Computations with p-adic numbers

Xavier Caruso

January 23, 2017

Abstract

This document contains the notes of a lecture I gave at the "Journées Nationales du Calcul Formel[1]" (JNCF) on January 2017. The aim of the lecture was to discuss low-level algorithmics for p-adic numbers. It is divided into two main parts: first, we present various implementations of p-adic numbers and compare them and second, we introduce a general framework for studying precision issues and apply it in several concrete situations.

Contents

Introduction

The field of p-adic numbers, $\mathbb{Q}_p$, was first introduced by Kurt Hensel at the end of the 19th century in a short paper written in German [36]. From that time, the popularity of p-adic numbers has grown without interruption throughout the 20th century. Their first success was materialized by the famous Hasse–Minkowski's Theorem [75] that states that a Diophantine equation of the form $P(x_1, \dots, x_n) = 0$ where P is a polynomial of total degree at most 2 has a solution over $\mathbb{Q}$ if and only if it has a solution over $\mathbb{R}$ and a solution over $\mathbb{Q}_p$ for all prime numbers p. This characterization is quite interesting because testing whether a polynomial equation has a p-adic solution can be carried out in a very efficient way using *analytic* methods just like over the reals. This kind of strategy is nowadays ubiquitous in many areas of Number Theory and Arithmetic

[1] (French) National Computer Algebra Days

Geometry. After Diophantine equations, other typical examples come from the study of number fields: we hope deriving interesting information about a number field K by studying carefully all its p-adic incarnations $K \otimes_{\mathbb{Q}} \mathbb{Q}_p$. The ramification of K, its Galois properties, *etc.* can be — and are very often — studied in this manner [69, 65]. The class field theory, which provides a precise description of all Abelian extensions[2] of a given number field, is also formulated in this language [66]. The importance of p-adic numbers is so prominent today that there is still nowadays very active research on theories which are dedicated to purely p-adic objects: one can mention for instance the study of p-adic geometry and p-adic cohomologies [6, 58], the theory of p-adic differential equations [50], Coleman's theory of p-adic integration [24], the p-adic Hodge theory [14], the p-adic Langlands correspondence [5], the study of p-adic modular forms [34], p-adic ζ-functions [52] and L-functions [22], *etc.* The proof of Fermat's last Theorem by Wiles and Taylor [81, 78] is stamped with many of these ideas and developments.

Over the last decades, p-adic methods have taken some importance in Symbolic Computation as well. For a long time, p-adic methods have been used for factoring polynomials over $\mathbb{Q}$ [56]. More recently, there has been a wide diversification of the use of p-adic numbers for effective computations: Bostan et al. [13] used Newton sums for polynomials over $\mathbb{Z}_p$ to compute composed products for polynomials over $\mathbb{F}_p$; Gaudry et al. [32] used p-adic lifting methods to generate genus 2 CM hyperelliptic curves; Kedlaya [49], Lauder [54] and many followers used p-adic cohomology to count points on hyperelliptic curves over finite fields; Lercier and Sirvent [57] computed isogenies between elliptic curves over finite fields using p-adic differential equations.

The need to build solid foundations to the algorithmics of p-adic numbers has then emerged. This is however not straightforward because a single p-adic number encompasses an infinite amount of information (the infinite sequence of its digits) and then necessarily needs to be truncated in order to fit in the memory of a computer. From this point of view, p-adic numbers behave very similarly to real numbers and the questions that emerge when we are trying to implement p-adic numbers are often the same as the questions arising when dealing with rounding errors in the real setting [62, 26, 63]. The algorithmic study of p-adic numbers is then located at the frontier between Symbolic Computation and Numerical Analysis and imports ideas and results coming from both of these domains.

Content and organization of this course. This course focuses on the low-level implementation of p-adic numbers (and then voluntarily omits high-level algorithms making use of p-adic numbers) and pursues two main objectives. The first one is to introduce and discuss the most standard strategies for implementing p-adic numbers on computers. We shall detail three of them, each of them having its own spirit: (1) the *zealous arithmetic* which is inspired by interval arithmetic in the real setting, (2) the *lazy arithmetic* with its *relaxed* improvement and (3) the *p-adic floating-point arithmetic*, the last two being inspired by the eponym approaches in the real setting.

The second aim of this course is to develop a general theory giving quite powerful tools to study the propagation of accuracy in the p-adic world. The basic underlying idea is to linearize the situation (and then model the propagation of accuracy using differentials); it is once again inspired from classical methods in the real case. However, it turns out that the non-archimedean nature of $\mathbb{Q}_p$ (*i.e.* the fact that $\mathbb{Z}$ is bounded in $\mathbb{Q}_p$) is the source of many simplifications which will allow us to state much more accurate results and to go much further in the p-adic setting. As an example, we shall see that the theory of p-adic precision yields a general strategy for increasing the numerical stability of any given algorithm (assuming that the problem it solves is well-conditioned).

This course is organized as follows. §1 is devoted to the introduction of p-adic numbers: we define them, prove their main properties and discuss in more details their place in Number Theory, Arithmetic Geometry and Symbolic Computation. The presentation of the standard implementations of p-adic numbers mentioned above is achieved in §2. A careful comparison

[2] An abelian extension is a Galois extension whose Galois group is abelian.

between them is moreover proposed and supported by many examples coming from linear algebra and commutative algebra. Finally, in §3, we expose the aforementioned theory of p-adic precision. We then detail its applications: we will notably examine many very concrete situations and, for each of them, we will explain how the theory of p-adic precision helps us either in quantifying the qualities of a given algorithm regarding to numerical stability or, even better, in improving them.

Acknowledgments. This document contains the (augmented) notes of a lecture I gave at the "Journées Nationales du Calcul Formel[3]" (JNCF) on January 2017. I heartily thank the organizers and the scientific committee of the JNCF for giving me the opportunity to give these lectures and for encouraging me to write down these notes. I am very grateful to Delphine Boucher, Nicolas Brisebarre, Claude-Pierre Jeannerod, Marc Mezzarobba and Tristan Vaccon for their careful reading and their helpful comments on an earlier version of these notes.

Notation. We use standard notation for the set of numbers: $\mathbb{N}$ is the set of natural integers (*including* 0), $\mathbb{Z}$ is the set of relative integers, $\mathbb{Q}$ is the set of rational numbers and $\mathbb{R}$ is the set of real numbers. We will sometimes use the soft-O notation $\tilde{O}(-)$ for writing complexities; we recall that, given a sequence of positive real numbers (u_n), $\tilde{O}(u_n)$ is defined as the union of the sets $O(u_n \log^k u_n)$ for k varying in $\mathbb{N}$.

Throughout this course, the letter p always refers to a fixed prime number.

1 Introduction to p-adic numbers

In this first section, we define p-adic numbers, discuss their basic properties and try to explain, by selecting a few relevant examples, their place in Number Theory, Algebraic Geometry and Symbolic Computation. The presentation below is voluntarily very summarized; we refer the interested reader to [2, 35] for a more complete exposition of the theory of p-adic numbers.

1.1 Definition and first properties

p-adic numbers are very ambivalent objects which can be thought of under many different angles: computational, algebraic, analytic. It turns out that each point of view leads to its own definition of p-adic numbers: computer scientists often prefer viewing a p-adic number as a sequence of digits while algebraists prefer speaking of projective limits and analysts are more comfortable with Banach spaces and completions. Of course all these approaches have their own interest and understanding the intersections between them is often the key behind the most important advances.

In this subsection, we briefly survey all the standard definitions of p-adic numbers and provide several mental representations in order to try as much as possible to help the reader to develop a good p-adic intuition.

1.1.1 Down-to-earth definition

Recall that each positive integer n can be written in *base* p, that is as a finite sum:

$$n = a_0 + a_1 p + a_2 p^2 + \cdots + a_\ell p^\ell$$

where the a_i's are integers between 0 and $p-1$, the so-called *digits*. This writing is moreover unique assuming that the most significant digit a_ℓ does not vanish. A possible strategy to compute the expansion in base p goes as follows. We first compute a_0 by noting that it is necessarily the remainder in the Euclidean division of n by p: indeed it is congruent to n modulo p and lies in the range $[0, p-1]$ by definition. Once a_0 is known, we compute $n_1 = \frac{n-a_0}{p}$, which is also the

[3] (French) National Computer Algebra Days

$$
\begin{aligned}
a_0 :\quad 1742 &= 248 \times 7 + \mathbf{6} \\
a_1 :\quad 248 &= 35 \times 7 + \mathbf{3} \\
a_2 :\quad 35 &= 5 \times 7 + \mathbf{0} \\
a_3 :\quad 5 &= 0 \times 7 + \mathbf{5}
\end{aligned}
$$

$$
\Longrightarrow \quad 1742 = \overline{5036}^{7}
$$

Figure 1.1: Expansion of 1742 in base 7

$$
\begin{array}{rcccccccc}
 & \ldots & 2 & 3 & 0 & 6 & 2 & 4 & 4 \\
+ & \ldots & 1 & 6 & 5 & 2 & 3 & 3 & 2 \\
\hline
 & \ldots & \mathbf{4} & \mathbf{2} & \mathbf{6} & \mathbf{1} & \mathbf{6} & \mathbf{0} & \mathbf{6}
\end{array}
\qquad\qquad
\begin{array}{rcccccccc}
 & \ldots & 2 & 3 & 0 & 6 & 2 & 4 & 4 \\
\times & \ldots & 1 & 6 & 5 & 2 & 3 & 3 & 2 \\
\hline
 & \ldots & 4 & 6 & 1 & 5 & 5 & 2 & 1 \\
 & \ldots & 2 & 2 & 5 & 0 & 6 & 5 & \\
 & \ldots & 2 & 5 & 0 & 6 & 5 & & \\
 & \ldots & 5 & 5 & 2 & 1 & & & \\
 & \ldots & 6 & 1 & 6 & & & & \\
 & \ldots & 6 & 3 & & & & & \\
 & \ldots & 4 & & & & & & \\
\hline
 & \ldots & \mathbf{4} & \mathbf{3} & \mathbf{2} & \mathbf{0} & \mathbf{3} & \mathbf{0} & \mathbf{1}
\end{array}
$$

Figure 1.2: Addition and multiplication in $\mathbb{Z}_7$

quotient in the Euclidean division of n by p. Clearly $n_1 = a_1 + a_2 p + \cdots + a_\ell p^{\ell-1}$ and we can now compute a_1 repeating the same strategy. Figure 1.1 shows a simple execution of this algorithm.

By definition, a *p-adic integer* is an infinite formal sum of the shape:

$$
x = a_0 + a_1 p + a_2 p^2 + \cdots + a_i p^i + \cdots
$$

where the a_i's are integers between 0 and $p-1$. In other words, a p-adic integer is an integer written in base p with an infinite number of digits. We will sometimes alternatively write x as follows:

$$
x = \overline{\ldots a_i \ldots a_3 a_2 a_1 a_0}^{p}
$$

or simply

$$
x = \ldots a_i \ldots a_3 a_2 a_1 a_0
$$

when no confusion can arise. The set of p-adic integers is denoted by $\mathbb{Z}_p$. It is endowed with a natural structure of commutative ring. Indeed, we can add, subtract and multiply p-adic integers using the schoolbook method; note that handling carries is possible since they propagate on the left. The ring of natural integers $\mathbb{N}$ appears naturally as a subring of $\mathbb{Z}_p$: it consists of p-adic integers $\ldots a_i \ldots a_3 a_2 a_1 a_0$ for which $a_i = 0$ when i is large enough. Note in particular that the integer p writes $\ldots 0010$ in $\mathbb{Z}_p$ and more generally p^n writes $\ldots 0010 \ldots 0$ with n ending zeros. As a consequence, a p-adic integer is a multiple of p^n if and only if it ends with (at least) n zeros. Remark that negative integers are p-adic integers as well: the opposite of n is, by definition, the result of the subtraction $0-n$.

Similarly, we define a *p-adic number* as a formal infinite sum of the shape:

$$
x = a_{-n} p^{-n} + a_{-n+1} p^{-n+1} + \cdots + a_i p^i + \cdots
$$

where n is an integer which may depend on x. Alternatively, we will write:

$$
x = \overline{\ldots a_i \ldots a_2 a_1 a_0 \,.\, a_{-1} a_{-2} \ldots a_{-n}}^{p}
$$

and, when no confusion may arise, we will freely remove the bar and the trailing p. A p-adic number is then nothing but a "decimal" number written in base p with an infinite number of

digits before the decimal mark and a finite amount of digits after the decimal mark. Addition and multiplication extend to p-adic numbers as well.

The set of p-adic numbers is denoted by $\mathbb{Q}_p$. Clearly $\mathbb{Q}_p = \mathbb{Z}_p[\frac{1}{p}]$. We shall see later (*cf* Proposition 1.1, page 5) that $\mathbb{Q}_p$ is actually the fraction field of $\mathbb{Z}_p$; in particular it is a field and $\mathbb{Q}$, which is the fraction field of $\mathbb{Z}$, naturally embeds into $\mathbb{Q}_p$.

1.1.2 Second definition: projective limits

From the point of view of addition and multiplication, the last digit of a p-adic integer behaves like an integer modulo p, that is an element of the finite field $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$. In other words, the application $\pi_1 : \mathbb{Z}_p \to \mathbb{Z}/p\mathbb{Z}$ taking a p-adic integer $x = a_0 + a_1 p + a_2 p^2 + \cdots$ to the class of a_0 modulo p is a ring homomorphism. More generally, given a positive integer n, the map:

$$\begin{array}{rccc} \pi_n : & \mathbb{Z}_p & \to & \mathbb{Z}/p^n\mathbb{Z} \\ a_0 + a_1 p + a_2 p^2 + \cdots & & \mapsto & (a_0 + a_1 p + \cdots + a_{n-1}p^{n-1}) \bmod p^n \end{array}$$

is a ring homomorphism. These morphisms are compatible in the following sense: for all $x \in \mathbb{Z}_p$, we have $\pi_{n+1}(x) \equiv \pi_n(x) \pmod{p^n}$ (and more generally $\pi_m(x) \equiv \pi_n(x) \pmod{p^n}$ provided that $m \geqslant n$). Putting the π_n's all together, we end up with a ring homomorphism:

$$\begin{array}{rccc} \pi : & \mathbb{Z}_p & \to & \varprojlim_n \mathbb{Z}/p^n\mathbb{Z} \\ & x & \mapsto & (\pi_1(x), \pi_2(x), \ldots) \end{array}$$

where $\varprojlim_n \mathbb{Z}/p^n\mathbb{Z}$ is by definition the subring of $\prod_{n=1}^{\infty} \mathbb{Z}/p^n\mathbb{Z}$ consisting of sequences $(x_1, x_2, \ldots)$ for which $x_{n+1} \equiv x_n \pmod{p^n}$ for all n: it is called the *projective limit* of the $\mathbb{Z}/p^n\mathbb{Z}$'s.

Conversely, consider a sequence $(x_1, x_2, \ldots) \in \varprojlim_n \mathbb{Z}/p^n\mathbb{Z}$. In a slight abuse of notation, continue to write x_n for the unique integer of the range $[\![0, p^n - 1]\!]$ which is congruent to x_n modulo p^n and write it in base p:

$$x_n = a_{n,0} + a_{n,1} p + \cdots + a_{n,n-1} p^{n-1}$$

(the expansion stops at $(n-1)$ since $x_n < p^n$ by construction). The condition $x_{n+1} \equiv x_n \pmod{p^n}$ implies that $a_{n+1,i} = a_{n,i}$ for all $i \in [\![0, n-1]\!]$. In other words, when i remains fixed, the sequence $(a_{n,i})_{n>i}$ is constant and thus converges to some a_i. Set:

$$\psi(x_1, x_2, \ldots) = \ldots a_i \ldots a_2 a_1 a_0 \in \mathbb{Z}_p.$$

We define this way an application $\psi : \varprojlim_n \mathbb{Z}/p^n\mathbb{Z} \to \mathbb{Z}_p$ which is by construction a left and a right inverse of π. In other words, π and ψ are isomorphisms which are inverses of each other.

The above discussion allows us to give an alternative definition of $\mathbb{Z}_p$, which is:

$$\mathbb{Z}_p = \varprojlim_n \mathbb{Z}/p^n\mathbb{Z}.$$

The map π_n then corresponds to the projection onto the n-th factor. This definition is more abstract and it seems more difficult to handle as well. However it has the enormous advantage of making the ring structure appear clearly and, for this reason, it is often much more useful and powerful than the down-to-earth definition of §1.1.1. As a typical example, let us prove the following proposition.

Proposition 1.1. *(a) An element $x \in \mathbb{Z}_p$ is invertible in $\mathbb{Z}_p$ if and only if $\pi_1(x)$ does not vanish.*

(b) The ring $\mathbb{Q}_p$ is the fraction field of $\mathbb{Z}_p$; in particular, it is a field.

Proof. *(a)* Let $x \in \mathbb{Z}_p$. Viewing $\mathbb{Z}_p$ as $\varprojlim_n \mathbb{Z}/p^n\mathbb{Z}$, we find that x is invertible in $\mathbb{Z}_p$ if and only if $\pi_n(x)$ is invertible in $\mathbb{Z}/p^n\mathbb{Z}$ for all n. The latest condition is equivalent to requiring that $\pi_n(x)$ and p^n are coprime for all n. Noting that p is prime, this is further equivalent to the fact that $\pi_n(x) \bmod p = \pi_1(x)$ does not vanish in $\mathbb{Z}/p\mathbb{Z}$.

(b) By definition $\mathbb{Q}_p = \mathbb{Z}_p[\frac{1}{p}]$. It is then enough to prove that any nonzero p-adic integer x can be written as a product $x = p^n u$ where n is a nonnegative integer and u is a unit in $\mathbb{Z}_p$. Let n be the number of zeros at the end of the p-adic expansion of x (or, equivalently, the largest integer n such that $\pi_n(x) = 0$). Then x can be written $p^n u$ where u is a p-adic integer whose last digit does not vanish. By the first part of the proposition, u is then invertible in $\mathbb{Z}_p$ and we are done. □

We note that the first statement of Proposition 1.1 shows that the subset of non-invertible elements of $\mathbb{Z}_p$ is exactly the kernel of π_1. We deduce from this that $\mathbb{Z}_p$ is a local ring with maximal ideal $\ker \pi_1$.

1.1.3 Valuation and norm

We define the *p-adic valuation* of the nonzero p-adic number

$$x = \ldots a_i \ldots a_2 a_1 a_0 \,.\, a_{-1} a_{-2} \ldots a_{-n}$$

as the smallest (possibly negative) integer v for which a_v does not vanish. We denote it $\mathrm{val}_p(x)$ or simply $\mathrm{val}(x)$ if no confusion may arise. Alternatively $\mathrm{val}(x)$ can be defined as the largest integer v such that $x \in p^v\mathbb{Z}_p$. When $x = 0$, we put $\mathrm{val}(0) = +\infty$. We define this way a function $\mathrm{val} : \mathbb{Q}_p \to \mathbb{Z} \cup \{+\infty\}$. Writing down the computations (and remembering that p is prime), we immediately check the following compatibility properties for all $x, y \in \mathbb{Q}_p$:

(1) $\mathrm{val}(x+y) \geqslant \min\big(\mathrm{val}(x), \mathrm{val}(y)\big)$,

(2) $\mathrm{val}(xy) = \mathrm{val}(x) + \mathrm{val}(y)$.

Note moreover that the equality $\mathrm{val}(x+y) = \min\big(\mathrm{val}(x), \mathrm{val}(y)\big)$ does hold as soon as $\mathrm{val}(x) \neq \mathrm{val}(y)$. As we shall see later, this property reflects the tree structure of $\mathbb{Z}_p$ (see §1.1.5).

The *p-adic norm* $|\cdot|_p$ is defined by $|x|_p = p^{-\mathrm{val}(x)}$ for $x \in \mathbb{Q}_p$. In the sequel, when no confusion can arise, we shall often write $|\cdot|$ instead of $|\cdot|_p$. The properties (1) and (2) above immediately translate as follows:

(1') $|x+y| \leqslant \max\big(|x|, |y|\big)$ and equality holds if $|x| \neq |y|$,

(2') $|xy| = |x| \cdot |y|$.

Remark that (1') implies that $|\cdot|$ satisfies the triangular inequality, that is $|x+y| \leqslant |x| + |y|$ for all $x, y \in \mathbb{Q}_p$. It is however much stronger: we say that the p-adic norm is *ultrametric* or *non Archimedean*. We will see later that ultrametricity has strong consequences on the topology of $\mathbb{Q}_p$ (see for example Corollary 1.3 below) and strongly influences the calculus with p-adic (univariate and multivariate) functions as well (see §3.1.4). This is far from being anecdotic; on the contrary, this will be the starting point of the theory of p-adic precision we will develop in §3.

The p-adic norm defines a natural distance d on $\mathbb{Q}_p$ as follows: we agree that the distance between two p-adic numbers x and y is $|x-y|_p$. Again this distance is ultrametric in the sense that:

$$d(x,z) \leqslant \max\big(d(x,y), d(y,z)\big).$$

Moreover the equality holds as soon as $d(x,y) \neq d(y,z)$: all triangles in $\mathbb{Q}_p$ are isosceles! Observe also that d takes its values in a proper subset of $\mathbb{R}^+$ (namely $\{0\} \cup \{p^n : n \in \mathbb{Z}\}$) whose unique accumulation point is 0. This property has surprising consequences; for example, closed balls of positive radius are also open balls and *vice et versa*. In particular $\mathbb{Z}_p$ is open (in $\mathbb{Q}_p$) and compact according to the topology defined by the distance. From now on, we endow $\mathbb{Q}_p$ with this topology.

Clearly, a p-adic number lies in $\mathbb{Z}_p$ if and only if its p-adic valuation is nonnegative, that is if and only if its p-adic norm is at most 1. In other words, $\mathbb{Z}_p$ appears as the closed unit ball in $\mathbb{Q}_p$. Viewed this way, it is remarkable that it is stable under addition (compare with $\mathbb{R}$); it is however a direct consequence of the ultrametricity. Similarly, by Proposition 1.1, a p-adic integer is invertible in $\mathbb{Z}_p$ if and only if it has norm 1, meaning that the group of units of $\mathbb{Z}_p$ is then the unit sphere in $\mathbb{Q}_p$. As for the maximal ideal of $\mathbb{Z}_p$, it consists of elements of positive valuation and then appears as the open unit ball in $\mathbb{Q}_p$ (which is also the closed ball of radius p^{-1}).

1.1.4 Completeness

The following important proposition shows that $\mathbb{Q}_p$ is nothing but the completion of $\mathbb{Q}$ according to the p-adic distance. In that sense, $\mathbb{Q}_p$ arises in a very natural way... just as does $\mathbb{R}$.

Proposition 1.2. *The space $\mathbb{Q}_p$ equipped with its natural distance is complete (in the sense that every Cauchy sequence converges). Moreover $\mathbb{Q}$ is dense in $\mathbb{Q}_p$.*

Proof. We first prove that $\mathbb{Q}_p$ is complete. Let $(u_n)_{n\geqslant 0}$ be a $\mathbb{Q}_p$-valued Cauchy sequence. It is then bounded and rescaling the u_n's by a uniform scalar, we may assume that $|u_n| \leqslant 1$ (*i.e.* $u_n \in \mathbb{Z}_p$) for all n. For each n, write :

$$u_n = \sum_{i=0}^{\infty} a_{n,i} p^i$$

with $a_{n,i} \in \{0, 1, \ldots, p-1\}$. Fix an integer i_0 and set $\varepsilon = p^{-i_0}$. Since (u_n) is a Cauchy sequence, there exists a rank N with the property that $|u_n - u_m| \leqslant \varepsilon$ for all $n, m \geqslant N$. Coming back to the definition of the p-adic norm, we find that $u_n - u_m$ is divisible by p^{i_0}. Writing $u_n = u_m + (u_n - u_m)$ and computing the sum, we get $a_{n,i} = a_{m,i}$ for all $i \leqslant i_0$. In particular the sequence $(a_{n,i_0})_{n\geqslant 0}$ is ultimately constant. Let $a_{i_0} \in \{0, 1, \ldots, p-1\}$ denote its limit. Now define $\ell = \sum_{i=0}^{\infty} a_i p^i \in \mathbb{Z}_p$ and consider again $\varepsilon > 0$. Let i_0 be an integer such that $p^{-i_0} \leqslant \varepsilon$. By construction, there exists a rank N for which $a_{n,i} = a_i$ whenever $n \geqslant N$ and $i \leqslant i_0$. For $n \geqslant N$, the difference $u_n - \ell$ is then divisible by p^{i_0} and hence has norm at most ε. Hence (u_n) converges to ℓ.

We now prove that $\mathbb{Q}$ is dense in $\mathbb{Q}_p$. Since $\mathbb{Q}_p = \mathbb{Z}_p[\frac{1}{p}]$, it is enough to prove that $\mathbb{Z}$ is dense in $\mathbb{Z}_p$. Pick $a \in \mathbb{Z}_p$ and write $a = \sum_{i\geqslant 0} a_i p^i$. For a nonnegative integer n, set $b_n = \sum_{i=0}^{n-1} a_i p^i$. Clearly b_n is an integer and the sequence $(b_n)_{n\geqslant 0}$ converges to a. The density follows. □

Corollary 1.3. *Let $(u_n)_{n\geqslant 0}$ be a sequence of p-adic numbers. The series $\sum_{n\geqslant 0} u_n$ converges in $\mathbb{Q}_p$ if and only if its general term u_n converges to 0.*

Proof. Set $s_n = \sum_{i=0}^{n-1} u_i$. Clearly $u_n = s_{n+1} - s_n$ for all n. If (s_n) converges to a limit $s \in \mathbb{Q}_p$, then u_n converges to $s - s = 0$. We now assume that (u_n) goes to 0. We claim that (s_n) is a Cauchy sequence (and therefore converges). Indeed, let $\varepsilon > 0$ and pick an integer N for which $|u_i| \leqslant \varepsilon$ for all $i \geqslant N$. Given two integers m and n with $m > n \geqslant N$, we have:

$$|s_m - s_n| = \left| \sum_{i=n}^{m-1} u_i \right| \leqslant \max\big(|u_n|, |u_{n+1}|, \ldots, |u_{m-1}|\big)$$

thanks to ultrametricity. Therefore $|s_m - s_n| \leqslant \varepsilon$ and we are done. □

1.1.5 Tree representation

Geometrically, it is often convenient and meaningful to represent $\mathbb{Z}_p$ as the infinite full p-ary tree. In order to explain this representation, we need a definition.

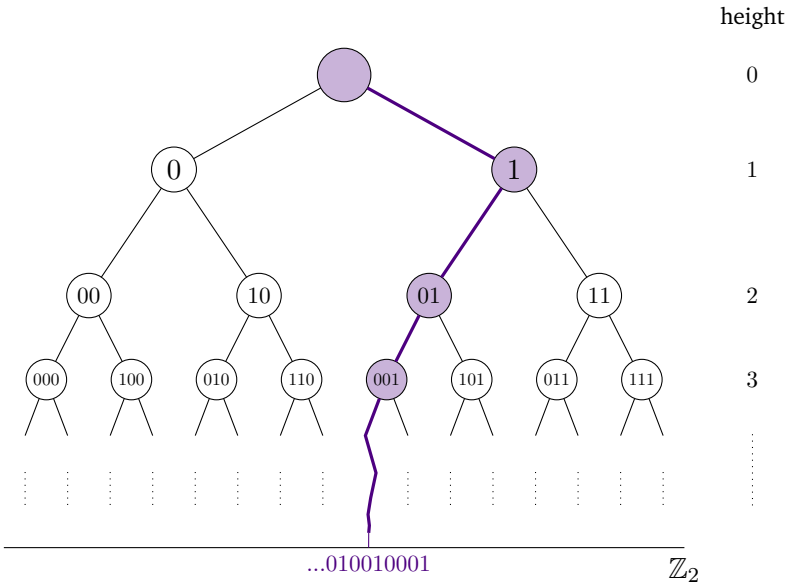

Figure 1.3: Tree representation of $\mathbb{Z}_2$

Definition 1.4. For $h \in \mathbb{N}$ and $a \in \mathbb{Z}_p$, we set:

$$I_{h,a} = \left\{ x \in \mathbb{Z}_p \quad \text{s.t.} \quad x \equiv a \pmod{p^h} \right\}.$$

An *interval* of $\mathbb{Z}_p$ is a subset of $\mathbb{Z}_p$ of the form $I_{h,a}$ for some h and a.

If a decomposes in base p as $a = a_0 + a_1 p + a_2 p^2 + \cdots + a_{h-1} p^{h-1} + \cdots$, the interval $I_{h,a}$ consists exactly of the p-adic integers whose last digits are $a_{h-1} \ldots a_1 a_0$ in this order. On the other hand, from the analytic point of view, the condition $x \equiv a \pmod{p^h}$ is equivalent to $|x-a| \leqslant p^{-h}$. Thus the interval $I_{h,a}$ is nothing but the closed ball of centre a and radius p^{-h}. Even better, the intervals of $\mathbb{Z}_p$ are exactly the closed balls of $\mathbb{Z}_p$.

Clearly $I_{h,a} = I_{h,a'}$ if and only if $a \equiv a' \pmod{p^h}$. In particular, given an interval I of $\mathbb{Z}_p$, there is exactly one integer h such that $I = I_{h,a}$. We will denote it by $h(I)$ and call it the *height* of I. We note that there exist exactly p^h intervals of $\mathbb{Z}_p$ of height h since these intervals are indexed by the classes modulo p^h (or equivalently by the sequences of h digits between 0 and $p-1$).

From the topological point of view, intervals behave like $\mathbb{Z}_p$: they are at the same time open and compact.

We now define the *tree* of $\mathbb{Z}_p$, denoted by $\mathcal{T}(\mathbb{Z}_p)$, as follows: its vertices are the intervals of $\mathbb{Z}_p$ and we put an edge $I \to J$ whenever $h(J) = h(I) + 1$ and $J \subset I$. A picture of $\mathcal{T}(\mathbb{Z}_2)$ is represented on Figure 1.3. The labels indicated on the vertices are the last h digits of a. Coming back to a general p, we observe that the height of an interval I corresponds to the usual height function in the tree $\mathcal{T}(\mathbb{Z}_p)$. Moreover, given two intervals I and J, the inclusion $J \subset I$ holds if and only if there exists a path from I to J.

Elements of $\mathbb{Z}_p$ bijectively correspond to infinite paths of $\mathcal{T}(\mathbb{Z}_p)$ starting from the root through the following correspondence: an element $x \in \mathbb{Z}_p$ is encoded by the path

$$I_{0,x} \to I_{1,x} \to I_{2,x} \to \cdots \to I_{h,x} \to \cdots .$$

Under this encoding, an infinite path of $\mathcal{T}(\mathbb{Z}_p)$ starting from the root

$$I_0 \to I_1 \to I_2 \to \cdots \to I_h \to \cdots$$

corresponds to a uniquely determined p-adic integer, which is the unique element lying in the decreasing intersection $\bigcap_{h \in \mathbb{N}} I_h$. Concretely each new I_h determines a new digit of x; the whole

collection of the I_h's then defines x entirely. The distance on $\mathbb{Z}_p$ can be visualized on $\mathcal{T}(\mathbb{Z}_p)$ as well: given $x, y \in \mathbb{Z}_p$, we have $|x-y| = p^{-h}$ where h is the height where the paths attached to x and y separate.

The above construction easily extends to $\mathbb{Q}_p$.

Definition 1.5. For $h \in \mathbb{Z}$ and $a \in \mathbb{Q}_p$, we set:

$$I_{h,a} = \{\, x \in \mathbb{Q}_p \quad \text{s.t.} \quad |x-a| \leqslant p^{-h} \,\}.$$

A *bounded interval* of $\mathbb{Q}_p$ is a subset of $\mathbb{Q}_p$ of the form $I_{h,a}$ for some h and a.

Similarly to the case of $\mathbb{Z}_p$, a bounded interval of $\mathbb{Q}_p$ of height h is a subset of $\mathbb{Q}_p$ consisting of p-adic numbers whose digits at the positions $< h$ are fixed (they have to agree with the digits of a at the same positions).

The graph $\mathcal{T}(\mathbb{Q}_p)$ is defined as follows: its vertices are the intervals of $\mathbb{Q}_p$ while there is an edge $I \to J$ if $h(J) = h(I) + 1$ and $J \subset I$. We draw the attention of the reader to the fact that $\mathcal{T}(\mathbb{Q}_p)$ is a tree but it is not rooted: there does not exist a largest bounded interval in $\mathbb{Q}_p$. To understand better the structure of $\mathcal{T}(\mathbb{Q}_p)$, let us define, for any integer v, the subgraph $\mathcal{T}(p^{-v}\mathbb{Z}_p)$ of $\mathcal{T}(\mathbb{Q}_p)$ consisting of intervals which are contained in $p^{-v}\mathbb{Z}_p$. From the fact that $\mathbb{Q}_p$ is the union of all $p^{-v}\mathbb{Z}_p$, we derive that $\mathcal{T}(\mathbb{Q}_p) = \bigcup_{v \geqslant 0} \mathcal{T}(p^{-v}\mathbb{Z}_p)$. Moreover, for all v, $\mathcal{T}(p^{-v}\mathbb{Z}_p)$ is a rooted tree (with root $p^{-v}\mathbb{Z}_p$) which is isomorphic to $\mathcal{T}(\mathbb{Z}_p)$ except that the height function is shifted by $-v$. The tree $\mathcal{T}(p^{-v-1}\mathbb{Z}_p)$ is thus obtained by juxtaposing p copies of $\mathcal{T}(p^{-v}\mathbb{Z}_p)$ and linking the roots of them to a common parent $p^{-v}\mathbb{Z}_p$ (which then becomes the new root).

1.2 Newton iteration over the p-adic numbers

Newton iteration is a well-known tool in Numerical Analysis for approximating a zero of a "nice" function defined on a real interval. More precisely, given a differentiable function $f : [a,b] \to \mathbb{R}$, we define a recursive sequence $(x_i)_{i \geqslant 0}$ by:

$$x_0 \in [a,b] \quad ; \quad x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)},\ i = 0, 1, 2, \ldots \tag{1.1}$$

Under some assumptions, one can prove that the sequence (x_i) converges to a zero of f, namely x_∞. Moreover the convergence is very rapid since, assuming that f is twice differentiable, we usually have an inequality of the shape $|x_\infty - x_i| \leqslant \rho^{2^i}$ for some $\rho \in (0,1)$. In other words, the number of correct digits roughly doubles at each iteration. The Newton recurrence (1.1) has a nice geometrical interpretation as well: the value x_{i+1} is the x-coordinate of the intersection point of the x-axis with the tangent to the curve $y = f(x)$ at the point x_i (see Figure 1.4).

1.2.1 Hensel's Lemma

It is quite remarkable that the above discussion extends almost *verbatim* when $\mathbb{R}$ is replaced by $\mathbb{Q}_p$. Actually, extending the notion of differentiability to p-adic functions is quite subtle and probably the most difficult part. This will be achieved in §3.1.3 (for functions of class C^1) and §3.2.3 (for functions of class C^2). For now, we prefer avoiding these technicalities and restricting ourselves to the simpler (but still interesting) case of polynomials. For this particular case, the Newton iteration is known as *Hensel's Lemma* and already appears in Hensel's seminal paper [36] in which p-adic numbers are introduced.

Let $f(X) = a_0 + a_1 X + \cdots + a_n X^n$ be a polynomial in the variable X with coefficients in $\mathbb{Q}_p$. Recall that the derivative of f can be defined in a purely algebraic way as $f'(X) = a_1 + 2a_2 X + \cdots + n a_n X^{n-1}$.

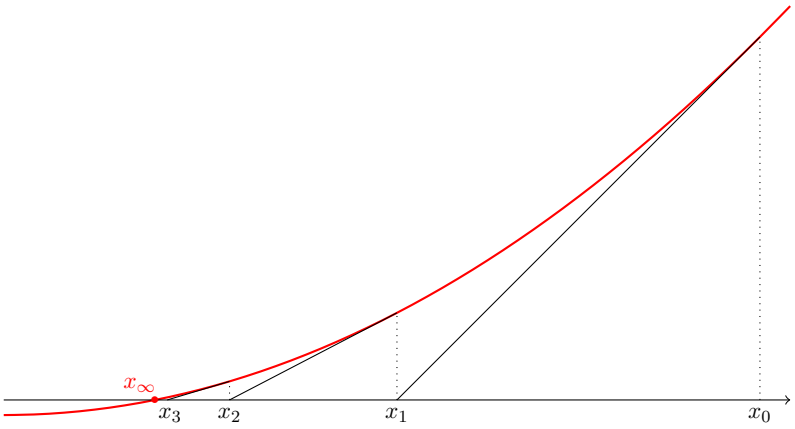

Figure 1.4: Newton iteration over the reals

Theorem 1.6 (Hensel's Lemma)**.** *Let $f \in \mathbb{Z}_p[X]$ be a polynomial with coefficients in $\mathbb{Z}_p$. We suppose that we are given some $a \in \mathbb{Z}_p$ with the property that $|f(a)| < |f'(a)|^2$. Then the sequence $(x_i)_{i\geqslant 0}$ defined by the recurrence:*

$$x_0 = a \quad ; \quad x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

is well defined and converges to $x_\infty \in \mathbb{Z}_p$ with $f(x_\infty) = 0$. The rate of convergence is given by:

$$|x_\infty - x_i| \leqslant |f'(a)| \cdot \left(\frac{|f(a)|}{|f'(a)|^2}\right)^{2^i}.$$

Moreover x_∞ is the unique root of f in the open ball of centre a and radius $|f'(a)|$.

The proof of the above theorem is based on the next lemma:

Lemma 1.7. *Given $f \in \mathbb{Z}_p[X]$ and $x, h \in \mathbb{Z}_p$, we have:*

(i) $|f(x+h) - f(x)| \leqslant |h|$.

(ii) $|f(x+h) - f(x) - hf'(x)| \leqslant |h|^2$.

Proof. For any nonnegative integer i, define $f^{[i]} = \frac{1}{i!} f^{(i)}$ where $f^{(i)}$ stands for the i-th derivative of f. Taylor's formula then reads:

$$f(x+h) - f(x) = hf'(x) + h^2 f^{[2]}(x) + \cdots + h^n f^{[n]}(x). \tag{1.2}$$

Moreover, a direct computation shows that the coefficients of $f^{[i]}$ are obtained from that of f by multiplying by binomial coefficients. Therefore $f^{[i]}$ has coefficients in $\mathbb{Z}_p$. Hence $f^{[i]}(x) \in \mathbb{Z}_p$, *i.e.* $|f^{[i]}(x)| \leqslant 1$, for all i. We deduce that each summand of the right hand side of (1.2) has norm at most $|h|$. The first assertion follows while the second is proved similarly. □

Proof of Theorem 1.6. Define $\rho = \frac{|f(a)|}{|f'(a)|^2}$. We first prove by induction on i the following conjunction:

$$(H_i): \quad |f'(x_i)| = |f'(a)| \quad \text{and} \quad |f(x_i)| \leqslant |f'(a)|^2 \cdot \rho^{2^i}.$$

Clearly (H_0) holds. We assume now that (H_i) holds for some $i \geqslant 0$. We put $h = -\frac{f(x_i)}{f'(x_i)}$ so that $x_{i+1} = x_i + h$. We write $f'(x_{i+1}) = \big(f'(x_{i+1}) - f'(x_i)\big) + f'(x_i)$. Observe that the first summand $f'(x_{i+1}) - f'(x_i)$ has norm at most $|h|$ by the first assertion of Lemma 1.7, while $|f'(x_i)| \geqslant |h| \cdot \rho^{-2^i} > |h|$ by the induction hypothesis. The norm of $f'(x_{i+1})$ is then the maximum of the norms of the two summands, which is $|f'(x_i)| = |f'(a)|$. Now, applying again Lemma 1.7, we get $|f(x_{i+1})| \leqslant |h|^2 \leqslant |f'(a)|^2 \cdot \rho^{2^i}$ and the induction goes.

Coming back to the recurrence defining the x_i's, we get:

$$|x_{i+1} - x_i| = \frac{|f(x_i)|}{|f'(x_i)|} \leqslant |f'(a)| \cdot \rho^{2^i}. \tag{1.3}$$

By Corollary 1.3, this implies the convergence of the sequence $(x_i)_{i\geqslant 0}$. Its limit x_∞ is a solution to the equation $x_\infty = x_\infty + \frac{f(x_\infty)}{f'(x_\infty)}$. Thus $f(x_\infty)$ has to vanish. The announced rate of convergence follows from Eq. (1.3) thanks to ultrametricity.

It remains to prove uniqueness. For this, consider $y \in \mathbb{Z}_p$ with $f(y) = 0$ and $|y - x_0| < |f'(a)|$. Since $|x_\infty - x_0| \leqslant |f'(a)| \cdot \rho < |f'(a)|$, we deduce $|x_\infty - y| < |f'(a)|$ as well. Applying Lemma 1.7 with $x = x_\infty$ and $h = y - x_\infty$, we find $|hf'(x_\infty)| \leqslant |h|^2$. Since $|h| < |f'(a)| = |f'(x_\infty)|$, this implies $|h| = 0$, *i.e.* $x = x_\infty$. Uniqueness is proved. □

Remark 1.8. All conclusions of Theorem 1.6 are still valid for any sequence (x_i) satisfying the weaker assumption:

$$\left| x_{i+1} - x_i + \frac{f(x_i)}{f'(x_i)} \right| \leqslant |f'(a)| \cdot \rho^{2^{i+1}}$$

(the proof is entirely similar). Roughly speaking, this stronger version allows us to work with approximations at *each* iteration. It will play a quite important role for algorithmic purpose (notably in §2.1.3) since computers cannot handle exact p-adic numbers but always need to work with truncations.

1.2.2 Computation of the inverse

A classical application of Newton iteration is the computation of the inverse: for computing the inverse of a real number c, we introduce the function $x \mapsto \frac{1}{x} - c$ and the associated Newton scheme. This leads to the recurrence $x_{i+1} = 2x_i - cx_i^2$ with initial value $x_0 = a$ where a is sufficiently close to $\frac{1}{c}$.

In the p-adic setting, the same strategy applies (although it does not appear as a direct consequence of Hensel's Lemma since the mapping $x \mapsto \frac{1}{x} - c$ is *not* polynomial). Anyway, let us pick an invertible element $c \in \mathbb{Z}_p$, *i.e.* $|c| = 1$, and define the sequence $(x_i)_{i\geqslant 0}$ by:

$$x_0 = a \quad ; \quad x_{i+1} = 2x_i - cx_i^2,\ i = 0, 1, 2, \ldots$$

where a is any p-adic number whose last digit is the inverse modulo p of the last digit of c. Computing such an element a reduces to computing a modular inverse and thus is efficiently feasible.

Proposition 1.9. *The sequence $(x_i)_{i\geqslant 0}$ defined above converges to $\frac{1}{c} \in \mathbb{Z}_p$. More precisely, we have $|cx_i - 1| \leqslant p^{-2^i}$ for all i.*

Proof. We prove the last statement of the proposition by induction on i. By construction of a, it holds for $i = 0$. Now observe that $cx_{i+1} - 1 = 2cx_i - c^2x_i^2 - 1 = -(cx_i - 1)^2$. Taking norms on both sides, we get $|cx_{i+1} - 1| = |cx_i - 1|^2$ and the induction goes this way. □

Quite interestingly, this method extends readily to the non-commutative case. Let us illustrate this by showing how it can be used to compute inverses of matrices with coefficients in $\mathbb{Z}_p$. Assume then that we are starting with a matrix $C \in M_n(\mathbb{Z}_p)$ whose reduction modulo p is invertible, *i.e.* there exists a matrix A such that $AC \equiv I_n \pmod p$ where I_n is the identity matrix of size n. Note that the computation of A reduces to the inversion of an $n \times n$ matrix over the finite field $\mathbb{F}_p$ and so can be done efficiently. We now define the sequence $(X_i)_{i\geqslant 0}$ by the recurrence:

$$X_0 = A \quad ; \quad X_{i+1} = 2X_i - X_iCX_i$$

(be careful with the order in the last term). Mimicking the proof of Proposition 1.9, we write:

$$CX_{i+1} - I_n = 2CX_i - CX_iCX_i - I_n = -(CX_i - I_n)^2$$

and obtain this way that each entry of $CX_i - 1$ has norm at most p^{2^i}. Therefore X_i converges to a matrix X_∞ satisfying $CX_\infty = I_n$, *i.e.* $X_\infty = C^{-1}$ (which in particular implies that C is invertible). A similar argument works for p-adic skew polynomials and p-adic differential operators as well.

1.2.3 Square roots in $\mathbb{Q}_p$

Another important application of Newton iteration is the computation of square roots. Again, the classical scheme over $\mathbb{R}$ applies without (substantial) modification over $\mathbb{Q}_p$.

Let c be a p-adic number. If the p-adic valuation of c is odd, then c is clearly not a square in $\mathbb{Q}_p$ and it does not make sense to compute its square root. On the contrary, if the p-adic valuation of c is even, then we can write $c = p^{2v}c'$ where v is an integer and c' is a unit in $\mathbb{Z}_p$. Computing a square root then reduces to computing a square root of c'; in other words, we may assume that c is invertible in $\mathbb{Z}_p$, *i.e.* $|c| = 1$.

We now introduce the polynomial function $f(x) = x^2 - c$. In order to apply Hensel's Lemma (Theorem 1.6), we need a first rough approximation a of $\sqrt{c}$. Precisely, we need to find some $a \in \mathbb{Z}_p$ with the property that $|f(a)| < |f'(a)|^2 = |2a|^2$. From $|c| = 1$, the above inequality implies $|a| = 1$ as well and therefore can be rewritten as $a^2 \equiv c \pmod{q}$ where $q = 8$ if $p = 2$ and $q = p$ otherwise. We then first need to compute a square root of c modulo q. If $p = 2$, this can be achieved simply by looking at the table of squares modulo 8:

a	0	1	2	3	4	5	6	7
a^2	0	1	4	1	0	1	4	1

Observe moreover that c is necessarily odd (since it is assumed to be invertible in $\mathbb{Z}_2$). If it is congruent to 1 modulo 8, $a = 1$; otherwise, there is no solution and c has no square root in $\mathbb{Q}_2$. When $p > 2$, we have to compute a square root in the finite field $\mathbb{F}_p$ for which efficient algorithms are known [23, §1.5]. If c mod p is not a square in $\mathbb{F}_p$, then c does not admit a square root in $\mathbb{Q}_p$ either.

Once we have computed the initial value a, we consider the recursive sequence $(x_i)_{i\geqslant 0}$ defined by $x_0 = a$ and

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} = \frac{1}{2}\left(x_i + \frac{c}{x_i}\right),\ i = 0, 1, 2, \ldots$$

By Hensel's Lemma, it converges to a limit x_∞ which is a square root of c. Moreover the rate of convergence is given by:

$$\begin{aligned} |x_\infty - x_i| &\leqslant p^{-2^i} && \text{if } p > 2 \\ |x_\infty - x_i| &\leqslant 2^{-(2^i+1)} && \text{if } p = 2 \end{aligned}$$

meaning that the number of correct digits of x_i is at least 2^i (resp. $2^i + 1$) when $p > 2$ (resp. $p = 2$).

1.3 Similarities with formal and Laurent series

According to the very first definition of $\mathbb{Z}_p$ we have given (see §1.1.1), p-adic integers have formal similitudes with formal series over $\mathbb{F}_p$: they are both described as infinite series with coefficients between 0 and $p-1$, the main difference being that additions and multiplications involve carries in the p-adic case while they do not for formal series. From a more abstract point of view, the parallel between $\mathbb{Z}_p$ and $\mathbb{F}_p[\![t]\!]$ is also apparent. For example $\mathbb{F}_p[\![t]\!]$ is also endowed with a valuation; this valuation defines a distance on $\mathbb{F}_p[\![t]\!]$ for which $\mathbb{F}_p[\![t]\!]$ is complete and Hensel's Lemma (together with Newton iteration) extends *verbatim* to formal series. In the same fashion,

char. 0		char. p
$\mathbb{Z}$	$\longleftrightarrow$	$\mathbb{F}_p[t]$
$\mathbb{Q}$	$\longleftrightarrow$	$\mathbb{F}_p(t)$
$\mathbb{Z}_p$	$\longleftrightarrow$	$\mathbb{F}_p[\![t]\!]$
$\mathbb{Q}_p$	$\longleftrightarrow$	$\mathbb{F}_p((t))$

Figure 1.5: Similarities between characteristic 0 and characteristic p

the analogue of $\mathbb{Q}_p$ is the field of Laurent series $\mathbb{F}_p((t))$. Noticing further that $\mathbb{F}_p[t]$ is dense in $\mathbb{F}_p[\![t]\!]$ and similarly that $\mathbb{F}_p(t)$ is dense in $\mathbb{F}_p((t))$ [4], we can even supplement the correspondence between the characteristic 0 and the characteristic p, ending up with the "dictionary" of Figure 1.5.

Algebraists have actually managed to capture the essence of there resemblances in the notion of *discrete valuation rings/fields* and *completion* of those [74]. Recently, Scholze defined the notion of *perfectoid* which allows us (under several additional assumptions) to build an actual bridge between the characteristic 0 and the characteristic p. It is not the purpose of this lecture to go further in this direction; we nevertheless refer interested people to [71, 72, 9] for the exposition of the theory.

1.3.1 The point of view of algebraic geometry

Algebraic Geometry has an original and interesting point of view on the various rings and fields introduced before, showing in particular how p-adic numbers can arise in many natural problems in arithmetics. The underlying ubiquitous idea in algebraic geometry is to associate to any ring A a geometrical space Spec A — its so-called *spectrum* — on which the functions are the elements of A. We will not detail here the general construction of Spec A but just try to explain informally what it looks like when A is one of the rings appearing in Figure 1.5.

Let us start with $k[t]$ where k is a field (of any characteristic). For the simplicity of the exposition, let us assume further that k is algebraically closed. One thinks of elements of $k[t]$ as (polynomial) functions over k, meaning that this spectrum should be thought of as k itself. Spec $k[t]$ is called the affine line over k and is usually drawn as a straight line. The spectrum of $k(t)$ can be understood in similar terms: a rational fraction $f \in k(t)$ defines a function on k as well, except that it can be undefined at some points. Therefore Spec $k(t)$ might be thought as the affine line over k with a "moving" finite set of points removed. It is called the *generic point* of the affine line.

What about $k[\![t]\!]$? If k has characteristic 0 (which we assume for the simplicity of the discussion), the datum of $f \in k[\![t]\!]$ is equivalent to the datum of the values of the $f^{(n)}(0)$'s (for n varying in $\mathbb{N}$); we sometimes say that f defines a function on a *formal neighborhood* of 0. This formal neighborhood is the spectrum of $k[\![t]\!]$; it should then be thought as a kind of thickening of the point $0 \in$ Spec $k[t]$ which does not include any other point (since a formal series $f \in k[\![t]\!]$ cannot be evaluated in general at any other point than 0). Finally Spec $k((t))$ is the *punctured formal neighborhood* of 0; it is obtained from Spec $k[\![t]\!]$ by removing the point 0 but not its neighborhood!

The embedding $k[t] \to k[\![t]\!]$ (resp. $k(t) \to k((t))$) given by Taylor expansion at 0 corresponds to the restriction of a function f defined on the affine line (resp. the generic point of the affine line) to the formal neighborhood (resp. the punctured formal neighborhood) of 0.

Of course, one can define similarly formal neighborhoods and punctured formal neighborhoods around other points: for $a \in k$, the corresponding rings are respectively $k[\![h_a]\!]$ and $k((h_a))$ where h_a is a formal variable which plays the role $t-a$. The algebraic incarnation of the restrictions to Spec $k[\![h_a]\!]$ and $k((h_a))$ are the Taylor expansions at a.

[4] The natural embedding $\mathbb{F}_p(t) \to \mathbb{F}_p((t))$ takes a rational function to its Taylor expansion at 0.

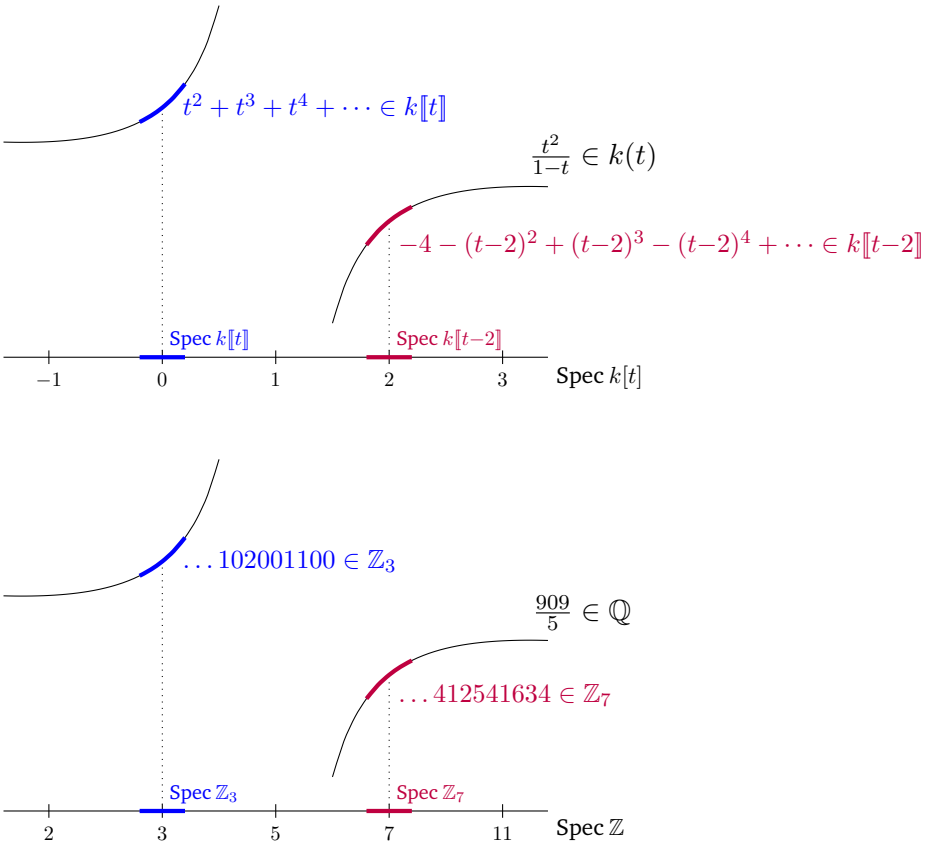

Figure 1.6: The point of view of algebraic geometry

The above discussion is illustrated by the drawing at the top of Figure 1.6; it represents more precisely the rational fraction $f(t)=\frac{t^2}{1-t}$ viewed as a function defined on Spec $k(t)$ (with 1 as point of indeterminacy) together its Taylor expansion around 0 (in blue) and 2 (in purple) which are defined on formal neighborhoods.

We now move to the arithmetical rings ($\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{Z}_p$ and $\mathbb{Q}_p$) for which the picture is surprisingly very similar. In order to explain what is the spectrum of $\mathbb{Z}$, we first need to reinterpret Spec $k[t]$ in a more algebraic way. Given $f\in k[t]$, its evaluation at a, namely $f(a)$, can alternatively be defined as the remainder in the Euclidean division of a by $t-a$. The affine line over k then appears more intrinsically as the set of monic polynomials of degree 1 over k or, even better, as the set of irreducible monic polynomials (recall that we had assumed that k is algebraically closed).

Translating this to the arithmetical world, we are inclined to think at the spectrum of $\mathbb{Z}$ as the set of prime numbers. Moreover, the evaluation of an integer $a\in\mathbb{Z}$ at the prime p should be nothing but the remainder in the Euclidean division of a by p. The integer a defines this way a function over the spectrum of $\mathbb{Z}$.

A rational number $\frac{a}{b}$ can be reduced modulo p for all prime p not dividing b. It then defines a function on the spectrum of $\mathbb{Z}$ except that there are points of indeterminacy. Spec $\mathbb{Q}$ then appears as the generic point of Spec $\mathbb{Z}$ exactly as Spec $k(t)$ appeared as the generic point of Spec $k[t]$.

Now sending an integer or a rational number to $\mathbb{Q}_p$ is the exact algebraic analogue of writing the Taylor expansion of a polynomial or a rational fraction. That is the reason why Spec $\mathbb{Z}_p$ should be thought of as a formal neighborhood of the point $p\in$ Spec $\mathbb{Z}$. Similarly Spec $\mathbb{Q}_p$ is the punctured formal neighborhood around p.

The second drawing of Figure 1.6 (which really looks like the first one) displays the rational number $\frac{909}{5}$ viewed as a function over Spec $\mathbb{Z}$ (with 5 as point of indeterminacy) together with its

local p-adic/Taylor expansion at $p = 3$ and $p = 7$.

1.3.2 Local-global principle

When studying equations where the unknown is a function, it is often interesting to look at local properties of a potential solution. Typically, if we have to solve a differential equation:

$$a_d(t)y^{(d)}(t) + a_{d-1}(t)y^{(d-1)}(t) + \cdots + a_1(t)y'(t) + a_0y(t) = b(t)$$

a standard strategy consists in looking for analytic solutions of the shape $y(t) = \sum_n c_n(t-a)^n$ for some a lying in the base field. The differential equation then rewrites as a recurrence on the coefficients c_n which sometimes can be solved. This reasoning yields local solutions which have to be glued afterwards.

Keeping in mind the analogy between functions and integers/rationals, we would like to use a similar strategy for studying Diophantine equations over $\mathbb{Q}$. Consider then a Diophantine equation of the shape:

$$P(x_1, \ldots, x_n) = 0 \tag{1.4}$$

where P is a polynomial with rational coefficients and the x_i's are the unknowns. If Eq. (1.4) has a global solution, *i.e.* a solution in $\mathbb{Q}^n$, then it must have local solutions everywhere, *i.e.* a solution in $\mathbb{Q}_p^n$ for each prime number p. Indeed $\mathbb{Q}$ embeds into each $\mathbb{Q}_p$. We are interested in the converse: assuming that Eq. (1.4) has local solutions everywhere, can we glue them in order to build a global solution? Unfortunately, the answer is negative in general. There is nevertheless one remarkable case for which this principle works well.

Theorem 1.10 (Hasse–Minkowski)**.** *Let $P(x_1, \ldots, x_n)$ be a multivariate polynomial. We assume that P is quadratic,* i.e. *that the total degree of P is 2. Then, the equation (1.4) has a solution in $\mathbb{Q}^n$ if and only if it has a solution in $\mathbb{R}^n$ and in $\mathbb{Q}_p^n$ for all prime numbers p.*

We refer to [75] for the proof of this theorem (which is not really the purpose of this course). Understanding the local-global principle beyond the case of quadratic polynomials has motivated a lot of research for more than 50 years. In 1970, at the International Congress of Mathematicians in Nice, Manin highlighted a new obstruction of cohomological nature to the possibility of glueing local solutions [61]. This obstruction is called nowadays the *Brauer–Manin obstruction*[5]. Exhibiting situations where it can explain, on its own, the non-existence of rational solutions is still an active domain of research today, in which very recent breakthrough progresses have been done [39, 38].

1.4 Why should we implement p-adic numbers?

We have seen that p-adic numbers are a wonderful mathematical object which might be quite useful for arithmeticians. However it is still not clear that it is worth implementing them in mathematical software. The aim of this subsection is to convince the reader that it is definitely worth it. Here are three strong arguments supporting this thesis:

(A) *p-adic numbers provide sometimes better numerical stability;*

(B) *p-adic numbers provide a solution for "allowing for division by p in $\mathbb{F}_p$";*

(C) *p-adic numbers really appear in nature.*

In the next paragraphs (§§1.4.1–1.4.3), we detail several examples illustrating the above arguments and showing that p-adic numbers appear as an essential tool in many questions of algorithmic nature.

[5] The naming comes from the fact this obstruction is written in the language of Brauer groups... the latter being defined by Grothendieck in the context we are interested in.

1.4.1 The Hilbert matrix

The first aforementioned argument, namely the possibility using p-adic numbers to have better numerical stability, is used in several contexts as factorization of polynomials over $\mathbb{Q}$ or computation of Galois groups of number fields. In order to avoid having to introduce too advanced concepts here, we have chosen to focus on a simpler example which was pointed out in Vaccon's PhD thesis [79, §1.3.4]: the computation of the inverse of the Hilbert matrix. Although this example is not directly related to the most concrete concerns of arithmeticians, it already very well highlights the potential of p-adic numbers when we are willing to apply numerical methods to a problem which is initially stated over the rationals.

We recall that the Hilbert matrix of size n is the square matrix H_n whose (i,j) entry is $\frac{1}{i+j-1}$ $(1 \leqslant i,j \leqslant n)$. For example:

$$H_4 = \begin{pmatrix} 1 & 1/2 & 1/3 & 1/4 \\ 1/2 & 1/3 & 1/4 & 1/5 \\ 1/3 & 1/4 & 1/5 & 1/6 \\ 1/4 & 1/5 & 1/6 & 1/7 \end{pmatrix}.$$

Hilbert matrices are famous for many reasons. One of them is that they are very ill-conditioned, meaning that numerical computations involving Hilbert matrices may lead to important numerical errors. A typical example is the computation of the inverse of H_n. Let us first mention that an exact formula giving the entries of H_n^{-1} is known:

$$(H_n^{-1})_{i,j} = (-1)^{i+j} \cdot (i+j-1) \cdot \binom{n+i-1}{n-j} \cdot \binom{n+j-1}{n-i} \cdot \binom{i+j-2}{i-1}^2. \tag{1.5}$$

(see for example [21]). We observe in particular than H_n^{-1} has integral coefficients.

We now move to the numerical approach: we consider H_n as a matrix with real coefficients and compute its inverse using standard Gaussian elimination (with choice of pivot) and IEEE floating-point arithmetics (with 53 bits of precision) [47]. Here is the result we get with SageMath [77] for $n = 4$:

$$H_4^{-1} \approx \begin{pmatrix} 15.9999999999998 & -119.999999999997 & 239.999999999992 & -139.999999999995 \\ -119.999999999997 & 1199.99999999996 & -2699.99999999989 & 1679.99999999993 \\ 239.999999999992 & -2699.99999999989 & 6479.99999999972 & -4199.99999999981 \\ -139.999999999995 & 1679.99999999993 & -4199.99999999981 & 2799.99999999987 \end{pmatrix}.$$

We observe that the accuracy of the computed result is acceptable but not so high: the number of correct binary digits is about 45 (on average on each entry of the matrix), meaning then that the number of incorrect digits is about 8. Let us now increase the size of the matrix and observe how the accuracy behaves:

size of the matrix	5	6	7	8	9	10	11	12	13
number of correct digits	40	34	28	25	19	14	9	4	0

We see that the losses of accuracy are enormous.

On the other hand, let us examine now the computation goes when H_n^{-1} is viewed as a matrix over $\mathbb{Q}_2$. Making experimentations in SageMath using again Gaussian elimination and the straightforward analogue of floating-point arithmetic (see §2.3) with 53 bits of precision, we observe the following behavior:

size of the matrix	5	6	7	8	9	10	11	12	13	50	100
number of correct digits	52	52	51	51	51	51	51	51	51	49	48

The computation of H_n^{-1} seems quite accurate over $\mathbb{Q}_2$ whereas it was on the contrary highly inaccurate over $\mathbb{R}$. As a consequence, if we want to use numerical methods to compute the exact inverse of H_n over $\mathbb{Q}$, it is much more interesting to go through the 2-adic numbers. Of course this approach does not make any sense if we want a *real* approximation of the entries of H_n^{-1}; in particular, if we are only interesting in the size of the entries of H_n^{-1} (but not in their exact values) passing through the p-adics is absurd since two integers of different sizes might be very close in the p-adic world.

The phenomena occurring here are actually easy to analyze. The accuracy of the inversion of a real matrix is governed by the condition number which is defined by:

$$\mathrm{cond}_{\mathbb{R}}(H_n) = \|H_n\|_{\mathbb{R}} \cdot \|H_n^{-1}\|_{\mathbb{R}}$$

where $\|\cdot\|_{\mathbb{R}}$ is some norm on $M_n(\mathbb{R})$. According to Eq. (1.5), the entries of H_n are large: as an example, the bottom right entry of H_n is equal to $(2n-1)\cdot\binom{2n-2}{n-1}^2$ and thus grows exponentially fast with respect to n. As a consequence the condition number $\mathrm{cond}_{\mathbb{R}}(H_n)$ is quite large as well; this is the source of the loss of accuracy observed for the computation of H_n^{-1} over $\mathbb{R}$.

Over $\mathbb{Q}_p$, one can argue similarly and consider the p-adic condition number:

$$\mathrm{cond}_{\mathbb{Q}_p}(H_n) = \|H_n\|_{\mathbb{Q}_p} \cdot \|H_n^{-1}\|_{\mathbb{Q}_p}$$

where $\|\cdot\|_{\mathbb{Q}_p}$ is the infinite norm over $M_n(\mathbb{Q}_p)$ (other norms do not make sense over $\mathbb{Q}_p$ because of the ultrametricity). Since H_n^{-1} has integral coefficients, all its entries have their p-adic norm bounded by 1. Thus $\|H_n^{-1}\|_{\mathbb{Q}_p} \leqslant 1$. As for the Hilbert matrix H_n itself, its norm is equal to p^v where v is the highest power of p appearing in a denominator of an entry of H_n, *i.e.* v is the unique integer such that $p^v \leqslant 2n-1 < p^{v+1}$. Therefore $\|H_n\|_{\mathbb{Q}_p} \leqslant 2n$ and $\mathrm{cond}_{\mathbb{Q}_p}(H_n) = O(n)$. The growth of the p-adic condition number in then rather slow, explaining why the computation of H_n^{-1} is accurate over $\mathbb{Q}_p$.

1.4.2 Lifting the positive characteristic

In this paragraph, we give more details about the mysterious sentence: *p-adic numbers provide a solution for "allowing for divisions by p in $\mathbb{F}_p$"*. Let us assume that we are working on a problem that makes sense over any field and always admits a unique solution (we will give examples later). To fix ideas, let us agree that our problem consists in designing a fast algorithm for computing the aforementioned unique solution. Assume further that such an algorithm is available when k has characteristic 0 but does not extend to the positive characteristic (because it involves a division by p at some point). In this situation, one can sometimes take advantage of p-adic numbers to attack the problem over the finite field $\mathbb{F}_p$, proceeding concretely in three steps as follows:

1. we lift our problem over $\mathbb{Q}_p$ (meaning that we introduce a p-adic instance of our problem whose reduction modulo p is the problem we have started with),
2. we solve the problem over $\mathbb{Q}_p$ (which has characteristic 0),
3. we finally reduce the solution modulo p.

The existence and uniqueness of the solution together ensure that the solution of the p-adic problem is defined over $\mathbb{Z}_p$ and reduces modulo p to the correct answer. Concretely, here are two significant examples where the above strategy was used with success: the fast computation of composed products [13] and the fast computation of isogenies in positive characteristic [57, 53].

In order to give more substance to our thesis, we briefly detail the example of composed products. Let k be a field. Given two monic polynomials P and Q with coefficients in k, we recall that the composed product of P and Q is defined by:

$$P \otimes Q = \prod_{\alpha} \prod_{\beta} (X - \alpha\beta)$$

where α (resp. β) runs over the roots of P (resp. of Q) in an algebraic closure of k, with the convention that a root is repeated a number of times equal to its multiplicity. Note that $P \otimes Q$ is always defined over k since all its coefficients are written as symmetric expressions in the α's and the β's. We address the question of the fast multiplication of composed products.

If k has characteristic 0, Dvornicich and Traverso [27] proposed a solution based on Newton sums. Indeed, letting $S_{P,n}$ and $S_{Q,n}$ be the Newton sums of P and Q respectively:

$$S_{P,n} = \sum_{\alpha} \alpha^n \quad and \quad S_{Q,n} = \sum_{\beta} \beta^n.$$

It is obvious that the product $S_{P,n} \cdot S_{Q,n}$ is the n-th Newton sum of $P \otimes Q$. Therefore, if we design fast algorithms for going back and forth between the coefficients of a polynomial and its Newton sums, the problem of fast computation of the composed product will be solved.

Schönhage [73] proposed a nice solution to the latter problem. It relies on a remarkable differential equation relating a polynomial and the generating function of the sequence of its Newton sums. Precisely, let A be a monic polynomial of degree d over k. We define the formal series:

$$S_A(t) = \sum_{n=0}^{\infty} S_{A,n+1} t^n.$$

A straightforward computation then leads to the following remarkable relation:

$$(E): \quad A'_{\text{rec}}(t) = -S_A(t) \cdot A_{\text{rec}}(t)$$

where A_{rec} is the reciprocal polynomial of A defined by $A_{\text{rec}}(X) = X^d \cdot A(\frac{1}{X})$. If the polynomial A is known, then one can compute $S_A(t)$ by performing a simple division in $k[\![t]\!]$. Using fast multiplication and Newton iteration for computing the inverse (see §1.2.2), this method leads to an algorithm that computes $S_1, S_2, \ldots, S_N$ for a cost of $\tilde{O}(N)$ operations in k.

Going in the other direction reduces to solving the differential equation (E) where the unknown is now the polynomial A_{rec}. When k has characteristic 0, Newton iteration can be used in this context (see for instance [12]). This leads to the following recurrence:

$$B_1(t) = 1 \quad ; \quad B_{i+1}(t) = B_i(t) - B_i(t) \int \left(\frac{B'_i(t)}{B_i(t)} + S_A(t) \right) dt$$

where the integral sign stands for the linear operator acting on $k[\![t]\!]$ by $t^n \mapsto \frac{t^{n+1}}{n+1}$. One proves that $B_i(t) \equiv A_{\text{rec}}(t) \pmod{t^{2^i}}$ for all i. Since $A_{\text{rec}}(t)$ is a polynomial of degree d, it is then enough to compute it modulo t^{d+1}. Hence after $\log_2(d+1)$ iterations of the Newton scheme, we have computed all the relevant coefficients. Moreover all the computations can be carried by omitting all terms of degree $> d$. This leads to an algorithm that computes A_{rec} (and hence A) for a cost of $\tilde{O}(d)$ operations in k.

In positive characteristic, this strategy no longer works because the integration involves divisions by integers. When k is the finite field with p elements[6], one can tackle this issue by lifting the problem over $\mathbb{Z}_p$, following the general strategy discussed here. First, we choose two polynomials $\hat{P}$ and $\hat{Q}$ with coefficients in $\mathbb{Z}_p$ with the property that $\hat{P}$ (resp. $\hat{Q}$) reduces to P (resp. Q) modulo p. Second, applying the characteristic 0 method to $\hat{P}$ and $\hat{Q}$, we compute $\hat{P} \otimes \hat{Q}$. Third, we reduce the obtained polynomial modulo p, thus recovering $P \otimes Q$. This strategy was concretized by the work of Bostan, Gonzalez-Vega, Perdry and Schost [13]. The main difficulty in their work is to keep control on the p-adic precision we need in order to carry out the computation and be sure to get the correct result at the end; this analysis is quite subtle and was done by hand in [13]. Very general and powerful tools are now available for dealing with such questions; they will be presented in §3.

[6] The same strategy actually extends to all finite fields.

1.4.3 p-adic numbers in nature

We have already seen in the introduction of this course that p-adic numbers are involved in many recent developments in Number Theory and Arithmetic Geometry (see also §1.3.2). Throughout the 20th century, many p-adic objects have been defined and studied as well: p-adic modular/automorphic forms, p-adic differential equations, p-adic cohomologies, p-adic Galois representations, *etc.* It turns out that mathematicians, more and more often, feel the need to make "numerical" experimentations on these objects for which having a nice implementation of p-adic numbers is of course a prerequisite.

Moreover p-adic theories sometimes have direct consequences in other areas of mathematics. A good example in this direction is the famous problem of counting points on algebraic curves defined over finite fields (*i.e.* roughly speaking counting the number of solutions in finite fields of equations of the shape $P(X, Y) = 0$ where P is bivariate polynomial). This question has received much attention during the last decade because of its applications to cryptography (it serves as a primitive in the selection of secure elliptic curves). Since the brilliant intuition of Weil [80] followed by the revolution of Algebraic Geometry conducted by Grothendieck in the 20th century [31], the approach to counting problems is now often cohomological. Roughly speaking, if C is an algebraic variety (with some additional assumptions) defined over a finite field $\mathbb{F}_q$, then the number of points of C is related to the traces of the Frobenius map acting on the cohomology of C. Counting points on C then "reduces" to computing the Frobenius map on the cohomology. Now it turns out that traditional algebraic cohomology theory yields vector spaces over $\mathbb{Q}_\ell$ (for an auxiliary prime number ℓ which does not need to be equal to the characteristic of the finite field we are working with). This is the way ℓ-adic numbers enter naturally into the scene.

2 Several implementations of p-adic numbers

Now we are all convinced that it is worth implementing p-adic numbers, we need to discuss the details of the implementation. The main problem arising when we are trying to put p-adic numbers on computers is the precision. Indeed, remember that a p-adic number is defined as an infinite sequence of digits, so that it *a priori* cannot be stored entirely in the memory of a computer. From this point of view, p-adic numbers really behave like real numbers; the reader should therefore not be surprised if he/she often detects similarities between the solutions we are going to propose for the implementation of p-adic numbers and the usual implementation of reals.

In this section, we design and discuss three totally different paradigms: zealous arithmetic (§2.1), lazy arithmetic together with the relaxed improvement (§2.2) and p-adic floating-point arithmetic (§2.3). Each of these ways of thinking has of course its own advantages and disadvantages; we will try to compare them fairly in §2.4.

2.1 Zealous arithmetic

Zealous arithmetic is by far the most common implementation of p-adic numbers in usual mathematical software: MAGMA [11], SAGEMATH [77] and PARI [4] use it for instance. It appears as the exact analogue of the interval arithmetic of the real setting: we replace p-adic numbers by intervals. The benefit is of course that intervals can be represented without errors by finite objects and then can be stored and manipulated by computers.

2.1.1 Intervals and the big-O notation

Recall that we have defined in §1.1.5 (see Definitions 1.4 and 1.5) the notion of interval: a bounded interval of $\mathbb{Q}_p$ is by definition a closed ball lying inside $\mathbb{Q}_p$, *i.e.* a subset of the form:

$$I_{N,a} = \{\, x \in \mathbb{Q}_p \quad \text{s.t.} \quad |x-a| \leqslant p^{-N} \,\}.$$

The condition $|x-a| \leqslant p^{-N}$ can be rephrased in a more algebraic way since it is equivalent to the condition $x \equiv a \pmod{p^N\mathbb{Z}_p}$, *i.e.* $x \in a + p^N\mathbb{Z}_p$. In short $I_{N,a} = a + p^N\mathbb{Z}_p$. In symbolic computation, we generally prefer using the big-O notation and write:

$$I_{N,a} = a + O(p^N).$$

The following result is easy but fundamental for our propose.

Proposition 2.1. *Each bounded interval I of $\mathbb{Q}_p$ may be written as:*

$$I = p^v s + O(p^N)$$

where (N, v, s) is either of the form $(N, N, 0)$ or a triple of relative integers with $v < N$, $s \in [0, p^{N-v})$ and $\gcd(s, p) = 1$. Moreover this writing is unique.

In particular, bounded intervals of $\mathbb{Q}_p$ are representable by exact values on computers.

Proof. It is a direct consequence of the fact that $I_{N,a}$ depends only on the class of a modulo p^N. □

The interval $a + O(p^N)$ has a very suggestive interpretation if we are thinking at p-adic numbers in terms of infinite sequences of digits. Indeed, write

$$a = a_v p^v + a_{v+1} p^{v+1} + \cdots + a_{N-1} p^{N-1} + \cdots$$

with $0 \leqslant a_i < p$ and agree to define $a_i = 0$ for $i < v$. A p-adic number x then lies in $a + O(p^N)$ if its i-th digit is a_i for all $i < N$. Therefore one may think of the notation $a + O(p^N)$ as a p-adic number of the shape:

$$a_v p^v + a_{v+1} p^{v+1} + \cdots + a_{N-1} p^{N-1} + ?\, p^N + ?\, p^{N+1} + \cdots$$

where the digits at all positions $\geqslant N$ are unspecified. This description enlightens Proposition 2.1 which should become absolutely obvious now.

We conclude this paragraph by introducing some additional vocabulary.

Definition 2.2. Let $I = a + O(p^N)$ be an interval. We define:

- the *absolute precision* of I: $\quad \mathrm{abs}(I) = N$;
- the *valuation* of I: $\quad \mathrm{val}(I) = \mathrm{val}(a)$ if $0 \notin I$, $\quad \mathrm{val}(I) = N$ otherwise;
- the *relative precision* of I: $\quad \mathrm{rel}(I) = \mathrm{abs}(I) - \mathrm{val}(I)$.

We remark that the valuation of I is the integer v of Proposition 2.1. It is also always the smallest valuation of an element of I. Moreover, coming back to the interpretation of p-adic numbers as infinite sequences of digits, we observe that the relative precision of $a + O(p^N)$ is the number of known digits (with rightmost zeroes omitted) of the family of p-adic numbers it represents.

2.1.2 The arithmetics of intervals

Since intervals are defined as subsets of $\mathbb{Q}_p$, basic arithmetic operations on them are defined in a straightforward way. For example, the sum of the intervals I and J is the subset of $\mathbb{Q}_p$ consisting of all the elements of the form $a+b$ with $a \in I$ and $b \in J$. As with real intervals, it is easy to write down explicit formulas giving the results of the basic operations performed on p-adic intervals.

Proposition 2.3. *Let $I = a + O(p^N)$ and $I' = a' + O(p^{N'})$ be two bounded intervals of $\mathbb{Q}_p$. Set $v = \mathrm{val}(I)$ and $v' = \mathrm{val}(I')$. We have:*

$$I + I' = a + a' + O\big(p^{\min(N,\, N')}\big) \tag{2.1}$$

$$I - I' = a - a' + O\big(p^{\min(N,\, N')}\big) \tag{2.2}$$

$$I \times I' = aa' + O\big(p^{\min(v+N',\, N+v')}\big) \tag{2.3}$$

$$I \div I' = \frac{a}{a'} + O\big(p^{\min(v+N'-2v',\, N-v')}\big) \tag{2.4}$$

where in the last equality, we have further assumed that $0 \notin I'$.

Remark 2.4. Focusing on the precision, we observe that the two first formulae stated in Proposition 2.3 immediately imply $\mathrm{abs}(I + I') = \mathrm{abs}(I - I') = \min(\mathrm{abs}(I), \mathrm{abs}(I'))$. Concerning multiplication and division, the results look ugly at first glance. They are however much more pretty if we translate them in terms of relative precision; indeed, they become simply $\mathrm{rel}(I \times I') = \mathrm{rel}(I \div I') = \min(\mathrm{rel}(I), \mathrm{rel}(I'))$ which is parallel to the case of addition and subtraction and certainly easier to remember.

Proof of Proposition 2.3. The proofs of Eq. (2.1) and Eq. (2.2) are easy and left as an exercise to the reader. We then move directly to multiplication. Let $x \in I \times I'$. By definition, there exist $h \in p^N\mathbb{Z}_p$ and $h' \in p^{N'}\mathbb{Z}_p$ such that:

$$x = (a+h)(a'+h') = aa' + ah' + a'h + hh'.$$

Moreover, coming back to the definition of the valuation of an interval, we find $v \leqslant \mathrm{val}(a)$. The term ah' is then divisible by $p^{v+N'}$. Similarly $a'h$ is divisible by $p^{N+v'}$. Finally hh' is divisible by $p^{v+v'}$; it is then *a fortiori* also divisible by $p^{v+N'}$ because $v \leqslant N$. Thus $ah' + a'h + hh'$ is divisible by $p^{\min(v+N',\, N+v')}$ and we have proved one inclusion:

$$I \times I' \subset aa' + O\big(p^{\min(v+N',\, N+v')}\big).$$

Conversely, up to swapping I and I', we may assume that $\min(v+N',\, N+v') = v+N'$. Up to changing a, we may further suppose that $\mathrm{val}(a) = v$. Pick now $y \in aa' + O(p^{v+N'})$, *i.e.* $y = aa' + h$ for some h divisible by $p^{v+N'}$. Thanks to our assumption on a, we have $\mathrm{val}(\frac{h}{a}) = \mathrm{val}(h) - \mathrm{val}(a) \geqslant v + N' - v = N'$, *i.e.* $p^{N'}$ divides $\frac{h}{a}$. Writing $y = a \times (a' + \frac{h}{a})$ then shows that $y \in I \times I'$. The converse inclusion follows and Eq. (2.3) is established.

We now assume that $0 \notin I'$ and start the proof of Eq. (2.4). We define $1 \div I'$ as the set of inverses of elements of I'. We claim that:

$$1 \div I' = \frac{1}{a'} + O(p^{N'-2v'}). \tag{2.5}$$

In other to prove the latter relation, we write $I' = a' \times (1 + O(p^{N-v'}))$. We then notice that the function $x \mapsto x^{-1}$ induces a involution from $1 + O(p^{N-v'})$ to itself; in particular $1 \div (1 + O(p^{N-v'})) = 1 + O(p^{N-v'})$. Dividing both sides by a', we get (2.5). We finally derive that $I \div I' = I \times \big(\frac{1}{a'} + O(p^{N'-2v'})\big)$. Eq. (2.4) then follows from Eq. (2.3). □

Using Proposition 2.3, it is more or less straightforward to implement addition, subtraction, multiplication and division on intervals when they are represented as triples (N, v, s) as in Proposition 2.1 (be careful however at the conditions on v and s). This yields the basis of the *zealous arithmetic*.

2.1.3 Newton iteration

A tricky point when dealing with zealous arithmetic concerns Newton iteration. To illustrate it, let us examine the example of the computation of a square root of c over $\mathbb{Q}_2$ for

$$c = 1 + 2^3 + 2^4 + 2^5 + 2^{10} + 2^{13} + 2^{16} + 2^{17} + 2^{18} + 2^{19} + O(2^{20}).$$

We observe that $c \equiv 1 \pmod 8$. By Hensel's Lemma, c has then a unique square root in $\mathbb{Q}_2$ which is congruent to 1 modulo 4. Let us denote it by $\sqrt{c}$. Following §1.2.3, we compute $\sqrt{c}$ using Newton iteration: if $(x_i)_{i\geqslant 0}$ is the recursive sequence defined by

$$x_0 = 1 \quad ; \quad x_{i+1} = \frac{1}{2} \cdot \left(x_i + \frac{c}{x_i} \right), \; i = 0, 1, 2, \ldots$$

we know that x_i and $\sqrt{c}$ share the same $(2^i + 1)$ final digits. Here is the result we get if we compute the first x_i's using zealous arithmetic:

$$\begin{aligned}
x_1 &= \ldots 1111001001000011101 \\
x_2 &= \ldots 001110100011110101 \\
x_3 &= \ldots 101110100010101 \\
x_4 &= \ldots 0111010001010101 \\
x_5 &= \ldots 111010001010101 \\
x_6 &= \ldots 11010001010101
\end{aligned}$$

Here the 2^i+1 rightmost digits of x_i (which are the correct digits of $\sqrt{c}$) are colored in purple and the dots represent the unknown digits, that is the digits which are absorbed by the $O(-)$. We observe that the precision decreases by 1 digit at each iteration; this is of course due to the division by 2 in the recurrence defining the x_i's. The maximal number of correct digits is obtained for x_4 with 16 correct digits. After this step, the result stabilizes but the precision continues to decrease. Combining naively zealous arithmetic and Newton iteration, we have then managed to compute $\sqrt{c}$ at precision $O(2^{16})$.

It turns out that the losses of precision we have just highlighted has no intrinsic meaning but is just a consequence of zealous arithmetic. We will now explain that it is possible to slightly modify Newton iteration in order to completely eliminate this unpleasant phenomenon. The starting point of your argument is Remark 1.8 which tells us that Newton iteration still converges at the same rate to $\sqrt{c}$ as soon as the sequence (x_i) satisfies the weaker condition:

$$\left| x_{i+1} - \frac{1}{2} \cdot \left(x_i + \frac{c}{x_i} \right) \right| \leqslant 2^{2^{i+1}+1}.$$

In other words, we have a complete freedom on the choice of the "non-purple" digits of x_{i+1}. In particular, if some digit of x_{i+1} was not computed because the precision on x_i was too poor, we can just assign freely our favorite value (*e.g.* 0) to this digit without having to fear unpleasant consequences. Even better, we can assign 0 to each "non-purple" digit of x_i as well; this will not affect the final result and leads to computation with smaller integers. Proceeding this way, we obtain the following sequence:

$$\begin{aligned}
x_1 &: \ldots 0000000000000000101 \\
x_2 &: \ldots 0000000000000010101 \\
x_3 &: \ldots 0000000000001010101 \\
x_4 &: \ldots 0001011101010001010101 \\
x_5 &: \ldots 1010111010001010101
\end{aligned}$$

and we obtain the final result with 19 correct digits instead of 16. Be very careful: we cannot assign arbitrarily the 20-th digit of x_5 because it is a "purple" digit and not a "non-purple" one.

More precisely if we do it, we have to write it in black (of course, we cannot decide randomly what are the correct digits of $\sqrt{c}$) and a new iteration of the Newton scheme again loses this 20-th digit because of the division by 2. In that case, one may wonder why we did not try to assign a 21-st digit to x_4 in order to get one more correct digit in x_5. This sounds like a good idea but does not work because the input c — on which we do not have any freedom — appears in the recurrence formula defining the x_i's and is given at precision $O(2^{20})$. For this reason, each x_i cannot be computed with a better precision than $O(2^{19})$.

In fact, we can easily convince ourselves that $O(2^{19})$ is the optimal precision for $\sqrt{c}$. Indeed c can be lifted at precision $O(2^{21})$ either to:

$$\begin{aligned} & c_1 = 1 + 2^3 + 2^4 + 2^5 + 2^{10} + 2^{13} + 2^{16} + 2^{17} + 2^{18} + 2^{19} + O(2^{21}) \\ \text{or} \quad & c_2 = 1 + 2^3 + 2^4 + 2^5 + 2^{10} + 2^{13} + 2^{16} + 2^{17} + 2^{18} + 2^{19} + 2^{20} + O(2^{21}). \end{aligned}$$

The square roots of these liftings are:

$$\begin{aligned} \sqrt{c_1} &= 1 + 2^2 + 2^4 + 2^6 + 2^{10} + 2^{12} + 2^{13} + 2^{14} + 2^{16} + 2^{18} + O(2^{20}) \\ \sqrt{c_2} &= 1 + 2^2 + 2^4 + 2^6 + 2^{10} + 2^{12} + 2^{13} + 2^{14} + 2^{16} + 2^{18} + 2^{19} + O(2^{20}) \end{aligned}$$

and we now see clearly that the 19-th digit of $\sqrt{c}$ is affected by the 20-th digit of c.

2.2 Lazy and relaxed arithmetic

The very basic idea behind lazy arithmetic is the following: in every day life, we are not working with all p-adic numbers but only with *computable* p-adic numbers, *i.e.* p-adic numbers for which there exists a program that outputs the sequence of its digits. A very natural idea is then to use these programs to represent p-adic numbers. By operating on programs, one should then be able to perform additions, multiplications, *etc.* of p-adic numbers.

In this subsection, we examine further this idea. In §2.2.1, we adopt a very naive point of view, insisting on ideas and not taking care of doability/performances. We hope that this will help the reader to understand more quickly the mechanisms behind the notion of lazy p-adic numbers. We will then move to complexity questions and will focus on the problem of designing a framework allowing our algorithms to take advantage of fast multiplication algorithms for integers (as Karatsuba's algorithm, Schönhage–Strassen's algorithm [33, §8] or Fürer's algorithm and its improvements [30, 40]). This will lead us to report on the theory of *relaxed algorithms* introduced recently by Van der Hoeven and his followers [42, 43, 44, 7, 8, 55].

Lazy p-adic numbers have been implemented in the software MATHEMAGIX [45].

2.2.1 Lazy p-adic numbers

Definition 2.5. A *lazy p-adic number* is a program x that takes as input an integer N and outputs a *rational number* $\mathrm{x}(N)$ with the property that:

$$\big|\mathrm{x}(N+1) - \mathrm{x}(N)\big|_p \leqslant p^{-N}$$

for all N.

The above condition implies that the sequence $\mathrm{x}(N)$ is a Cauchy sequence in $\mathbb{Q}_p$ and therefore converges. We call its limit the *value* of x and denote it by $\mathrm{value}(\mathrm{x})$. Thanks to ultrametricity, we obtain $|\mathrm{value}(\mathrm{x}) - \mathrm{x}(N)| \leqslant p^{-N}$ for all N. In other words, $\mathrm{x}(N)$ is nothing but an approximation of $\mathrm{value}(\mathrm{x})$ at precision $O(p^N)$.

The p-adic numbers that arise as values of lazy p-adic numbers are called *computable*. We observe that there are only countably many lazy p-adic numbers; the values they take in $\mathbb{Q}_p$ then

form a countable set as well. Since $\mathbb{Q}_p$ is uncountable (it is equipotent to the set of functions $\mathbb{N} \to \{0, 1, \ldots, p-1\}$), it then exists many uncomputable p-adic numbers.

Our aim is to use lazy p-adic numbers to implement actual (computable) p-adic numbers. In order to do so, we need (at least) to answer the following two questions:

- Given (the source code of) two lazy p-adic numbers x and y, can we decide algorithmically whether value(x) $=$ value(y) or not?
- Can we lift standard operations on p-adic numbers at the level of lazy p-adic numbers?

Unfortunately, the first question admits a negative answer; it is yet another consequence of Turing's halting problem. For our purpose, this means that it is impossible to implement equality at the level of lazy p-adic numbers. Inequalities however can always be detected. In order to explain this, we need a lemma.

Lemma 2.6. *Let* x *be a lazy p-adic number and set $x =$* value(x)*. For all integers N, we have:*

$$\begin{aligned} &\mathrm{val}_p(x) = \mathrm{val}_p(\mathrm{x}(N)) && \text{if } \mathrm{val}_p(\mathrm{x}(N)) < N \\ &\mathrm{val}_p(x) \geqslant N && \text{otherwise.} \end{aligned}$$

Proof. Recall that the norm is related to the valuation through the formula $|a| = p^{-\mathrm{val}_p(a)}$ for all $a \in \mathbb{Q}_p$. Moreover by definition, we know that $|x - \mathrm{x}(N)| \leqslant p^{-N}$. If $\mathrm{val}_p(\mathrm{x}(N)) \geqslant N$, we derive $|\mathrm{x}(N)| \leqslant p^{-N}$ and thus $|x| \leqslant p^{-N}$ by the ultrametric inequality. On the contrary if $\mathrm{val}_p(\mathrm{x}(N)) < N$, write $x = (x - \mathrm{x}(N)) + \mathrm{x}(N)$ and observe that the two summands have different norms. Hence $|x| = \min\big(|x - \mathrm{x}(N)|, |\mathrm{x}(N)|\big) = \mathrm{x}(N)$ and we are done. □

Lemma 2.6 implies that value(x) $=$ value(y) if and only if $\mathrm{val}_p(\mathrm{x}(N) - \mathrm{y}(N)) \geqslant N$ for all integers N. Thus, assuming that we have access to a routine val_p computing the p-adic valuation of a rational number, we can write down the function `is_equal` as follows (Python style[7]):

```
def is_equal(x,y):
    for N in ℕ:
        v = valₚ(x(N) - y(N))
        if v < N: return False
    return True
```

We observe that this function stops if and only if it answers False, *i.e.* inequalities are detectable but equalities are not.

We now move to the second question and try to define standard operations at the level of lazy p-adic numbers. Let us start with addition. Given two computable p-adic numbers x and y represented by the programs x and y respectively, it is actually easy to build a third program `x_plus_y` representing the sum $x + y$. Here it is:

```
def x_plus_y(N):
    return x(N) + y(N)
```

The case of multiplication is a bit more subtle because of precision. Indeed going back to Eq. (2.3), we see that, in order to compute the product xy at precision $O(p^N)$, we need to know x and y at precision $O(p^{N-\mathrm{val}(y)})$ and $O(p^{N-\mathrm{val}(x)})$ respectively. We thus need to compute first the valuation of x and y. Following Proposition 2.6, we write down the following procedure:

```
def val(x):
    for N in ℕ:
        v = valₚ(x(N))
        if v < N: return v
    return Infinity
```

[7] We emphasize that all the procedures we are going to write are written in pseudo-code in Python *style* but certainly not in pure Python: they definitely do not "compile" in Python.

However, we observe one more time that the function `val` can never stop; precisely it stops if and only if x does not vanish. For application to multiplication, it is nevertheless not an issue. Recall that we needed to know $\mathrm{val}(x)$ because we wanted to compute y at precision $O(p^{N-\mathrm{val}(x)})$. But obviously computing y at higher precision will do the job as well. Hence it is in fact enough to compute an integer v_x with the guarantee that $\mathrm{val}(x) \geqslant v_x$. By Lemma 2.6, an acceptable value of v_x is $\min(0, \mathrm{val}_p(\mathrm{x}(0)))$ (or more generally $\min(i, \mathrm{val}_p(\mathrm{x}(i)))$ for any value of i). A lazy p-adic number whose value if xy is then given by the following program:

```
def x_mul_y(N):
    vx = min(0, val_p(x(0))
    vy = min(0, val_p(y(0))
    return x(N-vy) * y(N-vx)
```

The case of division is similar except that we cannot do the same trick for the denominator: looking at Eq. (2.4), we conclude that we do not need a lower bound on the valuation of the denominator but an upper bound. This is not that surprising since we easily imagine that the division becomes more and more "difficult" when the denominator gets closer and closer to 0. Taking the argument to its limit, we notice that a program which is supposed to perform a division should be able, as a byproduct, to detect whether the divisor is zero or not. But we have already seen that the latter is impossible. We are then reduced to code the division as follows:

```
def x_div_y(N):
    vx = min(0, val_p(x(0))
    vy = val(y)   # This step might never stop
    return x(N + vy) / y(N + 2*vy - vx)
```

keeping in mind that the routine never stops when the denominator vanishes.

2.2.2 Relaxed arithmetic

The algorithms we have sketched in §2.2.1 are very naive and inefficient. In particular, they redo many times a lot of computations. For example, consider the procedure `x_plus_y` we have designed previously and observe that if we had already computed the sum $x + y$ at precision $O(p^N)$ and we now ask for the same sum $x + y$ at precision $O(p^{N+1})$, the computation restarts from scratch without taking advantage of the fact that only one digit of the result remains unknown.

One option for fixing this issue consists basically in implementing a "sparse cache": we decide in advance to compute and cache the $\mathrm{x}(N)$'s only for a few values of N's (typically the powers of 2) and, on the input N, we output $\mathrm{x}(N')$ for a "good" $N' \geqslant N$. Concretely (a weak form of) this idea is implemented by maintaining two global variables `current_precision[x]` and `current_approximation[x]` (attached to each lazy p-adic number x) which are always related by the equation:

```
current_approximation[x] = x(current_precision[x])
```

If we are asking for $\mathrm{x}(N)$ for some N which is not greater than `current_approximation[x]`, we output `current_precision[x]` without launching a new computation. If, instead, N is greater than `current_approximation[x]`, we are obliged to redo the computation but we take the opportunity to double the current precision (at least). Here is the corresponding code:

```
def x_with_sparse_cache(N):
    if current_precision[x] < N:
        current_precision[x] = max(N, 2*current_precision[x])
        current_approximation[x] = x(current_precision[x])
    return current_approximation[x]
```

This solution is rather satisfying but it has nevertheless several serious disadvantages: it often outputs too large results[8] (because they are too accurate) and often does unnecessary computations.

Another option was developed more recently first by van der Hoeven in the context of formal power series [42, 43, 44] and then by Berthomieu, Lebreton, Lecerf and van der Hoeven for p-adic numbers [7, 8, 55]. It is the so-called *relaxed arithmetic* that we are going to expose now. For simplicity we shall only cover the case of p-adic *integers*, *i.e.* $\mathbb{Z}_p$. Extending the theory to $\mathbb{Q}_p$ is more or less straightforward but needs an additional study of the precision which makes the exposition more technical without real benefit.

Data structure and addition

The framework in which relaxed arithmetic takes place is a bit different and more sophisticated than the framework we have used until now. First of all, relaxed arithmetic does not view a p-adic number as a sequence of more and more accurate approximations but as the sequence of its digits. Moreover it needs a richer data structure in order to provide enough facilities for designing its algorithms. For this reason, it is preferable to encode p-adic numbers by classes which may handle its own internal variables and its own methods.

We first introduce the class `RelaxedPAdicInteger` which is an abstract class for all relaxed p-adic integers (*i.e.* relaxed p-adic integers must be all instances of a derived class of `RelaxedPAdicInteger`). The class `RelaxedPAdicInteger` is defined as follows:

```
class RelaxedPAdicInteger:
    # Variable
    digits              # list of (already computed) digits

    # Virtual method
    def next(self)      # compute the next digit and append it to the list
```

We insist on the fact that the method `next` is an internal (private) method which is not supposed to be called outside the class. Of course, although it does not appear above, we assume that the class `RelaxedPAdicInteger` is endowed with several additional public methods making the interface user-friendly. For instance if x is a relaxed p-adic integer inheriting from `RelaxedPAdicInteger`, we shall often use the construction x$[N]$ to access to the N-th digit of x. In pure Python, this functionality can be implemented by adding to the class a method `__getitem__` written as follows:

```
    def __getitem__(self,N):
        n = len(self.digits)    # Index of the first uncomputed digit
        while n < N+1:          # We compute the digits until the position N
            self.next()
            n += 1
        return self.digits[N]   # We return the N-th digit
```

In order to highlight the flexibility of the construction, let us explain as a warm-up how addition is implemented in this framework. One actually just follows the schoolbook algorithm: the n-th digit of a sum is obtained by adding the n-th digit of the summands plus possibly a carry. We add an additional local variable to the class in order to store the current carry and end up with the following implementation:

```
class x_plus_y(RelaxedPAdicInteger):
    # Additional variable
    carry               # variable for storing the current carry

    def next(self):     # compute the next digit and append it to the list
        n = len(self.digits)            # index of the first uncomputed digit
        s = x[n] + y[n] + self.carry    # perform the addition
```

[8] Of course it is always possible to reduce the result modulo p^N but this extra operation has a non-negligible cost.

```
        self.digits[n] = s % p          # compute and store the new digit
        self.carry = s // p             # update the carry
```

That's all. Although this does not appear in the pseudo-code above, the relaxed p-adic integers x and y (which are supposed to be instances of the class `RelaxedPAdicInteger`) should be passed as attributes to the constructor of the class.

The subtraction is performed similarly (exercise left to the reader).

Multiplication

A priori, multiplication can be treated in a similar fashion. Given x and y two p-adic integers with digits x_i's and y_j's, the n-th digit of a product xy is obtained by adding the contribution of all cross products $x_i y_j$ for $i + j = n$ plus a possible carry. This approach leads to the following straightforward implementation:

```
class x_mul_y(RelaxedPAdicInteger):
    # Additional variable
    carry               # variable for storing the current carry

    def next(self):     # compute the next digit and append it to the list
        n = len(self.digits)
        s = self.carry
        for i in 0,1,...,n:
            s += x[i] * y[n-i]
        self.digits[n] = s % p
        self.carry = s // p
```

Nevertheless this strategy has one important defect: it does not take advantage of the fast multiplication algorithms for integers, that is, it computes the first N digits of xy in $O(N^2 \log p)$ bit operations while we would have expected only $\tilde{O}(N \log p)$.

The relaxed multiplication algorithm from [7] fixes this drawback. The rough idea behind it is to gather digits in x and y as follows:

$$x = x_0 + (x_1 + x_2 p)p + (x_3 + x_4 p + x_5 p^2 + x_6 p^3)p^3 + \cdots$$
$$y = y_0 + (y_1 + y_2 p)p + (y_3 + y_4 p + y_5 p^2 + y_6 p^3)p^3 + \cdots$$

and to implement a kind of block multiplication. More precisely, we materialize the situation by a grid drawn in the half plane (see Figure 2.1). The coordinate on the x-axis (resp. the y-axis) corresponds to the integers i's (resp. j's) that serve as indices for the positions of the digits of the p-adic integer x (resp. y). As for the cell (i, j), it represents the product $x_i y_j$. The computation of the n-th digit of xy needs to know all the digits appearing on the anti-diagonal $i + j = n$; we refer again to Figure 2.1 where this anti-diagonal is displayed for $n = 10$.

We now pave the grid using larger and larger squares as follows. We pave the first row[9] and the first column by squares of size 1. Once this has been done, we forget temporarily the cells that have been already paved (*i.e.* the cells lying on the first row or the first column), we group the two next lines and the two next columns together and we pave them with squares of size 2. We continue this process and double the size of the squares at each iteration. The obtained result is displayed on Figure 2.1. This construction exhibits the following remarkable property:

Lemma 2.7. *If (i, j) is a cell located on a square of size 2^ℓ of the paving, then $2^\ell < i$ and $2^\ell < j$.*

Proof. The squares of size 2^ℓ are all located by construction on the region where both coordinates i and j are greater than or equal to $1 + 2 + 2^2 + \cdots + 2^{\ell-1} = 2^\ell - 1$. The Lemma follows from this observation. □

[9] It is of course the row corresponding to $j = 1$ which is drawn on the bottom in Figure 2.1.

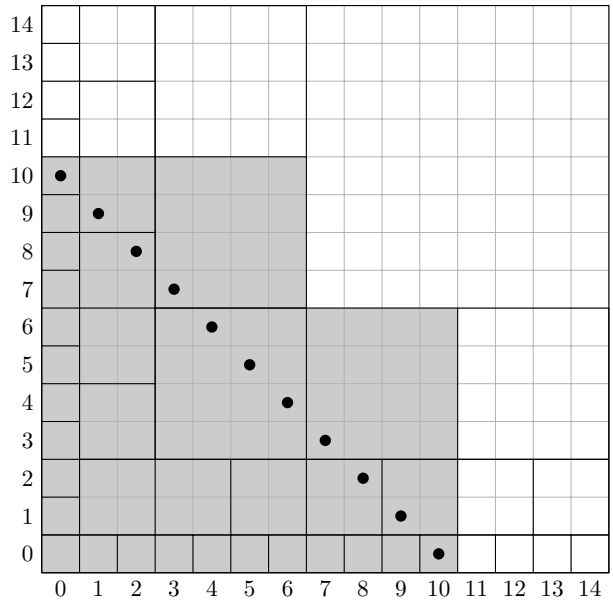

Figure 2.1: Relaxed multiplication scheme

As an example, look at the cell $(7, 3)$ on Figure 2.1; it is the bottom left corner of a square of size 4. Let us denote by $S_{i,j}$ the square of the paving with bottom left corner located at position (i, j). For such any (i, j), we define:

$$C_{i,j}(t) = \left(x_i + tx_{i+1} + \cdots + t^{\ell} x_{i+2^{\ell}-1}\right) \times \left(y_j + ty_{j+1} + \cdots + t^{\ell} y_{j+2^{\ell}-1}\right) \in \mathbb{Z}[t]$$

where 2^{ℓ} is the size of $S_{i,j}$ and t is a new formal variable. We are now ready to explain the concrete idea behind the relaxed multiplication algorithm. In the `next` procedure, when we encounter a pair $(i, j = n-i)$ we distinguish between two cases:

- if (i, j) is the position of a cell located at the bottom left corner of square of the paving, we add the contribution of $C_{i,j}(p)$ (instead of adding the sole contribution of $x_i y_j$),
- otherwise, we do nothing (since the contribution of $x_i y_j$ has already been taken into account previously).

Here are several several important remarks. First, by Lemma 2.7, the computation of the $C_{i,j}(p)$'s in the first step only requires the digits of x and y up to the position n. Second, notice that listing all the pairs (i, j) fitting into the first case is easy. Indeed, notice that (i, j) is the bottom left corner of a paving square of size 2^{ℓ} if and only if i and j are both congruent to $2^{\ell} - 1$ modulo 2^{ℓ} and one of them is actually equal to $2^{\ell} - 1$. Hence, for a given nonnegative integer ℓ, there exist at most two such cells (i, j) for which $i + j = n$ and they are moreover given by explicit formulas.

At that point, one may wonder why we have introduced the polynomials $C_{i,j}(t)$'s instead of working only with the values $C_{i,j}(p)$'s. The reason is technical: this modification is needed to get a good control on the size of the carries[10]. We will elaborate on this later (see Remark 2.8 below). By introducing the construction $\mathrm{x}[i, \ldots, j]$ for giving access to the polynomial

$$x_i + x_{i+1}t + x_{i+2}t^2 + \cdots + x_j t^{j-i} \in \mathbb{Z}[t]$$

the above ideas translate into the concrete algorithm of Figure 2.2.

We now briefly sketch its complexity analysis. The first observation is that the while loop of the `next` method of Figure 2.2 is repeated until ℓ reaches the 2-adic valuation of $n + 2$. Writing

[10] Another option, used in [7], would be to build an algorithm for performing operations on integers written in base p. This can be achieved for instance using Kronecker substitution [33, Corollary 8.27].

```
class x_mul_y(RelaxedPAdicInteger):
    # Additional variable
    carry                # variable in ℤ[t] for storing the current carry

    def next(self):    # compute the next digit and append it to the list
        n = len(self.digits)
        m = n + 2; ℓ = 0; s = 0
        while m > 1:
            # The contribution of the first square of size 2^ℓ
            s += x[2^ℓ - 1, ..., 2^(ℓ+1) - 2]
               * y[(m-1)*2^ℓ - 1, ..., m*2^ℓ - 2]
            # The contribution of the second square
            if m > 2:     # Case where the two squares are indeed not the same
                s += y[2^ℓ - 1, ..., 2^(ℓ+1) - 2]
                   * x[(m-1)*2^ℓ - 1, ..., m*2^ℓ - 2]
            if m is odd: break
            m = m // 2
            ℓ += 1
        s += self.carry
        self.digits[n] = s(0) % p
        self.carry = (s(0) // p) + (s // t)
```

Figure 2.2: An implementation of the relaxed multiplication

$v = \mathrm{val}_2(n+2)$, the total complexity for the execution of the while loop therefore stays within:

$$\sum_{\ell=0}^{v} \tilde{O}(2^\ell \log p) = \tilde{O}(2^v \log p)$$

if we use fast algorithms for polynomial multiplication [33, §8]. Moreover at the end of the while loop, the degree of s remains bounded by 2^v and its coefficients have all at most $O(\log(np))$ binary digits. The second step, which is more technical, consists in bounding the size of the carry. The key point is to notice that after the computation of the $(n-1)$-st digit, the carry takes the form:

$$c + (c_0 + c_1 t + c_2 t^2 + \cdots + c_n t^n)$$

where the c_k's come from the contribution of the cells (i', j') with $i' + j' \geqslant n$ that have been already discovered (*i.e.* belonging to a paving square $S_{i,j}$ for which $C_{i,j}(t)$ has already been computed) and c comes from the carries appearing while computing the t^{n-1}-term of the product $(x_0 + x_1 t + x_2 t^2 + \cdots + x_n t^n) \times (y_0 + y_1 t + y_2 t^2 + \cdots + y_n t^n)$. Once this has been noticed, we deduce that the c_k's are all bounded from above by $n \cdot (p-1)^2$ while:

$$c \leqslant \frac{1}{p^n} \sum_{i'+j' \leqslant n} x_{i'} y_{j'} \leqslant np \cdot (p-1).$$

Hence the carry itself is a polynomial of degree at most n and each of its coefficients has at most $O(\log(pn))$ binary digits.

Remark 2.8. While the growth of the coefficients of the carry remains under control, its value at p may — and does — grow exponentially fast with respect to n. That is the reason why we had to switch to polynomials.

Noticing finally that the division by t on polynomials is free (it is just a shift of the coefficients), we find that the last three lines of the `next` method displayed on Figure 2.2 have a cost of

$\tilde{O}(2^v \log(np))$ bit operations. The total complexity of the computation of the n-th digit then stays within $\tilde{O}(2^v \log(np))$ bit operations as well.

Summing up all these contributions up to N, we find that the relaxed multiplication algorithm computes the first N digits of a product in $\tilde{O}(N \log p)$ bit operations. We refer to [7] for a more precise complexity analysis (of a slightly different version of the algorithm we have presented here).

Computation of fixed points

A quite interesting application of the relaxed approach is the possibility to define and compute very easily fixed points of contraction mappings defined over $\mathbb{Z}_p$ (or more generally $\mathbb{Z}_p^d$ for some d). Before going into this, we need to define the notion of *function* over relaxed p-adic numbers. Recall that a relaxed p-adic integer is an instance of `RelaxedPAdicInteger`. A function over relaxed p-adic numbers can then be defined as a class deriving from `RelaxedPAdicInteger` endowed with a constructor accepting x as parameter and constructing $F(\mathrm{x})$. Actually we have already seen such examples: the classes `x_plus_y` and `x_mul_y` we have designed before fit exactly in this framework. Below is yet another easy example modeling the function $s : x \mapsto 1 + px$.

```
class S(RelaxedPAdicInteger):
    def next(self):
        n = len(self.digits)
        if n == 0: self.digits[0] = 1
        else:      self.digits[n] = x[n-1]
```

Definition 2.9. Let F be a function defined over relaxed p-adic integers. We say that F is a *contraction* if its `next` method computes the n-th digit by making only use of the first $n-1$ digits of x.

Concretely F is a contraction if its `next` method never calls x[i] for some $i \geqslant n$ when n denotes (as usual) the position of the digit the `next` method is currently computing. For example, the function `S` introduced above is a contraction.

Remark 2.10. If F is a contraction modeling a function $f : \mathbb{Z}_p \to \mathbb{Z}_p$, the first n digits of $f(x)$ depends only on the first $n-1$ digits of x. This implies that f is a $\frac{1}{p}$-contraction in the usual sense:

$$|f(x) - f(y)| \leqslant \tfrac{1}{p} \cdot |x - y|.$$

Note that the converse is not necessarily true: it may happen that f is indeed a contraction but is modeled by the function F which misses this property.

If F is a contraction modeling a function $f : \mathbb{Z}_p \to \mathbb{Z}_p$, the Banach fixed point Theorem implies that f has a unique fixed point in $\mathbb{Z}_p$. Moreover this sequence is the limit of any sequence $(x_n)_{n \geqslant 0}$ solution to the recurrence $x_{n+1} = f(x_n)$. This property translates immediately in the world of programs: it says that we can compute the fixed point of f just by replacing each occurrence of x by the current instance of the class (`self`) in the `next` function of F. As an example, applying this treatment to the function S, we get:

```
class SFixed(RelaxedPAdicInteger):
    def next(self):
        n = len(self.digits)
        if n == 0: self.digits[0] = 1
        else:      self.digits[n] = self[n-1]
```

which is[11] a relaxed p-adic number representing the fixed point of s, namely $\frac{1}{1-p}$.

[11] Or more precisely: "whose any instance is"

Division

The above toy example has a real interest because it can be generalized to handle arbitrary divisions $a \div b$ where b is invertible in $\mathbb{Z}_p$. We assume first that b is congruent to 1 modulo p, *i.e* $b = 1 + pb'$ for some $b' \in \mathbb{Z}_p$. The assumption means that the 0-th digit of b is 1; as for the digits of b', they are those of b shifted by one position. We introduce the affine function $f : \mathbb{Z}_p \to \mathbb{Z}_p$ defined by:

$$f(x) = a + (1 - b)x = a - pb'x.$$

It is a contraction which can be modeled (using relaxed multiplication and relaxed subtraction) by a contraction F defined over relaxed p-adic integers. Applying the recipe discussed above, we then build a new relaxed p-adic integer that models the unique fixed point of x, which is $\frac{a}{1+pb'} = \frac{a}{b}$. We have therefore computed the division of a by b in the relaxed world.

For a general b, one can proceed as follows. Let b_0 be the 0-th digit of b. We compute an integer c such that $b_0 c \equiv 1 \pmod{p}$ and view c as a p-adic number. Since $b \equiv b_0 \pmod{p}$, we get $bc \equiv 1 \pmod{p}$ as well. Moreover the fraction $\frac{a}{b}$ rewrites $\frac{a}{b} = \frac{ac}{bc}$ and we can now apply the method above. Using this techniques, a general relaxed division reduces to one inversion modulo p and two relaxed multiplication for the preparation plus one more relaxed multiplication and one relaxed subtraction for the computation of the fixed point.

2.3 Floating-point arithmetic

The most common representation of real numbers on computers is by far floating-point arithmetic which is normalized by the IEEE 754-2008 Standard [47, 63]. Let us recall very briefly that the rough idea behind the floating-point arithmetic is to select a finite fixed subset $\mathcal{R}$ of $\mathbb{R}$ (the so-called set of *floating-point numbers*) on which we define the operators $\oplus$ and $\otimes$ which are supposed to mimic the usual addition and multiplication of reals.

In the p-adic world, this approach has an analogue which was first proposed (as far as we know) in a unpublished note by Kedlaya and Roe in 2008 [51]. Unfortunately it seems that it has never been implemented yet[12]. The aim of the subsection is to report on Kedlaya and Roe's proposal.

2.3.1 Definition of p-adic floating-point numbers

The construction of p-adic floating-point numbers depends on the initial choice of three positive integers N, $e_{\min}$, $e_{\max}$ that we fix now until the end of §2.3. We assume $e_{\min} < e_{\max}$ and $N \geqslant 1$. The integer N is called the *precision* of the system while $e_{\min}$ and $e_{\max}$ are respectively the minimal and the maximal *exponent* of the system.

Definition 2.11. A *p-adic floating-point number* is either

- a p-adic number $x \in \mathbb{Q}_p$ of the shape $p^e s$ with the conditions:

$$e_{\min} \leqslant e \leqslant e_{\max} \quad ; \quad -\tfrac{p^N-1}{2} < s \leqslant \tfrac{p^N-1}{2} \quad ; \quad \gcd(s, p) = 1 \tag{2.6}$$

- the number $0 \in \mathbb{Q}_p$,
- the special symbol ∞,
- the special symbol NaN (not a number).

Note that the conditions (2.6) imply that a p-adic floating-point number $x \in \mathbb{Q}_p$ of the shape $p^e s$ cannot be zero and moreover that e and s are uniquely determined by x. Indeed, since s is prime with p, it cannot vanish and e has to be equal to the p-adic valuation of x. It is thus determined by x and, consequently, so is $s = p^{-e}\, x$.

[12] Although an implementation was very recently proposed by Roe for integration in SageMath; see https://trac.sagemath.org/ticket/20348.

Definition 2.12. Given a p-adic floating-point number x of the shape $x = p^e s$,

- the integer e is called the *exponent* of x,
- the integer s is called the *significand* (or sometimes the *mentissa*) of x.

Representation of p-adic floating-point numbers.

We represent p-adic floating-point numbers on machine by pairs of integers as follows:

- the p-adic number $p^e s$ is represented by (e, s),
- the p-adic number 0 is represented by $(e_{\text{max}}, 0)$,
- the symbol ∞ is represented by $(e_{\text{min}}-1, 1)$,
- the symbol NaN is represented by $(e_{\text{min}}-1, 0)$.

Remark 2.13. The above definitions omit several components of the IEEE standard. First, they do not include subnormal numbers. These are meant to allow for gradual underflow to zero, which is reasonable to provide when working in base 2 with relatively small e. However, for e as large as the bit length of a modern machine word (*i.e.* no less than 32), this benefit is sufficiently minor that it does not outweigh the added complexity needed to handle subnormal numbers properly. Second, the above definitions do not include signed infinities, because p-adic numbers do not have a meaningful notion of sign. For similar reasons, the comparison operators $<, \leqslant, >, \geqslant$ are not defined for p-adic numbers. Third, they do not provide for multiple types of NaN's. For instance, in the IEEE specification, a distinction is made between signaling NaN's (those which raise an exception upon any operation) and quiet NaN's.

For some computations, it may be useful to allow some p-adic floating-point numbers to be represented in more than one way. We thus define an operation called *normalization* on the set of pairs of integers (e, s) as follows:

- if $e < e_{\text{min}}$ and $s = 0$, return $(e_{\text{min}}-1, 0)$ (which represents NaN);
- if $e < e_{\text{min}}$ and $s \neq 0$, return $(e_{\text{min}}-1, 1)$ (which represents ∞);
- if $e > e_{\text{max}}$, return $(e_{\text{max}}, 0) = 0$ (which represents 0);
- if $e_{\text{min}} \leqslant e \leqslant e_{\text{max}}$ and $s = 0$, return $(e_{\text{max}}, 0) = 0$;
- if $e_{\text{min}} \leqslant e \leqslant e_{\text{max}}$ and $s \neq 0$, let $k \in \{0, \ldots, m-1\}$ be the largest integer such that p^k divides s and return the normalization of $(e + k, s')$ where s' is the unique integer such that $-\frac{p^N-1}{2} < s' \leqslant \frac{p^N-1}{2}$ and $s' \equiv p^{-k} m \pmod{p^N}$.

For many operations, it is safe to leave an unnormalized pair (e, s) as long as $s \neq 0$ and $e_{\text{min}} \leqslant e \leqslant e_{\text{max}} - \text{val}_p(s)$ since $p^e s$ is then guaranteed to equal the value of the normalization of (e, s). However, one should normalize before testing for equality.

2.3.2 Operations on p-adic floats

Let $\mathbb{Q}_p^{\text{FP}}$ be the set of p-adic floating-point numbers. Define also $\mathbb{Q}'_p = \mathbb{Q}_p \sqcup \{\infty, \text{NaN}\}$. Clearly $\mathbb{Q}_p^{\text{FP}} \subset \mathbb{Q}'_p$.

Lemma 2.14. *Given $x \in \mathbb{Q}_p$ with $e_{\text{min}} \leqslant \text{val}(x) \leqslant e_{\text{max}}$, there exists a unique element $y \in \mathbb{Q}_p \cap \mathbb{Q}_p^{\text{FP}}$ such that $|x - y| \leqslant |x| \cdot p^{-N}$.*

Proof. We write $x = p^v \cdot x_0$ where $v = \text{val}(x)$ and x_0 is invertible in $\mathbb{Z}_p$. Set $e = v$ and let s be the unique integer of the range $\left(-\frac{p^N-1}{2}, \frac{p^N-1}{2}\right]$ which is congruent to x_0 modulo p^N. Define $y = p^e s$; it is a p-adic floating-point number by the assumptions of the Lemma. Moreover:

$$|x - y| = p^{-v} \cdot |x_0 - s| \leqslant p^{-v-N} = |x| \cdot p^{-N}.$$

The existence is proved.

As for uniqueness, remark first that $y = 0$ cannot do the job. Assume now that $y' = p^{e'}s'$ is a p-adic floating-point number written in normalized form which satisfies the required inequality. First notice that $e' = v$ necessarily. Indeed, from $e' > v$, we would deduce $|x-y'| = |x| > |x| \cdot p^{-N}$. Similarly from $e_i < v$, we would get $|x - y'| = p^{-e_1} > |x|$. Consequently $|x - y'| = |x| \cdot |x_0 - s'|$ and our assumption on y' translates to the congruence $x_0 \equiv s' \pmod{p^N}$. Therefore we derive $s \equiv s' \pmod{p^N}$ and then $s = s'$ since both s and s' have to lie in the interval $\big(-\frac{p^N-1}{2}, \frac{p^N-1}{2}\big]$. Hence $y = y'$ and we are done. □

Definition 2.15. The *rounding function* is the function $o : \mathbb{Q}'_p \to \mathbb{Q}_p^{\text{FP}}$ defined as follows:

- if $x \in \{\infty, \text{NaN}\}$, then $o(x) = x$;
- if $x \in \mathbb{Q}_p$ and $\text{val}(x) < e_{\text{min}}$, then $o(x) = \infty$ (*overflow*);
- if $x \in \mathbb{Q}_p$ and $\text{val}(x) > e_{\text{max}}$, then $o(x) = 0$ (*underflow*);
- if $x \in \mathbb{Q}_p$ and $e_{\text{min}} \leqslant \text{val}(x) \leqslant e_{\text{max}}$, then $o(x)$ is the unique element of $\mathbb{Q}_p^{\text{FP}}$ such that $|x - o(x)| \leqslant |x| \cdot p^{-N}$.

Remark 2.16. We emphasize that overflow (resp. underflow) occurs when e is small (resp. large) which is the exact opposite of the real case. This is *not* a typo; the reason behind that is just that, in the p-adic world, p^N goes to 0 when N goes to $+\infty$.

Remark 2.17. Note that the uniqueness in the last case means that there is no analogue of the notion of a rounding mode in real arithmetic.

Given any k-ary function $f : (\mathbb{Q}'_p)^k \to \mathbb{Q}'_p$, we define its *compliant approximation* $f_{\text{FP}} : (\mathbb{Q}_p^{\text{FP}})^k \to \mathbb{Q}_p^{\text{FP}}$ by:

$$f_{\text{FP}}(x_1, \ldots, x_k) = o\big(f(x_1, \ldots, x_k)\big).$$

We are now in position to specify the four basic arithmetic operations (addition, subtraction, multiplication, division) on $\mathbb{Q}_p^{\text{FP}}$. In order to do so, we need first to extend them on $\mathbb{Q}'_p$. Let us agree to set (here $\div$ stands for the division operator):

$$\begin{array}{ll}
\forall x \in \mathbb{Q}'_p, & x \pm \text{NaN} = \text{NaN} \pm x = \text{NaN} \\
\forall x \in \mathbb{Q}'_p \backslash \{\text{NaN}\}, & x \pm \infty = \infty \pm x = \infty \\
\forall x \in \mathbb{Q}'_p, & x \times \text{NaN} = \text{NaN} \times x = \text{NaN} \\
\forall x \in \mathbb{Q}'_p \backslash \{0, \text{NaN}\}, & x \times \infty = \infty \times x = \infty \\
 & 0 \times \infty = \infty \times 0 = \text{NaN} \\
\forall x \in \mathbb{Q}'_p, & x \div \text{NaN} = \text{NaN} \div x = \text{NaN} \\
\forall x \in \mathbb{Q}'_p \backslash \{0, \text{NaN}\}, & x \div 0 = \infty, \quad 0 \div 0 = \text{NaN} \\
\forall x \in \mathbb{Q}'_p \backslash \{\infty, \text{NaN}\}, & x \div \infty = 0, \quad \infty \div \infty = \text{NaN}
\end{array}$$

Addition, subtraction, multiplication and division on floating-point p-adic numbers are, by definition, the operations $+_{\text{FP}}$, $-_{\text{FP}}$, $\times_{\text{FP}}$ and $\div_{\text{FP}}$ respectively.

Let us examine concretely how the computation goes at the level of p-adic floating-point numbers. We start with multiplication with is the easiest operator to handle. The formula we derive immediately from the definition is:

$$\begin{array}{rll}
(p^{e_1}s_1) \times_{\text{FP}} (p^{e_2}s_2) = p^{e_1+e_2} \cdot (s_1 s_2 \bmod p^N) & & \text{if } e_1 + e_2 \leqslant e_{\text{max}} \\
= 0 & & \text{otherwise}
\end{array}$$

where $a \bmod p^N$ is the unique representation of a modulo p^N lying in the interval $\big(-\frac{p^N-1}{2}, \frac{p^N-1}{2}\big]$. We emphasize moreover that if (e_1, s_1) and (e_2, s_2) are both written in normalized form then

so is $(e_1 + e_2, s_1 s_2 \bmod p^N)$ (under the assumption $e_1 + e_2 \leqslant e_{\max}$). Indeed the product of two numbers which are coprime to p is coprime to p as well. The case of the division is similar:

$$\begin{aligned}(p^{e_1} s_1) \div_{\text{FP}} (p^{e_2} s_2) &= p^{e_1 - e_2} \cdot (s_1 s_2^{-1} \bmod p^N) && \text{if } e_1 - e_2 \geqslant e_{\min} \\ &= \infty && \text{otherwise.}\end{aligned}$$

The cases of addition and subtraction are more subtle because they can introduce cancellations of the rightmost digits. When $e_1 \neq e_2$, this cannot happen and the formula writes as follows:

$$\begin{aligned}(p^{e_1} s_1) +_{\text{FP}} (p^{e_2} s_2) &= p^{e_1} \cdot (s_1 + p^{e_2 - e_1} s_2 \bmod p^N) && \text{if } e_1 < e_2 \\ &= p^{e_2} \cdot (p^{e_1 - e_2} s_1 + s_2 \bmod p^N) && \text{if } e_2 < e_1.\end{aligned}$$

On the contrary, when $e_1 = e_2$, the naive formula $(p^e s_1) +_{\text{FP}} (p^e s_2) = p^e \cdot (s_1 + s_2 \bmod p^N)$ might fail because $s_1 + s_2$ can have a huge valuation. Instead we introduce $v = \mathrm{val}_p(s_1 + s_2)$ and the correct formula writes:

$$\begin{aligned}(p^e s_1) +_{\text{FP}} (p^e s_2) &= p^{e+v} \cdot \left(\tfrac{s_1 + s_2}{p^v} \bmod p^N\right) && \text{if } v \leqslant e_{\max} - e \\ &= 0 && \text{otherwise.}\end{aligned}$$

As an alternative, we may prefer using non-normalized representations of p-adic floating-point numbers; it is indeed an option but it requires to be *very* careful with underflows.

2.4 Comparison between paradigms

We have just presented three quite different ways of thinking of p-adic numbers from the computational point of view. Of course, each of them has its own advantages and disadvantages regarding flexibility, rapidity, accuracy, *etc.* In this subsection we compare them, trying as much as possible to support our arguments with many examples.

2.4.1 Zealous arithmetic vs. lazy/relaxed arithmetic

The main difference between zealous and lazy/relaxed arithmetic is certainly that they are designed for different uses! In the zealous perspective, the inputs are given with a certain precision which is supposed to be fixed once for all: one can imagine for instance that the inputs come from measures[13] or that computing them requires a lot of effort and is then not an option. So the precision on the inputs is immutable and the goal of zealous arithmetic is to propagate the precision to the output (trying to be as sharp as possible).

On the other hand, the point of view of lazy/relaxed arithmetic is the exact opposite: instead of fixing the precision on the inputs, we fix a *target* precision and we propagate it back to the inputs. Indeed suppose, as an easy example, that we have to evaluate the product xy. In the lazy/relaxed world, this is done by creating and returning the program `x_mul_y` *without running it*. The execution (*i.e.* the actual computation of the product) is postponed until somebody asks for the value of xy at a certain precision $O(p^N)$ (through the call `x_mul_y`$[N]$). At that moment, the program runs and infers the needed precision on the inputs x and y.

Except maybe in very particular situations, the point of view of lazy/relaxed arithmetic is certainly much closer to the needs of the mathematician. Indeed, it is quite rare to have to work with p-adic numbers that do not come from a previous computation. Mathematicians are more relaxed than zealous!

Apart from that, still regarding precision, the zealous and the lazy/relaxed approaches are both based on the arithmetic of intervals and, for this reason, they exhibit similar behaviors in the following sense. Assume that we are given a certain function $f : \mathbb{Z}_p \to \mathbb{Z}_p$ to be evaluated and define the two numbers ℓ_1 and ℓ_2 (depending *a priori* on extra parameters) as follows:

[13] Through I have to confess that I have never seen a relevant physical p-adic measure yet.

- when we ask for the value of $f(a + O(p^N))$, the zealous approach answers $b + O(p^{N-\ell_1})$ (loss of ℓ_1 digits of precision),
- when we ask for the value of $f(a)$ at precision $O(p^N)$, the lazy/relaxed technique requires the computation of a at precision $O(p^{N+\ell_2})$.

It turns out that ℓ_1 and ℓ_2 do not very much depend neither on a, nor on N. Moreover they are almost equal[14]. On the other hand, we underline that this value $\ell_1 \approx \ell_2$ is very dependent on the function f and even on the algorithm we use for the evaluation. We will present and discuss this last remark in §2.4.3 below.

The lazy/relaxed arithmetic has nevertheless several more or less annoying disadvantages. First of all, it is certainly more difficult to implement efficiently (although the implementation in MATHEMAGIX [45] is very well polished and quite competitive; we refer to [7] for timings). It is also supposed to be a bit slower; for instance, the relaxed multiplication looses asymptotically a factor $\log N$ (for the precision $O(p^N)$) compared to the zealous approach. Another more serious issue is memory. Indeed the relaxed arithmetic needs to keep stored all intermediate results by design. As an easy toy example, let us have a quick look at the following function

```
def nth_term(n)
    u = a
    for i in 1,2,...,n:
        u = f(u)
    return u
```

that computes the n-th term of a recursive sequence defined by its initial value $u_0 = a$ and the recurrence $u_{i+1} = f(u_i)$. Here a and f are given data. In zealous arithmetic, the above implementation requires to store only one single value of the sequence $(u_i)_{i\geqslant 0}$ at the same time (assuming that we can rely on a good garbage collector); indeed at the i-th iteration of the loop, the value of u_i is computed and stored in the variable `u` while the previous value of `u`, *i.e.* the value of u_{i-1}, is destroyed. On the other hand, in relaxed arithmetic, the variable representing u_i stores the definition of u_i, including then the value of u_{i-1}. The relaxed p-adic number returns by the function `nth_term` is then a huge structure in which all the relaxed p-adic numbers $u_1, u_2, \ldots, u_n$ have to appear. When we will then ask for the computation of the first N digits of u_n, the relaxed machinery will start to work and compute — and store — the values of all the u_i's at the desired precision. This is the price to pay in order to be able to compute one more digit of u_n without having to redo many calculations.

2.4.2 Interval arithmetic vs. floating-point arithmetic

p-adic floating-point arithmetic has a very serious limitation for its use in mathematics: the results it outputs are *not proved* and even often wrong! Precisely, the most significand digits of the output are very likely to be correct while the least significand digits are very likely to be incorrect... and we have *a priori* no way to know which digits are correct and which ones are not. On the other hand, at least in the real setting, interval arithmetic is known for its tendency to yield pessimistic enclosures. What about the p-adic case? At first, one might have expected that ultrametricity may help. Indeed ultrametricity seems to tell that rounding errors do not accumulate as it does in the real world. Unfortunately this simple rationale is too naive: in practice, p-adic interval arithmetic does overestimate the losses of precision exactly as real interval arithmetic does.

The causes are multiple and complex but some of them can be isolated. The first source of loss of precision comes from the situation where we add two p-adic numbers known at different precision (*e.g.* two p-adic numbers of different sizes). As a toy example, consider the function $f : \mathbb{Z}_p^2 \to \mathbb{Z}_p^2$ mapping (x, y) to $(x + y, x - y)$ (it is a similarity in the p-adic plane) and suppose

[14] We refer to §3.2.1 for a clarification of this statement based on a theoretical study (which is itself based on the theory of p-adic precision we shall develop in §3.1).

that we want to evaluate $f \circ f$ on the entry $x = 1 + O(p^2)$, $y = 1 + O(p^{20})$. We then code the following function:

```
def F(x,y):
    return x+y, x-y
```

and call F(F(x,y)). Let us have a look at precision. According to Proposition 2.3, the first call F(x,y) returns the pair $(2 + O(p^2), O(p^2))$ and the final result is then $(2 + O(p^2), 2 + O(p^2))$. On the other hand, observing that $f \circ f$ is the mapping taking (x, y) to $(2x, 2y)$. Using this, we end up with:

$$\begin{aligned} f \circ f(x,y) &= \big(2 + O(p^2), 2 + O(p^{20})\big) \quad \text{if } p > 2 \\ &= \big(2 + O(2^3), 2 + O(2^{21})\big) \quad \text{if } p = 2 \end{aligned}$$

which is much more accurate. Interval arithmetic misses the simplification and consequently loses accuracy.

A second more subtle source of inaccuracy comes from the fact that interval arithmetic often misses *gain* of precision. A very basic example is the computation of px; written this way, the absolute precision increases by 1. However if, for some good reason[15], the product px does not appear explicitly but instead is computed by adding x to itself p times, interval arithmetic will not see the gain of precision. It turns out that similar phenomena appear for the multiplicative analogue of this example, *i.e.* the computation of x^p for $x = a + O(p^N)$. For simplicity assume that $\mathrm{val}(x) = 0$ and $N \geqslant 1$. Using again Proposition 2.3, we find that zealous arithmetic leads to the result $x^p = a^p + O(p^N)$. However, surprisingly, this result is not optimal (regarding precision) as shown by the next lemma.

Lemma 2.18. *If $a \equiv b \pmod{p^N}$, then $a^p \equiv b^p \pmod{p^{N+1}}$.*

Proof. Write $b = a + p^N c$ with $c \in \mathbb{Z}_p$ and compute:

$$b^p = (a + p^N c)^p = \sum_{i=0}^{p} \binom{p}{i} a^{p-i} (p^N c)^i.$$

When $i \geqslant 2$, the corresponding term is divisible by p^{2N} while when $i = 1$, it equals $p^{N+1} a^{p-1} c$ and is therefore apparently divisible by p^{N+1}. As a consequence $b^p \equiv a^p \pmod{p^{N+1}}$ as claimed. □

According to Lemma 2.18, the correct precision for x^p is (at least) $O(p^{N+1})$ which is not detected by arithmetic interval. SageMath [77] fixes this issue by an *ad-hoc* implementation of the power function which knows about Lemma 2.18. However similar behaviors happen in quite a lot of different other situations and they of course cannot be all fixed by *ad-hoc* patches.

Remark 2.19. Recall that we have seen in §2.1.3 that the computation of square roots in $\mathbb{Q}_2$ looses one digit of precision. It is absolutely coherent with the result above which says that the computation of squares gains one digit of precision.

As a conclusion, a very common situation where p-adic floating-point arithmetic can be very helpful is that situation where the mathematician is experimenting, trying to understand how the new mathematical objects he has just introduced behave. At the moment, he does not really need proofs[16]; he needs fast computations and plausible accuracy, exactly what p-adic floating-point arithmetic can offer.

2.4.3 Comparison by everyday life examples

We analyze many examples and compare for each of them the accuracy we get with p-adic floating-point arithmetic on the one hand and with interval arithmetic (restricting ourselves to the zealous approach for simplicity) on the other hand. Examples are picked as basic and very common primitives in linear algebra and commutative algebra.

[15] For example, imagine that all the values $x, 2x, 3x, \ldots, px$ are needed.

[16] Well, it is of course often better to have proofs... but it is maybe too early.

Determinant

Our first example concerns the computation of the determinant of a square matrix with entries in $\mathbb{Z}_p$. In order to be as accurate as possible, we shall always use a division-free algorithm (see for instance [10] or [48] and the references therein for faster solutions). When M is a generic matrix there is actually not so much to say: if the entries of the matrix are given with N significand digits, the same holds for the determinant and this precision is optimal. Nonetheless, quite interesting phenomena show up for matrices having a special form. In the sequel, we will examine the case of matrices M of the form $M = PDQ$ where $P, Q \in \mathrm{GL}_d(\mathbb{Z}_p)$ and D is a diagonal matrix with diagonal entries $p^{a_1}, \ldots, p^{a_d}$. We assume that the three matrices P, D and Q are given at precision $O(p^N)$ for some N. Here is a concrete example (picked at random) with $p = 2$, $d = 4$ and $N = 10$:

$$a_1 = 0 \quad ; \quad a_2 = 2 \quad ; \quad a_3 = 3 \quad ; \quad a_4 = 5$$

$$P = \begin{pmatrix} \ldots 1111100100 & \ldots 0110110101 & \ldots 0101011000 & \ldots 1101010001 \\ \ldots 1010001101 & \ldots 0110011001 & \ldots 1101111000 & \ldots 1010100100 \\ \ldots 1011101111 & \ldots 0100100111 & \ldots 0000111101 & \ldots 0010010001 \\ \ldots 1011111001 & \ldots 1000100011 & \ldots 1100100110 & \ldots 0111100011 \end{pmatrix}$$

$$Q = \begin{pmatrix} \ldots 1010110001 & \ldots 0010011111 & \ldots 1010010010 & \ldots 1010001001 \\ \ldots 1111101111 & \ldots 0111100101 & \ldots 0110101000 & \ldots 0111100000 \\ \ldots 0111110100 & \ldots 0010010101 & \ldots 0000101111 & \ldots 1001100010 \\ \ldots 0101111111 & \ldots 0101110111 & \ldots 1110000011 & \ldots 1110000110 \end{pmatrix}$$

so that:

$$M = \begin{pmatrix} \ldots 0101110000 & \ldots 0011100000 & \ldots 1011001000 & \ldots 0011000100 \\ \ldots 1101011001 & \ldots 1101000111 & \ldots 0111001010 & \ldots 0101110101 \\ \ldots 0111100011 & \ldots 1011100101 & \ldots 0010100110 & \ldots 1111110111 \\ \ldots 0000111101 & \ldots 1101110011 & \ldots 0011010010 & \ldots 1001100001 \end{pmatrix} \tag{2.7}$$

Remark 2.20. Each entry of M is known at precision $O(2^{10})$ as well. Indeed any permutation of M by an element $H \in 2^{10} M_4(\mathbb{Z}_2)$ can be induced by a perturbation of D by the element $P^{-1}HQ^{-1}$ which lies in $2^{10} M_4(\mathbb{Z}_2)$ as well because P and Q have their inverses in $M_4(\mathbb{Z}_2)$.

Here are the values for the determinant of M computed according to the two strategies we want to compare:

$$\begin{array}{ll} \text{Interval (zealous) arithmetic:} & \det M = O(2^{10}) \\ \text{Floating-point arithmetic:} & \det M = 2^{10} \times \ldots 0001001101 \end{array}$$

We observe the interval arithmetic does not manage to decide whether $\det M$ vanishes or not. On the other hand, the floating-point approach outputs a result with 10 significand digits by design; the point is that we do not know *a priori* whether these digits are correct or not. In our particular case, we can however answer this question by computing $\det M$ as the product $\det P \cdot \det D \cdot \det Q = 2^{10} \det P \cdot \det Q$ using zealous arithmetic. We find this way:

$$\det M = 2^{10} \times \ldots 01101$$

which means that the result computed by floating-point arithmetic has (at least) 5 correct digits. This is actually optimal as we shall see later in §3.2.2.

Characteristic polynomial

We now move to the characteristic polynomial keeping first the same matrix M. Here are the results we get:

Interval (zealous) arithmetic:

$$\begin{aligned}\chi_M(X) = X^4 + &\quad (\ldots 0001000010)\, X^3 + &\quad (\ldots 1000101100)\, X^2\\ + &\quad (\ldots 0011100000)\, X \;+ &\quad (\ldots 0000000000)\end{aligned}$$

Floating-point arithmetic:

$$\begin{aligned}\chi_M(X) = X^4 + &\quad (\ldots 00001000010)\, X^3 + &\quad (\ldots 111000101100)\, X^2\\ + &\,(\ldots 110100011100000)\, X \;+ &\,(2^{10} \times \ldots 0001001101)\end{aligned}$$

where, in the second case, the correct digits are written in purple[17]. We observe again that, although the computation is not proved mathematically, the floating-point arithmetic outputs more accurate results. As we shall see later (see §3.2.2) the accuracy it gives is even optimal for this particular example.

We consider now the matrix $N = I_4 + M$ where I_4 is the identity matrix of size 4. The computation of χ_N leads to the following results:

Interval (zealous) arithmetic:

$$\begin{aligned}\chi_N(X) = X^4 + &\quad (\ldots 0000111110)\, X^3 + &\quad (\ldots 0101101100)\, X^2\\ + &\quad (\ldots 0101001010)\, X \;+ &\quad (\ldots 0100001011)\end{aligned}$$

Floating-point arithmetic:

$$\begin{aligned}\chi_N(X) = X^4 + &\quad (\ldots 00000111110)\, X^3 + &\quad (\ldots 000101101100)\, X^2\\ + &\quad (\ldots 10101001010)\, X \;+ &\quad (\ldots 0100001011)\end{aligned}$$

On that example, we remark that interval arithmetic is as accurate as floating-point arithmetic. More interesting is the evaluation of χ_N at 1, which is nothing but the determinant of M we have already computed before. The values we get are the following:

$$\begin{aligned}&\text{Interval (zealous) arithmetic:} &&\chi_N(1) = O(2^{10})\\ &\text{Floating-point arithmetic:} &&\chi_N(1) = 0\end{aligned}$$

Although there is no surprise with interval arithmetic, we observe that floating-point arithmetic now fails to produce any correct digit.

LU factorization

LU factorization is a classical tool in linear algebra which serves as primitives for many problems. We refer to [1, §2] for an introduction to the topic. Recall briefly that a LU factorization of a matrix M is a decomposition of the form $M = LU$ where L (resp. U) is a lower triangular (resp. upper triangular) matrix. In the sequel, we require moreover that the diagonal entries of L are all equal to 1. With the normalization, one can prove that any matrix M defined over a field admits a *unique* LU factorization as soon as all its principal minors[18] do not vanish.

LU factorizations can be computed using standard Gaussian elimination: starting from M, we first multiply the first column by the appropriate scalar in order to make the top left entry equal to 1 and use it as pivot to cancel all coefficients on the first line by operating on columns. We

[17] We decided which digits are correct simply by computing at higher precision (with zealous arithmetic in order to get a guaranteed result).

[18] Recall the i-th principal minor of M is the determinant of the submatrix of M obtained by selecting the first i rows and first i columns.

then get a matrix of the shape:

$$\begin{pmatrix} 1 & 0 & \cdots & 0 \\ \star & \star & \cdots & \star \\ \vdots & \vdots & & \vdots \\ \star & \star & \cdots & \star \end{pmatrix}$$

and we continue this process with the submatrix obtained by deleting the first row and the first column. The matrix we get at the end is the L-part of the LU factorization of M.

We propose to explore the numerical stability of LU factorization *via* Gaussian elimination in the p-adic case. Let us start with an example. Consider first the same matrix M as above and write its LU factorization $M = LU$ (one can check that its principal minors do not vanish). Performing Gaussian elimination as described above within the framework of zealous arithmetic on the one hand and within the framework of floating-point arithmetic on the other hand, we end up with the following matrices:

Interval (zealous) arithmetic:

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 2^{-4} \times \ldots 001111 & 1 & 0 & 0 \\ 2^{-4} \times \ldots 010101 & \ldots 100011 & 1 & 0 \\ 2^{-4} \times \ldots 001011 & \ldots 010101 & \ldots 110 & 1 \end{pmatrix} \tag{2.8}$$

Floating-point arithmetic:

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 2^{-4} \times \ldots 1010001111 & 1 & 0 & 0 \\ 2^{-4} \times \ldots 0110010101 & \ldots 0011100011 & 1 & 0 \\ 2^{-4} \times \ldots 0101001011 & \ldots 0111010101 & 00010110110 & 1 \end{pmatrix}$$

As before the correct digits are displayed in purple. We remark once again that the accuracy of p-adic floating-point arithmetic is better than that of interval arithmetic (it is actually even optimal on that particular example as we shall see in §3.2.2). More precisely, we note that the zealous precision is sharp on the first column but the gap increases when we are moving to the right.

Remark 2.21. It is somehow classical [46, §1.4] that the entries of L and U can all be expressed as the quotient of two appropriate minors of M (Cramer-like formulas). Evaluating such expressions, it is possible to compute the LU factorization and stay sharp on precision within the zealous framework. The drawback is of course complexity since evaluating two determinants for each entry of L and U is clearly very time-consuming. A stable and efficient algorithm, combining the advantages of the two approaches, is designed in [16].

Bézout coefficients and Euclidean algorithm

We now move to polynomials and examine the computation of Bézout coefficients *via* the Euclidean algorithm. We pick at random two monic polynomials P and Q of degree 4 over $\mathbb{Z}_2$:

$$\begin{aligned} P = X^4 &+ (\ldots 1101111111)\, X^3 + (\ldots 0011110011)\, X^2 \\ &+ (\ldots 1001001100)\, X \ + (\ldots 0010111010) \\ Q = X^4 &+ (\ldots 0101001011)\, X^3 + (\ldots 0111001111)\, X^2 \\ &+ (\ldots 0100010000)\, X \ + (\ldots 1101000111) \end{aligned}$$

We can check that P and Q are coprime modulo p (observe that $P + Q \equiv 1 \pmod{p}$); they are therefore *a fortiori* coprime in $\mathbb{Q}_p[X]$. By Bézout Theorem, there exist two polynomials $U, V \in \mathbb{Z}_2[X]$ such that $UP + VQ = 1$. Moreover these polynomials are uniquely determined if we require $\deg U, \deg V \leqslant 3$. Computing them with the extended Euclidean algorithm (without

any kind of algorithmic optimization), we get:

Interval (zealous) arithmetic:

$$\begin{aligned} U = &\quad (\ldots 101100)\, X^3 + &&(\ldots 101100)\, X^2 \\ &+ \quad (\ldots 100)\, X \;+ &&(\ldots 1011) \\ V = &\quad (\ldots 010100)\, X^3 + &&(\ldots 100100)\, X^2 \\ &+ \quad (\ldots 100)\, X \;+ &&(\ldots 101) \end{aligned}$$

Floating-point arithmetic:

$$\begin{aligned} U = &\quad (\ldots 101011101100)\, X^3 + (\ldots 111100101100)\, X^2 \\ &+ (\ldots 100000110100)\, X \;+ \quad (\ldots 0110001011) \\ V = &\quad (\ldots 010100010100)\, X^3 + (\ldots 001011100100)\, X^2 \\ &+ (\ldots 000100111100)\, X \;+ \quad (\ldots 1111100101) \end{aligned}$$

Floating-point arithmetic provides again better accuracy, though far for being optimal this time. Indeed the theory of subresultants [82, §4.1] shows that the coefficients of U and V can be all expressed as quotients of some minors of the Sylvester matrix of (P, Q) by the resultant of P and Q, denoted hereafter by $\mathrm{Res}(P, Q)$. From the fact that P and Q are coprime modulo p, we deduce that $\mathrm{Res}(P, Q)$ does not vanish modulo p. Thus $\mathrm{val}_p(\mathrm{Res}(P, Q)) = 0$ and dividing by $\mathrm{Res}(P, Q)$ does not decrease the absolute precision according to Proposition 2.3. As a consequence, following this path and staying within the zealous framework, we can calculate the values of U and V at precision $O(2^{10})$. Here is the result we get:

$$\begin{aligned} U &= (\ldots 0011101100)\, X^3 + (\ldots 0100101100)\, X^2 + (\ldots 1101110100)\, X \;+\; (\ldots 0010001011) \\ V &= (\ldots 1100010100)\, X^3 + (\ldots 1011100100)\, X^2 + (\ldots 0110111100)\, X \;+ (\ldots 0001100101). \end{aligned}$$

We observe that the latter values (in addition to being proved) are even more accurate that the result which was computed "naively" using floating-point arithmetic. The drawback is complexity since evaluating many determinants requires a lot of time. The theory of p-adic precision we are going to introduce in the next section (see §3 and, more especially, §3.3.1) provides the tools for writing a stabilized version of Euclidean algorithm that computes provably U and V at (almost) optimal precision. We refer to [17] for more details (the material developed in §3 is necessary to read this reference).

Polynomial evaluation and interpolation

Polynomial evaluation and polynomial interpolation are very classical and useful primitives involved in many algorithms in symbolic computation. In this last paragraph, we examine how they behave from the point of view of precision. We focus on a very basic (but already very instructive) example. We consider the following two procedures:

- `evaluation`: it takes as input a polynomial $P \in \mathbb{Q}_p[X]$ of degree at most d and outputs $P(0), P(1), \ldots, P(d)$;
- `interpolation`: it takes as input a list of values $y_0, \ldots, y_d \in \mathbb{Q}_p$ and returns the interpolation polynomial $P \in \mathbb{Q}_p[X]$ of degree at most d such that $P(i) = y_i$ for $i \in \{0, 1, \ldots, d\}$.

Algorithms (and notably fast algorithms) for these tasks abound in the literature (see for instance [33, §10]). For our propose, we choose naive algorithms: we implement `evaluation` by evaluating separately the $P(i)$'s (using Hörner scheme say) and we implement `interpolation` using the method of divided differences [41, §2]. Under our assumptions (interpolation at the first integers), it turns out that it takes a particularly simple form that we make explicit now.

Define the *difference operator* Δ on $\mathbb{Q}_p[X]$ by $\Delta A(X) = A(X + 1) - A(X)$. The values taken by A at the integers are related to the values $\Delta^n A(0)$ by a simple closed formula, as shown by the next lemma.

Lemma 2.22. *For all polynomials $A \in \mathbb{Q}_p[X]$ of degree at most d, we have:*

$$A(X) = \sum_{n=0}^{d} \Delta^n A(0) \cdot \binom{X}{n} \quad \textit{where} \quad \binom{X}{n} = \frac{X(X-1)\cdots(X-n+1)}{n!}.$$

Proof. We proceed by induction on d. When $d = 0$, the lemma is trivial. We assume now that it holds for all polynomials A of degree at most d and consider a polynomial $A \in \mathbb{Q}_p[X]$ with $\deg A = d+1$. We define $B(X) = \sum_{n=0}^{d} \Delta^n A(0) \cdot \binom{X}{n}$. Remarking that $\Delta\binom{X}{n} = \binom{X}{n-1}$ for all positive integers n, we derive:

$$\Delta B(X) = \sum_{n=1}^{d} \Delta^n A(0) \cdot \binom{X}{n-1} = \sum_{n=0}^{d-1} \Delta^{n+1} A(0) \cdot \binom{X}{n}.$$

From the induction hypothesis, we deduce $\Delta B = \Delta A$ (since ΔA has degree at most d). Furthermore, going back to the definition of B, we find $A(0) = B(0)$. All together, these two relations imply $A = B$ and the induction goes. □

Remark 2.23. Lemma 2.22 extends by continuity to all continuous functions f on $\mathbb{Z}_p$. In this generality, it states that any continuous function $f : \mathbb{Z}_p \to \mathbb{Z}_p$ can be uniquely written as a convergent series of the shape:

$$f(x) = \sum_{n=0}^{\infty} a_n \cdot \binom{x}{n}$$

where the a_n's lie in $\mathbb{Z}_p$ and converge to 0 when i goes to infinity. The a_n's are moreover uniquely determined: we have $a_n = \Delta^n f(0)$. They are called the *Mahler coefficients* of f. We refer to [60] for much more details on this topic.

From Lemma 2.22, we easily derive an algorithm for our interpolation problem. Given $y_0, \ldots, y_d$, we define the "divided" differences $y_{n,i}$ for $0 \leqslant i \leqslant n \leqslant d$ by (decreasing) induction on n by $y_{d,i} = y_i$ and $y_{n-1,i} = y_{n,i+1} - y_{n,i}$. These quantities can be easily computed. Moreover, thanks to Lemma 2.22, the interpolation polynomial P we are looking for writes $P(X) = \sum_{n=0}^{d} y_{n,0}\binom{x}{n}$. This provides an algorithm for computing the interpolation polynomial.

Let us now try to apply successively `evaluation` and `interpolation` taking as input a random polynomial P of degree $d = 8$ with coefficients in $\mathbb{Z}_p$ (for $p = 2$ as usual):

$$\begin{aligned} P = {} & (\ldots 0111001110)\, X^8 + (\ldots 0101010001)\, X^7 + (\ldots 1000001100)\, X^6 \\ & + (\ldots 1010001101)\, X^5 + (\ldots 1111000100)\, X^4 + (\ldots 0011101101)\, X^3 \\ & + (\ldots 1010010111)\, X^2 + (\ldots 0011011010)\, X \ + (\ldots 0001011110) \end{aligned}$$

The results we get are:

Interval (zealous) arithmetic:

$$\begin{aligned} & (\ldots 110)\, X^8 + && (\ldots 001)\, X^7 + && (\ldots 100)\, X^6 \\ + \ & (\ldots 101)\, X^5 + && (\ldots 100)\, X^4 + && (\ldots 1101)\, X^3 \\ + \ & (\ldots 10111)\, X^2 + && (\ldots 1011010)\, X \ + && (\ldots 0001011110) \end{aligned}$$

Floating-point arithmetic:

$$\begin{aligned} & (\ldots 00001001110)\, X^8 + && (\ldots 1011010001)\, X^7 + && (\ldots 001010001100)\, X^6 \\ + \ & (\ldots 0010001101)\, X^5 + && (\ldots 000011000100)\, X^4 + && (\ldots 01011011101)\, X^3 \\ + \ & (\ldots 0010010111)\, X^2 + && (\ldots 11011011010)\, X \ + && (\ldots 00001011110) \end{aligned}$$

The result computed by floating-point arithmetic is again a bit more accurate. However this observation is not valid anymore where the degree d gets larger. Figure 2.3 shows an example with a polynomial (picked at random) of degree $d = 19$. We see that almost all digits are incorrect,

Initial polynomial:

$$\begin{aligned}
&(\ldots 0101101001)\, X^{19} + (\ldots 1101000011)\, X^{18} + (\ldots 0011001110)\, X^{17} + (\ldots 1001011010)\, X^{16} \\
+\ &(\ldots 0011100111)\, X^{15} + (\ldots 0110101110)\, X^{14} + (\ldots 0111111001)\, X^{13} + (\ldots 1011010111)\, X^{12} \\
+\ &(\ldots 0100000100)\, X^{11} + (\ldots 0000110000)\, X^{10} + (\ldots 1110101010)\, X^{9} + (\ldots 1111101100)\, X^{8} \\
+\ &(\ldots 0100010001)\, X^{7} + (\ldots 0101010000)\, X^{6} + (\ldots 0111101111)\, X^{5} + (\ldots 1100010011)\, X^{4} \\
+\ &(\ldots 0100000001)\, X^{3} + (\ldots 1000010010)\, X^{2} + (\ldots 0000100000)\, X + (\ldots 0001111110)
\end{aligned}$$

Interval (zealous) arithmetic:

$$\begin{aligned}
&O(2^{-6})\, X^{19} + O(2^{-6})\, X^{18} + O(2^{-6})\, X^{17} + O(2^{-5})\, X^{16} \\
+\ &O(2^{-6})\, X^{15} + O(2^{-6})\, X^{14} + O(2^{-6})\, X^{13} + O(2^{-5})\, X^{12} \\
+\ &O(2^{-6})\, X^{11} + O(2^{-6})\, X^{10} + O(2^{-6})\, X^{9} + O(2^{-5})\, X^{8} \\
+\ &O(2^{-4})\, X^{7} + O(2^{-3})\, X^{6} + O(2^{-2})\, X^{5} + O(2^{-1})\, X^{4} \\
+\ &(\ldots 1)\, X^{3} + (\ldots 010)\, X^{2} + (\ldots 100000)\, X + (\ldots 0001111110)
\end{aligned}$$

Floating-point arithmetic:

$$\begin{aligned}
&(2^{-3} \times \ldots 1110011011)\, X^{19} + (2^{-5} \times \ldots 0000000011)\, X^{18} + (2^{-3} \times \ldots 0001011111)\, X^{17} + (2^{-5} \times \ldots 1100111101)\, X^{16} \\
+\ &(\ldots 11111100110)\, X^{15} + (2^{-4} \times \ldots 0110100011)\, X^{14} + (2^{-2} \times \ldots 0000010011)\, X^{13} + (2^{-4} \times \ldots 1010001101)\, X^{12} \\
+\ &(2^{-3} \times \ldots 0010000011)\, X^{11} + (2^{-5} \times \ldots 0100101111)\, X^{10} + (2^{-3} \times \ldots 0000110011)\, X^{9} + (2^{-5} \times \ldots 1010101001)\, X^{8} \\
+\ &(2^{-2} \times \ldots 0010000101)\, X^{7} + (2^{-2} \times \ldots 1101100111)\, X^{6} + (\ldots 1101101111)\, X^{5} + (2^{-1} \times \ldots 0011100111)\, X^{4} \\
+\ &(\ldots 0101110101)\, X^{3} + (\ldots 11011101010)\, X^{2} + (\ldots 000000001100000)\, X + (\ldots 00001111110)
\end{aligned}$$

Figure 2.3: Evaluation and re-interpolation of a polynomial of degree 19 over $\mathbb{Z}_2$

many coefficients have negative valuations, *etc.* For this problem, floating-point arithmetic is then not well suited.

One may wonder whether another algorithm, specially designed for stability, would lead to better results. Unfortunately, the answer is negative: we shall see later (in §3.2.2) that the problem of polynomial evaluation and interpolation is *very* ill-conditioned in the p-adic setting, so that numerical methods are ineffective.

3 The art of tracking p-adic precision

In many examples presented in §2.4.3, we have often recognized similar behaviors: interval arithmetic often overestimates the losses of precision while floating-point arithmetic provides more accurate — but unproved — results. Understanding precisely the origin of these phenomena is a quite stimulating question (that has been widely studied in the real setting).

Recently Caruso, Roe and Vaccon [18] proposed a general theory for dealing with precision in the p-adic setting and this way provided powerful tools for attacking the aforementioned question. Their rough idea was to develop an analogue of interval arithmetic in higher dimensions. In other words, instead of attaching a precision $O(p^N)$ to each p-adic variable, they group variables and attach to the collection of all of them a *unique global* precision datum materialized by an "ellipsoid" in some normed p-adic vector space. The magic of ultrametricity then operates: ellipsoids are rather easy to deal with and behave very well with respect to tracking of precision.

In this section, we report on Caruso, Roe and Vaccon's work. §3.1 is dedicated to the foundations of their theory; it is mostly of mathematical nature and deals with p-adic analysis in several variables. It culminates with the statement (and the proof) of the precision Lemma (Theorem 3.16). Some first applications are discussed in §3.2 where the precision Lemma is used for finding the maximal precision one can expect in many concrete situations. Finally we propose in §3.3 general methods for reaching this optimal precision and apply them in concrete cases.

3.1 Foundations of the theory of p-adic precision

The aforementioned theory of p-adic precision is based on a single result of p-adic analysis — the so-called *precision Lemma* — controlling how ellipsoids transform under mappings of class C^1. In fact the terminology "ellipsoid" is not quite appropriate to the p-adic setting (though

very suggestive for the comparison with the real setting) because vector spaces over $\mathbb{Q}_p$ are not equipped with some L^2-norm. Mathematicians then prefer using the term "lattice" because, as we shall see below, the p-adic lattices behave like usual $\mathbb{Z}$-lattices in $\mathbb{R}$-vector spaces.

This subsection is organized as follows. We first introduce the notion of p-adic lattice (§§3.1.1–3.1.2) together with the necessary material of p-adic analysis (§3.1.3). After this preparation, §3.1.4 is devoted to the precision Lemma: we state it and prove it. Applications to p-adic precision will be discussed in the next subsections (§3.2 and §3.3).

3.1.1 Lattices in finite-dimensional p-adic vector spaces

Let E be a finite-dimensional vector space over $\mathbb{Q}_p$. A (ultrametric) *norm* on E is a mapping $\|\cdot\|_E : E \to \mathbb{R}^+$ satisfying the usual requirements:

(i) $\|x\|_E = 0$ if and only if $x = 0$;

(ii) $\|\lambda x\|_E = |\lambda| \cdot \|x\|_E$;

(iii) $\|x + y\|_E \leqslant \max\big(\|x\|_E, \|y\|_E\big)$.

Here x and y refer to elements of E while λ refers to a scalar in $\mathbb{Q}_p$. We notice that, without further assumption, one can prove that equality holds in (iii) as soon as $\|x\|_E \neq \|y\|_E$: all triangles are isosceles in all normed p-adic vector spaces! Indeed, by symmetry, we may assume $\|x\|_E < \|y\|_E$. Now we remark that from $\|x + y\|_E < \|y\|_E$, we would deduce:

$$\|y\|_E \leqslant \max\big(\|x + y\|_E, \|-x\|_E\big) = \max\big(\|x + y\|_E, \|x\|_E\big) < \|y\|_E$$

which is a contradiction. Our assumption was then absurd, meaning that $\|x + y\|_E = \|y\|_E$.

Given a real number r, we let $B_E(r)$ denote the closed ball in E of centre 0 and radius r, *i.e.*:

$$B_E(r) = \big\{\, x \in E \text{ s.t. } \|x\|_E \leqslant r \,\big\}.$$

It is worth remarking that $B_E(r)$ is a module over $\mathbb{Z}_p$. Indeed, on the one hand, multiplying by a scalar in $\mathbb{Z}_p$ does not increase the norm (since elements of $\mathbb{Z}_p$ have norm at most 1) and, on the other hand, the ultrametric triangular inequality implies that E is stable under addition. Balls in p-adic vector spaces have then two faces: one is analytic and one is algebraic. Being able to switch between these two points of view is often very powerful.

The very basic (but still very important) example of a normed $\mathbb{Q}_p$-vector space is $\mathbb{Q}_p^d$ itself endowed with the infinite norm defined by $\|(x_1, x_2, \ldots, x_d)\|_\infty = \max\big(|x_1|, |x_2|, \ldots, |x_d|\big)$ for $x_1, \ldots, x_d \in \mathbb{Q}_p$. The unit ball of $\big(\mathbb{Q}_p^d, \|\cdot\|_\infty\big)$ is $\mathbb{Z}_p^d$ and, more generally, $B_{\mathbb{Q}_p^d}(r) = p^n\mathbb{Z}_p^d$ where n is the unique relative integer defined by $p^{-n} \leqslant r < p^{-(n-1)}$. We notice that they are indeed $\mathbb{Z}_p$-modules. Other standard norms over $\mathbb{R}^d$ do not have a direct p-adic analogue since they violate the ultrametric triangular inequality.

The general notion of lattice is modeled on balls:

Definition 3.1. Let E be a finite-dimensional vector space over $\mathbb{Q}_p$. A *lattice* in E is a subset of E of the form $B_E(1)$ for some p-adic norm on E.

Remark that if H is a lattice in E and a is a non zero scalar in $\mathbb{Q}_p$, then aH is a lattice in E as well. Indeed if H is the closed unit ball for some norm $\|\cdot\|_E$, then aH is the closed unit ball for $\frac{1}{|a|}\|\cdot\|_E$. More generally, if $H \subset E$ is a lattice and $f : E \to E$ is a bijective linear transformation, then $f(H)$ is also a lattice. A consequence of Proposition 3.3 below is that all lattices take the form $f(H_0)$ where H_0 is a fixed lattice and f vary in $\mathrm{GL}(E)$.

Lattices might be thought of as "ellipsoids" centered at 0. They form a class of special neighborhoods of 0 that we will use afterwards to model the precision (see §3.3). From this perspective, they will appear as the natural generalization to higher dimension of the notion of

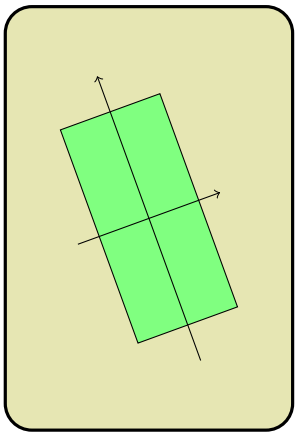

Figure 3.1: Picture of a lattice in the ultrametric world

bounded interval (centered at 0) of $\mathbb{Q}_p$ we have introduced in §1.1.5 and already widely used in §2.1 to model precision in zealous arithmetic.

Figure 3.1 shows a possible picture of a lattice drawn in the p-adic plane $\mathbb{Q}_p^2$ (endowed with the infinite norm). Nevertheless, we need of course to be very careful with such representations because the topology of $\mathbb{Q}_p$ has nothing to do with the topology of the paper sheet (or the screen). In particular, it is quite difficult to reflect ultrametricity.

Here is another purely algebraic definition of lattices which justifies the wording (compare with the case of $\mathbb{Z}$-lattice in $\mathbb{R}$-vector spaces).

Definition 3.2. Let E be a finite-dimensional vector space over $\mathbb{Q}_p$. A *lattice* in E is a $\mathbb{Z}_p$-module generated by a basis of E over $\mathbb{Q}_p$.

The fact that a lattice in the sense of Definition 3.2 is a lattice in the sense of Definition 3.1 is rather easy: if $(e_1, \dots, e_d)$ is a basis of E over $\mathbb{Q}_p$, the $\mathbb{Z}_p$-span of the e_i's is the closed unit ball for the norm $\|\cdot\|_E$ defined by:

$$\|\lambda_1 e_i + \cdots + \lambda_d e_d\|_E = \max\big(|\lambda_1|, \dots, |\lambda_d|\big).$$

As for the converse, it follows from the next proposition.

Proposition 3.3. *Let E be a d-dimensional normed vector space over $\mathbb{Q}_p$. There exists a basis $(e_1, \dots, e_d)$ of E over $\mathbb{Q}_p$ such that $B_E(1)$ is the $\mathbb{Z}_p$-module generated by the e_i's.*

Proof. Set $L = B_E(1)$. We have already seen that L is a module over $\mathbb{Z}_p$; thus the quotient L/pL makes sense and is a vector space over $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$. Consider $(\bar{e}_i)_{i\in I}$ a basis of L/pL. (We shall see later that L/pL is finite dimensional of dimension d over $\mathbb{F}_p$ but, for now, we do not know this and we do not assume anything on the set I indexing the basis.) For all $i \in I$, consider $e_i \in L$ which reduces to $\bar{e}_i$ modulo p.

We first claim that the family $(e_i)_{i\in I}$ (where e_i is considered as an element of E) is free over $\mathbb{Q}_p$. Indeed consider a relation of the form $\sum_{i\in I} \lambda_i e_i = 0$ with $\lambda_i \in \mathbb{Q}_p$ and $\lambda_i = 0$ for almost all i. Assume by contradiction that this relation is non trivial. Then $v = \min_i \mathrm{val}_p(\lambda_i)$ is finite. Up to multiplying the λ_i's by p^{-v}, we may assume that $\lambda_i \in \mathbb{Z}_p$ for all $i \in I$ and that there exists at least one index i for which λ_i does not reduce to 0 modulo p. Now reducing our dependency relation modulo p, we get $\sum_{i\in I} \bar{\lambda}_i \bar{e}_i = 0$ where $\bar{\lambda}_i \in \mathbb{F}_p$ is the class of λ_i modulo p. Since the family $(\bar{e}_i)_{i\in I}$ is free over $\mathbb{F}_p$ by construction, we derive $\bar{\lambda}_i = 0$ which contradicts our assumption.

It follows from the freedom of the e_i's that I is finite of cardinality at most d. Let us then write $I = \{1, 2, \dots, d'\}$ with $d' \leqslant d$. We now prove that $(e_1, \dots, e_{d'})$ generates L over $\mathbb{Z}_p$. Let

then $x \in L$. Using that the $\bar{e}_i$'s generate L/pL, we find p-adic integers $\lambda_{i,1}$ ($0 \leqslant i \leqslant d'$) such that:

$$\begin{aligned} & x \equiv \lambda_{1,1}e_1 + \lambda_{2,1}e_2 + \cdots + \lambda_{d',1}e_{d'} \pmod{pL} \\ \textit{i.e.} \quad & x = \lambda_{1,1}e_1 + \lambda_{2,1}e_2 + \cdots + \lambda_{d',1}e_{d'} + px_1 \end{aligned} \tag{3.1}$$

for some $x_1 \in L$. Applying the same reasoning to x_1 and re-injecting in Eq. (3.1), we end up with an equality of the form $x = \lambda_{1,2}e_1 + \lambda_{2,2}e_2 + \cdots + \lambda_{d',2}e_{d'} + px_2$ with $x_2 \in L$ and $\lambda_{i,2} \equiv \lambda_{i,1} \pmod{p}$ for all i. Continuing this process we construct d' sequences $(\lambda_{i,n})_{n\geqslant 0}$ with the property that

$$x \equiv \lambda_{1,n}e_1 + \lambda_{2,n}e_2 + \cdots + \lambda_{d',n}e_{d'} \pmod{p^n L}$$

and $\lambda_{i,n+1} \equiv \lambda_{i,n} \pmod{p^n}$ for all n and i. The latter congruence shows that, for all i, the sequence $(\lambda_{i,n})_{n\geqslant 0}$ is Cauchy and then converges to some $\lambda_i \in \mathbb{Z}_p$. Passing to the limit, we find that these λ_i's furthermore satisfy $x = \lambda_1 e_1 + \cdots + \lambda_{d'} e_{d'}$.

It now remains to prove that the e_i's generate E over $\mathbb{Q}_p$. For this, remark that any vector $x \in E$ can be written as $x = p^v y$ with $\|y\|_E \leqslant 1$, *i.e.* $y \in B_E(1)$. By what we have done before, we know that y can be written as a linear combination of the e_i's. Hence the same holds for x. □

Remark 3.4. Proposition 3.3 is a particular case of Nakayama's Lemma (which is a classical result of commutative algebra). We refer to [28] for a general introduction to commutative algebra including an exposition of Nakayama's Lemma in a much more general context.

Proposition 3.3 has other important consequences. It shows for instance that the inclusion of a lattice H in the ambient space E is homeomorphic to the inclusion of $\mathbb{Z}_p^d$ in $\mathbb{Q}_p^d$ (where d is the dimension of E). In particular lattices are all open and compact at the same time.

Three other consequences are enumerated in the next corollary.

Corollary 3.5. *(i) All finite dimension normed vector spaces over $\mathbb{Q}_p$ are complete.*

(ii) All norms over a given finite dimension vector space over $\mathbb{Q}_p$ are equivalent.

(iii) All linear applications between finite dimensional vector spaces over $\mathbb{Q}_p$ are continuous.

Proof. Let E be a finite dimension normal vector space over $\mathbb{Q}_p$. Use Proposition 3.3 to pick a basis $e_1, \ldots, e_d$ of E whose $\mathbb{Z}_p$-span is $B_E(1)$. We claim that an element $x \in E$ lies in the ball $B_E(p^{-n})$ if and only if all its coordinates on the basis $(e_1, \ldots, e_d)$ are divisible by p^n. Indeed the latter assertion implies easily the former by factoring out p^n. Conversely, let $x \in B_E(p^{-n})$. Then $p^{-n}x$ has norm at most 1 and thus can be written as $p^{-n}x = \lambda_1 e_1 + \cdots + \lambda_d e_d$ with $\lambda_i \in \mathbb{Z}_p$ for all i. Multiplying by p^n on both side and identifying coefficients, we get the claim.

Let $(x_n)_{n\geqslant 0}$ be a Cauchy sequence with values in E. For all n, write $x_n = \lambda_{n,1}e_1 + \cdots + \lambda_{n,d}e_d$ with $\lambda_{n,i} \in \mathbb{Q}_p$. It follows from the result we have proved in the previous paragraph that the sequences $(\lambda_{n,i})_{n\geqslant 0}$ are Cauchy for all i. They thus converge and hence so does $(x_n)_{n\geqslant 0}$. This proves (i).

The two other assertions are easy (after Proposition 3.3) and left to the reader. □

3.1.2 Computation with lattices

The algebraic side of lattices provides the tools for representing and manipulating p-adic lattices on computers (at least if the underlying vector space E is reasonable) in a quite similar fashion as usual integral lattices are represented and manipulated *via* integral matrices.

For simplicity, let us expose the theory in the case where $E = \mathbb{Q}_p^d$ (endowed with the infinite norm). We then represent a lattice $H \subset E$ by the matrix whose row vectors form a basis of L (in the sense of Definition 3.2). Equivalently, we can take the matrix, in the canonical basis, of the linear transformation f mentioned just above Definition 3.1.

For example, if $d = 4$ and H is generated by the vectors:

$$\begin{aligned} e_1 &= \big(\ldots 0101110000, \ldots 0011100000, \ldots 1011001000, \ldots 0011000100\big) \\ e_2 &= \big(\ldots 1101011001, \ldots 1101000111, \ldots 0111001010, \ldots 0101110101\big) \\ e_3 &= \big(\ldots 0111100011, \ldots 1011100101, \ldots 0010100110, \ldots 1111110111\big) \\ e_4 &= \big(\ldots 0000111101, \ldots 1101110011, \ldots 0011010010, \ldots 1001100001\big) \end{aligned}$$

we build the matrix

$$M = \begin{pmatrix} \ldots 0101110000 & \ldots 0011100000 & \ldots 1011001000 & \ldots 0011000100 \\ \ldots 1101011001 & \ldots 1101000111 & \ldots 0111001010 & \ldots 0101110101 \\ \ldots 0111100011 & \ldots 1011100101 & \ldots 0010100110 & \ldots 1111110111 \\ \ldots 0000111101 & \ldots 1101110011 & \ldots 0011010010 & \ldots 1001100001 \end{pmatrix} \tag{3.2}$$

Remark 3.6. By convention, our vectors will always be row vectors.

The matrix M we obtain this way is rather nice but we can further simplify it using Hermite reduction [23, §2.4]. In the p-adic setting, Hermite reduction takes the following form.

Theorem 3.7 (p-adic Hermite normal form)**.** *Any matrix $M \in \mathrm{GL}_d(\mathbb{Q}_p)$ can be uniquely written as a product $M = UA$ where $U \in \mathrm{GL}_d(\mathbb{Z}_p)$ and A has the shape:*

$$A = \begin{pmatrix} p^{n_1} & a_{1,2} & \cdots & \cdots & a_{1,d} \\ 0 & p^{n_2} & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & p^{n_{d-1}} & a_{d-1,d} \\ 0 & \cdots & \cdots & 0 & p^{n_d} \end{pmatrix}$$

where the n_i's are relative integers and the $a_{i,j}$'s are rational numbers of the form $a_{i,j} = \frac{b_{i,j}}{p^{v_{i,j}}}$ with $0 \leqslant b_{i,j} < p^{n_j + v_{i,j}}$.

The matrix A is called the Hermite normal form *of M.*

Remark 3.8. The left multiplication by the matrix U corresponds to operations on the rows of M, that are operations on the vectors e_i. The invertibility of U over $\mathbb{Z}_p$ ensures that the row vectors of A continue to generate the lattice H we have started with.

The proof of Theorem 3.7 is constructive and can be done by row-echelonizing the matrix M. Instead of writing it down for a general M, let us just show how it works on the example (3.2). We first select in the first column an entry with minimal valuation, we then move it to the top left corner by swapping rows and we normalize it so that it becomes a power of p by rescaling the first row by the appropriate invertible element of $\mathbb{Z}_p$. In our example, one can for instance choose the second entry of the first column. After swap and renormalization, we get:

$$\begin{pmatrix} 1 & \ldots 1110011111 & \ldots 0011011010 & \ldots 1101111101 \\ \ldots 0101110000 & \ldots 0011100000 & \ldots 1011001000 & \ldots 0011000100 \\ \ldots 0111100011 & \ldots 1011100101 & \ldots 0010100110 & \ldots 1111110111 \\ \ldots 0000111101 & \ldots 1101110011 & \ldots 0011010010 & \ldots 1001100001 \end{pmatrix}$$

We now use the top left entry as pivot to clear the other entries of the first column, obtaining this way the new matrix:

$$\begin{pmatrix} 1 & \ldots 1110011111 & \ldots 0011011010 & \ldots 1101111101 \\ 0 & \ldots 0001010000 & \ldots 0101101000 & \ldots 0100010100 \\ 0 & \ldots 0111101000 & \ldots 0101011000 & \ldots 1100100000 \\ 0 & \ldots 1010010000 & \ldots 0011100000 & \ldots 0011011000 \end{pmatrix}$$

Remark that these row operations do not affect the $\mathbb{Z}_p$-span of the row vectors (*i.e.* they correspond to a transformation matrix U which lies in $\mathrm{GL}_d(\mathbb{Z}_p)$). We continue this process with the 3×3 matrix obtained by erasing the first row and the first column. Since we have forgotten the first row, the smallest valuation of an entry of the second column is now 3 and the corresponding entry is located on the third row. We then swap the second and the third rows, rescale the (new) second row in order to put 2^3 on the diagonal and use this value 2^3 as pivot to cancel the remaining entries on the second column. After these operations, we find:

$$\begin{pmatrix} 1 & \ldots 1110011111 & \ldots 0011011010 & \ldots 1101111101 \\ 0 & 2^3 & \ldots 0000111000 & \ldots 0110100000 \\ 0 & 0 & \ldots 1100111000 & \ldots 0011010100 \\ 0 & 0 & \ldots 1011110000 & \ldots 0001011000 \end{pmatrix}$$

Iterating again one time this process, we arrive at:

$$\begin{pmatrix} 1 & \ldots 1110011111 & \ldots 0011011010 & \ldots 1101111101 \\ 0 & 2^3 & \ldots 0000111000 & \ldots 0110100000 \\ 0 & 0 & 2^3 & \ldots 000001100 \\ 0 & 0 & 0 & \ldots 111110000 \end{pmatrix}$$

Interestingly, observe that the precision on the last two entries of the last column has decreased by one digit. This is due to the fact that the pivot 2^3 was not the element with the smallest valuation on its *row*. This loss of precision may cause troubles only when the initial matrix M is known at precision $O(p^N)$ with $N \leqslant \max_i n_i$; in practice such a situation very rarely happen and will never appear in this course.

The next step of Hermite reduction is the normalization of the bottom right entry:

$$\begin{pmatrix} 1 & \ldots 1110011111 & \ldots 0011011010 & \ldots 1101111101 \\ 0 & 2^3 & \ldots 0000111000 & \ldots 0110100000 \\ 0 & 0 & 2^3 & \ldots 000001100 \\ 0 & 0 & 0 & 2^4 \end{pmatrix}$$

It remains now to clean up the upper triangular part of the matrix. For this, we proceed again column by column by using the pivots on the diagonal. Of course there is nothing to do for the first column. We now reduce the $(1,2)$ entry modulo 2^3 which is the pivot located on the same column. In order to do so, we add to the first row the appropriate multiple of the second row. We obtain this way the new matrix:

$$\begin{pmatrix} 1 & 7 & \ldots 1110110010 & \ldots 0010011101 \\ 0 & 2^3 & \ldots 0000111000 & \ldots 0110100000 \\ 0 & 0 & 2^3 & \ldots 000001100 \\ 0 & 0 & 0 & 2^4 \end{pmatrix}$$

We emphasize that the $(1,2)$ entry of the matrix is now *exact*! Repeating this procedure several times we end up finally with

$$\begin{pmatrix} 1 & 7 & 2 & 5 \\ 0 & 2^3 & 0 & 12 \\ 0 & 0 & 2^3 & 12 \\ 0 & 0 & 0 & 2^4 \end{pmatrix}$$

which is the expected Hermite normal form.

Remark 3.9. The algorithmic problem of computing the Hermite normal form has been widely studied over the integers and we now know much more efficient algorithms (taking advantage of fast matrix multiplication) to solve it [48]; these algorithms extend without trouble to the case of $\mathbb{Z}_p$ (with the same complexity). Other algorithms specially designed for the p-adic setting are also available [16].

Here is a remarkable corollary of Theorem 3.7 (compare with Proposition 2.1):

Corollary 3.10. *Lattices in $\mathbb{Q}_p^d$ are representable on computers by exact data.*

Operations on lattices can be performed on Hermite forms without difficulty. The sum of two lattices, for example, is computed by concatenating the two corresponding Hermite matrices and re-echelonizing. In a similar fashion, one can compute the image of a lattice under a surjective[19] linear mapping f: we apply f to each generator and echelonize the (possibly rectangular) matrix obtained this way.

3.1.3 A small excursion into p-adic analysis

Another important ingredient we will need is the notion of differentiable functions in the p-adic world. All the material presented in this section is classical. We refer to [70, §I.4] and [25] for a much more detailed and systematic exposition.

The case of univariate functions

The notion of differentiability at a given point comes with no surprise: given an open subset $U \subset \mathbb{Q}_p$, $x \in U$ and a function $f : U \to \mathbb{Q}_p$, we say that f is differentiable at x if $\frac{f(x)-f(y)}{x-y}$ has a limit when y tends to x. When f is differentiable at x, we define:

$$f'(x) = \lim_{y \to x} \frac{f(x) - f(y)}{x - y}.$$

Alternatively one can define the derivative using Taylor expansion: one says that f is differentiable at x is there exists $f'(x) \in \mathbb{Q}_p$ for which:

$$\big|f(y) - f(x) - (y-x)f'(x)\big| = o\big(|y-x|\big) \quad \text{for } y \to x. \tag{3.3}$$

Usual formulas for differentiating sums, products, composed functions, *etc.* extend *verbatim* to the p-adic case.

The fact that $\mathbb{Q}_p$ is (highly) disconnected has however unpleasant consequences. For instance, the vanishing of f' on an interval does not imply the constancy of f. Indeed, consider as an example the indicator function of $\mathbb{Z}_p$: it is a locally constant function defined on $\mathbb{Q}_p$. It is then differentiable with zero derivative but it is not constant on $\mathbb{Q}_p$. One can cook up even worse examples. Look for instance at the function $f : \mathbb{Z}_p \to \mathbb{Z}_p$ defined by:

$$a_0 + a_1 p + \cdots + a_n p^n + \cdots \quad \mapsto \quad a_0 + a_1 p^2 + \cdots + a_n p^{2n} + \cdots$$

where the a_i's are the digits, *i.e.* they lie in the range $[0, p-1]$. A straightforward computation shows that $|f(x) - f(y)| = |x - y|^2$, *i.e.* f is 2-Hölder. In particular, f is differentiable everywhere and f' vanishes on $\mathbb{Z}_p$. Yet f is apparently injective and *a fortiori* not constant on any interval.

The notion of p-adic function of class C^1 is more subtle. Indeed the notion of differentiable function with continuous derivative is of course well defined but not that interesting in the p-adic setting. Precisely, this definition is too weak and encompasses many "irregular" functions. A more flexible definition is the following.

Definition 3.11. Let U be an open subset of $\mathbb{Q}_p$ and $f : U \to \mathbb{Q}_p$ be a function. We say that f is of class C^1 on U if there exists a function $f' : U \to \mathbb{Q}_p$ satisfying the following property: for all $a \in U$ and all $\varepsilon > 0$, there exists a neighborhood $U_{a,\varepsilon} \subset U$ of a on which:

$$\big|f(y) - f(x) - (y-x)f'(a)\big| \leqslant \varepsilon \cdot |y-x|. \tag{3.4}$$

[19] Surjectivity ensures that the image remains a lattice in the codomain.

Given a function $f : U \to \mathbb{Q}_p$ of class C^1 on U, it is clear that f is differentiable on U and that the function f' appearing in Definition 3.11 has to be the derivative of f. Moreover f' is necessarily continuous on U. Indeed consider $a \in U$ and $\varepsilon > 0$. Let $b \in U_{a,\varepsilon}$. The intersection $U = U_{a,\varepsilon} \cap U_{b,\varepsilon}$ is open and not empty. It then contains at least two different points, say x and y. It follows from the definition that:

$$\begin{aligned} &\big|f(y) - f(x) - (y-x)f'(a)\big| \leqslant \varepsilon \cdot |y-x| \\ \text{and} \quad &\big|f(y) - f(x) - (y-x)f'(b)\big| \leqslant \varepsilon \cdot |y-x|. \end{aligned}$$

Combining these inequalities, we derive $\big|f'(b) - f'(a)\big| \leqslant \varepsilon$ which proves the continuity of f' at x. In the real setting, the continuity of f' implies conversely the inequality (3.4)[20] but this implication fails in the p-adic world. A counter-example is given by the function $f : \mathbb{Z}_p \to \mathbb{Q}_p$ defined by:

$$\begin{aligned} x = a_v p^v + a_{v+1}p^{v+1} + \cdots + a_{2v}p^{2v} + a_{2v+1}p^{2v+1} + \cdots \\ \mapsto \quad f(x) = a_v p^{2v} + \big(a_{2v}p^{2v} + a_{2v+1}p^{2v+2} + a_{2v+2}p^{2v+4} + \cdots\big) \end{aligned}$$

where the a_i's are integers between 0 and $p-1$ with $a_v \neq 0$. One checks that $|f(x)| \leqslant |x|^2$ and $|f(x) - f(y)| = \frac{|x-y|^2}{|x|^2}$ if $|x-y| \leqslant |x|^2 < 1$. These inequalities ensure that f is differentiable on $\mathbb{Z}_p$ and that its derivative vanishes everywhere (and thus is continuous). On the other hand when $|x-y| = |x|^2 < 1$, we have $|f(x) - f(y)| = |x-y|$, showing that f cannot be of class C^1.

Remark 3.12. Alternatively, following [25], one may define the notion of class C^1 for a function $f : U \to F$ by requiring the existence of a covering $(U_i)_{i\in I}$ of U and of *continuous* real-valued functions $\varepsilon_i : \mathbb{R}^+ \to \mathbb{R}^+$ with $\varepsilon_i(0) = 0$ such that:

$$\forall i \in I, \quad \forall x, y \in U_i, \quad \big|f(y) - f(x) - (y-x)f'(x)\big| \leqslant |y-x| \cdot \varepsilon_i\big(|y-x|\big) \tag{3.5}$$

(compare with (3.4)). In brief, a function f is of class C^1 when the estimation (3.3) is *locally uniform* on x. When the domain U is compact (*e.g.* $U = \mathbb{Z}_p$), one can remove the word "locally", *i.e.* one can forget about the covering and just take $I = \{\star\}$ and $U_\star = U$.

Proof of the equivalence between the two definitions. Assume first that f satisfies the definition of Remark 3.12. Let $a \in U$ and $\varepsilon > 0$. We have $a \in U_i$ for some i. Let δ be a positive real number such that $\varepsilon_i < \varepsilon$ on the interval $[0, \delta)$. Define $U_{a,\varepsilon}$ as the intersection of U_i with the open ball of centre a and radius δ. On $U_{a,\varepsilon}$, the estimation:

$$\big|f(y) - f(x) - (y-x)f'(x)\big| \leqslant \varepsilon \cdot |y-x| \tag{3.6}$$

holds. It remains then to relate $f'(x)$ to $f'(a)$. In order to do so, we write:

$$\begin{aligned} &\big|f(x) - f(a) - (x-a)f'(a)\big| \leqslant \varepsilon \cdot |x-a| \\ &\big|f(a) - f(x) - (a-x)f'(x)\big| \leqslant \varepsilon \cdot |a-x|. \end{aligned}$$

Combining these inequalities, we obtain $|f'(x) - f'(a)| \leqslant \varepsilon$. Re-injecting this new input in (3.6), we finally get (3.4) as desired.

Conversely, assume that f is of class C^1 in the sense of Definition 3.11. Since the problem is local on the domain, we may assume that U is an open ball. Up to translating and rescaling f, one may further suppose without loss of generality that $U = \mathbb{Z}_p$. In particular, note that U is compact. We define the function ε on $(0, \infty)$ by:

$$\varepsilon(\delta) = \sup_{\substack{x,y\in\mathbb{Z}_p \\ 0<|y-x|\leqslant\delta}} \frac{|f(y) - f(x) - f'(x)(y-x)|}{|y-x|}.$$

[20] Indeed, by the mean value theorem we can write $f(y) - f(x) = (y-x)f'(g(x,y))$ for some $g(x,y) \in [x,y]$. Thus $|f(y) - f(x) - (y-x)f'(a)| = |y-x| \cdot |f'(g(x,y)) - f'(a)|$ and the conclusion follows from Heine–Cantor Theorem.

Compacity ensures that the supremum is finite, so that ε is well defined. We have to show that ε goes to 0 when δ goes to 0. Let $\varepsilon' > 0$. By assumption, for all $a \in \mathbb{Z}_p$, there exists an open neighborhood $U_{a,\varepsilon'}$ of a on which $\frac{|f(y)-f(x)-f'(x)(y-x)|}{|y-x|} \leqslant \varepsilon'$. Up to shrinking $U_{a,\varepsilon'}$, one may assume that $U_{a,\varepsilon'} = a + p^{n_a}\mathbb{Z}_p$ for some positive integer n_a. Now observe that the family of all $U_{a,\varepsilon'}$ when a varies is a covering of $\mathbb{Z}_p$; by compacity, one can extract from it a finite subcovering $(U_{a_i,\varepsilon'})_{1\leqslant i\leqslant m}$ that continues to cover $\mathbb{Z}_p$. If n denotes the supremum of the n_{a_i}'s ($1 \leqslant i \leqslant m$), we thus derive $\varepsilon(\delta) \leqslant \varepsilon'$ for $\delta \leqslant p^{-n}$ and we are done. □

The case of multivariate functions

Let E and F be two finite dimensional normed vector spaces over $\mathbb{Q}_p$. We denote by $\mathcal{L}(E,F)$ the space of $\mathbb{Q}_p$-linear mappings from E to F. The definition of differentiability and "of class C^1" is mimicked from the univariate case.

Definition 3.13. Let U be an open subset of E.

A function $f : U \to F$ is *differentiable* at the point $x \in U$ if there exists a linear mapping $df_x \in \mathcal{L}(E,F)$ such that:

$$\|f(y) - f(x) - df_x(y-x)\|_F = o\big(\|y-x\|_E\big) \quad \text{when} \quad y \to x.$$

A function $f : U \to F$ is *of class* C^1 on U if there exists a function $df : U \to \mathcal{L}(E,F)$, $x \mapsto df_x$ satisfying the following property: for all $v \in U$ and all $\varepsilon > 0$, there exists a neighborhood $U_{a,\varepsilon} \subset U$ of a on which:

$$\|f(y) - f(x) - df_a(y-x)\|_F \leqslant \varepsilon \cdot \|y-x\|_E. \tag{3.7}$$

Of course, if f is of class C^1 on U, it is differentiable at every point $x \in U$. When $E = \mathbb{Q}_p^n$ and $F = \mathbb{Q}_p^m$ (or more generally when E and F are equipped with distinguished bases) the matrix of the linear mapping df_x is the Jacobian matrix $J(f)_x$ defined by:

$$J(f)_x = \left(\frac{\partial f_j}{\partial x_i}(x)\right)_{1\leqslant i\leqslant n,\, 1\leqslant j\leqslant m}$$

where $x_1, \ldots, x_n$ are the coordinates of E and $f_1, \ldots, f_m$ are the components of f.

Remark 3.14. Similarly to the univariate case (see Remark 3.12), a function $f : U \to F$ is of class C^1 if and only if there exist a covering $(U_i)_{i\in I}$ of U and some *continuous* real-valued functions $\varepsilon_i : \mathbb{R}^+ \to \mathbb{R}^+$ with $\varepsilon_i(0) = 0$ such that:

$$\forall i \in I, \quad \forall x, y \in U_i, \quad \|f(y) - f(x) - (y-x)f'(x)\|_F \leqslant \|y-x\|_E \cdot \varepsilon_i\big(\|y-x\|_E\big). \tag{3.8}$$

and one can just take $I = \{\star\}$ and $U_\star = U$ when U is compact.

3.1.4 The precision Lemma

The precision Lemma is a result of p-adic analysis controlling how lattices transform under sufficiently regular mappings. Before stating it, we need a definition.

Definition 3.15. Let E be a normed vector space over $\mathbb{Q}_p$. Given $\rho \in (0,1]$ and $r > 0$, we say that a lattice $H \subset E$ is ρ-rounded, r-small if $B_E(\rho r) \subset H \subset B_E(r)$.

When $E = \mathbb{Q}_p^d$ (endowed with the infinite norm), one can determine ρ and r satisfying the conditions of Definition 3.15 by looking at any matrix M representing the lattice H (see §3.1.2). Indeed, if n is an integer for which the matrix $p^n M$ has all its entries in $\mathbb{Z}_p$, the lattice $p^n H$ is included in $\mathbb{Z}_p^n = B_E(1)$, meaning that $H \subset B_E(p^n)$. Similarly $B_E(p^{-m}) \subset H$ whenever m is an integer such that $p^m M^{-1} \in M_d(\mathbb{Z}_p)$. Therefore if n is the smallest valuation of an entry of M and m is the smallest valuation of an entry of M^{-1}, the corresponding lattice H is (p^{n+m})-bounded, p^n-small.

Theorem 3.16 (Precision Lemma). *Let E and F be two finite-dimensional p-adic normed vector spaces and let $f : U \to F$ be a function of class C^1 defined on an open subset U of E.*

Let $v \in U$ be such that df_v is surjective. Then, for all $\rho \in (0, 1]$, there exists a positive real number δ such that, for any $r \in (0, \delta)$, any lattice ρ-bounded r-small lattice H satisfies:

$$f(v + H) = f(v) + df_v(H). \tag{3.9}$$

Unfortunately the precision Lemma is a bit technical; understanding its precise content is then probably not easy at first glance. In order to help the reader, let us say that the most important part of the precision Lemma is the conclusion, namely Eq. (3.9). This equation explains how f transforms a "shifted" lattice (*i.e.* a lattice translated by some vectors) and teaches us that f transforms a shifted lattice into another shifted lattice! From people coming from the real world, this result should appear as really amazing: in the real case, the image of an ellipsoid under a function f is in general definitely *not* another ellipsoid (unless f is affine). In the p-adic case, this happens for any function f of class C^1 with *surjective differential* (this assumption is important) and almost any lattice H (the assumptions on H are actually rather weak though technical). This is the magic of ultrametricity.

Below, we give the proof of the precision Lemma and discuss several extensions. We advise the reader who is more interested by applications (than by computations with ε) to skip the end of the subsection and go directly to §3.2, page 53.

Proof of Theorem 3.16. Without loss of generality, we may assume $v = 0$ and $f(0) = 0$. From the surjectivity of df_0, we derive that there exists a positive constant C such that $B_F(1) \subset df_0(B_E(C))$. Pick $\varepsilon < \frac{\rho}{C}$, and choose a positive real number δ for which

$$\|f(b) - f(a) - df_0(b-a)\|_F \leqslant \varepsilon \cdot \|b-a\|_E \tag{3.10}$$

on the ball $B_E(\delta)$. Let $r \in (0, \delta)$. We suppose that H is a lattice with $B_E(\rho r) \subset H \subset B_E(r)$. We seek to show that f maps H surjectively onto $df_0(H)$. We first prove that $f(H) \subset df_0(H)$. Suppose $x \in H$. Applying Eq. (3.10) with $a = 0$ and $b = x$, we get $\|f(x) - df_0(x)\|_F \leqslant \varepsilon\|x\|_E$. Setting $y = f(x) - df_0(x)$, we have $\|y\|_F \leqslant \varepsilon r$. The definition of C implies that $B_F(\varepsilon r) \subset df_0(B_E(\rho r))$. Thus there exists $x' \in B_E(\rho r)$ such that $df_0(x') = y$. Then $f(x) = df_0(x - x') \in df_0(H)$.

We now prove surjectivity. Let $y \in df_0(H)$. Let $x_0 \in H$ be such that $y = df_0(x_0)$. We inductively define two sequences (x_n) and (z_n) by the following cross requirements:

- z_n is an element of E satisfying $df_0(z_n) = y - f(x_n)$ and $\|z_n\|_E \leqslant C \cdot \|y - f(x_n)\|_F$, and
- $x_{n+1} = x_n + z_n$.

For convenience, let us also define $x_{-1} = 0$ and $z_{-1} = x_0$. We claim that the sequences (x_n) and (z_n) are well defined and take their values in H. We do so by induction, assuming that x_{n-1} and x_n belong to H and showing that z_n and x_{n+1} do as well. Noticing that

$$\begin{aligned} y - f(x_n) &= f(x_{n-1}) + df_0(z_{n-1}) - f(x_n) \\ &= f(x_{n-1}) - f(x_n) - df_0(x_{n-1} - x_n) \end{aligned} \tag{3.11}$$

we deduce using differentiability that $\|y - f(x_n)\|_F \leqslant \varepsilon \cdot \|x_n - x_{n-1}\|_E$. Since we are assuming that x_{n-1} and x_n lie in $H \subset B_E(r)$, we find $\|y - f(x_n)\| \leqslant \varepsilon r$. Thus $\|z_n\| \leqslant C \cdot \varepsilon r < \rho r$ and then $z_n \in H$. From the relation $x_{n+1} = x_n + z_n$, we finally deduce $x_{n+1} \in H$.

Using (3.11) and differentiability at 0 once more, we get

$$\|y - f(x_n)\| \leqslant \varepsilon \cdot \|z_{n-1}\| \leqslant \varepsilon C \cdot \|y - f(x_{n-1})\|,$$

for all $n > 0$. Therefore, $\|y - f(x_n)\| = O(a^n)$ and $\|z_n\| = O(a^n)$ for $a = \varepsilon C < \rho \leqslant 1$. These conditions show that $(x_n)_{n\geqslant 0}$ is a Cauchy sequence, which converges since E is complete (see Corollary 3.5). Write x for the limit of the x_n; we have $x \in H$ because H is closed (see again Corollary 3.5). Moreover, f is continuous on $H \subseteq U_\varepsilon$ since it is differentiable, and thus $y = f(x)$. □

For the applications we have in mind, Theorem 3.16 is actually a bit too weak because it is not effective: the value of δ is not explicit whereas we will often need it in concrete applications. In the most general case, it seems difficult to say much more about δ. Nevertheless, there are many cases of interest where the dependence on δ can be made explicit. The simplest such case is that of multivariate polynomials and is covered by the next proposition.

Proposition 3.17. *Let $f : \mathbb{Q}_p^n \to \mathbb{Q}_p^m$ be a function whose coordinates are all multivariate polynomials with coefficients in $\mathbb{Z}_p$. Let $v \in \mathbb{Z}_p^n$ such that df_v is surjective. Then Eq. (3.9) holds as soon as $H \subset B_E(r)$ and $B_F(pr^2) \subset df_v(H)$ for some positive real number r.*

Remark 3.18. The case where f is a multivariate polynomial with coefficients in $\mathbb{Q}_p$ reduces to the above proposition by multiplying f by an appropriate constant. Similarly, if the point v does not lie in $\mathbb{Z}_p^n$ but in $\mathbb{Q}_p^n$, we can shift and rescale the polynomial f in order to place ourselves within the scope of application of Proposition 3.17.

The proof of Proposition 3.17 is based on the following Lemma.

Lemma 3.19. *Let $g : \mathbb{Q}_p^n \to \mathbb{Q}_p^m$ be a function whose coordinates are all multivariate polynomials with coefficients in $\mathbb{Z}_p$. Then*

$$\|g(b) - g(a) - dg_0(b-a)\| \leqslant \max\big(\|a\|, \|b\|\big) \cdot \|b-a\|$$

for all $a, b \in \mathbb{Z}_p^n$.

Proof. We may assume $m = 1$. By linearity, we may further assume that $g(x_1, \dots, x_n) = x_1^{\alpha_1} \cdots x_n^{\alpha_n}$. If the sum of the α_i's is at most 1, the quantity $g(b) - g(a) - dg_0(b-a)$ vanishes and the lemma is clear. Otherwise the differential dg_0 vanishes and we write:

$$\begin{aligned} &g(b_1, \dots, b_n) - g(a_1, \dots, a_n) \\ &\quad = (b_1^{\alpha_1} - a_1^{\alpha_1}) b_2^{\alpha_2} \cdots b_n^{\alpha_n} + a_1^{\alpha_1} (b_2^{\alpha_2} - a_2^{\alpha_2}) b_3^{\alpha_3} \cdots b_n^{\alpha_n} + \cdots + a_1^{\alpha_1} \cdots a_{n-1}^{\alpha_{n-1}} (a_n^{\alpha_n} - b_n^{\alpha_n}). \end{aligned}$$

Noting that $b_i^{\alpha_i} - a_i^{\alpha_i} = (b_i - a_i) \sum_{k=0}^{\alpha_i - 1} b_i^k a_i^{\alpha_i - 1 - k}$ and remembering that $\alpha_1 + \cdots + \alpha_n \geqslant 2$, we get the announced result by applying the ultrametric triangular inequality. □

Proof of Proposition 3.17. Let H be a lattice satisfying the assumptions of the proposition. Let n be the largest relative integer for which $p^n \leqslant r$. Then $B_E(r) = B_E(p^n)$ and $B_F(p^{2n+1}) \subset B_F(pr^2)$, so that $H \subset B_E(p^n)$ and $B_F(p^{2n+1}) \subset df_v(H)$.

Set $H' = p^n H$; clearly H' is a lattice contained in $B_E(1)$. Let $\varphi : \mathbb{Q}_p^n \to \mathbb{Q}_p^n$ be a linear mapping taking bijectively $B_E(1)$ to H'. Consider the composite $g = f \circ (v + \varphi)$. From $H' \subset B_E(1)$, we deduce that the entries of the matrix of φ (in the canonical basis) lie in $\mathbb{Z}_p$. Since moreover $v \in \mathbb{Z}_p^n$, we deduce that all the coordinates of g are given by multivariate polynomials with coefficients in $\mathbb{Z}_p$. Furthermore, we have:

$$\begin{aligned} f(v + H) = f(v + p^{-n} H') = f(v + \varphi(B_E(p^n)) = g(B_E(p^n)) \\ \text{and} \quad f(v) + df_v(H) = f(v) + df_v \circ \varphi(B_E(p^n)) = g(0) + dg_0(B_E(p^n)). \end{aligned}$$

We then have to prove that $g(B_E(p^n)) = g(0) + dg_0(B_E(p^n))$ knowing that g is a multivariate polynomial function with coefficients in $\mathbb{Z}_p$ and $B_F(p^{n+1}) \subset dg_0(B_E(1))$. This can be done by following the lines of the proof of Theorem 3.16 and refining Eq. (3.10) by the estimation of Lemma 3.19. □

Other results concerning more general f and building on the machinery of Newton polygons are available in the literature. We refer the reader to [18] for the case of locally analytic functions and to [19] for the case of solutions of partial differential equations of order 1.

3.2 Optimal precision and stability of algorithms

The precision Lemma (Theorem 3.16) provides the mathematical tools for finding the intrinsic optimal loss/gain of precision of a given problem and then for studying the stability of numerical p-adic algorithms.

3.2.1 The precision Lemma at work

Consider an "abstract p-adic problem" encoded by a mathematical function $\varphi : U \to F$ where U is an open subset in a finite dimensional p-adic vector space E and F is another finite dimensional p-adic vector space. The modelling is straightforward: the domain U is the space of inputs, the codomain F is the space of outputs and φ is the mapping sending an input to the expected output. Many concrete problems fit into this framework: for instance φ can be the function mapping a pair of p-adic numbers to their sum or their product, but it can also be the function mapping a polynomial (of a given degree) to its derivative, a matrix (of given dimensions) to its inverse, a family of multivariate polynomials (of given size and degrees) to its reduced Gröbner basis, *etc.*

In order to apply the precision Lemma, we shall assume that φ is of class C^1. This hypothesis is not restrictive at all in practice since in many common cases, the function φ has much more regularity than that: it is often obtained by composing sums, products, divisions and if-statements so that it is locally given by multivariate rational fractions. We suppose furthermore that E and F are endowed with distinguished bases $(e_1, \dots, e_n)$ and $(f_1, \dots, f_m)$ respectively. This choice models the way the elements of E and F are represented on the computer. For instance if $E = \mathbb{Q}_p[X]_{<n}$ (the vector space of polynomials of degree less than n) and polynomials are represented internally by the list of their coefficients, we will just choose the canonical basis. On the contrary, if polynomials are represented as linear combinations of the shape:

$$\lambda_0 + \lambda_1 X + \lambda_2 X(X-1) + \cdots + \lambda_{n-1} X(X-1)\cdots(X-n+2)$$

then the distinguished basis we will choose is $(1, X, X(X-1), \dots, X(X-1)\cdots(X-n+2))$.

Context of zealous arithmetic

Recall that, in the zealous point of view, p-adic numbers are modeled by intervals of the form $a + O(p^N)$. As a consequence the input of the problem we are studying will not be an actual element of U but a quantity of the form:

$$\big(a_1 + O(p^{N_1})\big) \cdot e_1 + \big(a_2 + O(p^{N_2})\big) \cdot e_2 + \cdots + \big(a_n + O(p^{N_n})\big) \cdot e_n$$

which can be rewritten as the shifted lattice $v + H$ with:

$$\begin{aligned} v &= a_1 e_1 + a_2 e_2 + \cdots + a_n e_n \\ H &= \mathrm{Span}_{\mathbb{Z}_p}\big(p^{N_1} e_1,\, p^{N_2} e_2, \cdots,\, p^{N_n} e_n\big). \end{aligned}$$

Similarly the output takes the form:

$$\big(b_1 + O(p^{M_1})\big) \cdot f_1 + \big(b_2 + O(p^{M_2})\big) \cdot f_2 + \cdots + \big(b_m + O(p^{M_m})\big) \cdot f_m$$

where the b_j's are the coordinates of $\varphi(v)$ and then do not depend (fortunately) on the algorithm we are using to evaluate φ. On the other hand, the M_j's may — and do — depend on it. The question we address here can be formulated as follows: what is the maximal value one can expect for the M_j's? In other words: what is the optimal precision one can expect on the output (in terms of the initial precision we had on the inputs)?

The precision Lemma provides a satisfying answer to this question. Indeed, for $j \in \{1, \dots, m\}$, let $\mathrm{pr}_{F,j} : F \to \mathbb{Q}_p$ be the linear mapping taking a vector to its j-th coordinate (in the distinguished

basis $(f_1, \dots, f_m)$ we have fixed once for all). From the precision Lemma applied to the composite $\varphi_j = \mathrm{pr}_{F,j} \circ \varphi$, we get:

$$\varphi_j(v + H) = \varphi_j(v) + d\varphi_{j,v}(H) = b_j + d\varphi_{j,v}(H) \tag{3.12}$$

as soon as $d\varphi_{j,v}$ is surjective (and H satisfies some additional mild assumptions that we will ignore for now). Under these assumptions, Eq. (3.12) holds and the equality signs in it show that the optimal precision $O(p^{M_j})$ we were looking for is nothing but $d\varphi_{j,v}(H)$; note that the latter is a lattice in $\mathbb{Q}_p$ and then necessarily takes the form $p^{M_j}\mathbb{Z}_p = O(p^{M_j})$ for some relative integer M_j. Note moreover that the surjectivity of $d\varphi_{j,v}$ is equivalent to its non-vanishing (since $d\varphi_{j,v}$ takes its value in a one-dimensional vector space). Furthermore, by the chain rule, we have $d\varphi_{j,v} = \mathrm{pr}_{F,j} \circ d\varphi_v$ because $\mathrm{pr}_{F,j}$ is linear and then agrees with its differential at any point.

Finding the optimal precision $O(p^{M_j})$ can then be done along the following lines: we first compute the Jacobian matrix $J(\varphi)_v$ of φ at the point v and form the product:

$$A = \begin{pmatrix} p^{N_1} & & \\ & \ddots & \\ & & p^{N_n} \end{pmatrix} \cdot J(\varphi)_v \tag{3.13}$$

whose rows generate $d\varphi_{j,v}(H)$. The integer M_j appears then as the smallest valuation of an entry of the j-th column of A (unless this column vanishes in which case the precision Lemma cannot be applied). Of course, the computation of the Jacobian matrix can be boring and/or time-consuming; we shall see however in §3.2.2 below that, in many situations, simple mathematical arguments provide closed formulas for $J(\varphi)_v$, avoiding this way their direct computation.

The notion of diffused digits of precision. One important remark is that the lattice:

$$H' = \mathrm{Span}_{\mathbb{Z}_p}\left(p^{M_1} f_1,\, p^{M_2} f_2, \cdots,\, p^{M_m} e_m\right) \subset F$$

(for the M_j's we have constructed) does satisfy $f(v + H) \subset f(v) + H'$ but does *not* satisfy $f(v + H) = f(v) + H'$ in general. In other words, although we cannot expect more correct digits on each component of the output separately, the precision on the output is globally not optimal. Below is a very simple example illustrating this point.

Example. Assume that we want to evaluate an affine polynomial at the two points 0 and p. The function φ modeling this problem is:

$$\varphi : \mathbb{Q}_p[X]_{\leqslant 1} \to \mathbb{Q}_p^2, \quad P(X) = aX + b \mapsto (P(0), P(p)) = (b,\, ap{+}b).$$

It is a linear function so its Jacobian is easy to compute. We find $J(\varphi) = \begin{pmatrix} 0 & p \\ 1 & 1 \end{pmatrix}$. We assume furthermore that the inputs a and b are both given at precision $O(p^N)$. We then find that the optimal precision on $P(0)$ and $P(p)$ is $O(p^N)$ as well. Nevertheless the determinant of $J(\varphi)$ is apparently p; hence $J(\varphi)$ is not invertible in $M_2(\mathbb{Z}_p)$ and its row vectors generate a lattice which is strictly smaller than $\mathbb{Z}_p^2$. What does happen concretely? The main observation is that the difference $P(p) - P(0) = ap$ can be computed at precision $O(p^{N+1})$ because of the multiplication by p. There is thus one more digit that we know but this digit is "diffused" between $P(0)$ and $P(p)$ and only appears when we look at the difference.

The phenomenon enlighten in the above example is quite general: *we cannot in general reach the optimal precision by just giving individual precision on each coordinate of the output (we say that some digits of precision are diffused over several coordinates)... but this becomes always*[21] *possible after a suitable base change (which recombines the diffused digits of precision into actual digits)*. In order to make this sentence more precise, we first formulate a rigorous definition concerning the diffusion of the digits of precision.

[21] At least when the precision Lemma applies...

Definition 3.20. Let F be a finite dimensional vector space over $\mathbb{Q}_p$ with basis $(f_1, \dots, f_m)$.
We say that a lattice H is *diagonal* with respect to $(f_1, \dots, f_m)$ if it has the shape $H = \mathrm{Span}_{\mathbb{Z}_p}(p^{\nu_1} f_1, \dots, p^{\nu_m} f_m)$ for some relative integers $\nu_1, \dots, \nu_m$.
The *number of diffused digits* (with respect to $(f_1, \dots, f_m)$) of a given lattice H is the logarithm in base p of the index of H in the smallest diagonal lattice containing H.

Roughly speaking, the number of diffused digits of a lattice H with respect to the basis $(f_1, \dots, f_m)$ quantifies the quality of the basis for expressing the precision encoded by H: if H has no diffused digit, it is represented by a diagonal matrix and the precision splits properly into each coordinate. On the contrary, if H has diffused digits, writing the precision in the system of coordinates associated with $(f_1, \dots, f_m)$ leads to non optimal results. We remark furthermore that there always exists a basis in which the number of diffused digits of a given lattice H is zero: it suffices to take a basis whose $\mathbb{Z}_p$-span is H (such a basis always exists by Proposition 3.3).

Before going further, let us emphasize that, given a set of generators of H, it is not difficult to compute effectively its number of diffused digits of precision. Indeed, let M be a matrix whose row vectors span H. Pick in addition a diagonal lattice $H' = \mathrm{Span}_{\mathbb{Z}_p}(p^{\nu_1} f_1, \dots, p^{\nu_m} f_m)$ with $H \subset H'$. By projecting on the j-th coordinate we see that ν_j is at least the smallest valuation v_j of an entry of the j-th column of M. Conversely, one easily checks that the diagonal lattice $\mathrm{Span}_{\mathbb{Z}_p}(p^{v_1} f_1, \dots, p^{v_m} f_m)$ contains H. Thus the smallest diagonal matrix containing H is $\mathrm{Span}_{\mathbb{Z}_p}(p^{v_1} f_1, \dots, p^{v_m} f_m)$. It follows that the number of diffused digits of precision of H with respect to the basis $(f_1, \dots, f_m)$ is given by:

$$(v_1 + v_2 + \cdots + v_m) - \mathrm{val}_p(\det M). \tag{3.14}$$

If M is in Hermite normal form (see §3.1.2) this quantity vanishes if and only if M is indeed diagonal.

We now go back to our initial question: we want to analyze the optimal precision for the computation of $\varphi(v)$ for v is given at precision $H = \mathrm{Span}_{\mathbb{Z}_p}(p^{N_1} e_1, p^{N_2} e_2, \cdots, p^{N_n} e_n)$. The precision Lemma applied with φ itself (assuming that the hypotheses of the precision Lemma are fulfilled of course) yields the equality:

$$\varphi(v + H) = \varphi(v) + d\varphi_v(H).$$

The number of diffused digits of precision resulting from the computation of $\varphi(v)$ is then the number of diffused digits of the lattice $d\varphi_v(H)$. If we are working in the basis $(f_1, \dots, f_n)$, this number can be large, meaning that writing down the precision in this basis can be weak. However, if we are allowing a change of basis on the codomain, we can always reduce the number of diffused digits to 0; in such a basis, the precision can be made optimal!

Context of lazy/relaxed arithmetic

As we have explained in §2.4.1, the point of view of lazy/relaxed arithmetic differs from the zealous one on the following point: instead of fixing the precision on the input and propagating it to the outputs, it fixes a target precision and infers from it the precision needed on the inputs. Of course, in order to save time and memory, we would like to avoid the computation of unnecessary digits. Regarding precision, the question then becomes: what is the minimal number of digits we have to compute on the inputs in order to ensure a correct output at the requested precision?

This question translates into our precision language as follows: we fix some relative integers $M_1, \dots, M_m$ and look for relative integers $N_1, \dots, N_n$ for which:

$$\varphi(v + H) \subset \varphi(v) + H' \tag{3.15}$$

with $H = \mathrm{Span}_{\mathbb{Z}_p}(p^{N_1} f_1, \dots, p^{N_n} e_n)$ (the unknown) and $H' = \mathrm{Span}_{\mathbb{Z}_p}(p^{M_1} f_1, \dots, p^{M_m} f_m)$. Assuming that the precision Lemma applies for φ and H, Eq. (3.15) is equivalent to $d\varphi_v(H) \subset H'$,

i.e. $H \subset d\varphi_v^{-1}(H')$. The problem then reduces to find the largest diagonal lattice sitting in $d\varphi_v^{-1}(H')$.

The comparison with the zealous situation is instructive: in the zealous case, one had the lattice $d\varphi_v(H_{\text{Zeal}})$ and we were looking for the smallest diagonal lattice containing it. The problem is now "reversed" and can actually be related to the zealous problem using duality. Before going further, we need to recall basic facts about duality and lattices.

Given V a finite dimensional $\mathbb{Q}_p$-vector space, we define its *dual* $V^\star$ as the space of $\mathbb{Q}_p$-linear forms on V. We recall that there is a canonical isomorphism between V and its bidual $V^{\star\star}$; it maps an element $x \in V$ to the linear form taking $\ell \in V^\star$ to $\ell(x)$. We recall also that each basis $(v_1, \dots, v_d)$ of V has a dual basis $(v_1^\star, \dots, v_d^\star)$ defined by $v_j^\star(v_i) = 1$ if $i = j$ and $v_j^\star(v_i) = 0$ otherwise.

If $L \subset V$ is any $\mathbb{Z}_p$-module, its dual $L^\star$ is defined by as the subset of $V^\star$ consisting of linear forms $\ell : V \to \mathbb{Q}_p$ for which $\ell(L) \subset \mathbb{Z}_p$. Clearly $L^\star$ is a sub-$\mathbb{Z}_p$-module of $V^\star$. One can check that the construction $L \mapsto L^\star$ is involutive (*i.e.* $L^{\star\star} = L$), order-reversing (*i.e.* if $L_1 \subset L_2$ then $L_2^\star \subset L_1^\star$) and preserves diagonal lattices (*i.e.* if L is a diagonal lattice with respect to some basis then $L^\star$ is a diagonal lattice with respect to the dual basis).

Remark 3.21. If L is a lattice, so is $L^\star$. More precisely, if $(v_1, \dots, v_d)$ is a basis of V and M is a square matrix whose rows span L (with respect to the above basis), it is easily checked that $L^\star$ is spanned by the rows of ${}^t M^{-1}$ (with respect to the dual basis). On the contrary if L does not generate V as a $\mathbb{Q}_p$-vector space, the dual space $L^\star$ contains a $\mathbb{Q}_p$-line; conversely if L contains a $\mathbb{Q}_p$-line then $L^\star$ does not generate $V^\star$ over $\mathbb{Q}_p$.

Applying the above formalism to our situation, we find that the condition $H \subset d\varphi_v^{-1}(H')$ we want to ensure is equivalent to $d\varphi_v^{-1}(H')^\star \subset H^\star$. Our problem then reduces to finding the smallest diagonal lattice containing $d\varphi_v^{-1}(H')^\star$. By the analysis we have done in the zealous context, the integer N_i we are looking for then appears as the smallest valuation of an entry of i-th column of a matrix whose rows span the space $d\varphi_v^{-1}(H')^\star$ (with respect to the dual basis $(e_1^\star, \dots, e_n^\star)$). Computing $d\varphi_v^{-1}(H')^\star$ using Remark 3.21 (and taking care of the possible kernel of $d\varphi_v$), we finally find that N_i is the opposite of the smallest valuation of an entry of the i-th *row* of the matrix:

$$J(\varphi)_v \cdot \begin{pmatrix} p^{-M_1} & & \\ & \ddots & \\ & & p^{-M_m} \end{pmatrix}$$

where $J(\varphi)_v$ is the Jacobian matrix of φ at v (compare with (3.13)).

3.2.2 Everyday life examples revisited

We revisit the examples of §2.4.3 in light of the theory of p-adic precision.

The p-power map

As a training example, let us first examine the computation of the p-power map (though technically it was not treated in §2.4.3 but at the end of §2.4.2). Recall that we have seen that raising an invertible p-adic integer x at the power p gains one digit of precision: if x is given at precision $O(p^N)$, one can compute x^p at precision $O(p^{N+1})$.

We introduce the function $\varphi : \mathbb{Q}_p \to \mathbb{Q}_p$ taking x to x^p. It is apparently of class C^1 and its differential at x is the multiplication by $\varphi'(x) = px^{p-1}$. The Jacobian matrix of φ at x is then the 1×1 matrix whose unique entry is px^{p-1}. By the results of §3.2.1 (see particularly Eq. (3.12)), the optimal precision on x^p is the valuation of

$$(px^{p-1}) \times p^N = p^{N+1}x^{p-1}.$$

We recover this way the exponent $N+1$ thanks to the factor p which appears in the Jacobian. Unlike Lemma 2.18, the method we have just applied here teaches us in addition that the exponent $N+1$ cannot be improved.

Determinant

We take here $\varphi = \det : M_d(\mathbb{Q}_p) \to \mathbb{Q}_p$. We endow $M_d(\mathbb{Q}_p)$ with its canonical basis and let $x_{i,j}$ denote the corresponding coordinate functions: for $A \in M_d(\mathbb{Z}_p)$, $x_{i,j}(A)$ is the (i,j) entry of A. Fix temporarily a pair (i,j). Expanding the determinant according to the i-th row, we find that φ is an affine function of $x_{i,j}$ whose linear term is $(-1)^{i+j} x_{i,j} \cdot \det G_{i,j}$ where $G_{i,j}$ is the matrix obtained from the "generic" matrix $G = (x_{i,j})_{1 \leqslant i,j \leqslant d}$ by erasing the i-th row and the j-th column. Therefore $\frac{\partial \varphi}{\partial x_{i,j}} = (-1)^{i+j} \det G_{i,j}$. Evaluating this identity at some matrix $M \in M_d(\mathbb{Q}_p)$, we get the well-known formula:

$$\frac{\partial \varphi}{\partial x_{i,j}}(M) = (-1)^{i+j} \det M_{i,j} \tag{3.16}$$

where $M_{i,j}$ is the matrix obtained from M by erasing the i-th row and the j-th column. The Jacobian matrix of φ at M is then the column matrix (whose rows are indexed by the pairs (i,j) for $1 \leqslant i,j \leqslant d$) with entries $(-1)^{i+j} \det M_{i,j}$. According to the results of §3.2.1 (see especially Eq. (3.12)), the optimal precision for $\det M$, when the (i,j) entry of M is given at precision $O(p^{N_{i,j}})$, is:

$$O\big(p^{N'}\big) \quad \text{for} \quad N' = \min_{i,j}\big(N_{i,j} + \mathrm{val}(M_{i,j})\big). \tag{3.17}$$

Remember of course that this conclusion is only valid when the assumptions of the precision Lemma are all fulfilled. Nonetheless, relying on Proposition 3.17 (after having noticed that the determinant is a multivariate polynomial function), we can further write down explicit sufficient conditions for this to hold. These conditions take a particularly simple form when all the entries of M are given at the *same* precision $O(p^N)$: in this case, we find that the precision (3.17) is indeed the optimal precision as soon as $N' > \mathrm{val}(\det M)$, *i.e.* as soon as it is possible to ensure that $\det M$ does not vanish.

The general result above applies in particular to the matrix M considered as a running example in §2.4.3 (see Eq. (2.7)):

$$M = \begin{pmatrix} \ldots 0101110000 & \ldots 0011100000 & \ldots 1011001000 & \ldots 0011000100 \\ \ldots 1101011001 & \ldots 1101000111 & \ldots 0111001010 & \ldots 0101110101 \\ \ldots 0111100011 & \ldots 1011100101 & \ldots 0010100110 & \ldots 1111110111 \\ \ldots 0000111101 & \ldots 1101110011 & \ldots 0011010010 & \ldots 1001100001 \end{pmatrix} \in M_4(\mathbb{Z}_2). \tag{3.18}$$

Recall that all the entries of M were given at the same precision $O(2^{10})$. A simple computation (using zealous arithmetic) shows that the comatrix of M (*i.e.* the matrix whose (i,j) entry is $(-1)^{i+j} \det M_{i,j}$) is:

$$\mathrm{Comatrix}(M) = \begin{pmatrix} \ldots 1111000000 & \ldots 0001100000 & \ldots 1101000000 & \ldots 1001100000 \\ \ldots 0111000000 & \ldots 0011100000 & \ldots 1001000000 & \ldots 1011100000 \\ \ldots 0111000000 & \ldots 0001100000 & \ldots 0111000000 & \ldots 0011100000 \\ \ldots 1010000000 & \ldots 0111000000 & \ldots 1110000000 & \ldots 0011000000 \end{pmatrix}$$

We observe that the minimal valuation of its entries is 5 (it is reached on the second column), meaning that the optimal precision on $\det M$ is $O(2^{15})$. Coming back to §2.4.3, we see that the interval floating-point arithmetic reached this accuracy whereas zealous interval arithmetic missed many digits (without further help).

Computation of determinants with optimal precision. It is worth remarking that one can design a simple algorithm that computes the determinant of a p-adic matrix at the optimal precision *within the framework of zealous arithmetic* when the coefficients of the input matrix are all given at the same precision $O(p^N)$. This algorithm is based on a variation of the Smith normal form that we state now.

Proposition 3.22. *Any matrix $M \in M_d(\mathbb{Q}_p)$ admits a factorization of the following form:*

$$M = P \cdot \begin{pmatrix} a_1 & & \\ & \ddots & \\ & & a_d \end{pmatrix} \cdot Q \tag{3.19}$$

where P and Q are have determinant ± 1, the matrix in the middle is diagonal and its diagonal entries satisfy $\mathrm{val}(a_1) \leqslant \cdots \leqslant \mathrm{val}(a_d)$.

The factorization (3.19) is quite interesting for us for two reasons. First of all, it reveals the smallest valuation of an entry of the comatrix of M: it is

$$v = \mathrm{val}(a_1) + \mathrm{val}(a_2) + \cdots + \mathrm{val}(a_{d-1}). \tag{3.20}$$

Indeed, taking exterior powers, one proves the entries of $\mathrm{Comatrix}(M)$ and $\mathrm{Comatrix}(P^{-1}MQ^{-1})$ span the same $\mathbb{Z}_p$-submodule of $\mathbb{Q}_p$. Moreover, since $P^{-1}MQ^{-1}$ is diagonal, its comatrix is easily computed; we find:

$$\mathrm{Comatrix}(P^{-1}MQ^{-1}) = \begin{pmatrix} b_1 & & \\ & \ddots & \\ & & b_d \end{pmatrix} \quad \text{with} \quad b_i = a_1 \cdots a_{i-1} a_{i+1} \cdots a_d$$

(recall that $\mathrm{Comatrix}(A) = \det A \cdot A^{-1}$ if A is invertible) and we are done. The direct consequence of this calculation is the following: if the entries of M are all given at precision $O(p^N)$, the optimal precision on $\det M$ is $O(p^{N+v})$ (assuming $v < N$).

The second decisive advantage of the factorization (3.19) is that it can be computed without any loss of precision. The underlying algorithm basically follows the lines of the Hermite reduction we have presented in §3.1.2, except on the three following points:

- instead of choosing the pivot as an entry of smallest valuation on the working column, we choose as an entry of smallest valuation on the whole matrix,
- we do not only clear the entries behind the pivot but also the entries located on the right of the pivot,
- in order to keep the transformation matrices of determinant ± 1, we do not rescale rows.

For example, starting with our favorite matrix M (see Eq. (3.18) above), we first select the $(2,1)$ entry (which has valuation 0), we put it on the top left corner and use it as pivot to clear the entries on the first row and the first column. Doing so, we obtain the intermediate form:

$$\begin{pmatrix} \ldots 1101011001 & 0 & 0 & 0 \\ 0 & \ldots 0001010000 & \ldots 0101101000 & \ldots 0100010100 \\ 0 & \ldots 0111101000 & \ldots 0101011000 & \ldots 1100100000 \\ 0 & \ldots 1010010000 & \ldots 0011100000 & \ldots 0110011000 \end{pmatrix}$$

and we retain that we have swapped two rows, implying that the determinant of the above matrix is the opposite to the determinant of M. We now select the $(2,4)$ entry (which as valuation 2) and continue the reduction:

$$\begin{pmatrix} \ldots 1101011001 & 0 & 0 & 0 \\ 0 & \ldots 0100010100 & 0 & 0 \\ 0 & 0 & \ldots 0100011000 & \ldots 0101101000 \\ 0 & 0 & \ldots 0000110000 & \ldots 0000110000 \end{pmatrix}$$

We observe that we have not lost any precision in this step (all the entries of the matrix above are known at precision $O(2^{10})$), and this even if our pivot had positive valuation. It is actually

a general phenomenon due to the fact that we always choose the pivot of smallest valuation. Continuing this process, we end up with the matrix:

$$\begin{pmatrix} \dots 1101011001 & 0 & 0 & 0 \\ 0 & \dots 0100010100 & 0 & 0 \\ 0 & 0 & \dots 0100011000 & 0 \\ 0 & 0 & 0 & \dots 1101100000 \end{pmatrix}$$

Now multiplying the diagonal entries, we find $\det M = 2^{10} \times ...01101$ with precision $O(2^{15})$!

For a general M, the same precision analysis works. Indeed, when the entries of M are all given at precision $O(p^N)$, the reduction algorithm we have sketched above outputs a diagonal matrix whose determinant is the same as the one of M and whose diagonal entries $a_1, \dots, a_d$ are all known at precision $O(p^N)$. By Proposition 2.3 (see also Remark 2.4), zealous arithmetic can compute the product $a_1 a_2 \cdots a_d$ at precision $O(p^{N+w})$ with

$$w = \mathrm{val}(a_1) + \cdots + \mathrm{val}(a_d) - \max_i \mathrm{val}(a_i).$$

We then recover the optimal precision given by Eq. (3.20).

Characteristic polynomial

The case of characteristic polynomials has many similarities with that of determinants. We introduce the function $\varphi : M_d(\mathbb{Q}_p) \to \mathbb{Q}_p[X]_{<d}$ that maps a matrix M to $\varphi(M) = \det(XI_d - M) - X^d$ where I_d is the identity matrix of size d. The renormalization (consisting in subtracting X^d) is harmless but needed in order to ensure that φ takes its values in a vector space (and not an affine space) and has a surjective differential almost everywhere. The partial derivatives of φ are:

$$\frac{\partial \varphi}{\partial x_{i,j}}(M) = (-1)^{i+j} \cdot \det(XI_d - M)_{i,j}$$

where $(XI_d - M)_{i,j}$ is the matrix obtained from $XI_d - M$ by deleting the i-th row and the j-th column (compare with Eq. (3.16)). The Jacobian matrix $J(\varphi)_M$ of φ at M is then a matrix with d^2 rows, indexed by pairs (i,j) with $1 \leqslant i,j \leqslant d$, and d columns whose row with index (i,j) contains the coefficients of the polynomial $(-1)^{i+j} \cdot \det(XI_d - M)_{i,j}$. Using again the results of §3.2.1, the optimal precision on each coefficient of the characteristic polynomial of M can be read off from $J(\varphi)_M$.

Let us explore further our running example: the matrix M given by Eq. (3.18) whose entries are all known at precision $O(2^{10})$. A direct computation leads to the Jacobian matrix displayed on the left of Figure 3.2. We observe that the minimal valuation of an entry of the column of X^j is 0, 0, 2 and 5 when j is 3, 2, 1 and 0 respectively. Consequently the optimal precision on the coefficients of X^3, X^2, X and on the constant coefficient are $O(2^{10})$, $O(2^{10})$, $O(2^{12})$ and $O(2^{15})$ respectively. Coming back to §2.4.3, we notice that floating-point arithmetic found all the correct digits. We observe moreover that the Hermite normal form of the Jacobian matrix (shown on the right on Figure 3.2) is diagonal. There is therefore no diffused digit of precision for this example; in other words the precision written as $O(p^{10})X^3 + O(p^{10})X^2 + O(p^{12})X + O(p^{15})$ is sharp.

Remark 3.23. Making more extensive tests, we conclude that p-adic floating point arithemtic was pretty lucky with the above example: in general, it indeed sometimes finds all (or almost all) relevant digits but it may also sometimes not be more accurate than zealous arithmetic. We refer to [20] for precise statistics thereupon.

Let us now have a look at the matrix $N = I_4 + M$. Playing the same game as before, we obtain the Jacobian matrix displayed on the left of Figure 3.3. Each column of this matrix contains an entry of valuation zero. The optimal precision on each individual coefficient of the characteristic polynomial of N is then no more than $O(p^{10})$. However the Hermite reduced form of the Jacobian

$$
\begin{array}{c}
(i,j) \\
(1,1) \\
(1,2) \\
(1,3) \\
(1,4) \\
(2,1) \\
(2,2) \\
(2,3) \\
(2,4) \\
(3,1) \\
(3,2) \\
(3,3) \\
(3,4) \\
(4,1) \\
(4,2) \\
(4,3) \\
(4,4)
\end{array}
\begin{array}{c}
\begin{array}{cccc} X^3 & X^2 & X & 1 \end{array} \\
\begin{pmatrix}
1 & \ldots 0110110010 & \ldots 0111111000 & \ldots 0001000000 \\
0 & \ldots 0011100000 & \ldots 1011010100 & \ldots 1110100000 \\
0 & \ldots 1011001000 & \ldots 1001001000 & \ldots 0011000000 \\
0 & \ldots 0011000100 & \ldots 1111100100 & \ldots 0110100000 \\
0 & \ldots 1101011001 & \ldots 1010010000 & \ldots 1001000000 \\
1 & \ldots 1110001001 & \ldots 1001001100 & \ldots 1100100000 \\
0 & \ldots 0111001010 & \ldots 0110011000 & \ldots 0111000000 \\
0 & \ldots 0101110101 & \ldots 0111111100 & \ldots 0100100000 \\
0 & \ldots 0111100011 & \ldots 1010000000 & \ldots 1001000000 \\
0 & \ldots 1011100101 & \ldots 1110100000 & \ldots 1110100000 \\
1 & \ldots 0011101000 & \ldots 1001000100 & \ldots 1001000000 \\
0 & \ldots 1111110111 & \ldots 1111100100 & \ldots 1100100000 \\
0 & \ldots 0000111101 & \ldots 0010111000 & \ldots 0110000000 \\
0 & \ldots 1101110011 & \ldots 1101011000 & \ldots 1001000000 \\
0 & \ldots 0011010010 & \ldots 1101001000 & \ldots 0010000000 \\
1 & \ldots 1010100011 & \ldots 0111010000 & \ldots 1101000000
\end{pmatrix}
\end{array}
\xrightarrow{\text{Hermite}}
\begin{array}{c}
\begin{array}{cccc} X^3 & X^2 & X & 1 \end{array} \\
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 2^2 & 0 \\
0 & 0 & 0 & 2^5 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{pmatrix}
\end{array}
$$

Figure 3.2: The Jacobian of the characteristic polynomial at M

$$
\begin{array}{c}
(i,j) \\
(1,1) \\
(1,2) \\
(1,3) \\
(1,4) \\
(2,1) \\
(2,2) \\
(2,3) \\
(2,4) \\
(3,1) \\
(3,2) \\
(3,3) \\
(3,4) \\
(4,1) \\
(4,2) \\
(4,3) \\
(4,4)
\end{array}
\begin{array}{c}
\begin{array}{cccc} X^3 & X^2 & X & 1 \end{array} \\
\begin{pmatrix}
1 & \ldots 0110101111 & \ldots 1010010111 & \ldots 1111111001 \\
0 & \ldots 0011100000 & \ldots 0100010100 & \ldots 0110101100 \\
0 & \ldots 1011001000 & \ldots 0010111000 & \ldots 0101000000 \\
0 & \ldots 0011000100 & \ldots 1001011100 & \ldots 1010000000 \\
0 & \ldots 1101011001 & \ldots 1111011110 & \ldots 1100001001 \\
1 & \ldots 1110000110 & \ldots 1100111101 & \ldots 0001011100 \\
0 & \ldots 0111001010 & \ldots 1000000100 & \ldots 0111110010 \\
0 & \ldots 0101110101 & \ldots 1100010010 & \ldots 0010011001 \\
0 & \ldots 0111100011 & \ldots 1010111010 & \ldots 0110100011 \\
0 & \ldots 1011100101 & \ldots 0111010110 & \ldots 1011100101 \\
1 & \ldots 0011100101 & \ldots 0001110111 & \ldots 0011100011 \\
0 & \ldots 1111110111 & \ldots 1111110110 & \ldots 1100110011 \\
0 & \ldots 0000111101 & \ldots 0000111110 & \ldots 0100000101 \\
0 & \ldots 1101110011 & \ldots 0001110010 & \ldots 1001011011 \\
0 & \ldots 0011010010 & \ldots 0110100100 & \ldots 1000001010 \\
1 & \ldots 1010100000 & \ldots 0010001101 & \ldots 0000010010
\end{pmatrix}
\end{array}
\xrightarrow{\text{Hermite}}
\begin{array}{c}
\begin{array}{cccc} X^3 & X^2 & X & 1 \end{array} \\
\begin{pmatrix}
1 & 0 & 1 & 30 \\
0 & 1 & 2 & 29 \\
0 & 0 & 2^2 & 28 \\
0 & 0 & 0 & 2^5 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{pmatrix}
\end{array}
$$

Figure 3.3: The Jacobian of the characteristic polynomial at I_4+M

is now no longer diagonal, meaning that diffused digits of precision do appear. One can moreover count them using Eq. (3.14) (applied to the Hermite normal form of the Jacobian): we find 7. This means that the precision

$$O(p^{10})X^3 + O(p^{10})X^2 + O(p^{10})X + O(p^{10})$$

is not sharp; more precisely, after a suitable base change on $\mathbb{Q}_p[X]_{<d}$, it should be possible to visualize seven more digits. Given that $\chi_N(X) = \chi_M(X-1)$, this base change is of course the one which is induced by the change of variable $X \mapsto X-1$.

LU factorization

Given a positive integer d, define $L_d(\mathbb{Z}_p)$ as the subspace of $M_d(\mathbb{Z}_p)$ consisting of lower triangular matrices with zero diagonal. Similarly let $U_d(\mathbb{Z}_p)$ be the subspace space of $M_d(\mathbb{Z}_p)$ consisting of upper triangular matrices. Clearly $L_d(\mathbb{Z}_p)$ and $U_d(\mathbb{Z}_p)$ are p-adic vector spaces of respective dimensions $\frac{d(d-1)}{2}$ and $\frac{d(d+1)}{2}$. Let $\mathcal{U}$ be the open subset of $M_d(\mathbb{Z}_p)$ defined by the non-vanishing of all principal minors. We define two mappings:

$$\varphi : \mathcal{U} \to L_d(\mathbb{Z}_p),\ M \mapsto L - I_d \quad \text{and} \quad \psi : \mathcal{U} \to L_d(\mathbb{Z}_p),\ M \mapsto U$$

where $M = LU$ is the (unique) LU factorization of M. In order to compute the differentials of φ and ψ[22], we simply differentiate the defining relation $M = LU$. We get this way the relation $dM = dL \cdot U + L \cdot dU$. We rewrite it as:

$$L^{-1} \cdot dM \cdot U^{-1} = \big(L^{-1} \cdot dL\big) + \big(dU \cdot U^{-1}\big)$$

and observe that the first summand of the right hand side is strictly lower triangular whereas the second summand of upper triangular. We therefore derive $dL = L \cdot \mathrm{Lo}\big(L^{-1} \cdot dM \cdot U^{-1}\big)$ and $dU = \mathrm{Up}\big(L^{-1} \cdot dM \cdot U^{-1}\big) \cdot U$ where Lo (resp. Up) is the projection on the first factor (resp. on the second factor) of the decomposition $M_d(\mathbb{Z}_p) = L_d(\mathbb{Z}_p) \oplus U_d(\mathbb{Z}_p)$. The differentials of φ and ψ at $M \in \mathcal{U}$ are then given by the linear mappings:

$$\begin{aligned} d\varphi_M &: M_d(\mathbb{Z}_p) \to L_d(\mathbb{Z}_p), \quad dM \mapsto L \cdot \mathrm{Lo}\big(L^{-1} \cdot dM \cdot U^{-1}\big) \\ d\psi_M &: M_d(\mathbb{Z}_p) \to U_d(\mathbb{Z}_p), \quad dM \mapsto \mathrm{Up}\big(L^{-1} \cdot dM \cdot U^{-1}\big) \cdot U \end{aligned}$$

where L and U are the "L-part" and the "U-part" of the LU factorization of M respectively.

As an example, take again the matrix M given by Eq. (3.18). The Jacobian matrix of φ at this point is the matrix displayed on Figure 3.4. Looking at the minimal valuation of the entries of $J(\varphi)_M$ column by column, we find the optimal precision for each entry of the matrix L (defined as the L-part L of the LU factorization of M):

$$\begin{pmatrix} - & - & - & - \\ O(2^2) & - & - & - \\ O(2^2) & O(2^9) & - & - \\ O(2^2) & O(2^{10}) & O(2^7) & - \end{pmatrix} \tag{3.21}$$

(we recall that the entries of M were all initially given at precision $O(2^{10})$). Comparing with the numerical results obtained in §2.4.3 (see Eq. (2.8)), we see that floating-point arithmetic, one more time, found all the relevant digits.

A closer look at Figure 3.4 shows that the lattice $d\varphi_M(2^{10}M_4(\mathbb{Z}_2)) = 2^{10} \cdot d\varphi_M(M_4(\mathbb{Z}_2))$ has exactly 9 diffused digits (apply Eq. (3.14)). The precision (3.21) is then globally not optimal although it is on each entry separately. For example, if $\ell_{i,j}$ denotes the (i,j) entry of L, the linear

[22] These two functions are indeed differentiable. One can prove this by noticing for instance that the entries of L and U are given by explicit multivariate rational fractions [46, §1.4].

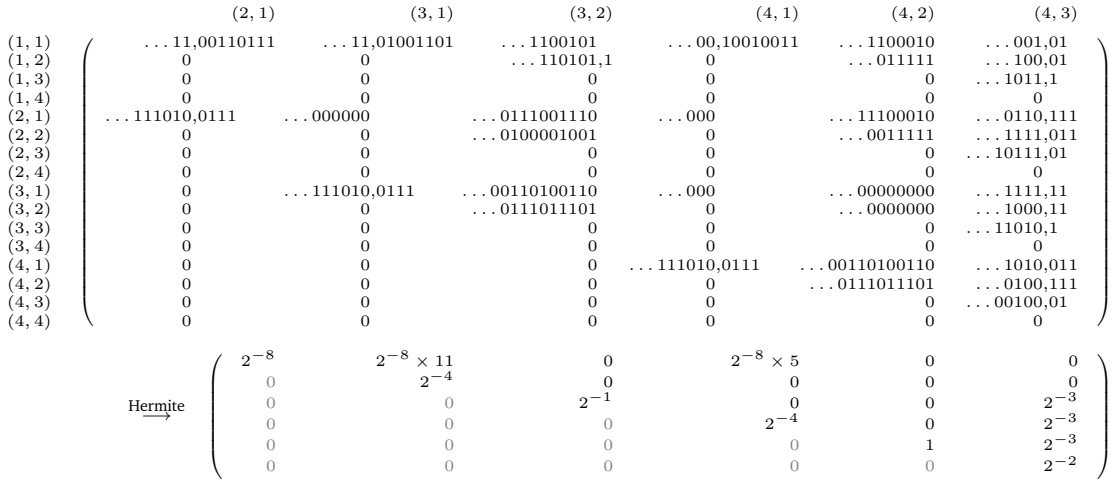

Figure 3.4: The Jacobian of the L-part of the LU factorization at M

combination $\ell_{3,2} - 11 \cdot \ell_{3,1}$ can be known at precision $O(2^6)$ although $\ell_{3,2}$ and $\ell_{3,1}$ cannot be known at higher precision than $O(2^2)$. It is remarkable that floating-point arithmetic actually "saw" these diffused digits; indeed, taking the values computed in floating-point arithmetic for $\ell_{3,2}$ and $\ell_{3,1}$, we compute:

$$\ell_{3,2} - 11 \cdot \ell_{3,1} = \ldots 0000010111.$$

It turns out that the last six digits of the latter value are correct!

Remark 3.24. For a general input $d \times d$ matrix M whose principal minors have valuation $v_1, \ldots, v_d$, we expect that zealous arithmetic looses about $\Omega(v_1 + \cdots + v_d)$ significant digits (for the computation of the LU factorization of M using standard Gaussian elimination) while floating-point arithmetics looses only $O(\max(v_1, \ldots, v_d))$ significant digits. If M is picked at random in the space $M_d(\mathbb{Z}_p)$ (equipped with its natural Haar measure), the former is $O(\frac{d}{p-1})$ on average while the second is $O(\log_p d)$ on average [16].

Bézout coefficients

We now move to commutative algebra. We address the question of the computation of the Bézout coefficients of two monic polynomials of degree d which are supposed to be coprime. Let $\mathcal{U}$ be the subset of $\mathbb{Q}_p[X]_{<d} \times \mathbb{Q}_p[X]_{<d}$ consisting of pairs $(\tilde{P}, \tilde{Q})$ for which $P = X^d + \tilde{P}$ and $Q = X^d + \tilde{Q}$ are coprime. Observe that $\mathcal{U}$ is defined by the non-vanishing of some resultant and so is open. We introduce the function:

$$\begin{array}{rccl} \varphi : & \mathcal{U} & \longrightarrow & \mathbb{Q}_p[X]_{<d} \times \mathbb{Q}_p[X]_{<d} \\ & (\tilde{P}, \tilde{Q}) & \mapsto & (U, V) \quad \text{s.t.} \quad UP + VQ = 1 \end{array}$$

for $P = X^d + \tilde{P}$ and $Q = X^d + \tilde{Q}$. (Note that U and V are uniquely determined thanks to the conditions on the degree.) Once again, the differential of φ can be easily computed by differentiating the defining relation $UP + VQ = 1$; indeed doing so, we immediately get:

$$\big(dU \cdot P\big) + \big(dV \cdot Q\big) = dR \tag{3.22}$$

for $dR = -U{\cdot}dP - V{\cdot}dQ$. Eq (3.22) gives $dU{\cdot}P \equiv dR \pmod{Q}$ for what we derive $dU \equiv U{\cdot}dR \pmod{Q}$. Comparing degrees we find $dU = U{\cdot}dR \bmod Q$. Similarly $dV = V{\cdot}dR \bmod P$. The differential of φ at a pair $(\tilde{P}, \tilde{Q})$ is then the linear mapping:

$$\begin{array}{rccl} d\varphi_{(\tilde{P},\tilde{Q})} : & \mathbb{Q}_p[X]_{<d} \times \mathbb{Q}_p[X]_{<d} & \longrightarrow & \mathbb{Q}_p[X]_{<d} \times \mathbb{Q}_p[X]_{<d} \\ & (dP, dQ) & \mapsto & (dU, dV) = \big(U{\cdot}dR \bmod Q,\, V{\cdot}dR \bmod P\big) \\ & & & \text{where} \quad dR = -U \cdot dP - V \cdot dQ. \end{array}$$

	dU				dV			
(dP, dQ)	X^3	X^2	X	1	X^3	X^2	X	1
$(X^3, 0)$	$\ldots 0100111$	$\ldots 0001000$	$\ldots 1000000$	$\ldots 1010000$	$\ldots 1011001$	$\ldots 0100000$	$\ldots 1000000$	$\ldots 0100000$
$(X^2, 0)$	$\ldots 1010000$	$\ldots 0010111$	$\ldots 0111000$	$\ldots 1000000$	$\ldots 0110000$	$\ldots 0101001$	$\ldots 0110000$	$\ldots 0000000$
$(X, 0)$	$\ldots 1000000$	$\ldots 0010000$	$\ldots 1010111$	$\ldots 0111000$	$\ldots 1000000$	$\ldots 1110000$	$\ldots 1101001$	$\ldots 0110000$
$(1, 0)$	$\ldots 1111000$	$\ldots 1101000$	$\ldots 0011000$	$\ldots 1010111$	$\ldots 0001000$	$\ldots 0111000$	$\ldots 0001000$	$\ldots 1001001$
$(0, X^3)$	$\ldots 1011001$	$\ldots 1111000$	$\ldots 0010000$	$\ldots 1110000$	$\ldots 0100111$	$\ldots 1100000$	$\ldots 1100000$	$\ldots 1100000$
$(0, X^2)$	$\ldots 1110000$	$\ldots 0101001$	$\ldots 0001000$	$\ldots 0010000$	$\ldots 0010000$	$\ldots 0010111$	$\ldots 0010000$	$\ldots 0100000$
$(0, X)$	$\ldots 0010000$	$\ldots 0100000$	$\ldots 0011001$	$\ldots 0001000$	$\ldots 1110000$	$\ldots 0100000$	$\ldots 1100111$	$\ldots 1010000$
$(0, 1)$	$\ldots 1001000$	$\ldots 0101000$	$\ldots 1011000$	$\ldots 0011001$	$\ldots 0111000$	$\ldots 0111000$	$\ldots 1001000$	$\ldots 0000111$

Figure 3.5: The Jacobian of the "Bézout function" at (P, Q)

It is important to note that $d\varphi_{(\tilde{P},\tilde{Q})}$ is never injective because it maps $(-V, U)$ to 0. Consequently, it is never surjective either and one cannot apply the precision Lemma to it. The easiest way to fix this issue is to decompose φ as $\varphi = (\varphi_U, \varphi_V)$ and to apply the precision Lemma to each component separately (though this option leads to less accurate results).

For the particular example we have studied in §2.4.3, namely:

$$\begin{aligned} P = X^4 &+ (\ldots 1101111111)\, X^3 + (\ldots 0011110011)\, X^2 \\ &+ (\ldots 1001001100)\, X \;\; + (\ldots 0010111010) \\ Q = X^4 &+ (\ldots 0101001011)\, X^3 + (\ldots 0111001111)\, X^2 \\ &+ (\ldots 0100010000)\, X \;\; + (\ldots 1101000111) \end{aligned}$$

the Jacobian matrix we find is shown on Figure 3.5. Each column of this matrix contains an entry of valuation zero, meaning that the optimal precision on each coefficient of U and V is $O(2^{10})$. Observe that it was not reached by interval arithmetic or floating-point arithmetic! Continuing our analysis, we further observe that the lattice generated by the dU-part (resp. the dV-part) of $J(\varphi)_{(\tilde{P},\tilde{Q})}$ has no diffused digit. The precision

$$O(2^{10})X^3 + O(2^{10})X^2 + O(2^{10})X + O(2^{10})$$

on both U and V is then optimal (but the joint precision induced on the pair (U, V) is not).

Another option for fixing the default of surjectivity of $d\varphi_{(P,Q)}$ is to observe that φ in fact takes its value in the hyperplane H of $\mathbb{Q}_p[X]_{<d} \times \mathbb{Q}_p[X]_{<d}$ consisting of pairs of polynomials (P, Q) whose coefficients in degree $d-1$ agree. Restricting the codomain of φ to H, we then obtain a well-defined mapping whose differential is almost everywhere surjective. Applying the precision Lemma, we find that the pair (U, V) has 16 diffused digits!

Remark 3.25. A careful study of the precision in Euclidean algorithm is presented in [17]: we prove in this reference that if P and Q are two monic polynomials of degree d, Euclidean algorithm executed within the framework of zealous arithmetic (resp. floating-point arithmetic) looses $\Omega(v_0 + \cdots + v_{d-1})$ (resp. $O(\max(v_0, \ldots, v_{d-1}))$) digits on each coefficient where v_j is the valuation of the j-th scalar subresultant of P and Q. Moreover, for random polynomials, we have by [17, Corollary 3.6]:

$$\begin{aligned} \mathbb{E}\big[v_0 + \cdots + v_{d-1}\big] &\geqslant \tfrac{d}{p-1} \\ \mathbb{E}\big[\max(v_0, \ldots, v_{d-1})\big] &\leqslant \log_p d + O\big(\sqrt{\log_p d}\big). \end{aligned}$$

Compare this result with the case of LU factorization (see Remark 3.24).

Polynomial evaluation and interpolation

Recall that the last example considered in §2.4.3 was about evaluation and interpolation of polynomials: precisely, starting with a polynomial $P \in \mathbb{Z}_p[X]$ of degree d, we first evaluated it at the points $0, 1, \ldots, d$ and then reconstructed it by interpolation from the values $P(0), P(1), \ldots, P(d)$. We have observed that this problem seemed to be numerically highly

unstable: for example, for $d = 19$ and an initial polynomial P given at precision $O(2^{10})$, we were only able to reconstruct a few number of digits and even found not integral coefficients in many places (see Figure 2.3). We propose here to give a theoretical explanation of this phenomenon.

Consider the function $\varphi : \mathbb{Q}_p[X]_{\leqslant d} \to \mathbb{Q}_p^{d+1}$, $P \mapsto (P(0), P(1), \dots, P(d))$. Note that it is linear, so that $d\varphi_P = \varphi$ for all P. The numerical stability of our evaluation problem is governed by the lattice $H = \varphi(p^N \mathbb{Z}_p[X]_{\leqslant d}) = p^N \cdot \varphi(\mathbb{Z}_p[X]_{\leqslant d})$ where $O(p^N)$ is the initial precision we have on each coefficient of P (assuming for simplicity that it is the same for all coefficients). Applying φ to the standard basis of $\mathbb{Z}_p[X]_{\leqslant d}$, we find that H is generated by the row vectors of the Vandermonde matrix:

$$M = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 0 & 1 & 2 & \cdots & d \\ 0 & 1 & 2^2 & \cdots & d^2 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 1 & 2^d & \cdots & d^d \end{pmatrix}.$$

According to Eq. (3.14), the number of diffused digits of H is then equal to the p-adic valuation of the determinant of M whose value is:

$$\det M = \prod_{0 \leqslant i < j \leqslant d} (j - i) = 1! \times 2! \times \cdots \times d!.$$

In order to estimate its p-adic valuation, we recall the following result.

Proposition 3.26 (Legendre's formula)**.** *For all positive integer n, we have:*

$$\mathrm{val}_p(n!) = \left\lfloor \frac{n}{p} \right\rfloor + \left\lfloor \frac{n}{p^2} \right\rfloor + \cdots + \left\lfloor \frac{n}{p^k} \right\rfloor + \cdots$$

where $\lfloor \cdot \rfloor$ is the usual floor function.

Proof. Exercise. □

It follows in particular from Legendre's formula that $\mathrm{val}_p(i!) \geqslant \frac{i}{p} - 1$, from what we derive:

$$\mathrm{val}_p(\det M) \geqslant \sum_{i=1}^{d} \left(\tfrac{i}{p} - 1\right) = \frac{d(d+1)}{2p} - d.$$

When d is large compared to p, the number of diffused digits is much larger than the number of p-adic numbers on which these diffused digits "diffuse" (which are the $d+1$ values $P(0), \dots, P(d)$). They will then need to have an influence at high precision[23]. This is why it is so difficult to get a sharp precision on the tuple $(P(0), \dots, P(d))$. Concretely the n-th digits of the coefficients of P may influence the $(n + \frac{d}{2p})$-th digit of some linear combination of the $P(i)$'s and conversely the n-th digits of the $P(i)$'s may influence the $(n - \frac{d}{2p})$-th digits of the coefficients of P. In other words, in order to be able to recover all the coefficients of P at the initial precision $O(p^N)$, one should have used at least $N + \frac{d}{2p}$ digits of precision (in the model of floating-point arithmetic).

Evaluation at general points. Until now, we have only considered evaluation at the first integers. We may wonder whether to what extend the instability we have observed above is related to this particular points. It turns out that it is totally independant and that similar behaviors show up with any set of evaluation points. Indeed, let $a_0, \dots, a_d$ be pairwise distinct elements of $\mathbb{Z}_p$ and

[23] Roughly speaking, if we want to dispatch n (diffused) digits between m places, we have to increase the precision by at least $\lceil \frac{n}{m} \rceil$ digits; this is the pigeonhole principle.

consider the function $\varphi : \mathbb{Q}_p[X]_{\leqslant d} \to \mathbb{Q}_p^{d+1}$, $P \mapsto (P(a_0), P(a_1), \dots, P(a_d))$. It is linear and its Jacobian at any point is the following Vandermonde matrix:

$$\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ a_0 & a_1 & a_2 & \cdots & a_d \\ a_0^2 & a_1^2 & a_2^2 & \cdots & a_d^2 \\ \vdots & \vdots & \vdots & & \vdots \\ a_0^d & a_1^d & a_2^d & \cdots & a_d^d \end{pmatrix}$$

whose determinant is $V(a_0, \dots, a_d) = \prod_{0 \leqslant i < j \leqslant d} (a_i - a_j)$. Thanks to Eq. (3.14), the number of diffused digits of the lattice $\varphi(p^N \mathbb{Z}_p[X]_{\leqslant d})$ is the valuation of $V(a_0, \dots, a_d)$. The next lemma ensures that this number is always as large as it was in the particular case we have considered at first (*i.e.* $a_i = i$ for all i).

Lemma 3.27. *For all $a_0, \dots, a_d \in \mathbb{Z}_p$, we have:*

$$\operatorname{val}\big(V(a_0, \dots, a_d)\big) \geqslant \operatorname{val}\big(1! \times 2! \times \cdots \times d!\big).$$

Proof. Given a finite subset $A \subset \mathbb{Z}_p$, we denote by $\nu(A)$ the valuation of $V(x_1, \dots, x_n)$ where the x_i's are the elements of A. We note that $V(x_1, \dots, x_n)$ depends up to a sign to the way the elements of A are enumerated but $\nu(A)$ depends only on A. We are going to prove by induction on the cardinality n of A that $\nu(A) \geqslant \operatorname{val}\big(1! \times 2! \times \cdots \times (n-1)!\big)$.

If A has cardinality 2, the result is obvious. Let us now consider a finite subset A of $\mathbb{Z}_p$ of cardinality n. For $i \in \{1, \dots, p\}$, let A_i be the subset of A consisting of elements which are congruent to i modulo p and write $n_i = \operatorname{Card} A_i$. Define in addition B_i as the set of $\frac{x-r}{p}$ for x varying in A_i. Clearly B_i is a subset of $\mathbb{Z}_p$ of cardinality n_i. Moreover, we easily check that:

$$\nu(A) = \sum_{i=1}^{p} \nu(A_i) = \sum_{i=1}^{p} \left(\frac{n_i\,(n_i-1)}{2} + \nu(B_i) \right). \tag{3.23}$$

Define the function $v : \mathbb{N} \to \mathbb{N}$, $n \mapsto \frac{n(n-1)}{2} + \operatorname{val}(1!) + \cdots + \operatorname{val}((n-1)!)$. It follows from the induction hypothesis that $\nu(A) \geqslant v(n_1) + \cdots + v(n_p)$. Observe that $f(n+1) - f(n) = n + \operatorname{val}(n!)$ and deduce from this that $f(n) + f(m) \leqslant f(n+1) + f(m-1)$ as soon as $m \geqslant n + 2$. Applying again and again this inequality and noting in addition that $n_1 + \cdots + n_p = n$, we end up with:

$$v(n_1) + \cdots + v(n_p) \geqslant r \cdot v(q+1) + (p-r) \cdot v(q) \tag{3.24}$$

where q and r are the quotient and the remainder of the Euclidean division of n by p respectively. It is enough to prove that the right hand side of Eq. (3.24) is equal to $\operatorname{val}(1!) + \cdots + \operatorname{val}((n-1)!)$. This follows from Eq. (3.23) applied with the particular set $A = \{0, 1, \dots, n-1\}$. □

Remark 3.28. The above proof shows in a similar fashion that if $a_0, \dots, a_d$ are integers, then $V(a_0, \dots, a_d)$ is divisible by $1! \times 2! \times \cdots \times d!$.

The conclusion of the above analysis is that the evaluation-interpolation strategy — which is used for instance in FFT — cannot be used in the p-adic setting without further care. Precisely, from this point of view, one should really think of $\mathbb{Z}_p$ as if it were a ring of characteristic p and reuse all the techniques developed in the case of finite fields for avoiding divisions by p.

3.2.3 Newton iteration

We have already seen that Hensel's Lemma is a powerful tool in the p-adic context. However, as highlighted in §2.1.3, it may have a strange behavior regarding precision. Below we use the precision Lemma in order to study carefully this phenomenon. We also take the opportunity to extend the Newton scheme (on which Hensel's Lemma is based) to the framework of multivariate functions of class C^2.

Functions of class C^2 in the p-adic setting

Recall that, given two $\mathbb{Q}_p$-vector spaces E and F, we denote by $\mathcal{L}(E,F)$ the space of linear mappings from E to F. Similarly, Given three $\mathbb{Q}_p$-vector spaces E_1, E_2 and F, we denote by $\mathcal{B}(E_1\times E_2,F)$ the space of bilinear functions $E_1\times E_2\to F$. We recall that there exist canonical isomorphisms — the so-called *curryfication isomorphisms* — between $\mathcal{B}(E_1\times E_2,F)$, $\mathcal{L}(E_1,\mathcal{L}(E_2,F))$ and $\mathcal{L}(E_2,\mathcal{L}(E_1,F))$. When $E_1=E_2=E$, we define further $\mathcal{B}^{\mathrm{s}}(E\times E)$ as the subspace of $\mathcal{B}(E\times E,F)$ consisting of *symmetric* bilinear functions.

We recall also that when E and F are endowed with norms, the space $\mathcal{L}(E,F)$ inherits the norm operator defined by $\|f\|_{\mathcal{L}(E,F)}=\sup_{x\in B_E(1)}\|f(x)\|_F$. Similarly, we endow $\mathcal{B}(E_1\times E_2,F)$ with the norm $\|\cdot\|_{\mathcal{B}(E_1\times E_2,F)}$ defined by $\|b\|_{\mathcal{B}(E_1\times E_2,F)}=\sup_{x_1\in B_{E_1}(1),x_2\in B_{E_2}(1)}\|b(x_1,x_2)\|_F$. The curryfication isomorphisms do preserve the norm.

The definition of "being of class C^2" is inspired from the alternative definition of "being of class C^1" provided by Remark 3.14 (see also Remark 3.12).

Definition 3.29. Let E and F be two normed finite dimensional vector spaces and let U be an open subset of E. A function $f:U\to F$ is of *class* C^2 if there exist:

- a function $df:U\to\mathcal{L}(E,F)$, $x\mapsto df_x$
- a function $d^2f:U\to\mathcal{B}^{\mathrm{s}}(E\times E,F)$, $x\mapsto d^2f_x$
- a covering $(U_i)_{i\in I}$ of U, and
- for all $i\in I$, a *continuous* real-valued function $\varepsilon_i:\mathbb{R}^+\to\mathbb{R}^+$ vanishing at 0

such that, for all $i\in I$ and all $x,y\in U_i$:

$$\|f(y)-f(x)-df_x(y-x)-\tfrac{1}{2}d^2f_x(y-x,y-x)\|_F\leqslant\|y-x\|_E^2\cdot\varepsilon_i\big(\|y-x\|_E\big)\tag{3.25}$$

Proposition 3.30. *Let E and F be two normed finite dimensional vector spaces and let U be an open subset of E. Let $f:U\to F$ be a function of class C^2. Then:*

(i) The function f is of class C^1 on U and its differential is the function df of Definition 3.29;

(ii) the function df is of class C^1 on U as well and its differential is the function d^2f (viewed as a element of $\mathcal{L}(E,\mathcal{L}(E,F))$) of Definition 3.29.

Proof. We consider the data df, d^2f, U_i, ε_i given by Definition 3.29. Without loss of generality, we may assume that the U_i's are all balls and that the ε_i's are all non-decreasing functions. By this additional assumption, U_i is compact; the function $x\mapsto\|d^2f_x\|_{\mathcal{B}(E\times E,F)}$ is then bounded on U_i by a constant C_i. Therefore Eq. (3.25) implies:

$$\|f(y)-f(x)-df_x(y-x)\|_F\leqslant\|y-x\|_E^2\cdot\max\big(C_i,\varepsilon_i\big(\|y-x\|_E\big)\big)$$

for all $x,y\in U_i$. The first assertion of the Proposition follows.

We now prove the second assertion. We fix $i\in I$. Let $x,y\in U_i$. We set $\delta=\|y-x\|_E$,

$$\begin{aligned}&\ell=df_x-df_y-d^2f_x(y-x)\in\mathcal{L}(E,F)\\ \text{and}\quad&b=\tfrac{1}{2}\big(d^2f_x-d^2f_y\big)\in\mathcal{B}^{\mathrm{s}}(E\times E,F).\end{aligned}$$

Observe that, thanks to our assumption on U_i (and ultrametricity), the ball $B_E(y,\delta)$ of centre y and radius δ is included in U_i. For $z\in B_E(y,\delta)$, we define:

$$\begin{aligned}\delta_{z,x}&=f(z)-f(x)-df_x(z-x)-\tfrac{1}{2}d^2f_x(z-x,z-x)\\ \delta_{z,y}&=f(z)-f(y)-df_y(z-y)-\tfrac{1}{2}d^2f_y(z-y,z-y)\\ \delta_{y,x}&=f(y)-f(x)-df_x(y-x)-\tfrac{1}{2}d^2f_x(y-x,y-x).\end{aligned}$$

By our assumptions, these three vectors have norm at most $\delta^2 \cdot \varepsilon_i(\delta)$. A simple computation using linearity and bilinearity yields $\delta_{z,x} - \delta_{z,y} - \delta_{y,x} = \ell(z-y) - b(z-y, z-y)$. Consequently, we get the estimation:

$$\|\ell(h) - b(h,h)\|_F \leqslant \delta^2 \cdot \varepsilon_i(\delta)$$

for all $h \in E$ with $\|h\|_E \leqslant \delta$. Applying it with $(p+1)h$ in place of h, we find:

$$\|\ell(h) - (p+1) \cdot b(h,h)\|_F \leqslant \delta^2 \cdot \varepsilon_i(\delta)$$

and combining now the two above inequalities, we end up with $\|b(h,h)\|_F \leqslant p\delta^2 \cdot \varepsilon_i(\delta)$ and so $\|\ell(h)\|_F \leqslant p\delta^2 \cdot \varepsilon_i(\delta)$ as well. This inequality being true for all $h \in B_E(\alpha)$, we derive $\|\ell\|_{\mathcal{L}(E,F)} \leqslant p^2\delta \cdot \varepsilon_i(\delta)$. This proves the differentiability of df together with the fact that its differential is $d^2 f$. □

Hensel's Lemma for functions of class C^2

We want to develop an analogue of Hensel's Lemma (see Theorem 1.6) for a function $f : U \to F$ of class C^2 whose domain U is an open subset of a finite dimensional p-adic vector space E and whose codomain F is another p-adic vector space of the same dimension. For simplicity, we assume further that $U = B_E(1)$ (the general case is deduced from this one by introducing coverings). Under this additional assumption, the function $d^2 f$ is bounded, *i.e.* there exists a constant C such that $\|d^2 f_x\|_{\mathcal{B}(E\times E,F)} \leqslant C$ for all $x \in B_E(1)$. Plugging this into Eq. (3.25) and possibly enlarging a bit C, we derive the two estimations:

$$\forall x, y \in B_E(1), \quad \|f(y) - f(x) - df_x(y-x)\|_F \leqslant C \cdot \|y-x\|_E^2 \tag{3.26}$$

$$\|df_y - df_x\|_{\mathcal{L}(E,F)} \leqslant C \cdot \|y-x\|_E \tag{3.27}$$

which are the key points for making Newton iteration work. We consider a point $v \in B_E(1)$ for which:

$$df_v \text{ is invertible} \quad \text{and} \quad \|f(v)\|_F \cdot \|df_v^{-1}\|_{\mathcal{L}(F,E)}^2 < C^{-1} \leqslant \|df_v^{-1}\|_{\mathcal{L}(F,E)}. \tag{3.28}$$

We set $r = \big(C \cdot \|df_v^{-1}\|_{\mathcal{L}(F,E)}\big)^{-1}$ and denote by V the open ball of centre v and radius r. Observe that $r \leqslant 1$, so that $V \subset B_E(1)$.

Lemma 3.31. *Under the assumptions (3.26), (3.27) and (3.28), the linear function df_x is invertible and $\|df_x^{-1}\|_{\mathcal{L}(F,E)} = \|df_v^{-1}\|_{\mathcal{L}(F,E)}$ for all $x \in V$. Moreover, the Newton iteration mapping $\mathcal{N} : V \to E$, $x \mapsto x - df_x^{-1}(f(x))$ takes its values in V and satisfies:*

$$\|f(\mathcal{N}(x))\|_F \leqslant C \cdot \|df_v^{-1}\|_{\mathcal{L}(F,E)}^2 \cdot \|f(x)\|_F^2. \tag{3.29}$$

Proof. Let $x \in V$. Set $g = df_v - df_x$. By the inequality (3.27), we have:

$$\|g\|_{\mathcal{L}(E,F)} = \|df_v - df_x\|_{\mathcal{L}(E,F)} \leqslant C \cdot \|v-x\|_E < Cr = \big(\|df_v^{-1}\|_{\mathcal{L}(F,E)}\big)^{-1}.$$

Write $df_x = df_v - g = df_v \circ (\mathrm{id}_E - df_v^{-1} \circ g)$. The above estimation shows that the norm of $df_v^{-1} \circ g$ is strictly less than 1. Therefore the series $\sum_{n=0}^{\infty} (df_v^{-1} \circ g)^{\circ n}$ converges to an inverse of $(\mathrm{id}_E - df_v^{-1} \circ g)$. In particular $(\mathrm{id}_E - df_v^{-1} \circ g)$ is invertible and so is df_x. Remark in addition that the norm of $\sum_{n=0}^{\infty} (df_v^{-1} \circ g)^{\circ n}$ is at most 1 and so $\|df_x^{-1}\|_{\mathcal{L}(F,E)} \leqslant \|df_v^{-1}\|_{\mathcal{L}(F,E)}$. Reverting the roles of v and x, we find even better that equality does hold. Applying now (3.26), we get:

$$\|f(v) - f(x) - df_x(v-x)\|_F \leqslant C \cdot \|v-x\|_E^2 < Cr^2 = \frac{r}{\|df_v^{-1}\|_{\mathcal{L}(F,E)}^2}.$$

Applying df_x^{-1}, we deduce $\|df_x^{-1}(f(v)) - df_x^{-1}(f(x)) - (v-x)\|_F < r$. Notice furthermore that $\|v-x\|_E < r$ by assumption and $\|df_x^{-1}(f(v))\|_F \leqslant \|df_v^{-1}\|_{\mathcal{L}(F,E)} \cdot \|f(v)\|_F < r$. Consequently we find $\|df_x^{-1}(f(x))\|_F < r$ as well, for what we derive $\mathcal{N}(x) \in V$. Eq. (3.29) finally follows by applying again Eq. (3.26) with x and $\mathcal{N}(x)$. □

Corollary 3.32 (Hensel's Lemma for function of class C^2)**.** *Under the assumptions* (3.26)*,* (3.27) *and* (3.28)*, the sequence* $(x_i)_{i\geqslant 0}$ *defined by the recurrence*

$$x_0 = v \quad ; \quad x_{i+1} = \mathcal{N}(x_i) = x_i - df_{x_i}^{-1}(f(x_i))$$

is well defined and converges to $x_\infty \in V$ *such that* $f(x_\infty) = 0$*. The rate of convergence is given by:*

$$\|x_i - x_\infty\|_E \leqslant r \cdot \Big(C \cdot \|f(v)\|_F \cdot \|df_v^{-1}\|^2_{\mathcal{L}(F,E)}\Big)^{2^i} = \frac{\Big(C \cdot \|f(v)\|_F \cdot \|df_v^{-1}\|^2_{\mathcal{L}(F,E)}\Big)^{2^i}}{C \cdot \|df_v^{-1}\|_{\mathcal{L}(F,E)}}.$$

Moreover x_∞ *is the unique solution in* V *to the equation* $f(x) = 0$*.*

Proof. We define $\rho = C \cdot \|f(v)\|_F \cdot \|df_v^{-1}\|^2_{\mathcal{L}(F,E)}$ By Lemma (3.31), all the x_i's lie in V and:

$$\|f(x_{i+1})\|_F \leqslant C \cdot \|df_v^{-1}\|^2_{\mathcal{L}(F,E)} \cdot \|f(x_i)\|^2_F \tag{3.30}$$

for all i. By induction, we derive $\|f(x_i)\|_F \leqslant \big(C \cdot \|df_v^{-1}\|^2_{\mathcal{L}(F,E)}\big)^{-1} \cdot \rho^{2^i}$. Therefore:

$$\|x_{i+1} - x_i\|_E = \|df_{x_i}^{-1}(f(x_i))\|_E \leqslant \|df_v^{-1}\|_{\mathcal{L}(F,E)} \cdot \|f(x_i)\|_F \leqslant r \cdot \rho^{2^i}. \tag{3.31}$$

The sequence $(x_i)_{i\geqslant 0}$ is a Cauchy sequence and therefore converges to some x_∞ for which $f(x_\infty) = 0$. Eq. (3.31) implies moreover the announced rate of convergence together with the fact that x_∞ belongs to V. It then only remains to prove the uniqueness of the solution to the equation $f(x) = 0$ in V. For this, assume that $y \in V$ satisfies $f(y) = 0$. Instantiating Eq. (3.26) with $x = x_\infty$, we obtain $\|df_{x_\infty}(y - x_\infty)\|_F \leqslant C \cdot \|y - x_\infty\|^2_E$. Hence:

$$\|y - x_\infty\|_E = \|df_{x_\infty}^{-1}\big(df_{x_\infty}(y - x_\infty)\big)\|_E \leqslant C \cdot \|df_v^{-1}\|_{\mathcal{L}(F,E)} \cdot \|y - x_\infty\|^2_E = r^{-1} \cdot \|y - x_\infty\|^2_E.$$

Since moreover $\|y - x_\infty\|_E$ has to be strictly less than r by our assumptions, we derive $y = x_\infty$ and uniqueness follows. □

Remark 3.33. Hensel's Lemma is stable in the sense that its conclusion remains correct for a sequence $(x_i)_{i\geqslant 0}$ satisfying the weaker recurrence:

$$x_{i+1} = x_i - df_{x_i}^{-1}(f(x_i)) \ + \ (\text{some small perturbation})$$

as soon as the perturbation is small enough to continue to ensure that the estimation (3.30) holds.

Precision

In practice, the function f of which we want to find a root is often not exact but given with some uncertainty; think typically of the case where f is a polynomial (of given degree) with coefficients in $\mathbb{Q}_p$ given at some finite precision. In order to model this, we introduce $C^2(B_E(1), F)$, the set of functions $f : B_E(1) \to F$ of class C^2. We endow it with the C^2-norm defined by

$$\|f\|_{C^2} = \max\big(\|f\|_\infty, \|df\|_\infty, \|d^2 f\|_\infty\big)$$

where the infinite norm of a function is defined as usual as the supremum of the norms of its values. We consider in addition a finite dimensional subspace $\mathcal{F}$ of $C^2(B_E(1), F)$. The uncertainty on f will be modeled by some lattice in $\mathcal{F}$.

We fix $v \in B_E(1)$ together with two positive real numbers C and r. We assume $r \leqslant 1$. Let V be the open ball in E of centre v and radius r; clearly $V \subset B_E(1)$. Let $\mathcal{U}$ be the open subset of $\mathcal{F}$ consisting of function f for which Eqs. (3.26), (3.27) and (3.28) hold and $\|df_v^{-1}\|_{\mathcal{L}(F,E)} = \frac{r}{C}$. By Hensel's Lemma (Corollary 3.32), any $f \in \mathcal{U}$ has a unique zero in V. The function:

$$\begin{array}{rccl} \mathcal{Z} : & \mathcal{U} & \longrightarrow & V \\ & f & \mapsto & x \quad \text{s.t.} \quad x \in V \text{ and } f(x) = 0 \end{array}$$

is then well defined. In the idea of applying the precision Lemma, we study its differentiability.

Lemma 3.34. *The function $\mathcal{Z}$ is differentiable on $\mathcal{U}$. Moreover, its differential at $f \in \mathcal{U}$ is the linear mapping $d\mathcal{Z}_f : \varphi \mapsto -df^{-1}_{\mathcal{Z}(f)}(\varphi(\mathcal{Z}(f)))$.*

Proof. The lemma can be seen as an application of the p-adic implicit functions Theorem (see [70, Proposition 4.3]). Below, we give a direct proof avoiding it. Let $f, g \in \mathcal{U}$. We set $x = \mathcal{Z}(f)$, $\varphi = g - f$ and $y = x - df_x^{-1}(\varphi(x))$. Since $f(x)$ vanishes, Eq. (3.26) reads:

$$\|g(y) - (\mathrm{id}_E - dg_x \circ df_x^{-1})(\varphi(x))\|_F \leqslant C \cdot \|y - x\|_E^2 \leqslant C \cdot \|df_x^{-1}\|^2_{\mathcal{L}(F,E)} \cdot \|\varphi\|^2_\infty. \tag{3.32}$$

From the identity $\mathrm{id}_E - dg_x \circ df_x^{-1} = (df_x - dg_x) \circ df_x^{-1}$, we deduce moreover that

$$\|\mathrm{id}_E - dg_x \circ df_x^{-1}\|_{\mathcal{L}(E,E)} \leqslant \|df_x^{-1}\|_{\mathcal{L}(F,E)} \cdot \|df_x - dg_x\|_{\mathcal{L}(E,F)} \leqslant \|df_x^{-1}\|_{\mathcal{L}(F,E)} \cdot \|d\varphi\|_\infty. \tag{3.33}$$

Combining (3.32) and (3.33), we derive $\|g(y)\|_F \leqslant \|df_x^{-1}\|_{\mathcal{L}(F,E)} \cdot \|\varphi\|^2_{C^2}$. Corollary 3.32 then implies $\|y - \mathcal{Z}(g)\|_E \leqslant \|df_x^{-1}\|^2_{\mathcal{L}(F,E)} \cdot \|\varphi\|^2_{C^2} = \|df_v^{-1}\|^2_{\mathcal{L}(F,E)} \cdot \|\varphi\|^2_{C^2}$, which proves the lemma. □

Applying the precision Lemma, we end up with the next corollary.

Corollary 3.35. *If the uncertainty on f is given by a (sufficiently rounded and small) lattice $\mathcal{H} \subset \mathcal{U}$, then the optimal precision on $x = \mathcal{Z}(f)$ is $df_x^{-1} \circ \mathrm{ev}_x(\mathcal{H})$ where $\mathrm{ev}_x : \mathcal{H} \to F$ is the evaluation morphism at x.*

It is quite instructive to compare this result with the optimal precision we get after a single iteration of the Newton scheme. In order to do so, we introduce the map:

$$\begin{array}{rccc} \mathcal{N} : & \mathcal{U} \times V & \longrightarrow & V \\ & (f, x) & \mapsto & f(x) - df_x^{-1}(f(x)) \end{array}$$

which is well defined thanks to Lemma 3.31. It is moreover clear that $\mathcal{N}$ is differentiable since it appears as a composite of differentiable functions. A straightforward (but tedious) calculation shows that its differential at (f, x) is given by:

$$d\mathcal{N}_{(f,x)} : (\varphi, \xi) \mapsto -df_x^{-1}\big(\varphi(x)\big) + df_x^{-1}\big(d^2 f_x(\xi, f(x))\big) + df_x^{-1}\big(d\varphi_x(f(x))\big).$$

In particular, we observe that if $f(x) = 0$, the last two terms vanish as well, so that the linear mapping $d\mathcal{N}_{(f,x)}$ does not depend on the second variable ξ and agrees with $d\mathcal{Z}_f$. In the language of precision, this means that the optimal precision on $\mathcal{Z}(f)$ is governed by the last iteration of the Newton scheme.

In practice, this result suggests the following strategy: *in order to compute $\mathcal{Z}(f)$, we start with some v satisfying the requirements of Hensel's Lemma (Corollary 3.32), we iterate the Newton scheme* without taking care (so much) of precision *until the obtained approximation looks correct and we finally perform a last iteration reintroducing the machinery for tracking precision.* By Remark 3.33, the omission of precision tracking is harmless while the equality $d\mathcal{Z}_f = d\mathcal{N}_{(f,\mathcal{Z}(f))}$ shows (under mild assumptions on the lattice $\mathcal{H}$) that the precision we will get at the end (using the above strategy) will be very good because the loss of precision will just come from the last iteration and so will be limited.

This behavior is very well illustrated by the example of the computation of square roots detailed in §2.1.3. (We then encourage the reader to read it and study it again in light of the discussion of this paragraph.) More recently Lairez and Vaccon applied this strategy for the computation of the solutions of p-adic differential equations with separation of variables [53]; they managed to improve this way a former algorithm by Lercier and Sirvent for computing isogenies between elliptic curves in positive characteristic [57].

3.3 Lattice-based methods for tracking precision

Until now, we have explained how the precision Lemma can be used for finding the optimal precision and analyzing this way the stability of algorithms. It turns out that the precision Lemma is also useful for *stabilizing* algorithms, *i.e.* for producing a stable procedure by altering slightly another given procedure which might be highly unstable.

3.3.1 The method of adaptive precision

We place ourselves in the following general setting. Let `Phi` be a routine that computes a mathematical function $\varphi : U \to V$ whose domain U is an open subset of a finite dimensional $\mathbb{Q}_p$-vector space E (*i.e.* `Phi` takes as input a tuple of p-adic numbers but may possibly fails for particular instances lying a closed subset) and whose codomain V is an open subset of another finite dimensional $\mathbb{Q}_p$-vector space (*i.e.* `Phi` outputs a tuple of p-adic numbers as well). We assume that f is of class C^1 on U. We moreover fix an input $x \in U$ for which the differential df_x is surjective.

Remark 3.36. While the C^1-assumption is usually harmless, the surjectivity assumption is often more serious. For instance, observe that it implies that the dimension of F (*i.e.* the size of the output) is not greater than the dimension of E (*i.e.* the size of the input). Clearly there exist many interesting algorithms that do not satisfy this requirement and therefore do not fit into our framework. One can always however workaround this issue as follows. We write the space of outputs F as a direct sum $F = F_1 \oplus \cdots \oplus F_m$ and decompose the procedure `Phi` accordingly, *i.e.* for each i, we introduce the algorithm $\texttt{Phi}_i$ that outputs only the i-th part of `Phi`. Clearly $\texttt{Phi}_i$ is modeled by the mathematical function $\varphi_i = \mathrm{pr}_i \circ \varphi$ where $\mathrm{pr}_i : F \to F_i$ is the canonical projection. As a consequence $d\varphi_{i,x} = \mathrm{pr}_i \circ d\varphi_x$; if the F_i's are small enough, it will then be unlikely that one of them is not surjective. For instance, in the special case where the F_i's are all lines, the writing $F = F_1 \oplus \cdots \oplus F_m$ corresponds to the choice of a basis of F and the surjectivity of the $d\varphi_{i,x}$'s is equivalent to the fact that each column of the Jacobian matrix has a nonzero entry (which is clearly much weaker than requiring its surjectivity).

Our aim is to compute $\varphi(x)$ using the procedure `Phi`; we would like moreover to be sharp in terms of precision and get proved results. We have seen that neither zealous arithmetic, nor floating-point arithmetic can ensure these two requirements at the same time: zealous arithmetic is usually not sharp (see §2.4.3, §3.2.2 for many examples) while floating-point arithmetic is not proved. The case of lazy/relaxed arithmetic is a bit aside but it is not quite satisfactory either. Indeed recall that in the lazy approach, we are fixing a target precision; the output of `Phi` will then be sharp by design. The issue is elsewhere and comes from the fact that the lazy/relaxed machinery will generally compute the input x at a higher precision than needed, consuming then more resources than necessary (see §3.2.1 for a theoretical discussion about this and/or §3.3.2 below for a concrete example).

Context of zealous arithmetic: precision on inputs

We assume that the input x is given at some precision which is represented by a lattice H in E: if the i-th coordinate of x is given at precision $O(p^{N_i})$, the lattice H is the diagonal lattice

$$H = p^{N_1}\mathbb{Z}_p \oplus p^{N_2}\mathbb{Z}_p \oplus \cdots \oplus p^{N_d}\mathbb{Z}_p \quad (\text{with } d = \dim E)$$

(see also §3.2.1). We underline nevertheless that non diagonal lattices H (corresponding to precision data with diffused digits, see Definition 3.20) are also permitted. We assume that H is nice enough so that the precision Lemma applies, giving:

$$\varphi(x + H) = \varphi(x) + d\varphi_x(H). \tag{3.34}$$

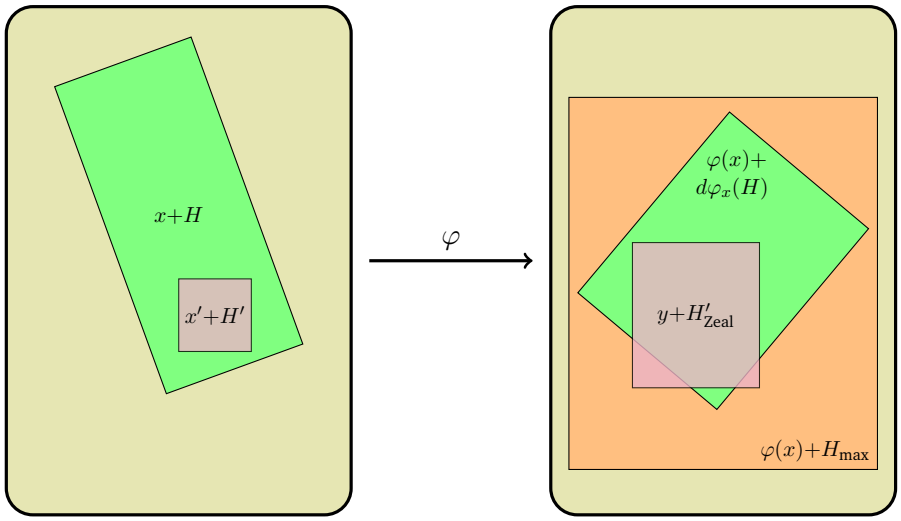

Figure 3.6: The method of adaptive precision: first attempt

Separation of approximation and precision. The formula above strongly suggests to split the computation of $\varphi(x+H)$ into two independent parts corresponding to the two summands. Concerning precision, we have to compute the lattice $d\varphi_x(H)$. One option for this is to rely on automatic differentiation techniques [3, 64]; this approach however often leads to costly computations and really affects the complexity of our algorithm. For this reason, alternative strategies are often preferable (when available). The simplest one is probably to precompute $d\varphi_x$ by hand (we have seen in §3.2.2 that it is tractable in many situations) and plug the obtained result into our algorithm. On the other hand, we observe that $d\varphi_x(H)$ is a lattice in F and is therefore encoded by a $\dim F \times \dim F$ matrix, which might be quite a large object (compare with $\dim F$ which is the size of the output). Therefore, if we do not need to be *extremely* careful on precision, we could prefer replacing $d\varphi_x(H)$ by a slightly larger lattice $H_{\max}$ which is more easily representable, *e.g.* $H_{\max}$ could be a diagonal lattice.

For now on, we assume that we have at our disposal a lattice $H_{\max}$ containing $d\varphi_x(H)$ and we focus on the computation of the first term of Eq. (3.34), that is the approximation. Since $d\varphi_x(H)$ is stable under addition, Eq. (3.34) yields $\varphi(x+H) = y + d\varphi_x(H) \subset y + H_{\max}$ for any y lying in $\varphi(x+H)$. Using now that $H_{\max}$ is stable under addition as well, we end up with $\varphi(x+H) \subset y + H_{\max}$ as soon as $y \in \varphi(x+H) + H_{\max}$. To conclude, it is then enough to compute the value of *any* element of $\varphi(x+H)$ at precision $H_{\max}$. For doing so, we can rely on zealous arithmetic: we increase sufficiently the precision on x (picking arbitrary values for digits) so that $\varphi(x)$ can be computed at precision at least $H_{\max}$ and we output $\varphi(x)$ at precision $H_{\max}$. In more conceptual terms, what we do is to choose an element $x' \in x+H$ together with a diagonal lattice $H' \subset H$ having the property that zealous arithmetic computes y and H'_{Zeal} such that $f(x'+H') \subset y + H'_{\text{Zeal}}$ and $H'_{\text{Zeal}} \subset H_{\max}$. We thus have:

$$y \in \varphi(x'+H') + H'_{\text{Zeal}} \subset \varphi(x+H) + H_{\max}$$

as wanted (see also Figure 3.6). These ideas lead to the following implementation:

```
def Phi_stabilized_v1(x):
    lift x at precision H'
    y = Phi(x)      # here y is computed at precision H'_Zeal
    return y at precision H_max
```

The method presented above works but has a serious drawback: in practice, if often happens that the precision encoded by H' is very high, leading then to very costly computations. In the next paragraph, we explain a strategy for dealing with this issue.

Viewing `Phi` *as a sequence of steps.* Roughly speaking, the problem with the above approach comes from the fact that the precision is tracked using interval arithmetic all along the execution of `Phi`. In order to avoid this, the idea consists in decomposing `Phi` into several steps; we then let interval arithmetic do the job of tracking precision within each individual step but we readjust the precision (using informations coming from the precision Lemma) each time we switch from one step to the next one. Concretely the readjustment is made by applying the method of the previous paragraph to each individual step.

Let us now go further into the details. As said above, we view the routine `Phi` as a finite sequence of steps $\texttt{Step}_1, \dots, \texttt{Step}_n$. Let $\sigma_i : U_{i-1} \to U_i$ be the mathematical function modeling the i-th step; its codomain U_i is the space of outputs of the i-th step, that is the space of "active" variables after the execution of i-th step. Mathematically, it is an open subset in a finite dimensional $\mathbb{Q}_p$-vector space E_i. For $i \in \{0, \dots, n\}$, we set $\varphi_i = \sigma_i \circ \cdots \circ \sigma_1$ and $x_i = \varphi_i(x)$; they are respectively the function modeling the execution of the first i steps of `Phi` and the state of the memory after the execution of the i-th step of `Phi` on the input x. In what follows, we always assume that $d\varphi_{i,x}$ is surjective[24] and, even more, that the precision Lemma applies with φ_i, x and H, meaning that, for all i:

$$\varphi_i(x + H) = x_i + d\varphi_{i,x}(H). \tag{3.35}$$

We define $H_i = d\varphi_{i,x}(H)$ and we assume that we are given a "simple" lattice $H_{i,\text{max}}$ containing H_i. However, for the application we have in mind, this datum will not be sufficient; indeed, we shall need to be *extremely* careful with precision at intermediate levels. In order to do so *without having to manipulate the "complicated" lattice* H_i, we assume that we are given in addition another "simple" lattice $H_{i,\text{min}}$ which is *contained* in H_i. With these notations, the stabilized version of `Phi` takes the following schematic form:

```
def Phi_stabilized_v2(x):
    y_0 = x
    for i in 1,2,...,n:
        lift y_{i-1} at enough precision
        y_i = Step_i(y_{i-1})    # here we want y_i to be computed at precision at least H_{i,min}
    return y_n at precision H_{n,max}
```

The locution "enough precision" in the above code means that we want to ensure that zealous arithmetics is able to compute $\texttt{y}_i$ at precision $H_{i,\text{min}}$. Let H'_{i-1} be a lattice encoding such an acceptable precision. Rigorously, it can be defined as follows. We set $y_0 = x$ and for $i \in \{1, \dots, n\}$, we define inductively H'_{i-1}, y_i and $H'_{i,\text{Zeal}}$ by requiring that the routine $\texttt{Step}_i$ called on the input y_{i-1} given at precision H'_{i-1} computes y_i at precision $H'_{i,\text{Zeal}}$ with $H'_{i,\text{Zeal}} \subset H_{i,\text{min}}$ (see Figure 3.7). Of course the value y_i we have just defined corresponds exactly to the variable $\texttt{y}_i$ of the procedure `Phi_stabilized`. The following lemma is the key for proving the correctness of the algorithm `Phi_stabilized_v2`.

Lemma 3.37. *For all $i \in \{1, \dots, n\}$, we have $y_i \in x_i + H_i$.*

Proof. We argue by induction on i. The initialization is obvious given that $y_0 = x = x_0$. We now assume that $y_{i-1} \in x_{i-1} + H_{i-1}$. Applying σ_i, we derive

$$\sigma_i(y_{i-1}) \in \sigma_i(x_{i-1} + H_{i-1}) = \sigma_i \circ \varphi_{i-1}(x + H) = \varphi_i(x + H) = x_i + H_i \tag{3.36}$$

thanks to Eq. (3.35) applied twice. Moreover, by construction, we know that $\sigma_i(y_{i-1}) \in y_i + H'_{i,\text{Zeal}} \subset y_i + H_{i,\text{min}} \subset y_i + H_i$. As a consequence the difference $\sigma_i(y_{i-1}) - y_i$ lies in the lattice H_i; this can be rewritten again as $y_i \in \sigma_i(y_{i-1}) + H_i$. Combining with Eq. (3.36), we get $y_i \in x_i + H_i$ as desired. □

[24] This assumption is rather restrictive but really simplifies the discussion.

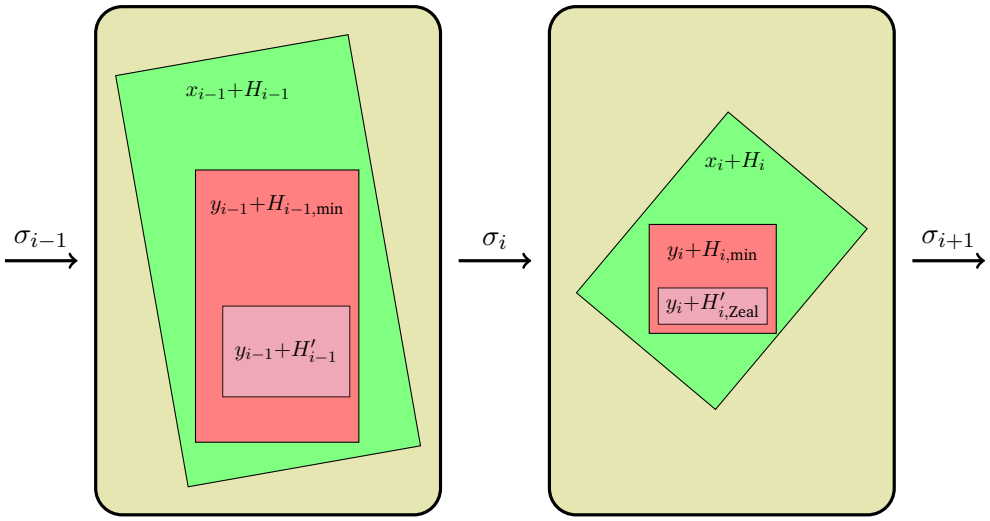

Figure 3.7: The method of adaptive precision: second attempt

We derive from Lemma 3.37 that $x_n + H_n = y_n + H_n$. Combining this equality with Eq. (3.35), we obtain $\varphi(x + H) = x_n + H_n = y_n + H_n \subset y_n + H_{\max,n}$. This inclusion exactly means that `Phi_stabilized` called on the input x given at precision H outputs an approximation of $\varphi(x)$ at precision $H_{\max,n}$; in other words, the algorithm `Phi_stabilized` is correct.

The main advantage of the algorithm `Phi_stabilized_v2` (compared to `Phi_stabilized_v1`) is that the precision encoded by the lattices H'_{i-1} are usually acceptable (compared to the precision encoded by H'). Actual computations are then carried out at a reasonable precision and then consume moderated resources.

Remark 3.38. Interestingly, observe that the spirit of algorithm `Phi_stabilized_v2` is close to the behavior of p-adic floating-point arithmetic; precisely, in p-adic floating-point arithmetic, we are decomposing `Phi` as a sequence of elementary steps (each of them consists of a single arithmetic operation) and the lattices $H_{i,\min}$ are chosen in such a way that the number of significand digits of each p-adic variable is kept constant. For this reason, the method of adaptive precision can be also used to derive *proofs* on the results obtained *via* p-adic floating-point arithmetic.

Context of lazy/relaxed arithmetic: target precision

We recall that the question addressed in the context of lazy/relaxed arithmetic is formulated as follows: we fix a target precision on the output and we want to avoid as much as possible the computation with unnecessary digits of the input, that are the digits that do not influence the output at the requested precision. For simplicity, it is convenient to assume that the output of `Phi` consists of a single p-adic number[25], that is $F = \mathbb{Q}_p$. The target precision is then given by a single integer N.

We recall from §3.2.1 that, if H' is a lattice in E, then the condition:

$$\varphi(x + H') \subset \varphi(x) + p^N \mathbb{Z}_p \tag{3.37}$$

ensures that the knowledge of x at precision H' is enough to compute $\varphi(x)$ at precision $O(p^N)$. Moreover, assuming that H' is nice enough so that the precision Lemma applies, the inclusion (3.37) is equivalent to $H' \subset d\varphi_x^{-1}(p^N \mathbb{Z}_p)$. We draw the attention of the reader to the fact that $d\varphi_x^{-1}(p^N \mathbb{Z}_p)$ itself is usually *not* a lattice because $d\varphi_x^{-1}$ often has a nontrivial kernel. Anyway, the above analysis gives a theoretical answer to the question we are interested in: the unnecessary digits on the inputs are those lying in some nice enough lattice contained in $d\varphi_x^{-1}(p^N \mathbb{Z}_p)$.

[25] This assumption is harmless since one can always reduce to this case by projection on each coordinate.

Mimicking what we have done before in the zealous context, we can use the above result to write down a first stabilized version of `Phi`. Set $H = d\varphi_x^{-1}(\mathbb{Z}_p)$ and choose a lattice $H_{\min}$ contained in H. Clearly $p^N H_{\min}$ is a lattice contained in $d\varphi_x^{-1}(p^N\mathbb{Z}_p)$. Moreover, examining closely the statement of the precision Lemma, we find that there exists an integer N_0 such that:

$$\varphi(x + p^N H) = \varphi(x) + p^N d\varphi_x(H) \subset \varphi(x) + p^N\mathbb{Z}_p \tag{3.38}$$

for all $N \geqslant N_0$. The first stabilized version of `Phi` then can be described as follows.

```
def Phi_stabilized_v1(x):
    def Phi_of_x(N):
        xapp = x(p^max(N,N0) H)      # Evaluation of x at precision p^min(N,N0) H
        return Phi(xapp) % p^N
    return Phi_of_x
```

Remark 3.39. In the code above, the variable `x` is a tuple of lazy p-adic numbers (as described in §2.2), not a tuple of relaxed p-adic numbers. Making the method of adaptive precision work within the framework of relaxed arithmetic is possible but tedious; for simplicity, we shall omit these technicalities and restrict ourselves to lazy p-adic numbers in this course.

The attentive reader has certainly noticed that the value N_0 has appeared explicitly in the function `Phi_stabilized_v1` above. This implies that we need to precompute it. This has to be done by hand *a priori* by explicating the constants in the precision Lemma in the special case we are considering; we do not hide that this may require some effort (through general techniques for that are available [18, 19]).

Eq. (3.38) shows that the nested function `Phi_of_x` returns the correct answer. The main benefit of `Phi_stabilized_v1` is that it asks for the computation of the input at sharp precision. Nevertheless, it carries all the computations with exact rational numbers and so still consumes a lot of resources.

In order to get rid of this disadvantage, we can follow the strategy already used with success in the zealous context: we decompose `Phi` into n steps $\texttt{Step}_1, \ldots, \texttt{Step}_n$. Again we call $\sigma_i : U_{i-1} \to U_i$ the mathematical function modeling $\texttt{Step}_i$. We set $\varphi_i = \sigma_i \circ \cdots \circ \sigma_1$, $x_i = \varphi_i(x)$ and introduce moreover the function $\psi_i = \sigma_n \circ \cdots \sigma_{i+1}$. Clearly $\varphi = \psi_i \circ \varphi_i$ for all i. Interestingly, observe that the relation $d\varphi_x = d\psi_{i,x_i} \circ d\varphi_{i,x}$ implies that all the differentials $d\psi_{i,x_i}$'s are surjective without any further assumption. For each $i \in \{0, \ldots, n\}$, we define $H_i = d\psi_{i,x_i}^{-1}(\mathbb{Z}_p)$ (which is generally *not* a lattice) and choose a lattice $H_{i,\min}$ contained in H_i. The optimized form of the stabilized version of `Phi` then writes as follows.

```
def Phi_stabilized_v2(x):
    def Phi_of_x(N):
        y_0 = x(p^N H_{0,min})      # Evaluation of x at precision p^N H_{0,min}
        for i in 1,2,...,n:
            y_i = Step_i(y_{i-1}) % p^N H_{i,min}
        return y_n % p^N
    return Phi_of_x
```

Roughly speaking, the above optimization allows us to "do the modulo" after each step, avoiding then the explosion of the size of the intermediate values.

Proposition 3.40. *There exists an integer N_0 (which can be made explicit with some effort) such that the nested function* `Phi_of_x` *outputs a correct answer on the inputs $N \geqslant N_0$.*

Remark 3.41. We underline that Proposition 3.40 does not assume the surjectivity of the differentials $d\varphi_{i,x}$; recall that this assumption was unavoidable in the case of zealous arithmetic and appeared as a serious limitation of the method. We have to mention however that the analysis of the algorithm `Phi_stabilized_v2` is easier under this additional hypothesis (see §3.3.2 below for a concrete example).

Sketch of the proof of Proposition 3.40. Given an integer N, we let $(y_{N,i})_{0\leqslant i\leqslant n}$ be a sequence such that $y_{N,0} = x$ and $y_{N,i} = \sigma_i(y_{N,i-1}) + p^N H_{i,\min}$. The value $y_{N,i}$ corresponds to the value of the variable y_i computed by the function `Phi_of_x` called on the input N. Moreover, for all fixed i, the sequence $(y_{N,i})_{N\geqslant 0}$ converges to x_i when N goes to infinity. Since ψ_i is a function of class C^1, this implies that $d\psi_{i,y_{N,i}}$ converges to $d\psi_{i,x_i}$ when N goes to infinity. From this, we derive that $d\psi_{i,y_{N,i}}(H_{i,\min}) = d\psi_{i,x_i}(H_{i,\min})$ for N large enough. On the other hand, analyzing closely the proof of the precision Lemma, we find that there exists an integer N_0 such that:

$$\begin{aligned}\psi_i\big(y_{N,i} + p^N H_{i,\min}\big) &= \psi_i(y_{N,i}) + d\psi_{i,y_{N,i}}\big(p^N H_{i,\min}\big)\\ &= \psi_i(y_{N,i}) + p^N d\psi_{i,x_i}\big(H_{i,\min}\big)\end{aligned} \tag{3.39}$$

for all i and all $N \geqslant N_0$. We claim that `Phi_of_x` outputs a correct answer (*i.e.* that $y_{N,n} \equiv \varphi(x) \pmod{p^N}$) whenever $N \geqslant N_0$. More precisely, we are going to prove by induction on i that $\psi_i(y_{i,N}) \equiv \varphi(x) \pmod{p^N}$ for $i \in \{0, 1, \ldots, n\}$ and $N \geqslant N_0$. This is obvious when $i = 0$ given that $\psi_0 = \varphi$ and $y_{0,N} = x$. Let us now assume $\psi_{i-1}(y_{i-1,N}) \equiv \varphi(x) \pmod{p^N}$. Noting than $\psi_i \circ \sigma_i = \psi_{i-1}$, we derive from $\sigma_i(y_{N,i-1}) \in y_{N,i} + p^N H_{i,\min}$ that:

$$\begin{aligned}\psi_{i-1}(y_{N,i-1}) \in \psi_i\big(y_{N,i} + p^N H_{i,\min}\big) &= \psi_i(y_{N,i}) + p^N d\psi_{i,x_i}\big(H_{i,\min}\big)\\ &\subset \psi_i(y_{N,i}) + p^N \mathbb{Z}_p\end{aligned}$$

using Eq. (3.39). Consequently $\psi_i(y_{N,i}) \equiv \psi_{i-1}(y_{N,i-1}) \pmod{p^N}$ and we are done thanks to the induction hypothesis. □

3.3.2 Example: The Somos 4 sequence

The method of adaptive precision (presented in §3.3.1) has been used with success in several concrete contexts: the computation of the solutions of some p-adic differential equations [53] and the computation of GCD's and Bézout coefficients *via* the usual extended Euclidean algorithm [17]. Below, we detail another example, initially pointed out by Buhler and Kedlaya [15], which is simpler (it actually looks like a toy example) but already shows up all the interesting phenomena: it is the evaluation of the p-adic Somos 4 sequences.

The Somos 4 sequence: definition, properties and calculations

The Somos 4 sequences were first introduced in [76] by Somos. It is a recursive sequence defined by the recurrence:

$$u_1 = a \quad ; \quad u_2 = b \quad ; \quad u_3 = c \quad ; \quad u_4 = d$$

$$u_{n+4} = \frac{u_{n+1}u_{n+3} + u_{n+2}^2}{u_n} \quad \text{for} \quad n \geqslant 1$$

where a, b, c and d are given initial values. The first values of the generic Somos 4 sequence (that is the Somos 4 sequence where the initial values are left as indeterminates) are shown on Figure 3.8. We observe a quite surprising property: the denominators of the first u_i's are all monomials. We insist on the fact that this behavior is *a priori* not expected at all. For instance the definition of u_9 involves a division by u_5 but a miraculous simplification occurs. This phenomenon is in fact general. Fomin and Zelevinsky proved in [29] (as an application of their theory of cluster algebras) that the Somos 4 sequence exhibits the *Laurent phenomenon*: for all n, the term u_n of the generic Somos 4 sequence is a Laurent polynomial in a, b, c, d, that is a polynomial in a, b, c, d and their inverses.

We address the question of the computation of the n-th term (for n large) of a Somos 4 sequence whose initial values a, b, c and d are invertible elements of $\mathbb{Z}_p$. At least two options are available: (1) we unroll the recurrence starting from the given initial values and (2) we precompute the n-th term of the generic Somos 4 sequence and, in a second time, we plug the values of a, b, c and d. Concretely, they correspond to the following codes respectively:

$$u_5 = \frac{1}{a}\,(c^2 + bd)$$

$$u_6 = \frac{1}{ab}\,(c^3 + bcd + ad^2)$$

$$u_7 = \frac{1}{a^2bc}\,(bc^4 + 2b^2c^2d + ac^3d + b^3d^2 + abcd^2 + a^2d^3)$$

$$u_8 = \frac{1}{a^3b^2cd}\,(b^2c^6 + ac^7 + 3b^3c^4d + 3abc^5d + 3b^4c^2d^2 + 3ab^2c^3d^2 + 2a^2c^4d^2 + b^5d^3 + ab^3cd^3 + 3a^2bc^2d^3 + a^2b^2d^4 + a^3cd^4)$$

$$u_9 = \frac{1}{a^3b^2c^2d}\,(b^2c^8 + ac^9 + 4b^3c^6d + 3abc^7d + 6b^4c^4d^2 + 6ab^2c^5d^2 + 3a^2c^6d^2 + 4b^5c^2d^3 + 7ab^3c^3d^3 + 6a^2bc^4d^3 + b^6d^4 + 3ab^4cd^4 + 5a^2b^2c^2d^4 + 3a^3c^3d^4 + 2a^2b^3d^5 + 3a^3bcd^5 + a^4d^6)$$

$$u_{10} = \frac{1}{a^5b^3c^3d^2}\,(b^4c^{10} + 2ab^2c^{11} + a^2c^{12} + 5b^5c^8d + 11ab^3c^9d + 6a^2bc^{10}d + 10b^6c^6d^2 + 24ab^4c^7d^2 + 17a^2b^2c^8d^2 + 4a^3c^9d^2 + 10b^7c^4d^3 + 26ab^5c^5d^3 + 28a^2b^3c^6d^3 + 15a^3bc^7d^3 + 5b^8c^2d^4 + 14ab^6c^3d^4 + 27a^2b^4c^4d^4 + 24a^3b^2c^5d^4 + 6a^4c^6d^4 + b^9d^5 + 3ab^7cd^5 + 14a^2b^5c^2d^5 + 19a^3b^3c^3d^5 + 12a^4bc^4d^5 + 3a^2b^6d^6 + 6a^3b^4cd^6 + 9a^4b^2c^2d^6 + 4a^5c^3d^6 + 3a^4b^3d^7 + 3a^5bcd^7 + a^6d^8)$$

Figure 3.8: The first terms of the generic SOMOS 4 sequence

	Laurent method	Zealous arith.	Float. arith.	Relaxed arith.
u_1	...0000000001	...0000000001	...0000000001	—
u_2	...0000000001	...0000000001	...0000000001	—
u_3	...0000000001	...0000000001	...0000000001	—
u_4	...0000000001	...0000000001	...0000000001	—
u_5	...0000000010	...0000000010	...0000000010	precision on u_1: 10
u_6	...0000000011	...0000000011	...0000000011	precision on u_1: 10
u_7	...0000000111	...0000000111	...0000000111	precision on u_1: 10
u_8	...0000010111	...0000010111	...0000010111	precision on u_1: 10
u_9	...0000111011	...000111011	...0000111011	precision on u_1: 11
u_{10}	...0100111010	...100111010	...10100111010	precision on u_1: 11
u_{50}	...0000011010	...0	...11000011010	precision on u_1: 19
u_{54}	...0100101001	BOOM	...1100101001	precision on u_1: 20
u_{500}	...1111110010	—	...01111110010	precision on u_1: 109

Figure 3.9: The Somos 4 sequence with initial values $1, 1, 1, 1 \in \mathbb{Z}_2$

```
def somos_option1(a,b,c,d,n):        # We assume n ⩾ 5
    x,y,z,t = a,b,c,d
    for i in 1, 2, ..., n−4:
        x,y,z,t = y, z, t, (y*t + z*z)/x
    return t

def somos_option2(a,b,c,d,n):        # We assume n ⩾ 5
    X,Y,Z,T = A,B,C,D                # A, B, C, D are formal variables
    for i in 1, 2, ..., n−4:
        X,Y,Z,T = Y, Z, T, (Y*T + Z*Z)/X
    return T(A=a, B=b, C=c, D=d)
```

The second option has of course a terrible complexity because the size of the terms of the generic Somos 4 sequence grows very rapidly (see Figure 3.8). However, if we are mostly interested in numerical stability, this option is rather attractive; indeed, we deduce from the Laurent phenomenon that if the initial values a, b, c and d (which are supposed to be invertible in $\mathbb{Z}_p$) are given at precision $O(p^N)$ then all the next terms will be computed at precision $O(p^N)$ as well. On the contrary, the first option may lead to quite important losses of precision.

The table of Figure 3.9 shows the results obtained for the particular Somos 4 sequence initialized with $a = b = c = d = 1 \in \mathbb{Z}_2$ using various methods. The first column corresponds to the second option discussed above; as expected, no losses of precision have to be reported. On the second column (resp. the third column), we have displayed the results obtained by unrolling the recurrence using the algorithm `somos_option1` within the framework of zealous arithmetic (resp. p-adic floating-point arithmetic). We observe that the precision decreases rapidly in the second

	Laurent method	Zealous arith.	Float. arith.	Relaxed arith.
u_1	...0000000001	...0000000001	...0000000001	—
u_2	...0000000001	...0000000001	...0000000001	—
u_3	...0000000001	...0000000001	...0000000001	—
u_4	...0000000011	...0000000011	...0000000011	—
u_5	...0000000100	...0000000100	...000000000100	precision on u_1: 10
u_6	...0000001101	...0000001101	...0000001101	precision on u_1: 10
u_7	...0000110111	...0000110111	...0000110111	precision on u_1: 10
u_8	...0111010111	...0111010111	...0111010111	precision on u_1: 10
u_9	...0111101111	...11101111	...1111101111	precision on u_1: 12
u_{10}	...0000010010	...00010010	...11000010010	precision on u_1: 12
u_{11}	...1000111001	...00111001	...0000111001	precision on u_1: 12
u_{12}	...0011111101	...11111101	...0011111101	precision on u_1: 12
u_{13}	...0000110101	...00110101	...0000110101	precision on u_1: 12
u_{14}	...1101010011	...1010011	...1101010011	precision on u_1: 13
u_{15}	...0000000000	...0000000	0	precision on u_1: 13
u_{16}	...0101011101	...1011101	...0101011101	precision on u_1: 13
u_{17}	...1001101011	...1101011	...1001101011	precision on u_1: 13
u_{18}	...0011110011	...1110011	...0011110011	precision on u_1: 13
u_{19}	...0000000111	BOOM	BOOM	precision on u_1: 23

Figure 3.10: The Somos 4 sequence with initial values $1, 1, 1, 3 \in \mathbb{Z}_2$

column. For example, one digit of precision is lost when computing u_9 because of the division by u_5 which has valuation 1. These losses of precision continue then regularly and, at some point, we find a value which becomes indistinguishable from 0. Dividing by it then produces an error and the computations cannot continue. The third column exhibits much better results though certain digits are still wrong (compare with the first column). Finally, the last column concerned lazy/relaxed arithmetic; it shows the number of digits of u_1 that have been asked for in order to compute the term u_n at precision $O(2^{10})$. We observe that this number increases regularly in parallel with the losses of precision observed in the zealous context. For instance, the relaxed technology needs one more digit of precision on u_1 to compute u_9 while the zealous approach loses one digit of precision on u_9. There phenomena have of course a common explanation: they are both due to the division by u_5 which has valuation 1.

The Somos 4 sequence starting with $(a, b, c, d) = (1, 1, 1, 3)$ is even more unstable (see Figure 3.10); indeed its 15-th term is divisible by 2^{10} and so vanishes at the precision $O(2^{10})$ we are considering. Zealous arithmetic and floating-point arithmetic are both puzzled after this underflow and crash on the computation of u_{19} (which requires a division by u_{15}).

Stabilization

We now apply the theory of adaptive precision, trying to stabilize this way the algorithm `somos_version1`. The decomposition of the routine `somos_version1` into steps in straightforward: each step corresponds to an interaction of the for loop. They are all modeled by the same mathematical function:

$$\begin{array}{rccc} \sigma : & \mathbb{Q}_p^4 & \longrightarrow & \mathbb{Q}_p^4 \\ & (x, y, z, t) & \mapsto & \left(y, z, t, \frac{yt+z^2}{x}\right). \end{array}$$

Remark 3.42. Of course σ is not defined at the points of the form $(0, y, z, t)$. In the sequel, it will be nevertheless more convenient to regard σ as a partially defined function over $\mathbb{Q}_p^4$ (than as a well-defined function over $\mathbb{Q}_p^\star \times \mathbb{Q}_p^3$).

Differential computations. For $i \geqslant 0$, we set $\varphi_i = \sigma \circ \cdots \circ \sigma$ (i times). Clearly σ and the φ_i's are all functions of class C^1 on their domain of definition. The Jacobian matrix of σ at the point (x, y, z, t) is easily seen to be:

$$J(\sigma)_{(x,y,z,t)} = \begin{pmatrix} 0 & 0 & 0 & -\frac{yt+z^2}{x^2} \\ 1 & 0 & 0 & \frac{t}{x} \\ 0 & 1 & 0 & \frac{2z}{x} \\ 0 & 0 & 1 & \frac{y}{x} \end{pmatrix}.$$

The computation of the Jacobian matrix of φ_i is more complicated. It is however remarkable that we can easily have access to its determinant. Indeed, observe that the determinant of $J(\sigma)_{(x,y,z,t)}$ is equal to $\frac{t'}{x}$ where $t' = \frac{yt+z^2}{x}$ is the last coordinate of $\sigma(x, y, z, t)$. By the chain rule, we then deduce that:

$$\det J(\varphi_i)_{(a,b,c,d)} = \frac{u_5}{u_1} \cdot \frac{u_6}{u_2} \cdots \frac{u_{i+4}}{u_i} = \frac{u_{i+1} \cdot u_{i+2} \cdot u_{i+3} \cdot u_{i+4}}{u_1 \cdot u_2 \cdot u_3 \cdot u_4} = \frac{u_{i+1} \cdot u_{i+2} \cdot u_{i+3} \cdot u_{i+4}}{abcd} \tag{3.40}$$

where the u_i's are the terms of the Somos 4 sequence initialized by $(u_1, u_2, u_3, u_4) = (a, b, c, d)$. Apart from that, the matrix $J(\varphi_i)_{(a,b,c,d)}$ has another remarkable property: if a, b, c, d are invertible in $\mathbb{Z}_p$, all the entries of $J(\varphi_i)_{(a,b,c,d)}$ lie in $\mathbb{Z}_p$; indeed they can be obtained alternatively as the partial derivatives of the terms of the generic Somos 4 sequence in which only divisions by a, b, c and d occur. As a consequence, we derive the double inclusion:

$$\begin{aligned} & p^{v(i)} \cdot \mathbb{Z}_p^4 \subset d\varphi_{i,(a,b,c,d)}\big(\mathbb{Z}_p^4\big) \subset \mathbb{Z}_p^4 \\ \text{with} \quad & v(i) = \mathrm{val}(u_{i+1}) + \mathrm{val}(u_{i+2}) + \mathrm{val}(u_{i+3}) + \mathrm{val}(u_{i+4}) \end{aligned} \tag{3.41}$$

which is the key for making the method of adaptive precision work.

Lemma 3.43. *For all i, there is at most one non invertible element among four consecutive terms of the Somos 4 sequence. In particular:*

$$v(i) = \max\big(\mathrm{val}(u_{i+1}), \mathrm{val}(u_{i+2}), \mathrm{val}(u_{i+3}), \mathrm{val}(u_{i+4})\big).$$

Proof. It is enough to prove that if $\mathrm{val}(u_{i-3}) = \mathrm{val}(u_{i-2}) = \mathrm{val}(u_{i-1}) = 0$ and $\mathrm{val}(u_i) > 0$, then $\mathrm{val}(u_{i+1}) = \mathrm{val}(u_{i+2}) = \mathrm{val}(u_{i+3}) = 0$. This can be derived easily from the defining formula of the Somos 4 sequence. □

Example. As an instructive example, we take the values $(a, b, c, d) = (1, 1, 1, 3)$ (given at precision $O(2^{10})$) and $i = 14$ corresponding to the annoying situation reported on Figure 3.10 where a term of the Somos sequence vanishes at the working precision. An explicit computation yields:

$$J(\varphi_{14})_{(1,1,1,3)} = \begin{pmatrix} \ldots0001110110 & \ldots0100010110 & \ldots1110111100 & \ldots0111110000 \\ \ldots1100101001 & \ldots1110001010 & \ldots1101010100 & \ldots0111111101 \\ \ldots0001001100 & \ldots0101001110 & \ldots1010110101 & \ldots0011101100 \\ \ldots0000000111 & \ldots0100100101 & \ldots1011100010 & \ldots0101011110 \end{pmatrix}.$$

According to the results of §3.2.1, the j-th column of this matrix (for $1 \leqslant j \leqslant 4$) encodes the loss/gain of precision on the computation of u_{j+14}. Precisely, we observe that each column of $J(\varphi_{14})_{(1,1,1,3)}$ contains an element of valuation zero, meaning that the optimal precision on u_{15}, u_{16}, u_{17} and u_{18} separately is $O(2^{10})$. In particular, we cannot decide whether u_{15} actually vanishes or does not. This sounds quite weird because we were able in any case to compute u_{19} using algorithm `somos_option1`. The point is that the formula given the value of u_{19} has the shape $u_{19} = \frac{\text{numerator}}{u_{15}}$ where the numerator is divisible by u_{15}. The approach based on the Laurent phenomenon views this simplification and is therefore able to compute u_{19} even when u_{15} vanishes!

The analysis of the optional precision on the quadruple $(u_{15}, u_{16}, u_{17}, u_{18})$ is interesting as well. However, according to Eq. (3.40), the determinant of $J(\varphi_{14})_{(1,1,1,3)}$ vanishes at the working precision; we cannot then certify its surjectivity and it seems that we are in a deadlock. In order to go further, we increase a bit the precision on the initial values: for now on, we assume that a, b, c and d (whose values are 1, 1, 1 and 3 respectively) are given at precision $O(2^N)$ with $N > 10$. The computation then shows that the determinant of $J(\varphi_{14})_{(1,1,1,3)}$ has valuation 10. Thanks to Eq. (3.14) that the quadruple $(u_{15}, u_{16}, u_{17}, u_{18})$ has 10 diffused digits of precision. We can even be more precise by computing the Hermite normal form of $J(\varphi_{14})_{(1,1,1,3)}$, which is:

$$\begin{pmatrix} 1 & 0 & 0 & 179 \\ 0 & 1 & 0 & 369 \\ 0 & 0 & 1 & 818 \\ 0 & 0 & 0 & 2^{10} \end{pmatrix}$$

meaning that the particular linear combination $179\,u_{15} + 369\,u_{16} + 818\,u_{17} - u_{18}$ could in principle be determined at precision $O(2^{N+10})$. Using this, it becomes possible to determine u_{19} at precision $O(2^N)$ (while a naive computation will lead to precision $O(2^{N-10})$ because of the division by u_{15}).

Explicit precision Lemma. Another task we have to do in order to use the theory of adaptive precision is to make explicit the constants in the precision Lemma for the particular problem we are studying. In the case of the Somos 4 sequence, the result we need takes the following form.

Proposition 3.44. *We keep the above notations and assumptions*[26]*. For all $N > v(i)$, we have:*

$$\varphi_i\big((a,b,c,d) + p^N\mathbb{Z}_p^4\big) = (u_{i+1}, u_{i+2}, u_{i+3}, u_{i+4}) + p^N\, d\varphi_{i,(a,b,c,d)}\big(\mathbb{Z}_p^4\big).$$

Proof. The proof is a slight generalization (to the case of multivariate *Laurent* polynomials) of Proposition 3.17. In fact, the only missing input is the generalization of Lemma 3.19 to our setting. Concretely, we then just have to establish that:

$$\|\varphi_i(v_2) - \varphi_i(v_1) - d\varphi_{i,v}(v_2{-}v_1)\| \leqslant \max\big(\|v_1{-}v\|, \|v_2{-}v\|\big) \cdot \|v_2{-}v_1\| \tag{3.42}$$

for all integers i and all vectors $v, v_1, v_2 \in \mathbb{Z}_p^4$ whose all coordinates are invertible. (Here $\|\cdot\|$ denotes as usual the infinite norm.) For doing so, we write φ_i as the compositum $\varphi_i = \tilde{\varphi}_i \circ \iota$ where ι is the (partially defined) embedding:

$$\iota : \mathbb{Q}_p^4 \to \mathbb{Q}_p^4, \quad (x,y,z,t) \mapsto (x,y,z,t,x^{-1},y^{-1},z^{-1},t^{-1})$$

and $\tilde{\varphi}_i : \mathbb{Q}_p^8 \to \mathbb{Q}_p^4$ is a function whose each coordinate is given by a multivariate polynomial with coefficients in $\mathbb{Z}_p$. We set $w = \iota(v)$, $w_1 = \iota(v_1)$, $w_2 = \iota(v_2)$ and compute:

$$\begin{aligned} \varphi_i(v_2) - \varphi_i(v_1) - d\varphi_{i,v}(v_2{-}v_1) &= \tilde{\varphi}_i(w_2) - \tilde{\varphi}_i(w_1) - d\tilde{\varphi}_{i,w} \circ d\iota_v(v_2{-}v_1) \\ &= \tilde{\varphi}_i(w_2) - \tilde{\varphi}_i(w_1) - d\tilde{\varphi}_{i,w}(w_2{-}w_1) \\ &\qquad + d\tilde{\varphi}_{i,w}(w_2 - w_1 - d\iota_v(v_2{-}v_1)\big). \end{aligned}$$

From the shape of $\tilde{\varphi}_i$, we deduce that the norm of its differential $d\tilde{\varphi}_{i,w}$ is at most 1. Therefore:

$$\begin{aligned} &\|\varphi_i(v_2) - \varphi_i(v_1) - d\varphi_{i,v}(v_2{-}v_1)\| \\ &\quad\leqslant \max\big(\|\tilde{\varphi}_i(w_2) - \tilde{\varphi}_i(w_1) - d\tilde{\varphi}_{i,w}(w_2{-}w_1)\|,\ \|w_2 - w_1 - d\iota_v(v_2{-}v_1)\|\big) \\ &\quad\leqslant \max\big(\|w_1{-}w\|{\cdot}\|w_2{-}w_1\|,\ \|w_2{-}w\|{\cdot}\|w_2{-}w_1\|,\ \|w_2{-}w_1 - d\iota_v(v_2{-}v_1)\|\big). \end{aligned} \tag{3.43}$$

[26] In particular, we continue to assume that the initial values a, b, c and d are invertible in $\mathbb{Z}_p$.

the last inequality coming from Lemma 3.19 applied to a shift of the function $\tilde{\varphi}_i$. Given three invertible elements x, x_1, x_2 in $\mathbb{Z}_p$, we have:

$$\left|\frac{1}{x_1} - \frac{1}{x_2} - \left(\frac{-1}{x^2}\right)\cdot(x_1{-}x_2)\right| = \left|\frac{(x_1{-}x_2)\cdot(x_1x_2 - x^2)}{x^2x_1x_2}\right| = |x_1{-}x_2|\cdot|x_1x_2 - x^2|.$$

Writing $x_1x_2 - x^2 = x(x_1 - x) + x_1(x_2 - x)$ shows that $|x_1x_2 - x^2| \leqslant \max\big(|x_1{-}x|, |x_2{-}x|\big)$. Thus:

$$\|w_2{-}w_1 - d\iota_v(v_2{-}v_1)\| \leqslant \|v_2{-}v_1\|\cdot\max\big(\|v_1{-}v\|,\ \|v_2{-}v\|\big).$$

Finally, a straightforward computation shows that $\|w_1{-}w\| = \|v_1{-}v\|$, $\|w_2{-}w\| = \|v_2{-}v\|$ and $\|w_2{-}w_1\| = \|v_2{-}v_1\|$. Inserting these inputs into Eq. (3.43), we get Eq. (3.42) as desired. □

Context of zealous arithmetic. We fix a positive integer N. We assume that the initial values a, b, c and d are given at precision $O(p^N)$. The corresponding lattice is $H = p^n\mathbb{Z}_p^4$. By Proposition 3.44, the precision Lemma applies with φ_i, x and H as soon as $N > v(i)$.

We now follow step by step the method of adaptive precision (see §3.3.1). We first set $H_i = d\varphi_{i,(a,b,c,d)}(H) = p^N\, d\varphi_{i,(a,b,c,d)}(\mathbb{Z}_p^4)$. We then have to exhibit two "simple" lattices $H_{i,\min}$ and $H_{i,\max}$ that approximates H_i from below and from above respectively. The answer is given by the inclusions (3.41): we take $H_{i,\min} = p^{N+v(i)}\mathbb{Z}_p^4$ and $H_{i,\max} = p^N\mathbb{Z}_p^4$. Concerning the lattice H'_{i-1}, it can be any lattice for which the following requirement holds: if the quadruple (x, y, z, t) is known at precision H'_{i-1}, then zealous arithmetic is able to compute $\sigma(x, y, z, t)$ at precision $H'_{i,\min} = p^{N+v(i)}\mathbb{Z}_p^4$. Since the division by x induces a loss of precision of at most $\mathrm{val}(x)$ digits, we can safely take $H'_{i-1} = p^{N+v(i)+\mathrm{val}(u_i)}\mathbb{Z}_p^4$. Instantiating the routine `Phi_stabilized_v2` (page 72) to our setting, we end up with the following stabilized version of the procedure `somos_option1`.

```
def somos_stabilized(a,b,c,d,n):
    # We assume n ⩾ 5
    # a,b,c,d are units in ℤ_p given at precision O(p^N)
    x,y,z,t = a,b,c,d
    for i in 1, 2, ..., n−4:
        u = (y*t + z*z)/x     # Precomputation at small precision
        v = val_p(y) + val_p(z) + val_p(t) + val_p(u)
        if v ⩾ N: raise PrecisionError
        lift x,y,z,t at precision O(p^(N+v+val_p(x)))
        x,y,z,t = y, z, t, (y*t + z*z)/x
    return t at precision O(p^N)
```

This algorithm always returns a correct answer at precision $O(p^N)$. It might fail — and then it raises an error — if the precision on the entries is not sufficient; by Lemma 3.43, this occurs exactly when one term of the Somos 4 sequence we are computing is indistinguishable from 0. One possible solution in that case is to increase arbitrarily the precision on a, b, c and d, rerun the algorithm and finally truncate the obtained result back to precision $O(p^N)$.

Context of lazy/relaxed arithmetic. The literal translation to the lazy world of the stabilized algorithm we have designed just above is:

```
def somos_stabilized(a,b,c,d,n):
    # We assume n ⩾ 5
    # a,b,c,d are lazy invertible p-adic numbers
    def nth_term(N):
        x,y,z,t = a(N),b(N),c(N),d(N)
        for i in 1, 2, ..., n−4:
            u = (y*t + z*z) / x
            v = val_p(y) + val_p(z) + val_p(t) + val_p(u)
            if v ⩾ N: raise PrecisionError
            x,y,z,t = y % p^(N+v), z % p^(N+v), t % p^(N+v), u % p^(N+v)
        return t % p^N
    return nth_term
```

Remark 3.45. Instead of raising an error if N is too small, it is more clever to rerun the computation at higher precision. We can implement this idea simply by replacing the statement "`raise PrecisionError`" by "`return ntn_term(2*N)`" in the code above.

The correctness of this algorithm is proved similarly to the case of zealous arithmetic by applying the precision Lemma to the function φ_i (see Proposition 3.44). We notice, in particular, that we do not need here Proposition 3.40; this is nice because making explicit the constant N_0 of Proposition 3.40 is tedious. This simplification is due to the fact that the differential $d\varphi_{i,(a,b,c,d)}$ is surjective (actually even bijective).

References

[1] Jounaïdi Abdeljaoued and Henri Lombardi. *Méthodes matricielles, introduction à la complexité algébrique*. Springer Verlag, Berlin, Heidelberg (2004)

[2] Yvette Amice. *Les nombres p-adiques*. PUF (1975)

[3] Micheal Bartholomew-Biggs, Steven Brown, Bruce Christianson and Laurence Dixon. *Automatic differentiation of algorithms*. Journal of Comp. and App. Math. **124** (2000), 171–190

[4] Christian Batut, Karim Belabas, Dominique Benardi, Henri Cohen and Michel Olivier. *User's guide to PARI-GP*. (1985–2013).

[5] Laurent Berger. *La correspondance de Langlands locale p-adique pour* $\mathrm{GL}_2(\mathbb{Q}_p)$. Astérisque **339** (2011), 157–180

[6] Pierre Berthelot and Arthur Ogus. *Notes on Crystalline Cohomology*. Princeton University Press (1978)

[7] Jérémy Berthomieu, Joris van der Hoeven, and Grégoire Lecerf. *Relaxed algorithms for p-adic numbers*. J. Number Theor. Bordeaux **23** (2011), 541–577

[8] Jérémy Berthomieu and Romain Lebreton. *Relaxed p-adic Hensel lifting for algebraic systems*. In ISSAC'12 (2012), 59–66

[9] Bhargav Bhatt. *What is... a Perfectoid Space?* Notices of the Amer. Math. Soc. **61** (2014), 1082–1084

[10] Richard Bird. *A simple division-free algorithm for computing determinants*. Information Processing Letters **111** (2011), 1072–1074

[11] Wieb Bosma, John Cannon, and Catherine Payoust. *The Magma algebra system. I. The user language.* J. Symbolic Comput. **24** (1997), 235–265

[12] Alin Bostan, Frédéric Chyzak, Grégoire Lecerf, Bruno Salvy and Éric Schost. *Differential equations for algebraic functions*. In ISSAC'07 (2007), 25–32

[13] Alin Bostan, Laureano González-Vega, Hervé Perdry and Éric Schost. *From Newton sums to coefficients: complexity issues in characteristic p*. In *MEGA'05* (2005)

[14] Olivier Brinon and Brian Conrad. *CMI Summer School notes on p-adic Hodge theory*. Available at `http://math.stanford.edu/~conrad/papers/notes.pdf` (2009)

[15] Joe Buhler and Kiran Kedlaya. *Condensation of determinants and the Robbins phenomenon*. Microsoft Research Summer Number Theory Day, Redmond (2012), available at `http://kskedlaya.org/slides/microsoft2012.pdf`

[16] Xavier Caruso. *Random matrices over a DVR and LU factorization*. J. Symbolic Comput. **71** (2015), 98–123

[17] Xavier Caruso. *Numerical stability of Euclidean algorithm over ultrametric fields*. to appear at J. Number Theor. Bordeaux

[18] Xavier Caruso, David Roe and Tristan Vaccon. *Tracking p-adic precision*. LMS J. Comp. and Math. **17** (2014), 274–294

[19] Xavier Caruso, David Roe and Tristan Vaccon. *p-adic stability in linear algebra*. In ISSAC'15 (2015)

[20] Xavier Caruso, David Roe and Tristan Vaccon. *Characteristic polynomials of p-adic matrices*. In preparation

[21] Man-Duen Choi. *Tricks or treats with the Hilbert matrix.*, Amer. Math. Monthly (1983), 301–312

[22] John Coates. *On p-adic L-functions*. Astérisque **177** (1989), 33–59

[23] Henri Cohen, *A course in computational algebraic number theory*, Springer Verlag, Berlin (1993)

[24] Pierre Colmez. *Integration sur les variétés p-adiques*. Astérisque **248** (1998)

[25] Pierre Colmez, *Fonctions d'une variable p-adique*, Astérisque **330** (2010), 13–59

[26] Marc Daumas, Jean-Michel Muller et al. *Qualité des Calculs sur Ordinateur*. Masson, Paris (1997)

[27] Roberto Dvornicich and Carlo Traverso. *Newton symmetric functions and the arithmetic of algebraically closed fields*. In AAECC-5, LNCS **356**, Springer, Berlin (1989), 216–224

[28] David Eisenbud. *Commutative Algebra: with a view toward algebraic geometry*. Springer Science & Business Media **150** (1995)

[29] Sergey Fomin and Andrei Zelevinsky. *The Laurent phenomenon*. Advances in Applied Math. **28** (2002), 119–144

[30] Martin Fürer. *Faster integer multiplication*. SIAM J. Comput. **39** (2009), 979–1005

[31] Alexander Grothendieck et al. *Séminaire de géométrie algébrique du Bois-Marie*. (1971–1977)

[32] Pierrick Gaudry, Thomas Houtmann, Annegret Weng, Christophe Ritzenthaler and David Kohel. *The 2-adic CM method for genus 2 curves with application to cryptography*. In Asiacrypt 2006, LNCS 4284, 114–129

[33] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, Cambridge (2003)

[34] Fernando Gouvêa. *Arithmetic of p-adic modular forms*. Lecture Notes in Math. **1304**, Springer-Verlag, Berlin, New-York (1988)

[35] Fernando Gouvêa. *p-adic Numbers: An Introduction*. Springer (1997)

[36] Kurt Hensel. *Über eine neue Begründung der Theorie der algebraischen Zahlen.* Jahresbericht der Deutschen Mathematiker-Vereinigung **6** (1897), 83–88

[37] James Hafner and Kevin McCurley. *Asymptotically fast triangularization of matrices over rings*. SIAM Journal of Comp. **20** (1991), 1068–1083

[38] David Harari. *Zéro-cycles et points rationnels sur les fibrations en variétés rationnellement connexes (d'après Harpaz et Wittenberg)*. Séminaire Bourbaki, Exp. 1096, Astérisque **380** (2016), 231–262

[39] Yonatan Harpaz and Olivier Wittenberg. *On the fibration method for zero-cycles and rational points*. Ann. of Math. **183** (2016), 229–295.

[40] David Harvey, Joris van der Hoeven and Grégoire Lecerf. *Even faster integer multiplication*. J. Complexity **36** (2015), 1–30

[41] Francis Hilbebrand. *Introduction to Numerical Analysis*. McGraw-Hill, New York (1956)

[42] Joris van der Hoeven. *Lazy multiplication of formal power series*. In ISSAC'97 (1997), 17–20

[43] Joris van der Hoeven. *Relax, but don't be too lazy*. J. Symbolic Comput. **34** (2002), 479–542

[44] Joris van der Hoeven. *New algorithms for relaxed multiplication*. J. Symbolic Comput. **42** (2007), 792–802

[45] Joris van der Hoeven, Grégoire Lecerf and Bernard Mourrain *The Mathemagix Language*, 2002–2012

[46] Alston Householder, *The Theory of Matrices in Numerical Analysis*. (1975)

[47] IEEE Computer Society. *IEEE Standard for Floating-Point Arithmetic, IEEE Standard 754-2008*. IEEE Computer Society, New York (2008)

[48] Erich Kaltofen and Gilles Villard. *On the complexity of computing determinants*. Comp. Complexity **13** (2005), 91–130

[49] Kiran Kedlaya. *Counting points on hyperelliptic curves using Monsky–Washnitzer cohomology*. J. Ramanujan Math. Soc. **16** (2001), 323–338

[50] Kiran Kedlaya. *p-adic Differential Equations*. Cambridge University Press (2010)

[51] Kiran Kedlaya and David Roe. *Two specifications for p-adic floating-point arithmetic: a Sage enhancement proposal*. Personal communication

[52] Neal Koblitz. *p-adic Numbers, p-adic Analysis, and Zeta-Functions*. Graduate Texts in Math. **58**, Berlin, New York, Springer-Verlag (1984)

[53] Pierre Lairez and Tristan Vaccon, *On p-adic differential equations with separation of variables*. In ISSAC'16 (2016)

[54] Alan Lauder. *Deformation theory and the computation of zeta functions*. Proc. London Math. Soc. **88** (2004), 565–602

[55] Romain Lebreton. *Relaxed Hensel lifting of triangular sets*. In MEGA'13 (2013)

[56] Arjen Lenstra, Hendrik Lenstra and László Lovász. *Factoring polynomials with rational coefficients*. Math. Ann. **261** (1982), 515–534

[57] Reynald Lercier and Thomas Sirvent. *On Elkies subgroups of ℓ-torsion points in elliptic curves defined over a finite field*. J. Number Theor. Bordeaux **20** (2008), 783–797

[58] Bernard Le Stum. *Rigid cohomology*. Cambridge University Press (2007)

[59] Carla Limongelli and Roberto Pirastu. *Exact solution of linear systems over rational numbers by parallel p-adic arithmetic*. In Parallel Processing: CONPAR 94-VAPP VI (1994), 313–323

[60] Kurt Mahler. *An interpolation series for continuous functions of a p-adic variable*. J. reine und angew. Mathematik **199** (1958), 23–34

[61] Yuri Manin. *Le groupe de Brauer-Grothendieck en géométrie diophantienne*. In Actes du Congrès International des Mathématiciens (Nice, 1970), Tome 1, Gauthier-Villars, Paris (1971), 401–411

[62] Jean-Michel Muller. *Arithmétique des ordinateurs*. Masson, Paris (1989)

[63] Jean-Michel Muller et al. *Handbook of floating-point arithmetic*. Birkhauser, Boston (2009)

[64] Richard Neidinger, *Introduction to Automatic Differentiation and MATLAB Object-Oriented Programming*. SIAM Review **52** (2010), 545–563

[65] Jurgen Neurkich. *Algebraic Number Theory*. Springer, Berlin, New York (1999)

[66] Jurgen Neurkich, *Class Field Theory. The Bonn lectures*. Edited by Alexander Schmidt. Springer, Berlin, London (2013)

[67] Alain Robert. *A course in p-adic analysis*. Springer Science & Business Media **198** (2000)

[68] R. Tyrell Rockafellar. *Variational Analysis*. Grundlehren der Mathematischen Wissenschaften **317**, Springer-Verlag (1997)

[69] Pierre Samuel. *Algebraic theory of numbers*. Paris, Hermann (1972)

[70] Peter Schneider. *p-Adic Lie groups*. Grundlehren der mathematischen Wissenschaften 344. Springer, Berlin (2011)

[71] Peter Scholze. *Perfectoid spaces: A survey*. Current Developments in Mathematics (2012)

[72] Peter Scholze. *Perfectoid spaces and their Applications*. Proceedings of the ICM 2014 (2014)

[73] Arnold Schönhage, *The fundamental theorem of algebra in terms of computational complexity*. Technical report, Univ. Tübingen (1982)

[74] Jean-Pierre Serre. *Corps locaux*. Hermann Paris (1962)

[75] Jean-Pierre Serre. *Cours d'arithmétique*. PUF (1970)

[76] Michael Somos. *Problem 1470*. Crux Mathematicorum **15** (1989), 208–208.

[77] William Stein et al. *Sage Mathematics Software*, The Sage Development Team, 2005–2017.

[78] Richard Taylor and Andrew Wiles. *Ring theoretic properties of certain Hecke algebras*. Ann. of Math. **141** (1995), 553–572

[79] Tristan Vaccon. *Précision p-adique*. PhD Thesis (2015). Available at https://tel.archives-ouvertes.fr/tel-01205269

[80] André Weil. *Numbers of solutions of equations in finite fields*. Bull. Amer. Math. Soc. **55** (1949), 497–508

[81] Andrew Wiles. *Modular elliptic curves and Fermat's Last Theorem*. Ann. of Math. **141** (1995), 443–551

[82] Franz Winkler. *Polynomial Algorithms in Computer Algebra*. Springer Wien New Work (1996)