X, Y]$). To describe Z , one may use a Gröbner basis of I , say for the lexicographic order $Y > X$. Such bases can however be unwieldy (they may involve a large number of polynomials, making modular computations difficult). Triangular decompositions are an alternative for which this issue is alleviated.

Geometrically, performing a triangular decomposition of the defining ideal of Z amounts to writing Z as the disjoint union of finitely many *equiprojectable sets*. Let $\pi : \overline{\mathbb{K}}^2 \rightarrow \overline{\mathbb{K}}$

be the projection on the X -space given by $(x, y) \mapsto x$. To $p = (x, y)$ in Z , we associate the positive integer $N(Z, p)$ defined as the cardinality of the fiber $\pi^{-1}(x) \cap Z$: this is the number of points in Z lying above x . We say that Z is *equiprojectable* if there exists a positive integer n such that $N(Z, p) = n$ for all $p \in Z$ (see [23] for illustrations).

It is proved in [7] that Z is equiprojectable if and only if its defining ideal I admits a Gröbner basis for the lexicographic order $Y > X$ that is a *monic triangular set*, i.e. of the form $\mathbf{T} = (U(X), V(X, Y))$, with U and V monic in respectively X and Y and with coefficients in \mathbb{K} (that result holds over a perfect field, so it applies over $\overline{\mathbb{K}}$; the fact that I has generators in $\mathbb{K}[X, Y]$ implies that \mathbf{T} has coefficients in \mathbb{K}). The degree $m = \deg(U, X)$ is the cardinality of $\pi(Z)$, and the equalities $n = \deg(V, Y)$ and $\delta = mn$ hold; we will say that \mathbf{T} has *bidegree* (m, n) .

When Z is not equiprojectable, it can be decomposed into equiprojectable sets, usually in a non-unique manner. The *equiprojectable decomposition* [23] is a canonical way to do so: it decomposes Z into subsets Z_{n_1}, \dots, Z_{n_s} , where for all $i \in \{1, \dots, s\}$, Z_{n_i} is the set of all $p \in Z$ for which $N(Z, p) = n_i$. This decomposition is implicit in the Cerlienco-Murredu description of the lexicographic Gröbner basis of the defining ideal of Z [18]; it can also be derived from Lazard's structure theorem for bivariate Gröbner bases [38].

If Z is defined over \mathbb{K} , then all Z_{n_i} are defined over \mathbb{K} as well, so they can be represented by monic triangular sets

$$\mathbf{T}_1 \left| \begin{array}{l} V_1(X, Y) \\ U_1(X) \end{array} \right. \quad \dots \quad \mathbf{T}_s \left| \begin{array}{l} V_s(X, Y) \\ U_s(X) \end{array} \right. \quad (2.1)$$

with coefficients in \mathbb{K} . If we let $m_i = |\pi(Z_{n_i})|$, then \mathbf{T}_i has bidegree (m_i, n_i) for all i , and $\sum_{i \leq s} m_i n_i = \delta$.

By abuse of notation, we will call the family of monic triangular sets $\mathcal{T} = (\mathbf{T}_1, \dots, \mathbf{T}_s)$ the *equiprojectable decomposition* of Z . If I is a radical ideal of $\mathbb{K}[X, Y]$ that remains radical in $\overline{\mathbb{K}}[X, Y]$, its zero-set Z is defined over \mathbb{K} ; then, we define the equiprojectable decomposition of I as that of Z .

Solving systems. Let now F and G be in $\mathbb{A}[X, Y]$. In this paper, we are interested in the set $Z(F, G)$ of *non-singular solutions* of the system $F = G = 0$, that is, the points (x, y) in $\overline{\mathbb{K}}^2$ such that $F(x, y) = G(x, y) = 0$ and $J(x, y) \neq 0$, where J is the Jacobian determinant of (F, G) . Remark that $Z(F, G)$ is a finite set, defined over \mathbb{K} ; if F and G have total degree at most d , then $Z(F, G)$ has cardinality $\delta \leq d^2$.

For instance, for generic F and G , $Z(F, G)$ coincides with their whole zero-set $V(F, G)$, it is equiprojectable ($s = 1$), the corresponding triangular set $\mathbf{T} = \mathbf{T}_1$ takes the form

$\mathbf{T} = (U(X), Y - \eta(X))$ and U is (up to a constant in \mathbb{K}) the resultant of F and G in Y .

Given F and G , our goal will be, up to a minor adjustment, to compute the triangular sets $\mathcal{T} = (\mathbf{T}_1, \dots, \mathbf{T}_s)$ that define the equiprojectable decomposition of $Z(F, G)$.

Representing these polynomials requires $O(d^2)$ elements of \mathbb{K} . We will show below that one can compute them using $O(d^3)$ operations in \mathbb{K} , where $O(\cdot)$ indicates the omission of logarithmic factors. It is a major open problem to compute \mathcal{T} in time $O(d^2)$, just like it is an open problem to compute the resultant of F and G in such a cost [30, Problem 11.10].

Size considerations. In this paper, we are mainly interested in a refinement of this situation to cases where \mathbb{A} is endowed with a “length” function; in such cases, the cost analysis must take this length into account. Rather than giving an axiomatic treatment, we will assume that we are in one of the following situations:

- $\mathbb{A} = k[T]$ and $\mathbb{K} = k(T)$, for a field k , where we use the length function $\lambda(a) = \deg(a)$, for $a \in \mathbb{A} - \{0\}$;
- $\mathbb{A} = \mathbb{Z}$ and $\mathbb{K} = \mathbb{Q}$, where we use the length function $\lambda(a) = \log(|a|)$, for $a \in \mathbb{A} - \{0\}$.

In both cases, the length of $a \in \mathbb{A}$ represents the amount of storage needed to represent it, in terms of elements of k , resp. bits. It will be useful to introduce a notion of length for polynomials with coefficients in \mathbb{K} : if P is such a polynomial, $\lambda(P)$ denotes the maximum of the lengths $\lambda(n_i)$ and $\lambda(d_i)$, where n_i and d_i are the numerators and denominators of the coefficients of P , when written in reduced form using a common denominator.

When $\mathbb{A} = k[T]$, we are studying the intersection of two surfaces in a 3-dimensional space with coordinates T, X, Y ; the output describes the solution curve for generic T .

In that case, write again $d = \max(\deg(F), \deg(G))$, as well as $\ell = \max(\lambda(F), \lambda(G))$. Then, the polynomials U_1, \dots, U_s in the equiprojectable decomposition (2.1) of $Z(F, G)$ are in $k(T)[X]$, and the sum of their degrees in X is at most d^2 . These polynomials are all factors of the resultant $\text{res}(F, G, Y)$, which implies that $\lambda(U_i)$ is at most $2d\ell$ for each i , so that representing them involves $O(d^3\ell)$ coefficients in k .

For the polynomials V_1, \dots, V_s , however, the bounds are worse: [24] proves that $\lambda(V_i)$ only admits a weaker bound of order $d^3\ell + d^4$, so they involve $O(d^5\ell + d^6)$ coefficients in k . Practice shows that these bounds are realistic: the polynomials V_i are usually much larger than the polynomials U_i . In order to resolve this issue, we will use the polynomials N_1, \dots, N_s defined by $N_i = U_i'V_i \bmod U_i$ for all i . Then, Theorem 2 from [24] combined with the bi-homogeneous Bézout bound shows that $\lambda(N_i) \leq 2d\ell + d^2$ for all i ; thus, storing these polynomials uses $O(d^3\ell + d^4)$ coefficients in k .

Entirely similar considerations apply in the case $\mathbb{A} = \mathbb{Z}$; in that case, Theorem 1 from [24] and an arithmetic Bézout theorem [37] prove that $\lambda(U_i) \leq 2d\ell + 24d^2$, and similarly for $\lambda(N_i)$, so $O(d^3\ell + d^4)$ bits are sufficient to store them.

We call *modified equiprojectable decomposition* of $Z(F, G)$ the set of polynomials $\mathcal{C} = (\mathbf{C}_1, \dots, \mathbf{C}_s)$, with $\mathbf{C}_i = (U_i, N_i)$. These are not monic triangular sets anymore (N_i is not monic in Y), but *regular chains* [6]. In the particular case where $s = 1$ and $V = V_1$ has the form $V(X, Y) = Y - \eta(X)$, it coincides with the rational univariate representation [56].

Main results. Our main results are the following theorems, that give upper bounds on the cost of computing the modified equiprojectable decomposition. We start with the case $\mathbb{A} = k[T]$, where we count operations in k at unit cost. Our second result concerns the case $\mathbb{A} = \mathbb{Z}$; in this case, we measure the cost of our algorithm using bit operations.

In what follows, we let $\mathbf{M} : \mathbb{N} \rightarrow \mathbb{N}$ be such that over any ring, univariate polynomials of degree less than d can be multiplied in $\mathbf{M}(d)$ ring operations, under the super-linearity conditions of [30, Ch. 8]: using FFT techniques, we can take $\mathbf{M}(d) \in O(d \log(d) \log \log(d))$. We also let ω be such that we can multiply $n \times n$ matrices using $O(n^\omega)$ ring operations, over any ring. The best known bound is $\omega < 2.38$ [65].

Theorem 1. *Let k be a field and let F, G be in $k[T][X, Y]$, with $d = \max(\deg(F), \deg(G))$ and $\ell = \max(\lambda(F), \lambda(G))$. If k has characteristic at least $4d^2(6d^2 + 9d\ell)$, one can compute the modified equiprojectable decomposition of $Z(F, G)$ over $k(T)[X, Y]$ by a probabilistic algorithm with probability of success at least $1/2$, using*

$$O(\mathbf{M}(d^2)\mathbf{M}(d\ell + d^2)d^{(\omega-1)/2} \log(d\ell)) \subset O^\sim(d^{3.69}\ell + d^{4.69})$$

operations in k .

Theorem 2. *Let $\varepsilon > 0$, let F, G be in $\mathbb{Z}[X, Y]$, and write $d = \max(\deg(F), \deg(G))$ and $\ell = \max(\lambda(F), \lambda(G))$. One can compute the modified equiprojectable decomposition of $Z(F, G)$ over $\mathbb{Q}[X, Y]$ by a probabilistic algorithm with probability of success at least $1/2$, using $O(d^{3+\varepsilon}\ell + d^{4+\varepsilon})$ bit operations.*

In both cases, one can easily obtain a cost of $O^\sim(d^4\ell + d^5)$ using modular methods: e.g., over $\mathbb{A} = k[T]$, solve the system at $O(d\ell + d^2)$ values of T , each of which costs $O^\sim(d^3)$ operations in k , and use rational function interpolation. Our main contribution is to show that this direct approach is sub-optimal; over $\mathbb{A} = \mathbb{Z}$, the cost of our algorithm almost matches the known upper bounds on the output size.

The structure of our algorithm is the same in both cases: we compute $Z(F, G)$ modulo an ideal \mathfrak{m} of \mathbb{A} , lift the result modulo a high power of \mathfrak{m} and reconstruct all rational

function coefficients. This approach is similar to the algorithm of [23]; the key difference is in how we implement the lifting process. The result in [23] assumes that the input system is given by means of a straight-line program; here, we assume that the input is dense, and we rely on fast *modular composition techniques*.

Our results imply similar bounds for computing the resultant $R = \text{res}(F, G, Y)$, at least for systems without singular roots: one can reconstruct R from U_1, \dots, U_s , taking care if needed of the leading coefficients of F and G in Y ; we leave the details to the reader. The main challenge is to handle systems with multiplicities without affecting the complexity. We expect that deflation techniques will make this possible.

After a section of preliminaries, we give (Section 2.3) an algorithm to compute $Z(F, G)$ over an arbitrary field in time $O(d^3)$. Section 2.4 is devoted to computing normal forms modulo triangular sets by means of modular composition techniques; this is the key ingredient of the main algorithm given in Section 2.5. Section 2.6 presents experimental results.

2.2 Preliminaries

2.2.1 Notation and basic results

In the introduction, we defined monic triangular sets with coefficients in a field. We will actually allow coefficients in a ring \mathbb{A} ; as in the introduction, all monic triangular sets will be bivariate, that is, in $\mathbb{A}[X, Y]$.

For positive integers m, n , $\mathbb{A}[X]_m$ denotes the set of all $F \in \mathbb{A}[X]$ such that $\deg(F) < m$, and $\mathbb{A}[X, Y]_{m,n}$ the set of all $F \in \mathbb{A}[X, Y]$ such that $\deg(F, X) < m$ and $\deg(F, Y) < n$. Using Kronecker's substitution, we can multiply polynomials in $\mathbb{A}[X, Y]_{m,n}$ in $O(\mathbf{M}(mn))$ operations in \mathbb{A} .

For a monic triangular set \mathbf{T} in $\mathbb{A}[X, Y]$, the monicity assumption makes the notion of remainder modulo the ideal $\langle \mathbf{T} \rangle$ well-defined; if \mathbf{T} has bidegree (m, n) , then for any F in $\mathbb{A}[X, Y]$, the remainder $F \bmod \langle \mathbf{T} \rangle$ is in $\mathbb{A}[X, Y]_{m,n}$. In terms of complexity, we have the following result about computations with such a triangular set (see [50, 44]).

Lemma 1. *Let \mathbf{T} be a monic triangular set in $\mathbb{A}[X, Y]$ of bidegree (m, n) . Then, given $F \in \mathbb{A}[X, Y]_{m',n'}$, with $m \leq m'$ and $n \leq n'$, one can compute $F \bmod \langle \mathbf{T} \rangle$ in $O(\mathbf{M}(m'n'))$ operations in \mathbb{A} . Additions, resp. multiplications modulo $\langle \mathbf{T} \rangle$ can be done in $O(mn)$, resp. $O(\mathbf{M}(mn))$ operations in \mathbb{A} .*

We continue with a result on polynomial matrix multiplication. The proof is the same as that of [13, Lemma 8], up to replacing univariate polynomials by bivariate

ones. Remark that for such rectangular matrix multiplications, one could actually use an algorithm of Huang and Pan's [35], which features a slightly better exponent (for current values of ω).

Lemma 2. *Let $\mathbf{M}_1, \mathbf{M}_2$ be matrices of sizes $(a \times b)$ and $(b \times c)$, with entries in $\mathbb{A}[X, Y]_{m,n}$. If $a = O(\ell^{1/2})$, $b = O(\ell^{1/2})$ and $c = O(\ell)$, one can compute $\mathbf{M} = \mathbf{M}_1 \mathbf{M}_2$ using $O(\mathbf{M}(mn)\ell^{(\omega+1)/2})$ operations in \mathbb{A} .*

2.2.2 Chinese Remainder techniques

Let $\mathcal{T} = (\mathbf{T}_1, \dots, \mathbf{T}_s)$ be a family of monic triangular sets in $\mathbb{A}[X, Y]$, where \mathbb{A} is a ring. In such situations, we write $\langle \mathcal{T} \rangle = \langle \mathbf{T}_1 \rangle \cap \dots \cap \langle \mathbf{T}_s \rangle$; if \mathbb{A} is a field, we write $V(\mathcal{T}) = V(\mathbf{T}_1) \cup \dots \cup V(\mathbf{T}_s)$, where $V(\mathbf{T})$ denotes the zero-set of \mathbf{T} over the algebraic closure of \mathbb{A} .

Following [23], we say that \mathcal{T} is *non-critical* if for i in $\{1, \dots, s\}$, $F_i = U_1 \cdots U_{i-1} U_{i+1} \cdots U_s$ is invertible modulo U_i ; if \mathbb{A} is a field, this simply means that U_1, \dots, U_s are pairwise coprime. The family \mathcal{T} is a *non-critical decomposition* of an ideal I if \mathcal{T} is non-critical and $\langle \mathcal{T} \rangle = I$.

Let $\mathcal{T} = (\mathbf{T}_1, \dots, \mathbf{T}_s)$ be a non-critical family of triangular sets, with $\mathbf{T}_i = (U_i(X), V_i(X, Y))$ of bidegree (m_i, n_i) , and suppose that there exists n such that $n_i = n$ for all i ; let also $m = m_1 + \dots + m_s$. Under these conditions, the following lemma shows how to merge \mathcal{T} into a single monic triangular set \mathbf{T} of bidegree (m, n) . Because \mathbb{A} may not be a field, we assume that $\mathcal{R} = (R_1, \dots, R_s)$ is part of the input, with $R_i = 1/F_i \bmod U_i$; we call them the *cofactors* of \mathcal{T} .

Lemma 3. *Given a non-critical family \mathcal{T} as above, under the assumption $n_i = n$ for all i , and given the cofactors \mathcal{R} , we can compute a monic triangular set \mathbf{T} of bidegree (m, n) such that $\langle \mathbf{T} \rangle = \langle \mathcal{T} \rangle$ using $(n\mathbf{M}(m) \log(m))$ operations in \mathbb{A} .*

Given $F \in \mathbb{A}[X, Y]$ reduced modulo $\langle \mathbf{T} \rangle$, we can compute all polynomials $F_i = F \bmod \langle \mathbf{T}_i \rangle$ using $O(n\mathbf{M}(m) \log(m))$ operations in \mathbb{A} .

PROOF. For $i = 1, \dots, s$, write $V_i = \sum_{j=0}^n v_{i,j} Y^j$, with all $v_{i,j}$ in $\mathbb{A}[X]$. Algorithm 10.22 in [30], where our polynomials R_i are written s_i , allows us to apply the Chinese Remainder Theorem, yielding v_0, \dots, v_n in $\mathbb{A}[X]$ such that $v_{i,j} = v_j \bmod U_i$ for all i, j . Since $v_{i,n} = 1$ for all i , $v_n = 1$ as well, so we let $U = U_1 \cdots U_s$, $V = \sum_{j=0}^n v_j Y^j$ and $\mathbf{T} = (U, V)$. Computing U takes $O(\mathbf{M}(m) \log(m))$ by [30, Lemma 10.4] and computing V takes a total time of $O(n\mathbf{M}(m) \log(m))$ by [30, Coro. 10.23].

To prove the second point, write $F = \sum_{j=0}^{n-1} f_j Y^j$, with all f_j in $\mathbb{A}[X]$. For $j = 0, \dots, n-1$, we apply the modular reduction algorithm of [30, Algo. 10.16] to compute

$f_{1,j}, \dots, f_{s,j}$, with $f_{i,j} = f_j \bmod U_i$; we return $F_i = \sum_{j=0}^{n-1} f_{i,j} Y^j$, for $i = 1, \dots, s$. The total time is n times the cost of modular reduction, that is, $O(nM(m) \log(m))$. \square

Corollary 1. *Let \mathbb{K} be a field and let $\mathcal{T} = (\mathbf{T}_1, \dots, \mathbf{T}_s)$ be a non-critical family of monic triangular sets in $\mathbb{K}[X, Y]$, with $\mathbf{T}_i = (U_i, V_i)$ of bidegree (m_i, n_i) for all i . Suppose that the ideal $\langle \mathcal{T} \rangle \cdot \overline{\mathbb{K}}[X, Y]$ is radical. Then one can compute the equiprojectable decomposition of the ideal $\langle \mathcal{T} \rangle$ using $O(M(\delta) \log(\delta))$ operations in \mathbb{K} , with $\delta = \sum_{1 \leq i \leq s} m_i n_i$.*

PROOF. Partition \mathcal{T} in the classes of the equivalence relation where $(U, V) \equiv (U^*, V^*)$ if and only if $\deg(V, Y) = \deg(V^*, Y)$. Let $\mathcal{T}_1, \dots, \mathcal{T}_t$ be these classes; for $j \in \{1, \dots, t\}$, let $\mu_j = \sum_{(U,V) \in \mathcal{T}_j} \deg(U, X)$ and let ν_j be the common value of $\deg(V, Y)$ for $(U, V) \in \mathcal{T}_j$; then, $\sum_{1 \leq j \leq t} \mu_j \nu_j = \delta$.

For $j = 1, \dots, t$, let \mathbf{T}_j^* be the monic triangular set obtained by applying the previous lemma to \mathcal{T}_j . Since \mathbb{K} is a field, the cofactors \mathcal{B}_j are computed in time $O(M(\mu_j) \log(\mu_j))$ using [30, Algo. 10.18], so the total time for any fixed j is $O(\nu_j M(\mu_j) \log(\mu_j))$, which is $O(M(\nu_j \mu_j) \log(\nu_j \mu_j))$. Summing over all j , the total cost is seen to be $O(M(\delta) \log(\delta))$.

Since $\langle \mathcal{T} \rangle$ is radical in $\overline{\mathbb{K}}[X, Y]$, we deduce that for all $i \in \{1, \dots, s\}$, the zero-set Z_i of \mathbf{T}_i is equiprojectable, with fibers for the projection $\pi : \overline{\mathbb{K}}^2 \rightarrow \overline{\mathbb{K}}$ all having cardinality n_i . Thus, the triangular sets $\mathbf{T}_1^*, \dots, \mathbf{T}_t^*$ form the equiprojectable decomposition of $\langle \mathcal{T} \rangle$. \square

2.2.3 Specialization properties

Consider a domain \mathbb{A} , its fraction field \mathbb{K} , and a maximal ideal $\mathfrak{m} \subset \mathbb{A}$ with residual field $k = \mathbb{A}/\mathfrak{m}$. Given F and G in $\mathbb{A}[X, Y]$, our goal here is to relate the equiprojectable decomposition of $Z(F, G)$ to that of $Z(F \bmod \mathfrak{m}, G \bmod \mathfrak{m})$, where the former is defined over \mathbb{K} and the latter over k .

The following results give quantitative estimates for ideals of “good reduction” in the two cases we are interested in, $\mathbb{A} = k[T]$ and $\mathbb{A} = \mathbb{Z}$; in both cases, we use the length function λ defined in the introduction. The case $\mathbb{A} = k[T]$, while not treated in [23], is actually the simpler, so we only sketch the proof; for $\mathbb{A} = \mathbb{Z}$, we can directly apply [23, Th. 1].

Lemma 4. *Let F, G be in $k[T][X, Y]$ and let $\mathbf{T}_1, \dots, \mathbf{T}_s \subset k(T)[X, Y]$ be the equiprojectable decomposition of $Z(F, G)$. If $d = \max(\deg(F), \deg(G))$ and $\ell = \max(\lambda(F), \lambda(G))$, there exist $A \in k[T] - \{0\}$ of degree at most $2d^2(6d^2 + 9d\ell)$ and with the following property.*

If an element $t_0 \in k$ does not cancel A , then none of the denominators of the coefficients of $\mathbf{T}_1, \dots, \mathbf{T}_s$ vanishes at $T = t_0$ and their evaluation at $T = t_0$ forms the equiprojectable decomposition of $Z(F(t_0, X, Y), G(t_0, X, Y))$.

PROOF. The approach of [23, Section 3] still applies in this case, and shows that if an element $t_0 \in k$ satisfies three assumptions (denoted by H_1, H_2, H_3 in [23]), then the specialization property holds. These properties imply the existence of a non-zero $A \in k[T]$ as claimed in the lemma; its degree can be bounded using the results of [59, 24]. \square

Lemma 5. *Let F, G be in $\mathbb{Z}[X, Y]$ and let $\mathbf{T}_1, \dots, \mathbf{T}_s \subset \mathbb{Q}[X, Y]$ be the equiprojectable decomposition of $Z(F, G)$. If $d = \max(\deg(F), \deg(G))$ and $\ell = \max(\lambda(F), \lambda(G))$, there exists $A \in \mathbb{N} - \{0\}$, with $\lambda(A) \leq 8d^5(3\ell + 10 \log(d) + 22)$ and with the following property.*

If a prime $p \in \mathbb{N}$ does not divide A , then none of the denominators of the coefficients of $\mathbf{T}_1, \dots, \mathbf{T}_s$ vanishes modulo p , and their reduction modulo p forms the modified equiprojectable decomposition of $Z(F \bmod p, G \bmod p)$.

2.3 A direct algorithm

In this section, we work over a field \mathbb{K} . We give an algorithm that takes as input F, G in $\mathbb{K}[X, Y]$ and computes the equiprojectable decomposition $\mathbf{T}_1, \dots, \mathbf{T}_s$ of $Z(F, G)$. If F and G have degree at most d , the running time is $O^\sim(d^3)$, that is, essentially the same as computing $\text{res}(F, G, Y)$ (we count all operations in \mathbb{K} at unit cost). This result is by no means surprising (a particular case appears in [43]) and certainly not enough to prove our main theorems. We will only use it as the initialization step of our lifting process.

The rest of this section is devoted to prove this proposition, following a few preliminaries.

Proposition 1. *Let F, G be in $\mathbb{K}[X, Y]$, of total degree at most d . If the characteristic of \mathbb{K} is greater than $2d^2 + d + 1$, one can compute the equiprojectable decomposition of $Z(F, G)$ using $O(\mathbf{M}(d)\mathbf{M}(d^2) \log(d)^2)$ operations in \mathbb{K} .*

Regular GCDs and quotients. Let R be a nonzero, squarefree polynomial in $\mathbb{K}[X]$, and let F, G be in $\mathbb{K}[X, Y]$. A *regular GCD* of (F, G) modulo R is a non-critical decomposition of the ideal $\langle R, F, G \rangle$; a *regular quotient* of F by G modulo R is a non-critical decomposition of the ideal $\langle R, F \rangle : G$. If $\mathcal{S} = (\mathbf{T}_1, \dots, \mathbf{T}_s)$ is a regular GCD of (F, G) modulo R , with $\mathbf{T}_i = (U_i, V_i)$ for all i , and if F is monic in Y , then $\mathcal{S} = (\xi_1, \dots, \xi_s)$, with $\xi_i = (U_i, F/V_i \bmod U_i)$ for all i , is a regular quotient of F by G modulo R .

If F, G have degree at most d in Y , and R, F, G have degree at most m in X , then using the algorithm of [3], both operations can be done in time $O(\mathbf{M}(d)\mathbf{M}(m) \log(d) \log(m))$.

Radical computation. Regular quotients allow us to compute radicals. Let indeed $\mathbf{T} = (U, V)$ be a monic triangular set of bidegree (m, n) in $\mathbb{K}[X, Y]$, with U squarefree

and with m and n less than the characteristic of \mathbb{K} ; we prove that $I = \langle U, V \rangle : \partial V / \partial Y$ is the radical of the ideal $\langle \mathbf{T} \rangle$.

Let I' be the extension of I in $\overline{\mathbb{K}}[X, Y]$. Over $\overline{\mathbb{K}}$, the assumption on m ensures that U is still squarefree, so the ideal $\langle U, V \rangle$ is the intersection of primary ideals of the form $\mathfrak{p}_i = \langle X - x_i, (Y - y_i)^{e_i} \rangle$, where $(x_i, y_i)_{1 \leq i \leq t}$ are the zeros of \mathbf{T} , and $e_i \in \mathbb{N}_{>0}$ is the multiplicity of the factor $Y - y_i$ in $V(x_i, Y)$. Then, I' is the intersection of the ideals $\mathfrak{p}_i : \partial V / \partial Y$, which we can rewrite as

$$I' = \bigcap_{1 \leq i \leq t} \langle X - x_i, (Y - y_i)^{e_i} \rangle : (e_i(Y - y_i)^{e_i - 1}).$$

The assumption on n implies that $e_i \neq 0$ in \mathbb{K} , so that I' is the intersection of the maximal ideals $\langle X - x_i, Y - y_i \rangle$; our claim is proved. As a consequence, under the above assumption on \mathbf{T} , we can compute a non-critical decomposition of the radical of $\langle \mathbf{T} \rangle$ in time $O(\mathbf{M}(n)\mathbf{M}(m) \log(n) \log(m))$.

After these preliminaries, we can turn to the algorithm. In what follows, J is the Jacobian determinant of (F, G) , $H = \gcd(F, G)$, $F^* = F/H$ and $G^* = G/H$. The idea is classical: we compute the resultant $R = \text{res}(F^*, G^*, Y)$, then a regular GCD of (F^*, G^*) modulo R ; make the result radical and finally we remove all points where J vanishes.

Step 0. We compute H, F^*, G^* as defined above. Corollary 11.9 in [30] gives an *expected* $O(d\mathbf{M}(d) \log(d))$ operations for computing H ; we briefly explain how to make it deterministic, up to an acceptable increase in running time (this is routine; some details are left to the reader).

Choosing $2d^2 + d + 1$ values x_i in \mathbb{K} , we compute $H_i = \gcd(F(x_i, Y), G(x_i, Y))$, $F_i^* = F(x_i, Y)/H_i$ and $G_i^* = G(x_i, Y)/H_i$. Lucky values of x_i are those where the leading coefficient of (say) F in Y and the resultant of (F^*, G^*) in Y are non-zero. We are sure to find at least $d^2 + 1$ of them; these will be those x_i 's where H_i has maximal degree. These are enough to reconstruct H, F^* and G^* by interpolation, hence a total running time of $O(\mathbf{M}(d)\mathbf{M}(d^2) \log(d))$.

Step 1. Compute the (nonzero) resultant $R = \text{res}(F^*, G^*, Y)$. Using Reischert's algorithm [55], this takes time $O(\mathbf{M}(d)\mathbf{M}(d^2) \log(d))$.

Step 2. Replace R by its squarefree part, which takes time $O(\mathbf{M}(d^2) \log(d))$. Note that $V(R, F^*, G^*) = V(F^*, G^*)$.

Step 3. Compute a regular GCD $\mathcal{S} = (\mathbf{T}_1, \dots, \mathbf{T}_s)$ of (F^*, G^*) modulo R , in time $O(\mathbf{M}(d)\mathbf{M}(d^2) \log(d)^2)$. Note that $V(\mathcal{S})$ is equal to $V(R, F^*, G^*)$, that is, $V(F^*, G^*)$.

Step 4. For $i = 1, \dots, s$, writing $\mathbf{T}_i = (U_i, V_i)$, compute a regular quotient of V_i by

$\partial V_i / \partial Y$ modulo U_i .

Letting (m_i, n_i) be the bidegree of \mathbf{T}_i , the cost for each i is $O(\mathbf{M}(d)\mathbf{M}(m_i) \log(d) \log(m_i))$. Using the super-linearity of \mathbf{M} , the total is seen to be $O(\mathbf{M}(d)\mathbf{M}(d^2) \log(d)^2)$.

Let $\mathcal{S} = (\xi_1, \dots, \xi_t)$ be the union of all triangular sets obtained this way, with $\xi_i = (P_i, Q_i)$. Since d^2 is less than the characteristic of \mathbb{K} , this is also the case for all m_i and n_i . As a result, by the discussion above, $\langle \mathcal{S} \rangle$ is the defining ideal of $V(F^*, G^*)$; in particular, it is radical in $\overline{\mathbb{K}}[X, Y]$.

Step 5. For $i = 1, \dots, t$, compute $J_i = J \bmod P_i$, where J is the Jacobian determinant of (F, G) . Using fast simultaneous modular reduction, this costs $O(d\mathbf{M}(d^2) \log(d))$.

Step 6. For $i = 1, \dots, t$, compute a regular quotient of Q_i by J_i modulo P_i ; again, the cost is $O(\mathbf{M}(d)\mathbf{M}(d^2) \log(d)^2)$. Let \mathcal{U} be the union of all resulting triangular sets; then, $\langle \mathcal{U} \rangle$ is the defining ideal of $V(F^*, G^*) - V(J)$, and one verifies that the latter set is $Z(F, G)$.

Step 7. Finally, apply the algorithm of Corollary 1 to \mathcal{U} to obtain the equiprojectable decomposition of $Z(F, G)$. Since $Z(F, G)$ has size at most d^2 , the cost is $O(\mathbf{M}(d^2) \log(d))$.

2.4 Normal form algorithms

We consider now the problem of reducing $F \in \mathbb{A}[X, Y]$ modulo several triangular sets. Our input is as follows:

- $\mathcal{T} = (\mathbf{T}_1, \dots, \mathbf{T}_s)$ is a non-critical family of monic triangular sets in $\mathbb{A}[X, Y]$, where $\mathbf{T}_i = (U_i, V_i)$ of bidegree (m_i, n_i) for all i and \mathbb{A} is a ring;
- $\mathcal{R} = (R_1, \dots, R_s)$ is the family of cofactors associated to \mathcal{T} , as in Subsection 2.2.2;
- F is in $\mathbb{A}[X, Y]$, of total degree less than d .

We make the following assumptions:

$$\sum_{i=1, \dots, s} m_i n_i \leq d^2 \quad \text{and} \quad n_i \leq d \quad \text{for all } i. \quad (\mathbf{H})$$

Then, the size of input and output is $\Theta(d^2)$ elements of \mathbb{A} .

Already for $s = 1$, in which case we write (m, n) instead of (m_1, n_1) , the difficulty of the problem can vary significantly: if both m and n are $O(d)$, Lemma 1 shows that the reduction can be done in optimal time $O^\sim(d^2)$; however, when $m \simeq d^2$ and $n \simeq 1$, that same lemma gives a sub-optimal $O^\sim(d^3)$.

In this section, we prove the following propositions, which give algorithms with better exponents. The first one applies over any ring \mathbb{A} ; it uses fast matrix multiplication to achieve an exponent $(\omega + 3)/2 \simeq 2.69$.

Proposition 2. *Under assumption (H), there exists an algorithm that takes as input polynomials \mathcal{T} , \mathcal{R} and F as above and returns all $F \bmod \langle \mathbf{T}_i \rangle$, for i in $\{1, \dots, s\}$, using $O(\mathbf{M}(d^2)d^{(\omega-1)/2} \log(d))$ operations in \mathbb{A} .*

When $\mathbb{A} = \mathbb{Z}/N\mathbb{Z}$, for some prime power N , better can be done in a *boolean* model: this second proposition shows that we can get arbitrarily close to linear time (in the boolean model, input and output sizes are $\Theta(d^2 \log(N))$).

Proposition 3. *Under assumption (H), for any $\varepsilon > 0$, there exists an algorithm that takes as input a prime power N , and polynomials \mathcal{T} , \mathcal{R} and F as above, all with coefficients in $\mathbb{Z}/N\mathbb{Z}$, and returns all $F \bmod \langle \mathbf{T}_i \rangle$, for i in $\{1, \dots, s\}$, using $d^{2+\varepsilon} O^\sim(\log(N))$ bit operations.*

2.4.1 Reduction modulo one triangular set

We first discuss a simplified version of the problem, where we reduce F modulo a single monic triangular set. In other words, we take $s = 1$; then, we simply write $\mathbf{T} = (U, V)$ instead of \mathbf{T}_1 , and we denote its bidegree by (m, n) instead of (m_1, n_1) . The polynomial F is in $\mathbb{A}[X, Y]$, of degree less than d ; thus our assumptions are the following:

$$mn \leq d^2 \quad \text{and} \quad n \leq d. \quad (\mathbf{H}')$$

In [54], Poteaux and Schost give two algorithms computing $F \bmod \langle \mathbf{T} \rangle$. The first one, originating from [53, Ths. 4-6], applies only in a boolean model, when $\mathbb{A} = \mathbb{Z}/p\mathbb{Z}$ for a prime p . Only a small change is needed to make it work modulo a prime power $N = p^\ell$. In both cases, when the base ring, or field, is too small, we need to enlarge it, by adding elements whose differences are invertible. In our case, we extend the basering $\mathbb{Z}/N\mathbb{Z}$ by a polynomial that is irreducible modulo p (since if $x - x'$ is a unit modulo p , it is a unit modulo N). The analysis of [53, Ths. 4-6] remains valid with this minor modification, and yields the following result.

Proposition 4. [53, 54] *Under assumption (H'), for any $\varepsilon > 0$, there exists an algorithm that takes as input a prime power N and F and \mathbf{T} as above, with coefficients in $\mathbb{Z}/N\mathbb{Z}$, and returns $F \bmod \langle \mathbf{T} \rangle$ using $d^{2+\varepsilon} O^\sim \log(N)$ bit operations.*

Since in this case the input and output size is $\Theta(d^2 \log(N))$ bits, this algorithm is close to being optimal.

If we consider the question over an abstract ring \mathbb{A} , no quasi-optimal algorithm is known. Under assumption (\mathbf{H}') , the second algorithm of [54] runs in time $O(d^{\omega+1})$; this is subquadratic in the size d^2 of the problem, but worse than $O^\sim(d^3)$. The following proposition gives an improved result.

Proposition 5. *Under assumption (\mathbf{H}') , there exists an algorithm that takes as input F and \mathbf{T} as above and returns $F \bmod \langle \mathbf{T} \rangle$ using $O(\mathbf{M}(d^2)d^{(\omega-1)/2})$ operations in \mathbb{A} .*

The rest of this subsection is devoted to prove this proposition. As in [54], we use a baby steps / giant steps approach inspired by Brent and Kung's algorithm [17], but with a slightly more refined subdivision scheme.

Let thus F be in $\mathbb{A}[X, Y]$, of total degree less than d , and let $\mathbf{T} = (U, V)$ be of bidegree (m, n) . The steps of the algorithm are given below: they consist in computing some powers of Y modulo $\langle \mathbf{T} \rangle$ (baby steps, at Step 3), doing products of matrices with entries in $\mathbb{A}[X, Y]$ (Step 4), and concluding using Horner's scheme (giant steps, at Step 6).

Step 0. Replace F by $F \bmod U$; as a consequence, we can assume $F \in \mathbb{A}[X, Y]_{r,d}$, with $r = \min(d, m)$. For future use, note that $mn \leq rd$: if $r = d$, this is because $mn \leq d^2$. Else, $r = m$, and the claim follows from the fact that $n \leq d$.

We do d reductions of polynomials of degree less than d by a polynomial of degree m in $\mathbb{A}[X]$; this is free if $d < m$ and costs $O(d\mathbf{M}(d))$ otherwise.

Step 1. Let $t = \lceil d/n \rceil - 1$ and write F as $F = F_0 + F_1Y^n + \dots + F_tY^{nt}$, with all F_i in $\mathbb{A}[X, Y]_{r,n}$.

Step 2. Let $\rho = \lfloor d/n^{1/2} \rfloor$ and $\mu = \lceil (t+1)/\rho \rceil$; note that since $d \geq n$, $\rho \geq 1$ so μ is well-defined. Furthermore, both ρ and μ are $O(d/n^{1/2})$ and $\rho\mu \geq t+1$. Set up the $(\mu \times \rho)$ matrix $\mathbf{M}_1 = [F_{i\rho+j}]_{0 \leq i < \mu, 0 \leq j < \rho}$ with entries in $\mathbb{A}[X, Y]_{r,n}$, where we set $F_k = 0$ for $k > t$.

Step 3. For $i = 0, \dots, \rho$, compute $\sigma_i = Y^{ni} \bmod \langle \mathbf{T} \rangle$. Cost: since $\deg(V, Y) = n$, $\sigma_1 = (Y^n \bmod \langle \mathbf{T} \rangle)$ is equal to $Y^n - V$, so computing it takes time $O(mn)$. Then, it takes time $O(\rho\mathbf{M}(mn))$ to deduce all other σ_i 's.

Step 4. Let $\nu = \lceil m/r \rceil - 1$; for $i = 0, \dots, \rho - 1$, write $\sigma_i = \sigma_{i,0} + \sigma_{i,1}X^r + \dots + \sigma_{i,\nu}X^{r\nu}$, with all $\sigma_{i,j}$ in $\mathbb{A}[X, Y]_{r,n}$. Build the $\rho \times (\nu+1)$ matrix $\mathbf{M}_2 = [\sigma_{i,j}]_{0 \leq i < \rho, 0 \leq j \leq \nu}$ and compute $\mathbf{M} = \mathbf{M}_1\mathbf{M}_2$.

Cost: we have seen that $mn \leq rd$, so that $m/r \leq d/n$ and thus $\nu = O(d/n)$. Using the bounds on ρ, μ, ν and Lemma 2, we deduce that the cost is $O(\mathbf{M}(rn)(e/n)^{(\omega+1)/2})$.

Step 5. Let $[m_{i,j}]_{0 \leq i < \mu, 0 \leq j \leq \nu}$ be the entries of \mathbf{M} , which are in $\mathbb{A}[X, Y]_{2r-1, 2n-1}$. For $i = 0, \dots, \mu - 1$, compute $G_i = m_{i,0} + m_{i,1}X^r + \dots + m_{i,\nu}X^{r\nu}$ and $H_i = G_i \bmod \langle \mathbf{T} \rangle$.

Because $m_{i,j} = F_{i\rho}\sigma_{0,j} + F_{i\rho+1}\sigma_{1,j} + \cdots + F_{(i+1)\rho-1}\sigma_{\rho-1,j}$, we deduce that $G_i = F_{i\rho}\sigma_0 + F_{i\rho+1}\sigma_1 + \cdots + F_{(i+1)\rho-1}\sigma_{\rho-1}$. Since $\sigma_j = Y^{nj} \bmod \langle \mathbf{T} \rangle$ for all j , this proves that $H_i = F_{i\rho} + F_{i\rho+1}Y^n + \cdots + F_{(i+1)\rho-1}Y^{n(\rho-1)} \bmod \langle \mathbf{T} \rangle$.

Computing a single G_i takes $O(rn\nu)$ additions in \mathbb{A} , which is $O(mn)$ since $r\nu = O(m)$. By construction, G_i is in $\mathbb{A}[X, Y]_{r(\nu+2)-1, 2n-1}$; since $r(\nu+2) = O(m)$, Lemma 1 implies that reducing G_i to obtain H_i takes time $O(\mathbf{M}(mn))$. The total for all G_i 's is $O(\mu\mathbf{M}(mn))$.

Step 6. Compute $H = H_0 + H_1\sigma_\rho + \cdots + H_{\mu-1}\sigma_\rho^{\mu-1} \bmod \langle \mathbf{T} \rangle$ using Horner's scheme; the expression given above for the polynomials H_i implies that $H = F \bmod \langle \mathbf{T} \rangle$. Cost: $O(\rho)$ additions and multiplications modulo \mathbf{T} , each of which costs $O(\mathbf{M}(mn))$ operations in \mathbb{A} .

Summary. Summing all contributions, we obtain

$$O(\mathbf{M}(d)d + \mathbf{M}(mn)(d/n)^{1/2} + \mathbf{M}(rn)(d/n)^{(\omega+1)/2}).$$

The first two terms are easily seen to be $O(\mathbf{M}(d^2)d^{1/2})$. To deal with the last term, note that $r \leq d$ implies $\mathbf{M}(rn) \leq \mathbf{M}(dn)$, and the super-linearity of \mathbf{M} implies that $\mathbf{M}(dn) \leq \mathbf{M}(d^2)n/d$. Thus, the third term is $O(\mathbf{M}(d^2)(d/n)^{(\omega-1)/2})$, which is $O(\mathbf{M}(d^2)d^{(\omega-1)/2})$. Proposition 5 is proved.

2.4.2 Reduction modulo several triangular sets

We now prove Proposition 2 and 3. To simplify the presentation, we give details for the first result (in the algebraic model); the proof in the boolean model requires no notable modification (just use Proposition 10 instead of 5 below).

Let thus $\mathcal{T} = (\mathbf{T}_1, \dots, \mathbf{T}_s)$ be monic triangular sets of the form $\mathbf{T}_i = (U_i, V_i)$, with coefficients in \mathbb{A} and bidegrees (m_i, n_i) . We also assume that the cofactors $\mathcal{R} = (R_1, \dots, R_s)$ are given. Given F in $\mathbb{A}[X, Y]$ of degree less than d , we consider the question of reducing F modulo all $\langle \mathbf{T}_i \rangle$, under assumption **(H)**. Our proof distinguishes three cases, from the most particular to the general case.

Identical n_i 's. Assume first that there exists n such that $n_i = n$ for all i . Writing $m = m_1 + \cdots + m_s$, Lemma 3 shows that we can build a monic triangular set \mathbf{T} in $\mathbb{A}[X, Y]$ of bidegree (m, n) , such that the ideal $\langle \mathbf{T} \rangle$ is the intersection of all $\langle \mathbf{T}_i \rangle$, in time $O(n\mathbf{M}(m) \log(m))$

To compute all $F \bmod \langle \mathbf{T}_i \rangle$, because **(H)** implies $mn \leq d^2$, we first compute $H = F \bmod \langle \mathbf{T} \rangle$ using Proposition 5, in time $O(\mathbf{M}(d^2)d^{(\omega-1)/2})$. Then, we use Lemma 3 to reduce H modulo all $\langle \mathbf{T}_i \rangle$ in time $O(n\mathbf{M}(m) \log(m))$. Since $n\mathbf{M}(m)$ is $O(\mathbf{M}(d^2))$, the cost

of this step is negligible.

Similar n_i 's. We now relax the assumption that all n_i 's are the same; instead, we assume that there exists n such that $n_i \in \{n, \dots, 2n - 1\}$ for all i ; as above, we write $m = m_1 + \dots + m_s$, and we introduce $n' = 2n - 1$.

For $i = 1, \dots, s$, define $V_i^* = Y^{n'-n_i}V_i$ and $\mathbf{T}_i^* = (U_i, V_i^*)$, so that the \mathbf{T}_i^* 's are monic triangular sets of bidegrees (m_i, n') . These new triangular sets and F may not satisfy **(H)** anymore, but they will, provided we replace d by $d' = 2d$. Indeed, notice that the inequality $n' \leq 2n_i$ holds for all i ; using **(H)**, this yields

$$\sum_{i=1, \dots, s} m_i n' \leq 2 \sum_{i=1, \dots, s} m_i n_i \leq 2d^2 \leq d'^2,$$

and similarly $n' \leq d'$. The algorithm in the previous paragraph then allows us to compute all $H_i = F \bmod \langle \mathbf{T}_i^* \rangle$, still in time $O(\mathbf{M}(d^2)d^{(\omega-1)/2})$.

Then, for all i , we compute the remainder $H_i \bmod \langle \mathbf{T}_i \rangle$. Using Lemma 1, this can be done in time $O(\mathbf{M}(m_i n))$ for each i , for a negligible total cost of $O(\mathbf{M}(mn)) \subset O(\mathbf{M}(d^2))$.

Arbitrary degrees. Finally, we drop all assumptions on the degrees n_i . Instead, we partition the set $S = \{1, \dots, s\}$ into S_0, \dots, S_κ such that i is in S_ℓ if and only if n_i is in $\{2^\ell, \dots, 2^{\ell+1} - 1\}$. Because all n_i satisfy $n_i \leq d$, κ is in $O(\log(d))$. We write as usual $m = m_1 + \dots + m_s$.

We are going to apply the algorithm of the previous paragraph to all S_ℓ independently. Remark that if all \mathbf{T}_i and F satisfy assumption **(H)**, the subset $\{\mathbf{T}_i \mid i \in S_\ell\}$ and F still satisfy this assumption. Let us thus fix $\ell \in \{0, \dots, \kappa\}$. The only thing that we need to take care of are the cofactors required for Chinese Remaindering. As input, we assumed that we know all $R_i = 1/(U_1 \cdots U_{i-1} U_{i+1} \cdots U_s) \bmod U_i$; what we need are the inverses $R_{\ell,i} = 1/\prod_{i' \in S_\ell, i' \neq i} U_{i'} \bmod U_i$ for $i \in S_\ell$. These polynomials are computed easily: first, we form the product $B_\ell = \prod_{i \notin S_\ell} U_i$; using [30, Lemma 10.4], this takes $O(\mathbf{M}(m) \log(m))$ operations in \mathbb{A} . Then, we reduce B_ℓ modulo all U_i , for $i \in S_\ell$, for the same amount of time as above. Finally, we obtain all $R_{\ell,i}$ as $R_{\ell,i} = R_i B_\ell \bmod U_i$; the time needed for these products is $O(\mathbf{M}(m))$.

Once the polynomials $R_{\ell,i}$ are known, the algorithm above gives us $F \bmod \langle \mathbf{T}_i \rangle$, for $i \in S_\ell$, in time $O(\mathbf{M}(d^2)d^{(\omega-1)/2})$; this dominates the cost of computing the polynomials $R_{\ell,i}$. Summing over ℓ finishes the proof of Prop. 2.

2.5 Proof of the main results

We assume here that we are one of the cases $\mathbb{A} = k[T]$ or $\mathbb{A} = \mathbb{Z}$ and we prove Theorems 1 and 2. Given F, G in $\mathbb{A}[X, Y]$ and writing as before $\mathcal{T} = (\mathbf{T}_1, \dots, \mathbf{T}_s)$ for the equiprojectable decomposition of $Z(F, G)$ and $\mathcal{C} = (\mathbf{C}_1, \dots, \mathbf{C}_s)$ for its modified version, we show here how to compute the latter.

The algorithm follows the template given in [23]: compute the equiprojectable decomposition modulo a randomly chosen maximal ideal \mathfrak{m} of \mathbb{A} , lift it modulo \mathfrak{m}^N , for N large enough, and reconstruct all rational fractions that appear as coefficients in \mathcal{C} from their expansion modulo \mathfrak{m}^N .

We suppose that $\lambda(F), \lambda(G) \leq \ell$ and $\deg(F), \deg(G) \leq d$, where λ is the length function defined in the introduction. For $i \leq s$, we write (m_i, n_i) for the bidegree of \mathbf{T}_i ; then, we have the upper bound $\sum_{i \leq s} m_i n_i \leq d^2$; besides, each n_i is at most d , so assumption **(H)** of Section 2.4 holds.

2.5.1 One lifting step

Here, \mathfrak{m} is a maximal ideal of \mathbb{A} ; we assume that none of the denominators of the coefficients of the polynomials in \mathcal{T} vanishes modulo \mathfrak{m} . Thus, for $N \geq 1$, we can define $\mathcal{T}_N = \mathcal{T} \bmod \mathfrak{m}^N$ by reducing all coefficients of $\mathcal{T} \bmod \mathfrak{m}^N$. Given \mathcal{T}_N , we show how to compute \mathcal{T}_{2N} ; this will be the core of our main algorithm. We start by describing some basic operations in $\mathbb{A}_N = \mathbb{A}/\mathfrak{m}^N$ (when $N = 1$, we also use the notation k to denote the residual field \mathbb{A}/\mathfrak{m}).

For complexity analyzes, we assume that $\mathbb{A} = k[T]$ and that \mathfrak{m} has the form $\langle T - t_0 \rangle$, for some t_0 in k ; we discuss the case $\mathbb{A} = \mathbb{Z}$ afterwards. Remark in particular that operations $(+, -, \times)$ in \mathbb{A}_N can be done in $O(\mathbf{M}(N))$ operations in k .

Univariate inversion. Given Q monic of degree m in $\mathbb{A}_N[X]$ and $F \in \mathbb{A}_N[X]$ of degree less than m , consider the problem of computing $1/F$ in $\mathbb{A}_N[X]/\langle Q \rangle$, if it exists.

This is done by computing the inverse modulo \mathfrak{m} (*i.e.*, over $k[X]$) by an extended GCD algorithm and lifting it by Newton iteration [30, Ch. 9]. The first step uses $O(\mathbf{M}(m) \log(m))$ operations in k , the second one $O(\mathbf{M}(m)\mathbf{M}(N))$.

Bivariate inversion. Given a monic triangular set \mathbf{T} in $\mathbb{A}_N[X, Y]$ of bidegree (m, n) and $F \in \mathbb{A}_N[X, Y]_{m,n}$, consider the computation of $1/F$ in $\mathbb{A}_N[X, Y]/\langle \mathbf{T} \rangle$, assuming it exists.

We use the same approach as above, but with bivariate computations. For inversion modulo \mathfrak{m} , we use [22, Prop. 6], which gives a cost $O(\mathbf{M}(m)\mathbf{M}(n) \log(m)^3 \log(n)^3)$. The lifting now takes $O(\mathbf{M}(mn)\mathbf{M}(N))$.

Lifting \mathcal{T}_N . We can now explain the main algorithm, called **Lift** in the next subsection. In what follows, we write $\mathcal{T}_N = (\mathbf{T}_{N,1}, \dots, \mathbf{T}_{N,s})$; all computations take place over \mathbb{A}_{2N} .

Step 0. First, as in the proof of Corollary 1, we compute the cofactors \mathcal{R}_N associated to \mathcal{T}_N using [30, Algo. 10.18]; this time, though, we work over the ring \mathbb{A}_{2N} . Steps 1 and 2 of that algorithm work over any ring; Step 3, which computes inverses modulo the polynomials $\mathbf{T}_{N,j}$, is dealt with using the remarks made above on univariate inversion. Because $\mathbf{T}_{N,j}$ has bidegree (m_j, n_j) for all j , with $\sum_{j \leq s} m_j n_j \leq d^2$, the total cost is $O(\mathbf{M}(d^2)\mathbf{M}(N) \log(d))$ operations in k .

Step 1. We will use formulas from [59] to lift from \mathcal{T}_N to \mathcal{T}_{2N} . First, we reduce the polynomials F , G and the entries of their Jacobian matrix J modulo \mathfrak{m}^{2N} ; as a result, we will now see these polynomials as elements of $\mathbb{A}_{2N}[X, Y]$.

Over $\mathbb{A} = k[T]$, the assumption that $\lambda(F), \lambda(G) \leq \ell$ means that F and G have degree at most ℓ in T ; we are reducing them modulo the polynomial $(T - t_0)^{2N}$. The time for one coefficient reduction is $O(\mathbf{M}(\ell))$, since when $2N > \ell$, no work is needed. The total time is $O(d^2 \mathbf{M}(\ell))$.

Step 2. We compute $F_{N,j} = F \bmod \langle \mathbf{T}_{N,j} \rangle$ over $\mathbb{A}_{2N}[X, Y]$ for all $j \in \{1, \dots, s\}$, as well as $G_{N,j} = G \bmod \langle \mathbf{T}_{N,j} \rangle$ and $J_{N,j} = J \bmod \langle \mathbf{T}_{N,j} \rangle$. This is the most costly part of the algorithm: because we know the cofactors \mathcal{R}_N associated to \mathcal{T}_N , and because assumption **(H)** of Section 2.4 is satisfied, Proposition 2 shows that one can compute all $F_{N,j}$ using $O(\mathbf{M}(d^2)\mathbf{M}(N)d^{(\omega-1)/2} \log(d))$ operations in k . The same holds for all $G_{N,j}$ and $J_{N,j}$.

Step 3. Finally, for all j , we compute the (2×2) Jacobian matrix $M_{N,j}$ of $\mathbf{T}_{N,j}$ in $\mathbb{A}_{2N}[X, Y]$ and the vector

$$\delta_{N,j} = M_{N,j} J_{N,j}^{-1} \begin{bmatrix} F_{N,j} \\ G_{N,j} \end{bmatrix} \text{ over } \mathbb{A}_{2N}[X, Y] / \langle \mathbf{T}_{N,j} \rangle.$$

Proposition 4 in [59] then proves that $\mathbf{T}_{2N,j} = \mathbf{T}_{N,j} + \delta_{N,j}^*$, where $\delta_{N,j}^*$ is the canonical preimage of $\delta_{N,j}$ over $\mathbb{A}_{2N}[X, Y]$.

The dominant cost is the inversion of the matrices $J_{N,j}$. By the remark above, the cost for a given j is $O(\mathbf{M}(m_j)\mathbf{M}(n_j) \log(m_j)^3 \log(n_j)^3 + \mathbf{M}(m_j n_j)\mathbf{M}(N))$; summing over j , this step is negligible compared to Step 2.

Summary. When $\mathbb{A} = k[T]$, the cost of deducing \mathcal{T}_{2N} from \mathcal{T}_N is $O(d^2 \mathbf{M}(\ell) + \mathbf{M}(d^2)\mathbf{M}(N)d^{(\omega-1)/2} \log(d))$ operations in k , which is $O^\sim(d^2 \ell + d^{(\omega+3)/2} N)$.

When $\mathbb{A} = \mathbb{Z}$ and $\mathfrak{m} = \langle p \rangle$, for a prime p , the algorithm does not change, but the complexity analysis does. Using the fact that computations modulo p^r can be done

in $O(\log(p^r))$ bit operations, and using Proposition 3, we obtain a cost of $d^{2+\varepsilon}O(\ell + N \log(p))$ bit operations, for any $\varepsilon > 0$.

2.5.2 Main algorithm

We will now analyze the main steps of the following algorithm, proving our main theorems. For simplicity, we suppose that $\mathbb{A} = k[T]$; the modifications for $\mathbb{A} = \mathbb{Z}$ follow.

Algorithm 1: Main Algorithm

Input: F, G in $\mathbb{A}[X, Y]$, $\mathfrak{m} \subset \mathbb{A}$, $\ell \in \mathbb{N}$, $d \in \mathbb{N}$

Output: $\mathcal{C} = (\mathbf{C}_1, \dots, \mathbf{C}_s)$

- 1 (1) $\mathcal{T}_1 \leftarrow Z(F \bmod \mathfrak{m}, G \bmod \mathfrak{m})$
 - 2 (2) $i \leftarrow 1$
 - 3 (3) **while** $\lambda(\mathfrak{m}^{2^i}) < 4d\ell + 48d^2$ **do**
 - 4 (3.a) $\mathcal{T}_{2^i} \leftarrow \text{Lift}(\mathcal{T}_{2^{i-1}}, F, G)$
 - 5 (3.b) $i \leftarrow i + 1$
 - 6 **end**
 - 7 (4) $\mathcal{C}_{2^{i-1}} \leftarrow \text{Convert}(\mathcal{T}_{2^{i-1}})$
 - 8 (5) **return** $\text{RationalReconstruction}(\mathcal{C}_{2^{i-1}})$
-

Step 1. Over $\mathbb{A} = k[T]$, the maximal ideal \mathfrak{m} has the form $\mathfrak{m} = \langle T - t_0 \rangle$, for some $t_0 \in k$. Reducing F and G modulo \mathfrak{m} takes $O(\ell d^2)$ operations in k by the plain algorithm.

We assume that t_0 is not a root of the polynomial A defined in Lemma 4. By assumption, the cardinality of k is at least twice the degree of A , so choosing t_0 at random, our assumption is satisfied with probability at least $1/2$.

We use the algorithm of Section 2.3 over k to compute the equiprojectable decomposition \mathcal{T}_1 of $Z(F \bmod \mathfrak{m}, G \bmod \mathfrak{m})$; under our assumption on t_0 , \mathcal{T}_1 coincides with $\mathcal{T} \bmod \mathfrak{m}$. This step takes $O(\mathbf{M}(d)\mathbf{M}(d^2) \log(d)^2)$ operations in k .

Step 3. We saw in the introduction that over either $\mathbb{A} = k[T]$ or $\mathbb{A} = \mathbb{Z}$, all polynomials U_i and N_i in \mathcal{C} satisfy $\lambda(U_i), \lambda(N_i) \leq 2d\ell + 24d^2$. In order to reconstruct the coefficients of these polynomials from their expansion modulo \mathfrak{m}^N , it is thus enough to ensure that $2(2d\ell + 24d^2) \leq \lambda(\mathfrak{m}^N)$; this accounts for the bound in the **while** loop. If we wanted to compute \mathcal{T} instead, the bound would be of order $d^3\ell + d^4$.

Step 3.a. For each value of i , we call the algorithm described in the previous subsection; we saw that the cost is $O(d^2\mathbf{M}(\ell) + \mathbf{M}(d^2)\mathbf{M}(2^i)d^{(\omega-1)/2} \log(d))$ operations in k . The last value i_0 of the loop index is such that $2^{i_0} < 4d\ell + 48d^2 \leq 2^{i_0+1}$. We deduce the total

running time:

$$O(d^2 \mathbf{M}(\ell) \log(\ell) + \mathbf{M}(d^2) \mathbf{M}(d\ell + d^2) d^{(\omega-1)/2} \log(d)).$$

Step 4. We obtain $\mathcal{C} \bmod \mathfrak{m}^{2^{i_0}}$ from $\mathcal{T} \bmod \mathfrak{m}^{2^{i_0}}$ by applying subroutine **Convert**, which does the following: for all $i \leq s$, \mathbf{T}_i has the form (U_i, V_i) and $\mathbf{C}_i = (U_i, N_i)$, with $N_i = V_i U_i' \bmod U_i$, over the ring $\mathbb{A}_{2^{i_0}}[X, Y]$. The cost is negligible compared to that of the lifting.

Step 5. Finally, **RationalReconstruction** recovers the rational coefficients appearing in \mathcal{C} from their expansion modulo $\mathfrak{m}^{2^{i_0}}$ (the index i_0 was chosen such that this precision is sufficient). There are $O(d^2)$ coefficients, each of them having numerator and denominator of degree $O(d\ell + d^2)$, so the total time is $O(d^2 \mathbf{M}(d\ell + d^2) \log(d\ell))$ operations in k .

Summary. Summing all previous costs, we see that the total time admits the upper bound claimed in Theorem 1,

$$O(\mathbf{M}(d^2) \mathbf{M}(d\ell + d^2) d^{(\omega-1)/2} \log(d\ell)).$$

Over $\mathbb{A} = \mathbb{Z}$, \mathfrak{m} is of the form $\langle p \rangle$, for a suitable p chosen as follows: let $B = 6 \cdot 8d^5(3\ell + 10\log(d) + 22)$. Using [30, Th. 18.10], we can compute in time $O(\log(d\ell))$ an integer $p \in [B+1, \dots, 2B]$ such that with probability at least $1/2$, p is prime and does not divide the integer A of Lemma 5. We apply the same algorithm as above (in particular, since $p \geq B$, the computation modulo p will satisfy the requirement on the characteristic of the field $k = \mathbb{Z}/p\mathbb{Z}$ of Proposition 1).

Using the analysis in the previous subsection and the bounds on the bit-size of the output, it is straightforward to derive an upper bound of $d^{2+\varepsilon} O(\log(d\ell))$ bit operations, for any $\varepsilon > 0$. Up to doubling ε , the polylogarithmic terms can be discarded, and we get the result of Theorem 2.

2.6 Experimental results

We report here on preliminary results obtained with an experimental implementation of our main algorithm in the case $\mathbb{A} = \mathbb{Z}$, based on Shoup's NTL [60]. Although Theorem 2 features the best complexity, it relies ultimately on an idea of Kedlaya and Umans' [36], and we are not aware of an efficient implementation of it, nor do we know how to derive one. Instead, we used the baby steps / giant steps idea underlying Theorem 1, which applies over any ring.

Our prototype is limited to inputs with word-size coefficients, and handles only the generic case described in the introduction, with only one triangular set of the form

degree	precision	Lifting	CRT, ZZ_p	CRT, lzz_p
100	32	295.67	1474.88	899.48
100	64	558.75	2949.76	1798.96
100	128	1241.4	5899.52	3597.93
120	32	421.78	2711.36	1990.40
120	64	774.14	5422.72	3980.80
120	128	1728.1	10845.44	7961.60
140	32	818.97	4902.24	2671.89
140	64	1486.35	9804.48	5343.79
140	128	3045.91	19608.96	10687.59
160	32	1072.1	7610.6	5293.64
160	64	1896.64	15221.2	10587.28
160	128	3958.17	30442.4	21174.56
180	32	1394.61	11121.48	6541.90
180	64	2399.61	22242.96	13097.57
180	128	4951.37	44485.92	26195.15

Table 2.1: Lifting vs CRT

$U(X), Y - \eta(X)$ in \mathcal{S} . We did implement some classical optimizations not described above in the lifting process, such as halving the precision needed for the Jacobian matrix [33, § 4.4]. In the size ranges below, we choose our prime p of about 50 bits (this agrees with the bound given in the previous section; also, in this generic case, it is easy to verify that such a prime is “lucky”). Our implementation does polynomial matrix multiplication with exponent $\omega = 3$. Nevertheless, this step was carefully implemented, using FFT techniques for evaluation / interpolation and fast multiplication of matrices modulo small primes.

We compare our results to a Chinese Remainder approach that computes the resultant and the last subresultant modulo many primes. NTL only computes resultants, so we used an implementation of the fast subresultant algorithm already used in [31] that mimics NTL’s built-in resultant implementation. We give timings for the two kinds of modular arithmetic supported by NTL, ZZ_p and lzz_p, for respectively “large” primes and word-size primes. The latter is usually faster, as confirmed below, but the former allows us to choose fewer but larger primes for modular computations, which may be advantageous.

The above table shows timings needed to compute the output modulo p^N , where p is a 50 bit prime, and N is a power of 2, using these various approaches; inputs are random dense polynomials, and correctness was verified by comparing that the results

of all approaches agreed. On these examples, our lifting algorithm does better than our CRT-based resultant implementation. The next step in our implementation will be to confirm whether this is still the case when we lift the general position assumption.

Chapter 3

Solving bivariate polynomial systems: the case of singular solutions

3.1 Introduction

Overview. Newton’s iterator is one of the most popular components of polynomial system solvers, either from the numeric or symbolic points of view. This iterator usually handles situations without multiplicities only, since it requires that the Jacobian matrix of the given system is invertible at the roots we are looking for.

In this paper, we are interested in applying such techniques to the solution of bivariate polynomial systems $F = G = 0$, with F and G in $\mathbb{Z}[X, Y]$. This work is in the continuation of [39], where Newton iteration techniques were used to handle solutions *without multiplicities* of system $F = G = 0$. In this work, using results and ideas from [39], as well as Lecerf’s extension of Newton’s iterator to systems with multiplicities [40], we extend this approach to all solutions.

A large body of recent work on this question [28, 25, 57, 11, 27] focuses on the problem computing real solutions of such systems, using a combination of symbolic elimination and real root isolation or approximation techniques. Our interest here is on complexity of the symbolic component of such algorithms. In a nutshell, one of our main results says that bivariate systems with integer coefficients can be solved “symbolically” in essentially optimal time.

Over an arbitrary field. Let us first discuss known results for solving a bivariate system $F = G = 0$ over $\mathbb{K}[X, Y]$, where \mathbb{K} is an abstract field. All along, we will suppose

that the zero-set $V(F, G)$ is finite. In this case, if F and G have total degree at most d , the Bézout theorem implies that the system $F = G = 0$ has at most d^2 solutions in an algebraic closure $\overline{\mathbb{K}}$ of \mathbb{K} .

Several approaches exist to describe the solutions of our system: Gröbner bases, triangular representations, or descriptions based on univariate polynomials. For instance, in [39], the authors relied on a canonical description of a zero-dimensional variety, called the equiprojectable decomposition from [23]. Although it would be natural to use this kind of description here as well, the techniques we rely on are slightly easier to apply when working in generic coordinates.

Indeed, if we are in generic coordinates, the zeros of $F = G = 0$ can simply be described by a pair of polynomials in $\mathbb{K}[X]$, of the form

$$P(X) = 0, \quad Y = S(X). \quad (3.1)$$

Remark that for such a description to make sense, no two points on $V(F, G)$ should have the same abscissa; this is precisely what is ensured once we are in generic coordinates. In such an output, our choice is to take P square-free; in other words, our representation of the solutions will forget multiplicities.

The input polynomials F and G have degree d in two variables; the polynomials P and S have degree at most d^2 in one variable. Thus, representing both input and output involves only (d^2) elements in \mathbb{K} . However, the best algorithms known so far all use at least $O(d^3)$ operations in \mathbb{K} .

Systems over the integers. In this paper, we are going to work in the particular case where $\mathbb{K} = \mathbb{Q}$; our approach would extend to cases such as $\mathbb{K} = k(T)$ (as was discussed in [39]), but one key complexity estimate still eludes us in those other cases. In such cases, it becomes crucial to take into account the bit-size of the output as well.

For a non-zero integer a , we write $\text{ht}(a) = \log(|a|)$; this essentially represents the amount of bits needed to store a . We call this the *height* of a . It will be useful to introduce a notion of height for polynomials with coefficients in \mathbb{Q} : if P is such a polynomial, the height $\text{ht}(P)$ denotes the maximum of the heights $\lambda(n_i)$ and $\lambda(d_i)$, where n_i and d_i are the numerators and denominators of the coefficients of P , when written in reduced form using a common denominator. Thus, height and degree combined give us an estimate of the total amount of bits, or machine words, \dots , needed to store P .

Suppose then that F and G have coefficients in \mathbb{Z} , with degree at most d and height at most h . Assuming that we are in generic coordinates, so that a representation of

the solutions of $F = G = 0$ makes sense, both P and S have coefficients in \mathbb{Q} . As is well-known (since at least [5, 56]), the bit-size bounds for the coefficients of S are much worse than those for P (and this is reflected in practice very accurately). Explicitly, the following results are known (we will reprove them):

$$\text{ht}(P) = O(dh + d^2), \quad \text{ht}(S) = O(d^3h + d^4).$$

The usual workaround is to replace S by another polynomial, R , defined as $R = P'S \bmod P$; equivalently, the solutions are now described by

$$P(X) = 0, \quad Y = \frac{R(X)}{P'(X)}.$$

This construction was highlighted in [5, 56], but goes back to early work of Kronecker and Macaulay. For the polynomial R , much better height bounds are known, of the form $\text{ht}(R) = O(dh + d^2)$. Thus, representing (P, R) involves $O(d^3h + d^4)$ bits.

Following Rouillier, we will call this representation the Rational Univariate Representation of $V(F, G)$. Remark that a similar construction for triangular representation is in [24].

Let us finally say a word about generic position questions. As was mentioned above, our requirement for the existence of an output such as polynomials (P, S) or (P, R) is that the coordinates X *separates* the points in $V(F, G)$. A simple change of variables of the form $X \leftarrow X + tY$ will ensure that this is the case, for almost all values of $t \in \mathbb{Z}$ (that is, all values except a finite number).

Main result. In all that follows, we will say that a solution (x, y) of the system $F = G = 0$ is *simple* if the Jacobian determinant of (F, G) is nonzero at (x, y) .

Let us temporarily let Z' be the set of such simple solutions. In [39], we gave with Lebreton an algorithm to compute a triangular representation of Z' ; this algorithm could be adapted to give a univariate representation, after putting the equations in generic coordinates.

In a nutshell, the idea is to compute the output modulo a prime p , then *lift* this representation modulo powers of p using a suitable form of Newton iteration. Looking only at points in Z' makes it straightforward to apply such techniques, since by assumption, at such points, the Jacobian matrix of (F, G) is invertible.

In this paper, we show that we can extend these ideas to find a univariate representation of $V(F, G)$, in a time close to optimal; this matches the results of [39] in the case

where all solutions of $F = G = 0$ are simple. The algorithm uses a combination of the lifting techniques introduced by Lecerf in [40] and a modification of the Kedlaya-Umans modular composition algorithm [36], following the same strategy as in [39].

Our algorithm is probabilistic of the Monte Carlo kind: one can choose an arbitrary error threshold, say $1/2^P$, and the algorithm guarantees that the result is correct with probability at least $1 - 1/2^P$. Part of the randomness simply amounts to choosing an integer in a finite set. Another component is more involved, as it amounts to choosing primes. Since this is a delicate question in itself, and not the topic of this paper, we will use the following device: we assume that we are given an oracle \mathcal{O} , which takes as input an integer B , and returns a prime number in $\{B + 1, \dots, 2B\}$, uniformly distributed within this set of primes.

Theorem 4. *Let $F, G \in \mathbb{Z}[X, Y]$ with degree at most d and height at most h , that have no nontrivial factor in $\mathbb{Q}[X, Y]$.*

For any $\varepsilon > 0$, there exists an algorithm with the following characteristics. Given $P \geq 1$, the algorithm computes the Rational Univariate Representation of the system $F_t = G_t = 0$, where t is an integer of height $O(P + \log(d))$, $F_t = F(X + tY, Y)$ and $G_t = G(X + tY, Y)$. The running time is $d^{3+\varepsilon}O^\sim((h + d)P)$ bit operations, and the probability of success is at least $1 - 1/2^P$.

For fixed P , this is thus $d^{3+\varepsilon}O^\sim(h + d)$ bit operations, which almost matches the known upper bounds on the output size. We are not aware of previous result that would be comparable in terms of complexity. As reviewed in [39], previous approaches to this question had cost at least $O^\sim(d^4h + d^5)$, for Monte Carlo algorithms. For Las Vegas algorithms, the known bounds are higher yet, namely $O^\sim(d^5h + d^6)$, see [15] for latest results in this context.

For the case where our base ring is $\mathbb{A} = k[T]$ instead of \mathbb{Z} , we were not able to obtain the same result as in [39, Theorem 1], as we were not able to find a fast algorithm for one specific question of modular composition appearing in the process of evaluating the derivatives of the given system. However, the approach given for the case $\mathbb{A} = \mathbb{Z}$ remains correct in the case of $\mathbb{A} = k[T]$ as our base ring.

Multiplicities. Before explaining how our result compares to previous work, it will be useful for us to recall the definition of the *multiplicity* of an isolated solution of a polynomial system. We will only need to discuss systems in one or two variables. As per the above convention, our base ring in this discussion is a domain \mathbb{A} with fraction field \mathbb{K} , and $\overline{\mathbb{K}}$ is an algebraic closure of \mathbb{K} . To give the most general definition, we assume that our polynomials have coefficients in $\overline{\mathbb{K}}$.

First, consider a nonzero univariate polynomial F in $\overline{\mathbb{K}}[X]$, and a root x^* of F . The *multiplicity* of F at x^* is the highest integer M such that $(X - x^*)^M$ divides F . The multiplicity M is one if and only if $F'(x^*)$ is nonzero.

Next, consider an ideal ψ in $\overline{\mathbb{K}}[X, Y]$, and an isolated solution (x^*, y^*) of the system of equations $\psi = 0$. Define the ideal $\psi' = \{f(X + x^*, Y + y^*) \mid f \in \psi\}$, so that $(0, 0)$ is a solution of the system $\psi' = 0$. Then, the multiplicity M of the system ψ at (x^*, y^*) is the dimension of the $\overline{\mathbb{K}}$ -vector space $\overline{\mathbb{K}}[[X, Y]]/\psi'$, see for instance [21, Chapter 4]; the fact that (x, y) is an isolated solution is equivalent to this dimension being finite. If $\psi = \langle F, G \rangle$, the multiplicity M is one if and only if the Jacobian determinant of (F, G) is nonzero at (x^*, y^*) .

It will also be useful to remember the following extension of the Bézout bound on the number of isolated solutions of a bivariate system: if (F, G) is a system in $\overline{\mathbb{K}}[X, Y]$, with both F and G having degree at most d , then the sum of the multiplicities of the isolated solutions of the system $F = G = 0$ is at most d^2 . Examples such as $F = X^d, G = Y^d$ show that multiplicities as high as d^2 are possible.

Lecerf's deflation algorithm. Our main idea is to apply lifting techniques to the triangular sets $\mathbf{T}_1, \dots, \mathbf{T}_s$: given these polynomials modulo an ideal \mathfrak{m} of \mathbb{A} , we intend to compute them modulo successive powers \mathfrak{m}^k ; when k is large enough, we deduce $\mathbf{T}_1, \dots, \mathbf{T}_s$, or rather $\mathbf{C}_1, \dots, \mathbf{C}_s$, since we saw that the latter have better size bounds.

However, the algorithm for lifting triangular sets from [59] requires that all points our triangular sets define are simple solutions of the input system (F, G) ; this is why the algorithm of [39] was restricted to such points.

In order to handle all solutions, including the multiple ones, one has to remember that the lifting algorithm from [59] is a variation around Newton iteration. Thus, it makes sense to attempt to use variants of Newton iteration techniques for multiple roots.

Our approach is based on a result of Lecerf's [40], which generalizes the usual Newton iterator to multiple roots, in our context of \mathfrak{m} -adic lifting (that is, of lifting modulo the powers of a maximal ideal \mathfrak{m} in the domain \mathbb{A}). However, this algorithm does not deal with our problem of lifting triangular sets; instead, in line with the classical presentation of Newton iteration, it deals with iterative approximations of a unique isolated root x^* of a polynomial system ψ , that has multiplicity $M > 1$.

The main idea behind this approach is classical: it boils down to replacing the given polynomial system ψ with a new one, say $\tilde{\psi}$, such that for the given root x^* of ψ with multiplicity M , x^* is still a root of the new system $\tilde{\psi}$, but with multiplicity $\tilde{M} < M$. The ideal generated by the new polynomial system $\tilde{\psi}$ is usually called a *deflation* ideal. We

can then find a new deflated system $\tilde{\psi}$ in which x^* is a non-singular root, by repeating the process sufficiently many times.

Lecerf proved that his construction of deflation ideals leads to an iteration with quadratic convergence, with an overhead that grows like the square of the multiplicity M of the root.

Our main technical contribution in this paper lies in the adaptation of this result to our context of lifting triangular sets, with an admissible complexity. One difficulty lies in the very fact that we are lifting whole triangular sets, not only a single root. In addition, Lecerf's algorithm and cost analysis assume that the input polynomials are given by a straight-line program, which is not the case for us. Finally, we saw that the cost of Lecerf's algorithm is quadratic in the multiplicity; we saw above that multiplicities as large as d^2 are possible for a bivariate system of degree d , so we will have to show that the cost can actually be reduced to quasi-linear in M . On the other hand, the fact that we consider only bivariate systems simplifies considerably the constructions.

Other deflation algorithms. Generalizing Newton iteration to singular situations, and in particular designing an efficient iterator with quadratic convergence in degenerate cases, are still ongoing challenging problems. We briefly review some of the previous work on this question. Remark that all algorithms below work for an arbitrary number of variables, not only bivariate systems.

In order to be develop Newton-type methods that converge to multiple roots, a common idea is to use *deflation techniques*, which consist in adding new equations in order to reduce the multiplicity.

An early result in this area is due to Ojika, Watanabe, and Mitsui [61]: by applying a triangulation preprocessing step on the Jacobian matrix at the approximate root, minors of the Jacobian matrix are added to the system to reduce the multiplicity.

In [2, 1], instead of triangulating the Jacobian matrix, the number of variables is doubled and new equations are introduced, which are linear in the new variables; it is proved that the multiplicity decreases through this process. In [8], this construction is related to Macaulay's *inverse systems*; Macaulay [47] dialytic method is revisited for this purpose. These deflation methods are applied iteratively until the root becomes simple, doubling each time the number of variables. Other algorithms for the construction of inverse systems are described e.g. in [45], reducing the size of the intermediate linear systems, or in [49] using an integration method.

In [52], a minimization approach is used to reduce the value of the equations and their derivatives at the approximate root, assuming a basis of the inverse system is known.

In [68], the inverse system is constructed via Macaulays method; multiplication table of the local algebras are deduced and their eigenvalues are used to improve the approximated root. It is proved that the convergence is quadratic when the Jacobian has co-rank one at the multiple root.

Unfortunately, even when the input system is bivariate, it seems difficult to control the complexity of the above algorithms. In addition, several of these results rely on purely numerical techniques, such as the Singular Value Decomposition, which will not carry over to our context. This explains why we rely on the approach of [40].

Notation. For positive integers m, n , and for a ring \mathbb{A} , $\mathbb{A}[X]_m$ denotes the set of all $F \in \mathbb{A}[X]$ such that $\deg(F) < m$, and $\mathbb{A}[X, Y]_{m,n}$ the set of all $F \in \mathbb{A}[X, Y]$ such that $\deg(F, X) < m$ and $\deg(F, Y) < n$.

3.2 Quantitative estimates

3.2.1 Polynomials in general position

In this subsection, we describe a classical notion of system in *general position*, and we discuss conditions that ensure that this property is preserved through reduction at a prime. These results are in essence classical (they go back to Kronecker and Macaulay), and their quantitative versions appear for instance in [37, 59, 23, 25, 16], among many other references. Nevertheless, we chose to give short self-contained proofs of the facts we need.

In all this section, π denotes the mapping $(x, y) \mapsto x$ of projection on the first factor; although the points x, y will be taken in various fields, we keep the same notation throughout, since no ambiguity can arise. In the beginning of this section, \mathbb{A} is a domain with field of fractions \mathbb{K} ; we let $\overline{\mathbb{K}}$ denote an algebraic closure of \mathbb{K} .

Representing zero-dimensional algebraic sets. Let $V \subset \overline{\mathbb{K}}^2$ be a finite set, and assume that V can be written as $V = V(F_1, \dots, F_t)$ for some F_1, \dots, F_t in $\mathbb{K}[X, Y]$. Suppose that the following conditions are satisfied:

- G₁. \mathbb{K} is perfect.
- G₂. X is a *separating element* for V , that is, the restriction of π to V is one-to-one.

Under these assumptions, there exist uniquely defined polynomials (P, S) in $\mathbb{K}[X]$, with P squarefree and monic, and S of degree less than P , such that the ideal $\langle P(X), Y - S(X) \rangle$

is the defining ideal of V in $\mathbb{K}[X, Y]$ (remark that the existence of such polynomials with coefficients in \mathbb{K} requires that \mathbb{K} be perfect).

Following [32], we call polynomials (P, S) the *Shape Lemma representation* of V , and denote them by $(P, S) = \text{SL}(V)$. Over a field such as $\mathbb{K} = \mathbb{Q}$, it is well known that this representation suffers from coefficient size bloat: the coefficients of S are in many cases significantly larger than those of P . A workaround is to use an alternative description, the *Rational Univariate Representation* of V , for which this issue usually disappears. It consists in polynomials (P, R) , with $R = SP' \bmod P \in \mathbb{K}[X]$; we denote these polynomials by $(P, R) = \text{RUR}(V)$.

The notion of Rational Univariate Representation is from [5, 56]; note the original definition is able to incorporate multiplicities, which we do not take into consideration here.

Polynomials in general position. Let F and G be in $\mathbb{A}[X, Y]$ and let $V = V(F, G) \subset \overline{\mathbb{K}}^2$. We say that F, G are in *general position* if the following holds:

H₁. F and G have no common factor in $\mathbb{K}[X, Y]$, so V is finite.

H₂. The leading coefficients f and g of respectively F and G with respect to Y are in \mathbb{A} .

H₃. V satisfies G_1 and G_2

When this is the case, polynomials P , S and R associated to V as above are well-defined; the polynomial P appearing in the Shape Lemma representation of V is the squarefree part of the resultant of F and G with respect to Y (once the latter has been made monic). As a matter of notation, we will write $(P, S) = \text{SL}(F, G)$ and $(P, R) = \text{RUR}(F, G)$.

For t in \mathbb{A} , we will denote by F_t and G_t the polynomials $F_t = F(X + tY, Y)$ and $G_t = G(X + tY, Y)$; similarly, we will write $V_t = V(F_t, G_t)$, so that

$$V_t = \{(x, y) \in \overline{\mathbb{K}}^2 \mid (x + ty, y) \in V\} = \phi_t(V),$$

where ϕ_t is the mapping $\overline{\mathbb{K}}^2 \rightarrow \overline{\mathbb{K}}^2$ given by $\phi_t(x, y) = (x - ty, y)$. Letting T be an indeterminate over \mathbb{A} , we use the same notation, using a subscript T instead of t , to denote the polynomials

$$F_T = F(X + TY, Y) \quad \text{and} \quad G_T = G(X + TY, Y),$$

and their zero-set V_T in $\overline{\mathbb{K}(T)}^2$ (actually, V_T lies in $\overline{\mathbb{K}(T)}^2 \subset \overline{\mathbb{K}(T)}^2$). If F and G are polynomials in $\mathbb{A}[X, Y]$, with no common factor (so they satisfy \mathbf{H}_1), one easily verifies that F_T and G_T satisfy \mathbf{H}_1 and \mathbf{H}_2 , that V_T satisfies \mathbf{G}_2 , over the ring $\mathbb{A}[T]$ instead of \mathbb{A} , and that V_T has the same cardinality as V .

Over the integers. Let us now restrict our attention to the case $\mathbb{A} = \mathbb{Z}$ and $\mathbb{K} = \mathbb{Q}$; as before, we take F and G that satisfy \mathbf{H}_1 and we write $V = V(F, G)$.

Let then \mathfrak{A} be the resultant of F_T and G_T with respect to Y ; this is a nonzero polynomial in $\mathbb{Z}[T, X]$, and we denote by $\mathfrak{a} \in \mathbb{Z}[T]$ its leading coefficient with respect to X . Let next $\mathfrak{B} \in \mathbb{Z}[T, X]$ be the *squarefree part* of \mathfrak{A} , that is, $\mathfrak{B} = \mathfrak{A} / \gcd(\mathfrak{A}, \mathfrak{A}')$, where \mathfrak{A}' is the derivative of \mathfrak{A} with respect to X ; remark that the gcd, and thus \mathfrak{B} itself, are formally defined only up to sign, but this will be inconsequential.

Lemma 1. In $\overline{\mathbb{Q}}[T, X]$, \mathfrak{B} factors as

$$\mathfrak{B} = \mathfrak{b} \prod_{(x,y) \in V} (X - (x - Ty)), \quad (3.2)$$

where \mathfrak{b} is the leading coefficient of \mathfrak{B} with respect to X , and belongs to $\mathbb{Z}[T]$.

PROOF. In $\overline{\mathbb{Q}}(T)[X]$, the roots of \mathfrak{B} are the values $(x - Ty)$, for $(x, y) \in V$, so there is indeed a factorization of the form $\mathfrak{B} = \mathfrak{b} \prod_{(x,y) \in V} (X - (x - Ty))$. *A priori*, \mathfrak{b} is in $\overline{\mathbb{Q}}(T)$, but the right-hand side expression shows that it is the leading coefficient of \mathfrak{B} with respect to X . Since \mathfrak{B} is in $\mathbb{Z}[T, X]$, \mathfrak{b} is in $\mathbb{Z}[T]$.

The following lemma shows how the polynomial \mathfrak{B} and its factors allows us to give formula for the Rational Univariate Representation of V_t and its subsets, when F_t and G_t are in general position. To state this lemma, remark that if W is a subset of V , we may rewrite the factorization in (3.2) as

$$\mathfrak{B} = \mathfrak{b} \mathfrak{C}_W \mathfrak{C}_{W^c},$$

where we write $W^c = V \setminus W$ and

$$\mathfrak{C}_W = \prod_{(x,y) \in W} (X - (x - Ty)) \quad \text{and} \quad \mathfrak{C}_{W^c} = \prod_{(x,y) \in W^c} (X - (x - Ty)).$$

If in addition W is defined over \mathbb{Q} (for instance, for $W = V$), both \mathfrak{C}_W and \mathfrak{C}_{W^c} are in $\mathbb{Q}[T, X]$.

Lemma 2. *Let W be a subset of V defined over \mathbb{Q} . Then, \mathfrak{C}_W can be written as $\mathfrak{D}_W/\mathfrak{c}_W$, where \mathfrak{D}_W is a primitive polynomial in $\mathbb{Z}[T, X]$ that divides \mathfrak{B} in $\mathbb{Z}[T, X]$, and \mathfrak{c}_W is a nonzero integer that divides the content of \mathfrak{b} .*

PROOF. Start from the factorization $\mathfrak{B} = \mathfrak{b} \mathfrak{C}_W \mathfrak{C}_{W^c}$, which holds between polynomials in $\mathbb{Q}[T, X]$. Since \mathfrak{C}_W and \mathfrak{C}_{W^c} are in $\mathbb{Q}[T, X]$ and are monic in X , they can be written as $\mathfrak{C}_W = \mathfrak{D}_W/\mathfrak{c}_W$ and $\mathfrak{C}_{W^c} = \mathfrak{D}_{W^c}/\mathfrak{c}_{W^c}$, with \mathfrak{c}_W and \mathfrak{c}_{W^c} in \mathbb{Z} , and \mathfrak{D}_W and \mathfrak{D}_{W^c} primitive in $\mathbb{Z}[T, X]$. Similarly, we can write $\mathfrak{b} = rs$, where $r \in \mathbb{Z}$ is the content of \mathfrak{b} and s is primitive in $\mathbb{Z}[T]$, and $\mathfrak{B} = RS$, where $R \in \mathbb{Z}$ is the content of \mathfrak{B} and S is primitive in $\mathbb{Z}[T, X]$.

Clearing denominators, we obtain $\mathfrak{c}_W \mathfrak{c}_{W^c} RS = rs \mathfrak{D}_W \mathfrak{D}_{W^c}$. Using Gauss' Lemma over $\mathbb{Z}[T, X]$, we deduce that $\mathfrak{c}_W \mathfrak{c}_{W^c} R = r$; this implies that \mathfrak{c}_W divides r , as claimed. We also obtain $S = s \mathfrak{D}_W \mathfrak{D}_{W^c}$, so that \mathfrak{D}_W divides S in $\mathbb{Z}[T, X]$, and thus \mathfrak{B} .

The explicit factorization of \mathfrak{B} and of the polynomials \mathfrak{C}_W allows us to give formula for the Rational Univariate Representation of V , or of one of its subsets W .

Lemma 3. *Let $t \in \mathbb{Z}$ be such that $\mathfrak{b}(t) \neq 0$ and such that F_t and G_t are in general position. Let further W be a subset of V , defined over \mathbb{Q} . Then, W_t is in general position, and the associated Rational Univariate Representation $(P_{W_t}, R_{W_t}) = \text{RUR}(W_t)$ is given by*

$$P_{W_t} = \mathfrak{C}_W(t, X) \quad \text{and} \quad R_{W_t} = \frac{\partial \mathfrak{C}_W}{\partial T}(t, X).$$

In addition, all denominators appearing in P_{W_t} and R_{W_t} divide $\mathfrak{b}(t)$.

PROOF. The proof of the first properties is classical, see for instance [5, 56, 59], which actually apply in more general cases. The last property follows from the previous lemma, together with the remark that the content of \mathfrak{b} divides $\mathfrak{b}(t)$, for any integer t .

Let now $\Delta \in \mathbb{Z}[T]$ be the discriminant of \mathfrak{B} with respect to X and define finally Γ as the product of Δ by the leading coefficients \mathfrak{f} and \mathfrak{g} of respectively F_T and G_T with respect to Y , and by the leading coefficient \mathfrak{a} of \mathfrak{A} in X . This is a nonzero element of $\mathbb{Z}[T]$.

The following lemma gives upper bounds on the degree and height of Δ and of the various polynomials \mathfrak{C}_W , for W a subset of V . These bounds are far from sharp as far as the constants involved are concerned, in particular in terms of degrees, but this will be harmless for the overall cost analysis.

Lemma 4. *Suppose that F and G have degree at most d and height at most h . Then, the following holds:*

- for any subset W of V defined over Q , the polynomial $\mathfrak{C}_W \in \mathbb{Q}[T, X]$ satisfies $\deg(\mathfrak{C}_W, T) \leq d^2$ and $\text{ht}(\mathfrak{C}_W) \leq 2hd + 16d^2$.
- Γ satisfies $\deg(\Gamma) \leq 12d^4$ and $\text{ht}(\Gamma) \leq 12hd^3 + 120d^4$.

PROOF. The polynomials F_T and G_T have degree at most d in Y and (X, T) and, by [37, Lemma 1.2.1.c], height at most $h + 4d$. Their resultant \mathfrak{A} has total degree at most $2d^2$, and the same lemma as above (and the remark following it) implies that its height is at most $2hd + 12d^2$. The factor bound of [37, Lemma 1.2.1.d] then implies that \mathfrak{B} has degree at most $2d^2$ and height at most $2hd + 16d^2$.

From this, we can prove the bounds we claim on \mathfrak{C}_W . The degree bound in T is obvious, since at most d^2 linear factors appear in the product giving \mathfrak{C}_W . For the height bound, remark that \mathfrak{C}_W and the polynomial \mathfrak{D}_W defined in Lemma 2 have the same height, and that \mathfrak{D}_W divides \mathfrak{B} , and thus \mathfrak{A} in $\mathbb{Z}[T, X]$. Thus, they admit the same height bound as \mathfrak{B} itself.

On the other hand, applying again the remark following [37, Lemma 1.2] to \mathfrak{B} , the degree and height bounds for this polynomial imply that Δ has degree at most $8d^4$ and height at most $8hd^3 + 80d^4$. Multiplying by the leading coefficients \mathfrak{f} and \mathfrak{g} of respectively F_T and G_T in Y , and by the leading coefficient \mathfrak{a} of \mathfrak{A} in X , which are all in $\mathbb{Z}[T]$, we deduce that Γ has degree at most $12d^4$ and height at most $12hd^4 + 120d^4$, using [37, Lemma 1.2.1.b].

The following specialization lemma shows how Δ controls (in particular) prime of bad reduction. We give it in the general form we will need below.

Lemma 5. *Let ϕ be a ring morphism $\mathbb{Z}[T] \rightarrow \mathbb{A}$, where \mathbb{A} is a domain; this morphism extends to a ring morphism $\phi : \mathbb{Z}[T, X, Y] \rightarrow \mathbb{A}[X, Y]$.*

Let $V' = V(\phi(F_T), \phi(G_T)) \subset \overline{\mathbb{K}}^2$, where $\overline{\mathbb{K}}$ is an algebraic closure of the fraction field \mathbb{K} of \mathbb{A} . Then, if $\phi(\Gamma)$ is nonzero, $\phi(F_T)$ and $\phi(G_T)$ satisfy \mathbf{H}_1 and \mathbf{H}_2 , and the cardinality of $\pi(V') \subset \overline{\mathbb{K}}$ is equal to the cardinality of V .

PROOF. First, let us establish that the cardinality of $\pi(V')$ is the number of pairwise distinct roots of $\phi(\mathfrak{A})$ in $\overline{\mathbb{K}}$. Because $\phi(\mathfrak{f})$ and $\phi(\mathfrak{g})$ are nonzero, they remain the leading terms of respectively $\phi(F_T)$ and $\phi(G_T)$ with respect to Y (which proves \mathbf{H}_2); in addition, the resultant $\text{res}(\phi(F_T), \phi(G_T), Y)$ coincides with the image $\phi(\mathfrak{A})$. On the other hand, because $\phi(\mathfrak{f})$ and $\phi(\mathfrak{g})$ are nonzero, the number of pairwise distinct roots of $\text{res}(\phi(F_T), \phi(G_T), Y)$ is the cardinality of $\pi(V') \subset \overline{\mathbb{K}}$. Our claim above is thus proved.

In addition, since $\phi(\Gamma)$ is nonzero, $\phi(\mathfrak{a})$ is nonzero, where \mathfrak{a} is the leading coefficient of \mathfrak{A} with respect to X (recall that \mathfrak{a} divides Γ). Thus, $\phi(\mathfrak{A})$ itself is nonzero, which

implies that $\text{res}(\phi(F_T), \phi(G_T), Y)$ is nonzero. As a result, $\phi(F_T)$ and $\phi(G_T)$ can only have a common factor in $\mathbb{A}[X]$. However, since their leading coefficients with respect to Y lie in \mathbb{A} , they have no such common factor. This proves H_1 .

Since $\mathfrak{B} = \mathfrak{A}/\text{gcd}(\mathfrak{A}, \mathfrak{A}')$, we deduce that \mathfrak{B} divides \mathfrak{A} , and that \mathfrak{A} divides some polynomial of the form $\mathfrak{a}\mathfrak{B}^k$ in $\mathbb{Z}[T, X]$, for some positive integer k . This relationship remains true through ϕ ; this implies that $\phi(\mathfrak{B})$ and $\phi(\mathfrak{A})$ have the same roots in $\overline{\mathbb{K}}$.

Now, we claim that because $\phi(\Delta)$ is nonzero, $\phi(\mathfrak{B})$ has no multiple root in $\overline{\mathbb{K}}$. Indeed, the leading term \mathfrak{b} of \mathfrak{B} divides its discriminant Δ , so $\phi(\mathfrak{b})$ must be nonzero. This implies that the resultant computation that gives $\Delta = \text{res}(\mathfrak{B}, \mathfrak{B}', X)$ carries over through ϕ , so that $\phi(\Delta)$ is the discriminant of $\phi(\mathfrak{B})$. Our claim above is thus proved.

This implies that the number of roots $\phi(\mathfrak{B})$, or equivalently of $\phi(\mathfrak{A})$, is equal to the degree of $\phi(\mathfrak{B})$. Since $\phi(\mathfrak{b})$ is nonzero, this degree is the degree of \mathfrak{B} in X , which is the cardinality of V , by Eq. (3.2). We are done.

Corollary 1. *Take F and G as above, that satisfy H_1 , with degree at most d and height at most h . Then, the following holds.*

- For t in \mathbb{Z} , if $\Gamma(t)$ is nonzero, then F_t and G_t are in general position.
- For t as above, if t has height at most ℓ , and if W is a subset of V defined over \mathbb{Q} , then the polynomials $(P_{W_t}, R_{W_t}) = \text{RUR}(W_t)$ have degree bounded by d^2 and height bounded by an explicitly computable integer $B_{\text{RUR}}(d, h, \ell) = O^{\sim}(hd + d^2\ell)$. The polynomial S_{W_t} appearing in $\text{SL}(W_t)$ has degree at most d^2 and height bounded by an explicitly computable integer $B_{\text{SL}}(d, h, \ell) = O^{\sim}(hd^4 + d^4\ell)$.

In particular, the polynomials in $\text{RUR}(F_t, G_t)$ satisfy these bounds.

- Let in addition p be a prime. If $\Gamma(t) \bmod p$ is nonzero, then $F_t \bmod p$ and $G_t \bmod p$ are in general position, and the leading terms of $F_t \bmod p$ and $G_t \bmod p$ with respect to Y are the images of those of F_t and G_t modulo p .
- For t and p as above, for any subset W of V defined over \mathbb{Q} , p cancels no denominator in either $\text{SL}(W_t)$ or $\text{RUR}(W_t)$. In addition, we have

$$\text{SL}(F_t, G_t) \bmod p = \text{SL}(F_t \bmod p, G_t \bmod p)$$

and

$$\text{RUR}(F_t, G_t) \bmod p = \text{RUR}(F_t \bmod p, G_t \bmod p).$$

PROOF. Suppose that $t \in \mathbb{Z}$ is such that $\Gamma(t)$ is nonzero. Properties \mathbf{H}_1 and \mathbf{G}_1 clearly hold for F_t and G_t . Applying the previous lemma to $\phi : \mathbb{Z}[T] \rightarrow \mathbb{Z}$ given by $\phi(f) = f(t)$, we deduce that \mathbf{H}_2 holds for F_t and G_t , and that the cardinality of $\pi(V_t) \subset \overline{\mathbb{Q}}$ is equal to the cardinality of V . Since V and V_t have the same cardinality, this proves that V_t satisfies \mathbf{G}_2 . Thus, F_t and G_t are in general position.

To prove the second item, recall the formula for (P_{W_t}, R_{W_t}) given in Lemma 3. Using the bounds on the degree and height of \mathfrak{C}_W given in Lemma 4, together with the bounds for evaluation given in [37, Lemma 1.2.1], a few simplifications show that upon evaluation at $T = t$, the height of \mathfrak{C}_W and its derivative with respect to T remains at most $4hd + 40d^2 + 4d^2\ell$.

From this, we can deduce bounds for $S_{W_t} = R_{W_t}/P'_{W_t} \bmod P_{W_t}$, by applying the Hadamard bound to the Sylvester matrix associated to P_{W_t} and P'_{W_t} , followed by the analysis of the height growth induced by multiplication by R_{W_t} modulo P_{W_t} . The most significant factor here is the Hadamard bound, which induces an height overhead of $O(d^2)$ compared to the bounds for R_{W_t} . This proves the second item.

Suppose next that the prime p is such that $\Gamma(t) \bmod p$ is nonzero. Consider first $\phi' : \mathbb{Z}[T] \rightarrow \mathbb{F}_p[T]$ given by $\phi'(f) = f \bmod p$. Because $\Gamma(t) \bmod p$ is nonzero, we have in particular that $\phi'(\Gamma)$ is nonzero.

Let then V'_t be the zero-set of $F_t \bmod p$ and $G_t \bmod p$ (in an algebraic closure of \mathbb{F}_p) and V'_T be the zero-set of $F_T \bmod p$ and $G_T \bmod p$ (in an algebraic closure of $\mathbb{F}_p[T]$). Because evaluation of T at t commutes with reduction modulo p , we deduce that $|V'_T| = |V'_t|$. As pointed out previously, V'_T satisfies \mathbf{G}_2 , so that $|\pi(V'_T)| = |V'_T| = |V'_t|$. On the other hand, applying the previous lemma to ϕ' implies that the cardinality of $|\pi(V'_t)|$ is equal to $|V|$. We deduce that $|V| = |V'_t|$.

Consider now the mapping $\phi'' : \mathbb{Z}[T] \rightarrow \mathbb{F}_p[T]$ given by $\phi''(f) = f(t) \bmod p$. Applying the previous lemma to ϕ'' , we deduce that $F_t \bmod p$ and $G_t \bmod p$ satisfy \mathbf{H}_1 and \mathbf{H}_2 , and that $|\pi(V'_t)| = |V|$. Since we saw above that $|V| = |V'_t|$, this proves that $|\pi(V'_t)| = |V'_t|$, so that $F_t \bmod p$ and $G_t \bmod p$ are in general position. This proves the third item.

To conclude, consider again a subset W of V , defined over \mathbb{Q} , together with the formula that yield $(P_{W_t}, R_{W_t}) = \text{RUR}(W_t)$. Let us also simply write $(P_t, R_t) = \text{RUR}(V_t)$ and $(P_t, S_t) = \text{SL}(V_t)$, forgetting the index V .

Again, we use the fact (established in the proof of the previous lemma) that $\mathfrak{b}(t)$ is nonzero modulo p . Using Lemma 3, this proves that none of the denominators of the coefficients of either P_{W_t} or R_{W_t} vanishes at p . On the other hand, $\mathfrak{B}(t, X)$, or equivalently $P_t(X)$, remains squarefree modulo p (because $\Gamma(t)$ does not vanish mod p), so this is the case as well for the polynomial \mathfrak{C}_W appearing in the Shape Lemma

representation of V_t . We deduce that the computation of the polynomial S_{W_t} appearing in $(P_{W_t}, S_{W_t}) = \text{SL}(W_t)$, given by $S_{W_t} = R_{W_t}/P'_{W_t} \bmod P_{W_t}$, specializes well modulo p .

Notice that the polynomials F_t and G_t reduce to zero modulo $\langle P_t(X), Y - S_t(X) \rangle$. This relationship remains true modulo p , so that the polynomials $(P_t(X) \bmod p, Y - S_t(X) \bmod p)$ define a subset of $V(F_t \bmod p, G_t \bmod p)$. However, both sets have the same cardinality $|V|$, so these sets are equal. By uniqueness, we conclude that $\text{SL}(F_t, G_t) \bmod p = \text{SL}(F_t \bmod p, G_t \bmod p)$; multiplying by $P'_t \bmod P_t$, this carries over to $\text{RUR}(F_t, G_t) \bmod p = \text{RUR}(F_t \bmod p, G_t \bmod p)$. The proof is complete.

3.2.2 Non-vanishing conditions

Let \mathbb{K} be a field, let P and S be in $\mathbb{K}[X]$, with P monic of degree e , and S of degree less than e . Consider a further polynomial H in $\mathbb{K}[X, Y]$, and assume that the following properties hold:

C_0 . P is squarefree.

C_1 . H vanishes nowhere on the set $V = V(P(X), Y - S(X))$.

In this short section, we focus on the case $\mathbb{K} = \mathbb{Q}$. Assuming that H has integer coefficients, we give conditions under which these two properties are maintained through reduction at a prime p .

Proposition 1. *There exists an explicitly computable function $\Delta_1(d, h, e, \ell) = (dhe\ell)^{O(1)}$ such that the following holds.*

Suppose that P and S have degree at most e and height at most ℓ , and that $H \in \mathbb{Z}[X, Y]$ has degree at most d and height at most h . If (P, S, H) satisfy C_0 and C_1 , there exists a nonzero integer δ_1 such that:

- δ_1 has height at most $\Delta_1(d, h, e, \ell)$;
- for any prime p that does not divide δ_1 , $P \bmod p$ and $S \bmod p$ are well-defined, and $(P, S, H) \bmod p$ satisfy C_0 and C_1 over \mathbb{F}_p .

The proof of this result will occupy the rest of this section. Let c_P and c_S be common denominators respectively P and S , so that we can write $P = P^*/c_P$ and $S = S^*/c_S$, with P^* and S^* in $\mathbb{Z}[X]$. Remark that we can take c_P and c_S of height at most ℓ and that the same holds for P^* and S^* .

Suppose that p is a prime that does not divide c , and such that P remains squarefree modulo p . Thus, C_0 is maintained through reduction at such a prime.

Starting from $H = \sum_{i,j} h_{i,j} X^i Y^j$, let us then define the polynomial with integer coefficients

$$\tilde{H} = \sum_{i \leq d, j \leq d} c^{d-j} h_{i,j} X^i Y^j,$$

so that $K(X) = c^d H(X, S)$ satisfies $K(X) = \tilde{H}(X, S^*)$. By assumption C_1 , this polynomial is coprime with P , and C_1 holds modulo p if K and P remain coprime modulo p . This is the case as soon as p does not divide the resultant of K and P^* , which is a nonzero integer. Thus, we can define δ_1 as

$$\delta_1 = c \operatorname{res}(P^*, P^{*'}, X) \operatorname{res}(P^*, K, X).$$

It remains to estimate the height of this integer. First, recall that c has height at most ℓ and that the same holds for P^* and S^* . This implies that $P^{*'}$ has height at most $2\ell + \log(e)$.

- The matrix giving the resultant $\operatorname{res}(P^*, P^{*'}, X)$ has size at most $2e$ and integer entries of height at most $2\ell + \log(e)$. Hence, its determinant is a nonzero integer of height at most $4e\ell + 4e \log(2e)$.
- The polynomial $K = \tilde{H}(X, S^*)$ is obtained by evaluating a polynomial of degree at most d in 2 variables, with coefficients of height at most $h + d\ell$, at univariate polynomials of degree at most e and height at most ℓ . Using Lemma 1.2.1.c in [37], we deduce that K has degree at most de and height at most $h + d(2\ell + e + 1)$.
- As a result, the matrix giving the resultant $\operatorname{res}(P^*, K, X)$ has for determinant a nonzero integer of height at most $2de \log(2de) + eh + de(2\ell + e + 1) + de(h + d\ell)$.

Adding all estimates gives an explicit formula for the upper bound Δ_1 , which is easily seen to be polynomial in d, h, e, ℓ .

3.2.3 Conservation of intersection multiplicity

Our context in this section is similar to the one of the previous section. We still consider a field \mathbb{K} , P and S in $\mathbb{K}[X]$, with P monic of degree e , and S of degree less than e . Now, we also take two further polynomials H, K in $\mathbb{K}[X, Y]$, not necessarily coprime, and we assume that the following properties hold:

M_0 . P is squarefree.

M_1 . All points in $V = V(P(X), Y - S(X))$ are isolated points of $V(H, K)$.

We are interested in describing situations under which the following extra property is verified:

$M_2(n)$. There exists $n \geq 1$ such that for all (x, y) in V , $\langle H, K \rangle$ has multiplicity n at (x, y) .

Define the new polynomials $G = \gcd(H, K)$, $H' = H/G$ and $K' = K/G$. Then, the points in V are still isolated points of $V(H', K')$, and for $(x, y) \in V$, the intersection multiplicities $\mu((H, K), (x, y))$ and $\mu((H', K'), (x, y))$ are the same.

Intersection multiplicity is invariant through linear change of coordinates. Thus, reusing the notation of Subsection 3.2.1, we deduce that for any value of t in \mathbb{K} , and for (x, y) in V , the equality $\mu((H', K'), (x, y)) = \mu((H'_t, K'_t), (x - ty, y))$ holds. Consider then a value t such that H'_t and K'_t are in general position.

In particular, for such a t , since V_t is a subset of $V(H'_t, K'_t)$ of cardinality e , there exist polynomials $P_{[t]}$ and $S_{[t]}$ in $\mathbb{K}[X]$, with $P_{[t]}$ monic and squarefree of degree e , such that $V_t = V(P_{[t]}(X), Y - S_{[t]}(X))$. We use the $[t]$ in our subscripts, since the subscript t is reserved for polynomials obtained by applying a linear change of variable. The same will hold below for $A_{[t]}$.

Lemma 6. *Let t be such that H'_t and K'_t are in general position, and let $A_{[t]} \in \mathbb{K}[X]$ be their resultant with respect to Y . For $n \geq 1$, condition $M_2(n)$ holds if and only if we have both:*

- $P_{[t]}^n$ divides $A_{[t]}$ in $\mathbb{K}[X]$
- $P_{[t]}$ and $A_{[t]}/P_{[t]}^n$ are coprime in $\mathbb{K}[X]$.

PROOF. Because H'_t and K'_t are in general position, for any (x, y) in V_t , we know that $\mu((H'_t, K'_t), (x, y))$ is the valuation of the resultant $A_{[t]} = \text{res}(H'_t, K'_t, Y)$ at x , that is, the highest exponent n such that $(X - x)^n$ divides $A_{[t]}$. Equivalently, $\mu((H'_t, K'_t), (x, y))$ is characterized as being the unique integer n such that $(X - x)^n$ divides $A_{[t]}$ and $(X - x)$ and $A_{[t]}/(X - x)^n$ are coprime.

Taking all (x, y) in V into account, this leads to the condition given in the statement of the lemma.

We will now focus on the particular case where $\mathbb{K} = \mathbb{Q}$. We suppose that H and K are in $\mathbb{Z}[X, Y]$, that P and S are in $\mathbb{Q}[X]$, and that P, S, H, K satisfy M_0 , M_1 and $M_2(n)$, for some $n \geq 1$. Our goal is to give conditions on a prime p such that the same polynomials taken modulo p are well-defined and still satisfy M_0 , M_1 and $M_2(n)$.

Proposition 2. *There exists an explicitly computable function $\Delta_2(d, h, e, \ell) = (dhe\ell)^{O(1)}$ such that the following holds.*

Suppose that P and S have degree at most e and height at most ℓ , and that $H, K \in \mathbb{Z}[X, Y]$ have degree at most d and height at most h . If (P, S, H, K) satisfy M_0, M_1 and $M_2(n)$, for some $n \geq 1$, there exists a nonzero integer δ_2 such that:

- δ_2 has height at most $\Delta_2(d, h, e, \ell)$;
- for any prime p that does not divide δ_2 , $P \bmod p$ and $S \bmod p$ are well-defined, and $(P, S, H, K) \bmod p$ satisfy M_0, M_1 and $M_2(n)$ over \mathbb{F}_p .

The proof of this proposition will occupy the rest of this section. As a preliminary remark, recall that we let G be the gcd of H and K in $\mathbb{Z}[X, Y]$. Since V consists entirely of isolated points of $V(H, K)$, the polynomials (P, S, G) satisfy conditions C_0 and C_1 of the previous section. Our first constraint is that p does not divide the integer δ_1 defined in Proposition 1. For such a prime p , the polynomials $(P, S, G) \bmod p$ are well-defined, P remains squarefree modulo p , and $(P, S, G) \bmod p$ still satisfy conditions C_0 and C_1 . In particular, the polynomials $(P, S, H, K) \bmod p$ still satisfy M_0 , but we cannot conclude that they satisfy M_1 yet.

Let then Γ be the polynomial in $\mathbb{Z}[T]$ associated to H' and K' by the construction of Section 3.2.1. In all that follows, we take t in \mathbb{Z} such that $\Gamma(t)$ is nonzero; in particular, by Corollary 1, H'_t and K'_t are in general position. We let $A_{[t]} \in \mathbb{Z}[X]$ and $P_{[t]} \in \mathbb{Q}[X]$ be as defined above; then, by the previous lemma, $P_{[t]}^n$ divides $A_{[t]}$ in $\mathbb{Q}[X]$, and $P_{[t]}$ and $A_{[t]}/P_{[t]}^n$ are coprime in $\mathbb{Q}[X]$.

We will give conditions on p for which the same statement remains true modulo p ; then, using the converse direction in the previous lemma will allow us to conclude.

The resultant $A_{[t]}$ is in $\mathbb{Z}[X]$, not necessarily monic. The polynomial $P_{[t]}$ is monic, so we may write it as $P_{[t]} = P_{[t]}^*/c_{[t]}$, with $c_{[t]}$ in \mathbb{Z} and $P_{[t]}^*$ primitive in $\mathbb{Z}[X]$. Since $P_{[t]}^n$ divides $A_{[t]}$ in $\mathbb{Q}[X]$, we deduce that $P_{[t]}^{*n}$ divides $A_{[t]}$ in $\mathbb{Z}[X]$, so $N_{[t]} = A_{[t]}/P_{[t]}^{*n}$ is a polynomial with integer coefficients. By assumption, $P_{[t]}$ and $N_{[t]}$ are coprime, and thus so are $P_{[t]}^*$ and $N_{[t]}$. We deduce that their resultant is a nonzero integer.

Let us then add the following conditions on our prime p : $\Gamma(t) \bmod p$ is nonzero, and the resultant $\text{res}(P_{[t]}^*, N_{[t]}, X)$ does not vanish modulo p . We are going to prove that the construction of $P_{[t]}$ and $S_{[t]}$ specializes modulo p . Since $\Gamma(t) \bmod p$, we can apply Corollary 1, and we deduce the following facts:

- $H'_t \bmod p$ and $K'_t \bmod p$ are in general position; in particular, $H' \bmod p$ and $K' \bmod p$ have finitely many common solutions. Since $H = GH'$ and $K = GK'$, and since

by Proposition 1 the points defined by $(P(X) \bmod p, Y - S(X) \bmod p)$ do not cancel G , we deduce that these points are isolated points on $V(H \bmod p, K \bmod p)$, and that the multiplicities of $(H, K) \bmod p$ and $(H', K') \bmod p$ are the same at these points. In particular, we have proved that M_1 still holds.

- Let $\alpha_{[t]} \in \mathbb{F}_p[X]$ be the resultant of $H'_t \bmod p$ and $K'_t \bmod p$ with respect to Y . By Corollary 1, the leading terms of $H'_t \bmod p$ and $K'_t \bmod p$ are the reductions modulo p of those of H'_t and K'_t . As a consequence, $\alpha_{[t]} = A_{[t]} \bmod p$.
- Since $(P_{[t]}, S_{[t]})$ are the polynomials in the Shape Lemma representation of $V_t \subset V(H'_t, K'_t)$, none of the denominators of the coefficients of $P_{[t]}$ or $S_{[t]}$ vanishes modulo p . In particular, $c_{[t]}$ does not vanish modulo p .

Since P remains squarefree modulo p , the polynomials $P(X) \bmod p, Y - S(X) \bmod p$ define a subset V' of cardinality e of $V(H' \bmod p, K' \bmod p)$. Applying the change of coordinates ϕ_t , we obtain a subset V'_t of $V(H'_t \bmod p, K'_t \bmod p)$ of cardinality e . Since we saw that the latter equations are in general position, we can deduce from the discussion prior to Lemma 6 (that held over an arbitrary field, provided we are in general position) that there exist polynomials $\pi_{[t]}, \sigma_{[t]}$ in $\mathbb{F}_p[X]$ such that $V'_t = V(\pi_{[t]}(X), Y - \sigma_{[t]}(X))$.

Lemma 7. $\pi_t = P_{[t]} \bmod p$ and $\sigma_t = S_{[t]} \bmod p$.

PROOF. By uniqueness of the Shape Lemma representation, it is enough to prove that $V'_t = V(P_{[t]}(X) \bmod p, Y - S_{[t]}(X) \bmod p)$. Because both sets have cardinality e , it is even sufficient to prove only one inclusion.

Now, V'_t is simply the zero-set of $P(X + tY) \bmod p$ and $Y - S(X + tY) \bmod p$. By construction, both $P(X + tY)$ and $Y - S(X + tY)$ reduce to zero modulo $P_{[t]}(X), Y - S_{[t]}(X)$, and this relationship remains true modulo p , so we have indeed established that $V(P_{[t]}(X) \bmod p, Y - S_{[t]}(X) \bmod p) \subset V'_t$. As explained above, this implies that $\pi_t = P_{[t]} \bmod p$ and $\sigma_t = S_{[t]} \bmod p$.

We can now prove that $M_2(n)$ is satisfied for $(P, S, H, K) \bmod p$. By assumption on p , the resultant $\text{res}(P_{[t]}^*, N_{[t]}, X)$ does not vanish modulo p . Using the previous lemma, we deduce that $N_{[t]} \bmod p$ is equal, up to a nonzero constant, to $\alpha_{[t]}/\pi_{[t]}^n$. Since the degree of $P_{[t]}^* \bmod p$ remains equal to e , $\text{res}(P_{[t]}^*, N_{[t]}, X) \bmod p$ is thus equal (up to a power of $c_{[t]}$) to $\text{res}(\pi_{[t]}, \alpha_{[t]}/\pi_{[t]}^n, X)$. Since we say that this quantity is nonzero, Lemma 6 implies that $M_2(n)$ is satisfied for $(P, S, H, K) \bmod p$, and we are done.

It remains to roughly quantify the conditions on p . First of all, since $\deg(\Gamma) = O(d^4)$, there exists $t \in \mathbb{Z}$ that does not cancel Γ and such that $t = O(d^4)$; its height is $O(\log(d))$. By Lemma 4, the height of Γ is $O(hd^3 + d^4)$, so the height of $\Gamma(t)$ is

roughly of the same order, and an upper bound $(hd)^{O(1)}$ be calculated for it, see for instance [37, Lemma 1.2.1.c]. The polynomials $P_{[t]}^*$ and $N_{[t]}$ both divide the resultant $A_{[t]}$ in $\mathbb{Z}[X]$. Using bounds on the resultant to estimate $A_{[t]}$ [37, Lemma 1.2.1.(ab)], then factor bounds for $P_{[t]}^*$ and $N_{[t]}$ [37, Lemma 1.2.1.d], and once again resultant bounds for $\text{res}(P_{[t]}^*, N_{[t]}, X)$, we deduce the existence of bounds of the form $(hd)^{O(1)}$ for the height of the latter integer, that can be computed explicitly.

Putting these bounds together, and taking into account as well the fact that p does not divide the integer δ_1 defined in Proposition 1, we conclude the proof the proposition.

3.3 Finding zeros in a list

Consider the following question: take a field \mathbb{K} , an element x in \mathbb{K} (or, as below, in an algebraic closure of it called $\overline{\mathbb{K}}$) and polynomials $r = [r_1, \dots, r_N]$ in $\mathbb{K}[X]$. To x and R , we can associate the index $v(x, r)$, which is the smallest i such that $r_i(x)$ is nonzero; if no such i exists, take $v(x, r) = \infty$. Computing $v(x, r)$ is easy, by evaluating all r_i 's at x one after the other.

Let r be as before and let now P be non-constant and square-free in $\mathbb{K}[X]$; let also V be the set of roots of P in $\overline{\mathbb{K}}$. The finite set V can be partitioned into non-empty sets V_{v_1}, \dots, V_{v_s} , for some indices $v_i \in \mathbb{N} \cup \{\infty\}$, where V_{v_i} is the subset of all points x in V such that $v(x, r) = v_i$. Computing the partition V_{v_1}, \dots, V_{v_s} amounts to factoring P into (non-necessarily irreducible) factors P_1, \dots, P_s , and finding the indices v_1, \dots, v_s in $\mathbb{N} \cup \{\infty\}$, such that for all i in $\{1, \dots, s\}$, the set of roots of P_i in $\overline{\mathbb{K}}$ is precisely V_{v_i} (remark that the P_i 's and v_i 's are uniquely defined, up to order). This is the object of the following algorithm called `zero_index`.

Lemma 8. *Suppose that P has degree e , and that all r_i have degree less than e . Algorithm `zero_index` correctly returns $(P_1, v_1), \dots, (P_s, v_s)$ as specified above, using $O^\sim(eN)$ operations in \mathbb{K} .*

PROOF. Correctness is proved by seeing that at the beginning of each step i of the for loop, the roots of C are exactly the roots x of P for which $v(x, R) \geq i$, and that the roots of Z are then those roots x of P for which $v(x, R) = i$. Each pass through the loop takes $O^\sim(e)$ operations for GCD and exact division, so the cost estimate follows.

Slightly more generally, consider polynomials

$$R = [R_{1,1}, \dots, R_{1,N}], \dots, [R_{M,1}, \dots, R_{M,N}]$$

Algorithm 2: zero_index(P, r)

Input: P in $\mathbb{K}[X]$, $r = (r_1, \dots, r_N)$ in $\mathbb{K}[X]^N$ **Output:** $L = [(P_1, v_1), \dots, (P_s, v_s)]$, with $v_i \in \mathbb{N} \cup \{\infty\}$

```
1  $L = []$ 
2  $C = P$ 
3 for  $i = 1, \dots, N$  do
4    $Z = \gcd(C, r_i)$ 
5   if  $Z$  is not constant then
6      $\text{append}(C/Z, i)$  to  $L$ 
7    $C = Z$ 
8 end
9 if  $C$  is not constant then
10   $\text{append}(C, \infty)$  to  $L$ 
11 return  $L$ 
```

in $\mathbb{K}[X]$, and x as above. Then, to x and R , we want to associate the smallest index i such that one of $R_{1,i}(x), \dots, R_{M,i}(x)$ is nonzero (if it exists); we also want to compute the smallest index j such that $R_{j,i}(x)$ is nonzero, so that our output is $w(x, R) = (i, j)$. If no such i exists, instead of the pair (i, j) , we return $w(x, R) = (\infty, -1)$.

Given R and a polynomial P as before, we can then partition the zero-set $V \subset \overline{\mathbb{K}}$ of P into V_{w_1}, \dots, V_{w_t} , such that V_{w_i} is the set of all $x \in V$ such that $w(x, R) = w_i$. As output, we thus return a sequence of polynomials P_1, \dots, P_t , together with indices w_1, \dots, w_t in $(\{1, \dots, N\} \times \{1, \dots, M\}) \cup \{(\infty, -1)\}$, such that for all i in $\{1, \dots, t\}$, the set of roots of P_i in $\overline{\mathbb{K}}$ is precisely V_{w_i} .

This is done by the following algorithm, called `zero_index_vectorial`, which now takes as input P and the sequence of sequences of polynomials R . We use a subroutine called `infinity(L)` which takes as input a sequence $[(P_1, v_1), \dots, (P_s, v_s)]$ such as the one computed by `zero_index`, and returns the polynomial P_i in it corresponding to $v_i = \infty$, if one such polynomial exists; otherwise, this subroutine returns 1.

Lemma 9. *Suppose that P has degree e , and that all $R_{j,i}$ have degree less than e . Algorithm `zero_index_vectorial` correctly returns $(P_1, w_1), \dots, (P_t, w_t)$ as specified above, using $\tilde{O}(eMN)$ operations in \mathbb{K} .*

PROOF. Correctness is proved by seeing that at the beginning of each step i of the for loop, the roots of C are exactly the roots x of P for which we have not found a nonzero $R_{j,i'}$, for any $i' < i$. After the call `zero_index(C, r)`, L' contains the zero indices for $[R_{j,i} \bmod C \mid j \in [1, \dots, M]]$. We remove from it the factor $C = \text{infinity}(L')$ (if it exists), which corresponds to those roots for which we will continue the process. At the

Algorithm 3: zero_index_vectorial(P, R)

Input: P in $\mathbb{K}[X]$, $R = [R_{1,1}, \dots, R_{1,N}], \dots, [R_{M,1}, \dots, R_{M,N}]$ in $\mathbb{K}[X]^{M \times N}$

Output: $L = [(P_1, w_1), \dots, (P_t, w_t)]$, $w_i \in (\{1, \dots, N\} \times \{1, \dots, M\}) \cup \{(\infty, -1)\}$

```
1  $L = []$ 
2  $C = P$ 
3 for  $i = 1, \dots, N$  do
4    $r = [R_{j,i} \bmod C \mid j \in [1, \dots, M]]$ 
5    $L' = \text{zero\_index}(C, r)$ 
6    $C = \text{infinity}(L')$ 
7   if  $C$  is not constant then
8      $\mid$  remove  $(C, \infty)$  from  $L'$ 
9      $L = L \text{ cat } L'$ 
10 end
11 if  $C$  is not constant then
12    $\mid$  append  $(C, (\infty, -1))$  to  $L$ 
13 return  $L$ 
```

end of the loops, C defines those roots of P that cancel all $R_{j,i}$, so we associate it with $(\infty, -1)$.

For a given index i , the reductions at step 4 take $O(Me)$ operations in \mathbb{K} , using fast Euclidean division. Calling `zero_index` takes $O(Me)$ operations as well, in view of the previous lemma. Summing these costs, we conclude the proof.

3.4 Normal forms for derivatives

In this section, we discuss some normal form algorithms for derivatives, inspired by those in [39], together with ideas from [40]. In all this section, we work over the ring $\mathbb{A} = \mathbb{Z}/N\mathbb{Z}$, for some prime power $N = p^\ell$, using indeterminates X, ξ, ζ . Our input is as follows:

- $L = [(n_1, m_1), \dots, (n_t, m_t)]$ is a list of pairs of integers.
- $L' = [P_1, \dots, P_t]$ is a list of polynomials, with for all i , P_i monic of degree e_i in $\mathbb{A}[X]$. In addition, we suppose that for all i, j , with $i \neq j$, P_i and P_j generate the unit ideal in $\mathbb{A}[X]$. Equivalently, $P_i \bmod p$ and $P_j \bmod p$ are coprime in $\mathbb{F}_p[X]$.
- $L'' = [J_1, \dots, J_t]$ is a list of polynomials, with for all i , J_i in $\mathbb{A}[X, \xi]_{e_i, n_i+1}$.
- F is a polynomial in $\mathbb{A}[X, Y]$.

As output, we want to compute the normal forms

$$D_{i,\mu} = \frac{\partial^\mu F}{\partial Y^\mu}(X + \xi, J_i) \bmod \langle P_i(X), \xi^{n_i+1} \rangle,$$

for all $i = 1, \dots, t$ and $\mu = 0, \dots, m_i$. This will be done by computing

$$F_i = F(X + \xi, J_i + d_Y) \bmod \langle P_i(X), \xi^{n_i+1}, d_Y^{m_i+1} \rangle, \quad (3.3)$$

since Taylor expansion shows that

$$F_i = \sum_{\mu=0}^{m_i} \frac{1}{\mu!} D_{i,\mu} d_Y^\mu.$$

We will focus on the computation of the F_i 's, since the overhead to deduce all $D_{i,\mu}$'s by coefficient extraction and multiplication by $\mu!$'s will be negligible.

If, for instance, J_i does not depend on ξ , so it lies in $\mathbb{A}[X]_{e_i}$, $D_{i,\mu}$ can be written

$$D_{i,\mu} = \sum_{\nu=0}^{n_i} \frac{1}{\nu!} \frac{\partial^{\mu+\nu} F}{\partial X^\nu \partial Y^\mu}(X, J_i) \xi^\nu \bmod \langle P_i(X) \rangle;$$

knowing $D_{i,\mu}$ thus allows us to compute the normal forms of the derivatives $\frac{\partial^{\mu+\nu} F}{\partial X^\nu \partial Y^\mu}$ modulo $\langle P_i(X), Y - J_i(X) \rangle$, for all $i = 1, \dots, t$, $\nu = 0, \dots, n_i$ and $\mu = 0, \dots, m_i$.

Suppose that F has degree d . We make the following assumption regarding the quantities n_i, m_i, e_i :

\mathbf{H}_{NF} . The inequality $\sum_{1 \leq i \leq t} (n_i + 1)(m_i + 1)e_i = O(d^2)$ holds.

Representing F requires approximately d^2 coefficients in \mathbb{A} . On the other hand, for all i , F_i lies in $\mathbb{A}[\xi, \zeta, X]_{n_i+1, m_i+1, e_i}$, representing all of them uses $\sum_{1 \leq i \leq t} (n_i + 1)(m_i + 1)e_i$ coefficients in \mathbb{A} . Thus, assumption \mathbf{H}_{NF} means that input and output size are not too far off.

The main result in this section is the following proposition, which shows that all F_i can be computed in essentially linear-time.

Proposition 3. *Under assumption \mathbf{H}_{NF} , for any $\varepsilon > 0$, there exists an algorithm normal forms that takes as input a prime power $N = p^\ell$, sequences L, L', L'' and polynomial F as above, and returns all F_i , for i in $\{1, \dots, t\}$, using $d^{2+\varepsilon} O^\sim(\log(N))$ bit operations.*

3.4.1 Auxiliary results

A first normal form algorithm. The central problem for these normal form questions is normal form computation modulo a single *triangular set* $\mathbf{T} = (P(X), Q(X, Y))$, with P monic in X and Q monic in Y , reduced with respect to P . Given F in $\mathbb{A}[X, Y]$, the question is to compute $F \bmod \langle P, Y - Q \rangle$. This apparently simple question is actually quite challenging; so far, no algorithm is known to solve it in optimal time in an *algebraic* complexity model.

In our particular context of computations modulo N , however, better results are available. Building on seminal results by Kedlaya and Umans [36], Theorem 6 in [53] gives a quasi-linear *bit complexity* result for such a task (as pointed out in [39], this result was originally proved for N a prime, but carries over to the case of a prime power).

Lemma 10. *For any $\varepsilon > 0$, there exists an algorithm `normal_form` with the following input:*

- a prime power N ;
- F in $\mathbb{A}[X, Y]_{m,n}$, with $\mathbb{A} = \mathbb{Z}/N\mathbb{Z}$,
- a triangular set $\mathbf{T} = (P(X), Q(X, Y))$, with P in $\mathbb{A}[X]$, monic of degree f , and Q in $\mathbb{A}[X, Y]$, monic in Y of degree g and of degree in X less than e .

This algorithm returns $F \bmod \langle P, Y - Q \rangle$ using $(mn + fg)^{1+\varepsilon} O^\sim(\log(N))$ bit operations.

Remark that up to the exponent ε , this algorithm is optimal, since both input and output involve $O(mn + ef)$ coefficients in $\mathbb{A} = \mathbb{Z}/N\mathbb{Z}$, which require a total of $O((mn + ef) \log(N))$ bits of storage.

Using this, Proposition 3 in [39] states the following result regarding the reduction of one polynomial F modulo several bivariate triangular sets.

Lemma 11. *Let $\mathbf{T}_1, \dots, \mathbf{T}_s$ be triangular sets in $\mathbb{A}[X, Y]$, where for all i , $\mathbf{T}_i = (P_i(X), Q_i(X, Y))$, with P_i monic in X of degree f_i and $Q_i(X, Y)$ monic in Y of degree g_i , and reduced with respect to X . Suppose that for all i, j in $\{1, \dots, s\}$, with $i \neq j$, P_i and P_j generate the unit ideal in $\mathbb{A}[X]$.*

Let F be in $\mathbb{A}[X, Y]$ with degree d , and suppose that $\sum_{i \leq s} f_i g_i = O(d^2)$. Then, for any $\varepsilon > 0$, there exists an algorithm `normal_forms_bivariate` that takes as input the prime power N , $\mathbf{T}_1, \dots, \mathbf{T}_s$ and F as above, and returns all $F \bmod \langle \mathbf{T}_i \rangle$, for i in $\{1, \dots, s\}$, using $d^{2+\varepsilon} O^\sim(\log(N))$ bit operations.

As in Proposition 3, the input and output sizes are $O(d^2)$ elements of \mathbb{A} , so the running time is close to optimal. This lemma will be our main tool to prove our proposition; most of the work in this section will consist in turning our original problem into an instance of the bivariate problem above.

There are two slight differences between the statement above and the one given in reference [39]. First, that result seemingly required another assumption, namely that all g_i should satisfy $g_i \leq d$. This is actually not needed: paper [39] gave an alternative solution to this problem, valid in an algebraic complexity model (over an arbitrary ring), that did require such an assumption. In our context, we can safely omit it.

Another slight difference is that the result in [39] required as an extra input the inverses of $(P_1 \cdots P_{i-1} P_{i+1} \cdots P_s)$ modulo P_i , for all $i = 1, \dots, s$. It was then pointed out that in the case $\mathbb{A} = \mathbb{Z}/N\mathbb{Z}$, for N a prime power, they can be computed in $O(d^2 \log(N))$ operations, which will be negligible. Thus, our assumptions are not restrictive.

An easy change of order. Our next auxiliary result is an explicit change-of-order for a particular bivariate ideal in $\mathbb{A}[X, Z]$. Several references give algorithms to perform this kind of operations [14, 50], but we are not aware of a complexity result that would apply in this particular case (for instance, the algorithm of [50] requires a radical ideal over a field, none of which conditions applying here). Nevertheless, the situation is simple enough that we can give an explicit solution.

Lemma 12. *Let P be in $\mathbb{A}[X]$ of degree e , such that $P \bmod p$ is squarefree, and let n be a positive integer. One can compute using $O(en \log(N))$ bit operations a polynomial V in $\mathbb{A}[Z]$ of degree less than en , such that in $\mathbb{A}[X, Z]$, we have the following equality between ideals:*

$$\langle P(X), (Z - X)^n \rangle = \langle P(Z)^n, X - V(Z) \rangle.$$

PROOF. Let P^* be an arbitrary monic lift of P to $\mathbb{Z}_p[X]$, where \mathbb{Z}_p is the ring of p -adic integers. Because $P \bmod p$ is squarefree, P^* is squarefree as well. In the first part of the proof, we work over \mathbb{Z}_p , its field of fractions \mathbb{Q}_p , and an algebraic closure of it, $\overline{\mathbb{Q}_p}$.

Let a_1, \dots, a_e be the (unknown) pairwise distinct roots of P^* in $\overline{\mathbb{Q}_p}$. Then, the ideal $\langle P^*(X), (Z - X)^n \rangle$ is the product of the pairwise coprime ideals

$$\left| \begin{array}{l} (Z - a_i)^n \\ X - a_i, \end{array} \right. \quad i = 1, \dots, e.$$

For such ideals, changing the order of X and Z is straightforward. We deduce that the

polynomial $V^*(Z)$ of degree less than en defined by the Chinese Remainder conditions

$$V^* \bmod (Z - a_i)^n = a_i, \quad i = 1, \dots, e$$

satisfies the equality $\langle P^*(X), (Z - X)^n \rangle = \langle Q(Z), X - V^*(Z) \rangle$, except that V^* is *a priori* in $\overline{\mathbb{Q}_p}[Z]$, and the equality holds in $\overline{\mathbb{Q}_p}[X, Z]$.

Let us write $Q = P^*(Z)^n$. To compute V^* , we define the polynomials

$$A = \sum_{i=1}^e a_i \prod_{i' \neq i} (Z - a_{i'})^n \quad \text{and} \quad B = \sum_{i=1}^e \prod_{i' \neq i} (Z - a_{i'})^n.$$

First, let us show how to compute A and B ; we will show as we go that both A and B are in $\mathbb{Z}_p[Z]$.

Let \tilde{A} and \tilde{B} be the polynomials $Z^{(e-1)n}A(1/Z)$ and $Z^{(e-1)n}B(1/Z)$; define similarly $\tilde{Q} = Z^{en}Q(1/Z)$, so that we have

$$\tilde{A} = \sum_{i=1}^e a_i \prod_{i' \neq i} (1 - a_{i'}Z)^n, \quad \tilde{B} = \sum_{i=1}^e \prod_{i' \neq i} (1 - a_{i'}Z)^n$$

and

$$\tilde{Q} = \prod_{i=1}^e (1 - a_iZ)^n.$$

Let us first show how to compute the power series expansions of the rational functions \tilde{A}/\tilde{Q} and \tilde{B}/\tilde{Q} . Consider the power series

$$\frac{1}{(1 - Z)^n} = \sum_{j \geq 0} c_j Z^j \quad \text{and} \quad S = \sum_{j \geq 0} s_j Z^j,$$

where $s_j = a_1^j + \dots + a_n^j$ is the j th power sum of P^* , so that all c_j 's and s_j 's are in \mathbb{Z}_p . The rational functions \tilde{A}/\tilde{Q} and \tilde{B}/\tilde{Q} can then be written as

$$\begin{aligned} \frac{\tilde{A}}{\tilde{Q}} &= \sum_{i=1}^e \frac{a_i}{(1 - a_iZ)^n} = \sum_{i=1}^e \sum_{j \geq 0} a_i^{j+1} c_j Z^j = \sum_{j \geq 0} c_j s_{j+1} Z^j, \\ \frac{\tilde{B}}{\tilde{Q}} &= \sum_{i=1}^e \frac{1}{(1 - a_iZ)^n} = \sum_{i=1}^e \sum_{j \geq 0} a_i^j c_j Z^j = \sum_{j \geq 0} c_j s_j Z^j. \end{aligned}$$

It is enough to compute both series expansions at precision en . Upon multiplication by \tilde{Q} , we deduce that \tilde{A} and \tilde{B} are both in $\mathbb{Z}_p[Z]$, as claimed.

In addition, we claim that B is invertible modulo Q , not only in $\mathbb{Q}_p[Z]$, but actually in $\mathbb{Z}_p[Z]$. Indeed, the resultant of Q and B is (up to sign) the n^2 -th power of the discriminant of P^* , which is by assumption a unit in \mathbb{Z}_p . Finally, one verifies that $V^* = A/B \bmod Q$, so that V^* is in $\mathbb{Z}_p[Z]$, as announced before.

So far, we established the equality $\langle P^*(X), (Z - X)^n \rangle = \langle Q(Z), X - V^*(Z) \rangle$ in $\overline{\mathbb{Q}_p}[X, Z]$. However, since all polynomials are in $\mathbb{Z}_p[X, Z]$, and monic in their leading variables, we deduce that the underlying membership identities hold in $\mathbb{Z}_p[X, Z]$ as well. Truncating modulo N , and defining $V = V^* \bmod N \in \mathbb{A}[Z]$, we conclude that the equality $\langle P(X), (Z - X)^n \rangle = \langle P(Z)^n, X - V(Z) \rangle$ holds in $\mathbb{A}[X, Z]$.

Finally, we turn to the cost analysis. We can compute all coefficients c_j and s_j at precision en using $\tilde{O}(en)$ operations in \mathbb{A} , and thus $\tilde{O}(en \log(N))$ bit operations: for the former, this is for instance done by computing $(1 - X)^n$ by binary powering and inverting it; for the latter, this is in [58].

Once we know the coefficients c_j and s_j , we recover \tilde{A} and \tilde{B} through multiplication by Q and reversal, for another $\tilde{O}(en)$ operations in \mathbb{A} , and A and B are deduced for free. The last non-obvious step is the computation of $1/B \bmod Q$ (since the rest is just another multiplication modulo Q). This is done using Newton iteration: the inverse of B modulo $\langle p, Q \rangle$ can be computed using the fast extended GCD algorithm in $\mathbb{F}_p[Z]$ in $\tilde{O}(en)$ operations modulo p ; then, Newton iteration for inverse gives us $1/B \bmod Q$ in $\mathbb{A}[Z]$ in quasi-linear time $\tilde{O}(en \log(N))$. Summing all costs above gives the claimed overall running time.

All notation being as in the lemma, we deduce that we have an isomorphism

$$\psi : \mathbb{A}[X, Z] / \langle P(X), (Z - X)^n \rangle \rightarrow \mathbb{A}[Z] / \langle P(Z)^n \rangle.$$

Taking $\mathbb{A}[X, Z]_{e,n}$ and $\mathbb{A}[Z]_{en}$ for representatives of respectively the left and right-hand sides, ψ is given by

$$\psi(R) = R \bmod \langle P(Z)^n, X - V(Z) \rangle$$

for R in $\mathbb{A}[X, Z]_{n,e}$, and

$$\psi^{-1}(S) = S \bmod \langle P(X), (Z - X)^n \rangle$$

for S in $\mathbb{A}[Z]_{en}$. Once V is known, applying Lemma 10, we deduce in particular that for any $\varepsilon > 0$, both change-of-bases ψ and ψ^{-1} can be performed in $(en)^{1+\varepsilon} \tilde{O}(\log(N))$ bit operations.

3.4.2 Proof of Proposition 3

Recall that on input sequences L, L', L'' our goal is to compute normal forms

$$F_i = F(X + \xi, J_i + \zeta) \bmod \langle P_i(X), \xi^{n_i+1}, \zeta^{m_i} \rangle,$$

for $i = 1, \dots, t$. Let us fix i in $\{1, \dots, t\}$, and let Z and T be two new variables. We will use them through the change of variables $Z = X + \xi$, $T = J_i + \zeta$.

First change of variables. First, we consider the introduction of the variable Z , that stands for $X + \xi$. In most of this paragraph, the index $i \in \{1, \dots, t\}$ is fixed. Then, there is an \mathbb{A} -algebra isomorphism

$$\phi_i : \mathbb{A}[X, \xi] / \langle P_i(X), \xi^{n_i+1} \rangle \rightarrow \mathbb{A}[X, Z] / \langle P_i(X), (Z - X)^{n_i+1} \rangle.$$

The left-hand side and right-hand side admit respectively the polynomials in $\mathbb{A}[X, \xi]_{e_i, n_i+1}$ and $\mathbb{A}[X, Z]_{e_i, n_i+1}$ as canonical representatives. With these representatives, we have, for R in $\mathbb{A}[X, \xi]_{e_i, n_i+1}$, $\phi_i(R) = R(X, Z - X) \bmod P_i$. The inverse mapping is given by $\phi_i^{-1}(S) = S(X, \xi + X) \bmod P_i$, for S in $\mathbb{A}[X, Z]_{e_i, n_i+1}$.

Lemma 13. *The following holds:*

- For R in $\mathbb{A}[X, \xi]_{e_i, n_i+1}$, one can compute $\phi_i(R)$ using $O^\sim(e_i n_i + 1 \log(N))$ bit operations.
- For S in $\mathbb{A}[X, Z]_{e_i, n_i+1}$, one can compute $\phi_i^{-1}(S)$ using $O^\sim(e_i n_i + 1 \log(N))$ bit operations.

PROOF. We give the proof for ϕ_i ; that for ϕ_i^{-1} is entirely similar. Define $\mathbb{B} = \mathbb{A}[X] / \langle P_i \rangle$. Computing $\phi_i(R)$ amounts to seeing R in $\mathbb{B}[\xi]$, and computing $R(\xi - X)$ in that ring (and finally, formally replacing ξ by Z). This is thus an instance of *shifting* a polynomial, in this case by $-X$. Since R has degree less than $n_i + 1$ in ξ , the divide-and-conquer algorithm of [29] solves this problem in $O^\sim(n_i + 1)$ operations $(+, \times)$ in \mathbb{B} , which is $O^\sim(e_i n_i + 1)$ operations $(+, \times)$ in \mathbb{A} , and thus $O^\sim(e_i n_i + 1 \log(N))$ bit operations.

The mapping ϕ_i can then be extended to a change of a variables

$$\Phi_i : \mathbb{A}[X, \xi, \zeta] / \langle P_i(X), \xi^{n_i+1}, \zeta^{m_i} \rangle \rightarrow \mathbb{A}[X, Z, \zeta] / \langle P_i(X), (Z - X)^{n_i+1}, \zeta^{m_i} \rangle,$$

which acts coefficient-wise in ζ . Both Φ_i and its inverse Φ_i^{-1} can thus be computed in $O^\sim(e_i(n_i + 1)\mu_i \log(N))$ bit operations. Let us finally write $J_i^{(1)} = \phi_i(J_i)$, so that $J_i^{(1)}$ lies

in $\mathbb{A}[X, Z]_{e_i, n_i+1}$. Defining

$$F_i^{(1)} = F(Z, J_i^{(1)} + \zeta) \bmod \langle P_i(X), (Z - X)^{n_i+1}, \zeta^{m_i} \rangle,$$

we see that F_i can be recovered as $\Phi_i^{-1}(F_i^{(1)})$.

Since we saw that applying the changes of variables takes quasi-linear time, we can now focus on computing the polynomials $F_i^{(1)}$, for $i = 1, \dots, t$.

Second change of variables. Our second change of variables is actually a change of order. As before, for the following discussion, we fix an index i in $\{1, \dots, t\}$.

Applying Lemma 12, we deduce that we can compute in $O^\sim(e_i(n_i + 1))$ a polynomial V_i in $\mathbb{A}[Z]$ such that we have the equality between ideals

$$\langle P_i(X), (Z - X)^{n_i+1} \rangle = \langle P_i(Z)^{n_i+1}, X - V_i(X) \rangle$$

in $\mathbb{A}[X, Z]$. In addition, we saw that the change of basis

$$\psi_i : \mathbb{A}[X, Z] / \langle P_i(X), (Z - X)^{n_i+1} \rangle \rightarrow \mathbb{A}[Z] / \langle P_i(Z)^{n_i+1} \rangle$$

and its inverse can be performed in $(e_i(n_i + 1))^{1+\varepsilon} O^\sim(\log(N))$ bit operations. As above, the mapping ψ_i can be extended to a change-of-basis

$$\Psi_i : \mathbb{A}[X, Z, \zeta] / \langle P_i(X), (Z - X)^{n_i+1}, \zeta^{m_i} \rangle \rightarrow \mathbb{A}[Z, \zeta] / \langle P_i(Z)^{n_i+1}, \zeta^{m_i} \rangle$$

which acts coefficient-wise in ζ . Both Ψ_i and its inverse Ψ_i^{-1} can thus be computed in $(e_i(n_i + 1))^{1+\varepsilon} O^\sim(m_i \log(N))$ bit operations.

Let us finally write $J_i^{(2)} = \psi_i(J_i^{(1)})$, so that $J_i^{(2)}$ lies in $\mathbb{A}[Z]_{e_i(n_i+1)} \simeq \mathbb{A}[Z] / \langle P_i(Z)^{n_i+1} \rangle$. Defining

$$F_i^{(2)} = F(Z, J_i^{(2)} + \zeta) \bmod \langle P_i(Z)^{n_i+1}, \zeta^{m_i} \rangle,$$

we see that $F_i^{(1)}$ can be recovered as $\Psi_i^{-1}(F_i^{(2)})$. Thus, since the change of bases take quasi-linear time, we can now focus on computing the normal forms $F_i^{(2)}$, for $i = 1, \dots, t$.

Third change of variables. Our last change of variables introduces a new variable T which will stand for $J_i^{(2)} + \zeta$. In the same vein as what we said for the introduction of variable Z , we can now notice that there is an \mathbb{A} -algebra isomorphism

$$\gamma_i : \mathbb{A}[Z, \zeta] / \langle P_i(Z)^{n_i+1}, \zeta^{m_i} \rangle \rightarrow \mathbb{A}[Z, T] / \langle P_i(Z)^{n_i+1}, (T - J_i^{(2)})^{m_i} \rangle.$$

The left-hand side and right-hand side admit respectively the elements of $\mathbb{A}[Z, \zeta]_{e_i(n_i+1), m_i}$ and $\mathbb{A}[Z, T]_{e_i(n_i+1), m_i}$ as canonical representatives. With these representatives, we have, for R in $\mathbb{A}[Z, \zeta]_{e_i(n_i+1), m_i}$, $\gamma_i(R) = R(Z, T - J_i^{(2)}) \bmod P_i^{n_i+1}$. The inverse mapping is given by $\gamma_i^{-1}(S) = S(Z, \zeta + J_i^{(2)}) \bmod P_i^{n_i+1}$, for S in $\mathbb{A}[Z, T]_{e_i(n_i+1), m_i}$.

Proceeding exactly as in Lemma 13, we deduce that we can compute γ_i or its inverse in $O^\sim(e_i n_i + 1 m_i \log(N))$ bit operations. Defining finally

$$F_i^{(3)} = F(Z, T) \bmod \langle P_i(Z)^{n_i+1}, (T - J_i^{(2)})^{m_i} \rangle,$$

we deduce that $F_i^{(2)} = \gamma_i^{-1}(F_i^{(3)})$. Once more, the change of variables takes quasi-linear time, so we are left with the problem of computing the polynomials

$$F(Z, T) \bmod \langle P_i(Z)^{n_i+1}, (T - J_i^{(2)})^{m_i} \rangle,$$

for $i = 1, \dots, t$. Since the polynomials $(P_i(Z)^{n_i+1}, P_j(Z)^{n_j+1})$ generate the unit ideal in $\mathbb{A}[Z]$ (for $i \neq j$), this can be done as a direct application of Lemma 11, with the announced cost of $d^{2+\varepsilon} O^\sim(\log(N))$ bit operations, for any $\varepsilon > 0$.

Adding up all of the costs seen so far, we conclude the proof of Proposition 3.

3.5 The deflation lemma

Consider a polynomial system $F = G = 0$ in $\mathbb{K}[X, Y]$ and an isolated solution (x^*, y^*) of it. Most extensions of Newton iteration to the case where (x, y) has multiplicity $M > 1$ seek to replace the given system with a new one, say ψ , such that the multiplicity of ψ at the root (x, y) is less than M – eventually, we reach $M = 1$, where we can apply Newton iteration without difficulty. Such a process is called *deflation*.

Following Lecerf's approach, the deflated systems are constructed by considering suitable derivatives of the given system $\langle F, G \rangle$. For the complexity analysis, we will need to set a bound on the order of partial derivatives. The following construction assigns to an isolated solution (x, y) of $F = G = 0$ a *signature* $\sigma(x^*, y^*)$, of the form $\sigma(x^*, y^*) = (m, \mathbb{H}, n, a, \mathbb{K})$. In essence, this signature predicts which derivatives of F, G should be taken to reach a deflated ideal ψ satisfying the multiplicity reduction requirement.

The following *deflation* lemma is the key to this construction. It follows very closely Lemma 4 from [40]. That reference deals with systems in an arbitrary number of variables, but relies on a generic change of variables, which we avoid here (by slightly changing the definition of integer m below).

Lemma 14 (Deflation lemma). *Let $\langle F, G \rangle$ be an ideal in $\mathbb{K}[X, Y]$, with F and G of degree at most d , and let $(x^*, y^*) \in \mathbb{K}^2$ be an isolated root of $\langle F, G \rangle$ with multiplicity M . Define*

$$m := \min \left\{ \mu : \frac{\partial^\mu F}{\partial Y^\mu}(x^*, y^*) \neq 0 \quad \text{or} \quad \frac{\partial^\mu G}{\partial Y^\mu}(x^*, y^*) \neq 0 \right\}.$$

If in addition $\overline{\mathbb{K}}$ has characteristic at least d , then:

(a) $m > 0$;

(b) $m \leq d$;

(c) (x^*, y^*) is a root of ψ with multiplicity n , for some integer n satisfying $1 \leq n \leq M/m$, where

$$\psi := \left\langle F, G, \frac{\partial F}{\partial Y}, \frac{\partial G}{\partial Y}, \dots, \frac{\partial^{m-1} F}{\partial Y^{m-1}}, \frac{\partial^{m-1} G}{\partial Y^{m-1}} \right\rangle.$$

PROOF. Upon translating the origin to (x^*, y^*) , we can assume without loss of generality that $x^* = y^* = 0$. To prove the first item, note that $F(0, 0) = G(0, 0) = 0$, which implies $m > 0$. Let us further denote by I the ideal $\langle F, G \rangle$.

Let us next prove that m is finite. If all partial derivatives of F with respect to Y, Y^2, \dots, Y^d vanish at $(0, 0)$, $F(0, Y)$ must be the zero polynomial (recall that F has degree at most d), so that X divides F . If this is the case for G as well, X divides both F and G , so $(0, 0)$ is not an isolated solution of $F = G = 0$, a contradiction.

Using the following facts,

$$\frac{\partial^\mu F}{\partial Y^\mu}(0, 0) = 0 \quad \text{and} \quad \frac{\partial^\mu G}{\partial Y^\mu}(0, 0) = 0, \quad 0 \leq \mu \leq m - 1$$

and

$$\psi = \left\langle F, G, \frac{\partial F}{\partial Y}, \frac{\partial G}{\partial Y}, \dots, \frac{\partial^{m-1} F}{\partial Y^{m-1}}, \frac{\partial^{m-1} G}{\partial Y^{m-1}} \right\rangle,$$

it is clear that $(0, 0)$ is a root of ψ , so $n \geq 1$. It remains to give an upper bound on it.

We are going to work locally, by looking at F, G and their derivatives in $\mathbb{K}[[X, Y]]$. So, by the definition of multiplicity, we have

$$M = \dim_{\mathbb{K}} \mathbb{K}[[X, Y]]/I \quad \text{and} \quad n = \dim_{\mathbb{K}} \mathbb{K}[[X, Y]]/\psi.$$

We are going to describe more precisely these residue class rings. Let us endow $\mathbb{K}[[X, Y]]$ with the order defined by

$$X^{a_1} Y^{b_1} > X^{a_2} Y^{b_2} \iff a_1 < a_2 \quad \text{or} \quad a_1 = a_2 \quad \text{and} \quad b_1 < b_2.$$

One verifies that this order is compatible with multiplication, and that $1 > X$ and $1 > Y$ both hold. This is thus a *local monomial order*, as in [21, Chapter 4]. To any power series S in $\mathbb{K}[[X, Y]]$, we can associate its *leading monomial* $\text{lm}(S)$ with respect to this order; this notation carries over to ideals in $\mathbb{K}[[X, Y]]$.

From [21, Theorem 4.3], we infer that the monomials in $\text{lm}(I)^c$ and $\text{lm}(\psi)^c$ – where the exponent c denotes complement – form bases of respectively $\mathbb{K}[[X, Y]/I$ and $\mathbb{K}[[X, Y]/\psi$. In particular, the numbers of these monomials are respectively M and n . Define

$$T := \{X^a Y^{m-1} \in \text{lm}(I)^c \mid a \geq 0\}.$$

Because $\text{lm}(I)$ is stable by multiplication, for each element $X^a Y^{m-1}$ of T , all monomials $X^a Y^b$, for $0 \leq b \leq m-1$, are in $\text{lm}(I)^c$, whence $M = |\text{lm}(I)^c| \geq m|T|$. We now prove that n is at most $|T|$.

- The definition of m implies that at least one of $\frac{\partial^m F}{\partial Y^m}$ or $\frac{\partial^m G}{\partial Y^m}$ does not vanish at $(0, 0)$; let us assume without loss of generality that this is the case for $\frac{\partial^m F}{\partial Y^m}$. This implies that for $b = 0, \dots, m-1$, the coefficient of the monomial Y^b in F is zero, while that of Y^m is nonzero. The definition of our local order then implies that Y^m is the leading term of F . Thus, Y^m is in $\text{lm}(I)$, so that $X^a Y^m$ is in $\text{lm}(I)$ for any $a \geq 0$.

Consider an element $P \in \mathbb{K}[[X, Y]]$ having leading monomial $X^a Y^m$. Because $m \leq d$, and due to our assumption on the characteristic of \mathbb{K} , we deduce that the leading monomial of $\frac{\partial^{m-1} P}{\partial Y^{m-1}}$ is $m! X^a Y$. This shows that for $a \geq 0$, $X^a Y$ is in $\text{lm}(\psi)$.

- Similarly, let a_0 be the smallest integer such that $X^{a_0} Y^{m-1}$ is in $\text{lm}(I)$; thus, $a_0 = |T|$. Differentiating $m-1$ times as above, we deduce that X^{a_0} is in $\text{lm}(\psi)$.

The two items above prove that $\text{lm}(\psi)^c$ is contained in $\{X^a \mid 0 \leq a < a_0\}$, so it has cardinality at most $a_0 = |T|$. This proves the lemma.

This lemma allows us to define the first components m, \mathbb{H} of $\sigma(x^*, y^*)$:

- m is defined as in the lemma;
- the string $\mathbb{H} \in \{\text{"F"}, \text{"G"}\}$ indicates which of $\frac{\partial^m F}{\partial Y^m}$ and $\frac{\partial^m G}{\partial Y^m}$ is nonzero at (x^*, y^*) ; in case of a tie, for definiteness, we choose F .

Using the same notation as above, let us define $H = F$ (if $\mathbb{H} = \text{"F"}$) or $H = G$ (if $\mathbb{H} = \text{"G"}$), so that

$$m = \min \left\{ \mu : \frac{\partial^\mu H}{\partial Y^\mu}(x^*, y^*) \neq 0 \right\}$$

and (x^*, y^*) is a root of $\frac{\partial^{m-1}H}{\partial Y^{m-1}}$.

Define H^c (the ‘‘complement’’ of H) as either $H^c = G$ if $H = F$ and $H^c = F$ if $H = G$. We could replace the system (F, G) by $(\frac{\partial^{m-1}H}{\partial Y^{m-1}}, H^c)$ to find the root (x^*, y^*) , but the multiplicity of (x^*, y^*) for that new system is not necessarily less than M/m . To fix the problem, based on the deflation lemma, we consider the following system instead:

$$\psi := \left\langle F, G, \frac{\partial F}{\partial Y}, \frac{\partial G}{\partial Y}, \dots, \frac{\partial^{m-1}F}{\partial Y^{m-1}}, \frac{\partial^{m-1}G}{\partial Y^{m-1}} \right\rangle.$$

The lemma then implies that (x^*, y^*) is a root of ψ of multiplicity n , with $n \leq M/m$.

The invertibility assumption of $\frac{\partial^m H}{\partial Y^m}(x^*, y^*)$ allows us to apply the implicit function theorem to $\frac{\partial^{m-1}H}{\partial Y^{m-1}}$ at (x^*, y^*) . Replacing X by $x^* + \xi$, where ξ is a new variable, we can find a power series J in $\mathbb{K}[[\xi]]$ and A in $\mathbb{K}[[\xi]][Y]$ such that

$$\frac{\partial^{m-1}H}{\partial Y^{m-1}}(x^* + \xi, Y) = (Y - J)A, \quad J(0) = y^* \quad \text{and} \quad A(0, y^*) \neq 0.$$

Let us further replace Y by $y^* + \zeta$, where ζ is a new variable, and let us work in the power series ring $\mathbb{K}[[\xi, \zeta]]$. Since $A(0, y^*)$ is nonzero, $A(\xi, y^* + \zeta)$ is a unit in $\mathbb{K}[[\xi, \zeta]]$. We deduce that in $\mathbb{K}[[\xi, \zeta]]$, we have the equality

$$\frac{\partial^{m-1}H}{\partial Y^{m-1}}(x^* + \xi, y^* + \zeta) = \zeta - (J - y^*).$$

This implies that in $\mathbb{K}[[\xi, \zeta]]$, $(0, 0)$ is a root of multiplicity n of the ideal generated by

$$\left(\frac{\partial^\alpha H}{\partial Y^\alpha}(x^* + \xi, J) \right)_{0 \leq \alpha < m-1}, \quad \zeta - (J - y^*), \quad \left(\frac{\partial^\alpha H^c}{\partial Y^\alpha}(x^* + \xi, J) \right)_{0 \leq \alpha \leq m-1}.$$

For $\alpha \geq 0$, define

$$H_\alpha := \frac{\partial^\alpha H}{\partial Y^\alpha}(x^* + \xi, J) \quad \text{and} \quad H_\alpha^c := \frac{\partial^\alpha H^c}{\partial Y^\alpha}(x^* + \xi, J),$$

so that the above ideal is generated by

$$\langle H_0, H_0^c, H_1, H_1^c, \dots, H_{m-2}, H_{m-2}^c, \zeta - (J - y^*), H_{m-1}^c \rangle.$$

Remark that, with the exception of $\zeta - (J - y^*)$, all the above generators are in $\mathbb{K}[[\xi]]$. Since $\zeta - (J - y^*)$ has degree one in ζ , we deduce that 0 is a root of multiplicity n of the ideal

$$\langle H_0, H_0^c, H_1, H_1^c, \dots, H_{m-2}, H_{m-2}^c, H_{m-1}^c \rangle \subset \mathbb{K}[[\xi]].$$

This proves in particular the following lemma.

Lemma 15. *The integer n satisfies*

$$n = \min (\{ \text{val}(H_\alpha) \}_{0 \leq \alpha < m-1} \cup \{ \text{val}(H_\alpha^c) \}_{0 \leq \alpha < m}),$$

where val denotes the ξ -adic valuation.

This allows us to complete the definition of the signature $\sigma(x^*, y^*)$: the last three components are n , the integer a that realizes the minimum above (in case of a tie, choose the minimum), and a string $K \in \{ "H", "H^c" \}$ that indicates whether the minimum occurs for H or H^c (in case of a tie, choose H). Associated to string K , we have the corresponding polynomial $K \in \{F, G\}$, obviously defined so $K = H$ if $K = "H"$ and $K = H^c$ otherwise.

The following lemma will help us give conditions on the preservation of the signature through specialization at primes, when for instance $\mathbb{K} = \mathbb{Q}$.

Lemma 16. *Suppose that (x^*, y^*) has signature (m, H, n, a, K) . Then (x^*, y^*) is a root of multiplicity n of $\langle \frac{\partial^{m-1}H}{\partial Y^{m-1}}, \frac{\partial^a K}{\partial Y^a} \rangle$.*

PROOF. The proof amounts to going backward the previous derivation, but taking fewer polynomials into account. By definition of a and K , and using Lemma 15, we see that n is the ξ -adic valuation of

$$\frac{\partial^a K}{\partial Y^a}(x^* + \xi, J)$$

in $\mathbb{K}[[\xi]]$, that is, the multiplicity of 0 as a root of the equation $\frac{\partial^a K}{\partial Y^a}(x^* + \xi, J)$. Equivalently, it is the multiplicity of $(0, 0)$ as a root of the system $\langle \frac{\partial^a K}{\partial Y^a}(x^* + \xi, J), \zeta - (J - y^*) \rangle$ in $\mathbb{K}[[\xi, \zeta]]$; as we saw previously, this ideal coincides with $\langle \frac{\partial^a K}{\partial Y^a}(x^* + \xi, y^* + \zeta), \frac{\partial^{m-1}H}{\partial Y^{m-1}}(x^* + \xi, y^* + \zeta) \rangle$. Translating back the origin, this proves our claim.

3.6 The σ -decomposition

In this section, we consider two polynomials F and G in $\mathbb{K}[X, Y]$, over some field \mathbb{K} , with degree at most d . We assume that \mathbb{K} has characteristic greater than or equal to $\deg(F)$ and $\deg(G)$, and that F and G have no nontrivial common factor in $\mathbb{K}[X, Y]$.

In this case, the signature $\sigma(x, y)$ of any element (x, y) of $V = V(F, G)$ is well-defined. Since V is finite, we can partition it into non-empty subsets $V_{\sigma_1}, \dots, V_{\sigma_s}$, indexed by signatures $\sigma_1, \dots, \sigma_s$, such that V_{σ_i} is the subset of V consisting of all (x, y) having signature σ_i . This decomposition will be called the σ -decomposition of V , it is unique up to order.

In this section, we give an algorithm that computes the σ -decomposition of V ; we work under the assumption that F, G are in general position. This assumption implies that there exist polynomials $(P, S) = \text{SL}(F, G)$ in $\mathbb{K}[X]$ such that the defining ideal of V (which is by definition a radical ideal) admits the generators $\langle P(X), Y - S(X) \rangle$.

Starting by computing P and S , we return tuples $[(P_i, S_i, m_i, \mathbf{H}_i, n_i, a_i, \mathbf{K}_i)]_{1 \leq i \leq t}$, where all P_i 's and S_i 's are in $\mathbb{K}[X]$, and for each i , the defining ideal of V_{σ_i} is $\langle P_i(X), Y - S_i(X) \rangle$, with $\sigma_i = (m_i, \mathbf{H}_i, n_i, a_i, \mathbf{K}_i)$. By a slight abuse of notation, we still call this sequence the σ -decomposition of V , and we denote it by $\sigma\text{-dec}(F, G)$ (as above, it is uniquely defined up to order only).

The first main result of this section is the following complexity bound on this calculation, when working over a finite field.

Proposition 4. *Suppose that F and G are polynomials in $\mathbb{K}[X, Y]$, with no nontrivial common factor and in general position. There exists an algorithm σ -decomposition that takes as input F, G and $\text{SL}(F, G)$, and returns the σ -decomposition of $V(F, G)$. When $\mathbb{K} = \mathbb{F}_p$, this algorithm can be implemented so as to take $d^{3+\varepsilon} O(\log(p))$ bit operations.*

In addition, when $\mathbb{K} = \mathbb{Q}$ and F, G have coefficients in \mathbb{Z} , we give conditions under which the computation reduces well at a prime p . The data $\sigma\text{-dec}(F, G)$ consists of a sequence of polynomials, integers and strings; by *reducing* such an object modulo p , we refer to the sequence obtained by reducing the coefficients of all polynomials in $\sigma\text{-dec}(F, G)$ modulo p , if no denominator vanishes. We denote this new sequence $\sigma\text{-dec}(F, G) \bmod p$.

Proposition 5. *There exists an explicitly computable function $\Delta_3(d, h, \ell) = (dh\ell)^{O(1)}$ such that the following holds.*

Suppose that F and G are polynomials in $\mathbb{Z}[X, Y]$, with no nontrivial common factor and in general position, with degree at most d and height at most h . Suppose as well all polynomials appearing in $\text{SL}(W)$, for any subset W of $V(F, G)$ defined over \mathbb{Q} , have height at most ℓ . Then, there exists a nonzero integer δ_3 such that:

- δ_3 has height at most $\Delta_3(d, h, \ell)$;
- for any prime p that satisfies the following conditions:
 - p does not divide δ_3 ,
 - $\text{SL}(F, G) \bmod p = \text{SL}(F \bmod p, G \bmod p)$,
 - for any subset W of $V(F, G)$ defined over \mathbb{Q} , p cancels no denominator in either $\text{SL}(W)$ or $\text{RUR}(W)$,

the equality $\sigma\text{-dec}(F, G) \bmod p = \sigma\text{-dec}(F \bmod p, G \bmod p)$ holds.

3.6.1 Computing all m_i 's and H_i 's

In order to motivate the general algorithm, we first briefly explain how to compute the integer m and string H at a rational point $(x, y) \in \mathbb{K}^2$ of $V(F, G)$, assuming such a point exists. In this case, the process is straightforward: simply evaluate all required derivatives at (x, y) , and stop as soon as we find a nonzero value. This is detailed in Algorithm `m_H_rational`, where we use a function `zero_index((x, y), [r1, ..., rN])` that returns the smallest index i such that $r_i(x, y)$ vanishes (with indices starting at one).

Algorithm 4: `m_H_rational`(F, G, x, y)

Input: (F, G) in $\mathbb{K}[X, Y]$, a point (x, y) in $V = V(F, G)$

Output: (m, H)

```

1  $d = \max(\deg(F), \deg(G))$ 
2  $R = [\frac{\partial F}{\partial Y}, \frac{\partial G}{\partial Y}, \dots, \frac{\partial^d F}{\partial Y^d}, \frac{\partial^d G}{\partial Y^d}]$ 
3  $n = \text{zero\_index}((x, y), R)$ 
4 if  $n$  is odd then
5   | return  $((n + 1)/2, "F")$ 
6 else
7   | return  $(n/2, "G")$ 
8 end

```

Given the Shape Lemma representation (P, S) of V , we follow the same approach. The only significant difference is that zero-tests are replaced by the splitting mechanism of Algorithm `zero_index`.

To describe the output, note that we can partition V into subsets $V_{m_1, H_1}, \dots, V_{m_s, H_s}$, for pairwise distinct (m_i, H_i) , where V_{m_i, H_i} is the subset of V consisting of all (x, y) such that $\sigma(x, y) = (m_i, H_i, \dots)$. The output of the following algorithm `compute_m_H` is the sequence $[(P_i, S_i, m_i, H_i)]_{1 \leq i \leq s}$ such that (P_i, S_i) is the Shape Lemma representation of V_{m_i, H_i} (this output, just like the partition $V_{m_1, H_1}, \dots, V_{m_s, H_s}$, is uniquely defined up to order).

In particular, notice that for all i , $\frac{\partial^{m_i-1} H_i}{\partial Y^{m_i-1}}(X, S_i) = 0$ modulo P_i and $\frac{\partial^m H_i}{\partial Y^m}(X, S_i)$ is a unit modulo P_i .

Algorithm 5: compute_m_H(F, G, P, S)

Input: (F, G) in $\mathbb{K}[X, Y]$, the Shape Lemma representation (P, S) of $V = V(F, G)$

Output: a sequence $[(P_i, S_i, m_i, H_i)]_{1 \leq i \leq s}$

```
1  $d = \max(\deg(F), \deg(G))$ 
2  $R_0 = [\frac{\partial F}{\partial Y}, \frac{\partial G}{\partial Y}, \dots, \frac{\partial^d F}{\partial Y^d}, \frac{\partial^d G}{\partial Y^d}]$ 
3  $R = [r \bmod \langle P, Y - S \rangle \mid r \in R_0]$ 
4  $K = \text{zero\_index}(P, R)$   $K$  is a sequence of the form  $[(P_i, n_i)]$ 
5  $W = []$ 
6 for  $(P_i, n_i)$  in  $K$  do
7    $S_i = S \bmod P_i$ 
8   if  $n_i$  is odd then
9      $\text{append}(P_i, S_i, (n_i + 1)/2, \text{"F"})$  to  $W$ 
10  else
11     $\text{append}(P_i, S_i, n_i/2, \text{"G"})$  to  $W$ 
12  end
13 end
14 return  $S$ 
```

Lemma 17. *Algorithm compute_m_H is correct. When $\mathbb{K} = \mathbb{F}_p$, one can implement it so that to take $d^{3+\varepsilon}O(\log(p))$ bit operations.*

PROOF. Correctness of the algorithm directly follows from the correctness of zero_index, and the fact that all m_i 's are at most d , as proved in the deflation lemma.

For the complexity analysis in the particular case $\mathbb{K} = \mathbb{F}_p$, we know that the cost of zero_index is quasi-linear, so all that matters is the cost of computing polynomials R , at steps 2 and 3.

This is achieved by calling Algorithm normal_forms of Proposition 3, with input $t = 1$, and L, L', L'', F , where L is the list $[(0, d)]$, L' is the list $[P]$ and L'' is the list $[S]$. In order to satisfy the required assumption H_{NF} , let us write $d' = \lceil d^{3/2} \rceil$; then, the product $me = de$ admits the upper bound $d^3 = O(d'^2)$, so we are under the assumptions of that proposition, up to replacing d by d' . As noted in Section 3.4, the output (F_1) of this algorithm takes the form

$$F_1 = \sum_{\mu=0}^d \frac{1}{\mu!} D_\mu \zeta^\mu;$$

since the index n was chosen to be zero, the entry of L'' depends only on X , D_μ writes

$$D_\mu = \frac{\partial^\mu F}{\partial Y^\mu}(X, S) \bmod \langle P(X) \rangle,$$

so it gives half the polynomials we wanted. Doing the same with G , we obtain all normal forms we required.

In terms of complexity, for any $\varepsilon > 0$, calling Proposition 3 can be done in $d^{2+\varepsilon}O^-(\log(p))$ bit operations; this is $d^{3+\varepsilon}O^-(\log(p))$, as claimed. All other costs are negligible.

Suppose that we are over $\mathbb{K} = \mathbb{Q}$, and that F and G are in $\mathbb{Z}[X, Y]$. The following discussion gives conditions under which the above calculation admits a good reduction at a prime p .

Lemma 18. *There exists an explicitly computable function $\Delta_{3,1}(d, h, \ell) = (dhl)^{O(1)}$ such that the following holds.*

Suppose that F and G are polynomials in $\mathbb{Z}[X, Y]$, with no nontrivial common factor and in general position, with degree at most d and height at most h . Suppose as well that P and S have height at most ℓ . There exists a nonzero integer $\delta_{3,1}$ such that:

- $\delta_{3,1}$ has height at most $\Delta_{3,1}(d, h, \ell)$;
- for any prime p that satisfies the following conditions:
 - p does not divide $\delta_{3,1}$,
 - $\text{SL}(F, G) \bmod p = \text{SL}(F \bmod p, G \bmod p)$,
 - for any subset W of $V(F, G)$ defined over \mathbb{Q} , p cancels no denominator in $\text{SL}(W)$,

the sequence obtained from $\text{compute_m_H}(F, G, P, S) \bmod p$ coincides with the output of $\text{compute_m_H}(F \bmod p, G \bmod p, P \bmod p, S \bmod p)$.

PROOF. Let $[(P_i, S_i, m_i, H_i)]_{1 \leq i \leq s}$ be the output of $\text{compute_m_H}(F, G, P, S) \bmod p$. For a given index i in $\{1, \dots, s\}$, the corresponding integer m_i is characterized as follows: for each entry, say H , of index less than m_i in the sequence $[\frac{\partial F}{\partial Y}, \frac{\partial G}{\partial Y}, \dots, \frac{\partial^d F}{\partial Y^d}, \frac{\partial^d G}{\partial Y^d}]$, $H(X, S_i(X)) = 0 \bmod P_i$; for the entry H_i , $\gcd(H_i(X, S_i(X)), P_i) = 1$. This latter condition is equivalent to H_i vanishing nowhere on $V(P_i(X), Y - S_i(X))$.

Thus, the polynomials P_i, S_i, H_i satisfy conditions C_0 and C_1 of Proposition 1. We claim that we can take for $\delta_{3,1}$ the product of the integers δ_1 associated by that proposition to the systems P_i, S_i, H_i , for $i = 1, \dots, s$.

Let then p be a prime such that $(P \bmod p, S \bmod p) = \text{SL}(F \bmod p, G \bmod p)$ and such that for any subset W of $V(F, G)$ defined over \mathbb{Q} , p cancels no denominator in $\text{SL}(W)$; assume as well that p does not divide $\delta_{3,1}$.

Then, all P_i 's and S_i 's can be reduced modulo p ; besides, because P remains squarefree modulo p , this is also the case for all P_i 's. Thus, the polynomials $[(P_i \bmod p, S_i \bmod p)]_{1 \leq i \leq s}$ form the Shape Lemma representations of *some* partition of $V(F \bmod p, G \bmod p)$. It remains to see whether this is the same partition as the one induced by running the algorithm over \mathbb{F}_p , with input $(F, G, P, S) \bmod p$.

The calculation of Algorithm 14 reduces well modulo p as soon as `zero_index` does (all other steps clearly admits a good reduction modulo p). In view of the discussion in the first paragraph, we are led to consider how the relations $H(X, S_i(X)) = 0 \bmod P_i$ or $H(X_i, S_i(X)) = 0 \bmod P_i$ that hold over \mathbb{Q} reduce modulo p .

Of course, a relation of the form $H(X, S_i(X)) = 0 \bmod P_i$ will remain true modulo p , for any p which both sides make sense. The more delicate question is whether the relation $\text{gcd}(H(X, S_i(X)), P_i) = 1$ remains true after reduction. Proposition 1 shows that as soon as p does not divide the integer δ_1 associated to P_i, S_i, H_i , the gcd remains one modulo p , as requested.

Thus, our claims are proved, except for the upper bound on the height $\delta_{3,1}$. This follows directly from Proposition 1, and the fact that there are at most d^2 indices i to take into account.

3.6.2 Computing all J_i 's

Suppose that we have determined the sequence $[(P_i, S_i, m_i, H_i)]_{1 \leq i \leq s}$ of the previous subsection; we now want to compute a power series J as defined in Section 3.5. Compared to that section, there is a slight difference: we are not working at a point (x, y) with coordinates in $\overline{\mathbb{K}}^2$, but with points given through Shape Lemma representations.

Let us thus fix an index i in $\{1, \dots, s\}$. Associated to P_i , one can define the ring $\mathbb{B}_i = \mathbb{K}[X]/\langle P_i \rangle$; this is in general not a field, but only a product of fields. Two elements will be highlighted in \mathbb{B}_i : the residue class x_i of X , and the residue class y_i of $T_i(X)$. Thus, by construction, $F(x_i, y_i) = G(x_i, y_i) = 0$ (where F and G are viewed as polynomials in \mathbb{B} , through the injection $\mathbb{K} \rightarrow \mathbb{B}$). In addition, the polynomial H_i associated to P_i and T_i is such that

$$\frac{\partial^{m_i-1} H_i}{\partial Y^{m_i-1}}(x_i, y_i) = 0 \text{ in } \mathbb{B}_i \quad \text{and} \quad \frac{\partial^{m_i} H_i}{\partial Y^{m_i}}(x_i, y_i) \text{ is a unit in } \mathbb{B}_i. \quad (3.4)$$

This is sufficient for us to apply Newton iteration, and compute a power series J_i in

$\mathbb{B}_i[[\xi]]$ such that

$$\frac{\partial^{m_i} H_i}{\partial Y^{m_i}}(x_i + \xi, J_i) = 0 \quad \text{and} \quad J_i(0) = y_i.$$

The following algorithm describes this process. This algorithm will be used in a slightly more general context: instead of taking as input the exact polynomials P_i and S_i computed in the previous section, we will as well call it using only *a factor* of the actual polynomial P_i (with S_i being correspondingly reduced modulo this factor); this will not change the analysis. These polynomials will be written (C_i, T_i) instead of (P_i, S_i) .

As input, this algorithm also takes extra parameters n_i , which give the required precision in ξ for the power series J_i .

Algorithm 6: compute $\mathcal{J}(F, G, [(C_i, T_i, m_i, H_i, n_i)]_{1 \leq i \leq s})$

Input: F, G , a sequence of polynomials C_i and T_i in $\mathbb{K}[X]$, strings H_i and indices m_i and n_i

Output: a sequence $[J_i]_{1 \leq i \leq s}$, where $J_i \in \mathbb{B}_i[[\xi]]$ is known modulo ξ^{n_i} and satisfies (3.4)

```

1   $\lambda = 1$ 
2   $[J_i]_{1 \leq i \leq s} = [T_i]_{1 \leq i \leq s}$ 
3   $I = [i \mid 1 \leq i \leq s \text{ and } \lambda < n_i]$ 
4   $I_F = [i \mid 1 \leq i \leq s \text{ and } H_i = \text{"F"}]$ 
5   $I_G = [i \mid 1 \leq i \leq s \text{ and } H_i = \text{"G"}]$ 
6  while  $I$  is not empty do
7       $[\eta_i]_{i \in I} = [\frac{\partial^{m_i-1} F}{\partial Y^{m_i-1}}(X + \xi, J_i) \bmod \langle C_i(X), \xi^{2\lambda} \rangle]_{i \in I \cap I_F}$  cat
8           $[\frac{\partial^{m_i-1} G}{\partial Y^{m_i-1}}(X + \xi, J_i) \bmod \langle C_i(X), \xi^{2\lambda} \rangle]_{i \in I \cap I_G}$ 
9       $[\eta'_i]_{i \in I} = [\frac{\partial^{m_i} F}{\partial Y^{m_i}}(X + \xi, J_i) \bmod \langle C_i(X), \xi^{2\lambda} \rangle]_{i \in I \cap I_F}$  cat
10          $[\frac{\partial^{m_i} G}{\partial Y^{m_i}}(X + \xi, J_i) \bmod \langle C_i(X), \xi^{2\lambda} \rangle]_{i \in I \cap I_G}$ 
11     for  $i$  in  $I$  do
12          $J_i = J_i - \eta/\eta' \bmod \langle C_i(X), \xi^{2\lambda} \rangle$ 
13     end
14      $\lambda = 2\lambda$ 
15      $I = [i \mid 1 \leq i \leq s \text{ and } \lambda < n_i]$ 
16 end
17 return  $[J_i \bmod \xi^{n_i}]_{1 \leq i \leq s}$  we may know  $J_i$  at a slightly higher precision than  $n_i$ 

```

Lemma 19. *Algorithm compute \mathcal{J} is correct. When $\mathbb{K} = \mathbb{F}_p$, for any $\varepsilon > 0$, one can implement it so that it takes $d^{2+\varepsilon} O(\log(p))$ bit operations, provided the inequality $\sum_{1 \leq i \leq s} (n_i + 1)(m_i + 1) \deg(C_i) = O(d^2)$ holds.*

PROOF. The algorithm essentially implements Newton iteration, over all $\mathbb{B}_i[[\xi]]$ independently. We saw that by construction, for all i , $\frac{\partial^{m_i-1} H_i}{\partial Y^{m_i-1}}(x_i, y_i) = 0$ in \mathbb{B}_i and $\frac{\partial^{m_i} H_i}{\partial Y^{m_i}}(x_i, y_i)$ is a unit in \mathbb{B}_i ; thus, we can indeed run Newton iteration. The sequence I indicates the indices for which we have not reached the required precision yet; these are the indices for which we do further iteration steps. Sequences I_F and I_G indicate which indices use F or G to do the lifting.

It remains to do the cost analysis, in the case where $\mathbb{K} = \mathbb{F}_p$; all the cost is spent in the main loop (at the beginning, the T_i 's are already reduced modulo the respective C_i 's; at the end, truncation is free).

First, remark that the highest value λ will reach will be $O(\max_i n_i)$, which is $O(d^2)$ by assumption. As a consequence, the number of times we will enter the loop is $O(\log(d))$, which we will be able to absorb in the term $d^{2+\varepsilon}$. Thus, we can focus on the cost of a single pass through the loop.

The inversion and multiplication at Step 12 take $O(\sum_{i \in I} \deg(C_i) \lambda)$ operations in \mathbb{F}_p , or $O(\sum_{i \in I} \deg(C_i) \lambda \log(p))$ bit operations. The most delicate steps are 7 and 9. To keep their cost admissible, we use algorithm `normal_forms` to compute the values η_i and η'_i , simultaneously for all indices i in I . We call this algorithm twice: once for the indices i in I_F , then for those indices i in I_G ; it is enough to analyze the cost for, say, F .

We call algorithm `normal_forms` with an input size t_F (the cardinality of $I \cap I_F$), lists L, L', L'' and polynomial F ; we set $L = [(2\lambda - 1, m_i)]_{i \in I \cap I_F}$, $L' = [C_i]_{i \in I \cap I_F}$ and $L'' = [J_i]_{i \in I \cap I_F}$. The input size satisfies

$$\sum_{i \in I \cap I_F} 2\lambda(m_i + 1) \deg(C_i) \leq \sum_{i \in I \cap I_F} (4n_i + 1)(m_i + 1) \deg(C_i),$$

which is $O(d^2)$ by assumption. Thus, for any $\varepsilon > 0$, we can compute

$$D_{i,\mu} = \frac{\partial^\mu F}{\partial Y^\mu}(X + \xi, J_i) \bmod \langle C_i(X), \xi^{2\lambda} \rangle,$$

for all i in $I \cap I_F$ and $\mu = 0, \dots, m_i$, using $d^{2+\varepsilon} O(\log(p))$ bit operations. Keeping those derivatives of order m_i and m_{i-1} gives us the requires values η_i and η'_i .

When $\mathbb{K} = \mathbb{Q}$, all computations can be reduced modulo p , for any prime p that satisfies the assumptions of Lemma 18. Indeed, for such a p and for i in $\{1, \dots, s\}$, $\frac{\partial^{m_i} H_i}{\partial Y^{m_i}}(X, S_i)$ is a unit modulo $P_i \bmod p$; this remains true for any factor C_i of P_i .

3.6.3 Computing all n_i 's, a_i 's and K_i 's

Finally, we want to compute the values of n , a and K at all points in V . As input, we start from the sequence $[(P_i, S_i, m_i, H_i)]_{1 \leq i \leq s}$ computed in Section 3.6.1; recall that these sequences define the partition of V into sets $(V_{m_i, H_i})_{1 \leq i \leq s}$

The σ -partition of V that we wish to compute is a refinement of the partition $(V_{m_i, H_i})_{1 \leq i \leq s}$; in other words, we obtain it by partitioning each V_{m_i, H_i} into subsets $(V_{\sigma_{i,j}})_{j \in D_i}$, for some index set D_i ; each $\sigma_{i,j}$ takes the form $\sigma_{i,j} = (m_i, H_i, n_{i,j}, a_{i,j}, K_{i,j})$. Our output will consist in a similarly indexed array of the form $[(C_{i,j}, T_{i,j}, m_i, H_i, n_{i,j}, a_{i,j}, K_{i,j})]_{1 \leq i \leq s, j \in D_i}$, such that for all i, j , $(C_{i,j}, T_{i,j})$ is the Shape Lemma representation of $V_{\sigma_{i,j}}$.

To describe the idea the algorithm, we can fix the index i . Then, we need to compute the power series J_i defined in the previous section, and deduce the expansions of $\frac{\partial^\mu F}{\partial Y^\mu}(X + \xi, J_i)$ and $\frac{\partial^\mu G}{\partial Y^\mu}(X + \xi, J_i)$ for suitable values of μ ; this will be done at successive precisions $\lambda = 1, 2, 4, \dots$ in ξ .

Suppose we have obtained these expansions modulo ξ^λ . If we were over a field, we would then look for the expansion with smallest valuation in ξ ; however, we are working over $\mathbb{B}_i = \mathbb{K}[X]/\langle P_i \rangle$, which is not necessarily a field. Thus, we apply Algorithm `zero_index_vectorial` of Section 3.3; it returns factors of P_i for which we have found the correct valuation, together with possibly a residual factor, for which we have to increase the precision in ξ . Thus, we replace P_i by this factor, multiply λ by 2, and start over. In order to distinguish between the input P_i 's and their factors, we use new variables called C_i as our current polynomials.

The following algorithm implements this idea; the fact that we have to handle as well the strings H_i to decide with partial derivatives to consider makes for some admittedly heavy bookkeeping (which we explain in the proof of the following lemma). In the pseudo-code, we use the subroutine `cf`(P, ξ^j) which returns the coefficient of ξ^j in polynomial P ; further subroutines `infinity`, `index_of` and `polynomial`, that are only designed for said bookkeeping purposes, are explained in the proof of the lemma.

Algorithm 7: compute_n_a_K $(F, G, [(P_i, S_i, H_i, m_i)]_{1 \leq i \leq s})$

Input: the sequence $[(P_i, S_i, m_i, H_i)]_{1 \leq i \leq s}$ computed in Section 3.6.1

Output: a sequence $[(C_{i,j}, T_{i,j}, m_i, H_i, n_{i,j}, a_{i,j}, K_{i,j})]_{1 \leq i \leq s, j \in D_i}$

```
1  $\lambda = 1$ 
2  $L = []$ 
3  $I = [1, \dots, s]$ 
4  $I_F = [i \mid 1 \leq i \leq s \text{ and } H_i = \text{"F"}]$ 
5  $I_G = [i \mid 1 \leq i \leq s \text{ and } H_i = \text{"G"}]$ 
6  $[C_i, T_i]_{i \in I} = [P_i, S_i]_{i \in I}$ 
7 while  $I$  is not empty do
8    $[J_i]_{i \in I} = \text{compute\_J}(F, G, [C_i, T_i, H_i, m_i, 2\lambda]_{i \in I})$ 
9    $[\eta_{i,\alpha}]_{i \in I, \alpha \in [0, \dots, m_i]} = [\frac{\partial^\alpha F}{\partial Y^\alpha}(X + \xi, J_i) \bmod \langle C_i(X), \xi^{2\lambda} \rangle]_{i \in I, \alpha \in [0, \dots, m_i]}$ 
10   $[\gamma_{i,\alpha}]_{i \in I, \alpha \in [0, \dots, m_i]} = [\frac{\partial^\alpha G}{\partial Y^\alpha}(X + \xi, J_i) \bmod \langle C_i(X), \xi^{2\lambda} \rangle]_{i \in I, \alpha \in [0, \dots, m_i]}$ 
11  for  $i$  in  $I$  do
12    if  $i$  is in  $I_F$  then
13       $R_i =$ 
14       $[[\text{cf}(\eta_{i,0}, \xi^j), \dots, \text{cf}(\eta_{i,m_i-2}, \xi^j)] \text{ cat } [\text{cf}(\gamma_{i,0}, \xi^j), \dots, \text{cf}(\gamma_{i,m_i-1}, \xi^j)]]_{j=1, \dots, 2\lambda-1}$ 
15    else
16       $R_i =$ 
17       $[[\text{cf}(\gamma_{i,0}, \xi^j), \dots, \text{cf}(\gamma_{i,m_i-2}, \xi^j)] \text{ cat } [\text{cf}(\eta_{i,0}, \xi^j), \dots, \text{cf}(\eta_{i,m_i-1}, \xi^j)]]_{j=1, \dots, 2\lambda-1}$ 
18     $L_i = \text{zero\_index\_vectorial}(C_i, R_i)$   $L_i$  has the form  $[(C_{i,j}, (n_{i,j}, \ell_{i,j}))]_{j \in D_i}$ 
19     $C_i = \text{infinity}(L_i)$ 
20    if  $C_i$  is not constant then
21      remove  $C_i$  from  $L_i$  update  $C_i$  and  $T_i$ 
22       $T_i = T_i \bmod C_i$ 
23    else
24      remove  $i$  from  $I$  we are done with this index
25       $T_{i,j} = [T_i \bmod A_{i,j}]_{A_{i,j} \in L_i}$ 
26       $L = L \text{ cat } [(C_{i,j}, T_{i,j}, m_i, H_i, n_{i,j}, \text{index\_of}(n_{i,j}, R_i), \text{polynomial}(n_{i,j}, R_i))]_{j \in D_i}$ 
27    end
28   $\lambda = 2\lambda$ 
29 end
30 return  $L$ 
```

Lemma 20. Algorithm compute_n_a_K terminates and is correct. When $\mathbb{K} = \mathbb{F}_p$, for any $\varepsilon > 0$, one can implement it so that it takes $d^{2+\varepsilon} O^{\sim}(\log(p))$ bit operations.

PROOF. To establish correctness, we first prove that the following invariant is preserved throughout the for loop: at the beginning of the loop,

- for all indices i in I , (C_i, S_i) is the Shape Lemma representation of the union of all subsets $V_{\sigma_{i,j}}$, for $\sigma_{i,j}$ of the form $\sigma_{i,j} = (m_i, H_i, n, a, K)$, for some $n \geq \lambda$.
- L contains the entries $[(C_{i,j}, T_{i,j}, m_i, H_i, n_{i,j}, a_{i,j}, K_{i,j})]_{i,j}$, for all indices i in $\{1, \dots, t\}$ and j in D_i such that $n_{i,j} < \lambda$.

Initially, $\lambda = 0$; since all $n_{i,j}$ are at least equal to one, our loop invariant holds. Supposing that we maintained the invariant up to some exponent λ , we prove that they will be maintained through the next pass in the loop.

Step 8 computes the sequence $[J_i]_{i \in I}$; those are power series known modulo $\xi^{2\lambda}$, such that, for all i ,

$$\frac{\partial^{m_i} H_i}{\partial Y^{m_i}}(X + \xi, J_i) = 0 \pmod{\xi^{2\lambda}} \quad \text{and} \quad J_i(0) = T_i$$

holds modulo C_i .

Consider an index i in I ; without loss of generality, we assume that I is in I_F , that is, that H_i is "F". In this case, Lemma 15 shows that for any (x, y) in $V(C_i, Y - S_i)$, the integer n appearing in its signature satisfies

$$n = \min \left(\left\{ \text{val} \left(\frac{\partial^\alpha F}{\partial Y^\alpha}(x + \xi, J_i(x, \xi)) \right) \right\}_{0 \leq \alpha < m_i - 1} \cup \left\{ \text{val} \left(\frac{\partial^\alpha G}{\partial Y^\alpha}(x + \xi, J_i(x, \xi)) \right) \right\}_{0 \leq \alpha < m_i} \right),$$

where $J_i(x, \xi)$ denotes the (truncated) power series in $\overline{\mathbb{K}}[[\xi]]$ obtained by evaluating X at x in J_i (this is valid, since J has coefficients in $\mathbb{K}[X]/\langle C_i \rangle$, and x is a root of C_i). In view of the calculations at Steps 9 and 10, we see that n can be rewritten as

$$n = \min \left(\left\{ \text{val}(\eta_{i,\alpha}(x)) \right\}_{0 \leq \alpha < m_i - 1} \cup \left\{ \text{val}(\gamma_{i,\alpha}(x)) \right\}_{0 \leq \alpha < m_i} \right);$$

if H_i was equal to "G", indices would be changed here. Remark that we know the power series $\eta_{i,\alpha}$ and $\gamma_{i,\alpha}$ modulo $\xi^{2\lambda}$; on the other hand, since we are still dealing with C_i , their valuation must be at least λ . The sequence R_i then precisely contains the coefficients of power series $\eta_{i,\alpha}$ and $\gamma_{i,\alpha}$ that we have to test for zero at the roots x of C_i .

The call to `zero_index_vectorial`(C_i, R_i) returns a sequence $L_i = [(C_{i,j}, (n_{i,j}, \ell_{i,j}))]_{j \in D_i}$, such that $C_i = \prod_{j \in D_i} C_{i,j}$, with either $(n_{i,j}, \ell_{i,j})$ in $\{1, \dots, 2\lambda - 1\} \times \{1, \dots, 2m_i - 1\}$ (the former is the length of the entries in R_i , the latter the length of R_i itself) or $(n_{i,j}, \ell_{i,j}) = (\infty, -1)$; the roots of $C_{i,j}$ are the roots x of C_i having $n = n_{i,j}$, when $n_{i,j} < \infty$, or for which all we can say is $n \geq 2\lambda$, when $n_{i,j} = \infty$.

If $n_{i,j} = \infty$ does not show up, we have found the valuation for all roots of C_i ; otherwise, the roots corresponding to $n_{i,j} = \infty$ will have to enter the next pass in the for loop. This is decided at Step 17, where subroutine `infinity` extracts the entry $(C_{i,j}, (n_{i,j}, \ell_{i,j}))$ in L_i having $n_{i,j} = \infty$, if such an entry exists, and replaces C_i by this polynomial $C_{i,j}$. If no such entry exists, `infinity` returns and assign 1 to C_i .

If the new value of C_i has positive degree, we update T_i as well, so that (C_i, T_i) is the Shape Lemma representation of all roots of P_i having $n \geq 2\lambda$ — this proves that the first half of our loop invariant will be satisfied for the next iteration. If the new value of C_i is equal to 1, we are done with all roots of P_i , so we can remove i from index set I .

It remains to update the sequence L with those entries of L_i corresponding to $n_{i,j} < \infty$. For all these entries, index $\ell_{i,j} \in \{1, \dots, 2m_i - 1\}$ tells us which polynomial in R_i yielded a nonzero value. Subroutines `index_of` and `polynomial` deduce the corresponding index $\alpha_{i,j}$ (in either $\{0, \dots, m_i - 1\}$ or $\{0, \dots, m_i\}$), and the corresponding string $K_{i,j}$ indicates whether the nonvanishing occurred for one of the $\eta_{i,\alpha}$'s or $\gamma_{i,\alpha}$'s. This construction shows that the second half of our loop invariant will be satisfied for the next iteration, so we are done with our induction proof.

Next, remark that the algorithm terminates: indeed, all $n_{i,j}$ satisfy the crude upper bound $n_{i,j} \leq d^2$. Our loop invariant then proves that the output of the algorithm is correct.

It remains to do the cost analysis, when $\mathbb{K} = \mathbb{F}_p$. The number of passes through the for loop is $O(\log(d))$. Let us consider a given pass through the for loop, for some precision λ , and let us write $c_i = \deg(C_i)$ for i in I . The crucial inequality to notice is that

$$\sum_{i \in I} (2\lambda + 1)(m_i + 1)c_i = O(d^2).$$

Indeed, for all indices i remaining in I at this stage, the index n of any root x of C_i is at least λ (as per our loop invariant). For such a root x , by the deflation lemma, the product of the indices m and n is at most $\mu((F, G), (x, S(x)))$, the intersection multiplicity of F and G at $(x, S(x)) \in V$. The summand above is $O(\lambda m_i c_i)$, and the sum over all $i \in I$ of this quantity is thus at most $\sum_{(x,y) \in V} \mu((F, G), (x, y))$, which we know to be at most d^2 .

As a consequence, we can apply Lemma 19, which proves that the cost of computing all J_i is $d^{2+\varepsilon} O(\log(p))$ bit operations. The computation of all $[\eta_{i,\alpha}]_{i \in I, \alpha \in [0, \dots, m_i]}$ is handled by Algorithm `normal_forms`, with input lists L, L', L'' given by $L = [(2\lambda - 1, m_i)]_{i \in I}$, $L' = [C_i]_{i \in I}$ and $L'' = [J_i]_{i \in I}$; in view of Proposition 3, the cost is again $d^{2+\varepsilon} O(\log(p))$ bit operations. All other arithmetic operations (remainder at Step 20 and multiple remainders at Step 23) are cheaper, so we are done.

The main algorithm of this section, Algorithm σ -dec, is simply the combination of `compute_m_H`, `compute_J` and `compute_n_a_K`. Combining the results of Lemmas 17, 19 and 20 proves the complexity statement in Proposition 4. It remains to prove Proposition 5.

Let us thus assume that $\mathbb{K} = \mathbb{Q}$. Let p be a prime that satisfies the assumptions of Lemma 18. Then, the computations in Algorithms `compute_m_H` and `compute_J` reduce well modulo p . Consider now the output $[(C_{i,j}, T_{i,j}, m_i, H_i, n_{i,j}, a_{i,j}, K_{i,j})]_{1 \leq i \leq s, j \in D_i}$ of `compute_n_a_K`.

Let us slightly simplify the discussion using the following argument. Consider a component $(C_{i,j}, T_{i,j}, m_i, H_i, n_{i,j}, a_{i,j}, K_{i,j})$ of the σ -decomposition of $V(F, G)$. Running Algorithm `compute_n_a_K` on input F, G and the sequence consisting only of $[(C_{i,j}, T_{i,j}, m_i, H_i)]$ returns again $(C_{i,j}, T_{i,j}, m_i, H_i, n_{i,j}, a_{i,j}, K_{i,j})$, since this set is its own σ -decomposition. Thus, it suffices to give conditions under which this remains the case modulo p .

With such an input, Algorithm `compute_n_a_K` performs a series of zero-test on the entries of a sequence of the form

$$R_i = [[\text{cf}(\eta_{i,0}, \xi^j), \dots, \text{cf}(\eta_{i,m_i-2}, \xi^j)] \text{ cat } [\text{cf}(\gamma_{i,0}, \xi^j), \dots, \text{cf}(\gamma_{i,m_i-1}, \xi^j)]]_{j=1, \dots, 2\lambda-1},$$

for increasing values of λ . Each such test amounts to verify whether one of the coefficients, of the form either $\text{cf}(\eta_{i,k})$ or $\text{cf}(\gamma_{i,k})$, is zero or invertible modulo $C_{i,j}$. All are found to be zero, until we discover one of them to be invertible modulo $C_{i,j}$. In view of the structure of the output, we know that this must be the coefficient of $\xi^{n_{i,j}}$ in $\frac{\partial^{a_{i,j}} K_{i,j}}{\partial Y^{a_{i,j}}}$.

Lemma 16 now implies that all points in $V(C_{i,j}(X), Y - T_{i,j}(X))$ must be roots of multiplicity $n_{i,j}$ of the system $(\frac{\partial^{m_i} H_i}{\partial Y^{m_i}}, \frac{\partial^{a_{i,j}} K_{i,j}}{\partial Y^{a_{i,j}}})$. The polynomials $C_{i,j}, T_{i,j}, \frac{\partial^{m_i} H_i}{\partial Y^{m_i}}, \frac{\partial^{a_{i,j}} K_{i,j}}{\partial Y^{a_{i,j}}}$ thus satisfy the assumptions of Proposition 2.

We deduce that if p does not divide the integer δ_2 associated to $C_{i,j}, T_{i,j}, \frac{\partial^{m_i} H_i}{\partial Y^{m_i}}, \frac{\partial^{a_{i,j}} K_{i,j}}{\partial Y^{a_{i,j}}}$, the multiplicity of $(\frac{\partial^{m_i} H_i}{\partial Y^{m_i}}, \frac{\partial^{a_{i,j}} K_{i,j}}{\partial Y^{a_{i,j}}})$ at any root of $C_{i,j}(X), Y - T_{i,j}(X)$ taken modulo p remains $n_{i,j}$. As a result, applying again Lemma 16 shows that the corresponding coefficient of either $\eta_{i,k}$ or $\gamma_{i,j}$ remains invertible modulo $C_{i,j} \bmod p$, in $\mathbb{F}_p[X]$. Thus, Algorithm `compute_n_a_K` behaves in the same manner modulo p as over \mathbb{Q} , and we are done.

Taking all $C_{i,j}$ into account, we see that to ensure success it is sufficient to ask that p divides none of the integers δ_2 associated to the systems $C_{i,j}, T_{i,j}, \frac{\partial^{m_i} H_i}{\partial Y^{m_i}}, \frac{\partial^{a_{i,j}} K_{i,j}}{\partial Y^{a_{i,j}}}$. If H and K have degree at most d and height at most h , there are at most d^2 such systems; all indices m_i and $n_{i,j}$ are at most d^2 , so the degree and height of polynomials $\frac{\partial^{m_i} H_i}{\partial Y^{m_i}}$ and $\frac{\partial^{a_{i,j}} K_{i,j}}{\partial Y^{a_{i,j}}}$ are respectively at most d and $O^-(h+d)$. If we assume that $C_{i,j}, T_{i,j}$ have height

at most ℓ , the bounds given in Proposition 2 show that the product of all δ_2 's we consider admits an explicitly computable upper bound of the form $(dh\ell)^{O(1)}$. Together with the bound given in Lemma 18, this finishes the proof of Proposition 5.

3.7 Newton iteration

In this section, we give the details of a deflated Newton iteration that follows naturally from the deflation lemma. The following is essentially a particular case of the general algorithm from [40]; however, we chose to give a self-contained proofs of the result we need.

Let \mathbb{B} be a ring and let (x^*, y^*) be in \mathbb{B} , as well as $m \in N$ and $H \in \mathbb{B}[X, Y]$ be such that the following holds:

$$\mathbf{X}_1. \quad \frac{\partial^{m-1}H}{\partial Y^{m-1}}(x^*, y^*) = 0.$$

$$\mathbf{X}_2. \quad \frac{\partial^m H}{\partial Y^m}(x^*, y^*) \text{ is a unit in } \mathbb{B}.$$

We suppose in addition that \mathfrak{m} is an ideal in \mathbb{B} such that $\mathfrak{m}^2 = (0)$. In what follows, we suppose that we know (x, y) in \mathbb{B} , with both $x - x^*$ and $y - y^*$ in \mathfrak{m} , and we will show how to recover x^* and y^* ; further assumptions on (x^*, y^*) will be introduced when needed.

Solving for Y . Given (x, y) in \mathbb{B} , with both $x - x^*$ and $y - y^*$ in \mathfrak{m} , Newton iteration applied (with respect to Y) to the polynomial $\frac{\partial^{m-1}H}{\partial Y^{m-1}}$ shows that there exist a unique y_x in \mathbb{B} such that

$$\frac{\partial^{m-1}H}{\partial Y^{m-1}}(x, y_x) = 0, \quad y_x = y \text{ mod } \mathfrak{m}. \quad (3.5)$$

Explicitly, y_x is given by

$$y_x = y - \frac{\frac{\partial^{m-1}H}{\partial Y^{m-1}}(x, y)}{\frac{\partial^m H}{\partial Y^m}(x, y)}. \quad (3.6)$$

This is well-defined, since $\frac{\partial^m H}{\partial Y^m}(x^*, y^*)$ being a unit implies that $\frac{\partial^m H}{\partial Y^m}(x, y)$ is a unit as well. Remark also that since $\mathfrak{m}^2 = (0)$ in \mathbb{B} , doing just one step of Newton iteration is sufficient to find the root y_x .

The implicit function(s) J . Let ξ be a new variable, which we will use for power series over \mathbb{B} . Given x in \mathbb{B} , with $x - x^*$ in \mathfrak{m} , our next goal is to compute, if it exists, a power series J in $\mathbb{B}[[\xi]]$ such that

$$\frac{\partial^{m-1}H}{\partial Y^{m-1}}(x + \xi, J) = 0, \quad J(0) = y^* \text{ mod } \mathfrak{m}. \quad (3.7)$$

Lemma 21. *A power series J satisfies (3.7) if and only if it satisfies*

$$\frac{\partial^{m-1}H}{\partial Y^{m-1}}(x + \xi, J) = 0, \quad J(0) = y_x. \quad (3.8)$$

PROOF. Of course, if J satisfies (3.8), it satisfies the seemingly weaker condition (3.7). Conversely, suppose that J satisfies (3.7); we only have to prove that $J(0) = y_x$. Start from condition $\frac{\partial^{m-1}H}{\partial Y^{m-1}}(x + \xi, J) = 0$, and evaluate ξ at 0. This shows that the element $J(0) \in \mathbb{B}$ satisfies $\frac{\partial^{m-1}H}{\partial Y^{m-1}}(x, J(0)) = 0$ and $J(0) = y \pmod{\mathfrak{m}}$; the uniqueness of the solution of (3.5) proves that $J(0) = y_x$.

Applying again Newton iteration, these time modulo the powers of ξ , we deduce that there exists a unique power series J_x in $\mathbb{B}[[\xi]]$ that satisfies (3.7), or equivalently (3.8). Remark that such power series were already considered in Sections 3.5 and 3.6. Of particular interest will be power series $J^* = J_{x^*}$, which thus satisfies

$$\frac{\partial^{m-1}H}{\partial Y^{m-1}}(x^* + \xi, J^*) = 0, \quad J^*(0) = y^*.$$

Of course, since x^* and y^* are unknown to us, we cannot compute J^* . However, modulo \mathfrak{m} , we see that for all x such that $x - x^* \in \mathfrak{m}$, $J_x = J^* \pmod{\mathfrak{m}}$. In fact, something more precise can be said.

Lemma 22. *Let x be in \mathbb{B} , such that $x = x^* \pmod{\mathfrak{m}}$. Then the equality*

$$J_x = J^* + (x - x^*) \frac{dJ^*}{d\xi}$$

holds.

PROOF. We prove that the power series $C_x = J^* + (x - x^*) \frac{dJ^*}{d\xi}$ is equal to J_x . In view of the uniqueness property, it suffices to prove that C_x satisfies both conditions in (3.7). We start by evaluating the functional at $x + \xi$ and C_x ; this gives

$$\frac{\partial^{m-1}H}{\partial Y^{m-1}}(x + \xi, C_x) = \frac{\partial^{m-1}H}{\partial Y^{m-1}}(x^* + \xi + (x - x^*), J^* + (x - x^*) \frac{dJ^*}{d\xi}).$$

Because $x - x^*$ is in \mathfrak{m} , and $\mathfrak{m}^2 = (0)$, we can do a Taylor expansion at the first order, and deduce that the previous quantity is

$$\frac{\partial^{m-1}H}{\partial Y^{m-1}}(x^* + \xi, J^*) + (x - x^*) \frac{\partial^m H}{\partial X \partial Y^{m-1}}(x^* + \xi, J^*) + (x - x^*) \frac{dJ^*}{d\xi} \frac{\partial^m H}{\partial Y^m}(x^* + \xi, J^*).$$

The first term above vanishes by definition of J^* , so we are left with

$$(x - x^*) \left(\frac{\partial^m H}{\partial X \partial Y^{m-1}}(x^* + \xi, J^*) + \frac{dJ^*}{d\xi} \frac{\partial^m H}{\partial Y^m}(x^* + \xi, J^*) \right).$$

The right-hand factor is identically zero, since it is the derivative with respect to ξ of the defining equation for J^* . Thus, we are done with the first condition for C_x .

To prove the second one, note that because $x - x^*$ is in \mathfrak{m} , $C_x = J^* \bmod \mathfrak{m}$. As a consequence, $C_x(0) = J^*(0) \bmod \mathfrak{m}$, and thus $C_x(0) = y^* \bmod \mathfrak{m}$. This proves the second condition for C_x , and thus that $C_x = J_x$.

Using the second equation. Let us now consider a further polynomial K in $\mathbb{B}[X, Y]$, together with an integer $a \geq 0$. To x in \mathbb{B} , such that $x = x^* \bmod \mathfrak{m}$, we now associate

$$S_x = \frac{\partial^a K}{\partial Y^a}(x + \xi, J_x),$$

which is a well-defined power series in $\mathbb{B}[[\xi]]$. Inspired by the notation above, we write S^* for the particular case $x = x^*$. The following lemma shows that S_x is a first-order approximation of S^* .

Lemma 23. *Let x be in \mathbb{B} , such that $x = x^* \bmod \mathfrak{m}$. The equality*

$$S_x = S^* + (x - x^*) \frac{dS_x}{d\xi}$$

holds.

PROOF. Let us write $P = \frac{\partial^a K}{\partial Y^a}$. We have to prove that

$$P(x + \xi, J_x) = P(x^* + \xi, J^*) + (x - x^*) \frac{d}{d\xi} (P(x + \xi, J_x)).$$

The proof is similar to that in the previous lemma. Recall that $J_x = J^* + (x - x^*) \frac{dJ^*}{d\xi}$. Thus, the left-hand side is

$$P(x^* + \xi + (x - x^*), J^* + (x - x^*) \frac{dJ^*}{d\xi}),$$

which gives, after a first-order Taylor expansion,

$$\begin{aligned} P(x^* + \xi, J^*) + (x - x^*) \frac{\partial P}{\partial X}(x^* + \xi, J^*) + (x - x^*) \frac{dJ^*}{d\xi} \frac{\partial P}{\partial Y}(x^* + \xi, J^*) \\ = P(x^* + \xi, J^*) + (x - x^*) \frac{d}{d\xi} P(x^* + \xi, J^*). \end{aligned}$$

Finally, the first equality above proves in particular that $P(x + \xi, J_x) = P(x^* + \xi, J^*) \pmod{\mathfrak{m}}$, and this remains true after differentiation with respect to ξ . Thus, the lemma is proved.

As in Section 3.6, let us write $\text{cf}(S, \xi^j)$ the coefficient of ξ^j in a power series $S \in \mathbb{B}[[\xi]]$. We can then make our last assumptions on (x^*, y^*) : there exists an integer $n \geq 1$ such that

$$\mathsf{X}_3. \text{ cf} \left(\frac{d^{n-1} S^*}{d\xi^{n-1}}, \xi^0 \right) = 0$$

$$\mathsf{X}_4. \text{ cf} \left(\frac{d^n S^*}{d\xi^n}, \xi^0 \right) \text{ is a unit in } \mathbb{B}.$$

The following lemma finally allows us to compute x^* , assuming we know $x^* \pmod{\mathfrak{m}}$ and $y^* \pmod{\mathfrak{m}}$.

Lemma 24. *Let x be in \mathbb{B} , such that $x = x^* \pmod{\mathfrak{m}}$. If (x^*, y^*) satisfies $\mathsf{X}_1 - \mathsf{X}_4$, then $\text{cf}(\frac{d^n S_x}{d\xi^n}, \xi^0)$ is a unit in \mathbb{B} and*

$$x^* = x - \frac{\text{cf} \left(\frac{d^n S_x}{d\xi^n}, \xi^0 \right)}{\text{cf} \left(\frac{d^{n-1} S_x}{d\xi^{n-1}}, \xi^0 \right)}.$$

PROOF. To prove the first item, remark that the previous lemma shows in particular that $S_x = S^* \pmod{\mathfrak{m}}$. This remains true after differentiating n times. Extracting the constant coefficient, we deduce that $\text{cf}(\frac{d^n S_x}{d\xi^n}, \xi^0) = \text{cf}(\frac{d^n S^*}{d\xi^n}, \xi^0) \pmod{\mathfrak{m}}$. Since the former is a unit, the latter must be a unit too.

To conclude, start from the equality $S_x = S^* + (x - x^*) \frac{dS_x}{d\xi}$ proved above, and take the $(n - 1)$ -th derivative with respect to ξ . This gives $\frac{d^{n-1} S_x}{d\xi^{n-1}} = \frac{d^{n-1} S^*}{d\xi^{n-1}} + (x - x^*) \frac{d^n S_x}{d\xi^n}$. Extracting the constant coefficient (with respect to ξ) on both sides, and using X_3 gives

$$\text{cf} \left(\frac{d^{n-1} S_x}{d\xi^{n-1}}, \xi^0 \right) = (x - x^*) \text{cf} \left(\frac{d^n S_x}{d\xi^n}, \xi^0 \right).$$

Since we proved that $\text{cf}(\frac{d^n S_x}{d\xi^n}, \xi^0)$ is a unit in \mathbb{B} , the claim follows.

Once x^* is known, we can also recover y^* . One option is to apply the Newton iteration of Eq. (3.6), but we will prefer the following method, which will not require further evaluations. We know that $J_x = J^* + (x - x^*)\frac{dJ^*}{d\xi}$; as in the proof of the above lemma, since $J^* = J_x \bmod \mathfrak{m}$, we can rewrite $J_x = J^* + (x - x^*)\frac{dJ_x}{d\xi}$. Since $\text{cf}(J^*, \xi^0) = y^*$, we deduce

$$y^* = \text{cf}(J_x, \xi^0) - (x - x^*)\text{cf}(J_x, \xi^1). \quad (3.9)$$

3.8 Main algorithm

We can finally present the main algorithm and analyze its complexity. All along this section, we use the following notation. The input is a pair of polynomials F and G with coefficients in \mathbb{Z} , with degree at most d and height at most h . We suppose that these polynomials satisfy assumption \mathbf{H}_1 of Section 3.2, so that the associated polynomial $\Gamma \in \mathbb{Z}[T]$ is well-defined.

3.8.1 Choosing parameters

We are going to apply a random change of variables, and compute modulo a random prime. In this first section, we discuss these choices, and quantify their probability of success. Given $P \geq 1$, our goal is to obtain a probability of success of at least $1 - 1/2^P$.

As in the introduction, we suppose that we have an oracle \mathcal{O} , which on input B returns a random prime in the interval $\{B + 1, \dots, 2B\}$, chosen uniformly among those primes.

Let us first choose an integer t at random in the set $\{1, \dots, 2^{P+5}d^4\}$; remark in particular that the height of t is at most $P + 5 + 4\log(d)$. Because Γ is nonzero of degree at most $12d^4$, the probability that t cancels it is at most $1/2^{P+1}$. In what follows, let us assume that $\Gamma(t)$ is nonzero, so that, by Corollary 1, F_t and G_t are in general position. Besides, remark that we have the following height bounds:

- From Corollary 1 again, all polynomials appearing in the Shape Lemma representation of V , and of all its \mathbb{Q} -definable subsets, have height at most $H_1(P, d, h) = B_{\text{SL}}(d, h, P + 5 + 4\log(d))$, which is $(Pdh)^{O(1)}$.
- For t in $\{1, \dots, 2^{P+5}d^4\}$, the height of $\Gamma(t)$ can be bounded by an explicitly computable integer $H_2(P, d, h) = (Pdh)^{O(1)}$.
- The polynomials F_t and G_t have degree d ; their height is at most $H_3(P, d, h) = h + d(P + 5 + 4\log(d) + 3)$, which is clearly $(Pdh)^{O(1)}$.

Let us then define $\Delta_4(P, d, h) = \Delta_3(d, H_3, H_1)H_2(d, h)$, where Δ_3 was defined in Proposition 5. This quantity is computable from P, d, h (in time polylogarithmic in P, d, h), and we have $\Delta_4(P, d, h) = (Pdh)^{O(1)}$.

Consider now the integer $\delta_4 = \Gamma(t)\delta_3$, where δ_3 was defined in Proposition 5. Thus, the above construction shows that δ_4 is a nonzero integer of height at most $\Delta_4(P, d, h)$. Let us finally define $\Delta'_4(P, d, h) = 2^{P+2}\Delta_4(P, d, h)$, and the set

$$\Lambda(P, d, h) = \{\Delta'_4(P, d, h) + 1, \dots, 2\Delta'_4(P, d, h)\}.$$

Then, we have the following quantitative estimates:

- The set $\Lambda(P, d, h)$ contains at least $\Delta'_4(P, d, h)/(2 \log \Delta'_4(P, d, h))$ primes [66, Theorem 18.8].
- There are at most $\log_{\Delta'_4(P, d, h)}(\delta_4)$ primes in $\Lambda(P, d, h)$ that divide δ_4 .

Let us call the oracle \mathcal{O} , with input $\Delta'_4(P, d, h)$; as output, we get a random prime p in $\Lambda(P, d, h)$. The probability of p dividing δ_4 is thus at most

$$\frac{\log_{\Delta'_4(P, d, h)}(\delta_4)}{\frac{\Delta'_4(P, d, h)}{2 \log \Delta'_4(P, d, h)}} = 2 \frac{\log(\delta_4)}{\Delta'_4(P, d, h)} \leq \frac{1}{2^{P+1}}.$$

We will see below that provided $\Gamma(t)$ is nonzero and δ_4 does not reduce to zero modulo p , our lifting algorithm will compute $\text{RUR}(F, G)$. The previous discussion shows that it happens with probability at least $1 - 1/2^P$, as we wanted.

The last remark we make here is that the prime p we compute with is at most $2^{P+3}\Delta_4(P, d, h)$, which is $2^{O(P)}(dh)^{O(1)}$. Thus, $\log(p)$ is $O(P + \log(dh))$.

3.8.2 Computations modulo p

In all that follows, we suppose that t and p have been chosen such that $\Gamma(t)$ is nonzero, and such that the integer δ_4 defined above does not vanish modulo p .

In particular, by Corollary 1, both systems (F_t, G_t) and $(F_t \bmod p, G_t \bmod p)$ are in general position, over respectively $\mathbb{Q}[X, Y]$ and $\mathbb{F}_p[X, Y]$, and we have the specialization property $\text{SL}(F_t, G_t) \bmod p = \text{SL}(F_t \bmod p, G_t \bmod p)$.

Lemma 25. *Given F and G , one can compute $F_t \bmod p$ and $G_t \bmod p$, as well as $\text{SL}(F_t \bmod p, G_t \bmod p)$, using $\tilde{O}(d^2h + d^3 \log(p))$ bit operations.*

PROOF. We first reduce F and G modulo p ; this takes $O^\sim(d^2(h+\log(p)))$ bit operations, by fast Euclidean division. Then, we apply the change of variable $X \mapsto X + tY$ for $F \bmod p$ and $G \bmod p$, which gives $F_t \bmod p$ and $G_t \bmod p$ in $O^\sim(d^2 \log(p))$ bit operations.

We know that $(F_t \bmod p, G_t \bmod p)$ is in general position. To compute its Shape Lemma representation, we apply the algorithm of [39, Proposition 1], which runs in time $O^\sim(d^3 \log(p))$. There is only one minor difference: one step in that algorithm should be avoided (Step 6, which removes multiple solutions from $V(F_t \bmod p, G_t \bmod p)$ – we do not want to discard them here).

Let $\Sigma = [(C_{i,j}, T_{i,j}, m_i, H_i, n_{i,j}, a_{i,j}, K_{i,j})]_{1 \leq i \leq s, j \in D_i}$ be the σ -decomposition of F_t and G_t , as computed by Algorithm σ -dec. Since p does not divide $\Gamma(t)$, Corollary 1 proves that p cancels the denominator of none of the coefficients of the polynomials $C_{i,j}$ or $T_{i,j}$. In addition, the fact that p does not cancel the integer δ_3 of Proposition 5, applied to F_t and G_t , implies that $\Sigma \bmod p$ is the σ -decomposition of $F_t \bmod p$ and $G_t \bmod p$.

Lemma 26. *Given $F_t \bmod p$ and $G_t \bmod p$, one can compute the σ -decomposition of $\text{SL}(F_t \bmod p, G_t \bmod p)$ using $d^{3+\varepsilon} O^\sim(\log(p))$ bit operations.*

PROOF. This is Proposition 4, applied to $F_t \bmod p$ and $G_t \bmod p$.

In particular, all polynomials above can be seen in $\mathbb{Z}_p[X]$. For a fixed index i, j , we can then define the residue class ring $\mathbb{B}^\infty = \mathbb{Z}_p[X]/\langle C_{i,j} \rangle$, as well as the power series $J_{i,j}^\infty \in \mathbb{B}^\infty[[\xi]]$ characterized by the conditions

$$\frac{\partial^{m_i} H_i}{\partial Y^{m_i}}(x_{i,j}^\infty + \xi, J_{i,j}^\infty) \neq 0 \quad \text{and} \quad J_{i,j}^\infty(0) = y_{i,j}^\infty,$$

where $x_{i,j}^\infty$ is the residue class of X in \mathbb{B}^∞ , and $y_{i,j}^\infty$ is the residue class of $T_{i,j}$. The following lemma is then a direct consequence of the previous discussion.

Lemma 27. *For any index i, j , the point $(x_{i,j}^\infty, y_{i,j}^\infty)$ and $(m_i, H_i, n_{i,j}, a_{i,j}, K_{i,j})$ satisfy conditions $X_1 - X_4$ of Section 3.7 over the ring \mathbb{B}^∞ .*

3.8.3 Analysis of one lifting step

Suppose that we know $\Sigma \bmod N$, for N some power of p . In this section, we show how to compute $\Sigma \bmod N^2$. The main result of this section is the following proposition.

Proposition 6. *Given $\Sigma \bmod N$, for N a power of p , one can compute $\Sigma \bmod N^2$ using $d^{2+\varepsilon} O^\sim(\log(N))$ bit operations.*

In what follows, we denote by \mathbb{A} the ring $\mathbb{Z}/N^2\mathbb{Z}$, where all computations will be done. For any i in $\{1, \dots, s\}$ and j in D_i , we thus assume that we know $c_{i,j} = C_{i,j} \bmod N$ and $y_{i,j} = T_{i,j} \bmod N$. Our goal is to compute $c'_{i,j} = C_{i,j} \bmod N^2$ and $y'_{i,j} = T_{i,j} \bmod N^2$.

Let us write

$$\mathbb{B}_{i,j} = \mathbb{A}[X]/\langle c_{i,j} \rangle \quad \text{and} \quad \mathbb{B}'_{i,j} = \mathbb{A}[X]/\langle c'_{i,j} \rangle,$$

where in the former case, we view $c_{i,j}$ as a polynomial in $\mathbb{A}[X] = \mathbb{Z}/N^2\mathbb{Z}$. Thus, we can compute in $\mathbb{B}_{i,j}$, but not in $\mathbb{B}'_{i,j}$, since $c'_{i,j}$ is unknown. In what follows, we write $x_{i,j}$ for the residue class of X in $\mathbb{B}_{i,j}$.

Starting from the claim in Lemma 27 and reducing modulo N^2 , we deduce that over the ring $\mathbb{B}'_{i,j}$, the point $(x_{i,j}, t'_{i,j}) \in \mathbb{B}'_{i,j}{}^2$ satisfies assumptions $X_1 - X_4$ of Section 3.7. Now, Proposition 1 in [40] proves that $\mathbb{B}_{i,j}$ and $\mathbb{B}'_{i,j}$ are isomorphic, through an isomorphism that reduces to the identity modulo N , and that leaves \mathbb{Z} invariant. Thus, there exists $(x^*_{i,j}, y^*_{i,j}) \in \mathbb{B}_{i,j}{}^2$ such that:

- $x^*_{i,j} = x_{i,j} \bmod N$ and $y^*_{i,j} = y_{i,j} \bmod N$
- $(x^*_{i,j}, y^*_{i,j})$ and $(m_i, H_i, n_{i,j}, a_{i,j}, K_{i,j})$ satisfy conditions $X_1 - X_4$ of Section 3.7 over the ring $\mathbb{B}_{i,j}$.

We will thus apply the algorithm of Section 3.7, in order to first compute $x^*_{i,j}$ and $y^*_{i,j}$ in $\mathbb{B}_{i,j}$. In a second stage, we will deduce $c'_{i,j}$ and $t'_{i,j}$.

The computation of $x^*_{i,j}$ and $y^*_{i,j}$ proceeds itself in several steps, which follow the description in Section 3.7. For the cost analysis, it will be useful to remark that

$$\sum_{1 \leq i \leq s, j \in D_i} n_{i,j} m_{i,j} \deg(C_{i,j}) \leq d^2 \tag{3.10}$$

holds. Indeed, the deflation lemma shows that for any root x of $C_{i,j}$, $n_{i,j} m_{i,j}$ is a lower bound on the multiplicity of (F, G) at $(x, T_{i,j}(x))$. The above inequality then follows from Bézout's theorem.

Algorithm 8: lift_y($F, G, \Sigma \bmod N$)

Input: polynomials F, G , the σ -decomposition $\Sigma \bmod N$

Output: $[v_{i,j}]_{1 \leq i \leq s, j \in D_i}$

- 1 $I = [(i, j) \mid 1 \leq i \leq s, j \in D_i]$
 - 2 $I_F = [(i, j) \mid 1 \leq i \leq s, j \in D_i \text{ and } H_i = \text{"F"}]$
 - 3 $I_G = [(i, j) \mid 1 \leq i \leq s, j \in D_i \text{ and } H_i = \text{"G"}]$
 - 4 $[\eta_{i,j}]_{(i,j) \in I_F} = [\frac{\partial^{m_i-1} F}{\partial Y^{m_i-1}}(x_{i,j}, y_{i,j})]_{i \in I_F}$ calculations are done in $\mathbb{B}_{i,j}$
 - 5 $[\eta'_{i,j}]_{(i,j) \in I_F} = [\frac{\partial^{m_i} F}{\partial Y^{m_i}}(x_{i,j}, y_{i,j})]_{i \in I_F}$
 - 6 $[\gamma_{i,j}]_{(i,j) \in I_G} = [\frac{\partial^{m_i-1} G}{\partial Y^{m_i-1}}(x_{i,j}, y_{i,j})]_{i \in I_G}$
 - 7 $[\gamma'_{i,j}]_{(i,j) \in I_G} = [\frac{\partial^{m_i} G}{\partial Y^{m_i}}(x_{i,j}, y_{i,j})]_{i \in I_G}$
 - 8 **return** $[v_{i,j}]_{(i,j) \in I} = [y_{i,j} - \eta_{i,j}/\eta'_{i,j}]_{i \in I_F} \text{ cat } [y_{i,j} - \gamma_{i,j}/\gamma'_{i,j}]_{i \in I_G}$
-

Computing all $v_{i,j}$'s. First, applying Eq. (3.6), we compute the elements $v_{i,j}$ such that for all i, j , we have

$$\frac{\partial^{m_i-1} H_i}{\partial Y^{m_i-1}}(x_{i,j}, v_{i,j}) = 0$$

in $\mathbb{B}_{i,j}$, and such that $v_{i,j} = y_{i,j} \bmod N$.

Correctness follows from Eq. (3.6). Regarding running time, the bulk of the cost is the computation of sequences $[\eta_{i,j}]$, $[\eta'_{i,j}]$, $[\gamma_{i,j}]$, $[\gamma'_{i,j}]$; the divisions at the last step can all be done in quasi-linear time.

Let us for instance explain how to compute $[\eta_{i,j}]$ and $[\eta'_{i,j}]$. This is a direct application of Algorithm `normal_forms`, with input the lists L, L', L'' and F , with $L = [(0, m_i)]_{(i,j) \in I}$, $L' = [c_{i,j}]_{(i,j) \in I}$ and $L'' = [y_{i,j}]_{(i,j) \in I}$. Inequality (3.10) implies that

$$\sum_{1 \leq i \leq s, j \in D_i} m_{i,j} \deg(C_{i,j}) \leq d^2,$$

so we are under the conditions of Proposition 3. This implies that all $[\eta_{i,j}]$ and $[\eta'_{i,j}]$ can be computed in $d^{2+\varepsilon} O^{\sim}(\log(N))$ bit operations. This concludes the cost analysis of this step.

Computing all $x_{i,j}^*$ and $y_{i,j}^*$'s. Next, we consider Algorithm `lift_x_y`, which computes all $x_{i,j}^*$ and $y_{i,j}^*$ in $\mathbb{B}_{i,j}$.

Algorithm 9: lift_{x,y}(N, F, G, Σ_N)

Input: polynomials F, G , the σ -decomposition $\Sigma \bmod N$

Output: $[(x_{i,j}^*, y_{i,j}^*)]_{1 \leq i \leq s, j \in D_i}$

```
1  $I = [(i, j) \mid 1 \leq i \leq s, j \in D_i]$ 
2  $I_F = [(i, j) \mid 1 \leq i \leq s, j \in D_i \text{ and } H_i = \text{"F"}]$ 
3  $I_G = [(i, j) \mid 1 \leq i \leq s, j \in D_i \text{ and } H_i = \text{"G"}]$ 
4  $[v_{i,j}]_{(i,j) \in I} = \text{lift}_y(F, G, \Sigma \bmod N)$ 
5  $[J_{i,j}]_{(i,j) \in I} = \text{compute\_J}(F, G, [(c_{i,j}, t_{i,j}, m_i, H_i, n_{i,j} + 1)]_{(i,j) \in I})$ 
6  $\eta = [\eta_{i,j,\alpha}]_{(i,j) \in I, \alpha \in [0, \dots, m_i]} = [\frac{\partial^\alpha F}{\partial Y^\alpha}(X + \xi, J_{i,j}) \bmod \langle c_{i,j}(X), \xi^{n_{i,j}+1} \rangle]_{(i,j) \in I, \alpha \in [0, \dots, m_i]}$ 
7  $\gamma = [\gamma_{i,j,\alpha}]_{(i,j) \in I, \alpha \in [0, \dots, m_i]} = [\frac{\partial^\alpha G}{\partial Y^\alpha}(X + \xi, J_{i,j}) \bmod \langle c_{i,j}(X), \xi^{n_{i,j}+1} \rangle]_{(i,j) \in I, \alpha \in [0, \dots, m_i]}$ 
8  $L = []_{(i,j) \in I}$ 
9 for  $(i, j)$  in  $I$  do
10    $S_{i,j} = \text{select}(H_i, K_{i,j}, a_{i,j}, \eta, \gamma)$ 
11    $x_{i,j}^* = x_{i,j} - \frac{\text{cf}\left(\frac{d^{n_{i,j}} S_{i,j}}{d\xi^{n_{i,j}}}, \xi^0\right)}{\text{cf}\left(\frac{d^{n_{i,j}-1} S_{i,j}}{d\xi^{n_{i,j}-1}}, \xi^0\right)}$ 
12    $y_{i,j}^* = \text{cf}(J_{i,j}, \xi^0) - (x_{i,j} - x_{i,j}^*)\text{cf}(J_{i,j}, \xi^1)$ 
13   append  $(x_{i,j}^*, y_{i,j}^*)$  to  $L$ 
14 end
15 return  $L$ 
```

This stage of the algorithm directly uses the formula derived in Lemma 24, and the subsequent Equation (3.9). For any index i, j , we need the corresponding power series $S_{i,j}$ modulo $\xi^{n_{i,j}+1}$, which in turn means that we need $J_{i,j}$ at the same precision.

The power series $J_{i,j}$ are computed at Step 5 using Algorithm `compute_J`, using the output $v_{i,j}$ of the previous step as a starting value. Algorithm `compute_J` was written so as to work modulo a prime, not a prime power; however, it still applies to work modulo a prime power N without any modification. Inequality (3.10) shows that we are under the assumptions of Lemma 19, so we deduce that Step 5 can be executed using $d^{2+\varepsilon}O^{\sim}(\log(N))$ bit operations.

Steps 6 and 7 are done using Algorithm `normal_forms`. Once more, Inequality (3.10) shows that we are under the assumptions of Proposition 3, so these steps take $d^{2+\varepsilon}O^{\sim}(\log(N))$ bit operations.

Subroutine `select` then extracts the power series $J_{i,j}$ from the vectors η and γ , using $H_i, K_{i,j}$, and $a_{i,j}$ to find the proper entry. The updates necessary to compute $x_{i,j}^*$ and $y_{i,j}^*$ take quasi-linear time. To summarize, the cost of this step is $d^{2+\varepsilon}O^{\sim}(\log(N))$ bit

operations.

Computing all $c'_{i,j}$'s and $t'_{i,j}$'s. At this stage, for any index (i, j) , we have found the root $(x_{i,j}^*, y_{i,j}^*)$ with coordinates in $\mathbb{B}_{i,j} = \mathbb{A}[X]/\langle c_{i,j} \rangle$. Our goal is to recover $c'_{i,j} = C_{i,j} \bmod N^2$, together with $t'_{i,j} = T_{i,j} \bmod N^2$.

Explicit formula exist for this conversion (see for instance [33, Section 6]); in our case, writing $\delta_{i,j} = x_{i,j}^* - x_{i,j}$, they read

$$c'_{i,j} = c_{i,j} - \left(\delta_{i,j} \frac{dc_{i,j}}{dX} \bmod c_{i,j} \right) \quad \text{and} \quad t'_{i,j} = t_{i,j} - \delta_{i,j} \frac{dt_{i,j}}{dX} \bmod c_{i,j}.$$

In particular, the new polynomials $c'_{i,j}$ and $t'_{i,j}$ can all be computed for a total of $O^\sim(d^2 \log(N))$ bit operations. Summing all costs seen so far, we conclude the proof of Proposition 6.

3.8.4 Total cost

We can finally finish the cost analysis of our algorithm. Since t has been chosen with height $O(P + \log(d))$, Corollary 1 shows that the output $\text{RUR}(F_t, G_t)$ has height $O^\sim(dh + d^2P)$.

We run lifting steps until the precision N goes beyond twice this bound. Using Chinese remainder techniques, we can then deduce from $\Sigma \bmod N$ the polynomials $(P \bmod N, S \bmod N) = \text{SL}(F_t, G_t) \bmod N$, in quasi-linear time $O^\sim(d^2 \log(N))$. From this, we compute $R = P'S \bmod P$ modulo N , and we apply rational reconstruction to all coefficients. This takes again quasi-linear time.

The dominant cost is that of lifting, which is $d^{2+\varepsilon} O^\sim(dh + d^2P)$, or simply $d^{3+\varepsilon} O^\sim((h + d)P)$.

Chapter 4

Univariate Polynomial Factorization

4.1 Introduction

In this chapter we will consider the problem of univariate polynomial factorization over number fields. As we already said in 1, there are two main approaches to factor a univariate polynomial over a number field. The first approach which is due to Trager [62], says that factoring over $F(\alpha)$ is doable if we can factor over F , where F is a field and α is in the algebraic closure of F . Since we know how to factor over \mathbb{Q} [69, 42, 63], so we can do it over any number fields, using [62].

The other approach is to apply a similar methods as we do over \mathbb{Q} , which is done by Lenstra in 1982 [41]. Then Belabas in [9] combined a modification of Lenstra method and van Hoeij's factors combination approach [63] to introduce a polynomial-time algorithm for factoring univariate polynomials over number fields. These methods are based on two main steps, *modular factorization* and *factors combination*. The first step is just applying the Hensel lifting on the factors of the given polynomial, say $h \in \mathbb{Q}(\alpha)[x]$, modulo some (lucky) prime ideal, say P , up to some precision. The other step deals with the problem of finding the true factors of h over $\mathbb{Q}(\alpha)$ from the P -adic factors.

Our main concern in this chapter is to solve the factorization problem, following the second approach, in a more efficient way, in practice. To make the chapter more readable, in Section 4.2, we first explain how to factor a polynomial over \mathbb{Q} following van Hoeij's approach [63]. Then in Section 4.3 we consider the problem of univariate polynomial factorization over number fields, using multivariate representation, while following a similar approach as Belabas approach in [9]. Finally in 4.3.5, we will give some experimental results, comparing our approach with the existed ones using splitting fields computations,

as one of the most important applications of polynomial factorization.

4.2 Factoring polynomials over \mathbb{Z} and \mathbb{Q}

This chapter deals with the question of factoring polynomials over the infinite domains \mathbb{Z} and \mathbb{Q} . As we will see, factoring over such finite field turns out to be an essential step to derive factoring algorithms over infinite domains. We first consider $\mathbb{Z}[x]$ as the polynomial ring of our choice. Later on, we will show that these algorithms can also be used to factor polynomials over the base domain \mathbb{Q} with only slight modifications.

We start with some basic consideration how to factor polynomials over the integers. Obviously, a prime factor of a monic polynomial over \mathbb{Z} is also a factor over any finite field \mathbb{F}_q . Unfortunately, the converse direction fails, i.e., a factor of f in \mathbb{F}_q is not necessarily a factor in \mathbb{Z} . As a small example illustrating this fact, let us take $q = 7$ and $f = x^2 + 6 \in \mathbb{Z}[x]$. Obviously, f is irreducible in $\mathbb{Z}[x]$, but in $\mathbb{F}_7[x]$ we have:

$$f = (x + 6)(x + 1)$$

More generally, if we assume that a factorization of f in $\mathbb{Z}[x]$ is given by $f = f_1 \cdots f_k$, then in $\mathbb{F}_q[x]$ the irreducible factorization of f is $f = g_1 \cdots g_r$, where $r \geq k$.

Now, the question arises whether a factorization over finite fields, i.e., the so-called modular factors, might be useful to factorize polynomials over \mathbb{Z} , and how this can be achieved most efficiently. The factors of f over the integers are usually called the *true* factors.

4.2.1 Some basic considerations

Let $f \in \mathbb{Z}[x]$ be a square-free polynomial. At first, we have to consider the question for which primes p the polynomial $\bar{f} = f \bmod p$ in $\mathbb{F}_p[x]$ is square-free, which will be a necessary condition for the factor recombination phase of the following algorithms. Here, and also in the following, the bar denotes the modular image of the given element. In order to determine the right choices of p , the following lemma [66, Lemma 15.1] is useful.

Lemma 28. *Let $f \in \mathbb{Z}[x]$ be a non-zero square-free polynomial, $p \in \mathbb{N}$ a prime not dividing $\text{lc}(f)$, the leading coefficient of f . Then \bar{f} is square-free if and only if p does not*

divide the discriminant of f .

Now, assume that we have found a prime p which fulfils the requirements of Lemma 4.1. Assume further that we have already computed a factorization of f in $\mathbb{F}_p[x]$, say $\bar{f} = \overline{\text{lc}(f)}g_1 \cdots g_r$ for some $r \in \mathbb{N}$, where $g_i \in \mathbb{F}_p[x]$ is a monic polynomial for $1 \leq i \leq r$, so we want to find those factors of that can be combined to prime factors over \mathbb{Z} . More formally, for some integers $r > k$, we have

$$f = f_1 \cdots f_k = \text{lc}(f)g_1 \cdots g_r \pmod{p}$$

Suppose, that we have already computed all the g_i , and let f_j be an irreducible factor of f . If we consider the factorization modulo p , some of the g_i 's divide f_j (but we do not know which ones). Hence, we want to compute a partition $S = \{S_j | j \in \{1, \dots, k\}\}$ of $\{1, \dots, r\}$, where the set S_j contains precisely those modular factors that can be combined to the factor f_j . If we could compute such a partition, for all $j \in \{1, \dots, k\}$, we would obtain

$$\frac{\text{lc}(f)}{\text{lc}(f_j)} f_j = \text{lc}(f) \prod_{g_i \in S_j} g_i \pmod{p}$$

Furthermore, we have to ensure that all prime factors f_j can be uniquely restored from the g_i 's, which can be achieved by a suitable choice of the considered modulus p . Using Mignottes bound

$$\frac{p}{2} > \sqrt{n+1} 2^n |\text{lc}(f)| \|f\|_\infty$$

we ensure that all coefficients of $\frac{\text{lc}(f)}{\text{lc}(f_j)} f_j$ are integers with absolute value less than $p/2$ [66, Corolary 6.33]. Thus, we obtain the desired equality if we use $\{-\frac{p-1}{2}, \dots, \frac{p-1}{2}\}$ as representative, and we can construct the true factors f_j from the modular factors $g_i \in S_j$ in this case.

By now, we can already outline the idea of how an algorithm for factoring polynomials over \mathbb{Z} may work, formulated in the following two steps:

- Compute a factorization of f in $\mathbb{F}_p[x]$ for a suitable choice of p , that is, \bar{f} should be square-free and unique restoration should be possible as discussed above.
- Compute the prime factors of f over \mathbb{Z} and reconstruct the true factors

Obviously, the first step can easily be achieved by choosing a sufficiently large prime p , which is usually called big-prime approach. However, this approach is very expensive in practice, since very large values of p are often needed to obtain the proper

factorization over the integers.

Another more sophisticated approach, called Hensel-lifting ([66] Theorem 15.18), tries to solve the same problem, first by factoring f modulo a suitable small prime p , satisfying Lemma 28 and then lifting the result up to a sufficiently large power of p , for which unique restoration should be possible.

A simple possibility to deal with the second step is to build all possible combinations of factors until we finally find the true factors over \mathbb{Z} . This approach has exponential runtime in the worst case, but fortunately, this worst case does only occur with a very small probability so this approach is nevertheless commonly used in practice. We only briefly mention here, that there is also an approach using only a polynomial number of steps which works by identifying the factors of f with the vectors of a lattice and computing short vectors in this lattice. Computing the shortest vectors in a lattice is NP-hard [4], but there is an algorithm computing an approximation that is sufficient for our purpose, [42] for more details. We will sketch the idea of the algorithm in the next section since it serves as a foundation for the sophisticated algorithm developed by Mark van Hoeij in 2002 [63].

4.2.2 The approach of van Hoeij

In this section we review the algorithm of Mark van Hoeij [63] for factoring polynomials over the integers.

Short vectors in lattices

As we already stated there also exists an algorithm [42] for factoring polynomials over the integers which ensures correct termination after a polynomial number of steps. This algorithm is based on short vectors in lattices, and the LLL algorithm by Lenstra, Lenstra and Lovacz. Since the ideas of this algorithm and of the actual algorithm of van Hoeij [63] are quite similar, we briefly review the key idea of how short vectors in lattices can be used to factor polynomials over the integers.

At first, we take a look at the following important lemma. Here and in the following, the norm $\| f \|$ of a polynomial f is defined as 2-norm of its coefficients regarded as a vector $v = (a_0, \dots, a_n)$, where $f = \sum_{i=0}^n a_i x^i$ and $\| f \| := \| v \| = \sqrt{a_0^2 + \dots + a_n^2}$.

Lemma 29. *Let $f, g \in \mathbb{Z}[x]$ with $\deg(f) = n$ and $\deg(g) = d$, and assume that $u \in \mathbb{Z}[x]$*

is non-constant, monic, and divides both f, g modulo m for some $m \in \mathbb{N}$ with $\|f\|^d \|g\|^n < m$. Then $\gcd(f, g) \in \mathbb{Z}[x]$ is non-constant.

A proof of this lemma can be found in [66, Lemma 16.20].

The idea of factoring polynomials using the above lemma is now as follows. Suppose that we are given a square-free polynomial $f \in \mathbb{Z}[x]$ of degree n and we have already computed a monic polynomial $u \in \mathbb{Z}[x]$ with degree $d < n$ that divides f modulo m for some $m \in \mathbb{N}$. Then we need to find a short polynomial g , such that $\|g\|^n < m \|f\|^{-d}$, that is also divisible by u modulo m . Then the above lemma gives us a nontrivial factor of f in $\mathbb{Z}[x]$.

Before going any further, we need the following definition.

Definition 2. Let $n \in \mathbb{N}$ and $f_1, \dots, f_n \in \mathbb{R}^n$ with $f_i = (f_{i1}, \dots, f_{in})$. Then

$$L := \sum_{i=1}^n \mathbb{Z}f_i = \left\{ \sum_{i=1}^n r_i f_i : r_1, \dots, r_n \in \mathbb{Z} \right\}$$

is the **Lattice** generated by f_1, \dots, f_n . The vectors f_1, \dots, f_n are called a basis for the lattice L .

In order to find such a polynomial g of degree less than some bound j , we consider the lattice $L \subset \mathbb{Z}^j$ generated by the coefficient vectors of the polynomials

$$\{ux^i \mid 0 \leq i < d - j\} \cup \{mx^i \mid 0 \leq i < d\}$$

In the following, we simply identify a polynomial with the vector of its coefficients. An element g of L has the form

$$g = qu + rm, \quad q, r \in \mathbb{Z}[x], \quad \deg(q) < d - j, \quad \deg(r) < d$$

and degree less than j . Thus, u divides g modulo m . On the other hand, if some $g \in \mathbb{Z}[x]$ is of degree less than j and divisible by u modulo m , then we have

$$g = q^*u + r^*m, \quad q^*, r^* \in \mathbb{Z}[x]$$

Division with remainder by the monic polynomial u yields $q^{**}, r^{**} \in \mathbb{Z}[x]$ with $q^* = q^{**}u + r^{**}$ and $\deg(r^{**}) < \deg(u)$. Let $q = q^* + mq^{**}$ and $r = r^*$, we see that the polynomial g has the claimed form and we conclude that

$$g \in L \iff \deg(g) < j \text{ and } u \text{ divides } g \text{ modulo } m$$

Thus, we only have to compute a short vector in the lattice L . Unfortunately, the problem of computing the shortest vector of a given lattice is NP-hard[4], so there is no hope for efficient algorithms in general. However, we can use the LLL algorithm [42] to compute the set S of short vectors for most instances of the problem. We omit a detailed description of the LLL algorithm here for the sake of readability. We only state that the LLL algorithm can find the set S of short vectors, provided that all vectors outside of $\text{Span}(S)$ are sufficiently long in comparison. We now present a small example which illustrates how factorization with short vectors actually works.

Example 3. Let $f = x^3 - 1 \in \mathbb{Z}[x]$ and $m = 7^6 = 117649$. Using factorization over finite fields and applying Hensel lifting, we obtain

$$f = (x - 1)(x - 2)(x + 3) \pmod{7}$$

and

$$f = (x - 1)(x - 34967)(x + 34968) \pmod{7^6}$$

We now set $u = x - 34967$ and $j = 3$, i.e. we consider the lattice $L \subset \mathbb{Z}^3$ generated by the coefficients of the vectors ux, u, m , namely,

$$(1, -34967, 0), (0, 1, -34967), (0, 0, 117649)$$

We now apply the LLL algorithm to the lattice L which yields the following three vectors:

$$(1, 1, 1), (132, 95, -228), (228, -132, -95)$$

If we take the first one, we obtain the polynomial $g = x^2 + x + 1$, since $\gcd(f, g) = g$, so g is a proper factor of f in $\mathbb{Z}[x]$.

Solving knapsack problem with LLL

We now present another possibility where the LLL algorithm can be efficiently used. We will show how to break instances of the subset sum problem and the knapsack problem.

The subset sum problem seeks the answer to the following:

given $a_1, \dots, a_n, s \in \mathbb{N}$, are there $x_1, \dots, x_n \in \{0, 1\}$ such that $\sum_{i=1}^n a_i x_i = s$?

It is a well-known fact that this problem is NP-complete. A slight generalization of it is called the knapsack problem which aims computing the x_i 's, if there is any solution. It is well-known that this problem is NP-complete, too [46].

The connection between the subset sum problem and short vectors in lattices is now given by the fact that a solution of the above problem yields a short vector in the lattice $L \subset \mathbb{Z}^{n+1}$ generated by the rows $r_1, \dots, r_{n+1} \in \mathbb{Z}^{n+1}$, of the matrix

$$\begin{pmatrix} 1 & 0 & \cdots & 0 & -a_1 \\ 0 & 1 & \cdots & 0 & -a_2 \\ \vdots & \vdots & \ddots & \vdots & \\ 0 & 0 & \cdots & 1 & -a_n \\ 0 & 0 & \cdots & 0 & s \end{pmatrix} \in \mathbb{Z}^{(n+1) \times (n+1)}$$

In order to see this, let the vector $(x_1, \dots, x_n) \in \{0, 1\}^n$ be a solution of the knapsack problem. Now consider the vector

$$v = \sum_{i=1}^n r_i x_i + r_{n+1} \in L$$

we obtain

$$v = (x_1, \dots, x_n, -\sum_{i=1}^n a_i x_i + s) = (x_1, \dots, x_n, 0)$$

Moreover, we have $\|v\|_2 \leq \sqrt{n}$ which is very small, at least if the a_i 's are reasonably large numbers. Thus v is a short vector in this case, which we can find using the LLL algorithm.

Example 4. Consider the lattice $L \subset \mathbb{Z}^7$ generated by the rows of the matrix

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & -366 \\ 0 & 1 & 0 & 0 & 0 & 0 & -385 \\ 0 & 0 & 1 & 0 & 0 & 0 & -392 \\ 0 & 0 & 0 & 1 & 0 & 0 & -401 \\ 0 & 0 & 0 & 0 & 1 & 0 & -422 \\ 0 & 0 & 0 & 0 & 0 & 1 & -437 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1215 \end{pmatrix} \in \mathbb{Z}^{7 \times 7}$$

Now, the LLL algorithm computes a short vector $v = (0, 0, 1, 1, 1, 0, 0) \in L$. We only consider the first six entries of the vector as the desired coefficients x_i 's, and indeed we obtain

$$366 \times 0 + 385 \times 0 + 392 \times 1 + 401 \times 1 + 422 \times 1 + 422 \times 0 + 437 \times 0 = 1215$$

The algorithm of van Hoeij will work similarly, i.e., it will encode the desired solution as a solution of a knapsack problem, which it will solve using the LLL algorithm.

General introduction

Let $f = \sum_{i=0}^n a_i x^i \in \mathbb{Z}[x]$ be a monic and square-free polynomial and define a natural number D as an upper bound for the number of digits of the coefficients of f .

We assume that we already factored the polynomial f in $\mathbb{Z}_p[x]$, where \mathbb{Z}_p denotes the ring of p -adic integers, yielding a factorization $\prod_{i=1}^r f_i$ where $f_i \in \mathbb{Z}_p[x]$ are monic irreducible factors of f in $\mathbb{Z}_p[x]$. Following the algorithm of Zassenhaus [69], we now have to decide which of these factors f_i can be combined to a factor of f in $\mathbb{Z}[x]$. Obviously, there are 2^r possibilities which can be encoded in a vector $v = (v_1, \dots, v_r) \in \{0, 1\}^r$, where $v_i = 1$ means that the factor f_i is included in the product to build the factor f in $\mathbb{Z}[x]$. More precisely, we set $g_v = \prod_{i=1}^r f_i^{v_i}$. Then

$$\{g_v : v \in \{0, 1\}^r\}$$

is the set of all monic factors of f in $\mathbb{Z}_p[x]$. Since we are interested in the factors of f in $\mathbb{Z}[x]$, we set

$$V := \{v : g_v \in \mathbb{Z}[x]\}$$

and so $\{g_v : v \in V\}$ is the set of all monic factors of f in $\mathbb{Z}[x]$. Finally, we restrict the

set V to all irreducible factors of f which yields

$$B = \{v \in V : g_v \text{ irreducible in } \mathbb{Z}[x]\}$$

Thus, the set B contains precisely the combinations of factors we are looking for. Since $\mathbb{Z}[x]$ is unique factorization domain, the set V consist of all $\{0,1\}$ -linear combinations of B .

To find the true factors of f , we still need to compute the set B , which corresponds to computing the 'good' combinations of modular factors. In the following, let $P_Z(n, D)$ be the complexity of computing the p -adic factorization of f , which is polynomial in n and D .

The factor recombination phase of Zassenhaus' algorithm tries out every factor combination, starting with single factors, then moving on to product of two, three factors and so on. For every combination, it tests whether the computed product g_v divides f in $\mathbb{Z}[x]$. We can assume the time of each test to be constant c .

We know that the complexity of Zassenhaus' algorithm, in the worst case, is exponential in the number of modular factors. If we set $|v| = \sum v_i$ then the complexity depends on $M := \max\{|v| : v \in B\}$. The worst case is $M = r$, i.e., f is irreducible and all 2^r combinations will be tried. Thus, the total cost is at most:

$$P_Z(n, D) + c2^r$$

or we can bound the cost by

$$P_Z(n, D) + cE_Z(M)$$

where $E_Z(M) := 2^M \leq 2^r$ depends exponentially on M .

In practice, even for large n , the number M is often low, so $P_Z(n, D)$ is the dominant part of the total cost and the Zassenhaus algorithm works fast. But, for instance, when f has a large number of factors in $\mathbb{Z}_p[x]$ for every p , but only a few factors in $\mathbb{Z}[x]$, then M is a large number and $cE_Z(M)$ dominates the running time and the algorithm of Zassenhaus is exponentially slow in this case.

On the other hand, fortunately, there is an algorithm that ensures correct termination after a polynomial number of steps for every considered polynomial. It is based on a

lattice reduction algorithm, the LLL algorithm introduced in [42] by Lenstra, Lenstra and Lovasz, which can be used to solve many combinatorial problems by encoding the solutions of the problem as short vectors of a lattice. Let P_L denotes the complexity of the lattice part. In the algorithm of [42] most cases $P_L \gg P_Z$, in other words, the algorithm of Zassenhaus is must faster, except when $cE_Z \gg P_Z$ holds, which only happens for a few polynomials in practice.

Putting it all together, there exists a polynomial-time algorithm [42], and an exponential time algorithm [69] which is nevertheless faster most of the time. Now the question naturally arises whether we can combine the advantages of both approaches, i.e., to guarantee polynomial-time termination, but nevertheless maintain the fast runtime in the overall case, this has been done by van Hoeij [63].

Suppose that $g_v \in \mathbb{Z}[x]$. The algorithm in [42] would find a $g = g_v$ by computing a vector U_g in an n -dimensional lattice whose entries are the coefficients of g . However, if g is large, i.e., it has high degree and large coefficients, the LLL algorithm will be very slow. Obviously, if we want a faster computation, we have to consider smaller vectors instead, which is the key idea behind the van Hoeij algorithm [63].

For this purpose, we need to design a new lattice in such a way that B can be obtained from the short vectors found by LLL. In order to keep the cost of LLL at a minimum, we should additionally make sure that the short vectors found by LLL do not contain any information (other than the set) about the coefficients of the factors f , the input polynomial, so the cost of LLL will be independent of g . So the LLL cost is independent of n, D and depends only on r , polynomially. Then, the total cost of the newly designed algorithm would be

$$P_Z(n, D) + P(r)$$

where $P(r)$ is the cost of LLL for the new lattice.

Thus, the resulting algorithm is faster than the one of Zassenhaus whenever $P(r) < cE_Z(M, r)$. Experiments show that the cut-off point is low. This means that when $P(r) > cE_Z(M, r)$ then both of them are small. However, when r is larger, then $P(r)$ can be much smaller than $cE_Z(M, r)$. It turns out in experiments that polynomials with $r > 400$ and $n, D > 2000$ can be handled, which is far beyond the reach of [42, 69].

The construction of the lattice

In order to construct the desired lattice, following [63], we are going to find *linear* conditions on the vectors $v \in \mathbb{B}$. However, the coefficients of the polynomial g_v can not be used because they do not depend linearly on v . In order to circumvent this problem, we define a vector $T_A(g) \in \mathbb{Z}_p^s$ with s entries, satisfying the following property for all polynomials $g_1, g_2 \in \mathbb{Z}_p[x]$:

$$T_A(g_1 g_2) = T_A(g_1) + T_A(g_2)$$

This vector $T_A(g)$ will be constructed in such a way that the entries of $T_A(g_v)$ are p -adic integers for all vectors $v \in \{0, 1\}^r$, and if furthermore $T_A(g_v) \in \mathbb{Z}[x]$ holds, then the entries are integers, bounded in absolute value by $\frac{1}{2}p^b$ for some integer b . Now, $T_A(g_v)$ is a linear combination of $T_A(f_i)$, $T_A(g_v) = \sum_{i=1}^r v_i T_A(f_i)$, and the entries are integers in the case of $T_A(g_v) \in \mathbb{Z}[x]$.

Using $T_A(f_i)$ directly as the input vectors of the lattice has two drawbacks:

- the entries of $T_A(f_i)$ are p -adic integers, i.e., they cannot be finitely expressed. This problem can be solved by cutting $T_A(f_i)$ from the right side.
- if $g_v \in \mathbb{Z}[x]$, then the entries of $T_A(g_v)$ are integers bounded by $\frac{1}{2}p^b$, giving some partial information about the coefficients of g_v . Including such unnecessary information in the lattice only leads to inefficiency. This problem can be solved by cutting $T_A(f_i)$ from the left side, removing the information as big as the size of g_v .

As we said, both problems can be solved by cutting each entry of $T_A(f_i)$ on two sides. Let $t = T_A(f_i)$ for some $1 \leq i \leq r$, so t is a vector of p -adic integers and can be written as $t = \sum_{k=0}^{\infty} t_k p^k$, where t_i 's are integers in the interval $(-\frac{p^b}{2}, \frac{p^b}{2}]$. Choose $a > b$ and replace t with $\sum_{k=b}^{a-1} t_k p^{k-b}$, i.e., the powers $\geq a$ and the powers $< b$ in t are removed and the whole sum is divided by p^b . Denote this cutting by $T_A^{b,a}(f_i)$.

Now let us introduce a candidate for the already mentioned vector T_A .

Definition 3. *the i 'th trace $Tr_i(g)$ of a polynomial g is defined as the sum of the i 'th powers of the roots (counted with multiplicity) of g .*

Obviously for all $f_1, f_2 \in \mathbb{Z}_p[x]$, we have

$$Tr_i(f_1 f_2) = Tr_i(f_1) + Tr_i(f_2)$$

However, what we want is to actually compute the i 'th trace of a given polynomial. This can be achieved using the i 'th elementary symmetric polynomials as follows. Consider an arbitrary polynomial as

$$g = (x - x_1) \cdots (x - x_d)$$

Then g can be written as

$$g = x^d + \tilde{E}_1 x^{d-1} + \cdots + \tilde{E}_d$$

where $\tilde{E}_i = (-1)^i E_i$ and $E_i = E_i(x_1, \dots, x_d)$ is the i 'th symmetric polynomial in the variables x_1, \dots, x_d . The i 'th power polynomial $P_i(x_1, \dots, x_d)$ is defined as $x_1^i + \cdots + x_d^i$, i.e., $P_i = Tr_i(g)$. Computationally, the polynomials E_i and P_i are related recursively, as follows.

$$P_i = -i\tilde{E}_i - \sum_{k=1}^{i-1} P_k \tilde{E}_{i-k}$$

$$i\tilde{E}_i = -P_i - \sum_{k=1}^{i-1} P_k \tilde{E}_{i-k}$$

A direct conclusion of the above equations, called Newton identities, can be stated as the following lemma.

Lemma 30. *Let \mathbb{F} be a field of characteristic 0, such as the field of p -adic numbers. Then a monic polynomial $g \in \mathbb{F}[x]$ of degree d has rational numbers as coefficients if and only if $Tr_i(g) \in \mathbb{Q}$ for all $1 \leq i \leq d$.*

Let

$$Tr_{1,\dots,d}(g) = \begin{pmatrix} Tr_1(g) \\ Tr_2(g) \\ \vdots \\ Tr_d(g) \end{pmatrix}$$

The following lemma is an extension of Lemma 30 stating that the polynomial g is in fact a polynomial over \mathbb{Z} if and only if $Tr_{1,\dots,d}(g)$ has integer entries, by imposing an extra assumption on f . The proof can be found in [63, Lemma 2.2].

Lemma 31. *Let f be a monic polynomial of degree n in $\mathbb{Z}[x]$, and \mathbb{F} be a field of characteristic zero. Then for any monic factor $g \in \mathbb{F}[x]$ of f with degree d , the following statements are equivalent:*

- $g \in \mathbb{Z}[x]$

- $Tr_{1,\dots,d}(g) \in \mathbb{Z}^d$

As we already stated, every monic factor $g \in \mathbb{Z}_p[x]$ can be encoded by a 0 – 1 vector $v = (v_1, \dots, v_r) \in \{0, 1\}^r$ as

$$g = \prod_{i=1}^r f_i^{v_i}$$

Let s and d be positive integers and A be a $s \times d$ matrix with integer entries. Then define

$$T_A := ATr_{1,\dots,d}$$

The key idea behind introducing the matrix A is only for practical efficiency. Indeed instead of using a vector $Tr_{1,\dots,d}$ with d entries, we use the vector $ATr_{1,\dots,d}$ with s entries where s is much smaller than d in practice. In order to increase understanding, the reader may as well think of the trivial choice $s = d$ and let A denote the $d \times d$ identity matrix in the following.

Lemma 32. *If $g \in \mathbb{Z}_p[x]$ is a monic factor of f then*

$$g \in \mathbb{Z}[x] \implies T_A(g) \in \mathbb{Z}^s$$

PROOF. A direct application of Newton identities, and the fact that A is an integer matrix.

Let S be a subset of p -adic factors and g be the product of S . Then

$$T_A(g) = \sum_{f_i \in S} T_A(f_i)$$

or equivalently, if we encode this set S as 0 – 1 vector $v = (v_1, \dots, v_r)$, then

$$T_A(g) = \sum v_i T_A(f_i)$$

So a necessary condition for g to be a true factor is that the sum of $T_A(f_i)$'s, $f_i \in S$, has integer entries. However, each $T_A(f_i)$ is a p -adic integer, so they can only be determined up to some finite accuracy a . Let B_{rt} be a bound on the absolute value of the complex roots of f . Then dB_{rt}^i is a bound for $|Tr_i(g)|$ for any rational factor g of f of degree less than d . This allows us to compute bounds for the entries of $T_A(g)$. Now, we can compute the symmetric remainder of $\sum_{f_i \in S} T_A(f_i)$ modulo p^a .

However, this remainder still includes the powers $< b$, i.e., at least partial information about the coefficients of g . For example, if we consider $T_A = Tr_{1,\dots,s}$ then the first s coefficients of g can be computed from $T_A(g)$ using the relation between the i 'th elementary symmetric polynomials and the i 'th power polynomials. However, we do not have any good reason for that, since we can compute as well by multiplying all polynomial $f_i \in S$. Therefore, we can cut off this unnecessary information since we are not interested in the precise value of $T_A(g)$, but we only want to know whether it satisfies the bound or not. Moreover, this cutting of information will speed up the upcoming algorithm since it significantly simplifies the inputs for the LLL algorithm. For this purpose, we define the vector T_A^b as below.

Assume the absolute value of the i 'th entry of $T_A(g)$ is bounded by B_i for each monic factor g of f . Also choose a list of positive integers $b = (b_1, \dots, b_s)$ such that $B_i < \frac{1}{2}p^{b_i}$.

Definition 4. Let $g \in \mathbb{Z}_p[x]$ be monic, r_i be the i 'th entry of $T_A(g)$, and \bar{r}_i be the symmetric remainder of r_i modulo p^{b_i} . Then $r_i - \bar{r}_i$ is divisible by p^{b_i} , so $u_i := \frac{r_i - \bar{r}_i}{p^{b_i}}$ is a p -adic integer. Then we define

$$T_A^b(g) = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_s \end{pmatrix}$$

If g is a true factor of f , then $r_i \leq B_i < \frac{1}{2}p^{b_i}$, hence $r_i - \bar{r}_i = 0$. Thus, all entries of $T_A(g)$ will be zero. This proves the following lemma.

Lemma 33. Let $g \in \mathbb{Z}_p[x]$ be a monic factor of f . then

$$g \in \mathbb{Z}[x] \implies T_A^b(g) = 0$$

We note that if f_j was approximated with accuracy a then the i 'th entry of $T_A^b(f_j)$ can be computed modulo p^{a-b_i} . So a should be greater than b_i . In particular, a needs to be larger than $\log(2B_i)/\log(p)$, since $B_i < \frac{1}{2}p^{b_i}$.

As we can see from the definition of $T_A^b(g)$, everything that is smaller than the bound B_i has simply been rounded off to 0, which means smaller entries compared to $T_A(g)$. But unfortunately, additivity is lost due to this round off, i.e., the equality $T_A^b(f_1 f_2) = T_A^b(f_1) + T_A^b(f_2)$ does not necessarily hold anymore. However, the following lemma states that $T_A^b(g)$ is additive up to a small error. The proof can be find in [63, Lemma 2.6].

Lemma 34. Let S be a subset of $\{f_1, \dots, f_r\}$ and let g be the product of the elements of S . Then

$$T_A^b(g) = \varepsilon + \sum_{f_i \in S} T_A^b(f_i)$$

where $\varepsilon = (\varepsilon_1, \dots, \varepsilon_s) \in \mathbb{Z}^s$. Furthermore,

$$|\varepsilon_i| \leq \frac{|S|}{2}$$

Combining the Lemma 33 and 34 yields to the following Lemma.

Lemma 35. Let S be a subset of $\{f_1, \dots, f_r\}$ and let g be the product of the elements of S . If $g \in \mathbb{Z}[x]$ then

$$\varepsilon + \sum_{f_i \in S} T_A^b(f_i) = 0$$

for some vector $\varepsilon \in \mathbb{Z}^s$ given as in Lemma 34.

Note that the entries of $T_A^b(g)$ are still p -adic integers which cannot be expressed using finite arithmetic. Therefore, we have to additionally cut them at some power p^a .

Definition 5. Let $c \in \mathbb{Z}_p$ and $0 \leq b \leq a$ be integers. The symmetric remainder $C^a(c)$, called the approximation of c with accuracy a , is the unique integer $-\frac{p^a}{2} < C^a(c) \leq \frac{p^a}{2}$ that is congruent to c modulo p^a . Now define $C_b^a(c)$, called a tow-sided cut of c , as $C^{a-b}((c - C^b(c))/p^b)$

Now choose integers a_i such that $b_i < a_i$, Let \bar{c}_{ji} be the i 'th entry of $T_A(f_j)$ and let c_{ji}^* be the i 'th entry of $T_A^b(f_j)$. Define

$$c_{ji} := C_{b_i}^{a_i}(\bar{c}_{ji}) = C^{a_i - b_i}(c_{ji}^*)$$

and let

$$C_j = (c_{j1}, \dots, c_{js})$$

Putting it all together, the vectors C_j are approximations of the vectors $T_A^b(f_j)$ which are now suitable inputs for the LLL algorithm. We can now reformulate Lemma 35 as follows. Let e_1, \dots, e_s be the standard basis for \mathbb{Z}^s , that is for $1 \leq i \leq s$, e_i is a vector of size s with 1 at the i th position, and zero at the others.

Theorem 5. (The factorization knapsack problem)

Let f be a monic square-free polynomial in $\mathbb{Z}[x]$ and f_1, \dots, f_r the irreducible p -adic

factor of f . For every $S \subset \{f_1, \dots, f_r\}$ if the product g of the elements of S is a rational factor of f then

$$\sum_{i=1}^s (\varepsilon_i + \gamma_i p^{a_i - b_i}) e_i + \sum_{j=1}^r v_j C_j = 0$$

for some integers ε_i and γ_i with absolute value at most $\frac{|S|}{2}$, and where

$$v_j = \begin{cases} 1 & \text{if } f_j \in S \\ 0 & \text{otherwise} \end{cases}$$

This theorem is the main theoretical result of the section, putting the previous lemmas all together in one statement as a knapsack problem. Indeed, based on Theorem 5 the factor recombination phase of Zassenhaus factorization algorithm can be converted to a knapsack problem that can be solved, as we will see in the following, using LLL technique.

The knapsack lattice

In this section we present the van Hoeij factorization algorithm [63] for univariate polynomials over \mathbb{Q} or \mathbb{Z} and explain its correctness and termination.

Let W be the set of all vectors $v = (v_1, \dots, v_r) \in \mathbb{Z}^r$ for which $\prod_{i=1}^r f_i^{v_i}$ is in $\mathbb{Z}[x]$. Note that if g_1, \dots, g_t are the monic irreducible factors of f in $\mathbb{Z}[x]$, then $\{w_1, \dots, w_t\}$ is a basis of W in reduced echelon form, where w_i is defined as the 0–1 vector (v_1, \dots, v_r) for which $g_k = \prod_{i=1}^r f_i^{v_i}$. Finding this reduced basis is the same as solving the combinatorial problem in the algorithm of Zassenhaus [69], that is choosing the set of modular factors correspond to the true irreducible factors of f . If $L \subseteq \mathbb{Z}^r$ is a lattice, then B_L denotes a basis for L . The matrix whose rows are the elements of B_L is denoted by (B_L) and $\text{rref}(B_L)$ denotes its row echelon form. If we can compute any basis B_W of W , then the combinatorial problem is solved because $\{w_1, \dots, w_t\}$ are the rows of $\text{rref}(B_W)$.

Lemma 36. *Let L be a lattice such that $W \subseteq L \subseteq \mathbb{Z}^r$ and $R = \text{rref}(B_L)$. Then $L = W$ if and only if the following two conditions hold:*

- each column of R contains precisely one 1, all other entries are 0
- if (v_1, \dots, v_r) is a row of R then $g = \prod_{i=1}^r f_i^{v_i}$ is in $\mathbb{Z}[x]$

PROOF. refer to the proof of [63, Lemma 2.8].

The above lemma gives us a sufficient condition for L to be the desired lattice W . So the van Hoeij algorithm starts from $L = \mathbb{Z}^r$ and at each step tests if L is equal to W . This test can be done using Lemma 36. Now suppose that $L \neq W$. Then the algorithm tries to calculate a new lattice L' with

$$W \subseteq L' \subseteq L$$

Then L is replaced by L' . The algorithm keeps repeating this until we finally obtain $L = W$, i.e., both conditions of Lemma 36 are satisfied.

Like any other algorithm, we need to address the following two questions:

- Does the algorithm terminate ?
- If the algorithm terminates, is the output correct ?

For the purpose, assume that we have chosen an $s \times d$ matrix A and integers a_i, b_i such that $a \geq a_i > b_i > \frac{\log(2B_i)}{\log(p)}$. We will now show how to compute the new lattice $L' \subseteq L$, hopefully of smaller dimension than L , that nevertheless contains all solutions of the knapsack problem introduced in Theorem 5. Note that the new lattice L' would depend on the size of the matrix A and the precisions a, a_i, b_i , so if we can not find a new lattice L' satisfying the required conditions, then we should change this parameters.

Let

$$M := \sqrt{C^2 r + s(r/2)^2}$$

where C is a positive integer chosen in such a way that the values of both terms under the square root do not deviate very much from each other, i.e., they should be roughly equal. Let B_L be a basis for L . Initially $L = \mathbb{Z}^r$ and B_L is the standard basis of \mathbb{Z}^r . In order to solve the knapsack-like problem given in Theorem 5 we will construct a knapsack lattice Δ such that the vector

$$v_S = (Cv_1, \dots, Cv_r, -\varepsilon_1, \dots, -\varepsilon_s)$$

is an element of Δ for every solution S of the knapsack problem. Note that if $(v_1, \dots, v_r, \varepsilon_1, \dots, \varepsilon_s, \gamma_1, \dots, \gamma_s) \in \mathbb{Z}^{r+2s}$ is a solution of the given knapsack problem, then the vector $(Cv_1, \dots, Cv_r, -\varepsilon_1, \dots, -\varepsilon_s) \in \mathbb{Z}^{r+s}$ is an element of the lattice Δ . Since in the following we do not need to know the value of γ_i 's. We do not put them in the lattice Δ to obtain a smaller lattice. Indeed, v_i 's are the actual values

that we are looking for and ε_i 's are the given information about the values of v_i 's, we will use information to separate the desired solution of the given knapsack problem from the others. Information will be encoded in terms of shortness of the desired solutions, which can be detected by applying LLL algorithm on the constructed lattice Δ .

A vector v in Δ will be called M -short if $\|v\| \leq M$. Note that v_S is M -short for every solution S of the knapsack problem, since $|S| \leq r$ and by applying Lemma 34, we have

$$\|v\|^2 \leq C^2|S| + s(|S|/2)^2 \leq M^2$$

Let (e_1, \dots, e_s) be the standard basis of \mathbb{Z}^s and 0^s denotes the zero element of \mathbb{Z}^s . All of these vectors are now in row notation. The notation $(v, w) \in \mathbb{Z}^{r+s}$ refers to the vector obtained by concatenating the vectors $v \in \mathbb{Z}^r$ and $w \in \mathbb{Z}^s$. The knapsack lattice $\Delta \in \mathbb{Z}^{r+s}$ is defined by the following basis: $B_\Delta = B_C \cup B_{p^*}$, where

$$B_{p^*} = \{(0^r, p^{a_i-b_i} e_i) : 1 \leq i \leq s\}$$

$$B_C = \{(Cv, vm) : v \in B_L\}, \quad m = \begin{pmatrix} C_1 \\ C_2 \\ \vdots \\ C_s \end{pmatrix}$$

In the first step, we have $L = \mathbb{Z}^r$. Then B_L has r elements and B_Δ has $r+s$ elements, and the matrix of the basis of Δ is

$$(B_\Delta) = \begin{pmatrix} C & 0 & \cdots & 0 & c_{11} & \cdots & c_{1s} \\ 0 & C & \cdots & 0 & c_{21} & \cdots & c_{2s} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & C & c_{r1} & \cdots & c_{rs} \\ 0 & 0 & \cdots & 0 & p^{a_1-b_1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & p^{a_s-b_s} \end{pmatrix}$$

The matrix (B_Δ) is square, so the determinant of the lattice is then given by

$$D = C^r p^{(a_1-b_1)+\cdots+(a_s-b_s)}$$

Let $l := |B_\Delta| = s + |B_L|$, where $|\cdot|$ denotes the cardinality of the set. Note that the matrix (B_Δ) is a matrix of size $l \times (r+s)$, which is not necessarily square

at each step of the algorithm. We can now use LLL to compute a reduced basis V_1, \dots, V_l of Δ . Let V_1^*, \dots, V_l^* be its Gram-Schmidt basis [66, Section 16.2]. We can compute approximation \bar{V}_k^* of V_k^* using floating point arithmetic. Let $t \leq l$ be the smallest integer such that $\bar{V}_k^* > M'$ for all $t < k \leq l$, where M' is M plus a bound of round-off errors in \bar{V}_k^* coming from the numerical computation. Thus, we obtain $\bar{V}_k^* > M$ for all $k > t$, i.e., those vectors are not M -short. Now define $\Delta' \subseteq \Delta$ as the span of $\{V_k : k \leq t\}$ and let $L' \subseteq L$ be the projection of $\frac{1}{C}\Delta'$ on the first r coordinates.

Roughly speaking, we distinguish between vectors of the basis, which are M -short, and those which are not. The vectors which are not M -short can obviously simply be thrown out of the computation, since they cannot yield the desired vectors. This is captured more formally in the following lemma.

Lemma 37. *$W \subseteq L'$, so if the algorithm terminates, then the output is correct.*

PROOF. Theorem (1.11) in [42] states that if $|V_k^*| > M$ for all $k > t$, then all M -short vectors are in the span of $\{V_k : k \leq t\}$. Now if S is a solution of the knapsack problem, then the vector v_S is M -short, and thus in Δ' . If S corresponds to an irreducible factor g_k and 0–1 vector w_k , then w_k is $1/C$ times the projection of v_S on the first r coordinates, and $v_S \in \Delta'$, hence $w_k \in L'$, and the lemma follows.

Before giving any proof for the termination of the process, let us first state the algorithm. The factorization algorithm over \mathbb{Z} or \mathbb{Q} introduced by Mark van Hoeij in 2002 can be stated using the following steps. Let $F \in \mathbb{Z}[x]$ be the input polynomial.

1. Apply the Berlekamp-Zassenhaus [69, 12] algorithm to F , but search only for the rational factors that consist of at most three p -adic factors (just for practical reason), Whenever a rational factor is found, remove the corresponding p -adic factors from the list.
2. Let f_1, \dots, f_r be the remaining p -adic factors, and let f be the polynomial F divided by the rational factors that were found in Step 1. If r is small then use the standard Berlekamp-Zassenhaus algorithm to do the rest of the work as well.
3. At this point, r is not small, the p -adic factors f_1, \dots, f_r have been computed modulo p^a , where the prime p has been chosen to minimize r , and a has been chosen using the Mignotte bound. Now the knapsack algorithm can begin.
4. Let $B_L = \{e_1, \dots, e_r\}$, the standard basis for \mathbb{Z}^r .

5. Choose a matrix A , compute an upper bound B_i for the i 'th entry of $T_A(g)$ for any rational factor g of f . Choose the integers $a_i > b_i > \log(2B_i)/\log(p)$. If $a_i > a$ then do additional Hensel lifting to increase a .
6. Compute the basis $B_{p^*} \cup B_C$ for the lattice Δ as described above. If the last s entries of the elements B_C are already small, then go back to step 5 and choose a different matrix A .
7. Apply the LLL algorithm to compute a reduced basis V_1, \dots, V_r of the lattice Δ . Do a floating point Gram-Schmidt computation to determine an as small as possible integer t such that all M -short vectors of Δ are in $\Delta' := \mathbb{Z}V_1 + \dots + \mathbb{Z}V_t$. Let L' be $1/C$ times the projection of Δ' on the first r coordinates. Then replace B_L by a basis of L' . If $|B_L|$ did not decrease then return to step and use larger values for $a_i - b_i$.
8. Let $t = |B_L|$. Note that $\dim(W) \leq r/4$, because all rational factors consisting of less than four p -adic factor have already been removed in step 1. If $t = 1$ then f must be irreducible and the computation ends. If $t > r/4$ then go to step 5, otherwise proceed to step 9.
9. Compute $R = \text{rref}(B_L)$. If R does not satisfy the first condition of Lemma 36 then go to step 5, otherwise proceed to step 10.
10. Check the second condition of Lemma 36 by executing step 11 for $k \in \{1, \dots, t\}$.
11. Let (v_1, \dots, v_r) be the k 'th row of R . Let $g_k = C^a(\prod(f_i^{v_i}))$. If g_k does not divide f in $\mathbb{Z}[x]$, then go back to step 5.
12. Now $f = g_1 \cdots g_t$, the irreducibility of the monic polynomials g_1, \dots, g_t has been proved, so the computation is done.

We refer the reader to [63, Section 2.3] for discussion on how to choose different parameters of the algorithm. Finally, we have the following lemma addressing the termination of the van Hoeij algorithm. For the proof see [63, Lemma 2.10].

Lemma 38. *The algorithm terminates.*

The following remarks are addressing the non-monic situation and how to choose an upper bound on the trace of the factors of a given polynomial over \mathbb{Z} .

Remark 1. If the polynomial $f = \sum_{i=0}^n a_i x^i \in \mathbb{Z}[x]$ is not monic, so $a_n \neq 1$, we need to make two changes to the algorithm. If $g \in \mathbb{Q}[x]$ is a monic rational factor of f then the coefficients of g are no longer automatically integers. As a consequence, $Tr_i(g)$ is no longer in \mathbb{Z} , which is something that the algorithm uses. However $a_n^i Tr_i(g) \in \mathbb{Z}$. Indeed, if $\deg(g) = l$, and $\gamma_1, \dots, \gamma_l$ are the roots of g in some field extension of \mathbb{Q} , then we have

$$\tilde{g}(x) := a_n^l g(x/a_n) = (x - a_n \gamma_1) \cdots (x - a_n \gamma_l)$$

which is a monic polynomial over \mathbb{Q} with

$$Tr_i(\tilde{g}) = a_n^i Tr_i(g)$$

. To prove $a_n^i Tr_i(g) \in \mathbb{Z}$, it is enough to show that $\tilde{g} \in \mathbb{Z}[x]$. For the purpose, assume that $f(x) = g(x)h(x)$ over \mathbb{Q} , then

$$\tilde{f}(x) := a_n^{n-1} f(x/a_n) = a_n^l g(x/a_n) a_n^{n-l-1} h(x/a_n) = \tilde{g}(x) a_n^{n-l-1} h(x/a_n)$$

Since \tilde{f}, \tilde{g} are monic polynomials over \mathbb{Q} , $\tilde{h}(x) := a_n^{n-l-1} h(x/a_n)$ must be a monic polynomial over \mathbb{Q} . Hence we have a factorization of a monic integer polynomial \tilde{f} into two monic rational polynomials over \mathbb{Q} as $\tilde{f} = \tilde{g}\tilde{h}$, so the Gauss lemma concludes that $\tilde{g}, \tilde{h} \in \mathbb{Z}[x]$, as desired. So in the non-monic case Tr_i needs to be replaced by $a_n^i Tr_i$ in the algorithm.

The second change is in step 11 of the algorithm which is identical to the difference between the monic and non-monic case in the Berlekamp-Zassenhaus algorithm. To find g_k , take $C^a(a_n \prod f_i^{v_i})$ instead of $C^a(\prod f_i^{v_i})$ and divide it by the gcd of its coefficients.

Remark 2. In step 5 of the algorithm, we need to compute an upper bound B_i for i th entry of $T_A(g)$ for any rational factor g of f . For the purpose it is enough to find an upper bound on $Tr_j(g)$ for $1 \leq j \leq n$, since we have $|Tr_j(g)| \leq |Tr_j(f)|$ for any rational factor g of f , and $T_A(g)$ is a linear combination of $Tr_j(g)$'s. Let B_{roots} be an upper bound on the modulus of the roots of f . For instance, B_{roots} can be $\sum |\frac{a_i}{a_{i+1}}|$ where $f = \sum_{i=0}^n a_i x^i$ [34]. Then by definition, we have

$$|Tr_j(f)| = \left| \sum_{f(\gamma)=0} \gamma^j \right| \leq n B_{root}^j$$

4.3 Factoring polynomials over number fields

In this section we consider the problem of factoring a univariate polynomial over number fields, finite algebraic extensions of \mathbb{Q} . Let

$$\{H_1(t_1), \dots, H_q(t_1, \dots, t_q)\} \subset \mathbb{Q}[t_1, \dots, t_q]$$

be a monic triangular set with $\deg(H_i, t_i) = h_i$, and for each $1 \leq i \leq q$, H_i is irreducible over

$$\mathbb{Q}[t_1, \dots, t_{i-1}] / \langle H_1, \dots, H_{i-1} \rangle$$

So then $\mathbb{K} = \mathbb{Q}[t_1, \dots, t_q] / \langle H_1, \dots, H_q \rangle$ is a number field with $N := [\mathbb{K} : \mathbb{Q}] = \prod_{i=1}^q h_i$. We let $\mathcal{O}_{\mathbb{K}}$ be its ring of algebraic integers [67, Section 2.2].

Now let $f \in \mathbb{K}[x]$ be a monic univariate polynomial of degree n over the number field \mathbb{K} . This section is concerned with the question of factoring f over \mathbb{K} . We are going to follow the same approach as in the previous section by extending the van Hoeij algorithm [63] to the number field case. Indeed, if $q = 1$ then our approach would be exactly the same as what Belabas does in [9]. What we will do in this section is to extend his method to the case of $q > 1$.

Before going any further, we first do some modifications on the given input polynomials, which are needed for the following discussions. Note that these modifications are not any restriction on the input of the factorization algorithm which will be introduced in this section.

First, we assume that $\{H_i\}_{i=1}^q \subset \mathbb{Z}[t_1, \dots, t_q]$, if not, use the change of variable $\tilde{t}_i = t_i/\lambda_i$, where λ_i is the common denominator of the coefficients of H_i .

Secondly, we assume that f is a square-free polynomial over \mathbb{K} , otherwise, replace f by $f/\gcd(f, f')$, and then we can easily find the complete factorization of f using $\gcd(f, f')$ and the factors of $f/\gcd(f, f')$.

So from now on, $f \in \mathbb{K}[x]$ is a monic square-free polynomial over the number field \mathbb{K} whose generators over \mathbb{Q} are all multivariate polynomials over \mathbb{Z} . As in the van Hoeij factorization algorithm, our algorithm follows two steps.

The first step is lifting the factors modulo some power of some suitable prime. Let p be a prime number such that

- $\langle H_1 \bmod p, \dots, H_q \bmod p \rangle$ is a radical ideal in $\mathbb{Z}/p\mathbb{Z}[t_1, \dots, t_q]$
- $\hat{\mathbb{K}} = \mathbb{Z}/p\mathbb{Z}[t_1, \dots, t_q]/\langle \hat{H}_1, \dots, \hat{H}_q \rangle$ is a field, where \hat{H}_1 is an irreducible factor of the image of H_1 modulo p , and for $2 \leq i \leq q$, \hat{H}_i is an irreducible factor of the image of H_i modulo p , $\hat{H}_1, \dots, \hat{H}_{i-1}$, $\deg(\hat{H}_i, t_i) = \hat{h}_i$, and $\{\hat{H}_1, \dots, \hat{H}_q\}$ is a monic triangular set.
- f remains square-free over $\hat{\mathbb{K}}$
- none of the denominators of f and its factors over \mathbb{K} vanishes modulo p

For the existence of such a prime p , we refer the reader to [59, 23]. Since $\hat{\mathbb{K}}$ is a finite field, then we can factor f over $\hat{\mathbb{K}}$ using, e.g. [12]. Then following [59], we can lift the factors to some required precision. The following theorem summarizes the lifting step.

Theorem 6. *Using the above notations and for a given $a \in \mathbb{N}$, one can compute the polynomials $f_{a,1}, \dots, f_{a,r}$ such that*

$$f = \prod_{i=1}^r f_{a,i} \bmod p^{2^a}, \hat{H}_1^a, \dots, \hat{H}_q^a$$

where $\hat{H}_i^a \bmod p = \hat{H}_i$, for $1 \leq i \leq q$, and $f_{a,i}$'s are monic irreducible polynomials in $\hat{\mathbb{K}}_p^a = \mathbb{Z}/p^{2^a}\mathbb{Z}[t_1, \dots, t_q]/\langle \hat{H}_1^a, \dots, \hat{H}_q^a \rangle$, called modular factors.

Now the next step is to find the true factors of f over \mathbb{K} . Following the same approach as before, we need to address two questions:

- Factor construction: which asks for constructing the irreducible factor of f over \mathbb{K} for a given corresponding modular factor
- Factor combination: which asks for which modular factors correspond to a true irreducible factor of f over \mathbb{K}

In the following we explain how to solve the above questions, using lattice structure discussed in Section 4.2.

4.3.1 Factor construction

Before going any further, we first need to figure out how to construct the true factor of f over \mathbb{K} from its p -adic approximation. In the case of factorization over \mathbb{Z} , the construction phase is nothing but just considering the p -adic factor as a polynomial over \mathbb{Z} . This is because there is an embedding from \mathbb{Z} into the ring of p -adic integers \mathbb{Z}_p for any prime number p . Since the image of each element of \mathbb{Z} under the embedding has only a finite number of non-zero coefficient as a p -series, so we can distinguish the element of \mathbb{Z} from the other elements of \mathbb{Z}_p . So we only need to know an upper bound on the size of the coefficients of the true factor of the given polynomial over \mathbb{Z} . Then any p -adic factor with coefficient size less than the bound can be a candidate for the true factor over \mathbb{Z} , as in the van Hoeij algorithm [63].

But here, when we are working over number fields, the situation is more complicated. Assume that $\hat{g} = \sum_{j=1}^d \hat{g}_j x^j \in \hat{\mathbb{K}}_p^a[x]$ is a modular factor of f over $\hat{\mathbb{K}}$ and let $g = \sum_{j=1}^d g_j x^j \in \mathbb{K}[x]$ be the corresponding true factor of f over \mathbb{K} . The goal of the section is to reconstruct the polynomial g from its modular image \hat{g} , which will be done coefficient-wisely. Since, for each $1 \leq j \leq d$, \hat{g}_j is the modular image of g_j in $\hat{\mathbb{K}}_p^a$, we have

$$\hat{g}_j = g_j \pmod{p^{2^a}, \hat{H}_1^a, \dots, \hat{H}_q^a}$$

Whence there exist $\beta_{j,i}^a \in \mathbb{Q}[t_1, \dots, t_q]$, for $0 \leq i \leq q$, such that over $\mathbb{Q}[t_1, \dots, t_q]$ we have

$$g_j - \hat{g}_j = \beta_{j,0}^a p^{2^a} + \beta_{j,1}^a \hat{H}_1^a + \dots + \beta_{j,q}^a \hat{H}_q^a$$

Note that $\deg(g_j - \hat{g}_j, t_i) < h_i$ for each i , regarding $g_j - \hat{g}_j$ as a polynomial in $\mathbb{Q}[t_1, \dots, t_q]$. The following lemma gives us an upper bound on the degrees of $\beta_{j,i}^a$'s in each variable for some suitably chosen $\beta_{j,i}^a$'s satisfying the above equality. In the following we consider \hat{H}_i as a polynomial over \mathbb{Q} .

Lemma 39. *Let $\gamma \in \mathbb{Q}[t_1, \dots, t_q]$ with $\deg(\gamma, t_i) < h_i$. Assume $\gamma = \gamma_0 p^{2^a} + \gamma_1 \hat{H}_1^a + \dots + \gamma_q \hat{H}_q^a$, where $\gamma_i \in \mathbb{Q}[t_1, \dots, t_q]$. Let $T_i = \{\hat{H}_1, \dots, \hat{H}_i\}$, for $1 \leq i \leq q-1$. Then there exist polynomials $\beta_i \in \mathbb{Q}[t_1, \dots, t_q]$ such that*

- $\gamma = \beta_0 p^{2^a} + \beta_1 \hat{H}_1^a + (\beta_2 \hat{H}_2^a \pmod{T_1}) + \dots + (\beta_q \hat{H}_q^a \pmod{T_{q-1}})$
- $\deg(\beta_0, t_k) < \hat{h}_k$, for $1 \leq k \leq q$
- for $i \geq 1$

- $\deg(\beta_i, t_k) < \hat{h}_k$, for $1 \leq k < i$
- $\deg(\beta_i, t_k) < h_k - \hat{h}_k$, for $k = i$
- $\deg(\beta_i, t_k) < h_k$, for $i < k \leq q$

PROOF. We will prove the lemma by induction over q . Let $q = 1$, then $\gamma = \gamma_0 p^{2^a} + \gamma_1 \hat{H}_1^a$ and $\deg(\gamma, t_1) < h_1$. Division with remainder by the monic polynomial \hat{H}_1^a yields $\gamma_{0,0}, \gamma_{0,1} \in \mathbb{Q}[t_1]$ with $\gamma_0 = \gamma_{0,0} + \gamma_{0,1} \hat{H}_1^a$ and $\deg(\gamma_{0,0}, t_1) < \hat{h}_1$. So then

$$\begin{aligned} \gamma &= \gamma_0 p^{2^a} + \gamma_1 \hat{H}_1^a = (\gamma_{0,0} + \gamma_{0,1} \hat{H}_1^a) p^{2^a} + \gamma_1 \hat{H}_1^a \\ &= \gamma_{0,0} p^{2^a} + (\gamma_{0,1} p^{2^a} + \gamma_1) \hat{H}_1^a \end{aligned}$$

Taking $\beta_0 = \gamma_{0,0}$ and $\beta_1 = \gamma_{0,1} p^{2^a} + \gamma_1$ prove the induction base. Note that $\deg(\beta_1, t_1) < h_1 - \hat{h}_1$, since $\deg(\beta_1 \hat{H}_1^a, t_1) \leq \deg(\gamma, t_1) < h_1$ and $\deg(\beta_1 \hat{H}_1^a, t_1) = \deg(\beta_1, t_1) + \hat{h}_1$.

Now assume the correctness of lemma for $q - 1$. Let

$$\gamma = \gamma_0 p^{2^a} + \gamma_1 \hat{H}_1^a + \cdots + \gamma_q \hat{H}_q^a$$

By reducing γ_q modulo the polynomials $\hat{H}_1^a, \dots, \hat{H}_{q-1}^a$, we have

$$\gamma_q = \gamma_{q,0} + \gamma_{q,1} \hat{H}_1^a + \cdots + \gamma_{q,q-1} \hat{H}_{q-1}^a$$

where $\gamma_{q,i} \in \mathbb{Q}[t_1, \dots, t_q]$ and $\deg(\gamma_{q,0}, t_i) < \hat{h}_i$, for $1 \leq i \leq q - 1$. Whence

$$\begin{aligned} \gamma &= \gamma_0 p^{2^a} + \gamma_1 \hat{H}_1^a + \cdots + \gamma_q \hat{H}_q^a \\ &= \gamma_0 p^{2^a} + \gamma_1 \hat{H}_1^a + \cdots + (\gamma_{q,0} + \gamma_{q,1} \hat{H}_1^a + \cdots + \gamma_{q,q-1} \hat{H}_{q-1}^a) \hat{H}_q^a \\ &= \gamma_0 p^{2^a} + (\gamma_1 + \gamma_{q,1} \hat{H}_1^a) \hat{H}_1^a + \cdots + (\gamma_{q-1} + \gamma_{q,q-1} \hat{H}_q^a) \hat{H}_{q-1}^a + \gamma_{q,0} \hat{H}_q^a \end{aligned}$$

Let $\hat{\gamma}_i := \gamma_i + \gamma_{q,i} \hat{H}_q^a$ for $1 \leq i \leq q - 1$, then

$$\gamma = \gamma_0 p^{2^a} + \hat{\gamma}_1 \hat{H}_1^a + \cdots + \hat{\gamma}_{q-1} \hat{H}_{q-1}^a + \gamma_{q,0} \hat{H}_q^a$$

Let $\gamma_{q,0} = \sum a_j t_q^j$, where $a_i \in L[t_1, \dots, t_{q-1}]$. Then $\gamma_{q,0} \hat{H}_q^a = \sum a_j \hat{H}_q^a t_q^j$. Using

reduction modulo T_{q-1} , for some $b_{j,i} \in L[t_1, \dots, t_q]$ one can write

$$\begin{aligned} \gamma_{q,0} \hat{H}_q^a &= \sum_j a_j \hat{H}_q^a t_q^j = \sum_j (a_j \hat{H}_q^a \bmod T_{q-1}) t_q^j + \sum_j \sum_{i=1}^{q-1} (b_{j,i} \hat{H}_i^a) t_q^j \\ &= \sum_j (a_j \hat{H}_q^a \bmod T_{q-1}) t_q^j + \sum_{i=1}^{q-1} \sum_j (b_{j,i} t_q^j) \hat{H}_i^a \end{aligned}$$

Taking $\alpha_i = \hat{\gamma}_i + \sum_j (b_{j,i} t_q^j)$ for $1 \leq i \leq q-1$, we have

$$\gamma = \gamma_0 p^{2^a} + \alpha_1 \hat{H}_1^a + \dots + \alpha_{q-1} \hat{H}_{q-1}^a + \sum_j (a_j \hat{H}_q^a \bmod T_{q-1}) t_q^j$$

then obviously,

$$\gamma - \sum_j (a_j \hat{H}_q^a \bmod T_{q-1}) t_q^j = \gamma_0 p^{2^a} + \alpha_1 \hat{H}_1^a + \dots + \alpha_{q-1} \hat{H}_{q-1}^a$$

Applying the induction hypothesis to the above equality coefficient-wise in t_q , there exist polynomials $\hat{\alpha}_i$ for $0 \leq i \leq q-1$, satisfying the degree constraints of the lemma, such that

$$\begin{aligned} \gamma &= \hat{\alpha}_0 p^{2^a} + \hat{\alpha}_1 \hat{H}_1^a + (\hat{\alpha}_2 \hat{H}_2^a \bmod T_1) + \dots + (\hat{\alpha}_{q-1} \hat{H}_{q-1}^a \bmod T_{q-2}) \\ &\quad + \sum_j (a_j \hat{H}_q^a \bmod T_{q-1}) t_q^j \end{aligned}$$

which is equal to

$$\gamma = \hat{\alpha}_0 p^{2^a} + \hat{\alpha}_1 \hat{H}_1^a + (\hat{\alpha}_2 \hat{H}_2^a \bmod T_1) + \dots + (\gamma_{q,0} \hat{H}_q^a \bmod T_{q-1})$$

Now we claim that the degree of the right side in t_q is less than h_q , which in turn proves the lemma. The proof of the claim is a direct conclusion of the fact that for any $1 \leq d \leq \deg((\gamma_{q,0} \hat{H}_q^a \bmod T_{q-1}), t_q)$ we have

$$c_{d,0} + c_{d,1} + c_{d,2} + \dots + c_{d,q-1} + c_{d,q} \neq 0$$

provided that $c_{d,q} \neq 0$, where $c_{d,0}$ is the coefficient of t_q^d in $\hat{\alpha}_0 p^{2^a}$, $c_{d,1}$ is the coefficient of t_q^d in $\hat{\alpha}_1 \hat{H}_1^a$, and $c_{d,i}$ is the coefficient of t_q^d in $(\hat{\alpha}_i \hat{H}_i^a \bmod T_{i-1})$ for $2 \leq i \leq q$. This is true because let ℓ be the smallest index for which $c_{d,\ell}$ is non-zero. Then the degree of $c_{d,\ell}$ in t_ℓ is at least \hat{h}_ℓ , while the degree of all other $c_{d,i}$ for $\ell < i \leq q$ in t_ℓ are strictly less than

\hat{h}_ℓ , since they are already reduced modulo T_i .

Replacing γ with $g_j - \hat{g}_j$, Lemma 39 states that there exist $\beta_{j,i}^a \in \mathbb{Q}[t_1, \dots, t_q]$, for $0 \leq i \leq q$, such that over \mathbb{K} we have

$$g_j - \hat{g}_j = \beta_{j,0}^a p^{2^a} + \beta_{j,1}^a \tilde{H}_1^a + (\beta_{j,2}^a \hat{H}_2^a \bmod T_1) + \dots + (\beta_{j,q}^a \hat{H}_q^a \bmod T_{q-1})$$

where $\beta_{j,i}^a$'s satisfy the degree constraints of the lemma. Looking at a polynomial as a vector of its coefficients, then the above equality states that the polynomial $g_j - \hat{g}_j$ is a vector in the vector space over \mathbb{Q} given by the following basis \mathbb{B} :

By convention let $T_0 = \{\}$. Assume that Γ_i be the set of all polynomials $(t_1^{\alpha_1} \dots t_q^{\alpha_q} \hat{H}_i^a \bmod T_{i-1})$ such that

- $0 \leq \alpha_k < \hat{h}_i$ for $1 \leq k < i$
- $0 \leq \alpha_k < h_i - \hat{h}_i$ for $k = i$
- $0 \leq \alpha_k < h_i$ for $i < k \leq q$

then define

$$\mathbb{B} := \{t_1^{\alpha_1} \dots t_q^{\alpha_q} p^{2^a} : 0 \leq \alpha_i < \hat{h}_i\} \cup \Gamma_1 \cup \dots \cup \Gamma_q$$

Now we know that the vector $g_i - \hat{g}_i$, that we are looking for, is an element of a vector space over \mathbb{Q} given by \mathbb{B} , but the question is how to find such a vector. This can be done using a lattice with the basis \mathbb{B} and its fundamental domain. Before introducing the lattice, we need some new definitions and notations.

Definition 6. Let E be a Euclidean space, $\Lambda \subset E$ a lattice with given basis $U = (u_i)$.

- We define the (open, centred) fundamental domain associated to U by

$$J := J(U) = \left\{ \sum \lambda_i u_i : (\lambda_i) \in \mathbb{R}^U, |\lambda_i| < 1/2 \right\}$$

- Let $B(0, r)$ be the open ball of radius r , centred at 0. We denote

$$r_{max} = r_{max}(U) := \sup\{r \in \mathbb{R}^+, B(0, r) \subset J\}$$

the radius of the largest ball inscribed in the closure of J .

Now let L be the lattice give by \mathbb{B} . The following lemma states how to find the vector g for a given vector \tilde{g} , such that they are congruent modulo the lattice L , which is the main lemma for factor reconstruction phase.

Lemma 40. *Let E, Λ, U, J , be as in Definition 6, and $|\cdot|^2$ the Euclidean norm. If $x \in E$, there exist one unique $y \in J$ such that*

$$x \equiv y \pmod{\Lambda}$$

In terms of coordinates (on a fixed arbitrary basis), let M be the matrix giving the (u_i) , then y is given by

$$y = x \pmod{M} := x - M \lfloor M^{-1}x \rfloor$$

As usual, $\lfloor x \rfloor := \lfloor x + 1/2 \rfloor$ is the operator rounding to nearest integer and is to be applied coordinate-wise.

PROOF. We prove the lemma for $y = x - M \lfloor M^{-1}x \rfloor$. It is clear that y is congruent to x modulo Λ , since $y - x = M \lfloor M^{-1}x \rfloor \in \Lambda$. Now we need to show that $y \in J$. Let $\lfloor M^{-1}x \rfloor = M^{-1}x + v$, where $v = (v_i)$ with $|v_i| < 1/2$. Then $M \lfloor M^{-1}x \rfloor = x + Mv$ and hence $y = x - M \lfloor M^{-1}x \rfloor = x - (x + Mv) = -Mv$. Now $|v_i| < 1/2$, for each i , implies $y \in J$. Finally we have to prove the uniqueness of $y \in J$. For the purpose assume there exist another $y' \in J$ congruent to x modulo Λ . Hence $y' - y \in \Lambda$. On the other hand, let $y' - y = \sum \lambda_i u_i$, then $|\lambda_i| < 1$, since both y and y' are in J . Since $y - y'$ is an element of the lattice Λ , it must be a linear combination of u_i 's with integer coefficients, so $\lambda_i = 0$ for all i 's, which implies $y = y'$.

In practice, working with fundamental domain is not easy, since we can not compute J efficiently. But instead, we can estimate the radius of the largest ball inscribed in the closure of J . Replacing J with $B(0, r)$ in Lemma 40 gives a weaker version of Lemma 40 as follows.

Lemma 41. *Let $E, \Lambda, U, J, r_{max}$ be as in Definition 6, and $|\cdot|^2$ the Euclidean norm. If $x \in E$, there is at most one $y \in E$ such that*

$$x \equiv y \pmod{\Lambda} \text{ and } |y| < r_{max}$$

If it exists, y is the unique element in J congruent to x modulo Λ . In terms of coordinates (on a fixed arbitrary basis), let M be the matrix giving the (u_i) , then y is given by

$$y = x \pmod{M} := x - M \lfloor M^{-1}x \rfloor$$

As usual, $\lfloor x \rfloor := \lfloor x + 1/2 \rfloor$ is the operator rounding to nearest integer and is to be applied coordinate-wise.

PROOF. From Lemma 40, we know that there exists a unique $y = x - M \lfloor M^{-1}x \rfloor \in J$ congruent to x modulo Λ . So If $|y| < r_{max}$ then the lemma follows.

We first recall the Gram-Schmidt orthogonalization process and LLL-reduced basis for a given set of vectors.

Definition 7. Given n linearly independent vectors b_1, \dots, b_n , the Gram-Schmidt orthogonalization of b_1, \dots, b_n is defined by $\hat{b}_i = b_i - \sum_{j=1}^{i-1} \mu_{i,j} \hat{b}_j$, where $\mu_{j,i} = \frac{\langle b_i, \hat{b}_j \rangle}{\langle \hat{b}_j, \hat{b}_j \rangle}$ and $\langle \cdot, \cdot \rangle$ stands for inner product of the input vectors.

Definition 8. A basis $\mathbb{B} = \{b_1, \dots, b_n\} \in \mathbb{R}^n$ is a LLL-reduced basis with quality ratio α if the following holds:

- for $1 \leq i \leq n$ and $j < i$, $|\mu_{j,i}| \leq 1/2$
- for $1 \leq i \leq n$, $\alpha \|\hat{b}_i\| \leq \|\mu_{i,i+1} \hat{b}_i + \hat{b}_{i+1}\|^2$

For the actual computation of such a LLL-reduced basis for a given basis \mathbb{B} , we refer to [20, Algorithm 2.6.3]. More precisely,

Theorem 7. Using the above notations, there exists an algorithm that computes the LLL-reduced basis with quality ratio $1/4 < \alpha \leq 1$ in at most $O(n^6 \log(B)^3)$ bit operations, where $|b_i|^2 \leq B$ for all $1 \leq i \leq n$.

Following Belabas [9, Proposition 3.10], one can find a lower bound for r_{max} as below.

Lemma 42. Assume (u_i) is LLL-reduced with quality ratio $1/4 < \alpha \leq 1$. Let $\gamma := (\alpha - 1/4)^{-1} \geq 4/3$ and $d := \dim(E)$. Then

$$r_{max}(U) \geq \frac{|u_1|}{2(3\sqrt{\gamma}/2)^{d-1}}$$

Let's go back to the main question of this section. As we said, we are going to reconstruct each coefficient g_j of the polynomial g from its image \hat{g}_j . Indeed we had the following relation between g_j and \hat{g}_j ,

$$g_j - \hat{g}_j = \beta_{j,0}^a p^{2^a} + \beta_{j,1}^a \tilde{H}_1^a + (\beta_{j,2}^a \hat{H}_2^a \bmod T_1) + \dots + (\beta_{j,q}^a \hat{H}_q^a \bmod T_{q-1})$$

whence $g_j - \hat{g}_j$ is an element of the vector space generated by \mathbb{B} . Multiplying both side of the equation by the common denominator θ of all $\beta_{j,i}^a$, we can consider $\theta(g_j - \hat{g}_j)$

as an element of the lattice generated by \mathbb{B} .

Let E be Euclidean space with dimension $[\mathbb{K} : \mathbb{Q}]$ and $\Lambda \subset E$ be a lattice given by the basis $U = \mathbb{B}$. Applying Lemma 41 on E, Λ and U for $x = \hat{g}_j$, there exist at most one vector y congruent to \hat{g}_i modulo Λ with $|y| < r_{max}$. So Lemma 41 can solve the problem of finding g_j if such a vector exists, and of course, the existence of such a vector depends on the radius r_{max} , which in turn depends on the precision a in our approximation of the modular factor \hat{g} .

Let B_j be a bound (using Euclidean norm) on the size of the coefficients of g_j , regarding as a polynomial in $\mathbb{Q}[t_1, \dots, t_q]$. Now if $r_{max} \geq B_j$, then we can be sure that such a vector exists and so Lemma 41 succeeds. If not, then we need to increase the precision a and try again. Now the question is why for any B_j there exists a precision a for which $r_{max} \geq B_j$. We need the following definition from [67].

Definition 9. Let \mathbb{K} be a number field with ring of integers $\mathcal{O}_{\mathbb{K}}$, and $I \subseteq \mathcal{O}_{\mathbb{K}}$ is an ideal. We define the norm $\text{Norm}(I)$ of an ideal I to be the cardinality of the quotient ring $\mathcal{O}_{\mathbb{K}}/I$.

Lemma 43. Let (u_i) be a basis of the lattice $U = I_a$, LLL reduced with quality ratio α and $\gamma := 1/(1 - \alpha)$, where $I_a = \langle p^{2^a}, \hat{H}_1^a, \dots, \hat{H}_q^a \rangle$. Let $\lambda > 0$ and $x \in \mathcal{O}_{\mathbb{K}}$ such that $T_2(x) < \lambda$ (see Section 4.3.3 for definition of T_2). Then we can apply Lemma 41 to reconstruct x uniquely from $x \bmod I_a$, provided that

$$\text{Norm}(I_a) \geq (2\sqrt{\lambda/N}(3\sqrt{\gamma}/2)^{N-1})^N$$

PROOF. We apply Lemma 41 to U . Since $u_1 \in I_a - \{0\}$, we have

$$T_2(u_1) \geq N(\text{Norm}(\langle u_1 \rangle))^{2/N} \geq N(\text{Norm}(I_a))^{2/N}$$

where the first inequality follows from the arithmetic geometric means and the second from the fact that $\langle u_1 \rangle \subseteq I_a$ and $\text{Norm}(\langle u_1 \rangle) \neq 0$. From Lemma 42, it follows that

$$r_{max} \geq \frac{\sqrt{N}(\text{Norm}(I_a))^{1/N}}{2(3\sqrt{\gamma}/2)^{N-1}}$$

and we can apply Lemma 41 as soon as $r_{max} \geq \sqrt{\lambda}$.

The above lemma states that if $\text{Norm}(I_a)$ is greater than some number, then x can be constructed from its image. But we need to prove that this can actually happen, which is equivalent to prove that for any λ there exist a precision a for which $r_{max} \geq \sqrt{\lambda}$, which

in turn is equivalent to prove that $\text{Norm}(I_a) > \text{Norm}(I_{a'})$ if $a > a'$. The following lemma addresses this problem.

Lemma 44. *Using the above notations, I_{a+1} is a proper subset of I_a for any $a \in \mathbb{N}$.*

PROOF. Since $\hat{H}_i^a = \hat{H}_i^{a+1} \pmod{p^{2^a}}$, so $I_{a+1} \subseteq I_a$. But $p^{2^a} \notin I_{a+1}$, since otherwise there exist polynomials $b_i \in \mathbb{Z}[t_1, \dots, t_q]$ such that

$$p^{2^a} = b_0 p^{2^{a+1}} + b_1 \hat{H}_1^{a+1} + \dots + b_q \hat{H}_1^{a+1}$$

taking the normal form of both sides with respect to the polynomials $\hat{H}_1^a, \dots, \hat{H}_q^a$, $p^{2^{a+1}}$ divides the valuation of the right side, but not that of the left side, a contradiction.

Following the above lemma, we have $\text{Norm}(I_{a+1}) > \text{Norm}(I_a)$, since the cardinality of the quotient ring $\mathcal{O}_{\mathbb{K}}/I_a$ is strictly less than the cardinality of the quotient ring $\mathcal{O}_{\mathbb{K}}/I_{a+1}$. Hence by increasing the precision a , after finitely many steps, we can reach a precision a for which $\text{Norm}(I_a)$ satisfies the condition of Lemma 43.

4.3.2 Factor Combination

In this section we solve the question of factor combination using lattice reductions introduced in Section 4.2.2 for a suitable lattice, similar to the one for factorization over \mathbb{Q} , in such a way that every true combination of the modular factors corresponds to a vector in the lattice.

Since the definition of trace is independent of the choice of the underlying field, so we will use all notations and definitions regarding to traces given in Section 4.2.2. It is also obvious that the recursive relation between the coefficients of a polynomial and its traces over a field \mathbb{F} is independent of the choice of \mathbb{F} and it holds over any field. Now we want to state the equivalent version of all lemmas given in Section 4.2.2 for the case of number field.

In the following, let $\mathbb{K}_p = \mathbb{Z}_p[t_1, \dots, t_q] / \langle \tilde{H}_1^*, \dots, \tilde{H}_q^* \rangle$, where \mathbb{Z}_p is the ring of p -adic integers, and \tilde{H}_i^* is the p -adic approximation of an irreducible factor of H_i such that $\tilde{H}_i^* = \tilde{H}_i$ modulo p .

The following lemma which is an extension of Lemma 30 to the case of number fields, is indeed a direct conclusion of the recursive relation between the coefficients of f and its traces.

Lemma 45. Let $g \in \tilde{\mathbb{K}}_p[x]$ be a monic p -adic factor of f over $\tilde{\mathbb{K}}_p$ and $\deg(g) = d$. The monic polynomial g has coefficients in \mathbb{K} if and only if $Tr_i(g) \in \mathbb{K}$ for all $1 \leq i \leq d$.

Lemma 47, which is the main lemma of this section, claims the same statement as Lemma 45 over the ring of algebraic integers $\mathcal{O}_{\mathbb{K}}$ instead of over \mathbb{K} , with an extra assumption $f \in \frac{1}{\lambda}\mathcal{O}_{\mathbb{K}}[x]$. For the proof we need the following lemma from [51].

Lemma 46. Let $f \in \frac{1}{\lambda}\mathcal{O}_{\mathbb{K}}[x]$ be a monic polynomial, where $0 \neq \lambda \in \mathbb{Z}$, and suppose $f(x) = g(x)h(x) \in \mathbb{K}[x]$, where g, h are monic. Then $g, h \in \frac{1}{\lambda}\mathcal{O}_{\mathbb{K}}[x]$.

Lemma 47. Let $f \in \frac{1}{\lambda}\mathcal{O}_{\mathbb{K}}[x]$ be a monic polynomial of degree n , where $0 \neq \lambda \in \mathbb{Z}$, and $g \in \tilde{\mathbb{K}}_p[x]$ be a monic p -adic factor of f over $\tilde{\mathbb{K}}_p$ and $\deg(g) = d$. The monic polynomial g of degree d has coefficients in $\frac{1}{\lambda}\mathcal{O}_{\mathbb{K}}$ if and only if $Tr_i(g) \in (\frac{1}{\lambda})^n\mathcal{O}_{\mathbb{K}}$ for all $1 \leq i \leq d$.

PROOF. If the monic polynomial g of degree d has coefficients in $\frac{1}{\lambda}\mathcal{O}_{\mathbb{K}}$, then the recursive relation between the coefficients of g and its traces implies $Tr_i(g) \in (\frac{1}{\lambda})^i\mathcal{O}_{\mathbb{K}} \subseteq (\frac{1}{\lambda})^n\mathcal{O}_{\mathbb{K}}$ for all $1 \leq i \leq d$. Now assume that all traces of g are in $(\frac{1}{\lambda})^n\mathcal{O}_{\mathbb{K}}$. Then lemma 45 implies $g \in \mathbb{K}[x]$, since $(\frac{1}{\lambda})^n\mathcal{O}_{\mathbb{K}} \subset \mathbb{K}$. Then Lemma 46 implies $g \in \frac{1}{\lambda}\mathcal{O}_{\mathbb{K}}[x]$, since $f \in \frac{1}{\lambda}\mathcal{O}_{\mathbb{K}}[x]$.

Next lemma is a weaker version of Lemma 47, concerning with only the necessary condition for g to be in $\frac{1}{\lambda}\mathcal{O}_{\mathbb{K}}[x]$. This lemma can be useful in practice, as it says rather than taking all traces of g , taking only a linear combination of them might be enough. So it reduces the number of information added to the lattice, resulting in applying LLL algorithm [42] on a smaller matrix which in turn can significantly improve the running time of the LLL algorithm, specially when g has a large degree.

Lemma 48. Let $f \in \frac{1}{\lambda}\mathcal{O}_{\mathbb{K}}[x]$ be a monic polynomial of degree n , where $0 \neq \lambda \in \mathbb{Z}$, and $g \in \tilde{\mathbb{K}}_p[x]$ be a monic p -adic factor of f over $\tilde{\mathbb{K}}_p$ and $\deg(g) = d$.

$$g \in \frac{1}{\lambda}\mathcal{O}_{\mathbb{K}}[x] \implies Tr_A(g) \in ((\frac{1}{\lambda})^n\mathcal{O}_{\mathbb{K}})^s$$

PROOF. A direct application of Newton identities, and the fact that the $s \times d$ matrix A is an integer matrix.

Following Section 4.3.4, there exists an integer D such that

$$\mathbb{Z}[t_1, \dots, t_q] \subseteq \mathcal{O}_{\mathbb{K}} \subseteq \frac{1}{D}\mathbb{Z}[t_1, \dots, t_q]$$

Then Lemma 47 and 48 still hold if we replace $\mathcal{O}_{\mathbb{K}}$ with $\frac{1}{D}\mathbb{Z}[t_1, \dots, t_q]$. On the other hand, since \mathbb{K} is the field of fractions of $\mathcal{O}_{\mathbb{K}}$, so every coefficient of $f \in \mathbb{K}[x]$ can be written as a fraction with numerator and denominator in $\mathcal{O}_{\mathbb{K}}$. So then $C_n f \in \mathcal{O}_{\mathbb{K}}[x]$,

where $C_n \in \mathcal{O}_{\mathbb{K}}$ is the least common divisor of the denominators of the coefficients of f . Hence we can always assume that f is a polynomial over $\mathcal{O}_{\mathbb{K}}$, but not necessarily monic. From now on, for simplicity of the following discussion, we will assume that $f \in \mathcal{O}_{\mathbb{K}}[x]$ is a monic polynomial. The non-monic case can be dealt in the same way as it was explained in Remark 1. More precisely, we assume that $f \in \frac{1}{D}\mathbb{Z}[t_1, \dots, t_q][x]$ for some $D \in \mathbb{Z}$. So then, using Lemma 46, each monic factor g of f over \mathbb{K} is in $\frac{1}{D}\mathbb{Z}[t_1, \dots, t_q][x]$, and Lemma 48 implies that $Tr_A(g) \in ((\frac{1}{D})^n \mathbb{Z}[t_1, \dots, t_q])^s$.

Now let $S \subset \mathbb{K}_p[x]$ be a subset of p -adic factors $\{f_1, \dots, f_r\}$ of f and let g be the product of the polynomials in S , and $\deg(g) = d$. Then

$$T_A(g) = \sum_{f_i \in S} T_A(f_i)$$

or equivalently, if we encode the set S as 0 – 1 vector $v = (v_1, \dots, v_r)$, where $v_i = 1$ if $f_i \in S$ and $v_i = 0$ otherwise, then

$$T_A(g) = \sum_{i=1}^r v_i T_A(f_i)$$

Following Lemma 48, a necessary condition for g to be in $\frac{1}{D}\mathbb{Z}[t_1, \dots, t_q][x]$ is that the sum of $T_A(f_i)$'s, $f_i \in S$, has entries in $(\frac{1}{D})^n \mathbb{Z}[t_1, \dots, t_q]$. However, the $T_A(f_i)$ is in $\mathbb{K}_p[x]$, so they can only be determined up to some finite precision. Let B_i be a bound on the norm of $Tr_i(g)$ for any factor g of f in $\mathbb{K}[x]$, which can be computed using Section 4.3.3. This allows us to compute an upper bound for the entries of $T_A(g)$. Now, a necessary condition for g to be a polynomial in $\frac{1}{D}\mathbb{Z}[t_1, \dots, t_q][x]$ is that $T_A(g)$ satisfies the bound in each row.

From now on, we want to solve the problem of factor combination using the result of Lemma 48 and following the same approach as in Section 4.2.2. So we need to build a lattice such that each solution of the factor combination problem be an element of the lattice and can be obtained as a short vector in the lattice.

Following the notations in Theorem 6, let W be the set of all vectors $v = (v_1, \dots, v_r) \in \mathbb{Z}^r$ for which $\prod_{i=1}^r f_{a_i}^{v_i}$ corresponds to a true factor of f over \mathbb{K} . Note that if g_1, \dots, g_t are the monic irreducible factors of f over \mathbb{K} , then $\{w_1, \dots, w_t\}$ is a basis of W in reduced echelon form, where w_j is defined as the 0 – 1

vector $(v_{j,1}, \dots, v_{j,r})$ for which $\prod_{i=1}^r f_{a,i}^{v_{j,i}}$ corresponds to the true factor g_j of f over \mathbb{K} . Note that reconstruction of g_j , for $1 \leq j \leq t$, from its modular image $\prod_{i=1}^r f_{a,i}^{v_{j,i}}$ can be done using Section 4.3.1. Finding this reduced basis is the same as solving the combinatorial problem in the algorithm of Zassenhaus [69], that is, finding a partition of the set of all modular factors $\{f_{a,i}\}_{i=1}^r$ of f into t distinct subsets, each of which, corresponds to one of the true factors $\{g_1, \dots, g_t\}$.

Let $L \subseteq \mathbb{Z}^r$ be a lattice, then B_L denotes a basis for L . The matrix whose rows are the elements of B_L is denoted by (B_L) and $\text{rref}(B_L)$ denotes its row echelon form. If we can compute any basis B_W of W , then the combinatorial problem is solved because $\{w_1, \dots, w_t\}$ are the rows of $\text{rref}(B_W)$.

Lemma 49. *Let L be a lattice such that $W \subseteq L \subseteq \mathbb{Z}^r$ and $R = \text{rref}(B_L)$. Then $L = W$ if and only if the following two conditions hold:*

- (a) *Each column of R contains precisely one 1, all other entries are 0*
- (b) *if (v_1, \dots, v_r) is a row of R then $g \in \mathcal{O}_{\mathbb{K}}$, where g is a true factor of f over \mathbb{K} .*

PROOF. If $L = W$ then $\{w_1, \dots, w_t\}$ must be the rows of R because of the uniqueness of row echelon form, and thus both conditions hold. Conversely, assume that both conditions hold. Since each row of R corresponds to a true factor of f and $\mathcal{O}_{\mathbb{K}}$ has unique factorization, so each row of R is a linear combination of w_1, \dots, w_t . But then (a) and the properties of row echelon form imply that the rows of R are actually the vectors w_1, \dots, w_t . Hence $L \subseteq W$, and so $L = W$.

Lemma 49 gives us a sufficient condition for L to be the desired lattice W . So, like in the van Hoeij algorithm [63], we start from $L = \mathbb{Z}^r$ and at each step we test whether L is equal to W or not. This test can be done using Lemma 49. Now suppose that $L \neq W$. Then the algorithm tries to find a new lattice L' with

$$W \subseteq L' \subseteq L$$

Then L replaced by L' . The algorithm keeps repeating this until we finally obtain $L = W$, i.e., both conditions of Lemma 49 are satisfied. Like any other algorithm, we need to address the questions of correctness and termination of the process.

Assume that we have chosen an $s \times d$ matrix A and all modular factors $\{f_{a,i}\}_{i=1}^r$ have been computed to the precision 2^a . We will now show how to compute the new lattice $L' \subseteq L$, hopefully of smaller dimension than L , that nevertheless contains all

solutions of the factor combination problem. Note that the new lattice L' would depend on the size of the matrix A and the precisions 2^a ; if we can not find a new lattice L' satisfying the required conditions using the following process, then we should change these parameters.

Let B_L be a basis for L . Initially $L = \mathbb{Z}^r$ and B_L is the standard basis of \mathbb{Z}^r . Now we will construct a new lattice Δ such that the vector

$$v_S = (Cv_1, \dots, Cv_r, T_A(\hat{g})_{11}, \dots, T_A(\hat{g})_{s1})$$

is an element of Δ for every solution S of the factor combination problem, where $\hat{g} = \prod_{i=1}^r f_{a,i}^{v_i}$ and C is a constant described below. Note that $v_i = 1$ if $f_{a,i} \in S$ and $v_i = 0$ otherwise.

The lattice Δ is given by

$$\begin{pmatrix} CI_{r \times r} & S \\ 0 & Q \end{pmatrix}$$

where

- $S = \text{lift}(D^n S_{\mathbb{K}_p})$ and

$$S_{\mathbb{K}_p} = \begin{pmatrix} T_A(f_1)_{11} & \cdots & T_A(f_1)_{s1} \\ \vdots & \ddots & \vdots \\ T_A(f_r)_{1r} & \cdots & T_A(f_r)_{sr} \end{pmatrix}$$

and $\text{lift}(x) = x \bmod p^{2^a}, \hat{H}_1^a, \dots, \hat{H}_q^a$.

- Q is a $Ns \times Ns$ block diagonal matrix, with blocks equal to M on the diagonal, where M is a $N \times N$ matrix obtained from the basis \mathbb{B} mentioned in Section 4.3.1.

To any true factor of f corresponds $u \in \{0, 1\}^r$ and $v \in \mathbb{Z}^{Ns}$ such that the image of $(u \ v)$ has squared L_2 norm bounded by

$$C^2 r + \|Su + Qv\|_2^2$$

and we can bound $\|Su + Qv\|_2 \leq B_{\text{trace}}$ using Section 4.3.3. The constant C is chosen so that $C^2 r \approx B_{\text{trace}}$. It is not necessary at this point that M be LLL-reduced, nor that we use the lift specified above, although both conditions certainly speed up the reduction.

So any factor of f over \mathbb{K} corresponds to a μ -short vector of the new lattice Δ , where μ is an upper bound on the square root of $C^2 r + \|Su + Qv\|_2^2$, and can be computed using Section 4.3.3.

Now we are ready to present our new algorithm for factoring polynomials over number fields using multivariate representation. Let \mathbb{K} be the algebraic number field, as before. For a given square-free monic polynomial $f \in \mathbb{K}[x]$, this algorithm computes the irreducible factors of f over \mathbb{K} using the following steps:

1. Determine $D \in \mathbb{N}$, such that f and its factors are in $\frac{1}{D}\mathbb{Z}[t_1, \dots, t_q][x]$
2. Choose a suitable prime $p \nmid D$, satisfying the conditions explained at the beginning of the section
3. Compute the modular factors $f_{a,1}, \dots, f_{a,r}$ using Theorem 6, up to some precision 2^a , where a should be at least $\lceil \log(\mu)/\log(p) \rceil$ and μ is obtained in the next step.
4. Choose a matrix A . Compute the upper bound μ , as explained above
5. Compute the lattice Δ as above, if the entries of S are already small, that is, all vectors in the basis of Δ are already μ -short, go back to step 4 and choose another matrix A .
6. Apply the LLL algorithm to compute a reduced basis V_1, \dots, V_r of the lattice Δ . Do a floating point Gram-Schmidt computation to determine an as small as possible integer t such that all μ -short vectors of Δ are in $\Delta' := \mathbb{Z}V_1 + \dots + \mathbb{Z}V_t$. Let L' be $1/C$ times the projection of Δ' on the first r coordinates with the basis $B_{L'}$. If the dimension of L' did not decrease then return to step 3 and increase the precision.
7. If $t = 1$ then f must be irreducible and the computation ends. Otherwise continue.
8. Compute $R = \text{rref}(B_{L'})$. If R does not satisfy the first condition of Lemma 49 then go to step 4, otherwise proceed to the next step.
9. Check the second condition of Lemma 49 by executing step 10 for $k \in \{1, \dots, t\}$
10. Let (v_1, \dots, v_r) be the k 'th row of R . Then check if g_k divides f over \mathbb{K} , where g_k is the reconstructed polynomial from its modular image $\prod_{i=1}^r f_i^{v_i}$ (Section 4.3.1). Otherwise go back to step 4.
11. Now $f = g_1 \cdots g_t$

The correctness of the algorithm is clear, since the output satisfies the conditions of Lemma 49 in steps 8-10. Indeed, in step 10, each output polynomial g_j is checked whether it is a true factor of f or not.

Finally we need to prove that the algorithm terminates. Since the termination of the algorithm depends only on the lattice Δ and the way that we generate the new lattice L' , and in this sense, it is the same as van Hoeff's algorithm [63], so the proof is similar to the one given for Lemma 8 in [63]. But for more clearness, we give the first part of the proof which is different

Lemma 50. *The algorithm terminates.*

PROOF. Since in each round of the algorithm, the main goal is to find a new lattice L' , such that $W \subseteq L' \subseteq L$ and $\dim(L') < \dim(L)$ if $L \neq W$, so we have to show that if $L \neq W$ then eventually $\dim(L') < \dim(L)$, so that after finitely many steps the algorithm reaches $L = W$.

Without loss of generality, we assume that the matrix A has only one row, with the i th entries equal to one and other entries zero. So then $s = 1$ and $T_A(f_j) = Tr_i(f_j)$. Denote

$$U(v) = \sum_{j=1}^r v_j Tr_i(f_j) \in \mathbb{Z}_p[t_1, \dots, t_q] \quad : \quad v = (v_1, \dots, v_r) \in L$$

where \mathbb{Z}_p is the ring of p -adic integers and define $U(v, a_1)$ as the image of $DU(v)$ modulo p^{a_1} . Denote $\tilde{\mu} = r2^{r/2}\mu$, $Sol_i(L) := \{v \in L : DU(v) \in \mathbb{Z}[t_1, \dots, t_q]\}$, and $B(L, a_1) := \{v \in L : |Cv|^2 + |U(v, a_1)|^2 \leq \tilde{\mu}^2\}$. Since $\tilde{\mu} > \mu$, then it follows that W is contained in the span of $B(L, a_1)$. The rest of the proof is exactly the same as the one given in [63] for Lemma 38.

Truncation As we said in Section 4.2.2, truncating the value of $T_A(g)$ can have a significant effect on the efficiency of the van Hoeff factorization algorithm over \mathbb{Z} in practice. Now we want to extend a similar idea to the case of factorization over number fields.

For $t > 1$ any integer, and any $x \in \tilde{\mathbb{K}}_p^a$, we define the truncation

$$T_{\mathbb{K}}^{a,t}(x) := \lfloor \text{lift}(x)/t \rfloor$$

Now write

$$\begin{aligned} S &= S_0 + tS_1, & \| S_0 \|_\infty &\leq t/2 \\ Q &= Q_0 + tQ_1, & \| Q_0 \|_\infty &\leq t/2 \end{aligned}$$

Hence

$$\begin{aligned} \| S_0 \|_2 &\leq \sqrt{rNs} \frac{t}{2} \\ \| Q_0 \|_2 &\leq \sqrt{s} N \frac{t}{2} \end{aligned}$$

From the previous section, there exists $u \in \{0, 1\}^r$ and $v \in \mathbb{Z}^{Ns}$ such that

$$\| Su + Qv \|_2 \leq B_{trace}$$

whence

$$\begin{aligned} \| S_1u + Q_1v \|_2 &\leq B_{trace}/t + \| S_0u + Q_0v \|_2 /t \\ &\leq B_{trace}/t + \frac{\sqrt{rNs}}{2} \| u \|_2 + \frac{\sqrt{s}N}{2} \| v \|_2 \end{aligned}$$

Note that in the case of van Hoeij algorithm, $Q_0 = 0$, since Q is divisible by $t = p^b$. So it is not important to control v , which is necessary in this case. For the purpose, we have to make the following two assumptions.

- the precision a is so large that we can apply Lemma 41 to the matrix M . From this we deduce that $v = -\lfloor Q^{-1}Su \rfloor$.
- the specified lift is chosen for S , such that $\| Q^{-1}S \|_\infty \leq 1/2$

If $x \in \mathbb{R}^N$, we have $\lfloor x \rfloor = x + \varepsilon$, where $\| \varepsilon \|_\infty \leq 1/2$, hence

$$\| \lfloor x \rfloor \|_2 \leq \| x \|_2 + \sqrt{N}/2$$

From which we have

$$\| v \|_2 \leq \| Q^{-1}Su \|_2 + \sqrt{N}/2 \leq \frac{\sqrt{rNs}}{2} \| u \|_2 + \sqrt{N}/2$$

since $\| u \|_2 \leq \sqrt{r}$, we obtain

$$\| S_1u + Q_1v \|_2 \leq B_{high} := B_{trace}/t + \frac{r\sqrt{Ns}}{2}(1 + N\sqrt{s}/2) + \frac{N\sqrt{Ns}}{4}$$

So our final knapsack lattice is given by

$$\begin{pmatrix} CI_{t \times r} & S_1 \\ 0 & Q_1 \end{pmatrix}$$

where $C \geq 1$ is chosen so that $C^2 r \approx B_{high}^2$.

Note that if t is a power of p and p divides Q , then $Q_0 = 0$, in which case the bound becomes $B_{trace}/t + \frac{r\sqrt{Ns}}{2}$. Here we essentially recover van Hoeij's bound of $r\sqrt{s}/2$ in the case $N = 1$, since $\mathbb{K} = \mathbb{Q}$, provided we can take $t \gg B_{trace}$, i.e., provided that modular factors have been sufficiently lifted.

4.3.3 Bound on the Traces

In this section we will discuss how to determine an upper bound on the size of the traces of a polynomial over \mathbb{K} . Before explaining how to measure the size of an element of a field, we first need to know how to measure it. It is classical to measure the size of an element $x \in \mathcal{O}_{\mathbb{K}}$ in terms of the quadratic form $T_2(x) = \sum_{\sigma} |x^{\sigma}|^2$ where σ runs through the $[\mathbb{K} : \mathbb{Q}]$ embeddings of \mathbb{K} into \mathbb{C} and $x^{\sigma} = \sigma(x)$.

Lemma 51. *Let g be a monic divisor of f over \mathbb{K} . then for all integer $k \geq 0$ we have*

$$T_2(Tr_k(g)) \leq \deg(f)^2 \sum_{\sigma} B_{root}^{2k}(f^{\sigma})$$

where $B_{root}(h)$ is any bound for the modulus of the complex roots of h .

PROOF. Following equalities come from the definitions of T_2 and Tr_k ,

$$Tr_k(g) = \sum_{r \in Roots(g)} r^k$$

$$T_2(Tr_k(g)) = \sum_{\sigma} |Tr_k(g)^{\sigma}|^2$$

where $Roots(g)$ contains all roots of $g \in \mathbb{K}[x]$ in algebraic closure of \mathbb{K} . Since $Roots(g) \subseteq Roots(f)$, then

$$T_2(Tr_k(g)) = \sum_{\sigma} |Tr_k(g)^{\sigma}|^2 = \sum_{\sigma} \left| \sum_{r \in Roots(g)} r^k \right|^2$$

$$\begin{aligned}
&\leq \sum_{\sigma} |(\sum_{r \in \text{Roots}(f)} r^k)^{\sigma}|^2 \leq \sum_{\sigma} (\sum_{r \in \text{Roots}(f)} |r^{\sigma}|^k)^2 \\
&\leq \sum_{\sigma} (\sum_{r \in \text{Roots}(f)} B_{\text{root}}^k(f^{\sigma}))^2 = \sum_{\sigma} (\deg(f) B_{\text{root}}^k(f^{\sigma}))^2 \\
&= \deg(f)^2 \sum_{\sigma} B_{\text{root}}^{2k}(f^{\sigma})
\end{aligned}$$

Now an upper bound B on the all traces of all factors of f over \mathbb{K} can be obtained by adding up $T_2(\text{Tr}_k(g))$ for all k , i.e.,

$$B = \deg(f)^2 \sum_k \sum_{\sigma} B_{\text{root}}^{2k}(f^{\sigma})$$

which only depends on f and the number of traces.

In practical computations, it is more convenient to use Euclidean norm related to our specified basis (ω_i) , or in our case usually standard basis. For $x = \sum x_i \omega_i \in \mathbb{K}$, we let $|x|^2 := \sum |x_i|^2$. The following lemma states a relation between this norm and T_2 .

Lemma 52. *Let $M = (m_{ij}) \in M_d(\mathbb{Q})$ be the matrix such that $(\omega_i) = (u_i)M$, where (u_i) is another basis for \mathbb{K} as a vector spaces over \mathbb{Q} and $d = [\mathbb{K} : \mathbb{Q}]$. Now let $V = ((u_i^{\sigma})^{j-1})_{\sigma, 1 \leq j \leq d}$. Then*

$$|x|^2 \leq C_{T_2} T_2(x)$$

where $C_{T_2} = \|M^{-1}V^{-1}\|_2^2$, and $\|(a_{ij})\|_2 = (\sum |a_{ij}|^2)^{1/2}$.

PROOF. Let $x = (\omega_i)^t(x_i) = (u_i)M(x_i)$, $(x_i) \in \mathbb{Q}^d$. Writing the d different embeddings of this equation in \mathbb{C} , we obtain

$$(x^{\sigma})^t = VM^t(x_i)$$

hence $|x|^2 \leq C_{T_2} T_2(x)$ by Cauchy-Schwarz.

The last note which is needed to be mentioned here is how to compute $B_{\text{root}}(f^{\sigma})$, indeed for any embedding σ , we need to compute an upper bound for the modulus of the complex roots of f^{σ} . This can be done using the following steps,

- Finding the embedding σ
- Computing f^{σ}
- Computing a bound on the modulus of the complex roots of f^{σ}

Let $V = v(\langle H_1, \dots, H_q \rangle) \subset \mathbb{C}^q$, the variety of the ideal $\langle H_1, \dots, H_q \rangle \subset \mathbb{C}[t_1, \dots, t_q]$. So, for a given $v \in V$, the embedding $\sigma_v : \mathbb{K} \rightarrow \mathbb{C}$ is just evaluation of each element of \mathbb{K} at v . So one way to compute σ_v is to compute V in \mathbb{C}^q up to some precision. This can be done using [?, Theorem 19.2] by controlling the precisions of the roots at each step, that is, we first find the set of roots V_1 of the polynomial $H_1(t_1)$ in \mathbb{C} up to some precision using [?, Theorem 19.2], then for each element $u \in V_1$ compute $H_2^u(t_2) = H_2(u, t_2)$ and then compute the set of roots H_2^u in \mathbb{C} up to some precision using [?, Theorem 19.2], and so on. Discussion about how to control the errors on the roots at each step to reach the required precisions on the elements of V is out of the scope of this section. Having σ , computing f^σ is just an evaluation. A bound on the modulus of the complex roots of f^σ can be found using [34].

4.3.4 Choosing a Denominator

Let \mathbb{K} be a number field, and $\mathcal{O}_{\mathbb{K}}$ its ring of integers. In this section, we will show that there exists a non-zero integer Δ such that

$$\mathbb{Z}[t_1, \dots, t_q] \subseteq \mathcal{O}_{\mathbb{K}} \subseteq \frac{1}{\Delta} \mathbb{Z}[t_1, \dots, t_q]$$

Recall that \mathbb{K} is a \mathbb{Q} -vector space of dimension $d = [\mathbb{K} : \mathbb{Q}]$. A natural question is whether or not a similar statement can be made about $\mathcal{O}_{\mathbb{K}}$ as a \mathbb{Z} -module. Remarkably, it turns out that the strongest analogue of the \mathbb{Q} -statement is true: $\mathcal{O}_{\mathbb{K}}$ is a free \mathbb{Z} -module of rank d . We will prove this fact in this section.

Let $\alpha_1, \dots, \alpha_d$ be a \mathbb{Q} -basis for \mathbb{K} . Further assume that the α_i are all algebraic integers; this can be done by applying the following Lemma to any \mathbb{Q} -basis for \mathbb{K} .

Lemma 53. *Let $\alpha \in \mathbb{K}$. Then there is some $a \in \mathbb{Z}$ such that $a\alpha \in \mathcal{O}_{\mathbb{K}}$.*

PROOF. Let $f(x) = x^n + a_{n-1}x^{n-1} + \dots + a_0 \in \mathbb{Q}[x]$ be the minimal polynomial of α . Let $a \in \mathbb{Z}$ be some integer such that $af(x) \in \mathbb{Z}[x]$. Let $g(x)$ be the monic polynomial

$$x^n + aa_{n-1}x^{n-1} + a^2a_{n-2}x^{n-2} + \dots + a^n a_0$$

which is in $\mathbb{Z}[x]$ since $af(x)$ is. We have

$$g(a\alpha) = a^n \alpha^n + a^n a_{n-1} x^{n-1} + \dots + a^n a_0 = a^n f(\alpha) = 0$$

Thus a satisfies a monic polynomial with integral coefficients, and therefore lies in $\mathcal{O}_{\mathbb{K}}$.

Since the α_i satisfy no linear dependence with \mathbb{Q} -coefficients, they certainly satisfy no linear dependence with \mathbb{Z} -coefficients. Thus

$$\mathbb{Z}\alpha_1 + \mathbb{Z}\alpha_2 + \cdots + \mathbb{Z}\alpha_d$$

is a free \mathbb{Z} -module of rank d . Furthermore, it is clearly contained in $\mathcal{O}_{\mathbb{K}}$, thus $\mathcal{O}_{\mathbb{K}}$ contains a free \mathbb{Z} -module of rank d . To prove that $\mathcal{O}_{\mathbb{K}}$ itself is a free \mathbb{Z} -module, we need to find some free \mathbb{Z} -module of rank d which contains $\mathcal{O}_{\mathbb{K}}$. for the purpose, we need the following definition.

Definition 10. Let \mathbb{K} be a number field of degree d with complex embeddings $\sigma_1, \dots, \sigma_d$. Let $\alpha_1, \dots, \alpha_d$ be elements of \mathbb{K} . The discriminant $\Delta(\alpha_1, \dots, \alpha_d)$ of this n -tuple is defined to be the square of the determinant of the $d \times d$ matrix

$$(\sigma_i(\alpha_j))$$

The following Lemma gives an equivalent definition of discriminant Δ .

Lemma 54. Let \mathbb{K} be a number field as above and let $\alpha_1, \dots, \alpha_d$ be elements of \mathbb{K} . Then $\Delta(\alpha_1, \dots, \alpha_d)$ is equal to the square of the determinant of the $d \times d$ matrix

$$(Tr_{\mathbb{K}/\mathbb{Q}}(\alpha_i\alpha_j))$$

PROOF. Let $A = (\sigma_i(\alpha_j))$. Since $\det(A^t) = \det(A)$ (where A^t is the transpose of A), we see that $\Delta(\alpha_1, \dots, \alpha_d)$ is equal to the square of the determinant of A^t . The ij entry of this matrix is

$$\sum_{k=1}^d \sigma_k(\alpha_i)\sigma_k(\alpha_j) = \sum_{k=1}^d \sigma_k(\alpha_i\alpha_j) = Tr_{\mathbb{K}/\mathbb{Q}}(\alpha_i\alpha_j)$$

by property of $Tr_{\mathbb{K}/\mathbb{Q}}$. This proves the lemma.

Lemma 55. $\Delta(\alpha_1, \dots, \alpha_d) \in \mathbb{Q}$. If the α_i are all algebraic integers, then $\Delta(\alpha_1, \dots, \alpha_d) \in \mathbb{Z}$.

PROOF. Since $Tr_{\mathbb{K}/\mathbb{Q}}(\alpha)$ for $\alpha \in \mathbb{K}$ is a map from \mathbb{K} into \mathbb{Q} , so $\Delta(\alpha_1, \dots, \alpha_d) \in \mathbb{Q}$ follows immediately from Lemma 54. Additionally, if $\alpha \in \mathcal{O}_{\mathbb{K}}$, then $Tr_{\mathbb{K}/\mathbb{Q}}(\alpha) \in \mathbb{Z}$, since its minimal polynomial is a monic polynomial with integer coefficients. Hence $\Delta(\alpha_1, \dots, \alpha_d) \in \mathbb{Z}$, if $\alpha_i \in \mathcal{O}_{\mathbb{K}}$ for all i .

The following lemma states that Δ is a common denominator of any \mathbb{Q} -linear combination of α_i 's in \mathbb{K} .

Lemma 56. Let \mathbb{K} be a number field of degree d and let $\alpha_1, \dots, \alpha_d$ be a \mathbb{Q} -basis for \mathbb{K} consisting entirely of algebraic integers. Set $\Delta = \Delta(\alpha_1, \dots, \alpha_d)$. Fix $\alpha \in \mathcal{O}_K$ and write

$$\alpha = a_1\alpha_1 + \dots + a_d\alpha_d$$

with each $a_i \in \mathbb{Q}$. Then $\Delta a_i \in \mathbb{Z}$ for all i .

PROOF. First of all, note that Δ is a non-zero integer, since α_i 's are a \mathbb{Q} -basis of \mathbb{K} . Now applying the embedding σ_i to the expression for α , yields

$$\sigma_i(\alpha) = a_1\sigma_i(\alpha_1) + \dots + a_d\sigma_i(\alpha_d)$$

This can be considered to be a system of d linear equations in the d unknowns a_1, \dots, a_d , that is, we have the matrix equation

$$\begin{pmatrix} \sigma_1(\alpha) \\ \sigma_2(\alpha) \\ \vdots \\ \sigma_d(\alpha) \end{pmatrix} = \begin{pmatrix} \sigma_1(\alpha_1) & \sigma_1(\alpha_2) & \cdots & \sigma_1(\alpha_d) \\ \sigma_2(\alpha_1) & \sigma_2(\alpha_2) & \cdots & \sigma_2(\alpha_d) \\ \vdots & \vdots & & \vdots \\ \sigma_d(\alpha_1) & \sigma_d(\alpha_2) & \cdots & \sigma_d(\alpha_d) \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_d \end{pmatrix}$$

By Cramers rule, this has the unique solution $a_i = \gamma_i/\delta$, where δ is the determinant of $A = (\sigma_i(\alpha_j))$ (so that $\delta^2 = \Delta$, in particular, the solution is unique since $\Delta \neq 0$) and γ_i is the determinant of the matrix obtained from A by replacing the i th column by $(\sigma_j(\alpha))$. Note that both γ_i and δ are algebraic integers, since each entry in each matrix is. Since $\delta^2 = \Delta$, we have

$$\Delta a_i = \delta \gamma_i$$

The left-hand side is rational and the right-hand side is an algebraic integer, so both sides must be rational integers. This proves the lemma.

Finally, the following theorem states that the ring of algebraic integers is a free \mathbb{Z} -module.

Theorem 8. Let \mathbb{K} be a number field with ring of integers \mathcal{O}_K . Let $d = [\mathbb{K} : \mathbb{Q}]$. Then \mathcal{O}_K is a free \mathbb{Z} -module of rank d .

PROOF. Let $\alpha_1, \dots, \alpha_d$ be a \mathbb{Q} -basis for \mathbb{K} consisting entirely of algebraic integers. We have

$$\mathbb{Z}\alpha_1 + \dots + \mathbb{Z}\alpha_d$$

and by Lemma 56 we have

$$\mathcal{O}_{\mathbb{K}} \subseteq \frac{1}{\Delta}(\mathbb{Z}\alpha_1 + \cdots + \mathbb{Z}\alpha_d)$$

where $\Delta = \Delta(\alpha_1, \dots, \alpha_d)$. Thus we have shown that $\mathcal{O}_{\mathbb{K}}$ lies between two free \mathbb{Z} -modules of rank d , thus $\mathcal{O}_{\mathbb{K}}$ itself is a free \mathbb{Z} -module of rank d .

So far, we have proved that there exists a non-zero integer Δ such that

$$\mathbb{Z}[t_1, \dots, t_q] \subseteq \mathcal{O}_{\mathbb{K}} \subseteq \frac{1}{\Delta}\mathbb{Z}[t_1, \dots, t_q]$$

Note that here by abuse of notation we are denoting t_i modulo H_1, \dots, H_q by t_i . Also it is necessary to be mentioned that the set $S = \{t_1^{\beta_1} \cdots t_q^{\beta_q} : 0 \leq \beta_i < h_i, 1 \leq i \leq q\}$ is a \mathbb{Q} -basis for \mathbb{K} consisting entirely of algebraic integers and $\mathbb{Z}[t_1, \dots, t_q] = \bigoplus_{\alpha \in S} \mathbb{Z}\alpha$.

But now the question is how to find such a Δ in practice. The following theorem states how the discriminant of elements of S is related to the resultants of generating polynomials of the number field \mathbb{K} .

Theorem 9. *Let S be as above, then*

$$\Delta(S) = (\text{res}(H_1, \dots, \text{res}(H_{q-1}, \text{res}(H_q, \frac{\partial H_1}{\partial t_1} \cdots \frac{\partial H_{q-1}}{\partial t_{q-1}} \frac{\partial H_q}{\partial t_q}))))^2$$

PROOF. We prove the theorem by induction on q . let $q = 1$, then $S = \{1, t_1, \dots, t_1^{h_1-1}\}$ and we need to prove

$$\Delta(S) = (\text{res}(H_1, \frac{\partial H_1}{\partial t_1}))^2$$

for the purpose assume that $\alpha_1, \dots, \alpha_{h_1}$ are the complex roots of H_1 in \mathbb{C} . Then

$$\Delta(S) = \det \begin{pmatrix} 1 & \alpha_1 & \cdots & \alpha_1^{h_1-1} \\ 1 & \alpha_2 & \cdots & \alpha_2^{h_1-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_{h_1} & \cdots & \alpha_{h_1}^{h_1-1} \end{pmatrix}^2 = \left(\prod_{i < j} (\alpha_i - \alpha_j) \right)^2 = (\text{res}(H_1, \frac{\partial H_1}{\partial t_1}))^2$$

which proves the induction base case. Now assume that the statement of theorem holds for $q - 1$. To prove it for q , let

$$V = v_{\mathbb{C}}(\langle H_1, \dots, H_q \rangle)$$

be the variety of the ideal $\langle H_1, \dots, H_q \rangle$ over \mathbb{C} , and take the standard basis $S = \{t_1^{\beta_1} \cdots t_q^{\beta_q} : 0 \leq \beta_i < h_i, 1 \leq i \leq q\}$ as the \mathbb{Q} -basis for \mathbb{K} . Note that $|V| = |S|$. Let embedding $\sigma_v : \mathbb{K} \rightarrow \mathbb{C}$ corresponds to evaluation at the complex root $v \in V$. Consider the following ordering on the set S ,

$$1 > t_1 > \cdots > t_q$$

$$t_1^{\beta_1} \cdots t_q^{\beta_q} > t_1^{\gamma_1} \cdots t_q^{\gamma_q} \iff (\gamma_1 - \beta_1, \dots, \gamma_q - \beta_q) \geq 0$$

Now let $\tilde{S} = \{t_1^{\beta_1} \cdots t_{q-1}^{\beta_{q-1}} : 0 \leq \beta_i < h_i, 1 \leq i \leq q-1\}$ and

$$\tilde{V} = v_{\mathbb{C}}(\langle H_1, \dots, H_{q-1} \rangle)$$

be the variety of the ideal $\langle H_1, \dots, H_{q-1} \rangle$ over \mathbb{C} , which is the projection of V on the first $q-1$ coordinates, and $\tilde{\sigma}_{\tilde{v}}$ the corresponding embeddings for $\tilde{v} \in \tilde{V}$. Hence

$$V = \{(\tilde{v}, \alpha_{\tilde{v}i}) : \tilde{v} \in \tilde{V}, 1 \leq i \leq h_q\}$$

Suppose

$$\Delta(\tilde{S}) = \det(M)^2$$

where by definition,

$$M = (\tilde{\sigma}_{\tilde{v}}(\tilde{u}))_{\tilde{v}, \tilde{u} \in \tilde{V}}$$

Define the diagonal matrix $W_i = (w_{\tilde{u}\tilde{v}})$ of size $|\tilde{V}| \times |\tilde{V}|$, where $1 \leq i \leq h_q$, and

$$w_{\tilde{v}\tilde{v}} = \alpha_{\tilde{v}i}$$

Then

$$\begin{aligned} \Delta(S) &= \det \begin{pmatrix} M & W_1 M & \cdots & W_1^{h_q} M \\ M & W_2 M & \cdots & W_2^{h_q} M \\ \vdots & \vdots & \ddots & \vdots \\ M & W_{h_q} M & \cdots & W_{h_q}^{h_q} M \end{pmatrix}^2 \\ &= \det \begin{pmatrix} 1 & W_1 & \cdots & W_1^{h_q} \\ 1 & W_2 & \cdots & W_2^{h_q} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & W_{h_q} & \cdots & W_{h_q}^{h_q} \end{pmatrix}^2 \times \det \begin{pmatrix} M & 0 & \cdots & 0 \\ 0 & M & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & M \end{pmatrix}^2 \end{aligned}$$

By reordering the rows and columns of the first matrix we have

$$\det \begin{pmatrix} 1 & W_1 & \cdots & W_1^{h_q} \\ 1 & W_2 & \cdots & W_2^{h_q} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & W_{h_q} & \cdots & W_{h_q}^{h_q} \end{pmatrix} = \prod_{\tilde{v} \in \tilde{V}} \det \begin{pmatrix} 1 & \alpha_{\tilde{v}1} & \cdots & \alpha_{\tilde{v}1}^{h_q} \\ 1 & \alpha_{\tilde{v}2} & \cdots & \alpha_{\tilde{v}2}^{h_q} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_{\tilde{v}h_q} & \cdots & \alpha_{\tilde{v}h_q}^{h_q} \end{pmatrix}$$

Each determinant in the product is, indeed, the square root of the discriminant of H_q after evaluation at \tilde{v} , whence

$$\det \begin{pmatrix} 1 & W_1 & \cdots & W_1^{h_q} \\ 1 & W_2 & \cdots & W_2^{h_q} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & W_{h_q} & \cdots & W_{h_q}^{h_q} \end{pmatrix} = \text{res}(H_1, \cdots, \text{res}(H_{q-1}, \text{res}(H_q, \frac{\partial H_q}{\partial t_q})))$$

Also, by induction hypothesis,

$$\det \begin{pmatrix} M & 0 & \cdots & 0 \\ 0 & M & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & M \end{pmatrix} = (\det(M))^{h_q}$$

$$= (\text{res}(H_1, \cdots, \text{res}(H_{q-2}, \text{res}(H_{q-1}, \frac{\partial H_1}{\partial t_1} \cdots \frac{\partial H_{q-2}}{\partial t_{q-2}} \frac{\partial H_{q-1}}{\partial t_{q-1}}))))^{h_q}$$

Since $\text{res}(f, ag) = a^{\deg(f)} \text{res}(f, g)$ for two polynomials f, g and a constant a , so

$$\Delta(S) = (\text{res}(H_1, \cdots, \text{res}(H_{q-1}, \text{res}(H_q, \frac{\partial H_1}{\partial t_1} \cdots \frac{\partial H_{q-1}}{\partial t_{q-1}} \frac{\partial H_q}{\partial t_q}))))^2$$

which proves the theorem.

4.3.5 Splitting Fields

In this section we want to talk about one of the important applications of polynomial factorization, called *splitting fields*. In abstract algebra, a splitting field of a polynomial with coefficients in a field is a smallest field extension of that field over which the polynomial splits or decomposes into linear factors. (Note that for two fields F and L , we say that L is a field extension of F , if $F \subseteq L$). More precisely,

Definition 11. A *splitting field* of a polynomial $f(x)$ with degree n over a field F is

the smallest field extension L of F over which f factors into linear factors,

$$f(x) = \prod_{i=1}^n (x - a_i) \in L[x]$$

Finding roots of polynomials has been an important problem since the time of the ancient Greeks. Some polynomials, however, have no roots such as $x^2 + 1$ over \mathbb{Q} , the rational numbers. By constructing the splitting field for such a polynomial, one can find the roots of the polynomial in the new field.

Now let us explain how to actually compute such a splitting field for a given polynomial using factorization. Let F be a field and $f(x)$ be a polynomial of degree n in the polynomial ring $F[x]$. The general process for constructing L , the splitting field of $f(x)$ over F , is to construct a sequence of fields $F = K_0, K_1, \dots, K_{r-1}, K_r = L$ such that K_i is an extension of K_{i-1} containing a new root of $f(x)$. Since $f(x)$ has at most n roots, the construction will require at most n extensions. The steps for constructing K_{i+1} from K_i for $i \geq 0$ are given as follows:

- Factorize $f(x)$ over K_i into irreducible factors $f_1(x), \dots, f_k(x)$
- Choose any nonlinear irreducible factor $g(x) = f_i(x)$
- Construct the field extension K_{i+1} of K_i as the quotient ring $K_{i+1} = K_i[x]/\langle g(x) \rangle$, where $\langle g(x) \rangle$ denotes the ideal in $K_i[x]$ generated by $g(x)$.

Constructing L can be done by repeating this process until $f(x)$ completely factors over K_r for some $r \geq 0$.

The irreducible factor f_i used in the quotient construction may be chosen arbitrarily. Although different choices of factors may lead to different subfield sequences, the resulting splitting fields will be isomorphic.

Note that since $g(x)$ is irreducible, $\langle g(x) \rangle$ is a maximal ideal and hence $K_i[x]/\langle g(x) \rangle$ is, in fact, a field. Moreover, if we let

$$\pi : K_i[x] \longrightarrow K_i[x]/\langle g(x) \rangle$$

$$h(x) \longmapsto h(x) \bmod g(x)$$

be the natural projection of the ring onto its quotient, then

$$g(\pi(x)) = \pi(g(x)) = g(x) \bmod g(x) = 0$$

so $\pi(x)$ is a root of $g(x)$ and of $f(x)$. The degree of a single extension $[K_{i+1} : K_i]$ is equal to the degree of the irreducible factor $g(x)$. The degree of the extension $[L : F]$ is given by $[K_r : K_{r-1}] \cdots [K_2 : K_1][K_1 : F]$ and is at most $n!$.

Example 5. Let $f(x) = x^2 + 1 \in F[x]$, where $F = \mathbb{Q}$, the field of rational numbers. One can show that f is irreducible over F . So we make a new field $L = K_1 = F[x]/\langle f(x) \rangle$, containing the roots of f , which are $\pi(x)$, $-\pi(x)$, where

$$\pi : F[x] \longrightarrow L = F[x]/\langle f(x) \rangle$$

$$h(x) \longmapsto h(x) \bmod f(x)$$

That is, considering f as a polynomial over L , we have

$$f(x) = (x - \pi(x))(x + \pi(x))$$

Our approach for computing splitting field is the same the above mentioned steps. In each step for finding a new field K_{i+1} , we need to use a factorization algorithm to find a non-linear irreducible factor of f over K_i . Note that for each $1 \leq i \leq r$, the field K_i is a number field, that is, an extension of F , where $F = \mathbb{Q}$ in our case. So we can use the two different representations of the number fields, introduced in previous sections, for the number field K_i . In the following we are going to apply these two representations and the corresponding factorization algorithms to find the sequence of fields K_i , and then we will compare the running time of computing such a sequence in practice, using our implementation in Magma.

Let us first talk about the univariate representation. In each step, we have a univariate representation of the number field $K_i = \mathbb{Q}[t]/\langle g(t) \rangle$ as an extension of \mathbb{Q} with degree $d = \deg(g(x))$, and a polynomial $f(x)$ over K_i , that is,

$$f(x) \in K_i[x] = \frac{\mathbb{Q}[t]}{\langle g(t) \rangle}[x]$$

so the coefficients of f are elements of the quotient field $\mathbb{Q}[t]/\langle g(t) \rangle$, which can be con-

sidered as polynomials of degree less than d . Note that

$$\frac{\mathbb{Q}[t]}{\langle g(t) \rangle} = \{h(t) \in \mathbb{Q}[t] : \deg(h) < d\}$$

Now we can choose any factorization algorithm for univariate representation, such as, the Belabas factorization algorithm [9], which is indeed our factorization algorithm, introduced in this chapter, for $q = 1$, or Trager's factorization algorithm [62], to factor the polynomial f over K_i . Assume $f_1 \in K_i[x]$ is a non-linear irreducible factor of f over K_i . Now we can define the new field K_{i+1} as an extension of K_i as below

$$K_{i+1} = \frac{K_i[x]}{\langle f_1(x) \rangle} = \frac{\frac{\mathbb{Q}[t]}{\langle g(t) \rangle}[x]}{\langle f_1(x) \rangle} \approx \frac{\mathbb{Q}[t, x]}{\langle g(t), f_1(t, x) \rangle}$$

which contains at least one root of f_1 as a polynomial over K_{i+1} . Note that any root of f_1 is indeed a root of the original polynomial f . Additionally, \approx denotes for the isomorphism of the fields.

Since the chosen factorization algorithm always requires the univariate representation of the base field over \mathbb{Q} , to factor f over K_{i+1} in the next step, if any, we need a univariate representation of K_{i+1} as a field extension of \mathbb{Q} . So we need to convert the above bivariate representation of K_{i+1} to a univariate representation over \mathbb{Q} .

Such a univariate representation can be found with the cost of computing the minimal polynomial $\alpha = \bar{t} + \lambda\bar{x} \in K_{i+1}$ over \mathbb{Q} , where \bar{t}, \bar{x} are the image of t, x in K_{i+1} and λ is a suitable element in \mathbb{Z} . So now assume $g^*(y) \in \mathbb{Q}[y]$ is the computed minimal polynomial. We have

$$K_{i+1} = \frac{K_i[x]}{\langle f_1(x) \rangle} = \frac{\frac{\mathbb{Q}[t]}{\langle g(t) \rangle}[x]}{\langle f_1(x) \rangle} \approx \frac{\mathbb{Q}[t, x]}{\langle g(t), f_1(t, x) \rangle} \approx \frac{\mathbb{Q}[y]}{\langle g^*(y) \rangle}$$

Then we are in the same situation as in step i , and we can repeat the process replacing K_i with K_{i+1} . In summary, each step of computing the splitting field contains the following computations:

- factorization over K_i
- converting bivariate representation to univariate one using minimal polynomial computation of an element of K_{i+1} over \mathbb{Q}

Now let us look at the same process, but from multivariate representation perspective. In each step we have a multivariate representation of $K_i = \mathbb{Q}[t_1, \dots, t_i] / \langle g_1, \dots, g_i \rangle$,

where for $1 \leq j \leq i$, g_j is a monic multivariate polynomial in $\mathbb{Q}[t_1, \dots, t_j]$ of degree d_j in t_j , and $[K_i : \mathbb{Q}] = \prod_{j=1}^i d_j$. Let f be a univariate polynomial over K_i ,

$$f \in K_i[x] = \frac{\mathbb{Q}[t_1, \dots, t_i]}{\langle g_1, \dots, g_i \rangle}[x]$$

Since we are in multivariate representation of the number field K_i , we can use our factorization algorithm to factor f over K_i directly. Assume that $f_1 \in K_i[x]$ is a non-linear irreducible factor of f over K_i . So now we can define the new field K_{i+1} as below,

$$K_{i+1} := \frac{K_i[x]}{\langle f_1(x) \rangle} \approx \frac{\mathbb{Q}[t_1, \dots, t_i, x]}{\langle g_1, \dots, g_i, f_1 \rangle}$$

Since in the next step, to factor f over K_{i+1} , we can apply directly any multivariate factorization algorithm, there is no need for any conversion to a new representation and we can keep K_{i+1} in the same format (but may be with different naming for consistency, taking t_{i+1} and g_{i+1} instead of x and f_1 in the representation of K_{i+1} , respectively). So there is no need for computing any minimal polynomial or any sort of evaluation, compared to univariate situation.

So far, we know how to compute the splitting fields of a given polynomial over \mathbb{Q} , using two different representations. Since the main goal of the following experiments are to compare the running time of the different factorization algorithms using different representations, in the cost of the following approaches for computing the splitting fields, we do not count the cost of computing minimal polynomials in the total cost of the splitting field computation in the univariate representation approach.

As we discussed above, the choice of different factorization algorithms in the splitting field computation can effect its total cost, which is what we are going to examine in this section. In our experiments, we consider the following four different factorization algorithms:

- **BelMR**: Our multivariate factorization algorithm, which needs a multivariate representation of the number field in each step
- **BelUR**: Belabas factorization algorithm [9], which needs a univariate representation of the number field in each step, which is basically BelMR algorithm when $q = 1$

- **Trager**: Trager’s factorization algorithm [62], which needs a univariate representation of the number field in each step
- **MagmaMR**: Magma implementation of factorization of a univariate polynomial over a number field with multivariate representation

We implemented BelMR and BelUR in Magma, and we are just using the built-in methods in Magma for Trager and MagmaMR. We will use BelMRSF, BelURSF, TragerSF, and MagmaMRSF to refer to the splitting field algorithms using the corresponding factorization algorithms, respectively. Additionally, Magma has an implementation of splitting field computation of a given univariate polynomial over \mathbb{Q} , which will be denoted by MagmaSF.

In the following table we compare the running times of the above five splitting field algorithms on different input polynomials. Some of the sample polynomials are taken from the database <http://www.mathematik.uni-kassel.de/~klueners/minimum/minimum.html> by Jürgen Klüners and Gunter Malle, while the others are just random polynomials. In the table, the degree of the input polynomials and the extension degree of the computed splitting fields over \mathbb{Q} are denoted by n and N , respectively. All computations were done using Magma 2.18 on a 2.40 GHz Intel(R) Xeon(R) processor.

Table 4.1: Splitting Fields Computation

n	N	BelMRSF	BelURSF	TragerSF	MagmaMRSF	MagmaSF
3	6	0.02	0.09	0.00	0.00	0.00
4	8	0.03	0.70	0.00	0.01	0.01
5	120	272.53	> 1	334.140	235.91	32.31
6	6	0.12	0.34	0.01	0.01	0.01
6	72	28.23	> 1h	> 1h	11.38	0.22
7	7	0.13	0.26	0.04	0.02	0.01
7	42	56.97	> 1h	243.72	31.21	0.73

As we can see from Table 4.1, BelMRSF and MagmaMRSF which are using multivariate representation for the splitting fields almost always beat the BelURSF and TragerSF which are using univariate representation for the number fields, as we expected. Note that timings given in Table 4.1 are just the cost of factorization parts, while ignoring all other costs including the minimal polynomial computation, which could even dominant the cost of factorization parts in splitting fields computation using

the univariate representation.

On the other hand, in our experiments, MagmaMRSF was always faster than BelMRSF, while both are using multivariate representation for the number fields and the same approach for computing the splitting fields. The only difference is using different factorization algorithms as the main core of the splitting fields computation in our approach. Since we do not know how MagmaMR has been implemented in Magma, so the comparison between MagmaMRSF and BelMRSF may not be fair. But it still shows that using multivariate representation in splitting field computation works much better than the univariate counterpart.

Additionally, timings showed in the last column of Table 4.1 related to MagmaSF has a huge gap with other columns in the most cases, specially when the extension degree of the splitting fields goes higher. Since again we are not aware of the method used in MagmaSF for computing splitting fields, we can not point out the reason behind such a huge gap happening in the running time.

Let us go back to the comparison between BelMRSF and BelURSF which is the main goal of the section. As we already said, since we only count the factorization cost in the splitting fields computation and also we are using the same approach for computing them, so the timings given in Table 4.1 can be used for comparing the running time of the two factorization algorithms BelMR and BelUR. At each step of splitting field computation, we factor a univariate polynomial, let say f , over the current field, let say F . BelMRSF uses BelMR as the factorization algorithm to factor f over F in multivariate representation, while BelURSF uses the BelUR to factor f over the same field but in univariate representation. Note that since we change the representation of F , so we need to find the image of f in the new representation; hence f is not necessarily the same polynomial in both cases. So if we want to compare the running time of BelMR and BelUR, we need to choose one number field in two representations and take one polynomial over one of them and compute the image of the polynomial over the other field. This is, Indeed, what is happening at each step of the splitting fields computation.

So the timings given in the columns of Table 4.1 related to BelMRSF and BelURSF are also meaningful for comparison of the factorization algorithms BelMR and BelUR. So again as we can see from the table, our experiments show that BelMR works better than BelUR. Of course, there could be some situations for which BelUR factors the given

polynomial faster than BelMR, since the size of the input polynomials and the generators play an important role in the complexity of the two main steps, modular factorization and factor combination, of the factorization algorithms BelMR and BelMR. Based on our experiments, in general, we can not claim which one can always beat the other, as we think this is not true at all. But up to some application of the polynomial factorization, such as splitting fields computation, our experiments show that using multivariate representation for the number field and the multivariate version of Belabas factorization algorithm [9] are better than the univariate counterpart.

Bibliography

- [1] J. V. A. Leykin and Z. A. Higher-order deflation for polynomial systems with isolated singular solutions. *Math. and its Appl.*, 146:79–97, 2008.
- [2] J. V. A. Leykin and A. Zhao. Newtons method with deflation for isolated singularities of polynomial systems. *TCS*, 359(1-3):111–122, 2006.
- [3] C. J. Accettella, G. M. D. Corso, and G. Manzini. Inversion of two level circulant matrices over \mathbb{Z}_p . *Lin. Alg. Appl.*, 366:5–23, 2003.
- [4] M. Ajitai. The shortest vector problem in l_2 is NP-hard for randomized reductions. *Electronic Colloquium on Computational Complexity*, TR97-047, 1997.
- [5] M. E. Alonso, E. Becker, M.-F. Roy, and T. Wörmann. Zeroes, multiplicities and idempotents for zerodimensional systems. In *MEGA '94*, volume 142 of *Progress in Mathematics*, pages 1–15. Birkhäuser, 1996.
- [6] P. Aubry, D. Lazard, and M. M. Maza. On the theories of triangular sets. *JSC*, 28(1,2):45–124, 1999.
- [7] P. Aubry and A. Valibouze. Using Galois ideals for computing relative resolvents. *JSC*, 30(6):635–651, 2000.
- [8] Z. Z. B. H. Dayton. Computing the multiplicity structure in solving polynomial systems. *ISSAC 05*, pages 116–123, 2005.
- [9] K. Belabas. A relative van Hoeij algorithm over number fields. *Symb. Comp.*, 37:641–668, 2004.
- [10] K. Belabas, M. van Hoeij, J. Kluners, and A. Steel. Factoring polynomials over global fields. *arXiv:math/0409510v1*, 2004.

- [11] E. Berberich, P. Emeliyanenko, and M. Sagraloff. An elimination method for solving bivariate polynomial systems: Eliminating the usual drawbacks. In *ALLENEX*, pages 35–47. SIAM, 2011.
- [12] E. Berlekamp. Factoring polynomials over large finite fields. *Math. Comp.*, 24:713735, 1970.
- [13] A. Bostan, C.-P. Jeannerod, and É. Schost. Solving structured linear systems with large displacement rank. *Theor. Comput. Sci.*, 407(1-3):155–181, 2008.
- [14] F. Boulier, F. Lemaire, and M. Moreno Maza. Pardi! In *ISSAC'01*, pages 38–47. ACM, 2001.
- [15] Y. Bouzidi. *Résolution de systèmes bivariés et topologie de courbes planes*. PhD thesis, Université de Lorraine, 2014.
- [16] Y. Bouzidi, S. Lazard, M. Pouget, and F. Rouillier. Separating linear forms for bivariate systems. In *ISSAC '13*, pages 117–124. ACM, 2013.
- [17] R. P. Brent and H. T. Kung. Fast algorithms for manipulating formal power series. *J. ACM*, 25(4):581–595, 1978.
- [18] L. Cerlienco and M. Mureddu. From algebraic sets to monomial linear bases by means of combinatorial algorithms. *Discrete Math.*, 139:73–87, 1995.
- [19] J. Cheng, S. Lazard, L. M. Peñaranda, M. Pouget, F. Rouillier, and E. P. Tsigaridas. On the topology of real algebraic plane curves. *Mathematics in Computer Science*, 4(1):113–137, 2010.
- [20] H. Cohen. *A course in computational algebraic number theory*. Springer-Verlag, Berlin Heidelberg, third edition, 1996.
- [21] D. A. Cox, J. B. Little, and D. O'Shea. *Using algebraic geometry*, volume 185. Springer, New-York, 1998.
- [22] X. Dahan, X. Jin, M. Moreno Maza, and É. Schost. Change of order for regular chains in positive dimension. *Theor. Comput. Sci.*, 392(1–3):37–65, 2008.
- [23] X. Dahan, M. Moreno Maza, É. Schost, W. Wu, and Y. Xie. Lifting techniques for triangular decompositions. In *ISSAC'05*, pages 108–115. ACM, 2005.
- [24] X. Dahan and É. Schost. Sharp estimates for triangular sets. In *ISSAC*, pages 103–110. ACM, 2004.

- [25] D. Diocnos, I. Emiris, and E. Tsigaridas. On the asymptotic and practical complexity of solving bivariate systems over the reals. *JSC*, 44(7):818–835, 2009.
- [26] M. El Kahoui. Topology of real algebraic space curves. *J. Symb. Comput.*, 43(4):235–258, 2008.
- [27] P. Emeliyanenko and M. Sagraloff. On the complexity of solving a bivariate polynomial system. In *ISSAC'12*. ACM, 2012.
- [28] I. Z. Emiris and E. P. Tsigaridas. Real solving of bivariate polynomial systems. In *CASC*, pages 150–161. Springer, 2005.
- [29] J. von zur Gathen and J. Gerhard. Fast algorithms for Taylor shifts and certain difference equations. In *ISSAC*, pages 40–47. ACM, 1997.
- [30] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, 1999.
- [31] P. Gaudry and É. Schost. Construction of secure random curves of genus 2 over prime fields. In *Eurocrypt'04*, volume 3027 of *LNCS*, pages 239–256. Springer, 2004.
- [32] P. Gianni and T. Mora. Algebraic solution of systems of polynomial equations using Groebner bases. In *Applied algebra, algebraic algorithms and error-correcting codes (Menorca, 1987)*, volume 356 of *Lecture Notes in Comput. Sci.*, pages 247–257. Springer, 1989.
- [33] M. Giusti, G. Lecerf, and B. Salvy. A Gröbner free alternative for polynomial system solving. *J. Comp.*, 17(1):154–211, 2001.
- [34] M. W. Hirst HP. Bounding the roots of polynomials. *Coll Math*, J 28(4), 1997.
- [35] X. Huang and V. Pan. Fast rectangular matrix multiplication and applications. *J. Complexity*, 14(2):257–299, 1998.
- [36] K. S. Kedlaya and C. Umans. Fast polynomial factorization and modular composition. *SICOMP*, 40(6):1767–1802, 2011.
- [37] T. Krick, L. M. Pardo, and M. Sombra. Sharp estimates for the arithmetic Nullstellensatz. *Duke Math. J.*, 109:521–598, 2001.
- [38] D. Lazard. Ideal bases and primary decomposition: case of two variables. *J. Symbolic Comput.*, 1(3):261–270, 1985.

- [39] R. Leberton, E. Mehrabi, and É. Schost. On the complexity of solving bivariate systems: the case of non-singular solutions. In *ISSAC*. ACM, 2013.
- [40] G. Lecerf. Quadratic newton iteration for systems with multiplicity. *Comp. Math.*, 2:247–293, 2002.
- [41] A. Lenstra. Lattices and factorization of polynomials over algebraic number fields. *LNCS*, 144:3239, 1982.
- [42] A. Lenstra, H. Lenstra, and L. Lovasz. Factoring polynomials with rational coefficients. *Math. Ann.*, 261(4):515–534, 1982.
- [43] X. Li, M. Moreno Maza, and W. Pan. Computations modulo regular chains. In *ISSAC*, pages 239–246. ACM, 2009.
- [44] X. Li, M. Moreno Maza, and É. Schost. Fast arithmetic for triangular sets: from theory to practice. *JSC*, 44:891–907, 2009.
- [45] T. M. M. G. Marinari and H. Moller. Groebner duality and multiplicities in polynomial system solving. *ISSAC '95*, pages 167–179, 1995.
- [46] D. S. J. M. R. Garey. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co. New York, 1979.
- [47] F. Macaulay. *The algebraic theory of modular systems*. Cambridge University Press, 1916.
- [48] M. Moreno Maza. On triangular decompositions of algebraic varieties. Technical Report 4/99, NAG, UK, Presented at the MEGA-2000 Conference, Bath, UK, 1999. <http://www.csd.uwo.ca/~moreno>.
- [49] B. Mourrain. Isolated points, duality and residues. *Math. and its Appl.*, pages 469–493, 1997.
- [50] C. Pascal and E. Schost. Change of order for bivariate triangular sets. In *ISSAC'06*, pages 277–284. ACM, 2006.
- [51] L. R. P.J. Weinberger. Factoring polynomials over algebraic number fields. *ACM*, 2 Issue 4:335–350, 1976.
- [52] S. Pope and A. Szanto. Nearest multivariate system with given root multiplicities. *SCJ*, 44(6):606–625, 2009.

- [53] A. Poteaux and É. Schost. Modular composition modulo triangular sets and applications. *Comput. Comp.* (to appear).
- [54] A. Poteaux and É. Schost. On the complexity of computing with zero-dimensional triangular sets. *JSC* (to appear).
- [55] D. Reischert. Asymptotically fast computation of subresultants. In *ISSAC'97*, pages 233–240. ACM, 1997.
- [56] F. Rouillier. Solving zero-dimensional systems through the rational univariate representation. *Appl. Algebra Engrg. Comm. Comput.*, 9(5):433–461, 1999.
- [57] F. Rouillier. On solving systems of bivariate polynomials. In *ICMS*, volume 6327 of *LNCS*, pages 100–104. Springer, 2010.
- [58] A. Schönhage. The fundamental theorem of algebra in terms of computational complexity. Technical report, Univ. Tübingen, 1982.
- [59] É. Schost. Computing parametric geometric resolutions. *Appl. Algebra Engrg. Comm. Comput.*, 13(5):349–393, 2003.
- [60] V. Shoup. A new polynomial factorization algorithm and its implementation. *JSC*, 20(4):363–397, 1995.
- [61] S. W. T. Ojika and T. Mitsui. Deflation algorithm for multiple roots of a system of nonlinear equations. *Math. An. and Appls.*, 96(2):463479, 1983.
- [62] B. Trager. Algebraic factoring and rational function intergration. In *SYMSAC*, pages 219–226. ACM, 1976.
- [63] M. van Hoeij. Factoring polynomials and the knapsack problem. *Number Theory*, 95(2):167189, 2002.
- [64] M. van Hoeij and A. Novocin. Gradual sub-lattice reduction and a new complexity for factoring polynomials. *LATIN*, page 539553, 2010.
- [65] V. Vassilevska Williams. Multiplying matrices faster than Coppersmith-Winograd. *STOC '12*, pages 887–898. ACM, 2012.
- [66] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, Cambridge, second edition, 2003.
- [67] T. Weston. *Algebraic Number Theory*. Lecture Note given at Harvard, 1999.

- [68] X. Wu and L. Zhi. Computing the multiplicity structure from geometric involutive form. *ISSAC'08*, pages 325–332, 2008.
- [69] H. Zassenhaus. On Hensel factorization. *Journal of Number Theory*, pages 291–311, 1969.

Curriculum Vitae

Name:

Esmaeil Mehrabi

**Education and
Degrees:**

The University of Western Ontario
London, Ontario, Canada

Ph.D. in Computer Science, 2010-2014

Sharif University of Technology
Tehran, Iran

M.Sc. in Mathematics, 2005-2007

Kharazmi University
Tehran, Iran

B.Sc. in Mathematics, 2001-2005