JOURNAL
de Théorie des Nombres
de BORDEAUX
anciennement Séminaire de Théorie des Nombres de Bordeaux

Xavier CARUSO

Numerical stability of Euclidean algorithm over ultrametric fields

Tome 29, n° 2 (2017), p. 503-534.

<http://jtnb.cedram.org/item?id=JTNB_2017__29_2_503_0>

Journal de Théorie des Nombres
de Bordeaux **29** (2017), 503–534

Numerical stability of Euclidean algorithm over ultrametric fields

par Xavier CARUSO

Résumé. Nous étudions le problème de la stabilité du calcul des résultants et sous-résultants des polynômes définis sur des anneaux de valuation discrète complets (e.g. $\mathbb{Z}_p$ ou $k[\![t]\!]$ où k est un corps). Nous démontrons que les algorithmes de type Euclide sont très instables en moyenne et, dans de nombreux cas, nous expliquons comment les rendre stables sans dégrader la complexité. Chemin faisant, nous déterminons la loi de la valuation des sous-résultants de deux polynômes p-adiques aléatoires unitaires de même degré.

Abstract. We address the problem of the stability of the computations of resultants and subresultants of polynomials defined over complete discrete valuation rings (e.g. $\mathbb{Z}_p$ or $k[\![t]\!]$ where k is a field). We prove that Euclidean-like algorithms are highly unstable on average and we explain, in many cases, how one can stabilize them without sacrifying the complexity. On the way, we completely determine the distribution of the valuation of the subresultants of two random monic p-adic polynomials having the same degree.

1. Introduction

As wonderfully illustrated by the success of Kedlaya-type counting points algorithms [9], p-adic techniques are gaining nowadays more and more popularity in computer science, and more specifically in Symbolic computation. A crucial issue when dealing with p-adics is that of stability. Indeed, just like real numbers, p-adic numbers are by nature infinite and thus need to be truncated in order to fit in the memory of a computer. The level of truncation is called the *precision*. Usual softwares implementing p-adics (e.g. magma [3], pari [12], sagemath [13]) generally tracks precision as follows: an individual precision is attached to any p-adic variable and this precision is updated after each basic arithmetic operation. This way of

Manuscrit reçu le 29 octobre 2015, révisé le 23 février 2016, accepté le 18 mars 2016.
Mathematics Subject Classification. 11S99, 11Y99, 68W30.
Mots-clefs. Euclidean algorithm, ultrametric precision, subresultants.

Degree	Loss of precision (in number of significant digits)	
	Euclidean algorithm	expected
5	6.3	3.1
10	14.3	3.2
25	38.9	3.2
50	79.9	3.2
100	160.0	3.2

FIGURE 1.1. Average loss of precision when computing the GCD of two random monic polynomial of fixed degree over $\mathbb{Z}_2$.

tracking precision can be seen as the analogue of the arithmetic intervals in the real setting. We refer to §2.1.2 for more details.

In the paper [7], the authors proposed a new framework to control p-adic precision. The aim of this paper is to illustrate the techniques of loc. cit. on the concrete example of computation of GCDs and subresultants of p-adic polynomials. There is actually a real need to do this due to the combination of two reasons: on the one hand, computating GCDs is a very basic operation for which it cannot be acceptable to have important instability whereas, on the other hand, easy experimentations show that all standard algorithms for this task (e.g. extended Euclidean algorithm) are *very* unstable.

Figure 1.1 illustrates the instability of the classical extended Euclidean algorithm (cf. Algorithm 1.1) when it is called on random inputs which are monic 2-adic polynomials of fixed degree (see also Example 2.3).

Algorithm 1.1: Extended Euclidean algorithm

Input : Two polynomials $A, B \in \mathbb{Q}_p[X]$ (whose coefficients are known at given precision)

Output: A triple D, U, V such that $D = AU + BV = \gcd(A, B)$

$S_1 \leftarrow A$; $U_1 \leftarrow 1$; $V_1 \leftarrow 0$
$S_2 \leftarrow B$; $U_2 \leftarrow 0$; $V_2 \leftarrow 1$
$k \leftarrow 2$
while $S_k \neq 0$ **do**
    $Q, S_{k+1} \leftarrow$ quotient and remainder in the Euclidean division of S_{k-1} by S_k
    $U_{k+1} \leftarrow U_{k-1} - QU_k$
    $V_{k+1} \leftarrow V_{k-1} - QV_k$
    $k \leftarrow k + 1$
return $S_{k-1}, U_{k-1}, V_{k-1}$

Looking at the last line, we see that extended Euclidean algorithm outputs the Bézout coefficients of two monic 2-adic polynomials of degree 100 with an average loss of 160 significant digits by coefficient whereas a stable algorithm should only loose 3.2 digits on average. This "theoretical" loss is computed as the double of the valuation of the resultant. Indeed Cramer-like formulae imply that Bézout coefficients can be computed by performing a unique division by the resultant, inducing then only the aforementioned loss of precision (see §2.1.2 and (2.8) for a full justification). Examining the table a bit more, we observe that the "practical" loss of precision due to Euclidean algorithm seems to grow linearly with respect to the degree of the input polynomials whereas the "theoretical" loss seems to be independent of it. In other words, the instability of Euclidean algorithm is becoming more and more critical when the degree of the input increases.

Content of the paper. The aim of this article is twofold. We first provide in §3 a theoretical study of the instability phenomenon described above and give strong evidences that the loss of precision grows linearly with respect to the degree of the input polynomials, as we observed empirically. In doing so, we determine the distribution of the valuation of the subresultants of random monic polynomials over $\mathbb{Z}_p$ (cf. Theorem 3.3). This is an independent result which has its own interest.

Our second goal, which is carried out in §4, is to rub out these unexpected losses of precision. Making slight changes to the standard subresultant pseudo-remainder sequence algorithm and using in an essential way the results of [7], we manage to design a stable algorithm for computing all subresultants of two monic polynomials over $\mathbb{Z}_p$ (satisfying an additional assumption). This basically allows to stably compute GCDs assuming that the degree of the GCD is known in advance.

Notation. Here is a summary of the main notations used in this paper. The definitions of many of them will be recalled in §2.

$\mathfrak{A}$	—	a commutative ring
W	—	a complete discrete valuation ring
π	—	a uniformizer of W
K	—	the fraction field of W
k	—	the residue field of W
$\mathfrak{A}_{<n}[X]$	—	the free $\mathfrak{A}$-module consisting of polynomials over $\mathfrak{A}$ of degree $< n$
$\mathfrak{A}_{\leq n}[X]$	—	the free $\mathfrak{A}$-module consisting of polynomials over $\mathfrak{A}$ of degree $\leq n$

$\mathfrak{A}_n[X]$ — the affine space consisting of *monic* polynomials over $\mathfrak{A}$ of degree n.

$\mathrm{Res}^{d_A,d_B}(A,B)$ — The resultant of A and B "computed in degree (d_A,d_B)"

$\mathrm{Res}_j^{d_A,d_B}(A,B)$ — The j-th subresultant of A and B "computed in degree (d_A,d_B)"

2. The setting

The aim of this section is to introduce the setting we shall work in throughout this paper (which is a bit more general than that considered in the introduction).

2.1. Complete discrete valuation rings.

Definition 2.1. A *discrete valuation ring* (DVR for short) is a domain W equipped with a map $\mathrm{val} : W \to \mathbb{N} \cup \{+\infty\}$ — the so-called *valuation* — satisfying the four axioms:

(1) $\mathrm{val}(x) = +\infty$ iff $x = 0$
(2) $\mathrm{val}(xy) = \mathrm{val}(x) + \mathrm{val}(y)$
(3) $\mathrm{val}(x+y) \geq \min(\mathrm{val}(x), \mathrm{val}(y))$
(4) any element of valuation 0 is invertible.

Throughout this paper, we fix a discrete valuation ring W and assume that the valuation on it is normalized so that it takes the value 1. We recall that W admits a unique maximal ideal $\mathfrak{m}$, consisting of elements of positive valuation. This ideal is principal and generated by any element of valuation 1. Such an element is called a *uniformizer*. Let us fix one of them and denote it by π. The *residue field* of W is the quotient $W/\mathfrak{m} = W/\pi W$ and we shall denote it by k.

The valuation defines a distance d on W by letting $d(x,y) = e^{-\mathrm{val}(x-y)}$ for all $x, y \in W$. We say that W is *complete* if it is complete with respect to d, in the sense that every Cauchy sequence converges. Assuming that W is complete, any element $x \in W$ can be written uniquely as a convergent series:

$$x = x_0 + x_1\pi + x_2\pi^2 + \cdots + x_n\pi^n + \cdots \tag{2.1}$$

where the x_i's lie in a fixed set S of representatives of classes modulo π with $0 \in S$.

Let K denote the fraction field of W. The valuation v extends uniquely to K by letting $\mathrm{val}(\frac{x}{y}) = \mathrm{val}(x) - \mathrm{val}(y)$. Moreover, it follows from axiom (4)

that K is obtained from W by inverting π. Thus, any element of K can be uniquely written as an infinite sum:

$$x = \sum_{i=i_0}^{\infty} x_i \pi^i \tag{2.2}$$

where i_0 is some relative integer and the x_i's are as above. The valuation of x can be easily read off this writing: it is the smallest integer i such that $x_i \neq 0$.

2.1.1. ***Examples.*** A first class of examples of discrete valuation rings are rings of formal power series over a field. They are equipped with the standard valuation defined as follows: $\mathrm{val}(\sum_{i\geq 0} a_i t^i)$ is the smallest integer i with $a_i \neq 0$. The corresponding distance on $k[\![t]\!]$ is complete. Indeed, denoting by $f[i]$ the term in t^i in a series $f \in k[\![t]\!]$, we observe that a sequence $(f_n)_{n\geq 0}$ is Cauchy if and only if the sequences $(f_n[i])_{n\geq 0}$ are all ultimately constant. A Cauchy sequence $(f_n)_{n\geq 0}$ therefore converges to $\sum_{i\geq 0} a_i t^i$ where a_i is the limit of $f_n[i]$ when n goes to $+\infty$. The DVR $k[\![t]\!]$ has a distinguished uniformizer, namely t. Its maximal ideal is then the principal ideal (t) and its residue field is canonically isomorphic to k. If one chooses $\pi = t$ and constant polynomials as representatives of classes modulo t, the expansion (2.1) is nothing but the standard writing of a formal series. The fraction field of $k[\![t]\!]$ is the ring of Laurent series over k and, once again, the expansion (2.2) corresponds to the usual writing of Laurent series.

The above example is quite important because it models all complete discrete valuation rings of equal characteristic, i.e. whose fraction field and residue field have the same characteristic. On the contrary, in the mixed characteristic case (i.e. when the fraction field has characteristic 0 and the residue field has positive characteristic), the picture is not that simple. Nevertheless, one can construct several examples and, among them, the most important is certainly the ring of p-adic integers $\mathbb{Z}_p$ (where p is a fixed prime number). It is defined as the projective limit of the finite rings $\mathbb{Z}/p^n\mathbb{Z}$ for n varying in $\mathbb{N}$. In concrete terms, an element of $\mathbb{Z}_p$ is a sequence $(x_n)_{n\geq 0}$ with $x_n \in \mathbb{Z}/p^n\mathbb{Z}$ and $x_{n+1} \equiv x_n \pmod{p^n}$. The addition (resp. multiplication) on $\mathbb{Z}_p$ is the usual coordinate-wise addition (resp. multiplication) on the sequences. The p-adic valuation of $(x_n)_{n\geq 0}$ as above is defined as the smallest integer i such that $x_i \neq 0$. We can easily check that $\mathbb{Z}_p$ equipped with the p-adic valuation satisfies the four above axioms and hence is a DVR. A uniformizer of $\mathbb{Z}_p$ is p and its residue field is $\mathbb{Z}/p\mathbb{Z}$. A canonical set of representatives of classes modulo p is $\{0, 1, \ldots, p-1\}$.

Given a p-adic integer $x = (x_n)_{n\geq 0}$, the i-th digit of x_n in p-basis is well defined as soon as $i < n$ and the compatibility condition $x_{n+1} \equiv x_n \pmod{p^n}$ implies that it does not depend on n. As a consequence, a p-adic

integer can alternatively be represented as a "number" written in p-basis having an infinite number of digits, that is a formal sum of the shape:

$$a_0 + a_1 p + a_2 p^2 + \cdots + a_n p^n + \cdots \quad \text{with } a_i \in \{0, 1, \ldots, p-1\}. \tag{2.3}$$

Additions and multiplications can be performed on the above writing according to the rules we all studied at school (and therefore taking care of carries). Similarly to the equal characteristic case, we prove that $\mathbb{Z}_p$ is complete with respect to the distance associated to the p-adic valuation. The writing (2.3) corresponds to the expansion (2.1) provided that we have chosen $\pi = p$ and $S = \{0, 1, \ldots, p-1\}$. The fraction field of $\mathbb{Z}_p$ is denoted by $\mathbb{Q}_p$.

2.1.2. ***Symbolic computations over DVR.*** We now go back to a general complete discrete valuation ring W, whose fraction field is still denoted by K. The memory of a computer being necessarily finite, it is not possible to represent exhaustively all elements of W. Very often, mimicing what we do for real numbers, we choose to truncate the expansion (2.1) at some finite level. Concretely, this means that we work with approximations of elements of W of the form

$$x = \sum_{i=0}^{N-1} x_i \pi^i + O(\pi^N) \quad \text{with } N \in \mathbb{N} \tag{2.4}$$

where the notation $O(\pi^N)$ means that the x_i's with $i \geq N$ are not specified.

Remark 2.2. From the theoretical point of view, the expression (2.4) does not represent a single element x of W but an open ball in W, namely the ball of radius e^{-N} centered at $\sum_{i=0}^{N-1} x_i \pi^i$ (or actually at any element congruent to it modulo π^N). In other words, on a computer, we cannot work with actual p-adic numbers and we replace them by balls which are more tractable (at least, they can be encoded by a finite amount of information).

The integer N appearing in (2.4) is the so-called *absolute precision* of x. The *relative precision* of x is defined as the difference $N - v$ where v denotes the valuation of x. Continuing the comparison with real numbers, the relative precision corresponds to the number of significant digits since x can be alternatively written:

$$x = p^v \sum_{j=0}^{N-v-1} y_j \pi^j + O(\pi^N) \quad \text{with } y_j = x_{j+v} \text{ and } y_0 \neq 0.$$

Of course, it may happen that all the x_i's ($0 \leq i < N$) vanish, in which case the valuation of x is undetermined. In this particular case, the relative precision of x is undefined.

There exist simple formulae for following precision after each single elementary computation. For instance, basic arithmetic operations can be handled using:

$$(a + O(\pi^{N_a})) + (b + O(\pi^{N_b})) = a + b + O(\pi^{\min(N_a, N_b)}), \tag{2.5}$$

$$(a + O(\pi^{N_a})) - (b + O(\pi^{N_b})) = a - b + O(\pi^{\min(N_a, N_b)}), \tag{2.6}$$

$$(a + O(\pi^{N_a})) \times (b + O(\pi^{N_b})) = ab + O(\pi^{\min(N_a + \mathrm{val}(b), N_b + \mathrm{val}(a))}). \tag{2.7}$$

$$(a + O(\pi^{N_a})) \div (b + O(\pi^{N_b})) = \frac{a}{b} + O(\pi^{\min(N_a - \mathrm{val}(b), N_b + \mathrm{val}(a) - 2\,\mathrm{val}(b))}).^{1} \tag{2.8}$$

with the convention that $\mathrm{val}(a) = N_a$ (resp. $\mathrm{val}(b) = N_b$) if all known digits of a (resp. b) are zero. Combining these formulae, one can track the precision while executing any given algorithm. This is the analogue of the standard *interval arithmetic* over the reals. Many usual softwares (as SAGEMATH, MAGMA) implement p-adic numbers and formal series this way. We shall see later that this often results in overestimating the losses of precision.

Example 2.3. As an illustration, let us examine the behaviour of the precision on the sequence (R_i) while executing Algorithm 1.1 with the input:

$$\begin{aligned} A &= X^5 + (27 + O(2^5))X^4 + (11 + O(2^5))X^3 \\ &\quad + (5 + O(2^5))X^2 + (18 + O(2^5))X + (25 + O(2^5)) \\ B &= X^5 + (24 + O(2^5))X^4 + (25 + O(2^5))X^3 \\ &\quad + (12 + O(2^5))X^2 + (3 + O(2^5))X + (10 + O(2^5)). \end{aligned}$$

The remainder in the Euclidean division of A by B is $S_3 = A - B$. According to (2.6), we do not loose precision while performing this subtraction and the result we get is:

$$\begin{aligned} S_3 &= (3 + O(2^5))X^4 + (18 + O(2^5))X^3 \\ &\quad + (25 + O(2^5))X^2 + (15 + O(2^5))X + (15 + O(2^5)). \end{aligned}$$

In order to compute S_4, we have now to perform the Euclidean division of $S_2 = B$ by S_3. Noting that the leading coefficient of S_2 has valuation 0 and using Eq (2.5)–(2.8), we deduce that this operation does not loose precision again. We get:

$$S_4 = (26 + O(2^5))X^3 + (17 + O(2^5))X^2 + (4 + O(2^5))X + (16 + O(2^5)).$$

[1] We observe that these formulae can be rephrased as follows: the absolute (resp. relative) precision on the result of a sum or a subtraction (resp. a product or a division) is the minimum of the absolute (resp. relative) precisions on the summands/factors.

We observe now that the leading coefficient of S_4 has valuation 1. According to (2.8), divising by this coefficient — and therefore *a fortioti* computing the euclidean division of S_3 by S_4 — will result in loosing at least one digit in relative precision. The result we find is:

$$S_5 = \underbrace{\left(\tfrac{3}{4} + O(2^2)\right)}_{\text{rel. prec.}=4} X^2 + \underbrace{\left(6 + O(2^3)\right)}_{\text{rel. prec.}=2} X + \underbrace{\left(3 + O(2^3)\right)}_{\text{rel. prec.}=3}.$$

Continuing this process, we obtain:

$$S_6 = \left(20 + O(2^5)\right)X + \left(12 + O(2^5)\right) \qquad \text{and} \qquad S_7 = \tfrac{7}{4} + O(2).$$

The relative precision on the final result S_7 is then 3, which is less than the initial precision which was 5.

2.2. Subresultants. A first issue when dealing with numerical computations of GCDs of polynomials over W is that the GCD function is not continuous: it takes the value 1 on an open dense subset without being constant. This of course annihilates any hope of computing GCDs of polynomials when only approximations of them are known. Fortunately, there exists a standard way to recover continuity in this context: it consists in replacing GCDs by subresultants which play an analoguous role. For this reason, in what follows, we will exclusively consider the problem of computing subresultants.

Definitions and notations. We recall briefly basic definitions and results about resultants and subresultants. For a more complete exposition, we refer to [2, §4.2], [8, §3.3] and [14, §4.1]. Let $\mathfrak{A}$ be an arbitrary commutative ring and let A and B be two polynomials with coefficients in $\mathfrak{A}$. We pick in addition two integers d_A and d_B greater than or equal to the degree of A and B respectively. We consider the Sylvester linear mapping:

$$\begin{aligned} \psi : \ \mathfrak{A}_{<d_B}[X] \times \mathfrak{A}_{<d_A}[X] &\to \mathfrak{A}_{<d_A+d_B}[X] \\ (U, V) &\mapsto AU + BV \end{aligned}$$

where $\mathfrak{A}_{<d}[X]$ refers to the finite free $\mathfrak{A}$-module of rank d consisting of polynomials over $\mathfrak{A}$ of degree strictly less than d. The Sylvester matrix is the matrix of ψ in the canonical ordered basis, which are

$$\begin{aligned} &((X^{d_B-1}, 0), \dots, (X, 0), (1, 0), (0, X^{d_A-1}), \dots, (0, 1)) \quad \text{for the source} \\ &\qquad\qquad \text{and} \quad (X^{d_A+d_B-1}, \dots, X, 1) \quad \text{for the target.} \end{aligned}$$

The *resultant* of A and B (computed in degree d_A, d_B) is the determinant of the ψ; we denote it by $\mathrm{Res}^{d_A,d_B}(A, B)$. We observe that it vanishes if $d_A > \deg A$ or $d_B > \deg B$. In what follows, we will freely drop the exponent d_A, d_B if d_A and d_B are the degrees of A and B respectively. Using

Cramer formulae, we can build polynomials $U^{d_A,d_B}(A,B) \in \mathfrak{A}_{<d_B}[X]$ and $V^{d_A,d_B}(A,B) \in \mathfrak{A}_{<d_A}[X]$ satisfying the two following conditions:

(1) their coefficients are, up to a sign, maximal minors of the Sylvester matrix, and
(2) $A \cdot U^{d_A,d_B}(A,B) + B \cdot V^{d_A,d_B}(A,B) = \mathrm{Res}^{d_A,d_B}(A,B)$.

These polynomials are called the *cofactors* of A and B (computed in degree d_A, d_B).

The subresultants are defined in the similar fashion. Given an integer j in the range $[0,d)$ where $d = \min(d_A,d_B)$, we consider the "truncated" Sylvester linear mapping:

$$\begin{aligned}\psi_j : \ \mathfrak{A}_{<d_B-j}[X] \times \mathfrak{A}_{<d_A-j}[X] &\to \mathfrak{A}_{<d_A+d_B-j}[X]/\mathfrak{A}_{<j}[X] \\ (U,V) &\mapsto AU + BV.\end{aligned}$$

Its determinant (in the canonical basis) is the j-th *principal subresultant* of A and B (computed in degree d_A, d_B). Just as before, we can construct polynomials $U_j^{d_A,d_B}(A,B) \in \mathfrak{A}_{<d_B-j}[X]$ and $V_j^{d_A,d_B}(A,B) \in \mathfrak{A}_{<d_A-j}[X]$ such that:

(1) their coefficients are, up to a sign, minors of the Sylvester matrix[2], and
(2) $A \cdot U_j^{d_A,d_B}(A,B) + B \cdot V_j^{d_A,d_B}(A,B) \equiv \det \psi_j \pmod{\mathfrak{A}_{<j}[X]}$.

We set $R_j^{d_A,d_B}(A,B) = A \cdot U_j^{d_A,d_B}(A,B) + B \cdot V_j^{d_A,d_B}(A,B)$: it is the j-th *subresultant* of A and B (computed in degree d_A, d_B). The above congruence implies that $R_j^{d_A,d_B}(A,B)$ has degree at most j and that its coefficient of degree j is the j-th principal subresultant of A and B. As before, we freely drop the exponent d_A, d_B when d_A and d_B are equal to the degrees of A and B respectively. When $j = 0$, the application ψ_j is nothing but ψ. Therefore, $\mathrm{Res}_0^{d_A,d_B}(A,B) = \mathrm{Res}^{d_A,d_B}(A,B)$ and, similarly, the cofactors agree: we have $U_0^{d_A,d_B}(A,B) = U^{d_A,d_B}(A,B)$ and $V_0^{d_A,d_B}(A,B) = V^{d_A,d_B}(A,B)$.

We recall the following very classical result.

Theorem 2.4. *We assume that $\mathfrak{A}$ is a field. Let A and B be two polynomials with coefficients in $\mathfrak{A}$. Let j be the smallest integer such that $\mathrm{Res}_j(A,B)$ does not vanish. Then $\mathrm{Res}_j(A,B)$ is a* GCD *of A and B.*

Since they are defined as determinants, subresultants behave well with respect to base change: if $f : \mathfrak{A} \to \mathfrak{A}'$ is a morphism of rings and A and B are polynomials over $\mathfrak{A}$ then $\mathrm{Res}_j^{d_A,d_B}(f(A),f(B)) = f\big(\mathrm{Res}_j^{d_A,d_B}(A,B)\big)$ where $f(A)$ and $f(B)$ denotes the polynomials deduced from A and B respectively by applying f coefficient-wise. This property is sometimes referred to as the *functoriality* of subresultants. We emphasize that, when f is not

[2]Indeed, observe that the matrix of ψ_j is a submatrix of the Sylvester matrix.

injective, the relation $\mathrm{Res}_j(f(A), f(B)) = f\big(\mathrm{Res}_j(A,B)\big)$ does *not* hold in general since applying f may decrease the degree. Nevertheless, if d_A and d_B remained fixed, this issue cannot happen.

The subresultant pseudo-remainder sequence. When $\mathfrak{A}$ is a domain, there exists a standard nice Euclidean-like reinterpreation of subresultants, which provides in particular an efficient algorithm for computing them. Since it will play an important role in this paper, we take a few lines to recall it.

This reinterpretation is based on the so-called *subresultant pseudo-remainder sequence* which is defined as follows. We pick A and B as above. Denoting by $(P \,\%\, Q)$ the remainder in the Euclidean division of P by Q, we define two recursive sequences (S_i) and (c_i) as follows:

$$\text{(2.9)} \qquad \begin{cases} S_{-1} = A,\ S_0 = B,\ c_{-1} = 1 & \\ S_{i+1} = (-s_i)^{\varepsilon_i+1} s_{i-1}^{-1}\, c_i^{-\varepsilon_i} \cdot (S_{i-1} \,\%\, S_i) & \text{for } i \geq 0 \\ c_{i+1} = s_{i+1}^{\varepsilon_{i+1}} \cdot c_i^{1-\varepsilon_{i+1}} & \text{for } i \geq -1. \end{cases}$$

Here $n_i = \deg S_i$, $\varepsilon_i = n_{i+1} - n_i$ and s_i is the leading coefficient of S_i if $i \geq 0$ and $s_{-1} = 1$ by convention. These sequences are finite and the above recurrence applies until S_i has reached the value 0.

Proposition 2.5. *With the above notations, we have:*

$$\begin{aligned} \mathrm{Res}_j(A,B) &= S_i && \text{if } j = n_{i-1} - 1 \\ &= 0 && \text{if } n_i < j < n_{i-1} - 1 \\ &= \left(\frac{s_i}{s_{i-1}}\right)^{\varepsilon_i - 1} \cdot S_i && \text{if } j = n_i \end{aligned}$$

for all i such that S_i is defined.

Remark 2.6. The Proposition 2.5 provides a formula for *all* subresultants. We note moreover that, in the common case where $n_{i-1} = n_i - 1$, the two formulae giving $\mathrm{Res}_{n_i}(A,B)$ agree.

Mimicing ideas behind extended Euclidean algorithm, one can define the "extended subresultant pseudo-remainder sequence" as well and obtains recursive formulae for cofactors at the same time.

Important simplifications occur in the "normal" case, which is the case where all principal subresultants do not vanish. Under this additional assumption, one can prove that the degrees of the S_i's decrease by one at each step; in other words, $\deg S_i = d_B - i$ for all i. The sequence (S_i) then stops at $i = d_B$. Moreover, the ε_i's and the c_i's are now all "trivial": we have $\varepsilon_i = 1$ and $c_i = s_i$ for all i. The recurrence formula then becomes:

$$S_{i+1} = s_i^2 \cdot s_{i-1}^{-2} \cdot (S_{i-1} \,\%\, S_i) \quad \text{for } i \geq 1.$$

and Proposition 2.5 now simply states that $R_j = S_{d_B - j}$. In other words, still assuming that all principal subresultants do not vanish, the sequence of subresultants obeys to the recurrence:

$$R_{d+1} = A, \quad R_d = B, \qquad R_{j-1} = r_j^2 \cdot r_{j+1}^{-2} \cdot (R_{j+1} \,\%\, R_j) \tag{2.10}$$

where r_j is the leading coefficient of R_j for $j \leq d$ and $r_{d+1} = 1$ by convention. Moreover, a similar recurrence exists for cofactors as well:

$$U_{d+1} = 1, \quad U_d = 0, \qquad U_{j-1} = r_j^2 \cdot r_{j+1}^{-2} \cdot (U_{j+1} - Q_j U_j) \tag{2.11}$$

$$V_{d+1} = 0, \quad U_d = 1, \qquad V_{j-1} = r_j^2 \cdot r_{j+1}^{-2} \cdot (V_{j+1} - Q_j V_j) \tag{2.12}$$

where Q_j is quotient in the Euclidean division of R_{j+1} by R_j.

Proposition 2.5 of course yields an algorithm for computing subresultants. In the normal case and assuming further for simplicity that the input polynomials are monic of same degree, it is Algorithm 2.1, which uses the primitive `prem` for computing pseudo-remainders. We recall that the pseudo-remainder of the division of A by B is the polynomial $\texttt{prem}(A, B)$ defined by $\texttt{prem}(A, B) = \mathrm{lc}(B)^{\deg B - \deg A + 1}(A \% B)$ where $\mathrm{lc}(B)$ denotes the leading coefficient of B.

Algorithm 2.1: Subresultant pseudo remainder sequence algorithm

Input : Two polynomials $A, B \in K_d[X]$ (given at finite precision)
Output: The complete sequence of subresultants of A and B.

$R_d \leftarrow B$; $r_d \leftarrow 1$
$R_{d-1} \leftarrow B - A$
for $j = (d-1), (d-2), \ldots, 1$ **do**
  $r_j \leftarrow$ coefficient in X^j of R_j
  if $r_j = 0$ **then raise** NotImplementedError;
  $R_{j-1} \leftarrow \texttt{prem}(R_{j+1}, R_j)/r_{j+1}^2$
return $R_{d-1}, \ldots, R_0$

Unfortunately, while working over a complete discrete valuation field K, the stability of Algorithm 2.1 is as bad as that of standard Euclidean algorithm. The use of Algorithm 2.1 is interesting because it avoids denominators (i.e. we always work over W instead K) but it does not improve the stability.

Example 2.7. Applying Algorithm 2.1 with the input (A, B) of Example 2.3, we obtain:

$$\begin{aligned}
R_4 &= (29+O(2^5))X^4 + (14+O(2^5))X^3 + (5+O(2^5))X^2 \\
&\qquad + (17+O(2^5))X + (17+O(2^5)) \\
R_3 &= (4+O(2^5))X^3 + (13+O(2^5))X^2 + (4+O(2^5))X + (16+O(2^5)) \\
R_2 &= (5+O(2^5))X^2 + (20+O(2^5))X + O(2^5) \\
R_1 &= (1+O(2))X + (1+O(2)) \\
R_0 &= 1+O(2)
\end{aligned}$$

We observe in particular that the absolute precision on R_0 is 1, although it should be at least 5 since R_0 is given by an integral polynomial expression in terms of the coefficients of A and B. We note moreover that the relative precision on R_0 (which is 1 as well) is worse that the relative precision we got on S_7 (which was 3) while executing Algorithm 1.1 (cf. Example 2.3).

3. Unstability of Euclidean-like algorithms

In this section, we provide strong evidences for explaining the average loss of precision observed while executing Algorithm 2.1. Concretely, in §3.1 we establish[3] a lower bound on the losses of precision which depends on extra parameters, that are the valuations of the principal subresultants. The next subsections (§§3.2 and 3.3) aim at studying the behaviour of these valuations on random inputs; they thus have a strong probabilistic flavour.

Remark 3.1. The locution *Euclidean-like algorithms* (which appears in the title of the Section) refers to the family of algorithms computing GCDs or subresultants by means of successive Euclidean divisions. We believe that the stability of all algorithms in this family is comparable since we are precisely loosing precision while performing Euclidean divisions. Among all algorithms in this family, we chose to focus on Algorithm 2.1 because it is simpler due to the fact that it only manipulates polynomials with coefficients in W. Nevertheless, our method extends to many other Euclidean-like algorithms including Algorithm 1.1; this extension is left as an exercice to the reader.

3.1. A lower bound on losses of precision. We consider two fixed polynomials A and B with coefficients in W whose coefficients are known with precision $O(\pi^N)$ for some positive integer N. For simplicity, we assume further that A and B are both monic and share the same degree d. For any integer j between 0 and $d-1$, we denote by R_j the j-th subresultant of A and B.

[3] in a model of precision which is slightly weaker that the usual one; we refer to §3.1 for a complete discussion about this.

In this subsection, we estimate the loss of precision if we compute the R_j's using the recurrence (2.10). In what follows, we are going to use a *flat precision model*: this means that a polynomial $P(X)$ is internally represented as:

$$P(X) = \sum_{i=1}^{n} a_i X^i + O(\pi^N) \quad \text{with } a_i \in K \text{ and } N \in \mathbb{Z}.$$

In other words, we assume that the software we are using does not carry a precision data on each coefficient but only a unique global precision datum. Concretely this means that, after having computing a polynomial, the software truncates the precision on each coefficient to the smallest one. One can argue that this assumption is too strong (compared to usual implementations of p-adic numbers). Nevertheless, it defines a simplified framework in which computations can be carried out and experiments show that it rather well reflects the behaviour of the loss of precision observed in practice.

Let V_j be the valuation of the principal j-th subresultant of A, B and W_j be the minimum of the valuations of the coefficients of R_j. Of course we have $V_j \geq W_j$ and we set $\delta_j = V_j - W_j$.

Proposition 3.2. *Let A and B as above. Either Algorithm 2.1 fails or it outputs the subresultants R_j's at precision $O(\pi^{N_j})$ with:*

$$N_j \leq N + V_{j+1} - 2 \cdot (\delta_{j+1} + \delta_{j+2} + \cdots + \delta_{d-1}).$$

Proof. Using that R_{j+1} and R_j have the expected degrees, the remainder $(R_{j+1} \,\%\, R_j)$ is computed as follows:

$$\begin{aligned} \text{we set:} \quad & S = R_{j+1} - r_{j+1} \cdot r_j^{-1} \cdot R_j \\ \text{and we have:} \quad & R_{j+1} \,\%\, R_j = S - s \cdot r_j^{-1} \cdot R_j \end{aligned}$$

where s is the coefficient of degree j of S. Let us first estimate the precision of S. Using (2.7)–(2.8), we find that the computed relative precision on $r_{j+1} \cdot r_j^{-1} \cdot R_j$ is $\min(N_{j+1} - V_{j+1}, N_j - V_j)$. The absolute precision of this value is then $M = \min(N_{j+1} - \delta_j, N_j - \delta_j + V_{j+1} - V_j)$. The latter is also the precision of S since the first summand R_{j+1} is known with higher precision. Repeating the argument, we find that the precision of $(R_{j+1} \,\%\, R_j)$ is equal to $\min(M - \delta_j, N_j - \delta_j + \mathrm{val}(s) - V_j)$ and therefore is lower bounded by $M - \delta_j \leq N_j - 2\delta_j + V_{j+1} - V_j$. From this, we derive $N_{j-1} \leq N_j - 2\delta_j - V_{j+1} + V_j$ and the proposition finally follows by summing up these inequalities. □

The difference $N - N_0 = -V_1 + 2\sum_{k=1}^{d} \delta_j$ is a lower bound on the number of digits lost after the computation of the resultant using the subresultant pseudo-remainder sequence algorithm. In the next subsection (cf. Corollary 3.6), we shall see that V_1 and all δ_j's are approximatively equal to $\frac{1}{p-1}$ on average. The loss of precision then grows linearly with respect to d. This

confirms the precision benchmarks shown in Figure 1.1. We emphasize one more time that this loss of precision is *not* intrinsic but an artefact of the algorithm we have used; indeed, one should not loose any precision when computing resultants because they are given by polynomial expressions.

3.2. Behaviour on random inputs. Proposition 3.2 gives an estimation of the loss of precision in Euclidean-like algorithms in terms of the quantities V_j and δ_j. It is nevertheless *a priori* not clear how large these numbers are. The aim of this paragraph is to compute their order of magnitude when A and B are picked randomly among the set of monic polynomials of degree d with coefficients in W. In what follows, we assume that the residue field $k = W/\pi W$ is finite and we use the letter q to denote its cardinality.

We endow W with its Haar measure. The set Ω of couples of monic polynomial of degree d with coefficients in W is canonically in bijective correspondence with W^{2d} and hence inherits the product measure. We consider V_j, W_j and δ_j as random variables defined on Ω.

Theorem 3.3. *We fix $j \in \{0, \dots, d-1\}$. Let $X_0, \dots, X_{d-1}$ be d pairwise independent discrete random variables with geometric law of parameter $(1 - q^{-1})$, i.e.*

$$\mathbb{P}[X_i = k] = (1 - q^{-1}) \cdot q^{-k} \qquad \text{with } 0 \leq i < d \text{ and } k \in \mathbb{N}.$$

Then V_j is distributed as the random variable

$$Y_j = \sum_{i=0}^{d} \min(X_{j-i}, X_{j-i+1}, \dots, X_{j+i})$$

with $X_i = +\infty$ if $i < 0$ and $X_i = 0$ if $i \geq d$.

Remark 3.4. The above Theorem does not say anything about the correlations between the X_j's. In particular, we emphasize that it is *false* that the tuple $(V_{d-1}, \dots, V_0)$ is distributed as $(Y_{d-1}, \dots, Y_0)$. For instance, one can prove that (V_{d-1}, V_{d-2}) is distributed as $(X,\ X' + \min(X', [X/2]))$ where X and X' are two independent discrete random variables with geometric law of parameter $(1 - q^{-1})$ and the notation $[\,\cdot\,]$ stands for the integer part function. In particular, we observe that $(V_{d-1}, V_{d-2}) \neq (2, 1)$ almost surely although the events $\{V_{d-1} = 2\}$ and $\{V_{d-2} = 1\}$ both occur with positive probability.

Nonetheless, a consequence of Proposition 3.10 below is that the variables $\bar{V}_j = \mathbb{1}_{\{V_j = 0\}}$ are mutually independent.

Theorem 3.5. *For all $j \in \{0, \dots, d-1\}$ and all $m \in \mathbb{N}$, we have:*

$$\mathbb{P}[\delta_j \geq m] \geq \frac{(q-1)(q^j - 1)}{q^{j+1} - 1} q^{-m}.$$

The proof of Theorem 3.3 and Theorem 3.5 will be given in §3.3. We now derive some consequences. Let σ denote the following permutation:

$$\begin{pmatrix} 1 & 2 & \cdots & \frac{d}{2} & \frac{d}{2}+1 & \frac{d}{2}+2 & \cdots & d \\ 1 & 3 & \cdots & d-1 & d & d-2 & \cdots & 2 \end{pmatrix} \quad \text{if } 2 \mid d$$

$$\begin{pmatrix} 1 & 2 & \cdots & \frac{d+1}{2} & \frac{d+3}{2} & \frac{d+5}{2} & \cdots & d \\ 1 & 3 & \cdots & d & d-1 & d-3 & \cdots & 2 \end{pmatrix} \quad \text{if } 2 \nmid d.$$

In other words, σ takes first the odd values in $[1, d]$ in increasing order and then the even values in the same range in decreasing order.

Corollary 3.6. *For all $j \in \{0, \ldots, d-1\}$, we have:*

(1) $\mathbb{E}[V_j] = \sum_{i=1}^{d-j} \frac{1}{q^{\sigma(i)}-1}$; *in particular* $\frac{1}{q-1} \leq \mathbb{E}[V_j] < \frac{q}{(q-1)^2}$

(2) $\frac{q^j-1}{q^{j+1}-1} \leq \mathbb{E}[\delta_j] \leq \mathbb{E}[V_j]$

(3) $\sigma[V_j]^2 = \sum_{i=1}^{d-j} \frac{(2i-1)\cdot q^{\sigma(i)}}{(q^{\sigma(i)}-1)^2}$; *in particular* $\frac{\sqrt{q}}{q-1} \leq \sigma[V_j] < \frac{q\sqrt{q+1}}{(q-1)^2}$

(4) $\mathbb{P}[V_j \geq m] \leq q^{-m+O(\sqrt{m})}$

(5) $\mathbb{E}[\max(V_0, \ldots, V_{d-1})] \leq \log_q d + O(\sqrt{\log_q d})$

Proof. By Theorem 3.3, we have $\mathbb{E}[V_j] = \sum_{i=0}^{d} \mathbb{E}[Z_i]$ with

$$Z_i = \min(X_{j-i}, \ldots, X_{j+i})$$

(j is fixed during all the proof). Our conventions imply that Z_i vanishes if $i \geq d-j$. Let us define $\tau(1), \ldots, \tau(d-j)$ as the numbers $\sigma(1), \ldots, \sigma(d-j)$ sorted in increasing order. For $i < d-j$, the random variable Z_i is the minimum of $\tau(i)$ independent random variables with geometric distribution of parameter $(1-q^{-1})$. Thus its distribution is geometric of parameter $(1-q^{-\tau(i)})$. Its expected value is then $\frac{1}{q^{\tau(i)}-1}$ and the first formula follows. The inequality $\frac{1}{q-1} \leq \mathbb{E}[V_j]$ is clear because $\frac{1}{q-1}$ is the first summand in the expansion of $\mathbb{E}[V_j]$. The upper bound is derived as follows:

$$\mathbb{E}[V_j] < \sum_{i=0}^{\infty} \frac{1}{q^i-1} \leq \sum_{i=0}^{\infty} \frac{1}{q^i-q^{i-1}} = \frac{q}{(q-1)^2}.$$

The first inequality of claim (2) is obtained from the relation

$$\mathbb{E}[\delta_j] = \sum_{m=1}^{\infty} m \cdot \mathbb{P}[\delta_j = m] = \sum_{m=1}^{\infty} \mathbb{P}[\delta_j \geq m]$$

using the estimation of Theorem 3.5. The second inequality is clear because $\delta_j \leq V_j$.

The variance of V_j is related to the covariance of Z_i's thanks to the formula

$$\mathrm{Var}(V_j) = \sum_{1 \leq i,i' \leq d-j} \mathrm{Cov}(Z_i, Z_{i'}).$$

Moreover, given X and X' two independent variables having geometric distribution of parameter $(1 - a^{-1})$ and $(1 - b^{-1})$ respectively, a direct computation gives:

$$\mathrm{Cov}(X, \min(X, X')) = \frac{ab}{(ab-1)^2}.$$

Applying this to our setting, we get:

$$\mathrm{Cov}(Z_i, Z_{i'}) = \frac{q^{e(i,i')}}{(q^{e(i,i')} - 1)^2}$$

where $e(i,i') = \min(\tau(i), \tau(i')) = \tau(\min(i,i'))$. Summing up these contributions, we get the equality in (3). The inequalities are derived from this similarly to what we have done in (1).

We now prove (4). Let $(G_i)_{i \geq 0}$ be a countable family of mutually independent random variables with geometric distribution of parameter $(1 - q^{-1})$. We set $G = \sum_{i=1}^{\infty} \min(G_1, \ldots, G_i)$. Thanks to Theorem 3.3, it is enough to prove that $\mathbb{P}[G \geq m] \leq q^{-m+O(\sqrt{m})}$. We introduce the event E_m formulated as follows:

(E_m) *There exists a partition* $\underline{m} = (m_1, \ldots, m_\ell)$ *of* m *such that* $G_i \geq m_i$ *for all* $i \leq \ell$.

We claim that E_m contains the event $\{G \geq m\}$. Indeed assume $G \geq m$ and set $m'_i = \min(G_1, \ldots, G_i)$. Clearly, the sequence $(m'_i)_{i \geq 1}$ is nondecreasing and $\sum_{i=1}^{\infty} m'_i \geq m$ by assumption. Therefore there exists a partition $(m_1, \ldots, m_\ell)$ of m with $m_i \leq m'_i$ for all i. These m_i's also satisfy $G_i \geq m_i$ and our clair is then proved. We derive $\mathbb{P}[G \geq m] \leq \mathbb{P}[E_m]$ and therefore:

$$\mathbb{P}[G \geq m] \leq \sum_{\underline{m}} \prod_{i=1}^{\ell} \mathbb{P}[G_i \geq m_i]$$

where the latter sum runs over all partitions $\underline{m} = (m_1, \ldots, m_\ell)$ of m. Replacing $\mathbb{P}[G_i \geq m_i]$ by q^{-m_i}, we get $\mathbb{P}[E_m] \leq p(m) \cdot q^{-m}$ where $p(m)$ denotes the number of partitions of m. By a famous formula [1], we know that $\log p(m)$ is equivalent to $\pi\sqrt{2m/3}$. In particular $p(m) \in q^{O(\sqrt{m})}$ and (4) is proved.

We now derive (5) by a standard argument. It follows from (4) that

$$\mathbb{P}[\max(V_0, \ldots, V_{d-1})] \leq d \cdot q^{-m+c\sqrt{m}}$$

for some constant c. Therefore:

$$\mathbb{E}[\max(V_0, \dots, V_{d-1})] \leq \sum_{m=1}^{\infty} \min(1, d \cdot q^{-m+c\sqrt{m}}).$$

Let m_0 denote the smallest index such that $d \cdot q^{-m_0+c\sqrt{m_0}}$, i.e. $m_0 - c\sqrt{m_0} \geq \log_q d$. Solving the latest equation, we get $m_0 = \log_q d + O(\sqrt{\log_q d})$. Moreover $\sum_{m=m_0}^{\infty} d\, q^{-m+c\sqrt{m}}$ is bounded independently of d. The result follows. □

3.3. Proof of Theorems 3.3 and 3.5. During the proof, A and B will always refer to monic polynomials of degree d and R_j (resp. U_j and V_j) to their j-th subresultant (resp. their j-th cofactors). If P is a polynomial and n is a positive integer, we use the notation $P[n]$ to refer to the coefficient of X^n in P. We set $r_j = R_j[j]$.

Preliminaries on subresultants. We collect here various useful relations between subresultants and cofactors. During all these preliminaries, we work over an arbitrary base ring $\mathfrak{A}$.

Proposition 3.7. *The following relations hold:*

- $U_{j-1}V_j - U_jV_{j-1} = (-1)^j r_j^2$;
- $U_j[d-j-1] = -V_j[d-j-1] = (-1)^j r_{j+1}$;
- $\mathrm{Res}_k^{j,j-1}(R_j, R_{j-1}) = r_j^{2(j-k-1)}\, R_k$ *for* $k < j$;
- $\mathrm{Res}_k^{d-j,d-j-1}(U_{j-1}, U_j) = r_j^{2(d-j-k-1)}\, U_{d-1-k}$ *for* $k < d-j$.

Moreover r_j depends only on the $2(d-j)-1$ coefficients of highest degree of A and B.

Proof. By functoriality of subresultants, we may assume that $\mathfrak{A}$ is the ring $\mathbb{Z}[a_0, \dots, a_{d-1}, b_0, \dots, b_{d-1}]$ and that A and B are the two generic monic polynomials $A = X^d + \sum_{i=0} a_i X^i$ and $B = X^d + \sum_{i=0} b_i X^i$. Under this additional assumption, all principal subresultant are nonzero. Therefore, the sequences $(R_j)_j$, $(U_j)_j$ and $(V_j)_j$ are given by the recurrences (2.10)–(2.12). The two first announced relations follow easily. Let now focus on the third one. We set $\tilde{R}_j = R_j$ and $\tilde{R}_k = r_j^{2(j-k-1)}\, R_k$ for $k < j$. An easy decreasing induction on k shows that this sequence obeys to the recurrence:

$$\tilde{R}_{k-1} = \tilde{r}_k^2 \cdot \tilde{r}_{k+1}^{-2} \cdot (\tilde{R}_{k+1} \,\%\, \tilde{R}_k)$$

where $\tilde{r}_j = 1$ and $\tilde{r}_k$ is the coefficient of $\tilde{R}_k$ of degree k for all $k < j$. Comparing with (2.10), this implies that $\tilde{R}_k$ is the k-th subresultant of the pair (R_j, R_{j-1}) and we are done. The fourth equality and the last statement are proved in a similar fashion. □

For any fixed index $j \in \{1, \dots, d-1\}$, we consider the function ψ_j that takes a couple $(A, B) \in \mathfrak{A}_d[X]^2$ to the quadruple $(U_j, U_{j-1}, R_j, R_{j-1})$. It follows from Proposition 3.7 that ψ_j takes its values in the subset $\mathcal{E}_j$ of

$$\big(\mathfrak{A}_{\leq d-j-1}[X]\big) \times \big(\mathfrak{A}_{\leq d-j}[X]\big) \times \big(\mathfrak{A}_{\leq j}[X]\big) \times \big(\mathfrak{A}_{\leq j-1}[X]\big)$$

consisting of the quadruples $(\mathcal{U}_j, \mathcal{U}_{j-1}, \mathcal{R}_j, \mathcal{R}_{j-1})$ such that:

$$\begin{aligned} \mathcal{U}_{j-1}[d-j] &= (-1)^{j-1}\, \mathcal{R}_j[j] \\ \text{and } \ \mathrm{Res}^{d-j,d-j-1}(\mathcal{U}_{j-1}, \mathcal{U}_j) &= -\mathcal{R}_j[j]^{2(d-j-1)}. \end{aligned}$$

Let $\mathcal{E}_j^\times$ be the subset of $\mathcal{E}_j$ defined by requiring that $\mathcal{R}_j[j]$ is invertible in $\mathfrak{A}$. In the same way, we define $\Omega_j^\times$ as the subset of $\mathfrak{A}_d[X]^2$ consisting of couples (A, B) whose j-th principal subresultants (in degree (d, d)) is invertible in $\mathfrak{A}$.

Proposition 3.8. *The function ψ_j induces a bijection between $\Omega_j^\times$ and $\mathcal{E}_j^\times$.*

Proof. We are going to define the inverse of ψ_j. We fix a quadruple $(\mathcal{U}_j, \mathcal{U}_{j-1}, \mathcal{R}_j, \mathcal{R}_{j-1})$ in $\mathcal{E}_j^\times$ and set $a = \mathcal{R}_j[j]$. Let $\mathcal{W}_j$ and $\mathcal{W}_{j-1}$ denote the cofactors of $(\mathcal{U}_{j-1}, \mathcal{U}_j)$ in degree $(d-j, d-j-1)$. Define $\mathcal{V}_j = \alpha \mathcal{W}_j$ and $\mathcal{V}_{j-1} = -\alpha \mathcal{W}_{j-1}$ where $\alpha = a^{2j-2d+4}$. The relation:

$$\mathcal{U}_{j-1}\mathcal{V}_j - \mathcal{U}_j\mathcal{V}_{j-1} = a^2. \tag{3.1}$$

then holds. We now define A and B using the formulae:

$$\begin{cases} A = (-1)^j \cdot a^{-2} \cdot (\mathcal{V}_j \mathcal{R}_{j-1} - \mathcal{V}_{j-1}\mathcal{R}_j) \\ B = (-1)^{j-1} \cdot a^{-2} \cdot (\mathcal{U}_j \mathcal{R}_{j-1} - \mathcal{U}_{j-1}\mathcal{R}_j). \end{cases} \tag{3.2}$$

They are both monic of degree d, so that we can define φ_j as the function taking $(\mathcal{U}_j, \mathcal{U}_{j-1}, \mathcal{R}_j, \mathcal{R}_{j-1})$ to (A, B). The composite $\varphi_j \circ \psi_j$ is easily checked to be the identity: indeed, if $\psi_j(A, B) = (\mathcal{U}_j, \mathcal{U}_{j-1}, \mathcal{R}_j, \mathcal{R}_{j-1})$, the relation (3.1) implies that $\mathcal{V}_{j-1}$ and $\mathcal{V}_j$ are the missing cofactors (up to a sign) and, consequently, A and B have to be given by the system (3.2).

To conclude the proof, it remains to prove that the composite in the other direction $\psi_j \circ \varphi_j$ is the identity as well. Since both φ_j and ψ_j are componantwise given by polynomials, we can use functoriality and assume that $\mathfrak{A}$ is the field $\mathbb{Q}(c_0, c_1, \dots, c_n)$ (with $n = 2d$) where each variable c_i corresponds to one coefficient of $\mathcal{U}_j$, $\mathcal{U}_{j-1}$, $\mathcal{R}_j$ and $\mathcal{R}_{j-1}$ with the convention that c_0 (resp. $(-1)^{j-1}c_0$) is used for the leading coefficients of $\mathcal{R}_j$ (resp. $\mathcal{U}_{j-1}$). Set:

$$\begin{aligned} (A, B) &= \varphi_j(\mathcal{U}_j, \mathcal{U}_{j-1}, \mathcal{R}_j, \mathcal{R}_{j-1}) \\ \text{and } \ (U_j, U_{j-1}, R_j, R_{j-1}) &= \psi_j(A, B) \end{aligned}$$

Since $\mathfrak{A}$ is a field and $\mathcal{R}_j[j]$ does not vanish, the Sylvester mapping

$$\begin{aligned}\mathfrak{A}_{<d-j}[X] \times \mathfrak{A}_{<d-j}[X] &\to \mathfrak{A}_{<2d-j}[X]/\mathfrak{A}_{<j}[X]\\ (U,V) &\mapsto AU+BV\end{aligned}$$

has to be bijective. Therefore there must exist $\lambda \in \mathfrak{A}$ such that $\mathcal{R}_j = \lambda \cdot R_j$ and $\mathcal{U}_j = \lambda \cdot U_j$. Similarly $(\mathcal{R}_{j-1},\mathcal{U}_{j-1}) = \mu \cdot (R_{j-1}, U_{j-1})$ for some $\mu \in \mathfrak{A}$. Identifying the leadings coefficients, we get $\lambda = \mu$. Now observe that:

$$\begin{aligned}\mathrm{Res}^{d-j,d-j-1}(\mathcal{U}_{j-1},\mathcal{U}_j) &= \mathrm{Res}^{d-j,d-j-1}(\lambda U_{j-1}, \lambda U_j)\\ &= \lambda^{2d-2j-1} \cdot \mathrm{Res}^{d-j,d-j-1}(U_{j-1}, U_j).\end{aligned}$$

Noting that $(\mathcal{U}_j,\mathcal{U}_{j-1},\mathcal{R}_j,\mathcal{R}_{j-1})$ and $(U_j,U_{j-1},R_j,R_{j-1})$ both belong to $\mathcal{E}_j$, we derive:

$$\mathcal{R}_j[j]^{2(d-j-1)} = \lambda^{2d-2j-1} \cdot R_j[j]^{2(d-j-1)}$$

from what we finally get $\lambda = 1$ since $\mathcal{R}_j = \lambda \cdot R_j$. □

Corollary 3.9. *We assume that $\mathfrak{A} = W$. Then the map $\psi_j : \Omega_j^\times \to \mathcal{E}_j^\times$ preserves the Haar measure.*

Proof. Proposition 3.8 applied with the quotient rings $\mathfrak{A} = W/\pi^n W$ shows that $(\psi_j \bmod \pi^n)$ is a bijection for all n. This proves the Corollary. □

The distribution in the residue field. We assume in this paragraph that $\mathfrak{A}$ is a finite field of cardinality q. We equip $\Omega_{\mathfrak{A}} = \mathfrak{A}_d[X]^2$ with the uniform distribution. For $j \in \{0,\dots,d-1\}$ and $(A,B) \in \Omega_{\mathfrak{A}}$, we set $\bar{V}_j(A,B) = 1$ if $r_j(A,B)$ vanishes and $\bar{V}_j(A,B) = 0$ otherwise. The functions $\bar{V}_j$'s define random variables over $\Omega_{\mathfrak{A}}$.

Proposition 3.10. *With the above notations, the $\bar{V}_j$'s are mutually independent and they all follow a Bernoulli distribution of parameter $\frac{1}{q}$.*

Proof. Given $J \subset \{0,\dots,d-1\}$, we denote by $\Omega_{\mathfrak{A}}(J)$ the subset of $\Omega_{\mathfrak{A}}$ consisting of couples (A,B) for which $r_j(A,B)$ does not vanish if and only if $j \in J$. We want to prove that $\Omega_{\mathfrak{A}}(J)$ has cardinality $q^{2d-\mathrm{Card}\,J}(q-1)^{\mathrm{Card}\,J}$. To do this, we introduce several additional notations. First, we write $J = \{n_1,\dots,n_\ell\}$ with $n_1 > n_2 > \cdots > n_\ell$ and set $n_{\ell+1} = 0$ by convention. Given n and m two integers with $m < n$, we let $V_{[m,n)}$ denote the set of polynomials of the form $a_m X^m + a_{m+1}X^{m+1} \cdots + a_n X^n$ with $a_i \in \mathfrak{A}$ and $a_n \neq 0$. Clearly, $V_{[m,n)}$ has cardinality $(q-1)q^{n-m}$. If P is any polynomial of degree n and $m < n$ is an integer, we further define $P[m:] \in V_{[m,n)}$ as the polynomial obtained from P by removing its monomials of degree $< m$. Finally, given (A,B) in $\Omega_{\mathfrak{A}}$, we denote by $(S_i(A,B))$ its subresultant pseudo-remainder sequence as defined in §2.2. We note that, if $(A,B) \in \Omega_{\mathfrak{A}}(J)$, the sequence

$(S_i(A,B))$ stops at $i=\ell$ and we have $\deg S_i = n_i$ for all i. We now claim that the mapping

$$\begin{aligned}\Lambda_J :\ \Omega_{\mathfrak{A}}(J) &\to V_{[n_1,n_2)} \times \cdots \times V_{[n_\ell,n_{\ell+1})} \\ (A,B) &\mapsto \big(S_i(A,B)[n_{i+1}:]\big)_{1\le i\le \ell}\end{aligned}$$

is injective. In order to establish the claim, we remark that the knowledge of $S_{i-1}(A,B)$ and $S_i(A,B)[n_{i+1}:]$ (for some i) is enough to reconstruct the quotient of the Euclidean division of $S_i(A,B)$ by $S_{i-1}(A,B)$. Thus, one can reconstruct $S_i(A,B)$ from the knowledge of $S_{i-2}(A,B)$, $S_{i-1}(A,B)$ and $S_i(A,B)[n_{i+1}:]$. We deduce that $\Lambda_J(A,B)$ determines uniquely all $S_i(A,B)$'s and finally A and B themselves. This proves the claim.

To conclude the proof, we note that the claim implies that the cardinality of $\Omega_{\mathfrak{A}}(J)$ is at most $q^{2d-\ell}(q-1)^\ell$. Summing up these inequalities over all possible J, we get $\operatorname{Card}\Omega_{\mathfrak{A}} \le q^{2d}$. This latest inequality being an equality, we must have $\operatorname{Card}\Omega_{\mathfrak{A}}(J) = q^{2d-\operatorname{Card} J}(q-1)^{\operatorname{Card} J}$ for all J. □

Proof of Theorem 3.5. We assume first that $j < d-1$. Proposition 3.10 above ensures that r_{j+1} is invertible in W with probability $(1-q^{-1})$. Moreover, assuming that this event holds, Corollary 3.9 implies that R_j is distributed in $W_{\le j}[X]$ according to the Haar measure. An easy computation gives $\mathbb{P}[\delta_j \ge m \,|\, r_{j+1} \in W^\times] = \frac{q(q^j-1)}{q^{j+1}-1}$ and therefore:

$$\mathbb{P}[\delta_j \ge m] \ge (1-q^{-1}) \cdot \frac{q(q^j-1)}{q^{j+1}-1} = \frac{(q-1)(q^j-1)}{q^{j+1}-1}.$$

The case $j = d-1$ is actually simpler. Indeed, the same argument works except that we know for sure that $r_{j+1} = r_d$ is invertible since it is equal to 1 by convention. In that case, the probability is then equal to $\frac{q(q^j-1)}{q^{j+1}-1}$. □

Proof of Theorem 3.3. We fix $j \in \{0,\dots,d-1\}$. We define the random variable $V_j^{(0)}$ as the greatest (nonnegative) integer v such that all principal subresultants $r_{j'}$ have positive valuation for j' varying in the open range $(j-v, j+v)$ (with the convention that $r_{j'} = 0$ whenever $j' < 0$). It is clear from the definition that r_{j-v} or r_{j+v} (with $v = V_j^{(0)}$) has valuation 0. Moreover, assuming first that $\operatorname{val}(r_{j+v}) = 0$, we get by Proposition 3.7:

$$\begin{aligned}\operatorname{val}(r_j) &= v + \operatorname{val}\big(r_v^{j-v,j-v+1}(A^{(1)},B^{(1)})\big) \\ \text{with}\quad A^{(1)} &= \frac{1}{r_{j+v}X^{j-v-1}} \cdot R_{j+v}[j-v-1:], \\ \text{and}\quad B^{(1)} &= A^{(1)} + \frac{1}{\pi X^{j-v-1}} \cdot R_{j+v-1}[j-v-1:]\end{aligned}$$

where we recall that, given a polynomial P and an integer m, the notation $P[m:]$ refers to the polynomial obtained from P by removing its monomials

of degree strictly less than m. We notice that all the coefficients of $B^{(1)}$ lie in W because $r_{j'}$ has positive valuation for $j' \in (j-v, j+v)$. Furthermore, Corollary 3.9 shows that the couple $(A^{(1)}, B^{(1)})$ is distributed according to the Haar measure on $(W_{2v-1}[X])^2$. If $\mathrm{val}(r_{j+v}) = 0$, one can argue similarly by replacing R_{j+v} and R_{j+v-1} by the cofactors U_{j-v} and U_{j-v+1} respectively. Replacing (A, B) by $(A^{(1)}, B^{(1)})$, we can now define a new random variable $V_j^{(1)}$ and, continuing this way, we construct an infinite sequence $V_j^{(m)}$ such that $V_j = \sum_{m\geq 0} V_j^{(m)}$.

We now introduce a double sequence $(X_i^{(m)})_{0\leq i<d, m\geq 0}$ of mutually independent random variables with Bernoulli distribution of parameter $\frac{1}{q}$ and we agree to set $X_{j'}^{(m)} = 0$ for $j' < 0$ and $X_{j'}^{(m)} = 1$ for $j \geq d$. It follows from Proposition 3.10 (applied with $\mathfrak{A} = k$) that $V_j^{(0)}$ has the same distribution than $Y_j^{(0)} = \sum_{i=1}^{d} \min(X_{j-i}^{(0)}, \ldots, X_{j+i}^{(0)})$. In the same way, keeping in mind that $A^{(1)}$ and $B^{(1)}$ have both degree $2V_j^{(0)} - 1$, we find that $V_j^{(1)}$ has the same distribution than $\sum_{i=1}^{V_j^{(0)}-1} \min(X_{j-i}^{(1)}, \ldots, X_{j+i}^{(1)})$, which can be rewritten as $Y_j^{(1)} = \sum_{i=1}^{d} \min(X_{j-i}^{(0)}, X_{j-i}^{(1)}, \ldots, X_{j+i}^{(0)}, X_{j+i}^{(1)})$. More precisely, the equidistribution of $(A^{(1)}, B^{(1)})$ shows that the joint distribution $(V_j^{(0)}, V_j^{(1)})$ is the same as that of $(Y_j^{(0)}, Y_j^{(1)})$. Repeating the argument, we see that $(V_j^{(m)})_{m\geq 0}$ is distributed as $(Y_j^{(m)})_{m\geq 0}$ where:

$$Y_j^{(m)} = \sum_{i=1}^{d} \min(X_{j-i}^{(0)}, \ldots X_{j-i}^{(m)}, \ldots, X_{j+i}^{(0)}, \ldots, X_{j+i}^{(m)}).$$

Setting finally $X_i = \sum_{m\geq 0} \min(X_1^{(0)}, \ldots, X_i^{(m)})$, we find the X_i's $(0 \leq i < d)$ are mutually independent and that they all follow a geometric distribution of parameter $(1 - q^{-1})$. We now conclude the proof by noting that Y_j equals $\sum_{i=1}^{d} \min(X_{j-i}, \ldots, X_{j+i})$ (recall that the $X_i^{(m)}$'s only take the values 0 and 1). □

4. A stabilized algorithm for computing subresultants

We have seen in the previous sections that Euclidean-like algorithm are unstable in practice. On the other hand, one can compute subresultants in a very stable way by evaluating the corresponding minors of the Sylvester matrix. Doing so, we do not loose any significant digit. Of course, the downside is the rather bad efficiency.

In this section, we design an algorithm which combines the two advantages: it has the same complexity than Euclidean algorithm and it is very stable in the sense that it does not loose any significant digit. This algorithm

is deduced from the subresultant pseudo-remainder sequence algorithm by applying a "stabilization process", whose inspiration comes from [7].

4.1. Crash course on ultrametric precision. In this subsection, we briefly report on and complete the results of [7] where the authors draw the lines of a general framework to handle a sharp (often optimal) track of ultrametric precision. We also refer to [6] for a complete exposition of the theory including many discussions and examples. In what follows, the letter W still refers to a complete DVR while the letter K is used for its fraction field.

4.1.1. ***The notion of lattice.*** As underlined in Remark 2.2, the usual way of tracking precision consists in replacing elements of W — which cannot fit entirely in the memory of a computer — by balls around them. Using this framework, a software manipulating d variables in W will work with d "independent" balls. The main proposal of [7] is to get rid of this "independence" and model precision using a unique object contained in a d-dimensional vector space. In order to be more precise, we need the following definition.

Definition 4.1. A W-lattice in a finite dimensional vector space E over K is a W-submodule of E generated by a K-basis of E.

Although the definition of a lattice is similar to that of $\mathbb{Z}$-lattice in $\mathbb{R}^d$, the geometrical representation of it is quite different. Indeed, the elements of W themselves are not distributed as $\mathbb{Z}$ is in $\mathbb{R}$ but rather form a ball inside K (they are exactly elements of norm ≤ 1). More generally, assume that E is equipped with a ultrametric norm $\|\cdot\|_E$ compatible with that on K (i.e. $\|\lambda x\|_E = |\lambda| \cdot \|x\|_E$ for $\lambda \in K$, $x \in E$). (A typical example is $E = K^n$ equipped with the sup norm.) One checks that the balls

$$B_E(r) = \{\, x \in E \;\mid\; \|x\|_E \leq r \,\}$$

are all lattices in E. Moreover, any lattice is deduced from $B_E(1)$ by applying a linear automorphism of E. Therefore, lattices should be thought as special neighborhoods of 0 (see Figure 4.1).

As a consequence, cosets of the form $x + H$, where H is a lattice, appear as interesting candidates to model precision. This feeling is consolidated by the following result which roughly speaking claims that such cosets behave quite well under differentiable maps.

Lemma 4.2 ([7, Lemma 3.4])**.** *Let E and F be two normed finite dimensional K-vector spaces. Let $f : E \to F$ be a function of class C^1 and let x be a point in K^n at which the differential of f, denoted by $f'(x)$, is surjective. Then, for all $\rho \in (0, 1]$, there exists $\delta > 0$ such that the following equality*

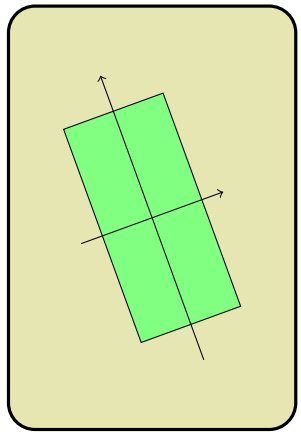

FIGURE 4.1. Picture of a lattice in the ultrametric world

holds:

$$f(x+H) = f(x) + f'(x)(H) \tag{4.1}$$

for any lattice H satisfying $B_E(\rho r) \subset H \subset B_E(r)$ for some $r < \delta$.

In what follows, we will often use Lemma 4.2 with $\rho = 1$. It states in this particular case that

$$f(x + B_E(r)) = f(x) + f'(x)(B_E(r)) \tag{4.2}$$

as soon as r is small enough. It is moreover possible to provide an explicit upper bound on r assuming that f has more regularity. The case of locally analytic functions is treated in [7] in full generality. Nevertheless, for the application we have in mind, it will be enough to restrict ourselves to the simpler case of *integral polynomial* functions. In order to proceed, we assume that E is endowed with distinguished "orthonormal" basis, that is a basis $(e_1, \ldots, e_n)$ with the property that $\| \sum_{i=1}^n x_i e_i \|_E = \max_{1 \leq i \leq n} |x_i|$ for all families of x_i's lying in K. In other words, the choice of this distinguished "orthonormal" basis defines a norm-preserving isomorphism between E and K^n endowed with the sup norm. We assume similarly that we are given a distinguished "orthonormal" basis $(f_1, \ldots, f_m)$ of F. Then any function $f : E \to F$ can be written in our distinguished system of coordinates as follows:

$$f(x) = \sum_{j=1}^{m} F_j(x_1, \ldots, x_n) f_j \quad \text{with} \quad x = \sum_{i=1}^{n} x_i e_i.$$

Definition 4.3. The function f is *integral polynomial* if all F_j's are polynomials functions with coefficients in W.

Example 4.4. Let us examine more closely the case of polynomial spaces since it will be considered repeatedly in the sequel. We take $E = K_{<n}[X]$ and $F = K_{<m}[X]$ and endow both with the Gauss norm, which is defined by:

$$\|a_0 + a_1X + \cdots + a_{n-1}X^{n-1}\|_E = \max(|a_0|, |a_1|, \ldots, |a_{n-1}|)$$
$$\|b_0 + b_1X + \cdots + b_{m-1}X^{m-1}\|_F = \max(|b_0|, |b_1|, \ldots, |b_{m-1}|)$$

It is clear from these definitions that the canonical basis $(1, X, \ldots, X^{n-1})$ and $(1, X, \ldots, X^{m-1})$ of E and F respectively are "orthonormal". Moreover the coordinates in these basis are the a_i's and the b_i's respectively. Hence, an integral polynomial function $f : E \to F$ is nothing but a function mapping a polynomial P to a polynomial Q whose coefficients are given by polynomial formulae which involve only the coefficients of P and constants in W.

Obviously, all integral polynomial functions are function of class C^1 (and even locally analytic), so that Lemma 4.2 applies to them. Proposition 4.5 below exhibits an explcit value for the bound δ appearing in Lemma 4.2 when f is integral polynomial and $r = 1$.

Proposition 4.5. *Let $f : E \to F$ be an integral polynomial function and $x \in B_E(1)$. Then,* (4.2) *holds as soon as $B_F(r) \subset f'(x)(B_E(1))$.*

Proof. It is a direct corollary of [7, Proposition 3.12]. □

4.1.2. *Application to precision.* Let us now briefly explain how Lemma 4.2 can be utilized for tracking precision.

Tracking precision locally. Assume first that we want to perform a given simple operation — corresponding, say, to an elementary step (e.g. an iteration of the main loop) of the algorithm we are executing — modeled by a function g of class C^1 defined on an open subset U of a finite dimensional normed K-vector space E and taking values in another finite dimensional normed K-vector space F. Our input is an approximated element of U which is represented by a coset C with respect to some lattice H, that is a subset of U of the form $C = x + H$ for some $x \in U$. We would like to insist on the following: the value of x is a priori *not* given; only is given the subset C. However, since H is stable under addtion, we have $C = x + H$ for *any* element $x \in C$.[4] As explained in §2.1.2, assuming that g is given as an algebraic expression, the naive solution for evaluating $g(C)$ consists in using formulae (2.5)–(2.8). However, this often results in an overestimation on the precision, in the following sense: this method leads to some inclusion

$$g(C) = g(x + H) \subset y + H_{\text{naive}}$$

[4]This assertion means that any element of the "rectangle" C is a center of it... which might be surprising if we are accustomed to real numbers.

where $y \in F$ and H_{naive} is a lattice which is generally much more larger that $g'(x)(H)$, the latter being the best possible one according to Lemma 4.2 (if the assumptions of this Lemma are fullfiled). In order to avoid this and be sharp on precision, another solution consists in splitting the computation of $g(C)$ into two parts as follows:

(A) compute $g'(x)(H)$, and
(B) compute $g(x)$ for some $x \in C$.

Part (A) is not easy to handle in full generality: in order to be efficient, a special close analysis taking advantage of the particular problem under consideration is often necessary. For now, we assume that we are given two lattices $H_{\min}$ and $H_{\max}$ with the property that:

$$H_{\min} \subset g'(x)(H) \subset H_{\max}. \tag{4.3}$$

We shall see later (cf. §4.2) how these lattices can be constructed — for a negligible cost — in the special case of subresultants.

We now focus on part (B), which also requires some discussion. Indeed, computing $g(x)$ is not straightforward because x itself lies in a K-vector space and therefore cannot be stored and manipulated on a computer. Nevertheless, one can take advantage of the fact that x may be chosen arbitrarily in C. More precisely, we pick a sublattice H' of H and consider the new approximated element $x + H' \subset x + H$. Concretely, this means that we arbitrarily increase the precision on the given input x. Now, applying the naive method with $x + H'$, we end up with some $y \in F$ and some lattice $H'_{\text{naive}} \subset F$ with the property that:

$$g(x + H') \subset y + H'_{\text{naive}}$$

(see Figure 4.2). If furthemore H' is chosen in such a way that $H'_{\text{naive}} \subset H_{\min}$, the two cosets $y + g'(x)(H)$ and $g(C)$ have a non-empty intersection because $g(x)$ lies in both. Therefore they must coincide. We deduce that $y \in g(C)$. This exactly means that y is an acceptable value for $g(x)$ and we are done. Moreover, estimating the dependance of H'_{naive} in terms of H' is usually rather easy (remember that g is supposed to model a simple operation). Hence since $H_{\min}$ is known — as we had assumed — finding H' satisfying the required assumption is generally not difficult.

Tracking precision globally. As already said, we shall use the above method for tracking precision while executing a single step in a complete algorithm. Let us now address the problem of "glueing". We consider an algorithm $\mathtt{F}$ consisting in a succession of n steps $\mathtt{G}_0, \ldots, \mathtt{G}_{n-1}$. It is modeled by a function $f : U \to F$ of class C^1 where U is an open subset in a finite dimensional normed K-vector space E and F is a finite dimensional normed K-vector space. The input of $\mathtt{F}$ is an approximated element in U represented as a coset $C = x + H$ where $x \in U$ and H is a lattice. We also

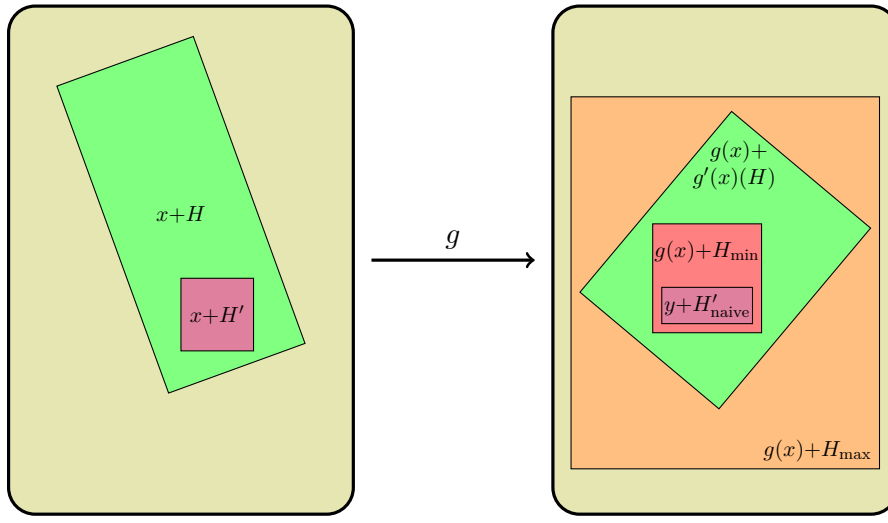

FIGURE 4.2. Method for tracking precision based on Lemma 4.2

introduce notations for each individual step. For all i, we assume that $\mathtt{G}_i$ is modeled by a function $g_i : U_i \to U_{i+1}$ of class C^1 where U_i is an open subset is some normed K-vector space E_i and, by convention, $U_0 = U$, $E_0 = E$ and $U_n = E_n = F$. We thus have:

$$f = g_{n-1} \circ g_{n-2} \circ \cdots \circ g_1 \circ g_0.$$

For all i, we set $f_i = g_{i-1} \circ \cdots \circ g_0$. It is the function modeling the execution of the i first steps of our algorithm. We further define $x_i = f_i(x)$ and $H_i = f_i'(x)(H)$. The chain rule for composing differentials readily implies the recurrence

$$H_{i+1} = g_i'(x_i)(H_i) \tag{4.4}$$

For simplicity, we make the following assumptions:

- the $\mathbb{Z}_p$-submodule H_i is a lattice in E_i such that $x_i + H_i \subset U_i$;
- the triple (g_i, x_i, H_i) satisfies the assumptions of Lemma 4.2;
- for all i, we have succeeded in finding (good enough) explicit lattices $H_{\min,i}$ and $H_{\max,i}$ such that $H_{\min,i} \subset H_i \subset H_{\max,i}$;
- for all i, we have succeeded in finding an explicit lattice H_i' such that, while tracking naively precision, we end up with an inclusion

$$g_i(x_i + H_i') = x_{i+1} + H_{\text{naive},i+1}$$

with $H_{\text{naive},i+1} \subset H_{\min,i+1}$.

We note that the first and the second assumptions are quite strong because they imply in particular that the sequence of $\dim E_i$ is non-increasing.

However, it really simplifies the forthcoming discussion and will be harmless for the application developed in this paper. As already mentionned, the construction of $H_{\min,i}$ and $H_{\max,i}$ will generally follow from a *theoretical* argument depending on the setting, while exhibiting H'_i will often be straightforward.

We are now in position to apply the method for tracking precision locally we have discussed earlier to all g_i's. This leads to a *stabilized* version of the algorithm `F` whose skeleton is depicted in Algorithm 4.1.

Algorithm 4.1: Stabilized version of `F`

Input : x given at precision $O(H)$
Output: $f(x)$ given at precision $O(H_{\max,n})$

$x_0 \leftarrow x$
for $i = 0, \ldots, n-1$ **do**
  lift x_i to precision $O(H'_i)$
  $x_{i+1} \leftarrow \mathtt{G}_i(x_i)$
return $x_n + O(H_{\max,n})$

The correctness of Algorithm 4.1 (under the assumptions listed above) is clear after Lemma 4.2.

4.2. Application to subresultants. We now apply the theory presented in §4.1 above to the problem of computing subresultants, i.e. the abstract Algorithm `F` is now instantiated to Algorithm 2.1. We split this algorithm into steps in the obvious manner, each step corresponding to an iteration of the main loop. We thus consider the functions:

$$\begin{aligned} g_d \,:\, K_d[X] \times K_d[X] &\to K_d[X] \times K_{\leq d-1}[X] \\ (A, B) &\mapsto (B, A - B) \end{aligned}$$

$$\begin{aligned} \text{and} \quad g_j \,:\, K_{\leq j+1}[X] \times K_{\leq j}[X] &\to K_{\leq j}[X] \times K_{\leq j-1}[X] \\ (R_{j+1}, R_j) &\mapsto (R_j, R_{j-1}) \end{aligned}$$

where R_{j-1} is defined as usual by $R_{j-1} = r_j^2 \cdot r_{j+1}^{-2} \cdot (R_{j+1} \,\%\, R_j)$ where r_j (resp. r_{j+1}) stands for the coefficient of degree j in R_j (resp. of degree $j+1$ in R_{j+1}). We remark that g_j is only defined on the subset consisting of pairs (R_{j+1}, R_j) for which R_{j+1} has degree $j+1$; this reflects the fact that Algorithm 2.1 fails on inputs for which at least one principal subresultant vanishes. The composite function $f = g_1 \circ \cdots \circ g_d$ (be careful with the order of the indices) models (a slight variant of) Algorithm 2.1. For all j, we put

$f_j = g_{j+1} \circ \cdots \circ g_d$; it is the function:

$$\begin{aligned} f_j \,:\, K_d[X] \times K_d[X] &\to K_{\leq j}[X] \times K_{\leq j-1}[X] \\ (A,B) &\mapsto (\mathrm{Res}_j(A,B), \mathrm{Res}_{j-1}(A,B)). \end{aligned}$$

For simplicity, we assume in addition that the precision on the input (A,B) is *flat*, meaning that all coefficients of A and B are known with the same absolute precision $O(\pi^N)$. In the language of §4.1, this flat precision corresponds to the lattice $H = \pi^N \mathcal{L}$ where $\mathcal{L} = W_{<d}[X] \times W_{<d}[X]$ is the unit ball in $K_d[X] \times K_d[X]$ with respect to the Gauss norm (cf. Example 4.4). Following §4.1, our first task consists in finding two lattices $H_{\min,j}$ and $H_{\max,j}$ having the property that $H_{\min,j} \subset f'_j(A,B)(H) \subset H_{\max,j}$. This is achieved by the Lemma.

Lemma 4.6. *For all $(A,B) \in K_d[X]^2$, we have:*

$$r_j^2 \cdot \mathcal{L}_j \subset f'_j(A,B)(\mathcal{L}) \subset \mathcal{L}_j$$

where r_j is the j-th principal subresultant of (A,B) and $\mathcal{L}_j = W_{\leq j}[X] \times W_{\leq j-1}[X]$ is the unit ball in $K_{\leq j}[X] \times K_{\leq j-1}[X]$.

Proof. The second inclusion is clear because f_j is a polynomial function. Let us prove the first inclusion. One may of course assume that r_j does not vanish, otherwise there is nothing to prove. Now, we remark that f_j factors through the function ψ_j introduced in §3.3. By continuity, the j-th principal subresultant function does not vanish on a neighborhood of (A,B). By Proposition 3.8, ψ_j is injective on this neighborhood. Therefore so is f_j. Furthermore, a close look at the proof of Proposition 3.8 indicates that a left inverse of f_j is the function mapping (S_j, S_{j-1}) to

$$(-1)^j \cdot r_j^{-2} \cdot (V_j S_{j-1} - V_{j-1} S_j,\ -U_j S_{j-1} + U_{j-1} S_j)$$

where U_j, V_j (resp. U_{j-1}, V_{j-1}) are the j-th (resp $(j-1)$-th) cofactors of (A,B). Differenting this, we get the announced result. □

Lemma 4.6 ensures that one can safely take $H_{\min,j} = r_j^2 \cdot \pi^N \mathcal{L}_j$ and $H_{\max,j} = \pi^N \mathcal{L}_j$. It finally remains to construct the lattice $H'_j \subset K_{\leq j}[X] \times K_{\leq j-1}[X]$. For this, we remark that a naive track of precision leads to a loss of at most $2 \cdot \mathrm{val}(r_{j+1})$ digits while executing the step $\mathtt{G}_j$ (see also proof of Proposition 3.2 for similar considerations). Therefore, one can take $H'_j = r_j^2 r_{j+1}^2 \cdot \pi^N \mathcal{L}_j$. Instantiating Algorithm 4.1 in this particular case, we end up with Algorithm 4.2 below which then appears as a stable version of Algorithm 2.1.

Algorithm 4.2: Stabilized version of Algorithm 2.1

Input : Two polynomials $A, B \in K_d[X]$ given at flat precision $O(\pi^n)$
Output: The sequence of subresultants of A and B given at flat precision $O(\pi^n)$

$R_d \leftarrow B$; $r_d \leftarrow 1$
$R_{d-1} \leftarrow B - A$
for $j = (d-1), (d-2), \ldots, 1$ **do**
  $r_j \leftarrow$ coefficient in X^j of R_j
  if $\mathrm{val}(r_j) \geq \frac{N}{2}$ **then raise** NotImplementedError;
  lift (R_{j+1}, R_j) at precision $O(\pi^{N+2\,\mathrm{val}(r_j)+2\,\mathrm{val}(r_{j+1}))})$
  $R_{j-1} \leftarrow \mathtt{prem}(R_{j+1}, R_j)/r_{j+1}^2$
return $R_{d-1} + O(\pi^N), \ldots, R_0 + O(\pi^N)$

Proposition 4.7. *Algorithm 4.2 computes all subresultants of* (A, B) *at precision* $O(\pi^N)$ *under the following assumption*[5]

(H) *all principal subresultants of* (A, B) *do not vanish modulo* $\pi^{N/2}$.

It runs in $O(d^2 \cdot \mathtt{M}(N + \max(V_0, \ldots, V_{d-1}))$ *bit operations where* V_j *denotes the valuation of* r_j *and* $\mathtt{M}(n)$ *is the number of bit operations needed to perform an arithmetic operation (addition, product, division) in* W *at precision* $O(\pi^n)$.

Remark 4.8. In all usual examples (p-adic numbers, Laurent series), one can choose $\mathtt{M}(n)$ to be quasi-linear in n and the size of the residue field k.

Proof. Correctness has been already proved (the assumption (H) ensures that Proposition 4.5 applies to each g_j). As usual Euclidean algorithm, Algorithm 1.1 requires $O(d^2)$ operations in the base ring W. Moreover, we observe that the maximal precision at which we are computing is upper bounded by $N + 2\max(V_0, \ldots, V_{d-1})$. This justifies the announced complexity. □

According to Corollary 3.6, the expected value of $\max(V_0, \ldots, V_{d-1})$ is in $O(\log_p d)$. Thus, the average complexity of Algorithm 1.1 is

$$O(d^2 \cdot \mathtt{M}(N + \log d))$$

bit operations. In all usual cases (cf. Remark 4.8), this complexity is also $\tilde{O}(d^2 N \cdot \log |k|)$ bit operations.

To conclude with, let us comment on briefly the hypothesis (H). We first remark that it is satisfied with high probability if N is large compared to $2 \cdot \log_d p$. Thus, replacing if necessary N by $3 \cdot \log_d p$ (which does not affect

[5] If this assumption is not fullfiled, the algorithms fails and returns an error.

the asymptotic behavior of the complexity), it is harmless on average — but might be not on particularly bad instances. We moreover underline that, if we are just interested in computing the j-th subresultant for a particular j, then we just need to assume the non-vanishing of the principal subresultants in the range $[j+1, d-1]$.

Open questions. The first hypothesis we would like to relax is, of course, (H). Actually, it seems quite plausible that one can produce a stabilized version of the "complete"[6] subresultant pseudo-remainder sequence algorithm following the same strategy. Nevertheless, this extension is not completely straightforward because designing it requires to understand precisely how the coefficients c_i's (appearing in (2.9)) alter the behaviour of the precision. We therefore let it as an open question.

As it was presented, Algorithm 4.2 only accepts inputs consisting of a pair of *monic* polynomials having the same degree. It is actually not difficult to make it work with all couples of polynomials (A, B) such that $\mathrm{lc}(B)$ is invertible in W and $\deg A \geq \deg B$. Indeed, it is enough for this to replace line 2 by:

$$R_{d-1} \leftarrow (-1)^{\deg A - \deg B}(A \,\%\, B).$$

However, writing an extension of Algorithm 4.2 that accepts all inputs seems much more tricky and this is the second open question we raise.

Beyond this, one may wonder if one can use similar techniques to compute not only subresultants but cofactors as well. For those indexes j such that r_j is invertible in W, the same analysis applies almost *verbatim*. However for other indexes j, the differential computation seems to be much more subtle. One can get around this issue by using lifting methods only when r_j is a unit in W and tracking precision naively otherwise: it is possible to get this way a stable algorithm whose average running time is acceptable but which seems to be bad in the worst case. Can we do better?

Another quite interesting question is that of designing an algorithm which combines the precision technology developed in this paper with the "half-GCD" methods. It is actually closely related to the previous question because "half-GCD" methods make an intensive use of cofactors in order to speed up the computation.

5. Conclusion: towards p-adic floats

When computing with real numbers, computers very often use floating point arithmetic. The rough idea of this model consists in representating all real numbers using the same number of digits (the so-called *precision*) and to apply rounding heuristics when final digits are unsettled. In comparison with arithmetic interval, floating point arithmetic has two main advantages.

[6] I.e. dealing with abnormal sequences as well.

First, it allows simple and fast implementations. Second, experiments show that the obtained results have generally more much correct digits than the number predicted by arithmetic interval. The counterpart is that obtaining proved results requires a specific (and often subtle) analysis.

In the p-adic setting, the analogue of floating point arithmetic was proposed by Kedlaya and Roe in a unpublished note [10] but has not been developed and studied so far. One reason for this is probably the well-known saying: "in the p-adic world, rounding errors do not accumulate". Consequently one might expect that interval arithmetic would provide sharp results. Nonetheless this hope is failing and examples are basic and numerous: p-adic differential equations [4, 11], LU factorization [5], SOMOS 4 sequence [7], resultants (this paper), *etc.* Consequently, interval arithmetic is not as good as one might have expected at first. Therefore, it probably makes sense to seriously study the analogue of floating point arithmetic in a ultrametric context.

Let us describe quickly what might be this analogue and what are its advantages and disadvantages. We keep the notations of the previous sections: the letter W denotes a complete discrete valuation ring with uniformizer π and K is its fraction field. In the model of ultrametric floating point arithmetic, we fix a positive integer N (the *precision*) and represent elements of K by approximations of the form:

$$\pi^e \cdot \sum_{i=0}^{N-1} x_i \pi^i \tag{5.1}$$

where e is a relative integer and the x_i's are elements of a fixed set of representatives of W modulo π with the convention that the representative of $0 \in k$ is $0 \in W$. We further assume that $x_0 \neq 0$, i.e. e is the valuation of the sum (5.1). We see that this framework is quite similar to usual floating point arithmetics: the integer e plays the role of exponent, the uniformizer π plays the role of the basis and the value $\sum_{i=0}^{N-1} x_i \pi^i$ plays the role of the significand (the mantissa). It remains to define operations $\oplus$ and $\odot$ on approximations modeling addition and multiplication on K respectively. We do this as follows: given x and y two elements of K of the form (5.1), we compute $x + y$ (resp. xy) in K, expand it as a convergent series $\sum_{i=v}^{\infty} s_i \pi^i$ (with $s_v \neq 0$) and define $x \oplus y$ (resp. $x \odot y$) by truncating the series at $i = v + N$.

Similarly to real floating point arithmetic, the main advantages of ultrametric floating point arithmetic are the simplicity and the efficiency while the counterpart is the difficulty to get proved results. Moreover, the aforementioned examples are evidences that ultrametric floating point arithmetic may often compute much more correct digits than the number

predicted by a naive analysis based on interval arithmetic. In order to illustrate this last assertion, let us go back to the case of resultants discussed earlier in this paper. Let A and B be two monic polynomials of degree d (picked at random) whose coefficients are all known at precision $O(\pi^N)$. We have proved that if we are using the model of interval arithmetic, then the subresultant pseudo-remainder sequence algorithm will output $\mathrm{Res}(A,B)$ at precision $O(\pi^{N-N_{\mathrm{int}}})$ where N_{int} grows linearly with respect to d on average. On the other hand, if we are using ultrametric floating point arithmetic, then the same algorithm will output $\mathrm{Res}(A,B)$ at precision $O(\pi^{N-N_{\mathrm{float}}})$ where N_{float} grows linearly with respect to $\log d$ on average. We emphasize furthermore that this result is *proved*! From this point of view, floating point arithmetics seems to behave better in the ultrametric setting: we may hope to get proved results relatively cheaply.

References

[1] G. E. ANDREWS, *The Theory of Partitions*, Encyclopedia of Mathematics and Its Applications, Cambridge University Press, 1976, xiv+255 pages.

[2] S. BASU, R. POLLACK & M.-F. ROY, *Algorithms in Real Algebraic Geometry*, 2nd ed., Algorithms and Computation in Mathematics, vol. 10, Springer, 2006, x+662 pages.

[3] W. BOSMA, J. CANNON & C. PLAYOUST, "The Magma algebra system. I. The user language.", *J. Symb. Comput.* **24** (1997), no. 3-4, p. 235-265.

[4] A. BOSTAN, L. GONZÁLEZ-VEGA, H. PERDRY & É. SCHOST, "From Newton sums to coefficients: complexity issues in characteristic p", in *MEGA'05*, 2005.

[5] X. CARUSO, "Random matrices over a DVR and LU factorization", *J. Symb. Comput.* **71** (2015), p. 98-123.

[6] ———, "Computations with p-adic numbers", preprint, 2017.

[7] X. CARUSO, D. ROE & T. VACCON, "Tracking p-adic precision", *LMS J. Comput. Math.* **17A** (2014), p. 274-294.

[8] H. COHEN, *A course in Computational Algebraic Number Theory*, Graduate Texts in Mathematics, vol. 138, Springer, 1993, xxi+534 pages.

[9] K. S. KEDLAYA, "Counting points on hyperelliptic curves using Monsky–Washnitzer cohomology", *J. Ramanujan Math. Soc.* **16** (2001), no. 4, p. 323-338, errata in *ibid.*, **18** (2003), no. 4, 417-418.

[10] K. S. KEDLAYA & D. ROE, "Two specifications for p-adic floating-point arithmetic: a Sage enhancement proposal", personal communication.

[11] P. LAIREZ & T. VACCON, "On p-adic differential equations with separation of variables", preprint, 2016.

[12] THE PARI GROUP, "PARI/GP version `2.9.0`", 2016, available from `http://pari.math.u-bordeaux.fr/`.

[13] THE SAGE DEVELOPERS, "SageMath, the Sage Mathematics Software System (Version 7.5.1)", 2016, `http://www.sagemath.org/`.

[14] F. WINKLER, *Polynomial Algorithms in Computer Algebra*, Texts and Monographs in Symbolic Computation, Springer, 1996, vii+272 pages.

Xavier CARUSO
Université Rennes 1, IRMAR
35042 Rennes Cedex, France
E-mail: `xavier.caruso@normalesup.org`
URL: `http://perso.univ-rennes1.fr/xavier.caruso`