Solving Multivariate Diophantine Equations and Their Role In Multivariate Polynomial Factorization.

by

YUSUF BARIS TUNCER

M.Sc., Tuebingen University, 2002
B.Sc., Bogazici University, 1999

Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of
Doctor of Philosophy

in the
Department of Mathematics
Faculty of Science

Approval

Name:	**YUSUF BARIS TUNCER**
Degree:	**Doctor of Philosophy (Mathematics)**
Title:	***Solving Multivariate Diophantine Equations and Their Role In Multivariate Polynomial Factorization.***
Examining Committee:	**Chair:** Tom Archibald Professor

Michael Monagan
Senior Supervisor
Professor _____

Marni Mishna
Supervisor
Associate Professor _____

Tamon Stephen
Internal Examiner
Associate Professor _____

Keith Geddes
External Examiner
Professor Emeritus
David R. Cheriton School of Computer Science
University of Waterloo _____

Date Defended: 14 August 2017

Abstract

Multivariate polynomial factorization over integers via multivariate Hensel lifting (MHL) is one of the central areas of research in computer algebra.

Most computer algebra platforms, such as Maple, Magma and Singular, implement Wang's incremental design of MHL which lifts the factors one variable at a time and one degree at a time. At each step MHL must solve a multivariate diophantine problem (MDP) which Wang solves using the same idea; lifting the solutions one variable and one degree at a time.

Although this performs well when the evaluation points are mostly zero, it performs poorly when there are many non-zero evaluation points as the number of MDP problems to be solved can be exponential in the number of variables. In this thesis we introduce a new non-recursive solution to the MDP which explicitly exploits the sparsity of the solutions to the MDP.

We use sparse interpolation techniques and exploit the fact that at each step of MHL, the solutions to MDP's are structurally related. We design a probabilistic sparse Hensel lifting algorithm (MTSHL) and give a complete average case complexity analysis for it.

We describe a series of experiments based on our implementation of MTSHL, compare its efficiency with Wang's algorithm, and show that MTSHL performs better for many practical applications. We also show that previous probabilistic approaches proposed for MHL as an alternative to Wang's algorithm are not practical whereas MTSHL is practical and the running time is predictable.

Keywords: Multivariate Polynomial Factorization; Multivariate Hensel Lifting; Multivariate Polynomial Diophantine problem; Sparse Interpolation; Sparse Hensel Lifting.

Dedication

This thesis is dedicated to the loving memory of my father, Ali Riza Tuncer.

Acknowledgements

I would like to take immense pleasure in expressing my sincere and deep sense of gratitude to my supervisor Michael Monagan for his invaluable guidance, creative suggestions and generous support during my PhD studies at SFU.

I also thank the members of my thesis committee Marni Mishna, Tamon Stephen, Keith Geddes for reading my thesis and Tom Archibald for being present at my defence session.

I extend my sincere thanks to Roman Pearce for his generous support and coding help, John Hebron for his CECM support and John Ogilvie for his friendship.

I also take this opportunity to thank my high school mathematics teacher Suat Gencel for being such a fantastic teacher and for being my inspiration.

I would like to extend my warmest gratitude to my close friends Gulsen Meral Yildirim and Burcak Dogan for being caring and supportive at difficult times.

I owe a lot to my family, Gulderen Tuncer, Melek Burcu Tuncer Karabina and Koray Karabina who encouraged and helped me at every stage of my PhD research.

Finally I am deeply thankful to Gokcen Yildiz. Without her psychological support this dissertation would not have been possible.

Table of Contents

List of Tables

a

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Evolution of Multivariate Polynomial Factorization

Multivariate polynomial factorization arises in many applications in symbolic computation. It is a significant sub-problem when computing primary decompositions of ideals, simplification of large formulae, symbolic integration and solving systems of polynomial equations. It also plays a significant role in other fields such as commutative algebra, algebraic geometry, cryptography and algebraic coding theory.

Polynomial factorization has a long history; and as Kaltofen indicates [Kal85], it is a success story. According to Knuth [Knu81], its beginnings can be traced back to 1707 Isaac Newton's *Arithmetica Universalis* and to the astronomer Schubert who in 1793 presented a finite step algorithm to compute the factors of a univariate polynomial with integer coefficients. Kronecker rediscovered Schubert's method in 1882 and also gave algorithms for factoring polynomials with two or more variables or with coefficients in algebraic extensions [Kro95].

In 1967 Berlekamp made an ingenious proposal for factoring univariate polynomials in $\mathbb{Z}_p[x]$ in which p is prime in [Ber67]. It is polynomial time in the degree of the polynomial unlike all previous algorithms which are exponential in the degree. Two years later Zassenhaus showed in [Zas69] how to apply the "Hensel lemma" to lift a factorization modulo a prime p to a factorization modulo p^{2k} in k iterations, provided that the integral polynomial is square-free and remains square-free modulo p. For p^{2k} sufficiently large, the integral factors can then be found from the factorization modulo p^{2k}.

In 1971 and 1975, Musser [Mus71, Mus75] formalized this idea, and described an outline for the multivariate factorization based on Zassenhaus's observations. Wang, in collaboration with Rothschild [WR75], extended the ideas of Musser to obtain a multivariate factorization algorithm for integral polynomials. Subsequently, Wang significantly improved this multivariate factorization algorithm [Wan78] by developing an incremental design of multivariate Hensel lifting (MHL) and designing a coefficient correction algorithm (LCC). Today

many modern computer algebra systems such as Maple, Magma and Singular use Wang's incremental design of multivariate Hensel lifting (MHL) to factor multivariate polynomials over integers. The details can be found in Chapters 6 and 8 of the Geddes, Czapor and Labahn text [GCL92]. Only this textbook has the details. The Maple, Magma and Singular implementations are all based on [GCL92].

As long as the necessary conditions are satisfied, Wang's design of MHL invariably returns the true unique lifting. In many practical applications the polynomials are sparse. Although Wang's method performs significantly better than the previous algorithm that he developed with Rothschild in [WR75], it does not explicitly take sparsity into account. Zippel's sparse interpolation [Zip90] was the first probabilistic method aimed to take sparsity into account. Based on sparse interpolation and multivariate Newton's iteration, Zippel then introduced a sparse Hensel lifting (ZSHL) algorithm in [Zip81], which uses an MHL organization different from that of Wang. In this method Zippel suggested that one use Wang's LCC.

A new approach for sparse Hensel lifting for the sparse case was proposed by Kaltofen (KSHL) in [Kal85]. Kaltofen's method is based on Wang's incremental design of MHL but it uses a LCC different from Wang's LCC and offers a distinct solution to the multivariate diophantine problem (MDP) that appears as a significant sub-problem in Wang's design of MHL.

This thesis presents a new sparse Hensel lifting algorithm proposed by Monagan and Tuncer (MTSHL) that appeared in [MT16-2]. It outperforms previous algorithms. It is based also on Wang's incremental design of MHL and LCC and offers a solution to the MDP different from the solutions offered by Zippel and Kaltofen. To solve the MDP problem appearing in MHL, the technique proposed by Monagan and Tuncer uses and improves the sparse interpolation techniques developed by Zippel.

MTSHL algorithm performs much better than the previous two approaches ZSHL and KSHL. Also for many cases it is significantly faster than Wang's algorithm.

What do we mean by performing better? According to the best of our knowledge the average complexity analysis of the approaches by Zippel and Kaltofen were not done likely because of the complicated nature of the algorithms and also because the term *sparse* is unclear in a mathematical sense. One can make worst case complexity estimates; these estimates are not based on sparseness. The papers do not mention an implementation and we are unaware of any implementation of these algorithms in the literature. We have implemented these algorithms in Maple and optimized them. We recognized that they do not behave as the authors expected. For many sparse examples Wang's proposal is quicker then these two algorithms. This effect led us to question the assumptions upon which ZSHL and KSHL are based. According to our experiments these proposals bring new hidden costs that are not negligible. We will discuss these details and give concrete examples to test

how these approaches work, to find their weak points and to compare their efficiency with our and Wang's algorithms to convince the reader. We note again that Wang's second algorithm [Wan78] performs better for the sparse case than the previous proposal by Wang and Rothschild [WR75] and it is the standard algorithm. If one aims to develop a new approach for a multivariate factorization for the sparse case, it should be compared with Wang's second algorithm [Wan78]. We compute the exact complexity analysis of MTSHL. The complicated nature of the algorithms make the exact complexity analysis tedious and difficult. Even for Wang's algorithm in [Wan78] an exact satisfactory complexity analysis was not done because of its highly recursive nature. For an attempt in this direction see [Wit04]. This analysis does not take into account the decrease in the number of terms of a polynomial after evaluation and therefore fails to give insight for the sparse cases. As we claim that for all cases our MTSHL algorithm performs much better than the previous two approaches ZSHL and KSHL and also Wang's algorithm for many cases, we are in a position not only to implement and to show that MTSHL is quicker for some data but also to make the term sparse mathematically precise and to compute the average complexity and to verify our claims by randomly generated sparse polynomials and families of polynomials with sparse factors. We do not give an exact complexity analysis of ZSHL, KSHL or Wang's approaches. After we implemented our approach, we recognized that for the sparse cases, and when the evaluation points chosen by the algorithm are not all zero, MTSHL performs better, so we focused on a detailed complexity analysis of MTSHL. This was not an easy task and required almost half of the thesis, namely Chapters 5 and 6.

For the examples in the thesis and the implementation of the algorithms presented in the thesis we used Maple, with two key parts implemented in C. This raises an obvious question, is the gain obtained by MTSHL because of an efficient implementation of MTSHL, but poor implementation of Wang's algorithm? Regarding other computer algebra platforms, three computer algebra systems with multivariate factorization are Magma, Singular and Maple. For multivariate factorization they all use Wang's MHL following the representation in [GCL92]. Table 1.1 and 1.2 below shows that the multivariate factorization timings of Maple are quicker then Magma and Singular.

1.2 The outline of the thesis

Consider the symmetric Toeplitz matrix T_n below:

$$T_n = \begin{pmatrix} x_1 & x_2 & \cdots & x_{n-1} & x_n \\ x_2 & x_1 & \cdots & x_{n-2} & x_{n-1} \\ & \ddots & \ddots & \ddots & \\ x_{n-1} & x_{n-2} & \cdots & x_1 & x_2 \\ x_n & x_{n-1} & \cdots & x_2 & x_1 \end{pmatrix}$$

n	$\#d_n$	sizes of factors	Maple	# MDP	MDP%	Magma	Singular
7	427	30, 56	0.035	161	30%	0.01	0.02
8	1628	167, 167	0.065	383	43%	0.04	0.05
9	6090	153, 294	0.166	1034	73%	0.10	0.28
10	23797	931, 931	0.610	2338	76%	0.89	1.77
11	90296	849, 1730	2.570	6508	74%	1.96	8.01
12	350726	5579, 5579	19.45	15902	80%	72.17	84.04
13	1338076	4983, 10611	84.08	45094	84%	181.0	607.99
14	5165957	34937, 34937	637.8	103591	77%	6039.0	20333.45
15	19732508	30458, 66684	4153.2	286979	84%	12899.2	–

Table 1.1: Factorization timings in CPU seconds for factoring d_n the determinant of the n by n Toeplitz matrix T_n evaluated at $x_n = 1$.

Let d_n be the determinant of T_n, a polynomial in $\mathbb{Z}[x_1, \ldots, x_n]$. The data in Table 1.1 for factoring d_n compares Maple 2017 with Magma 2.22–5 and Singular 3–1–6. Column $\#d_n$ is the number of terms in the determinant when expanded and the next column gives the number of terms in the two factors. Column # MDP shows the number of calls (including recursive calls) to Maple's MDP algorithm and the column MDP% shows the percentage of time in Hensel lifting spent solving MDPs. Notice that over 75% of the time is in solving MDP's. We also saw examples where over 90% of the time was in MDP. This was our initial motivation for studying algorithms for solving MDP.

The determinant $d_n = \det(T_n)$ is homogeneous. Since the difficulty of the factorization of $\det(T_n)$ depends on which variable is used to de-homogenize the determinant, we fixed $x_n = 1$ for all three systems. That is, for $\det(T_n)$ we time the factorization of $d_n(x_1, x_2, \ldots, x_{n-1}, 1)$.

The de-homogenization of d_{14} is of total degree 14 which has 2 factors. Both of them are in 13 variables of total degree 7 and each has 34937 terms for a density ratio 0.450684. The *density ratio* of a polynomial in n variables of total degree d with T non-zero terms is given by the ratio $T/\binom{n+d}{d}$.

The de-homogenization of d_{15} is of total degree 15 which has 2 factors. The first factor computed is in 14 variables of total degree 8, has 66684 terms and density ratio 0.208537. The second factor is in 14 variables of total degree 7, has 30458 terms and density ratio 0.261937.

Factoring d_n is a challenging problem. We note that these factors are atypical sparse examples. Their total degree is small and less than their total number of variables, which is contrary to our intuition of sparse examples. They are huge and not too sparse; they can be considered as dense in practical terms. Our natural expectation in this case is that Wang's approach is hence preferable to sparse approaches.

n	$\#d_n$	sizes of factors	Maple	# MDP	MDP%	Magma	Singular
7	246	7,924	0.045	330	90%	0.01	0.02
8	810	8,8,20,86	0.059	218	46%	0.07	0.06
9	2704	9,45,1005	0.225	1810	74%	0.74	0.24
10	7492	10,10,715,715	0.853	1284	62%	8.44	2.02
11	32066	11,184756	7.160	75582	91%	104.3	11.39
12	86500	12,12,42,78,78,621	19.76	1884	76%	7575.1	30.27
13	400024	13, 2704156	263.4	1790701	92%	30871.90	NA
14	1366500	14,14,27132,27132	1664.4	50381	77%	$> 10^6$	288463.17
15	4614524	15,120,3060,303645	18432.	477882	82%	–	NA

Table 1.2: Factorization data and timings in CPU seconds for factoring d_n the determinant of the n by n Cyclic matrix C_n evaluated at $x_n = 1$.

As can be seen from Table 1.1, if we factor d_{14} and d_{15} using Maple that uses Wang's algorithm for multivariate factorization, the calculation will take 637 s. and 4153 s. respectively. MTSHL factors d_{14} and d_{15} in 250s. and 1650 s. resp. 60% and 40% resp. spent on solving diophantine equations. (Timings were obtained on an Intel Core i5–4670 CPU running at 3.40GHz.)

Next, consider the Cyclic matrix C_n below:

$$C_n = \begin{pmatrix} x_1 & x_2 & \dots & x_{n-1} & x_n \\ x_n & x_1 & \dots & x_{n-2} & x_{n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_3 & x_4 & \dots & x_1 & x_2 \\ x_2 & x_3 & \dots & x_n & x_1 \end{pmatrix}$$

The data in Table 1.2 for factoring the determinant of C_n compares Maple 2017 with Magma 2.22–5 and Singular 3–1–6.

The determinant $d_n = \det(C_n)$ is also homogeneous.

The de-homogenization of d_{14} is of total degree 14 and has 4 factors. 2 factors are in 13 variables of total degree 1, has 14 terms and density ratio 1; the other 2 factors are in 13 variables of total degree 6, has 27132 terms and density ratio 1.

The de-homogenization of d_{15} is of total degree 15 and has 4 factors. The first 3 factors are of total degree 1, 2 and 4 with density ratio 1. The last factor is of total degree 8 with density ratio 0.949573.

For these full dense examples, as can be seen from Table 1.2, if we factor d_{14} and d_{15} using Maple that uses Wang's algorithm for multivariate factorization, the calculation took 1664 s. and 18432 s. respectively. MTSHL factors d_{14} and d_{15} in 523 s. and 7496 s. resp. 0.16% and 0.06% resp. spent on solving diophantine equations.

To make this thesis accessible, we begin with a brief introduction to show how Maple, Singular, Magma or other computer algebra platforms factor such huge multivariate polynomials. We try to keep the requirements to read the thesis to a minimum and follow a readable approach by giving non-trivial examples; these examples are important, since as we will see in Chapter 4, without considering some concrete problems the complexity estimates can be misleading. There can be hidden costs that are thought to be cheap but that in practice are not. We have a computational problem and it is a natural wish of the reader to see some live computations. We also supply pseudo-code so that a readers can generate their own examples to convince themselves.

1.2.1 Various approaches for sparse multivariate factorization

As mentioned above, the most efficient multivariate polynomial factorization algorithm is based on multivariate Hensel lifting (MHL), which is a natural generalization of univariate Hensel Lifting. To begin we give an overview of the univariate Hensel lifting so that the reader can see how the ideas behind univariate Hensel lifting naturally generalize to the multivariate case. Leading coefficient correction (LCC) plays an important role in this process; we will illustrate LCC with concrete examples.

In Chapter 2, we explain in detail the steps of the multivariate factorization developed by Wang. We will see that LCC for the multivariate case and MHL play an important role in multivariate factorization. We then observe that the MDP problem naturally appears as a sub-problem during incremental design of MHL and we define the multivariate diophantine problem (MDP) in detail.

Wang's design of MHL is incremental; that is, it recovers the variables one at a time. In a typical application of Wang's multivariate factorization algorithm, one applies LCC before the j^{th} lifting step of the MHL. The total complexity of the factorization can then be obtained as the cost of LCC + the cost of univariate factorization over $\mathbb{Z}[x_1]$+ the cost of MHL. One can readily construct examples in which LCC or factorization in $\mathbb{Z}[x_1]$ dominates the total complexity. But such examples are atypical in practice. Generally among these, the dominating cost is the total cost of MHL; moreover, within MHL solving the MDP dominates the cost (see section 6.6). So, in Chapter 3 we investigate the ways of solving MDP. We first discuss the natural attempts to solve the MDP via the use of Groebner bases and pseudo remainder sequences and then show that these methods are inefficient. Next we will describe Wang's solution of the MDP. As we have discussed, although Wang's approach is much better than the previous approaches, its efficiency decreases significantly when the factors to be computed are sparse but their number of terms are not that small and the evaluation points that the algorithm chooses are non-zero. This leads us to probabilistic approaches to solve the MDP to exploit the sparsity of the factors. We show that interpolation is an option to solve the MDP. It may seem that solution via interpolation is a natural solution to the MDP. But this idea is false. Solution to the MDP via interpolation is not deterministic

but probabilistic and the underlying probability is not immediate to see. Some evaluation points of the interpolation algorithm will turn out to be *unlucky*, i.e. useless and in fact we investigate the distribution of unlucky points. If the factors to be computed are sparse then the solutions to the MDP are also sparse. Next we will show how to use Zippel's sparse interpolation idea to solve the MDP. Although this approach seems promising, it comes with an additional evaluation cost that dominates as the number of terms in the factors to be computed increases. One way to decrease the evaluation cost that we explore in Chapter 3 is to decrease the number of evaluations required by evaluating to bivariate images instead of univariate images and solve bivariate diophantine equations instead of univariate ones. In the sparse interpolation one has to evaluate multivariate polynomials at many points. The final section of Chapter 3 describes the evaluation method that we have used for subsequent evaluation of a polynomial in sparse interpolation, which brings a significant advantage over classical evaluation methods.

As noted above, sparse Hensel lifting was first introduced by Zippel [Zip81] and then improved by Kaltofen [Kal85]. In Chapter 4, we inspect the general SHL ideas and discuss ZSHL and KSHL in detail. At the j^{th} step of the incremental design of MHL one computes the Taylor expansion of factors at a random point in a for loop. At the i^{th} step of the for loop one computes the i^{th} Taylor coefficient of the factors by solving an MDP problem. Let us denote by f_j one factor to be computed at the j^{th} step and by α_j the random point chosen by the algorithm. So at the j^{th} step the Taylor expansion $f_j = \sum_{i=0}^{\deg_{x_j} f_j} f_{ji}(x_j - \alpha_j)^i$ is computed. The coefficients f_{ji} are polynomials in the variables $x_1, \ldots, x_{j-1}$. We make a key observation that $\text{Supp}(f_{j,i}) \subseteq \text{Supp}(f_{j,i-1})$ with large probability; by $\text{Supp}(f_{j,i})$ we mean the set of non-zero monomials in polynomial f_{ji}. Therefore, with high probability, the number of terms of the polynomials to be computed by solving MDP decreases; so that the sparse approach to solve MDP's proposed in Chapter 3 becomes more effective as i increases, even for cases in which the factor to be computed is dense. Moreover, while we compute $f_{j,i}$ via solving the MDP, a sparse technique to solve the MDP can take $f_{j,i-1}$ as a skeleton. This observation means that solving a MDP reduces to solving linear equations modulo p. Moreover, the linear systems are Vandermonde systems which can be solved very efficiently. (See [Zip90].) In Chapter 4, we make these statements more precise, propose and give the pseudo-code of MTSHL. Finally we reconsider the case in which the prime p chosen by the algorithm is too small to compute the coefficients; in this case p-adic lifting is needed. The current approach is the one described in [GCL92]. In this case, at the j^{th} step of MHL, one projects down and computes the solutions to the MDP first in $\mathbb{Z}_p[x_1]$ then lifts them to $\mathbb{Z}_{p^l}[x_1]$ and then by staying in $\mathbb{Z}_{p^l}$ arithmetic iterates the solutions to $\mathbb{Z}_{p^l}[x_1, \ldots, x_{j-1}]$. At the end of Chapter 4 we show that a sparse MDP solver developed in Chapter 3 renders an improved option. Suppose one factor $f \in \mathbb{Z}[x_1, \ldots, x_n]$ has a p-adic representation $f = \sum_{k=0}^{l} f_k p^k$. We show that in this case also with high probability $\text{Supp}(f_k) \subseteq \text{Supp}(f_{k-1})$ if p is chosen randomly. Therefore we propose first computing the

factorization in $\mathbb{Z}_p[x_1, \ldots, x_n]$ by doing all arithmetic mod p where p is a machine prime (e.g 63 bits on a 64 bit computer), i.e. run the entire Hensel lifting modulo a machine prime. Then lift the solution to $\mathbb{Z}_{p^l}[x_1, \ldots, x_n]$ by computing f_k, again by solving each MDP appear in the lifting process using the sparse interpolation developed. By this approach we stay in modulo p to recover integer coefficients. We confirm with experimental data that this approach brings a significant gain over the previous approach.

Our solution to MDP via sparse interpolation was first presented as a poster at ISSAC 2015; then MTSHL was first presented in detail in CASC 2016 and published in the Proceedings of CASC 2016 [MT16-2]. The final chapter considering the p^l case will be submitted soon.

1.2.2 Efficiency and analysis of MTSHL

MTSHL is a probabilistic approach, like ZSHL and KSHL. To investigate the efficiency of the MTSHL algorithm we must consider the probabilistic assumptions made in the design of MTSHL. Given two multivariate polynomials in $\mathbb{F}_q[x_1, \ldots, x_n]$ in which $\mathbb{F}_q$ is a finite field of q elements and a random evaluation point $\gamma = (\gamma_2, \ldots, \gamma_n) \in \mathbb{F}_q^{n-1}$, the solution to the MDP proposed by MTSHL requires that $\gcd(f(x_1, \gamma), g(x_1, \gamma)) = 1$. This condition is true with high probability if q is big enough. A point γ which does not satisfy this condition is called an *unlucky* point. Let X be a random variable that counts the unlucky points for a given multivariate polynomial pair (f, g). In Chapter 5, we consider the distribution of unlucky points and calculate the expected value $\mathrm{E}[X]$ of X. It was a surprise to us that $\mathrm{E}[X] = 1$ independent of the degrees of f and g. One should also consider how smooth is the distribution. To see it we must calculate the variance $\mathrm{Var}[X]$ of X, which is difficult to compute. To attain this goal we must investigate techniques to compute $\mathrm{Var}[X]$. Chapter 5 shows also a pertinent result. Suppose in the discussion above, $n = 2$ and we change the field $\mathbb{F}_q$ with $\mathbb{Z}_m$ in which m is a composite number and X is the random variable which counts the number of roots of $f(x)$ in $\mathbb{Z}_m$. This case is interesting, because as we see in Chapter 5, in this case too, the expected value $\mathrm{E}[X] = 1$, but we have not a small variance as one might expect because of the zero divisors. The existence of such zero divisors makes the computation of $\mathrm{Var}[X]$ difficult, but the techniques we developed in Chapter 5 allow us to compute the variance in this case and to give an explanation to sequence A006579 in the On-line Encyclopedia of Integer Sequences. Most of the work explained in Chapter 5 was presented at FPSAC 2016 and published in DMTCS, see [MT16-1].

As a final step we give a precise complexity analysis of MTSHL. As the inner structure of the algorithm is complicated and as we claim that MTSHL behaves well for computing sparse factors we must be careful in our complexity analysis. As Chapter 2 makes clear, the incremental design of MHL is based on the evaluation of one variable of the input polynomial to be factored at a random point. Each time, the number of terms in the polynomial hence decreases after evaluation. To have an effective estimate of the complexity we must

estimate the number of terms of a polynomial after each evaluation. For this purpose, we will make the term sparse more precise by defining the density ratio of a polynomial. Let $f \in \mathbb{Z}_p[x_1, \ldots, x_n]$ be a randomly chosen multivariate polynomial in which p is a large prime. Suppose that the total degree of f is d, that f has T_f non-zero terms and that $\gamma \in \mathbb{Z}_p$ is chosen at random. Let X be the random variable that counts the number of terms of $f(x_1, \ldots, x_{n-1}, \gamma)$. We first compute the expected value $\mathrm{E}[X]$ of X and then give a good estimate for it. We also discuss the variance $\mathrm{Var}[X]$ of X. Each time we make an observation we confirm our theoretical estimations with experimental data. A well-known result of Zippel is the sparse interpolation [Zip79] paper. Many gcd computation algorithms implemented in computer algebra platforms use this approach, including Maple, Magma and Singular. Our observations in Chapter 6 show that one of the assumptions made by Zippel in his complexity analysis of sparse interpolation [Zip79] is false. We revise this assumption and correct the analysis. We hopefully modify the common perception that in Zippel's variable by variable sparse interpolation,that most of the work is done when interpolating the last variable x_n: For most sparse polynomials, the number of evaluation points needed to interpolate the last $n/2$ variables is the same. Next we investigate further and compute the expected number of terms of Taylor coefficients of f expanded around a non-zero random point γ. Then we compute the complexity of solving a single MDP. As the complexity analysis is tedious, we hope it will help the reader to follow the section in which we compute the complexity of MTSHL. Eventually we present some timing data to compare our factorization algorithms with Wang's algorithm, which is currently used in Maple. Most of the work explained in Chapter 6 has been submitted (to JSC) recently.

1.3 Univariate Hensel Lifting

The aim of this section is to give an overview of univariate Hensel lifting so that the reader can see how the ideas behind univariate Hensel lifting naturally generalize to the multivariate case which we will describe later. It is intended to be self-complete. For a comprehensive introduction and a detailed discussion on the subject we refer the reader to [GCL92]. We follow their treatment.

Suppose that we are given a polynomial $a(x) \in \mathbb{Z}[x]$, a prime number p and a pair of relatively prime factors $u_0(x), w_0(x) \in \mathbb{Z}_p[x]$ such that

$$a(x) = u_0(x)w_0(x) \mod p,$$

and our aim is to lift these factors to a pair of factors $u(x), w(x) \in \mathbb{Z}[x]$ such that $u(x) = u_0(x) \mod p$ and $w(x) = w_0(x) \mod p$.

In other words, we want to invert the modular homomorphism

$$\phi_p : \mathbb{Z}[x] \to \mathbb{Z}_p[x].$$

where $\phi_p(f) = f \mod p$. The idea of Hensel lifting is to choose $l \in \mathbb{N}$ big enough so that $p^l/2$ is bigger than any of the integer coefficients of $a(x)$ and the possible factors $u(x), w(x) \in \mathbb{Z}[x]$. This is done to identify $\mathbb{Z}$ with $\mathbb{Z}_{p^l}$ for this specific problem.

Before we explain this approach, it is important to note that such a solution pair in $\mathbb{Z}[x]$ may not exist, even if the polynomial in $\mathbb{Z}[x]$ splits into irreducible factors over some prime p. For example the polynomial $a(x) = x^4 + 1 \in \mathbb{Z}[x]$ is irreducible over $\mathbb{Z}$ and splits into quadratic factors for $p = 5$: $a(x) = (x^2 + 2)(x^2 - 2) \in \mathbb{Z}_5[x]$. In fact, $a(x)$ splits into quadratic factors for any choice of $p > 2$.

Now we explain the lifting process using the bivariate Newton iteration: We start by assuming that the polynomial solutions $\tilde{u}(x), \tilde{w}(x) \in \mathbb{Z}[x]$ to the problem exist. Given $a(x) \in \mathbb{Z}[x]$, it is possible to determine such a bound $l \in \mathbb{N}$ as described above by using the Mignotte's bound [Mig74] which is given in the Theorem 1 below.

For a given univariate polynomial $f = \sum_{i=0}^{n} f_i x^i \in \mathbb{Z}[x]$ the ∞-norm is defined as $\|f\|_\infty = \max_{0 \le i \le n} |f_i|$.

Theorem 1. *(Mignotte's bound [Mig74]) Let $f, g, h \in \mathbb{Z}[x]$ be univariate polynomials where degree of f, g, h is d, m, k respectively. If the product gh divides f in $\mathbb{Z}[x]$ then we have*

$$\|g\|_\infty \|h\|_\infty \le 2^{m+k}\sqrt{d+1}\|f\|_\infty.$$

Corollary 2. *Applying Theorem 1 to $g = 1$ and keeping f, h as before leads to the following bound for every coefficient of every factor h of f.*

$$\|h\|_\infty \le 2^{d}\sqrt{d+1}\|f\|_\infty.$$

Now consider the p-adic representations

$$\begin{aligned} \tilde{u} &= u_0 + u_1 p + \cdots + u_l p^l \\ \tilde{w} &= w_0 + w_1 p + \cdots + w_l p^l. \end{aligned}$$

For $1 \le k \le l$, let us define $u^{(1)} = u_0$ and

$$u^{(k)} = \tilde{u} \mod p^k = u_0 + u_1 p + \cdots + u_{k-1} p^{k-1}$$

and $\Delta u^{(k)} = u_k p^k$. Then we have $u^{(k+1)} = u^{(k)} + \Delta u^{(k)}$. We define similar notation for w.

We want to obtain the factorization $a = u^{(k+1)} w^{(k+1)} \mod p^{k+1}$ given a factorization $a = u^{(k)} w^{(k)} \mod p^k$. We have

$$\begin{aligned} a - u^{(k+1)} w^{(k+1)} &= a - (u^{(k)} + u_k p^k)(w^{(k)} + w_k p^k) \\ &= a - u^{(k)} w^{(k)} - (u_k w^{(k)} + w_k u^{(k)}) p^k - u_k w_k p^{2k} \\ &= a - u^{(k)} w^{(k)} - (u_k w^{(k)} + w_k u^{(k)}) p^k \mod p^{k+1}. \end{aligned}$$

Algorithm 1 Univariate Diophantine Solver (UniDio)

Input: A field $\mathbb{F}$, $a, b, c \in \mathbb{F}[x]$ such that $\gcd(a, b) \mid c$ with $g = \gcd(a, b)$.

Output: $\sigma, \tau \in \mathbb{F}[x]$ such that $\sigma a + \tau b = c$ with $\deg \sigma < \deg b - \deg g$. Moreover if $\deg c < \deg a + \deg b - \deg g$ then τ satisfies $\deg \tau < \deg a - \deg g$.

1: Find $s, t \in \mathbb{F}[x]$ such that $sa + tb = g$ via the extended Euclidean algorithm.
2: Set $\tilde{\sigma} := sc/g$ and $\tilde{\tau} = tc/g$.
3: Divide $\tilde{\sigma}$ by b/g so that $\tilde{\sigma} = (b/g)q + r$ for some unique $q, r \in \mathbb{F}[x]$.
4: Set $\sigma := r$ and $\tau := \tilde{\tau} + q(a/g)$.
5: **return** (σ, τ).

Therefore if $a = u^{(k+1)}w^{(k+1)} \mod p^{k+1}$ we should have

$$u_k w^{(k)} + w_k u^{(k)} = \frac{a - u^{(k)}w^{(k)}}{p^k} \mod p.$$

Observe that $\phi_p(u^{(k)}) = u_0$ and $\phi_p(w^{(k)}) = w_0$. Hence we get

$$w_0 u_k + u_0 w_k = \phi_p\left(\frac{a - u^{(k)}w^{(k)}}{p^k}\right). \tag{1.1}$$

It turns out that solving Eqn (1.1) for u_k and w_k is crucial to get the factors u and w. Eqn (1.1) is an instance of a univariate diophantine problem (UDP) in $\mathbb{Z}_p[x]$. We state it formally and give an algorithm for its solution :

Theorem 3. *Let $\mathbb{F}[x]$ be the Euclidean domain of univariate polynomials over a field $\mathbb{F}$. Let $a, b \in \mathbb{F}[x]$ be given non-zero polynomials and let $g = \gcd(a, b) \in \mathbb{F}[x]$. Then for any given polynomial $c \in \mathbb{F}[x]$ such that $g \mid c$ there exist unique polynomials $\sigma, \tau \in \mathbb{F}[x]$ such that*

$$\sigma a + \tau b = c \tag{1.2}$$

where $\deg \sigma < \deg b - \deg g$. Moreover if $\deg c < \deg a + \deg b - \deg g$ then τ satisfies $\deg \tau < \deg a - \deg g$.

Proof. An algorithm to find the unique solution pair is described in Algorithm 1, UniDio. *Correctness*: After finding $s, t \in \mathbb{F}[x]$ such that $sa+tb = g$ via extended Euclidean algorithm (EAA) we have $s(a/g) + t(b/g) = 1 \Rightarrow sc(a/g) + tc(b/g) = c \Rightarrow sa(c/g) + tb(c/g) = c$. Then let $\tilde{\sigma} := s(c/g)$ and $\tilde{\tau} := t(c/g)$. By dividing $\tilde{\sigma}$ by b/g we get $s(c/g) = (b/g)q + r$ for some unique $q, r \in \mathbb{F}[x]$ so that $\deg_x(r) < \deg_x(b/g) = \deg_x(b) - \deg_x(g)$. Now set $\sigma := r$ and $\tau := \tilde{\tau} + q(a/g)$. Then in $\mathbb{F}[x]$ we have

$$\sigma a+\tau b = ra+(t(c/g)+q(a/g))b = (s(c/g)-q(b/g))a+(t(c/g)+q(a/g))b = s(c/g)a+t(c/g)b = c.$$

Uniqueness: Suppose one also has $\bar{\sigma}, \bar{\tau} \in \mathbb{F}[x]$ such that $\deg \bar{\sigma} < \deg b - \deg g$ satisfying $\bar{\sigma}a + \bar{\tau}b = c$. Then $(\sigma - \bar{\sigma})(a/g) = -(\tau - \bar{\tau})(b/g)$. Since $\gcd(a/g, b/g) = 1 \Rightarrow (b/g) \mid (\sigma - \bar{\sigma})$. Then $\deg(b/g) \leq \deg(\sigma - \bar{\sigma}) < \deg(b/g)$. This is possible only if $\sigma - \bar{\sigma} = 0 \Rightarrow \sigma = \bar{\sigma} \Rightarrow \tau = \bar{\tau}$.

Finally suppose that $\deg(c) < \deg(a) + \deg(b) - \deg(g)$. We have shown above the existence of the solution. So we can write $\tau = (c - \sigma a)/b$. Then $\deg(\tau) = \deg(c - \sigma a) - \deg(b)$. Now if

- $\deg c \geq \deg(\sigma a) \Rightarrow \deg \tau \leq \deg c - \deg b < \deg a - \deg g$.
- $\deg c < \deg(\sigma a) \Rightarrow \deg \tau \leq \deg \sigma + \deg a - \deg b < (\deg b - \deg g) + \deg a - \deg b = \deg a - \deg g$.

□

Now we continue our discussion on univariate Hensel lifting (UHL).

Since $\gcd(u_0, w_0) = 1 \in \mathbb{Z}_p[x]$, we can solve Eqn(1.1) using Algorithm UniDio to find unique polynomials $\sigma_k, \tau_k \in \mathbb{Z}_p[x]$ such that

$$\sigma_k u_0 + \tau_k w_0 = \phi_p \left(\frac{a - u^{(k)} w^{(k)}}{p^k} \right)$$

where $\deg \sigma_k < \deg w_0$.
We then claim that the order $k+1$ p-adic approximations to the solutions $\tilde{u}$ and $\tilde{w}$ are

$$u^{(k+1)} = u^{(k)} + \tau_k p^k \text{ and } w^{(k+1)} = w^{(k)} + \sigma_k p^k.$$

Let us state and prove it formally.

Theorem 4. *(Hensel's Lemma). Let p be a prime in $\mathbb{Z}$ and let $a(x) \in \mathbb{Z}[x]$ be a given polynomial over the integers. Let $u^{(1)}, w^{(1)} \in \mathbb{Z}_p[x]$ be two relatively prime polynomials over the field $\mathbb{Z}_p$ such that*

$$a(x) = u^{(1)} w^{(1)} \mod p.$$

Then for any integer $k \geq 1$ there exists polynomials $u^{(k)}, w^{(k)} \in \mathbb{Z}_{p^k}[x]$ such that

$$a(x) = u^{(k)} w^{(k)} \mod p^k$$

and

$$u^{(k)} = u^{(1)} \mod p \text{ and } w^{(k)} = w^{(1)} \mod p.$$

Proof. The case $k = 1$ is already given. Assume that claim is true for $k \geq 1$ and we have $u^{(k)}, w^{(k)} \in \mathbb{Z}_{p^k}[x]$ satisfying the conditions given in the theorem. We define

$$c_k(x) = \phi_p \left(\frac{a(x) - u^{(k)} w^{(k)}}{p^k} \right) \tag{1.3}$$

where all operations are performed in $\mathbb{Z}[x]$ before applying ϕ_p. Since $\gcd(u^{(1)}, w^{(1)}) = 1$, by using Algorithm UniDio we can find the unique polynomials $\sigma_k, \tau_k \in \mathbb{Z}_p[x]$ such that

$$\sigma_k u^{(1)} + \tau_k w^{(1)} = c_k \mod p \tag{1.4}$$

where $\deg \sigma_k < \deg w^{(1)}$. We then define

$$u^{(k+1)} = u^{(k)} + \tau_k p^k \text{ and } w^{(k+1)} = w^{(k)} + \sigma_k p^k. \tag{1.5}$$

Let $u^{(k)} = \sum_{s=0}^{k-1} u_s p^s$ and $w^{(k)} = \sum_{s=0}^{k-1} w_s p^s$ where $u_0 = u^{(1)}$ and $w_0 = w^{(1)}$. Now passing to mod p^{k+1} we have

$$\begin{aligned}
u^{(k+1)}w^{(k+1)} &= (u^{(k)} + \tau_k p^k)(w^{(k)} + \sigma_k p^k) \mod p^{k+1} \\
&= u^{(k)}w^{(k)} + (\sigma_k u^{(k)} + \tau_k w^{(k)})p^k \mod p^{k+1} \\
&= u^{(k)}w^{(k)} + (\sigma_k \sum_{s=0}^{k-1} u_s p^s + \tau_k \sum_{s=0}^{k-1} w_s p^s)p^k \mod p^{k+1} \\
&= u^{(k)}w^{(k)} + (\sigma_k u^{(1)} + \tau_k w^{(1)})p^k \mod p^{k+1} \\
&= u^{(k)}w^{(k)} + c_k p^k \mod p^{k+1}, \text{ by Eqn(1.4)} \\
&= a \mod p^{k+1}, \text{ by Eqn(1.3)}.
\end{aligned}$$

From Eqn (1.5) it is clear that

$$u^{(k+1)} = u^{(k)} \mod p^k \text{ and } w^{(k+1)} = w^{(k)} \mod p^k.$$

Hence by the induction hypothesis

$$u^{(k+1)} = u^{(1)} \mod p \text{ and } w^{(k+1)} = w^{(1)} \mod p.$$

□

Example 5. This example considers the uniqueness of Hensel construction described in Theorem 4. Consider the example where $a(x) = x^2 - 1 \in \mathbb{Z}[x]$ and $p = 5$. Also let

$$u_1^{(1)} = x - 1,\ w_1^{(1)} = x + 1,\ u_1^{(2)} = x - 1,\ w_1^{(2)} = x + 1$$

and

$$u_2^{(1)} = x - 1,\ w_2^{(1)} = x + 1,\ u_2^{(2)} = -9(x - 1),\ w_2^{(2)} = 11(x + 1).$$

Since $-9 = 11 = 1 \mod 5$ we have $u_1^{(1)} = u_2^{(1)} \mod 5$, $w_1^{(1)} = w_2^{(1)} \mod 5$ and

$$\begin{aligned} a(x) &= u_1^{(2)} w_1^{(2)} = u_2^{(2)} w_2^{(2)} \mod 5^2 \\ a(x) &= u_1^{(1)} w_1^{(1)} = u_2^{(1)} w_2^{(1)} \mod 5. \end{aligned}$$

Thus, even if we impose monicness on $a(x) \in \mathbb{Z}[x]$ and the factors $u^{(1)}, w^{(1)} \in \mathbb{Z}_p[x]$, the conditions in Theorem 4 do not imply uniqueness of the factors $u^{(k)}, w^{(k)} \in \mathbb{Z}_{p^k}[x]$ for $k > 1$.

On the other hand, note that $u_2^{(2)}$ and $u_1^{(2)}$ are associates in the ring $\mathbb{Z}_{p^2}[x]$. The same is true for $w_1^{(2)}$ and $w_2^{(2)}$. In fact one can show that in such a case different liftings must be associates. (See [GCL92]). This observation will be important to solve the leading coefficient problem.

If we impose monicness on $u^{(k)}, w^{(k)}$ for each $k \geq 1$, then we have the uniqueness property of the Hensel construction as shown below.

Corollary 6. *(Uniqueness of the Hensel Construction) In Theorem 4, if the given polynomial $a(x) \in \mathbb{Z}[x]$ is monic and correspondingly if the relatively prime factors $u^{(1)}, w^{(1)} \in \mathbb{Z}_p[x]$ are chosen to be monic, then for any natural number $k \geq 1$, monic polynomials $u^{(k)}, w^{(k)} \in \mathbb{Z}_{p^k}[x]$ in Theorem 5 are unique.*

Proof. The proof is again by induction. For the case $k = 1$ the given polynomials are clearly unique since the solution pairs are relatively prime over $\mathbb{Z}_p[x]$. Assume that claim is true for $k \geq 1$ and we have unique monic $u^{(k)}, w^{(k)} \in \mathbb{Z}_{p^k}[x]$ satisfying the conditions given in the theorem. We need to prove the uniqueness of the monic polynomials $u^{(k+1)}, w^{(k+1)} \in \mathbb{Z}_{p^k}[x]$ satisfying

$$a(x) = u^{(k+1)} w^{(k+1)} \mod p^{k+1}$$

and

$$u^{(k+1)} = u^{(1)} \mod p \text{ and } w^{(k+1)} = w^{(1)} \mod p.$$

We have

$$a(x) = u^{(k+1)} w^{(k+1)} \mod p^{k+1} \Rightarrow a(x) = u^{(k+1)} w^{(k+1)} \mod p^k.$$

But then by induction we also have $a(x) = u^{(k)} w^{(k)} \mod p^k$. Hence by induction assumption on the uniqueness of lifting over p^k we have

$$u^{(k+1)} = u^{(k)} \mod p^k \text{ and } w^{(k+1)} = w^{(k)} \mod p^k.$$

Therefore p^k divides both $u^{(k+1)} - u^{(k)}$ and $w^{(k+1)} - w^{(k)}$. Hence we can write

$$u^{(k+1)} = u^{(k)} + \tau(x) p^k \text{ and } w^{(k+1)} = w^{(k)} + \sigma(x) p^k \tag{1.6}$$

for some polynomials $\sigma(x), \tau(x) \in \mathbb{Z}_p[x]$ and it remains to prove the uniqueness of σ and τ.

For a given polynomial f, let us by $\mathrm{LC}_{\mathrm{x_i}}(\mathrm{f})$ we denote the leading coefficient of f w.r.t variable x_i. We don't use the subscript if f is univariate. According to Eq (1.6) if $\deg \sigma \geq \deg w^{(k)}$ then $1 = \mathrm{LC}(\mathrm{w}^{(\mathrm{k}+1)}) = \mathrm{LC}(w^{(k)} + \sigma(x)p^k) \neq 1$ because of the multiple p^k and monicness of $w^{(k)}$. The argument is similar for τ. So, since $a, u^{(1)}, w^{(1)}$ are all monic for $k \geq 1$, we must have

$$\deg \sigma < \deg w^{(1)} \text{ and } \deg \tau < \deg u^{(1)}.$$

This means that $u^{(k+1)}$ and $w^{(k+1)}$ must always have the same leading terms as $u^{(1)}$ and $w^{(1)}$ respectively (This observation is important for the next section where we discuss the leading coefficient problem.). Now as before, over $\mathbb{Z}_{p^{k+1}}[x]$ we have

$$a = u^{(k)}w^{(k)} + (\sigma u^{(1)} + \tau w^{(1)})p^k \mod p^{k+1}.$$

It follows that

$$\sigma u^{(1)} + \tau w^{(1)} = \frac{a - u^{(k)}w^{(k)}}{p^k} \mod p.$$

But any solution σ satisfying this equation with

$$\deg(\sigma(x)) < \deg(w^{(1)}(x))$$

is unique by Theorem 3. By the uniqueness of univariate division in $\mathbb{Z}_p[x]$, it follows that $\tau(x)$ is also unique. □

1.4 Leading Coefficient Correction (LCC)

Corollary 6 that we have proved in the last section shows the uniqueness of the Hensel lifting in the monic case. We consider the non-monic case now. For a detailed discussion we refer [GCL92]. Here we sketch the idea behind the LCC.

A polynomial is called primitive, if the greatest common divisor of its coefficients is equal to 1. Let $a = uw$ with $a, u, w \in \mathbb{Z}[x]$ all primitive and let

$$\alpha = \mathrm{LC(a)};\ \mu = \mathrm{LC(u)};\ \nu = \mathrm{LC(w)}.$$

Suppose also $p \nmid \alpha$. We should satisfy $\alpha = \mu\nu$. Now we define $\tilde{a} := \alpha a$. Then

$$\tilde{a} = \mu\nu uw = [\nu u][\mu w].$$

So if we define $\tilde{u} := \nu u$ and $\tilde{w} := \mu w$, then there is a factorization of the form $\tilde{a} = \tilde{u}\tilde{w} \in \mathbb{Z}[x]$ where both factors have the leading coefficient α. This observation suggests that we can solve the leading coefficient problem by simply correcting the leading coefficient of the lifted factors by $\alpha \mod p^k$, i.e. each time we compute $\tilde{u}^{(k)}, \tilde{w}^{(k)}$ in the for loop in the Algorithm

UHL and update the correct the leading coefficients by $\alpha \mod p^k$. In other words, if $u^{(k)}, w^{(k)}$ denote modulo p^k factors $\tilde{u}^{(k)}, \tilde{w}^{(k)}$ which maintain the conditions of Hensel's lemma, we simply define

$$\begin{aligned} \tilde{u}^{(k)} &= \phi_{p^k}(\alpha\,\mathrm{LC}(\mathrm{u}^{(\mathrm{k})})^{-1}\mathrm{u}^{(\mathrm{k})}) \\ \tilde{w}^{(k)} &= \phi_{p^k}(\alpha\,\mathrm{LC}(\mathrm{w}^{(\mathrm{k})})^{-1}\mathrm{w}^{(\mathrm{k})}). \end{aligned}$$

Note that if $p \nmid \mathrm{LC(a)} = \alpha$ then $p \nmid \mathrm{LC}(\mathrm{u}^{(\mathrm{k})})$ and $p \nmid \mathrm{LC}(\mathrm{w}^{(\mathrm{k})})$ so $\mathrm{LC}(\mathrm{u}^{(\mathrm{k})})^{-1}$ and $\mathrm{LC}(\mathrm{w}^{(\mathrm{k})})^{-1}$ exist modulo p^k.

Finally since $a \in \mathbb{Z}[x]$ was assumed to be primitive, after the last step of lifting we define the actual factors as $u = \mathrm{pp}(\tilde{u})$, $w = \mathrm{pp}(\tilde{w})$, where $\mathrm{pp}(f(x))$ denotes the primitive part of a given polynomial $f(x) \in \mathbb{Z}[x]$.

But there is an ingenious modification described by Yun [Yun74] which is attributed to a suggestion by J. Moses. Modulo p the factors of $\tilde{a}$ must be of the form

$$\begin{aligned} \tilde{u}^{(1)} &= \alpha_p x^m + \beta_{m-1}x^{m-1} + \cdots + \beta_0 \\ \tilde{w}^{(1)} &= \alpha_p x^n + \gamma_{n-1}x^{n-1} + \cdots + \gamma_0 \end{aligned}$$

where $\alpha_p = \phi_p(\alpha)$. Now suppose that the factors $\tilde{u}^{(1)}, \tilde{w}^{(1)}$ are simply changed by replacing the leading coefficients α_p by α. To this end we define the **replaceLC** as follows:

Given a polynomial $a(x) \in R[x]$ over a coefficient ring R and a given $r \in R$, **replaceLC**$(a(x), r)$ is the polynomial replacing the leading coefficient of $a(x)$ by r.

Then after applying **replaceLC** we get

$$\begin{aligned} \tilde{u}^{(1)} &= \alpha x^m + \beta_{m-1}x^{m-1} + \cdots + \beta_0 \\ \tilde{w}^{(1)} &= \alpha x^n + \gamma_{n-1}x^{n-1} + \cdots + \gamma_0. \end{aligned}$$

In other words the leading coefficients of $\tilde{u}^{(1)}$, $\tilde{w}^{(1)}$ are no longer represented as elements of the field $\mathbb{Z}_p$, but nonetheless we still have $\tilde{a} = \tilde{u}\tilde{w} \in \mathbb{Z}_p[x]$.

The Hensel construction can therefore be applied using this modified factors: Let us consider the step 9 in the algorithm in the for loop. We first compute

$$c = \frac{\tilde{a} - \tilde{u}^{(1)}\tilde{w}^{(1)}}{p}.$$

Note that the domain of this operation is $\mathbb{Z}[x]$. Since $\mathrm{LC}(\tilde{\mathrm{a}}) = \alpha^2$ we have

$$\deg c < \deg \tilde{a} = \deg \tilde{u}^{(1)} + \deg \tilde{w}^{(1)}.$$

This means that the solution to the diophantine equation

$$\sigma\tilde{u}^{(1)} + \tau\tilde{w}^{(1)} = c$$

will satisfy the usual condition $\deg\sigma < \deg\tilde{w}^{(1)}$ and also satisfy the additional condition (See Theorem 3)

$$\deg\tau < \deg\tilde{u}^{(1)}.$$

Before we give a concrete example which covers what we explained so far, note that this last idea allows us to handle LCC outside the for loop as described in the algorithm UHL below. The fact that LCC can be handled outside the for loop will remain valid for the generalization of Hensel's construction for the multivariate case.

Univariate Hensel lifting algorithm described as Algorithm 2, UHL. It is based on [GCL92]. A concrete example is in order.

Example 7. Suppose that we are trying to factor

$$a(x) = 12x^3 + 10x^2 - 36x + 35 \in \mathbb{Z}[x].$$

We choose $p = 5$. Note that LC(a) $= \alpha = 12$ and $p \nmid \alpha$. Observe that a is primitive and

$$a = u^{(1)}w^{(1)} \mod 5 \tag{1.7}$$

where $u^{(1)} = 2x$, $w^{(1)} = x^2 + 2 \in \mathbb{Z}_5[x]$. Now we define

$$\tilde{a}(x) = 12a = 144x^3 + 120x^2 - 432x + 420 \in \mathbb{Z}[x].$$

Now, if there is a factorization $a = uw \in \mathbb{Z}[x]$ satisfying Eqn (1.7) then there is a factorization of $\tilde{a}$ where 12 is the leading coefficient of each factor .

As a first step we update $u^{(1)} \leftarrow \phi_5(12 \cdot 2^{-1} \cdot u^{(1)}) = 2x$ and $w^{(1)} \leftarrow \phi_5(12 \cdot 1^{-1} \cdot w^{(1)}) = 2x^2 - 1$. Then by solving $su^{(1)} + tw^{(1)} = 1$ via EEA we get $s = x$, $t = -1 \in \mathbb{Z}_5[x]$ such that $su^{(1)} + tw^{(1)} = 1$. Then by applying **replaceLC** we get

$$\tilde{u}^{(1)} = 12x\,,\ \tilde{w}^{(1)} = 12x^2 - 1.$$

Now the error is $e_1 = \tilde{a} - \tilde{u}^{(1)}\tilde{w}^{(1)} = 120x^2 - 420x + 420$. In iteration step $k = 1$ we first get

$$c_1 = e_1/5 = 24\,x^2 - 84\,x + 84.$$

Algorithm 2 Univariate Hensel Lifting (UHL)

Input

1. A primitive polynomial $a \in \mathbb{Z}[x]$.
2. A prime p such that $p \nmid \mathrm{LC(a)}$.
3. $u^{(1)}, w^{(1)} \in \mathbb{Z}_p[x]$ such that $\gcd(u^{(1)}, w^{(1)}) = 1$ and $a = u^{(1)}w^{(1)} \mod p$.
4. An integer B which bounds the magnitudes of all integer coefficients appearing in a and its possible factors with degrees not exceeding $\max\{\deg u^{(1)}, \deg w^{(1)}\}$.

Output

1. Polynomials $u, w \in \mathbb{Z}[x]$ such that $a = uw \in \mathbb{Z}[x]$ and

$$\mathbf{monic}(u) = \mathbf{monic}(u^{(1)}) \mod p, \ \mathbf{monic}(w) = \mathbf{monic}(w^{(1)}) \mod p$$

   where **monic** denotes the function which makes the polynomial monic as an element of the domain $\mathbb{Z}_p[x]$.

2. If there is no factorization of $a \in \mathbb{Z}[x]$ such that $a = u^{(1)}w^{(1)} \mod p$, it returns 'no such factorization'.

$\alpha \leftarrow \mathrm{lcoeff(a)}$
$a \leftarrow \alpha a$
$u^{(1)} \leftarrow \phi_p(\alpha \mathbf{monic}(u^{(1)}))$, $w^{(1)} \leftarrow \phi_p(\alpha \mathbf{monic}(w^{(1)}))$
$u^{(1)} \leftarrow \mathrm{mods}(u^{(1)}, p)$, $w^{(1)} \leftarrow \mathrm{mods}(w^{(1)}, p)$. *(Symmetric range)*
$u \leftarrow \mathbf{replaceLC}(u^{(1)}, \alpha)$, $w \leftarrow \mathbf{replaceLC}(w^{(1)}, \alpha)$
$e(x) \leftarrow a - uw$
$modulus \leftarrow p$
while $e \neq 0$ **and** $modulus < 2B\alpha$ **do**
    $c \leftarrow (e/modulus) \mod p$
    $(\sigma, \tau) \leftarrow \mathrm{UniDio}(u^{(1)}, w^{(1)}, c)$ (Algorithm 1 with $\mathbb{F} = \mathbb{Z}_p$).
    $\sigma \leftarrow \mathrm{mods}(\sigma, p)$, $\tau \leftarrow \mathrm{mods}(\tau, p)$. *(Symmetric range)*
    $u \leftarrow u + \tau \cdot modulus$, $w \leftarrow w + \sigma \cdot modulus$
    $e \leftarrow a - uw$
    $modulus \leftarrow modulus \cdot p$
end while
if $e = 0$ **then**
    return $(\mathrm{primpart}(u), \mathrm{primpart}(w))$
else
    return no such factorization exists
end if

By solving $\sigma w^{(1)} + \tau u^{(1)} = c_1$ by Algorithm UniDio we get $\sigma = 1$, $\tau = x - 2 \in \mathbb{Z}_5[x]$ and

$$\begin{aligned} u^{(2)} &= \tilde{u}^{(1)} + 1 \cdot 5 = 12x + 5 \\ w^{(2)} &= \tilde{w}^{(1)} + (x-2) \cdot 5 = 12x^2 + 5x - 11. \end{aligned}$$

Now the error is $e_2(x) = \tilde{a} - \tilde{u}^{(2)}\tilde{w}^{(2)} = -325x + 475$. In iteration step $k = 2$ we first get

$$c_2 = e_2/5^2 = -13x + 19.$$

By solving $\sigma w^{(1)} + \tau u^{(1)} = c_2$ by Algorithm UniDio we get $\sigma = 1$, $\tau = 1 - x \in \mathbb{Z}_5[x]$ and

$$\begin{aligned} u^{(3)} &= \tilde{u}^{(2)} + (1) \cdot 5^2 = 12x + 30 \\ w^{(3)} &= \tilde{w}^{(2)} + (1-x) \cdot 5^2 = 12x^2 - 20x + 14. \end{aligned}$$

At this point the error is $e_3 = \tilde{a} - \tilde{u}^{(3)}\tilde{w}^{(3)} = 0 \Rightarrow \tilde{a} = \tilde{u}^{(3)}\tilde{w}^{(3)}$. Finally taking the primitive parts we get the actual factors

$$\begin{aligned} u(x) &= \mathrm{pp}(\tilde{u}^{(3)}(x)) = \tilde{u}^{(3)}(x)/6 = 2x + 5 \\ w(x) &= \mathrm{pp}(\tilde{w}^{(3)}(x)) = \tilde{w}^{(3)}(x)/2 = 6x^2 - 10x + 7. \end{aligned}$$

1.5 Multiterm Hensel Lifting

A generalization of the two-term univariate Hensel lifting described in the previous sections is straightforward. This time we will start with the factorization

$$a = u_1^{(1)} \cdots u_r^{(1)} \in \mathbb{Z}_p[x]$$

where $\gcd(u_i^{(1)}, u_j^{(1)}) = 1 \,\mathrm{in}\, \mathbb{Z}_p[x]$ for $i \neq j$.

Then during the multiterm Hensel lifting, for each lifting step $k > 0$ we will find $\sigma_i^{(k+1)} \in \mathbb{Z}_p[x]$ to get

$$u_i^{(k+1)} = u_i^{(k)} + \sigma_i^{(k)} p^k \in \mathbb{Z}_{p^{k+1}}[x] \,\mathrm{so\,that}\, u_i^{(k)} = u_i^{(1)} \in \mathbb{Z}_p[x]$$

where $\deg(\sigma_i^{(k)}) < \deg(u_i^{(k)})$. Note that

$$\prod_{i=1}^{r} u_i^{(k+1)} = \prod_{i=1}^{r} (u_i^{(k)} + \sigma_i^{(k)} p^k) = \prod_{i=1}^{r} u_i^{(k)} + p^k \left(\sum_{i=1}^{r} \sigma_i^{(k)} \prod_{j=1, j\neq i}^{r} u_j^{(k)} \right) \mod p^{k+1},$$

hence

$$\left(\prod_{i=1}^{r} u_i^{(k+1)} - \prod_{i=1}^{r} u_i^{(k)} \right) / p^k = \sum_{i=1}^{r} \sigma_i^{(k)} \prod_{j=1, j\neq i}^{r} u_j^{(k)} \mod p.$$

Therefore we are faced with the multiterm version of the univariate diophantine problem:

Suppose that we are given polynomials $u_1, \ldots, u_r \in \mathbb{Z}_p[x]$ where $p \nmid \mathrm{LC}(\prod_{\mathrm{i=1}}^{\mathrm{r}} \mathrm{u_i})$. Let $b_j = \prod_{i=1, i\neq j}^{r} u_i$ so that $b_j u_j = \prod_{i=1}^{r} u_i$. Let $c \in \mathbb{Z}_p[x]$ such that $\deg c < \deg(\prod_{i=1}^{r} u_i)$. The aim is to find $\sigma_j \in \mathbb{Z}_p[x], j = 1, \ldots, r$ such that

$$\sigma_1 b_1 + \cdots + \sigma_r b_r = c$$

where $\sigma_j = 0$ or $\deg \sigma_j < \deg u_j$.

The solution is obtained by first solving the multiterm version of the EEA: Find $s_j \in \mathbb{Z}_p[x], j = 1, \ldots, r$ such that

$$s_1 b_1 + \cdots + s_r b_r = 1$$

where $\deg s_j < \deg u_j$.

Before we describe the general solution to the multiterm EEA, let us first consider the case where $r = 4$, to see the idea behind: The task is to find $s_1, s_2, s_3, s_4 \in \mathbb{Z}_p[x]$ such that

$$\begin{aligned} s_1 b_1 + s_2 b_2 + s_3 b_3 + s_4 b_4 &= 1 \\ \Rightarrow s_1 u_2 u_3 u_4 + s_2 u_1 u_3 u_4 + s_3 u_1 u_2 u_4 + s_4 u_1 u_2 u_3 &= 1 \\ \Rightarrow s_1 u_2 u_3 u_4 + u_1(s_2 u_3 u_4 + s_3 u_2 u_4 + s_4 u_2 u_3) &= 1 \\ \Rightarrow s_1 u_2 u_3 u_4 + u_1(s_2 u_3 u_4 + u_2(s_3 u_4 + s_4 u_3)) &= 1 \end{aligned}$$

We start by defining $\beta_0 := 1$ and solve the diophantine equation

$$s_1 u_2 u_3 u_4 + u_1 \beta_1 \;=\; 1 \mod p$$

for $s_1, \beta_1 \in \mathbb{Z}_p[x]$ such that $\deg s_1 < \deg u_1$. Then we solve the diophantine equation

$$s_2 u_3 u_4 + u_2 \beta_2 \;=\; \beta_1 \mod p$$

for $s_2, \beta_2 \in \mathbb{Z}_p[x]$such that $\deg s_2 < \deg u_2$. Finally we solve the diophantine equation

$$s_3 u_4 + u_3 \beta_3 \;=\; \beta_2 \mod p$$

for $s_3, \beta_3 \in \mathbb{Z}_p[x]$ such that $\deg s_3 < \deg u_4$ and define $s_4 = \beta_3$.

So, we can describe the general procedure to solve the multiterm EEA: We define $\beta_0 := 1$. Then for j from 1 to $r - 1$ we solve the diophantine equation

$$\beta_j u_j + s_j \prod_{i=j+1}^{r} u_i = \beta_{j-1} \mod p$$

for $\beta_j, s_j \in \mathbb{Z}_p[x]$ such that $\deg s_j < \deg u_j$. Finally define $s_r := \beta_{r-1}$.

The first task is done. Now we have

$$\sum_{j=1}^{r} s_j b_j = 1 \Rightarrow \sum_{j=1}^{r} c s_j b_j = c.$$

Then, since $p \nmid \mathrm{LC}(\mathrm{u_i})$ we can apply by univariate division to find the quotient and remainder $q_j, r_j \in \mathbb{Z}_p[x]$ such that

$$cs_j = u_j q_j + r_j$$

where $r_j = 0$ or $\deg r_j < \deg u_j$. Then we define $\sigma_j := r_j$. Observe that

$$\begin{aligned} cs_j &= u_j q_j + \sigma_j \\ \Rightarrow \sum_{j=1}^{r} b_j cs_j &= \sum_{j=1}^{r} b_j u_j q_j + \sum_{j=1}^{r} b_j \sigma_j \\ \Rightarrow c \sum_{j=1}^{r} b_j s_j &= \sum_{j=1}^{r} \prod_{i=1}^{r} u_i q_j + \sum_{j=1}^{r} b_j \sigma_j \\ \Rightarrow c &= \prod_{i=1}^{r} u_i \sum_{j=1}^{r} q_j + \sum_{j=1}^{r} b_j \sigma_j \end{aligned}$$

Since $\deg c < \deg(\prod_{i=1}^{r} u_i)$, it follows that $\sum_{j=1}^{r} q_j = 0$ and $c = \sum_{j=1}^{r} b_j \sigma_j$ where $\deg \sigma_j < \deg u_j$.

The leading coefficient problem is solved in a similar way as in two-term univariate Hensel lifting algorithm UHL. So, the algorithm for multiterm univariate Hensel lifting is a simple modification of the UHL algorithm by multiterm EEA and multiterm diophantine equations as we have described above.

1.6 The generalized Univariate Diophantine Problem (UDP)

We will see in Chapter 2 that during the process of MHL, Wang's algorithm needs to solve univariate diophantine problem Eq (1.4) over $\mathbb{Z}_{p^l}[x]$ which is not an Euclidean domain for $l > 1$. Instead, we will use the following Theorem.

Theorem 8. *[GCL92] For a prime integer p and a positive integer l, let $u, w \in \mathbb{Z}_{p^l}[x]$ be univariate polynomials satisfying*

(i) $p \nmid \mathrm{LC}(u)$ and $p \nmid \mathrm{LC}(w)$ *and*

(ii) $\gcd(\phi_p(u), \phi_p(w)) = 1$ in $\mathbb{Z}_p[x]$.

Then for any polynomial $c \in \mathbb{Z}_{p^l}[x]$ there exist unique polynomials $\sigma, \tau \in \mathbb{Z}_{p^l}[x]$ such that

$$\sigma u + \tau w = c \tag{1.8}$$

where $\deg\sigma < \deg w$.

Moreover if $\deg c < \deg u + \deg w$ *then* τ *satisfies* $\deg\tau < \deg u$.

To find the solution in the ring $\mathbb{Z}_{p^l}[x]$ for $l > 1$, the idea is to find the solution in $\mathbb{Z}_p[x]$ and lift it by using Newton iteration. We give the sketch of the proof.
The first step is to find $\tilde{s}, \tilde{w} \in \mathbb{Z}_{p^l}[x_1]$ such that $\tilde{s}u + \tilde{t}w = 1 \mod p^l$. To reach this aim, assume a context in which a polynomial solution is known to exist, and let the p-adic expansions of the solutions $\tilde{s}$ and $\tilde{t}$ be

$$\begin{aligned}\tilde{s} &= s + s_1 p + \cdots + s_k p^k + \cdots + s_l p^l \\ \tilde{t} &= t + t_1 p + \cdots + t_k p^k + \cdots + t_l p^l.\end{aligned}$$

Let also $s^{(k)} := \tilde{s} \mod p^k$ and $t^{(k)} := \tilde{t} \mod p^k$. The equation that we will apply Newton's iteration is: $G(s,t) = su + tw - 1 \in \mathbb{Z}[x_1]$ for the unknowns s, t. As we did in Section 1, if we have order k $p-$adic approximations $s^{(k)}, t^{(k)}$ to the actual solution pair $(\tilde{s}, \tilde{t})$ and if we obtain the correction terms $\triangle s^{(k)} := s_k p^k, \triangle t^{(k)} := t_k p^k$ by solving

$$G_s(s^{(k)}, t^{(k)})\triangle s^{(k)} + G_t(s^{(k)}, t^{(k)})\triangle s^{(k)} = -G(s^{(k)}, t^{(k)}) \mod p^{k+1},$$

then the order $(k+1)$ p-adic approximations will be $s^{(k+1)} = s^{(k)} + \triangle s^{(k)}$ and $t^{(k+1)} = t^{(k)} + \triangle t^{(k)}$. By substituting the derivatives we get

$$us_k + wt_k = \frac{1 - s^{(k)}u - t^{(k)}w}{p^k} \mod p. \tag{1.9}$$

Since $\gcd(u^{(1)}, w^{(1)}) = 1 \text{ in } \mathbb{Z}_p[x_1]$ we can apply Algorithm UniDio algorithm to find $s, t \in \mathbb{Z}_p[x]$ such that $su + tw = 1 \mod p$. Let $s^{(1)} := s$ and $t^{(1)} := t$. Then for $k = 1, 2, \ldots, l-1$ Eqn (1.9) can be used to compute s_k, t_k's. This recursive step will be finite. Then $\tilde{s} = s^{(l)}$ and $\tilde{\tau} = t^{(l)}$.

The next step is to compute σ and τ. We proceed as before. Let $\tilde{\sigma} := s^{(l)}c$ and $\tilde{\tau} := t^{(l)}c$. We have $p \nmid \mathrm{LC}(w)$ so $\mathrm{LC}(w)$ is a unit in $\mathbb{Z}_{p^l}[x]$. Then we can apply Euclidean division to compute $\tilde{\sigma} = wq + r \mod p^l$ for some $q, r \in \mathbb{Z}_{p^l}[x]$ such that $r = 0$ or $\deg r < \deg w$. Finally we define $\sigma := r$ and $\tau := \tilde{\tau} + qu$.

It remains to show the uniqueness of the solution and the degree constraints. Let $\sigma_1, \tau_1 \in \mathbb{Z}_{p^l}[x_1]$ and $\sigma_2, \tau_2 \in \mathbb{Z}_{p^l}[x_1]$ be two solution pairs satisfying the conditions (i) and (ii). Then we have

$$\begin{aligned}(\sigma_1 - \sigma_2)u &= -(\tau_1 - \tau_2)w \mod p^l \\ &\Downarrow \\ \phi_p(\sigma_1 - \sigma_2)\phi_p(u) &= -\phi_p(\tau_1 - \tau_2)\phi_p(w) \mod p.\end{aligned}$$

But we have $\gcd(\phi_p(u), \phi_p(w)) = 1 \mod p$. Then $\phi_p(w) \mid \phi_p(\sigma_1 - \sigma_2)$. Note that $p \nmid \mathrm{LC}(w)$, it follows that $\deg(\phi_p(w)) = \deg(w)$. On the other hand, by assumption we have

$$\deg(\sigma_1 - \sigma_2) \leq \max\{\deg(\sigma_1), \deg(\sigma_2)\} < \deg(w).$$

Thus $\sigma_1 - \sigma_2 = 0 \mod p \Rightarrow \tau_1 - \tau_2 = 0 \mod p$. Our aim is to show by induction that $p^k \mid (\sigma_1 - \sigma_2)$ and $p^k \mid (\tau_1 - \tau_2)$ for each $1 \leq k \leq l$. We have already proved the case $k = 1$. For the case $k \Rightarrow k + 1$: We define

$$\begin{array}{ccc} \alpha = (\sigma_1 - \sigma_2)/p^k & \text{and} & \beta = -(\tau_1 - \tau_2)/p^k \\ & \Downarrow & \\ \alpha u & = & \beta w \mod p^{l-k}. \end{array}$$

By repeating the degree argument above we get $\alpha = 0 \mod p$ and $\beta = 0 \mod p$. This means $p^{k+1} \mid (\sigma_1 - \sigma_2)$ and $p^{k+1} \mid (\tau_1 - \tau_2)$.

Finally we need to prove remaining degree constraints: $p \nmid \mathrm{LC}(w)$ implies that $\mathrm{LC}(w)$ is a unit in $\mathbb{Z}_{p^l}$. Hence the following division will be valid in $\mathbb{Z}_{p^l}[x_1]$:

$$\tau(x) = (c - \sigma u)/w.$$

Then $\deg(\tau(x)) = \deg(c - \sigma u) - \deg(w \leq \max\{\deg(c), \deg(\sigma) + \deg(u)\} - \deg(w)$

Now if $\deg(c) < \deg(u) + \deg(w)$ we have 2 cases:

(i) $\deg(c) \geq \deg(\sigma) + \deg(u) \Rightarrow \deg(\tau) \leq \deg(c) - \deg(w < \deg(u)$.

(ii) $\deg(c) < \deg(\sigma) + \deg(u) \Rightarrow \deg(\tau) \leq \deg(\sigma) + \deg(u) - \deg(w) < \deg(u)$ since we have already seen that $\deg(\sigma) < \deg(w)$. This completes the proof.

Chapter 2

Multivariate Factorization

2.1 The Steps of Multivariate Polynomial Factorization

In this section we will describe the steps of multivariate polynomial factorization as developed by Wang [Wan78]. This algorithm is currently being used by many computer algebra systems including Maple, Magma and Singular. Their implementations are all based on the description of the algorithm in Chapter 6 and 8 of the Geddes, Czapor and Labahn text [GCL92]. For the mathematics behind MHL we will follow the treatment of [GCL92] and also [Wan78] by providing concrete examples.

Suppose that the aim is to factor a polynomial $a \in \mathbb{Z}[x_1, \ldots, x_n]$. By choosing a main variable, say x_1, we may write a as a polynomial in x_1,

$$a(x_1, \ldots, x_n) = c_m x_1^m + c_{m-1} x_1^{m-1} + \cdots + c_1 x_1 + c_0$$

with coefficients $c_i \in \mathbb{Z}[x_2, \ldots, x_n]$ for $0 \leq i \leq m$ and the leading coefficient of a, $\mathrm{LC}_{x_1}(a) = c_m \neq 0$. The content of a, $\mathrm{cont}_x(a)$, with respect to the main variable x_1 is defined as

$$\mathrm{cont}_{x_1}(a) = \gcd(c_0, c_1, \ldots, c_m).$$

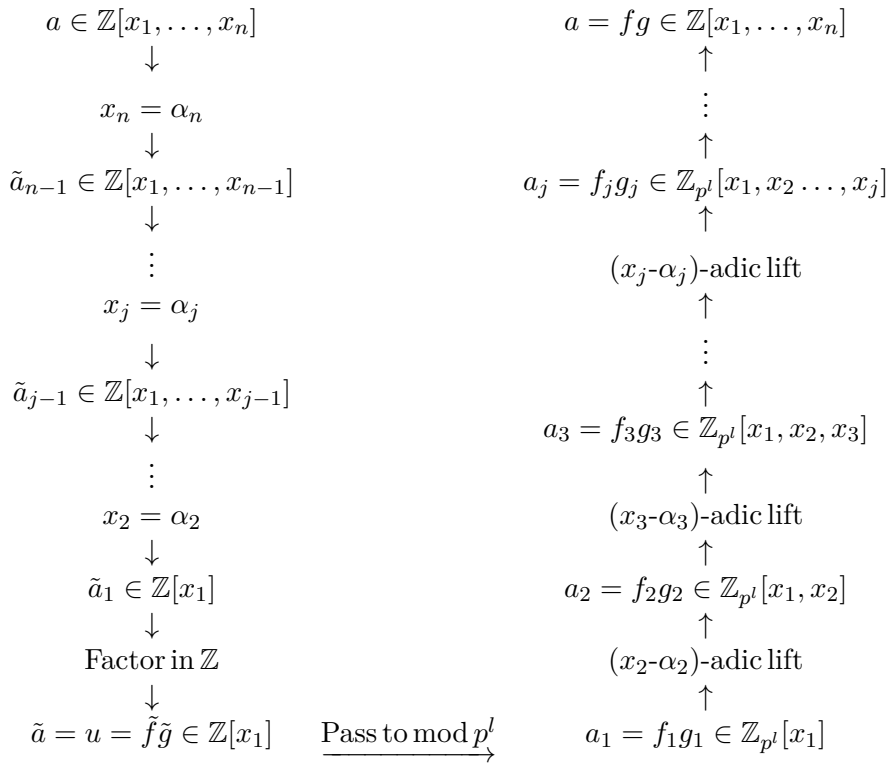
The primitive part of a, $\mathrm{pp}_{x_1}(a)$, is defined as $a/\mathrm{cont}_{x_1}(a)$. We say a is primitive if $\mathrm{cont}_{x_1}(a) = 1$.

We also say a is square-free if a has no repeated factors, that is there is no $b \in \mathbb{Z}[x_1, \ldots, x_n]$ with $\deg(b) > 0$ such that $b^2 \mid a$. We have the following useful lemma to determine whether a polynomial is square-free. For a proof of it see [GCL92].

Lemma 9. *Let $f(x)$ be a primitive polynomial in $R[x]$ where R is a unique factorization domain of characteristic 0. Let $g = \gcd(f, \partial f/\partial x)$. Then $a(x)$ is square-free if and only if $\deg_x(g) = 0$.*

Now we describe the steps of multivariate factorization:

Figure 2.1: **Main steps of Wang's Algorithm**



Example 10. With the choice of $p = 11$, $l = 1$ and $I = [x_2 = 2, x_3 = 5]$ the steps to factor

$$a = 27\,{x_1}^7 x_2 {x_3}^6 + 18\,{x_1}^4 {x_2}^2 {x_3}^8 + \cdots + 48\,{x_3}^4 + 18\,{x_2}^3 + 36\,x_1 + 36 \text{ (48terms)}$$

1. Determine the coefficient bound and factor in $\mathbb{Z}$

$$u(x_1) = \left(15\,{x_1}^3 + 4\,x_1 + 20\right)\left(750\,x_1 + 2503\right)\left(75\,{x_1}^3 + 12\,{x_1}^2 + 2503\right) \in \mathbb{Z}[x_1]$$

2. Determine the leading coefficients and pass to mod p

$$\tilde{u}(x_1) = \left(2\,x_1 - 5\right)\left(-2\,{x_1}^3 + {x_1}^2 - 5\right)\left(4\,{x_1}^3 + 4\,x_1 - 2\right) \in \mathbb{Z}_p[x_1]$$

3. After the first step of MHL in $\mathbb{Z}_p[x_1, x_2]$

$$\left(4\,{x_1}^3 + 2\,{x_2}^3 + 4\,x_1 + 4\right)\left(x_1 x_2 - 5\right)\left(3\,{x_1}^2 {x_2}^2 - 2\,{x_1}^3 - 4\,x_2 + 3\right)$$

4. After the second step of MHL in $\mathbb{Z}_p[x_1, x_2, x_3]$

$$\left(3\,{x_1}^3 x_3 + 2\,{x_2}^3 + 4\,x_1 + 4\right)\left(3\,x_1 x_2 {x_3}^3 + 4\,{x_3}^4 + 3\right)\left(3\,{x_1}^3 {x_3}^2 + 2\,x_2 {x_3}^4 + 3\,{x_1}^2 {x_2}^2 + 3\right)$$

These are in fact the true factors of a.

Step 1. Make the polynomial a primitive and square-free: If a is not primitive the $\mathrm{cont}_{x_1}(a)$ and $\mathrm{pp}_{x_1}(a)$ can be factored separately. Hence we may assume that a is primitive. Let

$$g = \gcd(a(x_1, \ldots, x_n), \partial a/\partial x_1).$$

Note that a has a repeated factor f if and only if $f \mid g$ and a/g is square-free. Then factorization can be continued by factoring a and a/g separately. Thus we may assume a is square-free.

Step 2. Determine the leading coefficients of factors: The leading coefficient of a is $\mathrm{LC}_{x_1}(a) = c_m \in \mathbb{Z}[x_2, \ldots, x_n]$. By a recursive call, c_m is factored over $\mathbb{Z}$. Let

$$c_m = \Omega f_1^{e_1} f_2^{e_2} \cdots f_k^{e_k}$$

where the f_i's are distinct irreducible polynomials of positive degree in $\mathbb{Z}[x_2, \ldots, x_n]$ and Ω is an integer. Let us assume that c_m is not an integer, for that case is trivial. As a next step, a set of integers $\{\alpha_2, \ldots, \alpha_n\}$ will be found satisfying the following conditions:

1. $\tilde{c}_m = c_m(\alpha_2, \ldots, \alpha_n) \neq 0$;
2. $a_1(x_1) = a(x_1, \alpha_2, \ldots, \alpha_n)$ is square-free;
3. For each f_i, $\tilde{f}_i = f_i(\alpha_2, \ldots, \alpha_n)$ has at least one prime divisor p_i which does not divide any $\tilde{f}_j$ for $j < i$, Ω or the content of $a(x_1, \alpha_2, \ldots, \alpha_n) \in \mathbb{Z}[x_1]$.

This can be done without factoring any integers by the Choose Evaluation Points (CEP) algorithm described below. CEP calls the algorithm Choose Candidate Points (CCP) which is also described below. CCP is a slight modification of an algorithm proposed by Musser [Mus71, Mus75] . It chooses α_n until $a(x_1, \ldots, x_{n-1}, \alpha_n)$ has the same degree in x_1 as a and is square-free. In the same way $\alpha_{n-1}, \ldots, \alpha_2$ are chosen so that the evaluated polynomial remains of the same degree and square-free at each stage. Therefore $(\alpha_2, \ldots, \alpha_n)$ satisfies the conditions (1) and (2) of Wang's LCC.

Termination of CCP can be shown by considering step 6: Note that $\tilde{a}(x_1, \ldots, x_k)$ is square-free, so $\gcd(\tilde{a}, \partial\tilde{a}/\partial x_1) \neq 1 \Longrightarrow r = \mathrm{resultant}_{x_1}(\tilde{a}, \partial\tilde{a}/\partial x_1) = 0$. Note that $r \in \mathbb{Z}[x_2, \ldots, x_k]$ and at step 4 α_k is chosen so that $\mathrm{LC}_{x_1}(\tilde{a}(x_1, \ldots, x_{k-1}, \alpha_k)) \neq 0$. Now, if at step 6 $g = \gcd(B, \partial B/\partial x_1)$ with $\deg_{x_1} g > 0$, then B is not square-free and $r(x_k = \gamma) = 0$ as a polynomial in $\mathbb{Z}[x_2, \ldots, x_k]$. If we write $r \in \mathbb{Z}[x_2, \ldots, x_k]$ as $r = \sum_{i=1}^{l} c_i(x_k) M_i$ for some $l \geq 0$, where $M_i \in \mathbb{Z}[x_2, \ldots, x_{k-1}]$ are distinct monomials, then $r = 0 \Leftrightarrow c_i(x_k) = 0$ for each $1 \leq x_k \leq l$. Then $r = 0 \Longleftrightarrow \mathrm{cont}_{[x_2, \ldots, x_{k-1}]}(r)(x_k = \alpha_k) = 0$. Hence the number of choices of $\alpha_k \in \mathbb{Z}$ which do not make $\tilde{a}(x_1, \ldots, x_{k-1}, \alpha_k)$ square-free is finite. (See Example 11 below.)

Algorithm 3 Choose Candidate Points (**CCP**)

Input A square-free polynomial $a \in \mathbb{Z}[x_1, \ldots, x_n]$.

Output Returns an ordered set of candidate evaluation points $\alpha_2, \ldots, \alpha_n$ such that $a(x_1, \alpha_2, \ldots, \alpha_n)$ has the same degree in x_1 as a and is square-free.

Set $\tilde{a} \leftarrow a$ and $d \leftarrow \deg_{x_1}(a)$
for k **from** n **to** 2 **by** -1 **do**
    repeat
        Pick α_k in $\mathbb{Z}$ at random (In practice from a large set.)
        $B \leftarrow \tilde{a}(x_k = \alpha_k)$
        Let $g \leftarrow \gcd(B, \partial B/\partial x_1))$
    until $\mathrm{LC}_{x_1} B \neq 0$, $\deg_{x_1} = d$ and $\deg_{x_1} g = 0$
    Set $\tilde{a} \leftarrow B$
end for
return $(\alpha_2, \ldots, \alpha_n)$

Algorithm 4 Choose Evaluation Points (**CEP**)

Input $a \in \mathbb{Z}[x_1, \ldots, x_n]$.

Output Returns an ordered set of evaluation points $(\alpha_2, \ldots, \alpha_n)$ for MHL and primes $(p_1, \ldots, p_k)$ satisfying the condition (3) of Wang's LCC.

Set $L = (\alpha_2, \ldots, \alpha_n) \leftarrow$ **CCP**(a)
Set $\delta \leftarrow \mathrm{cont}(a(x_1, \alpha_2, \ldots, \alpha_n)) \in \mathbb{Z}$
Factor $\mathrm{LC}_{x_1}(a) \in \mathbb{Z}[x_2, \ldots, x_n]$. Let $\mathrm{LC}_{x_1}(a) = \Omega f_1^{e_1} f_2^{e_2} \cdots f_k^{e_k}$
for i **from** 1 to k **do** $\tilde{f}_i \leftarrow f_i(\alpha_2, \ldots, \alpha_n)$ **end for**
Set $d_0 \leftarrow \delta \cdot \Omega$
for i **from** 1 to k **do**
    $q \leftarrow |\tilde{f}_i|$.
    for j **from** $i-1$ to 0 **by** -1 **do**
        $r \leftarrow d_j$.
        while $r \neq 1$ **do**
            $r \leftarrow \gcd(r, q)$, $q \leftarrow q/r$
            if $q = 1$ **then**
                return CEP(a) *(Restart)*
            end if
        end while
    end for
    $d_i \leftarrow q$
end for
for i **from** 1 to k **do**
    $p_k \leftarrow$ any prime divisor of d_k
end for
return L and $(p_1, \ldots, p_k)$

Example 11. Consider the polynomial f below which is primitive in the variable x_1:

$$f = x_4 {x_1}^2 + 2\,x_1 x_2 {x_4}^2 + 2\,x_1 x_3 x_4 + {x_2}^2 + 2\,x_2 x_3 + {x_3}^2 + 2\,x_1 + 2\,x_2 + 2\,x_3 + x_4.$$

We have $\mathrm{LC}_{x_1}(f) = x_4$ and

$$\partial f/\partial x_1 = 2\,x_2 {x_4}^2 + 2\,x_1 x_4 + 2\,x_3 x_4 + 2.$$

Then $\gcd(f, \partial f/\partial x_1) = 1$, and hence f is a square-free polynomial. Let $r = \mathrm{resultant}_{x_1}(f, \partial f/\partial x_1)$. If we compute r we get

$$r = (-4\,{x_4}^5 + 4\,{x_4}^2){x_2}^2 + (-8\,{x_4}^4 + 8\,{x_4}^2)x_2 x_3 + (-8\,{x_4}^3 + 8\,{x_4}^2)x_2 + (-4\,{x_4}^3 + 4\,{x_4}^2){x_3}^2 + 4\,{x_4}^3 - 4\,x_4.$$

We have $\mathrm{cont}_{[x_2,x_3]}(r) = 4\,x_4(x_4 - 1)$.

In fact, for $x_4 = 1$, $f(x_1, x_2, x_3, 1) = (1 + x_1 + x_2 + x_3)^2$ and for $x_4 = 0$ (which makes $\mathrm{LC}_{x_1}(f)(x_2, x_3, 0) = 0$) one has $f(x_1, x_2, x_3, 0) = {x_2}^2 + 2\,x_2 x_3 + {x_3}^2 + 2\,x_1 + 2\,x_2 + 2\,x_3$ which is square-free. Thus 0 and 1 are not viable choices for α_4.
If we pick $\alpha_4 = -1$ then

$$f(x_1, x_2, x_3, -1) = -{x_1}^2 + 2\,x_1 x_2 - 2\,x_1 x_3 + {x_2}^2 + 2\,x_2 x_3 + {x_3}^2 + 2\,x_1 + 2\,x_2 + 2\,x_3 - 1$$

which is a square-free polynomial and $\mathrm{LC}_{x_1}(f)(x_4 = -1) \neq 0$.

Ideally the random integers at steps 4 of CCP are chosen to be small integers including zero. This is to make the factorization of $a(x_1, \alpha_2, \ldots, \alpha_n) \in \mathbb{Z}[x_1]$ easier. Any possible choice of $\alpha_i = 0$ helps retain the sparseness in the intermediate steps of the multivariate Hensel lifting. This will be described later in Chapter 3 in detail.

As a next step CEP tests whether the choice $(\alpha_2, \ldots, \alpha_n)$ is suitable for choosing primes satisfying the condition (3) of Wang's LCC. It avoids integer factorization and instead uses integer gcd computations since factoring large integers is not easy. It first computes integers d_i for $i = 1, \ldots, k$ which satisfy condition (3). Then any prime divisor of d_i serves as p_i.

Correctness and termination of CEP can be shown easily by considering step 11. Common factors are simply removed by taking gcd's.

Now let $\delta = \mathrm{cont}(a_1(x_1))$ and $u(x) = \mathrm{pp}(a_1(x_1))$. Then $u(x_1)$ is factored over integers [Ber48, Ber67].

$$u(x) = u_1(x_1) \cdots u_r(x_1) \in \mathbb{Z}[x_1].$$

It follows from Hilbert's irreducibility theorem [[Lang62], p.14], that for any irreducible polynomial $h(x_1, \ldots, x_n)$ over $\mathbb{Z}$ the subset $\{(\alpha_2, \ldots, \alpha_n)\} \subset \mathbb{Z}^{n-1}$ of points such that $h(x_1, \alpha_2, \ldots, \alpha_n)$ remains irreducible over $\mathbb{Z}$ is dense. Therefore with high probability a factorization of $u(x) \in \mathbb{Z}[x_1]$ reflects the factorization pattern of a.

Now if none of $u_i(x_1)$, $i = 1, \ldots, r$ is extraneous, then a factors into r distinct irreducible polynomials,

$$a = \prod_{i=1}^{r} g_i(x_1, \ldots, x_n).$$

Let $b_i(x_2, \ldots, x_n) = \mathrm{LC}(g_i)$ and $\tilde{b}_i = b_i(\alpha_2, \ldots, \alpha_n)$. Then

$$g_i(x_1, \alpha_2, \ldots, \alpha_n) = \delta_i u_i(x)$$

for some divisor δ_i of δ. The following observation is central in Wang's LCC.

Lemma 12. *Following the notation above, if there are no extraneous factor, then for all i and m,*

$$f_k^m \mid b_i \Longleftrightarrow \tilde{f}_k^m \mid \mathrm{LC}(u_i)\delta.$$

Proof. If $f_k^m \mid b_i$, then $\tilde{f}_k^m \mid \tilde{b}_i$. We have $g_i(x_1, \alpha_2, \ldots, \alpha_n) = \delta_i u_i(x) \Rightarrow \tilde{b}_i = \delta_i \mathrm{LC}(u_i)$. Hence $\tilde{f}_k^m \mid \mathrm{LC}(u_i)\delta_i \Rightarrow \tilde{f}_k^m \mid \mathrm{LC}(u_i)\delta$. Suppose on the other hand $f_k^m \nmid b_i$. Then note that

$$b_i = \mathrm{LC}(g_i) \mid \prod_{j=1}^{r} \mathrm{LC}(g_j) = \Omega \prod_{j=1}^{k} f_j^{e_j}.$$

Therefore $b_i = \omega f_1^{s_1} f_2^{s_2} \cdots f_k^{sk} \Rightarrow \tilde{b}_i = \omega \tilde{f}_1^{s_1} \tilde{f}_2^{s_2} \cdots \tilde{f}_k^{s_k}$ where $\omega \mid \Omega$, $s_i \geq 0$ and $s_k < m$. Thus by the choice of p_k, $p_k^m \nmid \tilde{b}_i \Rightarrow \tilde{f}_k^m \nmid \mathrm{LC}(u_i)\delta_i \Rightarrow \tilde{f}_k^m \nmid \mathrm{LC}(u_i)\delta$. □

This lemma enables us to distribute all f_k first, then all f_{k-1}, etc. Thus all $d_i(x_2, \ldots, x_n) = \mathrm{pp}(b_i)$ can be determined as products of powers of f_i's. Then it comes to distribute Ω. Let $\tilde{d}_i = d_i(\alpha_2, \ldots, \alpha_n)$. If $\delta = 1$ then $b_i = (\mathrm{LC}(u_i)/\tilde{d}_i)d_i$. Otherwise if $\delta \neq 1$ the following steps are carried out for $i = 1, \ldots, r$.

1. Let $d = \gcd(\mathrm{LC}(u_i), \tilde{d}_i)$ and $b_i = (\mathrm{LC}(u_i)/d)d_i$.
2. Let $u_i = (\tilde{d}_i/d)u_i$.
3. Let $\delta = \delta/(\tilde{d}_i/d)$.

Now if $\delta = 1$ the process ends. Otherwise, let $u_i = \delta u_i, b_i = \delta b_i$ and $a = \delta^{r-1}a$. In this case when the true factors over a are found they may have integer contents which should be removed.

Example 13. Returning to the polynomial of Example 10, consider factoring the following polynomial a below and where we have only the knowledge of the expanded form of it.

$$a = (3\,{x_1}^3 x_3 + 2\,{x_2}^3 + 4\,x_1 + 4)(3\,x_1 x_2 {x_3}^3 + 4\,{x_3}^4 + 3)(3\,{x_1}^3 {x_3}^2 + 2\,x_2 {x_3}^4 + 3\,{x_1}^2 {x_2}^2 + 3).$$

Our aim is to get the correct distribution of the leading coefficients $[3x_3,\ 3x_2x_3^3,\ 3x_3^2]$ where the leading coefficient of a, $\mathrm{LC}_{x_1}(a) = 27x_2x_3^6$.

We have $\Omega = 27, f_1 = x_2, f_2 = x_3$. If we choose $L = [\alpha_2 = 2, \alpha_3 = 5]$ and then factor $a_1 = a(x_1, L)$ over $\mathbb{Z}$ we get

$$a_1 = u(x_1) = (15\,{x_1}^3 + 4\,x_1 + 20)(750\,x_1 + 2503)(75\,{x_1}^3 + 12\,{x_1}^2 + 2503).$$

Note that a_1 is square-free and we have $\delta = \mathrm{cont}(a_1) = 1$, $\tilde{f}_1 = 2, \tilde{f}_2 = 5$ so $p_1 = 2$, $p_2 = 5$ and hence the choice of L satisfies the conditions of Wang's LCC.
We start with the distribution of f_2:

$$\begin{aligned} \tilde{f}_2^3 = 125 \mid \mathrm{LC}(u_2) = 750 &\Rightarrow f_2^3 = x_3^3 \mid b_2 \\ \tilde{f}_2^2 = 25 \mid \mathrm{LC}(u_3) = 75 &\Rightarrow f_2^2 = x_3^2 \mid b_3 \\ \tilde{f}_2 = 5 \mid \mathrm{LC}(u_1) = 15 &\Rightarrow f_2 = x_3 \mid b_1. \end{aligned}$$

We continue with the distribution of f_1:

$$\tilde{f}_1 = 2 \mid \mathrm{LC}(u_2) = 750 \Rightarrow f_1 = x_2 \mid b_2.$$

Therefore we get

$$d_1 = \mathrm{pp}(b_1) = x_3,\ d_2 = \mathrm{pp}(b_2) = x_2 x_3^3,\ d_3 = \mathrm{pp}(b_3) = x_3^2.$$

As a final step

$$\begin{aligned} b_1 &= (\mathrm{LC}(u_1)/\tilde{d}_1)d_1 = (15/5)x_3 = 3x_3 \\ b_2 &= (\mathrm{LC}(u_2)/\tilde{d}_2)d_2 = (750/250)x_2 x_3^3 = 3x_2 x_3^3 \\ b_3 &= (\mathrm{LC}(u_1)/\tilde{d}_1)d_1 = (75/25)x_3^2 = 3x_3^2. \end{aligned}$$

These are the correct leading coefficients of the factors of a.

Actually the idea behind this process is simple. The aim is first to decompose $(\mathrm{LC}(a)/\Omega) = x_2 x_3^6$ into a product of 3 functions images of which will be $[\mathrm{LC}(u_1), \mathrm{LC}(u_2), \mathrm{LC}(u_3)] = [15, 750, 75]$ at the evaluation points $L = [\alpha_2 = 2, \alpha_3 = 5]$. This is done by choosing a prime p that divides $\tilde{f}_i$ which does not divide any $\tilde{f}_j$ with $j \neq i$. Then Ω is distributed in an obvious way. See also [GCL92], Chapter 8, page 377.

Step 3. Construction of multivariate factors:

(i) Coefficient bound: A coefficient bound, B, can be computed such that for any integer coefficient b of any divisor of a, $B > |b|$. The following bound can be used in the choice of a prime to lift the factors [Gel60].

Lemma 14. *Suppose $P_1(x_1, \ldots, x_n), \ldots, P_m(x_1, \ldots, x_n)$ are arbitrary polynomials in n variables with $\|.\|_\infty$ norms $H_1, \ldots, H_m$. Denoting the $\|.\|_\infty$ norm of and the degrees of*

the polynomial $P = \prod_{i=1}^{m} P_i$ *as* H *and* $d_1, \ldots, d_n$ *in the variables* $x_1, \ldots, x_n$ *respectively then the following inequality holds.*

$$H \geq e^{-d} H_1 H_2 \cdots H_m \text{ where } d = \sum_{i=1}^{n} d_i \text{ with e} = \sum_{\text{i}=0}^{\infty} 1/\text{i!}.$$

Corollary 15. *Let* $P_i(x_1, \ldots, x_n)$ *be an irreducible factor of the polynomial* $P(x_1, \ldots, x_n)$ *and the the total degree of* P *be* d*. Then the following inequality holds.*

$$\|P_i\|_\infty \leq e^d \|P\|_\infty.$$

(ii) Construction of factors: Recall that in step 2 we have defined $\delta = \text{cont}(a_1(x_1))$ and $u(x) = \text{pp}(a_1(x_1))$ and then factored $u(x_1)$ over integers.

$$u(x_1) = u_1(x_1) \cdots u_r(x_1) \in \mathbb{Z}[x_1].$$

If $r = 1$, then a is irreducible and we are done. Otherwise, we choose a prime p such that $p^l > 2B$ and $\tilde{u} = u \mod p$ has the same degree as u and is square-free modulo p. Then define $\tilde{u}_k = u_k \mod p$ so that

$$\tilde{u}(x_1) = \tilde{u}_1(x_1) \cdots \tilde{u}_r(x_1) \mod \langle p, I \rangle$$

where $I = \langle x_2 - \alpha_2, \ldots, x_n - \alpha_n \rangle$. This partial factorization of $u \mod p$ is used as a basis for the multivariate Hensel lifting (MHL) to get relatively prime polynomials $U_i(x_1, \ldots, x_n)$ for $i = 1, \ldots, r$ where $U_i = \tilde{u}_i \mod \langle p, I \rangle$ satisfying

$$a(x_1, \ldots, x_n) = U_1(x_1, \ldots, x_n) \cdots U_r(x_1, \ldots, x_n) \mod \left\langle p^l, I^h \right\rangle$$

where $h = 1 + \deg_{(x_2, \ldots, x_n)}(a)$. Details of MHL will be described in detail in Section 2.2.

The univariate factoring algorithm determines some primes in the process of factoring u and can be used as a prime p as described above. But it is not necessary to choose p equal to this prime. In fact, as indicated in [Mus75], it is better to choose a larger prime (since we are not looking for the complete factorization over $\mathbb{Z}_p$) to reduce the number of Hensel construction iterations. We could in fact choose p large enough to eliminate entirely the phase of construction which lifts from p to p^l but this might mean that p would be a multiple precision integer. The best approach seems to be to choose p as big as possible while constrained to be a single precision integer [Mus75]. The architecture of modern computers now allows us to choose a 64 bit prime (or 63 bit prime if signed 64 bits integers are used) which is big enough to handle the coefficients of the polynomial a to be factored for many practical multivariate problems.

(iii) Finding actual factors: If a is monic (with respect to the main variable x_1) then any irreducible factor of a over $\mathbb{Z}$ either is equal to some U_i or is equal to the product

of two or more U_i's $\mod \left\langle p^l, I^h \right\rangle$. If a is not monic these equivalences are up to the units in the coefficient domain of a. In any case, the irreducible factors of a are found from combinations of these U_i's by trial divisions.

2.2 An incremental design of Multivariate Hensel Lifting (MHL)

In this section we will give the mathematical details of Multivariate Hensel Lifting (MHL) which is the core of multivariate factorization. Then we will discuss Wang's incremental design of MHL. We start by the main theorem.

Theorem 16. *(Multivariate Hensel construction). Let p be a prime number, l be a positive integer. Suppose $a, u, w \in \mathbb{Z}_{p^l}[x_1, \ldots, x_j]$ where $p \nmid \mathrm{lcoeff}(\phi_I(a))$ and $I = \langle x_2 - \alpha_2, \ldots, x_j - \alpha_j \rangle$ is an ideal in $\mathbb{Z}_{p^l}[x_1, \ldots, x_j]$ with $\alpha_i \in \mathbb{Z}$. Let $u^{(1)}(x_1), w^{(1)}(x_1) \in \mathbb{Z}_{p^l}[x_1]$ be two univariate polynomials in $\mathbb{Z}_{p^l}[x_1]$ satisfying*

$$(i)\, a(x_1, \ldots, x_j) = u^{(1)}(x_1) w^{(1)}(x_1) \bmod \left\langle I, p^l \right\rangle \text{ with}$$

$$(ii)\ \gcd\left(\phi_p(u^{(1)}(x_1)), \phi_p(w^{(1)}(x_1))\right) = 1 \,\mathrm{in} \mathbb{Z}_p[x_1].$$

Then for any $k \geq 1$, there exist $u^{(k)}, w^{(k)} \in \mathbb{Z}_{p^l}[x_1, \ldots, x_j]/I^k$ such that

$$(iii)\, a(x_1, \ldots, x_j) = u^{(k)} w^{(k)} \bmod \left\langle I^k, p^l \right\rangle \text{ and}$$

$$(iv)\, u^{(k)} = u^{(1)}(x_1) \bmod \left\langle I, p^l \right\rangle,\ w^{(k)} = w^{(1)}(x_1) \bmod \left\langle I, p^l \right\rangle.$$

Proof. The proof is by induction on k. The case $k = 1$ is given by the condition (i). For $k \Rightarrow k+1$: Suppose that we have $u^{(k)}, w^{(k)} \in \mathbb{Z}_{p^l}[x_1, \ldots, x_j]/I^k$ satisfying the conditions (iii) and (iv). We define the error

$$e^{(k)} = a(x_1, \ldots, x_j) - u^{(k)} w^{(k)} \in \mathbb{Z}_{p^l}[x_1, \ldots, x_j]/I^k.$$

Then we have $e^{(k)} \in I^k$. Note that I^k is generated by all possible k-combinations of $(x_i - \alpha_i)$ for $2 \leq i \leq k$. Then for some $c_{\mathbf{i}}(x_1) \in \mathbb{Z}_{p^l}[x_1]$ where the index $\mathbf{i}$ is defined as $\mathbf{i} = (i_1, i_2, \ldots, i_k)$ we have

$$e^{(k)} = \sum_{i_1=2}^{j} \sum_{i_2=i_1}^{j} \cdots \sum_{i_k=i_{k-1}}^{j} c_{\mathbf{i}}(x_1)(x_{i_1} - \alpha_{i_1})(x_{i_2} - \alpha_{i_2}) \cdots (x_{i_k} - \alpha_{i_k})$$

Since $p \nmid \mathrm{lcoeff}(\phi_I(a))$, by using Theorem 8, for each $\mathbf{i}$ we can find unique pairs $\sigma_{\mathbf{i}}(x_1), \tau_{\mathbf{i}}(x_1) \in \mathbb{Z}_{p^l}[x_1]$ such that

$$\sigma_{\mathbf{i}}(x_1) u^{(1)}(x_1) + \tau_{\mathbf{i}}(x_1) w^{(1)}(x_1) = c_{\mathbf{i}}(x_1)$$

where $\deg_{x_1}(\sigma_{\mathbf{i}}(x_1)) < \deg_{x_1}(c_{\mathbf{i}}(x_1))$. Now define

$$\begin{aligned} u^{(k+1)} &= u^{(k)} + \sum_{i_1=2}^{j} \sum_{i_2=i_1}^{j} \cdots \sum_{i_k=i_{k-1}}^{j} \tau_{\mathbf{i}}(x_1)(x_{i_1} - \alpha_{i_1})(x_{i_2} - \alpha_{i_2}) \cdots (x_{i_k} - \alpha_{i_k}) \\ w^{(k+1)} &= w^{(k)} + \sum_{i_1=2}^{j} \sum_{i_2=i_1}^{j} \cdots \sum_{i_k=i_{k-1}}^{j} \sigma_{\mathbf{i}}(x_1)(x_{i_1} - \alpha_{i_1})(x_{i_2} - \alpha_{i_2}) \cdots (x_{i_k} - \alpha_{i_k}). \end{aligned}$$

Then, we compute $u^{(k+1)}w^{(k+1)}$ and pass to modulo $I^{(k)}$ and get

$$\begin{aligned} u^{(k+1)}w^{(k+1)} &= u^{(k)}w^{(k)} \\ &+ \sum_{i_1=2}^{j} \sum_{i_2=i_1}^{j} \cdots \sum_{i_k=i_{k-1}}^{j} \left(\sigma_{\mathbf{i}}(x_1)u^{(1)}(x_1) + \tau_{\mathbf{i}}(x_1)w^{(1)}(x_1)\right)(x_{i_1} - \alpha_{i_1})(x_{i_2} - \alpha_{i_2}) \cdots (x_{i_k} - \alpha_{i_k}) \\ &\qquad \mod (I^{k+1}, p^l). \end{aligned}$$

Hence

$$u^{(k+1)}w^{(k+1)} = u^{(k)}w^{(k)} + e^{(k)} = a(x_1, \ldots, x_j) \mod (I^{k+1}, p^l).$$

By definition of $u^{(k+1)}, w^{(k+1)}$ above we have

$$u^{(k+1)} = u^{(1)}(x_1) \bmod \left\langle I, p^l \right\rangle,\ w^{(k+1)} = w^{(1)}(x_1) \bmod \left\langle I, p^l \right\rangle.$$

□

The idea of the constructive proof of Theorem 16 was used in multivariate polynomial factorization by Wang and Rotschild [WR75]. One major difficulty in this approach is the intermediate expression swell when we compute the error term where we need to use non-zero's for α_i's. Let $I = \langle x_2 - 1, x_3 - 2, x_4 - 3 \rangle$ and consider the I-adic expansion of the monomial $x_2^r x_3^s x_4^t$. It has $(r+1)(s+1)(t+1)$ many terms. So when non-zero evaluation points are used an intermediate expression swell is inevitable. However it is not always possible to choose the evaluations points to be zero, because in applications of the MHL for multivariate polynomial factorization, the leading coefficient must not vanish under the evaluation homomorphism. As one may guess, for many applications the leading coefficient is a monomial which maps to zero for any choice of zero for one of its indeterminates.

A construction was proposed by Wang in [Wan78] behaves better in comparison with the previous approach. Before we explain it, let us state and prove the uniqueness of the multivariate Hensel construction.

Corollary 17. *(Uniqueness of the multivariate Hensel construction). In Theorem 16, if the given polynomial $a \in \mathbb{Z}_{p^l}[x_1, \ldots, x_j]$ is monic in the variable x_1 and correspondingly if the univariate factors $u^{(1)}(x_1), w^{(1)}(x_1) \in \mathbb{Z}_{p^l}[x_1]$ are chosen to be monic, then for any integer*

$k \geq 1$ conditions $(i), (ii)$ of the Theorem 13 uniquely determine the factors $u^{(k)}, w^{(k)} \in \mathbb{Z}_{p^l}[x_1, \ldots, x_j]/I^k$ which are monic in the variable x_1.

Proof. We follow the idea of the proof of Corollary 6 in Chapter 1 and use induction on k. Let

$$(u_1^{(k)}, w_1^{(k)}), (u_2^{(k)}, w_2^{(k)}) \in \mathbb{Z}_{p^l}[x_1, \ldots, x_j]/I^k$$

be two monic solution pairs in the variable x_1 which satisfy the conditions (i) and (ii) of the Theorem 16.

For the case $k = 1$ note that $\mathbb{Z}_{p^l}[x_1, \ldots, x_j]/I = \mathbb{Z}_{p^l}[x_1, \ldots, x_j]/\langle x_2 - \alpha_2, \ldots, x_j - \alpha_j\rangle \cong \mathbb{Z}_{p^l}[x_1]$. So the uniqueness follows from the uniqueness of univariate Hensel construction given by Corollary 6.

For the case $k \Rightarrow k+1$: Let $(u^{(k)}, w^{(k)}) \in \mathbb{Z}_{p^l}[x_1, \ldots, x_j]/I^k$ be a solution pair. Note that

$$a(x_1, \ldots, x_j) = u^{(k+1)}w^{(k+1)} \bmod \left\langle I^{k+1}, p^l \right\rangle \Rightarrow a(x_1, \ldots, x_j) = u^{(k+1)}w^{(k+1)} \bmod \left\langle I^k, p^l \right\rangle$$

Then by the induction assumption on uniqueness we have

$$u^{(k+1)} = u^{(k)} \bmod \left\langle I^k, p^l \right\rangle \text{ and } w^{(k+1)} = w^{(k)} \bmod \left\langle I^k, p^l \right\rangle.$$

Then $u^{(k+1)} - u^{(k)}$ and $w^{(k+1)} - w^{(k)}$ are both in I^k. Hence for some for some pairs $\sigma_{\mathbf{i}}(x_1), \tau_{\mathbf{i}}(x_1) \in \mathbb{Z}_{p^l}[x_1]$ where the index $\mathbf{i}$ is defined as $\mathbf{i} = (i_1, i_2, \ldots, i_k)$, we can write

$$\begin{aligned} u^{(k+1)} &= u^{(k)} + \sum_{i_1=2}^{j} \sum_{i_2=i_1}^{j} \cdots \sum_{i_k=i_{k-1}}^{j} \tau_{\mathbf{i}}(x_1)(x_{i_1} - \alpha_{i_1})(x_{i_2} - \alpha_{i_2}) \cdots (x_{i_k} - \alpha_{i_k}) \\ w^{(k+1)} &= w^{(k)} + \sum_{i_1=2}^{j} \sum_{i_2=i_1}^{j} \cdots \sum_{i_k=i_{k-1}}^{j} \sigma_{\mathbf{i}}(x_1)(x_{i_1} - \alpha_{i_1})(x_{i_2} - \alpha_{i_2}) \cdots (x_{i_k} - \alpha_{i_k}). \end{aligned}$$

Now if for some index $\mathbf{i}$, $\deg_{x_1}(\sigma_{\mathbf{i}}) \geq \deg_{x_1}(w_1^{(k)})$ then $\mathrm{LC}_{x_1}(w^{(k+1)}) \neq 1$ which is against the induction hypothesis. The argument is similar for $\tau_{\mathbf{i}}(x_1)$. Hence for all indices $\mathbf{i}$, we have $\deg_{x_1}(\sigma_{\mathbf{i}}) < \deg_{x_1}(w_1^{(k)})$ and $\deg_{x_1}(\tau_{\mathbf{i}}) < \deg_{x_1}(u_1^{(k)})$. Therefore the leading coefficient of $w^{(k+1)}$ in x_1, is the leading coefficient of $w^{(1)}$ in x_1 which is 1. Similarly for $u^{(k+1)}$. (**This observation is important and indicates that Yun's replaceLC procedure can be applied to the multivariate case as well**).

It remains to show the uniqueness of $\sigma_{\mathbf{i}}(x_1), \tau_{\mathbf{i}}(x_1) \in \mathbb{Z}_{p^l}[x_1]$. But this is easy by considering $u^{(k+1)}w^{(k+1)}$ as in the proof of Theorem 16 above and using Theorem 8 which states that the solution to the generalized univariate diophantine problem has unique solutions $\sigma_{\mathbf{i}}(x_1), \tau_{\mathbf{i}}(x_1) \in \mathbb{Z}_{p^l}[x_1]$ where $\deg_{x_1}(\sigma_{\mathbf{i}}) < \deg_{x_1}(w_1^{(k)})$ and $\deg_{x_1}(\tau_{\mathbf{i}}) < \deg_{x_1}(u_1^{(k)})$. □

We can now describe Wang's incremental design of MHL in [Wan78] to find $u^{(k)}$, $w^{(k)}$. It behaves much better when the factors to be computed are sparse. The idea is to lift the solution over $\mathbb{Z}_p[x_1]$ to $\mathbb{Z}_{p^l}[x_1]$ and then lift the solutions one variable at a time

$$\begin{aligned} \text{from } \mathbb{Z}_{p^l}[x_1] \quad &\text{to} \quad \mathbb{Z}_{p^l}[x_1, x_2] \\ \text{from } \mathbb{Z}_{p^l}[x_1, x_2] \quad &\text{to} \quad \mathbb{Z}_{p^l}[x_1, x_2, x_3] \\ &\vdots \\ \text{then from } \mathbb{Z}_{p^l}[x_1, \ldots, x_{j-1}] \quad &\text{to} \quad \mathbb{Z}_{p^l}[x_1, \ldots, x_j] \end{aligned}$$

Assuming that a polynomial solution is known to exist, the idea is to use Newton's iteration. Consider the $(x_j - \alpha_j)$-adic representation of the solution polynomial

$$u = u^{(1)} + u_1(x_j - \alpha_j) + u_2(x_j - \alpha_j)^2 + \cdots + u_k(x_j - \alpha_j)^k + \cdots$$

with $u^{(1)}, u_j \in \mathbb{Z}_{p^l}[x_1, \ldots, x_{j-1}]$. We define

$$u^{(k)} := u^{(1)} + u_1(x_j - \alpha_j) + u_2(x_j - \alpha_j)^2 + \cdots + u_{k-1}(x_j - \alpha_j)^{k-1}.$$

At this point we want to give an example to emphasize how important the choice of the evaluation point is. Let $p = 31$ and let

$$f = x^4 - xyz^2 - z^4.$$

be the factor to be computed. Notice f is sparse in z. The solutions to the MDP's are the Taylor coefficients of the factors. This means if $z = 0$ is the evaluation then at the 3^{rd} step of MHL the solutions to the MDP's will be simply the coefficients of f in z, namely, $-1, 0, -xy$ and 0. That is, one needs to solve only two MDP's. This is why, for Wang's design of MHL zero evaluation points are desirable. However it is not always possible to choose 0 as an evaluation point. For $z = 5$ the Taylor expansion of f is

$$\begin{aligned} f \;=\; & \underbrace{x^4 + 6\,xy - 5}_{\text{given } f_2} + \underbrace{(-10\,xy - 4)}_{\text{Solution to the 1st MDP}} \quad (z-5) \\ + \; & \underbrace{(-xy + 5)}_{\text{Solution to the 2nd MDP}} \quad (z-5)^2 \\ + \; & \underbrace{(+11)}_{\text{Solution to the 3rd MDP}} \quad (z-5)^3 \\ + \; & \underbrace{(-1)}_{\text{Solution to the 4th MDP}} \quad (z-5)^4 \end{aligned}$$

which is no longer sparse in z and one needs to solve 4 MDP's.

We continue our discussion. We have

$$u^{(k)} = u \bmod (x_j - \alpha_j)^k.$$

Let us define

$$u^{(k+1)} := u^{(k)} + \triangle u^{(k)},\ \triangle u^{(k)} := u_k (x_j - \alpha_j)^k$$

$$w^{(k+1)} := w^{(k)} + \triangle w^{(k)},\ \triangle w^{(k)} := w_k (x_j - \alpha_j)^k.$$

where $u_k, w_k \in \mathbb{Z}_{p^l}[x_1, \ldots, x_{j-1}]$.

To compute $u^{(k+1)}$and $w^{(k+1)}$from $u^{(k)}$and $w^{(k)}$, we first define $F(u, w) = a(x_1, \ldots, x_n) - uw \in \mathbb{Z}[x_1, \ldots, x_n][u, w]$. As before we apply Newton iteration to the function F and get

$$F(u^{(k)} + \triangle u^{(k)}, w^{(k)} + \triangle w^{(k)}) = F(u^{(k)}, w^{(k)}) + F_u(u^{(k)}, w^{(k)}) \triangle u^{(k)} + F_w(u^{(k)}, w^{(k)}) \triangle w^{(k)} + E$$

where the term E contains terms $(x_j - \alpha_j)^s$ with $s \geq k + 1$. Observe that $F(u, w) = 0 \bmod (x_j - \alpha_j)^k \Rightarrow F(u^{(k)}, w^{(k)}) = 0 \bmod (x_j - \alpha_j)^k$ for all $k \geq 1$. So, passing to modulo $(x_j - \alpha_j)^{k+1}$ we get

$$F_u(u^{(k)}, w^{(k)}) \triangle u^{(k)} + F_w(u^{(k)}, w^{(k)}) \triangle w^{(k)} = -F(u^{(k)}, w^{(k)})$$

where $\triangle u^{(k)}$ and $\triangle w^{(k)}$ are to be solved. Let $e^{(k)}(x_1, \ldots, x_n) = a(x_1, \ldots, x_n) - u^{(k)} w^{(k)}$. We have $F_u = -w$ and $F_w = -u$. Hence we get

$$w(x_1, \ldots, x_j) u_k(x_1, \ldots, x_{j-1})(x_j - \alpha_j)^k + u(x_1, \ldots, x_j) w_k(x_1, \ldots, x_{j-1})(x_j - \alpha_j)^k = e^{(k)}(x_1, \ldots, x_n)$$

This shows that at the k^{th} step $(x_j - \alpha_j)^k$ divides $e^k(x_1, \ldots, x_n)$ and we obtain

$$w(x_1, \ldots, x_j) u_k(x_1, \ldots, x_{j-1}) + u(x_1, \ldots, x_j) w_k(x_1, \ldots, x_{j-1}) = e^{(k)}(x_1, \ldots, x_n)/(x_j - \alpha_j)^k.$$

Evaluating this equation at $x_j = \alpha_j$ we get

$$w(x_1, \ldots, x_{j-1}, \alpha_j) u_k(x_1, \ldots, x_{j-1}) + u(x_1, \ldots, x_{j-1}, \alpha_j) w_k(x_1, \ldots, x_{j-1}) = e_k^{(k)} \qquad (2.1)$$

where $e_k^{(k)}$ is the k^{th} Taylor coefficient of $e^{(k)}$, that is

$$e_k^{(k)} = \frac{e^{(k)}(x_1, \ldots, x_n)}{(x_j - \alpha_j)^k}|_{x_j = \alpha_j}$$

Suppose that $a \in \mathbb{Z}_{p^l}[x_1, \ldots, x_{j-1}]$ is monic in x_1 and we have the factorization of

$$a(x_1,\ldots,x_{j-1},\alpha_j) = u(x_1,\ldots,x_{j-1},\alpha_j)w(x_1,\ldots,x_{j-1},\alpha_j).$$

Then if we can solve Eqn (2.1), we can lift $u(x_1,\ldots,x_{j-1},\alpha_j)$ and $w(x_1,\ldots,x_{j-1},\alpha_j)$ to get $a(x_1,\ldots,x_{j-1},x_j) = u(x_1,\ldots,x_{j-1},x_j)w(x_1,\ldots,x_{j-1},x_j)$ as follows:

1. We start with $u_0 := u(x_1,\ldots,x_{j-1},\alpha_j)$,$w_0 := w(x_1,\ldots,x_{j-1},\alpha_j)$ and compute $e^{(1)} = a(x_1,\ldots,x_j) - u_0w_0$.
2. Then we compute $e_1^{(1)} = e^{(1)}/(x_j-\alpha_j)|_{x_j=\alpha_j}$ and solve the MDP $u_0w_1 + w_0u_1 = e_1^{(1)}$ for $u_1, w_1 \in \mathbb{Z}_p[x_1,\ldots,x_{j-1}]$.
3. Then we get $u^{(1)} = u_0 + u_1(x_j-\alpha_j)$ and $w^{(1)} = w_0 + w_1(x_j-\alpha_j)$.
4. The next step is to compute $e^{(2)} = a - u^{(1)}w^{(1)}$. If $e^{(2)} = 0$ then we have the factorization.
5. Else we compute $e_2^{(2)} = e^{(2)}/(x_j-\alpha_j)^2|_{x_j=\alpha_j}$ and solve the MDP $u_0w_2 + w_0u_2 = e_2^{(2)}$ for $u_2, w_2 \in \mathbb{Z}_p[x_1,\ldots,x_{j-1}]$.
6. Then we get $u^{(2)} = u^{(1)} + u_2(x_j-\alpha_j)^2$ and $w^{(2)} = w^{(1)} + w_2(x_j-\alpha_j)^2$.
7. The next step is to compute $e^{(3)} = a - u^{(2)}w^{(2)}$...

We follow these steps above finitely many times until we get $e^{(m)} = 0$ for some $m \geq 0$ or we stop when we exceed a bound. Algorithm 5 below describes the j^{th} step of MHL for $j > 1$ and when the inputs are monic.

Eqn (2.1) is an example of a multivariate diophantine problem (MDP) which we will investigate in Section 2.3 in detail. Hence Wang's incremental design of MHL is reduced to finding an effective solution to the MDP problem.

Example 18. Suppose we seek to factor $a = fg$ where $f = {x_1}^5 + 3\,{x_1}^2x_2{x_3}^2 - 7\,{x_1}^4 - 4\,x_1x_3 + 1$ and $g = {x_1}^5 + {x_1}^2x_2x_3 - 7\,{x_3}^4 - 6$. Let $\alpha_3 = 2$ and $p = 2^{31}-1, l = 1$. Before lifting we have

$$\begin{aligned} f^{(0)} &:= f(x_3=2) = {x_1}^5 - 7\,{x_1}^4 + 12\,{x_1}^2x_2 - 8\,x_1 + 1 \\ g^{(0)} &:= g(x_3=2) = {x_1}^5 + 2\,{x_1}^2x_2 - 118. \end{aligned}$$

In the following all operations will be in mod p and $e_k^{(k)}$ denotes the coefficient of $(x_3-2)^k$ in the Taylor expansion of the error $e^{(k)}$ about $x_3 = 2$.

Let also $f_0 := f^{(0)}, g_0 := g^{(0)}, f^{(k)} := \sum_{i=0}^k f_i(x_3-2)^i, g^{(k)} := \sum_{i=0}^k g_i(x_3-2)^i$.
Step i = 1: $e^{(1)} = a - f^{(0)}g^{(0)}$. Then we compute the Taylor coefficient

$$\begin{aligned} e_1^{(1)} = 13\,{x_1}^7x_2 - 7\,{x_1}^6x_2 - 4\,{x_1}^6 + 36\,{x_1}^4{x_2}^2 - 224\,{x_1}^5 \\ + 1568\,{x_1}^4 - 16\,{x_1}^3x_2 - 4103\,{x_1}^2x_2 + 2264\,x_1 - 224 \end{aligned}$$

and solve the MDP $f_0 g_1 + g_0 f_1 = e_1^{(1)}$ for (f_1, g_1). The solution pair is (the next Chapter will show how to find f_1 and g_1). $(f_1, g_1) = (12\,{x_1}^2 x_2 - 4\,x_1, 12\,{x_1}^2 x_2 - 4\,x_1)$. Now we update

$$\begin{aligned} f^{(1)} &= f^{(0)} + f_1 \cdot (x_3 - 2) \\ &= {x_1}^5 - 7\,{x_1}^4 + 12\,{x_1}^2 x_2 x_3 - 12\,{x_1}^2 x_2 - 4\,x_1 x_3 + 1 \\ g^{(1)} &= g^{(0)} + g_1 \cdot (x_3 - 2) = {x_1}^5 + {x_1}^2 x_2 x_3 - 224\,x_3 + 330 \end{aligned}$$

Step i = 2: $e^{(2)} = a - f^{(1)} g^{(1)}$. Then we compute the Taylor coefficient

$$e_2^{(2)} = 3\,{x_1}^7 x_2 + 6\,{x_1}^4 {x_2}^2 - 168\,{x_1}^5 + 1176\,{x_1}^4 - 2370\,{x_1}^2 x_2 + 1344\,x_1 - 168.$$

and solve the MDP $f_0 g_2 + g_0 f_2 = e_2^{(2)}$ for (f_2, g_2). The solution pair is $(f_2, g_2) = (3\,{x_1}^2 x_2, -168)$. Now we update

$$\begin{aligned} f^{(2)} &= f^{(1)} + f_2 \cdot (x_3 - 2)^2 = {x_1}^5 + 3\,{x_1}^2 x_2 {x_3}^2 - 7\,{x_1}^4 - 4\,x_1 x_3 + 1 \\ g^{(2)} &= g^{(1)} + g_2 \cdot (x_3 - 2)^2 = {x_1}^5 + {x_1}^2 x_2 x_3 - 168\,{x_3}^2 + 448\,x_3 - 342 \end{aligned}$$

In fact, at the end of the 2^{nd} iteration we have recovered f already. $g = a/f$ can be obtained by trial division. Maple does this. But let's go further.
Step i = 3: $e^{(3)} = a - f^{(2)} g^{(2)}$. Then we compute the Taylor coefficient

$$e_3^{(3)} = -56\,{x_1}^5 + 392\,{x_1}^4 - 672\,{x_1}^2 x_2 + 448\,x_1 - 56.$$

and solve the MDP $f_0 g_3 + g_0 f_3 = e_3^{(3)}$ for (f_3, g_3). The solution pair is $(f_3, g_3) = (0, -56)$. Now we update

$$\begin{aligned} f^{(3)} &= f^{(2)} + f_3 \cdot (x_3 - 2)^3 = {x_1}^5 + 3\,{x_1}^2 x_2 {x_3}^2 - 7\,{x_1}^4 - 4\,x_1 x_3 + 1 \\ g^{(3)} &= g^{(2)} + g_3 \cdot (x_3 - 2)^3 = {x_1}^5 + {x_1}^2 x_2 x_3 - 56\,{x_3}^3 + 168\,{x_3}^2 - 224\,x_3 + 106. \end{aligned}$$

Step i = 4 : $e^{(4)} = a - f^{(3)} g^{(3)}$. Then we compute the Taylor coefficient

$$e_4^{(4)} = -7\,{x_1}^5 + 49\,{x_1}^4 - 84\,{x_1}^2 x_2 + 56\,x_1 - 7$$

and solve the MDP $f_0 g_4 + g_0 f_4 = e_4^{(4)}$. The solution pair is $(f_4, g_4) = (0, -7)$. Now we update

$$\begin{aligned} f^{(4)} &= f^{(3)} + f_4 \cdot (x_3 - 2)^4 = {x_1}^5 + 3\,{x_1}^2 x_2 {x_3}^2 - 7\,{x_1}^4 - 4\,x_1 x_3 + 1 \\ g^{(4)} &= g^{(3)} + g_4 \cdot (x_3 - 2)^4 = {x_1}^5 + {x_1}^2 x_2 x_3 - 7\,{x_3}^4 - 6 \end{aligned}$$

Algorithm 5 j^{th} step of Multivariate Hensel Lifting for $j > 1$.

Input : $\alpha_j \in \mathbb{Z}_{p^l}$, $a_j \in \mathbb{Z}_{p^l}[x_1, \ldots, x_j]$, $f_{j-1}, g_{j-1} \in \mathbb{Z}_{p^l}[x_1, \ldots, x_{j-1}]$ where a_j, f_{j-1}, g_{j-1} are monic in x_1 and $a_j(x_j = \alpha_j) = f_{j-1}g_{j-1}$.

Output : $f_j, g_j \in \mathbb{Z}_{p^l}[x_1, \ldots, x_j]$ such that $a_j = f_j g_j$.

$\sigma_{j0} \leftarrow f_{j-1}$, $\tau_{j0} \leftarrow g_{j-1}$, $\sigma_j \leftarrow \sigma_{j0}$, $\tau_j \leftarrow \tau_{j0}$, $monomial \leftarrow 1$
$error \leftarrow a_j - f_{j-1}\, g_{j-1}$
for i from 1 to $\deg(a_j, x_j)$ **while** $error \neq 0$ **do**
    $monomial \leftarrow monomial \times (x_j - \alpha_j)$
    $c \leftarrow$ Taylor coefficient of $(x_j - \alpha_j)^i$ of $error$ at $x_j = \alpha_j$
    if $c \neq 0$ **then**
        Solve the MDP $\sigma_{ji}\tau_{j0} + \tau_{ji}\sigma_{j0} = c$ in $\mathbb{Z}_{p^l}[x_1, \ldots, x_{j-1}]$ for σ_{ji} and τ_{ji}.
        $(\sigma_j, \tau_j) \leftarrow (\sigma_j + \sigma_{ji} \times monomial,\ \tau_j + \tau_{ji} \times monomial)$.
        $error \leftarrow a_j - \sigma_j\, \tau_j$.
    end if
end for
$f_j \leftarrow \sigma_j$ and $g_j \leftarrow \tau_j$

Step i = 5: $e^{(5)} = a - f^{(4)}g^{(4)} = 0$ and we have the factors! As explained in Section 1.2 following Yun's idea, using the **replaceLC** procedure the leading coefficient correction can be handled outside of the for loop for the non-monic case. So to make the explanation easier, in the Algorithm 5 below, the j^{th} step of Multivariate Hensel Lifting for $j > 1$ is described for the monic case.

2.3 The Multivariate Diophantine Problem (MDP)

As seen in Section 2.2, Eqn (2.1) is an example of a multivariate diophantine problem which we describe below in detail. We seek now an effective solution to the MDP problem.

Definition 19. (MDP) Let p be a prime number. Let $j, l \geq 1$ and $u, w, c \in \mathbb{Z}_{p^l}[x_1, \ldots, x_j]$. Let $I_j = \langle x_2 - \alpha_2, \ldots, x_j - \alpha_j \rangle$ be an ideal of $\mathbb{Z}_{p^l}[x_1, \ldots, x_j]$ with $\alpha_i \in \mathbb{Z}_{p^l}$ (not distinct). Let $\phi_{\langle I_j, p\rangle} : \mathbb{Z}_{p^l}[x_1, \ldots, x_j] \to \mathbb{Z}_p[x_1]$ be an evaluation homomorphism and $p \nmid \mathrm{LC}_{x_1}(\phi_{I_j}(uw))$. Let d be the maximum total degree of u and w wrt. the variables $x_2, \ldots, x_j$. Let also $u^{(1)}(x_1) = \phi_{\langle I_j, p\rangle}(u)$, $w^{(1)}(x_1) = \phi_{\langle I_j, p\rangle}(w)$. Then the MDP problem is to find multivariate polynomials $\sigma, \tau \in \mathbb{Z}_{p^l}[x_1, \ldots, x_j]$ such that

$$\sigma u + \tau w \;\; = \;\; c \mod \left\langle I_j^{d+1},\, p^l \right\rangle$$

with $\deg_{x_1}(\sigma) < \deg_{x_1}(w)$, where it is given that

$$\gcd\left(u^{(1)}(x_1), w^{(1)}(x_1)\right) = 1 \,\text{in}\, \mathbb{Z}_p[x_1].$$

The case where $j = l = 1$ is the univariate diophantine problem considered in Section 1.3, Theorem 3.

The case $j = 1$, $l > 1$ is the generalized univariate diophantine problem considered in Section 1.6, Theorem 8.

However the solution is not that obvious for the cases where $j > 1$. In Chapter 3 we will have a closer look at different ways of solving an MDP. Before we close this Chapter, let us establish the uniqueness of the solution and the fact that as long as the gcd condition is satisfied the solution to the MDP is independent of the choice of ideal.

Proof. Consider the case where $l = 1$: Let $I_j = \langle x_2 - \alpha_2, \ldots, x_j - \alpha_j \rangle, K_j = \langle x_2 - \beta_2, \ldots, x_j - \beta_j \rangle$ be ideals of $\mathbb{Z}_p[x_1, \ldots, x_j]$ with $\alpha_i, \beta_j \in \mathbb{Z}_{p^l}$ and suppose one has two solution pairs $(\sigma, \tau), (\tilde{\sigma}, \tilde{\tau})$ to the MDP defined for an ideals I_j, K_j resp. with the same initial conditions:

$$(i)\ \deg_{x_1}(\sigma) < \deg_{x_1}(w) \quad \text{and} \quad \gcd\left(\phi_{\langle I_j,p\rangle}(u), \phi_{\langle I_j,p\rangle}(w)\right) = 1 \in \mathbb{Z}_p[x_1]$$
$$(ii)\ \deg_{x_1}(\tilde{\sigma}) < \deg_{x_1}(w) \quad \text{and} \quad \gcd\left(\phi_{\langle K_j,p\rangle}(u), \phi_{\langle K_j,p\rangle}(w)\right) = 1 \in \mathbb{Z}_p[x_1]$$

Let $g = \gcd(u, w)$. We first claim that if the MDP conditions satisfied, then $g \in \mathbb{Z}_p[x_2, \ldots, x_j]$:

Since $g \mid u$, there exists $s \in \mathbb{Z}_p[x_1, \ldots, x_j]$ such that $gs = u$. Let by LC, we denote the leading coefficient in the variable x_1. We have $\mathrm{LC}(gs) = \mathrm{LC}(u) \Rightarrow \mathrm{LC}(g)\mathrm{LC}(s) = \mathrm{LC}(u)$. So, $\mathrm{LC}(g) \mid \mathrm{LC}(u)$. So $p \nmid \mathrm{LC}_{x_1}(\phi_I(g))$. Let's set $Y = [x_2, \ldots, x_j]$ and let $d_g = \deg_{x_1}(g)$. Then $g(x_1, Y) = \mathrm{LC}_{x_1}(g)x_1^{d_g} + h(x_1, Y)$ with $\deg_{x_1}(h) < d_g$ or $d_g = 0$, i.e. $g \in \mathbb{Z}_p[x_2, \ldots, x_j]$. On the other hand, since $p \nmid \mathrm{LC}_{x_1}(\phi_I(g))$ one has $g \mid u \Rightarrow \phi_{\langle I_j,p\rangle}(g) \mid \phi_{\langle I_j,p\rangle}(u)$. The same is true for w. Hence we have $\phi_{\langle I_j,p\rangle}(g) \mid \gcd(\phi_{\langle I_j,p\rangle}(u), \phi_{\langle I_j,p\rangle}(w)) = 1$. Again, since $p \nmid \mathrm{LC}_{x_1}(\phi_I(g))$ it follows that $\phi_{\langle I_j,p\rangle}(g) = \phi_{\langle I_j,p\rangle}(\mathrm{LC}_{x_1}(g)) \cdot x_1^{d_g} + \phi_{\langle I_j,p\rangle}(h) = 1 \Rightarrow d_g = 0$, i.e. $g \in \mathbb{Z}_p[x_2, \ldots, x_j]$.

At this point consider the natural onto ring homomorphism

$$\Phi : \mathbb{Z}_p[x_1, \ldots, x_j] \to \mathbb{Z}_p[x_1, \ldots, x_j]/I_j^{d+1}$$

given by $f(x_1, x_2, \ldots, x_j) \mapsto f(x_1, x_2 - \alpha_2, \ldots, x_j - \alpha_j) + I_j^{d+1}$. Then

$$\Phi(f) \in I_j^{d+1} \Longleftrightarrow \Phi(f) = f(x_1, x_2 - \alpha_2, \ldots, x_j - \alpha_j) = \sum_{k_2+\cdots+k_j=d+1} c_{k_2,\ldots,k_j}(x_1, \ldots, x_j)(x_2 - \alpha_2)^{k_2} \cdots (x_j - \alpha_j)^{k_j} \Longleftrightarrow$$
$$f(x_1, x_2, \ldots, x_j) = \sum_{k_2+\cdots+k_j=d+1} c_{k_2,\ldots,k_j}(x_1, x_2 + \alpha_2 \ldots, x_j + \alpha_j)x_2^{k_2} \cdots x_j^{k_j}$$

for some $c_{k_2,\ldots,k_j}(x_1, x_2, \ldots, x_j) \in \mathbb{Z}_p[x_1, \ldots, x_j]$ if and only if $f \in L_j^{d+1}$ where $L_j := \langle x_2, \ldots, x_j \rangle$. Hence we have a ring isomorphism $\mathbb{Z}_p[x_1, \ldots, x_j]/{L_j}^{d+1} \cong \mathbb{Z}_p[x_1, \ldots, x_j]/I_j^{d+1}$.

Therefore we have

$$\mathbb{Z}_{p^l}[x_1,\ldots,x_j]/K_j^{d+1} \cong \mathbb{Z}_{p^l}[x_1,\ldots,x_j]/{L_j}^{d+1} \cong \mathbb{Z}_{p^l}[x_1,\ldots,x_j]/I_j^{d+1}$$

and hence we may consider $\sigma,\tau,\tilde{\sigma},\tilde{\tau} \in \mathbb{Z}_{p^l}[x_1,\ldots,x_j]/{L_j}^{d+1}$ i.e. the solution polynomials up to total degree d with respect to $x_2,\ldots,x_j$ and represented in the natural way, via the coordinate translation isomorphism as described above.

Now as for the univariate case we have $(\sigma-\tilde{\sigma})u = -(\tau-\tilde{\tau})w$. Then $(\sigma-\tilde{\sigma})\frac{u}{g} = -(\tau-\tilde{\tau})\frac{w}{g} \Rightarrow \frac{w}{g} \mid (\sigma-\tilde{\sigma})$. But $\deg_{x_1}(\frac{w}{g}) = \deg_{x_1}(w) - \deg_{x_1}(g)$ and we have just shown that $g \in \mathbb{Z}_p[x_2,\ldots,x_j] \Rightarrow \deg_{x_1}(g) = 0$. So, $\deg_{x_1}(\frac{w}{g}) = \deg_{x_1}(w) - \deg_{x_1}(g) = \deg_{x_1}(w) \leq \deg_{x_1}(\sigma-\tilde{\sigma}) < \deg_{x_1}(w)$. Hence $\sigma-\tilde{\sigma} = 0 \Rightarrow \sigma = \tilde{\sigma} \Rightarrow \tau = \tilde{\tau}$.

For the case $l > 1$, we will use induction on l just as in the proof of the uniqueness part of Theorem 8. We have already seen the case $l = 1$. Now suppose that the claim is true for $1 \leq k < l$. Then we can define $\alpha = (\sigma-\tilde{\sigma})/p^k$ and $\beta = -(\tau-\tilde{\tau})/p^k$. Note that

$$(\sigma-\tilde{\sigma})u = -(\tau-\tilde{\tau})w \mod p^l \Rightarrow \alpha u = \beta w \mod p^{l-k}.$$

By repeating the degree argument above we get $\alpha = 0 \mod p \Rightarrow \sigma = \tilde{\sigma} \mod p^{k+1}$. Similarly we get $\beta = 0 \mod p \Rightarrow \tau = \tilde{\tau} \mod p^{k+1}$ and we are done. □

Chapter 3

Solving Multivariate Diophantine Equations

3.1 Solution via polynomial remainder sequences

Our first natural attempt to solve the MDP problem is to generalize the idea of the solution to the univariate case. The proof of the following well-known facts about resultants can be found in [CLO07].

Proposition 20. *Let $\mathbb{F}$ be a field, $A, B \in \mathbb{F}[x_1, \ldots, x_n]$ have positive degree in x_1 and let R be a Sylvester resultant of A and B, denoted $R = \mathrm{Res}_{x_1}(A, B)$. Then*
(i) $\mathrm{Res}_{x_1}(A, B) \in \mathbb{F}[x_2, \ldots, x_n]$ *(x_1 is eliminated),*
(ii) There exist $S, T \in \mathbb{F}[x_1, \ldots, x_n]$ such that $SA + TB = R$,
(iii) $\deg R \leq \deg A \deg B$ *(Bezout bound),*
(iv) $R = 0$ if and only if A and B have a common factor in $\mathbb{F}[x_1, \ldots, x_n]$ which has positive degree in x_1,
(v) For A and B monic in x_1 and $\alpha \in \mathbb{F}^{n-1}$, $\mathrm{Res}_{x_1}(A(x_1, \alpha), B(x_1, \alpha)) = R(\alpha)$.

As a first step let $l = 1$ and $u, w \in \mathbb{Z}_p[x_1, \ldots, x_j]$ be monic in the main variable, say x_1. According to Proposition 20-(ii), there exist $S, T \in \mathbb{Z}_p[x_1, \ldots, x_j]$ and $R = \mathrm{Res}_{x_1}(u, w) \in \mathbb{Z}_p[x_2, \ldots, x_j]$ such that $Su + Tw = R$.

Now we can proceed as in the univariate case: Given $u, w, c \in \mathbb{Z}_p[x_1, \ldots, x_j]$ as in the MDP problem, we must have that $\gcd(u, w) = 1$. Hence by Proposition 20-(iv) $R = \mathrm{Res}_{x_1}(u, w) \neq 0$. First we define $\tilde{\sigma} := Sc$ and then by pseudo division with respect to the variable x_1, we get $m\tilde{\sigma} = qw + r$ with $\deg_{x_1}(r) < \deg_x(w)$. Note that $m = 1$ since w is monic. Now, define $\bar{\sigma} := r$, $\bar{\tau} := cT + qu$. Then

$$\bar{\sigma}u + \bar{\tau}w = (\tilde{\sigma} - qw)u + (cT + qu)w = (Su + Tw)c = Rc.$$

Now since $R \in \mathbb{Z}_p[x_2, \ldots, x_j]$, if we define $\sigma := \frac{\bar{\sigma}}{R}$ and $\tau := \frac{\bar{\tau}}{R}$ then we have $\sigma u + \tau w = c$ with degree condition on x_1 satisfied (since R does not depend on x_1). By uniqueness we conclude R in fact divides both $\bar{\sigma}$ and $\bar{\tau}$ and we have the solution $\sigma, \tau \in \mathbb{Z}_p[x_1, \ldots, x_j]$.

In the procedure described above one can choose to compute σ first and then compute τ by multivariate division. One way of computing the resultant and the polynomials S, T described in Proposition 20 is the reduced-PRS (Pseudo Remainder) algorithm or Subresultant PRS algorithm. (For details, see [GCL92]). Before we discuss how efficient this method is, consider the following concrete example in which each of the polynomial solution pair (σ, τ) to the MDP has only 1 term.

Example 21. Let $p = 13$ and the parameters of the MDP be

$$\begin{aligned} u &= x^6 - 2\,xy^4 + 5\,y^2 \\ w &= x^6 - 2\,x^3z^2 - 5\,xy^4 + 4\,xy^2 \\ c &= -4\,x^8y + 4\,x^7yz - 5\,x^5yz^2 - 6\,x^3y^5 + 5\,x^2y^5z - 3\,x^3y^3 - 6\,xy^3z. \end{aligned}$$

The actual solution to this problem (that we will compute) is $(\sigma, \tau) = (4\,xyz, -4\,x^2y)$.

We first call reduced-PRS algorithm to compute

$$\begin{aligned} S &= 4\,x^5y^{20} + 5\,x^2y^{20}z^2 + \cdots + +4\,xy^{10} + 5\,y^{10} \text{ (79 terms)} \\ T &= -4\,x^5y^{20} - 5\,y^{24} + \cdots + -4\,xy^{10} - 5\,y^{10} \text{ (68 terms)} \end{aligned}$$

and the resultant

$$\begin{aligned} R = {} & 4\,y^{26} - 3\,y^{24} + 2\,y^{14}z^{10} - 4\,y^{16}z^6 + y^{12}z^{10} - y^{20} + 2\,y^{18}z^2 + y^{14}z^6 + 2\,y^{18} \\ & + y^{16}z^2 + y^{12}z^6 + 5\,y^6z^{12} + y^{16} + 4\,y^{14}z^2 + 3\,y^{10}z^6 - 4\,y^8z^8 + 5\,y^{14} + 5\,y^{12}z^2 - 5\,y^{10}z^4 - y^{12} \end{aligned}$$

We have $\tilde{\sigma} = Sc \bmod p$, which has 550 terms! As a next step we compute $\bar{\sigma} = \text{Prem}(\tilde{\sigma}, w, x_1) \bmod p$ and

$$\begin{aligned} \bar{\sigma} = {} & 3\,xy^{27}z + xy^{25}z - 5\,xy^{15}z^{11} - 3\,xy^{17}z^7 + 4\,xy^{13}z^{11} - 4\,xy^{21}z - 5\,xy^{19}z^3 + \\ & 4\,xy^{15}z^7 - 5\,xy^{19}z + 4\,xy^{17}z^3 + 4\,xy^{13}z^7 - 6\,xy^7z^{13} + 4\,xy^{17}z + 3\,xy^{15}z^3 \\ & - xy^{11}z^7 - 3\,xy^9z^9 - 6\,xy^{15}z - 6\,xy^{13}z^3 + 6\,xy^{11}z^5 - 4\,xy^{13}z \end{aligned}$$

Finally $\sigma = \frac{\bar{\sigma}}{R} = 4\,xyz$. Then we compute $\tau = (c - \sigma u)/w = -4x^2y$.

Although it is a short and a direct approach to solve the MDP problem, this method is very bad, since in the subresultant PRS (and also the primitive PRS) an intermediate expression swell occurs in all variables except x_1 and in the coefficients. This is inevitable because of pseudo-division. As Example 21 shows, even for very sparse solutions (solutions in this example have only 1 term), the polynomials S, T, R are much bigger than the input

polynomials. Also c in the MDP problems is a Taylor coefficient of the error which is in general huge. This means $\tilde{\sigma} = Sc$ is even larger (in this simple example it has 550 terms). These facts make the intermediate computations of the algorithm very expensive. Our experiments confirmed this. As, even for $l = 1$ and the monic inputs this method is not effective even for sparse solutions, therefore we look for another method to solve the MDP problem.

3.2 Solution via Groebner Basis

We can see the MDP problem as an ideal membership problem. If the solution exists then c is in the ideal $\mathcal{I}$ generated by the polynomials u, v in the ring $\mathbb{Z}_{p^l}[x_1, \ldots, x_j]$, i.e. $c \in \mathcal{I} = \langle u, v \rangle$. Once again, as a first step let $l = 1$ and $u, w \in \mathbb{Z}_p[x_1, \ldots, x_j]$ be monic in the main variable, say x_1.

For $j \geq 1$, by generalization of univariate division to multivariate division one can find quotients σ_1, τ_1 and a remainder r in $\mathbb{Z}_p[x_1, \ldots, x_j]$ so that $c = \sigma_1 u + \tau_1 v + r$. However by changing the order of u and v we may get a different remainder. So, this method does not result in a unique remainder r in general.

However this task can be accomplished by using Groebner basis. For an introduction to Groebner basis theory see [CLO07] or [GCL92]. Let $\mathcal{G} = (g_i)$ where $g_i \in \mathbb{Z}_p[x_1, \ldots, x_n]$ be the Groebner basis of $\mathcal{I}$ so that $\mathcal{I} = \langle u, v \rangle = \langle g_i \rangle$. Then by the property of Groebner basis, there exist quotients Q_i's and a unique remainder r in $\mathbb{Z}_p[x_1, \ldots, x_j]$ such that $c = \sum_i Q_i g_i + r$. For a different ordering of g_i's the Q_i's may be different, however this time r is unique. If $r \neq 0$ then $r \notin \langle g_i \rangle \Rightarrow r \notin \mathcal{I}$ and hence there is no solution to the MDP.

Now, if $r = 0$ then $c \in \langle g_i \rangle = \mathcal{I}$. Let us consider $\mathcal{I}$ and $\mathcal{G}$ as column vectors. The Groebner basis computation algorithm can be implemented in such a way that it computes $\mathcal{G}$ as well as the the transformation matrix U from $\mathcal{I}$ to $\mathcal{G}$ such that $\mathcal{G} = U\mathcal{I}$, with basically no extra effort. It is called as extended Groebner basis algorithm. (See for example [KR00].)

When the number of variables is finite, $\mathbb{Z}_p[x_1, \ldots, x_n]$ is a Noetherian ring, i.e. every ideal of it finitely generated. Hence the cardinality of the Groebner basis is finite. Suppose that the number of polynomials in the Groebner basis is m. Then U is a $m \times 2$ matrix and

$$c = [\ Q_1 \quad \ldots \quad Q_m\ ]\mathcal{G} = [\ Q_1 \quad \ldots \quad Q_m\ ]\,(U\mathcal{I}) = \Big([\ Q_1 \quad \ldots \quad Q_m\ ]U\Big)\,\mathcal{I}.$$

Therefore $[\ \tilde{\sigma} \quad \tilde{\tau}\ ] = [\ Q_1 \quad \ldots \quad Q_m\ ]U$ is a candidate solution: $\tilde{\sigma}u + \tilde{\tau}v = c \bmod p$. But the degree condition may not be satisfied, so the solution may not be unique. Note that v is monic in x_1. To reduce the degree of $\tilde{\sigma}$, we can proceed as if in the univariate case: We consider $\tilde{\sigma}$ and $v \in \mathbb{Z}_p[x_2, \ldots, x_j][x_1]$ and do the pseudo-division w.r.t the variable x_1 and define σ as the remainder of this division as we did in Section 3.1. Then we can compute τ accordingly.

The problem with this approach is the complexity of finding a Groebner basis and the size of the transition matrix U. Even for small examples the size of U can be very big and the computation cost of U can be very expensive. Although it is mathematically interesting, unfortunately this approach is not practical for solving the MDP's that appear in the multivariate factorization.

Example 22. Suppose that we're trying to solve the MDP problem $\sigma u + \tau w = c \bmod 13$ for (σ, τ) where:

$$\begin{aligned} u &= x^6 - x^2y^2 + 2\,y^2 \\ w &= x^6 - 2\,x^4y + 2\,y^2 \\ c &= -2\,x^{10} - 2\,x^7y^3 + 4\,x^8y + 2\,x^3y^5 - 4\,x^4y^2 - 4\,xy^5 \end{aligned}$$

The actual solution (that we will compute) to this problem is $(\sigma, \tau) = (-2\,y^3x, -2\,x^4)$. Note that each of the solution pair has only 1 term!

Let $\mathcal{I} = \langle u, w \rangle$. First we compute the Groebner Basis of the ideal $\mathcal{I}$. The `Basis` command in the Groebner package of Maple computes the Groebner basis and it also returns the transition matrix from the basis of $\mathcal{I}$ to Groebner basis $\mathcal{G} = (g_i)$ so that $\mathcal{I} = \langle u, w \rangle = \langle g_i \rangle$. By using the commands G,C := `Basis([u,w],tdeg(x,y,z),characteristic=13,output=extended)` and U:=`Matrix(C)` we have:

$$U = \begin{bmatrix} -6\,x^4 - x^2y - 3\,x^2 - 2\,y & 6\,x^4 + 3\,x^2 - 6\,y^2 - 5\,y \\ 5\,x^2 - y & -5\,x^2 + 4\,y \\ -6 & 6 \\ 1 & 0 \end{bmatrix}$$

So, if we use the command I:=`Vector([u,w])` in Maple, then we have $\mathcal{G} = U\mathcal{I}$:

$$\mathcal{G} = \begin{bmatrix} y^4 - y^3 \\ x^2y^3 + 6\,y^3 \\ x^4y + 6\,x^2y^2 \\ x^6 - x^2y^2 + 2\,y^2 \end{bmatrix} = U \begin{bmatrix} x^6 - x^2y^2 + 2\,y^2 \\ x^6 - 2\,x^4y + 2\,y^2 \end{bmatrix}.$$

The `NormalForm` command in Maple, computes the representation of c given the ideal $\mathcal{I}$ with the Groebner basis $\mathcal{G}$, i.e. it computes Q_i's and r such that $c = \sum_i Q_i g_i + r$. We call r:=`NormalForm(c,G,tdeg(x,y,z),'Q',characteristic = 13)` and Q:=`Vector[row](Q)` and obtain

$$Q = \begin{bmatrix} 2\,x^3y + 2\,x^3 - 4\,xy - 4\,x & -2\,x^5 - x^3 - 5\,x & 4\,x^4 & -2\,x^4 \end{bmatrix}$$

and $r = 0$. In the command above `tdeg(x,y,z)` represents the grlex order, for which Groebner basis computation is fastest in general. Then we compute QU and simplify it with the command `simplify(expand(Q.U) ) mod 13` , to get

$$QU = [\ \tilde{\sigma} \quad t\tilde{a}u\ ] = \begin{bmatrix} x^7y + 4\,x^7 - 2\,x^5y^2 + 5\,x^5y - 5\,xy^2 \\ -x^7y - 4\,x^7 + x^3y^3 + 4\,x^3y^2 - 2\,x^4 - 2\,y^3x + 5\,xy^2 \end{bmatrix}^T.$$

In fact $\tilde{\sigma}u + \tilde{\tau}w = c \bmod 13$. But note that $\deg_x(\tilde{\sigma}) \not< \deg_x(v)$. So, to get the unique solution, we proceed as in the univariate case: We consider $\tilde{\sigma}$ and $v \in \mathbb{Z}_{13}[y][x]$ and do the pseudo-division w.r.t the variable x and define σ as the remainder. (In Maple use the command `Rem(tildeSigma,v,x) mod 13`):

$$\sigma = -2\,y^3x \Rightarrow \tau = (c - \sigma u)/w = -2\,x^4$$

3.3 Wang's solution

Once again, the problem is to find multivariate polynomials $\sigma, \tau \in \mathbb{Z}_{p^l}[x_1, \ldots, x_j]$ such that

$$\sigma u + \tau w = c \bmod \left\langle I_j^{d+1},\, p^l \right\rangle. \tag{3.1}$$

Wang's solution idea is the same idea we used to solve MHL incrementally, namely, to use a $(x_j - \alpha_j)$-adic Newton iteration. Let $j > 1$ and

$$\triangle\sigma^{(k)} := s_k(x_1, \ldots, x_{j-1})(x_j - \alpha_j)^k,\ \sigma^{(k+1)} := \sigma^{(k)} + \triangle\sigma^{(k)}$$

$$\triangle\tau^{(k)} := t_k(x_1, \ldots, x_{j-1})(x_j - \alpha_j)^k,\ \tau^{(k+1)} := \tau^{(k)} + \triangle\tau^{(k)}.$$

Since $\deg_j(\sigma)$ and $\deg_j(\tau)$ are bounded, for some k we must have $\sigma = \sigma^{(k)}$ and $\tau = \tau^{(k)}$. Let now $G = \sigma u + \tau w - c$ and $I_{j-1} = \langle x_2 - \alpha_2, \ldots, x_{j-1} - \alpha_{j-1} \rangle$. Then for any $k \geqslant 1$ we have

$$G(\sigma^{(k)}, \tau^{(k)}) = 0 \bmod \left\langle (x_j - \alpha_j)^k, I_{j-1}^{d+1}, p^l \right\rangle.$$

As before we have

$$\begin{aligned} u(x_1, \ldots, x_j)s_k(x_1, \ldots, x_{j-1}) + w(x_1, \ldots, x_j)t_k(x_1, \ldots, x_{j-1}) \\ = \frac{c(x_1, \ldots x_j) - \sigma^{(k)}u(x_1, \ldots, x_j) + \tau^{(k)}w(x_1, \ldots, x_j)}{(x_j - \alpha_j)^k} \\ \bmod \left\langle (x_j - \alpha_j)^k, I_{j-1}^{d+1}, p \right\rangle. \end{aligned} \tag{3.2}$$

Note that the interpretation of I_1 is the empty ideal and the discussion above assumed that $j > 1$, since for $j = 1$ we know the solution already.

Algorithm 6 WDS (Wang's Diophant Solver)

Input A prime p, polynomials $u, w, c \in \mathbb{Z}_p[x_1, x_2, \ldots, x_j]$, ideal $I_j = \langle x_2 - \alpha_2, \ldots, x_j - \alpha_j \rangle$ where the MDP conditions satisfied. Also the maximum degree d of the solution w.r.t $x_2, \ldots, x_j$.

Output $(\sigma, \tau) \in \mathbb{Z}_p[x_1, x_2, \ldots, x_j]$ such that $\sigma u + \tau w = c \in \mathbb{Z}_p[x_1, x_2, \ldots, x_j]$ or **FAIL** . It returns **FAIL** if there is no solution.

1: **if** $j = 1$ **then** call Algorithm UniDio to **return** $(\sigma, \tau) \in \mathbb{Z}_p[x_1]^2$ or FAIL **end if**
2: $(u_{\alpha_j}, w_{\alpha_j}, c_{\alpha_j}) \leftarrow (u(x_j = \alpha_j), w(x_j = \alpha_j), c(x_j = \alpha_j)) \in \mathbb{Z}_p[x_1, x_2, \ldots, x_{j-1}]^3$
3: $I_{j-1} \leftarrow \langle x_2 - \alpha_2, \ldots, x_{j-1} - \alpha_{j-1} \rangle$
4: $(\sigma, \tau) \leftarrow \mathbf{WDS}(u_{\alpha_j}, w_{\alpha_j}, c_{\alpha_j}, I_{j-1}, p)$
5: **if** $(\sigma, \tau) =$ FAIL **then return** FAIL **end if**
6: $error \leftarrow c - \sigma u - \tau w$
7: $monomial \leftarrow 1$
8: **for** i from 1 to d **while** $error \neq 0$ **do**
9: $\quad monomial \leftarrow monomial \times (x_j - \alpha_j)$
10: $\quad c_i \leftarrow$ Taylor coefficient of $(x_j - \alpha_j)^i$ of $error$ at $x_j = \alpha_j$
11: $\quad$ **if** $c_i \neq 0$ **then**
12: $\quad\quad (s, t) \leftarrow \mathbf{WDS}(u_{\alpha_j}, w_{\alpha_j}, c_i, I_{j-1}, p)$
13: $\quad\quad$ **if** $(s, t) =$ FAIL **then return** FAIL **end if**
14: $\quad\quad (s, t) \leftarrow (s \times monomial, t \times monomial)$
15: $\quad\quad (\sigma, \tau) \leftarrow (\sigma + s, \tau + t)$
16: $\quad\quad error \leftarrow error - tu - sw$
17: $\quad$ **end if**
18: **end for**
19: **if** $error = 0$ **then return** (σ, τ) **else return** FAIL **end if**

We have thus a recursive algorithm to solve Eqn (3.1). To solve Eqn (3.1) we pass to modulo $(x_j - \alpha_j)$ and try to solve

$$\sigma(x_1, \ldots, \alpha_j)u(x_1, \ldots, \alpha_j) + \tau(x_1, \ldots, \alpha_j)w(x_1, \ldots, \alpha_j) = c(x_1, \ldots, \alpha_j) \bmod \left\langle I_{j-1}^{d+1}, p^l \right\rangle$$

$$\Rightarrow \sigma^{(1)}u(x_1, \ldots \alpha_j) + \tau^{(1)}w(x_1, \ldots \alpha_j) = c(x_1, \ldots \alpha_j) \bmod \left\langle I_{j-1}^{d+1}, p^l \right\rangle.$$

To do that we use Eqn (3.2) for $k = 1, 2, \ldots, d$, i.e. solve

$$us_k + wt_k = e_k^{(k)}(x_1, \ldots, x_{j-1}) \bmod \left\langle (x_j - \alpha_j)^k, I_{j-1}^{d+1}, p \right\rangle.$$

where $e_k(x_1, \ldots, x_{j-1})$ denotes the coefficient of $(x_j - \alpha_j)^k$ in the $\langle x_j - \alpha_j \rangle$-adic representation of

$$e^{(k)}(x_1, \ldots, x_j) = c - \sigma^{(k)}u - \tau^{(k)}w$$

Also by construction we will have $\text{degree}_{x_1}(\sigma(x_1, \ldots x_j)) < \text{degree}_{x_1}(w(x_1, \ldots x_j))$.

Algorithm 6 (WDS) gives the pseudo-code for Wang's recursive solution to MDP, for $l = 1$. It can easily be extended to the case $l > 1$ by use of Theorem 8.

3.3.1 On Wang's MDP solver

Let the factors to be computed at the j^{th} step of MHL be $f_j = \sum_{i=0}^{d_j} f_{ji}(x_j - \alpha_j)^i$ and $g_j = \sum_{i=0}^{d_j} g_{ji}(x_j - \alpha_j)^i$.

To compute f_{ji} and g_{ji} during the i^{th} iteration in the for loop, MHL calls WDS and WDS computes Taylor expansions of (f_{ji}, g_{ji}) at $x_{j-1} = \alpha_{j-1}$. (That is, to solve MDP, WDS simply mimics MHL.)

To compute the Taylor coefficients of f_{ji} and g_{ji} WDS makes a recursive call (just as MHL does). To see the approximate number of calls to Algorithm UniDio by WDS for α_j, let $C(x_s)$ be the total number of calls to UniDio to recover x_s. We have $C(x_1) = 1$.

If $\alpha_j \neq 0$, to recover x_j WDS makes $\mathcal{O}(d_j)C(x_{j-1})$ calls to UniDio as in this case we expect Taylor coefficients to be non-zero. Therefore if all evaluation points are non-zero $C(x_j) \in \mathcal{O}(\prod_{i=2}^{j} d_i)$.

If $H(x_n)$ is the number of calls made by MHL to recover factors then

$$H(x_n) \in \mathcal{O}(d_n \cdots d_2) + \mathcal{O}(d_{n-1} \cdots d_2) + \cdots + \mathcal{O}(d_3 d_2) + \mathcal{O}(d_2)$$

If $d \geq d_i$ we obtain

$$H(x_n) \in \mathcal{O}(d^{n-1} + d^{n-2} + \cdots + d) \in \mathcal{O}(d^n)$$

We see that as n increases the number of calls to Algorithm UniDio increases exponentially especially for the case where evaluation points are non-zero.

If the evaluation points are all-zero then sparse polynomials remain sparse in all stages of MHL and WDS. This is the case where WDS is very efficient (See Section 6.6), as the number of MDP to be solved significantly decreases. This is the main advantage of incremental design of MHL and explains why Wang's second proposal [Wan78] behaves better than his previous proposal with Rothschild [WR75].

For a non-zero evaluation, Taylor expansion most likely turns out to be dense, that is most likely the Taylor coefficients become non-zero polynomials (See Section 4.4) and the number of MDP to be solved by WDS increases rapidly.

As explained in Chapter 2 it is not always possible to choose evaluation points to be zero, because Wang's LCC does not work for all-zero (or many-zero) evaluation points, for many practical examples (See Example 13 and also Section 6.6)

This leads us to search better ways to solve MDP than WDS, especially for the sparse case where evaluation points are non-zero.

3.4 Solution via interpolation

Once again, as a first step let $l = 1$ and $u, w \in \mathbb{Z}_p[x_1, \ldots, x_j]$ be monic in the main variable, say x_1 and the problem is to find multivariate polynomials $\sigma, \tau \in \mathbb{Z}_p[x_1, \ldots, x_j]$ such that

$$\sigma u + \tau w = c \mod \left\langle I_j^{d+1}, p \right\rangle. \tag{3.3}$$

We consider whether we can interpolate $x_2, \ldots, x_j$ in σ. If $\beta \in \mathbb{Z}_p$ with $\beta \neq \alpha_j$, then we have

$$\sigma(x_j = \beta)u(x_j = \beta) + \tau(x_j = \beta)w(x_j = \beta) = c(x_j = \beta) \mod I_{j-1}^{d_{j-1}+1}.$$

For the ideal $K_j = \langle x_2 - \alpha_2, \ldots, x_{j-1} - \alpha_{j-1}, x_j - \beta \rangle$ and $G_j = \gcd(u \bmod K_j, w \bmod K_j)$, we obtain a unique solution $\sigma(x_1, \ldots, x_{j-1}, \beta)$ iff $G_j = 1$. However it is possible that $G_j \neq 1$. Let $R = \mathrm{Res}_{x_1}(u, w)$ be the Sylvester resultant of u and v taken in x_1.

Going back to our discussion, since u, w are monic in x_1 Proposition 20 implies that

$$G_j \neq 1 \Longleftrightarrow \mathrm{Res}_{x_1}(u \bmod K_j, w \bmod K_j) = 0 \Longleftrightarrow R \bmod K_j = 0.$$

Therefore $\Pr[G_j \neq 1] = \Pr[R \bmod K_j = 0]$. To compute an upper bound for this probability we need Schwartz-Zippel Lemma.

Lemma 23. *[Sch80, Zip90] Let $f \in \mathbb{Z}_p[x_1, \ldots, x_n]$ be a polynomial of total degree d. Assume that f is not identically zero. Let $S \subseteq \mathbb{Z}_p$ be any finite set. Then, if we pick $\alpha_1, \ldots, \alpha_n$ independently and uniformly from S*

$$\Pr[f(\alpha_1, \ldots, \alpha_n) = 0] \leq d/|S|.$$

Now, proposition 20 also implies that $\deg(R) \leq \deg(u)\deg(w)$. Then by the Schwartz-Zippel Lemma

$$\Pr[G_j \neq 1] \leq \frac{\deg(u)\deg(w)}{p-1}.$$

If $\beta \neq \alpha_j$ is chosen at random and p is large, the probability that $G_j = 1$ is high so interpolation is thus an option to solve the MDP. If $G_j \neq 1$, we could choose another β. The bound above for $\Pr[G_j \neq 1]$ is a worst case bound. We will show in Chapter 5 that the average probability for $\Pr[G_j \neq 1] = 1/p$.

A natural algorithm for the solution to the MDP via interpolation is: (All computations are in $\mod p$)

1. If $j = 1$ then use UniDio to compute $\sigma, \tau \in \mathbb{Z}_p[x_1]$ and return (σ, τ).
2. Initialize $\sigma = 0$, $H = 0$, $M = 1$.
3. Repeat

(a) Pick a random $\alpha \in \mathbb{Z}_p$ uniformly and pass to modulo $(x_j - \alpha)$:
$u_\alpha = u(x_1, \ldots, x_j = \alpha), w_\alpha = w(x_1, \ldots, x_j = \alpha)$ and
$c_\alpha = c(x_1, \ldots, x_j = \alpha) \in \mathbb{Z}_p[x_1, x_2, \ldots, x_{j-1}]$.
Here if $j = 2$ and $\gcd(u_\alpha, w_\alpha) \neq 1$ Restart.

(b) Apply the algorithm recursively to solve the equation $\sigma_\alpha u_\alpha + \tau_\alpha w_\alpha = c_\alpha$ to find $\sigma_\alpha, \tau_\alpha \in \mathbb{Z}_p[x_1, \ldots, x_{j-1}]$.

(c) Incrementally interpolate σ :
Solve $H = \sigma_\alpha \mod (x_j - \alpha)$ and $H = \sigma \mod M$ for $H \in \mathbb{Z}_p[x_1, \ldots, x_j]$.

(d) Update the monomial M: $M = M \cdot (x_j - \alpha)$.

(e) If $H = \sigma$ then check whether $w \mid c - \sigma u$. If true, define $\tau = (c - \sigma u)/w$ and return(σ, τ) else $\sigma \leftarrow H$.

Note that the division in step 3-(e) is in $\mathbb{Z}_p[x_1, \ldots, x_j]$ and checking the exactness of this division can be done by using the division algorithm for the multivariate case [CLO07]. For an efficient implementation using heaps which works well for the sparse case see also [MP11, Joh74]. Since the solution is to the MDP is unique there must be a stabilization and hence the result.

Solution for the non-monic case and $l > 1$: To see that the interpolation approach also works for the non-monic case and $l > 1$, first note the MDP condition that $p \nmid \mathrm{LC}_{x_1}(\phi_{I_j}(uw))$ implies that the leading coefficients of u and w will not vanish $\mod (p, I_j)$ and hence the argument based on the resultant is still valid in this case.

As a next step, we need to modify the interpolation procedure described above to make it work for mod p^l. We can choose to apply a Newton's iteration as before to extend the solution $\mod p$ to $\mod p^l$. But there is an easier way: This time the α's will be chosen from $\mathbb{Z}_{p^l}$. This may yield a problem at step 3-(c). To overcome this problem we look closely how the incremental interpolation works in $\mod p$:

The aim is to find $H \in \mathbb{Z}_p[x_1, \ldots, x_j]$ such that $H = \sigma_\alpha \mod (p, x_j - \alpha)$ and $H = \sigma \mod (p, M)$. Note that $M \in \mathbb{Z}_p[x_j]$. The division algorithm gives $M = q(x_j - \alpha) + r$ for some $q \in \mathbb{Z}_p[x_j]$ and $r \in \mathbb{Z}_p$. Then $r^{-1}M - r^{-1}q(x_j - \alpha) = 1 \mod p$. So if we define $s = r^{-1}$ then $sM = 1 \mod (p, x_j - \alpha)$. Now we define $H = \sigma + sMf$ where $f = (\sigma_\alpha - \sigma) \mid_{x_k = \alpha}$. Hence $H = \sigma \mod M$ and $H = \sigma_\alpha \mod (x_j - \alpha)$.

Now in $\mathbb{Z}_{p^l}[x_j]$ the division algorithm will work since $x_j - \alpha$ is monic. But if $r = M(\alpha) = 0 \mod p$ then r will not be invertible in mod p^l. Hence to make the interpolation algorithm work for $l > 0$, all we need in the procedure 3-(a) above to choose $\alpha \in \mathbb{Z}_{p^l}$ such that $M(\alpha) \neq 0 \mod p$.

Finally the condition $p \nmid \mathrm{LC}_{x_1}(\phi_{I_j}(uw))$ also implies that $p \nmid \mathrm{LC}(w) \in \mathbb{Z}_{p^l}$. Hence $\mathrm{LC}(w)$ will be invertible in $\mathbb{Z}_{p^l}$ and the division algorithm will work for the $(c - \sigma u)/w \mod p^l$ at step 3-(e).

Example 24. Let $p = 5$ and $q = p^2 = 25$. Consider the MDP example in $\mathbb{Z}_{25}$ where $I_3 = \langle x_2 = 1, x_3 = 2 \rangle$

$$u = -7\,{x_1}^4{x_2}^2 - 12\,{x_3}^4 - 12\,x_1 + 1\,,\; w = 6\,{x_1}^4x_3 + 4\,{x_1}^2{x_2}^2x_3 + 2 \text{ and}$$

$$\begin{aligned} c = {} & -12\,{x_1}^7{x_2}^2 + 4\,{x_1}^6{x_2}^3 + 9\,{x_1}^4{x_2}^5 + 3\,{x_1}^5{x_2}^2x_3 - 3\,{x_1}^5x_2{x_3}^2 - 6\,{x_1}^4{x_2}^2{x_3}^2 + 2\,{x_1}^3{x_2}^4x_3 \\ & - 2\,{x_1}^3{x_2}^3{x_3}^2 - 7\,{x_1}^5{x_3}^2 - 6\,{x_1}^4{x_2}^3 - {x_1}^4{x_2}^2x_3 + 12\,{x_1}^3{x_2}^2{x_3}^2 + 8\,{x_1}^3{x_3}^4 - 11\,{x_1}^2x_2{x_3}^4 \\ & - 6\,{x_2}^3{x_3}^4 + 4\,{x_3}^6 + 4\,x_2{x_3}^4 + 9\,{x_3}^5 + 8\,{x_1}^4 - 11\,{x_1}^3x_2 - 6\,x_1{x_2}^3 - 9\,{x_1}^3 + 3\,{x_1}^2x_2 + x_1{x_2}^2 \\ & - x_1x_2x_3 + 4\,x_1{x_3}^2 - 12\,{x_2}^3 + 4\,x_1x_2 - 10\,x_1x_3 + 8\,{x_3}^2 + 8\,x_2 - 7\,x_3 \end{aligned}$$

Note that

$$u^{(1)}(x_1) = \phi_{\langle I_3,5\rangle}(u) = -2\,{x_1}^4 - 2\,x_1 - 1,\; w^{(1)}(x_1) = \phi_{\langle I_3,5\rangle}(w) = {x_1}^4 - {x_1}^2 + 2$$

and $\gcd(u^{(1)}, w^{(1)}) = 1 \mod 5$. We first choose $\alpha = 0$ and get

$$H = -9\,{x_1}^3 + 3\,{x_1}^2x_2 - 12\,{x_2}^3 + 8\,x_2.$$

Now $M = x_3$. Then we choose $\alpha = 1$ and update H,

$$H = -9\,{x_1}^3 + 3\,{x_1}^2x_2 - 12\,{x_2}^3 + 8\,x_2 + x_3.$$

Now $M = x_3(x_3 - 1)$. Next we choose $\alpha = 11$. But $M(11) = 110 = 0 \mod 5$. So we skip it. Then we choose $\alpha = 9$ and update H,

$$H = -9\,{x_1}^3 + 3\,{x_1}^2x_2 - 12\,{x_2}^3 + 8\,{x_3}^2 + 8\,x_2 - 7\,x_3$$

For the next choice of α we realize the stabilization of H. Then we assume $\sigma = H$ and check $\frac{c-\sigma u}{w} = -12\,x_1{x_2}^2 + 12\,x_1x_2x_3 + 3\,x_1x_3$. Note that all computations are in mod 25 and we have in fact the true solution pair $(\sigma, \tau) \in \mathbb{Z}_{5^2}[x_1, x_2, x_3]$.

3.5 Sparse interpolation

Our experiments showed that the solution via dense interpolation is not efficient enough. It is exponential for sparse inputs. Therefore we are led to consider a sparse interpolation approach for sparse u and w. Assuming that the solutions to the MDP are sparse we can use the idea behind the computation of GCD via sparse interpolation by Zippel [Zip90].

We explain the idea of sparse interpolation with an example let

$$\begin{aligned} A &= ({x_1}^3 + x_1{x_2}^3x_3 + 2\,x_1x_2{x_3}^2 + 2)(2\,x_1x_2x_3 + {x_1}^2 + 3) \\ B &= ({x_1}^3 + x_1{x_2}^3x_3 + 2\,x_1x_2{x_3}^2 + 2)({x_1}^3 + x_1x_2 + x_1x_3 + 1) \end{aligned}$$

and suppose we want to compute

$$g = \gcd(A, B) = ({x_1}^3 + ({x_2}^3x_3 + 2\,x_2{x_3}^2)\,x_1 + 2) \in \mathbb{Z}_{11}[x_1, x_2, x_3]$$

given the expanded form of inputs A and B. Suppose we computed recursively at $x_3 = 2$

$$g_1 = \gcd(A(x_1, x_2, 2), B(x_1, x_2, 2)) = x_1^3 + (2x_2^3 - 3x_2)x_1 + 2 \bmod 11$$

and we want to compute a second image at $x_3 = -5$

$$g_2 = \gcd(A(x_1, x_2, -5), B(x_1, x_2, -5)) \bmod 11$$

From g_1 we will make the **sparse interpolation assumption** that

$$g = x_1^3 + (f_1(x_3)x_2^3 + f_2(x_3)x_2)x_1 + f_3(x_3)$$

for some polynomials f_1, f_2 and f_3 in $\mathbb{Z}_{11}[x_3]$. So g_2 will be in the form

$$g_f = x_1^3 + (c_1x_2^3 + c_2x_2)x_1 + c_3$$

for some constants $c_1, c_2, c_3 \in \mathbb{Z}_{11}$. Now focus on the coefficients c_1 and c_2. Picking the random values, say -3 and 2 for x_2 we compute in $\mathbb{Z}_{11}[x_1]$ using the Euclidean algorithm

$$\begin{aligned} \gcd(A(x_1, -3, -5), B(x_1, -3, -5)) = x_1^3 - 4x_1 + 2 = g_f(x_1, -3) \bmod 11 \\ \gcd(A(x_1, 2, -5), B(x_1, 2, -5)) = x_1^3 + 5x_1 + 2 = g_f(x_1, 2) \bmod 11 \end{aligned}$$

Equating the coefficients in x_1 of LHS and RHS of the equations above, we obtain the following linear systems mod 11: $c_3 = 2$ and

$$\begin{aligned} -5c_1 - 3c_2 &= -4 \\ -3c_1 + 2c_2 &= 5 \end{aligned}$$

Solving these linear systems mod 11 we get $c_1 = c_2 = -5$ and $c_3 = 2$. We conclude that

$$g_2 = x_1^3 + (-5x_2^3 - 5x_2)x_1 + 2 \bmod 11$$

Assuming that the form assumption is correct, to compute g_2, what this small example suggests to us is that we should focus on the non-zero coefficients of powers of x_1 separately. If the suggested form had a term say $(d_1x_2^4 + d_2)x_1^2$ we could focus on d_1, d_2 separately and work with a 2×2 linear system, instead of considering c_1, c_2, c_3, d_1, d_2 all together and work with a 5×5 linear system.

Let the assumed form of the monic-GCD be

$$g_f = \sum_{i=1}^{m} c_i(x_2, ..., x_n)x_1^{n_i} \text{ where } c_i(x_2, ..., x_n) = \sum_{j=1}^{t_i} c_{ij}x_2^{\gamma_{2j}} \dots x_n^{\gamma_{nj}} \text{ with } c_{ij} \neq 0.$$

where $c_1(x_2, \dots, x_n) = 1$. Suppose t is the maximum number of terms in the coefficients of g, i.e. $t = \max_{i=1}^{m} \#t_i$. In sparse interpolation we should get each $c_i(x_2, ..., x_n)$ by solving at most $(m-1)$ $t \times t$ linear systems (for the leading term we don't need that), by giving random values to the $(n-1)$-tuple $(x_2, \dots, x_n)$. That is much more efficient than solving a $t(m-1) \times t(m-1)$ linear system. As explained in [Zip90], by choosing evaluation points carefully each linear system can be solved in $\mathcal{O}(t^2)$ arithmetic operations in $\mathbb{Z}_p$.

Note that this algorithm is probabilistic, since each sparse interpolation based on the assumption that coefficients which are zero in one dense interpolation will probably be zero in all cases. For example in the example above g_1 does not have the term $x_1x_2^2$ and we guessed that the it won't have this term for the next step. If the actual GCD had the term $x_1x_2^2(x_3 - 2)$ and then this term would be evaluated to 0 for $x_3 = 2$ and the assumption for g_2 would be incorrect.

The nice thing is that the probability that the assumption is incorrect can be made arbitrarily small by using a large enough range for evaluation points, i.e. using big primes, based on Lemma 23, Schwartz-Zippel Lemma.

3.6 Solution via sparse interpolation

Once again, as a first step let $l = 1$ and $u, w \in \mathbb{Z}_p[x_1, \dots, x_j]$ be monic in the main variable, say x_1 and the problem is to find multivariate polynomials $\sigma, \tau \in \mathbb{Z}_p[x_1, \dots, x_j]$ such that

$$\sigma u + \tau w = c \mod \left\langle I_j^{d+1}, p \right\rangle. \tag{3.4}$$

Following the sparse interpolation idea of Zippel in [Zip90], given a sub-solution $\sigma_j(x_1, \dots, x_j = \alpha_j)$ for $\alpha_j \in \mathbb{Z}_p$ we use this information to create a sub-solution form σ_f and compute $\sigma_j(x_1, \dots, x_j = \beta_j)$ for other random β_j's in $\mathbb{Z}_p$ with high probability if p is big. Suppose the form of σ_j is

$$\sigma_f = \sum_{i=1}^{m} c_i(x_2, ..., x_j)x_1^{n_i} \text{ where } c_i = \sum_{k=1}^{t_i} c_{ik}x_2^{\gamma_{2k}} \cdots x_j^{\gamma_{jk}} \text{ with } c_{ik} \in \mathbb{Z}_p \backslash \{0\}.$$

Let $t = \max_{i=1}^{m} t_i$ be the maximum number of terms in the coefficients of σ. In sparse interpolation we obtain each c_{ik} by solving m linear systems of size at most $t \times t$. As explained in [Zip90], each linear system can be solved in $\mathcal{O}(t^2)$ arithmetic operations in $\mathbb{Z}_p$. We then interpolate x_j in σ_j from $\sigma_j(x_1, \ldots, x_{j-1}, \beta_k)$ for $k = 0, \ldots, \deg_{x_j}(\sigma_j)$. Finally we compute $\tau_j = (c_j - \sigma_j u_j)/w_j$.

This method will also work for the case $l > 1$ by a careful choice of evaluation points as explained in Section 3.5 with one more additional condition: The sparse interpolation routine method explained in [Zip90] uses Vandermonde matrices. To make the explanation easier consider the Vandermonde matrix

$$V = \begin{bmatrix} a & b & c \\ a^2 & b^2 & c^2 \\ a^3 & b^3 & c^3 \end{bmatrix}.$$

We have $\det(V) = -abc\,(b-c)\,(a-c)\,(a-b)$. If a, b, c are all distinct in $\mathbb{Z}_p$ then $\det(V)$ is non-zero and hence invertible in $\mathbb{Z}_p$. So to make it invertible in $\mathbb{Z}_{p^l}$ we should choose $p \nmid abc$ and the a, b, c shouldn't be p apart. This condition can easily be checked in the sparse interpolation routine. We take one more step before we give an outline of an algorithm to solve an MDP problem via sparse interpolation.

3.6.1 First Improvement

The approach introduced in the preceding section solves the interpolation problem based on projection down to $\mathbb{Z}_p[x_1]$. To reduce the cost we tried projecting down to $\mathbb{Z}_p[x_1, x_2]$ because this will likely reduce the number t of evaluation points needed. Let the total degree of σ in x_1, x_2 be bounded by d and let

$$\sigma_f = \sum_{i+k \le d} c_{ik}(x_3, ..., x_j) x_1^i x_2^k \text{ where } c_{ik} = \sum_{l=0}^{s_{ik}} c_{ikl} x_3^{\gamma_{3l}} \cdots x_j^{\gamma_{jl}} \text{ with } c_{ikl} \in \mathbb{Z}_p \backslash \{0\}.$$

Let $s = \max s_{ik}$ be the maximum number of terms in the coefficients of σ_f. Here we solve $\mathcal{O}(d^2)$ linear systems of size at most $s \times s$. For $s < t$, the complexity of solving the linear systems decreases by a factor of $(t/s)^2$. We also save a factor t/s in the evaluation cost, which is often the most costly operation in sparse interpolation.

To solve the MDP in $\mathbb{Z}_p[x_1, x_2]$ we have implemented an efficient dense bivariate diophantine solver (BDP) in C. The algorithm incrementally interpolates x_2 in both σ and τ from univariate images in $\mathbb{Z}_p[x_1]$. When σ and τ stabilize we test whether $\sigma(x_1, x_2)u(x_1, x_2) + \tau(x_1, x_2)w(x_1, x_2) = c(x_1, x_2)$ using sufficiently many evaluations to prove the correctness of the solution. The cost is $\mathcal{O}(d^3)$ arithmetic operations in $\mathbb{Z}_p$ where d bounds the total degree of c, u, w, σ and τ in x_1 and x_2. We do not compute τ using division because that

could cost $\Theta(d^4)$ arithmetic operations. This bivariate MDP solving algorithm is presented as algorithm BSDiophant below.

3.6.2 The evaluation cost

The drawback of the interpolation method is the evaluation cost it brings. This is especially the case when the number of terms and the number of the indeterminates of the polynomial that we want to evaluate is high. In our experiments we found that the sparse interpolation approach we propose reduces the time spent solving MDP's but evaluation becomes the most time dominating part of the factoring algorithm.

Suppose $f = \sum_{i=1}^{s} c_i X_i Y_i$ where X_i is a monomial in x_1, x_2, Y_i is a monomial in $x_3, \ldots, x_n$, $0 \neq c_i \in \mathbb{Z}_p$ and we want to compute

$$f_j := f(x_1, x_2, x_3 = \alpha_3^j, \ldots, x_n = \alpha_n^j), \text{ for } j = 1, \ldots, t.$$

To compute f_j efficiently, one way is to pre-compute the powers of α_i's in $(n-2)$ tables and then do the evaluation using tables. We implemented this first. Let $d_i = \deg(f, x_i)$ and $d = \max_{3 \leq i \leq n} d_i$. For a fixed j, computing the $n-2$ tables of powers of α_i^j's (i.e. $1, \alpha_i^j, \alpha_i^{2j}, \ldots, \alpha_i^{d_i j}$) costs $\leq (n-2)d$ multiplications. To evaluate one term $c_i Y_i$ at $(\alpha_3^j, \ldots, \alpha_n^j)$ costs $n-2$ multiplications using the tables. Then the cost of evaluating f at $(\alpha_3^j, \ldots, \alpha_n^j)$ is $s(n-2)$ multiplications. Hence the total cost of t evaluations is bounded above by $C_T = s(n-2)t + (n-2)dt = t(n-2)(s+d)$ multiplications using tables.

However when we use sparse interpolation points of the form $(\alpha_3^j, \ldots, \alpha_n^j)$ for $j = 1, \ldots, t$ [Zip90] we can reduce the evaluation cost by a factor of $(n-2)$ by a simple organization. As an example suppose

$$f = x_1^{22} + 72x_1^3x_2^4x_4x_5 + 37x_1x_2^5x_3^2x_4 - 92x_1x_2^5x_5^2 + 6x_1x_2^3x_3x_4^2$$

and we want to compute $f_j := f(x_1, x_2, \alpha_3^j, \alpha_4^j, \alpha_5^j)$ for $1 \leq j \leq t$. Before combining and sorting, we write the terms of each f_j as

$$\begin{aligned} f_j &= x_1^{22} + 72\alpha_4^j\alpha_5^j x_1^3x_2^4 + 37(\alpha_3^j)^2\alpha_4^j x_1x_2^5 - 92(\alpha_5^j)^2 x_1x_2^5 + 6\alpha_3^j(\alpha_4^j)^2 x_1x_2^3 \\ &= x_1^{22} + 72(\alpha_4\alpha_5)^j x_1^3x_2^4 + 37(\alpha_3^2\alpha_4)^j x_1x_2^5 - 92(\alpha_5^2)^j x_1x_2^5 + 6(\alpha_3\alpha_4^2)^j x_1x_2^3. \end{aligned}$$

Now let

$$c^{(0)} := [1, 72, 37, -92, 6] \quad \text{and} \quad \theta := [1, \alpha_4\alpha_5, \alpha_3^2\alpha_4, \alpha_5^2, \alpha_3\alpha_4^2].$$

Then in a for loop $j = 1, \ldots, t$ we can update the coefficient array $c^{(0)}$ by the monomial array θ by defining $c_i^{(j)} = c_i^{(j-1)}\theta_i$ for $1 \leq i \leq s$ so that each iteration computes the coefficient array

$$c^{(j)} = [1, 72(\alpha_4\alpha_5)^j, 37(\alpha_3^2\alpha_4)^j, -92(\alpha_5^2)^j, 6(\alpha_3\alpha_4^2)^j]$$

Algorithm 7 SparseInt

Input : A big prime p and $u, w, c \in \mathbb{Z}_p[x_1, x_2, \ldots, x_j]$ and $\mathrm{Supp}(\sigma)$ the support of $\sigma \in \mathbb{Z}_p[x_1, x_2, \ldots, x_j]$ and u, w are monic in x_1.

Output : The solution (σ, τ) for the MDP $\sigma u + \tau w = c \in \mathbb{Z}_p[x_1, x_2, \ldots, x_j]$ or FAIL

1: Let $\sigma_f = \sum_{0 \le i+k \le d} c_{ik}(x_3, ..., x_j) x_1^i x_2^k$ where $c_{ik} = \sum_{l=1}^{s_{ik}} c_{ikl} M_{ikl}$ with c_{ikl} unknown coefficients to be solved for and $M_{ikl} \in \mathrm{Supp}(\sigma)$.

2: Let $s = \max s_{ik} = \max \#c_{ik}$.

3: Pick $(\alpha_3, \ldots \alpha_j) \in (\mathbb{Z}_p \backslash \{0\})^{j-2}$ at random.

4: Compute monomial evaluation sets

$$\{S_{ik} = \{m_{ikl} = M_{ikl}(\alpha_3, \ldots, \alpha_j) : 1 \le l \le s_{ik}\} \text{ for } 0 \le i + k \le d\}.$$

If $|S_{ik}| \neq s_{ik}$ for some ik try a different choice for $(\alpha_3, \ldots, \alpha_j)$. If this fails repeatedly, **return** FAIL. (*p is not big enough*)

5: **for** i from 1 to s **do** (*Compute the bivariate images of σ*)

6: Let $Y_i = (x_3 = \alpha_3^i, \ldots, x_j = \alpha_j^i)$.

7: Evaluate $u(x_1, x_2, Y_i), w(x_1, x_2, Y_i), c(x_1, x_2, Y_i)$ by the method explained in section 5.

8: Solve $\sigma_i(x_1, x_2) u(x_1, x_2, Y_i) + \tau_i(x_1, x_2) w(x_1, x_2, Y_i) = c(x_1, x_2, Y_i)$ in $\mathbb{Z}_p[x_1, x_2]$ for $\sigma_i(x_1, x_2)$ using BDP (see Section 3.6).

9: **if** BDP returns FAIL **then**

10: **return** FAIL *(Y_i is unlucky or there is no solution to the BDP).*

11: **end if**

12: **end for**

13: **for** $0 \le i + k \le d$ **do**

14: Construct and solve the $s_{ik} \times s_{ik}$ linear system

$$\left\{ \sum_{l=1}^{s_{ik}} c_{ikl}\, m_{ikl}^n \;=\; \text{coefficient of } x_1^i x_2^k \text{ in } \sigma_n(x_1, x_2) \;\text{ for } 1 \le n \le s_{ik} \right\}$$

for the coefficients c_{ikl} of $c_{ik}(x_3, \ldots, x_j)$. Because it is a Vandermonde system in m_{ikl} which are distinct by Step 4 it has a unique solution.

15: **end for**

16: Substitute the values c_{ikl} in σ_f to get σ

17: **if** $w \,|\, (c - \sigma u)$ **then**

18: Set $\tau = (c - \sigma u)/w$ and **return** (σ, τ)

19: **else**

20: **return** FAIL *(σ_f is wrong)*

21: **end if**

Algorithm 8 BSDiophant

Input A big prime p and $u, w, c \in \mathbb{Z}_p[x_1, x_2, \ldots, x_j]$ where the MDP conditions satisfied.
Output $(\sigma, \tau) \in \mathbb{Z}_p[x_1, x_2, \ldots, x_j]$ such that $\sigma u + \tau w = c \in \mathbb{Z}_p[x_1, x_2, \ldots, x_j]$ or **FAIL** . It returns **FAIL** if condition 2 of the MDP is not satisfied for the choice of any β in the algorithm. This is detected in subroutine BDP.

1: **if** $n = 2$ **then** call BDP to **return** $(\sigma, \tau) \in \mathbb{Z}_p[x_1, x_2]^2$ or FAIL **end if**.
2: Pick $\beta_1 \in \mathbb{Z}_p$ at random
3: $(u_{\beta_1}, w_{\beta_1}, c_{\beta_1}) \leftarrow (u(x_j = \beta_1), w(x_j = \beta_1), c(x_j = \beta_1)) \in \mathbb{Z}_p[x_1, x_2, \ldots, x_{j-1}]^3$.
4: $(\sigma_1, \tau_1) \leftarrow$ **BSDiophant**$(u_{\beta_1}, w_{\beta_1}, c_{\beta_1}, p)$.
5: **if** $\sigma_1 =$ FAIL **then return** FAIL **end if**
6: $k \leftarrow 1$, $\sigma \leftarrow \sigma_1$, $q \leftarrow (x_j - \beta_1)$ and $\sigma_f \leftarrow$ skeleton of σ_1.
7: **repeat**
8: $h \leftarrow \sigma$
9: Set $k \leftarrow k + 1$ and pick $\beta_k \in \mathbb{Z}_p$ at random distinct from $\beta_1, \ldots, \beta_{k-1}$
10: $(u_{\beta_k}, w_{\beta_k}, c_{\beta_k}) \leftarrow (u(x_1, \ldots, x_j = \beta_k), w(x_1, \ldots, x_j = \beta_k), c(x_1, \ldots, x_j = \beta_k)$.
11: $(\sigma_k, \tau_k) \leftarrow$**SparseInt**$(u_{\beta_k}, w_{\beta_k}, c_{\beta_k}, \sigma_f, p)$ (Algorithm 7)
12: **if** $\sigma_k =$ FAIL **then return** FAIL **end if**
13: Solve $\sigma = h \bmod q$ and $\sigma = \sigma_k \bmod (x_j - \beta_k)$ for $\sigma \in \mathbb{Z}_p[x_1, x_2, \ldots, x_j]$.
14: $q \leftarrow q \cdot (x_j - \beta_k)$
15: **until** $\sigma = h$ and $w|(c - \sigma u)$
16: Set $\tau \leftarrow (c - \sigma u)/w$ and **return** (σ, τ).

using $s = \#f$ multiplications in the coefficient field to obtain

$$f_j = x_1^{22} + 72(\alpha_4\alpha_5)^j x_1^3 x_2^4 + 37(\alpha_3^2\alpha_4)^j x_1 x_2^5 - 92(\alpha_5^2)^j x_1 x_2^5 + 6(\alpha_3\alpha_4^2)^j x_1 x_2^3.$$

Then sorting the monomials and combining terms we get

$$f_j = x_1^{22} + 72(\alpha_4\alpha_5)^j x_1^3 x_2^4 + (37(\alpha_3^2\alpha_4)^j - 92(\alpha_5^2)^j) x_1 x_2^5 + 6(\alpha_3\alpha_4^2)^j x_1 x_2^3.$$

Note that sorting is time consuming so it should be done once at the beginning. Then compute the arrays $c^{(j)}$ and then combine according to the sorting rule. In the example above by looking at the terms of f we know that after the evaluation the first and the second, also the third and the fourth terms of f will collide. Hence after computing each $c^{(j)}$ we know that the sum of the first and the second, also the third and the fourth terms of each array will correspond to the coefficients of f_j, so we won't spend time to sort the terms of each unsorted f_j.

With the organization described above one evaluates Y_i at $(\alpha_3, \ldots, \alpha_n)$ in $(n-3)$ multiplications using tables. The cost of $n-2$ tables of powers is $\leq (n-2)d$. Then at the first step the cost of (creating monomial array) is $\leq s(n-3) + (n-2)d$. After that the cost of each t evaluations is st multiplications. Hence the total cost is upper bounded by $C_N = st + s(n-3) + (n-2)d$. Then $C_T - C_N = (t-1)((n-2)d + (n-3)s) > (t-1)(n-2)d$. Hence the relative gain is approximately a factor of $(n-2)$.

For a recent research along this direction, see [MW17]. See also [BLS03]. The complexity, $C_N \in \mathcal{O}(st)$ can be reduced to $\mathcal{O}(s \log^2 t)$ by using FFT based methods.

Chapter 4

Sparse Hensel Lifting (SHL)

4.1 The main assumption of SHL

We start Chapter 4 by recalling the notation. Also, by now so as not to complicate the explanation we will assume that the input polynomial to the MHL algorithm is monic and $l = 1$.

Suppose that we seek to factor a multivariate polynomial $a \in R = \mathbb{Z}[x_1, \ldots, x_n]$ and $a = fg$ with f, g in R. As explained in Chapter 2, the multivariate Hensel lifting algorithm (MHL) developed by Yun [Yun74] and improved by Wang [Wan78] uses a prime number p and an ideal $I = \langle x_2 - \alpha_2, \ldots, x_n - \alpha_n \rangle$ of $\mathbb{Z}_p[x_1, \ldots, x_n]$ where $\alpha_2, \alpha_3, \ldots, \alpha_n \in \mathbb{Z}_p$ is a random evaluation point chosen by the algorithm. As discussed in Section 2.1 we want $\alpha_i = 0$ if possible.

For a given polynomial $h \in R$, let us use the notation

$$h_j := h(x_1, \ldots, x_j, x_{j+1} = \alpha_{j+1}, \ldots, x_n = \alpha_n) \bmod p$$

so that $a_1 = a(x_1, \alpha_2, \ldots, \alpha_n) \bmod p$. The input to MHL is a, I, f_1, g_1 and p such that $a_1 = f_1 g_1$ and $\gcd(f_1, g_1) = 1$ in $\mathbb{Z}_p[x_1]$. The input factorization $a_1 = f_1 g_1$ is obtained by factoring $a(x_1, \alpha_2, \ldots, \alpha_n)$ over the integers.

Let d_j denote the total degree of a_j with respect to the variables $x_2, \ldots, x_j$ and $I_j = \langle x_2 - \alpha_2, \ldots, x_j - \alpha_j \rangle$ with $j \leq n$. Wang's MHL lifts the factorization $a_1 = f_1 g_1$ variable by variable to $a_j = f_j g_j \in \mathbb{Z}_p[x_1, \ldots, x_j]/I_j^{d_j+1}$. It turns out that $f_n \equiv f \bmod p$ and $g_n \equiv g \bmod p$. For sufficiently large p we recover the factorization of a over $\mathbb{Z}$.

Factoring multivariate polynomials via Sparse Hensel Lifting (SHL) uses the same idea of the sparse interpolation from Section 3.5 . At the $(j-1)^{\text{th}}$ step of MHL we have

$$\begin{aligned} f_{j-1} &= x_1^{df} + c_{j1} M_1 + \cdots + c_{jt_j} M_{t_j} \\ g_{j-1} &= x_1^{dg} + s_{j1} N_1 + \cdots + s_{jr_l} N_{r_l} \end{aligned}$$

where t_j is the number of non-zero terms that appear in f_{j-1}, M_k's and N_l's are distinct monomials in $x_1, \ldots, x_{j-1}$ and $c_{jk}, s_{jl} \in \mathbb{Z}_p$. Then SHL makes a probabilistic assumption that

$$\begin{aligned} f_j &= x_1^{df} + \Lambda_{j1}(x_j)M_1 + \cdots + \Lambda_{jt_j}(x_j)M_{t_j} \\ g_j &= x_1^{dg} + \Gamma_{j1}(x_j)N_1 + \cdots + \Gamma_{jr_l}(x_j)N_{r_l} \end{aligned}$$

for $1 \leq k \leq t_j$ and $1 \leq l \leq r_j$, where

$$\begin{aligned} \Lambda_{jk} &= c_{jk}^{(0)} + c_{jk}^{(1)}(x_j - \alpha_j) + c_{jk}^{(2)}(x_j - \alpha_j)^2 + \cdots + c_{jk}^{(d_{j_k})}(x_j - \alpha_j)^{d_{j_k}} \\ \Gamma_{jl} &= s_{jl}^{(0)} + s_{jl}^{(1)}(x_j - \alpha_j) + s_{jl}^{(2)}(x_j - \alpha_j)^2 + \cdots + s_{jl}^{(d_{j_l})}(x_j - \alpha_j)^{d_{j_l}} \end{aligned}$$

with $c_{jk}^{(0)} := c_{jk}$ and where $df = \deg_{x_1}(f)$, $d_{j_k} = \deg_{x_n}(\Lambda_{jk})$, with $c_{jk}^{(i)} \in \mathbb{Z}_p$ for $0 \leq i \leq d_{j_k}$, and $s_{jl}^{(0)} := s_{jl}$ and where $dg = \deg_{x_1}(g)$, $d_{j_l} = \deg_{x_n}(\Gamma_{jl})$, with $s_{jl}^{(i)} \in \mathbb{Z}_p$ for $0 \leq i \leq d_{j_l}$.

So for each step j, the assumption of SHL is pictorially

$$\begin{aligned} a_j = f_j g_j &= (x_1^{df} + \Lambda_{j1}M_1 + \cdots + \Lambda_{jt_j}M_{t_j})(x_1^{dg} + \Gamma_{j1}N_1 + \cdots + \Gamma_{jr_j}N_{r_j}) \\ &\Uparrow \\ a_{j-1} = f_{j-1}g_{j-1} &= (x_1^{df} + c_{j1}M_1 + \cdots + c_{jt_j}M_{t_j})(x_1^{dg} + s_{j1}N_1 + \cdots + s_{jr_j}N_{r_j}) \quad (4.1) \end{aligned}$$

where s_{jk}, r_j and N_k are defined as above.

Of course, this assumption is probabilistic. f_j (or g_j) may contain a term ΛM where M is a monomial in $x_1, \ldots, x_{j-1}$, $\Lambda \in \mathbb{Z}_p[x_j]$ and $\Lambda(\alpha_j) = 0$. If $\alpha_j \neq 0$ is chosen random then $\Pr[\Lambda(\alpha_j) = 0] \leq \deg_{x_j}(\Lambda)/p \leq d_j/p$. So at the j^{th} step, this probability is $\leq d_j(r_j + t_j)/p$.

Note that if $\Lambda \in \mathbb{Z}_p[x_j]$ is a monomial then $\alpha_j \neq 0 \Rightarrow \Lambda(\alpha_j) \neq 0$. Therefore, if the number of terms of f, g is T_f, T_g resp. then at the j^{th} step the probability that SHL assumption is wrong is $\leq d(r_j + t_j)/2p \leq d(T_f + T_g)/2p$. Hence, in total, the probability that SHL assumption is wrong is $\leq d(n-1)(T_f + T_g)/2p$.

Therefore, if the factors to be computed are sparse and the evaluation points $(\alpha_2, \ldots, \alpha_n)$ are random and p is big then the SHL assumption is reasonable and true with high probability. In fact, for many practical sparse problems the probability bound given above is very generous. (See Chapter 5).

We close this section with a historical remark. In [Zip93] Zippel claims that, although the sparse interpolation algorithm is more widely known than the SHL discussed in this section, the SHL dates to the spring of 1977. Trager suggested to Zippel that the same techniques could be used for modular interpolation in May 1978.

4.2 Sparse Hensel Lifting by Zippel (ZSHL)

Zippel's sparse Hensel lifting (ZSHL) is a combination of the SHL idea discussed in the previous section and the multivariate version of Newton's iteration [Zip81, Zip93]. It does not follow the MHL procedure described by Zassenhaus. We will briefly sketch the idea and give a concrete example to show how it works. For more detail see [Zip81].

The reader may want to read the explanation below along with the Example 26.

Recall the Eqn (4.1). At the j^{th} step of MHL we have the knowledge of a_j and by looking at the factorization of a_{j-1}, we have the assumption

$$a_j = (x_1^{df} + \Lambda_{j1}M_1 + \cdots + \Lambda_{jt_j}M_{t_j})(x_1^{dg} + \Gamma_{j1}N_1 + \cdots + \Gamma_{jr_j}N_{r_j}). \tag{4.2}$$

Expanding the RHS of Eqn (4.2) and equating the coefficients of the monomials in $x_2, \ldots, x_{j-1}$ on the LHS and RHS of Eqn (4.2), the aim of ZSHL is to compute $\Lambda_{jk}, \Gamma_{jl}$ by using the multivariate version of Newton's iteration and sparse interpolation techniques discussed in Section 3.5.

Now, in a more general setting, let $R = \mathbb{Z}_p[x_2, \ldots, x_n]$ and we are given a system of polynomial equations over R

$$\begin{aligned} f_1(\theta_1, \ldots, \theta_m) &= p_1(x_2, \ldots, x_m) \\ &\vdots \\ f_n(\theta_1, \ldots, \theta_m) &= p_n(x_2, \ldots, x_m) \end{aligned} \tag{4.3}$$

where $m \leq n$ and p_i are polynomials in R. In addition suppose we are given an oracle that provides a solution to this system modulo $I = \langle x_i - \alpha_i,\ i = 2, \ldots, n \rangle$ for any choice of $\alpha_i \in \mathbb{Z}_p$. Let us also be given bounds on the degree of each variable x_j of θ_i's.

Our aim is to solve Eqn (4.3) for θ_i's, i.e. if we define $g_i = f_i(\theta_1, \ldots, \theta_m) - p_i(x_2, \ldots, x_m)$ then $g_i \in R[\theta_1, \ldots, \theta_m]$ and we are looking for a common root of $\{g_i\}$ in R^m.

The following proposition is by Zippel [Zip93].

Proposition 25. *Let $\vec{g} = (g_i(\theta_1, \ldots, \theta_n),\ i = 1, \ldots, n)$ be a set of polynomials defined in an integral domain R with a prime ideal $\wp$. Let $\mathcal{J}$ denote their Jacobian with respect to θ_i. Let $\vec{s} = (s_1, \ldots, s_n)$ be a solution of $\vec{g}$ modulo $\wp$ such that the determinant of $\mathcal{J}(s_1, \ldots, s_n)$ has an inverse in $R/\wp$. Then there exist unique elements $\hat{s} = (\hat{s}_1, \ldots, \hat{s}_n)$ of $R_\wp$ with $\hat{s}_i = s_i \mod \wp$ for which $\vec{g}(\hat{s}_1, \ldots, \hat{s}_n) = 0$.*

Proof. Consider g_j for some $j \in \{1, \ldots, n\}$. Its Taylor series expansion at $\vec{s} \in R^m$ is

$$\begin{aligned} g_j &= g_j(\vec{s}) + \sum_{i=1}^{n} \frac{\partial g_j}{\partial \theta_i}(\vec{s})(\theta_i - s_i) \\ &+ \frac{1}{2}\sum_{i=1}^{n}\sum_{k=1}^{n} \frac{\partial^2 g_j}{\partial \theta_i \partial \theta_k}(\vec{s})(\theta_i - s_i)(\theta_k - s_k) + \cdots \end{aligned} \tag{4.4}$$

The first summation in Eqn(4.4) is the inner product of the vectors

$$\frac{\partial g_j}{\partial \vec{\theta}}(\vec{s}) = (\frac{\partial g_j}{\partial \theta_1}(\vec{s}), \ldots, \frac{\partial g_j}{\partial \theta_n}(\vec{s})) \text{ and } (\vec{\theta} - \vec{s}) = \begin{pmatrix} \theta_1 - s_1 \\ \vdots \\ \theta_n - s_n \end{pmatrix}.$$

Using this notation the first two terms of Eqn (4.4) is

$$g_j(\vec{\theta}) = g_j(\vec{s}) + \frac{\partial g_j}{\partial \vec{\theta}}(\vec{s}) \cdot (\vec{\theta} - \vec{s}) + \cdots$$

Now more generally $\overrightarrow{g} = (g_i(\theta_1, \ldots, \theta_n),\ i = 1, \ldots, n)$ is a map from $R^n \to R^n$. Since R is an integral domain and $\wp$ is prime, the natural map $R \to R_\wp$ is injective, $\vec{g}$ induces a map $R_\wp^n \to R_\wp^n$ which can locally be expressed as

$$\vec{g}(\vec{\theta}) = \vec{g}(\vec{s}) + \mathcal{J}(\vec{s}) \cdot (\vec{\theta} - \vec{s}) + \cdots$$

where $\mathcal{J}(\vec{s})$ is the Jacobian matrix

$$\mathcal{J}(\vec{s}) = \frac{\partial \vec{g}}{\partial \vec{\theta}} = \begin{pmatrix} \frac{\partial g_1}{\partial \theta_1} & \frac{\partial g_1}{\partial \theta_2} & \cdots & \frac{\partial g_1}{\partial \theta_n} \\ \frac{\partial g_2}{\partial \theta_1} & \frac{\partial g_2}{\partial \theta_2} & \cdots & \frac{\partial g_2}{\partial \theta_n} \\ \vdots & & \ddots & \vdots \\ \frac{\partial g_n}{\partial \theta_n} & \frac{\partial g_n}{\partial \theta_2} & \cdots & \frac{\partial g_n}{\partial \theta_n} \end{pmatrix}.$$

Now let $\vec{\alpha} \in R_\wp^n$ be in the kernel of $\vec{g}$. If we define $\vec{\alpha}^{(0)} = \vec{s}$ and $\vec{\alpha}^{(k)}$ as the image of $\vec{\alpha}$ in $(R/\wp^{(k+1)})^n$ then $\vec{g}(\vec{\alpha}) = \vec{g}(\vec{\alpha}^{(k)}) \mod \wp^k$. Now, by expanding $\vec{g}$ at $\vec{\alpha} - \vec{\alpha}^{(k-1)}$ we get

$$0 = \vec{g}(\vec{\alpha}^{(k-1)}) + \mathcal{J}(\vec{\alpha}^{(k-1)}) \cdot (\vec{\alpha} - \vec{\alpha}^{(k-1)}) + \mathcal{O}((\vec{\alpha} - \vec{\alpha}^{(k-1)})^2). \tag{4.5}$$

The term $\mathcal{O}((\vec{\alpha} - \vec{\alpha}^{(k-1)})^2)$ is in $\wp^2$. If the rank of the Jacobian $\mathcal{J}(\vec{s})$ is n, then it is invertible and we get

$$\vec{\alpha} - \vec{\alpha}^{(k-1)} = -\mathcal{J}(\vec{s})^{-1} \cdot (\vec{\alpha} - \vec{\alpha}^{(k-1)}) \mod \wp^k. \tag{4.6}$$

If we define $\hat{s}$ as the projective limit of the coherent sequence $\vec{\alpha}^{(k)}$'s constructed by an iterative use of Eqn(4.6), we have the desired solution. To see the uniqueness, assume that

$(\hat{y}_1, \ldots, \hat{y}_n)$ also satisfies the proposition and k is the smallest value for which

$$\hat{x}_i \neq \hat{y}_i \mod \wp^{k+1} \tag{4.7}$$

for some i. Note that $k > 0$ and

$$(\hat{s}_1, \ldots, \hat{s}_n) = (\hat{y}_1, \ldots, \hat{y}_n) = \vec{\alpha}^{(k-1)} \mod \wp^k.$$

But then

$$\begin{aligned}(\hat{s}_1, \ldots, \hat{s}_n) - \vec{\alpha}^{(k-1)} &= -\mathcal{J}(\vec{\alpha}^{(0)})^{-1} \cdot (\vec{\alpha} - \vec{\alpha}^{(k-1)}) \mod \wp^{k+1} \\ (\hat{y}_1, \ldots, \hat{y}_n) - \vec{\alpha}^{(k-1)} &= -\mathcal{J}(\vec{\alpha}^{(0)})^{-1} \cdot (\vec{\alpha} - \vec{\alpha}^{(k-1)}) \mod \wp^{k+1}\end{aligned}$$

which contradicts to 4.7. □

Based on Proposition 25, ZSHL applies a recursive procedure: At the k^{th} stage we have a system of the form

$$\begin{aligned}f_1(\theta_1, \ldots, \theta_m) &= p_1(x_2, \ldots, x_k) \\ &\vdots \\ f_n(\theta_1, \ldots, \theta_m) &= p_n(x_2, \ldots, x_k).\end{aligned} \tag{4.8}$$

Next we choose $\alpha_k \in \mathbb{Z}_p$ and solve Eqn(4.8) modulo $(x_k - \alpha_k)$ to get

$$\theta_i = c_{i1}M_1 + \cdots + c_{it_i}M_{t_i} \; i = 1, \ldots, n.$$

Let $T = \sum_{i=1}^{n} t_i$. Then we introduce T new variables $\Lambda_{ij}, 1 \leq j \leq t_i$, one for each possible coefficient θ_i. By using the sparse interpolation idea, each of the equations in Eqn (4.8) is now re-written using Λ_{ij}:

$$f_j(c_{i1}M_1 + \cdots + c_{it_i}M_{t_i}, \ldots, c_{i1}M_1 + \cdots + c_{it_i}M_{t_i}) = p_1(x_1, \ldots, x_{k-1}; x_k) \tag{4.9}$$

By equating the coefficients of the monomials in $x_1, \ldots, x_{k-1}$ on the LHS and RHS of Eqn(4.9) we get a new system of equations

$$\begin{aligned}g_1(\Lambda_{11}, \ldots, \Lambda_{mt_m}) &= q_1(x_k) \\ &\vdots \\ g_N(\Lambda_{11}, \ldots, \Lambda_{mt_m}) &= q_N(x_k)\end{aligned} \tag{4.10}$$

From Eqn (4.8) we know that

$$\Lambda_{ij} = c_{ij} \mod (x_k - \alpha_k) \tag{4.11}$$

If the Jacobian of Eqn (4.10) is not zero, we use this as a starting point for multivariate Newton's iteration to lift Eqn (4.11) to a solution modulo $(x_k - \alpha_k)^l$ for any l:

$$\Lambda_{ij} = c_{ij} + c_{ij}^{(1)}(x_k - \alpha_k) + c_{ij}^{(2)}(x_k - \alpha_k)^2 + \cdots$$

After sufficiently many iterations we expand and rewrite Λ_{ij} as

$$\Lambda_{ij} = d_{ij} + d_{ij}^{(1)} x_k + d_{ij}^{(2)} x_k^2 + \cdots$$

and finally the solution to Eqn (4.8) is computed as

$$\theta_i = (d_{i1} + d_{i1}^{(1)} x_k + d_{i1}^{(2)} x_k^2 + \cdots)M_1 + \cdots + (d_{it_i} + d_{it_i}^{(1)} x_k + d_{it_i}^{(2)} x_k^2 + \cdots)M_{t_i}$$

For some problems there will be more equations than unknowns and thus the Jacobian is not a square. In this case a subset of these equations is used. Since this is a probabilistic approach at every stage k, the solutions obtained should be verified.

Now we give a concrete example how to use this procedure to factor a multivariate polynomial.

Example 26. Suppose that our aim is to factor

$$F = {x_1}^3 + x_1{x_2}^3{x_3}^3 - {x_1}^2{x_2}^2x_3 - {x_1}^2x_2{x_3}^2 + 6\,x_1x_2{x_3}^3 - {x_2}^2{x_3}^2 - 6\,{x_1}^2x_3 + x_1x_2. \tag{4.12}$$

We choose $p = 31$, evaluation points $\alpha_2 = 2, \alpha_3 = 1$ and then compute

$$F(x_1, x_2 = 2, x_3 = 1) \;=\; (x_1 - 2)\left({x_1}^2 - 10\,x_1 + 2\right). \tag{4.13}$$

Following the sparse interpolation idea we assume that

$$F = (x_1 - \theta_1)\left({x_1}^2 - \theta_2 x_1 + \theta_3\right) \tag{4.14}$$

where $\theta_i = \theta_i(x_2, x_3)$ for $1 \le i \le 3$. Equating coefficients in x_1 of Eqn (4.12) with Eqn (4.14) we obtain the set of equations

$$E = [\theta_1\theta_3 = {x_2}^2{x_3}^2,\ \theta_1\theta_2 + \theta_3 = {x_2}^3{x_3}^3 + 6\,x_2{x_3}^3 + x_2,\ \theta_1 + \theta_2 = {x_2}^2x_3 + x_2{x_3}^2 + 6\,x_3].$$

To solve this system of equations for $\theta_i \in \mathbb{Z}_p[x_2, x_3]$, we first reduce to modulo $x_3 - 1$ by defining $w_i = \theta_i(x_2, x_3 = 1)$ for $1 \leq i \leq 3$ and obtain

$$G = E(x_3 = 1) = [{x_2}^2 - w_1 w_3 = 0,\ {x_2}^3 - w_1 w_2 - w_3 + 7\, x_2 = 0,\ {x_2}^2 - w_1 - w_2 + x_2 + 6 = 0].$$

Before we start the iteration, first we compute the Jacobian $\partial G_i / \partial w_j$ for $1 \leq i, j \leq 3$

$$J = \begin{bmatrix} -w_3 & 0 & -w_1 \\ w_2 & w_1 & 1 \\ -1 & -1 & 0 \end{bmatrix}.$$

Note that by construction none of the terms of G_i will be in the form $w_j x_2^k$. So when taking the derivatives $\partial G_i / \partial w_j$, the terms containing the variable x_2 will disappear and the entries of J will be formed by polynomials in w_i's. We have $\det(J) = -{w_1}^2 + w_1 w_2 - w_3$. We need to start the iteration with the choice of w_i such that $\det(J) \neq 0 \mod p$. We start by choosing $w = [3, 15, 3]$ and then compute $\det(J(w)) = 2 \mod p$ and

$$J_w = J(w) = \begin{bmatrix} -3 & 0 & -3 \\ 15 & 3 & 1 \\ -1 & -1 & 0 \end{bmatrix} \mod p \Rightarrow J_w^{-1} = \begin{bmatrix} 16 & 17 & 20 \\ 15 & 14 & 10 \\ 25 & 14 & 11 \end{bmatrix} \mod p.$$

So we start by initializing the array $s = [3, 15, 3]$ and start the multiterm version of Newton's iteration (all computations are in $\mod p$).

1. For j from 1 to 2 do
2. $t_i \leftarrow G_i(w_i = s_i)$ for $1 \leq i \leq 3$.
3. $q_i \leftarrow t_i / (x_2 - 3)^j$ for $1 \leq i \leq 3$.
4. $Q \leftarrow [q_1\, q_2\, q_3]^T$
5. $c \leftarrow J_w^{-1} \cdot Q$
6. $C \leftarrow c(x_2 = 3)$.
7. $s_i \leftarrow s_i + C_i (x_2 - 3)^j$
8. End for loop.

After the iteration we have $s = [x_2, x_2^2 + 6, x_2]$. Hence we conclude that

$$w = [x_2, x_2^2 + 6, x_2].$$

This time again following the sparse interpolation idea we assume that

$$\theta_1 = h_1(x_3)x_2,\ \theta_2 = h_2(x_3)x_2^2 + h_3(x_3),\ \theta_3 = h_4(x_3)x_2.$$

Now substituting these into E we obtain

$$L = [-{x_2}^2{x_3}^2 = -h_1{x_2}^2h_4, {x_2}^3{x_3}^3 + 6\,x_2{x_3}^3 + x_2 = h_1h_2{x_2}^3 + h_1h_3x_2 + h_4x_2, \\ -{x_2}^2x_3 - x_2{x_3}^2 - 6\,x_3 = -h_2{x_2}^2 - h_1x_2 - h_3].$$

From L we deduce

$$K = [-{x_3}^2 = -h_1h_4, 6\,{x_3}^3+1 = h_1h_3+h_4, {x_3}^3 = h_1h_2, -6\,x_3 = -h_3, -{x_3}^2 = -h_1, -x_3 = -h_2].$$

Following the same procedure this time by 4×4 Jacobian matrix which has a non zero determinant at the chosen initializations for h_i, $1 \leq i \leq 4$, as above we get

$$h_1 = x_3^2,\ h_2 = x_3,\ h_3 = 6x_3,\ h_4 = 1.$$

Finally substituting these equations into Eqn (4.14) we get

$$F = \left(x_1 - x_2{x_3}^2\right)\left({x_1}^2 - x_3\left({x_2}^2 + 6\right)x_1 + x_2\right).$$

This is in-fact the true factorization.

4.2.1 Some remarks on ZSHL

In [Zip81], it is claimed that this organization of Hensel lifting, for all practical purposes does not exhibit the exponential behaviour of Wang's incremental design of MHL. However it is a question how efficient this algorithm will be for practical purposes. ZSHL is trying to recover one variable at a time, and at each stage, tries to solve all diophantine equations at once with the help of the Jacobian matrix (See [Zip93]). Even for this small example Wang's incremental design behaves better then ZSHL without the SHL assumption. When we consider the examples where the number of terms of each factor is 100 terms, at the last step the Jacobian matrix becomes (200×200). In general if the factors f, g to be computed have s, t terms respectively, then at the last step the expected size of the Jacobian matrix is $(s+t) \times (s+t)$. Hence, extracting the equations, forming the Jacobian matrix and inverting it takes too much time. This hidden cost is not negligible according to our experiments. There is no complexity analysis given in [Zip81] for this method.

Although the idea of ZSHL is important as a first step to factor sparse polynomials, according to our experiments, in practice, this method is not effective. Next, we review a simpler and a better approach to SHL proposed by Kaltofen in [Kal85].

4.3 Sparse Hensel Lifting by Kaltofen (KSHL)

In a classical implementation of MHL, at the j^{th} step just before the loop one applies the leading coefficient correction (LCC). Kaltofen's proposal for SHL in [Kal85] follows the incremental design of MHL by Wang and offers a different LCC from Wang's LCC. It is done in the for loop. As we noted earlier we will be using Wang's LCC (before the for loop). So, to have a fair comparison with the solution that we will propose in the next section, from now on we will explain and develop an optimized version of Kaltofen's SHL for the monic case.

The reader may want to read the following discussion along with Example 27.

Following the same notation introduced in Section 4.1, let at $(j-1)^{\text{th}}$ step we have

$$f_{j-1} = c_{j1}M_1 + \cdots + c_{jt_j}M_{t_j}$$

where t_j is the number of non-zero terms that appear in f_{j-1}, M_k's are the distinct monomials in $x_1, \ldots, x_{j-1}$ and $c_{jk} \in \mathbb{Z}_p$ for $1 \leq k \leq t_j$. Then at the j^{th} step SHL assumes

$$f_j = \Lambda_{j1}M_1 + \cdots + \Lambda_{jt_j}M_{t_j}$$

where for $1 \leq k \leq t_j$

$$\Lambda_{jk} = c_{jk}^{(0)} + c_{jk}^{(1)}(x_j - \alpha_j) + c_{jk}^{(2)}(x_j - \alpha_j)^2 + \cdots + c_{jk}^{(d_{j_k})}(x_j - \alpha_j)^{d_{j_k}}.$$

with $c_{jk}^{(0)} := c_{jk}$ and where $d_{j_k} = \deg_{x_n}(\Lambda_{jk})$ with $c_{jk}^{(i)} \in \mathbb{Z}_p$ for $0 \leq i \leq d_{j_k}$. The assumption is the same for the factor g_{j-1}.

To recover f_j from f_{j-1} and g_j from g_{j-1}, during the j^{th} step of MHL (see Algorithm 5) one starts with $\sigma_{j0} = f_{j-1}$, $\tau_{j0} = g_{j-1}$, then in a for loop starting from $i = 1$ and incrementing it while the error term and its i^{th} Taylor coefficient is non-zero, by solving MDP's

$$\sigma_{j0}\tau_{ji} + \tau_{j0}\sigma_{ji} = e_j^{(i)} \quad \text{in} \quad \mathbb{Z}_p[x_1, \ldots, x_{j-1}] \tag{4.15}$$

for $1 \leq i \leq \max(\deg_{x_j}(f_j), \deg_{x_j}(g_j))$.

After the loop terminates we have $f_j = \sum_{k=0}^{\deg_{x_j}(f_j)} \sigma_{jk}(x_j - \alpha_j)^k$. On the other hand if the assumption of SHL is true then we have also

$$\begin{aligned} f_j &= x_1^{df} + (\sum_{i=0}^{d_j} c_{j1}^{(i)}(x_j - \alpha_j)^i)M_1 + \cdots + (\sum_{i=0}^{d_j} c_{jt_j}^{(i)}(x_j - \alpha_j)^i)M_{t_j} \\ &= x_1^{df} + \sum_{i=0}^{d_j}(c_{j1}^{(i)}M_1 + \cdots + c_{jt_j}^{(i)}M_{t_j})(x_j - \alpha_j)^i. \end{aligned}$$

Similarly for g_j. Hence we see that if the assumption of SHL is true then the support of each σ_{jk} will be a subset of the support of f_{j-1}. Therefore we can use f_{j-1} as the skeleton of the solution of each σ_{jk}. The same is true for τ_{jk}.

Now let $g_{j-1} = x_1^{dg} + s_{j1}N_1 + \cdots + s_{jr_j}N_{r_j}$. To solve the MDP's in Eqn(4.15) at the j^{th} step, before the loop, we introduce unknowns u_k for $1 \leq k \leq t_j + r_j$ and assume that

$$\begin{aligned} \sigma_{ji} &= u_1 M_1 + \cdots + u_{t_j} M_{t_j} \\ \tau_{ji} &= u_{t_j+1} N_1 + \cdots + u_{t_j+r_j} N_{r_j}. \end{aligned}$$

After this assumption, by expanding Eqn (4.15) and equating the coefficients of the same monomials appearing on the LHS and the RHS of Eqn (4.15) one obtains a system of linear equations. By construction these equations will be homogeneous linear equations in the u_k's. Hence one has to solve a system of linear equations to get the factors.

At the j^{th} step of MHL , throughout the loop σ_{j0}, τ_{j0} remains the same. So, if the SHL assumption is true the assumed solution structures of σ_{ji} and τ_{ji} will remain the same on the LHS but only the RHS of Eqn (4.15) will change. Hence just before the loop it is enough to find $r_j + t_j$ linearly independent equations among $\mathcal{O}(r_j t_j)$ linear equations while keeping track of which monomials they correspond. We call this monomial set Mon, construct the corresponding matrix L, and compute L^{-1}.

Then in the for loop, for each i, one simply has to compute the Taylor coefficient of $e_j^{(i)}$ of the error, extract the coefficients of it corresponding to each monomial in Mon, form the related vector v, and then simply compute $w = L^{-1}v$ to recover u_k's.

It may be possible that there is not enough linearly independent equations. In this case we terminate the algorithm. Algorithm 9 below describes the j^{th} step of KSHL.

Before we discuss further how effective this approach is, it is helpful to see a working example.

Example 27. Suppose we seek to factor $a = fg$ where

$$\begin{aligned} f &= {x_1}^5 + 3\,{x_1}^2 x_2 {x_3}^2 - 7\,{x_1}^4 - 4\,x_1 x_3 + 1 \\ g &= {x_1}^5 + {x_1}^2 x_2 x_3 - 7\,{x_3}^4 - 6 \end{aligned}$$

Let $\alpha_3 = 2, p = 2^{31} - 1$. Suppose MHL is at the last step, i.e. $j = 3$ and we want to recover the variable x_3. Before lifting we have a and

$$\begin{aligned} f^{(0)} &:= f(x_3 = 2) = {x_1}^5 - 7\,{x_1}^4 + 12\,{x_1}^2 x_2 - 8\,x_1 + 1 \\ g^{(0)} &:= g(x_3 = 2) = {x_1}^5 + 2\,{x_1}^2 x_2 - 118. \end{aligned}$$

Algorithm 9 j^{th} step of optimized KSHL

Input : $a_j \in \mathbb{Z}_p[x_1, \ldots, x_j]$, $f_{j-1}, g_{j-1} \in \mathbb{Z}_p[x_1, \ldots, x_{j-1}]$ and $\alpha_j \in \mathbb{Z}_p$ where a_j, f_{j-1}, g_{j-1} are monic in x_1. Also, $a_j(x_1, \ldots, x_{j-1}, x_j = \alpha_j) = f_{j-1}g_{j-1}$.
Let $f_{j-1} = x_1^{df} + c_{j1}M_1 + \cdots + c_{jt_j}M_{t_j}$ and $g_{j-1} = x_1^{dg} + s_{j1}N_1 + \cdots + s_{jr_j}N_{r_j}$ where $M_1, \ldots, M_{t_j}$, and $N_1, \ldots, N_{r_j}$ are monomials in $x_1, \ldots, x_{j-1}$ and $df = \deg_{x_1} f$ and $dg = \deg_{x_1} g$. **Output** : $f_j, g_j \in \mathbb{Z}_p[x_1, \ldots, x_j]$ such that $a_j = f_j g_j$
or FAIL (No such factorization exists.)

1: $(\sigma_{j0}, \tau_{j0}) \leftarrow (f_{j-1}, g_{j-1})$.
2: $(\sigma_j, \tau_j) \leftarrow (\sigma_{j0}, \tau_{j0})$.
3: $monomial \leftarrow 1$.
4: Introduce unknowns $u_1, \ldots, u_{r_j+t_j}$ and $D \leftarrow \sigma_{j0}(u_1N_1 + \cdots + u_{r_j}N_{r_j}) + \tau_{j0}(u_{r_j+1}M_1 + \cdots + u_{t_j}M_{t_j})$
5: Expand D and collect the coefficients of the monomials in $x_1, \ldots, x_{j-1}$. Each coefficient is a homogeneous linear equation in u_k's.
6: Let S be the array of all these homogeneous equations and Mon be the array of monomials such that S_i is the coefficient of Mon_i in the expansion of D.
7: Find $i_1, \ldots, i_{r_j+t_j}$ such that $E = \{S_{i_1}, \ldots, S_{i_{r_j+t_j}}\}$ is a linearly independent set. Do this choosing equations of the form of $c\,u_k$ for some constant c first.
8: **if** no such E exists **then return** FAIL (SHL assumption is wrong) **end if**
9: Construct the $(r_j + t_j) \times (r_j + t_j)$ matrix L corresponding to the set E such that the unknown u_i corresponds to i^{th} column of L
10: Compute L^{-1}.
11: $error \leftarrow a_j - f_{j-1}\,g_{j-1}$
12: **for** i from 1 to $\deg(a_j, x_j)$ **while** $error \neq 0$ **do**
13: $\quad monomial \leftarrow monomial \times (x_j - \alpha_j)$
14: $\quad c \leftarrow$ coefficient of $(x_j - \alpha_j)^i$ in the Taylor expansion of the $error$ about $x_j = \alpha_j$
15: $\quad$ **if** $c \neq 0$ **then**
16: $\quad\quad$ **for** k from 1 to $r_j + t_j$ **do**
17: $\quad\quad\quad v_k \leftarrow$ the coefficient of Mon_{i_k} of the polynomial c
18: $\quad\quad$ **end for**
19: $\quad\quad w \leftarrow L^{-1}v$
20: $\quad\quad \sigma_{ji} \leftarrow \sum_{k=1}^{t_j} w_k M_k$ and $\tau_{ji} \leftarrow \sum_{k=1}^{r_j} w_{k+t_j} N_k$.
21: $\quad\quad (\sigma_j, \tau_j) \leftarrow (\sigma_j + \sigma_{ji} \times monomial, \tau_j + \tau_{ji} \times monomial)$.
22: $\quad\quad error \leftarrow a_j - \sigma_j\tau_j$.
23: $\quad$ **end if**
24: **end for**
25: **if** $error \neq 0$ **then return** FAIL **else return** (σ_j, τ_j) **end if**

If the assumption of SHL is true then we have

$$f = \sum_{i=0}^{\deg_{x_3} f} f_i (x_3 - 2)^i \text{ and } g = \sum_{i=0}^{\deg_{x_3} g} g_i (x_3 - 2)^i$$

where each f_i and g_i is in the form

$$\begin{aligned} f_i &= u_1^{(i)} {x_1}^4 + u_2^{(i)} {x_1}^2 x_2 + u_3^{(i)} x_1 + u_4^{(i)} \\ g_i &= u_5^{(i)} {x_1}^2 x_2 + u_6^{(i)} \end{aligned}$$

for some unknowns $C = \{u_1^{(i)}, u_2^{(i)}, u_3^{(i)}, u_4^{(i)}, u_5^{(i)}, u_6^{(i)}\}$.

At the beginning the unknowns are $C = \{u_1, u_2, u_3, u_4, u_5, u_6\}$ in $\mathbb{Z}_p$. We construct

$$\begin{aligned} D = \left({x_1}^5 - 7\,{x_1}^4 + 12\,{x_1}^2 x_2 - 8\,x_1 + 1\right)\left(u_5 {x_1}^2 x_2 + u_6\right) \\ + \left({x_1}^5 + 2\,{x_1}^2 x_2 - 118\right)\left(u_1 {x_1}^4 + u_2 {x_1}^2 x_2 + u_3 x_1 + u_4\right). \end{aligned}$$

Expanding D we see the system of homogeneous linear equations as coefficients

$$\begin{aligned} D = u_1 {x_1}^9 + (u_2 + u_5)\,{x_1}^7 x_2 + (2\,u_1 - 7\,u_5)\,{x_1}^6 x_2 + u_3 {x_1}^6 + (u_4 + u_6)\,{x_1}^5 + (2\,u_2 + 12\,u_5)\,{x_1}^4 {x_2}^2 \\ + (-118\,u_1 - 7\,u_6)\,{x_1}^4 + (2\,u_3 - 8\,u_5)\,{x_1}^3 x_2 + (-118\,u_2 + 2\,u_4 + u_5 + 12\,u_6)\,{x_1}^2 x_2 \\ + (-118\,u_3 - 8\,u_6)\,x_1 - 118\,u_4 + u_6 \end{aligned}$$

We need 6 linearly independent equations from these. First we check whether there are any equations in one unknown. In this example we see that u_1, u_3 corresponding to monomials ${x_1}^9, {x_1}^6$ are there already. Then we go over the equations one by one to get a rank 6 system. In this example we find that equations corresponding to the set

$$Mon = \{{x_1}^9, {x_1}^6, {x_1}^7 x_2, {x_1}^6 x_2, {x_1}^5, {x_1}^4\}$$

are linearly independent. Then we prepare the matrix A

$$A := \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 0 & -7 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ -118 & 0 & 0 & 0 & 0 & -7 \end{bmatrix}$$

In the following $e_3^{(k)}$ denotes the coefficient of $(x_3-2)^k$ in the Taylor expansion of the *error* about $x_3 = 2$. Let also $f_0 := f^{(0)}, g_0 := g^{(0)}, f^{(k)} := \sum_{i=0}^{k} f_i(x_3-2)^i, g^{(k)} := \sum_{i=0}^{k} g_i(x_3-2)^i$.

In the algorithm 4.1, the vector v is constructed by extracting the coefficients of $e_3^{(k)}$ corresponding to monomials in $Mon = \{{x_1}^9, {x_1}^6, {x_1}^7x_2, {x_1}^6x_2, {x_1}^5, {x_1}^4\}$ and $w = A^{-1}v \bmod p$.

Now for the loop, $i = 1$:

$$\begin{aligned}
error &= a - f^{(0)}g^{(0)} \\
e_3^{(1)} &= 13\,{x_1}^7x_2 - 7\,{x_1}^6x_2 - 4\,{x_1}^6 + 36\,{x_1}^4{x_2}^2 - 224\,{x_1}^5 \\
&\quad +1568\,{x_1}^4 - 16\,{x_1}^3x_2 - 4103\,{x_1}^2x_2 + 2264\,x_1 - 224 \\
v &= \begin{bmatrix} 0 & -4 & 13 & -7 & -224 & 1568 \end{bmatrix} \\
w &= \begin{bmatrix} 0 & 12 & -4 & 0 & 1 & -224 \end{bmatrix} \\
f^{(1)} &= f^{(0)} + \left(12\,{x_1}^2x_2 - 4\,x_1\right)(x_3 - 2) \\
&= {x_1}^5 - 7\,{x_1}^4 + 12\,{x_1}^2x_2x_3 - 12\,{x_1}^2x_2 - 4\,x_1x_3 + 1 \\
g^{(1)} &= g^{(0)} + \left({x_1}^2x_2 - 224\right)(x_3 - 2) \\
&= {x_1}^5 + {x_1}^2x_2x_3 - 224\,x_3 + 330
\end{aligned}$$

$i = 2$:

$$\begin{aligned}
error &= a - f^{(1)}g^{(1)} \\
e_3^{(2)} &= 3\,{x_1}^7x_2 + 6\,{x_1}^4{x_2}^2 - 168\,{x_1}^5 + 1176\,{x_1}^4 - 2370\,{x_1}^2x_2 + 1344\,x_1 - 168 \\
v &= \begin{bmatrix} 0 & 0 & 3 & 0 & -168 & 1176 \end{bmatrix} \\
w &= \begin{bmatrix} 0 & 3 & 0 & 0 & 0 & -168 \end{bmatrix} \\
f^{(2)} &= f^{(1)} + 3\,{x_1}^2x_2(x_3 - 2)^2 \\
&= {x_1}^5 + 3\,{x_1}^2x_2{x_3}^2 - 7\,{x_1}^4 - 4\,x_1x_3 + 1 \\
g^{(2)} &= g^{(1)} - 168(x_3 - 2)^2 \\
&= {x_1}^5 + {x_1}^2x_2x_3 - 168\,{x_3}^2 + 448\,x_3 - 342
\end{aligned}$$

$i = 3$:

$$\begin{aligned}
error &= a - f^{(2)}g^{(2)} \\
e_3^{(3)} &= -56\,{x_1}^5 + 392\,{x_1}^4 - 672\,{x_1}^2x_2 + 448\,x_1 - 56 \\
v &= \begin{bmatrix} 0 & 0 & 0 & 0 & -56 & 392 \end{bmatrix} \\
w &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & -56 \end{bmatrix} \\
f^{(3)} &= f^{(2)} + 0 \\
&= {x_1}^5 + 3\,{x_1}^2x_2{x_3}^2 - 7\,{x_1}^4 - 4\,x_1x_3 + 1 \\
g^{(3)} &= g^{(2)} - 56\,(x_3 - 2)^3 \\
&= {x_1}^5 + {x_1}^2x_2x_3 - 56\,{x_3}^3 + 168\,{x_3}^2 - 224\,x_3 + 106
\end{aligned}$$

At the end of the 3rd iteration we have recovered f and g can be obtained by early detection via trial division. But let's go further.

for $i = 4$:

$$\begin{aligned}
e_3^{(4)} &= -7\,{x_1}^5 + 49\,{x_1}^4 - 84\,{x_1}^2x_2 + 56\,x_1 - 7 \\
v &= \begin{bmatrix} 0 & 0 & 0 & 0 & -7 & 49 \end{bmatrix} \\
w &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & -7 \end{bmatrix} \\
f^{(4)} &= f^{(3)} + 0 \\
&= {x_1}^5 + 3\,{x_1}^2x_2{x_3}^2 - 7\,{x_1}^4 - 4\,x_1x_3 + 1 \\
g^{(4)} &= g^{(3)} - 7\,(x_3 - 2)^4 \\
&= {x_1}^5 + {x_1}^2x_2x_3 - 7\,{x_3}^4 - 6
\end{aligned}$$

for $i = 5$, $error = a - f^{(4)}g^{(4)} = 0$ and we have the factors!

4.3.1 Some remarks on KSHL

We have implemented our optimized version KSHL in Maple and made experiments. The most time consuming step is the step 7 of Algorithm 9 where one has to find $r_j + t_j$ linearly independent equations out of $\mathcal{O}(r_j t_j)$ linear equations and invert the corresponding matrix. This is not an easy task, since as we noted one should keep track of the monomial to which the linear equation corresponds. Therefore the most obvious way to get the system is to start with a set of one equation and set the rank= 1, then each time add a new equation to the set if it increases the previous rank by 1, until we reach a full rank $r_j + t_j$. To do this we have used Maple's `RowReduce` function which performs in-place Gaussian elimination on the input mod p Matrix L. This function is implemented in C and optimized. The time complexity is the time complexity of Gaussian elimination $\mathcal{O}((r_j + t_j)^3)$ plus the time complexity of failed cases, which is according to our experiments not negligible.

One can argue that taking random linear combinations of equations would rapidly generate linearly independent equations but this would depend on the total number of equations which is quadratic in $r_j + t_j$. Another hidden cost is that it requires $\mathcal{O}((r_j + t_j)^2)$ space.

In our experiments we have observed that this algorithm works well for very sparse polynomials but relatively poorly otherwise. According to our experiments, although our optimized version of KSHL is significantly faster than described in [Kal85], it is still slower than Wang's algorithm for most of the cases (see Chapter 6.6).

Suppose that the degree of the factors are d_{1j} and d_{2j}. In practice one obtains the factor which has the smallest degree first, say in this case $d_{1j} \leq d_{2j}$, and gets the other factor by trial division. On the other hand, for the non-zero ideal point α_j, since even a sparse polynomial turns out to be dense in $\langle x_j - \alpha_j \rangle$-adic expansion, one expects to solve d_{1j} MDP's. If we proceed in the way described in [Kal85], we need to set up a system of equations d_{1j} times and hence need to find a linearly independent system d_{1j} many times. However in our organization we need to set up system of equations only once as can be seen from Algorithm 9. Hence the cost of Gauss elimination decreases from $\mathcal{O}(d_{1j}(r_j + t_j)^3)$ to $\mathcal{O}((r_j + t_j)^3)$.

It should be mentioned that for very sparse problems the system of homogeneous equations contains many equations of the form cu_k for some constant c. In our implementation we took advantage of this observation to reach the full rank as fast as possible. We use these equations first, substitute them and then try to find a set of linearly independent equations for the remaining. We have implemented this version too. The drawback is that in this case at each step in the for loop one should construct another matrix and a vector since the new equations are no longer homogeneous. We have observed that it does not bring any advantage and it is even worse for practical problems. For example when we are trying to factor the determinants of Toeplitz matrices or Cyclic matrices we have observed that there is no simple homogeneous equation of the form cu_k.

4.4 Sparse Hensel Lifting by Monagan & Tuncer (MTSHL)

In this section we state one of the main ideas of MTSHL and give the pseudo-code for it. Before we go into the details we explain the idea of MTSHL by an example. Let $f_j \in \mathbb{Z}_p[x_1, \ldots, x_j]$ be a factor to be computed at the j^{th} step of MHL and $\alpha_j \in \mathbb{Z}_p$ be the randomly chosen point by the algorithm (in the beginning). Say $j = 3, p = 11$, $\alpha_3 = 2$ and

$$f_3 = x_1^5 + 4\,x_1{}^2x_3{}^3 - 2\,x_1x_2{}^3x_3 + 4\,x_2x_3{}^4 - \,x_1x_2x_3{}^2 - \,x_1x_2$$

Now consider the Taylor expansion of $f_3 = \sum_{i=0}^{4} f_{3,i}(x_3-2)^i$ in $\mathbb{Z}_{11}[x_1,x_2,x_3]$.

$$f_3 = (x_1^5 - 4\,x_1x_2{}^3 - x_1{}^2 - 5\,x_1x_2 - 2\,x_2) + \left(-2\,x_1x_2{}^3 + 4\,x_1{}^2 - 4\,x_1x_2 - 4\,x_2\right)(x_3-2) + \left(2\,x_1{}^2 - x_1x_2 - 3\,x_2\right)(x_3-2)^2 + \left(4\,x_1{}^2 - x_2\right)(x_3-2)^3 + 4\,x_2\,(x_3-2)^4$$

We see that

$$\begin{aligned} \mathrm{Supp}(f_{3,4}) &= \{x_2\} \\ \subseteq \mathrm{Supp}(f_{3,3}) &= \{x_1{}^2, x_2\} \\ \subseteq \mathrm{Supp}(f_{3,2}) &= \{x_1{}^2, x_1x_2, x_2\} \\ \subseteq \mathrm{Supp}(f_{3,1}) &= \{x_1x_2{}^3, x_1{}^2, x_1x_2, x_2\} \\ \subseteq \mathrm{Supp}(f_{3,0}) &= \{x_1^5, x_1x_2{}^3, x_1{}^2, x_1x_2, x_2\} \end{aligned}$$

If f_3 is a factor that we want to compute via MHL at the 3^{rd} step, the SHL idea explained in the previous section makes an assumption that $\mathrm{Supp}(f_{3,0}) = \mathrm{Supp}(f_3(x_3=2))$ with high probability. This is the

$$\mathrm{Supp}(f_{j,0}) = \mathrm{Supp}(f_j(x_j=\alpha_j)) \text{ (}\textbf{SHL assumption}\text{)}$$

The example above shows that more is true. As the next Theorem shows, if p is big enough and α_j chosen at random from $[0,p-1]$, in fact at the j^{th} step of MHL, with high probability

MTSHL assumption :

$$\mathrm{Supp}(f_{j,0}) = \mathrm{Supp}(f_j(x_j=\alpha_j)) \text{ and } \mathrm{Supp}(f_{j,i+1}) \subseteq \mathrm{Supp}(f_{j,i}) \text{ for } 0 \le i \le \deg_{x_j} f_j.$$

What does this buy us? Note that at the j^{th} step of MHL, in the for loop one first solves an MDP to get $f_{j,1}$ then $f_{j,2}$ and so on. So if we use the sparse interpolation approach to solve an MDP as MTSHL does, while we are solving the MDP to compute $f_{j,i+1}$ we can use $f_{j,i}$ as a skeleton of the solution. That is, it eliminates the computational cost of recursive steps of Zippel's sparse interpolation. Also, since the size of the supports get smaller as i increases, this significantly decreases the evaluation cost of MTSHL. Because, during MTSHL, one solves MDP's of the form $\sigma u + \tau w = c$ and when the solutions σ, τ to be computed (which also means the actual factors to be computed) are huge, the expected size of c is huge. (See Chapter 6). It also eliminates the sparse interpolation and reduces solving an MDP to solving a single (Vandermonde) linear system. What does this cost us? It increases the failure probability of the algorithm and this must be investigated.

Theorem 28. *Let $f \in \mathbb{Z}_p[x_1,\ldots,x_n]$ and by $\mathrm{Supp}(f)$ we denote the set of monomials present in f. Let α chosen at random in $\mathbb{Z}_p$ and $f = \sum_{i=0}^{d_n} b_i(x_1,\ldots,x_{n-1})(x_n-\alpha)^i$ be the*

$(x_n - \alpha)$-*adic expansion of* f, *where* $d_n = \deg_{x_n} f$. *Then for a given* j *with* $0 \leq j < d_n$,

$$\Pr[\mathrm{Supp}(b_{j+1}) \nsubseteq \mathrm{Supp}(b_j)] \leq |\mathrm{Supp}(b_{j+1})| \frac{d_n - j}{p - d_n + j + 1}.$$

Proof. For simplicity assume that $p > j$, otherwise we will need to introduce Hasse derivatives but the idea will be the same. We have

$$b_j(x_1, \ldots, x_{n-1}) = \frac{1}{j!} \frac{\partial}{\partial x_n^j} f(x_1, \ldots, x_{n-1}, x_n = \alpha).$$

If we write $f \in \mathbb{Z}_p[x_n][x_1, \ldots, x_{n-1}]$ as

$$f = c_1(x_n)M_1 + c_2(x_n)M_2 + \cdots + c_k(x_n)M_k$$

where M_k's are the distinct monomials in $x_1, \ldots, x_{n-1}$ and we denote $\frac{\partial}{\partial x_n^j} c_i(x_n) = c_i^{(j)}(x_n)$ then

$$\begin{aligned} b_j = \frac{\partial}{\partial x_n^j} f(x_n = \alpha) &= c_1^{(j)}(\alpha)M_1 + c_2^{(j)}(\alpha)M_2 + \cdots + c_k^{(j)}(\alpha)M_k. \\ b_{j+1} = \frac{\partial}{\partial x_n^{j+1}} f(x_n = \alpha) &= c_1^{(j+1)}(\alpha)M_1 + c_2^{(j+1)}(\alpha)M_2 + \cdots + c_k^{(j+1)}(\alpha)M_k. \end{aligned}$$

For a given $j > 0$, if $c_i^{(j+1)}(\alpha) \neq 0$, but $c_i^{(j)}(\alpha) = 0$ then $M_i \notin \mathrm{Supp}(b_j)$. We need to compute $\Pr[c_i^{(j)}(\alpha) = 0 \,|\, c_i^{(j+1)}(\alpha) \neq 0]$. If A is the event that $c_i^{(j)}(\alpha) = 0$ and B is the event that $c_i^{(j+1)}(\alpha) = 0$ then

$$\Pr[A \,|\, B^c] = \frac{\Pr[A] - \Pr[B]\Pr[A \,|\, B]}{\Pr[B^c]} \leq \frac{\Pr[A]}{\Pr[B^c]}.$$

By the Schwartz-Zippel Lemma

$$\frac{\Pr[A]}{\Pr[B^c]} \leq \frac{\deg_{x_n}(c_i^{(j)}(y))/p}{1 - (\deg_{x_n}(c_i^{(j+1)}(y))/p)} \leq \frac{(d_n - j)/p}{1 - (d_n - j - 1)/p} = \frac{d_n - j}{p - d_n + j + 1}$$

□

Theorem 28 shows that for the sparse case, if p is big enough and α is chosen at random then the probability of $\mathrm{Supp}(b_{j+1}) \subseteq \mathrm{Supp}(b_j)$ is high.

4.4.1 Bad evaluation points for MTSHL

Following the notation of the Theorem 28 above, let $f \in \mathbb{Z}_p[x_1, \ldots, x_n]$, α is chosen at random in $\mathbb{Z}_p$ and $f = \sum_{i=0}^{d_n} b_i(x_1, \ldots, x_{n-1})(x_n - \alpha)^i$ be the $(x_n - \alpha)$-adic expansion of f, where $d_n = \deg_{x_n} f$. For a given $\alpha \in \mathbb{Z}_p$, let us call α a *badpoint*, if $\mathrm{Supp}(b_{j+1}) \nsubseteq \mathrm{Supp}(b_j)$

for some $0 \leq j < d_n$. As in the proof of Theorem 28, consider

$$b_j = \frac{\partial}{\partial x_n^j} f(x_n = \alpha) = c_1^{(j)}(\alpha)M_1 + c_2^{(j)}(\alpha)M_2 + \cdots + c_k^{(j)}(\alpha)M_k.$$

So, for a given f, if $c_i^{(j)}$ has a root but does not have a double root at $x_n = \alpha$, then α is badpoint for b_{j+1}, i.e. $\mathrm{Supp}(b_{j+1}) \nsubseteq \mathrm{Supp}(b_j)$: Consider the following example where $\mathrm{Supp}(b_{j+1}) \nsubseteq \mathrm{Supp}(b_j)$ for $j = 0, 2$. Let

$$f := (x_1^6 + x_1^5 + x_1^4)(x_2 - 1)^3 + (x_1^5 + x_1^4 + x_1^3)(x_2 - 1) + (x_1^7 + 1) \in \mathbb{Z}_{509}[x_1, x_2].$$

But if we choose another point say $\alpha_2 = 301$ and compute the $(x_2 - 301)$-adic expansion of $f = \sum_{i=0}^{3} b_i(x_1)(x_2 - 301)^i$ we have

$$\begin{aligned} b_0 &= x_1^7 + 95x_1^6 + 395x_1^5 + 395x_1^4 + 300x_1^3 + 1 \\ b_1 &= 230x_1^6 + 231x_1^5 + 231x_1^4 + x_1^3 \\ b_2 &= 391x_1^6 + 391x_1^5 + 391x_1^4 \\ b_3 &= x_1^6 + x_1^5 + x_1^4 \end{aligned}$$

and we see that $\mathrm{Supp}(b_{j+1}) \subseteq \mathrm{Supp}(b_j)$ for $0 \leq j \leq 2$. In fact for this example $\alpha = 1, 209, -207$ are the only badpoints as can be seen by considering $f \in \mathbb{Z}_{509}[x_2][x_1]$:

$$f = x_1^7 + (x_2 - 1)^3 x_1^6 + (x_2 - 209)(x_2 - 1)(x_2 + 207) + x_1^5(x_2 - 209)(x_2 - 1)x_1^4 + (x_2 - 1)x_1^3 + 1$$

Note that these points are badpoints only for b_2. Before we give an upper bound for the number of badpoint points we consider the following example. Let $p = 1021$,

$$\begin{aligned} f &= (x_2 - 841)(x_2 - 414)(x_2 - 15)(x_2 - 277)x_1^9 \\ &+ (x_2 - 339)(x_2 - 761)(x_2 - 752)(x_2 - 345)x_1^7 \end{aligned}$$

and $f^{(i)} = \frac{\partial}{\partial x_2^i} f(x_1, x_2)$. Then

$$\begin{aligned} f^{(1)} &= 4(x_2 - 384)(x_2 - 230)(x_2 - 291)x_1^9 + 4(x_2 - 441)(x_2 + 127)(x_2 + 453)x_1^7 \\ f^{(2)} &= 12(x_2 - 89)(x_2 - 174)x_1^9 + 12(x_2 - 473)(x_2 - 115)x_1^7 \\ f^{(3)} &= (24x_2 - 93)x_1^9 + (24x_2 + 91)x_1^7 \text{ and} \\ f^{(4)} &= 24x_1^9 + 24x_1^7. \end{aligned}$$

So, the maximum number of badpoints occurs if each $c_i^{(j)}$ splits for different points, hence $|\mathrm{Supp}(f)|\frac{d_n(d_n+1)}{p}$ is an upper bound for the probability of hitting a badpoint.

Algorithm 10 j^{th} step of MTSHL

Input : $a_j \in \mathbb{Z}_p[x_1, \ldots, x_j]$, $f_{j-1}, g_{j-1} \in \mathbb{Z}_p[x_1, \ldots, x_{j-1}]$ and $\alpha_j \in \mathbb{Z}_p$ where a_j, f_{j-1}, g_{j-1} are monic in x_1. Also, $a_j(x_1, \ldots, x_{j-1}, x_j = \alpha_j) = f_{j-1}g_{j-1}$.

Output : $f_j, g_j \in \mathbb{Z}_p[x_1, \ldots, x_j]$ such that $a_j = f_j g_j$ or FAIL (No such factorization exists.)

1: **if** $r_j > t_j$ **then** interchange f_{j-1} with g_{j-1} **end if**
2: $(\sigma_{j0}, \tau_{j0}) \leftarrow (f_{j-1}, g_{j-1})$.
3: $(\sigma_j, \tau_j) \leftarrow (\sigma_{j0}, \tau_{j0})$.
4: $monomial \leftarrow 1$.
5: $error \leftarrow a_j - f_{j-1}\, g_{j-1}$
6: **for** i from 1 to $\deg(a_j, x_j)$ **while** $error \neq 0$ **do**
7: $\quad monomial \leftarrow monomial \times (x_j - \alpha_j)$
8: $\quad c \leftarrow$ coefficient of $(x_j - \alpha_j)^i$ in the Taylor expansion of the $error$ about $x_j = \alpha_j$
9: $\quad$ **if** $c \neq 0$ **then**
10: $\quad\quad \sigma_g \leftarrow$ skeleton of $\tau_{j,i-1}$
11: $\quad\quad (\sigma_{ji}, \tau_{ji}) \leftarrow$**SparseInt**$(\sigma_{j0}, \tau_{j0}, c, \sigma_g, p)$ (Algorithm 7)
12: $\quad\quad$ **if** (σ_{ji}, τ_{ji})=FAIL **then**
13: $\quad\quad\quad (\sigma_{ji}, \tau_{ji}) \leftarrow$ **BSDiophant**$(\sigma_{j0}, \tau_{j0}, c, p)$ (Algorithm 8)
14: $\quad\quad\quad$ **if** (σ_{ji}, τ_{ji})=FAIL **then** restart the factorization with a different ideal
15: $\quad\quad\quad$ **end if**
16: $\quad\quad$ **end if**
17: $\quad\quad (\sigma_j, \tau_j) \leftarrow (\sigma_j + \sigma_{ji} \times monomial,\ \tau_j + \tau_{ji} \times monomial)$.
18: $\quad\quad error \leftarrow a_j - \sigma_j \tau_j$.
19: $\quad$ **end if**
20: **end for**
21: **if** $error \neq 0$ **then return** FAIL (No such factorization exists)
22: **else return** (σ_j, τ_j)
23: **end if**

For a sparse polynomial with 1000 terms, $d_n = 20$, for $p = 2^{31} - 1$, this probability is 0.000097. This observation also suggests that while solving the diophantine equation 4.15 at the i^{th} iteration of the j^{th} step of MHL, we use $\sigma_{i,j-1}$ (or $\tau_{i,j-1}$) as a form of the solution of σ_{ji} (or τ_{ij}).

Back to our discussion on SHL, based on the observation above the j^{th} step ($j > 1$) of MTSHL is summarized in Algorithm 10.

Example 29. Suppose we seek to factor $a = fg$ where

$$\begin{aligned} f &= {x_1}^8 + 2\,x_1{x_2}^2{x_4}^3x_5 + 4\,x_1{x_2}^2{x_3}^3 + 3\,x_1{x_2}^2x_4{x_5}^2 + {x_2}^2x_3x_4 - 5 \\ g &= {x_1}^8 + 3\,{x_1}^2x_2x_3{x_4}^2x_5 + 5\,{x_1}^2x_2{x_3}^2x_4 - 3\,{x_4}^2{x_5}^2 + 4\,x_5 \end{aligned}$$

Let $\alpha_3 = 1, p = 2^{31} - 1$. Before lifting we have a and

$$\begin{aligned} f^{(0)} &:= f(x_5 = 1) = {x_1}^8 + 4\,x_1{x_2}^2{x_3}^3 + 2\,x_1{x_2}^2{x_4}^3 + 3\,x_1{x_2}^2x_4 + {x_2}^2x_3x_4 - 5 \\ g^{(0)} &:= g(x_5 = 1) = {x_1}^8 + 5\,{x_1}^2x_2{x_3}^2x_4 + 3\,{x_1}^2x_2x_3{x_4}^2 - 3\,{x_4}^2 + 4 \end{aligned}$$

If the assumption of SHL is true with $x_5 = \alpha = 1$ then at the first step we assume that $f = \sum_{i=0}^{\deg_{x_5} f} f_i (x_5 - 1)^i$ and $g = \sum_{i=0}^{\deg_{x_5} g} g_i (x_5 - 1)^i$ where f_1 and g_1 are in the form

$$\begin{aligned} f_1 &= \left(c_1 {x_3}^3 + c_2 {x_4}^3 + c_3 x_4\right) x_1 {x_2}^2 + c_4 {x_2}^2 x_3 x_4 + c_5 \\ g_1 &= \left(c_6 {x_3}^2 x_4 + c_7 x_3 {x_4}^2\right) {x_1}^2 x_2 + c_8 {x_4}^2 + c_9 \end{aligned}$$

for some unknowns $C = \{c_1, \ldots, c_9\}$.

In the following $e_5^{(k)}$ denotes the coefficient of $(x_5 - 1)^k$ in the Taylor expansion of the *error* about $x_5 = 1$. Let also $f_0 := f^{(0)}, g_0 := g^{(0)}, f^{(k)} := \sum_{i=0}^{k} f_i (x_5 - 1)^i, g^{(k)} := \sum_{i=0}^{k} g_i (x_5 - 1)^i$.

We start by computing the first $error = a - f^{(0)} g^{(0)}$. We obtain

$$\begin{aligned} e_5^{(1)} = {} & 3\,{x_1}^{10} x_2 x_3 {x_4}^2 + 2\,{x_1}^9 {x_2}^2 {x_4}^3 + 6\,{x_1}^9 {x_2}^2 x_4 + 12\,{x_1}^3 {x_2}^3 x_3 {x_4}^4 {x_4}^2 + 10\,{x_1}^3 {x_2}^3 {x_3}^2 {x_4}^4 \\ & + 12\,{x_1}^3 {x_2}^3 x_3 {x_4}^5 - 6\,{x_1}^8 {x_4}^2 + 30\,{x_1}^3 {x_2}^3 {x_3}^2 {x_4}^2 + 27\,{x_1}^3 {x_2}^3 x_3 {x_4}^3 + 3\,{x_1}^2 {x_2}^3 {x_3}^2 {x_4}^3 \\ & + 4\,{x_1}^8 - 24\,x_1 {x_2}^2 {x_3}^3 {x_4}^2 - 18\,x_1 {x_2}^2 {x_4}^5 - 15\,{x_1}^2 x_2 x_3 {x_4}^2 + 16\,x_1 {x_2}^2 {x_3}^3 \\ & - 20\,x_1 {x_2}^2 {x_4}^3 - 6\,{x_2}^2 x_3 {x_4}^3 + 36\,x_1 {x_2}^2 x_4 + 4\,{x_2}^2 x_3 x_4 + 30\,{x_4}^2 - 20 \end{aligned}$$

The MDP to be solved is:

$$\begin{aligned} D: \; & f_0 \left(\left(c_6 {x_3}^2 x_4 + c_7 x_3 {x_4}^2\right) {x_1}^2 x_2 + c_8 {x_4}^2 + c_9\right) \\ & + g_0 \left(\left(c_1 {x_3}^3 + c_2 {x_4}^3 + c_3 x_4\right) x_1 {x_2}^2 + c_4 {x_2}^2 x_3 x_4 + c_5\right) = e_5^{(1)}. \end{aligned}$$

Our aim is first to get $\left(\left(c_6 {x_3}^2 x_4 + c_7 x_3 {x_4}^2\right) {x_1}^2 x_2 + c_8 {x_4}^2 + c_9\right)$, since it will create a smaller matrix. We need 2 evaluations only: We choose 2 evaluation points: $[x_3 = 2, x_4 = 3]$ and $[x_3 = 2^2, x_4 = 3^2]$ and compute

$$\begin{aligned} D([x_3 = 2, x_4 = 3]): \; & \left({x_1}^8 + 95\,x_1 {x_2}^2 + 6\,{x_2}^2 - 5\right) \left(\left(12\,c_6 + 18\,c_7\right) {x_1}^2 x_2 + 9\,c_8 + c_9\right) \\ & + \left({x_1}^8 + 114\,{x_1}^2 x_2 - 23\right) \left(\left(8\,c_1 + 27\,c_2 + 3\,c_3\right) x_1 {x_2}^2 + 6\,c_4 {x_2}^2 + c_5\right) \\ = {} & 54\,{x_1}^{10} x_2 + 72\,{x_1}^9 {x_2}^2 - 50\,{x_1}^8 + 13338\,{x_1}^3 {x_2}^3 + 324\,{x_1}^2 {x_2}^3 - 270\,{x_1}^2 x_2 - 6406\,x_1 {x_2}^2 - 300\,{x_2}^2 + 250 \end{aligned}$$

$$\begin{aligned} D([x_3 = 4, x_4 = 9]): \; & \left({x_1}^8 + 1741\,x_1 {x_2}^2 + 36\,{x_2}^2 - 5\right) \left(\left(144\,c_6 + 324\,c_7\right) {x_1}^2 x_2 + 81\,c_8 + c_9\right) \\ & + \left({x_1}^8 + 1692\,{x_1}^2 x_2 - 239\right) \left(\left(64\,c_1 + 729\,c_2 + 9\,c_3\right) x_1 {x_2}^2 + 36\,c_4 {x_2}^2 + c_5\right) \\ = {} & 972\,{x_1}^{10} x_2 + 1512\,{x_1}^9 {x_2}^2 - 482\,{x_1}^8 + 4250556\,{x_1}^3 {x_2}^3 \\ & + 34992\,{x_1}^2 {x_2}^3 - 4860\,{x_1}^2 x_2 - 1200530\,x_1 {x_2}^2 - 17352\,{x_2}^2 + 2410 \end{aligned}$$

Now calling BDP we see that the solutions to these bivariate diophantine equations are respectively

$$\begin{aligned}[\sigma_1, \tau_1] &= [54\,{x_1}^2 x_2 - 50, 72\,x_1 {x_2}^2] \\ [\sigma_2, \tau_2] &= [972\,{x_1}^2 x_2 - 482, 1512\,x_1 {x_2}^2].\end{aligned}$$

Hence we have

$$\begin{aligned}(12\,c_6 + 18\,c_7)\,{x_1}^2 x_2 + 9\,c_8 + c_9 &= 54\,{x_1}^2 x_2 - 50 \\ (144\,c_6 + 324\,c_7)\,{x_1}^2 x_2 + 81\,c_8 + c_9 &= 972\,{x_1}^2 x_2 - 482\end{aligned}$$

Then we form the (Vandermonde) linear systems

$$\begin{bmatrix} 12 & 18 \\ 144 & 324 \end{bmatrix} \begin{bmatrix} c_6 \\ c_7 \end{bmatrix} = \begin{bmatrix} 54 \\ 972 \end{bmatrix} \text{ and } \begin{bmatrix} 9 & 1 \\ 81 & 1 \end{bmatrix} \begin{bmatrix} c_8 \\ c_9 \end{bmatrix} = \begin{bmatrix} -50 \\ -482 \end{bmatrix}$$

and we obtain $c_6 = 0, c_7 = 3, c_8 = -6, c_9 = 4$. So $g_1 = \left(3\,{x_1}^2 x_2 x_3 {x_4}^2 - 6\,{x_4}^2 + 4\right)$. Then by division we get $f_1 = \dfrac{e_5^{(1)} - f_0 g_1}{g_0} = \left(2\,x_1 x_2 {x_4}^3 + 8\,x_2 {x_3}^4\right)$. Hence

$$\begin{aligned}f^{(1)} &= f_0 + \left(2\,x_1 {x_2}^2 {x_4}^3 + 6\,x_1 {x_2}^2 x_4\right)(x_5 - 1) \\ g^{(1)} &= g_0 + \left(3\,{x_1}^2 x_2 x_3 {x_4}^2 - 6\,{x_4}^2 + 4\right)(x_5 - 1)\,.\end{aligned}$$

Note that we use the division step above also as a check for the correctness of the SHL assumption. Since the solution to the MDP is unique, we would have $g_0 \nmid e_5^{(1)} - f_0 g_1$, if the assumption was wrong.

Now following Theorem 28 by looking at the monomials of f_1 and g_1, we assume that the form of the f_2 and g_2 are

$$\begin{aligned}f_2 &= c_1 x_1 {x_2}^2 {x_4}^3 + c_2 x_1 {x_2}^2 x_4 + c_3 \\ g_2 &= c_4 {x_1}^2 x_2 x_3 {x_4}^2 + c_5 {x_4}^2 + c_6\end{aligned}$$

for some unknowns $C = \{c_1, \ldots, c_6\}$. The next error is $error = a - f^{(1)} g^{(1)}$. Then

$$\begin{aligned}e_5^{(2)} = {} & 3\,{x_1}^9 {x_2}^2 x_4 - 3\,{x_1}^8 {x_4}^2 + 15\,{x_1}^3 {x_2}^3 {x_3}^2 {x_4}^2 + 9\,{x_1}^3 {x_2}^3 x_3 {x_4}^3 \\ & - 12\,x_1 {x_2}^2 {x_3}^3 {x_4}^2 - 6\,x_1 {x_2}^2 {x_4}^5 - 18\,x_1 {x_2}^2 {x_4}^3 - 3\,{x_2}^2 x_3 {x_4}^3 + 12\,x_1 {x_2}^2 x_4 + 15\,{x_4}^2\end{aligned}$$

The MDP to be solved is:

$$D:\ f_0 \left(c_4 {x_1}^2 x_2 x_3 {x_4}^2 + c_5 {x_4}^2 + c_6\right) + g_0 \left(c_1 x_1 {x_2}^2 {x_4}^3 + c_2 x_1 {x_2}^2 x_4 + c_3\right) = e_5^{(2)}.$$

We need 2 evaluations again: Choose $[x_3 = 5, x_4 = 6]$ and $[x_3 = 5^2, x_4 = 6^2]$ and compute

$$\begin{aligned} D(([x_3 = 5, x_4 = 6]) := & \left({x_1}^8 + 950\, x_1 {x_2}^2 + 30\, {x_2}^2 - 5\right)\left(180\, c_4 {x_1}^2 x_2 + 36\, c_5 + c_6\right) \\ & + \left({x_1}^8 + 1290\, {x_1}^2 x_2 - 104\right)\left(216\, c_1 x_1 {x_2}^2 + 6\, c_2 x_1 {x_2}^2 + c_3\right) \\ & = 18\, {x_1}^9 {x_2}^2 - 108\, {x_1}^8 + 23220\, {x_1}^3 {x_2}^3 - 104472\, x_1 {x_2}^2 - 3240\, {x_2}^2 + 540 \end{aligned}$$

$$\begin{aligned} D(([x_3 = 25, x_4 = 36]) := & \left({x_1}^8 + 155920\, x_1 {x_2}^2 + 900\, {x_2}^2 - 5\right)\left(32400\, c_4 {x_1}^2 x_2 + 1296\, c_5 + c_6\right) \\ & + \left({x_1}^8 + 209700\, {x_1}^2 x_2 - 3884\right)\left(46656\, c_1 x_1 {x_2}^2 + 36\, c_2 x_1 {x_2}^2 + c_3\right) \\ & = 108\, {x_1}^9 {x_2}^2 - 3888\, {x_1}^8 + 22647600\, {x_1}^3 {x_2}^3 - 606636432\, x_1 {x_2}^2 - 3499200\, {x_2}^2 + 19440 \end{aligned}$$

Now calling BDP we see that the solutions to these bivariate diophantine equations are respectively

$$\begin{aligned} [\sigma_1, \tau_1] &= [-108, 18\, x_1 {x_2}^2] \\ [\sigma_2, \tau_2] &= [-3888, 108\, x_1 {x_2}^2] \end{aligned}$$

Hence we have

$$\begin{aligned} 180\, c_4 {x_1}^2 x_2 + 36\, c_5 + c_6 &= -108 \\ 32400\, c_4 {x_1}^2 x_2 + 1296\, c_5 + c_6 &= -3888 \end{aligned}$$

Then we solve the Vandermonde linear systems

$$[180]\,[c_4] = [0] \text{ and } \begin{bmatrix} 36 & 1 \\ 1296 & 1 \end{bmatrix} \begin{bmatrix} c_5 \\ c_6 \end{bmatrix} = \begin{bmatrix} -108 \\ -3888 \end{bmatrix}$$

and obtain $c_4 = 0, c_5 = -3, c_6 = 0$. So $g_2 = -3\, {x_4}^2$. Then by division we get $f_2 = \dfrac{e_5^{(2)} - f_0 g_2}{g_0} = 3\, x_1 {x_2}^2 x_4\, (x_5 - 1)^2$. Hence

$$\begin{aligned} f^{(2)} &= f^{(1)} + \; 3\, x_1 {x_2}^2 x_4\, (x_5 - 1)^2 \\ &= \; {x_1}^8 + 2\, x_1 {x_2}^2 {x_4}^3 x_5 + 4\, x_1 {x_2}^2 {x_3}^3 + 3\, x_1 {x_2}^2 x_4 {x_5}^2 + {x_2}^2 x_3 x_4 - 5 \\ g^{(2)} &= g^{(1)} + \; \left(-3\, {x_4}^2\right)(x_5 - 1)^2 \\ &= \; {x_1}^8 + 3\, {x_1}^2 x_2 x_3 {x_4}^2 x_5 + 5\, {x_1}^2 x_2 {x_3}^2 x_4 - 3\, {x_4}^2 {x_5}^2 + 4\, x_5. \end{aligned}$$

The next error is $error = a - f^{(2)} g^{(2)} = 0$ and we have the factors!

We have used 4 evaluations and solved three $(2 \times 2)-$ system and one (1×1) system. For the same problem KSHL would need to find 9 linearly independent homogeneous linear equations out of 28 equations first. A natural question is, is it possible for KHL to focus on to some subset of the variables first? The answer is no. The system of equations that are constructed by KSHL are coupled.

4.4.2 Some remarks on MTSHL

We give a detailed complexity analysis of MTSHL in Chapter 6. Before doing so, several remarks on Algorithm 10 are in order:

Step 8 in the for loop computes the i^{th} Taylor coefficient of the error at $x_j = \alpha_j$. Maple used to compute this using the formula $c = g(x_j = \alpha_j)/i!$ where g is the i'th derivative of *error* wrt x_j. Instead, Maple now uses the more direct formula

$$c = \sum_{k=i}^{d} \text{coeff}(error, x^k)\alpha^{s-k}\binom{s}{k}$$

where $d = \deg_{x_j} error$ which is three times faster [MP14].

At step 10 MTSHL makes the assumption $\text{Supp}(\tau_{ji}) \subseteq \text{Supp}(\tau_{j,i-1})$ based on Theorem 28. Note that, if the minimum of the number of terms of each factor of a_j is $t = \min(\#f_j, \#g_j)$, then at step 11 the probability of failure of the assumption is $\leq t\frac{d_j - i}{p - d_j - (i-1)} \leq \frac{td_j}{p-2d_j}$ and cost of solving each MDP is the evaluation cost + cost of a system of linear equation solving which is bounded above by $\mathcal{O}(t^2)$.

Another costly operation is the cost of the multivariate division, $\sigma_{ji} = (c - \sigma_{j0}\tau_{ji})/\tau_{j0}$, which is hidden in sparse interpolation. If the algorithm fails to compute (σ_{ji}, τ_{ji}) at step 11 then it passes to a safer way at step 13.

Another expensive operation in the Algorithm 10 is the error computation, $error \leftarrow a_j - \sigma_j\tau_j$, in the for loop. To decrease this cost, one of the ideas in [MY74] can be generalized to MHL. Let $\sigma_j = \sum_{s=0}^{\deg_{x_j}\sigma_j} \sigma_{j,s}(x_j - \alpha_j)^s$ and $\sigma_j^{(i)} = \sum_{s=0}^{i} \sigma_{j,s}(x_j - \alpha_j)^s$ (similarly for τ). One has

$$\begin{aligned} e_j^{(i+1)} &= a_j - \sigma_j^{(i)}\tau_j^{(i)} = a_j - (\sigma_j^{(i-1)} + \sigma_{j,i}(x_j - \alpha_j)^i)(\tau_j^{(i-1)} + \tau_{j,i}(x_j - \alpha_j)^i) \\ &= a_j - \sigma_j^{(i-1)}\tau_j^{(i-1)} - (\sigma_j^{(i-1)}\tau_{j,i} + \tau_j^{(i-1)}\sigma_{j,i})(x_j - \alpha_j)^i \\ &= e_j^{(i)} - U^{(i)}(x_j - \alpha_j)^i \end{aligned}$$

where $U^{(i)} := (\sigma_j^{(i-1)}\tau_{j,i} + \tau_j^{(i-1)}\sigma_{j,i})$. Hence in the for loop we have the relation $e_j^{(i+1)} = e_j^{(i)} - (x_j - \alpha_j)^i U^{(i)}$ for a correction term $U^{(i)} \in \mathcal{O}((x_j - \alpha_j)^{i-1})$. Also for $i \geqslant 0$ it is known that $(x_j - \alpha_j)^i$ divides $e_j^{(i)}$. So if we define $c_j^{(i)} := e_j^{(i)}/(x_j - \alpha_j)^i$ then $c_j^{(i)}$ can be computed efficiently using

$$c_j^{(i+1)} = (c_j^{(i)} - U^{(i)})/(x_j - \alpha_j).$$

Hence we may compute $c_j^{(i)}$ for $i = 1, 2, \ldots$ until it becomes zero instead of computing $e_j^{(i)}$. According to our experiments, this observation decreases the cost when the number of factors is 2. For more than 2 factors, the generalization of it does not bring a significant advantage. So, in our implementations we only use this update formula when the number of factors is 2.

Also note that, MTSHL makes use of only one of the SHL assumptions because it aims to get one of the factors. Also it eliminates the recursive step in MHL to compute the skeleton of the solution.

4.5 Reconsidering the case modulo $\mathbf{p}^l$ with $l > 1$.

In the previous sections we have seen that the lifting process has two main stages. The first step is to find a prime p and a natural number $l > 0$ such that the ring $\mathbb{Z}_{p^l}$ can be identified with the ring $\mathbb{Z}$. That is, we find a bound B such that the integer coefficients of the polynomial a to be factored and its factors to be computed are bounded by B. One way to choose such an upper bound B is given by Lemma 14. Then we choose l such that $p^l > 2B$. Next the solution in $\mathbb{Z}_p[x_1]$ is lifted to the solution in $\mathbb{Z}_{p^l}[x_1]$. The second step is to lift the solution from $\mathbb{Z}_{p^l}[x_1]$ to $\mathbb{Z}_{p^l}[x_1, \ldots, x_n]$. Note that in the second stage all arithmetic is in $\mathbb{Z}_{p^l}$ with $p^l > 2B$.

This strategy is described in detail in [GCL92] and it is currently implemented by most of the computer algebra platforms including Maple, Singular [Lee13] and Magma [Steel]. In this section we will question whether the strategy described above is the best way to manage the task for the case $l > 1$.

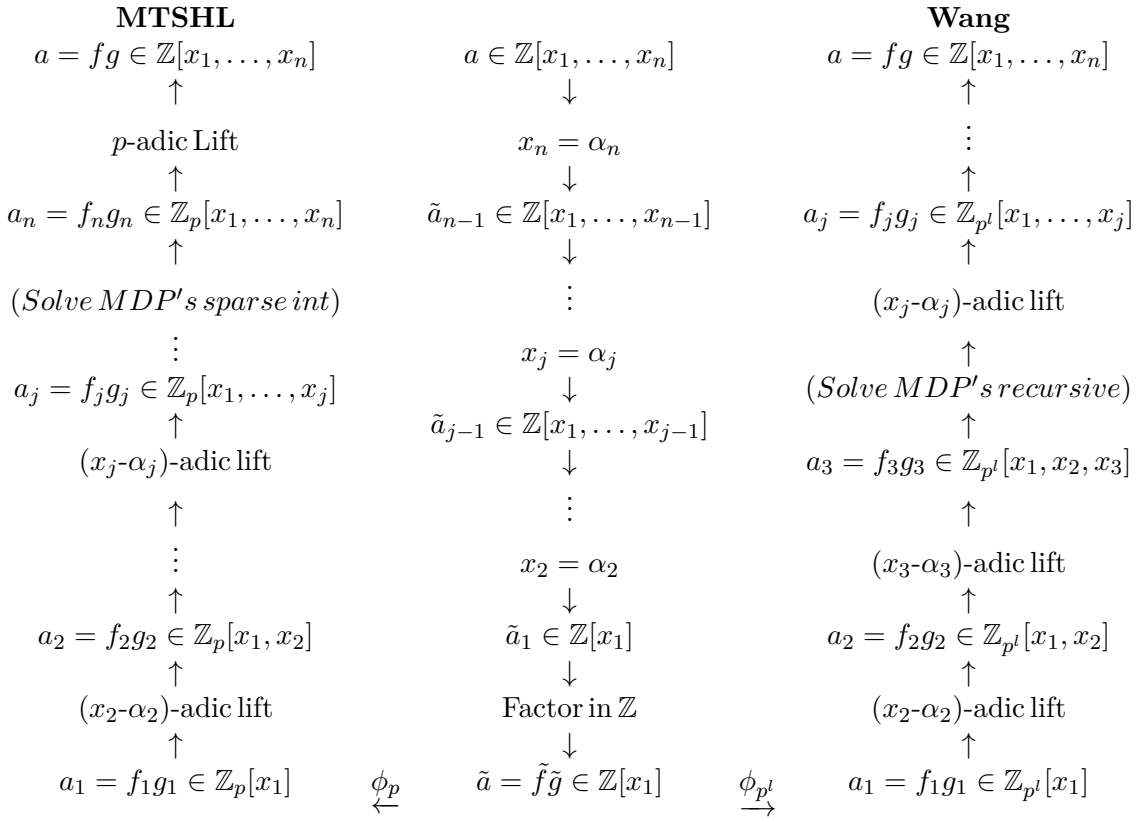
How should we choose p and how big should it be? As indicated previously by Musser, if we choose $p > 2B$ then p is large enough to eliminate entirely the phase of construction which lifts from p to p^l but this might mean that p would be a multiple precision integer which will increase significantly the cost of all coefficient arithmetic. The best approach seems to be to choose p as big as possible while constrained to be a single precision integer [Mus71].

Suppose for example that the coefficients of the factors are bounded by p^{10}. Before the factorization we don't have this information. Since most likely the coefficient bound $B > p^{20}$, this means that all arithmetic throughout MHL is in $\mathbb{Z}_{p^{20}}$, which will be very expensive.

Our design of an efficient sparse multivariate diophantine solver as we described in Section 3.6 allows us to propose a more efficient approach that eliminates most of the multi precision arithmetic. (See Figure 4.1)

- First compute the lifting upper bound $l = \lceil \log_p B \rceil$ by using Lemma 14.

Figure 4.1: Wang's Algorithm vs MTSHL



- Then choose a random $(m+1)$-bit machine prime p, i.e. $p \in [2^m < p < 2^{m+1}]$ and compute the factorization of a by lifting the factorization in $\mathbb{Z}_p[x_1]$ to in $\mathbb{Z}_p[x_1, \ldots, x_n]$ with MTSHL so that most of the work uses a machine prime p.

- Then as a second stage lift the factorization from $\mathbb{Z}_p[x_1, \ldots, x_n]$ to $\mathbb{Z}_{p^l}[x_1, \ldots, x_n]$ via p-adic lifting using the sparse MDP solver.

The following observations and the data in Table 4.1, which compares the efficiency of the both strategies, makes it clear, why this strategy is better than and should be preferred to the previous one.

Suppose that $a = uw$ where $a, u, w \in \mathbb{Z}[x_1, \ldots, x_n]$ and u, w are unknown to us. As a first step we choose an evaluation ideal $I = \langle x_2 - \alpha_2, \ldots, x_n - \alpha_n \rangle$ with randomly chosen α_i from $[0, p-1]$ such that the conditions of the Theorem 16 are satisfied with $l = 1$. Then there is a factorization $a = u^{(n)} w^{(n)} \in \mathbb{Z}_p[x_1, \ldots, x_n]$. This factorization is computed using MTSHL.

Now suppose that u (similarly w) has the form

$$\begin{aligned} u &= c_0M_0 + c_1M_1 + \cdots + c_kM_k \\ &= (\sum_{i=0}^{l-1} s_{0i}p^i)M_0 + (\sum_{i=0}^{l-1} s_{1i}p^i)M_1 + \cdots + (\sum_{i=0}^{l-1} s_{ki}p^i)M_k \end{aligned}$$

where $M_i \in \mathbb{Z}_{p^l}[x_1, \ldots, x_n]$ are distinct monomials, $0 \neq c_j \in \mathbb{Z}$ with $c_j = \sum_{i=0}^{l-1} s_{ji}p^j$ where $-p^l/2 < s_{ji} < p^l/2$. Then we have

$$\begin{aligned} u &= (\sum_{j=0}^{k} s_{j0}M_j) + (\sum_{j=0}^{k} s_{j1}M_j)p + \cdots + (\sum_{j=0}^{k} s_{j1}M_j)p^{l-1} \\ &= u_0 + u_1p + u_2p^2 + \cdots + u_{l-1}p^{l-1} \end{aligned}$$

where $u_i = \sum_{j=0}^{k} s_{ji}M_j$. It follows that

$$\frac{u - \sum_{i=0}^{k-1} u_ip^i}{p^k} = (\sum_{i=k}^{l-1} s_{0i}p^{i-k})M_0 + (\sum_{i=k}^{l-1} s_{1i}p^{i-k})M_1 + \cdots + (\sum_{i=k}^{l-1} s_{ki}p^{i-k})M_k.$$

Also, we have $u_0 = u \mod p \neq 0$ since in the first stage u is lifted from u_0. Now we make a key observation:

If p is an m-bit prime, $p \in [2^m, 2^{m+1}]$ and if p is chosen at random, the probability that $p \mid c_i$ is $\Pr[p \mid c_i] = \frac{\#\text{distinct (m+1)bit prime divisors of } c_i}{\#\text{m bit primes}}$. Let $\pi(s)$ be the number of primes $\leq s$. Since there are at most $\lfloor \log_{2^m}(c_i) \rfloor$ many $(m+1)$-bit primes dividing c_i we have

$$\Pr[p \mid c_i] \leq \frac{\lfloor \log_{2^m}(c_i) \rfloor}{\pi(2^{m+1}) - \pi(2^m)} \leq \frac{l}{\pi(2^{m+1}) - \pi(2^m)}$$

This probability is very small because according to the prime number theorem $\pi(s) \sim s/\log(s)$ and hence $\pi(2^{m+1}) - \pi(2^m) \sim \frac{2^m}{m\log(2)}$.

In fact, for $m = 30$, it has been shown in [Law17] that the exact number of 31-bit primes is $50697537 > 5 \cdot 10^7$. Therefore in our implementation the support of u_0 will contain all monomials M_i and $\text{Supp}\{u_j\} \subseteq \text{Supp}\{u_0\}$ with probability $> 1 - k\frac{l}{5 \cdot 10^7}$.

We make one more key observation and claim that $\text{Supp}\{u_j\} \subseteq \text{Supp}\{u_{j-1}\}$ for $1 \leq j \leq l$ with high probability: We have

$$\begin{aligned} u_j &= s_{0j}M_0 + s_{1j}M_1 + \cdots + s_{kj}M_k. \\ u_{j+1} &= s_{0,j+1}M_0 + s_{1,j+1}M_1 + \cdots + s_{k,j+1}M_k. \end{aligned}$$

For a given $j > 0$, if $s_{i,j+1} \neq 0$, but $s_{ij} = 0$ then $M_i \in \text{Supp}(u_{j+1})$ but $M_i \notin \text{Supp}(u_j)$. We consider $\Pr[s_{ij} = 0 \,|\, s_{i,j+1} \neq 0]$. If A is the event that $s_{ij} = 0$ and B is the event that

$s_{i,j+1} = 0$ then

$$\Pr[A \mid B^c] = \frac{\Pr[A] - \Pr[B]\Pr[A \mid B]}{\Pr[B^c]} \leq \frac{\Pr[A]}{\Pr[B^c]}.$$

It follows that

$$\frac{\Pr[A]}{\Pr[B^c]} \leq \frac{l/(\pi(2^{m+1}) - \pi(2^m))}{1 - l/(\pi(2^{m+1}) - \pi(2^m))} = \frac{l}{(\pi(2^{m+1}) - \pi(2^m)) - l}$$

Hence the probability that

$$\Pr[\text{Supp}\{u_j\} \subseteq \text{Supp}\{u_{j-1}\} \mid 1 \leq j \leq l] > 1 - k\frac{l}{((\pi(2^{m+1}) - \pi(2^m)) - l}$$

As an example for $m = 30, l = 5, k = 500$, this probability is > 0.99993.

Hardy and Ramanujan [HR17] proved that for almost all integers, the number of distinct primes dividing a number s is $\omega(s) \approx \log\log(s)$. This theorem was generalized by Erdos-Kac which shows that $\omega(s)$ is essentially normally distributed [EK40]. By this approximation note that

$$\frac{\Pr[A]}{\Pr[B^c]} \lesssim \frac{\log\log(s_{ij})/(\pi(2^{m+1}) - \pi(2^m))}{1 - \log\log(s_{i,j+1})/(\pi(2^{m+1}) - \pi(2^m))} = \frac{\log(l\log p)}{(\pi(2^{m+1}) - \pi(2^m)) - \log(l\log p)}.$$

Hence the probability that $\text{Supp}\{u_j\} \subseteq \text{Supp}\{u_{j-1}\}$ is $\gtrsim 1 - k\frac{m\log(lm)}{2^m - m\log(lm)}$. As an example for $m = 30, l = 5, k = 500$, this probability is > 0.99995.

What does this mean in the context of multivariate factorization over mod $\mathbb{Z}_{p^l}$ for $l > 1$? It means that the solutions to the multivariate diophantine problems occurring in the lifting process will, with high probability, be a subset of the monomials of the solutions of the previous step and these solutions can be computed simply by solving transposed Vandermonde systems by using a machine prime p and hence by an efficient arithmetic using a sparse MDP solver as described in Algorithm 7.

We sum up the observations made in this section in the Theorem 30 below.

Theorem 30. *Let p be a randomly chosen m-bit prime, i.e. $p \in [2^m < p < 2^{m+1}]$. With the notation introduced in this section*

$$\Pr(\text{Supp}\{u_j\} \subseteq \text{Supp}\{u_{j-1}\} \text{ for all } 1 \leq j \leq l) > 1 - \frac{kl}{((\pi(2^{m+1}) - \pi(2^m)) - l}.$$

This probability can be approximated by

$$\Pr[\text{Supp}\{u_j\} \subseteq \text{Supp}\{u_{j-1}\} \text{ for all } 1 \leq j \leq l] \gtrsim 1 - \frac{km\log(lm)}{2^m - m\log(lm)}.$$

Algorithm 11 LiftTheFactors

Input : $a \in \mathbb{Z}[x_1, \ldots, x_n]$, $f_0, g_0 \in \mathbb{Z}_p[x_1, \ldots, x_n]$ where a, f_0, g_0 are monic in x_1 and $a = f_0 g_0$ in $\mathbb{Z}_p[x_1, \ldots, x_n]$. Also an integer bound $l > 0$ (For example, [Lemma 14, [Gel60]]).

Output : $f, g \in \mathbb{Z}[x_1, \ldots, x_j]$ such that $a = fg \in \mathbb{Z}[x_1, \ldots, x_n]$ or FAIL

1: **if** $\#f_0 \leq \#g_0$ **then** $(u_0, w_0) \leftarrow (f_0, g_0)$ **else** $(w_0, u_0) \leftarrow (f_0, g_0)$ **end if**
2: $(f, g) \leftarrow (\text{mods}(u_0, p) \text{mods}(w_0, p))$. (# *use symmetric range*)
3: *modulus* $\leftarrow 1$.
4: *error* $\leftarrow a - fg$, $\sigma_f \leftarrow$ skeleton of f
5: **for** i from 1 to l **while** *error* $\neq 0$ **do**
6: *modulus* $\leftarrow$ *modulus* $\times p$, $c \leftarrow (error/modulus) \mod p$
7: Solve the MDP $\sigma\, u_0 + \tau\, w_0 = c$ for σ and τ in $\mathbb{Z}_p[x_1, \ldots, x_n]$
8: using σ_f and sparse interpolation (Algorithm 7)
9: $(\sigma, \tau) \leftarrow (\text{mods}(\sigma, p), \text{mods}(\tau, p))$. (# *use symmetric range*)
10: **if** (σ, τ)=FAIL **then return** FAIL **end if**
11: $\sigma_f \leftarrow \sigma$, $(f, g) \leftarrow (f + \sigma \times modulus, g + \tau \times modulus)$.
12: *error* $\leftarrow a - fg$.
13: **end for**
14: **if** *error* $\neq 0$ **then return** FAIL **else return** (f, g) **end if**

Algorithm 12 LiftTheFactors (optimized)

Input : $a \in \mathbb{Z}[x_1, \ldots, x_n]$, $f_0, g_0 \in \mathbb{Z}_p[x_1, \ldots, x_n]$ where a, f_0, g_0 are monic in x_1 and $a = f_0 g_0$ in $\mathbb{Z}_p[x_1, \ldots, x_n]$. Also an integer bound $l > 0$ (For example, [Lemma 14, [Gel60]]).

Output : $f, g \in \mathbb{Z}[x_1, \ldots, x_j]$ such that $a = fg \in \mathbb{Z}[x_1, \ldots, x_n]$ or FAIL

1: **if** $\#f_0 \leq \#g_0$ **then** $(u_0, w_0) \leftarrow (f_0, g_0)$ **else** $(w_0, u_0) \leftarrow (f_0, g_0)$ **end if**
2: $(f, g) \leftarrow (\text{mods}(u_0, p), \text{mods}(w_0, p))$. (# *use symmetric range*)
3: *modulus* $\leftarrow 1$.
4: *error* $\leftarrow (a - fg)/p$, $\sigma_f \leftarrow$ skeleton of f
5: **for** i from 1 to l **while** *error* $\neq 0$ **do**
6: *modulus* $\leftarrow$ *modulus* $\times p$, $c \leftarrow error \mod p$
7: Solve the MDP $\sigma\, u_0 + \tau\, w_0 = c$ for σ and τ in $\mathbb{Z}_p[x_1, \ldots, x_n]$
8: using σ_f and sparse interpolation (Algorithm 7)
9: $(\sigma, \tau) \leftarrow (\text{mods}(\sigma, p), \text{mods}(\tau, p))$. (# *use symmetric range*)
10: **if** (σ, τ)=FAIL **then return** FAIL **end if**
11: $\sigma_f \leftarrow \sigma$, $error \leftarrow (error - (f\tau + g\sigma) + \sigma\tau \times modulus)/p$
12: $(f, g) \leftarrow (f + \sigma \times modulus, g + \tau \times modulus)$.
13: **end for**
14: **if** *error* $\neq 0$ **then return** FAIL **else return** (f, g) **end if**

$n/d/T_{f_i}$	t_{f_i}	l	l_B	D − MTSHL	MTSHL	t_{lift}
5/10/300	0.07	2	5	5.866 (5.101)	0.438 (0.132)	0.241
5/10/500	0.11	2	5	9.265 (7.937)	1.194 (0.186)	0.48
5/10/1000	0.23	2	5	14.448 (12.826)	2.202 (0.264)	1.332
5/10/300	0.07	4	9	6.923 (6.104)	1.067 (0.156)	0.553
5/10/500	0.11	4	9	10.971 (9.737)	1.854 (0.219)	1.231
5/10/1000	0.23	4	9	16.943 (15.183)	3.552 (0.35)	2.632
5/10/300	0.07	8	17	8.638 (7.596)	2.553 (0.201)	2.076
5/10/500	0.11	8	17	13.118 (11.686)	3.101 (0.28)	2.396
5/10/1000	0.23	8	17	19.031 (17.225)	4.905 (0.459)	4.032

Table 4.1: The timing table for D − MTSHL vs MTSHL

4.5.1 Data for sparse lifting

In Chapter 6 we give data for the performance of MTSHL. In this section, we give some data in Table 4.1 to compare the -direct approach-, i.e. implementing MTSHL so that it computes a bound l_B and factors staying in modulo $\mathbb{Z}_{p^{l_B}}$ arithmetic, with the -lift at the last step approach- , i.e. staying in $\mathbb{Z}_p$ arithmetic approach-, as explained in this Section.

We generated 2 random polynomials in n variables of total degree d with T_f terms of density ratio t_f containing some terms that have coefficients in mod p^l. Then we multiplied them in $\mathbb{Z}$, factored the expanded polynomial and factored it with the direct approach. We called the algorithm which uses the direct approach as D − MTSHL. Since D − MTSHL does not know what the actual value of l is, it needs to compute the coefficient bound l_B (using Lemma 14) and stays in the $\mathbb{Z}_{p^{l_B}}$ arithmetic. It factored the polynomial in $tX(tY)$ seconds where tY denotes the time spent on solving MDP's. Then we factored the polynomial with MTSHL which uses the -lift at the last step approach- explained in this Section. It factored it in $tX(tY)$ seconds, spent t_{lift} s for l liftings.

Chapter 5

The Distribution of Unlucky Points

5.1 Unlucky points in computer algebra

To investigate the efficiency and determine the complexity of the MTSHL algorithm we need to consider the probabilistic assumptions made by the design of MTSHL. Given two multivariate polynomials in $\mathbb{Z}_p[x_1, \ldots, x_n]$ and a random evaluation point $\gamma = (\gamma_2, \ldots, \gamma_n) \in \mathbb{Z}_p^{n-1}$, the solution to the MDP proposed by MTSHL as described in Section 4.4 uses the condition that $\gcd(f(x_1, \gamma), g(x_1, \gamma)) = 1$, which is true with high probability if p is big enough. A point γ which does not satisfy this condition is called an *unlucky* point.

Unlucky points appear elsewhere in computer algebra. Let A, B be polynomials in $\mathbb{Z}[x_0, x_1, \ldots, x_n]$ and $G = \gcd(A, B)$. The cofactors $\widehat{A}$ of A and $\widehat{B}$ of B are defined as $\widehat{A} = A/G$ and $\widehat{B} = B/G$ respectively. Thus $A = G\widehat{A}$ and $B = G\widehat{B}$. Modular GCD algorithms compute G modulo a sequence of primes $p_1, p_2, p_3, \ldots$ and recover the integer coefficients of G using Chinese remaindering. The fastest algorithms for computing G modulo a prime p interpolate G from univariate images. Maple, Magma and Singular all currently use Zippel's algorithm for computing G [Zip90]. Let us write

$$A = \sum_{i=0}^{k} a_i x_0^i, \quad B = \sum_{i=0}^{l} b_i x_0^i, \quad \text{and} \quad G = \sum_{i=0}^{m} c_i x_0^i$$

where the coefficients $a_i, b_i, c_i \in \mathbb{Z}_p[x_1, \ldots, x_n]$. Roughly, Zippel's algorithm picks points $\alpha_i \in \mathbb{Z}_p^n$, computes monic univariate images of G

$$g_i = \gcd(A(x_0, \alpha_i), B(x_0, \alpha_i)),$$

scales them, then interpolates the coefficients $c_i(x_1, \ldots, x_n)$ of G from the coefficients of these (scaled) images.

Consider the unlikely situation that $\gcd(\widehat{A}(x_0,\alpha_j),\widehat{B}(x_0,\alpha_j)) \neq 1$ for some j. For example, if

$$\widehat{A} = x_0^2 + x_2 \quad \text{and} \quad \widehat{B} = x_0^2 + x_2 + (x_1 - 1)$$

then $\gcd(\widehat{A},\widehat{B}) = 1$ but $\gcd(\widehat{A}(x_0,1,\beta),\widehat{B}(x_0,1,\beta)) \neq 1$ for any $\beta \in \mathbb{Z}_p$. This is a problem for the algorithm and hence the evaluation point $(1,\beta)$ is said to be *unlucky* because we cannot use the images $\gcd(A(x_0,1,\beta),B(x_0,1,\beta))$ to interpolate G.

The same issue of unlucky evaluation points arises in MTSHL, where, given polynomials $a,b,c \in \mathbb{Z}_p[x_1,\ldots,x_n]$ we want to solve the diophantine equation $\sigma a + \tau b = c$ for σ and τ in $\mathbb{Z}_p[x_1,\ldots,x_n]$ by interpolating σ and τ modulo a prime p from univariate images (see Section 3.6).

What is the maximum number of unlucky evaluation points that can occur? What is the expected number of unlucky evaluation points? We answer the first question for A and B monic in x_0. Proposition 20 implies $\gamma \in \mathbb{Z}_p^{n-1}$ is unlucky if and only if $R(\gamma) = 0$ where $R = \operatorname{res}_{x_0}(\widehat{A},\widehat{B}) \in \mathbb{Z}_p[x_1,x_2]$. If α_j is chosen at random from $\mathbb{Z}_p^2$ then by the Schwarz-Zippel lemma implies

$$\Pr[\, R(\alpha_j) = 0 \,] \leq \frac{\deg R}{p}.$$

Applying Proposition 20 once again gives $\deg R \leq \deg \widehat{A} \deg \widehat{B} \leq \deg A \deg B$. If the algorithm need t images to interpolate G modulo p, then we can avoid unlucky evaluation points with high probability if we pick $p \gg t \deg A \deg B$.

This is an upper bound, it is the worst case bound for the GCD algorithm. Furthermore, one can easily construct very sparse polynomials A, B where $\deg A$ and $\deg B$ are big enough to make this bound useless. Researchers in computer algebra have observed that unlucky evaluation points are rare in practice and that we "never see them" when testing algorithms on random input. Theorems 33 and 34 give the first results on the distribution of unlucky evaluation points. In particular, for co-prime $\widehat{A}$ and $\widehat{B}$ of positive degree, Theorem 34 implies $\Pr[\, \alpha_j \text{ is unlucky} \,] < 1/p$.

5.2 An overview of the Chapter

The main goal this chapter is to investigate the distribution of unlucky points. To manage the task we use a generalization of the Inclusion Exclusion principle (Proposition 32) which allows us to determine $\mathrm{E}[X]$ and $\mathrm{Var}[X]$ without having explicit formulas for $\Pr[X = k]$. Then in the second part, by the techniques developed in the first part we generalize a Theorem of Schmidt [Sch76] which considers the distribution of the number of roots of a polynomial in a univariate polynomial ring $\mathbb{F}_q[x]$ in $\mathbb{F}_q$, where $\mathbb{F}_q$ the finite field with q elements, to the ring of integers modulo n $\mathbb{Z}_n$ where n need not to be a prime.

5.2.1 An overview of the first part

What we will prove in the first part is not only true for the case where the coefficients of the polynomials are in $\mathbb{Z}_p$ but also true if they are in $\mathbb{F}_q$. So we switch the notation from $\mathbb{Z}_p$ to $\mathbb{F}_q$ for this section. Let X be a random variable which counts the unlucky points for a given multivariate polynomial pair (f, g) with coefficients in $\mathbb{F}_q$. In the first part we calculate the expected value $\mathrm{E}[X]$ of X. But one should also consider how smooth is the distribution. To see this we need to calculate the variance $\mathrm{Var}[X]$ of X which is difficult to compute. The main result we will prove in the first part is the theorem below.

Theorem. *Let $\mathbb{F}_q$ be a finite field with q elements, and let $f, g \in \mathbb{F}_q[x_1, x_2, \ldots, x_n]$ be of the form $f = c_l x_1^l + \sum_{i=0}^{l-1} c_{l-i}(x_2, \ldots, x_n)x^i$ and $g = d_m x_1^m + \sum_{i=0}^{m-1} d_{m-i}(x_2, \ldots, x_n)x^i$ where $c_l \neq 0$, $d_m \neq 0$, $\deg c_{l-i} \leq l - i$, and $\deg d_{m-i} \leq m - i$. Thus f and g have total degree l and m respectively. Let X be a random variable which counts the number of points $\gamma = (\gamma_2, \ldots, \gamma_n) \in \mathbb{F}_q^{n-1}$ such that $\gcd(f(x_1, \gamma_2, \ldots, \gamma_n), g(x_1, \gamma_2, \ldots, \gamma_n)) \neq 1$. If $n > 1$, $l > 0$ and $m > 0$ then*

$$E[X] = q^{n-2} \quad \text{and} \quad \mathrm{Var}[X] = q^{n-2}(1 - 1/q).$$

We initially found this result by direct evaluation. For quadratic polynomials f, g of the form $f = x^2 + (a_1 y + a_2)x + a_3 y^2 + a_4 y + a_5$ and $g = x^2 + (b_1 y + b_2)x + b_3 y^2 + b_4 y + b_5$ over finite fields of size $q = 2, 3, 4, 5, 8, 9, 11$ we generated all q^{10} pairs and computed $X = |\{\alpha \in \mathbb{F}_q : \gcd(f(x, \alpha), g(x, \alpha)) \neq 1\}|$. We repeated this for cubic polynomials and some higher degree bivariate polynomials for $q = 2, 3$ to verify that $\mathrm{E}[X] = 1$ and $\mathrm{Var}[X] = 1 - 1/q$ holds more generally. For yet higher degree polynomials we used random samples. That $\mathrm{E}[X] = 1$ independent of the degrees of f and g was a surprise to us. We had expected a logarithmic dependence on the degrees of f and g.

5.2.2 An overview of the second part

Let f be a polynomial in $\mathbb{Z}_n[x]$ of a given degree $d > 0$ and let X be a random variable which counts the number of distinct roots of f in $\mathbb{Z}_n$.

In the case when $n = p$ is a prime, $\mathbb{Z}_p$ is a finite field. In the finite field case, $\mathbb{F}_q$, where q is a power of a prime, the distribution of X has been studied extensively, see for example [Pan13]. Schmidt proves in Ch. 4 of [Sch76] the following result.

Theorem. *[Sch76]. Let q be a power of a prime and $\mathbb{F}_q$ be a finite field with q elements. Let X be a random variable which counts the number of distinct roots of a polynomial of degree d in $\mathbb{F}_q[x]$. Then for $d > 1$,* $\mathrm{E}[X] = 1$ and $\mathrm{Var[X]} = 1 - 1/\mathrm{q}$.

This result has been generalized by A. Knopfmacher and J. Knopfmacher in [KK93] who count distinct irreducible factors of a given degree of f.

When n is composite, the ring $\mathbb{Z}_n$ has zero divisors which complicates the investigation of the distribution of X. It is interesting to see that just as in the finite field case, here also $\mathrm{E}[X] = 1$ although zero divisors exist. Naturally we don't expect a smooth distribution in this case and want to investigate further $\mathrm{Var}[X]$.

Let $\phi(n) = |\{\, 1 \le i \le n : \gcd(i, n) = 1\}|$ denote Euler's totient function. In the second part we will generalize Schmidt's results and prove the following theorem below.

Theorem. *Let X be a random variable which counts the number of distinct roots of a monic polynomial in $\mathbb{Z}_n[x]$ of degree $m > 0$. Then $\mathrm{E}[X] = 1$. For $m = 1$ $\mathrm{Var}[X] = 0$. For $m > 1$*

$$\mathrm{Var}[X] = \sum_{d|n, d\neq n} \frac{d}{n}\phi(\frac{n}{d}) = \sum_{d|n} \frac{d-1}{n}\phi(\frac{n}{d})$$

In particular, if $n = p^k$ where p is a prime number and $k \ge 1$, $\mathrm{Var}[X] = k(1 - 1/p)$.

We found this result by direct computation and using the Online Encyclopedia of Integer Sequences (OEIS) see [OEIS]. For polynomials of degree $2, 3, 4, 5$ in $\mathbb{Z}_n[x]$, we computed $\mathrm{E}[X]$ and $\mathrm{Var}[X]$ for $n = 2, 3, 4, \ldots, 20$ using Maple and found that $\mathrm{E}[X] = 1$ in all cases. Values for the variance are given in the table below.

n	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\mathrm{Var}[X]$	$\frac{1}{2}$	$\frac{2}{3}$	1	$\frac{4}{5}$	$\frac{3}{2}$	$\frac{6}{7}$	$\frac{3}{2}$	$\frac{4}{3}$	$\frac{17}{10}$	$\frac{10}{11}$	$\frac{7}{3}$	$\frac{12}{13}$	$\frac{25}{14}$	2	2
$a(n)$	1	2	4	4	9	6	12	12	17	10	28	12	25	30	32

When we first computed $\mathrm{Var}[X]$ we did not recognize the numbers. Writing $\mathrm{Var}[X] = a(n)/n$ we computed the sequence for $a(n)$ (see the table) and looked it up in the OEIS. We found it is sequence A006579 and that $a(n) = \sum_{k=1}^{n-1} \gcd(n, k)$. The OEIS also has the formula $a(n) = \sum_{d|n}(d-1)\phi(\frac{n}{d})$.

5.3 Generalization of the inclusion-exclusion principle

In this section we will first introduce a notation and remind the reader of some basic counting principles (See for example, [GR03]). Next we will present our first result, the generalization of the inclusion-exclusion principle.

Given a set U and the finite collection of sets $\Gamma = \{A_i, i = 0, \ldots, n-1\}$ where each $A_i \subseteq U$, let us define $C_0 = U$, $C_{n+1} := \emptyset$ and, for $1 \le k \le n$,

$$C_k := \bigcup_{0\le i_1<\cdots<i_k\le n-1} (A_{i_1} \cap A_{i_2} \cdots \cap A_{i_k}).$$

Then for $1 \le k \le n$, C_k is the union of all possible intersections of the $k-$subsets of the collection Γ . In particular $C_1 = A_0 \cup A_1 \cup \cdots \cup A_{n-1}$ and $C_n = A_0 \cap A_1 \cap \cdots \cap A_{n-1}$. Let

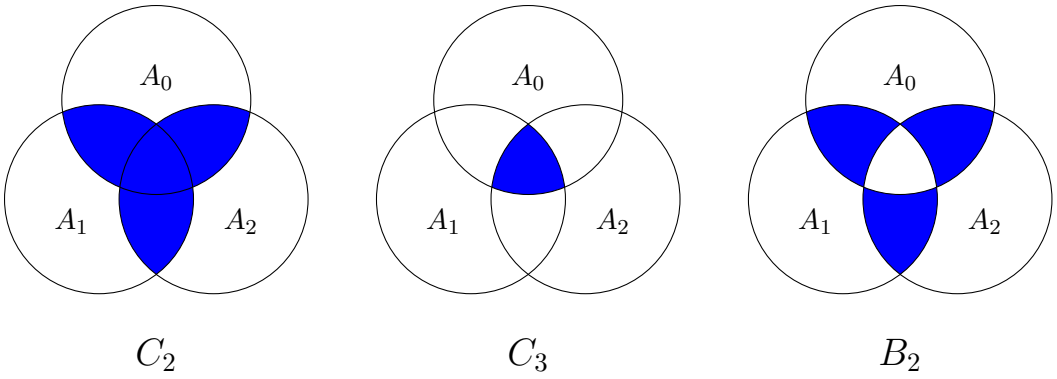


Figure 5.1: Show sets C_2, C_3, B_2 for three sets A_0, A_1, A_2

$B_k := C_k - C_{k+1}$ for $0 \leq k \leq n$. Observe that $C_k \supseteq C_{k+1}$, so $|B_k| = |C_k| - |C_{k+1}|$. Let us also define

$$b_k := |B_k| \text{ and } t_k := \sum_{0 \leq i_1 < \cdots < i_k \leq n-1} |A_{i_1} \cap A_{i_2} \cdots \cap A_{i_k}|.$$

Note that $t_1 = \sum_{i=0}^{n-1} |A_i|$ and $t_2 = \sum_{0 \leq i < j < n} |A_i \cap A_j|$. Also, $b_n = t_n$ and $b_{n-1} = t_{n-1} - \binom{n}{1} b_n$. Now $A_0 \cap A_1 \cap \cdots \cap A_{n-1}$ is a subset of $\binom{n}{n-2} = \binom{n}{2}$ sets of the form $A_{i_1} \cap A_{i_2} \cap \cdots \cap A_{i_{n-2}}$ and each $(n-1)$-section $A_{i_1} \cap A_{i_2} \cap \cdots \cap A_{i_{n-1}}$ is a subset of $\binom{n-1}{n-2} = \binom{n-1}{1}$ sets of the form $A_{i_1} \cap A_{i_2} \cap \cdots \cap A_{i_{n-2}}$ with $i_1 < i_2 < \cdots < i_{n-2}$. Therefore $b_{n-2} = t_{n-2} - \binom{n-1}{1} b_{n-1} - \binom{n}{2} b_n$.

Similarly, since each $(n-k+i)$-section is a subset of $\binom{n-k+i}{i}$ intersections of $(n-k)$ sets for $i = 1, \ldots, k$, we have the recursive formula

$$b_{n-k} = t_{n-k} - \sum_{i=1}^{k} \binom{n-k+i}{i} b_{n-k+i} \text{ for } k = 0, \ldots, n. \tag{5.1}$$

The following lemma is standard. See for example Theorem 8.2 in [GR03]. For completeness we give a proof.

Lemma 31. *Following the notation introduced above*

$$b_{n-k} = \sum_{i=0}^{k} (-1)^i \binom{n-k+i}{i} t_{n-k+i} \text{ for } k = 0, \ldots, n. \tag{5.2}$$

Proof. We will prove the claim by strong induction on k. For $k = 0$ we have $b_n = t_n$. Now assume that the claim is true for any integer $i \leq k$ in place of k.

By the recursive formula (5.1), we have

$$b_{n-(k+1)} = t_{n-(k+1)} - \binom{n-k}{1} b_{n-k} - \binom{n-k+1}{2} b_{n-k+1} - \cdots - \binom{n}{k+1} b_n.$$

On the other hand by using induction and multiplying each equation b_{n-j} with $-\binom{n-j}{k-j+1}$, we get the equations

$$\begin{aligned}
-\tbinom{n}{k+1}b_n &= -\tbinom{n}{k+1}t_n \\
-\tbinom{n-1}{k}b_{n-1} &= -\tbinom{n-1}{k}t_{n-1} + \tbinom{n-1}{k}\tbinom{n}{1}t_n \\
-\tbinom{n-2}{k-1}b_{n-2} &= -\tbinom{n-2}{k-1}t_{n-2} + \tbinom{n-2}{k-1}\tbinom{n-1}{1}t_{n-1} - \tbinom{n-2}{k-1}\tbinom{n}{2}t_n \\
&\vdots \\
-\tbinom{n-k}{1}b_{n-k} &= -\tbinom{n-k}{1}t_{n-k} + \tbinom{n-k}{1}\tbinom{n-k+1}{1}t_{n-k+1} - \cdots (-1)^{k+1}\tbinom{n-k}{1}\tbinom{n}{k}t_n.
\end{aligned}$$

After summing all these equalities, we consider the sum by focusing on the summands that are multiples of t_n on the right hand side:

$$c(t_n) = \sum_{i=0}^{k}(-1)^{k-i+1}\binom{n-k+i}{i+1}\binom{n}{k-i}.$$

For $d \leq k$ one has

$$\binom{n-d}{k-d+1}\binom{n}{d} = \binom{n}{k+1}\binom{k+1}{d}.$$

Then

$$c(t_n) = \binom{n}{k+1}\sum_{i=0}^{k}(-1)^{k-i+1}\binom{k+1}{k-i} = -\binom{n}{k+1}(-1)^k = (-1)^{k+1}\binom{n}{k+1}.$$

where the last equality follows from the binomial identity

$$\binom{k+1}{0} - \binom{k+1}{1} + \binom{k+1}{2} + \cdots + (-1)^k\binom{k+1}{k} = -(-1)^{k+1} = (-1)^k.$$

Similarly for $s = 1, \ldots, k$ we have

$$\begin{aligned}
c(t_{n-s}) &= \sum_{i=0}^{k-s}(-1)^{k-s-i+1}\binom{n-s-k+i}{i+1}\binom{n}{k-s-i} \\
&= \binom{n-s}{k-s+1}\sum_{i=0}^{k-s}(-1)^{k-s-i+1}\binom{n}{k-s-i} = (-1)^{k-s+1}\binom{n-s}{k-s+1}.
\end{aligned}$$

Finally, substituting $s = k - i$ in the formula above gives

$$b_{n-(k+1)} = \sum_{i=0}^{k}(-1)^{i+1}\binom{n-k+i}{i+1}t_{n-k+i}.$$

□

We are now ready to state the generalized inclusion-exclusion principle.

Proposition 32. *Following the same notation above, for* $1 \leq k \leq n$,

$$\sum_{i=0}^{n} i^k b_i = \sum_{i=1}^{k} i^k \left[\sum_{j=i}^{k} (-1)^{j-i} \binom{j}{j-i} t_j \right].$$

In particular:

(a) $\sum_{i=0}^{n} i b_i = t_1 = \sum_{i=0}^{n-1} |A_i|$ *(Inclusion Exclusion Principle) and;*

(b) $\sum_{i=0}^{n} i^2 b_i = t_1 + 2t_2 = \sum_{i=0}^{n-1} |A_i| + 2\sum_{i<j} |A_i \cap A_j|$.

Proof. According to Lemma 31 we have

$$\begin{aligned}
&b_n = t_n \\
&b_{n-1} = t_{n-1} - \tbinom{n}{1} t_n \\
&b_{n-2} = t_{n-2} - \tbinom{n-1}{1} t_{n-1} + \tbinom{n}{2} t_n \\
&\vdots \\
&b_2 = t_2 - \tbinom{3}{1} t_3 + \tbinom{4}{2} t_4 + \cdots + (-1)^{n-3} \tbinom{n-1}{n-3} t_{n-1} + (-1)^{n-2} \tbinom{n}{n-2} t_n \\
&b_1 = t_1 - \tbinom{2}{1} t_2 + \tbinom{3}{2} t_3 + \tbinom{4}{3} t_4 + \cdots + (-1)^{n-2} \tbinom{n-1}{n-2} t_{n-1} + (-1)^{n-1} \tbinom{n}{n-1} t_n
\end{aligned}$$

After summing $\sum_{i=1}^{n} i^k b_i$, for $1 \leq s \leq n$, we consider the sum by focusing on the summands that are multiples of t_s on the right hand side:

$$c(t_s) = \sum_{i=1}^{s} i^k \binom{s}{s-i} (-1)^{s-i}.$$

We claim that $c(t_s) = 0$ for $k < s \leq n$. We prove this by strong induction on k. For $k = 1$ we have

$$c(t_s) = \sum_{i=1}^{s} i \binom{s}{s-i} (-1)^{s-i} = \sum_{i=1}^{s} i \binom{s}{i} (-1)^{s-i} = \sum_{i=1}^{s} s \binom{s-1}{i-1} (-1)^{s-i}.$$

Since $s \geq 2$, we may substitute $m = s - 1 \geq 1$ and $j = i - 1$ to get

$$c(t_s) = s \sum_{j=0}^{m} \binom{m}{j} (-1)^{m-j} = s(1-1)^m = 0.$$

Now assume $\sum_{i=1}^{s} i^l \binom{s}{s-i} (-1)^{s-i} = 0$ for any $1 \leq l \leq k$ and $l + 1 \leq s \leq n$. Then

$$c(t_s) = \sum_{i=1}^{s} i^{k+1} \tbinom{s}{s-i} (-1)^{s-i} = \sum_{i=1}^{s} i^{k+1} \tbinom{s}{i} (-1)^{s-i} = \sum_{i=1}^{s} s i^k \tbinom{s-1}{i-1} (-1)^{s-i}.$$

Substitute $m = s - 1 \geq l \geq 1$ and $j = i - 1$ to obtain

$$\begin{aligned} c(t_s) &= s\sum_{j=0}^{m}(j+1)^k\binom{m}{j}(-1)^{m-j} = s\sum_{j=0}^{m}\sum_{l=0}^{k}\binom{k}{l}j^l\binom{m}{j}(-1)^{m-j} \\ &= s\sum_{l=0}^{k}\binom{k}{l}\sum_{j=0}^{m}j^l\binom{m}{j}(-1)^{m-j}. \end{aligned}$$

Since $l \leq k$, the induction hypothesis implies each summand

$$\sum_{j=0}^{m}j^l\binom{m}{j}(-1)^{m-j} = \sum_{j=1}^{m}j^l\binom{m}{m-j}(-1)^{m-j} = 0.$$

Hence $c(t_s) = 0$. On the other hand for $1 \leq s \leq k$ the sum of the summands containing i^s on the right hand side is $\sum_{j=i}^{k}(-1)^{j-i}\binom{j}{j-i}t_j$. Hence we have the result. In particular, for $k = 2$ the non-zero terms on the right-hand-side are $t_1 - \binom{2}{1}t_2 + 2^2t_2 = t_1 + 2t_2$. □

5.4 Results on the distribution of unlucky points

In this section we first consider the bivariate case and next we will prove the general multivariate version. The generalized inclusion exclusion principle will play a central role in the proofs .

Theorem 33. *Let $\mathbb{F}_q$ be a finite field with q elements, f, g be polynomials in $\mathbb{F}_q[x, y]$ of the form $f = c_n x^n + \sum_{i=0}^{n-1}\sum_{j=0}^{n-i} c_{ij}x^i y^j$ and $g = d_m x^m + \sum_{i=0}^{m-1}\sum_{j=0}^{m-i} d_{ij}x^i y^j$ with $c_n \neq 0$ and $d_m \neq 0$, thus of total degree n and m respectively. Let X be a random variable that counts the number of $\gamma \in \mathbb{F}_q$ such that $\gcd(f(x,\gamma), g(x,\gamma)) \neq 1$. If $n > 0$ and $m > 0$ then*

$$\mathrm{E}[X] = 1 \quad \text{and} \quad \mathrm{Var}[X] = 1 - 1/q.$$

Proof. Let us first explain the strategy of the proof. We note that without loss of generality we may assume f and g are monic in x because

$$\gcd(f(x,\gamma), g(x,\gamma)) = 1 \iff \gcd(c_n^{-1}f(x,\gamma), d_m^{-1}g(x,\gamma)) = 1$$

.

If we want to compute $\mathrm{E}[X]$ and $\mathrm{Var}[X]$ by using explicit formula of $\Pr[X = i]$ then the computations will be somewhat difficult. Instead we want to use the generalized inclusion-exclusion principle described in Proposition 33.

In the first part of the proof, to compute $E[X]$, for $\gamma \in \mathbb{F}_q$, let us define A_γ as the set of polynomial pairs $(f, g) \in \mathbb{F}_q[x, y]^2$ where f, g are monic in x with total degrees, $\deg(f) = n > 0$ and $\deg(g) = m > 0$ such that $\gcd(f(x,\gamma), g(x,\gamma)) \neq 1$. Our first goal will

be to compute $|A_0|$ and then show that $|A_0|=|A_\gamma|$ for $\gamma \neq 0$. Once we have this fact, we can invoke Proposition 32 and compute $\mathrm{E}[X]$ easily.

In the second part of the proof, to compute $\mathrm{Var}[X]$, for $(\gamma,\theta) \in \mathbb{F}_q^2$ with $\gamma < \theta$, let us define $A_{\gamma,\theta}$ as the set of bivariate polynomial pairs (f,g) where f,g are monic in x with total degrees, $\deg(f) = n > 0$ and $\deg(g) = m > 0$ such that $\gcd(f(x,\gamma), g(x,\gamma)) \neq 1$ and $\gcd(f(x,\theta), g(x,\theta)) \neq 1$. Our first goal will be to compute $|A_{0,1}|$ and then show that $|A_{0,1}| = |A_{\gamma,\theta}|$ for any pair (γ,θ) with $\gamma \neq \theta$. Once we have this fact we can invoke Proposition 32 again and compute $\mathrm{Var}[X]$ easily.

Let $(f,g) \in A_0$. Since f and g are monic in x, $f(x,0), g(x,0)$ are monic polynomials of degree n and m respectively in $\mathbb{F}_q[x]$. We have finitely many choices, say s, for non-relatively prime monic polynomial pairs $(h_i(x), l_i(x))$ with $\deg(h_i) = n$ and $\deg(l_i) = m$ with $i = 1,\ldots,s$ in $\mathbb{F}_q[x]^2$. Let $(f(x,0), g(x,0)) = (h_i(x), l_i(x))$ for some fixed i where $1 \leq i \leq s$. In fact $s = (q^n q^m)/q = q^{n+m-1}$, since there are $q^n q^m$ possible choices for monic polynomial pairs (h,l) in $\mathbb{F}_q[x]$ with $\deg(h) = n$, $\deg(l) = m$ and the probability that a given monic pair is non-relatively prime over $\mathbb{F}_q[x]$ is $1/q$ (see [Pan13, Ber68] and also [BC07] for an accessible proof).

Let $f(x,y) = x^n + c_{n-1}(y)x^{n-1} + \cdots + c_1(y)x + c_0(y)$ where $c_d(y) \in \mathbb{F}_q[y]$ of total degree $\deg(c_{n-d}(y)) \leq d$ and let $c_{n-d}(y) = a_d^{(n-d)} y^d + \cdots + a_0^{(n-d)}$ where $a_i^{(n-d)} \in \mathbb{F}_q$.

Let $h_i(x) = x^n + \alpha_{n-1}^{(i)} x^{n-1} + \cdots + \alpha_0^{(i)}$ with $\alpha_v^{(i)} \in \mathbb{F}_q$ for $0 \leq v \leq n-1$. Then for $1 \leq d \leq n$, we have $c_{n-d}(0) = a_0^{(n-d)} = \alpha_{n-d}^{(i)}$. It follows that there are q^d choices for such $c_{n-d}(y)$ and hence there are $q^1 q^2 \cdots q^n = q^{n(n+1)/2}$ choices for such $f(x,y)$. Similarly there are $q^{m(m+1)/2}$ choices for $g(x,y)$. Let us denote these numbers as $D = q^{n(n+1)/2}$ and $R = q^{m(m+1)/2}$. Since we have s choices for i, $|A_0| = sDR$.

On the other hand for a given $\gamma \in \mathbb{F}_q$ if $(f(x,y), g(x,y)) \in A_0$ then $(f(x,y-\gamma), g(x,y-\gamma)) \in A_\gamma$, since $f(x,y-\gamma)$ is again a bivariate polynomial which is a monic polynomial in x of total degree n and $g(x,y-\gamma)$ is again a bivariate polynomial which is a monic polynomial in x of total degree m. This correspondence (coordinate transformation) is bijective. Hence for any $\gamma \in \mathbb{F}_q$, one has $|A_\gamma| = sDR$.

For a general polynomial $f(x,y) \in \mathbb{F}_q[x,y]$ which is monic in x and of total degree $n > 0$, one has $q^2 q^3 \cdots q^{n+1} = q^n D$ choices. Similarly for a general polynomial $g(x,y) \in \mathbb{F}_q[x,y]$ which is monic in x and of total degree $m > 0$, one has $q^2 q^3 \cdots q^{m+1} = q^m R$ choices and therefore there are $q^{n+m}DR$ pairs (f,g) which are monic in x with total degrees $\deg(f) = n$ and $\deg(g) = m$.

Let $x_i := \Pr[X = i]$. This is the probability that $\gcd(f(x,\gamma), g(x,\gamma)) \neq 1$ for exactly i different γ's in $\mathbb{F}_q$, i.e. the probability that $(f,g) \in B_i$ in the notation introduced in section 5.3 considering the finite collection of sets $\Gamma = \{A_\gamma, \gamma \in \mathbb{F}_q\}$. Hence $x_i = \frac{b_i}{q^{n+m}DR}$. Then by

Proposition 32

$$\mathrm{E}[X] = \sum_{i=0}^{q} i x_i = \sum_{i=0}^{q} i \frac{b_i}{q^{n+m}DR} = \frac{\sum_{i=0}^{q-1} |A_i|}{q^{n+m}DR} = \frac{\sum_{i=0}^{q-1} sDR}{q^{n+m}DR} = \frac{qsDR}{q^{n+m}DR} = \frac{qq^{n+m-1}}{q^{n+m}} = 1.$$

To determine the variance of X, our proof assumes a set ordering of the elements of $\mathbb{F}_q$. For this purpose let us fix a generator α of $\mathbb{F}_q^*$ and use the ordering $0 < 1 < \alpha < \alpha^2 < \cdots < \alpha^{q-2}$.

Recall that $(\gamma, \theta) \in \mathbb{F}_q^2$ with $\gamma < \theta$, we defined $A_{\gamma,\theta}$ as the set of bivariate polynomial pairs (f, g) where f, g are monic in x with total degrees, $\deg(f) = n > 0$ and $\deg(g) = m > 0$ such that $\gcd(f(x, \gamma), g(x, \gamma)) \neq 1$ and $\gcd(f(x, \theta), g(x, \theta)) \neq 1$.

Let $(f, g) \in A_{0,1}$. Since f and g are monic in x, $f(x, 0), f(x, 1)$ are monic polynomials of degree n and $g(x, 0), g(x, 1)$ are monic polynomials of degree m in $\mathbb{F}_q[x]$. We have finitely many choices for non-relatively prime monic polynomial pairs $(h_i(x), l_i(x))$ with $\deg(h_i) = n$ and $\deg(l_i) = m$ with $i = 1, \ldots, s$ in $\mathbb{F}_q[x]^2$.

We have $(f(x, 0), g(x, 0)) = (h_i(x), l_i(x))$ and $(f(x, 1), g(x, 1)) = (h_j(x), l_j(x))$ for some fixed pair (i, j) where $1 \leq i, j \leq s$.

Now suppose $f(x, y) = x^n + c_{n-1}(y)x^{n-1} + \cdots + c_1(y)x + c_0(y)$ where $c_d(y) \in \mathbb{F}_q[y]$ of total degree $\deg(c_{n-d}(y)) \leq d$ and let $c_{n-d}(y) = a_d^{(n-d)} y^d + \cdots + a_0^{(n-d)}$ where $a_i^{(n-d)} \in \mathbb{F}_q$.

Let $h_i(x) = x^n + \alpha_{n-1}^{(i)} x^{n-1} + \cdots + \alpha_0^{(i)}$ and $h_j(x) = x^n + \beta_{n-1}^{(j)} x^{n-1} + \cdots + \beta_0^{(j)}$ with $\alpha_v^{(i)}, \beta_w^{(j)} \in \mathbb{F}_q$ for $0 \leq v, w \leq n-1$. Then for $1 \leq d \leq n$, we have $c_{n-d}(0) = a_0^{(n-d)} = \alpha_{n-d}^{(i)}$ and $c_{n-d}(1) = a_d^{(n-d)} + \cdots + a_1^{(n-d)} + a_0^{(n-d)} = \beta_{n-d}^{(j)}$.

It follows that there are q^{d-1} choices for such $c_{n-d}(y)$ and hence there are $q^0 q^1 \cdots q^{n-1} = q^{n(n-1)/2}$ choices for such $f(x, y)$.

Similarly there are $q^{m(m-1)/2}$ choices for $g(x, y)$. Let us call these numbers as $D_1 = q^{n(n-1)/2}$ and $R_1 = q^{m(m-1)/2}$. Since we have s^2 choices for (i, j) (i and j are need not be different, $|A_{0,1}| = s^2 D_1 R_1$.

On the other hand if $(f(x, y), g(x, y)) \in A_{0,1}$ then for $(\gamma, \theta) \in \mathbb{F}_q^2$ where $\gamma < \theta$, $(f(x, \frac{y-\gamma}{\theta-\gamma}), g(x, \frac{y-\gamma}{\theta-\gamma})) \in A_{\gamma,\theta}$ since $f(x, \frac{y-\gamma}{\theta-\gamma})$ is again a monic polynomial in x of total degree n and $g(x, \frac{y-\gamma}{\theta-\gamma})$ is again a monic polynomial in x of total degree m. This correspondence (coordinate transformation) is bijective and preserves relative primeness. Hence for a given $\gamma, \theta \in \mathbb{F}_q$ with $\gamma < \theta$, one has $|A_{\gamma,\theta}| = s^2 D_1 R_1$.

For a general bivariate polynomial $f(x, y) \in \mathbb{F}_q[x, y]$ which is monic in x and of total degree n, one has $q^2 q^3 \cdots q^{n+1} = q^{2n} D_1$ choices. Similarly for a general bivariate polynomial $g(x, y) \in \mathbb{F}_q[x, y]$ which is monic in x and of total degree m, one has $q^2 q^3 \cdots q^{m+1} = q^{2m} R_1$ choices. Therefore the number of bivariate polynomial pairs in (f, g) which are monic in x with total degrees, $\deg(f) = n$ and $\deg(g) = m$ is $q^{2n+2m} D_1 R_1$. Then with this notation we have $x_i = \frac{b_i}{q^{2n+2m} D_1 R_1}$.

Since we have $\binom{q}{2}$ choices for (γ, θ) with $\gamma < \theta$, $|A_{\gamma,\theta}| = s^2 D_1 R_1$ for all (γ, θ) with $\gamma < \theta$ and $\mathrm{E}[X] = 1$, by Proposition 32

$$
\begin{aligned}
\mathrm{Var}[X] &= \mathrm{E}[X^2] - \mathrm{E}[X]^2 \;=\; \mathrm{E}[X^2] - 1^2 \;=\; -1 + \sum_{i=0}^{q} i^2 x_i \\
&= -1 + \sum_{i=0}^{q} i^2 \frac{b_i}{q^{2n+2m} D_1 R_1} \;=\; -1 + \frac{\sum_{i=0}^{q} i^2 b_i}{q^{2n+2m} D_1 R_1} \\
&= -1 + \frac{\sum_{i=0}^{q-1} |A_i| + 2\sum_{i<j} |A_i \cap A_j|}{q^{2n+2m} D_1 R_1} \quad \text{(by Proposition33)} \\
&= -1 + \frac{\sum_{i=0}^{q-1} |A_i|}{q^{2n+2m} D_1 R_1} + \frac{2\sum_{i<j} s^2 D_1 R_1}{q^{2n+2m} D_1 R_1} \\
&= -1 + \frac{\sum_{i=0}^{q} i b_i}{q^{2n+2m} D_1 R_1} + \frac{2\binom{q}{2} s^2 D_1 R_1}{q^{2n} q^{2m} D_1 R_1} \\
&= -1 + \sum_{i=0}^{q} i \frac{b_i}{q^{2n+2m} D_1 R_1} + \frac{2\binom{q}{2} s^2 D_1 R_1}{q^{2n} q^{2m} D_1 R_1} \\
&= -1 + \sum_{i=0}^{q} i x_i + \frac{2\binom{q}{2} s^2 D_1 R_1}{q^{2n} q^{2m} D_1 R_1} \\
&= -1 + \mathrm{E}[X] + \frac{2\binom{q}{2} s^2 D_1 R_1}{q^{2n} q^{2m} D_1 R_1} \\
&= -1 + 1 + \frac{2\sum_{i<j} s^2 D_1 R_1}{q^{2n+2m} D_1 R_1} = \frac{2\binom{q}{2} s^2 D_1 R_1}{q^{2n} q^{2m} D_1 R_1} \\
&= \frac{q(q-1) q^{2n+2m-2}}{q^{2n+2m}} = \frac{q(q-1)}{q^2} \;=\; 1 - \frac{1}{q}.
\end{aligned}
$$

□

Theorem 34. *Let $\mathbb{F}_q$ be a finite field with q elements, $f, g \in \mathbb{F}_q[x_1, x_2, \ldots, x_n]$ be of the form $f = c_l x_1^l + \sum_{i=0}^{l-1} c_{l-i}(x_2, \ldots, x_n) x^i$ and $g = d_m x_1^m + \sum_{i=0}^{m-1} d_{m-i}(x_2, \ldots, x_n) x^i$ where $c_l \neq 0$, $d_m \neq 0$, $\deg c_{l-i} \leq l-i$, and $\deg d_{m-i} \leq m-i$, thus f and g have total degree l and m respectively. Let X be a random variable which counts the number of $\gamma = (\gamma_2, \ldots, \gamma_n) \in \mathbb{F}_q^{n-1}$ such that $\gcd(f(x_1, \gamma_2, \ldots, \gamma_n), g(x_1, \gamma_2, \ldots, \gamma_n)) \neq 1$. If $n > 1$, $l > 0$ and $m > 0$ then*

$$\mathrm{E}[X] = q^{n-2} \quad \text{and} \quad \mathrm{Var}[X] = q^{n-2}(1 - 1/q).$$

It follows that if γ is chosen at random from $\mathbb{F}_q^{n-1}$ then

$$\Pr[\, \gcd(f(x_1, \gamma_2, \ldots, \gamma_n), g(x_2, \gamma_2, \ldots, \gamma_n) \neq 1 \,] \;=\; \frac{q^{n-2}}{q^{n-1}} \;=\; \frac{1}{q}.$$

Proof. The proof runs along the same lines of the proof of Theorem 33. Let U be the set of all possible monic pairs $(f, g) \in \mathbb{F}_q[x_1, \ldots, x_n]^2$ where f, g are as described in the theorem. For $\alpha = (\alpha_2, \ldots, \alpha_n) \in \mathbb{F}_q^{n-1}$, let A_α be the set of all such polynomial pairs with $\gcd(f(x_1, \alpha), g(x_1, \alpha)) \neq 1$.

For the first part we will consider the monic pairs (f, g) with

$$\gcd(f(x_1, 0, 0 \ldots, 0), g(x_1, 0, 0 \ldots, 0)) \neq 1$$

and compute that the probability of this event is $1/q$ again. Then for a given non-zero $\alpha = (\alpha_2, \ldots, \alpha_n) \in \mathbb{F}_q^{n-1}$, considering the coordinate change

$$\bar{f}(x_1, \alpha_2, \ldots, \alpha_n) = f(x_1, x_2 - \alpha_2, \ldots, x_n - \alpha_n)$$

and using Proposition 32, since there are q^{n-1} possible such α's, we will see that $\mathrm{E}[X] = q^{n-1}q^{-1} = q^{n-2}$.

For the second part we will consider the monic pairs (f, g) with

$$\begin{aligned} \gcd(f(x_1, 0, 0 \ldots, 0), g(x_1, 0, 0 \ldots, 0)) &\neq 1 \\ \gcd(f(x_1, 1, 0 \ldots, 0), g(x_1, 1, 0 \ldots, 0)) &\neq 1 \end{aligned}$$

and see that probability of this event is $1/q^2$ again. For a given pair $(\alpha, \beta) \in \mathbb{F}_q^{n-1} \times \mathbb{F}_q^{n-1}$ with $\alpha \neq \beta$, this time the coordinate change of the second part of the proof that computes the variance may not be that obvious. We give the explicit construction below. Then by enumerating the elements of $\mathbb{F}_q^{n-1}$ from 0 to $q^{n-1} - 1$ and using Proposition 32, since there are $\binom{q^{n-1}}{2}$ possible pairs (α, β) with $\alpha < \beta$, we will see that

$$\begin{aligned} \mathrm{Var}[X] &= \mathrm{E}[X^2] - \mathrm{E}[X]^2 = -\mathrm{E}[X]^2 + \sum_{i=0}^{q^{n-1}-1} i^2 \mathrm{Pr}(X = i) \\ &= -\mathrm{E}[X]^2 + \sum_{i=0}^{q^{n-1}-1} i^2 \frac{b_i}{|U|} = -\mathrm{E}[X]^2 + \frac{\sum_{i=0}^{q^{n-1}-1} i^2 b_i}{|U|} \\ &= -\mathrm{E}[X]^2 + \frac{\sum_{i=0}^{q^{n-1}-1} |A_i| + 2\sum_{i<j} |A_i \cap A_j|}{|U|} \\ &= -\mathrm{E}[X]^2 + \sum_{i=0}^{q^{n-1}-1} \frac{1}{q} + 2\sum_{i<j} \frac{1}{q^2} \\ &= -\mathrm{E}[X]^2 + q^{n-1}\frac{1}{q} + 2\binom{q^{n-1}}{2}\frac{1}{q^2} \\ &= -q^{2n-4} + q^{n-2} + q^{n-1}(q^{n-1} - 1)q^{-2} \\ &= q^{n-2} - q^{n-3} = q^{n-2}(1 - \frac{1}{q}). \end{aligned}$$

For a given pair $(\alpha, \beta) \in \mathbb{F}_p^{n-1} \times \mathbb{F}_p^{n-1}$ with $\alpha \neq \beta$, let $\alpha = (\alpha_2, \ldots, \alpha_n)$ and $\beta = (\beta_2, \ldots, \beta_n)$. Our aim is to find a coordinate change such that

$$\begin{aligned} \bar{f}(x_1, \alpha_2, \ldots, \alpha_n) &= f(x_1, 0, 0 \ldots, 0) \\ \bar{f}(x_1, \beta_2, \ldots, \beta_n) &= f(x_1, 1, 0 \ldots, 0) \end{aligned}$$

where

$$\bar{f}(x_1, x_2, \ldots, x_n) = f(x_1, a_{20} + a_{22}x_2 + \cdots + a_{2n}x_n, \ldots, a_{n0} + a_{n2}x_n + \cdots + a_{nn}x_n).$$

Note that this transformation does not change the leading term in x_1, so it preserves monicness, degree in x_1, and preserves coprimality. To make this transformation bijective we need a $(n-1) \times (n-1)$ matrix

$$A = \begin{pmatrix} a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \vdots \\ a_{n2} & \ldots & a_{nn} \end{pmatrix}$$

which is invertible and to satisfy the relations

$$\begin{aligned} a_{22}\alpha_2 + \cdots + a_{2n}\alpha_n &= -a_{20} \\ a_{22}\beta_2 + \cdots + a_{2n}\beta_n &= 1 - a_{20} \end{aligned}$$

and for $3 \leq j \leq n$

$$\begin{aligned} a_{j2}\alpha_2 + \cdots + a_{jn}\alpha_n &= -a_{j0} \\ a_{j2}\beta_2 + \cdots + a_{jn}\beta_n &= -a_{j0}. \end{aligned}$$

Let us consider α and β as column vectors and suppose that α and β are linearly independent over $\mathbb{F}_p$. Then there exists a pair (i, j) such that $\begin{vmatrix} \alpha_i & \alpha_j \\ \beta_i & \beta_j \end{vmatrix} \neq 0$. Applying the necessary permutation if needed, we may assume that $\begin{vmatrix} \alpha_2 & \alpha_3 \\ \beta_2 & \beta_3 \end{vmatrix} \neq 0$. Then consider the invertible $(n-1) \times (n-1)$ matrix B

$$B := \begin{pmatrix} \alpha_2 & \alpha_3 & \ldots & \ldots & \alpha_n \\ \beta_2 & \beta_3 & \ldots & \ldots & \beta_n \\ 0 & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 1 \end{pmatrix}$$

Let a_j denote $a_j = (a_{22} \cdots a_{2n})^T$ for $2 \leq j \leq n$. Let $a_2 := B^{-1}(-1\,0 \cdots 0)^T$, so that $a_2 \cdot \alpha^T = -1$, $a_2 \cdot \beta^T = 0$. Let $a_3 := B^{-1}\left(1\,\mathbf{e}_1^T\right)^T$ so that $a_3 \cdot \alpha^T = 1$, $a_3 \cdot \beta^T = 1$ and let $a_j := B^{-1}(0\,\mathbf{e}_{j-2}^T)^T$ so that $a_j \cdot \alpha^T = 0$, $a_j \cdot \beta^T = 0$ for $3 < j < n$ where $\mathbf{e}_i$'s denote canonical basis vectors for $\mathbb{Z}_p^{n-2}$. Let also $a_{20} = 1, a_{30} = -1$ and $a_{j0} = 0$. Now, if we define $A = (a_2 \cdots a_n)^T$ then by construction of a_i's we have

$$A \cdot \alpha = \begin{pmatrix} a_2 \cdot \alpha^T \\ a_3 \cdot \alpha^T \\ \vdots \\ a_n \cdot \alpha^T \end{pmatrix} = \begin{pmatrix} -1 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \text{ and } A \cdot \beta = \begin{pmatrix} a_2 \cdot \beta^T \\ a_3 \cdot \beta^T \\ \vdots \\ a_n \cdot \beta^T \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Hence we get $a_2 \cdot \alpha^T + a_{20} = -1 + 1 = 0$ and $a_2 \cdot \beta^T + a_{20} = 0 + 1 = 1$ as needed. Also $a_3 \cdot \alpha^T + a_{30} = 1 - 1 = 0$ and $a_3 \cdot \beta^T + a_{30} = 1 - 1 = 0$ as needed. Also $a_j \cdot \alpha^T + a_{j0} = 0 + 0 = 0$ and $a_j \cdot \beta^T + a_{j0} = 0 + 0 = 0$ as needed. It remains to show that the set $\{a_2, a_3, \ldots, a_n\}$ is linearly independent.

Now, since B is invertible, the set $\{a_2, a_3, \ldots, a_n\}$ is linearly independent iff the set $\{Ba_2, Ba_3, \ldots, Ba_n\}$ is linearly independent. Let for some $\gamma_i \in \mathbb{F}_p, 2 \leq i \leq n$ we have

$$\gamma_2 \begin{pmatrix} -1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} + \gamma_3 \begin{pmatrix} 1 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} + \gamma_4 \begin{pmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix} + \cdots + \gamma_n \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix} = 0.$$

Then it can be easily seen that $\gamma_i = 0$ and hence $\{a_2, a_3, \ldots, a_n\}$ is linearly independent. It follows that A is invertible and the translation we have constructed

$$\begin{pmatrix} a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \vdots \\ a_{n2} & \ldots & a_{nn} \end{pmatrix} \begin{pmatrix} x_2 \\ \vdots \\ x_n \end{pmatrix} + \begin{pmatrix} 1 \\ -1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

satisfies the conditions we needed.

Example: Let $\alpha = (2, 0, 0, 0)$ and $\beta = (2, 3, 0, 1)$ in $\mathbb{Z}_5^4$. Then

$$B = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 2 & 3 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \Rightarrow B^{-1} = \begin{pmatrix} 3 & 0 & 0 & 0 \\ 3 & 2 & 0 & 3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \bmod 5.$$

$$a_2 = B^{-1}\begin{pmatrix} -1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \\ 0 \\ 0 \end{pmatrix}, a_3 = B^{-1}\begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 3 \\ 0 \\ 0 \\ 0 \end{pmatrix}, a_4 = B^{-1}\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, a_5 =$$
$$B^{-1}\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 3 \\ 0 \\ 1 \end{pmatrix}.$$ Then the transformation is

$$\begin{pmatrix} 2 & 2 & 0 & 0 \\ 3 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 3 & 0 & 1 \end{pmatrix}\begin{pmatrix} x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} + \begin{pmatrix} 1 \\ -1 \\ 0 \\ 0 \end{pmatrix}$$

and $\bar{f}(x_1, x_2, x_3, x_4, x_5) = f(x_1, 1 + 2x_2 + 2x_3, -1 + 3x_2, x_4, 3x_3 + x_5)$ and

$$\begin{aligned} \bar{f}(x_1, 2, 0, 0, 0) &= f(x_1, 0, 0, 0, 0) \\ \bar{f}(x_1, 2, 3, 0, 1) &= f(x_1, 1, 0, 0, 0). \end{aligned}$$

Now suppose that α and β are linearly dependent over $\mathbb{F}_p$. Again applying the necessary permutation if needed, we may assume that $0 \neq \alpha_2$ and $\alpha_2 \neq \beta_2$. Let $a_{20} = -\alpha_2/(\beta_2 - \alpha_2)$,$a_{22} = 1/(\beta_2 - \alpha_2)$ and $a_{2j} = 0$ for $3 \leq j \leq n$. Then we have $a_2 \cdot \alpha^T + a_{20} = \frac{\alpha_2}{\beta_2 - \alpha_2} - \frac{\alpha_2}{\beta_2 - \alpha_2} = 0$ and $a_2 \cdot \beta^T + a_{20} = \frac{\beta_2}{\beta_2 - \alpha_2} - \frac{\alpha_2}{\beta_2 - \alpha_2} = 1$ as needed.
Now consider the $1 \times (n-1)$ matrix $B = (\alpha_2 \cdots \alpha_n)$. Since $\alpha \neq 0$ $\dim(\mathrm{Ker}(B)) = n - 2$. Let $v_2, \ldots, v_n$ be a basis for $\mathrm{Ker}(B)$. Let for some $\gamma_i \in \mathbb{F}_p, 2 \leq i \leq n$ we have

$$\gamma_2 a_2 + \gamma_3 v_3 + \gamma_4 v_4 + \cdots + \gamma_n v_n = 0.$$

Then applying B from left hand side we have $\gamma_2 B a_2 = \gamma_2 \alpha_2/(\beta_2 - \alpha_2) = 0 \Rightarrow \gamma_2 = 0$. It follows that $\gamma_i = 0$ and $\{a_2, v_2, \ldots, v_n\}$ is linearly independent. Then $A := (a_2\, v_2 \cdots v_n)^T$ is invertible. It can be readily verified that the translation we have constructed

$$A\begin{pmatrix} x_2 \\ \vdots \\ x_n \end{pmatrix} + \begin{pmatrix} \frac{-\alpha_2}{\beta_2 - \alpha_2} \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

satisfies the remaining conditions needed.
Example: Let $\alpha = (1, 1, 0, 0)$ and $\beta = (2, 2, 0, 0)$ in $\mathbb{Z}_5^4$. Then $a_{20} = -\alpha_2/(\beta_2 - \alpha_2) = 4$ and

$a_{22} = 1/(\beta_2 - \alpha_2) = 1 \bmod 5$. Also, $B = (1\,1\,0\,0)$ and

$$v_3 = \begin{pmatrix} 3 \\ 2 \\ 0 \\ 0 \end{pmatrix}, v_4 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, v_5 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

and then the transformation is

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 3 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} + \begin{pmatrix} 4 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

where $\bar{f}(x_1, x_2, x_3, x_4, x_5) = f(x_1, 4 + x_2, 3x_2 + 2x_3, x_4, x_5)$ and

$$\begin{aligned} \bar{f}(x_1, 1, 1, 0, 0) &= f(x_1, 0, 0, 0, 0) \\ \bar{f}(x_1, 2, 2, 0, 0) &= f(x_1, 1, 0, 0, 0). \end{aligned}$$

□

5.4.1 A comparison with the binomial distribution.

Let Y be a random variable with a binomial distribution parameterized by n and p where n indicates the number of trials and p indicates the probability p. So $0 \leq Y \leq n$, $\Pr[Y = k] = \binom{n}{k} p^k (1-p)^{n-k}$, $E[Y] = np$ and $\mathrm{Var}[Y] = np(1-p)$. We noticed that the mean and variance of X in Theorem 34 is the same as the mean and variance of the binomial distribution $B(n, p)$ with $n = q$ trials and probability $p = 1/q$. In Table 1 below we compare the two distributions for

$f = x^2 + (a_1 y + a_2)x + (a_3 y^2 + a_4 y + a_5)$ and
$g = x^2 + (b_1 y + b_2)x + (b_3 y^2 + b_4 y + b_5)$

in $\mathbb{F}_q[x, y]$ with $q = 7$. Note that there are 7^{10} pairs for f, g. In Table 5.1 F_k is the number of pairs for which $\gcd(f(x, \alpha), g(x, \alpha)) \neq 1$ for exactly k values for $\alpha \in \mathbb{F}_7$. We computed F_k by computing this gcd for all distinct pairs using Maple. The values for B_k come from $B(7, 1/7)$. They are computed by $B_k = 7^{10} \Pr[Y = k]$.

k	0	1	2	3	4	5	6	7
F_k	96606636	110666892	56053746	17287200	1728720	0	0	132055
$F_k/7^{10}$	.34200036	.39177554	.19843773	0.061199	0.00612	0.	0.	0.00047
B_k	96018048	112021056	56010528	15558480	2593080	259308	14406	343
$B_k/7^{10}$	.33991668	.39656946	.19828473	0.0550791	0.0092	0.00092	0.0001	1.2e-6

Table 5.1: Data for quadratic (f, g) in $\mathbb{F}_7[x, y]$

The two zeros F_5 and F_6 can be explained as follows. Let $R(y)$ be the Sylvester resultant of f and g. Then applying Lemma 1 we have $R(\alpha) = 0 \iff \gcd(f(x,\alpha), g(x,\alpha)) \neq 1$ for $\alpha \in \mathbb{F}_q$. For our quadratic polynomials f and g, Lemma 1(ii) implies $\deg R \leq \deg f \deg g = 4$.

Hence $R(y)$ can have at most 4 distinct roots unless f and g are not co-prime in $\mathbb{F}_7[x,y]$ in which case $R(y) = 0$ and it has 7 roots. Therefore $F_5 = 0$, $F_6 = 0$ and $F_7 = 132055$ is the number pairs f, g which are not co-prime in $\mathbb{F}_7[x,y]$.

5.5 Results on the distribution of the roots of a monic polynomial in $\mathbb{Z}_n[x]$

Theorem 35. *Let X be a random variable which counts the number of distinct roots of a monic polynomial in $\mathbb{Z}_n[x]$ of degree $m > 0$. Then $\mathrm{E}[X] = 1$. For $m = 1$ $\mathrm{Var}[X] = 0$ and for $m > 1$*

$$\mathrm{Var}[X] = \sum_{d|n, d\neq n} \frac{d}{n}\phi(\frac{n}{d}) = \sum_{d|n} \frac{d-1}{n}\phi(\frac{n}{d}).$$

In particular, if $n = p^k$ where p is a prime number and $k \geq 1$, $\mathrm{Var}[X] = k(1 - 1/p)$.

Proof. Let A_i be the set of all monic univariate polynomials of degree $m > 0$ which have a root at $\alpha_i \in \mathbb{Z}_n$. Then since $x - \alpha_i$ is monic, for any $f \in A_i$ we have $f = (x - \alpha_i)q$ for a unique $q \in \mathbb{Z}_n[x]$ and n^{m-1} choices for such an f. Hence $|A_i| = n^{m-1}$.

Let $x_i := \Pr[X = i]$, this is the probability that f has exactly i distinct roots, i.e. $f \in B_i$ in the notation introduced in Section 5.3 considering the finite collection of sets $\Gamma = \{A_i, i = 0, \ldots, n-1\}$. Since we have n^{m-1} choices for a monic polynomial of degree m in $\mathbb{Z}_n[x]$ we have $x_i = \frac{b_i}{n^{m-1}}$. Then by Proposition 32,

$$\mathrm{E}[X] = \sum_{i=0}^{n} i x_i = \sum_{i=0}^{n} i \frac{b_i}{n^m} = \frac{\sum_{i=0}^{n} i b_i}{n^m} = \frac{\sum_{i=0}^{n-1} |A_i|}{n^m} = \frac{\sum_{i=0}^{n-1} n^{m-1}}{n^m} = \frac{n n^{m-1}}{n^m} = 1.$$

To prove the second part, if $m = 1$ then $f = x - \alpha$ for some $\alpha \in \mathbb{Z}_n$ and hence $X = 1$ and $\mathrm{Var}[X] = 0$. For $m > 1$ and $\alpha \in \mathbb{Z}_n^*$, our first aim is to find $|A_0 \cap A_\alpha|$. Let $f \in A_0 \cap A_\alpha$. It may not be the case that $f = x(x - \alpha)q$ for a unique $q \in \mathbb{Z}_n[x]$, since $\mathbb{Z}_n[x]$ is not a unique factorization domain in general. However, $f = xq_1 = (x - \alpha)q_2$ for unique $q_1, q_2 \in \mathbb{Z}_n[x]$. It follows that $\alpha q_2(0) = 0 \bmod n$. If $\gcd(\alpha, n) = d$, then $\gcd(\frac{\alpha}{d}, \frac{n}{d}) = 1$ and hence $q_2(0) = 0 \bmod \frac{n}{d}$. The general form is $q_2 = x^{m-1} + a_{m-2}x^{m-2} + \cdots + a_0$ where $a_i \in \mathbb{Z}_n$ for $i = 0, \ldots, m-2$. Since $q_2(0) = a_0 \bmod \frac{n}{d}$, there are d choices for a_0 and hence there are dn^{m-2} choices for q_2. Therefore $|A_0 \cap A_\alpha| = dn^{m-2}$.

For a given pair (γ, β) with $\beta > \gamma$, to compute $|A_\gamma \cap A_\beta|$, define $\alpha := \beta - \gamma$ and consider $A_0 \cap A_\alpha$. If $f \in A_\gamma \cap A_\beta$, then we have $f(x) = (x - \gamma)q_3(x) = (x - \beta)q_4(x)$ for unique $q_3, q_4 \in \mathbb{Z}_n[x]$. By the coordinate translation $x \mapsto x + \gamma$, we have $f(x + \gamma) \in A_0 \cap A_\alpha$ since

$f(x+\gamma) = xq_3(x+\gamma) = (x-\alpha)q_4(x+\gamma)$ where $f(x+\gamma), q_3(x+\gamma), q_4(x+\gamma)$ are monic and with the same degree before the translation. This correspondence is bijective and it follows that $|A_\gamma \cap A_\beta| = |A_0 \cap A_\alpha| = dn^{m-2}$.

Let $d = \gcd(\alpha, n)$. There are $k = \phi(\frac{n}{d})$ elements $\beta_1, \ldots, \beta_k$ in $\mathbb{Z}_{\frac{n}{d}}$ such that $\gcd(\beta_j, \frac{n}{d}) = 1$. If we define $\alpha_j := d\beta_j \in \mathbb{Z}_n$, then $\gcd(\alpha_j, n) = d$. For, if $s = \gcd(\alpha_j, n)$ and $d|s$, then $s|\alpha_j \Rightarrow s|d\beta_j \Rightarrow \frac{s}{d}|\beta_j$ and $\frac{s}{d}|\frac{n}{d} \Rightarrow \frac{s}{d}|\gcd(\beta_j, \frac{n}{d}) \Rightarrow \frac{s}{d}|1 \Rightarrow s = d$. Now, for each j consider the $n-\alpha_j$ pairs of the form $(i, i+\alpha_j)$ where $i = 0, \ldots, n-\alpha_j-1$. We have $|A_i \cap A_{i+\alpha_j}| = |A_0 \cap A_{\alpha_j}|$ and

$$\sum_{\beta>\gamma, d=\gcd(\beta-\gamma,n)} |A_\gamma \cap A_\beta| = \sum_{j=1}^{k} (n-\alpha_j)|A_0 \cap A_{\alpha_j}| = \sum_{j=1}^{k} (n-\alpha_j) dn^{m-2} = dn^{m-2} \sum_{j=1}^{k} n - \alpha_j$$

where $d = \gcd(\alpha_j, n)$ and $k = \phi(\frac{n}{d})$. Since $\gcd(n, \alpha_j) = d \iff \gcd(n, n-\alpha_j) = d$ we have $\sum_{j=1}^{k} n - \alpha_j = \sum_{j=1}^{k} \alpha_j$. Then

$$2\sum_{j=1}^{k} \alpha_j = \sum_{j=1}^{k} \alpha_j + \sum_{j=1}^{k} n - \alpha_j = \sum_{j=1}^{k} n = kn = \phi(\frac{n}{d})n \quad \Longrightarrow \quad \sum_{j=1}^{k} \alpha_j = \frac{n}{2}\phi(\frac{n}{d}).$$

It follows that

$$\sum_{\beta>\gamma, d=\gcd(\beta-\gamma,n)} |A_\gamma \cap A_\beta| = dn^{m-2} \sum_{j=1}^{k} n - \alpha_j = dn^{m-2} \sum_{j=1}^{k} \alpha_j = \frac{n}{2}\phi(\frac{n}{d}) dn^{m-2}.$$

Then by Proposition 32, it follows that

$$\begin{aligned}
\mathrm{Var}[X] &= \mathrm{E}[X^2] - \mathrm{E}[X]^2 = -1^2 + \mathrm{E}[X^2] \\
&= -1 + \sum_{i=0}^{n} i^2 x_i = -1 + \sum_{i=0}^{n} i^2 \frac{b_i}{n^m} = -1 + \frac{\sum_{i=0}^{n} i^2 b_i}{n^m} \\
&= -1 + \frac{\sum_{i=0}^{n-1} |A_i| + 2\sum_{i<j} |A_i \cap A_j|}{n^m} \\
&= -1 + \frac{nn^{m-1}}{n^m} + \frac{2\sum_{d|n\, d\neq n} \frac{n}{2}\phi(\frac{n}{d}) dn^{m-2}}{n^m} \\
&= 2 \sum_{d|n\, d\neq n} \frac{n}{2}\phi(\frac{n}{d}) dn^{-2} \;=\; \sum_{d|n\, d\neq n} \frac{d}{n}\phi(\frac{n}{d}).
\end{aligned}$$

Also, since by Gauss' Lemma $\sum_{d|n} \phi(\frac{n}{d}) = n$ we have

$$\begin{aligned}
\sum_{d|n} \frac{d-1}{n}\phi(\frac{n}{d}) &= \sum_{d|n} \frac{d}{n}\phi(\frac{n}{d}) - \frac{1}{n}\sum_{d|n} \phi(\frac{n}{d}) \\
&= \phi(1) + \sum_{d|n, d\neq n} \frac{d}{n}\phi(\frac{n}{d}) - \frac{1}{n}n \;=\; \sum_{d|n, d\neq n} \frac{d}{n}\phi(\frac{n}{d}).
\end{aligned}$$

To prove the last claim, let $n = p^k$ where p is a prime number and $k \geq 1$. Then

$$\sum_{d|n, d\neq n} \frac{d}{n}\phi(\frac{n}{d}) \quad = \quad \sum_{s=0}^{k-1} \frac{p^s}{p^k}\phi(\frac{p^k}{p^s}) = \sum_{s=0}^{k-1} p^{s-k}p^{k-s-1}(p-1) = k(1-1/p).$$

□

5.6 Summary

As we indicated in the beginning of the Chapter, to investigate the efficiency and determine the complexity of the MTSHL algorithm, we needed to consider the probabilistic assumptions made by the design of MTSHL.

To manage this task, we generalized the inclusion-exclusion principle in Section 5.3 as Proposition 32.

Theorem 34 in Section 5.4 shows that in the MTSHL algorithm the average probability of hitting an unlucky point is $1/p$. It assures us that if p is big enough, then the probability that MTSHL fails because of an unlucky event is very small. It also shows when p is big enough, the distribution of unlucky points is smooth.

Theorem 35 in Section 5.5 generalizes the Theorem of Schmidt which considers the distribution of number of roots of a polynomial in a univariate polynomial ring $\mathbb{F}_q[x]$ over a finite field $\mathbb{F}_q$ with q elements, to the ring of integers modulo n, $\mathbb{Z}_n$. The expected number of roots of a univariate polynomial is 1 as in the finite field case. The distribution is not smooth which reminds us that we can't rely on the expected value of a distribution without investigating the variance.

Chapter 6

Complexity of MTSHL

6.1 The steps of the analysis

In this Chapter we begin with studying what happens to the sparsity of multivariate polynomials when the variables are successively evaluated at numbers. We determine the expected number of remaining terms and the variance.

Next in Subsection 6.2.1 we question the assumptions made by Zippel in his complexity analysis of sparse interpolation [Zip79]. By the help of the results of Section 6.2, we revise this assumption and correct his analysis.

The solutions to the MDP are the coefficients of a Taylor series of a polynomial expanded about a random point; the expected number of terms of these coefficients is determined in Section 6.3.

The complicated nature of MTSHL makes the analysis tedious. Section 6.4 is written to help the reader follow Section 6.5 where we use these results to analyze the complexity of MTSHL.

Finally in Section 6.6 we confirm our theoretical estimations with experimental data. We give some timing data to compare MTSHL with Maple's factorization algorithm. To show that our comparison is fair, we also include timings for Magma and Singular's factorization algorithms.

6.2 The expected number of terms after evaluation

The complexity of MTSHL depends on the number of terms in the factors, and the number of terms of each factor is expected to decrease from the step $j+1$ to j after evaluation. To make our complexity analysis as precise as possible, we must give an upper bound for the expected sizes of the factors in each step j. In this section we compute these bounds and confirm our theoretical estimations by experimental data.

Let p be a big prime and $f \in \mathbb{Z}_p[x_1, \ldots, x_n]$ be a randomly chosen multivariate polynomial of degree $\leq d$ that has T non-zero terms. Let s be the number of all possible monomials of degree $\leq d$, i.e. $s = \binom{n+d}{n}$. An upper bound for T is then given by $T \leq s$. By randomly chosen we mean that the probability of occurrence of each monomial is the same and equal to $1/s$. We think of choosing a random polynomial of degree $\leq d$ that has T terms as choosing T distinct monomials out of s choices and choosing coefficients uniformly at random from $[0, p-1]$.

Let also $g = f(x_n = \alpha_n)$ for a randomly chosen non-zero element $\alpha_n \in \mathbb{Z}_p$. Before evaluation let $f = \sum_{i=1}^{t} \mathbf{m}_i c_i(x_n)$ where $\mathbf{m}_i$ are monomials in the variables $x_1, \ldots, x_{n-1}$. Then $T = \sum_{i=1}^{t} \#c_i(x_n)$ and $g = \sum_{i=1}^{t} \mathbf{m}_i c_i(\alpha_n)$. One has

$$\Pr[c_i(\alpha_n) = 0] \leq \frac{\deg c_i(x_n)}{p} \leq \frac{d}{p}$$

for each i. If p is much bigger than d and α_n is random, we don't expect any $c_i(\alpha_n) = 0$ for $1 \leq i \leq t$. In the following first we assume that $c_i(\alpha_n) \neq 0$ for $1 \leq i \leq t$ and compute the expected value of number of terms T_g of g, in terms of T, d and n. Then the upper bounds on the expected value of T_g will be valid even for the case in which some of the $c_i(\alpha_n) = 0$.

Let Y_k be a random variable that counts the number of terms of the k^{th} homogeneous component f_k of f which is of homogeneous of degree k in the variables $x_1, \ldots, x_{n-1}$. We have $f = \sum_{k=0}^{d} f_k$ and $\deg_{x_n} f_k \leq d - k$ for $0 \leq k \leq d$.

Example 36. Let $n = 3, d = 6$ and

$$f = \underbrace{1 + x_3^5}_{f_0} + \underbrace{2x_1x_3^4}_{f_1} + \underbrace{x_1x_2^2(3x_3 + 4x_3^2) + 5x_1^2x_2x_3^3}_{f_3} + \underbrace{x_1^2x_2^2(6x_3 + 7x_3^2)}_{f_4} + \underbrace{8x_1^3x_2^3}_{f_6}$$

with $f_2 = f_5 = 0$. $Y_0 = 2$, $Y_1 = 1$, $Y_3 = 3$, $Y_4 = 2$, $Y_6 = 1$, $Y_2 = Y_5 = 0$.

Let s_k be the number of all possible monomials in the variables $x_1, \ldots, x_{n-1}$with homogeneous degree k, i.e. $s_k = \binom{n-2+k}{n-2}$ and $d_k = d - k + 1$ for $0 \leq k \leq d$. Since the number of all possible monomials in the variables $x_1, \ldots, x_{n-1}$ and homogeneous of degree k in the variables $x_1, \ldots, x_{n-1}$ up to degree d is $s_k d_k$, we have $\sum_{k=0}^{d} s_k d_k = s$ and

$$\Pr[Y_k = j] = \frac{1}{\binom{s}{T}} \binom{s_k d_k}{j} \binom{s - s_k d_k}{T - j}.$$

This is a hyper-geometric distribution, its expected value and variance are well-known

$$E[Y_k] = T\frac{s_k d_k}{s} \text{ and } \operatorname{Var}[Y_k] = \frac{(s - s_k d_k)(s - T)T s_k d_k}{s^2(s-1)}.$$

Let X_k be a random variable that counts the number of terms of the k^{th} homogeneous component g_k of g which is homogeneous of degree k in the variables $x_1, \ldots, x_{n-1}$. Let us define the random variable $X = \sum_{k=0}^{d} X_k$. Then X counts the number of terms T_g of g.

Example 37. Let $n = 3, d = 6$ and $\alpha_3 = 1$. Below $g = f(x_3 = 1)$.

$$\begin{array}{rcccccc} f & = & \underbrace{1 + x_3^5}_{Y_0=2} & \underbrace{+2x_1x_3^4}_{Y_1=1} & \underbrace{+x_1x_2^2(3x_3 + 4x_3^2) + 5x_1^2x_2x_3^3}_{Y_3=3} & \underbrace{+x_1^2x_2^2(6x_3 + 7x_3^2)}_{Y_4=2} & \underbrace{+8x_1^3x_2^3}_{Y_6=1} \\ & & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ g & = & \underbrace{2}_{X_0=1} & \underbrace{+2x_1}_{X_1=1} & \underbrace{+7x_1x_2^2 + 5x_1^2x_2}_{X_3=2} & \underbrace{+13x_1^2x_2^2}_{X_4=1} & \underbrace{+8x_1^3x_2^3}_{X_6=1} \end{array}$$

So $Y = \sum_{k=0}^{6} Y_k = 9$ and $X = \sum_{k=0}^{6} X_k = 6$.

The expected number of terms of g is $E[X] = \sum_{k=0}^{d} E[X_k]$. We have

$$\begin{aligned} E[X_k] = \sum_{i=0}^{s_k} i \Pr[X_k = i] &= \sum_{i=0}^{s_k} i \sum_{j=0}^{s_k d_k} \Pr[X_k = i \,|\, Y_k = j] \Pr[Y_k = j] \\ &= \sum_{j=0}^{s_k d_k} \Pr[Y_k = j] \sum_{i=0}^{s_k} i \Pr[X_k = i \,|\, Y_k = j]. \end{aligned}$$

Our first goal is to find the conditional expectation

$$E[X_k \,|\, Y_k = j] = \sum_{i=0}^{s_k} i \Pr[X_k = i \,|\, Y_k = j].$$

To this end, let M_k be the set of all monomials in $x_1, \ldots, x_{n-1}$ with homogeneous degree k. We have $|M_k| = s_k$. Let

$$M_k = \{\mathbf{m}_1, \ldots, \mathbf{m}_{s_k}\}.$$

For $1 \leq i \leq s_k$ and $j > 0$, let A_i be the set of all non-zero polynomials in $\mathbb{Z}_p[x_1, \ldots, x_n]$ with j number of terms that are homogeneous of total degree k in the variables $x_1, \ldots, x_{n-1}$, have degree $< d_k$ in the variable x_n and does not include a term of the form $c\mathbf{m}_i x_n^r$ for any $0 \leq r < d_k$ for some non-zero $c \in \mathbb{Z}_p$. So using the table below, A_i is the set of all non-zero polynomials whose support does not contain any monomial from the i^{th} row. Note that if $f_k \in A_i$ then $\#f(x_n = \alpha_n) \leq s_k - 1$.

	1	x_n	$\ldots$	$x_n^{d_k-1}$
$\mathbf{m_1}$	$\mathbf{m_1}$	$\mathbf{m_1}x_n$	$\ldots$	$\mathbf{m_1}x_n^{d_k-1}$
$\mathbf{m_2}$	$\mathbf{m_2}$	$\mathbf{m_2}x_n$	$\ldots$	$\mathbf{m_2}x_n^{d_k-1}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\mathbf{m_{s_k}}$	$\mathbf{m_{s_k}}$	$\mathbf{m_{s_k}}x_n$	$\ldots$	$\mathbf{m_{s_k}}x_n^{d_k-1}$

If f_k is the k^{th} homogeneous component of f in the first $n-1$ variables, then if no zero evaluation occurs in the coefficients of f_k in the variables x_n (as it was assumed), we have

$$f_k \in \bigcup_{i=1}^{s_k} A_i \Longleftrightarrow \#f_k(x_1, \ldots, x_{n-1}, \alpha) \leq s_k - 1.$$

Example 38. Let $n = 3, k = 3, d_3 = 3, j = 4$. We have

$$M_k = \{\mathbf{m}_1 = x_1^3, \mathbf{m}_2 = x_1^2 x_2, \mathbf{m}_3 = x_1 x_2^2, \mathbf{m}_4 = x_2^3\}.$$

with $s_k = 4$. Consider the polynomials

$$F = x_1 x_2^2 + x_2^3 x_3^2 + x_1^3(x_3 + x_3^2) \in A_2\,,\; G = x_1 x_2^2 x_3^2 + x_1^3(1 + x_3 + x_3^2) \in A_2 \cap A_4.$$

So, $\#F(x_n = \alpha_n) \leq 4 - 1 = 3$ and $\#G(x_n = \alpha_n) \leq 4 - 2 = 2$.

Let us define

$$C_l := \bigcup_{i_1 < \cdots < i_l} (A_{i_1} \cap A_{i_2} \cdots \cap A_{i_l}) \text{ and } B_l := C_l - C_{l+1}$$

for $1 \leq l \leq s_k - 1$. Let us also define $B_{s_k} = C_{s_k} = \bigcap_{i=1}^{s_k} A_i$

$$b_l := |B_l| \text{ and } m_l := \sum\nolimits_{i_1 < \cdots < i_l} |A_{i_1} \cap A_{i_2} \cdots \cap A_{i_l}| \,\text{for}\, 1 \leq l \leq s_k.$$

so that $m_1 = \sum_{i=1}^{s_k} |A_i|$, and $m_2 = \sum_{1 \leq i < j \leq s_k} |A_i \cap A_j|$

With this notation we have

$$f_k \in B_l \Longleftrightarrow \#f_k(x_1, \ldots, x_{n-1}, \alpha) = s_k - l.$$

Let $v_j := \binom{s_k d_k}{j}$ and $q := (p-1)$. Assuming that no zero evaluation occurs, we have

$$\Pr[X_k = s_k - l \,|\, Y_k = j] = \frac{|B_l|}{q^j v_j} \tag{6.1}$$

It can be seen by counting that

$$\Pr[X_k = l \,|\, Y_k = j] = v_j^{-1} \sum_{i=0}^{l} (-1)^i \tbinom{s_k - (l-i)}{i} \tbinom{s_k}{l-i} \tbinom{d_k(l-i)}{j}.$$

As this formula is not easy to manipulate, we invoke the methods developed in Chapter 5.

We have $|A_i| = q^j \binom{(s_k-1)d_k}{j}$ and $|A_i \cap A_l| = q^j \binom{(s_k-2)d_k}{j}$ for $1 \leq i, l \leq s_k$ where $i \neq l$. By Proposition 32, $\sum_{i=1}^{s_k} i|B_i| = m_1$ and $\sum_{i=1}^{s_k} i^2|B_i| = m_1 + 2m_2$. To find the expected value and the variance of X, let us first define $w_j := \binom{(s_k-1)d_k}{j}$ and the random variable

$Z_k := s_k - X_k$. Then $Z_k = i \iff X_k = s_k - i$. Recall that $v_j := \binom{s_k d_k}{j}$. Then we have

$$\sum_{i=1}^{s_k} i\Pr[Z_k = i \,|\, Y_k = j] \overset{(6.1)}{=} \sum_{i=1}^{s_k} i\frac{|B_l|}{q^j v_j} = \frac{\sum_{i=1}^{s_k} i|B_l|}{q^j v_j} = \frac{\sum_{i=1}^{s_k} |A_i|}{q^j v_j} = s_k \frac{w_j}{v_j}.$$

Since $E[X_k \,|\, Y_k = j] = E[s_k - Z_k \,|\, Y_k = j] = s_k - E[Z_k \,|\, Y_k = j]$, we have

$$E[X_k \,|\, Y_k = j] = \sum_{i=0}^{s_k} i \Pr[X_k = i \,|\, Y_k = j] = s_k(1 - \frac{w_j}{v_j}).$$

To save some space let $y_{kj} := \Pr[Y_k = j]$. Then

$$\begin{aligned}
E[X] &= \sum_{k=0}^{d} E[X_k] = \sum_{k=0}^{d} \sum_{j=0}^{s_k d_k} y_{kj} \sum_{i=0}^{s_k} i \Pr[X_k = i \,|\, Y_k = j] \\
&= \sum_{k=0}^{d} \sum_{j=0}^{s_k d_k} y_{kj} s_k (1 - \frac{w_j}{v_j}) = \sum_{k=0}^{d} \sum_{j=0}^{s_k d_k} y_{kj} s_k - \sum_{k=0}^{d} \sum_{j=0}^{s_k d_k} y_{kj} s_k \frac{w_j}{v_j} \\
&= \sum_{k=0}^{d} s_k \sum_{j=0}^{s_k d_k} y_{kj} - \sum_{k=0}^{d} \sum_{j=0}^{s_k d_k} \frac{1}{\binom{s}{T}} \binom{s_k d_k}{j} \binom{s - s_k d_k}{T - j} s_k \frac{\binom{(s_k-1)d_k}{j}}{\binom{s_k d_k}{j}} \\
&= \sum_{k=0}^{d} s_k - \frac{1}{\binom{s}{T}} \sum_{k=0}^{d} s_k \sum_{j=0}^{s_k d_k} \binom{s - s_k d_k}{T - j} \binom{(s_k - 1)d_k}{j} \\
&= \sum_{k=0}^{d} s_k - \sum_{k=0}^{d} s_k \frac{\binom{s-d_k}{T}}{\binom{s}{T}} = \sum_{k=0}^{d} s_k \left(1 - \frac{\binom{s-d_k}{T}}{\binom{s}{T}}\right).
\end{aligned}$$

Note that what we have done so far can easily be generalized when the number of evaluation points are more than one : Let for $0 < m < n$

$$g = f(x_1, \ldots x_{n-m}, x_{n-m+1} = \alpha_{n-m+1}, \ldots, x_n = \alpha_n)$$

for m randomly chosen non-zero elements $\alpha_{n-m+1}, \ldots, \alpha_n \in \mathbb{Z}_p$ and $s = \binom{n+d}{n}$, $s_k = \binom{n-m-1+k}{n-m-1}$, $d_k = \binom{d-k+m}{m}$. If no zero evaluation occurs at the coefficients of f and we define the random variable $Y = \sum_{k=0}^{d} Y_k$ which counts the number of terms of f, then what we get is the conditional expectation

$$E[X \,|\, Y = T] = \sum_{k=0}^{d} s_k \left(1 - \frac{\binom{s-d_k}{T}}{\binom{s}{T}}\right). \tag{6.2}$$

From now on, to save some space, when it is clear from the context, we will use the notation $E[X]$ instead of $E[X \,|\, Y = T]$. Although it is not difficult to compute, the formula (6.2) is not useful. In order to have a smooth formulation in our complexity analysis, we

want a good approximation. First, we observe

$$\binom{s-d_k}{T}/\binom{s}{T} = (1-\frac{T}{s})(1-\frac{T}{s-1})\cdots(1-\frac{T}{s-d_k+1}).$$

For $0 \le i < d_k$ we have $(1-\frac{T}{s}) - (1-\frac{T}{s-i}) = \frac{iT}{s(s-i)} < \frac{d_k T}{s(s-d_k)}$. Let $\gamma_i := \frac{iT/s}{s-i}$. Then

$$\prod_{i=0}^{d_k-1}(1-\frac{T}{s-i}) = \prod_{i=0}^{d_k-1}(1-\frac{T}{s}-\gamma_i) = (1-\frac{T}{s})^{d_k} + Er$$

where

$$Er = \sum_{l=1}^{d_k}(-1)^l \left(\sum_{0\le i_1<\cdots<i_l\le d_k-1} \prod_{j=1}^{l} \gamma_{i_j} \right) (1-\frac{T}{s})^{d_k-l}.$$

Since

$$\prod_{i=1}^{l}\gamma_i < \left(\frac{d_k T/s}{s-d}\right)^l = \left(\frac{d_k}{s}\right)^l \left(\frac{T/s}{1-d_k/s}\right)^l \to 0 \text{ as } \frac{d_k}{s} \to 0$$

we see that $Er \to 0$ and hence the ratio $\binom{s-d_k}{T}/\binom{s}{T} \to (1-\frac{T}{s})^{d_k}$ as $\frac{d_k}{s} \to 0$.

Remark 39. From now on, unless indicated, whenever we use the symbol $\approx$ or $\lesssim$ we mean that in the calculation the approximation

$$\binom{s-d_k}{T}/\binom{s}{T} \approx (1-t_f)^{d_k} \tag{6.3}$$

is used (with error Er) where $t_f = T/s$ is the density ratio of f. When m is not close to n, since $s \in \mathcal{O}(n^d)$ and in the sparse case t_f is relatively small, the error Er is very close to zero not only asymptotically but also for practical values for n and d (see Example 40).

Now we stick with the case $m = 1$. For a single evaluation, we expect

$$E[X] \approx \sum_{k=0}^{d} s_k \left(1-(1-t_f)^{d_k}\right). \tag{6.4}$$

So, in the dense case where t_f is very close to 1, we expect approximately $\sum_{k=0}^{d} s_k$ many terms, i.e. most of the possible monomials in the variables $x_1, \ldots, x_{n-1}$ up to degree d. In the very sparse case where t_f is very close to zero, using the approximation $(1-t_f)^{d_k} \approx 1-d_k t_f$, we expect approximately $t_f \sum_{k=0}^{d} s_k d_k = t_f s = T$ terms as we intuitively expect.

Eqn (6.4) above is the expected number of terms $E[T_g]$ of g. Let γ be the number of all possible monomials in the variables $x_1, \ldots, x_{n-1}$ up to degree $d-1$, i.e. $\gamma = \binom{n+d-1}{n-1}$. Dividing the both sides of the equation (6.4) by γ we get the expected density ratio

$$E[T_g]/\gamma = E[T_g/\gamma] \Rightarrow E[t_g] = 1 - \gamma^{-1} \sum_{k=0}^{d} s_k \frac{\binom{s-d_k}{st_f}}{\binom{s}{st_f}}. \tag{6.5}$$

So, we have an induced function $e_t : t_f \mapsto E[t_g]$. Using Eqn (6.5) we get

$$E[t_g] \approx 1 - \gamma^{-1} \sum_{k=0}^{d} s_k (1 - t_f)^{d_k}. \tag{6.6}$$

Example 40. Table 6.1 below are the results of experiments with 4 random polynomials with $n = 7$ variables and degree $d = 15$. T_{g_i} and t_{g_i} are the actual number of terms and the density ratio of each $g_i = f_i(x_n = \alpha_i)$. $E[t_{g_i}]$ and eT_{g_i} are the expected number of terms of g_i based on Eqns (6.2) and (6.4) resp. $E[t_{g_i}]$ and et_{g_i} are the expected density ratio of g_i based on Eqns (6.5) and (6.6) resp.

	T_{f_i}	t_{f_i}	T_{g_i}	$E[T_{g_i}]$	eT_{g_i}	t_{g_i}	$E[t_{g_i}]$	et_{g_i}
f_1	17161	.100625	14356	14370.47	14370.36	.264558	.264825	.264823
f_2	19887	.116609	16196	16221.84	16221.73	.298466	.298943	.298941
f_3	29845	.174998	22303	22211.09	22210.96	.411009	.409315	.409313
f_4	39823	.233505	27244	27199.53	27199.41	.502063	.501244	.501242

Table 6.1: Expected number of terms after evaluation

Note that the polynomial function $e_t(y) = 1 - \gamma^{-1} \sum_{k=0}^{d} s_k (1-y)^{d_k}$ is strictly increasing on the interval $[0, 1]$, since $e_t'(y) > 0$ on the interval [0,1]. Also $e_t(0) = 0$ and $e_t(1) = 1$. For a given $0 \leq y_0 \leq 1$, consider the function $h(y) := e_t(y) - y_0$. We have $h(0) \leq 0$ and $h' > 0$ on [0,1]. Hence $h(y)$ has only 1 real root in $[0, 1]$. This helps us to guess T_f when we're given only T_g. Here is one example in the reverse direction:

Example 41. We call Maple's `randpoly` command to give a random sparse polynomial of degree 15 in 7 variables. It gives us a polynomial f with $T_f = 25050$. Suppose we don't know T_f. Then we choose a random point α and evaluate $g = f(x_n = \alpha)$. We compute $T_g = 19395$. Then we compute $t_g = 19395/\binom{7-1+15}{15} \approx 0.3574192835$ and $\gamma = \binom{7+15-1}{15}$. Using (6.6) we seek for the solutions of the polynomial equation

$$0.3574192835 = 1 - \gamma^{-1} \sum_{k=0}^{15} \binom{5+k}{k} (1-y)^{16-k}.$$

This polynomial equation has only one real root $y = 0.1461603065$ in [0,1]. So we guess $E[t_f] \approx 0.1461603065$. The actual density ratio is $t_f = 25050/\binom{7+15}{15} = 0.1461089220$. Our guess implies $E[T_f] \approx t_f \binom{7+15}{7} = 24926$, whereas $T_f = 25050$. We repeated this example with 4 random polynomials f_i where $g_i = f_i(x_n = \alpha_i)$. The results of the experiments are in Table 6.2.

Finally, following the notation in the beginning of the section, if some of the $c_i(\alpha_n) = 0$ for $1 \leq i \leq t$, then we consider $\tilde{f} = \sum_{k=0}^{t_{i_k}} c_{i_k}(x_n) \mathbf{m}_{i_k}$ where $\mathrm{Supp}(\tilde{f}) \subseteq \mathrm{Supp}(f)$ and

	Tg_i	t_{g_i}	$E[t_{f_i}]$	t_{f_i}	$E[T_{f_i}]$	T_{f_i}
f_1	14967	.2758182220	.1056824733	.1052983394	18023	17958
f_2	14597	.2689997051	.1025359262	.1020792288	17486	17409
f_3	14439	.2660880142	.1012024458	.1008713294	17259	17203
f_4	14375	.2649085950	.1006640188	.1005605592	17167	17150

Table 6.2: Expected density ratio after evaluation

$c_{i_k}(\alpha_n) = 0$. Then,

$$E[T_f] = E[T_{f-\tilde{f}}] \lesssim \sum_{k=0}^{d} s_k \left(1 - (1 - t_f)^{d_k}\right).$$

What we have found is an upper bound for $E[X]$. On the other hand since we choose $\alpha_n \neq 0$ a non-zero monomial does not evaluate to zero. That is, if $\#c_i(x_n) = 1$ then $\mathbf{m}_i c_i(\alpha_n) \neq 0$. We should apply the zero evaluation probability for the terms $\mathbf{m}_i c_i(x_n)$ for which $\#c_i(x_n) \geq 2$. Then for given $f \in \mathbb{Z}_p[x_1, \ldots, x_n]$ of degree $\leq d$ with T terms, the probability that zero evaluation occurs for a randomly chosen non-zero $\alpha_n \in \mathbb{Z}_p$ is $\leq \frac{dT}{2p}$. Note that, this is also true when the number of evaluations are more than one by using Swartz-Zippel Lemma.

To see how spread out the distribution from the mean is, we must compute the variance. First we consider the sum $A = \sum_{i=0}^{s_k} i^2 \Pr[Z_k = i \,|\, Y_k = j]$

$$\begin{aligned}
A &= \sum_{i=0}^{s_k} (s_k - i)^2 \Pr[Z_k = s_k - i \,|\, Y_k = j] \\
&= s_k^2 - 2s_k \sum_{i=0}^{s_k} i \Pr[X_k = i \,|\, Y_k = j] + \sum_{i=0}^{s_k} i^2 \Pr[X_k = i \,|\, Y_k = j] \\
&= s_k^2 - 2s_k(s_k - s_k \frac{w_j}{v_j}) + \sum_{i=0}^{s_k} i^2 \Pr[X_k = i \,|\, Y_k = j] \\
&= -s_k^2(1 - 2\frac{w_j}{v_j}) + \sum_{i=0}^{s_k} i^2 \Pr[X_k = i \,|\, Y_k = j].
\end{aligned}$$

Then

$$\sum_{i=0}^{s_k} i^2 \Pr[X_k = i \,|\, Y_k = j] = \sum_{i=0}^{s_k} i^2 \Pr[Z_k = i \,|\, Y_k = j] + s_k^2(1 - 2\frac{w_j}{v_j}).$$

Let $r_j := \binom{(s_k-2)d_k}{j}$. Recall that $|A_i| = q^j w_j$ and $|A_i \cap A_l| = q^j v_j$ for $i \neq l$. So

$$\begin{aligned}\sum_{i=1}^{s_k} i^2 \Pr[Z_k = i \,|\, Y_k = j] &= \sum_{i=1}^{s_k} i^2 \frac{|B_l|}{q^j v_j} = \frac{\sum_{i=1}^{s_k} i^2 |B_l|}{q^j v_j} \\ &= \frac{\sum_{i=1}^{s_k} |A_i|}{q^j v_j} + 2\frac{\sum_{1\le i,l\le s_k} |A_i \cap A_l|}{q^j v_j} \\ &= s_k \frac{w_j}{v_j} + 2\binom{s_k}{2}\frac{r_j}{v_j}.\end{aligned}$$

Then

$$\sum_{i=0}^{s_k} i^2 \Pr[X_k = i \,|\, Y_k = j] = s_k \frac{w_j}{v_j} + 2\binom{s_k}{2}\frac{r_j}{v_j} + s_k^2(1 - 2\frac{w_j}{v_j}).$$

Hence

$$\begin{aligned}\mathrm{Var}[X_k] &= E[X_k^2] - E[X_k]^2 = s_k \frac{w_j}{v_j} + 2\binom{s_k}{2}\frac{r_j}{v_j} + s_k^2(1 - 2\frac{w_j}{v_j}) - s_k^2(1 - \frac{w_j}{v_j})^2 \\ &= s_k \frac{w_j}{v_j} + s_k(s_k-1)\frac{r_j}{v_j} - \left(s_k \frac{w_j}{v_j}\right)^2.\end{aligned}$$

It follows that

$$\begin{aligned}\sum_{k=0}^{d} \mathrm{Var}[X_k] &\approx \sum_{k=0}^{d} s_k(1-t_f)^{d_k} + \sum_{k=0}^{d} s_k(s_k-1)(1-t_f)^{2d_k} - \sum_{k=0}^{d} s_k^2(1-t_f)^{2d_k} \\ &= \sum_{k=0}^{d} s_k \left((1-t_f)^{d_k} - (1-t_f)^{2d_k}\right).\end{aligned}$$

Note that $\sum_{k=0}^{d} \mathrm{Var}[X_k]$ is not equal to $\mathrm{Var}[X] = \sum_{k=0}^{d} \mathrm{Var}[X_k] + \sum_{k\neq l}^{d} \mathrm{Covar}[X_k, X_l]$, but it gives a good approximation. For example, for the sparse case where t_f is close to zero, using the approximation $(1-t_f)^{d_k} \approx 1 - d_k t_f$, we expect

$$\begin{aligned}\sum_{k=0}^{d} s_k \left((1-t_f)^{d_k} - (1-t_f)^{2d_k}\right) &\approx \sum_{k=0}^{d} s_k \left((1 - d_k t_f) - (1 - 2d_k t_f)\right) \\ &= t_f \sum_{k=0}^{d} s_k d_k = t_f s = T_f.\end{aligned}$$

The sum of the squares of the deviations of each X_k from $E[X_k]$ is T_f. We guess that the variance should be small. As an experiment, for $n = 7, d = 13$ we have generated 1000 random polynomials where $T_f = 1716$ for each of them. Each of them has density ratio $t_f = 0.01$. The expected number of terms after evaluation at a random non-zero point is

1684.14 and $\sum_{k=0}^{d} \mathrm{Var}[X_k] = 1606.30$. We calculated then the sample variance and found that it is 27.7.

Lemma 42. *Let p be a big prime and $f \in \mathbb{Z}_p[x_1, \ldots, x_n]$ be a random multivariate polynomial of degree of at most d that has T_f non-zero terms. Also let $0 < m < n$ and*

$$g = f(x_1, \ldots x_{n-m}, x_{n-m+1} = \alpha_{n-m+1}, \ldots, x_n = \alpha_n)$$

for m randomly chosen non-zero elements $\alpha_{n-m+1}, \ldots, \alpha_n \in \mathbb{Z}_p$. Let T_g be the expected number of terms of g and T_{g_k} be the number of monomials in g that are homogeneous of degree k in the variables $x_1, \ldots x_{n-m}$. Also let $s = \binom{n+d}{n}$, $s_k = \binom{n-m-1+k}{n-m-1}$, $\gamma = \binom{n-m+d}{n-m}$ and $d_k = \binom{d-k+m}{m}$. Then

$$E[T_g] \leq \sum_{k=0}^{d} s_k \left(1 - \frac{\binom{s-d_k}{T_f}}{\binom{s}{T_f}}\right). \tag{6.7}$$

Let the density of f, $t_f = T_f/s$. Eqn. (6.7) implies that if $\frac{d_k}{s} \to 0$, then with probability $\geq 1 - \frac{dT_f}{2(p-1)}$ one has

$$E[T_g] \approx \sum_{k=0}^{d} s_k \left(1 - (1 - t_f)^{d_k}\right). \tag{6.8}$$

Eqn. (6.8) implies that with probability $\geq 1 - \frac{dT_f}{2(p-1)}$

$$E[t_g] \approx 1 - \gamma^{-1} \sum_{k=0}^{d} s_k (1 - t_f)^{d_k} \tag{6.9}$$

$$\text{and } \mathrm{Var}[T_{g_k}] \approx s_k \left((1 - t_f)^{d_k} - (1 - t_f)^{2d_k}\right). \tag{6.10}$$

6.2.1 On Zippel's assumption

The sparse interpolation idea and the first gcd algorithm to use sparse interpolation was introduced and analyzed by Zippel in his research paper [Zip79]. The goal polynomial (the gcd) is $P(x_1, \ldots, x_n)$ and the starting point is $\vec{a} = (a_1, \ldots, a_n)$. According to our notation, $T_{f_{n-i}}$ denotes the number of terms of the polynomial $P(x_1, \ldots, x_i, a_{i+1}, \ldots, a_n)$. In subsection 3.2 of [Zip79] after computing a sum which depends on the $T_{f_{n-i}}$'s, Zippel claims "We need to make some assumptions about the structure of $T_{f_{n-i}}$ to get anything meaningful out of this. We will assume that the ratio of the terms $T_{f_{n-i}}/T_{f_{n-i+1}}$ is a constant k." [1]

Our observations in this section show that this assumption is false. To see a more accurate bound on the expected ratio of the subsequent number of terms, let us denote by

[1] In [Zip79] Zippel uses the notation t_i for $T_{f_{n-i}}$. Since we use the symbol t for the density, we use our notation as to not confuse the reader.

$d_k^{(i)} = \binom{d-k+i}{d-k}$, $s^{(i)} = \binom{n-i+d}{n-i}$, $s_k^{(i)} = \binom{n-i+k-1}{n-i-1}$, $r_k^{(i)} = \binom{s-d_k^{(i)}}{T}/\binom{s}{T}$ and $\beta_k^{(i)} = 1 - r_k^{(i)}$. Let

$$f_i = f(x_1, \dots x_{n-i}, x_{n-i+1} = \alpha_{n-i+1}, \dots, x_n = \alpha_n)$$

be the polynomial in $n-i$ variables after i random evaluations. According to Lemma 42 $E[T_{f_i}] = \sum_{k=0}^{d} s_k^{(i)} \beta_k^{(i)}$. Our aim is to find an upper and lower bound for

$$\frac{E[T_{f_i}]}{E[T_{f_{i+1}}]} = \frac{\sum_{k=0}^{d} s_k^{(i)} \beta_k^{(i)}}{\sum_{k=0}^{d} s_k^{(i+1)} \beta_k^{(i+1)}}.$$

Observe that

$$d_k^{(i+1)} > d_k^{(i)} \Rightarrow s - d_k^{(i+1)} < s - d_k^{(i)} \Rightarrow r_k^{(i+1)} < r_k^{(i)} \Rightarrow \beta_k^{(i+1)} > \beta_k^{(i)}.$$

Then

$$\frac{E[T_{f_i}]}{E[T_{f_{i+1}}]} = \frac{\sum_{k=0}^{d} s_k^{(i)} \beta_k^{(i)}}{\sum_{k=0}^{d} s_k^{(i+1)} \beta_k^{(i+1)}} < \frac{\sum_{k=0}^{d} s_k^{(i)} \beta_k^{(i)}}{\sum_{k=0}^{d} s_k^{(i+1)} \beta_k^{(i)}} \leq \max_k \left\{ \frac{s_k^{(i)}}{s_k^{(i+1)}} \right\}.$$

We have

$$\frac{s_k^{(i)}}{s_k^{(i+1)}} = \frac{n-i+k-1}{n-i-1} = 1 + \frac{k}{n-i-1} \leq 1 + \frac{d}{n-i-1}.$$

Our next aim is to show that $E[t_{f_{i+1}}] \geq E[t_{f_i}]$: We have

$$E[t_{f_i}] = 1 - \sum_{k=0}^{d} \frac{s_k^{(i)}}{s^{(i)}} r_k^{(i)} = 1 - \sum_{k=0}^{d} \frac{\binom{n-i+k-1}{n-i-1}}{\binom{n-i+d}{n-i}} \frac{\binom{s-d_k^{(i)}}{T}}{\binom{s}{T}} = 1 - \sum_{k=0}^{d} \frac{\binom{n-i+k-1}{n-i-1}}{\binom{s}{T}} \frac{\binom{s-d_k^{(i)}}{T}}{\binom{n-i+d}{n-i}}$$

Let $v_k^{(i)} = \binom{n-i+k-1}{n-i-1}/\binom{s}{T}$ and $w_k^{(i)} = \binom{s-d_k^{(i)}}{T}/\binom{n-i+d}{n-i}$ then $v_k^{(i+1)}/v_k^{(i)} < 1$ and $w_k^{(i+1)}/w_k^{(i)} < 1$. So $\sum_{k=0}^{d} v_k^{(i)} w_k^{(i)}$ decreases and hence $E[t_{f_i}]$ increases as i increases, i.e. we expect an increase in the density ratio after each evaluation. On the other hand

$$E[T_{f_i}] = E[t_{f_i}]\binom{n-i+d}{n-i} = E[t_{f_i}]\tfrac{n-i+d}{n-i}\binom{n-(i+1)+d}{n-(i+1)} \geq E[t_{f_i}]\tfrac{n-i+d}{n-i} E[T_{f_{i+1}}].$$

It follows that

$$\tfrac{E[T_{f_i}]}{E[T_{f_{i+1}}]} \geq E[t_{f_i}] \left(1 + \tfrac{d}{n-i}\right).$$

Hence we have

$$E[t_{f_i}] \left(1 + \tfrac{d}{n-i}\right) \leq \tfrac{E[T_{f_i}]}{E[T_{f_{i+1}}]} \leq 1 + \tfrac{d}{n-i-1}.$$

Now, since each of $E[t_{f_i}], 1 + \frac{d}{n-i}, 1 + \frac{d}{n-i-1}$ increases as i increases we expect an increase in the ratio $E[T_{f_i}]/E[T_{f_{i+1}}]$.

This means we expect that the ratio $T_{f_{n-i}}/T_{f_{n-i+1}}$ increases as i decreases, i.e. after each evaluation we expect an increase in the ratio of subsequent number of terms. See Example 43 for a sparse case and Example 44 for a (relatively) dense case.

Example 43. (**Sparse case with $\mathbf{t_f = 0.00004}$**). Table 6.3 below shows the result of a random experiment where $p = 2^{31} - 1$, $n = 12$, $d = 20$, $T_f = 10^4$, $t_f = 4 \cdot 10^{-5}$ and $f_i := f_{i-1}(x_{n-i+1} = \alpha_{n-i+1})$ with $f_0 = f$, for randomly chosen non-zero α_i's in $\mathbb{Z}_p$. Observe that when we evaluate the first 4 variables the number of terms didn't drop significantly.

i	T_{f_i}	$E[T_{f_i}]$	t_{f_i}	$t_{f_i}(1+\frac{d}{n-i})$	$T_{f_i}/T_{f_{i+1}}$	$1+\frac{d}{n-i-1}$
0	10000	10000	0.00004	0.00012	1.0000	2.8182
1	10000	9999.32	0.00012	0.00033	1.0006	3.0000
2	9994	9995.19	0.00033	0.00100	1.0025	3.2222
3	9969	9971.08	0.00100	0.00321	1.0135	3.5000
4	9836	9837.31	0.00316	0.01108	1.0686	3.8571
5	9205	9200.70	0.01037	0.03998	1.2767	4.3333
6	7210	7219.37	0.03132	0.13570	1.7470	5.0000
7	4127	4144.61	0.09710	0.48548	2.4435	6.0000
8	1689	1690.52	0.23090	1.38540	3.3512	7.6667
9	504	497.93	0.37895	2.90530	4.8932	11.000
10	103	104.50	0.54210	5.96310	7.3571	21.000

Table 6.3: The ratio of subsequent number of terms for a sparse case.

Example 44. (**Dense case with $\mathbf{t_f = 0.1}$**). Table 6.4 below shows the result of a random experiment where $p = 2^{31}-1$, $n = 7$, $d = 13$, $T_f = 7752$, $t_f = 0.1$ where $f_i := f_{i-1}(x_{n-i+1} = \alpha_{n-i+1})$ with $f_0 = f$, for randomly chosen non-zero α_i's in $\mathbb{Z}_p$. Observe that the number of terms dropped by 10% after the first evaluation.

i	T_{f_i}	$E[T_{f_i}]$	t_{f_i}	$t_{f_i}(1+\frac{d}{n-i})$	$T_{f_i}/T_{f_{i+1}}$	$1+\frac{d}{n-i-1}$
0	7752	7752	0.10	0.28	1.17	3.16
1	6670	6643.44	0.24	0.77	1.76	3.60
2	3774	3800.14	0.44	1.58	2.70	4.25
3	1398	1409.83	0.58	2.49	3.65	5.33
4	383	394.03	0.68	3.64	4.78	7.50
5	80	81.14	0.76	5.71	6.66	14

Table 6.4: The ratio of subsequent number of terms for a dense case.

The dominating term of the complexity analysis of Zippel is $\sum_{i=1}^{\nu} c_1 dT_{f_{n-i+1}}^3$ where c_1 is a constant. He assumes $T_{f_{n-i}}/T_{f_{n-i+1}} = k \Rightarrow T_{f_{n-i}} = T_{f_n} k^i$. It follows that

$$\sum_{i=1}^{\nu} c_1 dT_{f_{n-i+1}}^3 = c_1 d \sum_{i=1}^{\nu} T_{f_{n-i+1}}^3 = c_1 dT_{f_n}^3 \sum_{i=1}^{\nu} k^{3i} = c_1 dk^3 \frac{(k^{3\nu}-1)T_{f_n}^3}{k^3-1}.$$

Since $T_f = T_{f_0} = T_{f_n} k^n$, if k is large in comparison with 1 then this sum approaches $c_1 dT_f^3$ and if k is very close to 1 then this sum approaches to $c_1 ndT_f^3$. Then he concludes that if $T_f \gg d$ or n, the dominant behaviour is $\mathcal{O}\left(T_f^3\right)$.

The case where k is very close to 1 (where Zippel has a very sparse case in mind) is the worst case analysis. One can construct such an example. More precisely, Zippel's analysis shows that the complexity for the worst case is $\mathcal{O}\left(ndT_f^3\right)$ and for the average case (where he assumes that "k is large in comparison with 1") the complexity is $\mathcal{O}\left(dT_f^3\right)$. Below we show that this is not true and prove that even for the average case the expected complexity is $\mathcal{O}\left(ndT_f^3\right)$ for the sparse gcd computation.

Example 43 (the sparse case) shows that after 5 evaluations the number of terms is still 90% of T_f. See column T_f. This means the cost of each interpolation of the last few variables in Zippel's algorithm is the same and equal to $\mathcal{O}\left(dT_f^3\right)$. However in Example 44 (the relatively dense case) the number of terms starts to drop right away. Proposition 45 below qualifies this behaviour and makes the observation of Example 43 more precise.

Proposition 45. *Following the notation above, let* $\mathcal{I} = \left\{ i \in \mathbb{N} \mid t_f \le 1/\binom{i+d}{d} \text{ and } i \le n \right\}$. *Then with probability* $\ge 1 - \frac{dT_f}{2(p-1)}$ *one has*

(i) $\frac{E[T_{f_i}]}{T_f} \ge 1 - t_f^2$ for $i \in \mathcal{I}$.

(ii) If $n \le d$ *then* $|\mathcal{I}| \ge \max\{1, \lceil n - \log_r T_f \rceil\}$ *where* $r = 1 + \frac{d}{n}$.

Proof. To save some space we will use the notation $T = T_f$. With probability $\ge 1 - \frac{dT}{2(p-1)}$ one has

$$E[T_{f_i}] = \sum_{k=0}^{d} s_k^{(i)} \left(1 - \frac{\binom{s-d_k^{(i)}}{s}}{\binom{s}{T}}\right)$$

where $d_k^{(i)} = \binom{d-k+i}{i}$. Note that

$$t_f \le \frac{1}{\binom{i+d}{d}} \le \frac{1}{d_k^{(i)}} \Rightarrow t_f d_k^{(i)} \le 1 \tag{6.11}$$

It follows that

$$\frac{d_k^{(i)}}{s} = \frac{d_k^{(i)} t_f}{T} \le \frac{1}{T}. \tag{6.12}$$

We have

$$\frac{\binom{s-d_k^{(i)}}{s}}{\binom{s}{T}} \le (1-t_f)^{d_k^{(i)}}. \tag{6.13}$$

Note that we can use the approximation (6.3) because $d_k^{(i)}/s$ is small. But we don't need to use it. Now

$$\begin{aligned}
E[T_{f_i}] &= \sum_{k=0}^{d} s_k^{(i)}\left(1-\frac{\binom{s-d_k^{(i)}}{s}}{\binom{s}{T}}\right) \overbrace{\geq}^{\text{by (6.13)}} \sum_{k=0}^{d} s_k^{(i)}\left(1-(1-t_f)^{d_k^{(i)}}\right) \\
&= \sum_{k=0}^{d} s_k^{(i)}\left(1-\sum_{j=0}^{d_k^{(i)}}(-1)^j\binom{d_k^{(i)}}{j}t_f^j\right) \\
&= \sum_{k=0}^{d} s_k^{(i)}\left(1-1+d_k^{(i)}t_f-\sum_{j=2}^{d_k^{(i)}}(-1)^j\binom{d_k^{(i)}}{j}t_f^j\right) \\
&= \sum_{k=0}^{d} s_k^{(i)}d_k^{(i)}t_f-\sum_{k=0}^{d}\sum_{j=2}^{d_k^{(i)}}(-1)^j s_k^{(i)}\binom{d_k^{(i)}}{j}t_f^j \\
&= t_f s - t_f^2\sum_{k=0}^{d}\sum_{j=2}^{d_k^{(i)}}(-1)^j s_k^{(i)}\binom{d_k^{(i)}}{j}t_f^{j-2} \\
&= T - t_f^2\sum_{k=0}^{d}\sum_{j=2}^{d_k^{(i)}}(-1)^j s_k^{(i)}\binom{d_k^{(i)}}{j}t_f^{j-2}
\end{aligned}$$

The expected decrease in the number of terms is determined by the quantity

$$A = t_f^2\sum_{k=0}^{d}\sum_{j=2}^{d_k^{(i)}}(-1)^j s_k^{(i)}\binom{d_k^{(i)}}{j}t_f^{j-2}$$

We have

$$s_k^{(i)}\binom{d_k^{(i)}}{j}t_f^{j-2} < s_k^{(i)}\frac{\left(d_k^{(i)}\right)^j}{j!}t_f^{j-2} \overbrace{\leq}^{\text{by (6.11)}} \frac{s_k^{(i)}\left(d_k^{(i)}\right)^2}{j!}.$$

Then the absolute value $|A|$ of A is

$$|A| \leq t_f^2\sum_{k=0}^{d}\sum_{j=2}^{d_k^{(i)}}\frac{s_k^{(i)}\left(d_k^{(i)}\right)^2}{j!} = t_f^2\sum_{k=0}^{d}s_k^{(i)}\left(d_k^{(i)}\right)^2\underbrace{\sum_{j=2}^{d_k^{(i)}}\frac{1}{j!}}_{\leq 1} \leq t_f^2\sum_{k=0}^{d}s_k^{(i)}\left(d_k^{(i)}\right)^2$$

So we see that

$$\frac{|A|}{T} \leq \frac{t_f^2\sum_{k=0}^{d}s_k^{(i)}\left(d_k^{(i)}\right)^2}{t_f s} = t_f\sum_{k=0}^{d}s_k^{(i)}d_k^{(i)}\frac{d_k^{(i)}}{s} \overbrace{\leq}^{\text{by (6.12)}} t_f\frac{1}{T}s = t_f^2 \tag{6.14}$$

Then

$$\frac{E[T_{f_i}]}{T} \geq \frac{T-A}{T} \geq \frac{T-|A|}{T} = 1 - \frac{|A|}{T} \overbrace{\geq}^{\text{by(6.14)}} 1 - t_f^2.$$

This proves the first part. For the second part, note that

$$t_f \leq \frac{1}{\binom{i+d}{d}} \Leftrightarrow T \leq \frac{\binom{n+d}{d}}{\binom{i+d}{d}} \Leftrightarrow \frac{1}{T} \geq \frac{\binom{i+d}{d}}{\binom{n+d}{d}}$$

and that $\frac{n-i}{n+d} = \frac{n}{n+d} - \frac{i}{n+d} = \frac{1}{1+d/n} - \frac{i}{n+d}$. So if $n \leq d$ then $\frac{n-i}{n+d}$ is small and the following bound is useful

$$\frac{\binom{i+d}{d}}{\binom{n+d}{d}} = \frac{\binom{n+d-(n-i)}{d}}{\binom{n+d}{d}} \leq \left(1 - \frac{d}{n+d}\right)^{n-i}.$$

Now if

$$T \leq \left(1 - \frac{d}{n+d}\right)^{i-n} \Leftrightarrow \frac{1}{T} \geq \left(1 - \frac{d}{n+d}\right)^{n-i} \Rightarrow i \in \mathcal{I}$$

and if we define $w = \frac{n}{n+d}$ then $w^n T \leq w^i \Rightarrow i \geq n + \log_w T$. Finally if $r = \frac{1}{w} = 1 + \frac{d}{n}$ we have $i \geq n - \log_r T$. Hence if $n \leq d$ then $|\mathcal{I}| \geq \lceil n - \log_r T \rceil$.

For Example 43, $\max\{1, \lceil n - \log_r T_f \rceil\} = \max\{1, \lceil 12 - \log_{2.6} 10^4 \rceil\} = \max\{1, 3\} = 3$ whereas $|\mathcal{I}| = 5$ and for Example 44, $|\mathcal{I}| = 1$ whereas $\max\{1, \lceil n - \log_r T_f \rceil\} = \max\{1, \lceil 7 - \log_{2.86} 7752 \rceil\} = \max\{1, -1\} = 1$. □

Corollary 46. *Following the notation above, the average complexity of the sparse gcd algorithm of Zippel is* $\Omega(|I|dT^3)$. *If* $n \leq d$ *then the average complexity is in* $\Omega(\max\{1, \lceil n - \log_r T \rceil\} dT^3)$ *where* $r = 1 + \frac{d}{n}$.

Remark 47. According to Proposition 45 (i), when t_f is small, on average we expect that $T_{f_i} \approx T_f$ for $i \in \mathcal{I}$, i.e. for at least for $|\mathcal{I}|$ many steps we don't expect a significant decrease in the number of terms. Note that as t_f and hence T_f gets smaller, i.e. the inputs gets sparser, $|\mathcal{I}|$ gets closer to n. Also, in practice $n \leq d$ and as T_f gets smaller $\log_r T_f$ gets smaller and according to Proposition 45 (ii) $|\mathcal{I}| \geq \lceil n - \log_r T_f \rceil$ gets closer to n. This means for the sparse case the expected average complexity is in $\mathcal{O}(ndT_f^3)$. Thus, the common perception that in the sparse interpolation most of the work is done at the last step is not true: For most sparse examples, the work done at the last few steps is the same.

6.3 The expected number of terms of Taylor coefficients

Consider the Taylor series expansion of $f_j, g_j, e_j \in \mathbb{Z}_p[x_1, \ldots, x_n]$ about $x_n = \alpha_n$ for $0 \neq \alpha_n \in \mathbb{Z}_p$. Let $f_j = \sum_{i=0}^{d_n} f_{ji}(x_n - \alpha_n)^i, g_j = \sum_{i=0}^{d_n} g_{ji}(x_n - \alpha_n)^i$ and $e_j = \sum_{i=0}^{d_n} e_{ji}(x_n - \alpha_n)^i$ where $f_{ji}, g_{ji}, e_{ji} \in \mathbb{Z}_p[x_1, \ldots, x_{n-1}]$. In the i^{th} iteration of the for loop of the j^{th} step of

MHL (Algorithm 5) one solves the MDP $f_{ji}g_{j0} + g_{ji}f_{j0} = e_{ji}$ to compute f_{ji} and g_{ji} for $1 \leq i \leq d_n$. The cost of MDP depends on the sizes of the polynomials f_{ji}, g_{ji} and e_{ji}.

To make our complexity analysis as precise as possible, in this section we will compute theoretical estimations for the expected sizes of the Taylor coefficients f_j and the upper bounds for $E[T_{f_j}]$ of a randomly chosen $f \in \mathbb{Z}_p[x_1, \ldots, x_n]$ where $f = \sum_{i=0}^{d_n} f_j(x_n - \alpha_n)^j$ for a randomly chosen non-zero element $\alpha_n \in \mathbb{Z}_p$ and p is a big prime. We will confirm our theoretical estimations by experimental data.

We will use the notation $\#f$ and T_f interchangeably. For a random non-zero $\alpha_n \in \mathbb{Z}_p$, consider $f = \sum_{j=0}^{d_n} f_j(x_n - \alpha_n)^j$ where each $f_j \in \mathbb{Z}_p[x_1, \ldots, x_{n-1}]$. We expect $T_{f_{j+1}} \leq T_{f_j}$, that is, the size of the Taylor coefficients f_j decrease as j increases.

As a first step to find a upper bound on $E[T_{f_j}]$, we have the following Lemma.

Lemma 48. *Let $0 < d < p$ and $n > 0$. Then following the notation above, the probability of occurrence of each monomial in the support of $f' = \frac{\partial}{\partial x_n} f$ is the same.*

Let $\mathbf{m}' = cx_1^{\beta_1} \cdots x_n^{\beta_n} \in \mathrm{Supp}(f')$ where $\beta_1 + \cdots + \beta_n \leq d-1$. Then any monomial of the form $\mathbf{m} = (\beta_n + 1)^{-1} cx_1^{\beta_1} \cdots x_n^{\beta_n + 1} + \mathbf{n}$ where $\mathbf{n}$ is a monomial of degree $\leq d$ which does not contain the variable x_n lies over $\mathbf{m}'$. We have $\beta_1 + \cdots + (\beta_n + 1) \leq d$. On the other hand for $\mathbf{m} = cx_1^{\beta_1} \cdots x_n^{\beta_n} \in \mathrm{Supp}(f)$ one has $\frac{\partial}{\partial x_n}\mathbf{m} = c\beta_n x_1^{\beta_1} \cdots x_n^{\beta_n - 1} = 0 \iff p \,|\, c\beta_n \iff p \,|\, \beta_n$. So if $d < p$ we have $\frac{\partial}{\partial x_n}\mathbf{m} = 0 \iff \mathbf{m}$ does not contain the variable x_n, i.e. $\beta_n = 0$. Therefore the number of monomials lying over each distinct monomial in the support of f' is the same and equal to $(p-1)^\gamma + 1$ where $\gamma = \binom{n+d-1}{n}$. Since f is random, this implies that the probability of each monomial in the support of f' is the same.

After differentiation monomials which do not contain the variable x_n in f will disappear. Since the expected number of them is $= t_f \binom{n-1+d}{n-1}$, we expect

$$E[\#f'] = T_f - t_f \binom{n-1+d}{n-1}.$$

We must also compute the density ratio $t_{f'}$ of f'.

Lemma 49. *Following the notation above*

$$\begin{aligned}
E[t_{f'}] &= \left(T_f - t_f \tbinom{n-1+d}{n-1}\right) / \tbinom{n+d-1}{n} \\
&= t_f \left(\tbinom{n+d}{n} - \tbinom{n-1+d}{n-1}\right) / \tbinom{n+d-1}{n} \\
&= t_f \left(\tfrac{n+d}{n}\tbinom{n+d-1}{n-1} - \tbinom{n-1+d}{n-1}\right) / \tbinom{n+d-1}{n} \\
&= t_f \tbinom{n+d-1}{n-1} / \tfrac{n}{d}\tbinom{n+d-1}{d-1} = t_f \tbinom{n+d-1}{n-1} / \tbinom{n+d-1}{n-1} = t_f.
\end{aligned}$$

For simplicity let us assume that $p > j$, otherwise we need to introduce Hasse derivatives but the idea will be the same. We have $f_j = \frac{1}{j!}\frac{\partial}{\partial x_n^j} f(x_n = \alpha)$. Also $f^{(j)} := \frac{\partial}{\partial x_n^j} f$ is of degree

$\leq d_j := d - j$ .Using Lemma 48 and 49 repeatedly

$$E[f_j] \leq E[\#f^{(j)}] = E[t_{f^{(j)}}]\binom{n+d-j}{n} = t_f\binom{n+d-j}{n}.$$

It follows that

$$E[t_{f_j}] = E[f_j]/\binom{n+d-j}{n} \leq t_f.$$

We sum up the observations of this section in such a way that it will be helpful for the next sections.

Lemma 50. *Let p be a big prime and $f_j \in \mathbb{Z}_p[x_1, \ldots, x_j]$ be a multivariate polynomial of total degree $\leq d_j$ that has T_{f_j} non-zero terms. For a randomly chosen non-zero element $\alpha_j \in \mathbb{Z}_p$ consider $f_j = \sum_{i=0}^{d_j} f_{ji}(x_j - \alpha_j)^i$ where $f_{ji} \in \mathbb{Z}_p[x_1, \ldots, x_{j-1}]$. Let $s = \binom{j+d_j}{j}$ and $t_{f_j} = T_{f_j}/s$. Then*

$$E[T_{f_{ji}}] \lesssim t_{f_j}\binom{j+d_j-i}{j} \text{ and } E[t_{f_{ji}}] \lesssim t_{f_j}\frac{j+d_j-i}{j} \tag{6.15}$$

Example 51. Table 6.5 below shows the result of a random experiment where $p = 2^{31} - 1$, $j = 7$, $d_j = 13$, $T_{f_j} = 7752$. In the Table 6.5 $T_{f_{ji}}, t_{f_{ji}}$ and $t_{f_j^{(i)}}$ are the actual values. Also the expected number of terms $E[T_{f_{ji}}]$ of f_{ji}, the bound $bT_{f_{ji}}$ on the expected number of terms of f_{ji}, and the bound $bt_{f_{ji}}$ on the density ratio of f_{ji}, are based on (6.4) and (6.15) resp.

i	$T_{f_{ji}}$	$E[T_{f_{ji}}]$	$bT_{f_{ji}}$	$t_{f_j^{(i)}}$	$t_{f_{ji}}$	$bt_{f_{ji}}$
0	6651	6643.345	7752	0.09	0.085	0.285
1	4343	4366.828	5038.8	0.1	0.086	0.271
2	2773	2789.364	3182.4	0.09	0.088	0.257
3	1722	1724.183	1944.8	0.10	0.088	0.242
4	977	1025.981	1144.0	0.10	0.092	0.228
5	564	583.867	643.5	0.10	0.093	0.214
6	306	315.075	343.2	0.10	0.094	0.200
7	150	159.417	171.6	0.10	0.103	0.185
8	68	74.463	79.2	0.10	0.104	0.171
9	26	31.403	33.0	0.10	0.127	0.157
10	12	11.559	12.0	0.05	0.125	0.142
11	3	3.511	3.6	0.12	0.111	0.128
12	1	0.790	0.8	1	0.112	0.114

Table 6.5: The bounds on the expected number of terms and the density ratio.

6.4 The complexity of the MDP

Let p be a big prime and $u, w, h \in \mathbb{Z}_p[x_1, \ldots, x_n]$ where u, w are monic in x_1. Suppose we are trying to solve the MDP (which satisfies the MDP conditions)

$$D: \ fu + gw = h \tag{6.16}$$

to find the unique solution pair (f, g) via sparse interpolation as described in Section 3.6. Let d be a total degree bound for f, g, u, w, h. Our aim in this section is to estimate the expected complexity of solving D. Since the calculations of the complexity evaluation in the next section is somewhat tedious, this section is intended to help the reader to follow it easily.

Suppose the (probable) solution-form σ_f of f is true and

$$\sigma_f = \sum_{i+j \le d} c_{ij}(x_3, ..., x_n) x_1^i x_2^j \text{ where } c_{ij} = \sum_{l=1}^{m_{ij}} c_{ijl} x_3^{\gamma_{3l}} \cdots x_n^{\gamma_{jl}} \text{ with } c_{ijl} \in \mathbb{Z}_p \backslash \{0\}.$$

Let $m = \max m_{ij}$. Then in sparse interpolation the first step is to choose a random $(\beta_3, \ldots, \beta_n) \in (\mathbb{Z}_p \backslash \{0\})^{n-2}$ and solve bivariate MDP's

$$D_r: \ \tilde{f} \cdot u(x_1, x_2, \beta_3^r, \ldots, \beta_n^r) + \tilde{g} \cdot w(x_1, x_2, \beta_3^r, \ldots, \beta_n^r) = h(x_1, x_2, \beta_3^r, \ldots, \beta_n^r)$$

for $r = 1, \ldots, m$ where $(\tilde{f}, \tilde{g}) \in \mathbb{Z}_p[x_1, x_2]^2$ is to be solved.

As before let $s = \binom{n+d}{n}, r = \frac{n}{n+d}$ and $s_k = \binom{n-2+k}{n-2}$, $d_k = d - k + 1$ for $0 \le k \le d$. Suppose that the solution form σ_f of f is true. Then the expected number of monomials of the form $x_3^{\alpha_3} \cdots x_n^{\alpha_n} x_1^i x_2^j$ in the Supp(f) is $t_f \binom{n-2+d-k}{n-2} = t_f s_{d-k}$ when $i + j = k$. So we expect $\#c_{ij} = t_f s_{d-k}$. When $i = j = 0$, i.e. the number of monomials that are in the variables $x_3, \ldots, x_n$ in the Supp(f) expected to be greatest, the expected number of evaluations is $m = t_f s_d$. At this point we remark that

$$t_f s_d = T_f \frac{n(n-1)}{(n+d)(n+d-1)} \le T_f \left(\frac{n}{n+d}\right)^2.$$

So, if $T_f \left(\frac{n}{n+d}\right)^2 < 1$, our theoretical expectation of $m = t_f s_d$ will be less than 1 which in practice means that there won't be any evaluation. If d is big and T_f is small this inequality may occur, however the algorithm makes at least one evaluation and hence calls BDP at least once. So we should have $m = \lceil t_f s_d \rceil$.

Let $T = (T_f + T_u + T_w + T_h)$. (We are evaluating σ_f too, to get the linear system of equations in c_{ijl}). Then according to Subsection 3.6.2 the total cost C_E of the consecutive

evaluations is bounded above by

$$\begin{aligned} C_E &\leq \#\text{of terms} \times (\#\text{of evaluations} + n - 3) + (n-2)d \\ &\approx (T_f + T_u + T_w + T_h)\,(T_f \frac{s_d}{s} + 1 + n - 3) + (n-2)d \\ &\leq T(\lceil T_f r^2 \rceil + n) + nd \end{aligned}$$

If d is huge and T_f is small so that $T_f r^2 < n$ then nT or nd can dominate the sum above. So we obtain

$$C_E \in \mathcal{O}(T\lceil T_f r^2 \rceil + nT + nd) \tag{6.17}$$

After evaluation the sparse interpolation routine calls BDP to solve the bivariate diophantine equations D_r. For a given D_r, BDP solves it in $\mathcal{O}(d_2^3)$ arithmetic operations in $\mathbb{Z}_p$ via dense interpolation where d_2 is a bound for the total degrees of f, g, u, w, h in x_1, x_2 above. In our case $d_2 \leq d$. Hence the expected cost C_B of solving D_r's is

$$C_B \in \#\text{of evaluations} \times \mathcal{O}(d^3) = \mathcal{O}(\lceil T_f r^2 \rceil d^3) \tag{6.18}$$

BDP gives the unique solution for D_r iff the MDP conditions for D_r are satisfied. To have a unique solution for D_r, for $1 \leq t \leq m$, the first condition is

$$\gcd\left(u(x_1, x_2, \beta_3^t, \ldots, \beta_n^t), w(x_1, x_2, \beta_3^t, \ldots, \beta_n^t)\right) \mid h(x_1, x_2, \beta_3^t, \ldots, \beta_n^t)$$

which is the case if D has a solution. The second condition is, when BDP chooses a random $\gamma \in \mathbb{Z}_p$ while it is interpolating, it must be the case that

$$\gcd\left(u(x_1, \gamma, \beta_3^t, \ldots, \beta_n^t), w(x_1, \gamma, \beta_3^t, \ldots, \beta_n^t)\right) = 1 \text{ in } \mathbb{Z}_p[x_1].$$

The probability that the second condition fails is $\leq m \deg(u) \deg(w)/p \leq md^2/p$ [MT16-2]. This is a worst case upper bound. On average the expected number of failures is only m out of p trials [MT16-1]. In this case then the expected probability of failure is $\leq \lceil T_f r^2 \rceil / p$.

As seen in Section 6.2, we expect that the density ratio increases after each evaluation of f. Hence after evaluations we expect dense polynomials over $\mathbb{Z}_p[x_1, x_2]$ and this is why BDP uses the dense interpolation to solve bivariate MDP's. While solving the bivariate MDP's, BDP chooses a random $\gamma \in \mathbb{Z}_p$ and solves the univariate MDP over $\mathbb{Z}_p$. This is done by using the Euclidean algorithm (see [GCL92]). To do that it computes the univariate gcd and if it is not equal to 1 it detects it. So if such an unlucky evaluation occurs then BDP detects it and the algorithm terminates or it can be modified in such a way that it chooses another random γ.

If $i + j = k$ then to recover c_{ijl}'s one needs to solve a linear system which corresponds to a Vandermonde matrix of expected size $t_f s_{d-k}$. The cost of this operation is

$\mathcal{O}\left(t_f^2 s_{d-k}^2\right)$[Zip90]. We have $(k+1)$ monomials of the form $x_1^i x_2^j$ with $i+j=k$. So, the expected total cost C_V for the solution is in

$$C_V \in \mathcal{O}\left(\sum_{k=0}^{d}(k+1)t_f^2 s_{d-k}^2\right) = \mathcal{O}\left(T_f^2 \sum_{k=0}^{d}(k+1)\left(\frac{s_{d-k}}{s}\right)^2\right)$$

First note that

$$s = \binom{n+d}{n} = \frac{(n+d)(n+d-1)}{n(n-1)}\binom{n+d-2}{n-2} > r^{-2}\binom{n+d-2}{n-2} = r^{-2}s_{d-2}$$

Then, as we did in the first section we obtain

$$\frac{s_{d-k}}{s} < r^2\frac{s_{d-k}}{s_{d-2}} < r^2\left(1-\tfrac{n-2}{n+d-2}\right)^k = r^2(1-\theta)^k$$

where $\theta := \frac{n-2}{n+d-2}$. Then we get

$$T_f^2\sum_{k=0}^{d}(k+1)\left(\frac{s_{d-k}}{s}\right)^2 < T_f^2 r^4\sum_{k=0}^{d}(k+1)(1-\theta)^{2k}$$

By using the summation formula, a straightforward (but a bit tedious) calculation shows that if $n>2$ (which is in fact the case)

$$r^4\sum_{k=0}^{d}(k+1)(1-\theta)^{2k} \le r^4\frac{(n+d-2)^4}{(n-2)^2(n+2d-2)^2} < r^2\left(\frac{n}{n-2}\right)^2 \le 9r^2$$

Hence we see that the expected cost of solving linear systems is in ($r=\frac{n}{n+d}$)

$$C_V \in \mathcal{O}(T_f^2 r^2) \tag{6.19}$$

After computing f, the next step is the multivariate division $(h-fu)/w$ to get g. The expected cost C_M of the sparse multiplication and the sparse multivariate division C_D are

$$C_M \in \mathcal{O}\left(T_f T_u\right) \text{ and } C_D \in \mathcal{O}\left(T_w T_g\right) \tag{6.20}$$

arithmetic operations in $\mathbb{Z}_p$ ignoring the sorting cost.

Combining the equations (6.17), (6.18), (6.19) and (6.20) above wee see that the expected cost of solving the MDP is in

$$\underbrace{\mathcal{O}(\underbrace{T\lceil T_f r^2\rceil + nT + nd}_{C_E} + \underbrace{\lceil T_f r^2\rceil d^3}_{C_B} + \underbrace{T_f^2 r^2}_{C_V}}_{\text{to recover } f} + \underbrace{\underbrace{T_f T_u}_{C_M} + \underbrace{T_w T_g}_{C_D}}_{\text{to recover } g})$$

with the failure probability $\leq \lceil T_f r^2 \rceil d^3/p$ where $r = \frac{n}{n+d}$ and $T = (T_f + T_u + T_w + T_h)$.

Finally suppose that the guessed solution-form σ_f of f is wrong. Then the solution to f that the sparse interpolation routine computes will be wrong. Since the solution to the MDP is unique as long as the MDP conditions are satisfied then we will have $w \nmid h - fu$. So, in the sparse interpolation a possible failure, i.e. a possible false assumption is detected. In this case the cost of sparse division may increase. (We don't consider this.)

Theorem 52. *Let p be a big prime and $u, w, h \in \mathbb{Z}_p[x_1, \ldots, x_n]$ where u, w are monic in x_1. If the solution-form σ_f is true, then the cost of solving the MDP $fu + gw = h$ (which satisfies the MDP conditions) to find the unique solution pair (f, g) via sparse interpolation as described in section 2 is in*

$$\mathcal{O}\left(T\lceil T_f r^2 \rceil + nT + nd + \lceil T_f r^2 \rceil d^3 + T_f^2 r^2 + T_f T_u + T_w T_g\right)$$

where d is a total degree bound for f, g, u, w, h, $r = \frac{n}{n+d}$ and $T = T_f + T_u + T_w + T_h$ with the probability $> 1 - \lceil T_f r^2 \rceil d^3/p$.

6.5 The Complexity of MTSHL

For $j \geq 3$, during the j^{th} step of the MTSHL, one aims to reach the factorization $a_j = f_j g_j \in \mathbb{Z}_p[x_1, \ldots, x_j]$ from the knowledge of $a_j, f_{j-1}, g_{j-1} \in \mathbb{Z}_p[x_1, \ldots, x_{j-1}]$ satisfying $a_{j-1} = f_{j-1} g_{j-1}$. Let $f_j = \sum_{i=0}^{d_j} f_{ji}(x_j - \alpha_j)^i, g_j = \sum_{i=0}^{d_j} g_{ji}(x_j - \alpha_j)^i$ for a randomly chosen non-zero element α_j in $\mathbb{Z}_p$ and where $f_{j0} = f_{j-1}$, $g_{j0} = g_{j-1}$ and $\deg_{x_j}(a_j) = d_j$.

For $1 \leq i \leq d_j$ one recovers each f_{ji}, g_{ji} by solving the MDP problems

$$f_{ji} g_{j0} + g_{ji} f_{j0} = e_{ji} \quad \text{in} \quad \mathbb{Z}_p[x_1, \ldots, x_{j-1}]$$

via sparse interpolation in a for loop where e_{ji} is the i^{th} Taylor coefficient of $error$. Our aim in this section is first to estimate the complexity of this lifting process at the j^{th} step of the MTSHL algorithm, that is, finding f_j and g_j and then to estimate the expected complexity of the multivariate factorization via MTSHL. To this end let t_h, T_h denote the expected density ratio and the expected number of non-zero elements of a polynomial $h \in \mathbb{Z}_p[x_1, \ldots, x_j]$. By $(\#k)$ we will refer to the k^{th} line in the Algorithm 10.

Before continuing, to be able to follow the rest of discussion below easily, we suggest the reader to read the following explanations along with concrete Example 29 in Chapter 4.

6.5.1 Before we go into the details

Before we go into the details of tedious calculations, we want to make a guess what we will obtain, based on our observations from Section 6:

Suppose that the smallest factor is f and $T_f \gg \max\{n, d\}$. Based on Theorem 52 we may guess that the evaluation cost will be the most expensive. Now, at the j^{th} step of SHL, suppose that $T_{f_{j-1}} \leq T_{g_{j-1}}$. Then based on Lemma 42, MTSHL makes a probabilistic guess (#1) that T_{f_j} will be smaller than T_{g_j} and (in the for loop), for $1 \leq i \leq d_j$, in the sparse interpolation routine, it first computes f_{ji} and then recovers g_{ji} via the multivariate division. Since we expect $T_{f_{ji}} \leq T_{f_j}$ the expected number of evaluations for each i in the loop will be $\leq T_{f_j}\left(\frac{j}{j+d_j}\right)^2$(see Section 6.4). So at the j^{th} step the expected evaluation cost will be in $d_j\mathcal{O}(T_{a_j}T_{f_j}\left(\frac{j}{j+d_j}\right)^2) = d_j\mathcal{O}(T_{a_j}T_{f_j}\left(\frac{j}{d_j}\right)^2) = \mathcal{O}(\frac{j^2}{d_j}T_{a_j}T_{f_j})$. Then, running j from 1 to n,we expect that the average complexity will be close to or less than $n\mathcal{O}(\frac{n^2}{d}T_aT_f) = \mathcal{O}(\frac{n^3}{d}T_aT_f)$. Finally since $T_f \leq T_g$ and $T_a \in \mathcal{O}(T_fT_g)$ our guess is that the average complexity will be close to or less than $\mathcal{O}(\frac{n^3}{d}T_g^3)$.

In the following we make this guess more precise and prove that if $T_f \leq T_g$ and $T_g > nd^2$ the expected cost of MTSHL is in fact quadratic in n and $\in \mathcal{O}(\frac{n^2}{d}T_g^3)$.

6.5.2 In detail

Suppose that $T_{f_{j-1}} \leq T_{g_{j-1}}$. Then based on Lemma 42, MTSHL makes a probabilistic guess (#1) that T_{f_j} will be smaller than T_{g_j} and for $1 \leq i \leq d_j$, in the sparse interpolation routine, it first computes f_{ji} and then recovers g_{ji} via the multivariate division $(e_{ji} - g_{j0}f_{ji})/f_{j0}$.

The cost of (#5) is the cost of subtraction as we are given $a_{j-1} = f_{j-1}g_{j-1}$. In the sparse case this cost is for sorting the monomials which we will ignore for the rest of the discussion.

In the i^{th} iteration, updating the monomial (#7) has cost linear in i which is negligible. Then the algorithm computes the i^{th} Taylor coefficient of the error at $x_j = \alpha_j$ (#8).

Maple used to compute this using the formula $c = h(x_j = \alpha_j)/i!$ where h is the i'th derivative of *error* wrt x_j. Instead, Maple now uses the more direct formula $c = \sum_{k=i}^{d} \text{coeff}(error, x^k)\alpha^{s-k}\binom{s}{k}$ where $d = \deg_{x_j} error$ which is three times faster [MP14]. So the cost of this step is a sum of polynomials which can be done in almost linear time in $\#error < \#a_j$

Then to solve the MDP, it comes to use the sparse interpolation (#11).

Suppose that $f_{ji} = \sum c_{jikl}x_1^k x_2^l \in \mathbb{Z}_p[x_3, \ldots, x_{j-1}][x_1, x_2]$. Let $d_{ji} := d_j - i$ and $ev_{ji} := t_{f_{ji}}\binom{j-1-2+d_{ji}}{j-1-2}$. We have $\deg(f_{ji}) \leq d_{ji}$ and as explained in Section 6.4 we expect that $\#c_{ji00} = ev_{ji}$. Then by Lemma 50 we expect

$$ev_{ji} := t_{f_{ji}}\binom{j-3+d_{ji}}{j-3} \quad \leq \quad t_{f_j}\frac{j+d_{ji}}{j}\binom{j-3+d_{ji}}{j-3}.$$

Based on Theorem 28, MTSHL makes a probabilistic guess (#10) and assumes that in sparse interpolation the solution form $\sigma_{f_{ji}} = f_{j,i-1}$. So the expected number of evaluations at the i^{th} step is $\lceil ev_{j,i-1} \rceil$. According to Section 6.4 the expected cost $C_{Ev_{ji}}$ of evaluation

at the i^{th} step is bounded above by

$$C_{Ev_{ji}} < \left(T_{g_{j0}} + T_{f_{j0}} + T_{f_{j,i-1}} + T_{e_{ji}}\right)\left(\lceil ev_{j,i-1}\rceil + j - 4\right) + (j-3)\, d_{j,i}$$

After evaluation the sparse interpolation routine calls BDP. For a given bivariate diophantine equation the cost is $\mathcal{O}\left(d_{ji}^3\right)$. Hence the expected cost $C_{B_{ji}}$ of solving the bivariate diophantine equations via BDP in the i^{th} iteration is

$$C_{B_{ji}} \in \mathcal{O}\left(\lceil ev_{j,i-1}\rceil d_{ji}^3\right)$$

Note again that the sparse interpolation routine first computes f_{ji} and then recovers g_{ji} via the multivariate division. The linear systems to be solved to recover f_{ji} correspond to a Vandermonde matrices and they are constructed by the unknown coefficients of the solution form $\sigma_{f_{ji}}$ of f_{ji}. Hence if we define $r_{ji} = \frac{j-1}{j-1+d_{ji}}$, then the expected cost $C_{V_{ji}}$ of solving the linear system in the i^{th} iteration is (see Section 6.4 Eqn (6.19))

$$C_{V_{ji}} \in \mathcal{O}\left(T_{f_{j,i-1}}^2 r_{j,i-1}^2\right).$$

By Lemma 50 $E[\#g_{j0}] \leq t_{g_j}\binom{j+d_j}{j}$ and $E[\#f_{ji}] \leq t_{f_j}\binom{j+d_{ji}}{j}$. So, after computing f_{ji}, the expected cost $C_{M_{ji}}$ of sparse multiplication $g_{j0}f_{ji}$ and the expected cost $C_{D_{ji}}$ of sparse division $(e_{ji} - g_{j0}f_{ji})/f_{j0}$ is in

$$C_{M_{ji}} \text{ and } C_{D_{ji}} \in \mathcal{O}\left(t_{f_j} t_{g_j}\binom{j+d_j}{j}\binom{j+d_{ji}}{j}\right).$$

So far we have covered the (dominating) costs in sparse interpolation at the i^{th} iteration. Next we consider (#17):
The cost of updating, $C_{U_{ji}}$, i.e computing $f_{ji}(x_j - \alpha_j)^i$ and $g_{ji}(x_j - \alpha_j)^i$ are both in

$$C_{U_{ji}} \in \mathcal{O}\left(i(t_{f_j} + t_{g_j})\binom{j+d_{ji}}{j}\right).$$

Finally (#18) the cost of updating *error* is in

$$C_{Er_{ji}} \in \mathcal{O}\left(T_{f_j} T_{g_j}\right).$$

Let C_{Ev_j} be the expected total evaluation cost (in sparse interpolation) at the j^{th} step. Then $C_{Ev_j} = \sum_{i=1}^{d_j} C_{Ev_{ji}}$. To compute it we'll split the sum

$$\sum_{i=1}^{d_j}\left(T_{g_{j0}} + T_{f_{j0}} + T_{f_{j,i-1}} + T_{e_{ji}}\right)\left(\lceil ev_{j,i-1}\rceil + j - 4\right) + (j-3)\, d_{j,i-1}$$

and consider the parts separately: We first consider the sum

$$\begin{aligned}
\sum_{i=1}^{d_j}\lceil ev_{j,i-1}\rceil &= \sum_{i=0}^{d_j-1}\lceil ev_{ji}\rceil \leq \sum_{i=0}^{d_j}\lceil t_{f_j}\frac{j+d_{ji}}{j}\binom{j-3+d_{ji}}{j-3}\rceil \\
&\leq \sum_{i=0}^{d_j}\left(t_{f_j}\frac{j+d_{ji}}{j}\binom{j-3+d_{ji}}{j-3}+1\right) \\
&= \sum_{i=0}^{d_j}t_{f_j}\frac{j+d_{ji}}{j}\binom{j-3+d_{ji}}{j-3}+\sum_{i=0}^{d_j}1 \\
&\leq t_{f_j}\frac{j+d_j}{j}\sum_{i=0}^{d_j}\binom{j-3+d_{ji}}{j-3}+d_j = t_{f_j}\frac{j+d_j}{j}\binom{j-2+d_j}{j-2}+d_j \\
&= t_{f_j}\frac{j+d_j}{j}\frac{j-1}{j-1+d_j}\binom{j-1+d_j}{j-1}+d_j \leq t_{f_j}\binom{j-1+d_j}{j-1}+d_j \\
&= t_{f_j}\frac{j}{j+d_j}\binom{j+d_j}{j}+d_j = \frac{j}{j+d_j}T_{f_j}+d_j
\end{aligned}$$

As a next step, since we expect $T_{f_j} \leq T_{g_j}$, $T_{f_{j0}} \leq T_{f_j}$ and $T_{g_{j0}} \leq T_{g_j}$

$$\sum_{i=1}^{d_j}\left(T_{g_{j0}}+T_{f_{j0}}\right)\lceil ev_{j,i-1}\rceil \leq (T_{f_j}+T_{g_j})\left(\frac{j}{j+d_j}T_{f_j}+d_j\right) \in \mathcal{O}\left(\frac{j}{j+d_j}T_{f_j}T_{g_j}+d_jT_{g_j}\right) \tag{6.21}$$

On the other hand note that we expect $T_{e_{ji}} \leq T_{a_j}$. Then

$$\sum_{i=1}^{d_j}T_{e_{ji}}\lceil ev_{j,i-1}\rceil \leq T_{a_j}\sum_{i=1}^{d_j}\lceil ev_{j,i-1}\rceil \leq \frac{j}{j+d_j}T_{f_j}T_{a_j}+d_jT_{a_j}$$

So, since we expect $T_{f_{j,i-1}} \leq T_{f_{j0}}$ we see that

$$\sum_{i=1}^{d_j}\left(T_{g_{j0}}+T_{f_{j0}}+T_{f_{j,i-1}}+T_{e_{ji}}\right)(\lceil ev_{j,i-1}\rceil) \in \mathcal{O}(\frac{j}{j+d_j}(T_{f_j}T_{g_j}+T_{f_j}T_{a_j})+d_j(T_{g_j}+T_{a_j})) \tag{6.22}$$

Also

$$\sum_{i=0}^{d_j-1}(j-3)d_{j,i} \in \mathcal{O}(jd_j^2) \tag{6.23}$$

Now we need to consider the sum

$$\begin{aligned}\sum_{i=0}^{d_j} T_{e_{ji}} j &\leq \sum_{i=0}^{d_j} t_{a_j} j \binom{j+d_{ji}}{j} = \sum_{i=0}^{d_j} t_{a_j} j \frac{j+d_{ji}}{j} \binom{j-1+d_{ji}}{j-1} \\ &\leq t_{a_j}(j+d_j) \sum_{i=0}^{d_j} \binom{j-1+d_{ji}}{j-1} = t_{a_j}(j+d_j) \binom{j+d_j}{j} \\ &= (j+d_j) T_{a_j}\end{aligned}$$

Using the same idea we see that $\sum_{i=0}^{d_j} T_{f_{j,i-1}} j \leq (j+d_j) T_{f_j}$. So

$$\sum_{i=0}^{d_j} (T_{e_{ji}} + T_{f_{j,i-1}}) j \leq (j+d_j)(T_{f_j} + T_{a_j})$$

Also

$$\sum_{i=1}^{d_j-1} \left(T_{g_{j0}} + T_{f_{j0}} \right) j \leq (T_{f_j} + T_{g_j}) \sum_{i=1}^{d_j-1} j \leq (T_{f_j} + T_{g_j}) j d_j$$

So we get

$$\sum_{i=1}^{d_j-1} \left(T_{g_{j0}} + T_{f_{j0}} + T_{f_{j,i-1}} + T_{e_{ji}} \right) j \in \mathcal{O}\left(j d_j (T_{f_j} + T_{g_j}) + (j+d_j)(T_{f_j} + T_{a_j}) \right) \tag{6.24}$$

Let us consider the terms appearing in (6.22-6.23-6.24)

$$\underbrace{\frac{j}{j+d_j}(T_{f_j} T_{g_j} + T_{f_j} T_{a_j}) + d_j (T_{g_j} + T_{a_j})}_{(22)} + \underbrace{j d_j^2}_{(23)} + \underbrace{j d_j (T_{f_j} + T_{g_j}) + (j+d_j)(T_{f_j} + T_{a_j})}_{(24)}$$

We have $d_j T_{g_j} \leq j d_j T_{g_j}$ and since $T_{f_j} \leq T_{g_j}$ we get $j d_j (T_{f_j} + T_{g_j}) \in \mathcal{O}(j d_j T_{g_j})$. So by (6.22-6.23-6.24) the expected cost $C_{Ev_j} = \sum_{i=1}^{d_j} C_{Ev_{ji}}$ of evaluation at the j^{th} step is in

$$C_{Ev_j} \in \mathcal{O}\left(\frac{j}{j+d_j} T_{a_j} T_{f_j} + \frac{j}{j+d_j} T_{f_j} T_{g_j} + j d_j T_{g_j} + (j+d_j)(T_{f_j} + T_{a_j}) + j d_j^2 \right) \tag{6.25}$$

Let C_{B_j} be the expected cost of BDP at the j^{th} step. Then $C_{B_j} = \sum_{i=1}^{d_j} C_{B_{ji}} = \sum_{i=1}^{d_j} \mathcal{O}\left(\lceil ev_{j,i-1} \rceil d_{ji}^3 \right)$. First we consider

$$\sum_{i=1}^{d_j-1} \lceil ev_{j,i-1} \rceil d_{ji}^3 \leq d_j^3 (d_j + \frac{j}{j+d_j} T_{f_j}) \leq d_j^4 + j d_j^2 T_{f_j}$$

Hence

$$C_{B_j} \in \mathcal{O}\left(d_j^4 + j d_j^2 T_{f_j} \right) \tag{6.26}$$

Let C_{V_j} be the expected cost of solving linear systems (in sparse interpolation) at the j^{th} step. Then $C_{V_j} = \sum_{i=1}^{d_j} C_{V_{ji}} = \sum_{i=1}^{d_j} \mathcal{O}\left(T_{f_{j,i-1}}^2 r_{j,i-1}^2\right)$. We consider

$$\begin{aligned}
\sum_{i=0}^{d_j-1} T_{f_{ji}}^2 r_{ji}^2 &= \sum_{i=0}^{d_j-1} \left(t_{f_j}\binom{j+d_{ji}}{j}\frac{j-1}{j-1+d_{ji}}\right)^2 \\
&= \sum_{i=0}^{d_j-1} \left(t_{f_j}\frac{j-1}{j-1+d_{ji}}\frac{j+d_{ji}}{j}\binom{j-1+d_{ji}}{j-1}\right)^2 \\
&\leq t_{f_j}^2 \sum_{i=0}^{d_j} \binom{j-1+d_{ji}}{j-1}^2 \leq t_{f_j}^2 \left(\sum_{i=0}^{d_j}\binom{j-1+d_{ji}}{j-1}\right)^2 \\
&= t_{f_j}^2 \binom{j+d_j}{j}^2 = T_{f_j}^2
\end{aligned}$$

Hence, the expected cost $C_{V_j}$is in

$$C_{V_j} \in \mathcal{O}\left(T_{f_j}^2\right) \tag{6.27}$$

Let the expected cost of multiplication and division at the j^{th} step be C_{D_j} and C_{M_j} resp. Then $C_{M_j} = \sum_{i=1}^{d_j} C_{M_{ji}} = \sum_{i=1}^{d_j} \mathcal{O}\left(t_{f_j} t_{g_j} \binom{j+d_j}{j}\binom{j+d_{ji}}{j}\right)$. Similarly for C_{D_j}. Note that

$$\begin{aligned}
\sum_{i=1}^{d_j-1} t_{f_j} t_{g_j} \binom{j+d_j}{j}\binom{j+d_{ji}}{j} &= t_{f_j} t_{g_j}\binom{j+d_j}{j}\sum_{i=1}^{d_j-1}\binom{j-1+d_{ji}}{j-1} \\
\leq t_{g_j} t_{f_j}\binom{j+d_j}{j}^2 &= T_{f_j} T_{g_j}
\end{aligned}$$

So, the expected cost $C_{M_j}$of sparse multiplication and C_{D_j} of sparse division (in sparse interpolation) at the j^{th} step is in

$$C_{M_j} \in \mathcal{O}\left(T_{f_j} T_{g_j}\right) \text{ and } C_{D_j} \in \mathcal{O}\left(T_{f_j} T_{g_j}\right) \tag{6.28}$$

Let C_{U_j} be the cost of updating the factors at the the j^{th} step. Then $C_{U_j} = \sum_{i=1}^{d_j} C_{U_{ji}} = \sum_{i=1}^{d_j} \mathcal{O}\left(i(t_{f_j} + t_{g_j})\binom{j+d_{ji}}{j}\right)$. We have

$$\begin{aligned}
\sum_{i=1}^{d_j} i t_{g_j}\binom{j+d_{ji}}{j} &= t_{g_j}\sum_{i=1}^{d_j} i\binom{j+d_{ji}}{j} = t_{g_j}\frac{d_j(j+d_j+1)}{(j+1)(j+2)}\binom{j+d_j}{j} \\
&\leq \left(\frac{d_j(j+1)+d_j^2}{j^2}\right) t_{g_j}\binom{j+d_j}{j} = \left(\frac{d_j(j+1)+d_j^2}{j^2}\right) T_{g_j}
\end{aligned}$$

Since $T_{f_j} \leq T_{g_j}$, we get

$$C_{U_j} \in \mathcal{O}\left(\frac{d_j}{j} T_{g_j} + \frac{d_j^2}{j^2} T_{g_j}\right) \tag{6.29}$$

Let C_{Er_j} be the cost of updating *error* at the j^{th} step. Then $C_{Er_j} = \sum_{i=1}^{d_j} C_{Er_{ji}} = \sum_{i=1}^{d_j} \mathcal{O}\left(T_{f_j} T_{g_j}\right)$ is in

$$C_{Er_j} \in \mathcal{O}\left(d_j T_{f_j} T_{g_j}\right) \tag{6.30}$$

According to the equations from (6.25) to (6.30) we shall consider the dominating terms

$$\underbrace{\frac{j}{j+d_j} T_{a_j} T_{f_j} + \frac{j}{j+d_j} T_{f_j} T_{g_j} + j d_j T_{g_j} + (j+d_j)(T_{f_j} + T_{a_j}) + j d_j^2}_{C_{Ev_j}}$$

$$\underbrace{d_j^4 + j d_j^2 T_{f_j}}_{C_{B_j}} + \underbrace{T_{f_j}^2}_{C_{V_j}} + \underbrace{T_{f_j} T_{g_j}}_{C_{M_j} \text{ and } C_{D_j}} + \underbrace{(\frac{d_j}{j} + \frac{d_j^2}{j^2}) T_{g_j}}_{C_{U_j}} + \underbrace{d_j T_{f_j} T_{g_j}}_{C_{Er_j}}$$

The terms $T_{f_j}^2$, $\frac{j}{j+d_j} T_{f_j} T_{g_j}$, $\frac{d_j}{j} T_{g_j}$, $T_{f_j} T_{g_j}$ will be dominated by the term $d_j T_{f_j} T_{g_j}$. The term $j d_j^2$ will be dominated by $j d_j^2 T_{f_j}$. Hence the expected complexity at the j^{th} step is in

$$\mathcal{O}(\underbrace{\frac{j}{j+d_j} T_{a_j} T_{f_j} + j d_j T_{g_j} + (j+d_j)(T_{a_j} + T_{f_j})}_{C_{Ev_j}} + \underbrace{d_j^4 + j d_j^2 T_{f_j}}_{C_{B_j}} + \underbrace{\frac{d_j^2}{j^2} T_{g_j}}_{C_{U_j}} + \underbrace{d_j T_{f_j} T_{g_j}}_{C_{Er_j}})$$

Recall that $f_j := f(x_1, \ldots, x_j, x_{j+1} = \alpha_{j+1}, \ldots, x_n = \alpha_n) \bmod p$. Similarly for a and g. Let $\mathcal{J} = \left\{ j \in \mathbb{N} \mid \max\{t_a, t_f, t_g\} \leq 1/\binom{n-j+d}{d} \right\}$. Then as it was explained in Section 6.4 we expect $T_{f_j}, T_{g_j}, T_{a_j}$ very close to T_f, T_g, T_a resp. for $j \in \mathcal{J}$. Then

$$\sum_{j=3}^{n} T_{a_j} T_{f_j} \in \Omega(|\mathcal{J}| T_a T_f).$$

According to Remark 47, in the sparse examples $|\mathcal{J}| \in \mathcal{O}(n)$. Then $\sum_{j=3}^{n} \frac{j}{j+d_j} T_{a_j} T_{f_j} < \sum_{j=3}^{n} \frac{j}{d_j} T_{a_j} T_{f_j} < \frac{n}{d} \sum_{j=3}^{n} T_{a_j} T_{f_j} \in \mathcal{O}(\frac{n^2}{d} T_a T_f)$

On the other hand we have $\sum_{j=3}^{n} j d_j T_{g_j} \leq d T_g \sum_{j=3}^{n} j \in \mathcal{O}(n^2 d T_g)$, $\sum_{j=3}^{n} (j+d_j)(T_{a_j} + T_{f_j}) \leq (T_a + T_f) \sum_{j=3}^{n} (j+d_j) \leq (T_a + T_f)(n^2 + nd)$

So assuming that the inputs are sparse by running the index j from 3 to n, the expected complexity of MTSHL is in

$$\mathcal{O}(\underbrace{\frac{n^2}{d} T_a T_f + n^2 d T_g + (n^2 + nd)(T_a + T_f)}_{\text{Evaluation}} + \underbrace{nd^4 + n^2 d^2 T_f}_{\text{BDP}} + \underbrace{d^2 T_g}_{\text{Update}} + \underbrace{nd T_f T_g}_{\text{Er}})$$

Since $T_f \leq T_g$ the expected complexity is in

$$\mathcal{O}\left(\frac{n^2}{d}T_aT_g + n^2dT_g + (n^2+nd)(T_a+T_g) + nd^4 + n^2d^2T_g + ndT_g^2\right)$$

$$= \mathcal{O}\left(\frac{n^2}{d}T_aT_g + n^2T_a + ndT_a + n^2d^2T_g + ndT_g^2 + nd^4\right) \tag{6.31}$$

Finally since $T_a \in \mathcal{O}(T_fT_g)$ we have $T_a \in \mathcal{O}(T_g^2)$ and hence the expected complexity is in

$$\mathcal{O}\left(\frac{n^2}{d}T_g^3 + n^2T_g^2 + n^2d^2T_g + ndT_g^2 + nd^4\right) \tag{6.32}$$

Note that if T_g is big enough, for example $T_g > nd^2$ (which is the case for the most our experiments in the final section) then

$$ndT_g^2 > n^2d^3T_g > n^2d^2T_g \text{ and } n^2d^2T_g > n^3d^4 > nd^4, \text{ and}$$

$$ndT_g^2 < nd^2T_g^2 < T_g^3 \text{ and } nT_g^3 > n^2d^2T_g^2 > n^2T_g^2$$

Hence the expected complexity is in

$$\mathcal{O}\left(\frac{n^2}{d}T_g^3\right).$$

The cubic term is coming from evaluation and suggests the evaluation is the most time dominating step. This is what we have expected and will confirm by experimental data. Now according to Theorem 28

$$\Pr[\mathrm{Supp}(f_{j,i+1}) \nsubseteq \mathrm{Supp}(f_{ji})] \leq T_{f_{j,i+1}}\frac{d_{ji}}{p-d_{ji}+1} \leq T_{f_j}\frac{d-i}{p-2d+1}.$$

Then the probability that there is a false assumption on one of the assumptions of (#10) at the j^{th} step is $\leq \sum_{i=0}^{d_j-1} T_{f_{j,i+1}}\frac{d_{ji}}{p-d_{ji}+1} \leq \frac{d^2}{2(p-2d+1)}T_{f_j}$. Hence throughout the whole MHL process the probability of failure of MHL because of a false assumption at (#10) is $\leq \frac{(n-2)d^2}{2(p-2d+1)}T_{f_j}$. Also note that at the j^{th} step the algorithm used $\leq d_jT_{f_j}$ many evaluations and made an assumption on the gcd of univariate polynomials in BDP based on Schwartz-Zippel's lemma. Then the probability of a failure because of a false assumption on the gcd of polynomials in BDP is then $\leq \frac{d}{p-1}\sum_{j=3}^{n} d_jT_{f_j} \leq \frac{(n-2)(T_fd^2)}{p-1}$. Then the probability of failure of SHL is

$$\leq (n-2)d^2T_f\left(\frac{1}{p-1} + \frac{1}{2(p-2d+1)}\right)$$

This is a very generous bound. In our experiments to construct the data in the final section, we have used $p = 2^{31}-1$ and MTSHL has never failed.

As a final note, we consider the probabilistic assumption in Step 4 of the sparse interpolation routine SparseInt.

In the notation of Algorithm 7, SparseInt, consider the polynomials

$$\Delta_{ik} = \prod_{1 \le a < b \le s_{ik}} (M_{ika} - M_{ikb}) \in \mathbb{Z}_p[x_3, \ldots, x_j].$$

In Step 4, $|S_{ik}| = s_{ik}$ means the monomial evaluations are distinct and if this won't be the case then at least one of the Vandermonde matrices constructed in Step 14 won't be invertible. In that case then $\Delta_{ik}(\alpha_3, \ldots, \alpha_j) = 0$. We want to bound the probability that this may happen for any Δ_{ik}. Let $\Delta = \prod \Delta_{ik}$. Then $\Delta(\alpha_3, \ldots, \alpha_j) = 0$ means one or more monomial sets are not distinct. Since $\alpha_3, \ldots, \alpha_j$ were chosen at random from $[1, p-1]$ we have by Schwartz-Zippel

$$\Pr[\Delta(\alpha_3, \ldots, \alpha_j) = 0] \le \frac{\deg(\Delta)}{p-1}.$$

We have $\deg M_{ikl} \le d$ and $\deg \Delta \le \sum_{0 \le i+k \le d} d\binom{s_{ik}}{2}$. Note that $\sum s_{ik} = T_{f_j}$ and $\deg \Delta$ is maximized when one of the coefficients has all T_{f_j} terms, that is some $s_{ik} = T_{f_j} \le T_f$. Thus $\deg \Delta \le d\binom{T_f}{2}$ and we obtain

$$\Pr[\Delta(\alpha_3, \ldots, \alpha_j) = 0] \le \frac{dT_f^2}{2(p-1)}.$$

So when p is big enough so that $p \gg 1 + dT_f^2/2$ we expect that a different choice of $(\alpha_3, \ldots, \alpha_j)$ will satisfy the condition. This is again a very generous bound. In our experiments we have used $p = 2^{31} - 1$ and SparseInt routine has never failed at Step 4.

Theorem 53. *Let $a = fg \in \mathbb{Z}_p[x_1, \ldots, x_n]$ with f, g monic in x_1 and $T_f \le T_g$. Let $d = \deg(a)$, $r = \frac{n}{n+d}$ and p be a big prime with $p \gg d$. Then with the probability of failure $\le (n-2)d^2 T_f \left(\frac{1}{p-1} + \frac{1}{2(p-2d+1)}\right)$, the expected complexity of the MHL to recover the factors f, g via SHL algorithm is in*

$$\mathcal{O}\left(\frac{n^2}{d} T_g^3 + n^2 T_g^2 + n^2 d^2 T_g + n d T_g^2 + n d^4\right)$$

In particular if $T_g > nd^2$ then the expected complexity is in

$$\mathcal{O}\left(\frac{n^2}{d} T_g^3\right).$$

6.6 Some Timing Data

To compare the result of our ideas to Wang's, first we factored the determinants of Toeplitz and Cyclic matrices of different sizes as concrete examples.

Then we created sparse random polynomials A, B using

$$x_1^d + \texttt{randpoly}([x_2, .., x_n], \texttt{degree} = d, \texttt{terms} = t)$$

in Maple and computed $C = AB \in R$. We chose monic factors in x_1 so as not to complicate the algorithm with leading coefficient correction and to have a fair comparison with Maple's factorization algorithm.

We used $p = 2^{31} - 1$ and **two ideal types to factor C:**

ideal type 1: $I = \langle x_2 - 0, x_3 - 0, \cdots, x_n - 0 \rangle$ and

ideal type 2: $I = \langle x_2 - \alpha_1, x_3 - \alpha_2, \cdots, x_n - \alpha_n \rangle$ in which the α_i's in practice are chosen from a small interval including zero, for Wang's algorithm.

As noted it is not always possible to use ideal type 1. For example, ideal type 1 cannot be used to factor Cyclic or Toeplitz determinants. On the other hand, to make a fair comparison on the efficiency of the algorithms, when timing Wang's algorithm, we let Wang's algorithm choose its own ideal. It can include some zeros although it is not possible to choose all zero for these examples. That is, both MTSHL and Wang's algorithm are doing their best under fair conditions.

6.6.1 Factoring Toeplitz and Cyclic matrices: The state of art

Let C_n denote the $n \times n$ cyclic matrix and let T_n denote the $n \times n$ symmetric Toeplitz matrix below.

$$C_n = \begin{pmatrix} x_1 & x_2 & \dots & x_{n-1} & x_n \\ x_n & x_1 & \dots & x_{n-2} & x_{n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_3 & x_4 & \dots & x_1 & x_2 \\ x_2 & x_3 & \dots & x_n & x_1 \end{pmatrix} \text{ and } T_n = \begin{pmatrix} x_1 & x_2 & \cdots & x_{n-1} & x_n \\ x_2 & x_1 & \cdots & x_{n-2} & x_{n-1} \\ & \ddots & \ddots & \ddots & \\ x_{n-1} & x_{n-2} & \cdots & x_1 & x_2 \\ x_n & x_{n-1} & \cdots & x_2 & x_1 \end{pmatrix}$$

The determinants of C_n and T_n are homogeneous polynomials in n variables $x_1, x_2, \dots, x_n$.

As we implemented MTSHL in Maple (except two key parts: BDP (see Section 5.4) and the evaluation routine (see Section 3.6.2) are coded in C), we first compared Maple factorization timings with Singular and Magma to be sure that the main gain by MTSHL is independent of implementation.

The data in Tables 6.6 and 6.7 for factoring the determinants of C_n and T_n compares Maple 2017 with Magma 2.22–5 and Singular 3–1–6. Timings are in CPU seconds. Since

n	$\#d_n$	sizes of factors	Maple	# MDP	MDP%	Magma	Singular
7	427	30, 56	0.035	161	30%	0.01	0.02
8	1628	167, 167	0.065	383	43%	0.04	0.05
9	6090	153, 294	0.166	1034	73%	0.10	0.28
10	23797	931, 931	0.610	2338	76%	0.89	1.77
11	90296	849, 1730	2.570	6508	74%	1.96	8.01
12	350726	5579, 5579	19.45	15902	80%	72.17	84.04
13	1338076	4983, 10611	84.08	45094	84%	181.0	607.99
14	5165957	34937, 34937	637.8	103591	77%	6039.0	20333.45
15	19732508	30458, 66684	4153.2	286979	84%	12899.2	–

Table 6.6: Factorization timings in CPU seconds for factoring d_n the determinant of the n by n Toeplitz matrix T_n evaluated at $x_n = 1$

n	$\#d_n$	sizes of factors	Maple	# MDP	MDP%	Magma	Singular
7	246	7,924	0.045	330	90%	0.01	0.02
8	810	8,8,20,86	0.059	218	46%	0.07	0.06
9	2704	9,45,1005	0.225	1810	74%	0.74	0.24
10	7492	10,10,715,715	0.853	1284	62%	8.44	2.02
11	32066	11,184756	7.160	75582	91%	104.3	11.39
12	86500	12,12,42,78,78,621	19.76	1884	76%	7575.1	30.27
13	400024	13, 2704156	263.4	1790701	92%	30871.90	NA
14	1366500	14,14,27132,27132	1664.4	50381	77%	$> 10^6$	288463.17
15	4614524	15,120,3060,303645	18432.	477882	82%	–	NA

Table 6.7: Factorization data and timings in CPU seconds for factoring d_n the determinant of the n by n Cyclic matrix C_n evaluated at $x_n = 1$

$\det(T_n)$ and $\det(C_n)$ are homogeneous, and since the difficulty of the factorization of $\det(T_n)$ depends on which variable is used to de-homogenize the determinant, we fixed $x_n = 1$ for all three systems. That is, for $d_n = \det(T_n)$ we time the factorization of $d_n(x_1, x_2, \ldots, x_{n-1}, 1)$. The column (MDP) shows the number of calls (including recursive calls) to Maple's MDP algorithm and the percentage of time in Hensel lifting spent solving MDPs.

Data for the number of terms of $\det T_n$ and $\det C_n$ and the number of terms of their factors is given in Tables 6.6 and 6.7. In Table 6.7 notice that the second factor for $n = 7, 11, 13$ has more terms than $\det(C_n)$.

The data confirms the data from [MP14] that Maple's multivariate factorization code is relatively fast. This is partly because the underlying polynomial arithmetic is fast (see Monagan and Pearce [MP14]). We note here that Maple, Magma [Steel] and Singular [Lee] are all doing Hensel lifting one variable at a time (see Algorithm 6.4 of [GCL92]) and lifting solutions to MDP equations one variable at a time (see Algorithm 6.2 of [GCL92]). All

coefficient arithmetic is done modulo a prime p or or prime power p^l which bounds the size of the coefficients of any factors of the input. Singular [Lee] differs from Maple and Magma in that it first factors a bivariate image $f(x_1, x_2, \alpha_3, \ldots, \alpha_n)$ over $\mathbb{Z}$ then starts the Hensel lifting from bivariate images of the factors.

6.6.2 Factoring Toeplitz and Cyclic matrices with MTSHL

For MTSHL, it is important that α_i's are chosen from a large interval. For these we chose α_i's randomly from $\mathbb{Z}_q - \{0\}$ with $q = 65521$.

Maple, Magma, Singular all check for the homogeneity before factorization and if it is the case they first de-homogenize the polynomial to be factored. After factoring the de-homogenized polynomial, they homogenize the factors to obtain the actual factors. We followed the same procedure for MTSHL. As above we fixed $x_n = 1$ to de-homogenize.

In the tables we used the following notation

tW	is the time for Wang's algorithm which Maple currently uses (see[GCL92]),
tBS	is the time where factoring algorithm is based on Algorithm 6 ,
tKSHL	is the time where factoring algorithm is based on Algorithm 7 ,
tMTSHL	is the time where factoring algorithm is based on Algorithm 8 ,
tX(tY)	means factoring time tX with tY seconds spent on solving MDP,
t_{mul}	means time spent on multiplication in MTSHL,
t_{eval}	means time spent on evaluation in MTSHL,
T_{f_i}	denotes the number of terms of a factor
t_{f_i}	denotes the density ratio of a factor

The density ratio of factors of T_n can be seen in Table 6.8.

For C_n the density ratio is 1 for all factors except for $n = 12$, in which out of 6 factors one of them has $t_f = 0.53$ and the other has $t_f = 0.45$.

Table 6.8 presents timings for Hensel liftings to factor $\det T_n$ with MTSHL.

Table 6.9 presents timings for Hensel liftings to factor $\det C_n$ with MTSHL.

The values in the columns tW and tMTSHL shows that in general the most time dominating step of MHL is solving MDP and confirms that even for complicated examples MTSHL is quicker than Wang's algorithm. It spends less time to solve MDP. Also, the values in the columns t_{mul} and t_{eval} confirms the theoretical complexity analysis that evaluation and multiplication are the most time dominating operations in MTSHL.

n	t_{f_1}	t_{f_2}	tW	tMTSHL	t_{mul}	t_{eval}
7	0.27	0.36	0.035 (0.015)	0.046 (0.037)	0.001	0.003
8	0.50	0.50	0.065 (0.028)	0.073 (0.059)	0.007	0.005
9	0.31	0.23	0.166 (0.121)	0.122 (0.075)	0.018	0.001
10	0.47	0.47	0.610 (0.467)	0.418 (0.251)	0.099	0.024
11	0.22	0.28	2.570 (1.902)	1.138 (0.458)	0.339	0.053
12	0.45	0.45	19.45 (15.56)	13.165 (5.445)	3.779	0.897
13	0.27	0.21	84.08 (70.623)	21.769 (11.064)	6.904	4.361
14	0.45	0.45	637.8 (491.106)	249.961 (160.04)	71.351	102.918
15	0.21	0.26	4153.2 (1771.54)	1651.68 (689.634)	674.356	405.016

Table 6.8: Timings for factoring determinants of $n \times n$ symmetric Toeplitz matrices.

n	tW	tMTSHL	t_{mul}	t_{eval}
7	0.041 (0.012)	0.026 (0.015)	0.002	0.001
8	0.057 (0.025)	0.063 (0.046)	0.010	0.003
9	0.209 (0.152)	0.12 (0.042)	0.024	0.002
10	0.845 (0.642)	0.5 (0.22)	0.20	0.01
11	6.6 (4.884)	0.945 (0.094)	0.386	0.003
12	19.76 (15.808)	5.121 (1.385)	3.108	0.048
13	252.2 (211.848)	27.689 (1.474)	9.362	0.093
14	1861.8 (1563.912)	523.073 (85.326)	346.067	38.399
15	18432.0 (14929.2)	7496.94 (426.014)	3602.739	19.231

Table 6.9: Timings for factoring determinants of $n \times n$ cyclic matrices.

6.6.3 Factoring Random sparse data with MTSHL and KSHL

Table 6.10 presents timings for the random data for which ideal type 2 is used. It shows that when the evaluation points are chosen to be non-zero MTSHL is significantly faster than Wang's algorithm.

As can be seen from Tables 6.12, 6.10 that KSHL is only good for very sparse examples. Even for these cases MTSHL performs better.

We also included the timings for ideal type 1 case, as according to our experiments it is the only case where Wang's algorithm is quicker. This is because a sparse polynomial remains sparse for the ideal type 1 and hence the number of MDP to be solved significantly decreases and the evaluation cost of sparse interpolation becomes dominant which is not the case for Wang's algorithm for the ideal type 1.

Table 6.11 presents timings for the random data for which ideal type 1 is used. For the ideal type 1 case MTSHL or KSHL were not used, since the zero evaluation probability is large for the sparse case. According to our experiments Wang's algorithm is faster for

$n/d/T$	tW	tKSHL	tMTSHL
4/35/100	13.07 (11.95)	1.75 (1.18)	1.51 (0.24)
5/35/100	88.10 (86.28)	3.75 (2.57)	1.16 (0.36)
7/35/100	800.0 (797.0)	5.04 (4.08)	1.58 (0.59)
9/35/100	4451.6 (4449.4)	8.13 (6.22)	2.94 (0.56)
4/35/500	33.96 (26.48)	642.2 (635.1)	11.29 (0.82)
5/35/500	472.1 (402.5)	1916.2 (1899.6)	26.0 (4.86)
7/35/500	3870.5 (3842.2)	2329.4 (2305.5)	43.1 (6.84)
9/35/500	> 60000	3866.3 (3805.9)	79.6 (9.71)
11/35/500	NA	NA	243.2 (17.0)

Table 6.10: The timing table for random data with ideal type 2

$n/d/T_{f_i}$	t_{f_i}	tW	tBS
5/20/5000	0.1	7.08 (2.605)	11.804 (7.376)
5/15/3000	0.2	4.25 (1.554)	4.963 (2.29)
5/15/5000	0.3	6.882 (2.988)	6.471 (2.99)
4/20/5000	0.4	2.709 (1.211)	2.704 (1.267)
5/20/30000	0.56	86.224 (18.394)	90.111 (21.134)

Table 6.11: The timing table for random data with ideal type 1

$t_f < 0.2$. For $t_f \geq 0.2$ the performance of Wang's algorithm and Algorithm 6 (which uses sparse interpolation without SHL assumption) are almost the same.

Toeplitz	tKSHL	Cyclic	tKSHL
5	0.02 (0.018)	5	0.07 (0.06)
6	0.308 (0.306)	7	1157 (1157)
7	1157.5 (1157.5)	11	∞
8	119.88 (119.88)	13	∞
9	486.45 (485.41)		
10	25021 (25020)		

Table 6.12: The timing table for KSHL

Chapter 7

Conclusion

7.1 Summary

Within this dissertation we presented an efficient practical algorithm MTSHL for factoring multivariate polynomials over integers.

We have shown that solving the multivariate polynomial diophantine equations in multivarite Hensel lifting algorithm can be improved by using sparse interpolation techniques. This leads to an overall improvement in multivariate polynomial factorization.

We provided benchmarks comparing MTSHL with previous sparse Hensel lifting approaches developed by Kaltofen and Zippel. We also compared MTSHL with Wang's algorithm which is used in Maple, Magma and Singular.

We also studied what happens to the sparsity of multivariate polynomials when the variables are successively evaluated at numbers. We determined the expected number of terms remaining and the variance. We used these results to revisit and correct the complexity analysis of Zippel's original sparse interpolation.

Finally we computed the probability of success of MTSHL, presented an average case complexity analysis of it, and confirmed our theoretical analysis with experimental data.

7.2 Future Work

One future extension point is, to decrease the cost of MHL using our sparse interpolation techniques when there are more than 2 factors to be computed.

When the polynomial to be factored has more than 2 factors, say 4 factors, then the MDP to be has the form

$$\sigma_1 b_1 + \sigma_1 b_2 + \sigma_3 b_3 + \sigma_4 b_4 = c_k$$

in $\mathbb{Z}_p[x_1, \ldots, x_j]$ where $b_i = \prod_{j \neq i} u_j$. The current approach is to solve this equation by reducing it into 3 MDP's of the form $\sigma u + \tau w = c$ in $\mathbb{Z}_p[x_1, \ldots, x_j]$ as explained in Chapter 1. However one can invoke the inproved sparse interpolation idea used in MTSHL to recover

σ_i's simultaneoulsy without reducing it into 3 MDP's. With this way one does not need to compute $b_i = \prod_{j \neq i} u_j$ in $\mathbb{Z}_p[x_1, \ldots, x_j]$ but instead consider the bivariate images

$$b_i(x_1, x_2, \beta^j) = \prod_{j \neq i} u_j(x_1, x_2, \beta^j) \ \in \mathbb{Z}_p[x_1, x_2]$$

where β is chosen at random from $\mathbb{Z}_p^{n-2}$. Since this approach is probabilistic at the end one can check the correctness of the solutions by sufficiently many evaluations. We expect that this approach will decrease the computational cost of solving multi-polynomial diophantine equations.

Another extension point is to decrease the cost of error computation during MHL which is one of the most time dominating part of MHL.

At the j^{th} step of MHL, in a for loop one needs the compute the Taylor coefficient c_k of the error e_k in the k^{th} iteration. c_k appears on the RHS of the MDP equation to be solved . The computation of the error e_k is one of the most time dominating step in the k^{th} iteration. MTSHL solves the MDP by projecting down the MDP into the bivariate domain $\mathbb{Z}_p[x_1, x_2]$ and it needs the bivariate images of c_k. Instead of computing the error e_k in $\mathbb{Z}_p[x_1, \ldots, x_j]$, one can compute the bivarite images of e_k in $\mathbb{Z}_p[x_1, x_2]$ and so compute the bivariate images of c_k in $\mathbb{Z}_p[x_1, x_2]$. We expect that this approach will decrease the error cost.

Bibliography

[Ber48] E. R. Berlekamp. Factoring polynomials over large finite fields. *Math. Comp.*, v. 24, pp. 713-735. MR 43 #1948. (1970). 2.1

[Ber67] E. R. Berlekamp. Factoring polynomials over finite fields. Bell System Tech. J., v. 46, pp. 1853-1859. MR 36 #2314. (1967). 1.1, 2.1

[Ber68] E. Berlekamp. *Algebraic Coding Theory*, McGraw-Hill, (1968). 5.4

[BC07] A. Benjamin and C. Bennett. The Probability of Relatively Prime Polynomials. *Mathematics Magazine* **80** (3), 197–202, (2007). 5.4

[BLS03] A. Bostan, G. Lecerf, E. Schost. Tellegen's principle into practice. *Proceedings of the 2003 International Symposium on Symbolic and Algebraic Computation*, ISSAC '03, pages 37-44. ACM, (2003). 3.6.2

[CLO07] David Cox, John Little, Donal O'Shea. *Ideals, Varieties and Algorithms*. Springer-Verlag, 3rd ed., (2007). 3.1, 3.2, 3.4

[EK40] Paul Erdos, Mark Kac. The Gaussian Law of Errors in the Theory of Additive Number Theoretic Functions. *American Journal of Mathematics*. 62 (1/4): 738-742. (1940). 4.5

[Gel60] A.O. Gelfond, *Transcendental and Algebraic Numbers*, GITTL, Moscow, 1952; English translation by Leo F. Boron, Dover, New York, 1960. 2.1, 11, 12

[GCL92] K.O. Geddes, S.R. Czapor, G. Labahn, *Algorithms for Computer Algebra*, Kluwer Acad. Publ. (1992). 1.1, 1.2.1, 1.3, 5, 1.4, 8, 2.1, 2.1, 3.1, 3.2, 4.5, 6.4, 6.6.1, 6.6.2

[GR03] Ralph P. Grimaldi, *Discrete and Combinatorial Mathematics*, Pearson Addison Wesley, 5th ed., (2003). 5.3, 5.3

[HR17] G.H. Hardy, S. Ramanujan. The normal number of prime factors of a number n. *Quarterly Journal of Mathematics*. 48: 76-92. (1917). 4.5

[Joh74] S.C. Johnson. Sparse polynomial arithmetic, *ACM SIGSAM Bulletin*, 8 (3), pp. 63-71 (1974). 3.4

[Kal85] Kaltofen, E., Sparse Hensel lifting. Proc. EUROCAL '85, Springer Verlag LNCS, vol 204, pp. 4-17, (1985). 1.1, 1.2.1, 4.2.1, 4.3, 4.3.1

[Kal82] Kaltofen, E., On the complexity of factoring polynomials with integer coefficients, PhD Thesis. Rensselaer Polytechnic Institute. (1982).

[KK93] A. Knopfmacher and J. Knopfmacher. Counting irreducible factors of polynomials over finite fields. *Discrete Mathematics* **112** (1993) 103–118. 5.2.2

[Kro95] L. Kronecker. Grundzuge einer artihmetischen Theorie der algebraischen Grossen. *J. reine angew. Math.* **115**, pp. 53-78, (1895). 1.1

[KR00] Martin Kreuzer and Lorenzo Robbiano. *Computational Commutative Algebra 1*, Springer-Verlag Berlin Heidelberg, (2000). 3.2

[Knu81] D.E. Knuth. The Art of Computer Programing. Addison Wesley (1981). 1.1

[Lee] Martin M. Lee. Private Communication. 6.6.1

[Lee13] Martin M. Lee. Factorization of multivariate polynomials. Ph.D. Thesis. (2013). 4.5

[Law17] Marshall Law. *Computing Characteristic Polynomials of Matrices of Structured Polynomials*, Master Thesis, SFU. (2017). 4.5

[Lang62] S. Lang. *Diophantine Geometry*, Wiley, New York, (1962). 2.1

[Mig74] M. Mignotte. An equality about factors of polynomials. *Math. Comp*, **v.28**, pp. 1153-1157, (1974) 1.3, 1

[MY73] J. Moses & D. Y. Y. Yun. The EZ-GCD algorithm. Proceedings of ACM Annual Conference, August (1973).

[MP14] Michael Monagan and Roman Pearce. POLY. A New Polynomial Data Structure for Maple 17. *Computer Mathematics*, Springer Verlag, 325--348, (2014). 4.4.2, 6.5.2, 6.6.1

[MP11] Michael Monagan and Roman Pearce. Sparse polynomial division using a heap . *J. Symbolic Comput.*, **46**(7), pp. 807-822 (2011). 3.4

[Mus71] D. R. Musser. *Algorithms for Polynomial Factorization.* PhD Thesis. University of Wisconsin (1971). 1.1, 2.1, 4.5

[Mus75] D. R. Musser. Multivariate Polynomial Factorization. *J. Assoc. Comput. Mach.*, v.22, pp. 291-308, (1975). 1.1, 2.1, 2.1

[MT16-1] Michael Monagan and Baris Tuncer. Some results on counting roots of polynomials and the Sylvester resultant. *Proceedings of FPSAC 2016*, DMTCS. pp. 887--898, (2016). 1.2.2, 6.4

[MT16-2] Michael Monagan and Baris Tuncer. Using Sparse Interpolation in Hensel Lifting. *Proceedings of CASC 2016*, Springer-Verlag LNCS, (2016). 1.1, 1.2.1, 6.4

[MW17] Michael Monagan, Allen Wong. Fast Parallel Multi-point Evaluation of Sparse Polynomials *ACM Communications in Computer Algebra*,Volume 51 Issue 1, Pages 12-14, March (2017) 3.6.2

[MY74] Miola A., Yun D. Y. Y. Computational Aspects of Hensel-type Univariate Polynomial Greatest Common Divisor Algorithms. *Proceedings of EUROSAM '74*, ACM, pp. 46-54, (1974). 4.4.2

[OEIS] Sequence http://oeis.org/A006579 in *The On-Line Encyclopedia of Integer Sequences*, published electronically at http://oeis.org, 2010. 5.2.2

[Pan13] Daniel Panario and Gary Mullen. Greatest common divisors of polynomials. *Handbook of Finite Fields.* CRC Press, (2013). 5.2.2, 5.4

[Sch76] W. Schmidt. *Equations over Finite Fields: An Elementary Approach.* Springer-Verlag LNCS **536** (1976) Ch 4 pp. 157–159. 5.2, 5.2.2

[Sch80] Schwartz, Jack . Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM* **27**:701--717, (1980). 23

[Steel] Steel, Allan. Private Communication. 4.5, 6.6.1

[Wan78] Wang, P.S. An improved Multivariate Polynomial Factoring Algorithm, *Mathematics of Computation*, **32**, (1978). 1.1, 2.1, 2.2, 2.2, 3.3.1, 4.1

[WR75] Wang, P.S., Rothschild, L.P. Factoring multivariate polynomials over the integers. *Mathematics of Computation*, vol 29, NUMBER 131, pp. 935--950, (1975). 1.1, 2.2, 3.3.1

[Wit04] A.Wittkopf. Algorithms and implementations for differential elimination. PhD Thesis. Simon Fraser University (2004). 1.1

[Yun74] Yun, D.Y.Y. The Hensel Lemma in algebraic manipulation. Ph.D. Thesis. (1974) 1.4, 4.1

[Zas69] H. Zassenhaus. On Hensel factorization. I. *J. Number Theory*, v. 1, pp. 291-311. MR 39 #4120, (1969) 1.1

[Zip90] Zippel, R.E. Interpolating polynomials from their values. *J. Symbolic Comput.*, **9**(3), 375-403, (1990). 1.1, 1.2.1, 23, 3.5, 3.6, 3.6.2, 5.1, 6.4

[Zip79] Zippel, R.E. Probabilistic algorithms for sparse polynomials. *Proc. EUROSAM '79*, Springer Lec. Notes Comp. Sci., vol. 72, pp. 216--226, (1979). 1.2.2, 6.1, 6.2.1, 1

[Zip81] Zippel, R.E. Newton's iteration and the sparse Hensel algorithm. *Proc.* ACM Symp. Symbolic Algebraic Comp., 68--72, (1981). 1.1, 1.2.1, 4.2, 4.2.1

[Zip93] Zippel, R.E. *Effective Polynomial Computation*, Kluwer Acad. Publ. (1993). 4.1, 4.2, 4.2, 4.2.1