Computing Characteristic Polynomials of p-Curvatures in Average Polynomial Time

Raphaël Pagès

Computing Characteristic Polynomials of p-Curvatures in Average Polynomial Time

Raphaël Pagès
IMB, Université de Bordeaux, France
raphael.pages@u-bordeaux.fr

ABSTRACT

We design a fast algorithm that computes, for a given linear differential operator with coefficients in $\mathbb{Z}[x]$, all the characteristic polynomials of its p-curvatures, for all primes $p < N$, in asymptotically quasi-linear bit complexity in N. We discuss implementations and applications of our algorithm. We shall see in particular that the good performances of our algorithm are quickly visible.

CCS CONCEPTS

• **Computing methodologies** → **Algebraic algorithms**.

KEYWORDS

Algorithms, complexity, p-curvature, matrix factorial.

ACM Reference Format:
Raphaël Pagès. 2021. Computing Characteristic Polynomials of p-Curvatures in Average Polynomial Time. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The study of differential equations is a large part of mathematics which finds applications in many fields, particularly in physical sciences. Although the classical study of differential equations concerns essentially functions of real or complex variables, those equations can also be studied in an algebraic way. The functions in calculus get replaced by the elements of a so-called *differential ring*, and the "set of differential equations" is endowed with a ring structure. The resulting formalism is more flexible than that of calculus and makes it possible to study problems in positive characteristic.

In the algebraic context, the most relevant questions about a linear differential system $Y' = AY$, with A a matrix with coefficients in $\mathbb{Q}(x)$, differ a little from those in calculus. For example we may ask ourselves if such a system has an algebraic basis of solutions. This problem is especially difficult, though decidable, as was shown by Singer in [Sin80] (see also [BCDVW16]).

However, such a system can be reduced modulo p for any prime p not dividing the denominators of the matrix. Thus we can consider reductions modulo p of a given linear differential system. This construction turns out to be useful. Indeed if a system has an algebraic basis of solutions in characteristic 0, then its reduction modulo p also has one for almost all primes p. The well-known Grothendieck-Katz conjecture [Kat82] states that this is in fact an equivalence.

Thus it is very interesting for a given linear differential system in characteristic 0 to be able to determine if their reduction modulo p have a basis of algebraic solutions (or, more generally, to determine the dimension of its space of algebraic solutions) for a large amount of primes p, even if this only has heuristic applications for the time being. However, effective versions of the Grothedieck-Katz conjecture would turn this heuristic into a complete algorithm.

The resolution of this problem in positive characteristic is much easier than in characteristic 0 thanks to an invariant of linear differential systems in characteristic p: the p-curvature. This invariant is a linear map, whose kernel has the same dimension as the space of algebraic solutions of $Y' = AY$. Moreover, it is "easily computable", as its matrix is the p-th matrix A_p of the recursive sequence

$$A_1 = -A \quad \text{and} \quad A_{i+1} = A_i' - A \cdot A_i \ \text{ for } \ i \geqslant 1. \tag{1}$$

In this paper we are interested in computing the characteristic polynomials of the p-curvatures of a linear differential operator with coefficients in $\mathbb{Z}[x]$ for a whole range of primes $p < N$. This information contains an upper bound on the dimension of the kernel of the p-curvatures. It also enables us to tell whether the p-curvatures are nilpotent. This is interesting since Chudnovsky's theorem, of which a formulation can be found in [DGS94, Section VIII.1, Theorem 1.5], states that the minimal operator making a G-function vanish is globally nilpotent. As being globally nilpotent is quite an uncommon property, this provides a robust heuristic test when trying to post-certify a guessed annihilating differential operator.

The naive approach to this problem consists in computing the p-curvature with the recursive sequence (1) and then computing its characteristic polynomial. This strategy is sometimes referred to as *Katz's algorithm* [vdPS03, p. 324] and outputs the result in $\tilde{O}(p^2)$ bit operations (in this paper the notation $\tilde{O}$ will have the same meaning as O except we neglect logarithmic factors). Bostan, Caruso and Schost [BCS14] brought back the computation of the characteristic polynomial of the p-curvature to that of a factorial of matrices, and presented an algorithm finishing in $\tilde{O}(\sqrt{p})$ bit operations. It is unknown if the $1/2$ exponent is optimal for this problem. Indeed, the characteristic polynomial of the p-curvature is a polynomial P of degree $O(1)$ in x^p, and it is still unknown whether P is computable in polynomial time in $\log(p)$.

In this paper, we build upon [BCS14] to design an algorithm computing, for a given differential operator, almost all of the characteristic polynomials of its p-curvatures, for all primes $p < N$, in quasi-linear, thus quasi-optimal, time in N. This is a significant

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

improvement over previous algorithms for the given task, since the iterations of *Katz's algorithm* and of the algorithm from [BCS14] only terminate in respectively $\tilde{O}(N^3)$ and $\tilde{O}(N^{3/2})$ bit operations.

Since the number of primes smaller than N is also quasi-linear in N, this means that the average time spent on the computation of one characteristic polynomial is polynomial in $\log(N)$. It is important to note that "average" here is meant as average over the range of primes, and definitely not over the set of operators (even of fixed degree and order).

To achieve this goal, we reuse an idea of Costa, Gerbicz and Harvey, who designed an algorithm computing $(p-1)! \bmod p^2$ for all primes p less than N in quasi-linear time in N [CGH14]. This algorithm was originally designed to search for the so-called *Wilson primes*, but it soon found many applications, for instance in counting points on curves [Har14].

We begin this article with a quick reminder of the theoretical facts about differential operators which make our algorithm possible. We then present our algorithm and evaluate its complexity to see that it is indeed quasi-linear in N. Lastly we present the results of our implementation of the algorithm in the computer algebra software *SageMath*.

Acknowledgements. This work was supported by DeRerumNatura ANR-19-CE40-0018 and CLap–CLap ANR-18-CE40-0026-01. I address special thanks to my PhD thesis advisors, Alin Bostan and Xavier Caruso who helped me during the preparation of this article, whose roots are in my Master's thesis [Pag20]. I also warmly thank the reviewers for their relevant and numerous comments and the amazing amount of work they put on this paper.

2 DIFFERENTIAL OPERATORS

In this section, we outline the theoretical aspects necessary to our algorithm by following the exposition of [BCS14] (to which we refer for more detailed explanations) and extending the results of *loc. cit.* to characteristic 0. All results in Sections 2.1 and 2.2 come from [BCS14]. Besides, proofs were added when they were not given in *loc. cit.*

Let $\mathcal{R}$ be either $R[x]$ or $R(x)$, with $R = \mathbb{Z}$ or $\mathbb{F}_p$, equipped with their usual derivation $f \mapsto f'$. Throughout this article we will study the ring of differential operators with coefficients in $\mathcal{R}$, which we denote by $\mathcal{R}\langle\partial\rangle$. The elements of $\mathcal{R}\langle\partial\rangle$ are polynomials in ∂ of the form

$$f_n\partial^n + f_{n-1}\partial^{n-1} + \cdots + f_1\partial + f_0$$

with $f_i \in \mathcal{R}$. The (noncommutative) multiplication in this ring is deduced from the Leibniz rule $\partial f = f\partial + f'$ for all elements f of $\mathcal{R}$.

2.1 Euler and integration operators

In Sections 2.1 and 2.2 we will only consider the case $R = \mathbb{F}_p$. We study the Euler operator $x\partial$. One can show that

$$\partial \cdot (x\partial) = (x\partial + 1) \cdot \partial \quad \text{and} \quad x \cdot (x\partial) = (x\partial - 1) \cdot x.$$

We introduce a new variable θ and consider the noncommutative ring $\mathbb{F}_p[\theta]\langle\partial\rangle$ (resp. $\mathbb{F}_p(\theta)\langle\partial\rangle$) whose elements are polynomials in the variable ∂ with coefficients in $\mathbb{F}_p[\theta]$ (resp. $\mathbb{F}_p(\theta)$), with multiplication deduced from the rule $\partial\theta = (\theta + 1)\partial$.

We now want to rewrite operators in the variable x as operators in the variable θ with the association $\theta \mapsto x\partial$. In order to do this, we introduce the integration operator ∂^{-1} and the algebras $\mathbb{F}_p[x]\langle\partial^{\pm 1}\rangle$ (resp. $\mathbb{F}_p(x)\langle\partial^{\pm 1}\rangle$) of Laurent polynomials in the variable ∂ with coefficients in $\mathbb{F}_p[x]$ (resp. $\mathbb{F}_p(x)$). The same can be done in the variable θ.

Proposition 2.1 ([BCS14, Section 2]). *The rings $\mathbb{F}_p[x]\langle\partial^{\pm 1}\rangle \subset \mathbb{F}_p(x)\langle\partial^{\pm 1}\rangle$ (resp. $\mathbb{F}_p[\theta]\langle\partial^{\pm 1}\rangle \subset \mathbb{F}_p(\theta)\langle\partial^{\pm 1}\rangle$) of Laurent polynomials in the variable ∂ are all well defined. Furthermore, the multiplication satisfies $\partial^{-1}f = \sum_{i=0}^{p-1}(-1)^i f^{(i)}\partial^{-i-1}$ for all $f \in \mathbb{F}_p(x)$, and $\partial^i g(\theta) = g(\theta + i)\partial^i$ for all $g \in \mathbb{F}_p(\theta)$ and $i \in \mathbb{Z}$.*

Proof. One can show that ∂^p is central in $\mathbb{F}_p(\theta)\langle\partial\rangle$. This is also the case in $\mathbb{F}_p(x)\langle\partial\rangle$ since $f^{(p)} = 0$ for all $f \in \mathbb{F}_p(x)$, and thus $\partial^p f = \sum_{i=0}^{p}\binom{p}{i}f^{(i)}\partial^{p-i} = f\partial^p + f^{(p)} = f\partial^p$.

It follows that we only need to invert the central element ∂^p of both sets of rings, which can be done the same way as commutative localization.

The first relation comes from the fact that $\partial^{-1}f = \partial^{p-1}f\partial^{-p}$ and $\binom{p-1}{i} \equiv (-1)^i \mod p$ and the second one is trivial. □

Theorem 2.2 ([BCS14, Section 2.2]). *The following induces an isomorphism of $\mathbb{F}_p$-algebras:*

$$\begin{array}{rccc} & \mathbb{F}_p[x]\langle\partial^{\pm 1}\rangle & \tilde{\leftrightarrow} & \mathbb{F}_p[\theta]\langle\partial^{\pm 1}\rangle \\ \varphi_p : & x & \mapsto & \theta\partial^{-1} \\ & x\partial & \leftarrow\!\shortmid & \theta \qquad : \psi_p \\ & \partial & \leftrightarrow & \partial \end{array}$$

Proof. It is enough to check that $\varphi_p(\partial)\varphi_p(x) = \varphi_p(x)\varphi_p(\partial) + 1$ and $\psi_p(\partial)\psi_p(\theta) = (\psi_p(\theta) + 1)\psi_p(\partial)$ to see that φ_p and ψ_p are well defined. We check that ψ_p and φ_p are invertible by checking that $\psi_p \circ \varphi_p$ (resp. $\varphi_p \circ \psi_p$) is the only morphism mapping x to x (resp. θ to θ) and ∂ to ∂. □

Remark 2.3 ([BCS14, Section 2.2]). *The element $(x+1)\partial$ is invertible in $\mathbb{F}_p(x)\langle\partial^{\pm 1}\rangle$ but $\varphi_p((x+1)\partial) = \theta + \partial$ is not invertible in $\mathbb{F}_p(\theta)\langle\partial^{\pm 1}\rangle$. As such, φ_p does not extend to an isomorphism*

$$\mathbb{F}_p(x)\langle\partial^{\pm 1}\rangle \to \mathbb{F}_p(\theta)\langle\partial^{\pm 1}\rangle.$$

One can show that $\mathbb{F}_p[\theta^p - \theta]\langle\partial^{\pm p}\rangle$ is the center of $\mathbb{F}_p[\theta]\langle\partial^{\pm 1}\rangle$ and that $\varphi_p^{-1}(\theta^p - \theta) = x^p\partial^p$. This will be useful later on.

2.2 Operators and p-curvature

We recall that for $L \in \mathbb{F}_p(x)\langle\partial\rangle$, the left multiplication by the operator ∂^p defines an $\mathbb{F}_p(x)$-linear endomorphism of $\mathbb{F}_p(x)\langle\partial\rangle/\mathbb{F}_p(x)\langle\partial\rangle L$ since ∂^p is a central element. We define the p-curvature of L as being this $\mathbb{F}_p(x)$-linear endomorphism or, for computational purposes, its matrix in the canonical basis $(1, \partial, \partial^2, \ldots)$, which we denote by $A_p(L)$.

Remark 2.4. *It follows from the definition that the p-curvature of a differential operator L does not change if L is multiplied on the left by an element of $\mathbb{F}_p(x)$. Though Algorithm 3 presented in Section 3.4 will work for operators in $\mathbb{Z}[x]\langle\partial\rangle$ for convenience, this remark allows us to say that it in fact works for all operators in $\mathbb{Q}(x)\langle\partial\rangle$.*

As we did for operators with coefficients in $\mathbb{F}_p(x)$, we define the p-curvature of an operator L with coefficients in $\mathbb{F}_p(\theta)$ as the $\mathbb{F}_p(\theta)$-linear endomorphism of $\mathbb{F}_p(\theta)\langle\partial\rangle/\mathbb{F}_p(\theta)\langle\partial\rangle\cdot L$ induced by the left multiplication by ∂^p, and we denote by $B_p(L)$ its matrix in the canonical basis $(1, \partial, \partial^2, \ldots)$. By [BCS14, Lemma 2.3] which is proved by a straightforward computation, if $B(L)(\theta)$ is the companion matrix of L then

$$B_p(L) = B(L)(\theta) \cdot B(L)(\theta+1) \cdots B(L)(\theta+p-1).$$

As we are interested in computing the characteristic polynomial of the p-curvature we introduce the following (*cf* [BCS14, Section 3]): Let $L_x \in \mathbb{F}_p(x)\langle\partial\rangle$, and $L_\theta \in \mathbb{F}_p(\theta)\langle\partial\rangle$. We denote their respective leading coefficients by $l_x \in \mathbb{F}_p(x)$ and $l_\theta \in \mathbb{F}_p(\theta)$ respectively and define two new operators:

$$\Xi_{x,\partial}(L_x) := l_x^p \chi(A_p(L_x))(\partial^p)$$

$$\Xi_{\theta,\partial}(L_\theta) := \left(\prod_{i=0}^{p-1} l_\theta(\theta+i)\right) \chi(B_p(L_\theta))(\partial^p)$$

where $\chi(M)$, for a square matrix M, is its characteristic polynomial.

Remark 2.5. *Depending on the context, we may write $\Xi_{x,\partial,p}$ and $\Xi_{\theta,\partial,p}$ if we want to specify the characteristic.*

Proposition 2.6 ([BCS14, Section 3.1]). *The maps $\Xi_{x,\partial}$ and $\Xi_{\theta,\partial}$ are multiplicative and can thus be extended to maps on $\mathbb{F}_p(x)\langle\partial^{\pm 1}\rangle$ and $\mathbb{F}_p(\theta)\langle\partial^{\pm 1}\rangle$ respectively.*

Proof. Let $D := \mathbb{F}_p(x)\langle\partial\rangle$ (resp. $D := \mathbb{F}_p(\theta)\langle\partial\rangle$) and $L_1, L_2 \in D$. The right multiplication by L_2 induces a map $\zeta_1 : D/DL_1 \to D/DL_1L_2$. There is also a canonical map $\zeta_2 : D/DL_1L_2 \to D/DL_2$. We check that

$$0 \to D/DL_1 \xrightarrow{\zeta_1} D/DL_1L_2 \xrightarrow{\zeta_2} D/DL_2 \to 0$$

is an exact sequence. Furthermore the left multiplication by ∂^p induces an endomorphism of this exact sequence. It follows that in a suitable basis, the matrix of the p-curvature of L_1L_2 is an upper triangular block matrix, with the upper left block being the matrix of the p-curvature of L_1 and the bottom right block, that of L_2. The multiplicativity immediately follows. We extend those applications by setting $\Xi_{x,\partial}(L\partial^{-n}) = \Xi_{x,\partial}(L)\Xi_{x,\partial}(\partial)^{-n}$ (resp. $\Xi_{\theta,\partial}$) for all n and all operators L. □

Theorem 2.7 ([BCS14, Section 3]).

- *The map $\Xi_{x,\partial}$ (resp. $\Xi_{\theta,\partial}$) takes its values in $\mathbb{F}_p(x^p)[\partial^{\pm p}]$ (resp. $\mathbb{F}_p(\theta^p-\theta)[\partial^{\pm p}]$).*
- *Those two maps send an operator with polynomial coefficients to an operator with polynomial coefficients.*
- *The following diagram commutes:*

$$\begin{array}{ccc} \mathbb{F}_p[x]\langle\partial^{\pm 1}\rangle & \xrightarrow{\varphi_p} & \mathbb{F}_p[\theta]\langle\partial^{\pm 1}\rangle \\ \downarrow{\scriptstyle \Xi_{x,\partial}} & & \downarrow{\scriptstyle \Xi_{\theta,\partial}} \\ \mathbb{F}_p[x^p][\partial^{\pm p}] & \xrightarrow{\varphi_p} & \mathbb{F}_p[\theta^p-\theta][\partial^{\pm p}] \end{array}$$

This is the main result that makes our algorithm possible. Theorem 2.7 is interesting since it brings back the computation of the characteristic polynomial of the p-curvature to that of the "factorial of matrices" $B_p(L)$, and can thus be computed using factorial computation methods.

2.3 Extension to integral coefficients

Although the p-curvature is defined for operators of $\mathbb{F}_p(x)\langle\partial\rangle$, we can define the p-curvature of an element of $\mathbb{Z}[x]\langle\partial\rangle$, since the canonical morphism $\mathbb{Z} \to \mathbb{F}_p$ induces a ring homomorphism

$$\mathbb{Z}[x]\langle\partial\rangle \to \mathbb{F}_p[x]\langle\partial\rangle.$$

Our goal is to compute, for a differential operator with coefficients in $\mathbb{Z}[x]$, the characteristic polynomials of its p-curvatures, for nearly all primes p up to a certain integer N, in $\tilde{O}(N)$ bit operations.

Proposition 2.8. *The rings $\mathbb{Z}[x]\langle\partial^{\pm 1}\rangle$ and $\mathbb{Z}[\theta]\langle\partial^{\pm 1}\rangle$ (analogous to those of Section 2.1) are well defined and we have an isomorphism $\varphi : \mathbb{Z}[x]\langle\partial^{\pm 1}\rangle \xrightarrow{\sim} \mathbb{Z}[\theta]\langle\partial^{\pm 1}\rangle$ defined in a similar manner to φ_p (see Theorem 2.2).*

Proof. It is enough to check that the multiplicative part $S = \{\partial^n | n \in \mathbb{N}\}$ is a right denominator set of the ring $\mathbb{Z}[\theta]\langle\partial\rangle$ (see [Lam99, Section 10A]). Since this ring has no nontrivial zero divisor, we only have to check that S is right permutable, that is to say that

$$\forall g \in \mathbb{Z}[\theta]\langle\partial\rangle, \forall n \in \mathbb{N}, \exists g_1 \in \mathbb{Z}[\theta]\langle\partial\rangle, \exists n_1 \in \mathbb{N}, g\partial^{n_1} = \partial^n g_1.$$

This is the case since for all $n \in \mathbb{N}$ and all $g \in \mathbb{Z}[\theta]$, $\partial^n g(\theta-n) = g\partial^n$ and the fact that $\mathbb{Z}[\theta]\langle\partial^{\pm 1}\rangle$ is well defined follows by additivity.

The same can be done for operators with coefficients in the variable x. Let $f \in \mathbb{Z}[x]$ and suppose that $f^{(n_1)} = 0$. Then

$$f\partial^{n_1+1} = \partial \sum_{k=0}^{n_1-1} (-1)^k f^{(k)} \partial^{n_1-k}.$$

Now by induction on i, we show that for all $n_1 \in \mathbb{N}$, all $i \in \mathbb{N}^*$ and all $f \in \mathbb{Z}[x]$ such that $f^{(n_1)} = 0$, there exists $f_i \in \mathbb{Z}[x]\langle\partial\rangle$ such that $f\partial^{n_1+i} = \partial^i f_i$. We then conclude by additivity, which yields the fact that $\mathbb{Z}[x]\langle\partial^{\pm 1}\rangle$ is well defined. We show that φ is an isomorphism the same way we did for φ_p. □

By denoting $\pi_p : \mathbb{Z} \to \mathbb{F}_p$ the canonical reduction modulo p, we can easily see that $\pi_p \circ \varphi = \varphi_p \circ \pi_p$ (where we extend naturally π_p to suitable rings of operators). This enables us, for a given operator in $\mathbb{Z}[x]\langle\partial\rangle$, to compute the characteristic polynomials of its p-curvatures, by computing the isomorphism φ before the reduction modulo p. We will now see how to use this fact.

3 MAIN ALGORITHM

In this section, we present our algorithm and estimate its complexity. We denote by $2 \leqslant \omega \leqslant 3$ an exponent of matrix multiplication. From [AW21], we know that we can take $\omega < 2.3728596$. We will also have to address the cost of computing characteristic polynomials. Let us denote $\Omega_1 \in \mathbb{R}_+^*$ such that the computation of the characteristic polynomial of a square matrix of size m with coefficients in a ring R can be done in $\tilde{O}(m^{\Omega_1})$ arithmetic operations in R. From [KV05, Section 6], we know that it is theoretically possible to take $\Omega_1 \simeq 2.697263$. Finally, throughout this section, we assume that any two polynomials of degree d over a ring R (resp. integers of bit size n) can be multiplied in $\tilde{O}(d)$ operations in R (resp. $\tilde{O}(n)$ bit operations); FFT-like algorithms allow for these complexities [CK91, HvdH21]. We now give an outline of our algorithm.

Input: $L_x \in \mathbb{Z}[x]\langle\partial\rangle$, $N \in \mathbb{N}$
Output: A list of the characteristic polynomials of the p-curvatures of L_x, for all primes p with $p < N$ except a finite number not depending on N.

(1) Name l_x the leading coefficient of L_x.
(2) Compute $L_\theta := \varphi(L_x) \in \mathbb{Z}[\theta]\langle\partial^{\pm 1}\rangle$.
(3) Name l_θ the leading coefficient of L_θ.
(4) Compute $\mathcal{P}_{l_\theta}$, the list of all primes $p < N$ which do not divide l_θ.
(5) Construct $B(L_\theta)$.
(6) Compute $\left(\prod_{i=0}^{p-1} l_\theta(\theta+i)\right) \bmod p$ for all $p \in \mathcal{P}_{l_\theta}$.
(7) Compute $B(L)(\theta)\cdots B(L)(\theta+p-1) \bmod p$ for all $p \in \mathcal{P}_{l_\theta}$.
(8) Deduce all the $\Xi_{\theta,\partial,p}(L_\theta)$, for $p \in \mathcal{P}_{l_\theta}$.
(9) Deduce all $\chi(A_p(L_x)) = l_x^{-p}\varphi_p^{-1}(\Xi_{\theta,\partial,p}(L_\theta))$, for $p \in \mathcal{P}_{l_\theta}$.

Remark 3.1. *We only do the computation for the primes which do not divide the leading coefficient of L_θ because for those which do, the companion matrix of its reduction modulo p is not the reduction modulo p of its companion matrix.*

Lemma 3.2. *Let $L_\theta \in \mathbb{F}_p[\theta]\langle\partial\rangle$ be an operator with coefficients of degree at most $d \in \mathbb{N}$. Then $\Xi_{\theta,\partial}(L_\theta)$ has coefficients of degree at most dp.*

Proof. See [BCS14, Lemma 3.9]. □

From Lemma 3.2, we deduce that at the end of step (8) we have a list of (lists of) polynomials of degree linear in p, which means that the bit size of the output of this step is quadratic in N. This seems to remove all hope of ending up with a quasi-linear algorithm. Fortunately those polynomials lie in $\mathbb{F}_p[\theta^p - \theta]$(see Theorem 2.7). Thus each of them can be represented by data of bit size $O(d\log(p))$. We explain how in Section 3.1.

Remark 3.3. *This problem is also present at the end of step (9), but is easy to solve as we only need to determine the coefficients of x^i when i is a multiple of p. Thus we in fact compute polynomials $P_p \in \mathbb{F}_p[x,Y]$ such that $P_p(x^p, Y) = \chi(A_p(L))$ for all $p < N$.*

3.1 Reverse isomorphism, computation modulo θ^{d+1}

We know from Theorem 2.7 that for $L_\theta \in \mathbb{F}_p[\theta]\langle\partial\rangle$, the operator $\Xi_{\theta,\partial}(L_\theta)$ has coefficients in $\mathbb{F}_p[\theta^p - \theta]$.

Lemma 3.4. *Let $Q \in \mathbb{F}_p[\theta^p - \theta]$ be a polynomial of degree d in $\theta^p - \theta$ with $d < p$. Write:*

$$Q = \sum_{i=0}^{d} q_i(\theta^p - \theta)^i \quad \text{and} \quad Q = \sum_{i=0}^{dp} q'_i\theta^i.$$

For all $i \leqslant d$, we have $q_i = (-1)^i q'_i$.

Proof. This comes from the fact that $(-1)^i\theta^i$ is the only monomial of degree less than p in $(\theta^p - \theta)^i$. □

When p is strictly greater than d, it follows that we only need to compute the $\Xi_{\theta,\partial,p}$ modulo θ^{d+1} where d is the highest degree of the coefficients of the operator (in both variables x or θ), as one can see in Algorithm 1. We deduce the following lemma whose proof is obvious.

Algorithm 1 reverse_iso

Input: $Q_\theta \in \mathbb{F}_p[\theta^p - \theta][Y]$, of degree m in Y and degree at most dp in θ, known modulo θ^{d+1}.
Output: $Q_x \in \mathbb{F}_p[x,Y]$ such that $Q_x(x^p, \partial^p) = \varphi_p^{-1}(Q_\theta(\partial^p))$.

(1) $Q_x \leftarrow 0$.
(2) For all $i \leqslant m$:
  (a) Let $Q_{\theta,i}$ be the coefficient of ∂^i of Q_θ and write $Q_{\theta,i} = \sum_{j=0}^{d} q_{i,j}\theta^j + O(\theta^{d+1})$.
  (b) $Q_x \leftarrow Q_x + \sum_{j=0}^{d}(-1)^j q_{i,j}x^j Y^{i+j}$.
(3) *Return:* Q_x.

Lemma 3.5. *If $Q_\theta \in \mathbb{F}_p[\theta^p - \theta][Y]$ is of degree m in Y and dp in θ with $d < p$, then Algorithm 1 computes $Q_x \in \mathbb{F}_p[x,Y]$ such that $Q_x(x^p, \partial^p) = \varphi_p^{-1}(Q_\theta(\partial^p))$ in $O(dm\log(p))$ bit operations.*

Remark 3.6. *In fact we can still compute φ_p^{-1} if $p \leqslant d$ while only knowing the operator modulo θ^{d+1} but this is more tedious since there is no nice formula. In that case, with notation as in Lemma 3.4, we have $q'_i = \sum_{k=0}^{\lfloor i/(p-1)\rfloor}(-1)^{i-kp}\binom{i-k(p-1)}{k}q_{i-k(p-1)}$.*
This relation is easily invertible since it is given by a triangular matrix with no zero on the diagonal.

3.2 Translation before the computation

From the results of the previous subsection, we know that we only need to determine $\Xi_{\theta,\partial}$ modulo a small power of θ. Unfortunately, the companion matrix of an operator in $\mathbb{F}_p[\theta]\langle\partial\rangle$, even if the operator has polynomial coefficients, usually has its coefficient in $\mathbb{F}_p(\theta)$. In [BCS14], the authors solve this issue by injecting $\mathbb{F}_p(\theta)$ in $\mathbb{F}_p((\theta))$ and computing modulo a slightly higher power of θ. In order to minimize the degree of the polynomials used in the computation, we take a different approach based on the following proposition.

Proposition 3.7. *Let $a \in \mathbb{F}_p$. We denote by $\tau_a : \mathbb{F}_p[x] \to \mathbb{F}_p[x]$ the shift automorphism $Q \mapsto Q(x+a)$. This automorphism extends to automorphisms of $\mathbb{F}_p[x]\langle\partial\rangle$ and $\mathbb{F}_p[x,Y]$. Then*

$$\tau_a \circ \chi(A_p) = \chi(A_p) \circ \tau_a.$$

Proof. We know that $\tau_a(f)' = \tau_a(f')$ for all $f \in \mathbb{F}_p[x]$. We can thus extend τ_a to $\mathbb{F}_p[x]\langle\partial\rangle$. Now, since for any L, the operator $\tau_a(L)$ has the same order as L, we get that $A(\tau_a(L)) = \tau_a(A(L))$ (where $A(L)$ is the companion matrix of L). Now with the relation between τ_a and derivation we recursively extend that equality using (1) to get $\tau_a(A_p(L)) = A_p(\tau_a(L))$. Since τ_a is an endomorphism, the result follows. □

From Proposition 3.7, we deduce that we can shift an operator before computing the characteristic polynomials of its p-curvatures, and do the opposite translation on those to get the desired result. It is especially useful because of the following lemma.

Lemma 3.8. *Let $L_x \in \mathbb{Z}[x]\langle\partial\rangle$ be an operator and denote by $l_x \in \mathbb{Z}[x]$ its leading coefficient. If $l_x(0) \neq 0$ then $\varphi(L_x)$ has $l_x(0) \in \mathbb{Z}$ as its leading coefficient.*

Proof. A straightforward computation shows that $\varphi(x^i\partial^j) = p_i(\theta)\partial^{j-i}$ with $p_i(\theta)$ being a polynomial only dependent on i (and not on j). Thus the leading coefficient of $\varphi(L_x)$ can only come from the constant coefficient of l_x if this one is not 0. □

In our setting, the fact that $\varphi(L_x)$ has a constant leading coefficient means that its companion matrix (see §2.2) has its coefficients in $\mathbb{Q}[\theta]$, implying that we can do all the computations modulo θ^{d+1}. Lemma 3.8 shows that we can shift our starting operator by $a \in \mathbb{Z}$ where a is not a root of its leading coefficient to place ourselves in that setting.

Since translating back all the characteristic polynomials (the P_p in fact, see Remark 3.3) at the end of the computation is basically the same as translating a list of $O(Nm)$ univariate polynomials of degree d, it can be done in $\tilde{O}(Nmd)$ bit operations (for example with binary splitting), with m being the order of the operator and d the maximum degree of its coefficients.

3.3 Computing a matrix factorial modulo p for a large amount of primes p

Let $M(\theta) \in \mathscr{M}_m(\mathbb{Z}[\theta])$ be a square matrix of size m with coefficients of degree less than d. In this subsection we review the algorithm of [CGH14, Har14] applied to the computation of the following matrix factorial :

$$M(\theta)\cdot M(\theta+1)\cdots M(\theta+p-1) \mod (p,\theta^d)$$

for all primes $p < N$. Though very similar, the setting of [Har14] is slightly different from ours as it concerns only integer matrices and considers a different kind of products. For this reason, we prefer to take some time to restate the algorithm in full and, at the same time, take the opportunity to set up notations.

Since the method of [CGH14] computes products of $p-1$ entries modulo some power of p, we will compute $M(\theta+1)\cdots M(\theta+p-1) \mod (p,\theta^d)$ for all p, and then left-multiply by $M(\theta)$.

Let $\eta := \lceil \log_2(N) \rceil$. For all i and j with $0 \leqslant i \leqslant \eta$ and $0 \leqslant j < 2^i$, we denote $U_{i,j} := \left\{ k \in \mathbb{N} \;\middle|\; j\frac{N}{2^i} < k \leqslant (j+1)\frac{N}{2^i} \right\}$.

It follows from the definition that for all $0 \leqslant i < \eta$ and all $0 \leqslant j < 2^i$, $U_{i,j} = U_{i+1,2j} \cup U_{i+1,2j+1}$. Furthermore, for $i = \eta$, the $U_{i,j}$ are either empty or a singleton.

From this, we introduce $T_{i,j} := \prod_{k\in U_{i,j}} M(\theta+k) \mod \theta^d$, with the product being made by sorting elements of $U_{i,j}$ in ascending order, and $S_{i,j} := \prod_{\substack{p\in U_{i,j}\\ p \text{ prime}}} p$. From now on, we consider that the $T_{i,j}$ are elements of $\mathscr{M}_m(\mathbb{Z}[\theta]/\theta^d)$. From the properties of $U_{i,j}$, we deduce that $T_{i,j} = T_{i+1,2j}T_{i+1,2j+1}$ and $S_{i,j} = S_{i+1,2j}S_{i+1,2j+1}$.

These relations allow us to fill binary trees containing the $T_{i,j}$ and $S_{i,j}$ as their nodes from the bottom. Furthermore, filling those trees is nothing more than computing a factorial by binary splitting, and keeping the intermediate steps in memory.

To see how to apply this to our problem we suppose that $p \in U_{\eta,j}$ for a certain j. A direct computation gives:

$$\begin{aligned} &M(\theta+1)\cdot M(\theta+2)\cdots M(\theta+p-1) \mod (p,\theta^d)\\ &\quad = T_{\eta,0}T_{\eta,1}\cdots T_{\eta,j-1} \mod S_{\eta,j}. \end{aligned}$$

Algorithm 2 matrix_factorial

Input: $M(\theta) \in \mathscr{M}_m(\mathbb{Z}[\theta])$ with coefficients of degree less than d, $\mathcal{P}$ a list of primes smaller than N.
Output: A list containing $M(\theta)M(\theta+1)\cdots M(\theta+p-1) \mod (p,\theta^d)$ for all p in $\mathcal{P}$.

(1) $\eta \leftarrow \lceil \log_2(N) \rceil$.
(2) Fill $T_{\eta,\_}$ and $S_{\eta,\_}$.
(3) Compute the binary trees T and S.
(4) $W_{0,0} \leftarrow 1$.
(5) For i going from 0 to $\eta - 1$:
  (a) For j going from 0 to $2^i - 1$:
    (i) $W_{i+1,2j} \leftarrow W_{i,j} \mod S_{i+1,2j}$.
    (ii) $W_{i+1,2j+1} \leftarrow W_{i,j}T_{i+1,2j} \mod S_{i+1,2j+1}$.
(6) Construct $\prod$ the list of $W_{\eta,j}$ where $S_{\eta,j} \in \mathcal{P}$.
(7) Do the left multiplication by $M(\theta)$ on the elements of $\prod$.
(8) Return: $\prod$.

This motivates the following definition: for all i, j with $0 \leqslant i \leqslant \eta$ and $0 \leqslant j < 2^i$, we set $W_{i,j} := \prod_{k=0}^{j-1} T_{i,k} \mod S_{i,j}$. The following lemma is easily checked.

Lemma 3.9. *For all i and j such that the following quantities are well defined, $W_{i+1,2j} = W_{i,j} \mod S_{i+1,2j}$ and $W_{i+1,2j+1} = W_{i,j}T_{i+1,2j} \mod S_{i+1,2j+1}$.*

Thus we can compute the $W_{\eta,j}$ by filling a binary tree from the top starting from $W_{0,0} = 1$. This proves the correctness of Algorithm 2, while its complexity is addressed in the next proposition.

Proposition 3.10. *This algorithm has a cost of*

$$\tilde{O}\big(m^\omega dN(n + d\log(N) + \log(m))\big)$$

bit operations, where n is the maximum bit size of the integers in the matrix $M(\theta)$.

Proof. The computation of the binary tree S is less costly than that of T, so we do not consider it. Let us evaluate the complexity of the computation of T. We need to know the bit size of the integers at each level of T. We use the following lemma.

Lemma 3.11. *For any $a \leqslant N$, all the integers appearing in $M(\theta+a)$ have bit size at most $n + d(1 + \log_2(N))$.*

Proof. Let $Q \in \mathbb{Z}[\theta]$ of degree less than d appearing in $M(\theta)$. Then we can write

$$Q(\theta+a) = \sum_{j=0}^{d-1}\left(\sum_{i=j}^{d-1}\binom{i}{j}q_i a^{i-j}\right)\theta^j$$

where the q_i are the coefficients of Q. Moreover, we know that all the q_i are at most 2^n. Thus the coefficients of $Q(\theta+a)$ are less than $2^n N^{d-1}\sum_{i=j}^{d-1}\binom{i}{j} \leqslant 2^{n+d}N^d$. □

We now resume the proof of Proposition 3.10. If Δ_1 and Δ_2 are matrices in $\mathscr{M}_m(\mathbb{Z}[\theta]/\theta^d)$ with integers of bit size at most n_1, then $\Delta_1\Delta_2$ has integers of bit size at most $2n_1 + \log_2(dm)$. It follows that the integers in the matrices $A_{i,j}$ are of bit size at most:

$$\begin{aligned} &2^{\eta-i}(n + d(1+\log_2(N))) + (2^{\eta-i}-1)\log_2(dm)\\ &\quad = O(2^{\eta-i}(n + d\log_2(N) + \log_2(m))). \end{aligned}$$

The computation of T is reduced to the computation of its two sub-trees, followed by a multiplication of two square matrices of size m with polynomial coefficients of degree d and integers of bit size $O(2^{\eta-1}(n + d\log_2(N) + \log_2(m)))$. Since the bit size of the integers is halved at each level, we finally find, using that $2^\eta \leqslant 2N$, that the computation of T can be done in $\tilde{O}(m^\omega dN(n + d\log_2(N) + \log_2(m))$ bit operations.

The cost of computing W is the same as that of reducing $T_{i,j}$ mod $S_{i,j+1}$ whenever both quantities are well defined, and then of computing recursively the $W_{i,j}$ using only integers smaller than $S_{i,j}$. The first step can be done in $\tilde{O}(Nm^2d(n + d))$ bit operations, while the second requires $\tilde{O}(m^\omega dN)$ bit operations. □

3.4 Final algorithm

The most important pieces of our main algorithm are now in place, we are almost ready to write down its final version. Before doing this, we analyze the cost of converting an operator in $\mathbb{Z}[x]\langle\partial\rangle$ to its counterpart in $\mathbb{Z}[\theta]\langle\partial^{\pm1}\rangle$.

Proposition 3.12. *For any operator $L \in \mathbb{Z}[x]\langle\partial\rangle$, of order m with coefficients of degree at most d, with integer coefficients of bit size at most n, the computation of $\varphi(L)$, can be done in $\tilde{O}(d(m + d)(n + d))$ bit operations.*
Furthermore the resulting operator in the variable θ has its integer coefficients of bit size $O(n + d\log_2(d))$.

Proof. From [BCS14, Section 4.1] we get that this computation over a ring R can be done in $\tilde{O}((m + d)d)$ algebraic operations in R. Following their algorithm, we can show that, when $R = \mathbb{Z}$, intermediate computations do not produce integers larger than those of the final result. Moreover, if

$$\varphi\left(\sum_{\substack{0\leqslant i\leqslant d\\ 0\leqslant j\leqslant m}} l_{i,j}x^i\partial^j\right) = \sum_{\substack{0\leqslant i\leqslant d\\ -d\leqslant j\leqslant m}} l'_{i,j}\theta^i\partial^j$$

the estimation $|l_{i,j}| \leqslant 2^n$ implies $|l'_{i,j}| \leqslant 2^{n+d+1}d^d$. Putting all together, we get the announced result. □

Note that for an operator $L \in \mathbb{Z}[x]\langle\partial\rangle$ of order m with coefficients of degree at most d, $\varphi(L)$ has nonzero coefficients for powers of ∂ varying from $-d$ to m, making the square matrices used in Algorithm 3 of size at most $m + d$.

We now present the final algorithm in Algorithm 3.

Theorem 3.13. *For any operator $L \in \mathbb{Z}[x]\langle\partial\rangle$, Algorithm 3 computes a list of polynomials $P_p \in \mathbb{Q}[x, Y]$ for all primes $p < N$ except a finite number not depending on N, such that $P_p(x^p, Y) = \chi(A_p(L))$ in*

$$\tilde{O}\big(Nd((n + d)(m + d)^\omega + (m + d)^{\Omega_1})\big)$$

bit operations, where m is the order of the operator, d is the maximum degree of its coefficients and n is the maximum bit size of the integers appearing in L.

Proof. This is easily seen by summing the cost of each step of Algorithm 3. We observe that these complexities are correct whether or not 0 is a root of L_x. Indeed, when it is not, the new operator obtained after the translation of step (3) has integer coefficients of bit size $O(n + d\log(d))$, therefore our complexity analysis remains correct. □

Algorithm 3 charpoly_p_curv

Input: $L_x \in \mathbb{Z}[x]\langle\partial\rangle$ of order m, with coefficients of degree at most d and integer coefficients of bit size at most n, $N \in \mathbb{N}$.
Output: A list of polynomials $P_p \in \mathbb{F}_p[x, Y]$ such that $P_p(x^p, Y) = \chi(A_p(L))$ for all primes $p < N$, except a finite number not depending on N.

(1) $l_x \leftarrow$ the leading coefficient of L_x.
(2) $a \leftarrow 0$.
(3) If $l_x(0) = 0$ do:
  (a) Shift L_x by b with $b \in \mathbb{Z}$ not a root of l_x.
  (b) $a \leftarrow b$.
  Cost: $\tilde{O}(md(n + d))$ *bit operations.*
(4) Compute $L_\theta\partial^{-k} := \varphi(L_x)$ with `x_d_to_theta_d` from [BCS14, Section 4].
  Cost: $\tilde{O}((m + d)(n + d)d)$ *bit operations.*
(5) $d \leftarrow$ the maximum degree of the coefficients of L_θ.
(6) $l_\theta \leftarrow$ the leading coefficient of L_θ.
  It has been made to be an integer.
(7) Construct $M(\theta) = l_\theta \cdot B(L_\theta)$.
(8) Compute the list $\mathcal{P}$ of all primes p that do not divide l_θ with $d + 1 \leqslant p < N$.
  Cost: $\tilde{O}(N)$ *bit operations (see [CGH14, Proposition 2.1]).*
(9) Compute the list $\mathcal{L}$ of $M(\theta)\cdots M(\theta + p - 1)$ mod (θ^{d+1}, p) for all p in $\mathcal{P}$ using *matrix_factorial*.
  Cost: $\tilde{O}((m + d)^\omega(n + d)dN)$ *bit operations.*
(10) Divide all elements of $\mathcal{L}$ by l_θ.
  Cost: $O(N(m + d)^2d$ *bit operations.*
(11) Compute the list C of the characteristic polynomials of elements of $\mathcal{L}$.
  Cost: $\tilde{O}(N(m + d)^{\Omega_1}d)$ *bit operations.*
(12) Multiply the elements of C by l_θ.
  Cost: $\tilde{O}(N(m + d)d)$ *bit operations.*
(13) Compute the image by φ_p^{-1} of elements of C using *reverse_iso*.
  Cost: $\tilde{O}(Nd(m + d))$ *bit operations.*
(14) Divide the polynomials obtained by l_x and Y^{-k}.
  Cost: $\tilde{O}(Nmd)$ *bit operations.*
(15) If $a \neq 0$, shift the polynomials obtained by $-a$.
  Cost: $\tilde{O}(Nmd)$ *bit operations.*

As we have seen, Algorithm 3 does not compute the characteristic polynomial of the p-curvature for every $p < N$, as we have to remove all primes dividing $l_x(0)$, where l_x is the leading coefficient of the operator (provided of course that $l_x(0) \neq 0$). Primes less than the maximum degree of the coefficients of the operator are also not included; however, it is possible to remedy these with minor tweaks using Remark 3.6.

Proposition 3.14. *It is possible to compute all characteristic polynomials of the p-curvatures of an operator $L \in \mathbb{Z}[x]\langle\partial\rangle$ of order m and maximum degree of the coefficients d, for all primes p less than N, in asymptotically quasi-linear time in N.*

Proof. The computation for primes dividing $l_x(0)$ (with l_x being the leading coefficient of L) can be done using the main algorithm from [BCS14]. All other primes can be addressed using our new Algorithm 3.

As primes which cannot be computed using our algorithm only depend on the operator itself, the result immediately follows. □

4 IMPLEMENTATION AND TIMINGS

We have implemented Algorithm 3 in the Computer Algebra software *SageMath*. The source code can be downloaded from the following URL: `https://github.com/raphitek/p_curvatures`.

As mentioned earlier, the computation of the characteristic polynomial of a matrix of size m with coefficients in a ring can be performed in theory using $\tilde{O}(m^{\Omega_1})$ ring operations, with $\Omega_1 \simeq 2.697263$, see [KV05]. However, we did not implement the algorithm from [KV05], and instead used an algorithm computing a Hessenberg form of the matrix in $O(m^3)$ operations [CRV17]. Indeed, the latter algorithm is easier to implement and the computation of the characteristic polynomials is usually not the bottleneck and does not hinder the quasi-linear nature of our algorithm. Furthermore, experiments, as well as Theorem 3.13, showed that most of the running time is spent on the computation of trees T and W when the order of the operator is of the same magnitude as the degrees of its coefficients. We expect this trend to improve when the ratio of these two factors grows in favor of the order of the operator, but all experiments conducted so far showed that the computation of the characteristic polynomials is never the bottleneck by a wide margin. It is still more than six times faster on an operator of order 50 with coefficients of degree 2, for $N = 100$.

Remark 4.1. *In our experiments we do not consider cases where the degree d of the coefficients is higher than the order m of the operator because the complexity in d is worse than in m. As in [BBvdH12, Section IV], the general case reduces to this one using the transformation $x \mapsto -\partial, \partial \mapsto x$ which exchanges the roles of ∂ and x.*

4.1 Timings on random operators

Quasilinear as expected. Figure 1 shows computation timings of our implementation for operators in $\mathbb{Z}[x]\langle\partial\rangle$ of varying sizes on *SageMath* version 9.3.rc4 on an Intel(R) Core(TM) i3-40050 machine at 1.7Ghz, running ArchLinux. As expected, it does appear that our algorithm finishes in quasi-linear time in N. We can also see a floor phenomenon, with computation time varying very little between two powers of 2, and then doubling. This is an expected effect of the use of the complete binary tree structure in our algorithm. This effect however seems less visible, even if it is still perceptible, as the operator size increases. This is probably due to the fact that for operators of small sizes, the cost of manipulating empty nodes is non-negligible.

Comparison with the previous algorithm. We have compared the timings between our algorithm and the iteration of that of [BCS14] for an operator of order 3 and degree 2. Results are displayed on Figure 2 and show that the work presented in this paper is indeed a concrete progress for the considered task, compared to previous state of the art: experiments have shown that our algorithm was already more than twice as fast (on the same machine) than the algorithm of [BCS14] [*] for $N \sim 10^4$. The right part shows the ratio of computation times for operators of varying sizes. Results tend to

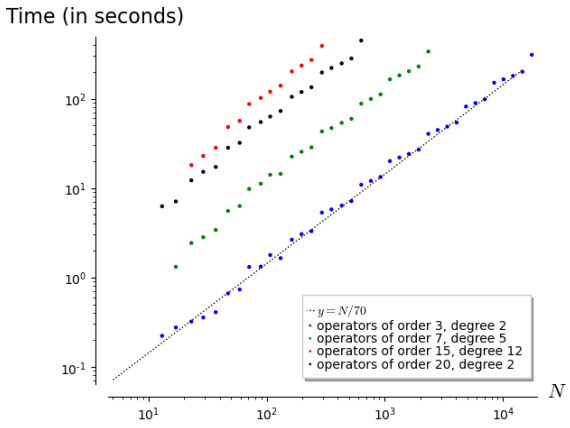


Figure 1: Computation time for random operators of varying orders and degrees

indicate that the good performances of our algorithm compared to the iteration of [BCS14] appear earlier when the order of the operator grows. Further experiments should be conducted to determine the influence of the degree of the coefficients.

4.2 Execution on special operators

Our algorithm was also tested on various "special" operators. One example is an operator proven in [BK10] to annihilate the generating function $G(t; 1, 0)$ of Gessel walks in the quarter plane ending on the horizontal axis. The result of this test indicates that this operator has a nilpotent p-curvature for all primes $p < 200$. This was of course expected since the generating function of Gessel walks is algebraic [BK10], hence the p-curvatures of its minimal-order differential operator are all zero. A similar test was performed on an operator proved in [BKV] to annihilate the generating function of Kreweras walks with interacting boundaries, which is not algebraic. Once again, the result of this test indicates that this operator has a nilpotent p-curvature for all primes $p < 200$[†]. Further testing was conducted on all the 76 operators for (specializations of) the D-finite generating functions for lattice walks classified in [BCvH[+]17] with $p < 200$, with yet again similar results[‡]. All those results were already predicted by Chudnovsky's theorem and make us quite confident in the accuracy of our implementation.

5 CONCLUSION AND FUTURE WORK

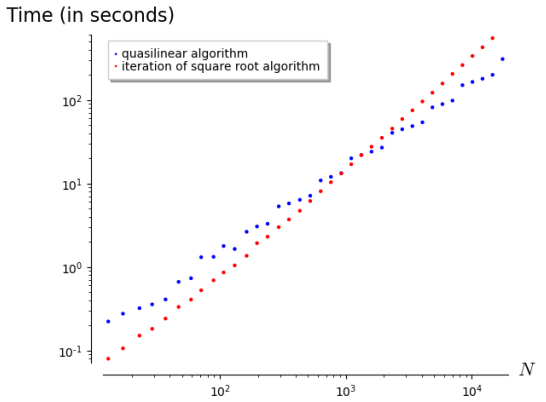
We have proposed an algorithm which computes the characteristic polynomials of the p-curvatures of a differential operator with coefficients in $\mathbb{Z}[x]$ for almost all primes $p < N$, in quasi-linear time in N.

We expect that the principle of this algorithm can theoretically be applied for differential operators with polynomial coefficients in any ring A by replacing $\mathbb{Z}/p\mathbb{Z}$ by A/pA. Especially we expect that this

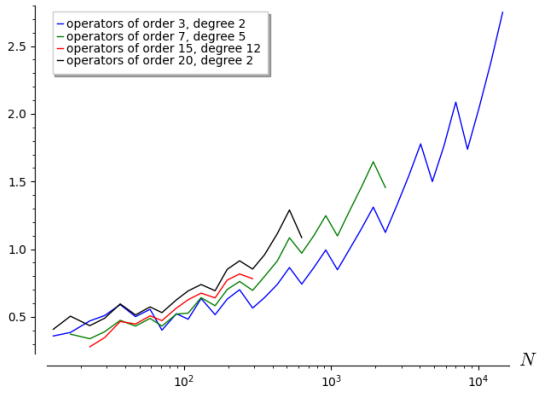
[*]The implementation of the algorithm from [BCS14] used can be found at `https://github.com/raphitek/p_curvatures/blob/main/p_curvature_single.sage`

[†]The program running the above mentioned tests can be found at `https://github.com/raphitek/p_curvatures/blob/main/test_p_curvature.sage`

[‡]The precise list of operators we considered can be found at `https://specfun.inria.fr/chyzak/ssw/ct-P.mpl` and the testing file can be found at `https://github.com/raphitek/p_curvatures/blob/main/ct-P.sage`



Computation time for operators of order 3 and degree 2



Ratio of computation times for operators of varying sizes

Figure 2: Comparison between the iteration of [BCS14]'s algorithm and our algorithm

algorithm extends nicely to operators with polynomial coefficients in the integer ring of a number field or with multivariate polynomial coefficients (which will allow us to deal with operators with parameters). In the latter case, we expect its time complexity in N to be in $\tilde{O}(N^s)$ where s is the number of variables. Furthermore, [BCS16] brought back the computation of the similarity class of the p-curvature of an operator in $K[x]\langle\partial\rangle$, with K a field of positive characteristic, to that of a matrix factorial. Thus we hope that the same principle can be applied to design an algorithm for computing the similarity classes of the p-curvatures of an operator in $\mathbb{Z}[x]\langle\partial\rangle$, for almost all primes $p < N$, in quasi-linear time in N.

This algorithm may also have applications to future works on factorisation of differential operators, as in [Clu03].

REFERENCES

[AW21] Josh Alman and Virginia Vassilevska Williams. *A Refined Laser Method and Faster Matrix Multiplication*, pages 522–539. SIAM, 2021.

[BBvdH12] Alexandre Benoit, Alin Bostan, and Joris van der Hoeven. Quasi-optimal multiplication of linear differential operators. In *FOCS 2012 - IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 524–530, New Brunswick, United States, October 2012. IEEE.

[BCDVW16] Moulay Barkatou, Thomas Cluzeau, Lucia Di Vizio, and Jacques-Arthur Weil. Computing the Lie algebra of the differential Galois group of a linear differential system. In *Proceedings of the 2016 ACM International Symposium on Symbolic and Algebraic Computation*, pages 63–70. ACM, New York, 2016.

[BCS14] Alin Bostan, Xavier Caruso, and Éric Schost. A fast algorithm for computing the characteristic polynomial of the p-curvature. In *ISSAC 2014—Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation*, pages 59–66. ACM, New York, 2014.

[BCS16] Alin Bostan, Xavier Caruso, and Éric Schost. Computation of the similarity class of the p-curvature. In *Proceedings of the 2016 ACM International Symposium on Symbolic and Algebraic Computation*, pages 111–118. ACM, New York, 2016.

[BCvH+17] Alin Bostan, Frédéric Chyzak, Mark van Hoeij, Manuel Kauers, and Lucien Pech. Hypergeometric expressions for generating functions of walks with small steps in the quarter plane. *European J. Combin.*, 61:242–275, 2017.

[BK10] Alin Bostan and Manuel Kauers. The complete generating function for Gessel walks is algebraic. *Proc. Amer. Math. Soc.*, 138(9):3063–3078, 2010. With an appendix by Mark van Hoeij.

[BKV] Alin Bostan, Manuel Kauers, and Thibaut Verron. The generating function of Kreweras walks with interacting boundaries is not algebraic. Proceedings of FPSAC'21, to appear.

[CGH14] Edgar Costa, Robert Gerbicz, and David Harvey. A search for Wilson primes. *Math. Comp.*, 83(290):3071–3091, 2014.

[CK91] David G. Cantor and Erich Kaltofen. On fast multiplication of polynomials over arbitrary algebras. *Acta Inform.*, 28(7):693–701, 1991.

[Clu03] Thomas Cluzeau. Factorization of differential systems in characteristic p. In *Proceedings of the 2003 International Symposium on Symbolic and Algebraic Computation*, pages 58–65. ACM, New York, 2003.

[CRV17] Xavier Caruso, David Roe, and Tristan Vaccon. Characteristic polynomials of p-adic matrices. In *ISSAC'17—Proceedings of the 2017 ACM International Symposium on Symbolic and Algebraic Computation*, pages 389–396. ACM, New York, 2017.

[DGS94] Bernard Dwork, Giovanni Gerotto, and Francis J. Sullivan. *An introduction to G-functions*, volume 133 of *Annals of Mathematics Studies*. Princeton University Press, Princeton, NJ, 1994.

[Har14] David Harvey. Counting points on hyperelliptic curves in average polynomial time. *Ann. of Math. (2)*, 179(2):783–803, 2014.

[HvdH21] David Harvey and Joris van der Hoeven. Integer multiplication in time $O(n \log n)$. *Ann. of Math. (2)*, 193(2):563–617, 2021.

[Kat82] Nicholas M. Katz. A conjecture in the arithmetic theory of differential equations. *Bull. Soc. Math. France*, 110(2):203–239, 1982.

[KV05] Erich Kaltofen and Gilles Villard. On the complexity of computing determinants. *Comput. Complex.*, 13(3–4):91–130, February 2005.

[Lam99] T. Y. Lam. *Lectures on modules and rings*, volume 189 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1999.

[Pag20] Raphaël Pagès. Étude d'algèbres d'opérateurs différentiels, techniques de calcul rapide de factorielles et applications au calcul de la p-courbure. Master's thesis, Univ. Paris 7, 2020. 91 pages.

[Sin80] Michael F. Singer. Algebraic solutions of nth order linear differential equations. In *Proceedings of the Queen's Number Theory Conference, 1979 (Kingston, Ont., 1979)*, volume 54 of *Queen's Papers in Pure and Appl. Math.*, pages 379–420. Queen's Univ., Kingston, Ont., 1980.

[vdPS03] Marius van der Put and Michael F. Singer. *Galois theory of linear differential equations*, volume 328 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer-Verlag, Berlin, 2003.