Modular Las Vegas Algorithms for Polynomial Absolute Factorization

Cristina Bertone [a,b], Guillaume Chèze [c], André Galligo [a]

[a] *Laboratoire J.-A. Dieudonné, Université de Nice - Sophia Antipolis, France*

[b] *Dipartimento di Matematica, Università degli Studi di Torino, Italy*

[c] *Institut de Mathématiques de Toulouse, Université Paul Sabatier Toulouse 3, France*

Abstract

Let $f(X,Y) \in \mathbb{Z}[X,Y]$ be an irreducible polynomial over $\mathbb{Q}$. We give a Las Vegas absolute irreducibility test based on a property of the Newton polytope of f, or more precisely, of f modulo some prime integer p. The same idea of choosing a p satisfying some prescribed properties together with LLL is used to provide a new strategy for absolute factorization of $f(X,Y)$. We present our approach in the bivariate case but the techniques extend to the multivariate case. Maple computations show that it is efficient and promising as we are able to construct the algebraic extension containing one absolute factor of a polynomial of degree up to 400.

Key words: Absolute factorization, modular computations, LLL algorithm, Newton polytope.

Introduction

Kaltofen's survey papers (Kaltofen, 1992) related the early success story of polynomial factorization. Since then, crucial progresses have been achieved : algorithms developed and implemented by Van Hoeij and his co-workers in the univariate case (Belabas et al., 2004), by Gao and his co-workers (see for instance Gao (2003)), then by Lecerf and his co-workers in the multivariate case (Bostan et al. (2004), Lecerf (2007)). Chèze (2004b), Chèze and Lecerf (2007) and Lecerf (2007) also improved drastically the multivariate absolute factorization (i.e. with coefficients in the algebraic closure): they produced an algorithm with the best known arithmetic complexity. Even if the situation evolved rapidly, there is still room for improvements and new points of view.

Here, we focus on absolute factorization of rationally irreducible polynomials with integer coefficients (see Chèze and Galligo (2005), Rupprecht (2004), Sommese et al.

Email addresses: cristina.bertone@unice.fr (Cristina Bertone), guillaume.cheze@math.ups-tlse.fr (Guillaume Chèze), galligo@math.unice.fr (André Galligo).

(2004) and the references therein). For such polynomials, the best current algorithm and implementation is Chèze's (Chèze (2004a), Chèze (2004b)) presented at Issac'04, it is based on semi-numerical computation, uses LLL and is implemented in Magma. It can factorize polynomials of high degrees, up to 200. One of the challenges is to improve its capabilities at least in certain situations.

We propose yet another strategy and algorithm to deal with (multivariate) absolute irreducibility test and factorization. This article will present a simple, but very efficient, irreducibility test. Then we extend our strategy to get a factorization algorithm based on modular computations, Hensel liftings and recognition of algebraic numbers via $p-$adic approximation using LLL (as explained in von zur Gathen and Gerhard (2003)).
Our absolute factorization algorithm can be viewed as a drastic improvement of the classical algorithm TKTD (see Dvornicich and Traverso (1989), Kaltofen (1985), Trager (1985) and Section 3). Indeed, we replace the computations in an algebraic extension of $\mathbb{Q}$ of degree n, the degree of the input polynomial, by computations in an extension of the minimal degree s, the number of factors of the input polynomial.

We made a preliminary implementation in Maple and computed several examples. It is very promising as it is fast and able to compute the researched algebraic extension for high degree polynomials (more than degree 200, see last section). The bottleneck of the procedure is now the final X-adic Hensel lifting, but we may avoid this problem, for instance, with a parallel version of our algorithm, as explained in Section 4.1.

In other words, our approach improves the practical complexity of absolute factorization of polynomials with integer coefficients.

Notations

$\mathbb{K}$ is a perfect field, $\overline{\mathbb{K}}$ is an algebraic closure of $\mathbb{K}$.
$\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$ is the finite field with p elements, where p is a prime integer.
tdeg f is the total degree of the polynomial f.

1. Absolute irreducibility test and Newton Polytope

Any implementation of an absolute factorization algorithm needs to first check if the polynomial is "trivially" absolutely irreducible. That is to say, test quickly a sufficient condition on f: when the test says yes, then f is absolutely irreducible and the factorization algorithm can be spared. The test should be fast and should, in "most" cases (i.e. with a good probability) say yes when the polynomial f is irreducible. For instance, for polynomials of degree 100, one might expect that such a test runs 100 time faster than a good general factorization algorithm. This is indeed the case for the test presented in this section: for a polynomial of degree 100, absolute factorization algorithms (e.g. the ones in Chèze (2004a) and Chèze and Lecerf (2007)) require 20 seconds to decide irreducibility while our test answers after only 0.07 seconds.

The absolute irreducibility test presented in this article is based on properties of the Newton polytope of a polynomial that we now review.

Definition 1. Let $f(X,Y) = \sum_{i,j} c_{i,j} X^i Y^j \in \mathbb{K}[X,Y]$. The Newton polytope of f, denoted by P_f, is the convex hull in $\mathbb{R}^2$ of all the points (i,j) with $c_{i,j} \neq 0$.

A point (i,j) is a *vertex* of P_f if it is not on the line segment of any other two points of the polytope. □

Remember that a polytope is the convex hull of its vertices.

We refer to Gao (2001) for basic results on absolute irreducibility and Newton polytopes and also for an interesting short history which goes back to the famous Eisenstein criterion.

Definition 2. Denote by (i_1, j_1), ..., $(i_l, j_l) \in \mathbb{Z}^2$ the vertices of P_f. We say that condition (C) is satisfied when $\gcd(i_1, j_1, \dots, i_l, j_l) = 1$. □

The aim of this section is to prove the following criterion.

Proposition 3 (Absolute irreducibility criterion).
Let $f(X, Y)$ be an irreducible polynomial in $\mathbb{K}[X, Y]$. If condition (C) is satisfied then f is absolutely irreducible.

Our statement in Proposition 3 bears similarities with one of Gao's result (Gao, 2001); but it differs since Gao assumed that P_f should be contained in a triangle when we assume that f is irreducible in $\mathbb{K}[X, Y]$. Although, our condition seems a strong theoretical hypothesis, in practice we can check it very quickly thanks to the algorithms developed in Bostan et al. (2004) and Lecerf (2006). The advantage of our criterion is that it applies to a larger variety of polytopes.

We first recall an important lemma about absolute factorization of (rationally) irreducible polynomials.

Lemma 4. *Let $f \in \mathbb{K}[X, Y]$ be an irreducible polynomial in $\mathbb{K}[X, Y]$, monic in Y:*

$$f(X, Y) = Y^n + \sum_{k=0}^{n-1} \sum_{i+j=k} a_{i,j} X^i Y^j.$$

Let $f = f_1 \cdots f_s$ be the monic factorization of f by irreducible polynomials f_l in $\overline{\mathbb{K}}[X, Y]$. Denote by $\mathbb{L} = \mathbb{K}[\alpha]$ the extension of $\mathbb{K}$ generated by all the coefficients of f_1. Then each f_l can be written:

$$f_l(X, Y) = Y^m + \sum_{k=0}^{m-1} \sum_{i+j=k} a_{i,j}^{(l)} X^i Y^j = Y^m + \sum_{k=0}^{m-1} \sum_{i+j=k} b_{i,j}(\alpha_l) X^i Y^j, \tag{1}$$

where $b_{i,j} \in \mathbb{K}[Z]$, $\deg_Z(b_{i,j}) < s$ and where $\alpha_1, \dots, \alpha_s$ are the different conjugates over $\mathbb{K}$ of $\alpha = \alpha_1$. □

See (Rupprecht, 2004, Lemma 2.2) for a proof.

As a corollary the number of absolute factors is equal to $[\mathbb{L} : \mathbb{K}]$.

In order to prove Proposition 3, we introduce the Minkowski sum and its properties concerning polytopes.

Definition 5. If A_1 and A_2 are two subsets of the vector space $\mathbb{R}^n$, we define their *Minkowski sum* as

$$A_1 + A_2 = \{a_1 + a_2 \mid a_1 \in A_1, a_2 \in A_2\}. \qquad \square$$

Lemma 6 (Ostrowski). *Let $f, g, h \in \mathbb{K}[X_1, X_2, \dots, X_n]$ with $f = gh$. Then $P_f = P_g + P_h$.*

Proof. See Ostrowski (1975). □

In particular (Schneider, 1993), if we sum up s times the same convex polytope A, then we have that

$$\underbrace{A + \cdots + A}_{s-times} = s \cdot A,$$

where $s \cdot A = \{s \cdot v \,|\, v \in A\}$. Furthermore the vertices $\{v_1, \ldots, v_l\}$ of $s \cdot A$ are exactly $v_i = s \cdot w_i$, where $\{w_1, \ldots, w_l\}$ is the set of vertices of A.

We now consider the irreducible polynomial $f \in \mathbb{K}[X, Y]$ and its absolute factors $f_1, \ldots, f_s \in \overline{\mathbb{K}}[X, Y]$. Observe that thanks to Lemma 4, we have that $P_{f_i} = P_{f_j}$ for every couple of indexes $i, j \in \{1, \ldots, s\}$.

We can then easily prove Proposition 3.

Proof. Suppose that f is not absolutely irreducible. Let $f_1, \ldots, f_s$ be the absolute factors of f. For what concerns the Newton polytopes, we have that

$$P_f = P_{f_1} + \cdots + P_{f_s} = s \cdot P_{f_1}.$$

Suppose in particular that the vertices of P_{f_1} are $\{(i_1, j_1), \ldots, (i_l, j_l)\}$. Then we have that the vertices of P_f are $\{(s \cdot i_1, s \cdot j_1), \ldots, (s \cdot i_l, s \cdot j_l)\}$. But then condition (C) is not satisfied. □

Corollary 7. *The number of absolute irreducible factors of a rationally irreducible polynomial $f(X, Y) \in \mathbb{K}[X, Y]$ divides* $\gcd(i_1, j_1, \ldots, i_l, j_l)$.

Proof. This is a consequence of the proof of Proposition 3. □

As all the arguments we used in this section extend to Newton polytopes in any number of variables we get:

Corollary 8. *Proposition 3 holds for a polynomial ring with any number of variables.* □

2. Evaluation of our irreducibility criterion

In Proposition 3, we established the validity of our criterion. In this section we address the natural question: does condition (C) happens frequently ?

When the polynomial f is dense, then the coordinates of the vertices of P_f are $(0, 0)$, $(n, 0)$, $(0, n)$, thus condition (C) is not satisfied and we cannot apply our test. However when f is sparse, in “most” cases, the Newton polytope is not the triangle of the previous situation and a direct use of Proposition 3 can quickly detect if f is absolutely irreducible.

We first provide time tables and statistic evidences of the efficiency of our criterion applied to a sparse polynomial $f(X, Y) \in \mathbb{Z}[X, Y]$. Then we consider its application to dense polynomials. In that case, modular computations are used to force a sparsity condition on a reduced polynomial modulo some prime p.

2.1. Statistics for a direct use of the test for sparse polynomials

To check the previous claim, we have constructed randomly 1000 polynomials of total degree n and applied our test. Our test is implemented in Magma and available at: http://www.math.univ-toulouse.fr/∼cheze/

The following table presents the obtained statistical results.

The entries are the degree n and a sparsity indicator *Prop*. When its value is $Prop = 1$ (respectively $Prop = 2$), each polynomial has about $n(n+1)/4$ (respectively $n(n+1)/6$) non-zero coefficients randomly chosen in $[-10^{12}; 10^{12}]$ and $n(n+1)/4$ (respectively $n(n+1)/3$) coefficients randomly chosen equal to zero. The outputs are: the number *Success* of absolute irreducible polynomials detected by our test, and the average running time T_{av} (in second).

n	*Prop*	*Success*	T_{av}
50	1	819	0.0134
50	2	943	0.0122
100	1	832	0.0787
200	1	849	0.6023
200	2	948	0.4432

This table shows that our test is well suited for sparse polynomials.

2.2. Irreducibility test with modular computations

Our aim is to construct a sparse polynomial associated to a dense polynomial, "breaking" its Newton polytope. For that purpose, we recall an easy corollary of Noether's irreducibility theorem. For a statement and some results about Noether's irreducibility theorem see e.g. Kaltofen (1995).

Proposition 9. *Let* $f(X,Y) \in \mathbb{Z}[X,Y]$ *and* $\overline{f}(X,Y) = f \mod p$, $\overline{f} \in \mathbb{F}_p[X,Y]$.
If tdeg $(f) =$ *tdeg* $(\overline{f})$ *and* $\overline{f}$ *is absolutely irreducible, then* f *is absolutely irreducible.* □

Now, even if f is dense, the idea is to choose p in order to force $\overline{f}$ to be sparse. Then we apply the test to $\overline{f}$ instead of applying it to f.

Let $a_1, \dots, a_r$ be the coefficients corresponding to the vertices of P_f and $L = [p_1, \dots, p_l]$ be the list of the primes dividing at least one of the a_i. Remark that:

$$\forall p_i \in L,\, P_f \neq P_{f \bmod p_i}.$$

Thus even when f is dense, if the coefficients $a_1, \dots, a_r$ are not all equal to 1, we can get polynomials $f \mod p_i$ such that $P_{f \bmod p_i}$ is not the triangle with vertices $(0,0)$, $(0,n)$, $(n,0)$. In Section 2.3, we will see that a linear change of coordinates permits to deal with the remaining case.

Example: $f(X,Y) = Y^3 + X^3 + 5X^2 + 3Y + 2$. Figure 1 clearly illustrates the effect of a reduction modulo $p = 2$.

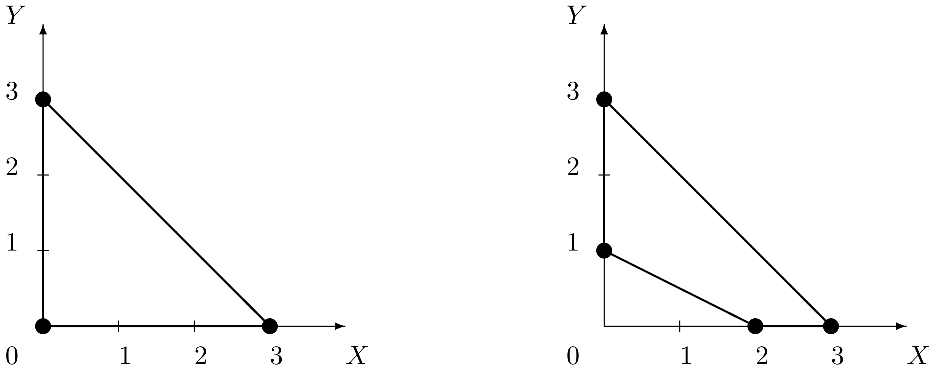

Fig. 1. Newton polytopes of f and $f \mod 2$

Therefore, thanks to Proposition 3 and Proposition 9, absolute irreducibility can be tested with a Las Vegas strategy (i.e. the output of the algorithm is *always* correct). However the output can be "I don't know". More precisely:
For each $p \in L$, test the absolute irreducibility of $\overline{f} \in \mathbb{F}_p[X, Y]$ with Proposition 3, and conclude with Proposition 9.

Newton-polytop-mod algorithm

```
Inputs: $f(X, Y) \in \mathbb{Z}[X, Y]$, irreducible in $\mathbb{Q}[X, Y]$.
Outputs: "$f$ is absolutely irreducible" or "I don't know".
 (1) Compute $P_f$ and the list $L$ of the primes dividing a coefficient corresponding to a
     vertex of $P_f$. Initialize test:=false: $i := 1$:
 (2) While(test=false) and ($i \leq |L|$) do $p := L[i]$;
     If $tdeg(f \mod p) = tdeg(f)$ then
           Compute $P_{f \mod p}$.
           If $f \mod p$ satisfies condition $(C)$ then
               If $f \mod p$ is irreducible in $\mathbb{F}_p[X, Y]$ then test:=true; End If;
           End If;   End If;   $i := i + 1$   End While:
 (3) If (test = true) then return "$f$ is absolutely irreducible" else return "I don't know"
     End If:
```

The following table shows that this algorithm is quite efficient. We constructed 1000 polynomials in $\mathbb{Z}[X, Y]$ of total degree n, with random integer coefficients in $[-10^{12}; 10^{12}]$. All these polynomials are dense. For each polynomial we test its absolute irreducibility with the previous algorithm. *Success* is the number of absolute irreducible polynomials detected with this algorithm. T_{av} (respectively T_{max}, T_{min}) is the average (respectively maximum, minimum) timing in second to perform one test.

n	*Success*	T_{av}	T_{max}	T_{min}
10	1000	0.0041	0.33	0
30	1000	0.0113	0.56	0
50	1000	0.0252	0.59	0.009
100	1000	0.1552	0.66	0.081
200	1000	1.7579	3.22	0.701

2.3. Modular computations and change of coordinates

A last task is to deal with polynomials whose coefficients are 0, 1 or -1 like $f(X,Y) = X^n + Y^n + 1$, because in that case the Newton polytope gives no information, even when one looks at the modular reduction $f \mod p$. The natural strategy is to perform a linear change of coordinates in order to obtain, after reduction, a polynomial satisfying condition (C). This is applied in the next algorithm.

Modular computation is performed in $\mathbb{F}_p$ where p is a prime between 2 and some value, here fixed to 101.

```
**Newton-Polytop-mod-chg-var algorithm**
Input: $f(X,Y) \in \mathbb{Z}[X,Y]$, irreducible in $\mathbb{Q}[X,Y]$.
Output: "$f$ is absolutely irreducible" or "I don't know".
  For each $p$ prime between 2 and 101 do:
    For $(a,b) \in \mathbb{F}_p^2$ do
      $f_{a,b}(X,Y) = f(X+a, Y+b) \mod p$;
      If tdeg $(f_{a,b})$ = tdeg $(f)$ then
        If $f_{a,b}$ satisfies condition $(C)$ then
          If $f_{a,b}$ is irreducible in $\mathbb{F}_p[X,Y]$ then return
  "$f$ is absolutely irreducible";
          End If; End If; End If;End If; End For; End For;
Return "I don't know".
```

This algorithm generalizes a test given by Ragot (2002) based on the following classical property.

Fact: *Let $f(X,Y) \in \mathbb{K}[X,Y]$ be an irreducible polynomial in $\mathbb{K}[X,Y]$. If there exists $(a,b) \in \mathbb{K}^2$ such that $f(a,b) = 0$ and $\dfrac{\partial f}{\partial X}(a,b) \neq 0$ or $\dfrac{\partial f}{\partial Y}(a,b) \neq 0$, then f is absolutely irreducible.*

Ragot's algorithm tests if $f \mod p$ has a simple root in $\mathbb{F}_p$. Remark that f has a simple root if and only if after a linear change of coordinates, which brings this root at the origin, the Newton polytope of f has at least one of the points $(1,0)$ and $(0,1)$ as vertex, while $(0,0)$ is not a vertex.

In that case, condition (C) is satisfied; thus Ragot's test is weaker than our test.

At http://www.mip.ups-tlse.fr/∼cheze/, we listed an example of polynomial for which absolute irreducibility is immediately detected by our algorithm reducing modulo $p = 2$, while Ragot's test needs to reduce and check iteratively for all primes until $p = 73$.

Let us remark that thanks to the following proposition, for $p \geq (n-1)^4$ our probabilistic test becomes deterministic.

Proposition 10 (Ragot (1997), Prop. 4.4.3 page 79)**.** *Let $f(X,Y) \in \mathbb{F}_p[X,Y]$ be an absolute irreducible polynomial of total degree n. If $p \geq (n-1)^4$ then f has simple roots in $\mathbb{F}_p$.* □

Indeed, if we have a simple root then after a change of coordinates we get a polynomial satisfying Ragot's test and thus satisfying condition (C). However, in practice, a probabilistic approach with a small prime is much faster.

We only considered the case of integer polynomials, however our tests can be extended to the case of polynomials with coefficients in a commutative ring. In this case, the computation modulo a prime number will be replaced by a computation modulo a prime ideal. The algorithms can also be extended to the case of polynomials with N variables, in which case the probability of success will increase with N. Indeed, there are more chances to obtain a gcd equal to 1 with more coordinates.

3. A toolbox for an absolute factorization algorithm

We aim to build a factorization algorithm by extending the analysis and strategy developed for the previous irreducibility test. We keep the notations introduced in Section 1 and specially in Lemma 4. A main task is to describe an algebraic extension $\mathbb{L} = \mathbb{Q}(\alpha)$ of $\mathbb{Q}$ which contains the coefficients of a factor f_1 of f.

This kind of strategy was already developed in the TKTD algorithm; TKTD is an acronym for Trager/ Kaltofen/Traverso/Dvornicich, (see Dvornicich and Traverso (1989), Kaltofen (1985) and Trager (1985)). The result of the TKTD algorithm is an algebraic extension $\mathbb{L}$ in which $f(X,Y)$ factors. Usually this extension is too big, that is to say: the degree extension of $\mathbb{L}$ is not minimal.

We aim to reach the same goal, obtain an algebraic extension in which $f(X,Y)$ is reducible, but the extension we will find is smaller, in fact minimal, and so more suitable for the computation of the factorization.

3.1. Algebraic extensions and primitive elements

We can describe the extension $\mathbb{L}$ of $\mathbb{Q}$ with a primitive element. Let us see that, generically, $\mathbb{L} = \mathbb{Q}[f_1(x_0, y_0)]$.

Lemma 11. *Let $f(X,Y) \in \mathbb{Z}[X,Y]$ be a rationally irreducible polynomial (i.e. over $\mathbb{Q}$) of degree n. Let $f_1(X,Y)$ be an absolute irreducible factor of $f(X,Y)$, $\deg f_1(X,Y) = m$. For almost all $(x_0, y_0) \in \mathbb{Z}^2$ we have $\mathbb{L} = \mathbb{Q}(f_1(x_0, y_0))$.*
More precisely, the following estimate on the probability holds:

$$\mathcal{P}\Big(\{(x_0,y_0) \in S^2 \mid \mathbb{L} = \mathbb{Q}(f_1(x_0,y_0))\}\Big) \geq 1 - \frac{n(s-1)}{2|S|} \qquad \text{with } s := n/m,$$

where S is a finite subset of $\mathbb{Z}$.

Proof. We denote by $a_{i,j}$ the coefficients of f_1, so $\mathbb{L} = \mathbb{Q}(a_{i,j})$. Let σ_l, $(1 \leq l \leq s)$ be s independent $\mathbb{Q}$-homomorphisms from $\mathbb{L}$ to $\mathbb{C}$.

Hence we have:

$$\forall u \neq v, \text{ there exists } (i,j) \text{ such that } \sigma_u(a_{i,j}) \neq \sigma_v(a_{i,j}). \qquad (*)$$

We consider $D(X,Y) = \prod_{u \neq v} \Big(\sum_{i,j} \big(\sigma_u - \sigma_v\big)(a_{i,j}) X^i Y^j \Big)$.

Property $(*)$ implies that $D(X,Y) \neq 0$. Then there exists $(x_0, y_0) \in \mathbb{Z}^2$ such that $D(x_0, y_0) \neq 0$. This means: for all $u \neq v$, $\sigma_u\big(f_1(x_0,y_0)\big) \neq \sigma_v\big(f_1(x_0,y_0)\big)$. Thus $f_1(x_0,y_0)$ is a primitive element of $\mathbb{L}$ and this gives the desired result.

The probability statement is a direct consequence of Zippel-Schwartz's lemma, applied to $D(X,Y)$, whose degree is bounded by $(ms(s-1))/2 = (n(s-1))/2$. □

Remark that the polynomial $D(X,Y)$ appearing in the previous proof is connected to another interesting polynomial: the discriminant, with respect to Z, of the 3-variate polynomial $F(X,Y,Z) = \prod_{j=1}^{s}(Z - f_j(X,Y))$. The discriminant with respect to Z of $F(X,Y,Z)$ is the square of $D(X,Y)$. F has coefficients in $\mathbb{Z}$ because its coefficients are invariant when we permute the f_j.

3.2. Number fields and p-adic numbers

Lemma 12. *Let $M(T) \in \mathbb{Z}[T]$ be a polynomial and p a prime number such that p divides $M(0)$, p does not divide the discriminant of $q(T)$ and and $p > \deg(M)$.*
Then there exists a root in $\mathbb{Q}_p$ of $M(T)$, considered as a polynomial in $\mathbb{Q}_p[T]$.

This lemma allows us to consider a number field $\mathbb{Q}(\alpha)$ as a subfield of $\mathbb{Q}_p$, for a well-chosen prime p. Indeed, if $q(T)$ is the minimal polynomial of α, then with a big enough integer c we can find a prime number p such that the polynomial $q(T+c)$ satisfies the hypothesis of Lemma 12. Thus we can consider $\alpha + c$ in $\mathbb{Q}_p$, then $\mathbb{Q}(\alpha) \subset \mathbb{Q}_p$. During our algorithm we are going to factorize $f(X,Y) \mod p$. We can consider this factorization as an "approximate" factorization of f in $\mathbb{Q}(\alpha)$ with the p-adic norm. Then this factorization gives information about the absolute factorization.

Proof. Since $M(0) = 0 \mod p$ and $p \nmid disc(q(T))$, 0 is also a root of $M_1(T) = \frac{M(T)}{\gcd(M(T), M'(T))}$ in $\mathbb{F}_p$. As $p > \deg(M)$ we have $M_1'(0) \neq 0$ in $\mathbb{F}_p$ and we can lift this root in $\mathbb{Q}_p$ by Hensel's liftings. This gives a root of $M_1(T)$ in $\mathbb{Q}_p$, thus a root of $M(T)$ in $\mathbb{Q}_p$. □

3.3. Choice of p

Lemma 13. *Let $f(X,Y) \in \mathbb{Z}[X,Y]$, $\deg f(X,Y) \geq 1$ and let $\mathcal{B}$ be a positive integer. There exist $(x_0,y_0) \in \mathbb{Z}^2$ and $p \in \mathbb{Z}$ such that p divides $f(x_0,y_0)$ and p does not divide $\mathcal{B}$.*

Proof. We can reduce to the case of one variable and use the classical argument of Dirichlet for proving that the set of prime numbers is infinite.

Consider the polynomial $f(X) \in \mathbb{Z}[X]$, $\deg f \geq 1$. Consider x_1 such that the constant term $c := f(x_1)$ is not zero.

Set $\tilde{f}(X) = f(X - x_1)$, so c is the constant term of $\tilde{f}(X)$. Consider $\tilde{f}(c\mathcal{B}X) = c(1 + \mathcal{B}Xq(X))$, where $q(X) \in \mathbb{Z}[X]$ is not zero (otherwise $deg f < 1$). We can find $x_0 \in \mathbb{Z}$, $x_0 \neq 0$ such that $\mathcal{B}x_0q(x_0) \neq 0$. Then, a prime p dividing $1 + \mathcal{B}x_0q(x_0)$ does not divide $\mathcal{B}$ and we are done. □

Definition 14. We say that the prime integer p gives a *bad reduction* of $f(X,Y)$ if the number of absolute factors of $f(X,Y) \mod p$ differs from the number of absolute factors of $f(X,Y)$. □

Proposition 15. *Let $f(X,Y)$ be a rationally irreducible polynomial, monic in Y. Then there is a finite number of prime integers p giving a bad reduction of $f(X,Y)$.*

Furthermore, if $d(X) = disc_Y(f(X,Y))$, $d_1(X) =$ square-free part of $d(X)$ and $D = disc_X(d_1(X))$, the set of prime integers p giving a bad reduction of f is contained in the set of prime divisors of D.

Proof. The finiteness of the set of p giving bad reductions comes from a theorem of Noether (1922). For the characterization using D, we can say with other words that $f(X, Y)$ has a good reduction mod$p$ if $d(X)$ and $d(X) \mod p$ have the same number of distinct roots. For the proof of this fact, see Trager (1989). Finally, for another proof, see Zannier (1997). □

3.4. Recognition strategy

We assume that we chose a good prime p, such that tdeg $(f) =$ tdeg $(f \mod p)$ and $f \mod p$ factors as $f(X, Y) = F^{(1)}(X, Y) \cdot G^{(1)}(X, Y) \mod p$ where $F^{(1)}$ is exactly the image $\mod p$ of an absolute factor f_1 of f.

In order to find the splitting field of $f(x_0, Y)$, relying on Proposition 11, we need to compute $q(T)$, the minimal polynomial with integer coefficients of $\alpha := f_1(x_0, y_0)$.

Starting from a factorization $f(x_0, Y) = F^{(1)}(x_0, Y)G^{(1)}(x_0, Y) \mod p$, we lift it through Hensel Lifting to the level of accuracy p^λ. We then consider the p-adic approximation $\overline{\alpha} := F^{(\lambda)}(x_0, y_0)$ of α. Using a “big enough” level of accuracy λ, we can compute the minimal polynomial of α from $\overline{\alpha}$.

Proposition 16. *Consider $\overline{\alpha} = F^{(\lambda)}(x_0, y_0)$, $0 \leq \overline{\alpha} \leq p^\lambda - 1$ constructed above, a positive integer Q bounding the size of the coefficients of $q(T)$, $Q \geq \|q(T)\|_\infty$, and a positive integer $\lambda \geq \log_p(2^{s^2/2}(s+1)^s Q^{2s})$.*

Then we can compute the minimal polynomial $q(T)$ of α using the LLL algorithm on an integer lattice whose basis is given using $\overline{\alpha}$ and p^λ.

Proof. We apply the same construction of von zur Gathen and Gerhard (2003, Section 16.4) for detecting rational factors of univariate polynomials.

We consider the polynomials

$$\{T^i(T - \overline{\alpha}) | i = 0, \ldots, s-1\} \cup \{p^\lambda\}.$$

We write as usual

$$T^i(T - \overline{\alpha}) = T^{i+1} - \overline{\alpha} T^i = \sum_{j=0}^{s} t_j T^j,$$

where, in this case, $t_j \neq 0$ for $j \in \{i+1, i\}$ and $t_j = 0$ otherwise. Then the associated vector for the polynomial $T^i(T - \overline{\alpha})$ is

$$b_i = (t_s, \ldots, t_0).$$

For the constant polynomial p^λ, we associate the vector $\tilde{b} = (0, \ldots, 0, p^\lambda)$. We can construct the $(s+1) \times (s+1)$ matrix B whose columns are the b_i, $i = 0, \ldots, s-1$ and $\tilde{b}$:

$$B = \begin{bmatrix} 1 & 0 & 0 & 0 \dots & 0 & 0 & 0 \\ -\overline{\alpha} & 1 & 0 & 0 \dots & 0 & 0 & 0 \\ 0 & -\overline{\alpha} & 1 & 0 \dots & 0 & 0 & 0 \\ 0 & 0 & -\overline{\alpha} & 1 \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 \dots & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \dots & -\overline{\alpha} & 1 & 0 \\ 0 & 0 & 0 & 0 \dots & 0 & -\overline{\alpha} & p^{\lambda} \end{bmatrix}$$

If we consider a point g of the integer lattice $\bigwedge(B) \subseteq \mathbb{R}^{s+1}$ generated by the columns of the matrix B, we can write its components with respect to the standard basis of $\mathbb{R}^{s+1}$

$$g = \sum_{i=0}^{s-1} g_i b_i + \tilde{g}\tilde{b} = (g_{s-1}, g_{s-2} - \overline{\alpha} g_{s-1}, \dots, g_0 - \overline{\alpha} g_1, \tilde{g} p^{\lambda} - \overline{\alpha} g_0)$$

and associate a polynomial:

$$G(T) = g_{s-1}T^s + (g_{s-2} - \overline{\alpha} g_{s-1})T^{s-1} + \cdots + (g_0 - \overline{\alpha} g_1)T + \tilde{g}p^{\lambda} - \overline{\alpha} g_0 =$$

$$= S(T)(T - \overline{\alpha}) + \tilde{g}p^{\lambda} \quad \text{with} \quad S(T) = \sum_{i=0}^{s-1} g_i T^i.$$

So if $g \in \bigwedge(B)$, the associated polynomial $G(T)$ has degree $\leq s$ and it is divisible by $(T - \overline{\alpha})$ modulo p^{γ}.

The *vice versa* holds:
If $G(T)$ is a polynomial of degree at most s and $G(T) \mod p^{\lambda}$ is divisible by $(T - \overline{\alpha})$, then we can write

$$G(T) = S^*(T)(T - \overline{\alpha}) + R^*(T)p^{\gamma} \quad \text{with } \deg S^*(T) \leq s - 1 \text{ and } \deg R^*(T) \leq s.$$

Using Euclidean division, we obtain $R^*(T) = S^{**}(T)(T - \overline{\alpha}) + Rp^{\gamma}$ with $\deg S^{**} \leq s - 1$ and R a costant. We define $S(T) := S^*(T) + p^{\gamma} S^{**}(T)$. We then have that

$$G(T) = S(T)(T - \overline{\alpha}) + Rp^{\gamma},$$

that is, $G(T)$ can be written as a point of the lattice $\bigwedge(B)$.

So if we consider the matrix B and we apply the LLL algorithm, we obtain as first vector of the reduced basis a "short"vector representing a polynomial $G(T)$ with "small" norm such that $G(T)$ has degree s and $G(T) \mod p^{\lambda}$ is divisible by $(T - \overline{\alpha})$. Using the hypothesis $\lambda \geq \log_p(2^{s^2/2}(s+1)^s Q^{2s})$ we can apply von zur Gathen and Gerhard (2003, Lemma 16.20): we then have that $q(T)$ and $G(T)$ have a non-constant gcd. But since $q(T)$ is irreducible and $\deg q(T) = \deg G(T)$, we have that $q(T) = G(T)$. □

To establish the level of accuracy λ, we need a bound on the size of the coefficients of the minimal polynomial of α, $q(T)$. Remember that

$$q(T) = \prod_{i=1}^{s}(T - \alpha_i) = T^s + \sigma_1(\tilde{\alpha}) + \cdots + \sigma_{s-1}(\tilde{\alpha})T + \sigma_s(\tilde{\alpha}),$$

where $\sigma_i(\tilde{\alpha})$ is the i-th symmetric function in the $\alpha = \alpha_1, \alpha_2, \ldots, \alpha_s$.

Observe that

$$|\sigma_k(\tilde{\alpha})| \leq \sum_{\tau \in \mathcal{S}_k} |\alpha_{\tau(1)}| \cdots |\alpha_{\tau(k)}| \leq \sum_{\tau \in \mathcal{S}_k} \prod_{j=1}^{m} |y_j^{\tau(1)}| \cdots \prod_{j=1}^{m} |y_j^{\tau(k)}|,$$

where $f_l(x_0, Y) = \prod_{j=1}^{m}(Y - y_j^{(l)})$ and $f(x_0, Y) = \prod_{i=1}^{s} f_l(x_0, Y)$.

As a bound on the coefficients of $f(x_0, Y)$ gives a bound on the $y_j^{(l)}$ (von zur Gathen and Gerhard, 2003), a bound on the coefficients of $f(x_0, Y)$ gives a bound for $\|q(T)\|_\infty$.

In practice, for "early detection", we rely on Proposition 16 replacing Q by

$$Q_1 = \|f(x_0, Y)\|_\infty.$$

Remark 17. We have to face two problems concerning the algorithm for the Hensel Lifting and the level of accuracy λ:

(1) *Leading coefficient problem*: Since $f(X, Y) \in \mathbb{Z}[X, Y]$ is not assumed to be monic, its modular factors too may not be monic; then the classical Hensel lifting of the modular factorization of $f(X, Y)$ is not necessarily unique (Geddes et al., 1992, Corollary of Theorem 6.2).

(2) In practical use of this construction of the minimal polynomial of α, we will avoid to lift the factorization until the level λ of Proposition 16 (this bound is usually very pessimistic). However, in this way we are not sure that the polynomial $G(T)$ is actually $q(T)$. We then need a quick method to check if we found a good candidate to define the field extension or if we have to lift the factorization to a higher level of accuracy.

Consider $f(x_0, Y) = \sum_{i=0}^{n} \phi_i Y^i$.

For what concerns the *leading coefficient problem*, we can simply consider the "modified" linear Hensel Lifting (Geddes et al., 1992, Algorithm 6.1). In this way we can lift the factorization modulo p, but the coefficients involved in the computations are bigger, since actually we lift a factorization of $\phi_n \cdot f(x_0, Y)$, obtaining a factor that we call $\tilde{F}^{(\lambda)}(Y)$.

For what concerns the second problem, we have to understand how the roots of a factor of $f(x_0, Y)$ are in connection with the coefficients of $q(T)$ and $\tilde{f}_1(Y)$, that is the factor of $f(x_0, Y)$ that we obtain after the "modified" Hensel Lifting. We call q_s the leading coefficient of the polynomial $q(T)$.

If $f_1(x_0, Y)$ is the factor of $f(x_0, Y)$ we are looking for, then the product of its roots is simply $\beta := (-1)^{\deg \tilde{f}_1(Y)} \tilde{f}_1(y_0)/\phi_n$.

Then the product of the conjugated of β is simply $q(0)/q_s$, but this is also the product of all the roots of $f(x_0, Y)$. So we have the following relation $\frac{q(0)}{q_s} = (-1)^s \frac{f(x_0, y_0)}{\phi_n}$.

When we apply the LLL algorithm to $\bigwedge(B)$ we can then proceed as follows: if the obtained polynomial $G(T)$ satisfies

$$\frac{G(0)}{G_s} = (-1)^s \frac{f(x_0, y_0)}{\phi_n} \qquad \text{with } G_s \text{ leading coefficient of } G(T) \qquad (\star)$$

then we will try to factor $f(x_0, Y)$ in the algebraic extension defined by $G(T)$, that is $\mathbb{Q}[T]/G(T)$. If $G(T)$ does not satisfy $(\star)$, then we have to rise the level of approximation of the Hensel lifting and then apply again LLL to the new lattice and test again.

In this way we have a necessary condition that can help us to recognize the minimal polynomial of α.

4. Absolute factorization algorithm

We use the results and methods of the previous section to compute an absolute factor f_1 of f (i.e. a representation of the field $\mathbb{L}$ of its coefficient and the coefficients).

To ease the presentation, we rely on the practical evidence that for random integer value x_0, $f(x_0, Y)$ is irreducible. In Section 4.2 we will present a variant using a weaker condition.

Abs-Fac algorithm

Input: $f(X,Y) \in \mathbb{Z}[X,Y]$, irreducible in $\mathbb{Q}[X,Y]$ of degree n, a finite subset S of $\mathbb{Z}^2$.

Output: $q(T) \in \mathbb{Q}[T]$ minimal polynomial of α defining the minimal algebraic extension $\mathbb{L} = \mathbb{Q}(\alpha) = \mathbb{Q}[T]/q(T)$ and $f_1(X,Y) \in \mathbb{L}[X,Y]$ an absolute irreducible factor of f, or "I don't know".

Preprocessing: Choose $(x_0, y_0) \in S^2$, such that $f(x_0, Y)$ is irreducible. If all of the points were used, then return "I don't know".

(1) Choose a prime p dividing $f(x_0, y_0)$ such that tdeg $(f \mod p) =$ tdeg (f).

(2) Factorize f in $\mathbb{F}_p[X,Y]$.

If $f \mod p$ is irreducible and satisfies an absolute irreducibility test then Return "f is absolutely irreducible", $f_1 := f$ and $q(T) := T$.

If $f \mod p$ is irreducible and not absolutely irreducible then go back to the Preprocessing step (choosing a point (x_0, y_0) not yet used and a different prime p).

Else $f(X,Y) = F^{(1)}(X,Y) \cdot G^{(1)}(X,Y) \mod p$ where $F^{(1)}$ is one of the irreducible factors in $\mathbb{F}_p[X,Y]$ with smallest degree m, check that $s := \frac{\text{tdeg }(f)}{m}$ is an integer else go back to the Preprocessing step (choosing a point (x_0, y_0) not yet used and a different prime p).

(3) Lift the factorization to $f(x_0, Y) = F^{(\lambda)}(x_0, Y) G^{(\lambda)}(x_0, Y) \mod p^{\lambda}$; λ is chosen according to Proposition 16 and Remark 17.

(4) Define $\overline{\alpha} := F^{(\lambda)}(x_0, y_0) \in \mathbb{Z}/p^{\lambda}\mathbb{Z}$. Find, using the lattice described in Section 3.4 and the LLL algorithm, the polynomial $q(T)$. If $q(T)$ does not satisfy $(\star)$ or it is not irreducible, go back to step (3) and double λ.

(5) Denote by α a root of $q(T)$ then factorize $f(x_0, Y)$ in $\mathbb{Q}(\alpha)[Y] = \mathbb{L}[Y]$ and denote by $F_1(x_0, Y)$ a factor with degree m and with $F_1(x_0, y_0) = \alpha$.

If we do not find such a factor, then go back to the Preprocessing step (choosing a point (x_0, y_0) not yet used and a different prime p).

(6) Perform m times X-adic Hensel liftings on $f(x_0, Y) = F_1(x_0, Y) F_2(x_0, Y)$ to determine a candidate for $f_1(X,Y)$ in $\mathbb{L}[X,Y]$ and check that it divides $f(X,Y)$. Else go back to the Preprocessing step (choosing a point (x_0, y_0) not yet used and a different prime p).

Return $q(T)$ and $f_1(X,Y)$.

Proposition 18. *The algorithm gives a correct answer.*

Proof. Since it is a Las Vegas algorithm, this algorithm is probably fast and always correct but the answer can be “I don't know”. So we just have to check that a given positive answer is correct.

The starting point of the proposed algorithm, as in the irreducibility test, is to determine a prime p such that the reduction modulo p kills the evaluation of f on an integer point (x_0, y_0). Then the constant term of the minimal polynomial of $\alpha := f_1(x_0, y_0)$ vanishes modulo p. Such a p is easily found. However we rely on randomness to expect with a good probability that $\mathbb{L} = \mathbb{Q}(\alpha)$ and that f has good reduction modulo p (using Proposition 15 and Lemma 12).

In the algorithm described above, we inserted some checks and a loop to change p if it is an “unlucky” choice. The algorithm can be made deterministic (but less efficient) by considering a large testing set for (x_0, y_0) and take p not dividing a huge constant $\mathcal{B}$ computed a la Trager, to avoid bad reduction. We would be able to do this thanks to Lemma 13.

The output of the algorithm, the factor f_1, is irreducible in $\mathbb{L}[X, Y]$. Indeed, $f_1(x_0, Y) = F_1(x_0, Y)$ and $F_1(x_0, Y)$ is irreducible in $\mathbb{L}[Y]$ because of the irreducibility of $f(x_0, Y)$ in the Preprocessing Step. Furthermore, the extension $\mathbb{L}$ is minimal. Indeed, at the end of the algorithm we have $\deg_Y f_1 = m$, $\deg q = s$ and $s.m = n$, see the definition of s in Step 2. $\square$

Remark: f_1 is irreducible modulo p and f_1 modulo p generically satisfies condition (C), so Proposition 3 guarantees the absolute irreducibility of f_1 in $\mathbb{L}[X, Y]$.

Instead of costructing the absolute factor $f_1(X, Y) \in \mathbb{L}[X, Y]$ using univariate factorization and X-adic Hensel lifting, one can substitute Steps 5 and 6 of the Abs-Fact Algorithm with the computation of the factorization of $f(X, Y)$ in $\mathbb{L}[X, Y]$. This is possible, for instance, using the software Magma (Bosma et al., 1997). In this way in the Preprocessing Step we do not need to look for (x_0, y_0) ensuring the irreducibility of $f(x_0, Y)$.
Anyhow, if such a routine for bivariate factorization in $\mathbb{L}[X, Y]$ is not available, we can weaken the request on the point (x_0, y_0) as described in Section 4.2.

4.1. Parallel version of the Algorithm

In step (5) of the Abs-Fac Algorithm we perform a factorization of $f(x_0, Y)$ in the polynomial ring $\mathbb{L}[Y]$. Then in Step (6) we use Hensel liftings to reconstruct the factor f_1. If we use parallel calculus in these steps, we can perform $(m+1)$ Lagrange interpolations to reconstruct the factor f_1. We have to assume that in the factorization of $f(x_0, Y)$ in $\mathbb{L}[Y]$ there is only one factor of degree m. This is not always verified, for instance if the extension $\mathbb{L}$ is normal we may have several factors of the same degree m.

We write the absolute factor f_1 as

$$f_1(X, Y) = Y^m + \sum_{k=0}^{m-1} \sum_{i+j=k} a_{i,j}^{(1)} X^i Y^j = Y^m + \sum_{j=0}^{m-1} b_j(\alpha, X) Y^j,$$

where $b_j(Z, X) \in \mathbb{Q}[Z, X]$ of degree $\leq m - j$ and α is a root of the polynomial $q(T)$ found in step (4).

We then want to find the polynomials $b_j(\alpha, X)$.

We substitute steps (5) and (6) with the following procedure:

(5bis) Denote by α a root of $q(T)$ (i.e. the command RootOf in Maple).
Choose points $x_1, \dots, x_m \in \mathbb{Z}$, $x_i \neq x_0$ for $i = 1, \dots, m$ such that $f(x_i, Y)$ is rationally irreducible.
Compute the factorization of $f(x_i, Y)$ in $\mathbb{L}[Y]$ and choose $F_{1,0}(Y)$ from the factorization of $f(x_0, Y)$ as in step (5) of the algorithm and $F_{1,j}(Y)$ a factor of minimal degree m in the factorization of $f(x_j, Y)$.

(6bis) Write $F_{1,j}(Y)$ as follows

$$F_{1,j} = \sum_{i=0}^{m} \gamma_{i,j} Y^j \text{ with } \gamma_j \in \mathbb{L}.$$

We then construct the polynomials $b_j(\alpha, X)$ of degree j using Lagrange interpolation (Burden and Faires, 1993, Section 3.1) on the set of nodes $\gamma_{0,j}, \dots, \gamma_{j,j}$. In this way we determine a candidate for $f_1(X, Y)$ in $\mathbb{L}[X, Y]$. We check that it divides $f(X, Y)$. Else go back to the Preprocessing step (choosing a point (x_0, y_0) not yet used and a different prime p).

The advantage of steps (5bis) and (6bis) is that in this way this part of the algorithm can be naturally parallelized and do not saturate the memory.

4.2. Hilbert's Irreducibility Theorem

In the preprocessing step we check that $f(x_0, Y)$ is irreducible. This situation happens very often in practice. With a more theoretical point of view, we know that there exists an infinite number of $x_0 \in \mathbb{Z}$ such that $f(x_0, Y)$ is irreducible, thanks to Hilbert's irreducibility theorem. There exists bounds for this theorem but unfortunately they are very big, see Dèbes and Walkowiak (2008).

Here we now use a weaker condition on the choice of (x_0, y_0) that allows us to reconstruct the factor $f_1(X, Y)$ even if $f(x_0, Y)$ is not rationally irreducible.

Choose an integer point $(x_0, y_0) \in \mathbb{Z}^2$ such that x_0 is not a root of the polynomial $\Delta(X) = \mathrm{disc}_Y(f(X, Y))$ and choose an integer p such that $\Delta(x_0) \mod p \neq 0$. With this choice of (x_0, y_0) we are sure that the univariate polynomial $f(x_0, Y)$ has no multiple roots in $\mathbb{Q}$ nor in $\mathbb{F}_p$.

We do not assume that $f(x_0, Y)$ is rationally irreducible. We computed the factorization $\mod p$

$$f(X, Y) = F(X, Y) \cdot G(X, Y) \in \mathbb{F}_p[X, Y] \quad \deg F = m.$$

Thanks to the choice of p as in step (1) of the algorithm, $F(X, Y)$ should be equal mod p to the researched absolute factor $f_1(X, Y)$ of f.

After applying step (5), we get the following factorization

$$f(x_0, Y) = \psi_1(Y) \cdots \psi_r(Y) \in \mathbb{Q}(\alpha)[Y] \tag{2}$$

and need to find the set of indexes $I \subseteq \{1, \dots, r\}$ such that

$$\prod_{i \in I} \psi_i(Y) = f_1(x_0, Y). \tag{3}$$

We reduce $\mod p$ the equalities (2) and (3). We obtain that $j \in I$ if and only if ψ_j mod p divides $F(x_0, Y) \mod p$.

5. Examples and practical complexity

We tested our algorithm on several examples, using (probably non-optimal) routines implemented in Maple 10.

We focused on the construction of the minimal polynomial $q(T)$ of α, that is on the construction of the splitting field $\mathbb{Q}(\alpha)$; in fact the last part of the algorithm, the factorization using coefficients in $\mathbb{Q}(\alpha)$, depends strongly on the used software: in some computer algebra systems one can find routines for multivariate factorization over algebraic extension of the rationals, such as in Magma (Bosma et al., 1997), or, alternatively, one can use X-adic Hensel lifting or Lagrange interpolation as described before.

The procedures, data and Maple files of several examples are available at http://math.unice.fr/∼cbertone/

Here we list some remarks about both the strong and the weak points of our algorithm arising from the computed examples.

- In general the algorithm is quite fast: it took around 30 sec (factorization $\bmod p$, Hensel lifting, construction of the minimal polynomial) to compute the polynomial $q(T)$ starting from a polynomial of degree 200, with 10 absolute factors of degree 20 each.
- If possible, it seems to be a good idea to choose a "small" prime p (in this way we can gain some time in the $\bmod p$-factorization). If the integers dividing $f(x_0, y_0)$ are quite big, it may be better to go back to the preprocessing step.
- On examples of high degree, most of the time is spent for the construction of the minimal polynomial from the approximation $\overline{\alpha}$. In our tests, we used the LLL function of Maple, but we may speed up this part of the computation using more performing algorithms for LLL (for example, see Nguên and Stehlé (2005) and Schnorr (2006)).
- For the computation of the p-adic Hensel Lifting, we have implemented a small procedure in Maple, both for the linear and the quadratic one, which can deal also with non-monic polynomials (von zur Gathen and Gerhard, 2003, Algorithm 6.1).

Benchmark

We consider random polynomials $g_1 \in \mathbb{Q}[x, y, z]$ and $g_2 \in \mathbb{Q}[z]$, of degrees d_1 and d_2 resp. both rationally irreducible. We compute $f(X, Y) = Res_z(g_1, g_2)$. In this way we obtain an irreducible polynomial $f(X, Y) \in \mathbb{Q}[x, y]$, monic in y, of degree $d_1 \cdot d_2$ with d_2 absolute irreducible factors each of degree d_1.

The polynomials g_1 and g_2 used are listed in the file "Polynomials.mws".

Here we summarize the time needed to obtain $q(T)$, the minimal rational polynomial of α, such that the absolute factors of $f(X, Y)$ are in $\mathbb{L}[x, y]$, $\mathbb{L} = \mathbb{Q}(\alpha) = \mathbb{Q}[T]/q(T)$ and we made a few remarks about the strategy one may adopt (for instance the choice of the prime).

In almost all of the examples, we compute the Hensel lifting both with the linear and the quadratic algorithm, this is why we always chose as level of accuracy a power of 2.

Example 19, we computed the factor $f_1(X, Y)$ using Lagrange Interpolation.

To repeat the examples, one need to change at the beginning of each Maple file the location of the file "proc.txt", in which there are (non-optimal) implementations for linear and quadratic Hensel Lifting (for non monic polynomials) and a procedure to compute the minimal polynomial of a p-adic approximation of α using the LLL algorithm.

The names of kind "Example1.2.mws" refer to the Maple files on the website.

Example 19. $f(X, Y)$ rational irreducible polynomial of degree 50 with 5 absolute factors of degree 10.

We need 1.5 sec to construct the example and factor $f(0, 0)$. We construct the minimal polynomial defining the field extension for 2 different choices of p.

Example1.1.mws: we choose $p = 11$.

- Time to factor $f(X, Y) \mod p$: 0.131 sec.

The estimation of the level of accuracy that ensures the correct computation of $q(T)$ is in this case 338; we choose to lift the factorization to the level p^{256}.

- Time to lift the factorization $f(0, Y) = g_1(0, Y)g_2(0, Y) \mod p$ to a factorization mod p^{256}, using:

Linear Hensel Lifting: less than 1 sec
Quadratic Hensel Lifting: less than 0.07 sec.

- Time to find the minimal polynomial of α through its approximation $\mod p^{256}$ using *LLL*: 0.22 sec.

We can complete the algorithm using steps (5bis) and (6bis):
we choose 10 nodes $x_1 \ldots, x_{10}$ randomly and factor the polynomials $f(x_j, Y)$ in $\mathbb{Q}(\alpha)[Y]$; using parallel calculus, these factorizations take less than 4 minutes, since the longest of them takes about 219 sec. Then we use Lagrange Interpolation and obtain $f_1(X, Y)$.

Example1.3.mws: if we use the software *Pari GP*, applying the function *polred()* to the obtained polynomial $q(T)$, we get $q_1(Z)$ which defines the same algebraic extension as $q(T)$ but has smaller coefficients. In this way, the factorization of $f(0, Y)$ in $\mathbb{Q}(\alpha)$ took only 8 sec, but the computation of the polynomial $q_1(Z)$ in Pari GP took more than 360 sec! □

Example 20. $f(X, Y)$ rational irreducible polynomial of degree 400 with 20 absolute factors of degree 20.

We need around 1260 sec to construct the example and factor $f(0, 0)$.

Example6.1.mws: we choose $p = 53259165137$.

- Time to factor $f(X, Y) \mod p$: 1924 sec.

The estimation of the level of accuracy that ensures the correct computation of $q(T)$ is in this case 398; we choose to lift the factorization to the level p^{256}.

- Time to lift the factorization $f(0, Y) = g_1(0, Y)g_2(0, Y) \mod p$ to a factorization mod p^{256}, using

Linear Hensel Lifting: less than 365 sec
Quadratic Hensel Lifting: less than 39 sec.

- Time to find the minimal polynomial of α through its approximation $\mod p^{256}$ using *LLL*: 1024 sec.

In order to compare the time needed for the construction of $q(T)$ computing modulo a "small" prime, we considered also the case with $p = 89$ dividing $f(-1, 0)$. In this case we obtained (Example6.2.mws):

- Time to factor $f(X, Y) \mod p$: 127 sec.

The estimation of the level of accuracy that ensures the correct computation of $q(T)$ is in this case 2194; we choose to lift the factorization to the level p^{1024}.

• Time to lift the factorization $f(0,Y) = g_1(0,Y)g_2(0,Y) \mod p$ to a factorization mod p^{1024}, using
Linear Hensel Lifting: 737 sec
Quadratic Hensel Lifting: 24 sec.

• Time to find the minimal polynomial of α through its approximation $\mod p^{1024}$ using LLL: 520 sec. □

For the details of other examples, see http://math.unice.fr/∼cbertone/

In the following table we resume the timings of a few more examples.

- n = tdeg (f), s=number of absolute factors of f, $m = n/s$=degree of an absolute factor of f;
- p= prime integer, λ = level of accuracy of Proposition 16, $\tilde{\lambda}$ =chosen level of accuracy;
- T_1 = time to factor $f(X,Y) \mod p$, T_2 =time to lift the factorization to $p^{\tilde{\lambda}}$, T_3 =time to find the minimal polynomial of α.

Example	n	s	m	p	λ	$\tilde{\lambda}$	T_1	T_2	T_3
Example 1.1	50	5	10	11	338	256	0.13 s	0.07 s	0.22 s
Example 1.2	50	5	10	307	141	128	0.13 s	0.08 s	0.4 s
Example 2.1	100	10	10	7	1105	512	3.4 s	0.3 s	2.25 s
Example 2.2	100	10	10	655379	160	128	6.2	0.4 s	5.7 s
Example 3.1	150	15	10	7	2246	1024	10 s	1.08 s	21 s
Example 4.1	200	10	20	47	853	512	33 s	2.8 s	14 s
Example 4.2	200	10	20	114041	282	256	128 s	3.8 s	30 s
Example 5	200	20	10	7682833	457	256	68 s	3.8 s	220 s
Example 6.1	400	20	20	53259165137	398	256	1924 s	39 s	1024 s
Example 6.2	400	20	20	127	2194	1024	127 s	24 s	520 s
Example 7	100	20	5	7	3029	2048	0.64 s	1.25 s	205 s

6. Conclusion

In this paper we have presented a new approach to absolute factorization improving the use of classical tools, in particular the TKTD algorithm and LLL algorithm.

In fact, we have refined the main idea of the TKTD algorithm (Dvornicich and Traverso (1989), Kaltofen (1985), Trager (1985)), because we construct a "small" algebraic extension field in which the polynomial $f(X,Y)$ splits. However the degree of the extension constructed by our algorithm is minimal, i. e. the number of absolute factors. In the TKTD algorithm the degree of the used extension is the degree of the polynomial $f(X,Y)$.

Furthermore, we use the LLL algorithm in a new way to define the field extension, while its classical applications are on the coefficients of a univariate rational polynomial in order to factor it (Lenstra et al., 1982), or on the exponents (see van Hoeij (2002) and Chèze (2004a)).

In our application, LLL is used on a lattice defined by $s+1$ vectors, where s is the number of absolute factors of the polynomial, which is smaller than the degree of the polynomial to factor. That is why in our algorithm the use of LLL is not a bottleneck.

Nevertheless, we may improve the fastness of the computations using, if it will be available in the future, a fast LLL (see Nguên and Stehlé (2005) and Schnorr (2006)) and a good implementation of the Polred algorithm (Cohen and Diaz y Diaz, 1991), which allows a better presentation of the algebraic field extension. Also the part concerning the construction of the absolute factor using X-adic Hensel lifting can be improved using an efficient strategy as the one described in Chèze and Lecerf (2007).

Our Maple prototype was able to construct the rational algebraic extension containing the coefficients of an absolute factor of a polynomial of high degree (up to 400), which were so far out of reach of all other absolute factorization algorithm; furthermore our construction is very fast on polynomials of middle degrees (about 100).

A complete and efficient implementation of our algorithm will also need good p-adic and X-adic Hensel liftings. We expect, in a near future, that the library Mathemagix (Mathemagix, 2009) will provide optimized implementations of these routines. Another point to improve is the parallel version of the algorithm, in order to be able to deal also with normal extensions of $\mathbb{Q}$. With these improvements we think that a complete implementation of our algorithm can be challenging with other absolute factorization algorithms such as Chèze and Lecerf (2007) and Chèze (2004a).

We would like to explore in the near future the possibility to use a "LLL technique" to directly recognize all of the algebraic coefficients of an absolute factor; in particular, we would like to test if such a technique may be more efficient than the X-adic Hensel lifting or the use of Lagrange interpolation on several univariate factorization in $\mathbb{Q}(\alpha)$.

Another related direction of research that we will soon explore, is extending some of these techniques to the decomposition of affine curves in dimension 3 or more.

Acknowledgments

The authors would like to thank Grégoire Lecerf for useful discussions and valuable suggestions concerning this paper and the anonymous referees for their helpful comments and remarks.

References

Belabas, K., Klueners, J., Steel, A., van Hoeij, M., 2004. Factoring polynomials over global fields, preprint arXiv:math/0409510v1, to appear in Journal de Théorie des Nombres de Bordeaux.

Bosma, W., Cannon, J., Playoust, C., 1997. The Magma algebra system. I: The user language. J. Symb. Comput. 24 (3-4), 235–265.

Bostan, A., Lecerf, G., Salvy, B., Schost, E., Wiebelt, B., 2004. Complexity issues in bivariate polynomial factorization. Gutierrez, Jaime (ed.), ISSAC 2004. Proceedings of the 2004 international symposium on symbolic and algebraic computation, Santander, Spain, July 4–7, 2004. New York, NY: ACM Press. 42-49 (2004).

Burden, R. L., Faires, J., 1993. Numerical analysis. 5th ed. Boston, MA: PWS Publishing Company. London: ITP International Thomson Publishing, xiv, 768 p.

Chèze, G., 2004a. Absolute polynomial factorization in two variables and the knapsack problem. Gutierrez, Jaime (ed.), ISSAC 2004. Proceedings of the 2004 international symposium on symbolic and algebraic computation, Santander, Spain, July 4–7, 2004. New York, NY: ACM Press. 87-94 (2004).

Chèze, G., 2004b. Des méthodes symboliques-numériques et exactes pour la factorisation absolue des polynômes en deux variables. Ph.D. thesis.

Chèze, G., Galligo, A., 2005. Four lectures on polynomial absolute factorization. Dickenstein, Alicia (ed.) et al., Solving polynomial equations. Foundations, algorithms, and applications. Berlin: Springer. Algorithms and Computation in Mathematics 14, 339-392, 393–418 (2005).

Chèze, G., Lecerf, G., 2007. Lifting and recombination techniques for absolute factorization. J. Complexity 23 (3), 380–420.

Cohen, H., Diaz y Diaz, F., 1991. A polynomial reduction algorithm. Sémin. Théor. Nombres Bordx., Sér. II (1), 351–360.

Dèbes, P., Walkowiak, Y., 2008. Bounds for Hilbert's irreducibility theorem. Pure Appl. Math. Q. 4 (4), 1059–1083.

Dvornicich, R., Traverso, C., 1989. Newton symmetric functions and the arithmetic of algebraically closed fields. Applied algebra, algebraic algorithms and error-correcting codes, Proc. 5th Int. Conference, AAECC-5, Menorca, Spain, 1987, Lect. Notes Comput. Sci. 356, 216-224 (1989).

Gao, S., 2001. Absolute irreducibility of polynomials via Newton polytopes. J. Algebra 237 (2), 501–520.

Gao, S., 2003. Factoring multivariate polynomials via partial differential equations. Math. Comput. 72 (242), 801–822.

Geddes, K. O., Czapor, S. R., Labahn, G., 1992. Algorithms for computer algebra. Dordrecht: Kluwer Academic Publishers Group. XVIII, 585 p. .

Kaltofen, E., 1985. Fast parallel absolute irreducibility testing. J. Symb. Comput. 1, 57–67.

Kaltofen, E., 1992. Polynomial factorization 1987-1991. In I. Simon, editor, Proc. LATIN '92.

Kaltofen, E., 1995. Effective Noether irreducibility forms and applications. J. Comput. System Sci. 50 (2), 274–295, 23rd Symposium on the Theory of Computing (New Orleans, LA, 1991).

Lecerf, G., 2006. Sharp precision in Hensel lifting for bivariate polynomial factorization. Math. Comput. 75 (254), 921–933.

Lecerf, G., 2007. Improved dense multivariate polynomial factorization algorithms. J. Symb. Comput. 42 (4), 477–494.

Lenstra, A., Lenstra, H., Lovász, L., 1982. Factoring polynomials with rational coefficients. Math. Ann. 261, 515–534.

Mathemagix, 2009. A free computer algebra system. Available at http://www.mathemagix.org.

Nguên, P. Q., Stehlé, D., 2005. Floating-point LLL revisited. Cramer, Ronald (ed.), Advances in cryptology – EUROCRYPT 2005. 24th annual international conference on the theory and applications of cryptographic techniques, Aarhus, Denmark, May 22–26, 2005. Proceedings. Berlin: Springer. Lecture Notes in Computer Science 3494, 215-233 (2005).

Noether, E., 1922. Ein algebraisches Kriterium für absolute Irreduzibilität. Math. Ann. 85, 26–33.

Ostrowski, A. M., 1975. On multiplication and factorization of polynomials. I: Lexicographic orderings and extreme aggregates of terms. Aequationes Math. 13, 201–228.

Ragot, J.-F., 1997. Sur la factorisation absolue des polynomes. Ph.D. thesis.

Ragot, J.-F., 2002. Probabilistic absolute irreducibility test for polynomials. J. Pure Appl. Algebra 172 (1), 87–107.

Rupprecht, D., 2004. Semi-numerical absolute factorization of polynomials with integer coefficients. J. Symb. Comput. 37 (5), 557–574.

Schneider, R., 1993. Convex bodies: the Brunn-Minkowski theory. Encyclopedia of Mathematics and Its Applications. 44. Cambridge: Cambridge University Press. xiii, 490 p. .

Schnorr, C. P., 2006. Fast LLL-type lattice reduction. Inf. Comput. 204 (1), 1–25.

Sommese, A. J., Verschelde, J., Wampler, C. W., 2004. Numerical factorization of multivariate complex polynomials. Theor. Comput. Sci. 315 (2-3), 651–669.

Trager, B., 1985. On the integration of algebraic functions. Ph.D. thesis.

Trager, B., 1989. Good reduction of curves and applications. Meeting on Computer and Commutative Algebra (COCOA II).

van Hoeij, M., 2002. Factoring polynomials and the knapsack problem. J. Number Theory 95 (2), 167–189.

von zur Gathen, J., Gerhard, J., 2003. Modern computer algebra. 2nd ed. Cambridge University Press.

Zannier, U., 1997. On the reduction modulo p of an absolutely irreducible polynomial $f(x, y)$. Arch. Math. 68 (2), 129–138.