Desingularization and p-Curvature of Recurrence Operators*

Yi Zhou
Florida State University
Tallahassee, USA
yzhou@math.fsu.edu

Mark van Hoeij
Florida State University
Tallahassee, USA
hoeij@math.fsu.edu

ABSTRACT

Linear recurrence operators in characteristic p are classified by their p-curvature. For a recurrence operator L, denote by $\chi(L)$ the characteristic polynomial of its p-curvature. We can obtain information about the factorization of L by factoring $\chi(L)$. The main theorem of this paper gives an unexpected relation between $\chi(L)$ and the true singularities of L. An application is to speed up a fast algorithm for computing $\chi(L)$ by desingularizing L first. Another contribution of this paper is faster desingularization.

CCS CONCEPTS

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

KEYWORDS

linear recurrence equations, singularities, p-curvature, algorithms

ACM Reference Format:
Yi Zhou and Mark van Hoeij. 2022. Desingularization and p-Curvature of Recurrence Operators. In *Proceedings of the 2022 International Symposium on Symbolic and Algebraic Computation (ISSAC '22), July 4–7, 2022, Villeneuve-d'Ascq, France.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3476446.3535478

1 INTRODUCTION

Singularities of linear difference operators can be divided into two groups, true (i.e. non-removable) singularities, and removable singularities. Desingularization (detecting or removing removable singularities) can expedite various algorithms for difference or differential equations. An early application [14] appeared in `DEtools[Homomorphisms]` in Maple 10. Other algorithms that benefit from reducing the number of singularities include finding closed form solutions and factoring, e.g. `LREtools[RightFactors]` in Maple 2021.

In characteristic p, linear recurrence operators can be classified by the so-called p-curvature. For a recurrence operator L in characteristic p, denote by $\chi(L)$ the characteristic polynomial of its p-curvature. Our main result states that the denominator of $\chi(L)$ determines the true singularities of L, including their multiplicities, up to shift equivalence.

*Both authors were supported by NSF grants 1618657 and 2007959.

ISSAC '22, July 4–7, 2022, Villeneuve-d'Ascq, France

ACM ISBN 978-1-4503-8688-3/22/07…$15.00
https://doi.org/10.1145/3476446.3535478

The algorithm `Xi_theta_d` from [3, p. 7] computes $\chi(L)$, multiplied by a denominator bound, by computing its Z-adic expansion. One application of our theorem is that we can replace the denominator bound by the exact denominator. This lowers the required Z-adic precision, which can speed up the computation, see subsection 5.2.

We want desingularization to take less time than the time it saves in applications. Then it is useful to compute a *partial desingularization* (where the goal is to remove most removable singularities, at a fraction of the cost of a full desingularization). We give various algorithms for this in section 6.

2 PRELIMINARIES

2.1 Desingularization

Let F be a field. Let $P = F[x][y]$ and $D = F(x)[y]$. If $f \in P$, then f is called *primitive* if the gcd of its coefficients in $F[x]$ is 1. If $f \in D - \{0\}$, then there is $c \in F(x) - \{0\}$, unique up to a factor in F, for which $c^{-1}f \in P$ is primitive. The content of f, denoted $\mathrm{Cont}(f)$, is this c, while the *primitive part* of f is $\mathrm{Prim}(f) = c^{-1}f \in P$. A version of Gauss's lemma says $\mathrm{Cont}(f_1 f_2) = \mathrm{Cont}(f_1)\mathrm{Cont}(f_2)$.

Let τ be the shift-operator. If $r(x)$ is a rational function then the product $\tau \cdot r(x)$ equals $r(x+1)\cdot\tau$. This product and the usual addition turn $\mathcal{P} := F[x][\tau]$ and $\mathcal{D} := F(x)[\tau]$ into non-commutative rings. The product corresponds to compositions of operators, where $L = \sum_{i=0}^{n} a_i(x)\tau^i \in \mathcal{D}$ operates on $y(x)$ as $L(y(x)) = \sum_{i=0}^{n} a_i(x)y(x+i)$.

If $L \in \mathcal{D} - \{0\}$ we can define $\mathrm{Cont}(L) \in F(x)$ and $\mathrm{Prim}(L) \in \mathcal{P}$ in the same way as before.

Definition 2.1. An operator $L \in \mathcal{P}$ is called *Gaussian* if

$$\forall_{A \in \mathcal{D}}\ AL \in \mathcal{P} \Longrightarrow A \in \mathcal{P}.$$

If L is Gaussian then L is primitive. In the commutative case, the two properties are equivalent by Gauss's lemma. It is known that Gauss's lemma does not hold in the non-commutative case, which is illustrated in Example 2.3 below (see also [9, p. 27]).

An element $L = \sum_{i=0}^{n} a_i(x)\tau^i \in \mathcal{P}$ corresponds to a recurrence relation $L(y(x)) = 0$, i.e.

$$a_n(x)y(x+n) + a_{n-1}(x)y(x+n-1) + \cdots + a_0(x)y(x) = 0. \quad (1)$$

So we can express $y(r)$ in terms of $y(r-1), \ldots, y(r-n)$ where $r = x + n$. The expression is not defined when r is a root of the denominator, which is $a_n(r-n)$. Hence we define:

Definition 2.2. Let $L = \sum_{i=0}^{n} a_i(x)\tau^i \in \mathcal{P}$ be primitive. If $a_n \neq 0$ then define $\mathrm{ord}(L) := n$ and let $\mathfrak{lc}(L)$ be $a_n(x-n)$ divided by its leading coefficient (to make it monic). The *singularities* of L are the monic irreducible factors of $\mathfrak{lc}(L)$ in $F[x]$ (or equivalently, their roots in $\overline{F}$) with their multiplicities.

Example 2.3. Let

$$L = x^2(x^2+1)\tau - (x+1)(x^2+2x+2) \in \mathbb{Q}[x][\tau].$$

Substituting $x \mapsto x - \mathrm{ord}(L)$ in the leading coefficient we obtain $\mathfrak{lc}(L) = (x-1)^2((x-1)^2+1)$. With repetition indicating multiplicity, the singularities are $x-1$, $x-1$, $(x-1)^2+1$, or equivalently 1, 1, $1 \pm \sqrt{-1}$. Let

$$A = \frac{1}{(x+1)\left(x^2+2x+2\right)}(10\tau + 11x^2 + 15x + 14).$$

Then

$$AL = 10(x+1)\tau^2 + \left(11x^3 - 18x^2 + 35x - 50\right)\tau - 11x^2 - 15x - 14,$$

so Gauss's lemma does not hold for $\mathcal{P} \subseteq \mathcal{D}$, since

$$\mathrm{Cont}(AL) = \mathrm{Cont}(L) = 1 \neq \mathrm{Cont}(A).$$

Now $\mathfrak{lc}(AL) = x + 1 - \mathrm{ord}(AL) = x - 1$ (we divided by 10 to make it monic). Compared to L, the singularity $(x-1)^2+1$ disappeared, as well as one of the two copies of $x-1$.

Lemma 2.4 ([9, Theorem 4.1.7, Corollary 4.1.9]). *Let $L \in \mathcal{P}$. For $k = 0, 1, 2, \ldots$ let*

$$\mathcal{I}_k = \{0\} \cup \{\mathfrak{lc}(AL) \ : \ A \in \mathcal{D}, AL \in \mathcal{P}, \mathrm{ord}(A) = k\} \qquad (2)$$

and let $\mathcal{I}_\infty$ be their union. Then $\mathcal{I}_0 \subseteq \mathcal{I}_1 \subseteq \mathcal{I}_2 \subseteq \cdots \subseteq \mathcal{I}_\infty$ are ideals in $F[x]$.

Proof. To show $\mathcal{I}_k \subseteq \mathcal{I}_{k+1}$, take a non-zero $a \in \mathcal{I}_k$. So $a = \mathfrak{lc}(AL)$ for some A of order k. Replacing A by τA shows that $a \in \mathcal{I}_{k+1}$. Clearly $\mathcal{I}_k$ is closed under $F[x]$-multiplication, and it is not difficult to show that it is closed under addition as well. □

Definition 2.5 (Essential parts, removable parts, [9, pp. 31-32]). With notations as in Lemma 2.4, let $\mathfrak{lc}_k(L)$ be the monic generator of $\mathcal{I}_k$ for $k = 0, 1, 2, \ldots, \infty$. Call $\mathfrak{lc}_k(L)$ the *essential part of the leading coefficient at order k*. Note that $\mathfrak{lc}_0(L) = \mathfrak{lc}(\mathrm{Prim}(L))$ and $\mathfrak{lc}_l(L)$ divides $\mathfrak{lc}_k(L)$ if $l \geq k$. Let $\mathfrak{rp}_k(L) = \frac{\mathfrak{lc}(L)}{\mathfrak{lc}_k(L)} \in F[x]$ and call it the *removable part of the leading coefficient at order k*.

Factors (or roots) of $\mathfrak{lc}(L)$ are divided into two (possibly overlapping) sublists: Factors of $\mathfrak{lc}_\infty(L)$ are the *true singularities* of L. Factors of $\mathfrak{rp}_\infty(L)$ are the *removable singularities*. In Example 2.3 the true singularity is $x-1$ and the removable singularities are $x-1$, $(x-1)^2-1$.

Definition 2.6. Let $L \in \mathcal{P}$ be primitive. We call $A \in \mathcal{D}$ a *desingularizer* if $AL \in \mathcal{P}$. Such A is said to be *trivial* if $A \in \mathcal{P}$ (that implies $\mathfrak{lc}(L) \mid \mathfrak{lc}(AL)$, so no singularities were removed). A desingularizer A is *optimal at order k* if $\mathfrak{lc}(AL) = \mathfrak{lc}_k(L)$ and $\mathrm{ord}(A) \leq k$; when $k = \infty$, we say A is *optimal* since it removes all removable singularities while introducing no new singularities.

An optimal desingularizer at order k exists because $\mathcal{I}_k$ is a principal ideal; $F[x]$ is a PID.

2.2 LCLM method for desingularization

Some algorithms for desingularizing linear recurrence operators can be found in the literature such as [1, 2, 4, 5, 9, 15], where [1, 2, 4, 9] aim for full desingularization and [15] focuses on desingularization over $R[x]$ where R is not a field. Here we describe the so-called *LCLM method*, published in [5]. LCLM stands for the least common left multiple and GCRD the greatest common right divisor. The main result of [5] is restated below, where we focus on the recurrence case while the original version also applies to other types of Ore operators.

Theorem 2.7 (Reformulation of Theorem 6 in [5]). *Suppose $L \in \mathcal{P}$. Introduce new constants $c_0, c_1, \ldots, c_k$ that are algebraically independent over F. Denote $\hat{F} = F(c_0, c_1, \ldots, c_k)$. Consider*

$$L' = \mathrm{Prim}(\mathrm{LCLM}(L, c_0 + c_1\tau + \cdots + c_k\tau^k)) \in \hat{F}[x][\tau].$$

Then $\mathfrak{lc}(L') = \mathfrak{lc}_k(L)f$ where $f \in \hat{F}[x]$ has no non-trivial factor in $F[x]$.

The original form of Theorem 6 in [5] is:
Let q be an irreducible polynomial which appears with multiplicity e in $\mathfrak{lc}(L)$ and let $m \leq e$ be maximal such that $q^m \mid \frac{\mathfrak{lc}(L)}{\mathfrak{lc}_k(L)}$ for $k \in \mathbb{N}$. Let $A = c_0 + c_1\tau + \cdots + c_k\tau^k$ in $F(c_0, \ldots, c_k)[x][\tau]$, where $c_0, \ldots, c_k$ are new constants that are algebraically independent over F. Denote $L' = \mathrm{Prim}(\mathrm{LCLM}(L, A))$. Then the multiplicity of q in $\mathfrak{lc}(L')$ is $e-m$.

The proof in [5] also holds when $e = m = 0$, which results in Theorem 2.7. It was stated for the case $\mathrm{char}(F) = 0$, but the proof is valid for positive characteristic as well.

Remark 1. *Theorem 2.7 implies $\mathfrak{lc}_k(L)$ stays the same if L is viewed as an operator in $E(x)[\tau]$ where E is a field extension of F, as the field extension does not affect $L' = \mathrm{Prim}(\mathrm{LCLM}(L, c_0 + c_1\tau + \cdots + c_k\tau^k))$.*

Theorem 2.7 implies the following desingularization algorithm.

Algorithm 1: LCLM_Method

Input : a primitive operator $L = \sum_{i=0}^{n} a_i\tau^i \in F[x][\tau]$ and positive integer k
Output : $\mathfrak{lc}_k(L)$

1 $A \leftarrow \sum_{i=0}^{k} c_i\tau^i$, where $c_0, c_1, \ldots, c_k$ are new constants that are algebraically independent over F;
2 $L' \leftarrow \mathrm{Prim}(\mathrm{LCLM}(A, L))$;
3 **return** $\gcd(\mathfrak{lc}(L), \mathfrak{lc}(L'))$;

The discussion following the main theorem in [5] states that in characteristic 0, instead of new constants, we can let $c_0, c_1, \ldots, c_k$ be random elements in F. In this case the algorithm is Monte-Carlo, meaning it returns the desired result with a high probability. The Monte-Carlo version is much faster since it avoids computations in a transcendental extension of F. It was implemented ([14]) with $k = 1$ by the second author in 2004.

We refer to the algorithm where $k = 1$ as *order-1 LCLM method*. The Monte-Carlo version of order-1 LCLM method is useful in practice since it strikes a good balance between benefit and cost. The LCLM computation is much faster for $k = 1$ than for larger k, and, in experiments, particularly section 5.2.2, $\mathfrak{lc}_1(L)$ is often very close to $\mathfrak{lc}_\infty(L)$. We will further speed up the algorithm in Section 6.

2.3 p-Characteristic polynomial

From here until Section 6, F will be a field of characteristic p, where p is a prime number.

A general theory of linear difference equations in positive characteristic is developed in [13, Chapter 5]. In [3], p-characteristic polynomials of recurrence operators (and differential operators) over $\mathbb{F}_p[x]$ are studied and an algorithm for computing them is

given. An algorithm for computing p-characteristic polynomials of operators in $\mathbb{Z}[x][\tau]$ for a number of p is presented in [10], based on the algorithm from [3]. We will give more information about these algorithms in subsection 2.4.

Let $Z = x^p - x = x(x+1)\cdots(x+p-1)$. A straight-forward calculation shows Z is fixed by τ and hence elements in $F(Z)$ are τ-constants. In fact, $F(Z)$ *is* the field of τ-constants; to see this, notice that $F(x)$ is a field extension of $F(Z)$ with $[F(x) : F(Z)] = p$ which is a prime and hence there is no proper intermediate field. (In general, if $f \in F(x) - F$, then $[F(x) : F(f)]$ is the maximum of the degrees of the numerator and denominator of f.)

Let $\mathcal{N} : F(x) \to F(Z)$ denote the norm map of the field extension $F(x)/F(Z)$. It is given by the formula

$$\mathcal{N} : f(x) \mapsto f(x)f(x+1)\cdots f(x+p-1).$$

Denote $T = \tau^p$. The center of $\mathcal{D}$ is $F(Z)[T]$. Since $F(x)[T] \subseteq \mathcal{D}$, any $\mathcal{D}$-module is naturally an $F(x)[T]$-module, that is, an $F(x)$-vector space equipped with an $F(x)$-linear map.

Definition 2.8. For a $\mathcal{D}$-module M, call the $F(x)$-linear map induced by T the *p-curvature* of M.

For an operator $L \in \mathcal{D}$, define its p-curvature to be that of the $\mathcal{D}$-module $\mathcal{D}/\mathcal{D}L$. Denote $\chi(L) \in F(x)[T]$ its characteristic polynomial with T as the variable. Call $\chi(L)$ the *p-characteristic polynomial* of L.

A characteristic polynomial is monic by definition so the leading coefficient of L is lost in $\chi(L)$. To reinsert it, denote $\tilde{\chi}(L) = \mathcal{N}(\mathfrak{lc}(L))\chi(L)$. It is called the *reduced norm* of L in [3].

Lemma 2.9. *Properties of p-characteristic polynomials.*

(i) *For $L \in \mathcal{D}$, $\chi(L) \in F(Z)[T]$.*
(ii) *For $L \in \mathcal{D}$, $\chi(L) \in \mathcal{D}L$.*
(iii) *For $L_1, L_2 \in \mathcal{D}$, $\chi(L_1L_2) = \chi(L_1)\chi(L_2)$ and $\tilde{\chi}(L_1L_2) = \tilde{\chi}(L_1)\tilde{\chi}(L_2)$.*
(iv) *For $L_1, L_2 \in \mathcal{D}$, if $\mathrm{GCRD}(L_1, L_2) = 1$, then*
$$\chi(\mathrm{LCLM}(L_1, L_2)) = \chi(L_1)\chi(L_2).$$
(v) *For $L \in \mathcal{P}$, $\tilde{\chi}(L) \in F[Z][T]$ and $\deg_Z(\tilde{\chi}(L)) \le \deg_x(L)$.*
(vi) *If $L \in F(Z)[T]$ then $\tilde{\chi}(L) = L^p$.*

Proof. All except item (iv) are proved in [3, Section 3] for the case $F = \mathbb{F}_p$ and the proofs are valid for a general field F with positive characteristic. We now prove (iv). Denote $L = \mathrm{LCLM}(L_1, L_2)$. If $\mathrm{GCRD}(L_1, L_2) = 1$ then $\mathcal{D}/\mathcal{D}L \cong \mathcal{D}/\mathcal{D}L_1 \oplus \mathcal{D}/\mathcal{D}L_2$ as $\mathcal{D}$-modules (and hence as $F(x)[T]$-modules). Now (iv) follows from the fact that characteristic polynomials are multiplicative on direct sums. □

Lemma 2.9(iii) implies that an operator factors only when its p-characteristic polynomial factors (as a polynomial in $F(Z)[T]$). In fact, the p-characteristic polynomial tells us even more. See [6] and [12] for discussions on this topic in the differential case. The p-characteristic polynomial is also useful for testing or proving irreducibility of operators in $\mathbb{Q}(x)[\tau]$ by reduction modulo p.

2.4 BCS algorithm and Pagès' algorithm

Bostan, Caruso and Schost (2015) present an algorithm for computing the p-characteristic polynomial of a linear recurrence operator in $\mathbb{F}_p[x][\tau]$, called `Xi_theta_d` in [3, p. 7]. We refer to it as the BCS algorithm. Their implementation in Magma is available at https://github.com/schost.

The BCS algorithm takes a prime p and a difference operator $L \in \mathbb{F}_p[x][\tau]$ as its input and computes $\tilde{\chi}(L) \in \mathbb{F}_p[Z][T]$ (making $\tilde{\chi}(L)$ monic gives $\chi(L)$). The algorithm computes $\tilde{\chi}(L)$ in $\mathbb{F}_p[[Z]][T]$ to precision $O(Z^{\deg_x(L)+1})$ which suffices by Lemma 2.9(v).

For $L \in \mathcal{P}$, the first part of Lemma 2.9(v) implies that $\mathcal{N}(\mathfrak{lc}(L))$ is a denominator bound for $\chi(L)$. The BCS algorithm uses this bound to ensure that what it computes in $\mathbb{F}_p[[Z]][T]$ is in $\mathbb{F}_p[Z][T]$, not just in $\mathbb{F}_p(Z)[T]$. We will show that (partial) desingularization leads to sharper denominator bounds. That reduces the required Z-adic precision, speeding up the computation. In fact, our main result Theorem 3.1 says that full desingularization gives the exact denominator.

For an operator $L \in \mathbb{Q}(x)[\tau]$, let $\chi_p(L)$ be the p-characteristic polynomial of its reduction modulo p. Pagès (2021) gives an algorithm for computing $\chi_p(L)$ for a number of primes at the same time, in the case where $L \in \mathbb{Z}[x][\tau]$ has a leading coefficient in $\mathbb{Z}$ ([10, Algorithm 3]). The algorithm is based on the BCS algorithm.

3 MAIN THEOREM AND COROLLARIES

Again, in the Main Theorem, its proof and its applications, F is a field of characteristic p.

Let $\mathrm{denom}(\cdot)$ be the monic denominator of a rational function, or of a polynomial with rational function coefficients. We will use this notation in the cases $F[x] \subset F(x)$ and $F[Z][T] \subset F(Z)[T]$.

Theorem 3.1. *For $L \in F[x][\tau]$, $\mathrm{denom}(\chi(L)) = \mathcal{N}(\mathfrak{lc}_\infty(L))$.*

The theorem quickly implies two corollaries, expressed in terms of the following definition.

Definition 3.2. Let $r_1, r_2 \in F(x) - \{0\}$. We say that r_1 and r_2 are *shift equivalent*, denoted $r_1 \sim r_2$, if $\tau - \frac{r_1}{r_2}$ has a non-zero solution in $F(x)$, in other words, if there exists $f \in F(x) - \{0\}$ for which $\frac{r_1}{r_2} = \frac{\tau(f)}{f}$.

If $r(x)$ has a factor $q(x)$ in the numerator or denominator, and one replaces $q(x)$ by its shift $q(x+1)$, then the result is shift-equivalent to $r(x)$. Note that $r_1 \sim r_2$ if and only if $\mathcal{N}(r_1) = \mathcal{N}(r_2)$.

Corollary 3.3. *If $\mathcal{D}/\mathcal{D}L_1 \cong \mathcal{D}/\mathcal{D}L_2$ for $L_1, L_2 \in \mathcal{D}$, then $\mathfrak{lc}_\infty(L_1)$ and $\mathfrak{lc}_\infty(L_2)$ are shift equivalent, so L_1 and L_2 have the same true singularities up to shifts.*

Corollary 3.4. *For $L_1, L_2 \in \mathcal{D}$, if*

- *$L = L_1L_2$, or*
- *$L = \mathrm{LCLM}(L_1, L_2)$ and $\mathrm{GCRD}(L_1, L_2) = 1$*

then $\mathfrak{lc}_\infty(L)$ and $\mathfrak{lc}_\infty(L_1)\mathfrak{lc}_\infty(L_2)$ are shift equivalent.

We did not expect Theorem 3.1 because these corollaries do not hold in characteristic 0:

Example 3.5. Let $L_1 = \tau - 1 \in \mathbb{Q}(x)[\tau]$ and

$$L = L_1x = \tau \cdot x - x = (x+1)\tau - x.$$

Then x is a true singularity by applying Lemma 4 in [4, p. 4]. However, neither L_1 nor x has true singularities, which disproves the analogue of Corollary 3.4 in characteristic 0. It is also a counter-example to the analogue of Corollary 3.3 since $\mathcal{D}/\mathcal{D}L_1 \cong \mathcal{D}/\mathcal{D}L$.

Another potential application of Theorem 3.1 is testing full desingularization; once we have $\chi(L)$ for $L \in F[x][\tau]$, the shift equivalence class of $\mathfrak{lc}(L)$ is known.

4 PROOF OF THE MAIN THEOREM

This section is devoted to the proof of Theorem 3.1. We start with an easy lemma.

Lemma 4.1. *For any $A, L \in \mathcal{D}$*

$$\operatorname{denom}(\chi(AL)) = \operatorname{denom}(\chi(A))\operatorname{denom}(\chi(L)).$$

Proof. Lemma 2.9(iii) and Gauss's lemma for $F(Z)[T]$ gives

$$\operatorname{Cont}(\chi(AL)) = \operatorname{Cont}(\chi(A))\operatorname{Cont}(\chi(L)).$$

Notice that $\chi(\cdot)$ is monic by definition, and the denominator of a monic polynomial is the reciprocal of the content. □

4.1 Special case, Gaussian operators

Gaussian operators are defined in Definition 2.1. In the following lemma we characterize them in terms of desingularization. In fact, Gaussian operators are exactly those that do not have any removable singularities.

Lemma 4.2. *Let $L \in \mathcal{P}$. The following are equivalent.*

1. *L is Gaussian, i.e. $\forall_{A\in\mathcal{D}}\ AL \in \mathcal{P} \Longrightarrow A \in \mathcal{P}$.*
2. *Every desingularizer is trivial.*
3. *$\operatorname{Cl}(L) = \mathcal{P}L$, where $\operatorname{Cl}(L) := \mathcal{D}L \bigcap \mathcal{P}$. (This is called the* Weyl closure *in [11].)*
4. *$\mathfrak{lc}(L) = \mathfrak{lc}_\infty(L)$, i.e. there are no removable singularities.*

Proof. Items 2 and 3 are reformulations of item 1, and immediately imply item 4. It remains to show that item 4 implies item 1. Suppose that $\mathfrak{lc}(L) = \mathfrak{lc}_\infty(L)$ and $AL \in \mathcal{P}$. To prove: $A \in \mathcal{P}$.

By partial fraction decomposition, $A = A_1 + A_2$ where $A_1 \in \mathcal{P}$ and $A_2 = \sum q_i\tau^i$ with the numerator of q_i having lower degree than its denominator. Since AL and A_1L are in $\mathcal{P}$, their difference A_2L is in $\mathcal{P}$ as well. If $A_2 \neq 0$, then the leading coefficient of A_2L will have lower degree than $\mathfrak{lc}(L)$, contradicting item 4. Thus $A_2 = 0$ and hence $A \in \mathcal{P}$. □

Lemma 2.9(v) says that $\tilde{\chi}(L) = \mathcal{N}(\mathfrak{lc}(L))\chi(L) \in F[Z][T]$ when $L \in \mathcal{P}$, in other words

$$\operatorname{denom}(\chi(L)) \mid \mathcal{N}(\mathfrak{lc}(L)). \tag{3}$$

Here we give a sharper denominator bound.

Lemma 4.3. *For $L \in F[x][\tau]$, $\operatorname{denom}(\chi(L)) \mid \mathcal{N}(\mathfrak{lc}_\infty(L))$.*

Proof. Let $A \in \mathcal{D}$ be an optimal desingularizer of L, then $\mathfrak{lc}(AL) = \mathfrak{lc}_\infty(L)$. From Lemma 4.1 and Equation (3) applied to AL, $\operatorname{denom}(\chi(L)) \mid \operatorname{denom}(\chi(AL)) \mid \mathcal{N}(\mathfrak{lc}(AL)) = \mathcal{N}(\mathfrak{lc}_\infty(L))$. □

Next we show that our denominator bound is exact for Gaussian operators. The next section will prove the general case by exploiting the fact that any operator has a Gaussian left multiple.

Lemma 4.4. *If $L \in F[x][\tau]$ is Gaussian, then*

$$\operatorname{denom}(\chi(L)) = \mathcal{N}(\mathfrak{lc}_\infty(L)).$$

Proof. Denote $f = \operatorname{Prim}(\chi(L)) \in F[Z][T]$. By Lemma 2.9(ii), $f \in \mathcal{D}L$ so there exists $Q \in \mathcal{D}$ such that

$$QL = f. \tag{4}$$

In fact $Q \in \mathcal{P}$ since L is Gaussian. Lemma 2.9(v) says $\tilde{\chi}(Q)$, $\tilde{\chi}(L) \in F[Z][T]$. Applying $\tilde{\chi}$ to Equation (4), and Lemma 2.9(vi), gives

$$\tilde{\chi}(Q)\tilde{\chi}(L) = \tilde{\chi}(f) = f^p. \tag{5}$$

Now $f^p \in F[Z][T]$ is primitive since f is primitive. Then Gauss's lemma implies $\tilde{\chi}(L) \in F[Z][T]$ is primitive. It follows that

$$\operatorname{denom}(\chi(L)) = \mathfrak{lc}(\tilde{\chi}(L)) = \mathcal{N}(\mathfrak{lc}(L)) = \mathcal{N}(\mathfrak{lc}_\infty(L))$$

where the last equality comes from Lemma 4.2, part 4. □

4.2 Proof for the general case

Lemma 4.5. *Suppose $L \in \mathcal{P}$ and $A = \sum_{i=0}^{k} \frac{n_i}{d_i}\tau^i \in \mathcal{D}$ is a desingularizer of L, where $\frac{n_i}{d_i} \in F(x)$ is in lowest terms for each i. Then $\mathcal{N}(d_k) \mid \mathcal{N}(\mathfrak{rp}_\infty(L))$.*

Proof. Let n be the order of L. The definition of $\mathfrak{lc}_\infty$ and the product of the leading terms of A and L gives

$$\mathfrak{lc}_\infty(L) \mid \tau^{-k-n}(\frac{n_k}{d_k})\mathfrak{lc}(L)$$

and hence $\tau^{-k-n}(\frac{n_k}{d_k})\mathfrak{rp}_\infty(L) \in F[x]$. Since $\frac{n_k}{d_k}$ is a reduced fraction, we have $\tau^{-k-n}(d_k) \mid \mathfrak{rp}_\infty(L)$, which leads to

$$\mathcal{N}(d_k) = \mathcal{N}(\tau^{-k-n}(d_k)) \mid \mathcal{N}(\mathfrak{rp}_\infty(L)).$$

□

Lemma 4.6. *Suppose $L \in \mathcal{P}$ and $A \in \mathcal{D}$ is an optimal desingularizer of L. Then there exists a positive integer N such that*

$$\operatorname{denom}(\chi(A)) \mid (\mathcal{N}(\mathfrak{rp}_\infty(L)))^N.$$

Proof. Write $A = \sum_{i=0}^{k} \frac{n_i}{d_i}\tau^i$, where $\frac{n_i}{d_i} \in F(x)$ is a reduced fraction for each i. We deduce $n_k = 1, d_k = \tau^{n+k}(\mathfrak{rp}_\infty(L))$ from the fact that $\mathfrak{lc}(AL) = \mathfrak{lc}_\infty(L)$. Clearly $d_0d_1\cdots d_kA \in \mathcal{P}$. Then by Equation (3) we have

$$\operatorname{denom}(\chi(A)) \mid \mathcal{N}(\mathfrak{lc}(d_0d_1\cdots d_kA)) = \mathcal{N}(d_0d_1\cdots d_{k-1}). \tag{6}$$

Now we bound $\mathcal{N}(d_j)$ in terms of $\mathfrak{rp}_\infty(L)$. Let

$$A_j = d_kd_{k-1}\cdots d_{j+1}(\sum_{i=0}^{j} \frac{n_i}{d_i}\tau^i)$$

for $j = 0, 1, \ldots, k-1$. Notice that

$$A_j - d_kd_{k-1}\cdots d_{j+1}A = -d_kd_{k-1}\cdots d_{j+1}(\sum_{i=j+1}^{k} \frac{n_i}{d_i}\tau^i) \in \mathcal{P}.$$

This implies $A_jL \in \mathcal{P}$, or equivalently, A_j is a desingularizer of L. Apply Lemma 4.5 to A_j:

$$\mathcal{N}(\operatorname{denom}(d_kd_{k-1}\cdots d_{j+1}\frac{n_j}{d_j})) \mid \mathcal{N}(\mathfrak{rp}_\infty(L)).$$

Notice that

$$d_j \mid d_kd_{k-1}\cdots d_{j+1}\cdot\operatorname{denom}(d_kd_{k-1}\cdots d_{j+1}\frac{n_j}{d_j}).$$

Therefore

$$\mathcal{N}(d_j) \mid \mathcal{N}(d_k d_{k-1} \cdots d_{j+1}) \cdot \mathcal{N}(\mathfrak{rp}_\infty(L)).$$

Recall that $\mathcal{N}(d_k) \mid \mathcal{N}(\mathfrak{rp}_\infty(L))$. By downward induction on j, we conclude that

$$\mathcal{N}(d_j) \mid (\mathcal{N}(\mathfrak{rp}_\infty(L)))^{k-j+1}$$

for $j = 0, 1, 2, \ldots, k-1$. Then $\operatorname{denom}(\chi(A))$ is bounded by a power of $\mathfrak{rp}_\infty(L)$ in terms of divisibility due to Equation 6.

□

We are now ready to finish the proof of Theorem 3.1.

Proof of Theorem 3.1. It remains to show that $\mathcal{N}(\mathfrak{lc}_\infty(L)) \mid \operatorname{denom}(\chi(L))$ for any $L \in \mathcal{P}$. There exists a sufficiently large k such that $\mathfrak{lc}_k(L) = \mathfrak{lc}_\infty(L)$. Introduce new constants $c_0, \ldots, c_k$ that are algebraically independent over F and denote $E = F(c_0, c_1, \ldots, c_k)$. Let

$$L' = \operatorname{Prim}(\operatorname{LCLM}(c_k\tau^k + \cdots + c_0, L)) \in E[x][\tau].$$

Theorem 2.7 says $\mathfrak{lc}(L') = \mathfrak{lc}_\infty(L)f$, where $f \in E[x]$ has no non-trivial factor in $F[x]$. It follows from Definition 2.5 (see also Equation 2) that $\mathfrak{lc}_\infty(L) \mid \mathfrak{lc}_\infty(L')$ and hence $\mathfrak{rp}_\infty(L') \mid f$. Remark 1 guarantees $\mathfrak{lc}_\infty(L)$ does not change as we shift from F to E. Let A be an optimal desingularizer of L'. By Lemma 2.9 ((iii) and (iv)), we have

$$\chi(AL') = \chi(A)\chi(c_k\tau^k + \cdots + c_0)\chi(L). \tag{7}$$

Since $c_0, \ldots, c_k$ are τ-constants, Lemma 2.9(v) implies

$$\operatorname{denom}(\chi(c_k\tau^k + \cdots + c_0)) = 1.$$

Applying Lemma 4.1 to Equation (7) gives

$$\operatorname{denom}(\chi(AL')) = \operatorname{denom}(\chi(A))\operatorname{denom}(\chi(L)).$$

Since AL' is Gaussian, we know from Lemma 4.4

$$\operatorname{denom}(\chi(AL')) = \mathcal{N}(\mathfrak{lc}_\infty(AL')),$$

which equals $\mathcal{N}(\mathfrak{lc}_\infty(L'))$ since A is an optimal desingularizer of L'. As a consequence,

$$\mathcal{N}(\mathfrak{lc}_\infty(L)) \mid \mathcal{N}(\mathfrak{lc}_\infty(L')) = \operatorname{denom}(\chi(A))\operatorname{denom}(\chi(L)). \tag{8}$$

Lemma 4.6 says $\operatorname{denom}(\chi(A))$ is a factor of f^N for some sufficiently large N, so $\operatorname{denom}(\chi(A))$ has no non-trivial factor in $F[x]$. By taking only factors in $F[x]$ in Equation 8, we obtain the desired result $\mathcal{N}(\mathfrak{lc}_\infty(L)) \mid \operatorname{denom}(\chi(L))$.

□

5 APPLICATION TO COMPUTATIONS

In this section $F = \mathbb{F}_p$.

5.1 Algorithm

Let $L \in \mathcal{P}$ and $\alpha = \mathfrak{rp}_k(L)$. Theorem 3.1 implies that $\mathcal{N}(\alpha)$, which is in $\mathbb{F}_p[Z]$, is a factor of $\tilde{\chi}(L)$. Dividing this factor away reduces the degree bound from Lemma 2.9(v) to

$$\deg_Z(\,\mathcal{N}(\alpha)^{-1}\tilde{\chi}(L)\,) \le \deg_x(L) - \deg_x(\alpha) \tag{9}$$

which becomes an equality when k is sufficiently large. However, we use $k = 1$ to minimize the time spent computing α. The reduced degree bound allows us to recover $\chi(L)$ from a lower precision Z-adic expansion. That leads to the following algorithm.

Algorithm 2: `Xi_p_desing`

Input : prime p and $L \in \mathbb{F}_p[x][\tau]$
Output : $\operatorname{Prim}(\chi(L)) \in \mathbb{F}_p[Z][T]$

1. Pick $k \ge 1$ and compute $\mathfrak{lc}_k(L)$ and $\alpha := \mathfrak{rp}_k(L) \in \mathbb{F}_p[x]$. We use $k = 1$ to minimize the time spent in this step.
2. Compute $\mathcal{N}(\alpha) \in \mathbb{F}_p[Z]$. Let v be its Z-adic valuation in $\mathbb{F}_p[[Z]]$ and let $\beta = Z^{-v}\mathcal{N}(\alpha) \in \mathbb{F}_p[Z]$. For computing $\mathcal{N}(\cdot)$ see Step 3 of `Xi_theta_d` in [3].
3. Let $d_1 = \deg_Z(\beta)$ and $d := \deg_x(L)$. Apply the BCS algorithm with d replaced by $d - d_1$ to L to obtain $\tilde{\chi}(L)$ up to the precision $O(Z^{d-d_1+1})$. Denote the result by χ_1.
4. Compute β^{-1} in $\mathbb{F}_p[[Z]]$ up to $O(Z^{d-d_1-v+1})$. This can be done by applying the extended Euclidean algorithm to β and Z^{d-d_1-v+1}.
5. Compute $\beta^{-1} \cdot (Z^{-v}\chi_1)$ in $\mathbb{F}_p[[Z]][T]$ up to the precision $O(Z^{d-d_1-v+1})$. This gives $\mathcal{N}(\alpha)^{-1}\chi_1 \in \mathbb{F}_p[Z][T]$. Return its primitive part (with respect to T). Note: $\mathcal{N}(\alpha)^{-1}\chi_1$ and $\mathcal{N}(\alpha)^{-1}\tilde{\chi}(L)$ agree to precision $O(Z^{d-d_1-v+1})$ which suffices by (9).

Step 3 is where we save CPU time over the original algorithm from [3] if $d_1 > 0$. If $d_1 = 0$ then there is no improvement in efficiency. In this case one would expect our algorithm to be slower due to some extra steps; however, as we will see in the following section, the extra steps cost so little time that the time difference is negligible.

5.2 Implementation and timings

Our Magma implementation of algorithm `Xi_p_desing` is available at https://www.math.fsu.edu/~yzhou/desingandpcurv/pcurv/
Our experiments can also be found at this URL. Note: first load the implementation of [3] at https://github.com/schost/pCurvature (file `pCurvature.mgm`).

5.2.1 OEIS operators. The following table presents the data of our experiments on two operators from OEIS ([7], [8]). Here d_1 is defined in the Step 3 of `Xi_p_desing`; each running time is the average of ten runs.

OEIS index	order	x-degree	d_1	BCS	`Xi_p_desing`
A151329	9	18	10	17.2s	9.6s
A002777	4	3	0	6.97s	7.01s

Table 1: Timings for operators from OEIS. $p = 27457$.

For OEIS A002777, we expect `Xi_p_desing` to be slower than BCS since $d_1 = 0$. However, as we can see from the table, the running time difference between two algorithms is nearly unnoticeable, which indicates that the attempted partial desingularization takes relatively little time.

The question now is what to expect asymptotically for larger examples? Should we expect small d_1 (few removable singularities) or high d_1 (most singularities are removable)? The answer depends on how the examples are constructed. For random operators d_1 is typically zero. However, random operators are unlikely to be useful.

Order-degree curves [4] suggest that d_1 is likely to be large for high order operators coming from creative telescoping.

Chapter 6 in [17] gives a bound for the number of removable singularities of right-factors. The bound can be large. Of course, the mere fact that a bound can be high does not imply that the actual number is also high, so we decided to test this experimentally by taking LCLM's of a sequence of operators with increasing orders and degrees.

5.2.2 LCLM operators. We generate random operators

$$L_1 = \sum_{i=0}^{N}\sum_{j=0}^{N} a_{ij}x^j\tau^i, \quad L_2 = \sum_{i=0}^{N}\sum_{j=0}^{N} b_{ij}x^j\tau^i$$

where a_{ij}'s and b_{ij}'s are random non-zero integers (i.e., L_1 and L_2 are dense), and calculate the $p = 1987$ characteristic polynomial of

$$L = \mathrm{Prim}(\mathrm{LCLM}(L_1, L_2)),$$

if its order does not drop modulo p. Table 2 exhibits the results. In the experiments, we observe that

$$\mathrm{ord}(L) = 2N, \quad \deg_x(L) = 2N^2 + 2N, \quad d_1 = 2N^2.$$

This suggests that high-order LCLM operators are likely to have quadratically more removable singularities than true singularities. If true, it would indicate an improvement that is more than a constant factor. We hope to soon prove this using Chapter 6 in [17].

N	d_1	BCS	Xi_p_desing
5	50	5.8s	1.7s
6	72	14.1s	3.7s
7	98	22.1s	4.9s
8	128	43.5s	8.6s
9	162	73.3s	14.4s
10	200	113.1s	18.2s

Table 2: Timings for LCLM operators.

6 FAST ALGORITHMS FOR DESINGULARIZATION AT ORDER 1

Convention throughout this section: $L = \sum_{i=0}^{n} a_i\tau^i \in F[x][\tau]$ is the operator to be desingularized; $a_i = 0$ for $i < 0$ and $i > n$.

6.1 First algorithm

We present our first speedup of the order-1 LCLM method, which is used in Step 1 of algorithm 2.

The order-1 LCLM method computes $L' = \mathrm{LCLM}(L, \tau - c)$ where c is a new constant (or a random number in the Monte-Carlo version). To speed this up, our idea is to obtain $\mathfrak{lc}_1(L)$ while only computing a portion of L'.

First we express L' in terms of c and coefficients of L. In fact, $L' = \sum_{i=0}^{n} c^i L_i$, where

$$L_i = a_i\tau L - \tau(a_{i-1})L = (a_i\tau - \tau(a_{i-1}))L. \tag{10}$$

Equation 10 Clearly this L' is a left multiple of L. To verify it is also a left-multiple of $\tau - c$, use the fact that the remainder of τ^i right-divided by $\tau - c$ is c^i. We skip the tedious computation. As a result L' is an LCLM of L and $\tau - c$.

The order-1 LCLM method computes L', which amounts to computing all L_i's. The following proposition shows that one can provably obtain $\mathfrak{lc}_1(L)$ from just a subset of the L_i's.

Proposition 6.1. *Let L_i be defined by Equation 10.If $\{i_1, i_2, \ldots, i_k\} \subseteq \{0, 1, 2, \ldots, k\}$ such that*

$$\gcd(a_{i_1}, a_{i_2}, \ldots, a_{i_k}) = 1, \tag{11}$$

then

$$\gcd(\mathfrak{lc}_0(L_{i_1}), \mathfrak{lc}_0(L_{i_2}), \ldots, \mathfrak{lc}_0(L_{i_k})) = \mathfrak{lc}_1(L).$$

The proof will be given in the next section. Note that there exist $i_1, i_2, \ldots, i_k$ satisfying the gcd condition (Equation 11) if and only if L is primitive. The proposition immediately gives rise to algorithm 3.

Algorithm 3: $\mathfrak{lc}_1$

Input : a primitive operator $L = \sum_{i=0}^{n} a_i\tau^i \in F[x][\tau]$
Output: $\mathfrak{lc}_1(L)$

1 Find $I \subset \{0, 1, \ldots, n\}$ such that $a_i \neq 0$ for any $i \in I$ and $\gcd(a_i \mid i \in I) = 1$. Note: the algorithm is still correct if we allow $a_i = 0$ for $i \in I$, but that i is redundant since it does not affect the gcd at all.
2 Compute L_i for $i \in I$ by Equation 10.
3 Return $\gcd(\mathfrak{lc}_0(L_i) : i \in I)$.

Remark 2. *Computing $\mathfrak{lc}_0(L_i) = \mathfrak{lc}(\mathrm{Prim}(L_i))$ is the most time-consuming part in the algorithm, because L_i has twice the x-degree as L.*

6.2 Proof

Always assume $L = \sum_{i=0}^{n} a_i\tau^i \in F[x][\tau]$ is primitive and L_i is defined by Equation 10.

Lemma 6.2. *There exists $b \in F[x]$ such that*

$$\mathrm{Cont}((\tau - b)L) = \tau^{n+1}(\mathfrak{rp}_1(L)).$$

Proof. Let $A \in \mathcal{D}$ be an optimal desingularizer of L at order 1. Then $A = \frac{1}{d_1}\tau - \frac{n_2}{d_2}$, where $d_1 = \tau^{n+1}(\mathfrak{rp}_1(L))$ and $\frac{n_2}{d_2} \in F(x)$ is a reduced fraction. Let $b = d_1\frac{n_2}{d_2}$. Observe that

$$bL = \tau \cdot L - d_1AL \in \mathcal{P}. \tag{12}$$

Due to L being primitive, b has to be a polynomial. Since A is an optimal desingularizer of L at order 1, $AL \in \mathcal{P}$ is primitive; otherwise dividing out the content of AL yields a more optimal desingularizer. By rearranging Equation 12 we see that $\frac{1}{d_1}(\tau - b)L = AL$ is primitive, which completes the proof. □

Theorem 6.3. *Let $C = (c_0, c_1, \ldots, c_{n+1}) \in F^{n+2}$. Denote*

$$C_1 = \sum_{i=0}^{n+1} c_ia_i, \quad C_0 = \sum_{i=0}^{n+1} c_i\tau(a_{i-1}), \quad L' = (C_1\tau - C_0)L.$$

Then

$$\mathfrak{lc}_1(L) \mid \mathfrak{lc}_0(L') \mid \tau^{-n-1}(C_1)\mathfrak{lc}_1(L).$$

Proof. Assume $C_1 \neq 0$ since otherwise it is trivial.

The relation $\mathfrak{lc}_1(L) \mid \mathfrak{lc}_0(L')$ immediately follows from the definition of $\mathfrak{lc}_1$. By Lemma 6.2, there exists $b \in F[x]$ such that $\mathrm{Cont}((\tau - b)L) = \tau^{n+1}(\mathfrak{rp}_1(L))$. Since

$$(\tau - b)L = \sum_{i=0}^{n+1} (\tau(a_{i-1}) - ba_i)\tau^j,$$

we have

$$\gcd(\tau(a_{i-1}) - ba_i \mid j = 0, 1, \ldots, n+1) = \tau^{n+1}(\mathfrak{rp}_1(L)). \quad (13)$$

Notice that

$$L' = C_1(\tau - b)L + (bC_1 - C_0)L,$$

and in particular

$$bC_1 - C_0 = \sum_{i=0}^{n+1} bc_i a_i - \sum_{i=0}^{n+1} c_i \tau(a_{i-1}) = \sum_{i=0}^{n+1} c_i (ba_i - \tau(a_{i-1}))$$

is a multiple of $\tau^{n+1}(\mathfrak{rp}_1)$ due to Equation 13. Hence $\frac{1}{\tau^{n+1}(\mathfrak{rp}_1)} L' \in F[x][\tau]$. When $C_1 \neq 0$,

$$\mathfrak{lc}\left(\frac{1}{\tau^{n+1}(\mathfrak{rp}_1(L))} L'\right) = \frac{1}{\mathfrak{rp}_1(L)} \tau^{-n-1}(C_1)\mathfrak{lc}(L) = \tau^{-n-1}(C_1)\mathfrak{lc}_1(L).$$

Then we have

$$\mathfrak{lc}_1(L') = \mathfrak{lc}(\mathrm{Prim}(L')) \mid \mathfrak{lc}\left(\frac{1}{\tau^{n+1}(\mathfrak{rp}_1(L))} L'\right) = \tau^{-n-1}(C_1)\mathfrak{lc}_1(L).$$

□

Proof of Proposition 6.1. In Theorem 6.3, setting $c_i = 1$ for some i and $c_j = 0$ for any $j \neq i$ yields

$$\mathfrak{lc}_1(L) \mid \mathfrak{lc}_0(L_i) \mid \tau^{-n-1}(a_i)\mathfrak{lc}_0(L).$$

The desired result follows immediately. □

6.3 Desingularizing both leading and trailing coefficients with one L_i

The variation in this section handles both leading and trailing singularities using only one L_i (defined in Equation 10) without checking the gcd condition (Equation 11). One goal of this variation is to check if it further speeds up the desingularization algorithm.

In the algorithm, $\mathfrak{tc}_1$ denotes the *essential part of the trailing coefficient at order 1*, which is the counterpart of $\mathfrak{lc}_1$ for the leading coefficient.

Algorithm 4: lc1_tc1

Input : a primitive operator $L = \sum_{i=0}^{n} a_i \tau^i \in F[x][\tau]$ with $a_0 a_n \neq 0$

Output: $l, t \in F[x]$ such that $\mathfrak{lc}_1(L) \mid l \mid \mathfrak{lc}(L)$ and $\mathfrak{tc}_1(L) \mid t \mid \mathfrak{tc}(L)$

$i \leftarrow \lfloor \frac{n}{2} \rfloor$
$L_i \leftarrow (a_i\tau - \tau(a_{i-1}))L$
$l, t \leftarrow \mathfrak{lc}_0(L_i), \mathfrak{tc}_0(L_i)$
$l, t \leftarrow \gcd(\tau^{-n}(a_n), l), \gcd(a_0, t)$
return l, t

algorithms	running time	x-degree in output
Order-1 LCLM	1.191s	6
lc1	0.055s	6
lc1_tc1	0.092s	6

Table 3: Comparison of different desingularization algorithms

6.4 Examples and comparisons

We have implemented algorithm 3 and algorithm 4 in Maple and SageMath, and done some experiments to compare the running time of our algorithm with the order-1 LCLM method. All can be found at [16]. Below we give an experiment we did in Maple.

Example 6.4. In this example the base field is $\mathbb{Q}$. We took random operators

$$L_1 = (26x^4 + 20)\tau^{11} - 96x^3\tau^9 + 64x^5\tau^8 + 45x^{11}\tau^4 - x^2\tau^3,$$
$$L_2 = -55x^3\tau^7 + 85x^3\tau^4 + 64x^4\tau^3 + (-14x^8 - 20x^4)\tau + 79x,$$

and then computed

$$L = \mathrm{Prim}(\mathrm{LCLM}(L_1, L_2)).$$

The x-degree of L is 109. We desingularize L using three different algorithms. For the LCLM method we used the Monte-Carlo version and randomly choose $c = 7$. The results are shown in the Table 3, where each time is the average of ten runs.

7 FUTURE WORK

7.1 Application to Pagès' algorithm

Pagès' algorithm computes $\chi_p(L)$ for $L \in \mathbb{Z}[x][\tau]$ with the restriction $\mathfrak{lc}(L) \in \mathbb{Z}$, but with minor adjustments it applies to all recurrence operators in $\mathbb{Z}[x][\tau]$. We expect desingularization to be beneficial here as well.

7.2 Differential case

The desingularization improvement should also work for the differential case or Ore operators. For a differential operator $L = \sum_{i=0}^{n} a_i \partial^i \in F[x][\partial]$, we can write $\mathrm{LCLM}(L, \tau - c) = \sum_{i=0}^{n} c^i L_i$, where

$$L_i = (a_i\partial - (a_{i-1} + a_i'))\, L.$$

We expect that there should also be a differential analog of our main result, Theorem 3.1.

REFERENCES

[1] S. A. Abramov, M. A. Barkatou, and M. van Hoeij. Apparent singularities of linear difference equations with polynomial coefficients. *Applicable Algebra in Engineering, Communication and Computing*, 17:117 – 133, 2006.

[2] Sergei Abramov. Eg-eliminations. *Journal of Difference Equations and Applications*, 5:393–433, 01 1999.

[3] Alin Bostan, Xavier Caruso, and Éric Schost. A fast algorithm for computing the characteristic polynomial of the p-curvature. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation*, ISSAC '14, pages 59–66, New York, NY, USA, 2014. ACM.

[4] Shaoshi Chen, Maximilian Jaroschek, Manuel Kauers, and Michael F. Singer. Desingularization explains order-degree curves for Ore operators. In *Proceedings of the 38th International Symposium on Symbolic and Algebraic Computation*, ISSAC '13, pages 157–164, New York, NY, USA, 2013. Association for Computing Machinery.

[5] Shaoshi Chen, Manuel Kauers, and Michael F. Singer. Desingularization of Ore operators. *Journal of Symbolic Computation*, 74:617 –– 626, 2016.

[6] Thomas Cluzeau. Factorization of differential systems in characteristic p. In *Proceedings of the 2003 International Symposium on Symbolic and Algebraic Computation*, ISSAC '03, pages 58–65, New York, NY, USA, 2003. Association for Computing Machinery.

[7] OEIS Foundation Inc. Entry A002777 in the on-line encyclopedia of integer sequences. http://oeis.org/A002777.

[8] OEIS Foundation Inc. Entry A151329 in the on-line encyclopedia of integer sequences. http://oeis.org/A151329.

[9] Maximilian Jaroschek. *Removable Singularities of Ore Operators*. PhD thesis, RISC, Johannes Kepler University Linz, 2013.

[10] Raphaël Pagès. Computing characteristic polynomials of p-curvatures in average polynomial time. In *Proceedings of the 2021 on International Symposium on Symbolic and Algebraic Computation*, ISSAC '21, pages 329–336, New York, NY, USA, 2021. Association for Computing Machinery.

[11] Harrison Tsai. Weyl closure of a linear differential operator. *Journal of Symbolic Computation*, 29:4–5, 2000.

[12] Marius van der Put. Reduction modulo p of differential equations. *Indagationes Mathematicae*, 7(3):367–387, 1996.

[13] Marius van der Put and Michael F. Singer. *Galois Theory of Difference Equations*. Lecture Notes in Mathematics. Springer Berlin Heidelberg, 1997.

[14] Mark van Hoeij. *Implementation of* `DEtools[Homomorphisms]`*, added to Maple in 2005. It uses* `LCLM` *to discard apparent singularities. http://www.math.fsu.edu/~hoeij/files/Hom*. 2004.

[15] Yi Zhang. Contraction of Ore ideals with applications. In *Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation*, ISSAC '16, pages 413–420, New York, NY, USA, 2016. Association for Computing Machinery.

[16] Yi Zhou. Implementations and examples. http://www.math.fsu.edu/~yzhou/desingandpcurv, 2021.

[17] Yi Zhou. *Algorithms for Factoring Linear Recurrence Operators*. PhD thesis, Florida State University, 2022.