Division and Slope Factorization of p-Adic Polynomials

Xavier Caruso
Université Rennes 1
xavier.caruso@normalesup.org

David Roe
University of Pittsburgh
roed.math@gmail.com

Tristan Vaccon
JSPS–Rikkyo University
vaccon@rikkyo.ac.jp

ABSTRACT

We study two important operations on polynomials defined over complete discrete valuation fields: Euclidean division and factorization. In particular, we design a simple and efficient algorithm for computing slope factorizations, based on Newton iteration. One of its main features is that we avoid working with fractional exponents. We pay particular attention to stability, and analyze the behavior of the algorithm using several precision models.

CCS Concepts

•**Computing methodologies** → **Algebraic algorithms;**

Keywords

Algorithms, p-adic precision, Newton polygon, factorization

1. INTRODUCTION

Polynomial factorization is a fundamental problem in computational algebra. The algorithms used to solve it depend on the ring of coefficients, with finite fields, local fields, number fields and rings of integers of particular interest to number theorists. In this article, we focus on a task that forms a building block for factorization algorithms over complete discrete valuation fields: the decomposition into factors based on the slopes of the Newton polygon.

The Newton polygon of a polynomial $f(X) = \sum a_i X^i$ over such a field is given by the convex hull of the points $(i, \operatorname{val}(a_i))$ and the point $(0, +\infty)$. The lower boundary of this polygon consists of line segments (x_j, y_j) – (x_{j+1}, y_{j+1}) of slope s_j. The slope factorization of $f(X)$ expresses $f(X)$ as a product of polynomials $g_j(X)$ with degree $x_{j+1} - x_j$ whose roots all have valuation $-s_j$. Our main result is a new algorithm for computing these $g_j(X)$.

Polynomial factorization over local fields has seen a great deal of progress recently [7–10] following an algorithm of Montes. Slope factorization provides a subroutine in such algorithms [10, Section 2]. Note that our algorithm lifts a *slope factor* corresponding to a segment of the Newton polygon, rather than the irreducible factor found by single factor lifting [8]. The slope factors are easier to find; as a consequence our iteration is less involved. We also remark that the methods introduced in this paper extend partially to the non-commutative setting and appear this way as an essential building block in several decomposition algorithms of p-adic Galois representations and p-adic differential equations [3].

Any computation with p-adic fields must work with approximations modulo finite powers of p, and one of the key requirements in designing an algorithm is an analysis of how the precision of the variables evolve over the computation. We work with precision models developed by the same authors [4, Section 4.2], focusing on the lattice and Newton models. As part of the analysis of the slope factorization algorithm, we describe how the precision of the quotient and remainder depend on the input polynomials in Euclidean division.

Main Results. Suppose that the Newton polygon of $P(X) = \sum_{i=0}^{n} a_i X^i$ has an extremal point at abscissa d. Set $A_0 = \sum_{i=0}^{d} a_i X^i$, $V_0 = 1$ and

$$\begin{aligned} A_{i+1} &= A_i + (V_i P \mathbin{\%} A_i) \\ B_{i+1} &= P \mathbin{/\!/} A_{i+1} \\ V_{i+1} &= (2V_i - V_i^2 B_{i+1}) \mathbin{\%} A_{i+1}. \end{aligned}$$

where $S \mathbin{/\!/} T$ and $S \mathbin{\%} T$ denotes the quotient and the remainder in the Euclidean division of S by T respectively. Our main result is Theorem 4.1, which states that the sequence (A_i) converges quadratically to a divisor of P. This provides a quasi-optimal simple-to-implement algorithm for computing slope factorizations. We moreover carry out a careful study of the precision and, applying a strategy coming from [4], we end up with an algorithm that outputs optimal results regarding to accuracy.

In order to prove Theorem 4.1, we also determine the precision of the quotient and remainder in Euclidean division, which may be of independent interest. These results are found in Section 3.2.

Organization of the paper. After setting notation, in Section 2 we recall various models for tracking precision in polynomial arithmetic. We give some background on Newton polygons and explain how using lattices to store precision can allow for extra *diffuse* p-adic digits that are not localized on any single coefficient.

In Section 3, we consider Euclidean division. We describe in Theorem 3.2 how the Newton polygons of the quotient and remainder depend on numerator and denominator. We

ISSAC'16, July 19–22, 2016, Waterloo, ON, Canada

ACM ISBN 978-1-4503-4380-0/16/07…$15.00

DOI: http://dx.doi.org/10.1145/2930889.2930897

use this result to describe in Proposition 3.3 the precision evolution in Euclidean division using the Newton precision model. We then compare the precision performance of Euclidean division in the jagged, Newton and lattice models experimentally, finding different behavior depending on the modulus.

Finally, in Section 4 we describe our slope factorization algorithm, which is based on a Newton iteration. Unlike other algorithms for slope factorization, ours does not require working with fractional exponents (compare for instance with [13, § 6]). In Theorem 4.1 we define a sequence of polynomials that will converge to the factors determined by an extremal point in the Newton polygon. We then discuss the precision behavior of the algorithm.

Notations. Throughout this paper, we fix a complete discrete valuation field K; we denote by $\operatorname{val} : K \to \mathbb{Z} \cup \{+\infty\}$ the valuation on it and by W its ring of integers (*i.e.* the set of elements with nonnegative valuation). We assume that val is normalized so that it is surjective and denote by π a uniformizer of K, that is an element of valuation 1. Denoting by $S \subset W$ a fixed set of representatives of the classes modulo π and assuming $0 \in S$, one can prove that each element in $x \in K$ can be represented uniquely as a convergent series:

$$x = \sum_{i=\operatorname{val}(x)}^{+\infty} a_i \pi^i \quad \text{with} \quad a_i \in S. \tag{1}$$

The two most important examples are the field of p-adic numbers $K = \mathbb{Q}_p$ and the field of Laurent series $K = k((t))$ over a field k. The valuation on them are the p-adic valuation and the usual valuation of a Laurent series respectively. Their ring of integers are therefore $\mathbb{Z}_p$ and $k[[t]]$ respectively. A distinguished uniformizer is p and t whereas a possible set S is $\{0, \dots, p-1\}$ and k respectively. The reader who is not familiar with complete discrete valuation fields may assume (without sacrifying too much to the generality) that K is one of the two aforementioned examples.

In what follows, the notation $K[X]$ refers to the ring of univariate polynomials with coefficients in K. The subspace of polynomials of degree at most n (resp. exactly n) is denoted by $K_{\leq n}[X]$ (resp. $K_n[X]$).

2. PRECISION DATA

Elements in K (and *a fortiori* in $K[X]$) carry an infinite amount of information. They thus cannot be stored entirely in the memory of a computer and have to be truncated. Elements of K are usually represented by truncating Eq.(1) as follows:

$$x = \sum_{i=v}^{N-1} a_i \pi^i + O(\pi^N) \tag{2}$$

where N is an integer called the *absolute precision* and the notation $O(\pi^N)$ means that the coefficients a_i for $i \geq N$ are discarded. If $N > v$ and $a_v \neq 0$, the integer v is the valuation of x and the difference $N - v$ is called the *relative precision*. Alternatively, one may think that the writing (2) represents a subset of K which consists of all elements in K for which the a_i's in the range $[v, N-1]$ are those specified. From the metric point of view, this is a ball (centered at any point inside it).

It is worth noting that tracking precision using this representation is rather easy. For example, if x and y are known with absolute (resp. relative) precision N_x and N_y respectively, one can compute the sum $x+y$ (resp. the product xy) at absolute (resp. relative) precision $\min(N_x, N_y)$. Computations with p-adic and Laurent series are often handled this way on symbolic computation softwares.

2.1 Precision for polynomials

The situation is much more subtle when we are working with a collection of elements of K (*e.g.* a polynomial) and not just a single one. Indeed, several precision data may be considered and, as we shall see later, each of them has its own interest. Below we detail three models of precision for the special case of polynomials.

Flat precision. The simplest method for tracking the precision of a polynomial is to record each coefficient modulo a fixed power of π. While easy to analyze and implement, this method suffers when applied to polynomials whose Newton polygons are far from flat.

Jagged precision. The next obvious approach is to record the precision of each coefficient individually, a method that we will refer to as *jagged* precision. Jagged precision is commonly implemented in computer algebra systems, since standard polynomial algorithms can be written for generic coefficient rings. However, these generic implementations often have suboptimal precision behavior, since combining intermediate expressions into a final answer may lose precision. Moreover, when compared to the Newton precision model, extra precision in the middle coefficients, above the Newton polygon of the remaining terms, will have no effect on any of the values of that polynomial.

Newton precision. We now move to *Newton precision* data. They can be actually seen as particular instances of jagged precision but there exist for them better representations and better algorithms.

Definition 2.1. A *Newton function of degree n* is a convex function $\varphi : [0, n] \to \mathbb{R} \cup \{+\infty\}$ which is piecewise affine, which takes a finite value at n and whose epigraph[1] $\operatorname{Epi}(\varphi)$ have extremal points with integral abscissa.

Remark 2.2. The datum of φ is equivalent to that of $\operatorname{Epi}(\varphi)$ and they can easily be represented and manipulated on a computer.

We recall that one can attach a Newton function to each polynomial. If $P(X) = \sum_{i=0}^{n} a_i X^i \in K_n[X]$, we define its Newton polygon $\operatorname{NP}(P)$ as the convex hull of the points $(i, \operatorname{val}(a_i))$ $(1 \leq i \leq n)$ together with the point at infinity $(0, +\infty)$ and then its Newton function $\operatorname{NF}(P) : [0, n] \to \mathbb{R}$ as the unique function whose epigraph is $\operatorname{NP}(P)$. It is well known [6, Section 1.6] that:

$$\begin{aligned} \operatorname{NP}(P+Q) &\subset \operatorname{Conv}\big(\operatorname{NP}(P) \cup \operatorname{NP}(Q)\big) \\ \operatorname{NP}(PQ) &= \operatorname{NP}(P) + \operatorname{NP}(Q) \end{aligned}$$

where Conv denotes the convex hull and the plus sign stands for the Minkowski sum. This translates to:

$$\begin{aligned} \operatorname{NF}(P+Q) &\geq \operatorname{NF}(P) \oplus \operatorname{NF}(Q) \\ \operatorname{NF}(PQ) &= \operatorname{NF}(P) \otimes \operatorname{NF}(Q) \end{aligned}$$

[1]Recall that the epigraph is the region above the graph.

where the operations $\oplus$ and $\otimes$ are defined accordingly. There exist classical algorithms for computing these two operations whose complexity is quasi-linear with respect to the degree.

In a similar fashion, Newton functions can be used to model precision: given a Newton function φ of degree n, we agree that a polynomial of degree at most n is given at precision $O(\varphi)$ when, for all i, its i-th coefficient is given at precision $O(\pi^{\lceil \varphi(i) \rceil})$ (where $\lceil \cdot \rceil$ is the ceiling function). In the sequel, we shall write $O(\varphi) = \sum_{i=0}^{n} O(\pi^{\lceil \varphi(i) \rceil}) \cdot X^i$ and use the notation $\sum_{i=0}^{n} a_i X^i + O(\varphi)$ (where the coefficients a_i are given by truncated series) to refer to a polynomial given at precision $O(\varphi)$.

It is easily checked that if P and Q are two polynomials known at precision $O(\varphi_P)$ and $O(\varphi_Q)$ respectively, then $P+Q$ is known at precision $O(\varphi_P \oplus \varphi_Q)$ and PQ is known at precision $O\big((\varphi_P \otimes \mathrm{NF}(Q)) \oplus (\mathrm{NF}(P) \otimes \varphi_Q)\big)$.

Definition 2.3. Let $P = P_{\mathrm{app}} + O(\varphi_P)$. We say that the Newton precision $O(\varphi_P)$ of P is *nondegenerate* if $\varphi_P \geq \mathrm{NF}(P_{\mathrm{app}})$ and $\varphi_P(x) > y$ for all extremal point (x, y) of $\mathrm{NP}(P_{\mathrm{app}})$.

We notice that, under the conditions of the above definition, the Newton polygon of P is well defined. Indeed, if δP is any polynomial whose Newton function is not less than φ_P, we have $\mathrm{NP}(P_{\mathrm{app}} + \delta P) = \mathrm{NP}(P_{\mathrm{app}})$.

Lattice precision. The notion of *lattice precision* was developed in [4]. It encompasses the two previous models and has the decisive advantage of precision optimality. As a counterpart, it might be very space-consuming and time-consuming for polynomials of large degree.

Definition 2.4. Let V be a finite dimensional vector space over K. A lattice in V is a sub-W-module of V generated by a K-basis of V.

We fix an integer n. A lattice precision datum for a polynomial of degree n is a lattice H lying in the vector space $K_{\leq n}[X]$. We shall sometimes denote it $O(H)$ in order to emphasize that it should be considered as a precision datum. The notation $P_{\mathrm{app}}(X) + O(H)$ then refers to any polynomial in the W-affine space $P_{\mathrm{app}}(X) + H$. Tracking lattice precision can be done using differentials as shown in [4, Lemma 3.4 and Proposition 3.12]: if $f : K_{\leq n}[X] \to K_{\leq m}[X]$ denotes any strictly differentiable function with surjective differential, under mild assumption on H, we have:

$$f(P_{\mathrm{app}}(X) + H) = f(P_{\mathrm{app}}(X)) + f'(P_{\mathrm{app}}(X))(H)$$

where $f'(P_{\mathrm{app}}(X))$ denotes the differential of f at $P_{\mathrm{app}}(X)$. The equality sign reflets the optimality of the method.

As already mentioned, the jagged precision model is a particular case of the lattice precision. Indeed, a precision of the shape $\sum_{i=0}^{n} O(\pi^{N_i}) X^i$ corresponds to the lattice generated by the elements $\pi^{N_i} X^i$ $(0 \leq i \leq n)$. This remark is the origin of the notion of *diffused digits of precision* introduced in [5, Definition 2.3]. We shall use it repeatedly in the sequel in order to compare the behaviour of the three aforementioned precision data in concrete situations.

3. EUCLIDEAN DIVISION

Euclidean division provides a building block for many algorithms associated to polynomials in one variable. In order to analyze the precision behavior of such algorithms, we need to first understand the precision attached to the quotient and remainder when dividing two polynomials. In the sequel, we use the notation $A \,/\!\!/\, B$ and $A \,\%\, B$ for the polynomials satisfying $A = (A \,/\!\!/\, B) \cdot B + (A \,\%\, B)$ and $\deg(A \,\%\, B) < \deg(B)$.

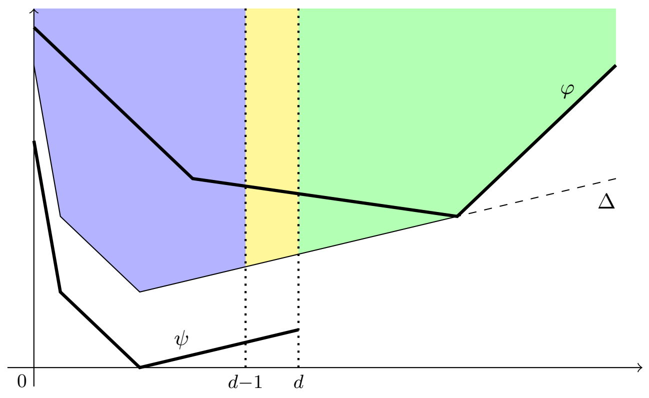

Figure 1: Euclidean division of Newton functions

3.1 Euclidean division of Newton functions

Definition 3.1. Let φ and ψ be two Newton functions of degree n and d respectively. Set $\lambda = \psi(d) - \psi(d-1)$. Letting Δ be the greatest affine function of slope λ with $\Delta \leq \varphi_{|[d,n]}$ and $\delta = \Delta(d) - \psi(d)$, we define:

$$\begin{aligned}
\varphi \,\%\, \psi &= \varphi_{|[0,d-1]} \oplus \big(\psi_{|[0,d-1]} + \delta\big) \\
\varphi \,/\!\!/\, \psi : [0, n-d] &\rightarrow \mathbb{R} \cup \{+\infty\} \\
x &\mapsto \inf_{h \geq 0} \varphi(x + d + h) - \lambda h.
\end{aligned}$$

Figure 1 illustrates the definition: if φ and ψ are the functions represented on the diagram, the epigraph of $\varphi \,\%\, \psi$ is the blue area whereas that of $\varphi \,/\!\!/\, \psi$ is the green area translated by $(-d, 0)$. It is an easy exercise (left to the reader) to design quasi-linear algorithms for computing $\varphi \,\%\, \psi$ and $\varphi \,/\!\!/\, \psi$.

Theorem 3.2. *Given $A, B \in K[X]$ with $B \neq 0$, we have:*

$$\mathit{NF}(A \,\%\, B) \geq \mathit{NF}(A) \,\%\, \mathit{NF}(B) \tag{3}$$
$$\textit{and} \quad \mathit{NF}(A \,/\!\!/\, B) \geq \mathit{NF}(A) \,/\!\!/\, \mathit{NF}(B) \tag{4}$$

PROOF. Write $A = A_{<d} + A_{\geq d}$ where $A_{<d}$ (resp. $A_{\geq d}$) consists of monomials of A of degree less than d (resp. at least d). Noting that:

$$A \,\%\, B = A_{<d} + (A_{\geq d} \,\%\, B) \quad \text{and} \quad A \,/\!\!/\, B = A_{\geq d} \,/\!\!/\, B$$

we may assume that $A = A_{\geq d}$.

Let us now prove Eq. (3). Replacing B by $c^{-1}B$ where c denotes the leading coefficient of B, we may assume that B is monic. Using linearity, we may further assume that A is a monomial. Set $R_n = X^n \,\%\, B$. The relation we have to prove is:

$$\mathrm{NF}(R_n)(x) \geq \mathrm{NF}(B)(x) - \lambda(n - d) \quad \text{for } n \geq d.$$

We proceed by induction. The initialisation is clear because R_d agrees with $(-B)$ up to degree $d-1$. We have the relation $R_{n+1} = XR_n - c_n B$ where c_n is the coefficient in X^{d-1} of R_n. Thanks to the induction hypothesis, we have:

$$\begin{aligned}
\mathrm{val}(c_n) \geq \mathrm{NF}(R_n)(d-1) &\geq \mathrm{NF}(B)(d-1) - \lambda(n-d) \\
&= -\lambda(n+1-d)
\end{aligned}$$

since $\lambda = -\text{NF}(B)(d{-}1)$ because B is monic. Therefore $\text{NF}(c_n B)(x) \geq \text{NF}(B)(x) - \lambda(n{+}1{-}d)$ for all x. On the other hand, for all x, we have:

$$\text{NF}(XR_n)(x) = \text{NF}(R_n)(x{-}1) \geq \text{NF}(R_n)(x) - \lambda$$

from what we get $\text{NF}(XR_n)(x) \geq \text{NF}(B)(x) - \lambda(n{+}1{-}d)$. As a consequence $\text{NF}(R_{n+1})(x) \geq \text{NF}(B)(x) - \lambda(n{+}1{-}d)$ and the induction follows. Eq. (4) is now derived from:

$$\text{NF}(A \mathbin{/\!/} B) \otimes \text{NF}(B) \geq \text{NF}(A) \oplus \text{NF}(A \mathbin{\%} B)$$

using the estimation on $\text{NF}(A \mathbin{\%} B)$ we have just proved (see Figure 1). □

3.2 Tracking precision

Newton precision. We first analyze the precision behavior of Euclidean division in the Newton model. Concretely, we pick $A, B \in K[X]$ two polynomials which are known at precision $O(\varphi_A)$ and $O(\varphi_B)$ respectively:

$$A = A_{\text{app}} + O(\varphi_A) \quad \text{and} \quad B = B_{\text{app}} + O(\varphi_B)$$

Here A_{app} and B_{app} are some approximations of A and B respectively and φ_A and φ_B denotes two Newton functions of degree $\deg A$ and $\deg B$ respectively. We are interested in determining the precision of $A \mathbin{\%} B$ and $A \mathbin{/\!/} B$. The following proposition gives a theoretical answer under mild assumptions.

Proposition 3.3. *We keep the above notations and assume that the Newton precisions $O(\varphi_A)$ and $O(\varphi_B)$ on A and B respectively are both nondegenerate (*cf *Definition 2.3). Then, setting:*

$$\varphi = \varphi_A \oplus \big[\varphi_B \otimes \big(NF(A) \oslash NF(B)\big)\big]$$

the polynomials $A \mathbin{/\!/} B$ and $A \mathbin{\%} B$ are known at precision $O(\varphi \oslash NF(B))$ and $O(\varphi \mathbin{\%\!\!\!\bigcirc} NF(B))$ respectively.

Proof. Let δA (resp. δB) be a polynomial whose Newton function is not less than φ_A (resp. φ_B) and define δQ and δR by:

$$Q_{\text{app}} + \delta Q = (A_{\text{app}} + \delta A) \mathbin{/\!/} (B_{\text{app}} + \delta B)$$
$$R_{\text{app}} + \delta R = (A_{\text{app}} + \delta A) \mathbin{\%} (B_{\text{app}} + \delta B)$$

where $Q_{\text{app}} = A_{\text{app}} \mathbin{/\!/} B_{\text{app}}$ and $R_{\text{app}} = A_{\text{app}} \mathbin{\%} B_{\text{app}}$. We have to show that $\text{NF}(\delta Q) \geq \varphi \oslash \text{NF}(B)$ and $\text{NF}(\delta R) \geq \varphi \mathbin{\%\!\!\!\bigcirc} \text{NF}(B)$. Set $\delta X = \delta A - Q_{\text{app}}\delta B$. Using Theorem 3.2, we obtain $\text{NF}(Q_{\text{app}}) \geq \text{NF}(A) \oslash \text{NF}(B)$ and consequently $\text{NF}(\delta X) \geq \varphi$. On the other hand, an easy computation yields

$$\delta X = (B + \delta B) \cdot \delta Q + \delta R$$

so that $\delta Q = \delta X \mathbin{/\!/} (B + \delta B)$ and $\delta R = \delta X \mathbin{\%} (B + \delta B)$. Using again Theorem 3.2, we get the desired result. □

With this result in hand, we may split the computation of Euclidean division into two pieces, first computing approximations Q_{app} and R_{app} and separately computing δQ and δR. Both the approximations and the precision can be computing in time that is quasi-linear in the degree.

Lattice precision. We now move to lattice precision. We pick A and B two polynomials of respective degree n and d and assume that they are known at precision $O(H_A)$ and $O(H_B)$ respectively:

$$A = A_{\text{app}} + O(H_A) \quad \text{and} \quad B = B_{\text{app}} + O(H_B).$$

where $H_A \in K_{\leq n}[X]$ and $H_B \in K_{\leq d}[X]$ are lattices. According to the results of [4], in order to determine the precision of $A \mathbin{/\!/} B$ and $A \mathbin{\%} B$, we need to compute the differential of the mappings $(X, Y) \mapsto X \mathbin{/\!/} Y$ and $(X, Y) \mapsto X \mathbin{\%} Y$ at the point $(A_{\text{app}}, B_{\text{app}})$. Writing $Q_{\text{app}} = A_{\text{app}} \mathbin{/\!/} B_{\text{app}}$ and $R_{\text{app}} = A_{\text{app}} \mathbin{\%} B_{\text{app}}$, this can be done by expanding the relation:

$$A_{\text{app}} + dA = (B_{\text{app}} + dB)(Q_{\text{app}} + dQ) + (R_{\text{app}} + dR)$$

and neglecting the terms of order ≥ 2. We get this way $dA = B_{\text{app}}dQ + Q_{\text{app}}dB + dR$ meaning that dQ and dR appears respectively as the quotient and the remainder of the Euclidean division of $dX = dA - Q_{\text{app}}dB$ by B_{app}.

Once this has been done, the strategy is quite similar to that explained for Newton precision: compute approximations and precision lattices separately for quotient and remainder.

3.3 An example: modular multiplication

For this example, we work over $W = \mathbb{Z}_2$ and fix a monic polynomial $M \in \mathbb{Z}[X]$ (known exactly) of degree 5. Our aim is to compare the numerical stability of the multiplication in the quotient $\mathbb{Z}_2[X]/M$ depending on the precision model we are using. In order to do so, we pick n random polynomials $P_1, \ldots, P_n$ in $\mathbb{Z}_2[X]/M(X)$ (according to the Haar measure) whose coefficients are all known at precision $O(2^N)$ for some large integer N. We then compute the product of the P_i's using the following quite naive algorithm.

```
1. set $P = 1$
2. for $i = 1, \ldots, n$ do compute $P = (P \cdot P_i) \mathbin{\%} M$
3. return $P$
```

The table of Figure 2 reports the average gain of *absolute* precision G which is observed while executing the algorithm above for various modulus and n. The average is taken on a sample of 1000 random inputs. We recall that G is defined as follows:

- in the case of jagged and Newton precision, the precision of the output may be written into the form $\sum_{i=0}^{4} O(2^{N_i})\, X^i$ and $G = \sum_{i=0}^{4} (N_i - N)$;
- in the case of lattice precision, the precision of the output is a lattice H and G is the index of H in $2^N \mathcal{L}$ where $\mathcal{L} = \mathbb{Z}_2[X]/M$ is the standard lattice; in that case, we write G as a sum $G_{\text{nd}} + G_{\text{d}}$ where G_{d} is the index of H in the largest lattice H_0 contained in H which can be generated by elements of the shape $2^{N_i} X^i$ $(0 \leq i \leq 4)$. (The term G_d corresponds to diffused digits according to [5, Definition 2.3].)

We observe several interesting properties. First of all, the gains for Newton precision and jagged precision always agree though one may have thought at first that Newton precision is weaker. Since performing precision computations in the Newton framework is cheaper, it seems (at least on this example) that using the jagged precision model is not relevant.

On the other hand, the lattice precision may end up with better results. Nevertheless this strongly depends on the modulus M. For instance, when M is irreducible modulo $p = 2$ or Eiseistein, there is apparently no benefit to using the lattice precision model. We emphasize that these two particular cases correspond to modulus that are usually used to define (unramified and totally ramified respectively) extensions of $\mathbb{Q}_2$.

For other moduli, the situation is quite different and the

Modulus M	n	Gain of precision		
		Jagged	Newton	Lattice (not dif. + dif.)
X^5+X^2+1 (Irred. mod 2)	10	0.2	0.2	0.2 + 0.0
	50	4.2	4.2	4.2 + 0.0
	100	11.2	11.2	11.2 + 0.0
X^5+1 (Sep. mod 2)	10	0.4	0.4	0.9 + 6.0
	50	5.6	5.6	11.1 + 42.0
	100	13.6	13.6	27.0 + 87.0
X^5+2 (Eisenstein)	10	6.2	6.2	6.2 + 0.0
	50	44.0	44.0	44.0 + 0.0
	100	92.5	92.5	92.5 + 0.0
$(X+1)^5+2$ (Shift Eisenstein)	10	0.6	0.6	4.7 + 1.4
	50	7.1	7.1	42.6 + 1.4
	100	15.1	15.1	91.8 + 1.4
X^5+X+2 (Two slopes)	10	1.7	1.7	7.9 + 9.8
	50	8.1	8.1	70.7 + 59.8
	100	16.1	16.1	152.6 + 125.9

Figure 2: Precision for modular multiplication

benefit of using the lattice precision model becomes more apparent. The comparison between the gain of precision in the jagged model and the number of not diffused digits in the lattice model makes sense: indeed the latter appears as a theoretical upper bound of the former and the difference between them quantifies the quality of the way we track precision in the jagged (or the Newton) precision model. We observe that this difference is usually not negligible (*cf* notably the case of $M(X) = (X+1)^5+2$) meaning that this quality is not very good in general. As for diffused digits, they correspond to digits that cannot be "seen" in the jagged precision model. Their number then measures the intrinsic limitations of this model. We observe that it can be very important as well in several cases.

The modulus $(X+1)^5+2$ shows the advantage of working with lattice precision in intermediate computations. Indeed, the precision behavior using the lattice model closely parallels that of X^5+2, since the lattices are related by a change of variables. But this structure is not detected in the Newton or jagged models.

4. SLOPE FACTORIZATION

A well-known theorem [6, Theorem 6.1] asserts that each extremal point M in the Newton polygon $\mathrm{NP}(P)$ of a polynomial $P \in K[X]$ corresponds to a factorization $P = AB$ where the Newton polygon of A (resp. B) is given by the part of $\mathrm{NP}(P)$ located at the left (resp. the right) of M. Such a factorization is often called a *slope factorization*.

The aim of this section is to design efficient and stable algorithms for computing these factorizations. Precisely the algorithm we obtain has a quasi-optimal complexity (compared to the size of the input polynomial) and outputs a result whose precision is (close to be) optimal. Two of its important additional features are simplicity and flexibility.

4.1 A Newton iteration

The factor A defined above is usually obtained *via* a Newton iteration after having prepared our polynomial by flattening the first slope using a change of variables involving possibly rational exponents. We introduce here a variant of this iteration which does not require the flattening step and is entirely defined over $K[X]$.

Theorem 4.1. *Let $P(X) = \sum_{i=0}^n a_i X^i$ be a polynomial of degree n with coefficients in K. We assume that $\mathrm{NP}(P)$ has an extremal point whose abscissa is d. We define the sequences $(A_i)_{i\geq 0}$ and $(V_i)_{i\geq 0}$ recursively by:*

$$A_0 = \sum_{i=0}^d a_i X^i, \quad V_0 = 1$$
$$A_{i+1} = A_i + (V_i P \ \% \ A_i),$$
$$V_{i+1} = (2V_i - V_i^2 B_{i+1}) \,\%\, A_{i+1}$$
$$\text{where } B_{i+1} = P \,/\!/\, A_{i+1}.$$

Then the sequence (A_i) converges to a divisor A_∞ of P of degree d whose leading coefficient is a_d and whose Newton function agrees with $NF(P)$ on $[0,d]$. Moreover, setting:

$$\kappa = NF(P)(d{+}1) + NF(P)(d{-}1) - 2 \cdot NF(P)(d)$$

(with $NF(P)(-1) = NF(P)(n{+}1) = +\infty$ if necessary), we have $\kappa > 0$ and the following rate of convergence:

$$\forall i \geq 0, \quad NF(A_\infty - A_i) \geq NF(P)_{|[0,d-1]} + 2^i\kappa. \tag{5}$$

Remark 4.2. The iteration formulae of the above theorem already appeared in [2, § 3.1]; nevertheless, as far as we know, the result of Theorem 4.1 is new.

Remark 4.3. The divisor A is uniquely determined by the conditions of Theorem 4.1. Indeed, consider two divisors A and A' of P such that $\mathrm{NF}(A) = \mathrm{NF}(A') = \mathrm{NF}(P)_{|[0,d]}$. Then $L = \mathrm{lcm}(A, A')$ is a divisor of P as well and the slopes of its Newton polygon are all at most $\lambda_0 = \mathrm{NF}(P)(d) - \mathrm{NF}(P)(d{-}1)$. Therefore $\deg L = d$ and L differs from A and A' by a multiplicative nonzero constant. Then, if A and A' share in addition the same leading coefficient, they must coincide.

The rest of this subsection is devoted to the proof of the theorem. If $d = n$ (resp. $d = 0$), the sequence A_i is constant equal to P (resp. to the constant coefficient of P) and theorem is clear. We then assume $0 < d < n$. We set:

$$\lambda_0 = \mathrm{NF}(P)(d) - \mathrm{NF}(P)(d{-}1)$$
$$\lambda_1 = \mathrm{NF}(P)(d{+}1) - \mathrm{NF}(P)(d),$$

so that $\kappa = \lambda_1 - \lambda_0$. The existence of an extremal point of $\mathrm{NP}(P)$ located at abscissa d ensures that $\lambda_1 > \lambda_0$, *i.e.* $\kappa > 0$. For all indices i, we define:

$$Q_i = V_i P \,/\!/\, A_i, \quad R_i = V_i P \,\%\, A_i = A_{i+1} - A_i,$$
$$S_i = P \,\%\, A_i, \quad T_i = (1 - V_i B_i) \,\%\, A_i$$

and when $\square$ is some letter, we put $\Delta\square_i = \square_{i+1} - \square_i$.

Lemma 4.4. *The following relations hold:*

$$\Delta B_i = -(R_i B_{i+1}) \,/\!/\, A_i, \tag{6}$$
$$\Delta S_i = -(R_i B_{i+1}) \,\%\, A_i, \tag{7}$$
$$S_i = (B_i R_i + T_i S_{i-1} + T_i \,\Delta S_{i-1}) \,\%\, A_i, \tag{8}$$
$$\Delta V_i = (V_i T_i - V_i^2 \,\Delta B_i) \,\%\, A_i, \tag{9}$$
$$1 - Q_i = T_i - (V_i S_i) \,/\!/\, A_i, \tag{10}$$
$$R_{i+1} = (\Delta V_i \, S_{i+1} + (1{-}Q_i) R_i) \,\%\, A_{i+1}, \tag{11}$$
$$T_{i+1} = (T_i + V_i \,\Delta B_i)^2 \,\%\, A_{i+1}. \tag{12}$$

PROOF. From $P = A_iB_i + S_i = A_{i+1}B_{i+1} + S_{i+1}$, we get $-R_iB_{i+1} = \Delta B_i \cdot A_i + \Delta S_i$. Hence, by consideration of degree, we obtain (6) and (7). On the other hand, from $V_iP = A_iQ_i + R_i = V_i(A_iB_i + S_i)$, we derive

$$(V_iB_i - Q_i) \cdot A_i = R_i - V_iS_i. \qquad (13)$$

Thus $R_i = V_iS_i \,\%\, A_i$. Hence $B_iR_i = (S_i - S_iT_i) \,\%\, A_i$ and $S_i = B_iR_i + S_iT_i \,\%\, A_i$, from which (8) follows directly By definition of V_i, we get $\Delta V_i = V_i(1 - V_iB_{i+1}) \,\%\, A_{i+1}$ and consequently (9). We now write $1 - Q_i = T_i + (V_iB_i - Q_i)$. Using (13) and noting that $\deg R_i < \deg A_i = d$, we get (10).

We have $V_iP = A_iQ_i + R_i = (A_{i+1} - R_i)Q_i + R_i$ and $V_{i+1}P = A_{i+1}Q_{i+1} + R_{i+1}$. Thus:

$$\begin{aligned} R_{i+1} &= \Delta V_i \, P + (1 - Q_i)R_i \\ &= (\Delta V_i \, S_{i+1} + (1 - Q_i)R_i) \,\%\, A_{i+1}, \end{aligned}$$

and (11) is proved. Finally

$$\begin{aligned} T_{i+1} &\equiv 1 - 2V_iB_{i+1} + V_i^2B_{i+1}^2 \pmod{A_{i+1}} \\ &\equiv (1 - V_iB_{i+1})^2 \pmod{A_{i+1}} \\ &\equiv (T_i + V_i \, \Delta B_i)^2 \pmod{A_{i+1}} \end{aligned}$$

which concludes the proof. □

If $\lambda_0 = -\infty$ or $\lambda_1 = +\infty$, the sequence (A_i) is constant and the theorem follows, so assume that λ_0 and λ_1 are finite.

We define the function $\varphi : \mathbb{R}^+ \to \mathbb{R} \cup \{+\infty\}$ by:

$$\varphi(x) = \begin{cases} \mathrm{NF}(P)(x) & \text{if } x \le d \\ \lambda_0(x - d) + \mathrm{NF}(P)(d) & \text{if } x > d \end{cases} \qquad (14)$$

For nonzero c in a finite extension of K, note that the theorem holds for P if and only if it holds for cP, since the A_i's change to cA_i's while the B_i's and the V_i's remained unchanged. We may thus assume that P is normalized so that $\mathrm{NF}(P)(d) = d\lambda_0$, *i.e.* $\varphi(x) = \lambda_0 x$ for $x > d$. For a polynomial $Q \in K[X]$ of degree n, we further define:

$$b_\varphi(Q) = \min_{x\in[0,n]} \mathrm{NF}(Q)(x) - \varphi(x) \qquad (15)$$

$$\text{and} \quad b_i(Q) = \min_{x\in[0,n]} \mathrm{NF}(Q)(x) - \lambda_i x \quad \text{for } i \in \{0, 1\}. \qquad (16)$$

Set also $b_\varphi(0) = b_0(0) = b_1(0) = +\infty$ by convention. With the normalization of P we chose above, we have $b_0(P) = b_\varphi(P) = 0$ and $b_\varphi(Q) \le b_0(Q)$ for all polynomial Q. Similarly $b_1(Q) \le b_0(Q)$ for all Q.

Lemma 4.5. *Let $b \in \{b_\varphi, b_0, b_1\}$. For $Q_1, Q_2 \in K[X]$:*

a) $b(Q_1 + Q_2) \ge \min\big(b(Q_1), b(Q_2)\big)$

b) $b(Q_1Q_2) \ge \min b(Q_1) + b(Q_2)$

c) $b_\varphi(Q_1Q_2) \ge b_\varphi(Q_1) + b_0(Q_2)$

For $Q, A \in K[X]$ with $\deg A = d$ and $NF(A) = NF(P)_{|[0,d]}$:

d) $b(Q \,\%\, A) \ge b(Q)$

e) $b_0(Q \,/\!\!/\, A) \ge b_\varphi(Q)$.

PROOF. *a)* is clear.

We skip the proof of *b)* which is similar to that of *c)*.

Let t_1 (resp. t_2) be the translation of vector $(0, b_\varphi(Q_1))$ (resp. $(0, b_0(Q_2))$. It follows from the definition of b_φ that $\mathrm{NP}(Q_1)$ is a subset of $t_1(\mathrm{Epi}(\varphi))$ where $\mathrm{Epi}(\varphi)$ denotes the epigraph of φ. Similarly $\mathrm{NP}(Q_2) \subset t_2(C)$ where C is the

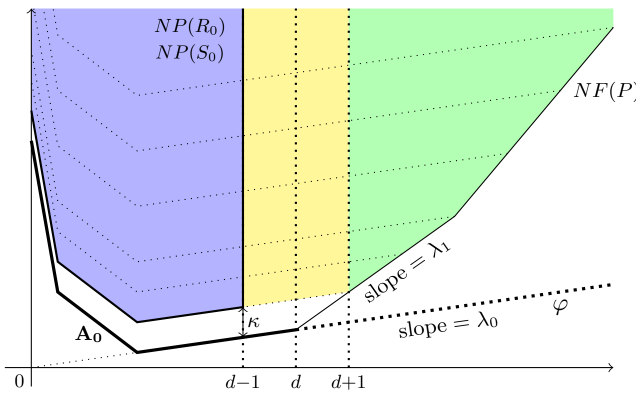

Figure 3: Bound on NF(R_0) and NF(S_0)

convex cone generated by the vectors starting from $(0, 0)$ to $(0, 1)$ and $(1, \lambda_0)$. Thus

$$\mathrm{NP}(Q_1Q_2) \subset t_2 \circ t_1\big(\mathrm{Epi}(\varphi) + C\big) = t_2 \circ t_1\big(\mathrm{Epi}(\varphi)\big)$$

and *c)* follows.

Finally *d)* and *e)* follows from Theorem 3.2. □

We are now going to prove by induction on i the conjunction of all equalities and inequalities below:

$$\begin{aligned} &\mathrm{NF}(A_i) = \varphi_{|[0,d]}, \\ &b_1(V_i) \ge 0, \quad b_\varphi(R_i) \ge 2^i\kappa \\ &b_\varphi(S_i) \ge 0, \quad b_0(T_i) \ge 2^i\kappa. \end{aligned} \qquad (17)$$

Noting that A_0 and P agree up to degree d and that $\mathrm{NP}(P)$ has an extremal point at abscissa d, we get $\mathrm{NF}(A_0) = \varphi_{|[0,d]}$. Clearly $b_1(V_0) \ge 0$ since $V_0 = 1$. It follows from the definitions that $R_0 = S_0 = P \,\%\, A_0 = (P - A_0) \,\%\, A_0$. We remark that $P - A_0 = \sum_{i=d+1}^n a_iX^i$. Using Theorem 3.2, we obtain that $b_\varphi(P - A_0) \ge \kappa$ and then $b_\varphi(R_0) = b_\varphi(S_0) \ge \kappa$ (see Figure 3). Finally observe that:

$$T_0 = (1 - B_0) \,\%\, A_0 = \big((A_0 - P) \,/\!\!/\, A_0\big) \,\%\, A_0.$$

Therefore $b_0(T_0) \ge \kappa$ results from $b_\varphi(A_0 - P) \ge \kappa$ thanks to Lemma 4.5. We have then established (17) when $i = 0$.

We now assume (17) for the index i. From $A_{i+1} = A_i + R_i$ and the estimation $b_\varphi(R_i) \ge 2^i\kappa > 0$, we derive $\mathrm{NF}(A_{i+1}) = \varphi_{|[0,d]}$. Therefore, Lemma 4.5 applies with $A = A_i$ and $A = A_{i+1}$. Now coming back to the the definition of B_{i+1} and using Theorem 3.2, we get $b_0(B_{i+1}) \ge b_1(B_{i+1}) \ge 0$. As a consequence:

$$b_\varphi(R_iB_{i+1}) \ge b_\varphi(R_i) + b_0(B_{i+1}) \ge 2^i\kappa$$

by Lemma 4.5 and the induction hypothesis. Using again Lemma 4.5, we then derive from (6) and (7) that $b_0(\Delta B_i) \ge 2^i\kappa$ and $b_\varphi(\Delta S_i) \ge 2^i\kappa$. Similarly, using (8) and the estimations we already know, we obtain $b_\varphi(S_i) \ge 2^i\kappa$. Combining this with $b_\varphi(\Delta S_i) \ge 2^i\kappa$, we find $b_\varphi(S_{i+1}) \ge 2^i\kappa$ as well. Applying again and again the same strategy, we deduce successively $b_0(\Delta V_i) \ge 2^i\kappa$ using (9), $b_0(1 - Q_i) \ge 2^i\kappa$ using (10), $b_\varphi(R_{i+1}) \ge 2^{i+1}\kappa$ using (11), and then $b_0(T_{i+1}) \ge 2^{i+1}\kappa$ using (12). Finally, coming back to the recurrence defining V_{i+1} and remembering that $b_1(V_i)$ and $b_1(B_{i+1})$ are both nonnegative, we find $b_1(V_{i+1}) \ge 0$. The equalities and inequalities of Eq. (17) have then all been established for the index $i + 1$ and the induction goes.

From the inequalities $b_\varphi(R_i) \geq 2^i\kappa$, we deduce that the sequence (A_i) is Cauchy and therefore converges. Its limit A_∞ certainly satisfies $\mathrm{NF}(A_\infty) = \varphi_{|[0,d]}$ because all the A_i's do. Moreover we know that $b_\varphi(S_i) \geq 2^i\kappa$ from what we derive that the sequence (S_i) goes to 0. Coming back to the definition of S_i, we find $P \,\%\, A_\infty = 0$, *i.e.* A_∞ divides P. Finally, Eq. (5) giving the rate of convergence follows from the writing $A_\infty - A_i = \sum_{j=i}^{\infty} R_j$ together with the facts that $b_\varphi(R_j) \geq 2^i\kappa$ and $\deg R_j \leq d-1$ for all $j \geq i$.

Remark 4.6. From the proof above, the sequence $(V_i)_{i\geq 0}$ converges as well. Its limit V_∞ is an inverse of $B_\infty = P \,/\!\!/\, A_\infty$ modulo A_∞ and it satisfies in addition $b_1(V_\infty) \geq 0$.

Moreover, the conclusion of Theorem 4.1 is still correct if A_0 is any polynomial of degree d with leading coefficient a_d and V_0 is any polynomial as soon as they satisfy:

$$b_\varphi\big(V_0 P \,\%\, A_0\big) > 0 \quad \text{and} \quad b_0\big((1 - V_0 B_0) \,\%\, A_0\big) > 0$$

except that the constant κ giving the rate of convergence should be now $\kappa = \min\big(b_\varphi\big(V_0 P \,\%\, A_0\big),\, b_0\big((1-V_0B_0) \,\%\, A_0\big)\big)$.

4.2 A slope factorization algorithm

Let $P \in K_n[X]$ and d be the abscissa of an extremal point of $\mathrm{NP}(P)$. Previously (*cf* Theorem 4.1), we have defined a sequence (A_i, V_i) converging to (A, V) where A is a factor of P whose Newton function is $\mathrm{NF}(P)_{|[0,d]}$ and V is the inverse of $B = P/A$ modulo A. We now assume that P is known up to some finite precision: $P = P_{\mathrm{app}} + O(\cdots)$ where the object inside the O depends on the chosen precision model. We address the two following questions: (1) what is the precision of the factor A, and (2) how can one compute in practice A at this precision?

In the sequel, it will be convenient to use a different normalization on A and B: if a_d is the coefficient of P of degree d, we set $A^{(1)} = a_d^{-1} A$ and $B^{(1)} = a_d B$ so that $A^{(1)}$ is monic and $P = A^{(1)}B^{(1)}$. We also assume the leading coefficient of P is *exactly* 1; the precision datum on P then only concerns the coefficients up to degree $n-1$. Similarly, $A^{(1)}$ and $B^{(1)}$ only carry precision up to degree $d-1$ and $n-d-1$ respectively.

Newton precision. We assume that the precision of the input P has the shape $O(\varphi_P)$ where φ_P is a Newton function of degree $n-1$. From now on, we assume that the precision $O(\varphi_P)$ is nondegenerate in the sense of Definition 2.3. This ensures in particular that the Newton polygon of P is well defined. We import the notations φ, b_φ and b_0 from §4.1 and refer to Eqs. (14)–(16) for the definitions.

Proposition 4.7. *We keep all the above notations and assumptions. We set:*

$$\delta = \min_{x\in[0,n-1]} \varphi_P(x) - \varphi(x)$$

and assume that $\delta > 0$. Then the factor $A^{(1)}$ is known with precision at least $O(\varphi_{A^{(1)}})$ with $\varphi_{A^{(1)}} = \varphi_{|[0,d-1]} - \varphi(d) + \delta$.

PROOF. Let $\delta_P \in K[X]$ be such that $\mathrm{NP}(\delta_P) \geq \varphi_P$. Let $A^{(1)}_{\mathrm{app}}$ and $A^{(1)}$ be the monic factors of P_{app} and $P_{\mathrm{app}} + \delta P$ respectively whose Newton functions are $\varphi_{|[0,d]} - \varphi(d)$.

We define the sequences (A_i) and (V_i) by the recurrence:

$$\begin{aligned} A_0 &= A_{\mathrm{app}}, \quad V_0 = V_{\mathrm{app}} \\ A_{i+1} &= A_i + \big(V_i(P_{\mathrm{app}} + \delta_P) \,\%\, A_i\big), \\ V_{i+1} &= (2V_i - V_i^2 B_{i+1}) \,\%\, A_{i+1} \\ &\qquad \text{where } B_{i+1} = (P_{\mathrm{app}} + \delta_P) \,/\!\!/\, A_{i+1}. \end{aligned}$$

where A_{app} and V_{app} are those related to P_{app}. Note that $A_{\mathrm{app}} = a_d \cdot A^{(1)}_{\mathrm{app}}$ if a_d denotes the coefficient of X^d in P_{app}. By Remark 4.6, we know that the sequence (A_i) converges to $A = a_d \cdot A^{(1)}$ and furthermore:

$$\mathrm{NP}(A - A_{\mathrm{app}}) \geq \varphi_{|[0,d-1]} + b_\varphi\big((V_{\mathrm{app}} \cdot \delta P) \,\%\, A_{\mathrm{app}}\big)$$

since $(V_{\mathrm{app}}P_{\mathrm{app}}) \,\%\, A_{\mathrm{app}} = (1 - V_{\mathrm{app}}B_{\mathrm{app}}) \,\%\, A_{\mathrm{app}} = 0$. Using repeatedly Lemma 4.5, we obtain:

$$b_\varphi\big((V_{\mathrm{app}} \cdot \delta P) \,\%\, A_{\mathrm{app}}\big) \geq b_0(V_{\mathrm{app}}) + b_\varphi(\delta P) \geq b_\varphi(\delta P) \geq \delta.$$

Thus $\mathrm{NP}(A - A_{\mathrm{app}}) \geq \varphi_{[0,d-1]} + \delta$. Dividing by a_d, we find $\mathrm{NP}(A^{(1)} - A^{(1)}_{\mathrm{app}}) \geq \varphi_{A^{(1)}}$ and we are done. $\square$

Remark 4.8. Under the hypothesis **(H)** introduced below, a correct precision on $A^{(1)}$ is also $O(\psi_{A^{(1)}})$ where:

$$\psi_{A^{(1)}} = \big(\varphi_P \otimes \big(\mathrm{NF}(V_{\mathrm{app}}) - \mathrm{NF}(P)(d)\big)\big) \oslash \mathrm{NF}(P)_{|[0,d]}.$$

This follows from Proposition 4.10 using $V^{(1)}_{\mathrm{app}} = a_d^{-1} V_{\mathrm{app}}$. It follows in addition from Remark 4.6 that $\mathrm{NF}(V_{\mathrm{app}})$ is bounded from below by $x \mapsto \lambda_1 x$. This yields the bound $\psi_{A^{(1)}} \geq \big(\varphi_P \otimes \psi\big) \oslash \mathrm{NF}(P)_{|[0,d]}$ where $\psi : [0, d-1] \to \mathbb{R}$ is the affine function mapping x to $\lambda_1 x - \mathrm{NF}(P)(d)$.

We can now move to the second question we have raised before, *i.e.* the design of an algorithm for computing $A^{(1)}$ with the precision given by Proposition 4.7. Our strategy consists in computing first the precision and then applying the Newton iteration until the expected precision is reached.

```
Algorithm slope_factorisation_Newton
Input: a monic polynomial $P + O(\varphi_P) \in K_n[X]$,
       a break point $d$ of $\mathrm{NP}(P)$
Output: the factor $A$ described above
1. Compute the functions $\mathrm{NF}(P)$ and $\varphi$
2. Compute $\varphi_A = \varphi_{|[0,d-1]} - \varphi(d) + \min_{x\in[0,n]} \varphi_P(x) - \varphi(x)$
3. Compute $\kappa = \mathrm{NP}(P)(d+1) + \mathrm{NF}(P)(d-1) - 2 \cdot \mathrm{NF}(P)(d)$
4. Set $i = 0$, $A_0 = \sum_{i=0}^{d} a_i X^i$ ($a_i$ = coeffs of $P$), $V_0 = 1$
5. repeat until $\varphi_{|[0,d-1]} + 2^i\kappa \geq \varphi_A$
6.     lift $A_i$, $V_i$ and $P$ at enough precision
7.     compute
          • $A_{i+1} = A_i + (PV_i \,\%\, A_i)$
            at precision $O(\varphi_{|[0,d-1]} + 2^{i+1}\kappa)$
          • $V_{i+1} = (2V_i - V_i^2 \cdot (P \,/\!\!/\, A_{i+1})) \,\%\, A_{i+1}$
            at precision $O(x \mapsto \lambda_1 x + 2^{i+1}\kappa)$
8.     set $i = i + 1$
9. return $A_i + O(\varphi_A)$
```

Remark 4.9. The precision needed at line 6 is of course governed by the computation performed at line 7. Note that it can be either computed *a priori* by using Proposition 3.3 or dynamically by using relaxed algorithms from [1, 11, 12]. In both cases, it is in $O(\pi^{N_i})$ with $N_i = O(2^i\kappa + \min \mathrm{NF}(P))$.

It follows from Theorem 4.1, Remark 4.6 and Proposition 4.7 that Algorithm `slope_factorisation_Newton` is correct and stable. Using the standard soft-O notation $\tilde{O}(\cdot)$ for hiding logarithmic factor, our algorithm performs at most $\tilde{O}(n)$ combinatorial operations and $\tilde{O}(n)$ operations in K at precision $O(\pi^N)$ with $N = O(\max \varphi_P - \min \mathrm{NF}(P))$ if one uses quasi-optimal algorithms for multiplication and Euclidean division of polynomials.

Lattice precision. The precision datum is given here by a lattice H_P in $K_{\leq n-1}[X]$; we shall then write $P = P_{\mathrm{app}} + O(H_P)$ where P_{app} is a *monic* approximation of the inexact polynomial P we want to factor. We assume from now that H_P is sufficiently small so that the Newton polygon of P is well defined. We then can define the function:

$$F = (F_A, F_B) : P_{\mathrm{app}} + H_P \to K_{=d}[X] \times K_{=n-d}[X]$$

mapping a polynomial P to the pair $(A^{(1)}, B^{(1)})$ obtained from it. We set $(A^{(1)}_{\mathrm{app}}, B^{(1)}_{\mathrm{app}}) = F(P_{\mathrm{app}})$.

We make the following hypothesis **(H)**:

The lattice H_P is a first order lattice at every point of $P_{\mathrm{app}} + H_P$ in the sense of [4, Definition 3.3], *i.e.* for all $P \in P_{\mathrm{app}} + H_P$:

$$F(P + H_P) = F(P) + F'(P)(H_P).$$

Obviously **(H)** gives an answer to the first question we have raised above: the precision of $A^{(1)}$ is the lattice $H_{A^{(1)}}$ defined as projection on the first component of $F'(P_{\mathrm{app}})(H_P)$. It turns out that it can be computed explicitely as shown by the next proposition.

Proposition 4.10. *The map $F_A : P \mapsto A^{(1)}$ is of class C^1 on $P_{app} + H_P$ and its differential at some point P is the linear mapping*

$$dP \mapsto dA^{(1)} = (V^{(1)}\, dP) \,\%\, A^{(1)}$$

where $(A^{(1)}, B^{(1)}) = F(P)$ and $V^{(1)}$ is the inverse of $B^{(1)}$ modulo $A^{(1)}$.

Proof. The function F is injective and a left inverse of it is $G : (A^{(1)}, B^{(1)}) \mapsto A^{(1)}B^{(1)}$. Clearly G is of class C^1 and its differential is given by

$$(dA^{(1)}, dB^{(1)}) \mapsto dP = A^{(1)}\, dB^{(1)} + B^{(1)}\, dA^{(1)}. \qquad (18)$$

Thanks to Bézout Theorem, it is invertible as soon as $A^{(1)}$ and $B^{(1)}$ are coprime, which is true because $\mathrm{NP}(A^{(1)})$ and $\mathrm{NP}(B^{(1)})$ do not shape a common slope. As a consequence F is of class C^1 and its differential is obtained by inverting Eq. (18). Reducing modulo $A^{(1)}$, we get $dP \equiv B^{(1)}\, dA^{(1)} \pmod{A^{(1)}}$. The claimed result follows after having noticed that $dA^{(1)}$ has degree at most $d-1$. ☐

A remarkable Corollary of Proposition 4.10 asserts the optimality of Proposition 4.7 in a particular case.

Corollary 4.11. *We assume **(H)**. Let $\delta \in \mathbb{R}$. When the precision of P is given by $O(NF(P)_{|[0,d-1]} + \delta)$, the precision of $A^{(1)}$ given by Proposition 4.7 is optimal.*

Proof. First remark that the δ defined in the statement of Proposition 4.7 coincide with the δ introduced in the Corollary. Define $\varphi_{A^{(1)}} = \varphi_{|[0,d-1]} - \varphi(d) + \delta$. Let H_P (resp. $H_{A^{(1)}}$) be the lattice consisting of polynomials of degree at most $n-1$ (resp. at most $d-1$) whose Newton function is not less that $\varphi_P = \mathrm{NF}(P)_{|[0,d-1]} + \delta$ (resp. $\varphi_{A^{(1)}}$). We have to show that $F'_A(P_{\mathrm{app}})(H_P) = H_{A^{(1)}}$. According to Proposition 4.10 the mapping $G : K_{\leq d-1}[X] \to K_{\leq n-1}[X]$, $dA^{(1)} \mapsto (B^{(1)}_{\mathrm{app}}\, dA^{(1)}) \,\%\, A^{(1)}_{\mathrm{app}}$ is a right inverse of $F'_A(P_{\mathrm{app}})$. It is then enough to prove that G takes $H_{A^{(1)}}$ to H_P, which can be done easily using Theorem 3.2. ☐

To compute $A^{(1)}$ in practice, we give a stopping criterion.

Proposition 4.12. *We assume **(H)**.*

(i) Let $\tilde{A}^{(1)} \in K_{=d}[X]$ such that $\tilde{A}^{(1)}\tilde{B}^{(1)} \in P_{app} + O(H_P)$ with $\tilde{B}^{(1)} = P_{app} \,/\!/\, \tilde{A}^{(1)}$. Then $\tilde{A}^{(1)} \in A^{(1)}_{app} + H_{A^{(1)}}$.

(ii) Let in addition $\tilde{V}^{(1)} \in K_{\leq d-1}[X]$ such that:

$$\big(\tilde{B}^{(1)}\tilde{V}^{(1)} \cdot H_P\big) \,\%\, \tilde{A}^{(1)} = H_P \,\%\, \tilde{A}^{(1)}.$$

Then $H_{A^{(1)}} = \big(\tilde{V}^{(1)} \cdot H_P\big) \,\%\, \tilde{A}^{(1)}$.

Proof. *(i)* Set $\tilde{P} = \tilde{A}^{(1)}\tilde{B}^{(1)}$. We know by assumption that $\tilde{P} = P_{\mathrm{app}} + O(H_P)$. Thus $F(\tilde{P})$ is well defined. The unicity of the slope factorization (*cf* Remark 4.3) further implies that $F(\tilde{P}) = (\tilde{A}^{(1)}, \tilde{B}^{(1)})$. The claimed result now follows from the hypothesis **(H)**.

(ii) By applying **(H)** with $P = \tilde{P}$ and replacing $F'(\tilde{P})$ by its expression given by Proposition 4.10, we find:

$$H_{A^{(1)}} = \big((\tilde{B}^{(1)})^{-1} \cdot H_P\big) \,\%\, \tilde{A}^{(1)}.$$

Dividing $\big(\tilde{B}^{(1)}\tilde{V}^{(1)} H_P\big) \,\%\, \tilde{A}^{(1)} = H_P \,\%\, \tilde{A}^{(1)}$ by $\tilde{B}^{(1)}$ modulo $\tilde{A}^{(1)}$, we get $H_{A^{(1)}} = \big(\tilde{V}^{(1)} \cdot H_P\big) \,\%\, \tilde{A}^{(1)}$ as expected. ☐

As a conclusion, the algorithm we propose consists in computing the Newton sequences (A_i) and (V_i) (following the strategy of Algorithm `slope_factorisation_Newton` regarding to precision) until we find a pair $(\tilde{A}^{(1)}, \tilde{V}^{(1)})$ satisfying the requirements (i) and (ii) of Proposition 4.12. Once this pair has been found, one may safely output $\tilde{A}^{(1)} + O\big((\tilde{V}^{(1)} \cdot H_P) \,\%\, \tilde{A}^{(1)}\big)$ under **(H)**. The resulting algorithm has quasi-optimal running time and optimal stability.

References

[1] Jérémy Berthomieu, Joris van der Hoeven, and Grégoire Lecerf, *Relaxed algorithms for p-adic numbers*, J. Théorie des Nombres des Bordeaux **23** (2011), no. 3, 541–577.

[2] Jérémy Berthomieu, Grégoire Lecerf, and Guillaume Quintin, *Polynomial root finding over local rings and application to error correcting codes*, Applicable Algebra in Engineering, Communication and Computing **24** (2013), no. 6, 413–443, DOI 10.1007/s00200-013-0200-5.

[3] Xavier Caruso, *Slope factorization of ore polynomials*, 2016. in preparation.

[4] Xavier Caruso, Tristan Vaccon, and David Roe, *Tracking p-adic precision*, LMS Journal of Computation and Mathematics **17 (Special issue A)** (2014), 274–294.

[5] ———, *p-adic stability in linear algebra*, Proceedings of the 2015 acm on international symposium on symbolic and algebraic computation, 2015, pp. 101–108.

[6] Bernard Dwork, Giovanni Gerotto, and Francis Sullivan, *An introduction to G-functions*, Princeton U.P., Princeton, 1994.

[7] Jordi Guàrdia, Jesús Montes, and Enric Nart, *Newton polygons of higher order in algebraic number theory*, Transactions of the AMS **364** (2012), no. 1, 361–416.

[8] Jordi Guàrdia, Enric Nart, and Sebastian Pauli, *Single-factor lifting and factorization of polynomials over local fields*, Journal of Symbolic Computation **47** (2012), no. 11, 1318 –1346.

[9] Jesús Montes, *Polígonos de newton de orden superior y aplicaciones aritméticas*, Ph.D. Thesis, 1999.

[10] Sebastian Pauli, *Factoring polynomials over local fields II*, Algorithmic number theory, 9th international symposium, 2010.

[11] Joris van der Hoeven, *Relax, but don't be too lazy*, J. Symbolic Comput. **34** (2002), no. 6, 479–542.

[12] ———, *New algorithms for relaxed multiplication*, J. Symbolic Comput. **42** (2007), no. 8, 792–802.

[13] Adrien Poteaux and Marc Rybowicz, *Improving Complexity Bounds for the Computation of Puiseux Series over Finite Fields*, ISSAC'15, 2015, pp. 299–306.