PARDI !

François Boulier
Université Lille I, LIFL
Villeneuve d'Ascq, France
boulier@lifl.fr

François Lemaire
Université Lille I, LIFL
Villeneuve d'Ascq, France
lemaire@lifl.fr

Marc Moreno Maza
Université Lille I, LIFL
Villeneuve d'Ascq, France
moreno@lifl.fr

ABSTRACT

We propose a new algorithm for converting a characteristic set of a prime differential ideal from one ranking into another. This differential algebra algorithm computes characteristic sets by change of ranking (ordering) for prime ideals. It identifies the purely algebraic subproblems which arise during differential computations and solves them algebraically. There are two improvements w.r.t. other approaches: formerly unsolved problems could be carried out; it is conceptually simple. Different variants are implemented.

Keywords

differential algebra. PDE. characteristic sets. change of rankings. gcd.

1. INTRODUCTION

In this paper, we propose an algorithm which solves the following problem: given a characteristic set C of a prime differential ideal $\mathfrak{a}$ w.r.t some ranking $\mathcal{R}$ and another ranking $\overline{\mathcal{R}} \neq \mathcal{R}$, compute a characteristic set $\overline{C}$ of $\mathfrak{a}$ w.r.t. $\overline{\mathcal{R}}$. The algorithm that we present, called[1] PARDI applies for systems of partial differential polynomial equations. It specializes to systems of ordinary differential polynomial equations and is then called[2] PODI. It specializes to nondifferential polynomial equations where it is called[3] PALGIE.

As far as we know, Ollivier was the first to solve the problem addressed in this paper. Let's quote [19, page 95]: "one can [design] a method for constructing a characteristic set of a finitely generated prime differential ideal as soon as one can effectively test membership to this ideal". An algorithm is given in SCRATCHPAD in [19, page 97]. In all approaches, a known characteristic set provides the membership test algorithm. Such a problem was also considered in [3]. However, the algorithms presented in [3] compute differential polynomials which are not necessarily part of the desired characteristic set but only help computing it. They are complementary to PARDI.

The restriction to prime ideals is realistic. Indeed most differential systems coming from real problems generate differential prime ideals. Quite often, nondifferential polynomial systems in positive dimension either generate prime ideals or can be decomposed into prime ideals. Assuming that prime ideals are given by characteristic sets is realistic too. In the differential case, it happens quite often (e.g. dynamical systems in nonlinear control theory) that the input equations already form characteristic sets w.r.t. some rankings. Let us observe that one can decide primality for any ideal presented as the set of the polynomials pseudoreduced to zero by a characteristic set. The algorithm we propose generalizes to ideals which are not necessarily prime. However, for the reasons explained above and the legibility of the paper, we prefer to restrict ourselves to the prime case. Our algorithm applies to perform invertible changes of coordinates on the dependent and independent variables. Such changes realize ring isomorphisms between two differential rings $\phi : R \to \overline{R}$, and one–to–one correspondences between the differential ideals of R and the ones of $\overline{R}$. However the image $\overline{C}$ of a characteristic set C of $\mathfrak{a}$ is usually not a characteristic set of the ideal $\overline{\mathfrak{a}} = \phi\mathfrak{a}$ and there is usually no ranking w.r.t. which a characteristic set of $\overline{\mathfrak{a}}$ could be easily deduced from $\overline{C}$. The idea is then to apply PARDI over $\overline{C}$ but to test membership in $\overline{\mathfrak{a}}$ by performing the inverse changes of coordinates and testing membership in $\mathfrak{a}$ using C.

Our approach offers several advantages. It identifies the algebraic subproblems which occur in the differential computations and solves them by a purely algebraic method. This improves the control of the coefficients growth and avoids many useless computations only due to differential considerations. This very important advantage w.r.t. all other approaches permits us to handle some unsolved problems, even using a preliminary implementation. The three variants were implemented: PARDI in MAPLE, PODI in C and PALGIE in MAPLE, C and Aldor. The application to the change of variables was implemented in MAPLE. A last contribution is the conceptual simplicity of our algorithm, which contrats with the high technicity of its implementation. Ev-

[1]PARDI is an acronym for Prime pARtial Differential Ideal. In French, "*pardi*" is an oldfashioned swearword such as, say, "*egad*" in English.
[2]PODI is an acronym for Prime Ordinary Differential Ideal.
[3]PALGIE is an acronym for Prime ALGebraic IdEal. However, since "*algie*" means "suffering" in French, one might also understand PALGIE as "polynomial suffering" say.

ISSAC 2001, UWO, Canada
©2001 ACM 1-58113-218-2/ 00/ 0008 $5.00

erybody knows that the common roots of two univariate polynomials over a field are given by their gcd. Our algorithm applies this very simple idea and replaces any two univariate polynomials by one of their gcd over the fraction field of some quotient ring which makes much more sense than speaking of full remainders as in the previous approaches.

2. DEFINITIONS AND NOTATIONS

Let X be an ordered alphabet (possibly infinite). Let $R = K[X]$ be a polynomial ring where K is a field. Let $p \in R \setminus K$ be a polynomial. The *leader* of p, denoted $\mathrm{ld}\, p$, is the greatest indeterminate x which occurs in p. The polynomial p can be written as $p = a_d\, x^d + \cdots + a_1\, x + a_0$ where $d = \deg(p, x)$ and the polynomials a_i are free of x. The polynomial $i_p = a_d$ is the *initial* of p. The *rank* of p is the monomial x^d. The *reductum* of p is the polynomial $p - i_p\, x^d$. If x^d and y^e are two ranks then $x^d < y^e$ if $x < y$ or $x = y$ and $d < e$. The *separant* of p is the polynomial $s_p = \partial p / \partial x$. Let $A \subset R \setminus K$ be a set of polynomials. Then I_A (resp. S_A) denotes the set of the initials (resp. the separants) of its elements. We denote $H_A = I_A \cup S_A$. The set A is said to be *triangular* if its elements have distinct leaders. Let q be a polynomial. We denote $\mathsf{prem}(q, p)$ the pseudoremainder of q by p, both polynomials being viewed as univariate polynomials in the leader of p. We denote $\mathsf{prem}(q, A)$ "*the*" pseudoremainder r of q by all the elements of A i.e. any polynomial r obtained from q and the elements of A by performing successive pseudoreductions and such that $\mathsf{prem}(r, p) = r$ for every $p \in A$. Without further precisions, r is not uniquely defined. Fix any precise algorithm.

If A is a subset of a ring R then (A) denotes the ideal generated by A. Let $\mathfrak{a}$ be an ideal of R. If $S = \{s_1, \ldots, s_t\}$ then the *saturation* $\mathfrak{a} : S^\infty$ of $\mathfrak{a}$ by S is the ideal $\mathfrak{a} : S^\infty = \{p \in R \mid \exists a_1, \ldots, a_t \in \mathbb{N} \text{ s.t. } s_1^{a_1} \cdots s_t^{a_t}\, p \in \mathfrak{a}\}$.

Reference books for differential algebra are [21] and [12]. We also refer to the MAPLE VR5 and VI `diffalg` package and to [5, 6, 20, 10]. A *derivation* over a ring R is a map $\delta : R \to R$ such that $\delta(a + b) = \delta a + \delta b$ and $\delta(a\, b) = (\delta a) b + a(\delta b)$ for every $a, b \in R$. A *differential ring* is a ring endowed with finitely many derivations which commute pairwise. The commutative monoid generated by the derivations is denoted by Θ. Its elements are the *derivation operators* $\theta = \delta_1^{a_1} \cdots \delta_m^{a_m}$ where the a_i are nonnegative integer numbers. The sum of the exponents a_i, called the *order* of the operator θ, is denoted by $\mathrm{ord}\, \theta$. The identity operator is the unique operator with order 0. The other ones are called *proper*. If $\phi = \delta_1^{b_1} \cdots \delta_m^{b_m}$ then $\theta\phi = \delta_1^{a_1 + b_1} \cdots \delta_m^{a_m + b_m}$. If $a_i > b_i$ for each $1 \le i \le m$ then $\theta / \phi = \delta_1^{a_1 - b_1} \cdots \delta_m^{a_m - b_m}$. A *differential ideal* $\mathfrak{a}$ of R is an ideal of R closed under derivation i.e. such that $a \in \mathfrak{a} \Rightarrow \delta a \in \mathfrak{a}$. Let A be a nonempty subset of R. We denote $[A]$ the differential ideal generated by A which is the smallest differential ideal which contains A.

Let $U = \{u_1, \ldots, u_n\}$ be a set of *differential indeterminates*. Derivation operators apply over differential indeterminates giving *derivatives* θu. We denote ΘU the set of all the derivatives. Let K be a differential field. The differential ring of the differential polynomials built over the alphabet ΘU with coefficients in K is denoted $R = K\{U\}$. A *ranking* is a total ordering over the set of the derivatives [12, page 75] satisfying the following axioms

1. $\delta v > v$ for each derivative v and derivation δ,
2. $v > w \Rightarrow \delta v > \delta w$ for all $v, w \in \Theta U$ and derivation δ.

Let us fix a ranking. The infinite alphabet ΘU gets ordered. Consider a polynomial $p \in R \setminus K$. Then the leader, initial, ... of p are well defined. Axioms of rankings imply that the separant of p is the initial of every proper derivative of p. Let $\mathrm{rank} p = v^d$. A differential polynomial q is said to be *partially reduced* w.r.t. p if no proper derivative of v occurs in q. It is said to be *reduced* w.r.t. p if it is partially reduced w.r.t. p and $\deg(q, v) < d$. A set A of differential polynomials is said to be *differentially triangular* if it is triangular and if its elements are pairwise partially reduced. It is said to be *autoreduced* if its elements are pairwise *reduced*. Autoreduced implies differentially triangular. If A is a set of differential polynomials and v is a derivative then $A_v = \{p \in \Theta A \mid \mathrm{ld}\, p \le v\}$. Thus R_v denotes the set of the differential polynomials with leader less than or equal to v.

One distinguishes the partial reduction algorithm, which is denoted $\mathsf{partial\_rem}$ from the full reduction algorithm, denoted $\mathsf{full\_rem}$. See [12, page 77]. Let q and p be two differential polynomials. The *partial remainder* $\mathsf{partial\_rem}\ (q, p)$ is the pseudoremainder of q by the (infinite) set of all the *proper* derivatives of p. The *full remainder* $\mathsf{full\_rem}\ (q, p)$ is the pseudoremainder of q by the set of all the derivatives of p (including p). Let A be a set of differential polynomials. We denote $\mathsf{partial\_rem}\ (q, A)$ and $\mathsf{full\_rem}\ (q, A)$ respectively the partial remainder and the full remainder of q by all the elements of A. Let $v = \mathrm{ld}\, q$ and $\overline{A} = A \cap R_v$. The partial remainder $\overline{q}$ of q by A is partially reduced w.r.t. all the elements of A and there exists a power product h of elements of $S_{\overline{A}}$ such that $h\, q \equiv \overline{q} \mod (\overline{A}_v)$. The full remainder $\overline{q}$ of q by A is reduced w.r.t. all the elements of A and there exists a power product h of elements of $H_{\overline{A}}$ such that $h\, q \equiv \overline{q} \mod (\overline{A}_v)$.

A pair $\{p_1, p_2\}$ of differential polynomials is said to be a *critical pair* if the leaders of p_1 and p_2 are derivatives of some same differential indeterminate u (say $\mathrm{ld}\, p_1 = \theta_1 u$ and $\mathrm{ld}\, p_2 = \theta_2 u$). Denote θ_{12} the least common multiple between θ_1 and θ_2. One distinguishes the *triangular situation* which arises when $\theta_{12} \ne \theta_1$ and $\theta_{12} \ne \theta_2$ from the *nontriangular* one which arises when $\theta_{12} = \theta_2$ (say). In the last case, the critical pair is said to be a *reduction critical pair*. In the triangular situation, the Δ–polynomial $\Delta(p_1, p_2)$ is

$$\Delta(p_1, p_2) = s_2\, \frac{\theta_{12}}{\theta_1}\, p_1 - s_1\, \frac{\theta_{12}}{\theta_2}\, p_2.$$

In the nontriangular one,

$$\Delta(p_1, p_2) = \mathsf{prem}(p_2,\ \frac{\theta_2}{\theta_1}\, p_1).$$

If A is a set of differential polynomials then critical_pairs(A) denotes all the critical pairs that can be formed with any two different elements of A and $\Delta(A) = \{\Delta(p_1, p_2) \mid \{p_1, p_2\} \in \text{critical\_pairs}(A)\}$. A critical pair $\{p_1, p_2\}$ is said to be *solved* by a system $A = 0,\ S \ne 0$ of differential polynomial equations and inequations if there exists a derivative $v < \theta_{12} u$ such that $\Delta(p_1, p_2) \in (A_v) : (S \cap R_v)^\infty$. Here is a criterion: if $H_A \subset S$ and $\mathsf{full\_rem}\ (\Delta(p_1, p_2), A) = 0$ then the critical pair $\{p_1, p_2\}$ is solved by $A = 0,\ S \ne 0$.

A differential system $A = 0,\ S \neq 0$ of a differential polynomial ring R is said to be a *regular differential system* w.r.t. a ranking $\mathcal{R}$ if (1) A is differentially triangular, (2) S contains the separants of the elements of A and is partially reduced w.r.t. A, (3) A is coherent[4] i.e. every critical pair in critical_pairs(A) is solved by $A = 0,\ S \neq 0$.

A subset C of a differential ideal $\mathfrak{a}$ is said to be a *characteristic set* of $\mathfrak{a}$ if C is autoreduced and $\mathfrak{a}$ contains no nonzero polynomial reduced w.r.t. C. If C is a characteristic set of $\mathfrak{a}$ and H_C contains no zero divisor in the quotient ring $R/\mathfrak{a}$ then $\mathfrak{a} = [C] : H_C^\infty$ and $p \in \mathfrak{a} \Leftrightarrow$full_rem $(p, C) = 0$. This is the case when $\mathfrak{a}$ is prime.

For algorithmic notations, we use pseudocode notation and an imperative programming style: we allow functions to modify some of their arguments. For this purpose, we put (as in ADA) the keywords in, out and in out in front of the formal parameters of the functions. The keyword in means that the formal parameter is readable only ; the keyword out means that the formal parameter is writable only ; the keyword in out means that the formal parameter is both readable and writable.

In the different algorithms, we often write the test $p \in \mathfrak{a}$ where p is some differential polynomial. Since a characteristic set C of $\mathfrak{a}$ is known, the test can translate as full_rem $(p, C) = 0$. These full reductions can be very expensive in practice and can be replaced by more subtle tests using first some inexpensive criteria (e.g. full reductions after evaluating derivatives which are not derivatives of leaders over $\mathbb{Z}/n\mathbb{Z}$ for some prime integer n).

3. THE OLD ALGORITHM

The Rosenfeld–Gröbner algorithm, which is implemented in the `diffalg` package of MAPLE VR5, solves the problem under consideration in this paper. See [6] for proofs and [4] for the pseudocode of Rosenfeld–Gröbner. In the general case, Rosenfeld–Gröbner needs to split the solutions of the current system which annihilate some differential polynomial p from the solutions which do not annihilate p. The differential polynomial p is usually the initial or the separant of some differential polynomial used with Ritt's reduction. When the characteristic set C of $\mathfrak{a}$ (prime) is known, only one branch needs being considered: the first one if $p \in \mathfrak{a}$; the second one if $p \notin \mathfrak{a}$. The specialized_Rosenfeld_Gröbner function is a specialized pseudocode of Rosenfeld–Gröbner. It handles quadruples $\langle A, D, P, S\rangle$ where A is the set of the differential polynomial equations already processed, D is the set of the critical pairs to be processed, P is the set of the differential polynomial equations to be processed, S is the set of the differential polynomial inequations. Before stating the loop invariants, we need to give the definition of a critical pair *nearly solved* by a quadruple $\langle A, D, P, S\rangle$. Two axioms are enough if the basic complete subfunction is used ; a third one is necessary if the advanced one is used.

1. every critical pair solved by $A \cup \Delta(D) = 0,\ S \neq 0$ is nearly solved by $\langle A, D, P, S\rangle$,
2. every critical pair in D is nearly solved by $\langle A, D, P, S\rangle$.

[4]Same definition as that of Rosenfeld [22] if $H_A \subset S$.

Loop invariants

1. $\mathfrak{a} = [A \cup \Delta(D) \cup P] : S^\infty$,
2. the set of ranks of A is autoreduced,
3. every critical pair in critical_pairs(A) is nearly solved by $\langle A, D, P, S\rangle$ (useful for PDE only),
4. $H_A \subset S$.

```
function specialized_Rosenfeld_Gröbner(in C, R, R̄)
begin
  ⟨A, D, P, S⟩ := ⟨∅, ∅, C, H_C⟩
      (H_C is computed w.r.t. R)
      (from now on, the ranking R̄ is implicitely used)
  while D ≠ ∅ or P ≠ ∅ do
    take and remove a new equation p ∈ P or a
          critical pair {p1, p2} ∈ D. In the latter
          case let p = Δ(p1, p2)
    p := full_rem (p, A)
    p := old_ensure_rank(p, C, P, S)
    if p ≠ 0 then
      ⟨A, D, P, S⟩ := complete(⟨A, D, P, S⟩, p)
    fi
  od
  return specialized_regCharacteristic(A, S)
end
```

At the end of the main loop, $\langle A, D, P, S\rangle = \langle A, \emptyset, \emptyset, S\rangle$ is such that A is coherent (because of the emptyness of D and the third loop invariant). The specialized_regCharacteristic function, described in appendix, partially reduces S w.r.t. A and makes the elements of A pairwise partially reduced. It thereby gets a regular differential system $\overline{A} = 0,\ \overline{S} \neq 0$ such that $\mathfrak{a} = [\overline{A}] : \overline{S}^\infty$ and then computes the desired characteristic set $\overline{C}$ by a purely algebraic method. Remark that the specialized_regCharacteristic function is a specialized version of [7, regCharacteristic] but that, in the different versions of the `diffalg` package the computation is performed with different (less efficient) algorithms based on Gröbner bases computations [6, 10]. The following function simplifies p as long as its initial or its separant lies in $\mathfrak{a}$.

```
function old_ensure_rank(in p, C, in out P, S)
begin
  while p ∉ K and (i_p ∈ 𝔞 or s_p ∈ 𝔞) do
    if i_p ∈ 𝔞 then
      P := P ∪ {i_p}
      p := reductum(p)
    else
      P := P ∪ {s_p}
      S := S ∪ {i_p}
      p := d p − v s_p where v^d = rank p
    fi
  od
  return p
end
```

The following complete function is basic. It enlarges A with the new equation p and removes from A the equations the leader of which is a derivative of the one of p (so that it preserves the second loop invariant of the calling function). It enlarges D with the critical pairs between p and the equations of A. It enlarges S with the initial and the separant of p. Let us observe that the equations removed from A now belong to reduction critical pairs whence this function does not modify the differential ideal, in the sense that

$$[A \cup \{p\} \cup \Delta(D) \cup P] : S^{\infty} = [A' \cup \Delta(D') \cup P'] : S'^{\infty}.$$

function complete(in$\langle A,\ D,\ P,\ S\rangle,\ p$)
begin
  $A' := \{p\} \cup \{q \in A \mid \mathrm{ld}\, q$ is not a derivative of $\mathrm{ld}\, p\}$
  $D' := D \cup \{\{q,\ p\} \mid q \in A$ and $\{q,\ p\}$ is a critical pair$\}$
  $P' := P$
  $S' := S \cup \{i_p,\ s_p\}$
  return $\langle A',\ D',\ P',\ S'\rangle$
end

3.1 The algebraic subproblems

We say that the specialized_Rosenfeld_Gröbner function encounters an "*algebraic subproblem*" when the differential polynomial p (let's denote it $p = p_2$) provided to complete has the same leader as some differential polynomial $p_1 \in A$. Let's look at complete to see how specialized_Rosenfeld_Gröbner behaves in that case. Assume $\mathrm{rank} p_1 = v^{d_1}$ and $\mathrm{rank} p_2 = v^{d_2}$. Since $p_1 \in A$ and p_2 is reduced w.r.t. A we have $d_2 < d_1$. The differential polynomial p_2 is stored in A, the differential polynomial p_1 is removed from A and the pair $\{p_1, p_2\}$ is stored in D. After a few loops, specialized_Rosenfeld_Gröbner extracts the pair $\{p_1, p_2\}$ from D and computes $\Delta(p_1, p_2)$. Since p_1 and p_2 have the same leader we have

$$\Delta(p_1, p_2) = \mathsf{prem}(p_1, p_2, v).$$

Denote p_3 this pseudoremainder and assume $\mathrm{ld}\, p_3 = v$. Then $\mathrm{rank} p_3 = v^{d_3}$ with $d_3 < d_2$. Arguing as above we see that p_3 is stored in A, that p_2 is removed from A and that the pair $\{p_2, p_3\}$ is stored in D. Let's summarize: the specialized_Rosenfeld_Gröbner function starts computing a very basic (bad) pseudoremainder sequence when it encounters an algebraic subproblem.

$$p_3 = \mathsf{prem}(p_1, p_2, v), \quad p_4 = \mathsf{prem}(p_2, p_3, v), \quad \ldots$$

Moreover, at each step i

- many critical pairs between the current pseudoremainder p_i and the other elements of A are generated (not only the reduction pair with p_{i-1}),
- the separant of p_i (not only the initial) is stored in S.

The points above come from differential considerations. We shall see they are useless. The analysis done in this section holds with the advanced version of complete too.

4. THE NEW ALGORITHM

The PARDI algorithm identifies as such the algebraic subproblems arising during the differential treatment. It optimizes the computation of the pseudoremainder sequence and avoids completely the computations due to the pointless differential considerations described above. Assume again that the new differential polynomial $p = p_2$ has the same leader as some $p_1 \in A$. PARDI relies on two key ideas.

First, replacing both p_1 and p_2 by their "*gcd*" g in $(R^-/\mathfrak{a}^-)[v]$ where $R^- = K[w \in \Theta U \mid w < v]$ and $\mathfrak{a}^- = \mathfrak{a} \cap R^-$. Actually g is the "*lsr*" i.e. the "*last nonzero subresultant*" of p_1 and p_2 in $(R^-/\mathfrak{a}^-)[v]$. It is also one of the gcds of p_1 and p_2 in $G[v]$ where G denotes the field of fractions of $R^-/\mathfrak{a}^-$. For legibility however, we shall speak of the gcd of p_1 and p_2 in $(R^-/\mathfrak{a}^-)[v]$ though this is slightly incorrect. We point out that the differential polynomial g always has positive degree in v (property (iii) of lsr1).

Second, applying a "*master—student*" relationship between C and A: every quantity which is zero in $R^-/\mathfrak{a}^-$ (i.e. reduced to zero by C) but not reduced to zero by A is stored in the list P of the equations to be processed. Roughly speaking, when P is empty then the "student" A is able to reproduce all the computations first performed by the "master" C.

4.1 Computing the gcd (the lsr, sorry)

The key ideas are the following.

1. Starting from a (good) algorithm for computing a pseudoremainder sequence. We choose the algorithm of Lionel Ducos [9] but the [15] algorithm would fit as well.

2. Verify at each step i that the leading coefficient of the current subresultant p_i is nonzero in $R^-/\mathfrak{a}^-$.

   If it is nonzero, just continue the [9] algorithm. Do not try to normalize this leading coefficient in any sense (idea explained below).

   If it is zero, restart the computation of the pseudoremainder sequence between the former subresultant p_{i-1} and reductum(p_i).

3. Do not store in S the separants of the subresultants but only their initials. Do not generate critical pairs.

The first and the third points show the advantages of the new algorithm w.r.t. the old one: the growth of the coefficients is controlled with the efficient way of [9]; the differential treatments are completely avoided.

Let's come back to the second point. This is a very simple but very important idea, already applied in [16], which permits to perform the [9] algorithm in the ring $(R^-/\mathfrak{a}^-)[v]$. The only difficult operation is the exact quotient computation in $R^-/\mathfrak{a}^-$. By not normalizing at all the leading coefficients of the subresultants we can perform it as if we were computing in $R^-[v]$, which is easy. *For pseudoremainder sequence algorithms, the most convenient choice is to represent the residue classes of the coefficients with representatives which make the exact quotient operation easy to perform.* The only precaution to take is to make sure that the leading coefficient of each subresultant p_i is nonzero. When it is, one theoretically could go on the [9] algorithm after replacing p_i by its reductum but the trick explained above would not work anymore: exact quotient operations would become very tricky to perform in $R^-/\mathfrak{a}^-$. Another much simpler possibility[5] consists in restarting the [9] algorithm over p_{i-1} and the reductum of p_i. This is what we do. However let us observe that in the context of [17], an algorithm is given which performs efficiently exact quotients in the ring $R^-/\mathfrak{a}^-$ by normalizing polynomials in the sense of [14]. The adaptation of that strategy to the general case does not seem to be efficient in practice [2].

Let's consider the implementation. The code is a slight modification of [8] and [16]. Functions Lazard2 and nsr can be found in [8, Lazard2 and next_sousResultant2 functions].

The parameters p and q satisfy the following properties: they both have the same leader v and belong to $\mathfrak{a}$. Their initials and separants do not belong to $\mathfrak{a}$. The parameters A, P and S are components of a quadruple as for specialized_Rosenfeld_Gröbner. In particular, the initials of the elements of A belong to S. Moreover, p and q are partially reduced w.r.t. A. The last nonzero subresultant g returned by the function satisfies the following property:

(i) the leading coefficient of g w.r.t. v does not lie in $\mathfrak{a}$.

All the subresultants computed from p and q belong to the ideal (p, q) in the ring $(R^-/\mathfrak{a}^-)[v]$. In other words, they belong to $(p, q) + \mathfrak{a}^-$ in $R^-[v]$ and, since $p, q \in \mathfrak{a}$

(ii) $g \in \mathfrak{a}$

(iii) $\mathrm{ld}\, g = v$ (the resultant of p and q belongs to $\mathfrak{a}$ since it is one of the subresultants ; it belongs to R^- ; it is thus zero in $(R^-/\mathfrak{a}^-)[v]$).

From (i) and (iii) we see that the initial of g does not lie in $\mathfrak{a}$. Moreover, since the separants of p, q do not lie in $\mathfrak{a}$ and since g is a gcd of p, q in $G[v]$ where G denotes the field of fractions of $R^-/\mathfrak{a}^-$, we see that $s_g \notin \mathfrak{a}$.

(iv) the initial and the separant of g do not lie in $\mathfrak{a}$.

The lsr1 function records in P all the coefficients which are zero in $R^-/\mathfrak{a}^-$ but not reduced to zero by A. It stores in S the initials $i_1, \ldots, i_n = i_g$ (after making sure that they do not belong to $\mathfrak{a}$) of the successive computed subresultants. Denote $\mathfrak{j}$ the ideal $((A \cup P) \cap R^-) : (S \cap R^-)^\infty$ where the variables A, P and S are considered at the end of the computation. We claim we have:

(v) $(p, q) \subset (g) : (i_1 \cdots i_n)^\infty$ in $(R^-/\mathfrak{j})[v]$.

Indeed $(p, q) \subset (g) : (i_1 \cdots i_n)^\infty$ in $(R^-/\mathfrak{a}^-)[v]$ (classical property). It suffices thus to prove that every leading coefficient which belongs to $\mathfrak{a}^-$ belongs to $\mathfrak{j}$. These leading coefficients are either recorded in P by ensure_lcoeff1 or reduced to zero by A. In the former case they belong to $(P \cap R^-)$, in the latter case they belong to $(A \cap R^-) : (S \cap R^-)^\infty$ for they are partially reduced w.r.t. A and the initials of the elements of A belong to S. They thus belong to $\mathfrak{j}$. This concludes the proof of the claim. □

```
function lsr1(in p, q, C, A, in out P, S)
begin
  v := ld p
  if deg(p, v) < deg(q, v) then swap p and q fi
  found := false
  while not found do
    δ := deg(p, v) − deg(q, v)
    s := i_q^δ
    S := S ∪ {−i_q}
    (p, q) := (q, prem(p, −q))
    z := p
    rankfall := false
    while not found and not rankfall do
      q := ensure_lcoeff1(q, C, A, P, rankfall)
      if q = 0 then
        found := true
      elif not rankfall then
        S := S ∪ {lcoeff(q, v)}
        δ := deg(p, v) − deg(q, v)
        z := Lazard2(q, lcoeff(q, v), s, δ)
            (computes q (i_q/s)^(δ−1))
        if deg(z, v) = 0 then
          found := true
        else
          (p, q) := (q, nsr(p, q, z, s))
              (computes the next subresultant)
          s := i_z
        fi
      fi
    od
  od
  return z
end
```

One can optimize the following function since $\deg(p, v) = 0$ implies $p \in \mathfrak{a}$.

```
function ensure_lcoeff1(in p, C, A, in out P, out rankfall)
begin
  v := ld p
  rankfall := false
  while p ≠ 0 and lcoeff(p, v) ∈ 𝔞 do
    rankfall := true
    if prem(lcoeff(p, v), A) ≠ 0 then
      P := P ∪ {lcoeff(p, v)}
    fi
    p := reductum(p)
  od
  return p
end
```

4.2 The main function

The PARDI function handles quadruples $\langle A, D, P, S\rangle$ in the same way as specialized_Rosenfeld_Gröbner and maintains the same loop invariants. A difference (which simplifies proofs) with specialized_Rosenfeld_Gröbner: the elements of every critical pair generated by PARDI always have different leaders (i.e. there are no “algebraic” critical pairs). The code

[5] According to Lionel Ducos himself (we take this opportunity to thank him very much for his comments).

can be simplified when $g = q$. One could also perform full remainders instead of partial ones but this would make the [9] algorithm less efficient.

```
function PARDI(in C, R, R̄)
begin
  ⟨A, D, P, S⟩ := ⟨∅, ∅, C, H_C⟩
      (H_C is computed w.r.t. R)
      (from now on, the ranking R̄ is implicitely used)
  while D ≠ ∅ or P ≠ ∅ do
    take and remove a new equation p ∈ P or a
        critical pair {p1, p2} ∈ D. In the latter
        case let p = Δ(p1, p2)
    p := partial_rem (p, A)
    p := ensure_rank(p, C, A, P)
    if p ≠ 0 then
      if ∃q ∈ A such that ld p = ld q then
        g := lsr1(p, q, C, A, P, S)
        ⟨A, D, P, S⟩ := complete(⟨A \ {q}, D, P, S⟩, g)
            (replaces p and q by their "gcd")
      else
        ⟨A, D, P, S⟩ := complete(⟨A, D, P, S⟩, p)
      fi
    fi
  od
  return specialized_regCharacteristic(A, S)
end
```

The following function simplifies the differential polynomial p while its initial and its separant lie in $\mathfrak{a}$.

```
function ensure_rank(in p, C, A, in out P)
begin
  while p ∉ K and (i_p ∈ 𝔞 or s_p ∈ 𝔞) do
    if i_p ∈ 𝔞 then
      if prem(i_p, A) ≠ 0 then P := P ∪ {i_p} fi
      p := reductum(p)
    else
      if prem(s_p, A) ≠ 0 then P := P ∪ {s_p} fi
      p := d p − v s_p where v^d = rank p
    fi
  od
  return p
end
```

4.2.1 *Proof of the first invariant*

We only give a sketched proof of the most interesting issue: proving that the differential ideal is not changed when the algorithm performs the purely algebraic treatment (call to lsr1). We consider the value of the quadruple $\langle A, D, P, S\rangle$ just after lsr1 is run. We claim we have $\mathfrak{a} = [A \cup \{p\} \cup \Delta(D) \cup P] : S^{\infty}$. This relation held before calling lsr1. This function enlarges P with elements of $\mathfrak{a}$ and enlarges S with elements which do not belong to $\mathfrak{a}$ (i.e. non zerodivisors elements modulo $\mathfrak{a}$ for the ideal is prime). This concludes the proof of the claim. □ The different versions of the complete function preserve this property for $g \in \mathfrak{a}$ and $i_g\, s_g \notin \mathfrak{a}$ (properties (ii) and (iv) of lsr1). At the beginning of the next loop, the differential ideal under consideration is $\mathfrak{b} = [(A \setminus \{q\}) \cup \Delta(D) \cup P \cup \{g\}] : S^{\infty}$. We must prove that $\mathfrak{a} = \mathfrak{b}$. The inclusion $\mathfrak{b} \subset \mathfrak{a}$ holds since $g \in \mathfrak{a}$. For the opposite inclusion $\mathfrak{a} \subset \mathfrak{b}$ it suffices to prove that the removed differential polynomials $p, q \in \mathfrak{b}$. According to property (v) of lsr1

$$p, q \in (g) : (i_1 \cdots i_n)^{\infty} \text{ in } (R^-/\mathfrak{j})[v].$$

The ideal $\mathfrak{j} \subset \mathfrak{b}$ for it is generated by polynomials of R^- and the differential polynomials p, q do not belong to that ring. Therefore

$$p, q \in (g) : (i_1 \cdots i_n)^{\infty} \text{ in } (R^-/(\mathfrak{b} \cap R^-))[v].$$

Since $g \in \mathfrak{b}$ and $i_1, \ldots, i_n$ are recorded in S by lsr1 we have

$$(g) : (i_1 \cdots i_n)^{\infty} \subset \mathfrak{b}$$

whence $p, q \in \mathfrak{b}$ and $\mathfrak{a} \subset \mathfrak{b}$. □

4.2.2 *Termination proof*

The rank of A decreases at each loop. This rank cannot strictly decrease infinitely many times (classical property [12, proposition 3, page 81] of autoreduced sets). It thus suffices to prove that this rank cannot be constant infinitely many times. This situation only arises if PARDI calls lsr1 and if $g = q$ or if all the coefficients of the differential polynomial p (viewed as a univariate polynomial in its leader) picked from P lie in $\mathfrak{a}$. In both cases, the algorithm suppresses a differential polynomial of P (i.e. p) and enlarges P with finitely many (possibly none) differential polynomials having leader strictly less than that of p. This cannot happen infinitely many times (classical argument of graph theory[6] [13, Satz 6.6] and the fact that [12, page 75] rankings are well orderings). Thus PARDI stops.

4.2.3 *Variant of algorithm*

It is interesting in practice to keep A as a regular chain in the sense [11, 1]. It allows to take an inequation into account as soon as it arises: either it is regular and A is left unchanged or it is not and A gets smaller. Let us observe that inequations need anyway to be kept until the end: this optimization does not avoid calling specialized_regCharacteristic. Thus, in the worst case, some computation time is lost. However, memory consumption is the most important issue and this variant improves it.

5. EXAMPLES

The following system C is a characteristic set w.r.t. the orderly ranking $\mathcal{R}$

$$\cdots > u_{xx} > u_{xy} > u_{yy} > v_x > v_y > u_x > u_y > v > u.$$

of the prime differential ideal $[C] : H_C^{\infty}$ in the ring $\mathbb{Q}\{u, v\}$ endowed with derivations w.r.t. x and y. Ranks appear on the left-hand side of the equal signs. The denominators of the right hand side are the initials of the differential polynomials.

$$C \left\{ \begin{array}{l} v_{xx} = u_x, \\ v_y = (u_x\, u_y + u_x\, u_y\, u)/(4\, u), \\ u_x^2 = 4\, u, \\ u_y^2 = 2\, u. \end{array} \right.$$

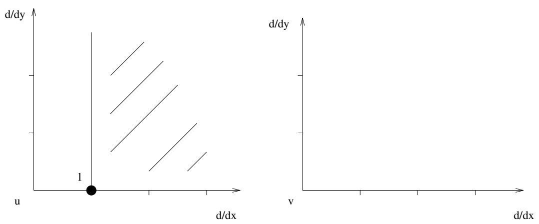
We are looking for a characteristic set $\overline{C}$ of the same prime differential ideal w.r.t. the elimination ranking $\overline{\mathcal{R}}$

$$\cdots > u_x > u_y > u > \cdots > v_{xx} > v_{xy} > v_{yy} > v_x > v_y > v.$$

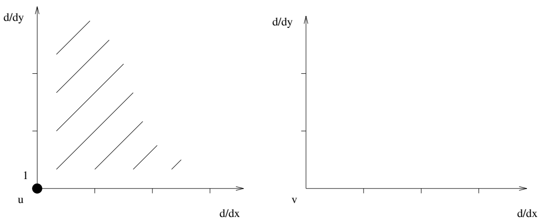
[6] Every infinite tree which is locally finite involves a branch of infinite length.

In the analysis below, we only give the ranks of the differential polynomials. On the diagrams, black circles give the leaders of the elements of A while integers give the degrees of the elements of A w.r.t. their leaders. Initially, $A = D = \emptyset$ and $S = \{4\,u,\, 2\,u_x,\, 2\,u_y\}$ contains H_C w.r.t. $\mathcal{R}$. We have $P = C$. The set of ranks of P (w.r.t. $\overline{\mathcal{R}}$ from now on) is rank$P = \{u_x,\, u_x,\, u_x^2,\, u_y^2\}$. The implemented version of PARDI is a draft version: it picks new equations from P first and critical pairs from D only if $P = \emptyset$ (the list D is not sorted). It was written in MAPLE VI by the second author. It slightly differs from the algorithm presented in this paper for it performs full remainders instead of partial remainders in the main loop. We chose this variant because it makes the analysis slightly shorter over that example.

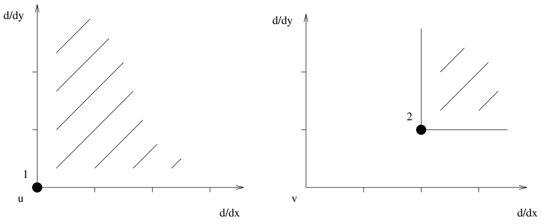
First, an equation with rank u_x is picked and removed from P and stored in A.



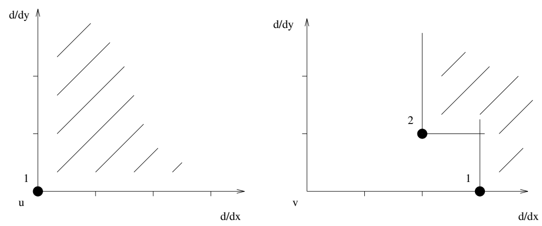
Now rank$P = \{u_x,\, u_x^2,\, u_y^2\}$. A new equation with rank u_x^2 is picked and removed from P. After full reduction, its rank is u. It forms a critical pair with the former equation, which is withdrawn from A.



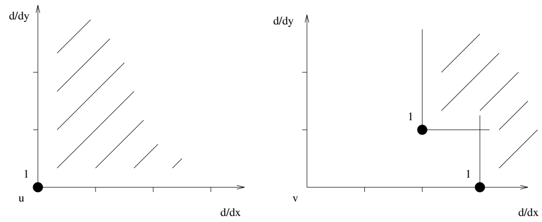
Now rank$P = \{u_x,\, u_y^2\}$ and rank$D = \{\{u,\, u_x\}\}$. A new equation with rank u_y^2 is picked and removed from P. After full reduction, its rank is v_{xxy}^2.



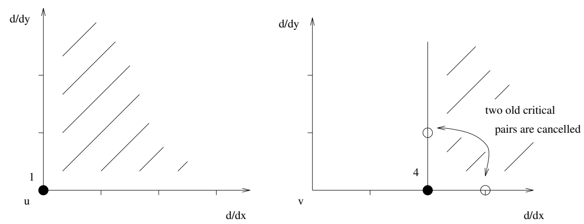
Now rank$P = \{u_x\}$ and rank$D = \{\{u,\, u_x\}\}$. A new equation with rank u_x is picked and removed from P. After full reduction, its rank is v_{xxx}. It forms a critical pair with the former equation.



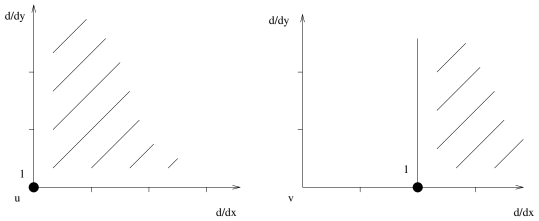
Now P is empty and rank$D = \{\{u,\, u_x\},\, \{v_{xxx},\, v_{xxy}^2\}\}$. The first critical pair is picked and removed from D. After full reduction, the Δ–polynomial has rank v_{xxy}. This is an algebraic subproblem. The situation is very easy for one of the differential polynomials has degree 1 while the other has degree 2 and there is basically no gain w.r.t. the old algorithm here. The gcd (last nonzero subresultant if you prefer) of these polynomials is the one with degree 1. It replaces the degree 2 polynomial in A and generates a critical pair with the differential polynomial with rank v_{xxx}. At the end of the computation of the gcd, the resultant of the two polynomials, which has rank v_{xx}^4 as a differential polynomial but which is zero in the quotient ring $(R^-/(\mathfrak{a} \cap R^-))[v_{xx}]$, was not reduced to zero by A (where $R^- = K[w \in \Theta U \mid w < v_{xx}$ w.r.t. $\overline{\mathcal{R}}])$. It is recorded in P.



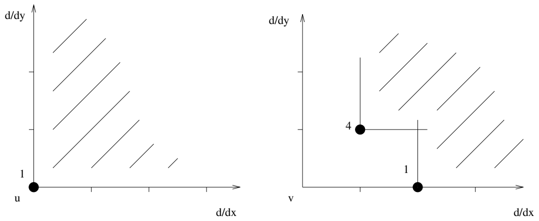
Now rank$P = \{v_{xx}^4\}$, rank$D = \{\{v_{xxx},\, v_{xxy}^2\},\, \{v_{xxx},\, v_{xxy}\}\}$. The resultant which has rank v_{xx}^4 is picked and removed from P. Two new critical pairs are generated between this differential polynomial and two of the differential polynomials in A. These two differential polynomials are withdrawn from A. Implemented in the advanced version of complete, the analogue of Buchberger's second criterion cancels the two old pairs of D.



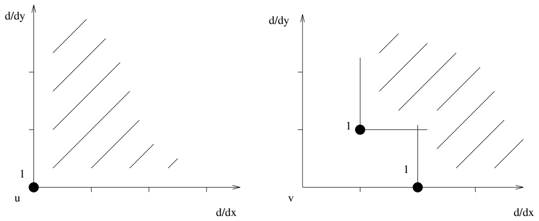
Now P is empty and rank$D = \{\{v_{xxx},\, v_{xx}^4\},\, \{v_{xxy},\, v_{xx}^4\}\}$. The first critical pair is picked and removed from D. After full reduction, the Δ–polynomial has rank v_{xx}^3. This is again an algebraic subproblem. The gcd between this differential polynomial and the element of A with rank v_{xx}^4 has degree 1 in v_{xx}. It replaces the differential polynomial with rank v_{xx}^4. During its computation no critical pair was generated and this is a very important gain w.r.t. specialized_Rosenfeld_Gröbner. At the end of its computation, the resultant which has rank v_{xy}^4 as a differential polynomial but is zero in the quotient ring, was not reduced to zero by A. It is recorded in P. No critical pair is generated.



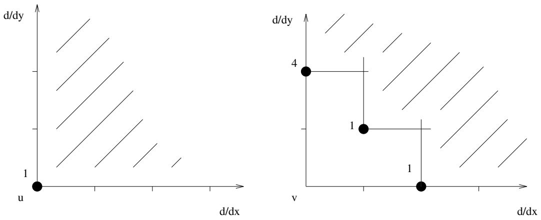
Now rank$P = \{v_{xy}^4\}$ and rank$D = \{\{v_{xxy},\, v_{xx}^4\}\}$. A new equation with rank v_{xy}^4 is picked and removed from P. It is put in A and generates a new critical pair.



P is empty and rank$D = \{\{v_{xx},\, v_{xy}^4\},\, \{v_{xxy},\, v_{xx}^4\}\}$. The first critical pair is picked and removed from D. After full reduction by A, the Δ–polynomial has rank v_{xy}^2. This is a purely algebraic subproblem. The gcd between this differential polynomial and the element of A with rank v_{xy}^4 has rank v_{xy}. It replaces the degree 4 polynomial in A. It generates one critical pair. At the end of its computation, the resultant, which has rank v_{yy}^4 is stored in P.



rank$P = \{v_{yy}^4\}$ and rank$D = \{\{v_{xx},\, v_{xy}\},\, \{v_{xxy},\, v_{xx}^4\}\}$. The resultant is picked and removed from P. It is recorded in A. By the analogue of Buchberger's criterion only one new critical pair is generated (the pair $\{v_{yy}^4,\, v_{xx}\}$ is avoided).



P is empty, rank$D = \{\{v_{xy},\, v_{yy}^4\},\, \{v_{xx},\, v_{xy}\},\, \{v_{xxy},\, v_{xx}^4\}\}$. The three Δ–polynomials are reduced to zero by A. After specialized_regCharacteristic is performed, one gets

$$\overline{C} \left\{ \begin{array}{l} u = v_{yy}^2, \\ v_{xx} = 2\, v_{yy}, \\ v_{xy} = (v_{yy}^3 - v_{yy})/v_y, \\ v_{yy}^4 = 2\, v_{yy}^2 + 2\, v_y^2 - 1. \end{array} \right.$$

Let us observe specialized_Rosenfeld_Gröbner could not carry this example out in the `diffalg` package in MAPLE VR5 (it cannot be carried out with `diffalg` in MAPLE VI because of a bug). Even if we could, the analysis would have been much more painful. Indeed, we have presented our example in terms of gcd and resultant. This makes much more sense than if we had presented it in terms of full remainders using Ritt's reduction.

5.1 Euler's equations for a perfect fluid

Expressed as differential polynomials, Euler's equations for a perfect fluid in two dimensions are

$$\Sigma \left\{ \begin{array}{l} v_t^1 + v^1 v_x^1 + v^2 v_y^1 + p_x = 0, \\ v_t^2 + v^1 v_x^2 + v^2 v_y^2 + p_y = 0, \\ v_x^1 + v_y^2 = 0. \end{array} \right.$$

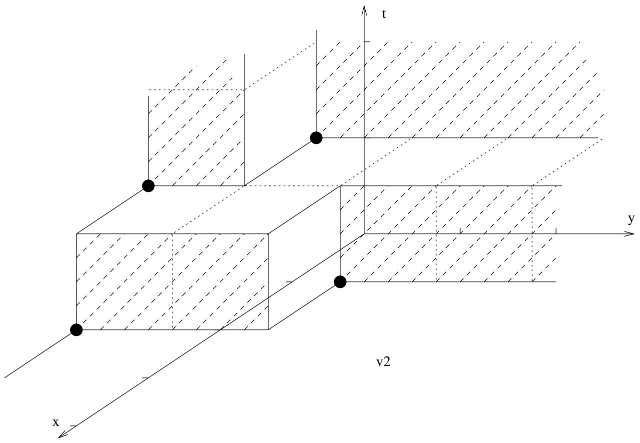
There are three differential indeterminates v^1, v^2 (components of the speed) and the pressure p. They depend on three independent variables x, y (space variables) and the time t. For some orderly ranking, Rosenfeld–Gröbner gets with nearly no computation the characteristic set

$$C \left\{ \begin{array}{l} p_{xx} = -2\, v_x^2\, v_y^1 - 2\,(v_y^2)^2 - p_{yy}, \\ v_t^1 = -v^2\, v_y^1 - p_x + v_y^2\, v^1, \\ v_x^1 = -v_y^2, \\ v_t^2 = -v^1\, v_x^2 - v^2\, v_y^2 - p_y. \end{array} \right.$$

For the elimination ranking $(p,\, v^1) \gg$ degrevlex(v^2) with $t > x > y$ the implementation of PARDI was able to compute a characteristic set of the differential prime ideal. This characteristic set cannot be written in this paper (the computer file is 600 kilobytes large). There are 7 equations involving more than 50 different derivatives. Intermediate computations took more than 500 megabytes on the computers of the UMS MEDICIS (GAGE Laboratory, of the École Polytechnique). We have

$$\mathrm{rank}\overline{C} = \{p_x,\, p_y,\, v^1,\, v_{xxxxt}^2,\, v_{xxxtt}^2,\, v_{xxytt}^2,\, v_{xxxyyt}^2\}.$$

The diagram of the differential indeterminate v^2 is[7]



This example could not be previously solved by any other algorithm. A remaining challenge (given by Pommaret) consists in computing a characteristic set for a ranking eliminating v^1 and v^2.

Conclusion

We have described a new algorithm for converting a characteristic set of a prime differential ideal from one ranking

[7]The authors would like to thank Marc Giusti for his help.

into another. As far as we know, it is the first time that algebraic subproblems are clearly identified during differential computations (i.e. before Rosenfeld's lemma applies). It is also the first time that advanced practical algorithmic methods coming from both polynomial algebra (gcd computation over quotient rings) and differential algebra (analogue of Buchberger's criteria) are merged.

6. REFERENCES

[1] AUBRY, P., LAZARD, D., AND MORENO MAZA, M. On the theories of triangular sets. *J. of Symb. Comp. 28* (1999), 105–124.

[2] AUBRY, P., AND MORENO MAZA, M. Triangular sets for solving polynomial systems: A comparative implementation of four methods. *J. of Symb. Comp. 28*, 1–2 (1999), 125–154.

[3] BOULIER, F. Efficient computation of regular differential systems by change of rankings using Kähler differentials. Tech. rep. LIFL 1999-14, presented at MEGA2000, 1999.

[4] BOULIER, F. *Triangularisation de systèmes différentiels.* série IC2. Hermès. (to publish).

[5] BOULIER, F., LAZARD, D., OLLIVIER, F., AND PETITOT, M. Representation for the radical of a finitely generated differential ideal. In *proc. of ISSAC'95*, 158–166.

[6] BOULIER, F., LAZARD, D., OLLIVIER, F., AND PETITOT, M. Computing representations for radicals of finitely generated differential ideals. (tech. rep. IT306 of the LIFL, dec. 1998 vers. in [20]).

[7] BOULIER, F., AND LEMAIRE, F. Computing canonical representatives of regular differential ideals. In *proc. of ISSAC 2000*, 37–46.

[8] DUCOS, L. source of the axiom package pseudoremaindersequence, 1995. (updated 1999).

[9] DUCOS, L. Optimizations of the subresultant algorithm. *J. of P. and A. Alg. 145* (2000), 149–163.

[10] HUBERT, É. Factorization free decomposition algorithms in differential algebra. *J. of Symb. Comp. 29*, 4,5 (2000), 641–662.

[11] KALKBRENER, M. A Generalized Euclidean Algorithm for Computing Triangular Representations of Algebraic Varieties. *J.S.C. 15* (1993), 143–167.

[12] KOLCHIN, E. R. *Differential Algebra and Algebraic Groups.* Academic Press, New York, 1973.

[13] KÖNIG, D. *Theorie der endlichen und unendlichen Graphen.* Chelsea publ. Co., New York, 1950.

[14] LAZARD, D. A new method for solving algebraic systems of positive dimension. *Disc. App. Math. 33* (1991), 147–160.

[15] LOMBARDI, H., ROY, M.-F., AND SAFEY EL DIN, M. New structure theorem for subresultants. *J. of Symb. Comp. 29*, 4,5 (2000), 663–690.

[16] MORENO MAZA, M. On Triangular Decompositions of Algebraic Varieties. NAG Tech. rep. 1999. (presented at MEGA2000).

[17] MORENO MAZA, M., AND RIOBOO, R. Polynomial gcd computations over towers of algebraic extensions. In *Proc. of AAECC11* (1995), Springer Verlag.

[18] MORRISON, S. The Differential Ideal $[P] : M^\infty$. *J. of Symb. Comp. 28* (1999), 631–656.

[19] OLLIVIER, F. *Le problème de l'identifiabilité structurelle globale : approche théorique, méthodes effectives et bornes de complexité.* PhD thesis, École Polytechnique, 1990.

[20] PETITOT, M. Quelques méthodes de Calcul Formel appliquéees à l'étude des équations différentielles, Feb. 1999. Mémoire d'habilitation à diriger des recherches, Univ. Lille I, LIFL.

[21] RITT, J. F. *Differential Algebra.* Dover Publications Inc., New York, 1950.

[22] ROSENFELD, A. Specializations in differential algebra. *Trans. Amer. Math. Soc. 90* (1959), 394–407.

Appendix

The advanced complete function is described in [6]. It applies an analogue of Buchberger's second criterion. Pseudocode can be found in [4]. The specialized_Rosenfeld_Gröbner function is a variant of [7, regCharacteristic]. Parameters satisfy the properties : $\mathfrak{a} = [A] : S^\infty$ and $\mathfrak{a} \cap K[N] = (0)$ (by [5; 18, Lazard's lemma]) where N denotes the derivatives which are not derivatives of any leader of A. The function builds a characteristic set $\overline{C}$ of $\mathfrak{a}$. In particular, $\overline{C}$ is a triangular set satisfying $\mathfrak{a} = [\overline{C}] : H_{\overline{C}}^\infty$. Here are invariants of all the loops.

1. $\mathfrak{a} = [\overline{A} \cup \overline{C}] : (\overline{S} \cup H_{\overline{C}})^\infty$
2. $\mathrm{rank}(\overline{A} \cup \overline{C}) = \mathrm{rank} A$; moreover, every leader of $\overline{A}$ is greater than every leader of $\overline{C}$.
3. $\overline{C}$ is a regular chain[8] in the sense of [11, 1].

At the end of the computation, $\overline{C}$ is a characteristic set of $(A) : S^\infty$ whence it is a characteristic set[9] of $[A] : S^\infty$. Observe that C is not provided to specialized_regCharacteristic as a parameter since it is not necessary in order to choose the right branch in the splitting tree.

function specialized_regCharacteristic(in A, S)
begin
  $\overline{A} := A$
  $\overline{S} := S \setminus K$
  $\overline{C} := \emptyset$
  while $\overline{A} \neq \emptyset$ do

[8] A polynomial is said to be *regular* w.r.t. a triangular set $\{p_1 < \cdots < p_n\}$ if it belongs to none of the associated prime ideals of $(p_1, \ldots, p_n) : (i_{p_1}, \ldots, i_{p_n})^\infty$. A triangular set $\{p_1 < \cdots < p_n\}$ is said to be a *regular chain* if $n = 1$ or i_{p_ℓ} is regular w.r.t. $\{p_1, \ldots, p_{\ell-1}\}$ for each $2 \leq \ell \leq n$.

[9] $\overline{C}$ is necessarily coherent at the end of the computation but not necessarily during the computation.

```
    take p ∈ A̅ with minimal leader
    while ∃s ∈ S̅ such that ld s ≤ ld p do
      S̅ := S̅ \ {s}
      regularize(partial_rem (s, C̅), C̅)
    od
    A̅ := A̅ \ {p}
    C̅ := C̅ ∪ {p}
  od
  while S̅ ≠ ∅ do
    take and remove s ∈ S̅
    regularize(partial_rem (s, C̅), C̅)
  od
  return C̅
end
```

The functions regularize and lsr0 recursively call each other. They are purely algebraic (i.e. nondifferential) functions much inspired from [16]. Computations are performed in "dimension zero". Both transform the regular chain $\overline{C}$ as a regular chain $\overline{C}'$ with the same set of leaders as $\overline{C}$ and s. t.

(a) $(\overline{C}) : I_{\overline{C}}^{\infty} \subset (\overline{C}') : I_{\overline{C}'}^{\infty} \subset \mathfrak{a}$;

(b) the prime ideals which are minimal over $(\overline{C}') : I_{\overline{C}'}^{\infty}$ are minimal over $(\overline{C}) : I_{\overline{C}}^{\infty}$.

The parameters of regularize are $\overline{C}$ and a polynomial $p \notin (\overline{C}) : I_{\overline{C}}^{\infty}$. It transforms $\overline{C}$ into a regular chain $\overline{C}'$ which satisfies the following property in addition of (a) and (b):

(c) p is regular w.r.t. $(\overline{C}') : I_{\overline{C}'}^{\infty}$.

Splittings are avoided in regularize: when $\deg(g, v) \neq 0$, a "pseudofactorization" of q is discovered i.e. for some nonnegative integer α we have $i_g^{\alpha}\, q = g\ \mathsf{pquo}(q, g)$. Then g is necessarily the factor to throw away (the one which does not belong to $\mathfrak{a}$). Last, the loop could be optimized by performing pseudoquotients while the pseudoremainder is zero.

```
function regularize(in p, in out C̅)
begin
  if p ∉ K then
    regularize(i_p, C̅)
    if ∃q ∈ C̅ such that ld q = ld p then
      q' := q
      C̅ := C̅ \ {q}
      g := lsr0(p, q', C̅)
      while deg(g, ld p) > 0 do
        q' := pquo(q', g)
        g := lsr0(p, q', C̅)
      od
      C̅ := C̅ ∪ {q'}
    fi
  fi
end
```

The parameters of lsr0 are two polynomials p, q having the same leader v and with leading coefficients w.r.t. v which are regular w.r.t. $(\overline{C}) : I_{\overline{C}}^{\infty}$. It transforms $\overline{C}$ into a regular chain $\overline{C}'$ and returns a polynomial g which satisfies the two following properties in addition of (a) and (b). Denote $\mathfrak{j} = (\overline{C} \cap R^-) : I_{\overline{C} \cap R^-}^{\infty}$ and $\mathfrak{j}' = (\overline{C}' \cap R^-) : I_{\overline{C}' \cap R^-}^{\infty}$ where R^- is the ring of the polynomials of leaders strictly less than v.

(d) g is the last nonzero subresultant of p and q in $(R^-/\mathfrak{j}')[v]$

(e) the leading coefficients $i_1, \ldots, i_n$ w.r.t. v of the successive subresultants computed by lsr0 are regular w.r.t. $(\overline{C}') : I_{\overline{C}'}^{\infty}$. In particular, the polynomial g is itself regular if it is the resultant of p and q.

```
function lsr0(in p, q, in out C̅)
begin
  v := ld p
  if deg(p, v) < deg(q, v) then swap p and q fi
  found := false
  while not found do
    if deg(q, v) = 0 then
      found := true
      z := q^deg(p,v)
    else
      δ := deg(p, v) − deg(q, v)
      s := i_q^δ
      (p, q) := (q, prem(p, −q))
      z := p
    fi
    rankfall := false
    while not found and not rankfall do
      q := ensure_lcoeff0(q, C̅, rankfall)
      if q = 0 then
        found := true
      else
        regularize(lcoeff(q, v), C̅)
        if not rankfall then
          δ := deg(p, v) − deg(q, v)
          z := Lazard2(q, lcoeff(q, v), s, δ)
          if deg(z, v) = 0 then
            found := true
          else
            (p, q) := (q, nsr(p, q, z, s))
            s := i_z
          fi
        fi
      fi
    od
  od
  return z
end

function ensure_lcoeff0(in p, C̅, out rankfall)
begin
  v := ld p
  rankfall := false
  while p ≠ 0 and lcoeff(p, v) ∈ (C̅) : I_C̅^∞ do
    rankfall := true
    p := reductum(p)
  od
  return p
end
```