Local polynomial factorisation: improving the Montes algorithm

Adrien Poteaux, Martin Weimann

Local polynomial factorisation: improving the Montes algorithm

Adrien Poteaux[1] and Martin Weimann[2]

[1]Univ. Lille, CNRS, Centrale Lille, UMR 9189 CRIStAL, F-59000 Lille, France
[2]LMNO, Université de Caen-Normandie

Abstract

We improve significantly the Nart-Montes algorithm for factoring polynomials over a complete discrete valuation ring $\mathbb{A}$. Our first contribution is to extend the Hensel lemma in the context of generalised Newton polygons, from which we derive a new divide and conquer strategy. Also, if $\mathbb{A}$ has residual characteristic zero or high enough, we prove that approximate roots are convenient representatives of types, leading finally to an almost optimal complexity both for irreducibility and factorisation issues, plus the cost of factorisations above the residue field. For instance, to compute an OM-factorisation of $F \in \mathbb{A}[x]$, we improve the complexity results of [3] by a factor δ, the discriminant valuation of F.

1 Introduction

Let $\mathbb{A}$ be a complete discrete valuation ring with residue field $\mathbb{F}$ and consider $F \in \mathbb{A}[x]$, monic and separable of degree d. The aim of this paper is to improve complexity bounds for the factorisation of F. Such a polynomial factorisation is a fundamental task of computer algebra with various applications in number theory and algebraic geometry. As such, our complexity results allow to fasten various computational problems, such as Okutsu frames, integral basis or genus of plane curves (see Section 6 for further details). Our work is based on the seminal Montes algorithm [10], for which the best known complexity is given in [3]. In [8], the authors conclude their paper by:

> Probably, an optimal local factorisation algorithm would consist in the application of the Montes algorithm as a fast method to get an Okutsu approximation to each irreducible factor, combined with an efficient "Hensel lift" routine able to improve these initial approximations by doubling the precision at each iteration. One may speculate that Newton polygons of higher order might also be used to design a similar acceleration procedure.

With S. Pauli, Guardia and Nart answered partially to this question thanks to the *single-factor lifting* algorithm [11], that can be viewed as a Newton-like method to lift a single factor with a quadratic convergence. This led to the overall complexity analysis of [3].
In this paper, we answer more precisely to this question, by showing that the classical Hensel algorithm can be adapted to the context of Newton polygon of higher order. We also provide a new divide and conquer strategy using this adapted Hensel algorithm, enabling us to lift all factors of F at the same time, with a complexity almost linear in the size of the output. These two elements allow us to gain a factor d in comparison to the complexity result of [3]. Moreover, following [27], we show that when $\mathrm{char}(\mathbb{F}) \nmid d$, we can use *approximate roots* as strongly optimal representatives of a type[1]. This induces an irreducibility test with a complexity almost linear in δ the valuation of the discriminant of F ; see Theorem 2. This improvement propagates for factorisation with a slightly greater assumption

Assumption 1. *$\mathrm{char}(\mathbb{F}) = 0$ or $\mathrm{char}(\mathbb{F}) > d$*

leading a complexity almost linear in $d\,n$ for a required precision $n \geq \delta$; see Theorem 4.

Related work. Classical implemented algorithms for factoring polynomials over $\mathbb{Q}_p$ (see e.g. [4, 6, 24, 25]) are based on the Zassenhaus Round Four algorithm, suffering from loss of precision in computing characteristic polynomials. In [11], the authors introduced a new technique as a combination of the Montes algorithm [9, 10] which exploits the Newton polygons of higher order (as initiated in [25]), and a Newton-like single factor lifting. Further complexity improvements are obtained in [3]. The present work is in the same vein, with the notable difference that we introduce a multi factor lifting, which is used in course of the Montes algorithm whenever a non trivial factorisation is discovered.
For rings of Laurent series $\mathbb{K}((t))$ of characteristic zero or high enough, Newton-Puiseux like algorithms can be used. The best complexity in this context is softly linear [29], as in the present paper, but more difficult to implement and slower for irreducibility issues. This led us to introduce in [27] approximate roots *à la Abhyankhar* [1, 26] in order to derive a faster and easy-to-implement irreducibility test, quite close to the algorithm [3] *à la Montes*, although not dealing with the small characteristic. This was a first step towards the present work, where we use now approximate roots in the factorisation context, as allowed by a systematic use of our generalised Hensel lifting. Note that the divide conquer in the present paper is quite different than the one of [29, Section 4.4]: in particular, the initialisation of the Hensel algorithm does not use the not yet implemented generalisation of the half gcd algorithm described in [20].
Finally, let us insist that following [11], our algorithm computes as a byproduct an Okutsu frame of each irreducible factors of F, containing the most significant arithmetic informations [8] and closely related to various computational problems of number theory and algebraic geometry, such as the computation of integral basis (see Section 6 for further details).

[1] see Sections 2 and 3 for the definitions of these terms

Organisation of the paper. We start by a summary of important definitions related to the Montes algorithm in Section 2. Then, we focus on the irreducibility test when $\mathrm{char}(\mathbb{F}) \nmid d$ in Section 3, leading to Theorem 2. In Section 4, we show how to adapt the Hensel algorithm in the context of Newton polygon of higher orders. Section 5 uses this latter algorithm on a well chosen type in order to derive a divide and conquer algorithm, leading to Theorem 4. Finally, we discuss some direct applications in Section 6.

Complexity model. Polynomials in $\mathbb{A}[x]$ considered in this paper are supposed given in a dense representation, with coefficients available up to an arbitrary precision (e.g. represented as tables, as we always use truncation bounds). We use the algebraic RAM model of Kaltofen [16, Section 2], counting only the number of arithmetic operations in the residue field $\mathbb{F}$. We classically denote $\mathcal{O}()$ and $\mathcal{O}^{\sim}()$ to respectively hide constant and logarithmic factors in our complexity results ; see e.g. [7, Chapter 25, Section 7]. We additionally let $\mathcal{O}_{\epsilon}(d) = \mathcal{O}(d^{1+\epsilon(d)})$ with $\epsilon(d) \to 0$. We have $\mathcal{O}^{\sim}(d) \subset \mathcal{O}_{\epsilon}(d)$, and freely speak of *almost linear* in d for both notations. Fast multiplication in $\mathbb{F}[y]$ is used, i.e. we multiply two polynomials of degree at most d within $\mathcal{O}^{\sim}(d)$ operations in $\mathbb{F}$ [7, Section 8.3]. We assume that univariate factorisation over $\mathbb{F}$ is available. Intermediate finite extensions $\mathbb{F}_k$ of degree f_{k-1} of $\mathbb{F}$ will occur (see Section 2), naturally represented as a quotient of $\mathbb{F}[y_0, \ldots, y_{k-1}]$ by a triangular prime ideal $(P_0(y_0), \ldots, P_{k-1}(y_0, \ldots, y_{k-1}))$.

Lemma 1. *An operation in $\mathbb{F}_k$ takes $\mathcal{O}_{\epsilon}(f_{k-1})$ operations over $\mathbb{F}$.*

Proof. If $\mathrm{Card}(\mathbb{F}) \geq \binom{d}{2}$, use [14, Theorem 4]. If $\mathrm{Card}(\mathbb{F}) < \binom{d}{2}$, proceed as in [27, proof of Theorem 1] (roughly speaking, keep the first i levels of the triangular set so that $\mathrm{Card}(\mathbb{F})\, f_{i-1} \geq \binom{d}{2}$ with i minimal, and apply [14, Theorem 4] over $\mathbb{F}_i$; as $i \in \mathcal{O}(\log \log d)$, an operation in $\mathbb{F}_i$ is $\mathcal{O}^{\sim}(f_{i-1})$ via [17, Proposition 2]). □

Remark 1. *Since some subroutines use the triangular representation of $\mathbb{F}_k$ (Remark 3), introducing a randomised Las Vegas subroutine to fasten the arithmetic in $\mathbb{F}_k$ via a primitive representation would not a priori be sufficient to express our complexity results in $\mathcal{O}^{\sim}()$ instead of $\mathcal{O}_{\epsilon}()$, in contrast to [29].*

2 Types and factorisation

Let $\mathbb{L}$ be a complete discrete valuation field, $v : \mathbb{L} \to \mathbb{Z} \cup \{+\infty\}$ any normalised and surjective valuation on $\mathbb{L}$ and π an uniformiser. We denote by $\mathbb{A} \subset \mathbb{L}$ the ring of integers of $(\mathbb{L}, v)$ and by $\mathbb{F} = \mathbb{A}/(\pi)$ the residue field of v. The two fields we have in mind in this paper are $\mathbb{L} = \mathbb{Q}_p$ the field of p-adic numbers and $\mathbb{L} = \mathbb{K}((t))$ the field of Laurent series over any field $\mathbb{K}$.

2.1 Types

We start with types of order 0, denoting the residue field $\mathbb{F}_0 := \mathbb{F}$.

Definition 1. *A type of order 0 is $\mathbf{t}_0 = [P_0]$, where $P_0 \in \mathbb{F}_0[y]$ is a monic irreducible polynomial.*

For any $G \in \mathbb{L}[x]$, a type of order 0 comes together with the Gauss valuation $v_0(\sum_i a_i\, x^i) := \min_i(v(a_i))$ and the residual polynomial operator $R_0(G) := G(y)/\pi^{v_0(G)} \mod \pi$.
Types $\mathbf{t}_k = [P_0, (\phi_1, \lambda_1, P_1), \ldots, (\phi_k, \lambda_k, P_k)]$ of order $k \geq 1$ are defined inductively below. If $1 \leq i \leq k-1$, we denote $\mathbf{t}_i = [P_0, (\phi_1, \lambda_1, P_1), \ldots, (\phi_i, \lambda_i, P_i)]$. For any field K and $P, Q \in K[y]$, we write $P(y) \sim Q(y)$ if there exists $c \in K^\times$ such that $P(y) = c\,Q(y)$. Also, we denote $\mathcal{P}$ the semigroup of polygons (i.e. the set of all open convex polygons of the plane, attached to finite formal sums of sides) and $\mathcal{P}^- \subset \mathcal{P}$ the semi-group of polygons with negative slopes (principal polygons). See [10, Section 1.1] for details.
Assume that types of order $k-1$ have been defined and that we can attach to a type of order $k-1$ a valuation $v_{k-1} : \mathbb{L}[x] \to \mathbb{Z}$, a field extension $\mathbb{F}_{k-1}$ of $\mathbb{F}$ and a residual polynomial operator $R_{k-1} : \mathbb{L}[x] \to \mathbb{F}_{k-1}[y]$.

Definition 2. *Let $k \geq 1$. $\mathbf{t}_k = [P_0, (\phi_1, \lambda_1, P_1), \ldots, (\phi_k, \lambda_k, P_k)]$ is a type of order k if $\mathbf{t}_{k-1}$ is a type of order $k-1$ and*

- *$\phi_k \in \mathbb{A}[x]$ is monic, irreducible and satisfies $R_{k-1}(\phi_k) \sim P_{k-1}$.*
- *$\lambda_k = -m_k/q_k \in \mathbb{Q}^-$, with $(q_k, m_k) \in \mathbb{N}^2$ coprime. We denote (α_k, β_k) s.t. $\alpha_k\, q_k - \beta_k\, m_k = 1$ with $0 \leq \beta_k < q_k$.*
- *$P_k \neq y \in \mathbb{F}_k[y]$ is monic, irreducible over $\mathbb{F}_k := \mathbb{F}_{k-1}[y]/(P_{k-1})$. We let $\ell_k := \deg(P_k)$ and $z_k := y \mod P_k(y) \in \mathbb{F}_{k+1}$.*

We will denote $e_k := q_1 \ldots q_k$ and $f_k := \ell_0 \ldots \ell_k = [\mathbb{F}_{k+1} : \mathbb{F}]$[2].

2.2 Associated operators and representatives

We fix a type $\mathbf{t} = [P_0, (\phi_1, \lambda_1, P_1), \ldots, (\phi_k, \lambda_k, P_k)]$ of order $k \geq 1$ and detail several operators associated to it. If $G \in \mathbb{L}[x]$, we denote $G = \sum a'_i \phi_{k-1}^i$ and $G = \sum a_i \phi_k^i$ their ϕ_{k-1} and ϕ_k-adic expansion[3].

Augmented valuation. $v_k : \mathbb{L}[x] \to \mathbb{Z}$ is defined from v_{k-1} as

$$v_k(G) := \min_i(q_{k-1} v_{k-1}(a'_i \phi_{k-1}^i) + m_{k-1} i). \tag{1}$$

This is indeed an "augmented valuation" as introduced by Mac Lane [18, 19] (see e.g. [10, page 379] for a detailed explanation). Notice in particular that $v_k(G) = \min_i v_k(a'_i \phi_{k-1}^i)$.

Newton polygon of higher order. The *polygon operator* $\mathcal{N}_k : \mathbb{L}[x] \to \mathcal{P}$ associates to G the lower convex hull of $\{(i, v_k(a_i \phi_k^i),\ a_i \neq 0\}$. We let $\mathcal{N}_k^-(G) \in \mathcal{P}^-$ stands for the principal part of $\mathcal{N}_k(G)$.

[2] A reader used to the work of Nart et al should pay attention that the notations e_k and f_k in [10] are here denoted q_k and ℓ_k. We rather use e_k and f_k for the ramification index and residual degree discovered so far, following [29].
[3] If $k = 1$, we let $\phi_0 = x$, $q_0 = 1$ and $m_0 = 0$, so that $v_1 = v_0$.

Residual polynomial operator. We need several intermediate operators. We let $S_k(G) := \{(i,j) \in \mathcal{N}_k(G),\ m_k i + q_k j \text{ is minimal}\}$, $I_k(G) := \{i \in \mathbb{N},\ v_k(a_i\phi_k^i)) \in S_k(G)\}$ and $i_k(G) := \min(I_k(G))$ (letting $i_0(G) = 0$ additionally). We let $\tau_{k,i} := \frac{i_{k-1}(a_i)+\beta_{k-1}v_k(a_i\phi_k^i)}{q_{k-1}} \in \mathbb{Z}$ (see [10], after Definition 2.19). Then $R_k : \mathbb{L}[x] \to \mathbb{F}_k[y]$ is defined inductively as

$$R_k(G) = \sum_{i \in I_k(G)} z_{k-1}^{\tau_{k,i}} R_{k-1}(a_i\phi_k^i)(z_{k-1})\, y^{\frac{i-i_k(G)}{q_k}}.$$

Remark 2. *Let us summarise the dependencies of these operators: v_k depends only of v_{k-1}, ϕ_{k-1} and λ_{k-1}, thus only of $\mathbf{t}_{k-1}$. $\mathcal{N}_k$ depends of v_k and ϕ_k. Finally, R_k depends of R_{k-1}, ϕ_k and λ_k.*

Representative of $\mathbf{t}_k$. They are defined as follows.

Definition 3. *Let $G \in \mathbb{A}[x]$ be monic. We say that G is of type $\mathbf{t}$ if for $0 \le i \le k$, $\mathcal{N}_i(G)$ is one-sided of slope λ_i, and for $1 \le i \le k$ $R_i(G) \sim P_i^{N_i}$ for some $N_i \in \mathbb{N}^\times$. We denote by $G_\mathbf{t}$ the product of all monic irreducible factors of G of type $\mathbf{t}$. We say that $\mathbf{t}$ divides G is $\deg(G_\mathbf{t}) > 1$. Finally, G is said to be a representative of $\mathbf{t}$ if additionally $ord_\mathbf{t}(G) := ord_{P_k} R_k(G) = 1$.*

2.3 Factorisation according to a type

The following theorem summarises the main results that led to the Montes algorithm:

Theorem 1. *Let $k \ge 1$ and $\mathbf{t}$ be a type of order $k-1$ together with a representative ϕ_k. Denote v_k the associated augmented valuation and assume $F \in \mathbb{A}[x]$ monic.*

1. *(Theorem of the polygon, [10, Theorem 3.1]) Suppose that $\mathcal{N}_k^-(F) = S_1 + \cdots + S_g$ where the polygons $S_1, \ldots, S_g$ are one-sided of distinct slopes $\lambda_{k,1}, \ldots, \lambda_{k,g}$. Denote by $R_{k,i}$ the residual polynomial operator associated to v_k, ϕ_k and the slope $\lambda_{k,i}$. The polynomial $F_\mathbf{t}$ admits a factorisation*

$$F_\mathbf{t} = F_{\mathbf{t},1} \cdots F_{\mathbf{t},g} \in \mathbb{A}[x]$$

where $\mathcal{N}_k^-(F_{\mathbf{t},i}) = S_i$ up to translation and $R_{k,i}(F_{\mathbf{t},i}) \sim R_{k,i}(F)$.

2. *(Theorem of the residual polynomial, [10, Theorem 3.7]) Let $R_{k,i}(F) \sim P_{k,i,1}^{a_1} \cdots P_{k,i,r}^{a_r}$ where the $P_{k,i,j}$ are pairwise coprime irreducible polynomials. Then $F_{\mathbf{t},i}$ admits a factorisation*

$$F_{\mathbf{t},i} = F_{\mathbf{t},i,1} \cdots F_{\mathbf{t},i,r_i} \in \mathbb{A}[x]$$

where $\mathcal{N}_k(F_{\mathbf{t},i,j})$ is straight of slope $\lambda_{k,i}$ and $R_{k,i}(F_{\mathbf{t},i,j}) \sim P_{k,i,j}^{a_j}$.

3. *(Irreducibility criterion) If $a_j = 1$, then $F_{\mathbf{t},i,j}$ is irreducible.*

3 Testing irreducibility

3.1 Approximate roots

Proposition 1. *Let $F \in \mathbb{A}[x]$ be monic of degree d, with $char(\mathbb{A}) \nmid d$. Let $N \in \mathbb{N}$ dividing d. There exists a unique polynomial $\psi \in \mathbb{A}[x]$ monic of degree d/N such that $\deg(F - \psi^N) < d - d/N$. We call it the N^{th} approximate roots of F, denoted by $\sqrt[N]{F}$. It can be computed in less than $\mathcal{O}(d)$ operations in $\mathbb{A}$.*

Proof. See e.g. [26, Proposition 3.1] for the existence, and [27, Proposition 11] for their computation. □

Proposition 2. *Let $F \in \mathbb{A}[x]$ be a monic separable polynomial of type $\mathbf{t}$ and denote $N = ord_{\mathbf{t}}(F)$. Suppose $char(\mathbb{F}) \nmid N$ and let $\psi = \sqrt[N]{F}$.*

1. *ψ is a representative of $\mathbf{t}$.*
2. *If F is of type $\mathbf{t} \cup (\psi, -m/q, P)$, then $q \deg(P) > 1$.*

Proof. Point 1 can be shown with arguments similar to [27, Lemma 7]. Point 2 is a direct consequence of the fact that the coefficient of ψ^{N-1} in the ψ-adic expansion of F is zero. □

Without dealing with the precision of computations in $\mathbb{A}$, this leads to the following irreducibility test algorithm:

Algorithm: `FastIrreducible`(F)
Input: $F \in \mathbb{A}[x]$ monic separable s.t. $\text{char}(\mathbb{F}) \nmid \deg(F)$.
Output: A Boolean (is F irreducible ?), a type $\mathbf{t}$ and a representative ϕ of $\mathbf{t}$.

if $R_0(F)$ *is not some* $P_0^{N_0}$ **then return** *False,* [];
$\mathbf{t} \leftarrow [P_0]$, $k \leftarrow 1$, $N \leftarrow N_0$;
while $N > 1$ **do**
  $\phi_k \leftarrow \sqrt[N]{F}$;
  if $\mathcal{N}_k^-(F)$ *is not one sided* **then return** *(False,* $\mathbf{t}$*,* ϕ_k*)*;
  if $R_k(F)$ *is not some* $P_k^{N_k}$ **then return** *(False,* $\mathbf{t}$*,* ϕ_k*)*;
  $\mathbf{t} \leftarrow \mathbf{t} \cup (\phi_k, \lambda_k, P_k)$; `//` λ_k `the slope of` $\mathcal{N}_k(F)$
  $N \leftarrow N_k$, $k \leftarrow k + 1$;
return *(True,* $\mathbf{t}, F$*)*

This algorithm is similar to the one of Montes et al, with the exception of the way we construct the representatives: approximate roots enable a quick computation of the representative ϕ_k, together with the additional property $q_k \ell_k > 1$ which ensures $k \in \log(d)$.

3.2 Precision and complexity

It remains to deal with the necessary precision to conduct operations in $\mathbb{A}$ and get complexity bounds for the computation of $\mathcal{N}_k^-(F)$ and $R_k(F)$. We proceed as in [27, Section 5.4]: starting with a small precision σ, we check at each iteration if σ is sufficient to certify that the computed data of $F \mod \pi^\sigma$ is truly the data of F. If the precision is not sufficient, we double it and restart the whole computation. This process multiplies the overall complexity by at most 2. We need a certificate that the current precision σ is high enough and an upper bound for σ.
Assume that we computed a type $\mathbf{t}_{k-1}$ dividing F using a precision σ. Compute $\mathcal{N}_k^-(F)$ with precision σ and denote $\lambda_{\min}$ its *right hand slope*, with convention $\lambda_{\min}(F) = +\infty$ if it is reduced to a vertex.

Lemma 2. *Let $F \in \mathbb{A}[x]$ monic divisible by $\mathbf{t}_{k-1}$. If $\sigma > \frac{v_k(F)+N\,|\lambda_{\min}|}{v_k(\pi)}$, then truncating computations modulo $\pi^{\sigma+1}$ will compute the correct right hand edge of $\mathcal{N}_k^-(F)$. If moreover F is of type $\mathbf{t}_{k-1}$, then $\frac{v_k(F)+N\,|\lambda_{\min}|}{v_k(\pi)} < \frac{2\delta}{d}$.*

Proof. This is [3, Lemmas 2.9 and 2.8]. □

Lemma 3. *One can compute $v_k(F)$ in $\mathcal{O}(d)$ operations in $\mathbb{A}$.*

Proof. Compute the ϕ_{k-1}-adic expansion of F in $\mathcal{O}(d)$ operations in $\mathbb{A}$ [7, Theorem 9.15]. If $k > 1$, compute recursively each $v_{k-1}(a_i\phi_{k-1}^i)$. As there is a closed formula for $v_{k-1}(\phi_{k-1})$ [10, Proposition 2.15], the bound $\deg(a_i) < \deg(\phi_{k-1})$ concludes. □

Proposition 3. *One can compute $\mathcal{N}_k^-(F)$ with precision σ in less than $\mathcal{O}(d\,\sigma)$ operations in $\mathbb{F}$.*

Proof. First compute the ϕ_k-adic expansion of F, then the different values $v_k(a_i\,\phi_k^i)$. The complexity then comes from [7, Theorem 9.15] and Lemma 3. □

Proposition 4. *Up to the cost of operations already done while computing $\mathcal{N}_k^-(F)$, one can compute $R_k(F)$ in $\mathcal{O}_\epsilon(d\,f_{k-1})$ op. in $\mathbb{F}$.*

Proof. This is [3, Lemma 5.6]. □

Lemma 4. *Let $F \in \mathbb{A}[x]$ monic of type $\mathbf{t}_{k-1}$ such that $ord_{\mathbf{t}_0} F > 1$. Then $f_{k-1}d \le 2\ell_0\delta$.*

Proof. We know that $F \equiv P_0^{N_0} \mod \pi$ for some irreducible polynomial $P_0 \in \mathbb{F}[x]$ of degree ℓ_0. Let θ be a root of F, say with minimal polynomial G (prime factor of F). Thus $\theta \mod \pi$ is a root of P_0 and there are $N_0 - 1$ other roots θ' of F such that $\theta \equiv \theta' \mod \pi$. It follows that $v_0(\theta - \theta') > 0$, hence $v_0(\theta - \theta') \ge \frac{1}{e_G}$ where e_G is the ramification index of G. Summing over all θ and all θ', we get $\delta \ge \sum_G \deg(G)(N_0 - 1)/e_G$, the sum over all prime factors of F. We have $e_G f_G = \deg(G)$ (f_G residual degree) and $f_{k-1} | f_G$ for all G. It follows that $\delta \ge f_{k-1}(N_0 - 1)$. Equality $N_0\,\ell_0 = d$ concludes. □

Theorem 2. *If $char(\mathbb{F}) \nmid d$, then there exists an algorithm that tests if F is irreducible in less than $\mathcal{O}_\epsilon(\delta\ell_0)$ operations in $\mathbb{F}$ and at most $\log_2(d)$ univariate irreducibility tests.*

Proof. The algorithm is `FastIrreducible` with the above modification to deal with the precision of the computations. As we use a precision $\sigma \leq 2\delta/d$, the complexity comes from all the intermediate results of this section. □

The case $\mathbb{A} = \mathbb{Z}_p$. We say that $F \in \mathbb{Z}_p[x]$ is Weierstrass if $F \equiv x^d \mod p$. Also, we say that p is small if it is polynomial in d and δ.

Corollary 1. *If p is small and does not divide d, we can test the irreducibility of a separable Weierstrass polynomial $F \in \mathbb{Z}_p[x]$ with $\mathcal{O}_\epsilon(\delta)$ operations in $\mathbb{F}_p$.*

Proof. F being Weierstrass, we have $\ell_0 = 1$ and $d \leq 2\delta$ by Lemma 4. We can check if $R_k(F) \in \mathbb{F}_k[x]$ is a prime power (and then compute P_k) within $\mathcal{O}_\epsilon(d\log(p))$ operations in $\mathbb{F}_p$ using [15, Corollary 2][4] together with $\deg(R_k(F))f_{k-1} \leq d$. □

Under the same hypothesis, Montes irreducibility test would require $\mathcal{O}_\epsilon(\delta^2)$ operations in $\mathbb{F}_p$, see [3, Theorem 5.10].

3.3 The case $\mathbf{char}(\mathbb{F}) \mid d$

When $\mathrm{char}(\mathbb{F}) \mid d$, approximate roots cannot be used as representatives of types. Following [3], we compute these representatives in another way (Proposition 6 below, in $\mathcal{O}_\epsilon(dw(F)/w(\pi)) \subset \mathcal{O}_\epsilon(\delta)$ operations in $\mathbb{F}$). Unfortunately, we might have now $q_k\ell_k = 1$ (refinement steps) and the number of iterations is not in $\mathcal{O}(\log(d))$ anymore. From [3, Lemma 5.11], we can bound the number of recursive call by $\mathcal{O}(\delta/\ell_0)$[5]. This leads to a complexity in $\mathcal{O}_\epsilon(\delta^2)$ operations in $\mathbb{F}$, plus the univariate irreducibility tests. We can still run only $k \leq \log_2(d)$ of them by first checking that $R_k(F)$ has a single root (i.e. $q_k\ell_k = 1$), using a univariate shift, in $\mathcal{O}(d)$ operations in $\mathbb{F}$, and use the univariate irreducibility test only when $q_k\ell_k > 1$. We proved the following:

Proposition 5. *If $\mathrm{char}(\mathbb{F})$ divides d, one can check if F is irreducible in less than $\mathcal{O}_\epsilon(\delta^2)$ operations in $\mathbb{F}$ and at most $\log_2(d)$ univariate irreducibility test.*

This result is very similar to [3, Theorem 5.10] (we just use some minor complexity improvements in some intermediate results, described in Section 3.2). We still call `FastIrreducible` the underlying algorithm.

4 Generalised slope factorisation

Algorithm `FastIrreducible` of the previous section will either prove that F is irreducible, either provide a type $\mathbf{t}_{k-1}$ together with a representative ϕ_k such that either $\mathcal{N}_k(F)$ has two or more distinct slopes, either R_k is not a power of an irreducible polynomial. To summarise these two possibilities, it is convenient to consider a slight modification of the residual polynomial operator R_k attached to the *right hand slope* $\lambda_k = -m_k/q_k$ of $\mathcal{N}_k^-(F)$:

[4] This result extends to towers of fields following the proof of [15, Corollary 3]
[5] Equation (5.4) of [3] actually shows that it is bounded by $\mathcal{O}(\delta/f_{k-1})$

Definition 4. *The* modified residual polynomial *of* $G \in \mathbb{L}[x]$ *(attached to* λ_k*) is defined by* $\tilde{R}_k(G)(y) := y^{i_k(G)} R_k(G)(y^{q_k})$.

If F is not irreducible, we get

$$\tilde{R}_k(F) = h_0 h_1 \cdots h_r$$

with $h_0 \sim y^{i_k(F)}$, $h_1, \ldots, h_r \in \mathbb{F}_k[y^{q_k}]$ coprime, this factorisation containing at least two different non trivial factors.
From Theorem 1, there exist $F_0, F_1, \cdots, F_r \in \mathbb{A}[x]$ monic such that

$$F = F_0 F_1 \cdots F_r \quad \text{with} \quad \tilde{R}_k(F_i) \sim h_i, \quad 0 \le i \le r.$$

This section describes a Hensel-like algorithm to compute such a factorisation up to an arbitrary precision with complexity almost linear in the size of the output.

4.1 A (slightly) more general problem

We express our problem in a slightly more general context, that will be useful in Section 5. Let $F \in \mathbb{A}[x]$ be a monic polynomial, $\mathbf{t} = \mathbf{t}_{k-1}$ be a type of order $k-1$ dividing F and $\phi = \phi_k$ a representative of $\mathbf{t}$. We assume that either $F_{\mathbf{t}}$ is a proper factor of F, either $F_{\mathbf{t}}$ admits a non trivial factorisation at order k induced by Theorem 1.
Let $v = v_k$ the augmented valuation built from $\mathbf{t}$ (i.e. from v_{k-1}, ϕ_{k-1} and λ_{k-1}) and $\mathcal{N}(F) = \mathcal{N}_k(F)$ the generalised Newton polygon defined by v and ϕ. Denote $\lambda = -\frac{m}{q} \in \mathbb{Q}^-$ the *right hand slope* of $\mathcal{N}^-(F)$ and $R(F)$ (resp. $\tilde{R}(F)$) the (modified) residual polynomial defined by $\mathbf{t}$, ϕ and λ.

Lemma 5. *Let* $G \in \mathbb{A}[x]$ *monic and* $g = \tilde{R}(G)$. *If* G *is of type* $\mathbf{t}$, *then*

$$\mathcal{N}(G) = \mathcal{N}^-(G), \ \deg(G) = \deg(\phi)\deg(g), \ lc(g) = R(\phi)^{\deg(g)}.$$

If $\mathbf{t}$ *does not divide* G *then* $g \in \mathbb{F}_k^\times$.

Proof. See e.g. [10]. □

Thanks to this lemma and our assumption on F, we get

$$\tilde{R}(F) = h_0 h_1 \cdots h_r h_\infty$$

with $h_0 = R(\phi)^s y^s$, $h_\infty \in \mathbb{F}_k^\times$ and $h_1, \ldots, h_r \in \mathbb{F}_k[y^q]$ powers of some coprime irreducible polynomials satisfying $h_i(0) \neq 0$ and $\mathrm{lc}(h_i) = R(\phi)^{\deg(h_i)}$. Moreover, there are uniquely determined monic polynomials $F_0, \ldots, F_r, F_\infty \in \mathbb{A}[x]$ such that

$$F = F_0 F_1 \cdots F_r F_\infty \quad \text{with} \quad \tilde{R}(F_i) = h_i, \quad i = 0, \ldots, t, \infty.$$

In particular, we have $F_{\mathbf{t}} = F_0 F_1 \cdots F_r$ and $F = F_{\mathbf{t}} F_\infty$.
Such a factorisation can be seen as a generalisation of the "slope factorisation" of [5]. It is natural to express the precision using the augmented valuation $w = v_{k+1}$ defined by v, ϕ and λ following (1). Thanks to a dichotomic argument, we are reduced to compute $F = \tilde{G}\tilde{H}$ (up to some precision w.r.t. w), with $\tilde{G}, \tilde{H}$ products of some of the F_i's above. We proceed as follows:

1. **Initialisation.** Compute $G, H \in \mathbb{A}[x]$ and $U, V \in \mathbb{L}[x]$ such that $w(F - G\,H) > w(F)$ and $w(U\,G + V\,H - 1) > 0$.

2. **Lifting.** Run the classical Hensel Lemma with adapted truncations to compute $\tilde{G}$ and $\tilde{H}$ such that $w(F - \tilde{G}\tilde{H}) \geq w(F) + n$ for a given precision n.

4.2 Initialisation

A key point is to compute polynomials with prescribed (modified) residual polynomial. We start with a definition.

Definition 5. *A polynomial $H \in \mathbb{A}[x]$ is said* monic in ϕ *if it has ϕ-adic expansion $H = \sum_{i=0}^{N} a_i \phi^i$ with $a_N = 1$. We say that H is strongly monic in ϕ (with respect to w) if moreover $w(H) = N w(\phi)$.*

Proposition 6. *Let $h = y^s\,\mathfrak{h}(y^q)$ with $\mathfrak{h} \in \mathbb{F}_k[y]$ and $W \in \mathbb{Z}$. We can compute $H \in \mathbb{L}[x]$ of smallest degree such that*

$$\tilde{R}(H) = h, \quad w(H) = W, \quad \deg(H) < (\deg(h) + 1)\deg(\phi).$$

It takes $\mathcal{O}\left(\deg(H)(\deg(h)+1)\,\frac{w(\phi)}{w(\pi)}\right)$ operations in $\mathbb{F}$. Furthermore:

1. *If $W \geq \deg(h) w(\phi)$, then $H \in \mathbb{A}[x]$.*

2. *If $W = \deg(h) w(\phi)$ and $lc(h) = R(\phi)^{\deg(h)}$, then H is strongly monic in ϕ.*

Proof. First use [3, Lemma 5.7] to compute $G \in \mathbb{A}[x]$ such that $\tilde{R}(G) = h$ and $w(G) = W + w(\pi)\,n$ with $n = \left\lceil \frac{\deg(h) w(\phi) - W}{w(\pi)} \right\rceil$. We can divide their complexity result by a factor $\deg(h)$ by replacing Horner evaluation therein by a divide and conquer strategy (see e.g. [12]). Finally, output $H = \pi^{-n}\,G$. □

Remark 3. *The representation of $\mathbb{F}_k$ as a tower of fields as described before Lemma 1 is used in the proof of [3, Lemma 5.7]. In particular, no operations in $\mathbb{F}_k$ is performed, so that the complexity can be expressed with $\mathcal{O}()$ and not $\mathcal{O}_\epsilon()$.*

Lemma 6. *Let $G, H \in \mathbb{L}[x]$ such that $w(G) = w(H)$.*

1. *$\tilde{R}_k(G) = \tilde{R}_k(H)$ if and only if $w(G - H) > w(G)$.*

2. *If $w(G + H) = w(G)$, then $\tilde{R}_k(G + H) = \tilde{R}_k(G) + \tilde{R}_k(H)$.*

Proof. (1) Since $R(G)$ and $R(H)$ have non zero constant term, Definition 4 implies that $\tilde{R}(G) = \tilde{R}(H)$ if and only if $R(G) = R(H)$ and $i_k(G) = i_k(H)$. As $w(G) = w(H)$, this is equivalent to that $R(G) = R(H)$ and $S(G) = S(H)$. [10, Proposition 2.8] concludes. (2) Equality $w(G + H) = w(G)$ implies that $S_k(G), S_k(H), S_k(G + H)$ lie on a same segment T of slope λ. We conclude with [10, Lemma 2.23 and eq. (19)], together with Definition 4. □

Let us consider now our initialisation problem. By the discussion above, we can assume that $\tilde{R}(F) = g\,h$ with g, h coprime, $g \in \mathbb{F}_k[y^q]$ and $h(y) = y^s\,\mathfrak{h}(y^q)$, with $\mathfrak{h} \in \mathbb{F}_k[y]$, $\mathfrak{h}(0) \neq 0$. We can additionally assume that $\mathrm{lc}_y(h) = R(\phi)^{\deg(h)}$. Notice that we might have $g \in \mathbb{F}_k^\times$.

Proposition 7. *Let g, h as above. One can compute $G, H \in \mathbb{A}[x]$ and $U, V \in \mathbb{L}[x]$ such that H is strongly monic in ϕ, $\tilde{R}(G) = g$, $\tilde{R}(H) = h$, $\deg(G) + \deg(H) \leq \deg(F)$, $w(F - G\,H) > w(F)$, $\deg(U) < \deg(H)$, $\deg(V) < \deg(G)$ and $w(U\,G + V\,H - 1) > 0$ in less than $\mathcal{O}\,(dw(F)/w(\pi))$ operations in $\mathbb{F}$.*

Proof. Use Proposition 6 to compute $G, H \in \mathbb{A}[x]$ with H strongly monic in ϕ and such that $\tilde{R}(H) = h$, $\tilde{R}(G) = g$, $w(F) = w(GH)$ with $\mathcal{O}\,(dw(F)/w(\pi))$ operations in $\mathbb{F}$ (this bound being a consequence of $w(F) = \deg(\tilde{R}(F))\,w(\phi)$). Notice that $\deg(G) + \deg(H) \leq \deg(F)$ as we construct G, H of minimal degrees. As $w(GH) = w(F)$ and $\tilde{R}(GH) = \tilde{R}(F)$, point 1 in Lemma 6 gives $w(F - GH) > w(F)$. For U, V, we first compute u, v in $\mathbb{F}_k[y]$ such that $u\,g + v\,h = 1$, $\deg(u) < \deg(h)$, $\deg(v) < \deg(g)$ with $\mathcal{O}_\epsilon(\deg(gh)\,f_{k-1}) \subset \mathcal{O}_\epsilon(d)$ operations in $\mathbb{F}$ [7, Corollary 11.9]. We necessarily have $v \in \mathbb{F}_k[y^q]$ and $u = y^t\,\mathfrak{u}$ with $\mathfrak{u} \in \mathbb{F}_k[y^q]$, so we may apply again Proposition 6 to compute $U, V \in \mathbb{L}[x]$ such that $\tilde{R}(U) = u$, $\tilde{R}(V) = v$, $\deg(U) < \deg(H)$, $\deg(V) < \deg(G)$, $w(U) = -w(H)$ and $w(V) = -w(G)$ within the same complexity bound. We get $\tilde{R}(UG) + \tilde{R}(VH) = ug + vh = 1 \neq 0$ and Lemma 6 (point 1) implies that $w(UG + VH) = w(UG) = w(VH)$, so that $\tilde{R}(UG + VH) = \tilde{R}(UG) + \tilde{R}(VH)$ (point 2). We get $\tilde{R}(UG + VH) = 1 = \tilde{R}(1)$ and $w(UG + VH) = 0 = w(1)$. Point 1 again gives $w(UG + VH - 1) > w(1) = 0$. □

4.3 Lifting: a valuated Hensel Lemma

Let `QuoRem` denotes the usual euclidean algorithm.

Lemma 7. *Let A, $B \in \mathbb{L}[x]$ with B strongly monic in ϕ. Then, $Q, R = \mathtt{QuoRem}(A, B)$ satisfies $w(R) \geq w(A)$ and $w(Q) \geq w(A) - w(B)$.*

Proof. We focus on the computation of R. First note that it be computed as follows[6]: write $A = \sum_{i=0}^{N} a_i\,\phi^i$ the ϕ-adic expansion of A and $B = \phi^b + \cdots$. If $N < b$, we get $R = A$; otherwise, compute $\tilde{A} = A - a_N\,\phi^{N-b}\,B$, and apply recursively this strategy to $\tilde{A}$. As $\tilde{A} = \sum_{i=0}^{N-1} \tilde{a}_i\,\phi^i$, this procedure converges towards the unique remainder R. We now prove the result by induction on the value $N \geq b$. By linearity, we can assume $A = a_N\,\phi^N$. We then have $w(\tilde{A}) \geq w(A)$ as $w(a_N\,\phi^{N-b}\,B) = w(A)$ from the assumption $w(B) = b\,w(\phi)$. By induction, this proves the lemma for R. Result for Q is then a straightforward consequence, as $w(Q\,B) = w(A - R)$. □

This lemma will enable us to prove that the classical Hensel lemma [7, Algorithm 15.10], when starting with correct initial polynomials, "double the precision" according to an extended valuation (w, ϕ). The only difference with the classical algorithm is the way we truncate polynomials: for any polynomial $F \in \mathbb{A}[x]$ and $n \in \mathbb{Z}$, we denote $\lceil F \rceil^n = \lceil F \rceil_{k+1}^n$

[6] in practice, we use the classical algorithm of $\mathbb{A}[x]$, this is only for this proof purpose

the "truncation of F according to the valuation $w = v_{k+1}$", defined recursively as follows. We let $\lceil F \rceil_0^n$ the usual truncation of F up to precision π^n and, if $F = \sum_i a_i \, \phi_k^i$ and $k \geq 0$, then $\lceil F \rceil_{k+1}^n = \sum_i \lceil a_i \rceil_k^{\frac{n - v_{k+1}(\phi_k^i)}{q_k}} \phi_k^i$. This is indeed a natural definition, as $\deg(a_i) < \deg(\phi_k)$ implies $v_{k+1}(a_i) = q_k \, v_k(a_i)$. In other words, we remove all terms of F that have w-valuation greater than n in its $(\phi_0, \ldots, \phi_k)$-multiadic expansion [27, Section 3].

Lemma 8. *Let $G \in \mathbb{L}[x]$ with $\deg(G) \leq d$ and $n \in \mathbb{N}$. We can compute $\lceil G \rceil^n$ in $\mathcal{O}\left(d(n/w(\pi) - v_0(G))\right)$ operations in $\mathbb{F}$.*

Proof. Compute $\lceil \pi^{-v_0(G)} G \rceil^{n - v_0(G) w(\pi)}$ with precision $\lceil n/w(\pi) \rceil - v_0(G)$ following the recursive definition above. The main cost is to compute the ϕ_k-adic expansions, as in the proof of Lemma 3. □

Algorithm `HenselStep` below takes as input $F, G, H \in \mathbb{A}[x]$ with H strongly monic in ϕ, $U, V \in \mathbb{L}[x]$ and $n \in \mathbb{N}^\times$ such that

- $w(F - G\,H) \geq w(F) + n$, with $\deg(F) \geq \deg(G) + \deg(H)$
- $w(U\,G + V\,H - 1) \geq n$, with $\deg(U) < \deg(H)$, $\deg(V) < \deg(G)$, $w(U) = -w(G)$ and $w(V) = -w(H)$.

It computes $\tilde{G}, \tilde{H} \in \mathbb{A}[x]$ with $\tilde{H}$ strongly monic in ϕ and $\tilde{U}, \tilde{V} \in \mathbb{L}[x]$ such that

- $w(F - \tilde{G}\,\tilde{H}) \geq w(F) + 2\,n$, with $\deg(\tilde{H}) = \deg(H)$, $\deg(F) \geq \deg(\tilde{G}) + \deg(\tilde{H})$, $w(\tilde{H} - H) \geq w(H) + n$ and $w(\tilde{G} - G) \geq w(G) + n$.
- $w(\tilde{U}\,\tilde{G} + \tilde{V}\,\tilde{H} - 1) \geq 2\,n$, with $\deg(\tilde{U}) < \deg(\tilde{H})$, $\deg(\tilde{V}) < \deg(\tilde{G})$, $w(\tilde{U}) = -w(\tilde{G})$ and $w(\tilde{V}) = -w(\tilde{H})$.

Algorithm: `HenselStep`(F, G, H, U, V, n)

$E \leftarrow \lceil F - G\,H \rceil^{w(F)+2n}$;
$Q, R \leftarrow \lceil \texttt{QuoRem}(U\,E, H) \rceil^{w(F)+2n}$;
$\tilde{G} \leftarrow \lceil G + E\,V + Q\,G \rceil^{w(G)+2n}$;
$\tilde{H} \leftarrow \lceil H + R \rceil^{w(H)+2n}$;
$B \leftarrow \lceil U\,\tilde{G} + V\,\tilde{H} - 1 \rceil^{2n}$;
$C, D \leftarrow \lceil \texttt{QuoRem}(U\,B, \tilde{H}) \rceil^{2n}$;
$\tilde{U} \leftarrow \lceil U - D \rceil^{2n - w(G)}$;
$\tilde{V} \leftarrow \lceil V - B\,V - C\,\tilde{G} \rceil^{2n - w(H)}$;
return $\tilde{H}$, $\tilde{G}$, $\tilde{U}$, $\tilde{V}$

Proposition 8. *Algorithm* `HenselStep` *is correct. It takes less than $\mathcal{O}\left(\frac{n + w(F)}{w(\pi)} d\right)$ operations in $\mathbb{F}$.*

Proof. We start with the correctness. H being strongly monic in H, Lemma 7 ensures $w(R) \geq w(H) + n > w(H)$ as $w(E) \geq w(F) + n$ by assumption. As $\deg(R) < \deg(H)$, this proves $\deg(\tilde{H}) = \deg(H)$, $w(\tilde{H} - H) \geq w(H) + n$ and $\tilde{H}$ strongly monic in ϕ. As $w(Q) \geq n$, we also get $w(\tilde{G} - G) = w(VE + QG) \geq w(G) + n$. Using these results and the equality

$$\lceil F - \tilde{G}\,\tilde{H} \rceil^{w(F)+2n} = \lceil E\,(1 - U\,G - V\,H) - R\,(E\,V + Q\,G) \rceil^{w(F)+2n},$$

we get $\lceil F - \tilde{G}\,\tilde{H} \rceil^{w(F)+2n} = 0$, i.e. $w(F - \tilde{G}\,\tilde{H}) \geq w(F) + 2\,n$, from which we also deduce $\deg(F) \geq \deg(\tilde{G}) + \deg(\tilde{H})$.
Similarly, using $\lceil C\,\tilde{H} + D \rceil^{2n} = \lceil U\,B \rceil^{2n}$, we get

$$\lceil \tilde{U}\,\tilde{G} + \tilde{V}\,\tilde{H} - 1 \rceil^{2n} = \lceil U\,\tilde{G} + V\,\tilde{H} - 1 - B\,(U\,\tilde{G} + V\,\tilde{H}) \rceil^{2n} = \lceil B^2 \rceil^{2n} = 0,$$

proving $w(\tilde{U}\,\tilde{G} + \tilde{V}\,\tilde{H} - 1) \geq 2\,n$. Using Lemma 7 once again, we can deduce easily the remaining properties of the output.
Finally, the complexity is a direct consequence of Lemma 8 (and the usual complexity of the Hensel algorithm [7, Theorem 15.11]). □

Corollary 2. *Let $F \in \mathbb{A}[x]$, $\mathbf{t}$ a type dividing F, a factorisation $\tilde{R}(F) = h_0 \cdots h_r h_\infty$ and $n \in \mathbb{N}$. One can compute $F_0, \ldots, F_r, F_\infty$ such that $\tilde{R}(F_i) = h_i$ and $w(F - F_0 \cdots F_r F_\infty) > n + w(F)$ in $\mathcal{O}\left(\frac{n+w(F)}{w(\pi)}\,d\right)$ operations in $\mathbb{F}$. If $\deg(F_{\mathbf{t}}) > d/2$, this is $\mathcal{O}\,(n\,d/w(\pi) + \delta)$.*

Proof. This algorithm is similar to the classical multifactor Hensel lifting [7, Algorithm 15.17]. We start by building a subproduct tree of the factorisation $\tilde{R}(F) = h_0 \cdots h_r\,h_\infty$. Then, for each node (from top to bottom), we initialise the polynomials G, H, U, V, and run `HenselStep` until we reach the required precision. Complexity and correctness follows from Propositions 7 and 8. If $\deg(F_{\mathbf{t}}) > d/2$, Lemma 2 implies $\frac{w(F)}{w(\pi)} < \frac{4\,\delta}{d}$. □

In order to get a factorisation $F = F_0 \cdots F_r F_\infty \mod \pi^{\sigma+1}$ for some given $\sigma \in \mathbb{N}$, we need to relate the valuations v_0 and w:

Lemma 9. *Let $G \in \mathbb{L}[x]$ and denote $N = \lfloor \deg(G)/\deg(\phi) \rfloor + 1$. Then we have $v_0(G)w(\pi) \leq w(G) \leq v_0(G)w(\pi) + Nw(\phi)$.*

Proof. First inequality is clear. For the second inequality, we can safely suppose that $v_0(G) = 0$. We first consider the case $\deg(G) < \deg(\phi)$ (so $N = 1$) and we proceed by induction (remind that $\phi = \phi_k$ and $w = v_{k+1}$). If $k = 0$, the claim is obvious. Assume $k \geq 1$. Then, from (1) and the assumption $\deg(G) < \deg(\phi_k)$, we have $v_{k+1}(G) = q_k v_k(G)$. We also get $v_{k+1}(\phi_k) = q_k v_k(\phi_k) + m_k$. It is thus sufficient to show that $v_k(G) \leq v_k(\phi_k)$. Write $G = \sum_{0 \leq i < q_{k-1}\ell_{k-1}} a'_i \phi_{k-1}^i$. As there is at least one a'_i not dividable by π, i.e. $v_k(a'_i) \leq v_k(\phi_{k-1})$ by recursion, we get $v_k(G) \leq q_{k-1}\ell_{k-1}v_k(\phi_{k-1})$. As $v_k(\phi_k) = q_{k-1}\ell_{k-1}v_k(\phi_{k-1})$ by [10, Theorem 2.11], this concludes. If $\deg(G) \geq \deg(\phi)$, we write $G = \sum_{i=0}^{N-1} a_i\,\phi^i$, $\deg(a_i) < \deg(\phi)$. Let i such that $v_0(a_i) = 0$. We get $w(G) \leq w(a_i\phi^i) \leq (i+1)w(\phi) \leq Nw(\phi)$. □

Theorem 3. *There exists an algorithm* `SlopeFacto` *that given* $F \in \mathbb{A}[x]$, $\mathbf{t}$ *a type dividing* F, *a factorisation* $\tilde{R}(F) = h_0 \cdots h_r h_\infty$ *and* $\sigma \in \mathbb{N}$, *computes a factorisation* $F = F_0 \cdots F_r F_\infty \mod \pi^{\sigma+1}$ *with* $\tilde{R}(F_i) = h_i$. *It takes* $\mathcal{O}\left(\sigma\, d + \frac{d^2\, w(\phi)}{\deg(\phi) w(\pi)}\right)$ *operations in* $\mathbb{F}$. *If* $\deg(F_\mathbf{t}) > d/2$, *this is* $\mathcal{O}(d\,\sigma + \delta)$.

Proof. Let $n = w(\pi^\sigma) - w(F) + (\lfloor d/\deg(\phi) \rfloor + 1)\, w(\phi)$ and apply Corollary 2 with n. Then $G = F - F_0 \cdots F_r F_\infty$ satisfies $\deg(G) < d$ and $w(G) > n + w(F)$, i.e. $v_0(G) > \sigma$ by Lemma 9. The first complexity bound follows from Corollary 2. As $F_\mathbf{t}$ is strongly monic in ϕ, we have $w(\phi)/\deg(\phi) = w(F_\mathbf{t})/\deg(F_\mathbf{t})$ and the second bound is a consequence of Lemma 2, which implies $\frac{w(F_\mathbf{t})}{w(\pi)} < \frac{2\,\delta}{\deg(F_\mathbf{t})}$. □

5 A fast factorisation algorithm

From the previous two sections, we can easily deduce a factorisation algorithm. Let $n \geq \delta$ ([3, Theorem 3.13] shows that this is a sufficient precision to detect the whole factorisation).

1. Run algorithm `FastIrreducible` with precision $2\delta/d$. We conclude that F is irreducible (and return F) or we get a type $\mathbf{t}_{k-1}$.
2. Compute the factorisation $\tilde{R}_k(F) = h_0 h_1 \cdots h_r$ in $\mathbb{F}_k[y]$ (we have $h_\infty = 1$ since F is of type $\mathbf{t}_{k-1}$).
3. Use Algorithm `SlopeFacto` of Section 4 to get a factorisation $F = F_0 F_1 \cdots F_r$ with precision n.
4. Go back to Step 1 for each H_i.

If ρ is the number of irreducible factors of F, this requires at most ρ univariate factorisations and $\log_2(d)$ univariate irreducible tests over $\mathbb{F}_k[y]$, plus a number of operations over $\mathbb{F}$ bounded by $\mathcal{O}_\epsilon(\rho\, n\, d)$ under Assumption 1, and $\mathcal{O}_\epsilon(\rho\, n\, d + \delta^2)$ otherwise.
This section describes a divide and conquer strategy that will reduce the computations over $\mathbb{F}$ to respectively $\mathcal{O}_\epsilon(d\, n)$ and $\mathcal{O}_\epsilon(d\, n + \delta^2)$. The idea is the following:

1. Find a type $\mathbf{t}$ such that $F_\mathbf{t}$ has degree $> d/2$ and that either $F_\mathbf{t}$ is irreducible, either any of its proper factor has degree $\leq d/2$.
2. Use Algorithm `SlopeFacto` of Section 4 to get a factorisation $F = F_0\, F_1 \cdots F_r F_\infty$ with $\mathsf{F}_\mathbf{t} = F_0\, F_1 \cdots F_r$.
3. Apply recursively this strategy to all factors that are not known irreducible.

5.1 Finding a dividing type

The aim of this section is to either find the irreducible factor of degree $> d/2$ if it exists, either find a factorisation $F = F_0\, \mathsf{F}_1 \cdots F_r\, F_\infty$ with each factor of degree $\leq d/2$, and to do so with precision $\sigma \leq 4\delta/d$. We start with the algorithm.

Algorithm: `DividingType`(F, σ)

Input: $F \in \mathbb{A}[x]$ monic separable, $\sigma \in \mathbb{N}$ the precision used.
Output: A type $\mathbf{t}$ with the properties described above.

1 $H \leftarrow F$, $d \leftarrow \deg(F)$;
2 **while** *True* **do**
3 $\quad (b, \mathbf{t}) \leftarrow$ `FastIrreducible`(H, σ);
4 $\quad$ **if** $b =$ *True* **then return** $\mathbf{t}$;
5 $\quad$ Compute $\tilde{R}(H) = g\,h$, $h = h_i$ with highest degree;
6 $\quad$ **if** $\deg(h)\,\deg(\phi) \leq d/2$ **then return** $\mathbf{t}$;
7 $\quad (G, H) \leftarrow$`SlopeFacto`$(H, \mathbf{t}, [g, h], \sigma)$;

Remark 4. *Note that the notation* `FastIrreducible`(H, σ) *means to run Algorithm* `FastIrreducible` *with parameter* H *and precision* σ*. Also, at Line 5,* H *is always of type* $\mathbf{t}$*, so that the factorisation* $\tilde{R}(H)$ *does not involve* h_∞*.*

Proposition 9. *The function call* `DividingType`$(F, 4\delta/d)$ *is correct. It takes less than* $\mathcal{O}_\epsilon(\rho\,\delta)$ *operations in* $\mathbb{F}$ *if Assumption 1 is satisfied, and* $\mathcal{O}_\epsilon(\rho\,\delta + \delta^2)$ *otherwise, plus at most* ρ *factorisations and* $\log_2(d)$ *irreducibility tests over some* $\mathbb{F}_k$*.*

Proof. Note first that $\deg(H)$ decreases at each loop (via the factorisation of Line 7), proving the ending of the algorithm. Also, the test of Line 6 ensures $\deg(H) > d/2$, so that $2\delta(H)/\deg(H) \leq 4\delta/d$ at any point of the algorithm. `FastIrreducible`$(H, 4\delta/d)$ will thus return a correct answer thanks to Lemma 3. This precision is also sufficient to compute $\tilde{R}(H)$ at Line 5 (use Proposition 4). As the precision used is high enough, correctness is straightforward: we output $\mathbf{t}$ when either $\mathbb{F}_\mathbf{t}$ is irreducible (Line 4), either all of its factor has degree $\leq d/2$ (Line 6). Finally, the complexity is a direct consequence of Theorems 2 and 3 (and Proposition 5). □

Remark 5. *There are various straightforward improvements to this algorithm: for instance, one can provide the type computed so far to* `FastIrreducible` *to avoid some already done computations. They do not appear to make the reading easier, but are needed to run less than* $\log_2(d)$ *irreducible tests over some* $\mathbb{F}_k[y]$*.*

5.2 The divide and conquer algorithm

Thanks to this result, we derive a fast factorisation algorithm:

Remark 6. *At Line 6,* F_0 *and* F_∞ *are* known irreducible *if they have degree* ≤ 1 *while* F_i *is* known irreducible *if* $ord_\mathbf{t} F_i = 1$ *for* $i = 1, \ldots, r$*.*

Theorem 4. *Let* $n \geq \delta$*. A function call* `Factorisation`(F, n) *returns the correct output. It requires at most* ρ *factorisations and* $\log_2(d)$ *irreducibility tests in some* $\mathbb{F}_k[y]$*, plus* $\mathcal{O}_\epsilon(d\,n)$ *operations in* $\mathbb{F}$ *if Assumption 1 is satisfied, and* $\mathcal{O}_\epsilon(d\,n + \delta^2)$ *otherwise.*

Algorithm: `Factorisation`(F, n)

Input: $F \in \mathbb{A}[x]$ monic separable, $n \in \mathbb{N}$ the precision.

Output: The list $F_1, \cdots, F_\rho$ of irreducible factors of $\mathbb{F}$ known with precision n.

1 $\mathbf{t} \leftarrow$`DividingType`$(F, 4\,n/d)$;
2 Compute $\tilde{R}(F) = h_0\, h_1 \cdots h_r\, h_\infty$;
3 $F_0, \cdots, F_r, F_\infty \leftarrow$`SlopeFacto`$(F, \mathbf{t}, [h_0, \cdots, h_r, h_\infty], n)$;
4 $\mathcal{R} \leftarrow \{\}$;
5 **for** $i \in \{0, \cdots, r, \infty\}$ **do**
6 **if** F_i *is known irreducible* **then** $\mathcal{R} \leftarrow \mathcal{R} \cup \{F_i\}$;
7 **else** $\mathcal{R} \leftarrow \mathcal{R} \cup \{$`Factorisation`$(F_i, n)\}$;
8 **return** $\mathcal{R}$;

Proof. This is a direct consequence of Proposition 9 and Theorem 3, as $\sum_i \deg(F_i) = d$ at Line 3, and $\deg(F_i) < d/2$, so that the number of lines of the recursive calls tree is less than $\log_2(d)$. Precision δ is sufficient to detect all irreducible factors from [3, Theorem 3.13]. □

Remark 7. *As in Section 3, we do not know in advance δ. We proceed similarly as explained in Section 3.2.*

When considering $n \in \mathcal{O}(\delta)$, then up to the cost of the residual factorisations, Theorem 4 improves [3, Theorem 5.15] by a factor δ if Assumption 1 is satisfied, and by a factor $\min(d, \delta)$ otherwise.

5.3 Residual factorisations over finite fields

If $\mathrm{Card}(\mathbb{F}) = q$, we factorise $F \bmod \pi$ with an expected $\mathcal{O}\tilde{}(d^2 + d\,\log(q))$ operations over $\mathbb{F}$ by [7, Corollary 14.30]. The remaining residual factorisations performed by algorithm `Factorisation` use then an expected $\mathcal{O}_\epsilon(d\delta\,\log(q))$ operations in $\mathbb{F}$, see the proof of [3, Theorem 5.14]. Together with Theorem 4, this proves:

Corollary 3. *Let $F \in \mathbb{Z}_p[x]$, separable with p small. Given the univariate factorisation of $F \bmod p$, we compute the irreducible factors of F with precision $n \geq \delta$ with an expected $\mathcal{O}_\epsilon(nd)$ operations over $\mathbb{F}_p$ if $p > d$ and $\mathcal{O}_\epsilon(nd + \delta^2)$ otherwise. The same result holds for $F \in \mathbb{F}_p[[t]][x]$.*

Corollary 3 improves significantly [3, Theorem 5.18] which leads to the complexity estimate $\mathcal{O}_\epsilon(d\delta^2 + nd^2)$ under the same hypothesis.

Remark 8. *Corollary 3 requires a priori to compute a primitive representation of $\mathbb{F}_k$ over $\mathbb{F}$ before applying the factorisation algorithm [7, Corollary 14.30] (this issue doesn't seem to be considered in [3]). To this aim, we may use a Las Vegas subroutine [29, Proposition 4] whose complexity fits in our aimed bound. There are recent faster factorisation algorithms [15, Corollary 3] (both for primitive or triangular representation), but not yet implemented.*

5.4 Avoiding residual factorisations

If $\mathbb{F}$ has large cardinal, then the residual factorisations will probably dominate the cost of the all algorithm. It might thus be preferable to rely on dynamic evaluation [14, 29] : we allow the P_k's to be square-free, hence the $\mathbb{F}_k$'s to be product of fields. If at some point we find a zero divisor (while computing some gcd's), then we pursue over each discovered summand of $\mathbb{F}_k$ (or we return false if we run an irreducibility test). At the end, we perform a unique residual factorisation (of expected small degree) for each discovered factor of F to deduce its irreducible factorisation. Notice that this last step might be useless, depending on the arithmetic information we want to compute. It is not needed for instance if we only want the ramification indices (e.g. for the computation of the genus of a plane curve [29]), or the valuation δ of the discriminant of F [22], or the equisingularity type of a germ of plane curve [28]. In these cases, we may only use square-free residual factorisations (fast gcd's) and the underlying algorithm is entirely deterministic.

6 Direct Applications

Our complexity estimates have several direct consequences for various tasks of computational number theory and algebraic geometry. In what follows, the complexity results are given up to the cost of the residual univariate factorisations.

OM-factorisation Let $\mathbb{K} = \mathbb{Q}[x]/(F)$ be a number field and let $p \in \mathbb{Z}$ a prime. The first main consequence of Theorem 4 is that we compute an *Okutsu-Montes (OM) representation* of the prime ideals dividing p with $\mathcal{O}_\epsilon(d\delta)$ operations in $\mathbb{F}_p$ if $p > d$ or $\mathcal{O}_\epsilon(d\delta + \delta^2)$ otherwise, improving [3, Theorem 5.15] respectively by a factor δ or $\min(d, \delta)$. These OM-representations carry on essential data about the corresponding extensions of local fields and give useful tools for various local and global arithmetic tasks (see e.g. [9]).

Valuations of discriminants and resultants As a straightforward application of fast OM-factorisation, we may use [22] to compute $\delta = v_p(\mathrm{Disc} F)$ with $\mathcal{O}_\epsilon(d\delta)$ operations in $\mathbb{F}_p$ if $p > d$ or $\mathcal{O}_\epsilon(d\delta + \delta^2)$ otherwise, improving [22, Theorem 2.5] respectively by a factor δ or $\min(d, \delta)$. If $G, H \in \mathbb{Z}[x]$ are coprime of degrees at most d, we compute $\delta = v_p(\mathrm{Res}(G, H))$ within the same bound, improving now [22, Theorem 3.3] by a factor δ or $\min(d, \delta)$. As mentioned in Section 5.4, we may rely on dynamic evaluation for this task: only square-free residual factorisation is required and the algorithm is deterministic.

Local integral basis Combined with Bauch's algorithm [2], our results allow to compute a $\mathbb{Z}_{(p)}$-basis of the integral closure of $\mathbb{Z}_{(p)}$ in $\mathbb{K}$ with $\mathcal{O}_\epsilon(d^2\delta)$ operations in $\mathbb{F}_p$ if $p > d$ or $\mathcal{O}_\epsilon(d^2\delta + \delta^2)$ otherwise, improving $\mathcal{O}_\epsilon(d^2\delta + d\delta^2)$ of [2, Lemma 3.10]. This impacts the computation of a global integral basis of $\mathbb{K}/\mathbb{Q}$, obtained from local ones via Hermite normal forms and Chinese remaindering.

Function fields All complexity results above extend trivially to function fields satisfying Assumption 1. In this context, we may use moreover the Riemann-Hurwitz formula to compute the genus of a degree d plane curve over $\mathbb{F}$ within $\mathcal{O}_\epsilon(d^3)$ operations in $\mathbb{F}$ (following a similar strategy than in [29], but using the easier to implement algorithm `Factorisation`). Here again, only square-free residual factorisation is required and the algorithm is deterministic.

More applications There are several other computational consequences for global fields, as computing the valuation at a prime ideal, factoring fractional ideals or Chinese remaindering [21], factoring bivariate polynomials [30], computing roots of polynomials [23], or computing Riemann-Roch spaces [13], but going into these details is beyond the scope of this paper.

References

[1] S. Abhyankar. *Algebraic Geometry for Scientists and Engineers*, volume 35 of *Mathematical surveys and monographs*. Amer. Math. Soc., 1990.

[2] J.-D. Bauch. Computation of integral bases. *Journal of Number Theory*, 165:382–407, 2016.

[3] J.-D. Bauch, E. Nart, and H. Stainsby. Complexity of the OM factorizations of polynomials over local fields. *LMS Journal of Computation and Mathematics*, 16:139–171, 2013.

[4] D. G. Cantor and D. Gordon. Factoring polynomials over p-adic fields. In *ANTS-IV*, volume 1838 of *LNCS*. Springer Verlag, 2000.

[5] X. Caruso, D. Roe, and T. Vaccon. Division and slope factorization of p-adic polynomials. In *Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation*, ISSAC '16, pages 159–166, New York, NY, USA, 2016. ACM.

[6] D. Ford, S. Pauli, and X.-F. Roblot. A fast algorithm for polynomial factorization over $\mathbb{Q}_p$. *Journal de Théorie des Nombres de Bordeaux*, 14:151–169, 2002.

[7] J. v. z. Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, New York, NY, USA, 3rd edition, 2013.

[8] J. Guàrdia, J. Montes, and E. Nart. Okutsu invariants and newton polygons. *Acta Arithmetica*, 145:83–108, 2010.

[9] J. Guàrdia, J. Montes, and E. Nart. Higher Newton polygons in the computation of discriminants and prime ideal decomposition in number fields. *Journal de Théorie des Nombres de Bordeaux*, 23(3):667–696, 2011.

[10] J. Guàrdia, J. Montes, and E. Nart. Newton polygons of higher order in algebraic number theory. *Transsactions of the American Mathematical Society*, 364:361–416, 2012.

[11] J. Guàrdia, E. Nart, and S. Pauli. Single-factor lifting and factorization of polynomials over local fields. *Journal of Symbolic Computation*, 47(11):1318 – 1346, 2012.

[12] W. Hart and A. Novocin. Practical divide-and-conquer algorithms for polynomial arithmetic. In V. P. Gerdt, W. Koepf, E. W. Mayr, and E. V. Vorozhtsov, editors, *Computer Algebra in Scientific Computing*, pages 200–214, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[13] F. Hess. Computing riemann–roch spaces in algebraic function fields and related topics. *Journal of Symbolic Computation*, 33(4):425–445, 2002.

[14] J. v. d. Hoeven and G. Lecerf. Directed evaluation. *Journal of Complexity*, 60:101498, 2020.

[15] J. v. d. Hoeven and G. Lecerf. Univariate polynomial factorization over finite fields with large extension degree. *Applicable Algebra in Engineering, Communication and Computing*, Jan 2022.

[16] E. Kaltofen. Greatest common divisors of polynomials given by straight-line programs. *J. ACM*, 35(1):231–264, Jan. 1988.

[17] R. Lebreton. Relaxed hensel lifting of triangular sets. *Journal of Symbolic Computation*, 68, Part 2:230 – 258, 2015. Effective Methods in Algebraic Geometry.

[18] S. Mac Lane. A construction for prime ideals as absolute values of an algebraic field. *Duke Math. J.*, 2(3):492–510, 1936.

[19] S. MacLane. A construction for absolute values in polynomial rings. *Trans. Amer. Math. Soc.*, 40(3):363–395, 1936.

[20] G. Moroz and É. Schost. A fast algorithm for computing the truncated resultant. In *ISSAC '16: Proceedings of the twenty-first international symposium on Symbolic and algebraic computation*, pages 1–8, New York, NY, USA, 2016. ACM.

[21] E. Nart. Okutsu-montes representations of prime ideals of one-dimensional integral closures. *Publicacions Matematiques - PUBL MAT*, 55, 07 2011.

[22] E. Nart. Local computation of differents and discriminants. *Mathematics of Computation*, 83(287):1513–1534, 2014.

[23] V. Neiger, J. Rosenkilde, and E. Schost. Fast computation of the roots of polynomials over the ring of power series. In *ISSAC'17*, pages 349–356. ACM, 2017.

[24] S. Pauli. Factoring polynomials over local fields. *J. Symb. Comp.*, 32:533–547, 2001.

[25] S. Pauli. Factoring polynomials over local fields, ii. In *ANTS-IX*, LNCS. Springer Verlag, 2010.

[26] P. Popescu-Pampu. Approximate roots. *Fields Institute Communiations*, 33:1–37, 2002.

[27] A. Poteaux and M. Weimann. A quasi-linear irreducibility test in $\mathbb{K}[[x]][y]$. *Preprint*, pages 1–21, 2018.

[28] A. Poteaux and M. Weimann. Computing the equisingularity type of a pseudo-irreducible polynomial. *Applicable Algebra in Engineering, Communication and Computing*, 31:435 – 460, 2020.

[29] A. Poteaux and M. Weimann. Computing Puiseux series: a fast divide and conquer algorithm. *Annales Henri Lebesgue*, 4:1061–1102, 2021.

[30] M. Weimann. Bivariate factorization using a critical fiber. *Journal of Foundations of Computational Mathematics*, pages 1–45, 2016.