Fast evaluation of some p-adic transcendental functions

Xavier Caruso, Marc Mezzarobba, Nobuki Takayama, Tristan Vaccon

Fast evaluation of some p-adic transcendental functions

Working draft, June 16, 2021

XAVIER CARUSO, Université de Bordeaux, CNRS, INRIA, France
MARC MEZZAROBBA, LIX, CNRS, École polytechnique, Institut polytechnique de Paris, France
NOBUKI TAKAYAMA, Kobe University, Japan
TRISTAN VACCON, Université de Limoges; CNRS, XLIM UMR 7252, France

We design algorithms for computing values of many p-adic elementary and special functions, including logarithms, exponentials, polylogarithms, and hypergeometric functions. All our algorithms feature a quasi-linear complexity with respect to the target precision and most of them are based on an adaptation to the p-adic setting of the binary splitting and bit-burst strategies.

CCS Concepts: • **Computing methodologies → Algebraic algorithms**.

Additional Key Words and Phrases: Algorithms, p-adic numbers, differential equations, binary splitting

1 INTRODUCTION

Special functions of a complex variable play a pivotal role in numerous questions arising from analysis, geometry, combinatorics and number theory. Examples include the link between the Riemann ζ-function and the distribution of prime numbers, or the Birch and Swinnerton-Dyer conjecture which relates special values of L-functions to arithmetical invariants of elliptic curves. Being able to evaluate these functions at high precision is invaluable for computing invariants or testing conjectures, and work on fast algorithms for this task over the last decades often makes it possible nowadays to reach accuracies in the millions of digits [*e.g.*, 12].

At the same time, mathematicians have realized that many complex special functions have interesting p-adic analogues. A famous example is that of p-adic L-functions, which encode subtle invariants of towers of number fields (*via* Iwasawa's theory) and, more generally, of algebraic varieties. The algorithmic counterpart of these questions also has attracted some interest. Efficient algorithms have been designed for computing the Morita p-adic Γ-function [22, §6.2] and, more recently, p-adic hypergeometric functions [1, 15] and some p-adic L-functions [3]. On a different but closely related note, since the pioneering works of Kedlaya [14], much effort has been devoted to computing the matrix of the Frobenius acting on the cohomology of p-adic algebraic varieties [*e.g.*, 17, 23].

The present paper continues this dynamic and provides new efficient algorithms for evaluating many p-adic elementary and special functions, including polylogarithms, hypergeometric functions and, more generally, solutions of "small" p-adic differential equations. In particular, our methods apply to the large class of matrices of the Frobenius acting on the cohomology of a fibration, since they satisfy differential equations of Picard-Fuchs type.

An important feature of our algorithms is that they all run in quasi-linear time in the precision. This contrasts with most previous work where the complexity was at least quadratic. The main

MM's work is supported in part by ANR grants ANR-19-CE40-0018 DeRerumNatura and ANR-20-CE48-0014-02 NuSCAP. XC's work is supported in part by ANR grant ANR-18-CE40-0026-01 CLap–CLap. TV's work is supported in part by CNRS-INSMI-PEPS-JCJC-2019 grant Patience.
Authors' addresses: Xavier Caruso, Université de Bordeaux, CNRS, INRIA, Bordeaux, France, xavier.caruso@normalesup.org; Marc Mezzarobba, LIX, CNRS, École polytechnique, Institut polytechnique de Paris, 91200, Palaiseau, France, marc@mezzarobba.net; Nobuki Takayama, Kobe University, Kobe, Japan, takayama@math.kobe-u.ac.jp; Tristan VacconUniversité de Limoges;, CNRS, XLIM UMR 7252, Limoges, France, tristan.vaccon@unilim.fr.

ingredient for reaching a quasi-optimal complexity is an adaptation to the p-adic setting of the so-called *bit-burst* method introduced by Chudnovsky and Chudnovsky [8, 9], building on the *binary splitting* technique [*e.g.*, 16] (see also [2, §178]) and other ideas dating back to Brent's work on elementary functions [6]. Our algorithms also incorporate later improvements from [24, 19, 18]. We refer to Bernstein's survey [4, esp. §12] for more history of the development of these techniques and further references.

Our starting point is the existence of recurrence relations on the partial sums of series expansions of the functions we are evaluating. Roughly speaking, the binary splitting method consists in expressing the Nth partial sum as a product of N matrices using this recurrence, and forming a balanced product tree to evaluate it (see §4.1). This approach reaches the desired quasi-linear complexity when the evaluation point x is a small integer. For more general x, we proceed in several steps: we find a sequence $x_1, \dots, x_n = x$ of intermediate evaluation points whose bit sizes increase at a controlled rate while they get closer and closer in the sense of p-adic distance. Doing this, we can use binary splitting to jump from x_i to x_{i+1}, and eventually reach x in quasi-linear time.

The remainder of the article is organized as follows. After preliminaries on the representation of p-adic numbers in §2, we introduce a p-adic analogue of bit-burst method in the simple case of $\log(t)$ in §3. The general case of solutions of linear differential equations is addressed in §4. Finally, in §5, we discuss several applications, including a fast algorithm for evaluating certain logarithmic derivatives related to the Dwork hypergeometric functions.

2 REPRESENTATION OF p-ADIC NUMBERS

Throughout this article, we fix a prime number p and a finite extension K of the field of p-adic numbers $\mathbb{Q}_p$. We recall that the p-adic valuation on $\mathbb{Q}_p$ extends uniquely to K. We denote it by val and assume that it is normalized by $\mathrm{val}(p) = 1$. We will use the notation $|\cdot|_p$ for the p-adic norm on K, defined by $|x|_p = p^{-\mathrm{val}(x)}$; in particular, we have $|p|_p = p^{-1}$.

It will be convenient to present K as an unramified extension of $\mathbb{Q}_p$ followed by a totally ramified extension given by an Eisenstein polynomial. Let us briefly recall how this works. We fix a uniformizer π of K, that is, an element of minimal positive valuation, and introduce, with k being the residue field of K:

- the ramification index e of $K/\mathbb{Q}_p$ defined by $e = \frac{1}{\mathrm{val}(\pi)}$,
- the residual degree f of $K/\mathbb{Q}_p$ defined by $f = [k : \mathbb{F}_p]$.

We choose a monic polynomial $U(X) \in \mathbb{Z}[X]$ of degree f whose reduction modulo p is irreducible. One easily checks that $U(X)$ remains irreducible in $\mathbb{Q}_p[X]$ and we can then form the field $\mathbb{Q}_q = \mathbb{Q}_p[X]/U(X)$. It follows from Hensel's lemma that $\mathbb{Q}_q$ embeds (non-canonically) into K. Let now $V(Y) \in \mathbb{Q}_q[Y]$ be the minimal polynomial of π over $\mathbb{Q}_q$. One can show that $V(Y)$ is an Eisenstein polynomial, and in particular that it lies in $\mathbb{Z}_p[X, Y]/U(X)$. Besides, Krasner's lemma [21, §3.1.5] indicates that we can assume (up to changing π) that $V(Y) \in \mathbb{Z}[X, Y]/U(X)$. Thus, viewing V as a bivariate polynomial over $\mathbb{Z}$, we have the presentations:

$$K \simeq \mathbb{Q}_p[X, Y]/(U, V), \quad \mathcal{O}_K \simeq \mathbb{Z}_p[X, Y]/(U, V) \tag{1}$$

where $\mathcal{O}_K$ denotes the ring of integers of K (which consists of the elements of nonnegative valuation). If $a \in K$ is represented by the polynomial $\sum_{i=0}^{f-1} \sum_{j=0}^{e-1} a_{ij} X^i Y^j$ (with $a_{ij} \in \mathbb{Q}_p$), one has:

$$\mathrm{val}(a) = \min_{i,j} \left(\mathrm{val}(a_{ij}) + \tfrac{j}{e} \right), \qquad |a|_p = \max_{i,j} \left(|a_{ij}|_p \cdot p^{-j/e} \right). \tag{2}$$

The presentation of K we have picked allows us to define a canonical *exact* subrings K^{ex} and $\mathcal{O}_K^{\mathrm{ex}}$ by (compare with Eq. (1))

$$K^{\mathrm{ex}} \simeq \mathbb{Q}[X, Y]/(U, V), \quad \mathcal{O}_K^{\mathrm{ex}} \simeq \mathbb{Z}[X, Y]/(U, V).$$

The ring O_K^{ex} is dense in the ring of integers O_K of K, which concretely means that given an element $a \in K$ with nonnegative valuation and an integer n, one can always find $b \in O_K^{\mathrm{ex}}$ such that $a \equiv b \pmod{p^n}$. Similarly, the ring K^{ex} is dense in K. Additionally, one can define a *height* function $h : O_K^{\mathrm{ex}} \to \mathbb{R}^+$ which measures the bit size of the elements by

$$h\Big(\sum_{i=0}^{f-1} \sum_{j=0}^{e-1} a_{ij} X^i Y^j\Big) = \max_{i,j} \log\left(1 + |a_{ij}|\right) \tag{3}$$

where the coefficients a_{ij} are integers and the notation $|a_{ij}|$ refers to the usual absolute value. If x and y are elements of O_K^{ex} of height bounded by H, one can compute the sum $x + y$ for a cost of $O(H)$ bit operations. Similarly, one can compute the product xy and reduce it for a total cost of $O\tilde{}(H)$ bit operations (where the hidden constant depends on U, V and hence on e, f).

Proposition 2.1. *Given a choice of defining polynomials U and V, there exists a constant $C \geq 0$ such that the function $h_K = h + C$ satisfies*

$$h_K(x_1 + \cdots + x_s) \leq \max(h_K(x_1), \ldots, h_K(x_s)) + \log(s),$$
$$h_K(x_1 x_2 \cdots x_s) \leq h_K(x_1) + \cdots + h_K(x_s),$$

for all $x_1, \ldots, x_s \in O_K^{\mathrm{ex}}$.

Proof. The first inequality follows (for any choice of C) from the observation that $|a_1 + \cdots + a_s| \leq s \cdot \max(|a_1|, \ldots, |a_s|)$ when $a_1, \ldots, a_s$ are integers. In order to prove the second inequality, it is enough to check that $h_K(xy) \leq h_K(x) + h_K(y)$, *i.e.* $h(xy) \leq h(x) + h(y) + C$ for some constant $C \geq 0$ and all $x, y \in O_K^{\mathrm{ex}}$. Using bilinearity of the product and the first part of the proposition, one finds that one can take $C = \log(ef) + \max_{\substack{0 \leq i \leq 2f-2 \\ 0 \leq j \leq 2e-2}} h\big(X^i Y^j\big)$. □

We fix a function $h_K : O_K^{\mathrm{ex}} \to \mathbb{R}^+$ satisfying the requirements of Proposition 2.1. It is advisable to minimize C because its value has a direct impact on the complexity of our algorithms. The proof of Proposition 2.1 shows that the value of C is related to the degrees and heights of the polynomials U and V. Since U is defined as a lift of a polynomial over $\mathbb{F}_p$, one can always assume $h(U) \leq \log p$. Bounding the height of V is more complicated but, by Krasner's lemma, it reduces to bounding the ramification of K, a problem that can be attacked using Newton polygons techniques.

3 ELEMENTARY FUNCTIONS

3.1 Logarithm

Let us start with the most basic transcendental functions, namely the p-adic logarithm and exponential [*e.g.*, 21, §4.5].

On the open unit disk centered at 1, the p-adic logarithm is defined by the usual convergent series $\log(1 - t) = -\sum_{i=1}^{\infty} t^i/i$. Given $x \in K$, $|x|_p < 1$, our aim is to compute $\log(1 - x)$ efficiently at high precision. The algorithm we describe is a straightforward adaptation of one of the classical algorithms for the same task over the reals. That the idea generalizes to $\mathbb{Q}_p$ is folklore (it is implemented in FLINT and a special case is mentioned in [4]) but, to our knowledge, no full analysis in the p-adic setting appears in the literature. Since the ideas behind this algorithm will return in the next sections, we discuss it in some detail.

First of all, it is useful to know how accurate the input has to be to determine $\log(1 - x)$ to an accuracy $O(\pi^\sigma)$.

Lemma 3.1. *Let $u, v \in K$ with $|u|_p < 1$ and $|v|_p < 1$. If one has $u = v + O(\pi^m)$ for an integer $m \geq e/(p-1)$, then $\log(1 - u) = \log(1 - v) + O(\pi^m)$.*

Proof. Writing $u = v + \pi^m w$ with $|w|_p \leq 1$ and expanding $u^p = (v + \pi^m w)^p$, we deduce that $u^p \equiv v^p \pmod{p\pi^m}$. Repeating the same argument, we find that $u^{p^s} \equiv v^{p^s} \pmod{p^s\pi^m}$ for all $s \geq 0$. Therefore $\frac{u^i}{i} \equiv \frac{v^i}{i} \pmod{\pi^m}$ for all positive integer i and the result follows from the definition of $\log(1-t)$. □

For the actual computation of $\log(1-t)$, we use the *"digit-burst"* strategy materialized by the following lemma, in which C denotes the constant of Proposition 2.1.

Lemma 3.2. *Given $x \in K$, $|x|_p < 1$, there exists a decomposition:*

$$1 - x \equiv (1 - x_1) \cdot (1 - x_2) \cdots (1 - x_\ell) \pmod{\pi^\sigma}$$

with $\ell = \lceil \log_2 \frac{\sigma}{e} \rceil$ and for all $s \in \{1, \dots, \ell\}$, $x_s \in O_K^{\mathrm{ex}}$ such that

$$\mathrm{val}(x_s) > 0,\ \mathrm{val}(x_s) \geq 2^{s-1} - 1 \text{ and } h_K(x_s) \leq (2^s - 1)\log p + C.$$

Proof. We recall that x is represented by a polynomial of the form $\sum_i \sum_j a_{ij} X^i Y^j$ with $a_{ij} \in \mathbb{Z}_p$ and $a_{00} \in p\mathbb{Z}_p$ (see §2). We define x_1 by $x_1 = \sum_i \sum_j a'_{ij} X^i Y^j$ where a'_{ij} is the unique integer in $[0, p-1]$ which is congruent to a_{ij} modulo p. Using (2) and (3), one has $\mathrm{val}(x_1) > 0$ and $h_K(x_1) \leq \log p + C$.

The quotient $(1-x)/(1-x_1)$ is congruent to 1 modulo p, hence it is represented by a polynomial of the form $\sum_i \sum_j b_{ij} X^i Y^j$ with $b_{00} \equiv 1 \pmod{p}$ and $b_{ij} \equiv 0 \pmod{p}$ for $(i,j) \neq (0,0)$. We set $1 - x_2 = \sum_i \sum_j b'_{ij} X^i Y^j$ where $b'_{ij} \in [0, p^3 - 1]$ is congruent to b_{ij} modulo p^3. One has $1 - x = (1 - x_1)\cdot(1 - x_2)\cdot(1 + O(p^3))$ with $\mathrm{val}(x_2) \geq 1$ and $h_K(x_2) \leq 3\log p + C$.

Repeating this process ℓ times, we obtain the lemma. □

Lemma 3.3. *Let $u \in O_K^{\mathrm{ex}}$ with $|u|_p < 1$. One can compute $\log(1-u)$ modulo π^σ For a cost of $O^{\tilde{}}\big(\sigma \cdot \frac{h_K(u)}{\mathrm{val}(u)}\big)$ bit operations.*

Proof. We have $\mathrm{val}\left(\frac{u^i}{i}\right) = i\,\mathrm{val}(u) - \mathrm{val}(i) \geq i\,\mathrm{val}(u) - \log_p i$. Thus $\log(1-u) \equiv \sum_{i=1}^N \frac{u^i}{i} \pmod{\pi^\sigma}$ if $N\,\mathrm{val}(u) - \log_p N \geq \frac{\sigma}{e}$. This occurs as soon as $N = O^{\tilde{}}\big(\frac{\sigma}{\mathrm{val}(u)}\big)$. Since the numerator (in O_K^{ex}) and denominator (in $\mathbb{Z}$) of any sum of the form $\sum_{i=1}^n \frac{u^i}{i_0+i}$ have height at most $n(h_K(u) + \log(N+1)) + \log n)$, one can compute the *exact* value of the finite sum $\sum_{i=1}^N \frac{u^i}{i} \in K^{\mathrm{ex}}$ in $O^{\tilde{}}(N h_K(u))$ bit operations using a divide-and-conquer strategy. The lemma follows. □

Putting everything together, we get the following theorem.

Theorem 3.4. *There exists an algorithm that takes as input an element $x \in K$, $|x|_p < 1$ and outputs $\log(1-x)$ at precision $O(\pi^\sigma)$ for a cost of $O^{\tilde{}}(\sigma)$ bit operations.*

Proof. Without loss of generality, one may assume $\sigma \geq \frac{e}{p-1}$. Combining Lemmas 3.1 and 3.2, we find a congruence of the form:

$$\log(1 - x) \equiv \log(1 - x_1) + \cdots + \log(1 - x_\ell) \pmod{\pi^\sigma}$$

with $\mathrm{val}(x_s) \geq 2^{s-1} - 1 + \frac{1}{e}$ and $h_K(x_s) \leq (2^s - 1)\log p + C$ for all $s \in \{1, \dots, \ell\}$. By Lemma 3.3, each summand $\log(1 - x_s)$ can be evaluated for a cost of $O^{\tilde{}}\big(\sigma \cdot \frac{h_K(x_s)}{\mathrm{val}(x_s)}\big) \subset O^{\tilde{}}(\sigma)$ bit operations. Since ℓ itself stays within $O(\log \sigma)$, the theorem is proved. □

It is possible to track the dependency in the field K of the complexity through the proof. Doing this, we obtain a total cost of $O^{\tilde{}}(\sigma \cdot (C + \log p))$ where C is the constant of Proposition 2.1 and the constants hidden in the $O^{\tilde{}}$ are now absolute.

3.2 Exponentiation

The exponential function. The p-adic exponential function is the function defined by $\exp(t) = \sum_{i=0}^{\infty} t^i/i!$. Using Legendre's formula, one shows that its radius of convergence is $R_{\exp} = p^{-1/(p-1)} < 1$ and that it assumes values in the open unit disk centered at 1.

Let $x \in K$ with $|x|_p < R_{\exp}$. We aim at computing $\exp(x)$ at precision $O(\pi^\sigma)$ in time $O^{\tilde{}}(\sigma)$. It is possible to use a similar "digit-burst" technique as for the logarithm. Instead, we present a different approach that we will reuse later on: we solve the equation $\log y = x$ (of unknown y) using a Newton scheme.

For this, we consider the function f defined for $|y-1|_p < 1$ by $f(y) = x - \log y$. The Newton iteration formula associated to f is

$$y_{s+1} = y_s - \frac{f(y_s)}{f'(y_s)} = y_s \cdot (1 + x - \log y_s). \tag{4}$$

Any sequence $(y_s)_{s\geq 0}$ satisfying (4) will rapidly converge to $\exp(x)$ provided that y_0 is close enough to $\exp(x)$. Noticing that $|f'(y)|_p = |f''(y)|_p = 1$ as soon as $|y-1|_p < 1$, we deduce from [7, Cor. 3.2.14] (applied with $C = R_{\exp}^{-1}$) that a sufficient condition for convergence is $|y_0 - \exp(x)|_p < R_{\exp}$. Now, observe that:

$$\operatorname{val}\left(\tfrac{x^i}{i!}\right) \geq i \cdot \left(\operatorname{val}(x) - \tfrac{1}{p-1}\right) \geq \tfrac{i}{e(p-1)}$$

the second inequality following from the fact that $\operatorname{val}(x) - \frac{1}{p-1}$ is a positive element of $\frac{1}{e(p-1)}\mathbb{Z}$. Hence, one can start the Newton iteration with $y_0 = \sum_{i=0}^{e-1} \frac{x^i}{i!} \bmod \pi^m$ where m is any integer strictly greater that $\frac{e}{p-1}$. The cost of the computation of y_0 is independent of the target precision σ. Finally, [7, Cor. 3.2.14] tells us that $y_s \equiv \exp(x) \pmod{\pi^\sigma}$ provided that $2^s\big(\operatorname{val}\big(y_0 - \exp(x)\big) - \frac{1}{p-1}\big) \geq \frac{\sigma}{e}$, which holds for $2^s \geq (p-1)\sigma$. One can take $s = O(\log \sigma)$, proving that the total cost of the algorithm is $O^{\tilde{}}(\sigma)$.

Powering. Given $\delta \in \mathbb{Z}_p$ and x in the open unit disk of K, one can give a meaning to the expression $(1+x)^\delta$ by setting:

$$(1+x)^\delta = \sum_{i=0}^{\infty} \frac{\delta(\delta-1)\cdots(\delta-i+1)}{i!} x^i. \tag{5}$$

A first idea to compute this value in quasi-optimal complexity is to write $(1+x)^\delta = \exp\big(\delta \cdot \log(1+x)\big)$. However, it does not quite work because the latter equality only makes sense when $\delta \cdot \log(1+x)$ falls inside the disk of convergence of the exponential. Instead, we observe that $y = (1+x)^\delta$ always satisfies the equation

$$\log y = \delta \cdot \log(1+x)$$

and solve it using a Newton scheme as we did in §3.2: we start by computing a first rough approximation y_0 of $(1+x)^\delta$ and then iterate the Newton operator (4). As before, the precision we need on y_0 is $O(p^{1/(p-1)})$, so that we can take the series (5) truncated after $m = \lceil e/(p-1) \rceil$ terms for y_0. The total complexity of the computation of $(1+x)^\delta$ is at most $O^{\tilde{}}(\sigma)$.

The same strategy applies to the *Artin-Hasse exponential* $\mathrm{AH}(t) = \exp(t + p^{-1}t^p + \cdots + p^{-n}t^{p^n} + \cdots)$, a useful renormalization of the p-adic exponential with a larger radius of convergence [21, §7.2].

4 SOLUTION OF DIFFERENTIAL EQUATIONS

Our goal is now to generalize the previous results to the evaluation of a large class of solutions of differential equations. We consider a linear differential equation of the form

$$a_r(t)y^{(r)}(t) + \cdots + a_1(t)y'(t) + a_0(t)y(t) = 0 \tag{6}$$

where the a_i are polynomials of degree at most d, with coefficients in O_K^{ex} of height at most ℓ. Substituting $y(t) = \sum_{n\geq 0} y_n t^n$ into (6) shows that the coefficient sequence (y_n) of a formal power series solution y must satisfy

$$b_0(n)y_n + b_1(n)y_{n-1} + \cdots + b_s(n)y_{n-s} = 0, \tag{7}$$

$$b_j(n) = \sum_{i=0}^{r} a_{i,i-r+j}(n-j)(n-j-1)\cdots(n-j-i+1), \tag{8}$$

where $s = r + d$ and $a_{i,j}$ is the coefficient of t^j in $a_i(t)$ (0 if $j < 0$). In particular, one has

$$b_0(n) = a_r(0)n(n-1)\cdots(n-r+1). \tag{9}$$

The recurrence (7) holds for all $n \in \mathbb{Z}$ if the sequence (y_n) is extended by $y_n = 0$ for $n < 0$.

4.1 Partial sums at ordinary points

The recurrence (7) can be used to evaluate partial sums of the series $y(t)$ efficiently by binary splitting. We recall and analyze an algorithm for this task, essentially the "optimized" version from [19] of a method first detailed in [8, §5–6].

We assume in this subsection that $a_r(0) \neq 0$. Then, the space of formal power series solutions of (6) has dimension r and admits a basis $(f_0, \ldots, f_{r-1})$ such that $f_j(t) = t^j + O(t^r)$. We denote by $\Phi_0(t) = \left(\frac{1}{i!}f_j^{(i)}(t)\right)$ the associated fundamental matrix. There exists $\rho > 0$ such that the f_j converge on the open disk of radius ρ centered at 0. For future use, we also define $\Phi_\xi(t) = \left(\frac{1}{i!}g_j^{(i)}(\xi + t)\right)$ at an arbitrary ξ with $a_r(\xi) \neq 0$, where g_j now is the solution such that $g_j(\xi + t) = t^j + O(t^r)$.

We are given an integer N and an element $x \in K^{\mathrm{ex}}$, written in the form $x = u/v$ with $u \in O_K^{\mathrm{ex}}$ and $v \in \mathbb{Z}$, and our task is to compute the N-th partial sum of $\Phi_0(x)$. The general idea of the algorithm is to encode the simultaneous computation of the entries of $\Phi_0(x)$ in a product of matrices that is computed in rational arithmetic by forming a balanced product tree. For space reasons, we limit ourselves here to a technical description of the procedure and refer to [18] for more context.

Given a ring R, an indeterminate Z, and $k \in \mathbb{Z}_{\geq 0}$, define the jet space $J_Z^k(R) = R[\![Z]\!]/Z^k$. Observe that computing each column of $\Phi_0(x)$ reduces to evaluating one of the f_j at $x + \Delta \in J_\Delta^r(K^{\mathrm{ex}})$. (Jet spaces in a second indeterminate Λ will be used in §4.3.) Let $\mathcal{M}$ be the set of tuples $T = (\mathsf{C}_T, \mathsf{d}_T, \mathsf{u}_T, \mathsf{v}_T, \mathsf{R}_T)$ with $\mathsf{C}_T \in (O_K^{\mathrm{ex}})^{s\times s}$, $\mathsf{d}_T \in O_K^{\mathrm{ex}}$, $\mathsf{u}_T \in J_\Delta^r(O_K^{\mathrm{ex}})$, $\mathsf{v}_T \in \mathbb{Z}$, and $\mathsf{R}_T \in J_\Delta^r(O_K^{\mathrm{ex}})^s$. We equip $\mathcal{M}$ with the product defined by

$$TT' = (\mathsf{C}_T\mathsf{C}_{T'}, \mathsf{d}_T\mathsf{d}_{T'}, \mathsf{u}_T\mathsf{u}_{T'}, \mathsf{v}_T\mathsf{v}_{T'}, \mathsf{R}_T\mathsf{C}_{T'}\mathsf{u}_{T'} + \mathsf{d}_T\mathsf{v}_T\mathsf{R}_{T'}) \tag{10}$$

where $\mathsf{R}_T, \mathsf{R}_{T'}$ are viewed as row vectors. The product is associative. In fact, multiplying elements of $\mathcal{M}$ amounts to multiplying $(s+1)\times(s+1)$ matrices of the form

$$\mathsf{M}_T = \begin{pmatrix} \mathsf{C}_T\mathsf{u}_T & 0 \\ \mathsf{R}_T & \mathsf{d}_T\mathsf{v}_T \end{pmatrix},$$

but the representation (10) makes fast computations with these special matrices easier to state and analyze.

For $n \in \mathbb{N}$, let $B(n)$ be the element of $\mathcal{M}$ defined by

$$\mathsf{C}_{B(n)} = \begin{pmatrix} 0 & b_0(n) & & \\ & & \ddots & \\ & & & b_0(n) \\ -b_s(n) & \cdots & \cdots & -b_1(n) \end{pmatrix}, \quad \begin{aligned} \mathsf{R}_{B(n)} &= (0, \ldots, 0, vb_0(n)), \\ \mathsf{d}_{B(n)} &= b_0(n), \\ \mathsf{u}_{B(n)} &= u + v\Delta, \\ \mathsf{v}_{B(n)} &= v. \end{aligned}$$

Also define $P(n_0, n_1) = B(n_1 - 1)\cdots B(n_0 + 1)B(n_0)$.

Algorithm 1 PartialSum$\big((a_i) \in O_K^{\mathrm{ex}}[t]^r, u \in O_K^{\mathrm{ex}}, v \in \mathbb{Z}, N \in \mathbb{Z}_{\geq r}, \sigma \in \mathbb{Z}_{\geq 0}\big)$

(1) Compute the polynomials b_j defined by (7), (8).
(2) Compute $\Pi = P(r, N)$ using recursively the formula $P(n_0, n_1) = P(m, n_1)P(n_0, m)$ with $m \approx (n_0 + n_1)/2$.
(3) Extract the last r columns of C_Π as a matrix $U \in (O_K^{\mathrm{ex}})^{s\times r}$.
(4) Compute and return the matrix

$$\tilde{\Phi} = [v^{-j}\mathrm{d}_\Pi^{-1}\mathrm{v}_\Pi^{-1}((u + v\Delta)^j(\mathrm{R}_\Pi)_{s-j})_i]_{0\leq i,j<r} \bmod \pi^\sigma \in K^{r\times r}$$

where $(\mathrm{R}_\Pi)_{s-j}$ is the entry of R_Π of index $s-j$, counting from zero, and ξ_i for $\xi \in J_\Delta^r(K)$ is the coefficient of Δ^i in ξ.

Proposition 4.1. *Write $f_j(t) = \sum_{n\geq 0} f_{j,n}t^n$ and let $f_j(t)_{<N} = \sum_{n=0}^{N-1} f_{j,n}t^n$. Algorithm 1 returns*

$$\left[\frac{1}{i!}\left(\frac{\mathrm{d}^i}{\mathrm{d}t^i}(f_j(t)_{<N})\right)_{t=x} \bmod \pi^\sigma\right]_{0\leq i,j<r}.$$

Proof. The recurrence relation (7) translates into

$$b_0(n)(y_{n-s+1}, \ldots, y_n)^{\mathrm{T}} = \mathrm{C}_{B(n)}(y_{n-s}, \ldots, y_{n-1})^{\mathrm{T}}.$$

Letting $Y_n = (y_{n-s}\xi^{n-1}, \ldots, y_{n-1}\xi^{n-1}, y(\xi)_{<n})^{\mathrm{T}}$ where $\xi = x + \Delta \in J_\Delta^r(K)$ and $y(t)_{<N}$ denotes the partial sum $\sum_{n=0}^{N-1} y_n t^n$, one has

$$Y_n = \begin{pmatrix} 0 & \xi & & & 0 \\ & & \ddots & & \vdots \\ & & & \xi & \vdots \\ c_s(n)\xi & \cdots & \cdots & c_1(n)\xi & 0 \\ 0 & \cdots & 0 & 1 & 1 \end{pmatrix} Y_{n-1}, \quad c_i(n) = -\frac{b_i(n)}{b_0(n)},$$

that is, $Y_n = (\mathrm{v}_{B(n)}\mathrm{d}_{B(n)})^{-1}\mathrm{M}_{B(n)}Y_{n-1}$, whenever $b_0(n) \neq 0$. Taking into account (9), it follows that

$$(y_{N-s+1}, \ldots, y_N)^{\mathrm{T}} = \mathrm{d}_\Pi^{-1}\mathrm{C}_\Pi(0, \ldots, 0, y_0, \ldots, y_{r-1})^{\mathrm{T}},$$

and $Y_N = (\mathrm{v}_\Pi\mathrm{d}_\Pi)^{-1}\mathrm{M}_\Pi Y_{r-1}$. When y is set to the element f_j of the distinguished basis defined above, the corresponding initial vector is $Y_r = ([d+j \text{ zeros}], u^{r-1}/v^{r-1}, [r-j-1 \text{ zeros}], u^j/v^j))^{\mathrm{T}}$. This leads to the formulas used at step 4. □

We turn to the complexity analysis. When A is an O_K^{ex}-algebra equipped with a distinguished O_K^{ex}-basis (e_i), such as $A = O_K^{\mathrm{ex}}[n]$, we extend h_K to A by setting $h_K(\sum_i \lambda_i e_i) = \max_i h_K(\lambda_i)$.

Lemma 4.2. *For $j = 0, \ldots, r$, the coefficient b_j of the recurrence relation (7) satisfies $h_K(b_j) \leq \ell + (s+1)\log s$.*

Proof. The coefficients of the polynomials $(n-j)\cdots(n-j-i+1)$ appearing in (8) are all bounded by $(1+j)\cdots(1+j+i-1) \leq s!$ since $i+j \leq s$ in all the terms. Coming back to the definition of h_K, we get $h_K(b_j) \leq \log(s!) + \max_{0\leq i\leq r} h_K(a_{i,i-r+j}) \leq s\log s + s + \ell$. □

Lemma 4.3. *Let H be such that $h_K(u), h_K(v) \leq H$. Let $0 \leq n_0 < n_1$ and $\Pi = P(n_0, n_1)$. We have the bounds*

$$h_K(\mathrm{u}_\Pi), h_K(\mathrm{v}_\Pi) \leq (n_1 - n_0)(H + \log(s)),$$
$$h_K(\mathrm{C}_\Pi), h_K(\mathrm{d}_\Pi) \leq (n_1 - n_0)(\ell + (s+2)(\log s + \log n_1) + 1),$$

$$h_K(\mathsf{R}_\Pi) \le (n_1 - n_0)(\ell + H + (s+3)(\log s + \log n_1) + 1).$$

Proof. We start by bounding the height of the elements of $B(n)$ for $n \le n_1$. We have by assumption $h_K(\mathsf{u}_{B(n)}), h_K(\mathsf{v}_{B(n)}) \le H$. Lemma 4.2 implies $h_K(b_j(n)) \le \beta(n) := \ell + (s+1)\log s + d\log n + 1$ for all j and n. We hence have $h_K(\mathsf{C}_{B(n)}), h_K(\mathsf{d}_{B(n)}) \le \beta(n)$ and $h_K(\mathsf{R}_{B(n)}) \le \beta(n) + H$. For $T, T' \in \mathcal{M}$, Proposition 2.1 yields

$$\begin{aligned}
h_K(\mathsf{C}_{TT'}) &\le h_K(\mathsf{C}_T) + h_K(\mathsf{C}_{T'}) + \log(s),\\
h_K(\mathsf{d}_{TT'}) &\le h_K(\mathsf{d}_T) + h_K(\mathsf{d}_{T'}),\\
h_K(\mathsf{v}_{TT'}) &\le h_K(\mathsf{v}_T) + h_K(\mathsf{v}_{T'}),\\
h_K(\mathsf{u}_{TT'}) &\le h_K(\mathsf{u}_T) + h_K(\mathsf{u}_{T'}) + \log(r).
\end{aligned}$$

The first inequality implies

$$h_K(\mathsf{C}_{T_1\cdots T_m}) \le h_K(\mathsf{C}_{T_1}) + \cdots + h_K(\mathsf{C}_{T_m}) + m\log(s)$$

whence $h_K(\mathsf{C}_\Pi) \le (n_1 - n_0)(\beta(n_1) + \log(s))$. The height bounds on d_Π, u_Π, and v_Π follow in the same way.

In the case of R_Π, Proposition 2.1 implies

$$h_K(\mathsf{R}_{TT'}) \le \max(h_K(\mathsf{R}_T\mathsf{C}_{T'}\mathsf{u}_{T'}), h_K(\mathsf{d}_T\mathsf{v}_T\mathsf{R}_{T'})).$$

With $T = B(n)$ and $T' = P(n-m, n-1)$, one has

$$\begin{aligned}
h_K(\mathsf{R}_T\mathsf{C}_{T'}\mathsf{u}_{T'}) &\le h_K(\mathsf{R}_T) + h_K(\mathsf{C}_{T'}) + h_K(\mathsf{u}_{T'}) + \log(s) + \log(r)\\
&\le \beta(n) + H + (m-1)(\beta(n) + \log(s) + H + \log(r))\\
&\le m(\beta(n) + H + 2\log(s)),\\
h_K(\mathsf{d}_T\mathsf{v}_T\mathsf{R}_{T'}) &\le h_K(\mathsf{d}_T) + h_K(\mathsf{v}_T) + h_K(\mathsf{R}_{T'}) + \log(r)\\
&\le \beta(n) + H + h_K(\mathsf{R}_{T'}) + \log(r).
\end{aligned}$$

Thus $h_K(\mathsf{R}_{TT'}) \le \max(h_K(\mathsf{R}_{T'}) + \gamma(n), m\gamma(n))$ where $\gamma(n) = \beta(n) + H + 2\log(s)$. By induction on m, one gets $h_K(\mathsf{R}_\Pi) \le (n_1 - n_0)\gamma(n_1)$. The claim follows. □

Proposition 4.4. *For u, v of height $\le H$ and $\sigma = O^{\tilde{}}(N)$, Algorithm 1 runs in $O^{\tilde{}}\big(s^\omega N(\ell + H + s)\big)$ bit operations, where ω is the exponent of matrix multiplication.*

Proof. The bulk of the cost comes from step 2. Step 2 decomposes into the computation of the tuples $B(n)$ for $n = r, \dots, N$, and the construction of a product tree from these tuples. The evaluation of b_j at n can be performed in $O^{\tilde{}}(h_K(b_j) + r\log n) = O^{\tilde{}}(\ell + s + \log n)$ operations in a divide-and-conquer fashion [11, 5], leading to a total cost of $O^{\tilde{}}(s(\ell + s + \log n) + H)$ for the construction of each $B(n)$.

Consider two subproducts $\Pi_0 = P(n_0, n_1)$ and $\Pi_1 = P(n_1, n_2)$ with $0 \le n_1 - n_0,\ n_2 - n_1 \le m$. Using the bounds from Lemma 4.3 and standard bounds on the complexity of arithmetic in $\mathbb{Z}[X]$, one sees that computing $\mathsf{u}_{\Pi_1\Pi_0}$ and $\mathsf{v}_{\Pi_1\Pi_0}$ from Π_0, Π_1 takes $O^{\tilde{}}(mr(H + \log s)) = O^{\tilde{}}(msH)$ operations. Similarly, the computation of $\mathsf{C}_{\Pi_1\Pi_0}$ and $\mathsf{d}_{\Pi_1\Pi_0}$ requires a total of $O^{\tilde{}}(m(s^{\omega+1}\ell\log N))$ operations. Finally, one can compute $\mathsf{d}_{\Pi_1}\mathsf{v}_{\Pi_1}\mathsf{R}_{\Pi_0}$ in $O^{\tilde{}}(msr \cdot (\ell + H + s\log N))$ operations, and $\mathsf{R}_{\Pi_1}\mathsf{C}_{\Pi_0}\mathsf{u}_{\Pi_0}$ in $O^{\tilde{}}(m(s^\omega + sr) \cdot (\ell + H + s\log N))$ operations by reinterpreting the product $J^r_\Delta(O^{\mathrm{ex}}_K)^s \times (O^{\mathrm{ex}}_K)^{s\times s} \to J^r_\Delta(O^{\mathrm{ex}}_K)^s$ as a matrix-matrix product $(O^{\mathrm{ex}}_K)^{r\times s} \times (O^{\mathrm{ex}}_K)^{s\times s} \to (O^{\mathrm{ex}}_K)^{r\times s}$.

Consequently, the cost of computing $P(r, N)$ from the $B(n)$ is $O^{\tilde{}}(s^\omega N(\ell + H + s))$ and dominates that of constructing the $B(n)$. Step 1 takes $O^{\tilde{}}(rs(\ell + s))$ operations. Step 4 takes $O^{\tilde{}}(r^2(\ell + H + s + \sigma))$ operations. □

When $|x| < \rho$, the series $y(x)$ converges geometrically, so $N = O(\sigma)$ terms suffice to attain the precision $O(\pi^\sigma)$. When additionally x is the image in K of an algebraic number of small bit size, one can take $H = O(1)$ in Proposition 4.4, and Algorithm 1 is enough to evaluate $y(x)$ to the precision $O(\pi^\sigma)$ in time $O^{\sim}(\sigma)$.

4.2 The "digit-burst" method

In general, though, computing $y(x)$ for $x \in K$ to a precision $O(\pi^\sigma)$ requires approximating x itself by an element of K^{ex} of height about σ, making the complexity bound roughly quadratic in σ. Chudnovsky and Chudnovsky [8] get around this issue by using the analytic continuation formula

$$\Phi_0(x_0 + x_1 + \cdots + x_m) = \Phi_{x_0+\cdots+x_{m-1}}(x_m) \cdots \Phi_{x_0}(x_1)\Phi_0(x_0)$$

along a path formed by approximations $x_0 + \cdots + x_m$ of x with an exponentially increasing number of correct digits, balancing the speed of convergence of the series with the height of the terms like in Theorem 3.4. Interestingly, the idea still applies in the p-adic case, even though the "analytic continuation" process does not allow one to escape from the disk of convergence of $\Phi_0(t)$.

We want to evaluate Φ_0 at a point x with $|x|_p < \rho$. For simplicity, we limit ourselves to $x \in O_K^{\mathrm{ex}}$ but the general case can be handled in a similar fashion. We first need some a priori bounds on the speed of convergence of series solutions of (6). Given $\rho > 0$, we denote by $\mathcal{A}_\rho$ the subring of $K[\![t]\!]$ consisting of series $f(t) = \sum_{n\geq 0} a_n t^n$ for which the sequence $(|a_n|_p \rho^n)_{n\geq 0}$ is bounded from above. The ring $\mathcal{A}_\rho$ is equipped with the *Gauss norm* $\|\cdot\|_\rho$ defined by $\|\sum_{n\geq 0} a_n t^n\|_\rho = \sup_{n\geq 0} |a_n|_p \rho^n$. It satisfies the ultrametric triangle inequality $\|f+g\|_\rho \leq \max(\|f\|_\rho, \|g\|_\rho)$ and it is multiplicative (*i.e.* $\|fg\|_\rho = \|f\|_\rho \|g\|_\rho$). Geometrically, series belonging to $\mathcal{A}_\rho$ converge on the *open* disk of center 0 and radius ρ and the Gauss norm corresponds to the sup norm on this disk taken over an algebraic closure.

Proposition 4.5. *For $f_0, f_1, \ldots, f_{r-1} \in \mathcal{A}_\rho$ and for any series $y \in K[\![t]\!]$ with $y^{(r)} + f_{r-1}y^{(r-1)} + \cdots + f_1 y' + f_0 y = 0$, one has:*

$$y \in \mathcal{A}_{\tilde{\rho}} \qquad \text{with} \qquad \tilde{\rho} = R_{\exp} \cdot \min\left(\rho, \min_{0\leq i<r} \|f_i\|_\rho^{\frac{1}{i-r}}\right),$$

$$\|y\|_{\tilde{\rho}} \leq \tilde{M} \qquad \text{with} \qquad \tilde{M} = \max_{0\leq i<r} \left|y^{(i)}(0)\right|_p,$$

where we recall that $R_{\exp} = p^{-1/(p-1)}$.

Sketch of the proof. Write $y = \sum_{n\geq 0} y_n t^n$. The coefficients y_n satisfy a recurrence of the form (8), except that the length of the recurrence in now unbounded because the f_i are series instead of polynomials. Using this recurrence, one checks by induction on n that $|n! y_n|_p \leq \tilde{M} \cdot (\tilde{\rho}/R_{\exp})^n$. Using $|n!|_p \geq R_{\exp}^{-n}$, we obtain $|y_n|_p \leq \tilde{M}\tilde{\rho}^n$ and the proposition follows. □

Define $\mathrm{slice}(x, \sigma_0, \sigma_1)$ as the result of replacing by zero the coefficients of p^k in the p-adic expansion of the coordinates of x except for $\sigma_0 \leqslant k < \sigma_1$, that is, using the notation of §2:

$$\mathrm{slice}\left(\textstyle\sum_{i,j} u_{i,j} X^i Y^j, \sigma_0, \sigma_1\right) = \sum_{i,j} ((u_{i,j} \bmod p^{\sigma_1}) - (u_{i,j} \bmod p^{\sigma_0}))X^i Y^j. \tag{11}$$

Algorithm 2 implements the computation of $\Phi_0(x)$.

Lemma 4.6. *If $a \in O_K^{\mathrm{ex}}[t]$ is a polynomial of degree at most r with $h_K(a) \leq \ell$ and $\xi \in O_K^{\mathrm{ex}}$ has height $h_K(\xi) \leq H$, then $\tilde{a} = a(\xi + t)$ has height $h_K(\tilde{a}) \leq \ell + (r+1)H + \log r$.*

Proof. With $a = \sum_i c_i t^i$, one has $\tilde{a} = \sum_j \sum_i \binom{i}{j} c_i \xi^{i-j} t^j$. Since $\log \binom{i}{j} \leq i \leq r$, the claim follows by Proposition 2.1. □

Algorithm 2 DigitBurstSolve($\boldsymbol{a} \in (O_K^{\mathrm{ex}})^r[t], \rho, M, x \in O_K^{\mathrm{ex}}, \sigma$)

(1) Let $\tilde{\rho} = R_{\exp} \cdot \min\left(\rho, \min_{0 \le i < r}\left(\frac{\|a_i\|_\rho}{\|a_r\|_\rho}\right)^{1/(i-r)}\right)$.
(2) Let $c = \lceil \max(\mathrm{val}(x), -1 + \log_p \tilde{\rho})\rceil$, $\sigma' = \sigma - \min(0, \lfloor \log_p M \rfloor)$.
(3) Set $Y = \mathrm{Id} \in (O_K^{\mathrm{ex}})^{r\times r}$, $\boldsymbol{a}_{-1} = \boldsymbol{a}$, and $x_{-1} = 0$.
(4) For $m = 0, 1, \ldots$ while $c2^m \leqslant \sigma$:
  (a) Set $\boldsymbol{a}_m(X) = \boldsymbol{a}_{m-1}(x_{m-1} + X)$.
  (b) Set $x_m = \mathrm{slice}(x,\ c\lfloor 2^{m-1}\rfloor,\ c2^m)$.
  (c) If $m = 0$ then set $N_0 = (\sigma' + \log_p M) \,/\, \log_p(\rho/|x|_p)$,
  otherwise set $N_m = \sigma' \,/\, \log_p(\tilde{\rho}/|x_m|_p)$
  (d) Set $Y = \tilde{\Phi} Y$ where $\tilde{\Phi} = \mathrm{PartialSum}(\boldsymbol{a}_m, x_m, 1, \lceil N_m\rceil, \sigma')$.
(5) Return Y.

Proposition 4.7. *Given $\boldsymbol{a} = (a_i) \in (O_K^{\mathrm{ex}})^r[t]$, $x \in O_K^{\mathrm{ex}}$ with $h_K(x) \le \sigma$ and $\rho, M \in \mathbb{R}_{>0}$ such that:*

(1) the leading coefficient a_r of $\boldsymbol{a}$ is invertible in $\mathcal{A}_\rho$ (equivalently, all roots of a_r in an algebraic closure have norm at least ρ),
(2) the entries of $\Phi_0(t)$ lie in $\mathcal{A}_\rho$ and have Gauss norm $\le M$,

Algorithm 2 computes $\Phi_0(x) \bmod \pi^\sigma$ in $O^{\tilde{}}(s^\omega\sigma(\ell + s))$ operations.

Proof. Consider iteration m of the loop. We have by construction $h_K(x_m) \le c2^m$ and $h_K(x_0 + \cdots + x_{m-1}) \le c2^{m-1}$. By Lemma 4.6, this implies $h_K(\boldsymbol{a}_m) \le c(r+1)2^{m-1} + \ell + \log r$. Using fast Taylor shift algorithms [25], one can compute the vector $\boldsymbol{a}_m$ from $\boldsymbol{a}_{m-1}$ in $O^{\tilde{}}(cr^3 h_K(x_m) + cr^2 h_K(\boldsymbol{a}_{m-1})) = O^{\tilde{}}(cr^3 2^m) = O^{\tilde{}}(s^3\sigma)$ operations. Since, by (2) and (11), $|x_m|_p \le p^{-c2^m}$, one gets $N_m = O(c^{-1}2^{-m}\sigma)$. By Proposition 4.4, the cost of the call to PartialSum is

$$O^{\tilde{}}(s^\omega N_m(h_K(\boldsymbol{a}_m) + h_K(x_m) + s) = O^{\tilde{}}(s^\omega\sigma(\ell + s)).$$

As the number of iterations is $O(\log\sigma)$, the total cost of the algorithm is $O^{\tilde{}}(s^\omega\sigma(\ell + s))$.

Regarding correctness, it follows from Proposition 4.5 applied with $f_i(t) = \frac{a_i}{a_r}(t - x_0 - \cdots - x_{m-1})$ and our choices of ρ, M and $\tilde{\rho}$ that the matrix $\tilde{\Phi}$ computed at step 4d is equal to $\Phi_{x_0+\cdots+x_{m-1}}(x_m)$ modulo $\pi^{\sigma'}$. Besides, the norm of its coefficients is bounded by M when $m = 0$ and by 1 otherwise. The product of all these matrices is then congruent to $\Phi_{x_0+\cdots+x_m}(0) = \Phi_x(0)$ modulo π^σ. □

4.3 Regular singularities

For many interesting examples, the assumption $a_r(0) \neq 0$ is not satisfied. In this case, there may not exist a full basis of formal power series solutions. Series solutions that do exist still satisfy the recurrence (7) (whose order drops since b_0 vanishes identically), but a solution $y(t) \in K[\![t]\!]$ is not necessarily characterized by its coefficients $y_0, \ldots, y_{r-1}$, and may not converge anywhere.

We focus here on the important special case where 0 is a *regular singular point*, which means, by definition, that the leading coefficient $Q_0 = b_{j_0}$ of (7), called the indicial polynomial, has degree r. It is a classical fact [*e.g.*, 20, §16] that one can then construct r linearly independent formal logarithmic series solutions

$$f_j(t) = t^{\delta_j} \sum_{\nu=0}^{\infty} \sum_{k=0}^{\kappa_j - 1} f_{j,\nu,k}\, t^\nu \frac{\log^k t}{k!}, \qquad f_{j,\nu,k} \in K(\delta_j), \tag{12}$$

where the δ_j are roots of Q_0 in an algebraic closure of K. Letting E be the set of (δ, k) such that δ is a root of Q_0 of multiplicity $\mu(\delta) > k$, the f_j can be chosen in such a way that, for each j, exactly one

Algorithm 3 RegSingPartialSum$\big((a_i), u \in O_K^{\mathrm{ex}}, v \in \mathbb{Z}, N, \sigma\big)$

(1) Compute the polynomials b_j defined by (7), (8).
Let $j_0 = \min\{j : b_j \neq 0\}$, $s' = s - j_0$ and $Q_j = b_{j_0+j}$ for $0 \leq j \leq s'$.
(2) Write $Q_0(n) = \zeta \prod_{q\in Q} \prod_{v\in\mathcal{N}_q} q(n+v)^{m_{q,v}}$ where $\zeta \in K^{\mathrm{ex}}$, $Q \subset K^{\mathrm{ex}}[n]$ is a set of monic, irreducible, non-constant polynomials, $\mathcal{N}_q \subset \mathbb{Z}_{\geq 0}$, and any two distinct $q(n+v)$ are coprime.
(3) Initialize $\tilde{\Phi}$ to an $r \times 0$ matrix over K^{ex}.
(4) For $q \in Q$:
  (a) Let γ be the image of X in $K^{\mathrm{ex}}[X]/q(X)$. Compute $\tau \in \mathbb{Z}$ such that $\tau\gamma$ is integral over O_K^{ex}.
  Let $\alpha = \tau\gamma$.
  (b) Let $\kappa = \sum_{v\in\mathcal{N}_q} m_{q,v}$.
  Initialize Ψ to an $(s'+1) \times 0$ matrix over $\mathcal{L}_{\alpha,\kappa}$.
  (c) For each pair (v_0, v_1) of consecutive elements of $\mathcal{N}_q \cup \{N\}$:
    (i) Right-shift all entries of Ψ by m_{q,v_0}, prepending zeros.
    (ii) For $k = 0, \ldots, m_{q,v_0} - 1$, append to Ψ a column of the form $([s'-1 \text{ zeros}], L^k_{\alpha,\kappa}, 0)^{\mathrm{T}}$ and attach to it the index v_0.
    (iii) Compute $\Pi = \tilde{P}(v_0, v_1)$ by binary splitting.
    (iv) Set $\Psi = \Pi \bullet \Psi$.
  (d) Compute the set $\mathcal{E}$ of roots of q in $\mathbb{Z}_p$. Fail if $|\mathcal{E}| < \deg q$.
  (e) For each c in the last row of Ψ and each $\gamma^* \in \mathcal{E}$, append to $\tilde{\Phi}$ the column vector of coefficients of $c(u/v, \tau\gamma^*, \gamma^* + v) \bmod \pi^\sigma$, cf. (14), where v is the index attached to the column of Ψ.
(5) Return $\tilde{\Phi}$.

of the coefficients $f_{j,v,k}$ for which $(\delta_j + v, k) \in E$ is nonzero, and one can take $\kappa_j \leq \sum_{\delta_i - \delta_j \in \mathbb{Z}} \mu(\delta_i)$. Moreover, by [19, Prop. 3], the relation (7) holds for the series (12) when $f_{j,v}$ is interpreted as the vector $(f_{j,v,k})_{0\leq k<\kappa_j}$ and n is set to $\delta_j + v + \Lambda$ where Λ is the operator mapping $(c_k)_k$ to $(c_{k+1})_k$. In other words, with $Q_j = b_{j_0+j}$ and $s' = s - j_0$, one has, for all $j \in \{1, \ldots, r\}$ and $v \in \mathbb{Z}$,

$$Q_0(\delta_j + v + \Lambda)(f_{j,v,k})_k + \cdots + Q_{s'}(\delta_j + v + \Lambda)(f_{j,v-s',k})_k = 0. \tag{13}$$

Making the bridge between these formal solutions and actual analytic solutions is the subject of the Dwork-Robba theory of p-adic exponents [*e.g.*, 13, §13]. While the general case seems difficult to attack, when the exponents δ_j all lie in $\mathbb{Z}_p$, the formal expression (12) does define an analytic function on each ball of the form $\{x_0(1+\xi) : |\xi|_p < 1\}$ with $|x_0|_p < \rho$ for a suitable $\rho > 0$, and binary splitting methods adapt, making it possible to evaluate the fundamental matrix $\Phi_0(t) = \big(\frac{1}{i!} f_j^{(i)}(t)\big)$ on any such ball in essentially linear time. We must limit ourselves here to a succinct description of the algorithm, leaving for future work a complete proof and complexity analysis (which however proceed along the same lines as in §4.1, see [18] for some details in the complex setting).

The procedure is summarized in Algorithm 3. Its input is similar to that of Algorithm 1, with the understanding that the number of terms N to be computed now applies separately to each set of solutions f_j whose exponents δ_j differ by integers. (We assume for simplicity that $N \geq \max(\{\delta_i - \delta_j\} \cap \mathbb{Z})$.) The differences with Algorithm 1 come from the need to deal with exponents δ_j lying in $\mathbb{Z}_p$ and with logarithmic terms.

Exponents in $\mathbb{Z}_p$ cause no serious trouble. The only subtlety is that, in order to keep the bit size of the coefficients of (13) small, we represent the δ_j as elements of formal integral extensions of O_K^{ex}. This is the role of step 4a. Computations performed in this representation are shared between solutions f_j that are Galois conjugates of each other, which roughly offsets the overhead of arithmetic in extension rings. When no two exponents δ_j differ by an element of $\mathbb{Z}$, all κ_j are

equal to 1, and Algorithm 3 reduces to something very similar to Algorithm 1, but slower by a factor $O^{\sim}(r)$.

Logarithmic terms are dealt with by viewing the operator Λ as a formal parameter in (13) and inverting the leading coefficient modulo Λ^{κ_j}. The main difficulty comes from zeros of Q_0 that differ by integers, leading to exceptional indices ν_0 where $Q_0(\delta_j + \nu_0 + \Lambda)$ is not invertible in $J_\Delta^{\kappa_j}(K^{\mathrm{ex}})$. Something special needs to be done to extend a sequence $(f_{j,\nu})_{\nu<\nu_0}$ past such a ν_0, whereas one can see that (13) leaves the choice of $f_{j,\nu,k}$ for $k < \mu(\delta_j+\nu)$ free, in accordance with the description of the basis in terms of E above.

To make this more precise, let us focus on one iteration of the loop starting at step 4. We freely use the notation of the algorithm; in particular, α and κ are fixed.

In order to generalize the binary splitting algorithm of §4.1 to the new setting, we extend some of the components of $\mathcal{M}$ to have $\mathsf{C} \in J_\Lambda^\kappa(\mathcal{O}_K^{\mathrm{ex}}[\alpha])^{s'\times s'}$, $\mathsf{d} \in \mathcal{O}_K^{\mathrm{ex}}[\alpha]$, and $\mathsf{R} \in J_\Lambda^\kappa(J_\Delta^r(\mathcal{O}_K^{\mathrm{ex}}[\alpha]))^{s'}$, the rest of the definition remaining formally the same. For $\nu \in \mathbb{Z}_{\ge 0}$, we define $\tilde{b}_0(\nu) \in \mathcal{O}_K^{\mathrm{ex}}[\alpha]$ and $\tilde{b}_j(\nu) \in J_\Lambda^\kappa(\mathcal{O}_K^{\mathrm{ex}}[\alpha])$ by $\tilde{b}_j(\nu)/\tilde{b}_0(\nu) = Q_j(\alpha+\nu+\Lambda)/(\Lambda^{-m_{q,\nu}}Q_0(\alpha+\nu+\Lambda))$ with the convention $m_{q,\nu} = 0$ when $\nu \notin \mathcal{N}_q$. This makes sense because, by definition, $\alpha+\nu$ has multiplicity $m_{q,\nu}$ as a root of Q_0. Then we define $\tilde{B}(\nu)$ and $\tilde{P}(\nu_0,\nu_1)$ similarly to $B(n)$ and $P(n_0,n_1)$ in §4.1, with s replaced by s' and each $b_j(n)$ replaced by $\tilde{b}_j(\nu)$.

We represent polynomials in $\log(t)$ appearing in coefficients and partial sums of series using elements of $\mathcal{L}_{\alpha,\kappa} = J_\Delta^r(K^{\mathrm{ex}}[\alpha])^\kappa$. The canonical basis of $\mathcal{L}_{\alpha,\kappa}$ over $J_\Delta^r(K^{\mathrm{ex}}[\alpha])$ is denoted $(L_{\alpha,\kappa}^k)_{k=0}^{\kappa-1}$. Interpreting Λ as the left-shift operator as above, we obtain an action $\bullet$ of $J_\Lambda^\kappa(K^{\mathrm{ex}}[\alpha])$ on $\mathcal{L}_{\alpha,\kappa}$. By identifying a tuple $T \in \mathcal{M}$ with the matrix M_T and viewing the latter as a matrix over $J_\Lambda^\kappa(J_\Delta^r(\mathcal{O}_K^{\mathrm{ex}}[\alpha]))$, it naturally extends to an action of $\mathcal{M}$ on $(s'+1)$-row matrices with entries in $\mathcal{L}_{\alpha,\kappa}$.

With these conventions, one can check that when $Q_0(\gamma+\nu) \neq 0$, applying $\tilde{B}(\nu)$ to a vector $Y \in \mathcal{L}_{\alpha,\kappa}^{s'+1}$ that encodes s' consecutive terms of a solution and a corresponding partial sum amounts to advancing to the next term using the recurrence (13). As in §4.1, the algorithm collects the $\tilde{B}(\nu)$ for ν between two roots of $Q_0(\gamma+n)$ in a product Π that is then applied to all solutions whose computation is in progress. When crossing a root ν_0 of $Q_0(\gamma+n)$, these solutions are "shifted to the right" (step 4(c)i) in a way that compensates for the factor $\Lambda^{m_{q,\nu_0}}$ missing in $\tilde{B}(\nu_0)$ compared to (13). "New" solutions of t-valuation $\gamma+\nu$ are added to the fundamental matrix.

Finally, at step 4e, the partial sums are converted to suitable specializations and are collected in a new matrix. More precisely, given $x = x_0(1+\xi)$ with $|x_0|_p < \rho$ and $|\xi|_p < 1$, we define the specialization $c(x,\alpha^*,\delta) \in J_\Delta^r(K^{\mathrm{ex}})$ of $c = \sum c_k L_{\alpha,\kappa}^k \in \mathcal{L}_{\alpha,\kappa}$ by

$$c(x,\alpha^*,\delta) = x_0^\delta\,(1+t)^\delta \cdot \sum_{k=0}^{\kappa-1} c_k(\alpha^*)\frac{\big(\log x_0 + \log(1+t)\big)^k}{k!} \tag{14}$$

with $t = \xi + x_0^{-1}\Delta$. Here $c_k(\alpha^*)$ is the image of c_k by the embedding of $J_\Delta^r(K^{\mathrm{ex}}[\alpha])$ into $J_\Delta^r(K)$ mapping α to α^*. The factors $(1+t)^\delta$ and $\log(1+t)$ are given by converging series and can be computed to the precision $O(\pi^\sigma)$ in $O^{\sim}(\sigma)$ operations using the algorithms of §3. As for x_0^δ and $\log x_0$, they can be chosen almost arbitrarily, any choice corresponding to a valid branch of the solution.

As in §4.1, the main contribution to the cost is that of step 4(c)iii, and it is not too hard to see that this step takes $O^{\sim}(N)$ bit operations, all other parameters being fixed. The overhead of arithmetic in $J_\Lambda^\kappa(\mathcal{O}_K^{\mathrm{ex}}[\alpha])$, summed over all α and κ, leads to an additional factor $O^{\sim}(r)$ compared to Proposition 4.4 in the complexity of the full algorithm. After using Algorithm 3 to move away

from a singularity, one can continue with the digit-burst method (since the next steps fall under the assumptions of §4.1), so that Proposition 4.7 adapts.

Formally, the algorithm also applies to partial sums of arbitrary logarithmic series solutions, even at irregular singular points. Only the existence of a full basis of the form (12) and its convergence properties depend on the regularity assumption. In particular, if we know by external arguments that a certain logarithmic series solution converges in a certain disk, we can evaluate it by binary splitting and the digit-burst method without trouble.

5 APPLICATIONS

5.1 Elementary and special functions

Elementary functions. The methods of sections §4.1 and §4.2 apply to the p-adic logarithm and exponential as these functions both satisfy simple differential equations. However, the specialized algorithms we presented in §3 perform much better in practice. In contrast, the general power function and the Artin-Hasse exponential are not covered by these methods because the differential equations annihilating them do not have small height degree.

Polylogarithms. Given a positive integer s, one can define the p-adic polylogarithm function Li_s by $\mathrm{Li}_s(t) = \sum_{i=1}^{\infty} \frac{t^i}{i^s}$. This function is solution to $(1-t)\cdot D^{s+1}(y) = D^s(y) = 0$ (with $D = t\cdot \mathrm{d}/\mathrm{d}t$), an equation with a regular singular point at the origin. It can be evaluated in essentially linear time using the algorithms of §4.2 and §4.3.

Gauss hypergeometric functions. Let a, b and c be three rational numbers with nonnegative p-adic valuation and $c \notin \mathbb{Z}^-$. To these parameters, we associate the Gauss hypergeometric function ${}_2F_1(a,b;c)$:

$$ {}_2F_1(a,b;c;t) = \sum_{i=0}^{\infty} \frac{(a)_i (b)_i}{(c)_i \cdot i!} t^i \tag{15}$$

where $(x)_i = x\cdot(x+1)\cdots(x+i-1)$. The function ${}_2F_1(a,b;c)$ has radius of convergence 1 and satisfies the differential equation $t(1-t)y'' - \big(c-(a+b+1)t\big)y' - aby = 0$, again with a regular singular point at the origin. The algorithms of §4 applied to ${}_2F_1(a,b;c;x)$ with $x \in K$, $|x|_p < 1$ run in essentially linear time for fixed a, b, c.

5.2 Gauss-Manin connections

The commutation of the Frobenius and the Gauss-Manin connection on the cohomology of p-adic varieties gives rise to differential equations with polynomial coefficients on the matrix of the Frobenius. This results in a class of equations to which one can hope using the methods of this paper to obtain interesting corollaries.

For an example of this phenomenon, start with the Gauss hypergeometric function (15) of parameters $(a,b,c) = (\frac{1}{2}, \frac{1}{2}, 1)$ and consider the logarithmic derivative

$$f(t) = {}_2F_1'(\tfrac{1}{2}, \tfrac{1}{2}; 1; t) / {}_2F_1(\tfrac{1}{2}, \tfrac{1}{2}; 1; t).$$

This formula defines a series that converges on the open unit disk. It turns out, however, that its sum f can be canonically extended to the *closed* unit disk as a consequence of [10, Lemma 3.1]. Evaluating f at points of norm 1 is in principle difficult as the series does not converge on the boundary. Recently, though, Asakura [1] and Kedlaya [15] independently noticed that values of f on the unit circle appear in the cohomology of certain algebraic fibrations.

One can try to combine this beautiful observation with the techniques of §4 to accelerate the computation of $f(x)$ when $|x|_p = 1$ and $x \not\equiv 1 \pmod{p}$. We conclude this paper with a short preview of results in this direction that we plan to develop in a future extended version.

precision σ	time (seconds)	value
12	2.08	1141554555
16	3.54	468670851430
20	5.96	372020184523305

Fig. 1. Log. derivative of ${}_2F_1(1/2, 1/2; 1; 3^p)$ with p=5

Let us first briefly review the main results of [1]. Let $X \subset \mathbb{P}^1_x \times \mathbb{P}^1_y \times \mathbb{P}^1_t$ be the variety defined by the equation $(x^2-1)\cdot(y^2-1) = t$; we view it as a fibration over $\mathbb{P}^1_t$. To this geometric situation, one can attach a cohomology space H (the log-crystalline cohomology of X), which is a module over $\mathbb{Z}_p[\![t]\!]$. For each choice of $c \in 1 + p\mathbb{Z}_p$, H is equipped with a Frobenius map $\phi_c : H \to H$, which is semi-linear in the sense that it is continuous, additive and it satisfies $\phi_c(th) = ct^p\phi(h)$ for all $h \in H$. Asakura shows that H is a free module of rank 2 and exhibits a canonical basis of it. Besides, he proves that, when $c = x^{1-p}$, the vector $\big(x(x-1)f(x), 1\big)^{\mathrm{T}}$ is the unique eigenvector of ϕ_c corresponding to an eigenvalue of norm 1. Thus, if we are able to compute ϕ_c (for $c = x^{1-p}$), we will be able to deduce the value $f(x)$ we are interested in.

For this, we use the so-called Gauss-Manin connection on H. The Gauss-Manin connection is a mapping $\nabla : H \to H\frac{\mathrm{d}t}{t}$ which encodes the variation of the cohomology with the parameter t. Writing that ∇ commutes with ϕ_c, we obtain the following differential equation, in which M_c is the matrix of ϕ_c in Asakura's basis:

$$t\,\frac{\mathrm{d}M_c(t)}{\mathrm{d}t} + \begin{pmatrix} 0 & \frac{t}{4} \\ \frac{1}{t-1} & 0 \end{pmatrix} M_c(t) - pM_c(t) \begin{pmatrix} 0 & \frac{ct^p}{4} \\ \frac{1}{ct^p-1} & 0 \end{pmatrix} = 0. \tag{16}$$

Moreover it turns out that $M_c(t)$ overconverges outside the open unit disk and actually defines an analytic function on the whole space punctured by the closed disk of center 1 and radius $R_{\exp}$. Paying particular attention to the initial conditions, we can then use the methods of §4 to evaluate M_c at any point in the domain of convergence. Since this includes all points x with $|x|_p = 1$ and $x \not\equiv 1 \pmod p$, we have reached our objective provided that $c = x^{1-p}$ is an integer of small height.

Roughly speaking, what precedes corresponds to the first step in the digit-burst method. In order to handle the next steps, we come back to the hypergeometric differential equation. Indeed, fix $x_0 \in \mathbb{Z}_p$ and let G be the solution to the Cauchy problem

$$\begin{aligned} &t(1-t)y'' + (2t-1)y' - \tfrac{1}{4}y = 0, \\ &y(x_0) = 1,\ y'(x_0) = f(x_0). \end{aligned} \tag{17}$$

Then G converges on the open disk of center x_0 and radius $R_{\exp}$ and it follows by analytic continuation that $f(x) = G'(x)/G(x)$ on this domain. Thus, once we know the value of $f(x_0)$, we can use (17) to compute $G(x)$ and $G'(x)$ by Algorithm 2, and eventually recover the value of $f(x)$.

Putting both ingredients together, we end up with an algorithm that evaluates $f(x)$ for $|x|_p = 1$ and $x \not\equiv 1 \pmod p$ with quasi-linear complexity in the output precision. Note that the complexity with respect to p is not as good because the coefficients appearing in the differential equation (16) have degree of the order of p. The estimates of Proposition 4.7 imply that the complexity in p of our algorithm is in $O^{\tilde{}}(p^{\omega+1})$, which makes it practical for small values of p only. It would be interesting to try to lower this complexity by capitalizing on the sparsity of the polynomials appearing in (16).

We have implemented part of the above algorithm in SageMath, based on ore_algebra[1]. Although it is still in development, the application of it seems to be promising as the timing data on Figure 1 demonstrates. Note that a naive evaluation of this function with precision σ requires to evaluate series of p^σ terms, e.g., $5^{20} = 95367431640625$.

[1] https://github.com/mkauers/ore_algebra, branch `padic`

Acknowledgements. We thank the anonymous referees, whose comments on a previous version of this paper led to major presentation improvements.

REFERENCES

[1] M. Asakura. 2020. An Algorithm of Computing Special Values of Dwork's p-Adic Hypergeometric Functions in Polynomial Time. arXiv: 1909.02700 [math].

[2] M. Beeler, R. W. Gosper, and R. Schroeppel. 1972. Hakmem. AI Memo 239. MIT Artificial Intelligence Laboratory.

[3] K. Belabas and B. Perrin-Riou. 2021. Symboles modulaires surconvergents et fonctions L p-adiques. arXiv: 2101.06960 [math].

[4] D. J. Bernstein. 2008. Fast multiplication and its applications. In *Algorithmic Number Theory*. Cambridge University Press, 325–384.

[5] A. Bostan, T. Cluzeau, and B. Salvy. 2005. Fast algorithms for polynomial solutions of linear differential equations. In *ISSAC '05*. ACM, 45–52.

[6] R. P. Brent. 1976. The complexity of multiple-precision arithmetic. In *The Complexity of Computational Problem Solving*, 126–165.

[7] X. Caruso. 2017. Computations with p-adic numbers. In *Journées Nationales de Calcul Formel*. Les cours du CIRM. Volume 5, 1–75.

[8] D. V. Chudnovsky and G. V. Chudnovsky. 1988. Approximations and complex multiplication according to Ramanujan. In *Ramanujan revisited*. Academic Press, 375–472.

[9] D. V. Chudnovsky and G. V. Chudnovsky. 1990. Computer algebra in the service of mathematical physics and number theory. In *Computers in Mathematics*. Dekker, 109–232.

[10] B. Dwork. 1969. p-adic cycles. *Inst. Hautes Études Sci. Publ. Math.*, 37, 27–115.

[11] G. Estrin. 1960. Organization of computer systems – the fixed plus variable structure computer. In *Proceedings of the Western Joint IRE-AIEE-ACM Computer Conference*. ACM, 33–40.

[12] F. Johansson. 2017. Arb: efficient arbitrary-precision midpoint-radius interval arithmetic. *IEEE Transactions on Computers*, 66, 8, 1281–1292.

[13] K. S. Kedlaya. 2010. *p-adic differential equations*. Cambridge University Press.

[14] K. S. Kedlaya. 2001. Counting points on hyperelliptic curves using Monsky-Washnitzer cohomology. *J. Ramanujan Math. Soc.*, 16, 4, 323–338.

[15] K. S. Kedlaya. 2019. Frobenius structures on hypergeometric equations. arXiv: 1912.13073 [math].

[16] P. Kogge and H. Stone. 1973. A parallel algorithm for the efficient solution of a general class of recurrence equations. *IEEE Transactions on Computers*, C-22, 786–793.

[17] A. G. B. Lauder. 2004. Deformation theory and the computation of zeta functions. *Proc. London Math. Soc. (3)*, 88, 3, 565–602.

[18] M. Mezzarobba. 2011. *Autour de l'évaluation numérique des fonctions D-finies*. Thèse de doctorat. École polytechnique.

[19] M. Mezzarobba. 2010. NumGfun: a package for numerical and analytic computation with D-finite functions. In *ISSAC '10*. ACM, 139–146.

[20] E. G. C. Poole. 1936. *Introduction to the theory of linear differential equations*. Clarendon Press.

[21] A. M. Robert. 2013. *A course in p-adic analysis*. Springer.

[22] F. Rodriguez Villegas. 2007. *Experimental number theory*. Oxford University Press.

[23] J. Tuitman. 2019. Computing zeta functions of generic projective hypersurfaces in larger characteristic. *Math. Comp.*, 88, 315, 439–451.

[24] J. van der Hoeven. 2001. Fast evaluation of holonomic functions near and in regular singularities. *Journal of Symbolic Computation*, 31, 6, 717–743.

[25] J. von zur Gathen and J. Gerhard. 1997. Fast algorithms for Taylor shifts and certain difference equations. In *ISSAC '97*. ACM, 40–47.