Fast computation of the Nth term of an algebraic series in positive characteristic

Article · February 2016

CITATIONS
2

READS
78

3 authors, including:

A. Bostan
National Institute for Research in Computer Science and Control
120 PUBLICATIONS **1,943** CITATIONS
SEE PROFILE

Gilles Christol
Sorbonne Université
65 PUBLICATIONS **936** CITATIONS
SEE PROFILE

Fast Computation of the Nth Term of an Algebraic Series over a Finite Prime Field*

Alin Bostan
Inria (France)
alin.bostan@inria.fr

Gilles Christol
IMJ (France)
gilles.christol@imj-prg.fr

Philippe Dumas
Inria (France)
philippe.dumas@inria.fr

ABSTRACT

We address the question of computing one selected term of an algebraic power series. In characteristic zero, the best algorithm currently known for computing the Nth coefficient of an algebraic series uses differential equations and has arithmetic complexity quasi-linear in $\sqrt{N}$. We show that over a prime field of positive characteristic p, the complexity can be lowered to $O(\log N)$. The mathematical basis for this dramatic improvement is a classical theorem stating that a formal power series with coefficients in a finite field is algebraic if and only if the sequence of its coefficients can be generated by an automaton. We revisit and enhance two constructive proofs of this result for finite prime fields. The first proof uses Mahler equations, whose sizes appear to be prohibitively large. The second proof relies on diagonals of rational functions; we turn it into an efficient algorithm, of complexity linear in $\log N$ and quasi-linear in p.

CCS Concepts

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

Keywords

algebraic series; finite fields; Mahler equations; diagonals; p-rational series; section operators; algebraic complexity

1. INTRODUCTION

One of the most difficult questions in modular computations is the complexity of computations mod p *for a large prime* p *of coefficients in the expansion of an algebraic function.*
D.V. Chudnovsky & G.V. Chudnovsky, 1990 [11].

Context. Algebraic functions are ubiquitous in all branches of pure and applied mathematics, notably in algebraic geometry, combinatorics and number theory. They also arise at the confluence of several fields in computer science: functional equations, automatic sequences, complexity theory. From a computer algebra perspective, a fundamental question is the efficient computation of power series expansions of algebraic functions. We focus on the particular question of computing *one selected term* of an *algebraic power series* f whose coefficients belong to a *field of positive characteristic* p. Beyond its relevance to complexity theory, this problem is important in applications to integer factorization and point-counting [4].

Setting. More precisely, we assume in this article that the ground field is the prime field $\mathbb{F}_p$ and that the power series f is (implicitly) given as the unique solution in $\mathbb{F}_p[[x]]$ of

$$E(x, f(x)) = 0, \qquad f(0) = 0,$$

where E is a polynomial in $\mathbb{F}_p[x, y]$ that satisfies

$$E(0, 0) = 0 \quad \text{and} \quad E_y(0, 0) \neq 0.$$

(Here, and hereafter, E_y stands for the partial derivative $\frac{\partial E}{\partial y}$.)

Given as input the polynomial E and an integer $N > 0$, our algorithmic problem is to efficiently compute the Nth coefficient f_N of $f = \sum_i f_i x^i$. The efficiency is measured in terms of number of arithmetic operations $(\pm, \times, \div)$ in the field $\mathbb{F}_p$, the main parameters being the index N, the prime p and the bidegree (h, d) of E with respect to (x, y).

In the particular case $d = 1$, the algebraic function f is actually a rational function, and thus the Nth coefficient of its series expansion can be computed in $O(\log N)$ operations in $\mathbb{F}_p$, using standard binary powering techniques [20, 14].

Therefore, it will be assumed in all that follows that $d > 1$.

Previous work and contribution. The most straightforward method for computing the coefficient f_N of the algebraic power series f proceeds by undetermined coefficients. Its arithmetic complexity is $O(N^d)$. Kung and Traub [19] showed that the formal Newton iteration can accelerate this to $\tilde{O}(dN)$. (The soft-O notation $\tilde{O}(\cdot)$ indicates that polylogarithmic factors are omitted.) Both methods work in arbitrary characteristic and compute f_N together with all $f_i, i < N$.

In characteristic zero, it is possible to compute the coefficient f_N faster, without computing all the previous ones. This result is due to the Chudnovsky brothers [10] and is based on the classical fact that the coefficient sequence $(f_n)_{n\geq 0}$ satisfies a linear recurrence with polynomial coefficients [12, 9, 3]. Combined with baby steps/giant steps techniques, this leads to an algorithm of complexity quasi-linear in $\sqrt{N}$. Except for the very particular case of rational functions ($d = 1$), no faster method is currently known.

Under restrictive assumptions, the baby step/giant step algorithm can be adapted to the case of positive characteristic [4]. The main obstacle is the fact that the linear recurrence satisfied by the sequence $(f_n)_{n\geq 0}$ has a leading coefficient that may vanish at various indices. In the spirit of [4, §8], p-adic lifting techniques could in principle be used, but we are not aware of any sharp analysis of the sufficient p-

*We warmly thank the referees for their very helpful comments.

ISSAC'16, July 19–22, 2016, Waterloo, ON, Canada
ACM ISBN 978-1-4503-4380-0/16/07.
DOI: http://dx.doi.org/10.1145/2930889.2930904

adic precision in the general case. Anyways, the best that can be expected from this method is a cost quasi-linear in $\sqrt{N}$.

We attack the problem from a different angle. Our starting point is a theorem due to the second author in the late 1970s [2, Th. 12.2.5]. It states that a formal power series with coefficients in a finite field is algebraic if and only if the sequence $(f_n)_{n\geq 0}$ of its coefficients can be generated by a p-automaton, i.e., f_N is the output of a finite-state machine taking as input the digits of N in base p. This implicitly contains the roots of a $\log N$-method for computing f_N, but the size of the p-automaton is at least $p^{(d+h)^2}$, see e.g. [22].

In its original version [7] the theorem was stated for $\mathbb{F}_2$, but the proof extends *mutatis mutandis* to any finite field. A different proof was given by Christol, Kamae, Mendès-France and Rauzy [8, §7]. Although constructive in essence, these proofs do not focus on computational complexity aspects.

Inspired by [1] and [13], we show that each of them leads to an algorithm of arithmetic complexity $O(\log N)$ for the computation of f_N, after a precomputation that may be costly for p large. On the one hand, the proof in [8] relies on the fact that the sequence $(f_n)_{n\geq 0}$ satisfies a divide-and-conquer recurrence. However, we show (Sec. 2) that the size of the recurrence is polynomial in p^d, making the algorithm uninteresting even for very moderate values of p. On the other hand, the key of the proof in [7] is to represent f as the diagonal of a bivariate rational function. We turn it (Sections 3–4) into an efficient algorithm that has complexity $O(\log N)$ after a precomputation whose cost is $\tilde{O}(p)$ only. To our knowledge, the only previous explicit occurrence of a $\log N$-type complexity for this problem appears in [11, p. 121], which announces the bound $O(p \cdot \log N)$ but without proof.

Structure of the paper. In Sec. 2, we propose an algorithm that computes the Nth coefficient f_N using Mahler equations and divide-and-conquer recurrences. Sec. 3 is devoted to the study of a different algorithm, based on the concept of diagonals of rational functions. We conclude in Sec. 4 with the design of our main algorithm.

Cost measures. We use standard complexity notation. The real number $\omega > 2$ denotes a feasible exponent for matrix multiplication i.e., there exists an algorithm for multiplying $n \times n$ matrices with entries in $\mathbb{F}_p$ in $O(n^\omega)$ operations in $\mathbb{F}_p$. The best bound currently known is $\omega < 2.3729$ from [16]. We use the fact that many arithmetic operations in $\mathbb{F}_p[x]_d$, the set of polynomials of degree at most d in $\mathbb{F}_p[x]$, can be performed in $\tilde{O}(d)$ operations: addition, multiplication, division, *etc.* The key to these results is a divide-and-conquer approach combined with fast polynomial multiplication [23, 5, 18]. A general reference on fast algebraic algorithms is [17].

2. USING MAHLER EQUATIONS

We revisit, from an algorithmic point of view, the proof in [8, §7] of the fact that the coefficients of an algebraic power series $f \in \mathbb{F}_p[[x]]$ can be recognized by a p-automaton. Starting from an algebraic equation for $f = \sum_n f_n x^n$, we first compute a Mahler equation satisfied by f, then we derive an appropriate divide-and-conquer recurrence for its coefficient sequence $(f_n)_n$, and use it to compute f_N efficiently. We will show that, although the complexity of this method is very good (i.e., logarithmic) with respect to the index N, the computation cost of the recurrence, and actually its mere size, are extremely high (i.e., exponential) with respect to the algebraicity degree d.

2.1 From algebraic equations to Mahler equations and DAC recurrences

Mahler equations are functional equations which use the Mahler operator M_p, or M for short, acting on formal power series by substitution: $Mg(x) = g(x^p)$. Their power series solutions are called *Mahler series*. In characteristic 0, there is no a priori link between algebraic equations and Mahler equations; moreover, a series which is at the same time algebraic and Mahler is a rational function [21, §1.3].

By contrast, over $\mathbb{F}_p$, a Mahler equation is nothing but an algebraic equation, because of the Frobenius endomorphism that permits writing $g(x^p) = g(x)^p$ for any $g \in \mathbb{F}_p[[x]]$. Hence a Mahler series in $\mathbb{F}_p[[x]]$ is an algebraic series. Conversely, an algebraic series is Mahler, since if f is algebraic of degree d then the $(d+1)$ power series f, $Mf = f^p$, ..., $M^d f = f^{p^d}$ are linearly dependent over $\mathbb{F}_p(x)$.

Concretely, the successive powers f^{p^k} $(0 \leq k \leq d)$ are expressed as linear combinations of 1, f, ..., f^{d-1} over $\mathbb{F}_p(x)$, and any dependence relation between them delivers a non-trivial Mahler equation with polynomial coefficients

$$c_0(x)f(x) + c_1(x)f(x^p) + \cdots + c_K(x)f(x^{p^K}) = 0, \qquad (1)$$

whose order K is at most d.

EXAMPLE 1 (A TOY EXAMPLE). Consider $E = x + y - y^3$ in $\mathbb{F}_5[x, y]$. Let $f = -x - x^3 + 2x^5 - 2x^7 + 2x^{11} + \cdots$ be the unique solution in $x\,\mathbb{F}_5[[x]]$ of $E(x, f(x)) = 0$. The expressions of f, f^5, f^{25}, f^{125} as linear combinations of 1, f, f^2 with coefficients in $\mathbb{F}_5(x)$ give the columns of the matrix

$$\begin{bmatrix} 0 & x & -2\,x^7 + x^5 + x & x^{41} - x^{37} + \cdots \\ 1 & 1 & x^8 - x^6 + 1 & x^{40} - 2\,x^{38} + \cdots \\ 0 & x & -2\,x^7 + x^5 + x & x^{41} - x^{37} + \cdots \end{bmatrix}.$$

The first three columns are linearly dependent and lead to the Mahler equation

$$x^4(1 - x^2 - x^4)f(x) - (1 + x^4 - 2x^6)f(x^5) + f(x^{25}) = 0. \qquad (2)$$

A Mahler equation instantly translates into a recurrence of divide-and-conquer type, in short a DAC recurrence. Such a recurrence links the value of the index-n coefficient f_n with some values for the indices n/p, $n/p^2 \ldots$, and some shifted indices. The translation is absolutely simple: a term $x^s f(x)$ in equation (1) translates into f_{n-s}; more generally, a term $x^s f(x^q)$ becomes $f_{(n-s)/q}$, with the convention that a coefficient f_ν is zero if ν is not a nonnegative integer.

EXAMPLE 2 (A BINOMIAL CASE). Let $p > 2$ be a prime number and let $f = x + x^2 + \cdots$ be the algebraic series in $x\,\mathbb{F}_p[[x]]$ solution of $E(x, y) = x + (1 + y)^{p-1} - 1$. The rewriting

$$x + (1+f)^{p-1} - 1 = x + \frac{1+f^p}{1+f} - 1 = \frac{(x-1)(1+f) + (1+f^p)}{1+f}$$

yields $f^p = -x + (1-x)f$ and $f^{p^2} = -x^p + (1-x^p)f^p$; hence $f^{p^2} = (1-x)^{p+1} - 1 + (1-x)^{p+1}f$. The situation is highly non-generic, since all powers f^{p^k} are expressible as linear combinations of 1 and f only. This delivers the second-order Mahler equation

$$(x^{p-1} - x^p)\,f(x) - (1 + x^{p-1} - x^p)\,f(x^p) + f(x^{p^2}) = 0,$$

which translates into a recurrence on the coefficients sequence

$$f_{n-p+1} - f_{n-p} - f_{\frac{n}{p}} - f_{\frac{n-p+1}{p}} + f_{\frac{n-p}{p}} + f_{\frac{n}{p^2}} = 0. \qquad (3)$$

It is possible to make the recurrence more explicit by considering the p^2 cases according to the value of the residue of n modulo p^2.

2.2 Nth coefficient via a DAC recurrence

The DAC recurrence relation attached to the Mahler equation (1), together with enough initial conditions, can be used to compute the Nth coefficient f_N of the algebraic series f. The complexity of the resulting algorithm is linear with respect to N. The reason is that the coefficient $c_0(x)$ in (1) generally has more than one monomial, and as a consequence, all the coefficients f_i, $i < N$ are needed to compute f_N.

EXAMPLE 3 (A BINOMIAL CASE, CONT.). We specialize Ex. 2 by taking $p = 7$ and compute f_N with $N = 100$. Applying (3) to $n = 106$, we need to know the value for $n = 99$, next for $n = 98$, $n = 15$, $n = 14$, and also for $n = 97, \ldots, n = 91$, $n = 14$, $n = 13$, $n = 2$. In the end, it appears that all values of f_n for the indices n smaller than $N = 100$ are needed to compute f_N.

It is possible to decrease dramatically the cost of the computation of f_N, from linear in N to logarithmic in N. The idea is to derive from (1) another Mahler equation with the additional feature that its trailing coefficient $c_0(x)$ is 1. To do so, it suffices to perform a change of unknown series. Putting $f(x) = c_0(x)g(x)$, we obtain the Mahler equation

$$g(x) + c_1(x)c_0(x)^{p-2}g(x^p) + \cdots + c_K(x)c_0(x)^{p^K-2}g(x^{p^K}) = 0. \quad (4)$$

Obviously this approach assumes that c_0 is not zero. But, as proved in [2, Lemma 12.2.3], this is the case if (1) is assumed to be a *minimal-order* Mahler equation satisfied by f.

The series $g(x)$ is no longer a power series, but a Laurent series in $\mathbb{F}_p((x))$. We cut it into two parts, its negative part $g_-(x)$ in $\mathbb{F}_p[x^{-1}]$ and its nonnegative part $h(x)$ in $\mathbb{F}_p[[x]]$:

$$g_-(x) = \sum_{n<0} g_n x^n, \qquad h(x) = \sum_{n\geq 0} g_n x^n.$$

Let us rewrite Eq. (4) in the compact form $L(x, M)g(x) = 0$, where $L(x, M)$ is a skew polynomial in the variable x and in the Mahler operator M. Plugging $g(x) = g_-(x)+h(x)$ in it yields three terms. The first is the power series $L(x, M)h(x)$. The second is the nonnegative part $-b(x)$ of $L(x, M)g_-(x)$, in $\mathbb{F}_p[x]$. The third is the negative part of $L(x, M)g_-(x)$, and it is zero because it is the only one which belongs to $x^{-1}\mathbb{F}_p[x^{-1}]$. Eq. (4) is rewritten as a new (inhomogeneous) Mahler equation $L(x, M)h(x) = b(x)$, namely

$$h(x) + c_1(x)c_0(x)^{p-2}h(x^p) + \cdots + c_K(x)c_0(x)^{p^K-2}h(x^{p^K}) = b(x). \quad (5)$$

EXAMPLE 4 (A TOY EXAMPLE, CONT.). The change of series $f(x) = x^4(1 - x^2 - x^4)h(x)$ in (2) gives the new equation

$$h(x) - x^{12}(1-x)(1+x)(1+x^2+2x^4)(1-x^2-x^4)^3h(x^5) + x^{92}(1-x^2-x^4)^{23}h(x^{25}) = -x - x^5 + x^7 + 2\,x^9 + \cdots + x^{149} - x^{157} - 2\,x^{159}$$

and a recurrence

$$h_n = h_{\frac{n-12}{5}} + 2h_{\frac{n-14}{5}} + \cdots - h_{\frac{n-92}{25}} + \cdots,$$

while f_n is given by $f_n = h_{n-4} - h_{n-6} - h_{n-8}$ for $n \geq 8$.

The right-hand side $b(x)$ of this new inhomogeneous equation is obtained as the nonnegative part of $-(c_1c_0^3g_-^5 + c_2c_0^{23}g_-^{25})$, where $c_0 = x^4(1-x^2-x^4)$, $c_1 = 2x^6 - x^4 - 1$, $c_2 = 1$ are the coefficients of Eq. (2), and where $g_- = -2x^{-1} - x^{-3}$, the negative part of f/c_0, can be determined starting from $(f \bmod x^4)$.

To compute the coefficient f_N for $N = 1251$, this approach only requires the computation of thirteen terms of the sequence h_n, namely for $n \in \{0, 3, 5, 7, 43, 45, 47, 243, 245, 247, 1243, 1245, 1247\}$. This number of terms behaves like $3\log_p N$, and compares well to the use of a recurrence like (3), which requires N terms.

At this point, it is intuitively plausible that the Nth coefficient f_N can be computed in arithmetic complexity $O(\log N)$: to obtain f_N, we need a few values h_N and these ones are essentially obtained from a bounded number of $h_{N/p}, h_{N/p^2}, \ldots$.

2.3 Nth coefficient via the section operators

This plausibility can be strengthened and made clearer with the introduction of the *section operators* (sometimes called *Cartier operators*, or *decimation operators*) and the concept of *p-rational series*.

DEFINITION 1. *The section operators $S_0, \ldots, S_{p-1}$, with respect to the radix p, are defined by*

$$S_r \sum_{n\geq 0} f_n x^n = \sum_{k\geq 0} f_{pk+r}x^k, \qquad \text{for} \quad 0 \leq r < p. \quad (6)$$

In other words, there is a section operator for each digit of the radix-p numeration system and the rth section operator extracts from a given series its part associated with the indices congruent to the digit r modulo p. These operators are linear and well-behaved with respect to the product:

$$S_r[f(x)g(x)] = \sum_{s+t\equiv r \bmod p} x^{\lfloor \frac{s+t}{p} \rfloor} S_s f(x) S_t g(x).$$

In particular, for $g(x) = h(x^p)$ the previous formula becomes

$$S_r[f(x)h(x^p)] = S_r[f(x)] \times h(x), \quad (7)$$

because of the obvious relationships with the Mahler operator $S_0M = \mathrm{I}_{\mathbb{F}_p[[x]]}$, $S_rM = 0$ for $r > 0$. The action of the section operators can be extended to the field $\mathbb{F}_p((x))$ of formal Laurent series and the same properties apply to the extended operators, which we denote in the same way.

The section operators permit to express the coefficient h_N of a formal power series $h(x)$ using the radix-p digits of N.

LEMMA 2. *Let h be in $\mathbb{F}_p[[x]]$ and let $N = (N_\ell \cdots N_1N_0)_p$ be the radix-p expansion of N. Then*

$$h_N = (S_{N_\ell} \cdots S_{N_1}S_{N_0}h)(0). \quad (8)$$

PROOF. We first apply the section operator S_{N_0} to the series $h(x)$ associated with the least significant digit of N, that is we start from N_0 and we pick every pth coefficient of $h(x)$. This gives $S_{N_0}h(x) = h_{N_0} + h_{N_0+p}x + h_{N_0+2p}x^2 + \cdots$. Iterating this process with each digit produces the series

$$S_{N_\ell} \cdots S_{N_1}S_{N_0}h(x) = h_N + h_{N+p^{\ell+1}}x + \cdots,$$

whose constant coefficient is h_N. □

2.4 Linear representation

Let us return to the algebraic series $f(x)$ and its relative $h(x)$. The Mahler equation (5) can be rewritten

$$h(x) = b(x) + a_1(x)h(x^p) + \cdots + a_K(x)h(x^{p^K}). \quad (9)$$

The coefficients $b(x)$ and $a_k(x)$, $1 \leq k \leq K$, are polynomials of degrees at most D, say. As a matter of fact, the vector space $\mathcal{W}$ of linear combinations

$$s(x) = a(x) + b_0(x)h(x) + b_1(x)h(x^p) + \cdots + b_K(x)h(x^{p^K})$$

with $a(x)$ and all $b_k(x)$ in $\mathbb{F}_p[x]_D$ is stable under the action of the section operators. Indeed, from

$$s(x) = (a(x) + b_0(x)b(x)) + (b_0(x)a_1(x) + b_1(x))h(x^p) + \cdots + (b_0(x)a_K(x) + b_K(x))h(x^{p^K})$$

we deduce

$$\begin{aligned} S_r s(x) = {} & S_r\left(a(x) + b_0(x)b(x)\right) \\ & + S_r\left(b_0(x)a_1(x) + b_1(x)\right)h(x) + \cdots \\ & + S_r\left(b_0(x)a_K(x) + b_K(x)\right)h(x^{p^{K-1}}), \end{aligned} \quad (10)$$

and the polynomials $S_r(a+b_0b)$ and $S_r(b_0a_k+b_k)$, $1 \le k \le K$, have degrees not greater than $2D/p \le D$.

The important consequence of this observation is that we have produced a finite dimensional $\mathbb{F}_p$-vector space $\mathcal{W}$ that contains $h(x)$ and that is stable under the sections $(S_r)_{0\le r<p}$. This will enable us to effectively compute the coefficient h_N using formula (8), by performing matrix computations.

The vector space $\mathcal{W}$ is spanned over $\mathbb{F}_p$ by the family $\mathcal{F} = \left\{x^i, 0 \le i \le D\right\} \cup \left\{x^j h(x^{p^k}), 0 \le j \le D, 0 \le k \le K\right\}$. Let A_r be a matrix of S_r with respect to $\mathcal{F}$, let C be the column vector containing only zero entries except an entry equal to 1 at index $(j,k) = (0,0)$, and L be the row matrix of the evaluations at 0 of the elements of $\mathcal{F}$.

With these notation, Eq. (8) rewrites in matrix terms:

$$h_N = LA_{N_\ell} \cdots A_{N_1} A_{N_0} C. \quad (11)$$

Definition 3. *The family of matrices $\left(L, (A_r)_{0\le r<p}, C\right)$ is a* linear representation *of the series $h(x)$. A power series that admits a linear representation is said to be* p-rational.

As it was pointed out in [1, p. 178], an immediate consequence of Eq. (11) is that the coefficient h_N can be computed using only $O(\log N)$ matrix-vector products in size $(D+1)(K+1)$, therefore in arithmetic complexity $O((DK)^2 \log N)$.

Actually, the matrices A_r are structured, and their product by a vector can be done faster than quadratically. Indeed, Eq. (10) shows that one can perform a matrix-vector product by the matrix A_r using K polynomial products in $\mathbb{F}_p[x]_D$, thus in quasi-linear complexity $\tilde{O}(DK)$. We deduce:

Proposition 4. *The Nth coefficient h_N of the solution h of (9) can be computed in time $\tilde{O}((DK)\log N)$.*

2.5 Nth coefficient using Mahler equations

We are ready to prove the main result of this section.

Theorem 5. *Let E be a polynomial in $\mathbb{F}_p[x,y]_{h,d}$ such that $E(0,0) = 0$ and $E_y(0,0) \ne 0$, and let $f \in \mathbb{F}_p[[x]]$ be its unique root with $f(0) = 0$. Algorithm 1 computes the Nth coefficient f_N of f using $\tilde{O}(d^3h^2p^{3d}\log N)$ operations in $\mathbb{F}_p$.*

To do this, we first need a preliminary result that will also be useful in Sec. 4. It provides size and complexity bounds for the remainder of the Euclidean division of a monomial in y by a polynomial in y with coefficients in $\mathbb{F}_p[x]$.

Lemma 6. *Let $E = \sum_i e_i(x)y^i$ be a polynomial in $\mathbb{F}_p[x][y]$ of degree d in y and at most h in x. Then, for $D \ge d$,*

$$y^D \bmod E = \frac{1}{e_d^{D-d+1}}\left(r_0(x) + \cdots + r_{d-1}(x)y^{d-1}\right),$$

where the r_i's are in $\mathbb{F}_p[x]$ of degree at most $h(D-d+1)$.

One can compute the r_i's using $\tilde{O}(hdD)$ operations in $\mathbb{F}_p$.

Algorithm **Nth coefficient via Mahler equations**

Input A polynomial $E(x,y) \in \mathbb{F}_p[x,y]$ with $E(0,0) = 0$ and $E_y(0,0) \ne 0$, and an integer $N \ge 0$.
Output The Nth coefficient f_N of the unique formal power series $f \in \mathbb{F}_p[[x]]$ solution of $E(x,f(x)) = 0$, $f(0) = 0$.

1. Use Algorithm 2 to compute a minimal-order Mahler equation (with $c_0 = c_{0,v_0}x^{v_0} + \cdots + c_{0,d_0}x^{d_0}$, $c_{0,v_0} \ne 0$):
   $L(x,M)f(x) = c_0(x)f(x) + \cdots + c_K(x)f(x^{p^K}) = 0$.
2. Deduce an equation $L'(x,M)g(x) = 0$ (4), with $c'_0 = 1$, satisfied by $g(x) = f(x)/c_0(x)$.
3. Compute $f(x) \bmod x^{v_0}$ by Newton iteration, and deduce the negative part $g_-(x)$ of $f(x)/c_0(x)$.
4. Compute the nonnegative part $b(x)$ of $-L'(x,M)g_-(x)$
   ▷ The nonnegative part $h(x)$ of $f(x)/c_0(x)$ satisfies the equation $L'(x,M)h(x) = b(x)$ (5).
5. Get a linear representation $(L, (A_r)_{0\le r<p}, C)$ for $h(x)$.
6. **for** N' in $\{N-d_0, \ldots, N-v_0\}$
   write $N' = (N'_\ell \ldots N'_0)_p$ and
   compute $h_{N'} = LA_{N'_\ell} \cdots A_{N'_0} C$;
7. **return** $f_N = c_{0,v_0}h_{N-v_0} + \cdots + c_{0,d_0}h_{N-d_0}$.

Algorithm 1. Nth coefficient via Mahler equations.

Proof. The degree bounds are proved by induction on D. The computation of $y^D \bmod E$ can be performed by binary powering. The last step dominates the cost of the whole process. It amounts to first computing the product of two polynomials in $\mathbb{F}_p[x,y]$ of degree at most $d-1$ in y and of degree $O(hD)$ in x, then reducing the result modulo E. Both steps have complexity $\tilde{O}(hdD)$: the multiplication is done via Kronecker's substitution [17, Cor. 8.28], the reduction via an adaptation of the Newton iteration used for the fast division of univariate polynomials [17, Th. 9.6]. □

The first step of Algorithm 1 is the computation of a Mahler equation for the algebraic series f. We begin by analyzing the sizes of this equation, and the cost of its computation.

Theorem 7. *Let E be a polynomial in $\mathbb{F}_p[x,y]_{h,d}$ such that $E(0,0) = 0$ and $E_y(0,0) \ne 0$, and let $f \in \mathbb{F}_p[[x]]$ be its unique root with $f(0) = 0$. Algorithm 2 computes a Mahler equation of minimal-order satisfied by f using $\tilde{O}(d^\omega hp^d)$ operations in $\mathbb{F}_p$. The output equation has order at most d and polynomial coefficients in $\mathbb{F}_p[x]$ of degree at most dhp^d.*

Proof. We first prove the correctness part. Since the R_i's all live in the vector space generated by $1, f, \ldots, f^{d-1}$ over $\mathbb{F}_p(x)$, the algorithm terminates and returns a Mahler operator of order K at most d.

At step s we have $R_s = y^{p^s} \bmod E$, thus $R_s(x,f) = f^{p^s}$. Since $\sum_{k=0}^K c_k R_k = 0$, this yields $L(f) = \sum_{k=0}^K c_k f^{p^k} = 0$, so L is a Mahler operator for f. It has minimal order, because otherwise $R_0, \ldots, R_{K-1}$ would be linearly dependent over $\mathbb{F}_p(x)$. Therefore, the algorithm is correct.

By Lemma 6, $e_d^{p^s-d+1}R_s$ is in $\mathbb{F}_p[x,y]$ of degree in y at most $d-1$ and degree in x at most $h(p^s-d+1)$. Moreover, $R_0, \ldots, R_K$ can be computed in time $\tilde{O}(hdp^K)$.

By Cramer's rule, the degrees of the c_k's are bounded by $dhp^K \le dhp^d$. Testing the rank condition, and solving for the c_k's amounts to polynomial linear algebra in size at most $d \times (d+1)$ and degree at most hp^d. This can be done in complexity $\tilde{O}(d^\omega hp^d)$, using Storjohann's algorithm [24]. □

Algorithm **AlgeqToMahler**

Input $E(x,y) \in \mathbb{F}_p[x,y]$ with $E(0,0)=0$ and $E_y(0,0) \neq 0$.
Output A minimal-order monic Mahler operator for the unique $f \in \mathbb{F}_p[[x]]$ with $E(x,f(x))=0$ and $f(0)=0$.

$R_0 = y$;
for $s = 1, 2, \ldots$ **do**
  $R_s = R_{s-1}^p \bmod E$ ▷ $R_s = y^{p^s} \bmod E$
  if $\operatorname{rank}_{\mathbb{F}_p(x)}(R_0, R_1, \ldots, R_s) < s+1$ **then**
    Solve $\sum_{k=0}^{s-1} \frac{c_k}{c_s} R_k = -R_s$ for $c_0, \ldots, c_s$ in $\mathbb{F}_p[x]$
    return $\sum_{k=0}^{s} c_k M^k$

Algorithm 2. From algebraic equations to Mahler equations.

We are now ready to prove Theorem 5.

Proof of Theorem 5. The correctness of Algorithm 1 follows from the discussion in Sec. 2.4. By Theorem 7, the output of Step 1 is a Mahler equation of order $K \leq d$ and coefficients c_j of maximum degree $H \leq dhp^K$. In particular, $d_0 \leq dhp^d$. The cost of Step 1 is $\tilde{O}(d^\omega hp^d)$. Step 2 consists in computing the coefficients $c'_j = c_j c_0^{p^j-2}$ for $1 \leq j \leq K$, of maximum degree $D \leq Hp^K$. It can be performed in time $\tilde{O}(KHp^K) = \tilde{O}(d^2hp^{2d})$. Step 3 has negligible complexity $\tilde{O}(dv_0) = \tilde{O}(d^2hp^d)$ using the Kung-Traub algorithm [19]. Step 4 requires $\tilde{O}(KHp^K) = \tilde{O}(d^2hp^{2d})$ operations. By Proposition 4 the last steps of the algorithm can be done using $\tilde{O}((DK)\log N) = \tilde{O}((d^2hp^{2d})\log N)$ for each N', thus for a total cost of $\tilde{O}((d^3h^2p^{3d})\log N)$. This dominates the whole computation. □

Generically, the size bounds in Theorem 5 are quite tight. This is illustrated by the following example.

Example 5 (A cubic equation). Let us consider the algebraic equation $E(x,y) = x - (1+x)y + x^2y^2 + (1+x)y^3 = 0$ in $\mathbb{F}_3[x,y]$. The assumption $E_y(0,0) = -1 \neq 0$ is fulfilled. We find a Mahler equation of order 3, whose coefficients $c_0 = x^{10} - x^{11} + \cdots - x^{40} + x^{45}$, c_1, c_2, c_3 have respectively heights 45, 47, 50, 32. We compute within precision $O(x^{10})$ the solution $f(x) = x - x^2 - x^3 + x^5 + \cdots - x^{44} + \cdots$, and the negative part $g_-(x) = x^{-9} + x^{-7} + x^{-6} - x^{-4} - x^{-2}$ of $g = f/c_0$. We find a new operator $L'(x,M)$ by computing the product $L(x,M)c_0(x)$ in $\mathbb{F}_3[x,M]$ and next a new equation $L'(x,M)y(x) = b(x)$ for the nonnegative part $h(x)$ of $g(x)$ by computing the polynomial $b(x) = L'(x,M)g_-(x)$. It appears that the coefficients $c'_0 = 1$, c'_1, c'_2, c'_3 and the right-hand side $b(x)$ have respectively heights 0, 92, 365, 1157, and 1103. We arrive at a linear representation with size $(1+3+1) \times (1+1157) = 5790$ for $p=3$. If we increase the prime number p, we successively find sizes 155190 for $p=5$, and 1342725 for $p=7$. We do not pursue further because it is clear that such sizes make the computation unfeasible.

3. USING DIAGONALS

To improve the arithmetic complexity with respect to the prime number p, we need a different way to represent the algebraic series f. It is given by Furstenberg's theorem, and relies on the concept of *diagonal of rational functions*.

3.1 From algebraic equation to diagonal

$$\frac{y-(2-x)y^2-3y^3+(4-2x^2)y^4+4xy^5}{(1-x)-(1-x)y-y^2+(1-x^2)y^3+xy^4} =$$
$$\begin{array}{llllllllll}
+y & +xy & +x^2y & +x^3y & +x^4y & +x^5y & +x^6y & +x^7y & \\
-y^2 & & & & & & & & \\
-3y^3 & -xy^3 & & +x^3y^3 & +2x^4y^3 & +3x^5y^3 & +4x^6y^3 & +5x^7y^3 & \\
-y^4 & & -x^2y^4 & & +x^4y^4 & +2x^5y^4 & +3x^6y^4 & +4x^7y^4 & \\
-3y^5 & & -3x^2y^5 & -2x^3y^5 & & +3x^5y^5 & -4x^6y^5 & +x^7y^5 & \\
-y^6 & +4xy^6 & -3x^2y^6 & -4x^3y^6 & -3x^4y^6 & & +5x^6y^6 & +x^7y^6 & \\
-3y^7 & +5xy^7 & -4x^2y^7 & +4x^3y^7 & +2x^4y^7 & +4x^5y^7 & & +2x^7y^7 & \\
 & & & & & & & +\cdots &
\end{array}$$

Figure 1. A diagonal of a bivariate rational series (see Ex. 6).

Furstenberg's theorem [15] tells us that every algebraic series is the diagonal of a bivariate rational function

$$f(x) = D\frac{a(x,y)}{b(x,y)}, \qquad b(0,0) \neq 0. \tag{12}$$

This means that $f(x) = \sum_i c_{i,i}x^i$, where $a/b = \sum_{i,j} c_{i,j}x^iy^j$.

Moreover, under the working assumption $E_y(0,0) \neq 0$, the rational function a/b is explicit in terms of E:

$$a(x,y) = yE_y(xy,y), \qquad b(x,y) = E(xy,y)/y.$$

(Notice that $b(0,0) = E_y(0,0) \neq 0$.) If the polynomial $E(x,y)$ has degree d and height h, then the partial degrees of a and b are bounded as follows

$$d_x = \max(\deg_x a, \deg_x b) \leq h, \qquad d_y = \max(\deg_y a, \deg_y b) \leq d+h. \tag{13}$$

Example 6 (A quartic equation). Consider

$$f(x) = x + x^3 + x^4 + 3x^5 + 5x^6 + 2x^7 + 4x^8 - x^9 - x^{10} - 2x^{11} + \cdots,$$

the unique solution in $x\,\mathbb{F}_{11}[[x]]$ of the algebraic equation

$$E(x,y) = -x + (1+x)y - (1+x^2)y^2 - y^3 + (1+x)y^4 = 0.$$

Then f is the diagonal of the rational function a/b with

$$a = y - (2-x)y^2 - 3y^3 + (4-2x^2)y^4 + 4xy^5,$$

$$b = (1-x) - (1-x)y - y^2 + (1-x^2)y^3 + xy^4.$$

Fig. 1 shows pictorially that $f(x)$ is the diagonal of a/b.

We will follow the path drawn in [6], and compute the Nth coefficient f_N using a/b as intermediate data structure.

3.2 Nth coefficient via diagonals

Bivariate section operators can be defined by mimicking (6). Although they depend on two residues modulo p, we will always use them with the same residue for both variables; thus we will simply denote them by S_r for $0 \leq r < p$. A nice feature is that they commute with the diagonal operator,

$$S_rD = DS_r.$$

Together with (12), this property translates the action of the section operators from the algebraic series f to the rational function a/b. Moreover, property (7) extends to bivariate section operators and justifies the following computation

$$S_r\frac{a(x,y)}{b(x,y)} = S_r\frac{a(x,y)b(x,y)^{p-1}}{b(x,y)^p} = S_r\frac{a(x,y)b(x,y)^{p-1}}{b(x^p,y^p)} = \frac{S_ra(x,y)b(x,y)^{p-1}}{b(x,y)}.$$

Algorithm **Nth coefficient via diagonals**

Input A polynomial $E(x,y) \in \mathbb{F}_p[x,y]$ with $E(0,0) = 0$ and $E_y(0,0) \neq 0$, and an integer $N \geq 0$.
Output The Nth coefficient f_N of the unique formal power series $f \in \mathbb{F}_p[[x]]$ solution of $E(x, f(x)) = 0$, $f(0) = 0$.

1. Compute $a(x,y) = yE_y(xy,y), b(x,y) = E(xy,y)/y$.
   ▷ At this point $f = D(a/b)$.
2. Set $d_x = \max(\deg_x a, \deg_x b)$, $d_y = \max(\deg_y a, \deg_y b)$, $L = [1, 0, \cdots, 0]$, and C the column vector expressing $a(x,y)$ in the canonical basis of $\mathbb{F}_p[x,y]_{d_x,d_y}$.
3. Compute $B = b^{p-1}$.
4. **for** r from 0 to $p-1$
   compute the image of the canonical basis $T_r x^i y^j = \sum_{n,m} A_{r;i,j;n,m} x^n y^m$
   and store the result in the matrix A_r
5. **return** $f_N = LA_{N_\ell} \cdots A_{N_0} \frac{C}{b(0,0)}$ for $N = (N_\ell \dots N_0)_p$.

Algorithm 3. Nth coefficient, via diagonals.

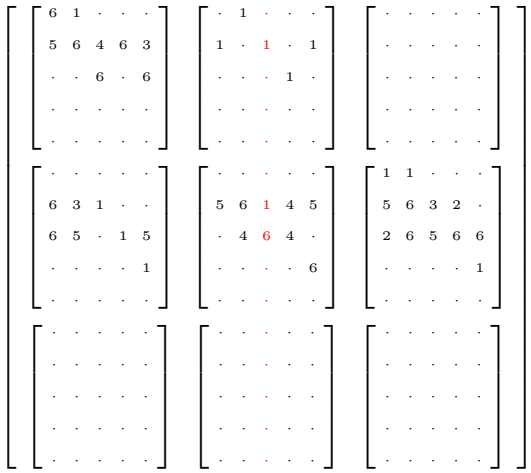

Figure 2. The matrix A_1 in Ex. 7.

To put it plainly

$$S_r \frac{a}{b} = \frac{S_r a b^{p-1}}{b}.$$

This yields an action on bivariate polynomials v of what we may call pseudo-section operators, defined by

$$T_r v = S_r v b^{p-1}, \qquad 0 \leq r < p,$$

hence

$$(S_{r_1} \cdots S_{r_k}) \left(\frac{a}{b}\right) = \frac{(T_{r_1} \cdots T_{r_k})(a)}{b}, \qquad 0 \leq r_i < p.$$

At this point, it is not difficult to see that the vector subspace $\mathbb{F}_p[x,y]_{d_x,d_y}$, where d_x and d_y are the maximal partial degrees of a and b defined in (13), is stable under the pseudo-section operators. The supports of b and of polynomials v in $\mathbb{F}_p[x,y]_{d_x,d_y}$ belong to a small rectangle $[0,d_x] \times [0,d_y]$. Raising b to the power $p-1$ enlarges the rectangle by a factor $p-1$ and the product by v results in a rectangle p times larger than the small rectangle. The pseudo-section operators, essentially based on a Euclidean division by p, send the large rectangle back into the small rectangle.

As a consequence we have at our disposal a new linear representation $\big(L, (A_r)_{0 \leq r < p}, C\big)$, this time based on the vector space $\mathbb{F}_p[x,y]_{d_x,d_y}$, and on its canonical basis. The row vector L has all its components equal to 0 except the first one, equal to 1, because all monomials of the canonical basis take the value 0 at $x = 0$, $y = 0$, except the monomial 1. The family $(A_r)_{0 \leq r < p}$ of matrices of size $(1+d_x)(1+d_y)$ expresses the pseudo-section operators T_r. The column vector C contains the coordinates of the polynomial a.

The crucial difference with the linear representation in Sec. 2.2 is that the dimension of the new one is much smaller.

It remains to compute f_N by using the rational nature with respect to the radix p of the power series $f(x)$. The process is summarized in Algorithm 3. Anew we conclude that the arithmetic complexity for the computing of f_N is $O(\log N)$. More precisely, since the linear representation has size $(1+d_x)(1+d_y)$, the computation of f_N is of order $O(d_x^2 d_y^2 \log N)$.

Example 7 (A cubic equation, cont.). We return to Ex. 5 with $p = 7$. We find $a = -y - xy^2 + 3y^3 + 3xy^4 + 2x^2y^4$ and $b = -1 + x - xy + y^2 + xy^3 + x^2y^3$, hence $d_x = 2$, $d_y = 4$. Next we compute $B = b^{p-1}$ and we build the linear representation. The main point is the size of the representation, namely $(1+d_x)(1+d_y) = 15$. This is ridiculously small as compared to the value obtained in Ex. 5. Even more importantly, the size remains the same if we change the prime number.

Although it is not necessary that we see the linear representation with our human eyes, it is quite normal that we watch it if only to control the computations. The matrices A_r, whose size is $(1+d_x)(1+d_y)$, have indices which are pairs of pairs. We can view them as square matrices of size $1+d_x$ whose elements are square blocks of size $1+d_y$. The coordinates over $x^i y^j$ of the image $T_r x^n y^m$ is in the block with indices (i,n) at place (j,m).

We only display the matrix A_1 (the zero coefficients are replaced by dots), in Fig. 2. It gives us for example the value of $T_1 xy^2$. Because the exponent of x is 1, we look at the column of blocks number 1 (the second one) and because the exponent of y is 2 we look at the column number 2 (the third one) in the matrices of this column (red column). In the first block, which corresponds to x^0, we see a coefficient 1 in row number 1, hence a term $x^0 y^1 = y$. In the second block, we see a coefficient 1 in row number 1 hence a term $x^1 y^1 = xy$ and a coefficient 6 in row number 2 hence a term $6x^1y^2 = 6xy^2$. We conclude $T_1 xy^2 = y + xy + 6xy^2$.

3.3 Precomputation

All the needed information to build the linear representation is contained in the polynomial $B = b^{p-1}$. Indeed, when we compute the image of an element $x^i y^j$ of the canonical basis by the pseudo-section operator T_r, we first multiply B by $x^i y^j$; this transformation is merely a translation of the exponents, that requires no arithmetic operation. Next, we extract the monomials whose exponents are congruent to r modulo p, again without any computation in $\mathbb{F}_p$. The conclusion is that, apart the final computation to obtain the value f_N, the cost comes from the computation of $B = b^{p-1}$. This can be done using binary powering and Kronecker's substitution. Since B has partial degrees at most pd_x in x and at most pd_y in y, the arithmetic complexity is $\tilde{O}(p^2 d_x d_y)$.

Gathering the study of the computation and the precomputation, and using Eq. (13), we obtain the following result.

Theorem 8. *Let E be in $\mathbb{F}_p[x,y]_{h,d}$ such that $E(0,0) = 0$ and $E_y(0,0) \neq 0$, and let $f \in \mathbb{F}_p[[x]]$ be its unique root with $f(0) = 0$. Algorithm 3 computes the Nth coefficient f_N of f in $O(h^2(d+h)^2 \log N) + \tilde{O}(p^2 h(d+h))$ operations in $\mathbb{F}_p$.*

The striking points are the decreasing of the exponent of p

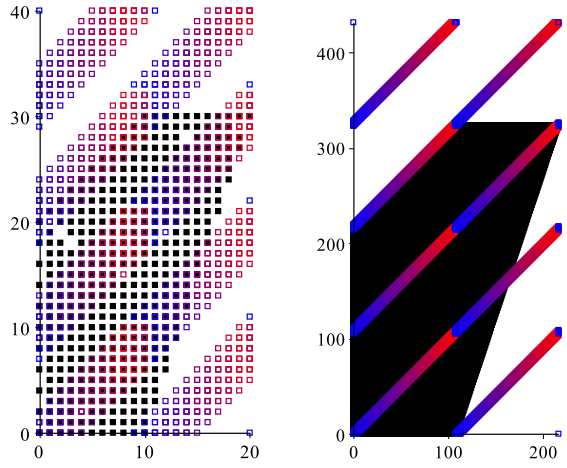

Figure 3. The useful coefficients of $B = b^{p-1}$ are located in the diagonal strips. See Ex. 8 for explanations.

from $3d$ to 2, and the replacement of the *multiplicative* complexity $p^{3d} \times \log N$ of Theorem 5 by the *additive* complexity $p^2 + \log N$. This is already a tremendous improvement. In the next section, we will present a further improvement.

4. A FASTER ALGORITHM

We will improve the algorithm of the previous section in order to decrease the exponent of the prime p from 2 to 1.

4.1 A small part of the large power is enough

As pointed out in Sec. 3.3, all the information needed to build the square matrices $A_0, \ldots, A_{p-1}$ of the linear representation is contained in the large power $B = b^{p-1}$. However building the A_i's does not require the whole polynomial B.

To see this, assume $0 \le \alpha \le (p-1)d_x$ and $0 \le \beta \le (p-1)d_y$ are such that the monomial $x^\alpha y^\beta$ of B contributes to one of the matrices A_r. Since $A_{r;i,j;n,m}$ is the coefficient of $x^n y^m$ in $T_r(x^i y^j) = S_r(x^i y^j B)$, this means that $i + \alpha \equiv j + \beta \bmod p$ for some $0 \le i \le d_x$ and $0 \le j \le d_y$. As a consequence, the difference exponent $\delta = \beta - \alpha$ necessarily belongs to $[-d_y, d_x] + p\mathbb{Z}$. For large p, this means that only a fraction $1/p$ of B is useful. The aim of this section is to take an algorithmic advantage of this remark.

EXAMPLE 8 (A CUBIC EQUATION, CONT.). The previous phenomenon is exemplified in Fig. 3 for the equation $E(x, y) = x - (1 + x)y + x^2y^2 + (1 + x)y^3 = 0$, already encountered in Ex. 5 and 7. We use $p = 11$ on the right-hand side, and $p = 109$ on the right-hand side. The polynomial B is represented by its monomial support (in black). Besides, the points (α, β) defined by $\alpha = pk + r - i$, $\beta = p\ell + r - j$, with $0 \le i \le d_x$, $0 \le j \le d_y$ are colored, with a color which goes from blue to red when r goes from 0 to $p - 1$ (but the pieces overlap). This generates strips with slope 1: only monomials in these strips contribute to the linear representation. For p small ($p = 11$), the strips cover almost the whole rectangle, but for p moderately large ($p = 109$) the proportion of useful coefficients in B becomes much smaller.

To emphasize the difference exponent $\delta = \beta - \alpha$, we perform a change of variables $x := x/t$, $y := t$, which produces Laurent polynomials with respect to t, namely $b(x/t, t)$ and

$$B(x/t, t) = b(x/t, t)^{p-1} = \sum_\delta \pi_\delta(x) t^\delta.$$

By construction, the coefficient $\pi_\delta(x)$ is a polynomial, namely the δ-diagonal $\sum_i B_{i,i+\delta} x^i$ of $B(x, y) = \sum_{i,j} B_{i,j} x^i y^j$.

Let δ_- be the opposite of the lowest degree with respect to t of $b(x/t, t)$, and δ_+ be the degree with respect to t of $b(x/t, t)$. An immediate bound is $\delta_- \le d_x$ and $\delta_+ \le d_y$.

By the previous discussion, all the information needed to build up the matrices of the linear representation is contained in the polynomials $\pi_\delta(x)$, for δ in

$$\Delta := ([-d_y, d_x] + p\mathbb{Z}) \cap [(1-p)\delta_-, (p-1)\delta_+].$$

This set is a union of small intervals of length $d_x + d_y + 1$ regularly spaced by p and within $[(1-p)d_x, (p-1)d_y]$.

4.2 Partial bivariate powering

Our aim is to show that the collection of $\pi_\delta(x)$ for $\delta \in \Delta$ can be computed in complexity quasi-linear in p. The starting point is to write

$$B(x/t, t) = \frac{b(x/t, t)^p}{b(x/t, t)} = \frac{b(x^p/t^p, t^p)}{b(x/t, t)}.$$

Here $b(x^p/t^p, t^p)$ is obtained by a mere rewriting of $b(x/t, t)$, hence at no cost, while $1/b(x/t, t)$ is a rational function. To be more concrete, we denote

$$b(x/t, t) = \sum_{v=-\delta_-}^{\delta_+} b_v(x) t^v, \qquad \frac{1}{b(x/t, t)} = \sum_{u \ge \delta_-} c_u(x) t^u,$$

where the coefficients $b_v(x)$ are polynomials in x and the coefficients $c_v(x)$ are rational functions in x.

Now, the polynomial $\pi_\delta(x)$ is expressed by a convolution

$$\pi_\delta(x) = \sum_{u+pv=\delta} c_u(x) b_v(x^p). \tag{14}$$

The constraint $u + pv = \delta$ implies $u \equiv \delta \bmod p$. Therefore, it is sufficient to compute the $c_\delta(x)$ for all $\delta \in \Delta' = p\delta_- + \Delta$.

To do this efficiently, we will exploit the fact that the sequence of rational functions $(c_u(x))_u$ satisfies a linear recurrence with coefficients in $\mathbb{F}_p[x]$. The coefficients of the recurrence relation are those of the polynomial $t^{\delta_-} b(x/t, t)$.

We are facing the following issue: since we need the values of $c_\delta(x)$ for indices of order $\delta = \Theta(p)$, we can not unroll the recurrence relation directly, otherwise the complexity (and actually the mere arithmetic size of the rational functions $c_i(x)$, $i < u$, computed along the way) becomes quadratic in p. Fortunately, one can compute the Nth term of a recurrent sequence without computing all the previous coefficients. The key is the following lemma.

LEMMA 9. [14] *Let R be a commutative ring and $a(t)/b(t)$ be a rational function in $R(t)$ with $b(0)$ invertible in R. Let d be the degree of $b(t)$ and $b^*(t) = t^d b(1/t)$ be its reciprocal. For any integer $N \ge 0$, the coefficients c_N, c_{N+1}, …, c_{N+d-1} of the formal power series expansion of $a(t)/b(t) = \sum_{n \ge 0} c_n t^n$ are the coefficients of t^{2d-2}, …, t^d, t^{d-1} in the product*

$$(t^N \bmod b^*(x)) \times (c_{2d-2} + \cdots + c_0 t^{2d-2}).$$

Fiduccia's algorithm [14] relies on Lemma 9 and computes $t^N \bmod b^*(t)$ by binary powering. As a consequence, the Nth term of a linear recurrent sequence of order d is computed in $\tilde{O}(d \log N)$ operations in R. In our setting, we use Fiduccia's algorithm for $R = \mathbb{F}_p(x)$.

THEOREM 10. *Let b be in $\mathbb{F}_p[x, y]_{d_x, d_y}$ and let $\delta \in \mathbb{Z}$. Algorithm 4 computes the δ-diagonal $\pi_\delta(x) = \sum_i B_{i,i+\delta} x^i$ of $B = b^{p-1}$ in $\tilde{O}(d_x(d_x + d_y)^3 p)$ operations in $\mathbb{F}_p$.*

PROOF. By (14), to compute π_δ it is sufficient to have the coefficients $c_{\delta-pv}$ for $-d_x \le v \le d_y$. The coefficient c_N has the form $p(x)/q(x)^{N+1}$, where $\deg p = O(Nd_x)$, $\deg q \le d_x$. The most expensive step in Fiduccia's algorithm for c_N is computing t^N modulo a polynomial in $\mathbb{F}_p[x,y]_{d_x,d_x+d_y}$. By Lemma 6, this can be done in time $T_N = \tilde{O}(Nd_x(d_x+d_y))$. Summing the costs T_N over all N's of the form $\delta - pv$ with $-d_x \le v \le d_y$ concludes the proof. □

Algorithm **PartialPowering**

Input A polynomial b in $\mathbb{F}_p[x,y]$ and $\delta \in \mathbb{Z}$.
Output The δ-diagonal of $B = b^{p-1}$, that is $\sum_i B_{i,i+\delta}x^i$.

for v from $\mathrm{val}_t\, b(x/t,t)$ to $\deg_t b(x/t,t)$ **do**
  Compute $c_{\delta-pv}(x) = [t^{\delta-pv}]\frac{1}{b(x/t,t)}$ via Fiduccia's algo
return $\pi_\delta(x) = \sum_{u+pv=\delta} c_u(x)b_v(x^p)$

Algorithm 4. δ-diagonal of a bivariate polynomial power.

4.3 Faster precomputation

We finally combine the results in Sections 4.1 and 4.2 with the ones of Sec. 3, and modify Algorithm 3 accordingly, by replacing its Step 3 with the partial powering routine (Algorithm 4). This yields our main result.

THEOREM 11. *Let E be in $\mathbb{F}_p[x,y]_{h,d}$ satisfy $E(0,0) = 0$ and $E_y(0,0) \neq 0$, and let $f \in \mathbb{F}_p[[x]]$ be its unique root with $f(0) = 0$. One can compute the coefficient f_N of f in $O(h^2(d+h)^2 \log N) + \tilde{O}(h(d+h)^5 p)$ operations in $\mathbb{F}_p$.*

PROOF. Only the cost of the precomputation changes. Instead of computing the whole power $B = b^{p-1}$, we only compute its δ-diagonals $\pi_\delta(x)$, for $\delta \in \Delta'$. Since $|\Delta'| = |\Delta| \le (d_x+d_y+1)^2$, we conclude using Theorem 10. □

EXAMPLE 9 (CATALAN). Let $E(x,y) = y - x - y^2$ be in $\mathbb{F}_p[x,y]$, with root $f = \sum_{n\ge 0} C_n x^{n+1}$, where C_n is $\frac{1}{n+1}\binom{2n}{n}$. Then f is the diagonal of a/b with $a = y(1-2y)$ and $b = 1-x-y$. Thus $d_x = 1, d_y = 2$, and the linear representation has size 6. The needed information to compute the 6×6 matrices $A_0, \ldots, A_{p-1}$ consists in the δ-diagonals of $B = b^{p-1}$ for $\delta \in \Delta := ([-2,1] + p\mathbb{Z}) \cap [1-p, p-1] = \{1-p, -2, -1, 0, 1, p-2, p-1\}$. Writing $B(x/t,t) = (1-x/t-t)^{p-1} = \sum_{\delta=1-p}^{p-1} \pi_\delta(x)t^\delta$, the goal is to compute the polynomials $\pi_\delta(x)$ for $\delta \in \Delta$. This is done via the formula $B(x/t,t) = (1-x^p/t^p - t^p)/(1-x/t-t)$ which gives $\pi_\delta(x) = -c_{\delta+p}(x)x^p + c_\delta(x) - c_{\delta-p}(x)$. Therefore, it is sufficient to compute the rational functions $c_\delta(x)$ for all $\delta \in \Delta' = \Delta + p$. This is done using Fiduccia's algorithm, which exploits the fact that the c_δ's satisfy the recurrence $xc_{k+2}(x) = c_{k+1}(x) - c_k(x)$.

For instance, $\pi_0(x) = -x^p c_p(x)$, where $c_p(x)$ is obtained as the coefficient of t^2 in the product $(-1/x^2 - t/x)\cdot(t^p \bmod -xt^2+t-1)$. This is done by binary powering. Checking that the algorithm correctly computes $\pi_0(x)$ amounts to checking the amusing identity:

$$\left[t^1\right]\ \left(t^p \bmod -xt^2+t-1\right) = \sum_{\ell=0}^{p-1}\binom{2\ell}{\ell}x^{\ell+1-p} \quad \text{in } \mathbb{F}_p[x].$$

4.4 Practical issues and future work

We have implemented the algorithms described in this section in Maple. Our prototype is available at the url
http://specfun.inria.fr/dumas/Research/AlgModp/.
The implementation permits getting an idea of the feasibility of the approach using diagonals. For instance, in the case of the quartic equation in Ex. 6, with $p = 9001$ instead of $p = 11$, the algorithm spends 142 seconds on the precomputation, then computes the N-th coefficient for $N = 10^{10,000}$, resp. $10^{100,000}$, resp. $10^{1,000,000}$ in: 3, resp. 25, resp. 344 seconds. (Timings on a personal laptop, Intel Core i5, 2.8 GHz, 3MB.)

In a forthcoming article, we plan to extend our algorithms to all finite fields, and to improve the complexity results.

5. REFERENCES

[1] J.-P. Allouche and J. Shallit. The ring of k-regular sequences. *Theoret. Comput. Sci.*, 98(2):163–197, 1992.

[2] J.-P. Allouche and J. Shallit. *Automatic sequences*. Cambridge University Press, 2003. Theory, applications, generalizations.

[3] A. Bostan, F. Chyzak, G. Lecerf, B. Salvy, and É. Schost. Differential equations for algebraic functions. In *ISSAC'07*, pages 25–32. ACM Press, 2007.

[4] A. Bostan, P. Gaudry, and É. Schost. Linear recurrences with polynomial coefficients and application to integer factorization and Cartier-Manin operator. *SIAM Journal on Computing*, 36(6):1777–1806, 2007.

[5] D. G. Cantor and E. Kaltofen. On fast multiplication of polynomials over arbitrary algebras. *Acta Inform.*, 28(7):693–701, 1991.

[6] G. Christol. Éléments algébriques. In *Groupe de travail d'Analyse Ultra-métrique (1ère année : 1973/74)*, number 14, 10 pp. Secrétariat Mathématique, Paris, 1974.

[7] G. Christol. Ensembles presque periodiques k-reconnaissables. *Theoret. Comput. Sci.*, 9(1):141–145, 1979.

[8] G. Christol, T. Kamae, M. Mendès France, and G. Rauzy. Suites algébriques, automates et substitutions. *Bull. Soc. Math. France*, 108(4):401–419, 1980.

[9] D. V. Chudnovsky and G. V. Chudnovsky. On expansion of algebraic functions in power and Puiseux series. I. *J. Complexity*, 2(4):271–294, 1986.

[10] D. V. Chudnovsky and G. V. Chudnovsky. Approximations and complex multiplication according to Ramanujan. In *Ramanujan revisited (Urbana-Champaign, 1987)*, pages 375–472. 1988.

[11] D. V. Chudnovsky and G. V. Chudnovsky. Computer algebra in the service of mathematical physics and number theory. In *Computers in mathematics (Stanford, 1986)*, volume 125 of *Lecture Notes in Pure and Appl. Math.*, pages 109–232. 1990.

[12] L. Comtet. Calcul pratique des coefficients de Taylor d'une fonction algébrique. *Enseignemen Math.*, 10:267–270, 1964.

[13] P. Dumas. *Récurrences mahlériennes, suites automatiques, études asymptotiques*. PhD thesis, Univ. Bordeaux I, 1993. Available at https://tel.archives-ouvertes.fr/tel-00614660.

[14] C. M. Fiduccia. An efficient formula for linear recurrences. *SIAM J. Comput.*, 14(1):106–112, 1985.

[15] H. Furstenberg. Algebraic functions over finite fields. *J. Algebra*, 7:271–277, 1967.

[16] F. L. Gall. Powers of tensors and fast matrix multiplication. In *ISSAC'14*, pages 296–303, 2014.

[17] J. von zur Gathen and J. Gerhard. *Modern computer algebra*. Cambridge University Press, third edition, 2013.

[18] D. Harvey, J. van der Hoeven, and G. Lecerf. Faster polynomial multiplication over finite fields. http://arxiv.org/abs/1407.3361, 2014.

[19] H. T. Kung and J. F. Traub. All algebraic functions can be computed fast. *J. Assoc. Comput. Mach.*, 25(2):245–260, 1978.

[20] J. C. P. Miller and D. J. Spencer Brown. An algorithm for evaluation of remote terms in a linear recurrence sequence. *Computer Journal*, 9:188–190, 1966.

[21] K. Nishioka. *Mahler Functions and Transcendence*. Number 1631 in Lecture Notes in Mathematics. Springer, Berlin, 1996.

[22] E. Rowland and R. Yassawi. Automatic congruences for diagonals of rational functions. *J. Théor. Nombres Bordeaux*, 27(1):245–288, 2015.

[23] A. Schönhage. Schnelle Multiplikation von Polynomen über Körpern der Charakteristik 2. *Acta Informat.*, 7:395–398, 1977.

[24] A. Storjohann. High-order lifting and integrality certification. *J. Symb. Comp.*, 36(3-4):613–648, 2003.