A Modular Algorithm to Compute the Exponential Solutions of a Linear Differential Operator

Thomas Cluzeau[*]
LACO, Université de Limoges
123 avenue Albert Thomas
87060 Limoges, France
thomas.cluzeau@unilim.fr

Mark van Hoeij[†]
Department of mathematics
Florida State University
Tallahassee, FL 32306, USA
hoeij@math.fsu.edu

February 18, 2004

Abstract

We present a new algorithm to compute exponential solutions of differential operators with rational function coefficients. We use a combination of local and modular computations, which allows to reduce the number of possibilities in the combinatorial part of the algorithm. We also show how unnecessarily large algebraic extensions of the constants can be avoided in the algorithm.

Introduction

An ordinary differential equation

$$y^{(n)} + a_{n-1}y^{(n-1)} + \cdots + a_0 y = 0 \tag{1}$$

corresponds to a differential operator

$$\partial^n + a_{n-1}\partial^{n-1} + \cdots + a_0\partial^0 \tag{2}$$

acting on y. Let $C \subset \overline{\mathbb{Q}}$ be some number field (we will allow more general C in Section 7.1). The topic of this paper is finding *exponential solutions* of (1) when the a_i are in $C(x)$. By this we mean: finding all $r \in \overline{\mathbb{Q}}(x)$ for which $y = \exp(\int r)$ is a solution of (1). This is equivalent to finding *first order right hand factors* $\partial - r$ with $r \in \overline{\mathbb{Q}}(x)$ of (2). Consequently, it can be viewed as part of the more general problem of factoring differential operators. Beke gave two factoring algorithms in [Bek94] (see also [PS03, 4.1 and 4.2.1]), one algorithm for first order factors (which we will refer to as *Beke's algorithm* in this paper) and one algorithm (which uses first order factors) for higher order factors. Other factoring algorithms are the local to global method of [Hoe97a, Hoe97b] and the eigenring method ([Sin96], [PS03, 4.2.2]). Computing exponential solutions (computing first order factors) with these algorithms can be slow. It is not difficult to find examples where Maple's

[*]Supported by a grant from the Région Limousin
[†]Supported by NFS grant 0098034

expsols takes a long time, even though it uses an almost rational version of Beke's algorithm, the local to global method, as well as the eigenring method. The goal in this paper is to give a faster algorithm. We combine modular methods with an almost rational version of Beke's algorithm to obtain Algorithm ExpSolsInC which computes the exponential solutions defined over a given field C. Then we study the field problem, which is necessary for giving an efficient algorithm ExpSols for computing all exponential solutions.

We sketch Beke's method using the vocabulary of this paper. First, at each *singularity*, one computes a finite number (at most n) of objects called *generalized exponents*. Then, for each *combination* of generalized exponents (choose one generalized exponent at each singularity), the problem reduces to computing polynomial solutions (*e.g.*, with [ABP95]) of some other differential operator. There are two main problems:

- *a combinatorial problem*: one has to choose a generalized exponent at each singularity which could lead to a large number (at most n^m where m is the number of singularities) of possible combinations.

- *a field problem*: singularities and generalized exponents are often defined over algebraic extensions of C, so some combinations can be defined over large algebraic extensions of C.

If neither problem occurs then Beke's algorithm usually works well in practice. Thus we are mainly interested in situations where one or both of these two problems do occur. If we can discard combinations in advance then that would address the combinatorial problem, and in some cases even the field problem if the discarded combinations include those that are defined over large extensions of C. Our first main goal is to do this with computations modulo a prime number p.

The algebraic theory of ordinary differential equations in characteristic p is studied in [Hon81, Kat82, CC85, Put95]. For the problem of factoring differential operators with coefficients in $\overline{\mathbb{F}}_p(x)$, van der Put showed in [Put95] (this is also recalled in [Put96, 2.1] and [PS03, 13.1]) that the relevant object is the *p-curvature* since it determines the differential module mod p and all factorizations of the operator. More recently, an algorithm to factor Ore polynomials with coefficients in $\mathbb{F}_p(x)$ is included in [GZ03]. This algorithm is based on the eigenring method adapted to characteristic p and generalizes the work of [Gie98] that tackles the problem of factoring differential operators over $\mathbb{F}_q$ where $q = p^l$ for p prime and $l \in \mathbb{N}^*$. An algorithm to factor differential systems over $\overline{\mathbb{F}}_p(x)$, giving links between the p-curvature and the eigenring, is found in [Clu03].

The natural idea to lift modular factorizations to characteristic zero has been studied in [Put96, 4.3] and [PS03, 13.2.3]. In this paper we propose a different approach because there are many problems that prevent lifting from becoming an efficient and complete algorithm for characteristic zero. For example, there could be many factorizations in characteristic p, and it is not clear which of these should be lifted. Also, it is not clear how to do efficient Hensel lifting. Using more than one prime leads to the combinatorial problem of combining factors of different characteristics, and using large prime(s) makes p-curvature computations or factoring operators mod p expensive. We propose not to use factors of the operator mod p for problems in characteristic zero, but to use only the characteristic polynomial of the p-curvature.

To reduce the number of combinations to check in Beke's algorithm, we prove links between eigenvalues in $\overline{\mathbb{F}}_p(x^p)$ of the p-curvature and the reduction mod p of the generalized exponents. Algorithm CombMatchRoot determines which generalized exponents match a given eigenvalue. Algorithm ExpSolsInC uses this information to compute all exponential solutions defined over a field C (this C must be given in the input). Unfortunately, there are cases (see Section 5) where our mod p computations yield no improvements for the combinatorial problem.

The *field problem* refers to the problem that the usual Beke's algorithm computes with a generalized exponent at *every singularity simultaneously.* So the degree of the field extension of C in Beke's algorithm depends *exponentially* on m, the number of singularities (an extension of degree $\leq m!$ for the singularities followed by an extension of degree $\leq n^m$ for the generalized exponents). Of course this can easily choke the computation. Algorithm ExpSolsInC does not have this large field problem because the highest degree extensions of C it uses are those given by *just one singularity.* In fact, on complicated examples usually most of the work in ExpSolsInC consists of rational (*i.e.*, over C) computations and not of computations in field extensions. However, for ExpSolsInC to find an exponential solution y, it needs in its input a field over which y can be defined. From now on, the *field problem* refers to the problem of finding such a field without constructing very large extensions that choke the computation. Our second main goal is solving this field problem.

Our final algorithm ExpSols first calls ExpSolsInC, then uses modular information to check if more exponential solutions could exist, and if so, it will first reduce the order as much as possible before tackling the field problem. After that, it calls the recursive FindASol algorithm which solves the field problem and returns an exponential solution defined over an extension, if such solution exists. We give several bounds (one of which is based on the p-curvature) for the degree of a field extension needed to find an exponential solution. Algorithm FindASol uses these bounds and makes an algebraic extension to decrease these bounds. It does this recursively and each time it calls ExpSolsInC to search for a solution over the new field. This continues until either a solution is found or until the bound becomes 1.

We summarize the two main new results in this paper:

- We use information mod p in several ways (in Section 5 for the combinatorial problem and Section 6 for improving bounds that are used in the field problem) to speed up computation of exponential solutions in characteristic 0 if possible.

  To accomplish this there is a subtle but important difficulty to be overcome. There are many results known that hold for almost all p, but if such p can not easily be identified (an example is given in Section 7.3) then the algorithm can not use the result. *The algorithm needs to know which p it can use.* This restriction has numerous consequences, and the algorithm needs to be designed with this issue in mind. We will give a notion of good primes in Definition 3.3, and then only use results that hold for all good p. Another important issue is that the definition of good primes should discard as few primes as possible (for large p the p-curvature computation becomes slow, so only small p are useful for speeding up ExpSols).

- Our second main new result is Algorithm FindASol in Section 6.2, a recursive algorithm that constructs a field extension over which an exponential solution (if one exists) can be defined.

  An efficiency advantage of Beke's method over the local to global method in [Hoe97b] or the eigenring method in [Sin96] is that it directly computes first order factors. In contrast, the local to global method often needs to compute higher order factors before it can reach the first order factors, and the eigenring method first has to compute the eigenring. The greatest disadvantage of Beke's method in comparison to the eigenring and local to global approaches is its use of a splitting field, and this is the issue that Algorithm FindASol resolves.

The paper is organized as follows. Section 1 contains the vocabulary needed to read this paper, in particular the notions of generalized exponents, reduction mod p, and p-curvature. Section 2 illustrates in a few examples how mod p computations can reduce the combinatorial problem. In Sections 3 and 4 we give links between the p-curvature and generalized exponents. Section 5 contains Algorithm CombMatchMod$p$ that reduces the number of combinations to check and Algorithm ExpSolsInC that computes all exponential solutions defined over a given field C. Section 6 addresses the field problem. Combining these algorithms in Section 7 we give Algorithm ExpSols that computes a basis of all exponential solutions, up to conjugation over C.

1 Preliminaries

In this section we briefly recall the concepts that are needed to read the next sections. The first subsection contains the necessary material in characteristic zero, the second subsection deals with characteristic p, and the third deals with reduction from characteristic 0 to characteristic p.

1.1 Differential operators in characteristic zero

Let $\overline{\mathbb{Q}}$ denote the algebraic closure of $\mathbb{Q}$. We endow the fraction field $\overline{\mathbb{Q}}(x)$ of the ring $\overline{\mathbb{Q}}[x]$ of polynomials in the indeterminate x with the usual differentiation $' := \frac{d}{dx}$. The *field of constants* of *the differential field* $(\overline{\mathbb{Q}}(x),')$ is then $\overline{\mathbb{Q}}$. Consider the non-commutative ring $\overline{\mathbb{Q}}(x)[\partial]$ of differential operators over the differential field $(\overline{\mathbb{Q}}(x),')$. In this ring the multiplication is given by $\forall u \in \overline{\mathbb{Q}}(x)$, $\partial u = u\partial + u'$. After multiplying if necessary by some element of $\overline{\mathbb{Q}}(x)$, a non-zero element $L \in \overline{\mathbb{Q}}(x)[\partial]$ can be written as

$$L = a_n\partial^n + a_{n-1}\partial^{n-1} + \ldots + a_0\partial^0,\ a_i \in \overline{\mathbb{Q}}[x],\ a_n \neq 0 \text{ and } \gcd(a_0, \ldots, a_n) = 1. \tag{3}$$

As usual, n will be called *the order* of the differential operator L.

1.1.1 Singularities and generalized exponents

We start with some classical definitions that can be found for example in [Inc26].

Definition 1.1. *Let $L \in \overline{\mathbb{Q}}(x)[\partial]$ be written as in (3). A point $x_0 \in \overline{\mathbb{Q}}$ is said to be* a singular point (or a singularity) of L *if it is a root of a_n. Otherwise, it is said to be* a regular (or ordinary) point of L.

In this paper we always consider the point at infinity as a singular point, whether it is actually singular or not. To each differential operator $L \in \overline{\mathbb{Q}}(x)[\partial]$, we associate the subset $\mathcal{S}$ of $\overline{\mathbb{Q}}$ containing all finite singularities. The set of all singularities of L is then $\overline{\mathcal{S}} := \mathcal{S} \cup \{\infty\}$.

Let $\delta := x\partial$ and remark that $\overline{\mathbb{Q}}(x)[\partial] = \overline{\mathbb{Q}}(x)[\delta] \subset \overline{\mathbb{Q}}((x))[\delta]$.

Definition 1.2. *Let L be a non-zero differential operator in $\overline{\mathbb{Q}}((x))[\delta]$. L can be written $\sum_{i=v}^{\infty} x^i p_i(\delta)$ for some polynomials p_i over $\overline{\mathbb{Q}}$ and $p_v \neq 0$. Then* the indicial equation of L *is the polynomial p_v. Its roots in $\overline{\mathbb{Q}}$ are called* the exponents of L at $x = 0$ *(see [Inc26, 7.21] or [HW97, Lemma 11]).*

The notion of *exponents* can be generalized in the sense of [HW97, 3.2]: there are several distinct ways to define *generalized exponents*. We choose here the algebraic definition of [Hoe97a, Hoe97b] that is easy to extend to characteristic p. We first give the definition at the point $x = 0$ and then extend it to all points. At a point $x_i \in \overline{\mathbb{Q}} \cup \{\infty\}$, we define the *local parameter* t_i by

$$t_i := \begin{cases} x - x_i & \text{if } x_i \in \overline{\mathbb{Q}}, \\ 1/x & \text{if } x_i = \infty. \end{cases}$$

Definition 1.3. *Let $e \in \overline{\mathbb{Q}}[x^{-\frac{1}{k}}]$ with k minimal in $\mathbb{N}^*$ and let $L_{\delta \to \delta+e}$ denote the differential operator $L \in \overline{\mathbb{Q}}(x)[\delta]$ in which we have replaced δ by $\delta + e$. Then e is said to be* a generalized exponent of L at $x = 0$ *if* 0 *is a root of the indicial equation of $L_{\delta \to \delta+e}$. The number k is called* the ramification (index) of e. *The multiplicity of the root* 0 *in the indicial equation is called* the multiplicity of e. *If $k = 1$ then e is called an* unramified generalized exponent. *If $k = 1$ and L has no generalized exponent $\tilde{e}$ for which $\tilde{e} - e$ is a negative integer then e is called a* $\mathbb{Z}$-minimal generalized exponent.

Definition 1.4. *Let $e \in \overline{\mathbb{Q}}[t_i^{-\frac{1}{k}}]$. Then e is said to be* a generalized exponent of L at $x = x_i$ *if $e_{t_i \to x}$ (e in which we have replaced t_i by x) is a generalized exponent of $L_{t_i \to x}$ at $x = 0$. Here $L_{t_i \to x}$ refers to the automorphism of $\overline{\mathbb{Q}}(x)[\partial]$ given by $x \mapsto x + x_i$ at a finite point and $x \mapsto 1/x$ at infinity. This leads to $\partial \mapsto \partial$ at a finite point and $\partial \mapsto -x^2\partial$ at infinity.*

Generalized exponents are useful for computing exponential solutions, see Lemma 1.8 and Remark 1.10. At each point x_i, there are exactly n generalized exponents counted with multiplicity. The point x_i is called a *regular singular* point of L when all generalized exponents are constants, in which case they are just the usual exponents. At a regular point x_i the (generalized) exponents are $0, 1, \ldots, n-1$. For further details on generalized exponents see [Hoe97a, Hoe97b, HW97].

1.1.2 Exponential solutions and first order right hand factors

Definition 1.5. *Let $L \in \overline{\mathbb{Q}}(x)[\partial]$ and let $y \neq 0$ belong to a differential field extension of $\overline{\mathbb{Q}}(x)$ with $\overline{\mathbb{Q}}$ as field of constants. Assume that $L(y) = 0$.*
Then, y is said to be an exponential solution of L *if $y'/y \in \overline{\mathbb{Q}}(x)$. It is said to be* a radical solution *if there exists a positive integer m such that $y^m \in \overline{\mathbb{Q}}(x)$ and finally, it is said to be* a rational *(respectively* polynomial*)* solution *if $y \in \overline{\mathbb{Q}}(x)$ (respectively $y \in \overline{\mathbb{Q}}[x]$).*

An exponential solution y of L corresponds to the first order right hand factor $\partial - y'/y$ of L.

Remark 1.6. *In this paper it will not be necessary to distinguish an exponential function y from a scalar multiple of y. Hence we may represent y by its logarithmic derivative $y'/y \in \overline{\mathbb{Q}}(x)$ or by its minimal annihilating operator $\partial - y'/y$.*

Definition 1.7. *Let $L \in \overline{\mathbb{Q}}(x)[\partial]$ be of order $n > 1$. $R \in \overline{\mathbb{Q}}(x)[\partial]$ of order $0 < m < n$ is said to be a* right hand factor of L *if there exists $l \in \overline{\mathbb{Q}}(x)[\partial]$ such that $L = lR$.*

Lemma 1.8 below (analogous to [Bou00, Proposition 8]) shows that the notions of generalized exponent, exponential solution and first order right hand factor are closely related.

Lemma 1.8. *Let $r \in \overline{\mathbb{Q}}(x)$. If $\partial - r$ is a first order right hand factor of $L \in \overline{\mathbb{Q}}(x)[\partial]$, then for all x_i in $\overline{\mathcal{S}}$, there exists a generalized exponent $e_{x_i} \in \overline{\mathbb{Q}}[t_i^{-1}]$ of L at x_i such that*

$$r = S + \frac{Q'}{Q} \quad \textit{where} \quad S = \sum_{x_i \in \mathcal{S}} \frac{e_{x_i}}{t_i} - t_\infty e_\infty^*,$$

and (Fuchs' relation)

$$\deg(Q) + \sum_{x_i \in \overline{\mathcal{S}}} \mathrm{Const}(e_{x_i}) = 0, \quad \text{in particular} \quad -\sum_{x_i \in \overline{\mathcal{S}}} \mathrm{Const}(e_{x_i}) \in \mathbb{N} \tag{4}$$

where $Q \in \overline{\mathbb{Q}}[x]$, the $\mathrm{Const}(e_{x_i})$ *are the constant terms of the e_{x_i} and $e_\infty^* := e_\infty - \mathrm{Const}(e_\infty)$.*

Proof. If $L = \sum a_i \partial^i$ then denote $L_{\partial \to \partial + S} = \sum a_i (\partial + S)^i$. Let e_{x_i} be the generalized exponent of $\partial - r$ at x_i. Since $\partial - r$ is a right hand factor of L, this is also a generalized exponent of L (see [Hoe97b, Lemma 3.2]). Now $L_{\partial \to \partial + S}$ has right hand factor $\partial - \tilde{r}$ where $\tilde{r} = r - S$. Let $\mathcal{S}'$ be the set of all finite singularities of $\partial - r$ and let $\Delta = \mathcal{S}' \setminus \mathcal{S}$. So $x_i \in \Delta$ if and only if x_i is a regular point of L (hence $e_{x_i} \in \mathbb{N}$) but a singular point (hence $e_{x_i} \neq 0$) of $\partial - r$. Now $r = \sum_{x_i} \frac{e_{x_i}}{t_i} - t_\infty e^*_\infty$ if we take the sum over all finite singularities of $\partial - r$. Hence $\tilde{r} = \sum_{x_i \in \Delta} \frac{e_{x_i}}{t_i}$. Then $\partial - \tilde{r}$ has a polynomial solution $Q = \prod_{x_i \in \Delta} t_i^{e_{x_i}}$. Fuchs' relation (which applied to $\partial - r$ is just a reformulation of the fact that the sum of the residues of r is zero) says that if we sum $\mathrm{Const}(e_{x_i})$ over all singularities of $\partial - r$ then the result is zero, $\sum_{x_i \in \overline{\mathcal{S}} \cup \Delta} \mathrm{Const}(e_{x_i}) = 0$. Then Equation (4) follows from the observation that $\deg(Q) = \sum_{x_i \in \Delta} e_{x_i}$. □

Remark 1.9. *In our algorithm we only need to compute* unramified *generalized exponents of L, those are the only ones that can be* relevant *for our algorithm.*

If at a singularity $x_i \in \mathcal{S}$ there are two generalized exponents $e_{x_i,1}$ and $e_{x_i,2}$ whose difference $d = e_{x_i,1} - e_{x_i,2}$ is a positive integer, then $e_{x_i,1}$ is not relevant *in the sense that the lemma stays true if we never use $e_{x_i,1}$ (because replacing $e_{x_i,1}$ and Q by $e_{x_i,2}$ and $t_i^d Q$ does not change $S + Q'/Q$). So, in our algorithm, we will only use* $\mathbb{Z}$-minimal *generalized exponents to construct r.*

The r from the previous lemma matches the exponential solution

$$\exp\left(\int S\right) Q.$$

If all e_{x_i} are rational numbers then this exponential solution can be written as

$$Q \prod_{x_i \in \mathcal{S}} t_i^{e_{x_i}} \tag{5}$$

which is a radical solution. So the radical solutions are those exponential solutions for which all associated generalized exponents are rational numbers.

Remark 1.10. *If we only know the generalized exponents defining some exponential solution modulo rational numbers (respectively modulo integers), then finding such exponential solution reduces to finding radical solutions (respectively rational solutions) of $L_{\partial \to \partial + S}$.*

1.2 Differential operators in characteristic p

Let $\overline{\mathbb{F}}_p$ denote the algebraic closure of the finite field $\mathbb{F}_p$. We define the ring $\overline{\mathbb{F}}_p(x)[\partial]$ of differential operators in the same way as in characteristic zero. The main difference is that the field of constants of $(\overline{\mathbb{F}}_p(x), ')$ is $\overline{\mathbb{F}}_p(x^p)$ (see [Put01] or [GZ03, Lemma 3.3]).

1.2.1 The p-curvature and the map $y \mapsto y^{(p-1)} + y^p$

The central tool for studying differential operators in characteristic p is the p-curvature; we recall a definition and we refer to [Kat82, CC85] and [PS03, 13.2.2] for more information. The differential field $\overline{\mathbb{F}}_p(x)$ is a finite dimensional vector space over its field of constants $\overline{\mathbb{F}}_p(x^p)$:

$$\overline{\mathbb{F}}_p(x) = \bigoplus_{i=0}^{p-1} \overline{\mathbb{F}}_p(x^p) x^i.$$

To any differential operator $L \in \overline{\mathbb{F}}_p(x)[\partial]$, one can associate the differential module

$$\mathcal{M}_L := \overline{\mathbb{F}}_p(x)[\partial]/\overline{\mathbb{F}}_p(x)[\partial]L.$$

This module is equipped with a $\overline{\mathbb{F}}_p(x^p)$-linear map $\tilde{\partial}$ (coming from the left multiplication by ∂ on $\overline{\mathbb{F}}_p(x)[\partial]$) satisfying

$$\tilde{\partial} f m = f' m + f \tilde{\partial} m \quad \text{for all} \;\; m \in \mathcal{M}_L, \; f \in \overline{\mathbb{F}}_p(x).$$

Definition 1.11. *Let $L \in \overline{\mathbb{F}}_p(x)[\partial]$. The* p-curvature *of L is the $\overline{\mathbb{F}}_p(x)$-linear map $\tilde{\partial}^p$ acting on the differential module $\mathcal{M}_L$ associated to L.*

An algorithm to compute the p-curvature matrix is given in [PS03, 13.2.2]. The map

$$\begin{array}{rccc} \tau: & \overline{\mathbb{F}}_p(x) & \to & \overline{\mathbb{F}}_p(x^p), \\ & y & \mapsto & y^{(p-1)} + y^p \end{array}$$

is very useful for order one right hand factors in characteristic p. Details can be found in [[PS03], 13.2.1] and [Put95]; we only recall a lemma that we will often use:

Lemma 1.12 ([Put95], Lemma 1.4.2). *The map $\tau : \overline{\mathbb{F}}_p(x) \to \overline{\mathbb{F}}_p(x^p)$ is a surjective additive map with kernel $\{ \frac{z'}{z} \mid z \in \overline{\mathbb{F}}_p(x)^* \}$.*

In [Put95], van der Put proves that for the problem of factoring a differential operator in characteristic p (or in an equivalent way its associated differential module $\mathcal{M}_L$), the p-curvature is the relevant object since its characteristic properties give all possible factorizations of the operator.

We will note $\chi_p(L)$ the characteristic polynomial of the p-curvature of $L \in \overline{\mathbb{F}}_p(x)[\partial]$. As we are only interested in first order factors, the only modular information that we will use are the roots in $\overline{\mathbb{F}}_p(x^p)$ (with multiplicities) of $\chi_p(L)$. We denote $\mathcal{R}(\chi_p(L))$ as the set of roots of $\chi_p(L)$ in $\overline{\mathbb{F}}_p(x^p)$.

Lemma 1.13. *Let $L \in \overline{\mathbb{F}}_p(x)[\partial]$.*
(a). *If $L = L_1 L_2$ then $\chi_p(L) = \chi_p(L_1)\chi_p(L_2)$ with $\deg(\chi_p(L_i)) = order(L_i)$.*
(b). *The map μ from the set of all first order monic right hand factors of L in $\overline{\mathbb{F}}_p(x)[\partial]$ to $\mathcal{R}(\chi_p(L))$ given by $\mu(\partial - r) := \tau(r)$ is well defined and surjective.*

Proof. The claim (a) follows from the classification (the equivalence of categories) of [Put95] and [PS03, 13.1]. That the map μ is well defined follows from the first claim and the fact that $\tau(r)$ is the p-curvature of $\partial - r$ (see [Put96, Lemma 2.2]). Surjectivity follows from the classification as well. □

1.3 Reduction modulo p and factorization

Let $v_p : \mathbb{Q} \to \mathbb{Z} \cup \{\infty\}$ denote the standard p-adic valuation on $\mathbb{Q}$. There are infinitely many ways to extend this valuation to a valuation $v_p : \overline{\mathbb{Q}} \to \mathbb{Q} \cup \{\infty\}$. We assume that one such choice is made. Let $R_p = \{a \in \overline{\mathbb{Q}} \mid v_p(a) \geq 0\}$, and consider the maximal ideal $I = \{a \in R_p \mid v_p(a) > 0\}$. Now choose an isomorphism $R_p/I \cong \overline{\mathbb{F}}_p$.

Definition 1.14. *Having fixed the above choices, we denote the image of $a \in R_p$ in $\overline{\mathbb{F}}_p$ by $a[p]$. This is called the* reduction *of a mod p.*

Note that $a[p]$, the reduction of a mod p, depends not only on a and p but depends on the choice of v_p as well. However, for compactness of notation we will only mention p even when we refer to both p and v_p. If $a \in \overline{\mathbb{Q}}$ is not in R_p then $a[p]$ is not defined. In our algorithm, we will apply the reduction map $[p] : R_p \to \overline{\mathbb{F}}_p$ to only finitely many algebraic numbers that are known in advance. Thus, it is not difficult to choose a prime p and valuation v_p in such a way that $a[p]$ is always defined for all numbers a for which the reduction map $[p]$ is used. The a's we reduce mod p are finite in number and thus they are elements of a finite extension K of $\mathbb{Q}$. There are only finitely many possible choices for v_p on K. We will assume that such p and v_p are chosen, so that all instances of the notation $a[p]$ are defined. For more details on this choice see Section 5.3.

In practice, reducing an algebraic number a mod p is done as follows: Let $a_1, \ldots, a_n$ be those algebraic numbers that have previously been reduced mod p, but take only those whose reduction mod p may appear in the same expression as $a[p]$ in the algorithm (if the reductions of $a_1, \ldots, a_n$ and some other numbers $b_1, \ldots, b_m$ are used completely independently, then the algorithm will still be valid if it computes the reductions of these two sets of numbers independently, using one valuation on $\mathbb{Q}(a_1, \ldots, a_n)$ and an independently chosen valuation on $\mathbb{Q}(b_1, \ldots, b_m)$. This way the larger field $\mathbb{Q}(a_1, \ldots, a_n, b_1, \ldots, b_m)$ does not need to be constructed). Then determine the minimal polynomial of a over $K := \mathbb{Q}(a_1, \ldots, a_n)$. Reduce this minimal polynomial mod p, then the coefficients are in a field we will denote by $K[p]$, which is defined as $\mathbb{F}_p(a_1[p], \ldots, a_n[p])$. Factor this polynomial over $K[p]$, and choose one of the irreducible factors. Then determine a field extension of $K[p]$ in which this irreducible factor has a root, and let $a[p]$ be that root.

We will apply the reduction map $[p]$ to the singularities of L, which are in $\overline{\mathbb{Q}} \cup \{\infty\}$. If $a = \infty$ then we define $a[p]$ as ∞. Two distinct singularities of L can have the same image under $[p]$, in which case we say that p is not a good prime (see condition $(C3)$ in Definition 3.3 in Section 3). Note that this definition of good prime depends not only on p, but depends on v_p as well (recall that when we refer to both p and v_p we will only mention p for brevity).

If a is some object defined over $\overline{\mathbb{Q}}$ (like L, a local parameter t_i, or a generalized exponent e_{x_i}) we can define $a[p]$ by applying the reduction map $[p]$ to all coefficients.

We show now that without too many assumptions on p a factorization of a differential operator in characteristic zero can be reduced mod p. This result can be found in [Put96].

Proposition 1.15. *Let $L \in \overline{\mathbb{Q}}(x)[\partial]$ and let p be such that L can be reduced mod p. If $L = L_1L_2$ then, after possibly replacing L_1 and L_2 by cL_1 and $\frac{1}{c}L_2$ for some constant $c \in \overline{\mathbb{Q}}$, we have $L[p] = L_1[p]\ L_2[p]$.*

We allow to replace L_1 and L_2 by cL_1 and $\frac{1}{c}L_2$ for some constant c to avoid pathologies of the form $L_1 = \frac{1}{p}\partial$ and $L_2 = p\partial$.

Proof. The only thing to prove is that cL_1 and $\frac{1}{c}L_2$ can be reduced mod p for some constant c. Any p-adic valuation v_p on $\overline{\mathbb{Q}}$ can be extended to $\overline{\mathbb{Q}}[x]$ (this statement is a form of Gauss' lemma. To define $v_p(f)$ for a non-zero polynomial one takes the smallest valuation of the coefficients). The valuation can then be extended to the fraction field $\overline{\mathbb{Q}}(x)$, and then be extended to $\overline{\mathbb{Q}}(x)[\partial]$ (again, take the minimal valuation of the coefficients). Then $L[p]$ is defined if and only if $v_p(L) \geq 0$. Let c be a constant with valuation $-v_p(L_1)$, so $v_p(cL_1) = 0$, hence cL_1 can be reduced mod p. Then $v_p(\frac{1}{c}L_2) = v_p(cL_1) + v_p(\frac{1}{c}L_2) = v_p(cL_1\frac{1}{c}L_2) = v_p(L_1L_2) = v_p(L) \geq 0$, so $\frac{1}{c}L_2$ can be reduced mod p as well. $\square$

One obtains the following two criteria (these can also be easily deduced from [Put95]):

Corollary 1.16. *Let $L \in \overline{\mathbb{Q}}(x)[\partial]$ and let p be a prime such that L can be reduced mod p. Assume further that the order n of L does not drop after the reduction.*
If $\chi_p(L)$ is irreducible over $\overline{\mathbb{F}}_p(x^p)$ then L is irreducible over $\overline{\mathbb{Q}}(x)$.
If $\chi_p(L)$ has no roots in $\overline{\mathbb{F}}_p(x^p)$ then L has no exponential solutions.

Proof. It follows directly from the previous proposition, [Clu03, Theorem 4.1-Step 2] and Lemma 1.13(b). □

As an example, let $L = (x^2+x+8)\partial^2 + (-x^8+x+6)\partial + 1$ and $p = 3$. Then $\chi_p(L)$ has no roots and this is a very fast way to show that L has no exponential solutions. Without modular methods this would have taken much longer because one of the degree bounds (one ends up searching for polynomial solutions of some other operators) turns out to be very high in this example.

In the corollary, the hypothesis that the order does not drop can not be omitted. For example, the operator $(p\partial + 1)\partial$ is reducible in characteristic 0 but its reduction $L[p] = \partial$ is irreducible.

2 Examples

We will illustrate our modular improvements for the combinatorial problem (see the introduction) with a few examples. These improvements rely on Proposition 1.15 and Lemma 1.13(b) in the following way. A first order factor in characteristic zero leads to a first order factor in characteristic p by Proposition 1.15, which in turn corresponds to a root of $\chi_p(L)$ by Lemma 1.13(b).

2.1 Example 1

Consider the following differential operator:

$$L = \partial^3 - \frac{2x^2 - x + 4}{2x^2}\partial^2 - \frac{3x^3 - 4x^2 - 3x - 2}{2x^4}\partial + \frac{2x^3 - 3x - 2}{2x^4}.$$

The singularities are 0 and ∞. Generalized exponents are 0, $\frac{5}{2} + \frac{1}{x}$, $2 + \frac{1}{x}$ at 0 and $-t_\infty^{-1}$, $-t_\infty^{-\frac{1}{2}}$, $t_\infty^{-\frac{1}{2}}$ at ∞ (recall that $t_\infty = \frac{1}{x}$). The usual Beke's algorithm will have three combinations to check since we only have one unramified generalized exponent at ∞, and three at 0. We will use this example to illustrate our modular method, even though our method generally does not speed up trivial examples (L is trivial for Beke's algorithm because the number of combinations is very small and there is no field problem).

Let $p = 3$ and note $c = x^p$. The characteristic polynomial $\chi_p(L)$ of the p-curvature matrix is $(\lambda^2 + \lambda/c^2 + 2/c + 1/c^4)(\lambda + 2)$ and has only the root $\lambda = 1$. We will see in Section 5.2 that this root excludes the choices $\frac{5}{2} + \frac{1}{x}$ and $2 + \frac{1}{x}$ at 0; consequently, there remains only one combination to check.

For each of the 3 combinations, Beke's algorithm will check Fuchs' relation (Equation (4) in Lemma 1.8). This check will be very fast because since there is no field problem in this example, we do not have to construct a complicated number field in order to do this check. One of the three combinations passes this check, and it yields a solution. In this example, the modular method would not provide a speedup because it discarded only two combinations (which otherwise would have been discarded anyway by Fuchs' relation). In the next example, our modular method discards more combinations, and moreover, the discarded combinations include all "hard" combinations, *i.e.*, those not invariant under the Galois group of $\overline{\mathbb{Q}}$ over $\mathbb{Q}$.

2.2 Example 2

Let

$$L = 9(x^3-2)^5\partial^3 + (x^3-2)(2x^{10}-12x^7+108x^5+24x^4-216x^2-16x-9)\partial$$
$$-2x(190x^6-274x^3-27x-212).$$

There are three finite singularities; the roots of x^3-2, with generalized exponents 2, $\frac{1}{36}\frac{\alpha^2}{(x-\alpha)}-\frac{1}{18}\alpha$ and $-\frac{1}{36}\frac{\alpha^2}{(x-\alpha)}+4+\frac{1}{18}\alpha$ where $\alpha := \text{RootOf}(x^3-2)$. At infinity, the generalized exponents are $0, -\frac{5}{3}, -\frac{4}{3}$. So the usual Beke's algorithm will have 81 combinations to try. Let $p=5$ and look at the reduction mod p of the singularities and generalized exponents. First, we factor $x^3-2=(x+2)(x^2+3x+4)$ mod p, so α reduces to 3 as well as to $\beta := \text{RootOf}(x^2+3x+4)$ in characteristic p. After reduction mod p the situation is then as follows: we have the singularity 3 with generalized exponents 2, $\frac{4}{x+2}-1$, $-\frac{4}{x+2}$ and the singularities β with generalized exponents 2, $\frac{\beta^2}{(x-\beta)}+3\beta$, $-\frac{\beta^2}{(x-\beta)}+4+2\beta$. At infinity the generalized exponents become 0 (with multiplicity 2) and 2. Now, we compute $\chi_p(L)$ and see that its only root is $\lambda = s$ where $s := \frac{4c^4+2c+1}{c^6+c^3+4}$ and $c=x^p$. Our method consists then in finding which generalized exponents reduced mod p could match such a root s. Here, the result is that s can only correspond to the choice $\frac{4}{x+2}-1$ at 3, $\frac{\beta^2}{(x-\beta)}+3\beta$ at β and any choice at infinity. This means that in characteristic zero, a possible exponential solution must involve the generalized exponent $\frac{1}{36}\frac{\alpha^2}{(x-\alpha)}-\frac{1}{18}\alpha$ at α. This reduces the number of possible combinations from 81 to 3. Moreover, we have eliminated all "hard" combinations, *i.e.,* those for which the sum S in Lemma 1.8 would not be in $\mathbb{Q}(x)$.

If two generalized exponents differ by a rational number, then our mod p computation can not detect which one to take, which is an important limitation. This is why three combinations remained after the mod p computation even though there was only one root of $\chi_p(L)$. Of the three remaining combinations only one satisfies Fuchs' relation, and this combination yields an exponential solution.

2.3 Example 3

Let p be a prime number and consider the operator $L = x^4\partial^2 - px\partial - 2x - 1$. Then p is a good prime in the sense of Definition 3.3 below. However, despite the fact that $L[p]$ exists and can even be factored, neither of the two generalized exponents $3+1/p+p/x^2$ and $-1/p$ of L at $x=0$ can be reduced mod p. So both can be discarded by Lemma 3.4 and hence L has no exponential solutions.

3 Roots of $\chi_p(L)$ and τ-reduced terms

Definition 3.1. *If r is an object in characteristic zero we set $\tau(r) := \tau(r\,[p])$.*
Let $e_{x_i} \in \overline{\mathbb{Q}}[t_i^{-1}]$ be an unramified generalized exponent of $L \in \overline{\mathbb{Q}}(x)[\partial]$ at a point $x_i \in \overline{\mathbb{Q}} \cup \{\infty\}$. Let p be such that x_i and e_{x_i} can both be reduced mod p. The τ-reduced term *corresponding to e_{x_i} is then defined as*

$$\tau_{red}(e_{x_i}) := \begin{cases} \tau(\frac{e_{x_i}}{t_i}) & \textit{if} \quad x_i \in \overline{\mathbb{Q}}, \\ -\tau(t_\infty e_\infty^*) & \textit{if} \quad x_i = \infty \end{cases}$$

where e_∞^* denotes e_∞ without its constant term.

Remark 3.2. *If $e \in \mathbb{Q}$ then $e[p] \in \mathbb{F}_p$ and Lemma 1.12 implies $\tau_{red}(e) = 0$.*

Definition 3.3. *A prime p is a* good prime *for L if the following conditions hold:*
$(C1)$- *The coefficients of L can be reduced* mod p *and the order of L does not decrease after reduction* mod p, i.e., $a_n[p] \neq 0$.
$(C2)$- *All singularities can be reduced* mod p.
$(C3)$- *Distinct singularities stay distinct after reduction* mod p.

If L is written in the form (3), then Condition $(C2)$ simply means "$\deg_x(a_n[p]) = \deg_x(a_n)$". And $(C3)$ means that the largest square-free factor of a_n remains square-free mod p. Checking if a prime p is good is fairly easy in practice. All three conditions are useful in our algorithm, for example, without $(C3)$ we would have to replace Algorithm CombMatchRoot in Section 5.2 by a more complicated algorithm. The lemma below uses $(C3)$ as well. Nevertheless, if p is not a good prime, then it is often still possible to obtain some useful information from a computation mod p, but we will not detail this.

Lemma 3.4. *Let $L \in \overline{\mathbb{Q}}(x)[\partial]$ and p be a good prime for L. If $\partial - r$ with $r \in \overline{\mathbb{Q}}(x)$ is a monic first order right hand factor of L and if $e_{x_i} \in \overline{\mathbb{Q}}[t_i^{-1}]$ is the generalized exponent of $\partial - r$ at x_i, then e_{x_i} can be reduced* mod p.

Proof. From Lemma 1.8, there exists a generalized exponent $e_{x_i} \in \overline{\mathbb{Q}}[t_i^{-1}]$ of L at each singularity x_i such that

$$r = \sum_{x_i \in \mathcal{S}} \frac{e_{x_i}}{t_i} - t_\infty e_\infty^* + \frac{Q'}{Q}, \tag{6}$$

where $Q \in \overline{\mathbb{Q}}[x]$ and e_∞^* is e_∞ without its constant term. Since p satisfies $(C1)$, r can be reduced mod p (see Proposition 1.15). Now Q is defined up to a multiplicative constant (which we can choose in such a way that $Q[p]$ is defined and is not zero), so Q'/Q can be reduced mod p and hence the remaining part $\sum_{x_i \in \mathcal{S}} e_{x_i}/t_i - t_\infty e_\infty^*$ can be reduced mod p as well. Conditions $(C2)$+$(C3)$ in Definition 3.3 imply that we can recover $e_\infty^*[p]$ and $e_{x_i}[p]$ for $x_i \in \mathcal{S}$ from $(\sum_{x_i \in \mathcal{S}} e_{x_i}/t_i - t_\infty e_\infty^*)[p]$ and therefore $e_{x_i}[p]$ is well defined. Then the same is true for $e_\infty[p]$ by Fuchs' relation. □

We now give two new results linking the τ-reduced terms and the eigenvalues of the p-curvature in $\overline{\mathbb{F}}_p(x^p)$, *i.e.*, the elements of $\mathcal{R}(\chi_p(L))$. First another definition:

Definition 3.5. *Let $L \in \overline{\mathbb{Q}}(x)[\partial]$. We say that $s \in \mathcal{R}(\chi_p(L))$* comes from characteristic zero *if there exists $r \in \overline{\mathbb{Q}}(x)$ such that $\partial - r$ is a right hand factor of L in characteristic zero and $s = \tau(r)$.*

Proposition 3.6. *Let $L \in \overline{\mathbb{Q}}(x)[\partial]$ and let p be a good prime for L. If $s \in \mathcal{R}(\chi_p(L))$ comes from characteristic zero, then $\forall\, x_i \in \overline{\mathcal{S}}$, there exists a generalized exponent e_{x_i} at x_i such that:*

$$s = \sum_{x_i \in \overline{\mathcal{S}}} \tau_{red}(e_{x_i}).$$

Proof. By definition, $s \in \mathcal{R}(\chi_p(L))$ comes from characteristic zero means that $s = \tau(r)$ for some $r \in \overline{\mathbb{Q}}(x)$ where $\partial - r$ is a right hand factor of L. From Lemma 1.8, there exists then a generalized exponent $e_{x_i} \in \overline{\mathbb{Q}}[t_i^{-1}]$ of L at each singularity x_i such that r can be written in the form (6) where $Q \in \overline{\mathbb{Q}}[x]$ and e_∞^* is e_∞ without its constant term. Since p is a good prime for L and Q is defined up to a multiplicative constant (which we can choose in such a way that $Q[p]$ is defined and is not zero), the reduction mod p of each term in Equation (6) is well defined (see also Lemma 3.4 above). Now using that $s = \tau(r)$, the result follows directly from Lemma 1.12 and the definition of the $\tau_{red}(e_{x_i})$. □

Proposition 3.7. *For almost all primes p the condition that s comes from characteristic 0 may be omitted.*

Proof. As we will not need this result for our algorithm, we will only sketch a proof. First we note that generalized exponents can be defined in characteristic p. Definitions 1.3 and 1.4 are completely algebraic: the operations required are additions and multiplications in $\overline{\mathbb{Q}}(x)$, tests to zero in $\overline{\mathbb{Q}}$ and computing the roots of a polynomial in $\overline{\mathbb{Q}}[x]$. Obviously these operations also make sense in characteristic p so the definitions are easily translated to characteristic p provided that the ramification k is not divisible by p (this is not an issue because we only consider $k = 1$ in this paper). The generalized exponents are here defined mod p, which is coherent with the fact that we want t_i^p to have generalized exponent 0 at x_i (because t_i^p is in the field of constants) as well as generalized exponent p at x_i because it vanishes with multiplicity p at x_i. Note that the notions of singularity, indicial equation and exponents in characteristic p have already been introduced in [Hon81] and [CC85].

Lemma 1.8 can also be adapted to characteristic p (the lemma is just a statement about the relation between a rational function and its local expansions). Then one can also give the analogue of Proposition 3.6 in characteristic p, with $L \in \overline{\mathbb{F}}_p(x)[\partial]$, with the x_i in $\overline{\mathbb{F}}_p \cup \{\infty\}$, the unramified generalized exponents e_{x_i} in $\overline{\mathbb{F}}_p[t_i^{-1}]$, but now s can be any root of $\chi_p(L)$.

Let $L \in \overline{\mathbb{Q}}(x)[\partial]$, let p be a good prime, and let s be any root of $\chi_p(L)$, not necessarily coming from characteristic 0. Applying the mod p analogue of Proposition 3.6 to $L[p]$ we see that even if s does not come from characteristic 0, the statement in Proposition 3.6 still holds if we replace the generalized exponents e_{x_i} of L by the generalized exponents of $L[p]$. This completes the proof because for all but finitely many good primes p, reduction mod p will induce a bijection between the (unramified) generalized exponents of L and those of $L[p]$. □

One obtains the following corollary which is related to the converse of [Hon81, Theorem 2] and [CC85, Corollary 1.4]. Indeed, their statement (attributed to Katz) is: if $L[p]$ has "sufficiently many solutions in a weak sense", which is equivalent to "L has nilpotent p-curvature", for almost all p, then all exponents of L are rational numbers.

Corollary 3.8. *Let $L \in \overline{\mathbb{Q}}(x)[\partial]$. If all generalized exponents of L are rational numbers, then for almost all primes, $\mathcal{R}(\chi_p(L)) \subseteq \{0\}$.*

Proof. If $e_{x_i} \in \mathbb{Q}$ then $\tau_{red}(e_{x_i}) = 0$ and the result follows from Proposition 3.7. □

4 Exponential solutions and roots of $\chi_p(L)$

4.1 Classes of exponential solutions

Definition 4.1. *The* class $\overline{e_{x_i}}$ *of a generalized exponent e_{x_i} is $\overline{e_{x_i}} := e_{x_i} \mod \mathbb{Q}$, the equivalence class of e_{x_i} modulo $\mathbb{Q}$.*

Two generalized exponents e_{x_i} and e'_{x_i} that have the same class $\overline{e_{x_i}}$ and that can both be reduced mod p satisfy $\tau_{red}(e_{x_i}) = \tau_{red}(e'_{x_i})$ by Remark 3.2. So we can define $\tau_{red}(\overline{e_{x_i}}) := \tau_{red}(e_{x_i})$.

Definition 4.2. *Let y be an exponential solution of L, let $\partial - r$ be the associated first order right hand factor of L (so $r = y'/y$) and let e_{x_i} be the associated generalized exponents in the sense of Lemma 1.8. Then the* class $\mathrm{Cl}(y)$ *of y is defined as the image of y'/y in $\overline{\mathbb{Q}}(x)/\sim$ where*

$a \sim b \Leftrightarrow a - b = \frac{g'}{g}$ *for some* radical *function* $g \neq 0$. *Viewing* $\overline{e_{x_i}}/t_i$ *as elements of* $\overline{\mathbb{Q}}(x)/\!\sim$ *we can write*

$$\mathrm{Cl}(y) \;=\; \frac{y'}{y} \bmod \sim \;=\; \sum_{x_i \in \mathcal{S}} \frac{\overline{e_{x_i}}}{t_i} - t_\infty e^*_\infty.$$

Two exponential solutions have the same class if and only if their quotient is a radical function, if and only if at every point, the associated generalized exponents have the same class. Now, to $L \in \overline{\mathbb{Q}}(x)[\partial]$ we associate the *class-set*

$$\mathrm{Cl}(L) := \{\, \mathrm{Cl}(y) \in \overline{\mathbb{Q}}(x)/\!\sim \;\mid y \text{ exponential solution of } L \,\}.$$

4.2 Links between $\mathrm{Cl}(L)$ and $\mathcal{R}(\chi_p(L))$

Lemma 4.3. *Let* $L \in \overline{\mathbb{Q}}(x)[\partial]$ *and* p *be a good prime for* L. *There is a map*

$$\begin{aligned} \Psi : \mathrm{Cl}(L) \;&\to\; \mathcal{R}(\chi_p(L)), \\ \mathrm{Cl}(y) \;&\mapsto\; \tau(\frac{y'}{y}) = \sum_{x_i \in \overline{\mathcal{S}}} \tau_{red}(\overline{e_{x_i}}). \end{aligned}$$

Proof. Let $\mathrm{Cl}(y) \in \mathrm{Cl}(L)$. An exponential solution y corresponds to a first order right hand factor $\partial - r$ of L in characteristic zero where $r = y'/y$. Since p is a good prime for L, this factorization can be reduced mod p (see Proposition 1.15) and we have thus a first order right hand factor $\partial - r[p]$ of $L[p]$. Now from Lemma 1.13, $\tau(r) \in \mathcal{R}(\chi_p(L))$. The lemma is then clear using Proposition 3.6 since $\tau(r)$ is an element of $\mathcal{R}(\chi_p(L))$ that comes from characteristic zero. □

Given L, there are two natural questions about Ψ:
$(Q1)$- Is Ψ surjective for almost all good primes?
$(Q2)$- Does there exist a good prime p such that Ψ is injective?

Proposition 4.4. *The answer to questions* $(Q1)$ *and* $(Q2)$ *can be no.*

Proof. We give one example for each question. For $(Q1)$, if we take a differential operator that has a basis of algebraic (but no exponential) solutions, then $\mathrm{Cl}(L) = \emptyset$. The proved part of Grothendieck's conjecture (see [Kat82], [Cha01] or [Put01] for statements and histories of this conjecture) says that for almost all primes the p-curvature is zero so that $\mathcal{R}(\chi_p(L)) = \{0\}$. For $(Q2)$, take an operator L with the following 8 exponential solutions:

$$x^{e_0}(x-1)^{e_1}(x-2)^{e_2}$$

for $e_0 \in \{0, \sqrt{2}\}, e_1 \in \{0, \sqrt{3}\}$ and $e_2 \in \{0, \sqrt{6}\}$. Modulo every prime p, at least one of the three square roots becomes an element of $\mathbb{F}_p$. Suppose for example that $\sqrt{6}[p] \in \mathbb{F}_p$. Consider then the exponential solutions y_1 and y_2 corresponding to choices respectively $(0,0,0)$ and $(0,0,\sqrt{6})$ for (e_0, e_1, e_2). Then $\mathrm{Cl}(y_1) \neq \mathrm{Cl}(y_2)$ but they have the same image under Ψ. □

The previous proposition says that we can not always choose a prime number for which each root of $\chi_p(L)$ corresponds to at most one class of exponential solutions. The example illustrates the cause of this, and one can show the following (note that we do not use this in our algorithm):

Proposition 4.5. *Let* $L \in \overline{\mathbb{Q}}(x)[\partial]$. *If there exists a good prime* p *for* L *for which:*

$$(*)\; \forall\, x_i \in \overline{\mathcal{S}},\; e_{x_i,2} - e_{x_i,1} \notin \mathbb{Q} \Rightarrow e_{x_i,2}[p] - e_{x_i,1}[p] \notin \mathbb{F}_p$$

then, for that prime, Ψ *is injective.*

Proof. Suppose that $\mathrm{Cl}(y_1) = \sum_{x_i \in \mathcal{S}} \frac{\overline{e_{x_i,1}}}{t_i} - t_\infty e^*_{\infty,1}$ and $\mathrm{Cl}(y_2) = \sum_{x_i \in \mathcal{S}} \frac{\overline{e_{x_i,2}}}{t_i} - t_\infty e^*_{\infty,2}$ are two distinct elements of $\mathrm{Cl}(L)$ and suppose that $\Psi(\mathrm{Cl}(y_1)) = \Psi(\mathrm{Cl}(y_2))$.

$$\begin{aligned}
\Psi(\mathrm{Cl}(y_1)) = \Psi(\mathrm{Cl}(y_2)) \quad &\Leftrightarrow \quad \textstyle\sum_{x_i \in \overline{\mathcal{S}}} \tau_{red}(e_{x_i,1}) = \sum_{x_i \in \overline{\mathcal{S}}} \tau_{red}(e_{x_i,2}),\\
&\Leftrightarrow \quad \tau\left(\textstyle\sum_{x_i \in \mathcal{S}} \frac{e_{x_i,1}}{t_i} - t_\infty e^*_{\infty,1}\right) = \tau\left(\sum_{x_i \in \mathcal{S}} \frac{e_{x_i,2}}{t_i} - t_\infty e^*_{\infty,2}\right),\\
&\Leftrightarrow \quad \textstyle\sum_{x_i \in \mathcal{S}} \frac{e_{x_i,2} - e_{x_i,1}}{t_i} - t_\infty (e^*_{\infty,2} - e^*_{\infty,1}) = \frac{g'}{g}, \text{ for } g \in \overline{\mathbb{F}}_p(x)^*.
\end{aligned}$$

This leads to: $\forall_{x_i \in \mathcal{S}}$, $e_{x_i,2}[p] - e_{x_i,1}[p] \in \mathbb{F}_p$ and $e^*_{\infty,2}[p] - e^*_{\infty,1}[p] = 0$ which, by Fuchs' relation, implies that also $e_{\infty,2}[p] - e_{\infty,1}[p] \in \mathbb{F}_p$. On the other hand, the hypothesis that $\mathrm{Cl}(y_1) \neq \mathrm{Cl}(y_2)$ means that $\exists\, i \in \overline{\mathcal{S}}, e_{x_i,2} - e_{x_i,1} \notin \mathbb{Q}$ which contradicts condition $(*)$. □

5 Modular improvements of the combinatorial problem

Definition 5.1. *Let $L \in \overline{\mathbb{Q}}(x)[\partial]$. L is said to be* rad-regular *if there exists $r \in \overline{\mathbb{Q}}(x)$ such that $L_{\partial \to \partial + r}$ has all its unramified generalized exponents in $\mathbb{Q}$. Otherwise, L is said to be* rad-singular*; it means that there exists a singularity x_i with two unramified generalized exponents whose difference is not in $\mathbb{Q}$.*

As mentioned in the introduction, we want to address two problems in Beke's algorithm: the combinatorial problem and the field problem. In this section we focus only on the combinatorial problem and not on the field problem; we suppose that a number field C is given and develop an algorithm that finds all exponential solutions defined over C. We show how a careful modular analysis can effectively reduce the combinatorial problem if L is rad-singular. Unfortunately, in the rad-regular case no reduction in the combinatorial problem is obtained (see also Section 7.2).

5.1 Some precisions on the vocabulary

Let C be a number field. In light of Remark 1.6 we use the terminology:

Definition 5.2. *An exponential function y* is defined over C *if $y'/y \in C(x)$.*

When C is not algebraically closed, we will not consider all roots of the leading coefficient a_n of L, we will only consider one root for each irreducible factor over C. So the number of singularities that our algorithm considers is the number of irreducible factors of a_n (this number depends on the field C) plus one (the point at infinity). The only extensions of C that we will work with in Section 5 are extensions given by a single irreducible factor of a_n in $C[x]$, so Algorithm ExpSolsInC in Section 5.4 does not use computationally expensive nested extensions.

Definition 5.3. *A* place *is either a monic irreducible polynomial P_i in $C[x]$ or ∞.*
At each place $P_i \neq \infty$, we define the point *x_i as the class of x in $C[x]/(P_i)$ noted* RootOf(P_i)*. At the place ∞, the point is also ∞.*

If the degree of P_i is one, then we can view x_i as an element of C. Recall that we have the local parameter $t_i = x - x_i$ at finite points and $t_i = 1/x$ at ∞. We adapt the notion of singularity and generalized exponents to the purpose of this section:

Definition 5.4. *The C-*singularities *of $L \in C(x)[\partial]$ written as in (3) are the points at the places P_i for all monic irreducible factors P_i of a_n in $C[x]$ and the point ∞. We keep the notations $\mathcal{S}$ for the set of finite C-singularities and $\overline{\mathcal{S}} = \mathcal{S} \cup \{\infty\}$. Furthermore, a C-*generalized exponent of L at *x_i is an unramified generalized exponent of L at the C-singularity x_i that further belongs to $C(x_i)[t_i^{-1}]$. The* degree *of a C-singularity x_i is the degree of P_i if $x_i \neq \infty$, and it is 1 if $x_i = \infty$.*

A first order operator $\partial - r \in C(x)[\partial]$ has a C-generalized exponent at every C-singularity. Hence, if at a C-singularity of L there are no C-generalized exponents, then L can not have exponential solutions defined over C.

Definition 5.5. *A* C-combination *is a list composed of one C-generalized exponent e_{x_i} at each C-singularity $x_i \in \overline{\mathcal{S}}$.*

The exponential solutions that we are looking for (those defined over C) match first order right hand factors $\partial - r$ with $r \in C(x)$ as in Lemma 1.8 but here we write:

$$r = \sum_{x_i \in \mathcal{S}} \mathrm{Tr}_i(\frac{e_{x_i}}{t_i}) - t_\infty e^*_\infty + \frac{Q'}{Q} \tag{7}$$

where Tr_i is the trace over the field extension $C(x_i, x) \supset C(x)$. The term $\mathrm{Tr}_i(e_{x_i}/t_i) \in C(x)$ is the sum of all conjugates of e_{x_i}/t_i over $C(x)$. So the above r is as in Lemma 1.8 with the additional property $r \in C(x)$, which means that for any σ in the Galois group of $\overline{\mathbb{Q}}(x)$ over $C(x)$, if we choose e_{x_i} at x_i in Lemma 1.8 then we must choose $\sigma(e_{x_i})$ at $\sigma(x_i)$ (σ maps the unramified generalized exponents of $L \in C(x)[\partial]$ at x_i to those at $\sigma(x_i)$). We extend Definition 3.1 to finite C-singularities x_i as follows:

$$\tau_{red}(e_{x_i}) := \tau(\mathrm{Tr}_i(\frac{e_{x_i}}{t_i})).$$

Definition 5.6. *Let P be a problem defined over a field C of characteristic 0 for which the following holds: if s is a solution of P defined over an algebraic extension of C, then all conjugates of s over C are solutions as well. Then we say that $s_1, \ldots, s_k$ are* the solutions of P up to conjugation over C *when:*

- for each s_i a finite extension C_i of C is given such that s_i is defined over C_i and,
- for any solution s of P defined over any algebraic extension C' of C there is precisely one i such that C_i can be embedded in C' over C and s_i corresponds to s under this embedding.

We define the degree *(or* algebraic degree*) of s_i over C as $[C_i : C]$. The total number of solutions defined over $\overline{C}$ equals the sum of the degrees over C of the s_i.*

Each such problem P that we need to solve (such as: find the singularities, find the generalized exponents at a singularity x_i, find the exponential solutions) will be solved up to conjugation, this in order to prevent computationally expensive splitting fields. In this terminology, "computing the C-singularities" and "computing the singularities up to conjugation over C" have identical meaning: factor a_n over C and construct a field $C_i = C(x_i) = C[x]/(P_i)$ for each factor P_i (the singularity at infinity has degree 1 hence its field is C). Solving a problem up to conjugation over C does not involve constructing fields that contain more than one C_i, which means that we are not able to perform arithmetic between objects defined over distinct C_i's (unless of course when the degrees are 1, *i.e.*, when these C_i's are just C). That is why the e_{x_i}/t_i in equation (7) must first be "traced down" to an object defined over C before we can add them.

When we compute generalized exponents at x_i (recall that we only need the unramified generalized exponents), we compute them up to conjugation over $C(x_i)$. The C-generalized exponents are those of (algebraic) degree 1 over $C(x_i)$. Generalized exponents of degree > 1 will not be used in Section 5, they will be used in Algorithm FindASol in Section 6.2.

5.2 Finding C-combinations matching the modular information

Let $L \in C(x)[\partial]$ where C is a number field. Suppose that L has m finite C-singularities $x_1, \dots, x_m$ and consider the following set:

$$\text{Comb} := \{(\overline{e_{x_1}}, \dots, \overline{e_{x_m}}, \overline{e_\infty}) \mid e_{x_i} \text{ is a } C\text{-generalized exponent of } L \text{ at } x_i\}.$$

The elements of Comb are called *C-combinations mod $\mathbb{Q}$*. Note that the set Comb depends on C in two ways. First, the x_i correspond to irreducible factors of a_n over C. Second, the e_{x_i} must be defined over $C(x_i)$. Consider now the map

$$\begin{array}{rccc} \Phi: & \text{Comb} & \to & \overline{\mathbb{F}}_p(x^p), \\ & (\overline{e_{x_1}}, \dots, \overline{e_{x_m}}, \overline{e_\infty}) & \mapsto & \sum_{x_i \in \overline{\mathcal{S}}} \tau_{red}(\overline{e_{x_i}}). \end{array}$$

Definition 5.7. *Let $s \in \mathcal{R}(\chi_p(L))$ and $h \in$ Comb. We say that h* matches *s if $\Phi(h) = s$. We say that h* satisfies Fuchs' relation *if $\sum_{x_i \in \mathcal{S}} \mathrm{Tr}_i(\mathrm{Const}(\overline{e_{x_i}})) + \mathrm{Const}(\overline{e_\infty}) = 0 \mod \mathbb{Q}$.*

Given an element s of $\mathcal{R}(\chi_p(L))$ for a good prime p, we can effectively find the finite set of elements h of Comb that match s. A naive way to do that consists in taking the image under Φ of all possible C-combinations mod $\mathbb{Q}$ and keeping those with image s. A reason for including Condition (C3) in the definition of a good prime is so that we can give a more practical method (Algorithm CombMatchRoot below).

In the sequel, $\mathbb{F} := C[p]$ denotes the image of the number field C after reduction mod p. In practice C is given by some (possibly nested) RootOf's so that this reduction makes sense (recall that choosing a reduction $[p]$ means choosing factors mod p of the polynomials defining the RootOf's).

Since $\mathbb{F}$ is finite, the map $U \mapsto U^p$ is an isomorphism from $\mathbb{F}[x]$ to $\mathbb{F}[x^p]$, and we denote the inverse isomorphism by $V \mapsto V^{1/p}$. If $r = \frac{a}{b} + c$ with $a, b, c \in \mathbb{F}[x]$ and $\deg(a) < \deg(b)$ then $\tau(r)$ can be written as $\frac{A}{B} + C$ with $\deg(A) < \deg(B)$, $B = b^p$ (here $\gcd(A, B)$ need not be 1) and $C = \tau(c)$. So if $V \in \mathbb{F}[x^p]$ divides the denominator of $\tau(r)$ then $V^{1/p} \in \mathbb{F}[x]$ divides the denominator of r.

Algorithm CombMatchRoot
Input: a number field C, a good prime p for L, the reduction $[p]$, $s \in \mathcal{R}(\chi_p(L)) \cap \mathbb{F}(x^p)$ and the data structure of C-singularities and C-generalized exponents of L.
Output: the elements h of Comb matching s and satisfying Fuchs' relation.

```
1 - For each $x_i \in \overline{\mathcal{S}}$ and each $e_{x_i}$ at $x_i$ compute $\tau_{red}(e_{x_i})$.
2 - Compute the partial fraction decomposition $\sum_{i=1}^{N_s} \frac{U_i}{V_i^{n_i}} + W$ of $s$ with $U_i$, $W \in \mathbb{F}[x^p]$,
    $V_i$ irreducible in $\mathbb{F}[x^p]$ and $\deg(U_i) < \deg(V_i^{n_i})$.
3 - Remove all $\overline{e_\infty}$ for which $\tau_{red}(\overline{e_\infty}) \neq W$; if none remain, then return $\emptyset$ and stop.
4 - For each $i = 1 \dots N_s$,
  4a - Find the $C$-singularity $x_j = \mathrm{RootOf}(P_j)$ such that $V_i^{1/p} \in \mathbb{F}[x]$ divides $P_j[p]$,
  4b - Remove all $\overline{e_{x_j}}$ for which the partial fraction decomposition of $\tau_{red}(\overline{e_{x_j}})$ does not
       contain the term $\frac{U_i}{V_i^{n_i}}$; if none remain, then return $\emptyset$ and stop.
5 - For all remaining $\overline{e_{x_j}}$, compute all combinations $h = (\overline{e_{x_1}}, \dots, \overline{e_{x_m}}, \overline{e_\infty})$ that satisfy
    Fuchs' relation and return those as output.
```

In step 1, if the reduction mod p fails then e_{x_i} can be discarded by the same argument as in Lemma 3.4. In steps 3 and $4b$, if nothing remains, then there are no first order right hand factors

$\partial - r$ of L with $r \in C(x)$ and $s = \tau(r)$. Condition $(C3)$ in the definition of good primes is used in step $4a$ to make sure that the P_j for which $V_i^{1/p}$ divides $P_j[p]$ is unique (the existence of such a P_j follows from the fact that a pole of a root of $\chi_p(L)$ is a singularity of $L[p]$).

Remark 5.8. *It follows from Lemma 3.4 that if e_{x_i} can not be reduced mod p then it can not be* relevant *for the algorithm, it can not correspond to any first order right hand factor. It follows from Algorithm* CombMatchRoot *above that a generalized exponent that can be reduced mod p but that does not match any element of $\mathcal{R}(\chi_p(L))$ can not be* relevant *either. If, adding these notions of relevant to those of Remark 1.9 (*i.e., *unramified, $\mathbb{Z}$-minimal), no relevant generalized exponents remain at a certain singularity x_i then there are no exponential solutions and we can stop the computation. If at a certain x_i only one relevant generalized exponent remains then we call x_i a* semi-apparent singularity *because it does not contribute to the combinatorial problem. In Algorithm FindASol in Section 6 it is important that we use only relevant generalized exponents to prevent making unnecessary field extensions.*

```
Algorithm CombMatchModp:
Input: an operator $L \in C(x)[\partial]$, a number field $C$, the data structure $E$ of $C$-singularities
and $C$-generalized exponents of $L$.
Output: the set of $C$-combinations mod $\mathbb{Q}$ that match a root of $\chi_p(L)$ for a good prime $p$
and that satisfy Fuchs' relation.
1 - If at some $C$-singularity there are no $C$-generalized exponents, return $\emptyset$ and stop.
2 - Choose a good prime $p$ for $L$ (choose a reduction $[p]$).
3 - Compute $\mathcal{R}(\chi_p(L))$ and (to be used in Section 7) Nroots := the total number of roots
    of $\chi_p(L)$ in $\overline{\mathbb{F}}_p(x^p)$ counted with multiplicity.
4 - Return the union of CombMatchRoot($C$,$p$,$[p]$,$s$,$E$) for $s \in \mathcal{R}(\chi_p(L)) \cap \mathbb{F}(x^p)$.
```

One can often (use Condition $(*)$ in Proposition 4.5) find a prime such that each s matches at most one C-combination mod $\mathbb{Q}$ (*i.e.*, the set CombMatchRoot(C,p,$[p]$,s,E) has ≤ 1 elements for each s). But Proposition 4.4 shows that such prime does not always exist so that, in practice, we do not try to find one.

5.3 Some remarks on the choice of p

We can choose p as follows. Start with $p = 2$, and as long as a reduction mod p of a coefficient of L fails, or if one of the other conditions of a good prime in Definition 3.3 is not met, we take the next prime. So we compute the smallest good prime. There are, however, certain cases where another choice could be better. In hard cases (when we need to use Algorithm FindASol, or when the number of combinations is very high) we may want to try a few more primes and select the best one. In very easy examples such as in Section 2.1 it can be best to use no primes at all because the combinatorial problem was already practically trivial without modular methods. Another situation where there is no improvement in the combinatorial problem is when L is rad-regular (see Definition 5.1 and also Section 7.2) and the p-curvature computation does not exclude existence of exponential solutions. In most cases we only use the smallest good p, and in the unusual event that this p is large then we skip the modular improvements given in Section 5 because p-curvature computations become expensive for large p. Exceptions are very easy cases (use no modular methods) and very hard cases (check more than one prime and select the best one).

5.4 Finding all exponential solutions defined over a given field

Algorithm CombMatchMod$p$ computes a set of possible combinations for C-generalized exponents mod $\mathbb{Q}$ that could appear in an exponential solution defined over C. We will use this to find all exponential solutions defined over C.

Recall that with the notations of Equation (7) and Lemma 1.8, an exponential solution can be written as $\exp(\int S)\,Q$ where $S = \sum_{x_i \in \mathcal{S}} \mathrm{Tr}_i(\frac{e_{x_i}}{t_i}) - t_\infty e_\infty^*$ and Q is a polynomial. After CombMatchMod$p$, it remains to check which C-combinations mod $\mathbb{Q}$ lead to exponential solutions and to compute these exponential solutions. A natural idea is: at each C-singularity, given a $\overline{e_{x_i}}$ we take the finitely many generalized exponents e_{x_j} satisfying $e_{x_j} \mod \mathbb{Q} = \overline{e_{x_i}}$ and we have thus a finite number of C-combinations $(e_{x_1}, \ldots, e_{x_m}, e_\infty)$, hence a finite number of possibilities for S after checking Fuchs' relation. Then for each S, we construct the operator $L_{\partial \to \partial + S}$ having as solutions those of L divided by $\exp(\int S)$, and we search for polynomial solutions Q of that operator.

We define an *E-combination* as a C-combination that uses only generalized exponents that appear in our data structure E. As already mentioned in Remark 1.9, replacing e_{x_i} by $e_{x_i} - d$ and Q by $t_i^d Q$ for some $d \in \mathbb{Z}$ leaves the expression $\exp(\int S)\,Q$ invariant. So we only need one generalized exponent in each equivalence class mod $\mathbb{Z}$. To assure that Q is a polynomial we only put $\mathbb{Z}$-minimal generalized exponents in our data structure E, *i.e.*, unramified generalized exponents that are minimal in their equivalence class mod $\mathbb{Z}$. This is also true for the point at infinity; although the constant term of e_∞ does not contribute to S in Lemma 1.8, it does contribute in Equation (4) to the degree bound for Q (the number N in step 4 is used as a degree bound for the Q_i in step 4b), and using only $\mathbb{Z}$-minimal generalized exponents at infinity ensures that this degree bound is not too low.

```
**Algorithm** ExpSolsIn$C$:
**Input**: an operator $L \in C(x)[\partial]$, a number field $C$.
**Output**: Sol, a basis of exponential solutions of $L$ defined over $C$.
1 - Compute the data structure $E$ of $C$-singularities and $\mathbb{Z}$-minimal $C$-generalized exponents.
2 - $M$ := CombMatchMod$p$($L$,$C$,$E$).
3 - Sol := $\emptyset$.
4 - For each $E$-combination $(e_{x_1}, \ldots, e_{x_m}, e_\infty)$ whose class mod $\mathbb{Q}$ is in $M$, and that satisfies:
      $N := -\mathrm{Const}(e_\infty) - \sum_{x_i \in \mathcal{S}} \mathrm{Tr}_i(\mathrm{Const}(e_{x_i})) \in \mathbb{N}$, do:
    4a - Let $S := \sum_{x_i \in \mathcal{S}} \mathrm{Tr}_i(\frac{e_{x_i}}{t_i}) - t_\infty e_\infty^*$ and $\tilde{L} := L_{\partial \to \partial + S}$,
    4b - Compute a basis $Q_1, \ldots, Q_w$ of polynomial solutions in $C[x]$ for $\tilde{L}$,
    4c - If $w > 0$ then add the $\exp(\int S)\,Q_j$ to Sol.
5 - Return Sol.
```

We can pre-compute the $\mathrm{Tr}_i(\mathrm{Const}(e_{x_i}))$ and $\mathrm{Tr}_i(\frac{e_{x_i}}{t_i})$, from then on all the work in the loop in step 4 (which may dominate the computation time, see also Section 7.1) takes place over C.

To use this procedure in the recursive algorithm FindASol in Section 6.2, we need to allow two extra inputs. First, an option called "just one"; if this option is given then we only need to return just one (if it exists) exponential solution defined over C but not a totally arbitrary one. We return a solution $\exp(\int S)\,Q$ with S as in the algorithm, but with Q of minimal possible degree. Second, an optional set F may be given in the input. Each element of F is a couple (x_i, e_{x_i}) where x_i is a C-singularity of degree 1 (see Definition 5.4) and e_{x_i} is a C-generalized exponent at x_i. This couple encodes the command: "in the data structure E in step 1 of ExpSolsInC, throw away all C-generalized exponents at x_i except e_{x_i}". This reduces the number of E-combinations to be checked. Note that usually not all C-singularities will appear in F, so in general there is still a combinatorial problem. Furthermore, if $F \neq \emptyset$, then it is of course possible that no exponential solutions over C are found even if such solutions exist.

6 A way to handle the field problem

Let $L \in C(x)[\partial]$ with $C \subset \overline{\mathbb{Q}}$. We want to find a basis of all exponential solutions, so we may have to search for exponential solutions defined over algebraic extensions of C. Before constructing such extensions, we first want to bound their degrees.

6.1 Two ways to obtain bounds

We start with a few definitions, in particular we need the notion of *type*. Many useful properties of the type of a differential operator can be found in [Ore32] (where type is called Art-Begriff). We will only use the type for first order operators, or equivalently, for exponential solutions:

Definition 6.1. *Two exponential functions y_1 and y_2 are said to be* of the same type *if y_1/y_2 is a rational function. This defines an equivalence relation, and* type(y_1)*, the* type *of y_1, is defined to be the equivalence class of y_1. We say that y_1 and y_2 have the same* local type *at x_i if y_1/y_2 is meromorphic at x_i, which is equivalent to saying that the generalized exponents of $\partial - y_1'/y_1$ and $\partial - y_2'/y_2$ at x_i differ by an integer.*

We will identify an exponential solution y with its minimal operator $\partial - y'/y \in \overline{\mathbb{Q}}(x)[\partial]$, and so we shall not distinguish y from a constant multiple of y. This makes it easy to describe what we mean by a *conjugate* of y; if σ is an element of the Galois group of $\overline{\mathbb{Q}}$ over some number field C, then by $\sigma(y)$ we simply mean a non-zero solution of $\partial - \sigma(y'/y)$.

Two exponential functions have the same type if and only if they have the same local type at every point x_i. The type of the exponential solution $y = \exp(\int S)\, Q$ with S and Q as in Lemma 1.8 can be uniquely represented as $\sum_{x_i \in \mathcal{S}} \frac{e_{x_i} \mod \mathbb{Z}}{t_i} - t_\infty e_\infty^*$ where e_∞^* is e_∞ without its constant term (recall that this constant term depends on the constant terms of the other e_{x_i} by Fuchs' relation). The Galois group of $\overline{\mathbb{Q}}$ over C acts on the set of all types.

Definition 6.2. *Let C be a number field, y be an exponential function. Let K be the* field of definition *of y over C, which is defined as the smallest number field K that contains C for which $y'/y \in K(x)$. Then the* algebraic degree *of y over C is $[K : C]$. We say that y is of* minimal algebraic degree m over C *if $[K : C] = m$ and there exist no exponential function $\tilde{y}$, of the same type as y, having smaller algebraic degree over C.*

Let C be a number field, let y be an exponential function, and let K be the field of definition of y over C. Let $L_1 = \partial - y'/y \in K(x)[\partial]$. Let $L_1, \ldots, L_r \in \overline{\mathbb{Q}}(x)[\partial]$ be the conjugates of L_1 over C, and let $L = \mathrm{LCLM}(L_1, \ldots, L_r) \in C(x)[\partial]$. It is easy to calculate a basis of all exponential solutions of L that have the same type as y; compute $\tilde{L} := L_{\partial \to \partial + y'/y} \in K(x)[\partial]$, compute a basis $b_1, \ldots, b_k \in K(x)$ of rational solutions of $\tilde{L}$, then take $b_1 y, \ldots, b_k y$. Let $R \in K(x)[\partial]$ be the monic operator with $b_1 y, \ldots, b_k y$ as solutions. Now R is the unique monic operator whose solution space is the set of all solutions of L with the same type as y, so R is uniquely determined by two things: $L \in C(x)$ and type(y). Hence, any element σ of the Galois group of $\overline{\mathbb{Q}}$ over C that leaves type(y) invariant leaves R invariant as well. Conversely, if σ leaves R invariant, it leaves type(y) invariant as well because R has precisely one type of exponential solutions. Thus, one can define a *field of definition* of type(y) over C, namely as the smallest field C' that contains C for which $R \in C'(x)[\partial]$.

Note that since $R \in K(x)[\partial]$, one has $C' \subseteq K$, hence: the field of definition of y over C contains the field of definition of type(y) over C.

We will now prove constructively that y is of minimal algebraic degree over C if and only if the fields of definition of y and type(y) are the same. To do this, we will show that R has an exponential solution defined over C', in fact, we will show that the algorithm ExpSolsInC with input R, C' will find such solutions. Since R has only one type, there is exactly one local type at each singularity x_i, so generalized exponents at x_i are unique modulo integers. So if σ is in the Galois group of $\overline{\mathbb{Q}}$ over $C'(x_i)$, then it can only move a generalized exponent e_{x_i} by an integer, but this implies that σ leaves e_{x_i} invariant, and thus $e_{x_i} \in C'(x_i)[t_i^{-1}]$, *i.e.*, e_{x_i} is a C'-generalized exponent. Hence there is exactly one $\mathbb{Z}$-minimal C'-generalized exponent at every singularity x_i, which implies that ExpSolsIn$C(R,C')$ will try exactly one C'-combination. This combination has the same type as y, and since all solutions of R have this type, ExpSolsInC will find a basis $y_1, \ldots, y_k$ of solutions of R, defined over C', where k is the order of R. We conclude the following:

Lemma 6.3. *Let C be a number field and $L \in C(x)[\partial]$. If y is an exponential solution of L defined over some algebraic extension K of C, then its conjugates over C are also solutions of L.*
Given L and y, we can compute $y_1, \ldots, y_k$ of minimal algebraic degree over C that form a basis of all exponential solutions of L of the same type as y.
If y is of minimal algebraic degree over C, then the distinct conjugates of y over C are of distinct type and hence linearly independent.

Proof. The fact that for $L \in C(x)[\partial]$, the conjugates over C of solutions are again solutions is clear. From the foregoing, given L and y, we can compute an operator $R \in K(x)[\partial]$ whose solution space is precisely the set of all exponential solutions of L of the same type as y. We can calculate the smallest field C' containing C for which $R \in C'(x)[\partial]$, and can find a basis $y_1, \ldots, y_k$ of solutions of R with ExpSolsIn$C(R,C')$. Then $y_1, \ldots, y_k$ are defined over C', which is the field of definition of their type, so they have minimal algebraic degree over C. And they form a basis of all exponential solutions of L of the same type as y.

If σ is in the Galois group of $\overline{\mathbb{Q}}$ over C, and if two objects have the same field of definition over C, then σ leaves the first object invariant if and only if σ acts as the identity on the field of definition, if and only if σ leaves the second object invariant. Now if y is of minimal algebraic degree over C, then it has the same field of definition as type(y), so σ leaves type(y) invariant if and only if σ leaves y (which we identify with $\partial - y'/y$ because we do not want to distinguish y from scalar multiples of y) invariant. Thus, distinct conjugates of y must have distinct types. $\square$

Therefore, if we have an exponential solution of minimal algebraic degree m, then this exponential solution gives in fact m linearly independent solutions. In the output of our algorithm ExpSols we will only return exponential solutions up to conjugation over C. In order to make it easier to count how many linearly independent solutions such an output represents, we will only allow exponential solutions in the output of ExpSols that have minimal algebraic degree over C.

Definition 6.4. *If C is a number field, x_i a singularity of L and e_{x_i} a generalized exponent at x_i, then* the field $C(x_i; e_{x_i})$ *is the extension of C given by x_i and the coefficients of e_{x_i}.*

Proposition 6.5. *Let C be a number field, $L \in C(x)[\partial]$, x_i a singularity, t_i the local parameter, and let $e \in \overline{\mathbb{Q}}[t_i^{-1}]$. Let b_1 be the number of distinct generalized exponents e_{x_i} of L at x_i with $e_{x_i} - e \in \mathbb{Z}$. Suppose that y is an exponential solution of minimal algebraic degree m over $C(x_i; e)$ with generalized exponent e at x_i. Then $m \leq b_1$.*

Proof. By the previous Lemma, the conjugates of y over $C(x_i;e)$ form m linearly independent exponential solutions, all of which have generalized exponent in $e+\mathbb{Z}$ at x_i. These exponential solutions (viewed as formal solutions at x_i) can be written as $\phi_j(t_i)\exp(\int e/t_i\,\mathrm{d}t_i)$ where $\phi_j(t_i)\in\overline{\mathbb{Q}}((t_i))$ for all $j\in\{1,\dots,m\}$. After Gaussian elimination on the vectors of coefficients of the $\phi_j(t_i)$, we obtain m formal solutions at x_i with distinct generalized exponents in $e+\mathbb{Z}$. Then $m\le b_1$ because there are only b_1 such generalized exponents at x_i. □

The modular approach yields a different bound:

Proposition 6.6. *Let C be a number field, let $L\in C(x)[\partial]$ and let p be a good prime for L. Let b_2 be the number of roots of $\chi_p(L)$ (counted with multiplicity) that match (in the sense of Algorithm* CombMatchRoot*) the choice $\overline{e_{x_i}}$ at x_i. Let y be an exponential solution of minimal algebraic degree m over $C(x_i;e_{x_i})$, and generalized exponent e_{x_i} at x_i. Then $m\le b_2$.*

Proof. Let $R\in C(x_i;e_{x_i})(x)[\partial]$ be the right hand factor whose solutions are spanned by all exponential solutions y of L that have generalized exponent e_{x_i} at x_i. Since there are at least m independent such solutions, the order d of R is at least m. All solutions of R have the same local type at x_i (namely the local type e_{x_i} mod $\mathbb{Z}$, see Definition 6.1). Now R factors as a product of first order factors $R=R_1R_2\cdots R_d$ because R has a basis of exponential solutions. All R_i have the same local type at x_i, so for each R_i, the generalized exponent at x_i must be congruent to e_{x_i} mod $\mathbb{Z}$. Hence the root of $\chi_p(R_i)$ matches $\overline{e_{x_i}}$ at x_i. The proposition now follows from the fact that R and $R_1,\dots,R_d$ can be reduced mod p (see Proposition 1.15) and Lemma 1.13(a). □

We now have two bounds b_1 and b_2 for the degree of the extension needed over $C(x_i;e_{x_i})$. We do not know in advance which is smaller, so we compute both and take the minimum.

Definition 6.7. *To each couple (x_i,e_{x_i}) we associate the bound $b_{(x_i,e_{x_i})}:=\min(b_1,b_2)$.*

6.2 An algorithm to find an exponential solution over an algebraic extension

Let $L\in C(x)[\partial]$ with $C\subset\overline{\mathbb{Q}}$. We suppose that L has no exponential solutions defined over C and we want to find (if it exists) an exponential solution defined over an algebraic extension of C. We will use Algorithm ExpSolsInC as well as the bounds from the previous section. To find an exponential solution over an algebraic extension of C, we increase the field step by step by adding extensions coming from singularities or generalized exponents, and at each step we use ExpSolsInC to search for an exponential solution over the new field. The bounds tell us when we can stop making field extensions.

For efficiency reasons we first want to reduce the order of the problem as much as possible before introducing field extensions, which is precisely what Algorithm ExpSols in Section 7 does before it calls Algorithm FindASol below. And since the order can be reduced whenever a solution is found, we want FindASol to stop computing as soon as it finds a solution.

In the algorithm we denote $e_{i,j}$, $j=1,\dots,n_i$ as all, up to conjugation over $C(x_i)$, see Definition 5.6, *relevant* (see Remarks 1.9 and 5.8) generalized exponents at x_i. If $n_i=0$ then we can stop the algorithm, the output is $\emptyset$. Let F be a data structure as in the comments after Algorithm ExpSolsInC. If x_i appears in F, which means that a generalized exponent at x_i has already been chosen, then $n_i:=1$. We denote $d_{i,j}:=[C(x_i;e_{i,j}):C(x_i)]$, which is the degree as in Definition 5.6 of $e_{i,j}$ over $C(x_i)$. Now we define $b_{(x_i,e_{i,j},F)}$ as in Definition 6.7 with as difference that in Proposition 6.6 we only count those roots of $\chi_p(L)$ that match all generalized exponents specified in F. Of course, before we enter step 5 we discard all $e_{i,j}$ for which $b_{(x_i,e_{i,j},F)}=0$ because we want n_i to be as small as possible, and if any n_i vanishes then we stop.

```
**Algorithm** FindASol:
**Input**: an operator $L \in C(x)[\partial]$, a field $C$, a data structure $F$ and a positive integer $b$.
**Output**: an exponential solution of $L$ or $\emptyset$.
1 - If $b = 0$, then return $\emptyset$ and stop.
2 - Run ExpSolsIn$C$($L$,$C$,options={"just one", $F$}).
3 - If this gives a solution, then return it and stop.
4 - If $b = 1$, then return $\emptyset$ and stop.
5 - If we can choose $x_i$ in step 6 with degree 1 over $C$, or if $b' \leq b/2$ for all $b'$ in step $6a$,
    then go to step 6, else go to step 8.
6 - Choose one $C$-singularity $x_i$, not appearing in $F$. Then, for $j = 1, \ldots, n_i$ do $6a$ and $6b$:
6a - Let $b'$ be the minimum of $b_{(x_i, e_{i,j}, F)}$ and $\lfloor \frac{b}{d_{i,j}} \rfloor$.
6b - If FindASol($L, C(x_i; e_{i,j}), F \cup \{(x_i, e_{i,j})\}, b'$) $\neq \emptyset$, then return it and stop.
7 - Return $\emptyset$ and stop.
8 - Do step $8a$ for all $x_i$ not appearing in $F$ for which $d_i := [C(x_i) : C]$ is not 1 and
    there are at least two $j$'s with $d_{i,j} = 1$.
8a - If FindASol($L, C(x_i), F, \lfloor b \frac{\lfloor d_i/2 \rfloor}{d_i} \rfloor$) $\neq \emptyset$, then return it and stop.
9 - For all $i, j$ with $x_i$ not appearing in $F$ and $1 < d_{i,j} \leq b$ do step $9a$:
9a - If FindASol($L, C(x_i; e_{i,j}), F \cup \{(x_i, e_{i,j})\}, \lfloor \frac{b}{d_{i,j}} \rfloor$) $\neq \emptyset$, then return it and stop.
10 - Return $\emptyset$.
```

For the modular computations in recursive calls of ExpSolsInC, we always use the same prime number so that the modular information that we need is only computed once. One can also re-use information on singularities and generalized exponents from previous computations, but this information needs to be updated to the new field which involves factoring polynomials. For the singularities this means: factoring a_n over the new field. The generalized exponents need to be updated as well, for example, if we extend C then the degree over $C(x_i)$ of a generalized exponent can decrease, which means that the polynomials defining the field extension $C(x_i; e_{i,j})$ over $C(x_i)$ have become reducible, so they need to be factored. This way one conjugacy class of generalized exponents can split up into several conjugacy classes of generalized exponents.

We are going to verify (we must prove this recursively, *i.e.*, by induction) that FindASol satisfies the following:

Algorithm specification: if an exponential solution y exists that satisfies the fixed choices in F and has degree $\leq b$ over C, then FindASol will return some exponential solution of L (but not necessarily of degree $\leq b$ over C).

Proof. We first show that step 6 is correct. Suppose such y exists. Then the conjugates of y over C are also solutions of L, and satisfy the same fixed choices made in F (because in F we only have singularities and generalized exponents defined over C) as well as the degree bound b. Now the $e_{i,j}$ constitute, up to conjugation, a complete list of relevant $\mathbb{Z}$-minimal generalized exponents at x_i, so there must exist j such that a conjugate of y has generalized exponent in $e_{i,j} + \mathbb{Z}$ at x_i. We may assume that conjugate is just y. So for this j, there is a solution y defined over a field C' with $[C' : C] \leq b$ having generalized exponent in $e_{i,j} + \mathbb{Z}$ at x_i. We now replace C by $C(x_i; e_{i,j})$ and F by $F \cup \{(x_i, e_{i,j})\}$ so we may use the degree bounds from the previous section. We included a new bound in the algorithm as well, namely $\lfloor \frac{b}{d_{i,j}} \rfloor$, which holds because:

$e_{i,j}$ is a C'-generalized exponent since it corresponds (mod $\mathbb{Z}$) to the solution y defined over C'. So $C'(x_i) = C'(x_i; e_{i,j})$. Let $\tilde{b} = [C'(x_i) : C(x_i)]$. Now $C'(x_i) = C'(x_i; e_{i,j})$ contains $C(x_i; e_{i,j})$, which has degree $d_{i,j}$ over $C(x_i)$, and so $[C'(x_i) : C(x_i; e_{i,j})] = \tilde{b}/d_{i,j}$. Now y is defined over C', hence over $C'(x_i)$, which is an extension of $C(x_i; e_{i,j})$ of degree $\tilde{b}/d_{i,j}$ and this is $\leq \lfloor \frac{b}{d_{i,j}} \rfloor$ because $\tilde{b} \leq [C' : C] \leq b$.

Now steps 8 and 9. Again suppose that an exponential solution y exists that satisfies the fixed choices in F and has degree $\leq b$ over C. Let again C' be the field of definition of y over C. If every C'-combination was also a C-combination then ExpSolsIn$C(L, C)$ and ExpSolsIn$C(L, C')$ would have to find an equal number of solutions; so FindASol will return a solution in step 3. Thus we may assume that not every C'-combination is a C-combination, so there are more C'-combinations than C-combinations. This can only happen when at least one of the following is true:

(I). There are more C'-singularities than C-singularities, or:
(II). There exists a C-singularity x_i at which there are more C'-generalized exponents than C-generalized exponents.

Case (I) will be handled by step 8 and case (II) by step 9. Suppose that (I) holds. Then the minimal polynomial of some C-singularity x_i must be reducible over C'. This minimal polynomial has degree d_i. One of the factors over C' must have degree $d' \leq \lfloor d_i/2 \rfloor$. By abuse of notation we denote x_i as a root of that factor as well. So then $[C'(x_i) : C'] = d'$, $[C' : C] \leq b$, so $[C'(x_i) : C] \leq d'b$. Now $C(x_i)$ is a field between $C'(x_i)$ and C having degree d_i over C, so $[C'(x_i) : C(x_i)] = [C'(x_i) : C]/d_i \leq d'b/d_i \leq b\epsilon$ where $\epsilon := \lfloor d_i/2 \rfloor / d_i$. Note that $\epsilon = 1/2$ when d_i is even, and $\epsilon < 1/2$ when d_i is odd. Now y is defined over C', hence also over $C'(x_i)$, which is an extension of degree $\leq b\epsilon$ over our new field $C(x_i)$. Hence the degree bound used in step $8a$ is correct.

Note that if there exists an i for which there is no j with $d_{i,j} = 1$ then a solution y necessarily involves a generalized exponent at x_i that is not defined over $C(x_i)$, the case that is treated by step 9, and in this case the algorithm stays correct (but becomes more efficient) if we skip 8 and use only this i in step 9. And if there is only one j with $d_{i,j} = 1$, then either we must choose this j (at x_i and its conjugates over C), or the problem is handled by step 9 (in which case it is enough to use only this i in 9).

Increasing the number of singularities does not increase the number of combinations if there is only one choice to be made at these singularities, thus skipping i in step 8 when there is only one j with $d_{i,j} = 1$ is correct.

Now assume that (I) does not hold (then every C'-singularity is a C-singularity) and assume that (II) holds. Then at some C-singularity x_i we have a C'-generalized exponent e_{x_i} that is not a C-generalized exponent. This generalized exponent is defined over $C'(x_i)$ but has degree $d_{i,j}$ over $C(x_i)$. The degree bound $\lfloor \frac{b}{d_{i,j}} \rfloor$ in step $9a$ can then be proved with the same arguments as before, which completes the proof that FindASol satisfies its specification. □

The degree bound drops at least a factor 2 each time we use recursion with an extension of C. So if the initial degree bound b is less than 4 then the extensions we construct are not greater than those given by a single x_i (in step 8) or a single couple (x_i, e_{x_i}) in step 9. If $b < 8$ then we will stack at most 2 such extensions (each given by an x_i or by a couple (x_i, e_{x_i})) on top of each other, which is generally feasible in practice. So unless the degree bound is very high we do not expect the algorithm to choke on these algebraic extensions. And even if the degree bound is very high (order $\geq$ Nroots ≥ 8), the alternatives currently used by Maple's *expsols* need not be better.

We can further improve the algorithm with modular methods. For example, certain x_i in step 8 may be omitted on the basis of modular information, namely when none of the roots of $\chi_p(L)$ allows a combination that uses distinct generalized exponents at two singularities conjugated to x_i.

If a root of $\chi_p(L)$ can only match a solution defined over C (this can happen when it must use an $e_{i,j}$ for which $C(x_i; e_{i,j}) = C$ and the bound in Definition 6.7 is 1) then we can discard this root after step 2. Reducing the number of roots of $\chi_p(L)$ in this way reduces the bounds and the n_i's which may help prevent making some unnecessary field extensions.

Step $8a$ can be improved in the following way. If the recursive call for x_i returns $\emptyset$, then later recursive calls in step $8a$ should be prevented from factoring the minimal polynomial of x_i over the new fields, and prevented from making that same extension x_i again.

A similar improvement is possible in step $6b$. If the recursive call for $e_{i,j}$ returns $\emptyset$, then we could discard $e_{i,j}$. This will prevent later recursive calls from trying $e_{i,j}$ again. But it also reduces the number of generalized exponents, which may improve the bound b (see the number N_i in comment [1] after Algorithm ExpSols in Section 7).

7 An algorithm to find all exponential solutions

In this section, we give a complete algorithm to find a basis, up to conjugation, for all exponential solutions of a differential operator $L \in C(x)[\partial]$ where $C \subset \overline{\mathbb{Q}}$. The general idea in this recursive algorithm is the following. Using our procedure ExpSolsInC, we first compute the "easy" first order factors, that is, those defined over C. Then by looking at the value of Nroots (from Algorithm CombMatchMod$p$) we check if there may exist more exponential solutions, which are then necessarily not definable over C. If such exponential solutions could exist then we first recursively remove all "easy" factors, both on the right as well as on the left using the *adjoint* L^* of L (see [Inc26, 5.3] or [PS03, 2.1]) until there are no more easy factors on either side. We do this because we want to make the order of the operator as small as possible before entering the "hard" case (*i.e.*, searching for exponential solutions defined over algebraic extensions using Algorithm FindASol).

The specifications of the following ExpSols algorithm are the following: first, it should not return the "same" solution more than once, more precisely: the output gives *only one* exponential solution in each conjugacy class. This is because we want the output to be a *basis up to conjugation.* Second, the solutions given in the output must all be of *minimal algebraic degree* over C, because this guarantees (see Lemma 6.3) that their conjugates over C are linearly independent. To count the number of elements in the basis of exponential solutions, we must count a solution y given in the output $d(y)$ times, where $d(y)$ is the degree over C of the field of definition of y. In the following, Card(Sol) denotes the number of elements of the set Sol. The number of solutions represented by Sol is the sum of $d(y)$ taken over $y \in$ Sol. Finally, we mark with a [.] the steps for which explanations or comments are given after the algorithm.

```
Algorithm ExpSols:
Input: a linear differential operator $L \in C(x)[\partial]$ ($C \subset \overline{\mathbb{Q}}$) and the field $C$.
Output: Sol, a basis of exponential solutions of $L$ up to conjugation over $C$.
– Sol := ExpSolsIn$C$($L$,$C$).
– If Card(Sol) $\geq$ Nroots $-1$, then return Sol and stop ([1]).
– If Sol $\neq \emptyset$ then,
  - Write $L = \tilde{L}$ LCLM($\partial - y'/y \,|\, y \in$ Sol),
  - ExpSols($\tilde{L}$,$C$) ([2]).
  - Keep only the types not defined over $C$ of the solutions found,
  - For each remaining type represented by $t$ ([3]),
    . Compute a basis $R_1, \ldots, R_s$ of rational solutions of $L_{\partial \to \partial + t}$,
    . Add the $\exp(\int t)\, R_i$ to Sol.
  - Return Sol and stop.
– Else,
  - If the order is 2 then go to the next Else below,
  - Make $L$ monic,
  - ExpSolsIn$C$($L^*$,$C$) where $L^*$ is the adjoint of $L$ ([4]).
  - If it finds solutions $y_1, \ldots, y_r$ where $r > 0$ then,
    . Write $L =$ LCLM$(\partial - y_1'/y_1, \ldots, \partial - y_r'/y_r)^*\, \tilde{L}$,
    . Return ExpSols($\tilde{L}$,$C$) and stop.
  - Else ([5]),
    . FindASol($L$,$C$,$\emptyset$,Nroots) ([6]). If $\emptyset$ then return $\emptyset$ and stop.
    . Optimize the solution $y$ found ([7]),
    . Compute a basis $R_1, \ldots, R_s$ of rational solutions of $L_{\partial \to \partial + y'/y}$,
    . Add the $y\, R_i$ to Sol,
    . Write $L = \tilde{L}$ LCLM($\partial - r_1, \ldots, \partial - r_s$, "and conjugates over $C$"), ([8])
    . Remove recursively from $\tilde{L}$ the solutions of the same type as $y$ ([9]),
    . ExpSols($\tilde{L}$,$C$).
    . Keep only the types not defined over $C$ of the solutions found,
    . For each remaining type represented by $t$ ([3]),
        Compute a basis $R_1, \ldots, R_s$ of rational solutions of $L_{\partial \to \partial + t}$,
        Add the $\exp(\int t)\, R_i$ to Sol.
    . Return Sol and stop.
```

We explain or comment the points marked with a [.]:
[1] - The number Nroots, which was calculated in CombMatchMod$p$, is used as upper bound for the number of linearly independent exponential solutions. We have already found Card(Sol) independent solutions and so there are at most $d :=$ Nroots $-$ CardSol linearly independent exponential solutions left. This d is an upper bound for the degree of the field extension of remaining exponential solutions of minimal algebraic degree. If $d \leq 1$ then no extensions are necessary so we can stop the computation.
At a singularity x_i, let N_i be the number of distinct unramified generalized exponents in $\overline{\mathbb{Q}}[t_i^{-1}]$: this is another upper bound for the total number of linearly independent exponential solutions. If $N_i <$ Nroots, then replace Nroots by N_i.
[2] - At this point the roots of the p-curvature have already been computed, and need not be computed again; we can simply take the roots (with multiplicity) for $\chi_p(L)$ and reduce the multiplicities according to the solutions found.

[3] - We have applied recursion on a left factor of L so the exponential solutions found are in general not exponential solutions of L. We will only use the types of the exponential solutions of this left factor, not the exponential solutions themselves. We skip types defined over C because all exponential solutions over C have already been found. From [Ore32] or [Hoe97b, Lemma 7.1], the types of exponential solutions of L that are not yet found must be among the types in the left factor $\tilde{L}$ of L.
[4] - We will apply FindASol only when all easy factors have been removed both on the left and on the right. Removal of easy factors, left or right, does not cause solutions to be lost because we only do this after all easy solutions have already been computed and stored in the set Sol, in the first step of the algorithm. To find the easy left factors, we apply ExpSolsInC to the adjoint operator L^* to find the easy exponential solutions of L^* which correspond to easy left factors of L (see [PS03, 2.1]). Then we apply recursion on the remaining right hand factor.
[5] - If the order of the operator is two, and if the bounds in FindASol do not immediately rule out a solution defined over an extension, then the formulas given in [UW96] would be a good alternative to FindASol. If the order is three, then an alternative would be the eigenring method (see [PS03]) but we do not expect that to be better than FindASol, especially if one implements some shortcuts for order 3 in FindASol (such as: we only need to consider extensions in FindASol that have a degree 3 subfield. And: if at x_i there are two non-conjugated generalized exponents that do not differ by an integer, then $C(x_i)$ is the only extension that needs to be considered).
[6] - Here, we reach the point where there is nothing left to do than entering the "hard case", trying to find a solution over an extension of C with FindASol. An *a priori* first bound is Nroots, the number of roots in $\overline{\mathbb{F}}_p(x^p)$ counted with multiplicity of $\chi_p(L)$ for a good prime p. Note that a more detailed analysis could lead to a sharper bound. For example, if two roots of $\chi_p(L)$ can not correspond to conjugated exponential solutions, or if one can conclude using the bounds in Section 6.1 that some root of $\chi_p(L)$ can not correspond to an exponential solution of minimal algebraic degree Nroots, then we could give FindASol a better bound.
[7] - Let y be an exponential solution. We can write $y'/y = P_1/P_2$ where P_1 and P_2 are polynomials with gcd 1 and P_2 is monic. The field of definition of y is then the field generated over C by the coefficients of P_1 and P_2. By "optimizing the solution" we mean two things: 1). Using this solution y to find a solution of minimal algebraic degree over C. And 2). Making sure that the field that the algorithm gives for y (this field contains the field of definition of y) is actually equal to the field of definition of y. Both 1) and 2) are important. We want y to be of minimal algebraic degree so that we know that its conjugates are linearly independent. But the way we count the number of conjugates of y, *i.e.*, the way we determine the number $d(y)$, is not by looking at y, but by looking at the field given for y. The field provided for y by Algorithm FindASol contains, but need not be not equal to, the field of definition of y. So to optimize y, we take the field given for y, and then determine the subfield generated over C by the coefficients of P_1 and P_2. Then we find defining equations (*i.e.*, new RootOf's) for this subfield, and use them to rewrite the coefficients of P_1 and P_2. This then takes care of 2). To do 1), we could use the approach in Lemma 6.3, however, this is not necessary because the special choice that Algorithm ExpSolsInC makes when the option "just one" is given causes 1) to be automatically satisfied. The polynomial Q is of minimal degree with this option given, which leads to uniqueness of y'/y which in turn causes y to already be of minimal algebraic degree over C.
[8] - Here $r_i = R_i'/R_i + y'/y$. One can compute this LCLM using the method of undetermined coefficients and solving linear equations over $C(x)$, this can be done without actually constructing the conjugates of y'/y.
[9] - What is meant here is to also remove solutions whose type is conjugated over C to the type

of y. We will remove such solutions recursively because if we do not then it would be non-trivial to ensure the specification of the algorithm that the "same" solution is not returned more than once. So before we call ExpSols on the remaining left factor $\tilde{L}$ of L, we first remove from $\tilde{L}$ all exponential solutions of the same type as (conjugates of) y, and we keep repeating this until $\tilde{L}$ no longer has solutions of this type. A recursive procedure to achieve that is the following: compute the rational solutions R_i of $\tilde{L}_{\partial \to \partial + y'/y}$; each one gives a first order right hand factor $\partial - r_i$ of L where $r_i = R_i'/R_i + y'/y$. If there are none then return $\tilde{L}$ else write $\tilde{L} = \tilde{L_1}$ LCLM($\partial - r_1, \ldots, \partial - r_s$, "and conjugates over C") and apply recursion on $\tilde{L_1}$.

7.1 Some remarks on the algorithm

Remark on ExpSolsInC: Even though we use Algorithm CombMatchMod$p$, there could still be combinations in step 4 that do not lead to an exponential solution (if for example the number of combinations is greater than Nroots then we already know in advance that this will be the case). If the degree bound (the number N in step 4) for polynomial solutions is high, or if C is a complicated field, or if there are many combinations to be checked, then to make the algorithm efficient we need a quick way to discard non-solution-combinations before doing all the work in steps 4a and 4b, because each of these two steps can dominate the computation time.

In step 4, we can of course pre-compute the traces, so that all computations in step 4 are rational (*i.e.*, over C). Computing S in step 4a then only involves additions in $C(x)$. However, that does not imply that computing S is cheap, because normalizing S (writing S as P_1/P_2 where P_1, P_2 are polynomials with no common factors) can be an expensive operation if C is a complicated field. Computation of $\tilde{L} = L_{\partial \to \partial + S}$ can be even more expensive because we need to multiply, differentiate, and add in $C(x)$. And since step 4a is applied to each combination, one can easily spend more time in step 4a than in steps 1 and 2 combined. And if N is large then 4b can take even more time.

Often the N's are very small, and one can improve the running time of ExpSolsInC substantially by implementing some cases for step 4b separately, such as the case $N = 0$ (in that case we should use a fast zero test for $\tilde{L}(1)$ without computing $\tilde{L}$ itself) and the case N small non-zero: here, one would be tempted to take an ansatz $\sum_{i=0}^{N} c_i x^i$, and to compute the coefficients of $\tilde{L}$(ansatz) which gives linear equations for the unknowns c_i. But for small N and large $\tilde{L}$ this results in much more equations than we need because there are only $N + 1$ unknowns. In this case, to make the ansatz approach efficient we should not fully evaluate $\tilde{L}$(ansatz) because then we computed much more equations than we need. Instead we should evaluate just enough coefficients of $\tilde{L}$(ansatz) (and do this without fully computing $\tilde{L}$ itself) so that we have enough equations to determine the c_i, and once the $c_i \in C$ are found then we finish with a quick zero test for $\tilde{L}(\sum_{i=0}^{N} c_i x^i)$. For large N we propose to compute $\tilde{L}$ and to use [ABP95].

If we find $w > 0$ polynomial solutions in step 4b then the computation time in 4a and 4b was well spent even if those steps took a lot of time. But when $w = 0$, and if C is a complicated field or N is a large number, then we want to avoid 4a and 4b with high probability. This is done as follows: take a sufficiently large prime number p (in general this p is not the same prime as we used for the p-curvature because for the p-curvature we only use small primes) such that $C[p] = \mathbb{F}_p$ (such primes have density at least $1/[C : \mathbb{Q}]$ so they are easy to find). Then compute $\tilde{L}[p] \in \mathbb{F}_p(x)[\partial]$ and check if it allows a polynomial solution of degree $\leq N$. If not, skip 4a and 4b.

More general fields: It is not difficult to generalize our algorithm to arbitrary fields of characteristic 0. We may assume that C is the field generated by the coefficients of L, so then C is a finitely generated extension of $\mathbb{Q}$, say $C = \mathbb{Q}(t_1, \ldots, t_k)[\alpha_1, \ldots, \alpha_l]$ where $t_1, \ldots, t_k$ are algebraically independent and the α_i are algebraic over $\mathbb{Q}(t_1, \ldots, t_k)$. Then the reduction mod p works as follows: choose random values for t_i in $\mathbb{Q}$. This gives a map f_1 from a subring R_1 of $\mathbb{Q}(t_1, \ldots, t_k)$ to $\mathbb{Q}$. If not all coefficients of the minimal polynomial of α_1 are in R_1, then choose different random values for t_i. After that, the minimal polynomial of α_1 is mapped to a polynomial over $\mathbb{Q}$, we can choose an irreducible factor, a root α'_1 of that factor, and map α_1 to α'_1. Do the same for $\alpha_2, \ldots, \alpha_l$. This way we have a map f_2 from a subring R_2 of C to the field $\mathbb{Q}(\alpha'_1, \ldots, \alpha'_l)$. We can then choose a reduction mod p in the same way as before, so we have a map g from a subring of $\mathbb{Q}(\alpha'_1, \ldots, \alpha'_l)$ to a finite field. Then let the map $[p]$ be the composition $g \circ f_2$ of these two maps, which will be defined on some subring R_3 of R_2. One can always find evaluations in $\mathbb{Q}$ for the t_i and a prime number p such that all elements of C that we want to reduce mod p are in R_3.

Logarithms: Generalized exponents correspond to formal solutions. Since we are only interested in exponential solutions, the formal solutions that contain a logarithm are not of interest to us. Because of this, we can disregard those generalized exponents that do not correspond to a formal solution without a logarithm. Note that in any given equivalence class mod $\mathbb{Z}$, this observation could eliminate some but never all unramified generalized exponents, so it does not help much for the combinatorial problem. However, this observation may reduce the number of distinct unramified generalized exponents in some equivalence classes mod $\mathbb{Z}$, which can reduce the bound in Proposition 6.5 as well as the number N_i in comment [1] after Algorithm ExpSols. This could speed up Algorithm FindASol. Note that if L has an exponential solution of minimal algebraic degree $n-1$ where n is the order of L, then the adjoint of L has an exponential solution defined over C. Algorithm ExpSols removes this solution, hence, if ExpSols calls FindASol with degree bound (the number b in FindASol) $N_i = n-1$ then we may use $n-2$ instead.

Deciding which generalized exponents match formal solutions without logarithms can be done if we compute $d+1$ terms of the formal solutions, where d is the exponent-difference. We propose to do this computation only when ExpSols has to call FindASol. To speed up this computation, one could compute the formal solutions modulo a suitably large prime number, if a logarithm occurs then a logarithm must occur in characteristic zero as well.

A related implementation: Although we do not have an implementation for the algorithm ExpSols in this paper, there is an implementation in Maple 9 for the difference case, *i.e.*, a procedure that computes the hypergeometric solutions of a difference equation. This implementation follows some of the ideas in this paper and performs very well in practice (a paper concerning the difference case is planned). We expect our ExpSols to perform equally well.

7.2 Computing radical solutions

Computing radical solutions is almost the same problem as computing exponential solutions, with only a few differences:

1. One only uses generalized exponents that are in $\mathbb{Q}$, see Equation (5) and Remark 1.10.
2. τ_{red} maps rational numbers to zero, which implies that the only root of $\chi_p(L)$ that matters is 0, but also implies that there is no reduction in the combinatorial problem.

This means that for computing radical solutions, the only useful modular information is the multiplicity of the root 0 in $\chi_p(L)$, which we can use as a bound for the number of linearly independent radical solutions, hence as a bound for the degree of the field extension over which such solutions (of minimal algebraic degree) are defined.

To save computation time, we can compute this multiplicity without computing $\chi_p(L)$, in the following way: Take a good prime p. Then compute a basis $z_1, \ldots, z_s$ of rational solutions of $L[p]$. If $s = 0$ then return 0, otherwise write $L[p] = L_1 L_2$ where $z_1, \ldots, z_s$ is a basis of solutions of L_2, apply recursion to L_1, then add s to the result for L_1, and return the answer. We expect this to be faster than computing $\chi_p(L)$ because rational solutions of a differential operator in characteristic p can be computed very quickly, $\mathcal{O}(\max(l,p)^2 p)$ field operations where l is a bound on the degree of the coefficients of L, see [Clu03].

7.3 Almost all primes

There are many results known that hold for all but finitely many p. For example, it is easy to show that

Lemma 7.1. *For all but finitely many p, the number of linearly independent radical solutions of L is bounded by the dimension of the space of rational solutions of $L[p]$.*

Thus, for almost all p, we could have used the number s in the previous section as a bound. The reader may wonder why we did not do so. The problem here is that of the many results that are known for all but finitely many p, the only ones we can use for the algorithm are those for which a *small prime p can be exhibited* with little computation. This problem is also the reason that the only information we use from the p-curvature is its characteristic polynomial.

References

[ABP95] S. A. Abramov, M. Bronstein, M. Petkovšek. On Polynomial Solutions of Linear Operator Equations. In *Proceedings of ISSAC'95*, ACM, New York, 1995.

[Bek94] E. Beke. Die Irreduzibilität der homogenen linearen differentialgleichungen. In *Math. Ann.*, **45**, 278-294, 1894.

[Bou00] D. Boucher. Sur les Équations Différentielles Linéaires Paramétrées, une Application aux Systèmes Hamiltoniens. Thèse, Université de Limoges, 2000.

[Cha01] A. Chambert-Loir. Théorèmes d'Algébricité en Géométrie Diophantienne. *Séminaire Bourbaki*, exposé No. **886**, Mars 2001.

[CC85] D. V. Chudnovsky, G. V. Chudnovsky. Applications of Pade Approximations to the Grothendieck Conjecture on Linear Equations. In *Lecture Notes in Mathematics*, Number Theory, **1135**: 52-100, 1985.

[Clu03] T. Cluzeau. Factorization of Differential Systems in Characteristic p. In *Proceedings of ISSAC'03*, August 3-6, Philadelphia, Pennsylvania: 58-65, 2003.

[Gie98] M. Giesbrecht. Factoring in Skew-Polynomial Rings over Finite Fields. In *Journal of Symbolic Computation*, **24(5)**: 463-486, 1998.

[GZ03] M. Giesbrecht, Y. Zhang. Factoring and Decomposing Ore Polynomials over $\mathbb{F}_p(t)$. In *Proceedings of ISSAC'03*, August 3-6, Philadelphia, Pennsylvania: 127-134, 2003.

[Hoe97a] M. van Hoeij. Formal Solutions and Factorization of Differential Operators with Power Series Coefficients. In *Journal Of Symbolic Computation*, **24**: 1-30, 1997.

[Hoe97b] M. van Hoeij. Factorization of Differential Operators with Rational Functions Coefficients. In *Journal Of Symbolic Computation*, **24**: 537-561, 1997.

[HW97] M. van Hoeij, J.-A. Weil. An Algorithm for Computing Invariants of Differential Galois Groups. In *Journal of Pure and Applied Algebra*, **117-118**: 353-379, 1997.

[Hon81] T. Honda. Algebraic Differential Equations. In *Symp. Math.*, **24**: 169-204, 1981.

[Inc26] E. L. Ince. Ordinary Differential Equations. *Dover Publications*, INC. New York, 1926.

[Kat82] N. Katz. A Conjecture in the Arithmetic Theory of Differential Equations. In *Bulletin de la Société Mathématique de France*, **110**: 515-534, 1982.

[Ore32] O. Ore. Formale Theorie der linearen Differentialgleichungen (Zweiter Teil), *J. für d. Reine u. angew. Math.*, **168**: 233-252, 1932.

[Pfl97] E. Pflügel. An Algorithm for Computing Exponential Solutions of First Order Linear Differential Systems. In *Proceedings of ISSAC'97*, ACM, New York, 1997.

[Put95] M. van der Put. Differential Equations in Characteristic p. In *Compositio Mathematica*, **97**: 227-251, 1995.

[Put96] M. van der Put. Reduction Modulo p of Differential Equations. In *Indag. Math.N.S*, **7(3)**: 367-387, 1996.

[Put01] M. van der Put. Grothendieck's Conjecture for the Risch Equation $y' = ay+b$. In *Indag. Mathem.*, **12(1)**: 113-124, 2001.

[PS03] M. van der Put, M. F. Singer. Galois Theory of Linear Differential Equations. In *Grundlehren der mathematischen Wissenschaften*, Vol. **328**, Springer, 2003.

[Sin96] M. F. Singer. Testing Reducibility of Linear Differential Operators: A Group Theoretic Perspective. In *Journal of Appl. Alg. in Eng. Comm. and Comp.* (AAECC), **7(2)**: 77-104, 1996.

[UW96] F. Ulmer, J.-A. Weil. Note on Kovacic's algorithm. In *Journal Of Symbolic Computation*, **22**: 179-200, 1996.