Constructive D-module Theory with SINGULAR

Daniel Andres Michael Brickenstein Viktor Levandovskyy
Jorge Martín-Morales Hans Schönemann

Abstract

We overview numerous algorithms in computational D-module theory together with the theoretical background as well as the implementation in the computer algebra system SINGULAR. We discuss new approaches to the computation of Bernstein operators, of logarithmic annihilator of a polynomial, of annihilators of rational functions as well as complex powers of polynomials. We analyze algorithms for local Bernstein-Sato polynomials and also algorithms, recovering any kind of Bernstein-Sato polynomial from partial knowledge of its roots. We address a novel way to compute the Bernstein-Sato polynomial for an affine variety algorithmically. All the carefully selected nontrivial examples, which we present, have been computed with our implementation. We address such applications as the computation of a zeta-function for certain integrals and revealing the algebraic dependence between pairwise commuting elements.

Mathematics Subject Classification (2010). 13P10, 14F10, 68W30.

Keywords. D-modules, non-commutative Gröbner bases, annihilator ideal, b-function, Bernstein-Sato polynomial, Bernstein-Sato ideal.

1 Introduction

Constructive D-module theory has been dynamically developing throughout the last years. There are new approaches, algorithms, implementations and applications. Our work on the implementation of procedures for D-modules started in 2003, motivated among other factors by challenging elimination problems in non-commutative algebras, which appear e. g. in algorithms for computing Bernstein-Sato polynomials. We reported on solving several challenges in [20]. A non-commutative subsystem SINGULAR:PLURAL [14] of the computer algebra system SINGULAR provides a user with possibilities to compute numerous Gröbner bases-based procedures in a wide class of non-commutative G-algebras [22]. It was natural to use this functionality in the context of computational D-module theory. Nowadays we present a D-module suite in SINGULAR consisting of the libraries `dmod.lib`, `dmodapp.lib`, `dmodvar.lib` and `bfun.lib`. There are many useful and flexible procedures for various aspects of D-module theory. These libraries are freely distributed together with SINGULAR [11].

There are several implementations of algorithms for D-modules, namely the experimental program KAN/SM1 by N. Takayama [38], the `bfct` package in RISA/ASIR [31] by

M. Noro [30] and the package `Dmodules.m2` in MACAULAY2 by A. Leykin and H. Tsai [40]. We aim at creating a D-module suite, which will combine flexibility and rich functionality with high performance, being able to treat more complicated examples.

In this paper we do not present any comparison between different computer algebra systems in the realm of D-modules, referring to [20] and [1]. However, comparison in the latter articles shows, that our implementation is superior to KAN/SM1 and MACAULAY2 and in many cases more powerful than RISA/ASIR.

Here is the list of problems we address in this paper:

- s-parametric annihilator of f (Section 3, see also [20, 1]),
- annihilator of f^α for $\alpha \in \mathbb{C}$ (Section 4, see also [34]),
- annihilator of a polynomial function f and of a rational function f/g (Section 4),
- b-function with respect to weights for an ideal (Section 5, see also [1]),
- global and local Bernstein-Sato polynomials of f (Section 6),
- partial knowledge of Bernstein-Sato polynomial (Section 6.4, see also [20]),
- Bernstein operator of f (Section 7),
- logarithmic annihilator of f (Section 8),
- Bernstein-Sato ideals for $f = f_1 \cdot \ldots \cdot f_m$ (Section 9, see also [20]),
- annihilator and Bernstein-Sato polynomial for a variety (Section 10, see also [1]).

We describe both theoretical and implementational aspects of the problems above and illustrate them with carefully selected nontrivial examples, computed with our implementation in SINGULAR. In Section 3.3, we give yet another alternative proof for the algorithm by Briançon-Maisonobe for computing $\mathrm{Ann}_{D_n[s]}(f^s)$, presented in [1]. Notably, this delivers additional structural information. In Section 7, we compare several approaches for the computation of Bernstein operators. Using the method of principal intersection, we formalize several methods for computing Bernstein-Sato polynomials. Following Budur et al. [7] and [1], we report on the implementation of two methods for the computation of Bernstein-Sato polynomials for affine varieties in a framework, which is a natural generalization of our approach to the algorithm by Briançon-Maisonobe.

Notations. Throughout the article $\mathbb{K}$ is assumed to be a field of characteristic zero. By R we denote the polynomial ring $\mathbb{K}[x_1, \ldots, x_n]$ and by $f \in R$ a non-constant polynomial.

We consider the n-th Weyl algebra as the algebra of linear partial differential operators with polynomials coefficients. That is $D_n = D(R) = \mathbb{K}\langle x_1, \ldots, x_n, \partial_1, \ldots, \partial_n \mid \{\partial_i x_i = x_i \partial_i + 1, \partial_i x_j = x_j \partial_i, i \neq j\}\rangle$. We denote by $D_n[s] = D(R) \otimes_{\mathbb{K}} \mathbb{K}[s_1, \ldots, s_n]$ and drop the index n depending on the context.

The ring R is a natural $D_n(R)$-module with the action

$$x_i \bullet f(x_1, \ldots, x_n) = x_i \cdot f(x_1, \ldots, x_n), \quad \partial_i \bullet f(x_1, \ldots, x_n) = \frac{\partial f(x_1, \ldots, x_n)}{\partial x_i}.$$

Working with monomial orderings in elimination, we use the notation $x \gg y$ for "x is greater than any power of y".

Given an associative $\mathbb{K}$-algebra A and some monomial well-ordering on A, we denote by $\mathrm{lm}(f)$ (resp. $\mathrm{lc}(f)$) the leading monomial (resp. the leading coefficient) of $f \in A$. Given a left Gröbner basis $G \subset A$ and $f \in A$, we denote by $\mathrm{NF}(f, G)$ the normal form of f with respect to the left ideal ${}_A\langle G\rangle$. We also use the shorthand notation $h \to_H f$ (and $h \to f$, if H is clear from the context) for the reduction of $h \in A$ to $f \in A$ with respect to the set H. If not specified, under *ideal* we mean *left ideal.* For $a, b \in A$, we use the Lie bracket notation $[a, b] := ab - ba$ as well as the skew Lie bracket notation $[a, b]_k := ab - k \cdot ba$ for $k \in \mathbb{K}^*$.

It is convenient to treat the algebras we deal with in a bigger framework of G-algebras of Lie type.

Definition 1.1. Let A be the quotient of the free associative algebra $\mathbb{K}\langle x_1, \ldots, x_n\rangle$ by the two-sided ideal I, generated by the finite set $\{x_j x_i - x_i x_j - d_{ij}\}$ $\forall 1 \le i < j \le n$, where $d_{ij} \in \mathbb{K}[x_1, \ldots, x_n]$. A is called a *G–algebra of Lie type* [22], if
- $\forall\ 1 \le i < j < k \le n$ the expression $d_{ij}x_k - x_k d_{ij} + x_j d_{ik} - d_{ik}x_j + d_{jk}x_i - x_i d_{jk}$ reduces to zero modulo I and,
- there exists a monomial ordering $\prec$ on $\mathbb{K}[x_1, \ldots, x_n]$, such that $\mathrm{lm}(d_{ij}) \prec x_i x_j$, $\forall i < j$.

G-algebras are also known as *algebras of solvable type* [17, 23] and *PBW algebras* [8]. We often use the following.

Lemma 1.2 (Generalized Product Criterion, [22])**.** *Let A be a G-algebra of Lie type and $f, g \in A$. Suppose* $\mathrm{lm}(f)$ *and* $\mathrm{lm}(g)$ *have no common factors, then* $\mathrm{spoly}(f, g) \to_{\{f,g\}} [f, g]$.

2 Challenges for Gröbner bases engines of Singular

Since the very beginning of implementation of algorithms for D-modules in Singular there have been intensive interaction with the developers of Singular. Numerous challenging examples and open problems from constructive D-module theory were approached both on the level of libraries and in the kernel of Singular and Singular:Plural. This resulted in several enhancements in kernel procedures and, among other, motivated M. Brickenstein to develop and implement the generalization of his `slimgb` [6] (slim Gröbner basis) algorithm to non-commutative G-algebras. Indeed `slimgb` is a variant of Buchberger's algorithm. It is designed to keep polynomials *slim*, that is short with small coefficients. The algorithm features parallel reductions and a strategy to minimize the weighted lengths of polynomials. A weighted length function of a polynomial can be seen as measure for the intermediate expression swell and it can consider not only the number of terms in a polynomial, but also their coefficients and degrees. Considering the degrees of the terms inside the polynomials, `slimgb` can often directly (that is, without using Gröbner Walk or similar algorithms) compute Gröbner bases with respect to e. g. elimination orderings. The procedure `slimgb` demonstrated very good performance on examples from the realm of D-modules [20], which require computations with elimination orderings.

As it will be seen in the paper, various computational questions, arising in D-module theory, use much more than Gröbner bases only. Among other, a transformation matrix between two bases (called LIFT in [15]), the kernel of a module homomorphism (called MODULO in [15]) and so on must be applied for complicated examples. On the other hand, the standard `std` routine for Gröbner bases, generalized to non-commutative G-algebras, is used together with `slimgb` for a variety of problems. Since the beginning of development of the D-module suite in SINGULAR, these functions have been enhanced: they became much faster and more flexible. The effect of the use of the generalized Chain Criterion (cf. [15]) in Gröbner engines is even bigger in the non-commutative case, due to the discarding of multiplications, which complexity is increased, compared with the commutative case. On the contrary, the generalized Product Criterion (Lemma 1.2) plays a minor role in the implementation, since the complete discarding of a pair generalizes to the computation of a Lie bracket of the pair members.

The concept of *ring list*, introduced in SINGULAR in 2004, enormously simplified the process of creation and modification of rings (like changing the monomial module ordering, regrouping of variables, modifying non-commutative relations, working with parameters of the ground field etc.). Especially in the D-module setting we modify rings often, create a new one from existing rings and equip a new ring with a new ordering. Thus, with ring lists the development of such procedures became much easier and the corresponding code became much more manageable.

We have to mention, that in the meantime the implementation of non-commutative multiplication in the kernel of SINGULAR:PLURAL has been improved as well.

3 s-parametric annihilator of f

Recall Malgrange's construction for $f = f_1 \cdots f_p \in \mathbb{K}[x_1, \ldots, x_n]$. Consider the algebra W_{p+n}, being the $(p+n)$-th Weyl algebra

$$\mathbb{K}\langle \{t_j, \partial t_j \mid 1 \le j \le p\} \mid \{[\partial t_j, t_k] = \delta_{jk}\}\rangle \otimes_{\mathbb{K}} \mathbb{K}\langle \{x_i, \partial_i \mid 1 \le i \le n\} \mid \{[\partial_i, x_k] = \delta_{ik}\}\rangle.$$

Moreover, consider the left ideal in W_{p+n}, called Malgrange ideal

$$I_f := \langle \ \{ \ t_j - f_j, \sum_{j=1}^{p} \frac{\partial f_j}{\partial x_i} \partial t_j + \partial_i \ \mid 1 \le j \le p, 1 \le i \le n\} \ \rangle.$$

Then for $s = (s_1, \ldots, s_p)$ we denote $f^s := f_1^{s_1} \cdots f_p^{s_p}$. Let us compute

$$I_f \cap \mathbb{K}[\{t_j \partial t_j\}]\langle x_i, \partial x_i \mid [\partial_i, x_i] = 1\rangle \subset D_n[\{t_j \partial t_j\}]$$

and furthermore, replace $t_j \partial t_j$ with $-s_j - 1$. The result is known (e. g. [34]) to coincide with the s-parametric annihilator of f^s, $\mathrm{Ann}_{D_n[s]}(f^s) = \{Q(x, \partial, s) \in D_n[s] \mid Q \bullet f^s = 0\}$.

There exist several methods for the computation of s-parametric annihilator of f^s.

3.1 Oaku and Takayama

The algorithm by Oaku and Takayama [32, 34] was developed in a wider context and uses homogenization. Consider the $\mathbb{K}$-algebras $T := \mathbb{K}[t_1, \ldots, t_p]$, $D'_p := D(T)$ and $H := D_n \otimes_{\mathbb{K}} D'_p \otimes_{\mathbb{K}} \mathbb{K}[u_1, \ldots, u_p, v_1, \ldots, v_p]$. Moreover, let I below be the (u, v)-homogenized Malgrange ideal, that is the left ideal in H

$$I = \left\langle \{t_j - u_j f_j, \sum_{k=1}^{p} \frac{\partial f_k}{\partial x_i} u_k \partial t_j + \partial_i, u_j v_j - 1\} \right\rangle.$$

Oaku and Takayama proved, that $\mathrm{Ann}_{D_n[s]}(f^s)$ can be obtained in two steps. At first $\{u_j, v_j\}$ are eliminated from I with the help of Gröbner bases, thus yielding $I' = I \cap (D_n \otimes_{\mathbb{K}} D'_p)$. Then, one calculates $I' \cap (D_n \otimes_{\mathbb{K}} \mathbb{K}[\{-t_j \partial t_j - 1\}])$ and substitutes every appearance of $t_j \partial t_j$ by $-s_j - 1$ in the latter. The result is then $\mathrm{Ann}_{D_n[s]}(f^s)$.

3.2 Briançon and Maisonobe

Consider $S_p = \mathbb{K}\langle \{\partial t_j, s_j\} \mid \partial t_j s_k = s_k \partial t_j - \delta_{jk} \partial t_j \rangle$ (the p-th shift algebra) and $S' = D_n \otimes_{\mathbb{K}} S_p$. Moreover, consider the following left ideal in S':

$$I = \left\langle \{s_j + f_j \partial t_j, \sum_{k=1}^{p} \frac{\partial f_k}{\partial x_i} \partial t_k + \partial_i\} \right\rangle.$$

Briançon and Maisonobe proved [5] that $\mathrm{Ann}_{D_n[s]}(f^s) = I \cap D_n[s_1, \ldots, s_p]$ and hence the latter can be computed via the left Gröbner basis with respect to an elimination ordering for $\{\partial t_j\}$.

3.3 Another alternative proof of Briançon-Maisonobe's method

Here we give yet another [1] computer algebraic proof for the method by Briançon and Maisonobe.

Throughout this section, we assume $1 \leq i \leq n$ and $1 \leq j \leq p$. Define

$$E := \mathbb{K}\langle \{t_j, \partial t_j, x_i, \partial_i, s_j\} \mid \{[\partial_i, x_i] = 1, [\partial t_j, t_j] = 1, [t_k, s_j] = \delta_{jk} t_j, [\partial t_k, s_j] = -\delta_{jk} \partial t_j\} \rangle.$$

Let $B = D_n[s]$ be a subalgebra of E, generated by $\{x_i, \partial_i, s_j\}$. Then the Briançon-Maisonobe method requires to prove [1], that

$$\langle \{t_j - f_j, \sum_{j=1}^{p} \frac{\partial f_j}{\partial x_i} \partial t_j + \partial_i, f_j \partial t_j + s_j\} \rangle \cap D_n[s] = \mathrm{Ann}_{D_n[s]}(f^s).$$

Theorem 3.1. *Let us define the following polynomials and sets:*

$$g_i := \partial_i + \sum_{k=1}^{p} \frac{\partial f_k}{\partial x_i} \partial t_k, \; G = \{g_i\}, T = \{t_j - f_j\}, S = \{s_j + f_j \partial t_j\}.$$

Let Λ be a (possibly empty) subset of $\{1, \ldots, p\}$. Define $M_\Lambda := G \cup \{t_k - f_k \mid k \in \Lambda\} \cup \{s_j + f_j \partial t_j \mid j \in \{1, \ldots, p\} \setminus \Lambda\}$.

(a) *For any Λ, the elements of M_Λ commute pairwise. In particular, so do $G \cup T$ and $G \cup S$.*

(b) *Consider an ordering $\prec$, satisfying $\{t_j\} \gg \{x_i\}$, $\{\partial_i, s_j\} \gg \{x_i, \partial t_j\}$. Then any subset of $G \cup T \cup S$ is a left Gröbner basis with respect to $\prec$. In particular, so is the set M_Λ for any Λ.*

(c) *The elements of M_Λ are algebraically independent.*

(d) *For any Λ, the Krull (and hence the Gel'fand-Kirillov) dimension of $\mathbb{K}[M_\Lambda]$ is $n+p$.*

(e) *For any Λ, $\mathbb{K}[M_\Lambda]$ is a maximal commutative subalgebra of E.*

Proof. (a) Computing commutators between elements, we obtain

$$[g_i, g_k] = \partial t_j \sum_j [\partial_i, \frac{\partial f_j}{\partial x_k}] + \partial t_j \sum_j [\frac{\partial f_j}{\partial x_i}, \partial_k] = \partial t_j \sum_j ([\partial_i, \frac{\partial f_j}{\partial x_k}] - [\partial_k, \frac{\partial f_j}{\partial x_i}]) = 0,$$

$$[t_k - f_k, g_i] = \sum_j \frac{\partial f_j}{\partial x_i}[t_k, \partial t_j] - [f_k, \partial_i] = 0, \qquad [t_i - f_i, t_k - f_k] = 0,$$

$$[s_i + f_i \partial t_i, s_j + f_j \partial t_j] = f_j [s_i, \partial t_j] - f_i [s_j, \partial t_i] = 0, \qquad [s_j + f_j \partial t_j, g_i] = [s_j, \partial_i] +$$

$$+ \partial t_j [f_j, \partial_i] + \sum_{k=1}^{p} \frac{\partial f_k}{\partial x_i}[s_j, \partial t_k] + [f_j \partial t_j, \sum_{k=1}^{p} \frac{\partial f_k}{\partial x_i} \partial t_k] = \frac{\partial f_j}{\partial x_i} \partial t_j - [\partial_i, f_j] \partial t_j = 0.$$

The only nonzero commutator arises from

$$[t_k - f_k, s_j + f_j \partial t_j] = [t_k, s_j] + f_j [t_k, \partial t_j] - [f_k, s_j] - [f_k, f_j \partial t_j] = \delta_{jk}(t_k - f_k).$$

However, according to the definition, only one of these elements belongs to M_Λ for any Λ.

(b) We run Buchberger's algorithm by hands. Due to the ordering property, for each pair the generalized Product Criterion is applicable. Hence using (a) we see, that most s-polynomials reduce to commutators, which are zero except $\mathrm{spoly}(t_k - f_k, s_j + f_j \partial t_j) = \delta_{jk}(t_k - f_k)$, which reduces to zero modulo the first polynomial. Thus, any subset including M_Λ is indeed a Gröbner basis.

(c) Using pairwise commutativity, we employ the Commutative Preimage Theorem from [19]. It states, that the ideal of algebraic dependencies between pairwise commuting elements $\{h_k \mid 1 \leq k \leq m\} \subset E$ can be computed as

$$E \otimes_{\mathbb{K}} \mathbb{K}[c_1, \ldots, c_m] \supset \langle \{h_i - c_i\} \rangle \cap \mathbb{K}[c_1, \ldots, c_m],$$

where c_i are new commutative variables, adjoint to E. In this elimination problem one requires an ordering on $E \otimes_{\mathbb{K}} \mathbb{K}[c]$, preferring variables of E to c_i's. For such an ordering, one needs to compute a Gröbner basis. Now, take $\{h_i\} := M_\Lambda$, $1 \leq i \leq p+n$, and run Buchberger's algorithm with respect to the same ordering as in (b). Thus we are again in the situation, where the Product Criterion applies, hence $[h_i - c_i, h_k - c_k] = 0$ since $[h_i, h_k] = 0$ by (b) and c_i are central. Hence, $\{h_i - c_i\}$ is a left Gröbner basis and by the elimination property $\langle\{h_i - c_i\}\rangle \cap \mathbb{K}[c_1, \dots, c_m] = 0$, that is $\{h_i\}$ are algebraically independent.

(d) By (c), M_Λ generates a commutative ring with no algebraic dependence between its elements, so the Krull dimension is the cardinality of M_Λ, that is $n+p$. Since M_Λ is isomorphic to a commutative polynomial ring by (c), its Gel'fand-Kirillov dimension over the field $\mathbb{K}$ is $n+p$ as well.

(e) With respect to the ordering from (b), the leading monomials of the generators are $\{\partial_1, \dots, \partial_n\} \cup \{t_k \mid k \in \Lambda\} \cup \{s_j \mid j \notin \Lambda\}$. Assume, that there exists an element in $E \setminus \mathbb{K}[M_\Lambda]$, which commutes with all elements in M_Λ. Then its leading monomial must belong to the subalgebra F, generated by $\{x_1, \dots, x_n\} \cup \{s_k \mid k \in \Lambda\} \cup \{t_j \mid j \notin \Lambda\} \cup \{\partial t_1, \dots, \partial t_p\}$. Since the center of E is $\mathbb{K}$, we consider centralizers of elements. Taking $F' = \cap_{k\in\Lambda} C(t_k - f_k) \cap F$, we see that an element from it can have no $\{\partial t_k, s_k \mid k \in \Lambda\}$. Considering $F'' = \cap_i C(g_i) \cap F'$, we exclude $\{x_1, \dots, x_n\}$. Thus we are left with the subalgebra, generated by $\tilde{F} = \{\partial t_j, s_j \mid j \notin \Lambda\}$. But no element of it can commute with $\{s_j + f_j \partial t_j \mid j \notin \Lambda\}$ except constants. Hence the claim. □

We want to eliminate both $\{t_j\}$ and $\{\partial t_j\}$ from an ideal, generated by $G \cup S \cup T$. By using an elimination ordering for $\{t_j\}$ we proved in (b) above, that $G \cup S \cup T$ is a Gröbner basis. Hence, the elimination ideal is generated by $G \cup S$ and we can proceed with eliminating $\{\partial t_j\}$ from $G \cup S$, which is exactly the statement of Briançon-Maisonobe in Section 3.2.

3.4 Implementation

We use the following acronyms in addressing functions in the implementation: *OT* for Oaku and Takayama, *LOT* for Levandovskyy's modification of Oaku and Takayama [20] and *BM* for Briançon-Maisonobe. Moreover, it is possible to specify the desired Gröbner basis engine (`std` or `slimgb`) via an optional argument.

For the classical situation $f = f_1$, the procedure `Sannfs(`f`)` computing $\mathrm{Ann}_{D_n[s]}(f^s) \subset D_n[s]$ uses a "minimal user knowledge" principle and chooses one of three mentioned algorithms. Alternatively, one can call the corresponding procedures `SannfsOT, SannfsLOT, SannfsBM` directly.

For the annihilator of $f = f_1 \cdots f_p$, see Section 9.

Example 3.2. We demonstrate, how to compute the s-parametric annihilator with `Sannfs`. This procedure takes a polynomial in a commutative ring as its argument and returns back a Weyl algebra of the type `ring` containing an object of the type `ideal` called `LD`. Note, that the latter ideal is a set of generators and not a Gröbner basis in general.

```
LIB "dmod.lib";
ring r = 0,(x,y),dp;          // set up the commutative ring
poly f = x^3 + y^2 + x*y^2;   // define the polynomial
def D = Sannfs(f); setring D; // call Sannfs and change to ring D
LD = groebner(LD); LD;        // compute and print Groebner basis
==> LD[1]=2*x*y*Dx-3*x^2*Dy-y^2*Dy+2*y*Dx
==> LD[2]=2*x^2*Dx+2*x*y*Dy+2*x*Dx+3*y*Dy-6*x*s-6*s
```

```
==> LD[3]=x^2*y*Dy+y^3*Dy-2*x^2*Dx-3*x*y*Dy-2*y^2*s+6*x*s
==> LD[4]=x^3*Dy+x*y^2*Dy+y^2*Dy-2*x*y*s-2*y*s
==> LD[5]=2*y^3*Dx*Dy+3*x^3*Dy^2+x*y^2*Dy^2-4*x^2*Dx^2-8*x*y*Dx*Dy-2*x^2*Dx
 -4*y^2*Dx*s+6*x*y*Dy+12*x*Dx*s-10*x*Dx-6*y*Dy+12*s
```

4 Annihilators of polynomial and rational functions

4.1 Annihilator of f^α for $\alpha \in \mathbb{C}$

It is known (e. g. [34]) that for any $\alpha \in \mathbb{C}$, $D_n/\operatorname{Ann}_{D_n}(f^\alpha)$ is a holonomic D-module. In the procedure `annfspecial` from `dmod.lib` we follow Algorithm 5.3.15 in [34]. Given f and α, we compute $\operatorname{Ann}_{D_n[s]}(f^s) \subset D_n[s]$, the Bernstein-Sato polynomial of f (cf. Section 6.1) and its minimal integer root s_0. Then, if $\alpha - (s_0+1) \in \mathbb{N}$, according to Algorithm 5.3.15 in [34] we have to compute a certain syzygy module in advance. Otherwise, $\operatorname{Ann}_{D_n}(f^\alpha) = \operatorname{Ann}_{D_n[s]}(f^s)\mid_{s=\alpha}$ is obtained via substitution.

Example 4.1. In this example we show, how one computes the annihilator of $2xy$.

```
LIB "dmod.lib"; option(redSB); option(redTail);
ring r = 0,(x,y),dp; poly g = 2*x*y;
def A = Sannfs(g); setring A;       // compute Ann(g^s)
LD = groebner(LD); LD;              // GB of the ideal Ann(g^s)
==> LD[1]=y*Dy-s
==> LD[2]=x*Dx-s
def B = annfs0(LD,2*x*y); setring B; // compute BS polynomial
BS; // the list of roots and multiplicities of BS polynomial
==> [1]:
==>    _[1]=-1
==> [2]:
==>    2
// so, the minimal integer root is -1
setring A;                          // need to work with Ann(g^s) again
ideal I = annfspecial(LD,2*x*y,-1,1);
                       // the last argument 1 indicates that we want to compute f^1
print(matrix(I));                   // condensed presentation
==> Dy^2, y*Dy-1, Dx^2, x*Dx-1
```

4.2 Alternative for an annihilator of f^m

Computing a syzygy module in the previous algorithm can be expensive. Therefore we note, that for $\alpha = m \in \mathbb{N}$ we better use an easier approach.

Lemma 4.2. *Let $g \in \mathbb{K}[x_1,\ldots,x_n]$. Consider the homomorphism of left D_n-modules $\psi : D_n \to D_n/\langle\partial_1,\ldots,\partial_n\rangle$, $\psi(1) = g$. Then $\operatorname{Ann}_{D_n}(g) = \ker\psi$.*

Proof. Note, that $\mathbb{K}[x_1,\ldots,x_n] \cong D_n/\langle\partial_1,\ldots,\partial_n\rangle$ as left D_n-modules. Hence we can view g as the image of 1 under ψ. Then $\operatorname{Ann}_{D_n}(g) = \{a \in D_n \mid a \bullet g = 0\} = \{a \in D_n \mid ag \in \langle\partial_1,\ldots,\partial_n\rangle\} = \ker\psi$. $\square$

Remark 4.3. Hence, given any element $f \in \mathbb{K}[x_1, \ldots, x_n]$, $\operatorname{Ann}_{D_n}(f)$ can be computed via the kernel of a module homomorphism (algorithm `Modulo`) which amounts to just one Gröbner basis computation. Moreover, it does not use elimination and hence is clearly more efficient in the special case $g = f^n$ for $f \in R, n \in \mathbb{N}$, than the more general method in Section 4.1. Notably this method can be generalized to various other operator algebras, see [35] for details. The corresponding procedure in `dmodapp.lib` is called `annPoly`.

Remark 4.4. Yet another improvement can be achieved in the computation of the minimal integer root of the Bernstein-Sato polynomial with the algorithms from Theorem 6.6 below. Namely, since we know, that for an integer root, say α, of the Bernstein-Sato polynomial of a polynomial in $n \geq 2$ variables $-n+1 \leq \alpha \leq -1$ holds (by [33, 41]) and -1 is always a root, we can run the `checkRoot` procedure (which is just one Gröbner basis computation with an arbitrary ordering, see Section 6.4) starting from $\alpha = -n+1$ to $\alpha = -2$. We stop at the first affirmative answer from `checkRoot` or output -1 if no positive answer appears. Thus, one executes `checkRoot` at most $n-2$ times.

Algorithm 4.5 (Heuristic for $\operatorname{Ann}_{D_n}(f^\alpha)$).
Input: $f \in \mathbb{C}[x_1, \ldots, x_n]$, $\alpha \in \mathbb{C}$
Output: $\operatorname{Ann}_{D_n}(f^\alpha)$
if $\alpha \in \mathbb{C} \setminus (\mathbb{Z} \cap [-n+1, -1])$ **then**

$$\operatorname{Ann}_{D_n}(f^\alpha) = \begin{cases} \langle \partial_1, \ldots, \partial_n \rangle & \text{if } \alpha = 0, \\ \ker(D_n \stackrel{1 \mapsto f^m}{\longrightarrow} D_n / \langle \partial_1, \ldots, \partial_n \rangle) & \text{if } \alpha = m \in \mathbb{N}, \quad \text{(cf. 4.2)}, \\ \operatorname{Ann}_{D_n[s]}(f^s) \mid_{s=\alpha} & \text{if } \alpha \in (\mathbb{C} \setminus \mathbb{Z}) \cup (\mathbb{Z} \cap (-\infty, -n]), \end{cases}$$

else (that is $\alpha \in \mathbb{Z} \cap [-n+1, -1]$)

$\mu := \min\{\beta \in \mathbb{Z}_{<0} \mid b_f(\beta) = 0\}$

$$\operatorname{Ann}_{D_n}(f^\alpha) = \begin{cases} \text{Procedure 4.1 with 4.4} & \text{if } \mu + 1 \leq \alpha \leq -1, \\ \text{Procedure 4.4 and } \operatorname{Ann}_{D_n[s]}(f^s) \mid_{s=\alpha} & \text{if } -n+1 \leq \alpha \leq \mu. \end{cases}$$

end if
return $\operatorname{Ann}_{D_n}(f^\alpha)$

4.3 Annihilator of a rational function

In order to compute the annihilator I of a rational function $\frac{f}{g}$ (it is known that D_n/I is holonomic) we use the following lemma.

Lemma 4.6. *Let $f, g \in \mathbb{K}[x_1, \ldots, x_n] \setminus \{0\}$. Consider the homomorphism of left D_n-modules $\tau : D_n \to D_n / \operatorname{Ann}_{D_n}(g^{-1}), q \mapsto qf$. Then $\operatorname{Ann}_{D_n}(\frac{f}{g}) = \ker \tau$.*

Proof. For $q \in \ker \tau = \{q \in D_n \mid qf \in \operatorname{Ann}_{D_n}(g^{-1})\}$, $(qf) \bullet g^{-1} = q \bullet (fg^{-1})$, hence $\operatorname{Ann}_{D_n}(\frac{f}{g}) = \ker \tau$. □

We compute $\operatorname{Ann}_{D_n}(g^{-1})$ with Algorithm 4.5 above. Although in the case, when -1 is not the minimal integer root of the Bernstein-Sato polynomial of g, we have to use expensive algorithms like 4.1, we know no other methods to compute the annihilator in Weyl algebras. Also, no general algorithm for computing a complete system of operator equations (with operators including along partial differentiation also partial (q-)differences et cetera) with polynomial coefficients, annihilating a rational function, is known to us. In our opinion, the existence of an algorithm for $\mathrm{Ann}_{D_n}(g^{-1})$ shows the intrinsic naturality of D-modules compared with other linear operators acting on $\mathbb{K}[x]$. The algorithm is implemented in `dmodapp.lib` and the corresponding procedure is called `annRat`.

Example 4.7. In this example we demonstrate the computation of annihilators of a rational function. The procedure `annRat` takes as arguments polynomials in a commutative ring and returns a Weyl algebra (of type `ring`) together with an object of type `ideal` called `LD` (cf. Example 3.2). Note, that `LD` is given in a Gröbner basis.

```
LIB "dmodapp.lib";
ring r = 0,(x,y),dp;
poly g = 2*x*y;  poly f = x^2 - y^3; // we will compute Ann(g/f)
option(redSB); option(redTail);      // get reduced minimal GB
def B = annRat(g,f); setring B;
LD;                                  // Groebner basis of Ann(g/f)
==> LD[1]=3*x*Dx+2*y*Dy+1
==> LD[2]=y^3*Dy^2-x^2*Dy^2+6*y^2*Dy+6*y
==> LD[3]=9*y^2*Dx^2*Dy-4*y*Dy^3+27*y*Dx^2+2*Dy^2
==> LD[4]=y^4*Dy-x^2*y*Dy+2*y^3+x^2
==> LD[5]=9*y^3*Dx^2-4*y^2*Dy^2+10*y*Dy-10
```

5 b-function with respect to weights for an ideal

Let $0 \neq w \in \mathbb{R}^n_{\geq 0}$ and consider the V-filtration $V = \{V_m \mid m \in \mathbb{Z}\}$ on D_n with respect to w, where V_m is spanned by $\left\{x^\alpha \partial^\beta \mid -w\alpha + w\beta \leq m\right\}$ over $\mathbb{K}$. That is, x_i and ∂_i get weights $-w_i$ and w_i respectively. Note that then the relation $\partial_i x_i = x_i \partial_i + 1$ is homogeneous of degree 0. It is known that the associated graded ring $\bigoplus_{m\in\mathbb{Z}} V_m/V_{m-1}$ is isomorphic to D_n, which allows us to identify it with the Weyl algebra.

From now on we assume, that $I \subset D_n$ is an ideal such that D_n/I is a holonomic module. Since holonomic D-modules are cyclic (e. g. [10]), for each holonomic D-module M there exists an ideal I_M such that $M \cong D_n/I_M$ as D-modules.

Definition 5.1. Let $0 \neq w \in \mathbb{R}^n_{\geq 0}$. For a non-zero polynomial

$$p = \sum_{\alpha,\beta\in\mathbb{N}_0^n} c_{\alpha\beta} x^\alpha \partial^\beta \in D_n \quad \text{with all but finitely many } c_{\alpha\beta} = 0$$

we put $m = \max_{\alpha,\beta}\{-w\alpha + w\beta \mid c_{\alpha\beta} \neq 0\} \in \mathbb{R}$ and define the *initial form* of p with respect to the weight w as follows:

$$\mathrm{in}_{(-w,w)}(p) := \sum_{\alpha,\beta\in\mathbb{N}_0^n:\ -w\alpha+w\beta=m} c_{\alpha\beta} x^\alpha \partial^\beta.$$

For the zero polynomial, we set $\mathrm{in}_{(-w,w)}(0) := 0$. Additionally, the ideal $\mathrm{in}_{(-w,w)}(I) := \mathbb{K} \cdot \{\mathrm{in}_{(-w,w)}(p) \mid p \in I\}$ is called the *initial ideal* of I with respect to w.

Definition 5.2. Let $0 \neq w \in \mathbb{R}^n_{\geq 0}$ and $s := \sum_{i=1}^n w_i x_i \partial_i$. Then $\text{in}_{(-w,w)}(I) \cap \mathbb{K}[s]$ is a principal ideal in $\mathbb{K}[s]$. Its monic generator $b_{I,w}(s)$ is called the *global b-function* of I with respect to the weight w.

Theorem 5.3. *The global b-function is nonzero.*

We will give a proof of this well-known result in Section 5.2.

Following its definition, the computation of the global b-function of I with respect to w can be done in two steps:

1. Compute the initial ideal I' of I with respect to w.

2. Compute the intersection of I' with the subalgebra $\mathbb{K}[s]$.

We will discuss both steps separately, starting with the initial ideal. It is important to mention, that although this procedure has been described in [34], this approach was completely treated by Noro in [30], accompanied with a very impressive implementation in `Risa/Asir`.

5.1 Computing the initial ideal

In order to compute the initial ideal, the method of weighted homogenization is proposed in [30], which we will describe below.

Let $u, v \in \mathbb{R}^n_{>0}$. The G-algebra $D^{(h)}_{(u,v)} := \mathbb{K}\langle x_1, \ldots, x_n, \partial_1, \ldots, \partial_n, h \mid \{x_j x_i = x_i x_j, \partial_j \partial_i = \partial_i \partial_j, x_i h = h x_i, \partial_i h = h \partial_i, \partial_j x_i = x_i \partial_j + \delta_{i,j} h^{u_i + v_j}\}\rangle$ is called the *n-th weighted homogenized Weyl algebra* with weights u, v, i. e. x_i and ∂_i get weights u_i and v_i respectively.

For $p = \sum_{\alpha,\beta} c_{\alpha\beta} x^\alpha \partial^\beta \in D_n$ we define the *weighted homogenization* of p as follows:

$$H_{(u,v)}(p) = \sum_{\alpha,\beta} c_{\alpha\beta} h^{\deg_{(u,v)}(p) - (u\alpha + v\beta)} x^\alpha \partial^\beta.$$

This definition naturally extends to a set of polynomials. Here, $\deg_{(u,v)}(p)$ denotes the weighted total degree of p with respect to weights u, v for x, ∂ and weight 1 for h.

For a monomial ordering $\prec$ on D_n, which is not necessarily a well-ordering, we define an associated homogenized global ordering $\prec^{(h)}$ in $D^{(h)}_{(u,v)}$ by setting $h \prec^{(h)} x_i, h \prec^{(h)} \partial_i$ for all i and,

$$\begin{aligned} p \prec^{(h)} q \quad \text{if} \quad & \deg_{(u,v)}(p) < \deg_{(u,v)}(q) \\ \text{or} \quad & \deg_{(u,v)}(p) = \deg_{(u,v)}(q) \quad \text{and} \quad p_{|h=1} \prec q_{|h=1}. \end{aligned}$$

Note that for $u = v = (1, \ldots, 1)$ this is exactly the standard homogenization as in [34] and [9]. Analogue statements of the following two theorems can be found in [34] and [30] respectively.

Theorem 5.4. *Let F be a finite subset of D_n and $\prec$ a global ordering. If $G^{(h)}$ is a Gröbner basis of $\langle H_{(u,v)}(F)\rangle$ with respect to $\prec^{(h)}$, then ${G^{(h)}}_{|h=1}$ is a Gröbner basis of $\langle F\rangle$ with respect to $\prec$.*

Theorem 5.5. *Let $\prec$ be a global monomial ordering on D_n and $\prec_{(-w,w)}$ the non-global ordering defined by*

$$\begin{aligned} x^\alpha \partial^\beta \prec_{(-w,w)} x^\gamma \partial^\delta \quad &\text{if} \quad -w\alpha + w\beta < -w\gamma + w\delta \\ &\text{or} \quad -w\alpha + w\beta = -w\gamma + w\delta \quad \text{and} \quad x^\alpha \partial^\beta \prec x^\gamma \partial^\delta. \end{aligned}$$

If $G^{(h)}$ is a Gröbner basis of $H_{(u,v)}(I)$ with respect to $\prec^{(h)}_{(-w,w)}$, then the set $\{\mathrm{in}_{(-w,w,0)}(g) \mid g \in G^{(h)}\}$ is a Gröbner basis of $\mathrm{in}_{(-w,w,0)}(H_{(u,v)}(I))$ with respect to $\prec^{(h)}$.

Proof. Let $f' \in \mathrm{in}_{(-w,w,0)}(H_{(u,v)}(I))$ be $(-w,w,0)$-homogeneous. There exist $f \in H_{(u,v)}(I)$ and $g \in G^{(h)}$ such that $f' = \mathrm{in}_{(-w,w,0)}(f)$ and $\mathrm{lm}_{\prec^{(h)}_{(-w,w)}}(g) \mid \mathrm{lm}_{\prec^{(h)}_{(-w,w)}}(f)$. Since f, g are (u,v)-homogeneous, we have

$$\mathrm{lm}_{\prec^{(h)}_{(-w,w)}}(g) = \mathrm{lm}_{\prec^{(h)}}(\mathrm{in}_{(-w,w,0)}(g)), \qquad \mathrm{lm}_{\prec^{(h)}_{(-w,w)}}(f) = \mathrm{lm}_{\prec^{(h)}}(\mathrm{in}_{(-w,w,0)}(f)),$$

which concludes the proof. $\square$

Summarizing the results from this section, we obtain the following algorithm to compute the initial ideal.

Algorithm 5.6 (`InitialIdeal`). **Input:** $I \subset D_n$ a holonomic ideal, $0 \neq w \in \mathbb{R}^n_{\geq 0}$, $\prec$ a global ordering on D_n, $u, v \in \mathbb{R}^n_{>0}$
Output: A Gröbner basis G of $\mathrm{in}_{(-w,w)}(I)$ with respect to $\prec$
$\prec^{(h)}_{(-w,w)}$:= the homogenized ordering as defined in theorem 5.5
$G^{(h)}$:= a Gröbner basis of $H_{(u,v)}(I)$ with respect to $\prec^{(h)}_{(-w,w)}$
return $G = \mathrm{in}_{(-w,w)}(G^{(h)}|_{h=1})$

5.2 Intersecting an ideal with a principal subalgebra

We will now consider a much more general setting than needed to compute the global b-function. Let A be an associative $\mathbb{K}$-algebra. We are interested in computing the intersection of a left ideal $J \subset A$ with the subalgebra $\mathbb{K}[s]$ of A where $s \in A$ is an arbitrary non-constant element. This intersection is always generated by one element since $\mathbb{K}[s]$ is a principal ideal domain. In other words, we want to find the monic generator $b \in A$ such that $\langle b \rangle = J \cap \mathbb{K}[s]$.

For this section, we will assume that there is an ordering on A such that there exists a finite left Gröbner basis G of J.

Then we can distinguish between the following four situations:

1. No leading monomials of elements in G divide the leading monomial of any power of s.

2. There is an element in G whose leading monomial divides the leading monomial of some power of s. In this situation, we have the following sub-situations.

2.1. $J \cdot s \subset J$ and $\dim_{\mathbb{K}}(\operatorname{End}_A(A/J)) < \infty$.

2.2. One of the two conditions in 2.1. does not hold.

2.2.1. The intersection is zero.

2.2.2. The intersection is not zero.

Lemma 5.7. *If there exists no $g \in G$ such that $\operatorname{lm}(g)$ divides $\operatorname{lm}(s^k)$ for some $k \in \mathbb{N}_0$, then $J \cap \mathbb{K}[s] = \{0\}$.*

The lemma covers the first case above. In the second case however, we cannot in general state whether the intersection is trivial or not as the following example illustrates.

Remark 5.8. The converse of the previous lemma is wrong. For instance, consider $\mathbb{K}[x, y]$ and $J = \langle y^2 + x\rangle$. Then $J \cap \mathbb{K}[y] = \{0\}$ while $\{y^2 + x\}$ is a Gröbner basis of J for any ordering.

In situation 2.1. though, the intersection is not zero as the following lemma shows, inspired by the sketch of the proof of Theorem 5.3 in [34].

Lemma 5.9. *Let $J \cdot s \subset J$ and $\dim_{\mathbb{K}}(\operatorname{End}_A(A/J)) < \infty$. Then $J \cap \mathbb{K}[s] \neq \{0\}$.*

Proof. Consider the right multiplication with s as a map $A/J \to A/J$ which is a well-defined A-module endomorphism of A/J as $a - a' \in J$ implies that $(a - a')s \in J \cdot s \subset J$, which holds by assumption for all $a, a' \in A$. Since $\operatorname{End}_A(A/J)$ is finite dimensional, linear algebra guarantees that this endomorphism has a well-defined non-zero minimal polynomial μ. Moreover, μ is precisely the monic generator of $J \cap \mathbb{K}[s]$ as $\mu(s) = [0]$ in A/J, hence $\mu(s) \in J \cap \mathbb{K}[s]$, and $\deg(\mu)$ is minimal by definition. □

Remark 5.10. In particular, the lemma holds if A/J itself is a finite dimensional A-module. In the case where A is a Weyl algebra and A/J is a holonomic module, we know that $\dim_{\mathbb{K}}(\operatorname{End}_A(A/J))$ is finite (cf. [34]).

For situation 2.1., we have reduced our problem of intersecting an ideal with a subalgebra generated by one element to a problem from linear algebra by the proof of the lemma, namely to the one of finding the minimal polynomial of an endomorphism.

Proof of Theorem 5.3. Let $0 \neq w \in \mathbb{R}^n_{\geq 0}$, $J := \operatorname{in}_{(-w,w)}(I)$ for a holonomic ideal $I \subset D_n$ and $s := \sum_{i=1}^n w_i x_i \partial_i$. Without loss of generality let $0 \neq p = \sum_{\alpha,\beta} c_{\alpha,\beta} x^\alpha \partial^\beta \in J$ be $(-w, w)$-homogeneous. Then we obtain for every monomial in p by using the Leibniz rule

$$\begin{aligned} x^\alpha \partial^\beta x_i \partial_i &= x^{\alpha+e_i} \partial^{\beta+e_i} + \beta_i x^\alpha \partial^\beta = (\partial_i x_i^{\alpha_i+1} - (\alpha_i + 1) x_i^{\alpha_i}) \frac{x^\alpha}{x_i^{\alpha_i}} \partial^\beta + \beta_i x^\alpha \partial^\beta \\ &= (\partial_i x_i - (\alpha_i + 1) + \beta_i) x^\alpha \partial^\beta = (x_i \partial_i - \alpha_i + \beta_i) x^\alpha \partial^\beta. \end{aligned}$$

Put $m = -w\alpha + w\beta$ for some term $c_{\alpha,\beta} x^\alpha \partial^\beta$ in p where $c_{\alpha,\beta}$ is non-zero. Since p is $(-w, w)$-homogeneous, m does not depend on the choice of this term. Hence,

$$\begin{aligned} p \cdot s = p \sum_{i=1}^n w_i x_i \partial_i &= \sum_{i=1}^n w_i \sum_{\alpha,\beta} (x_i \partial_i - \alpha_i + \beta_i) c_{\alpha,\beta} x^\alpha \partial^\beta \\ &= s \cdot p + \sum_{i=1}^n \sum_{\alpha,\beta} w_i (-\alpha_i + \beta_i) c_{\alpha,\beta} x^\alpha \partial^\beta = (s + m) \cdot p \in J. \end{aligned}$$

Since D_n/J is holonomic (cf. [34]) and $J \cdot s \subset J$, Remark 5.10 and Lemma 5.9 yield the claim. $\square$

If one knows in advance that the intersection is not zero, the following algorithm terminates.

Algorithm 5.11 (`principalIntersect`).
Input: $s \in A, J \subset A$ a left ideal such that $J \cap \mathbb{K}[s] \neq \{0\}$.
Output: $b \in \mathbb{K}[s]$ monic such that $J \cap \mathbb{K}[s] = \langle b \rangle$
 $G :=$ a finite left Gröbner basis of J (assume it exists)
 $i := 1$
 loop
  if there exist $a_0, \ldots, a_{i-1} \in \mathbb{K}$ such that $\mathrm{NF}(s^i, G) + \sum_{j=0}^{i-1} a_j \, \mathrm{NF}(s^j, G) = 0$ **then**
   return $b := s^i + \sum_{j=0}^{i-1} a_j s^j$
  else
   $i := i + 1$
  end if
 end loop

Note that because $\mathrm{NF}(s^i, G) + \sum_{j=0}^{i-1} a_j \, \mathrm{NF}(s^j, G) = 0$ is equivalent to $s^i + \sum_{j=0}^{i-1} a_j s^j \in J$, the algorithm searches for a monic polynomial in $\mathbb{K}[s]$ that also lies in J. This is done by going degree by degree through the powers of s until there is a linear dependency. This approach also ensures the minimality of the degree of the output. The algorithm terminates if and only if $J \cap \mathbb{K}[s] \neq \{0\}$. Note that this approach works over any field.

The check whether there is a linear dependency over $\mathbb{K}$ between the computed normal forms of the powers of s is done by the procedure `linReduce` in our implementation.

5.2.1 An enhanced computation of normal forms

When computing normal forms of the form $\mathrm{NF}(s^i, J)$ like in algorithm 5.11 we can speed up the reduction process by making use of the previously computed normal forms.

Lemma 5.12. *Let A be a $\mathbb{K}$-algebra, $J \subset A$ a left ideal and let $f \in A$. For $i \in \mathbb{N}$ put $r_i = \mathrm{NF}(f^i, J)$, $q_i = f^i - r_i \in J$ and $c_i = \frac{\mathrm{lc}(q_i r_1)}{\mathrm{lc}(r_1 q_i)}$ provided $r_1 q_i \neq 0$. For $r_1 q_i = 0$ we put $c_i = 0$. Then we have for all $i \in \mathbb{N}$*

$$r_{i+1} = \mathrm{NF}(f r_i, J) = \mathrm{NF}([f^i - r_i, r_1]_{c_i} + r_i r_1, J).$$

As a consequence, we obtain the following result for some $\mathbb{K}$-algebras of special importance.

Corollary 5.13. *If A is a G-algebra of Lie type (e. g. a Weyl algebra), then*

$$r_{i+1} = \mathrm{NF}(f r_i, J) = \mathrm{NF}([f^i - r_i, r_1] + r_i r_1, J) \textit{ holds.}$$

If A is commutative, we have $r_{i+1} = \mathrm{NF}(r_i r_1, J) = \mathrm{NF}(r_1, J)^{i+1} = \mathrm{NF}(r_1^{i+1}, J)$.

Note, that computing Lie bracket $[f, g]$ both in theory and in practice is easier and faster, than to compute $[f, g]$ as $f \cdot g - g \cdot f$, see e. g. [22].

5.2.2 Applications

Apart from computing global b-functions, there are various other applications of Algorithm 5.11.

Solving Zero-dimensional Systems. Recall that an ideal $I \subset \mathbb{K}[x_1, \ldots, x_n]$ is called zero-dimensional if one of the following equivalent conditions holds:

- $\mathbb{K}[x_1, \ldots, x_n]/I$ is finite dimensional as a $\mathbb{K}$-vector space.
- For each $1 \leq i \leq n$ there exist $0 \neq f_i \in I \cap \mathbb{K}[x_i]$.
- The cardinality of the zero-set of I is finite.

In order to compute the zero-set of I, one can use the classical triangularization algorithms. These algorithms require to compute a Gröbner basis with respect to some elimination ordering (like lexicographic one), which might be very hard.

By Algorithm 5.11, a generator of $I \cap \mathbb{K}[x_i]$ can be computed without these expensive orderings. Instead, any ordering, hence a better suited one, may be freely chosen.

A similar approach is used in the celebrated FGLM algorithm (cf. [12]).

Computing Central Characters and Algebraic Dependence. Let A be an associative $\mathbb{K}$-algebra. Intersection of a left ideal with the center of A, which is isomorphic to a commutative ring, is important for many algorithms, among other for the computation of central character decomposition of a finitely presented module (cf. [19] for the theory and [1] for an example with Principal Intersection). In the situation, where the center of A is generated by one element (which is not seldom), we can apply Algorithm 5.11 to compute the intersection (known to be often quite nontrivial) without engaging much more expensive Gröbner basis computation, which use elimination.

Example 5.14. Consider the quantum algebra $U'_q(\mathfrak{so}_3)$ (as defined by Fairlie and Odesskii) for q^2 being the n-th root of unity. It is known, that then, in addition to the single generator C of the center present over any field, three new elements Z_i, depending on n will appear. Since $U'_q(\mathfrak{so}_3)$ has Gel'fand-Kirillov dimension 3, four commuting elements in it obey a single polynomial algebraic dependency (the ideal of dependencies in principal). Computing such a dependency is a very tough challenge for Gröbner bases. But as we see, it is quite natural to apply Principal Intersection.

```
LIB "ncalg.lib"; LIB "bfun.lib";
def A = makeQso3(5);             // below Q^2 is the 5th root of unity
setring A;
// central elements, depending on Q in their classical form:
ideal I = x5+(Q3-Q2+2)*x3+(Q3-Q2+1)*x, y5+(Q3-Q2+2)*y3+(Q3-Q2+1)*y,
 z5+(Q3-Q2+2)*z3+(Q3-Q2+1)*z;
I = twostd(I);                   // two-sided Groebner basis
poly C = 5*xyz+(4Q3-3Q2+2Q-1)*x2+(-Q3+2Q2-3Q+4)*y2+(4Q3-3Q2+2Q-1)*z2;
poly v = vec2poly(pIntersect(C,I),1); // present vector as poly
poly t = subst(v,x,C); t;        // t as a polynomial in C of size 42
==> 3125*x5y5z5+(3125Q3-3125Q2+6250)*x5y5z3+(3125Q3-3125Q2+6250)*x5y3z5+ ...
```

```
// present matrix of cofactors of t as an element of I:
matrix T = lift(I,t);
poly a = 125*(25*I[1]*I[2]*I[3]+(Q3-7Q2+8Q-4)*(I[1]^2+I[2]^2+I[3]^2));
a-t;                            // a expresses t in the subalgebra gen. by I[1..3]
==> 0
// define univariate ring over algebraic extension:
ring r = (0,Q),c,dp; minpoly = Q4-Q3+Q2-Q+1;
poly v = fetch(A,v);            // map v from A to a univariate poly in c
factorize(v);
```

The latter factorization delivers the final touch to the answer: the algebraic dependency is described by the equation $C^2 \cdot (C + 4q^3 - 3q^2 - 3q + 4) \cdot (C + 3q^3 - q^2 - q + 3)^2 = 3125 \cdot Z_1 Z_2 Z_3 + 125 \cdot (q^3 - 7q^2 + 8q - 4) \cdot (Z_1^2 + Z_2^2 + Z_3^2)$.

6 Bernstein-Sato polynomial of f

6.1 Global Bernstein-Sato polynomial

One possibility to define the Bernstein-Sato polynomial of a polynomial $f \in \mathbb{K}[x_1, \ldots, x_n]$ is to apply the global b-function for specific weights.

Definition 6.1. Let $b_{I_f,w}(s)$ denote the global b-function of the univariate Malgrange ideal I_f of f (cf. Section 3) with respect to the weight vector $w = (1, 0, \ldots, 0) \in \mathbb{R}^{n+1}$, that is the weight of ∂_t is 1. Then $b_f(s) = (-1)^{\deg(b_{I_f,w})} b_{I_f,w}(-s-1)$ is called the *global b-function* (*Bernstein-Sato polynomial*) of f.

By Theorem 5.3, $b_f(s) \neq 0$ holds. Moreover, it is known that all roots of $b_f(s)$ are negative rational numbers. Kashiwara proved this result for local Bernstein-Sato polynomials over $\mathbb{C}$ [18]. This fact together with Theorem 6.3 below and classical flatness properties imply the claim for the global case over an arbitrary field of characteristic 0.

The following theorem gives us another option to define the Bernstein-Sato polynomial.

Theorem 6.2 ([4], see also [34, Lemma 5.3.11])**.** *The Bernstein-Sato polynomial $b_f(s)$ of f is the unique monic polynomial of minimal degree in $\mathbb{K}[s]$ satisfying the identity*

$$P \bullet f^{s+1} = b_f(s) \cdot f^s \qquad \textit{for some operator } P \in D_n[s].$$

Since $P \cdot f - b_f(s) \in \mathrm{Ann}_{D_n[s]}(f^s)$ holds, $b_f(s)$ is the monic polynomial satisfying

$$\langle b_f(s) \rangle = \mathrm{Ann}_{D_n[s]}(f^s) + \langle f \rangle \cap \mathbb{K}[s]. \tag{1}$$

Summarizing, there are several choices for computing the Bernstein-Sato polynomial:

1. Compute either
   (a) $J = \mathrm{in}_{(-w,w)}(I_f)$ or
   (b) $J = \mathrm{Ann}_{D_n[s]}(f^s) + \langle f \rangle$.
2. Intersect J with $\mathbb{K}[s]$ by
   (a) the classical elimination-driven approach or
   (b) using Algorithm 5.11.

It is very interesting to investigate the approach for the computation of Bernstein-Sato polynomial, which arises as the combination of the two methods:

1. $\mathrm{Ann}_{D_n[s]}(f^s)$ via Briançon-Maisonobe (cf. [20]),
2. $(\mathrm{Ann}(_{D_n[s]}f^s) + \langle f \rangle) \cap \mathbb{K}[s]$ via Algorithm 5.11.

For an efficient computation of $\mathrm{in}_{(-w,w)}(I_f)$ using the method of weighted homogenization as described in Section 5.1, Noro proposes [30] to choose the weights $\hat{u} = (\deg_u(f), u_1, \ldots, u_n)$, $\hat{v} = (1, \deg_u(f) - u_1 + 1, \ldots, \deg_u(f) - u_n + 1)$, such that the weight of t is $\deg_u(f)$ and the weight of ∂_t is 1. Here, $u \in \mathbb{R}^n_{>0}$ is an arbitrary vector and $\deg_u(f)$ denotes the weighted total degree of f with respect to u. The vector u may be chosen heuristically in accordance to the shape of f or by default, one can set $u = (1, \ldots, 1)$.

6.2 Implementation

For the computation of Bernstein-Sato polynomials, we offer the following procedures in the Singular library `bfun.lib`:

`bfct` computes $\mathrm{in}_{(-w,w)}(I_f)$ using weighted homogenization with weights $\hat{u}, \hat{v}$ for an optional weight vector u (by default $u = (1, \ldots, 1)$) as described above, and then uses Algorithm 5.11, where the occurring systems of linear equations are solved by the procedure `linReduce`.

`bfctAnn` computes $\mathrm{Ann}_{D_n[s]}(f^s)$ via Briançon-Maisonobe and intersects $\mathrm{Ann}_{D_n[s]}(f^s) + \langle f \rangle$ with $\mathbb{K}[s]$ analogously to `bfct`.

`bfctOneGB` computes the initial ideal and the intersection at once using a homogenized elimination ordering, a similar approach has been used in [16].

For the global b-function of an ideal $I \subset D_n$, `bfctIdeal` computes $\mathrm{in}_{(-w,w)}(I)$ using standard homogenization, i. e. weighted homogenization where all weights are equal to 1, and then proceeds the same way as `bfct`. Recall that D_n/I must be holonomic as in [34].

All these procedures work as the following example illustrates for `bfct` and the hyperplane arrangement $xyz(z-y)(y+z)$.

```
LIB "bfun.lib";
ring r = 0,(x,y,z),dp;    // commutative ring
poly f = x*y*z*(z-y)*(y+z);
list L = bfct(f);
print(matrix(L[1]));      // the roots of the BS-polynomial
==> -1,-5/4,-3/4,-3/2,-1/2
L[2];                     // the multiplicities of the roots above
==>  3,1,1,1,1
```

6.3 Local Bernstein-Sato Polynomial

Here we are interested in what kind of information one can obtain from the local b-functions for computing the global one and conversely. In order to avoid theoretical problems we will assume in this paragraph that the ground field $\mathbb{K} = \mathbb{C}$.

Several algorithms to obtain the local b-function of a hypersurface f have been known without any Gröbner bases computation but under some conditions on f. For instance, it was shown by Malgrange [24] that the minimal polynomial of $-\partial_t t$ acting on some vector space of finite dimension coincides with the reduced (local) Bernstein polynomial, assuming that the singularity is isolated.

The algorithms of Oaku [32] used Gröbner bases for the first time. Recently, Nakayama presented some algorithms, which use the global b-function as a bound and obtain a local b-function by Mora resp. approximate division [27], see also the work of Nishiyama and Noro [29].

Theorem 6.3 (Briançon-Maisonobe (unpublished), Mebkhout-Narváez [26])**.** *Let $b_{f,P}(s)$ be the local b-function of f at the point $P \in \mathbb{C}^n$ and $b_f(s)$ the global one. Then it is verified that $b_f(s) = \operatorname{lcm}_{P\in\mathbb{C}^n} b_{f,P}(s) = \operatorname{lcm}_{P\in\Sigma(f)} b_{f,P}(s)$, where $\Sigma(f) = V(\langle f, \frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_n}\rangle)$ denotes the singular locus of $V(f)$.*

Remark 6.4. Assume, that $\Sigma(f)$ consists of finitely many isolated singular points (the dimension of the corresponding defining ideal is 0). Then the computation of the global b-function with Theorem 6.3 becomes effective. Moreover, one needs just an algorithm for computing the local b-function of a hypersurface, having an isolated singularity at the origin.

The SINGULAR library `gmssing.lib`, developed and implemented by M. Schulze [36], contains the procedure `bernstein`, which computes the local b-function at the origin. It returns the list of roots and corresponding multiplicities.

Example 6.5. Let $\mathcal{C}$ be the curve in $\mathbb{C}^2$ given by $f = (x^3 - y^2)(3x - 2y - 1)(x + 2y)$. This curve has three isolated singular points $p_1 = (0,0)$, $p_2 = (1,1)$ and $p_3 = (1/4, -1/8)$.

```
LIB "gmssing.lib";
// note that one must use a local ordering for calling 'bernstein'
ring r = 0,(x,y),ds;   // ds stands for a local degrevlex ordering
poly f = (x^3-y^2)*(3x-2y-1)*(x+2y);
list L = bernstein(f); // local b-function at the origin p_1
print(matrix(L[1]));
==> -11/8,-9/8,-1,-7/8,-5/8
L[2];
==>    1,1,2,1,1
```

Moving to the corresponding points we also compute $b_{f,P_2}(s)$ and $b_{f,P_3}(s)$.

$$\begin{aligned} b_{f,P_1}(s) &= (s+1)^2(s+5/8)(s+7/8)(s+9/8)(s+11/8) \\ b_{f,P_2}(s) &= (s+1)^2(s+3/4)(s+5/4) \\ b_{f,P_3}(s) &= (s+1)^2(s+2/3)(s+4/3) \end{aligned}$$

From this information and using Theorem 6.3, the global b-function is

$$b_f(s) = (s+2/3)(s+5/8)(s+3/4)(s+7/8)(s+1)^2(s+4/3)(s+5/4)(s+9/8)(s+11/8).$$

Moreover, `gmssing.lib`, allows one to compute invariants related to the Gauss-Manin system of an isolated hypersurface singularity.

In the non-isolated case the situation is more complicated. For computing the local b-function in this case (which is important on its own) we suggest using two methods: Take the global b-function as an upper bound and a local version of the `checkRoot` algorithm, see below. Another method is to use a local version of `principalIntersect`, which is under development. Despite the existence of many algorithms, the effectiveness of the computation of local b-functions is still to be drastically enhanced.

6.4 Partial knowledge of Bernstein-Sato polynomial

As we have mentioned, several algorithms for computing the b-function associated with a polynomial have been known. However, in general it is very hard from computational point of view to obtain this polynomial, and in the actual computation a limited number of examples can be treated. For some applications only the integral roots of $b_f(s)$ are needed and that is why we are interested in obtaining just a part of the Bernstein-Sato polynomial.

Recall the algorithm `checkRoot` for checking whether a rational number is a root of the b-function of a hypersurface from [20]. Equation (1) was used to prove the following result.

Theorem 6.6. *([20]) Let R be a ring whose center contains $\mathbb{K}[s]$ as a subring. Let us consider $q(s) \in \mathbb{K}[s]$ a polynomial in one variable and I a left ideal in R satisfying $I \cap \mathbb{K}[s] \neq 0$. Then $(I + R\langle q(s)\rangle) \cap \mathbb{K}[s] = I \cap \mathbb{K}[s] + \mathbb{K}[s]\langle q(s)\rangle$. In particular, using the above equation* (1)*, we have*

$$\left(\operatorname{Ann}_{D_n[s]}(f^s) + D_n[s] \cdot \langle f, q(s)\rangle \right) \cap \mathbb{K}[s] = \langle b_f(s), q(s)\rangle.$$

As a consequence, let m_α be the multiplicity of α as a root of $b_f(-s)$ and let us consider the ideals $J_i = \operatorname{Ann}_{D_n[s]}(f^s) + \langle f, (s+\alpha)^{i+1}\rangle \subseteq D_n[s]$, $i = 0, \ldots, n$, then $[\, m_\alpha > i \iff (s+\alpha)^i \notin J_i \,]$.

Once we know a system of generators of the annihilator of f^s in $D_n[s]$, the last theorem provides an algorithm for checking whether a given rational number is a root of the b-function of f and for computing its multiplicity, using Gröbner bases for differential operators.

This algorithm is much faster, than the computation of the whole Bernstein polynomial via Gröbner bases, because no elimination ordering is needed for computing a Gröbner basis of J_i, once one knows a system of generators of $\operatorname{Ann}_{D_n[s]}(f^s)$. Also, the element $(s+\alpha)^{i+1}$, added as a generator, seems to simplify tremendously such a computation. Actually, when $i = 0$ it is possible to eliminate the variable s in advance and we can perform the whole computation in D_n. Let us see an example.

Example 6.7. Let A be the matrix given by

$$A = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \\ x_5 & x_6 & x_7 & x_8 \\ x_9 & x_{10} & x_{11} & x_{12} \end{pmatrix}.$$

Let us denote by Δ_i, $i = 1, 2, 3, 4$, the determinant of the minor resulting from deleting the i-th column of A, and consider $f = \Delta_1\Delta_2\Delta_3\Delta_4$. The polynomial f defines a non-isolated hypersurface in $\mathbb{C}^{12}$. Therefore, from [33] (see also [41]), the set of all possible integral roots of $b_f(-s)$ is $\{11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1\}$. It is known that $\mathrm{Ann}_{D_n[s]}(f^s) = \mathrm{Ann}^{(1)}_{D_n[s]}(f^s)$ (see Section 8) and this fact can be used to simplify the computation of the annihilator.

```
LIB "dmod.lib";
ring R = 0,(x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12),dp;
matrix A[3][4] = x1,x2,x3,x4, x5,x6,x7,x8, x9,x10,x11,x12;
poly Delta1 = det(submat(A,1..3,intvec(2,3,4)));
...                                 // analogous for Delta2 ... Delta4
poly f = Delta1*Delta2*Delta3*Delta4;
def D = Sannfslog(f); setring D;    // logarithmic annihilator
poly f = imap(R,f); number alpha = 11;
checkRoot1(LD1,f,alpha);
==> 0
```

Using the algorithm `checkRoot` we have proved that the minimal integral root of $b_f(s)$ is -1. This example was suggested by F. Castro-Jiménez and J. M. Ucha for testing the Logarithmic Comparison Theorem. A nice introduction to this topic can be found, for instance, in [39].

Let g be the polynomial resulting from f by substituting $x_1, x_2, x_3, x_4, x_5, x_9$ with 1. One can show that $b_g(s)$ divides $b_f(s)$ (see [21] for details). Using the `checkRoot` algorithm we have found that $(s+1)^4(s+1/2)(s+3/2)(s+3/4)(s+5/4)$ is a factor of $b_g(s)$ and therefore a factor of $b_f(s)$.

Remark 6.8. Using the notation from Section 5, given a holonomic D-module D/I, it is verified that $(\mathrm{in}_{(-w,w)}(I) + \langle q(s)\rangle) \cap \mathbb{K}[s] = \langle b_{I,w}(s), q(s)\rangle$, although Theorem 6.6 cannot be applied, since $s = \sum w_i x_i \partial_i$ does not commute with all operators. For some applications like integration and restriction the maximal and the minimal integral root of the b-function of I with respect to some weight vector have to be computed, see [34]. However, the above formula cannot be used to find the set of all integral roots, since no upper/lower bound exists in advance. For instance, as it was suggested by N. Takayama, $I = \langle t\partial_t + k\rangle$, $k \in \mathbb{Z}$ is D_1-holonomic and one has $\mathrm{in}_{(-1,1)}(I) \cap \mathbb{C}[s] = \langle s + k\rangle$ with $s = t\partial_t$.

We close this section by mentioning that there exist some well-known methods to obtain an upper bound for the Bernstein-Sato polynomial of a hypersurface singularity once we know, for instance, an embedded resolution of such singularity [18]. Therefore using this result by Kashiwara and the `checkRoot` algorithm, it is possible to compute the whole Bernstein-Sato polynomial without elimination orderings, see Example 1 in [20]. We investigate different methods in conjunction with the further development of the `checkRoot` family of algorithms in [21].

7 Bernstein operator of f

We define the Bernstein-Sato polynomial $b_f(s)$ to be the monic generator of a principal ideal, hence it is unique. But the so-called B-operator $P(s) \in D_n[s]$ from Theorem 6.2 is not unique.

Proposition 7.1. *Let G be a left Gröbner basis of $\operatorname{Ann}_{D_n[s]}(f^{s+1})$ and define a Bernstein operator to be the result of the reduced normal form $\operatorname{NF}(P(s), G)$ of some B-operator $P(s)$. Then, for a fixed monomial ordering on $D_n[s]$, the Bernstein operator is uniquely determined.*

Proof. Suppose that there is another $Q(s) \in D_n[s]$, such that the identities

$$P(s)f^{s+1} = b_f(s)f^s \text{ and } Q(s)f^{s+1} = b_f(s)f^s \text{ hold in the module } D_n[s]/\operatorname{Ann}_{D_n[s]}(f^s).$$

Then $(P(s) - Q(s))f^{s+1} = 0$, that is $P(s) - Q(s) \in \operatorname{Ann}_{D_n[s]}(f^{s+1})$. Hence, the set $\{R(s) \in D_n[s] \mid R(s)f^{s+1} = b_f(s)f^s\}$ can be viewed as an equivalence class and we can take the reduced normal form of any such operator (with respect to G) to be the canonical representative of the class. Since the reduced normal form with respect to a fixed monomial ordering on $D_n[s]$ is unique, so is the Bernstein operator $\operatorname{NF}(R(s), G)$. □

Note, that we can obtain the left Gröbner basis of $\operatorname{Ann}_{D_n[s]}(f^{s+1})$ via substituting s with $s+1$ in the left Gröbner basis of $\operatorname{Ann}_{D_n[s]}(f^s)$.

One can compute the Bernstein operator from the knowledge of $\operatorname{Ann}_{D_n[s]}(f^s)$ and $b_f(s)$ by the following methods.

7.1 B-operator via lifting

The algorithm $\textsc{Lift}(F, G)$ computes the transformation matrix, expressing the set of polynomials G via the set F, provided $\langle G\rangle \subseteq \langle F\rangle$. It is a classical application of Gröbner bases.

Lemma 7.2. *Suppose, that $\operatorname{Ann}_{D_n[s]}(f^s)$ is generated by $h_1, \dots, h_m$ and $b_f(s)$ is known. The output of $\textsc{Lift}(\{f, h_1, \dots, h_m\}, \{b_f(s)\})$ is the $1 \times (m+1)$ matrix $(a, b_1, \dots, b_m)$. Then a B-operator is computed as $P(s) = \operatorname{NF}(a, \operatorname{Ann}_{D_n[s]}(f^s))$.*

Proof. Because of Equation (1), if $(a, b_1, \dots, b_m)$ is the output of $\textsc{Lift}$ as in the statement,

$$b_f(s) = af + \sum_{i=1}^{m} b_i h_i \text{ holds,}$$

hence the first element of such a matrix is a B-operator. Thus, the Bernstein operator is obtained via $\operatorname{NF}(a, \operatorname{Ann}_{D_n[s]}(f^{s+1}))$. □

However, we have to mention, that the $\textsc{Lift}$ procedure is quite expensive in general.

Note that another method for the computation of a B-operator using lifting techniques is given by applying Algorithm 8 of [29] with $a(x) = 1$.

7.2 B-operator via kernel of module homomorphism

1. Consider the $D_n[s]$-module homomorphism

$$\varphi : D_n[s] \longrightarrow D_n[s]/(\operatorname{Ann}_{D_n[s]}(f^s) + \langle b_f(s)\rangle), \quad 1 \mapsto f,$$

then for $u \in \ker \varphi$, $uf \in \mathrm{Ann}_{D_n[s]}(f^s) + \langle b_f(s) \rangle$. That is, there exist $a, b_i \in D_n[s]$, such that

$$uf = ab_f(s) + \sum_{i=1}^{m} b_i h_i,$$

However, we are interested in such u, that $a \in \mathbb{K}$. This is possible, but the 2nd method above proposes a more elegant solution. Also one has to say, that in this case we have to compute a Gröbner basis of $\mathrm{Ann}_{D_n[s]}(f^s) + \langle b_f(s) \rangle$ as an intermediate step and also the kernel of a module homomorphism with respect to the latter. This combination is, in general, quite nontrivial to compute. In the Gröbner basis computation a monomial ordering, preferring $x, \partial x$ over s seems to be better because of numerous applications of the Product Criterion.

2. Consider the $D_n[s]$-module homomorphism

$$\vartheta : D_n[s]^2 \longrightarrow D_n[s]/\,\mathrm{Ann}_{D_n[s]}(f^s), \quad \epsilon_1 \mapsto b_f(s), \epsilon_2 \mapsto f.$$

Then $\ker \vartheta = \{(u, v)^T \in D_n[s]^2 \mid ub_f(s) + vf \in \mathrm{Ann}_{D_n[s]}(f^s)\}$ is a submodule of $D_n[s]^2$. Indeed, $\ker \vartheta$ has many generators. In order to get a vector of the form $(k, u(s))$ for $k \in \mathbb{K}$, we perform another Gröbner basis computation for a submodule with respect to a module monomial ordering, giving preference to the first component over the second one. Since in the reduced basis there is a single element of the form $(k, v(s)) \subset \ker \vartheta$ with $k \neq 0$, it follows that $P(s) = v(s)k^{-1}$.

This algorithm is implemented in `dmod.lib` as `operatorModulo`. The approach via lifting is used in the procedure `operatorBM`, which computes all the Bernstein data. The procedures can be used as follows.

Example 7.3. Consider the Reiffen curve $f = x^2 + y^3 + xy^2 \in \mathbb{K}[x, y]$. At first we use `operatorBM` and compare the length of an B-operator computed via lift to the length of the Bernstein operator.

```
LIB "dmod.lib";
ring r = 0,(x,y),dp;
poly F = x^2 + y^3 + x*y^2;
def A = operatorBM(F); setring A;
size(PS);                           // size of B-operator
==> 238
ideal LD2 = subst(LD,s,s+1);        // Ann(F^{s+1})
LD2 = groebner(LD2);                // LD is not a Groebner basis
poly PS2 = NF(PS,LD2); size(PS2);   // size of Bernstein operator
==> 41
```

So, computing with lifting potentially computes much longer operators. Let us compare with `operatorModulo`.

```
poly PS3 = operatorModulo(F,LD,bs); size(PS3);
==> 41
```

The size of the operator, returned by `operatorModulo` need not be minimal (e. g. by disabling some of the interactive options of Singular one can get a polynomial of length 50 in this example), but it is in general much shorter, than the one, delivered by `operatorBM`. Let us check the main property of the B-operator and print its highest terms:

```
NF(PS3*F - bs, subst(LD2,s,s-1));
==> 0
108*PS2;                              // i.e. the Bernstein operator
==> 6*x*Dx^2*Dy+9*y*Dx^2*Dy-2*x*Dx*Dy^2-4*y*Dx*Dy^2-y*Dy^3+ ...
```

In the last line we see the terms of highest degree with respect to $\partial x, \partial y$.

7.3 Gröbner free method

As a consequence of Theorem 6.2, one obtains that $P \cdot f - b_f(s) \in \operatorname{Ann}_{D_n[s]}(f^s)$ and $P, b_f(s) \notin \operatorname{Ann}_{D_n[s]}(f^s)$. If we fix an ordering such that $b_f(s) = \operatorname{NF}(b_f(s), \operatorname{Ann}_{D_n[s]}(f^s))$ holds, we may rewrite this relation to $b_f(s) = \operatorname{NF}(P \cdot f, \operatorname{Ann}_{D_n[s]}(f^s))$. Hence, we can compute P by searching for a linear combination of monomials $m \in D_n[s]$ that satisfy this equality when multiplied with f from the right side. Using the results from the beginning of this section, one only needs to consider monomials which span $D_n/\operatorname{Ann}_{D_n[s]}(f^{s+1})$ as $\mathbb{K}$-vector space. We get the following algorithm.

Algorithm 7.4.
Input: $f \in \mathbb{K}[x_1, \dots, x_n]$, the Bernstein-Sato polynomial $b_f(s)$ of f
Output: $P \in D_n[s]$, the Bernstein operator of f
  $d := 0$
  loop
    $M_d := \{m \in D_n[s] \mid m \text{ monomial}, \deg(m) \le d, \operatorname{lm}(p) \nmid m \,\forall\, p \in \operatorname{Ann}_{D_n[s]}(f^{s+1})\}$
    if there exist $a_m \in \mathbb{K}$ such that $b_f(s) = \sum_{m \in M_d} a_m \operatorname{NF}(m \cdot f, \operatorname{Ann}_{D_n[s]}(f^s))$ **then**
      return $P := \sum\limits_{m \in M_d} a_m m - b_f(s)$
    else
      $d := d + 1$
    end if
  end loop

The search for the coefficients a_m can be done using `linReduce` (cf. Algorithm 5.11) as one is in fact looking for a linear dependency between the Bernstein-Sato polynomial and the elements $m \cdot f$ in the vector space $D_n[s]/\operatorname{Ann}_{D_n[s]}(f^s)$.

Remark 7.5. Note, that Algorithm 7.4 can be extended to one searching for both B-operator and Bernstein-Sato polynomial simultaneously. We have to mention, that both algorithms of this kind are well suited for the search of operators and Bernstein-Sato polynomials in the case, when both of them are of relatively low total degree.

7.4 Computing integrals and zeta functions

Given a simplex $C \subset \mathbb{K}^n$ (for $\mathbb{K} = \mathbb{R}, \mathbb{C}$) and $f \in \mathbb{K}[x_1, \dots, x_n]$, we can define $\zeta(s) := \int_C f(x)^s dx$. Since $P(s) \bullet f^{s+1} = b_f(s) f^s$, we obtain

$$\zeta(s) = \int_C f(x)^s dx = \frac{1}{b_f(s)} \int_C P(s) \bullet f(x)^{s+1} dx$$

expanding the latter with e. g. the chain rule, we come to an in general inhomogeneous recurrence relation for $\zeta(s)$, which involves coefficients in $\mathbb{K}[s]$. Since $P(s)$ is globally defined (and is, of course, independent on C), one can obtain a generic formula for all integrals of this type.

Example 7.6. Let $f = x^2 - x \in \mathbb{K}[x]$. Then the Bernstein operator reads as $P(s) = (2x-1)\partial_x - 4(s+1)$ and $b_f(s) = s+1$. Any simplex in $\mathbb{K}^1$ is the interval $[a,b] =: C$.

$$\zeta(s) = \int_C f(x)^s dx = \frac{1}{s+1}\int_C ((2x-1)\partial_x - 4(s+1)) \bullet f(x)^{s+1} dx$$
$$= \frac{1}{s+1}\int_C (2x-1)\partial_x \bullet f(x)^{s+1} dx - 4\zeta(s+1)$$

By the chain rule, $\int_C (2x-1)(\partial_x \bullet f(x)^{s+1})dx = (2x-1)f(x)^{s+1} \mid_C -2\int_C f(x)^{s+1})dx$, hence

$$\zeta(s) = \frac{1}{s+1}\cdot(2x-1)f(x)^{s+1}\mid_C - \frac{2}{s+1}\zeta(s+1) - 4\zeta(s+1),$$

and thus

$$(4s+6)\zeta(s+1) + (s+1)\zeta(s) = (2b-1)(b^2-b)^{s+1} - (2a-1)(a^2-a)^{s+1}$$

The right hand side, say $R(s)$, satisfies the homogeneous recurrence $R(s+2) - (a^2 - a + b^2 - b)R(s+1) + (a^2-a)(b^2-b)R(s) = 0$ of order 2. Substituting the left hand side into it, we obtain a homogeneous recurrence with polynomial coefficients of order 3:

$$(a^2-a)(b^2-b)(s+1)\zeta(s) - ((s+2)(a^2-a+b^2-b) - (4s+6)(a^2-a)(b^2-b))\zeta(s+1)$$
$$-((4s+10)(a^2-a+b^2-b) - (s+3))\zeta(s+2) + (4s+14)\zeta(s+3) = 0.$$

To guarantee the uniqueness of a solution to this equation, we need to specify 3 initial values, which can be easily done. However, such recurrences very seldom admit a closed form solution, thus most information about $\zeta(s)$ is contained in the recurrence itself.

8 Logarithmic annihilator of f

Given a polynomial $f \in \mathbb{K}[x] = \mathbb{K}[x_1,\dots,x_n]$, consider the left ideal $\mathrm{Ann}^{(1)}_{D_n[s]}(f^s) \subseteq D_n[s]$ generated by those differential operators $P(s) \in D_n[s]$ of total order (in the partials) less than or equal to one, which annihilate f^s. This ideal is clearly contained in $\mathrm{Ann}_{D_n[s]}(f^s)$ and can be generated by elements of the form

$$P(s) = a_0(x,s) + a_1(x,s)\partial_{x_1} + \cdots + a_n(x,s)\partial_{x_n} \in D_n[s],$$

where $(a_0, a_1, \dots, a_n) \in \mathrm{syz}_{\mathbb{K}[x,s]}(f, s\frac{\partial f}{\partial x_1}, \cdots, s\frac{\partial f}{\partial x_n})$. Therefore, for each $f \in \mathbb{K}[x]$ one can compute, by using Gröbner bases in $\mathbb{K}[x,s]$, a system of generators of $\mathrm{Ann}^{(1)}_{D_n[s]}(f^s)$. The corresponding procedure in `dmod.lib` is called `Sannfslog`. Let us see the Reiffen curve $f = x^4 + y^5 + xy^4$ as an example with Singular.

```
LIB "dmod.lib";
ring R = 0,(x,y),dp;
poly f = x^4+y^5+x*y^4;
def A = Sannfslog(f); setring A; LD1;
==> LD1[1]=4*x^2*Dx+5*x*Dx*y+3*x*y*Dy-16*x*s+4*y^2*Dy-20*y*s
==> LD1[2]=16*x*Dx*y^2-125*x*Dx*y-4*x^2*Dy+4*Dx*y^3+5*x*y*Dy+12*y^3*Dy-100*y^2*Dy
 -64*y^2*s+500*y*s
// now we compute the whole annihilator with Sannfs and compare
setring R; def B = Sannfs(f); setring B;
map F = A,x,Dx,y,Dy,s;
ideal LD1 = F(LD1);
LD1 = groebner(LD1);
simplify( NF(LD,LD1), 2);
==> _[1]=36*y^3*Dx^2-36*y^3*Dx*Dy+1125/4*x*y*Dx^2-315/4*x*y*Dx*Dy+ ...
```

And the latter polynomial is not an element of $\operatorname{Ann}^{(1)}_{D_n[s]}(f^s)$ but of $\operatorname{Ann}^{(2)}_{D_n[s]}(f^s) = \operatorname{Ann}_{D_n[s]}(f^s)$.

8.1 The annihilator up to degree k

More generally, for a given $k \geq 1$ one can consider the left ideal $\operatorname{Ann}^{(k)}_{D_n[s]}(f^s) \subseteq D_n[s]$ generated by the differential operators $P(s) \in D_n[s]$ of total order less than or equal to k, such that $P(s)$ annihilate f^s. The tower of ideals

$$\operatorname{Ann}^{(1)}_{D_n[s]}(f^s) \subsetneq \cdots \subsetneq \operatorname{Ann}^{(k_0)}_{D_n[s]}(f^s) = \operatorname{Ann}_{D_n[s]}(f^s)$$

has been recently studied by Narváez in [28]. It is an open problem to find the minimal integer k_0 satisfying the above condition without computing the whole annihilator.

Computationally the annihilator up to degree k can be obtained using Gröbner bases in $\mathbb{K}[x,s]$ as follows. Consider $P(s) = \sum_{|\beta|\leq k} a_\beta \partial^\beta \in \operatorname{Ann}^{(k)}_{D_n[s]}(f^s)$ and let $g_\beta(x,s) \in \mathbb{K}[x,s]$ be the polynomial defined by the formula $\partial^\beta \cdot f^s = g_\beta \cdot f^{s-|\beta|}$. Then $(a_\beta)_{|\beta|\leq k} \in \operatorname{syz}(g_\beta f^{k-|\beta|})_{|\beta|\leq k}$. Eventually, the polynomials $g_\beta(x,s)$ can be computed using the expression given in Lemma 8.1.

Given $\beta = (\beta_1,\ldots,\beta_n) \in \mathbb{N}^n$, as usual $|\beta| = \beta_1 + \cdots + \beta_n$, $\beta! = \beta_1! \cdots \beta_n!$ and $\Delta^\beta_x = \frac{1}{\beta!}\partial^\beta_x$. A *partition* of β is a way of writing β as a sum of integral vectors with non-negative entries. Two sums which only differ in the order of their summands are considered to be the same partition. If $\beta = \sigma_1 + \ldots + \sigma_k$ with $\sigma_i \neq 0$, then σ is said to be a partition of *length* k. The set of all partitions of β (resp. of length k) is denoted by $\mathcal{P}(\beta)$ (resp. $\mathcal{P}(\beta;k)$). Obviously $\mathcal{P}(\beta) = \cup_{k=1}^{|\beta|}\mathcal{P}(\beta;k)$. Finally, we write $\ell(\sigma) := (\ell_{\sigma\tau})_\tau$, where $\ell_{\sigma\tau}$ is the number of times that τ appears in σ.

Lemma 8.1. *Using the above notation for all non-zero $\beta \in \mathbb{N}^n$ we have,*

$$\Delta^\beta_x \cdot f^s = \sum_{k=1}^{|\beta|} \binom{s}{k} \sum_{\sigma\in\mathcal{P}(\beta;k)} \frac{1}{\ell(\sigma)!}\ \Delta^{\sigma_1}_x(f)\cdots\Delta^{\sigma_k}_x(f)\cdot f^{s-k}.$$

This formula was suggested by Narváez and can be proved by induction on $|\beta|$. Similar expressions appear in [26, Prop. 5.3.5] and [25, Prop. 2.3.2].

However, despite this almost closed form, the set of polynomials, between which we have to compute syzygies, is growing fast and the size of polynomials increases. This results in quite hard computations even with the mentioned enhancements.

9 Bernstein-Sato ideals for $f = f_1 \cdot \ldots \cdot f_m$

Using the results from [13], which we confirmed through intensive testing (cf. [20]), it follows, that the method by Briançon-Maisonobe is the most effective one for the computation of s-parametric annihilators where $f = f_1$. Because of the structure of annihilators in the situation $f = f_1 \cdots f_p$, $p > 1$, basically the same principles stand behind the corresponding algorithms. Hence, we decided to implement only Briançon-Maisonobe's method for the s-parametric annihilator $\mathrm{Ann}_{D_n[s]}(f^s) \subset D_n[s]$, where $s = (s_1, \ldots, s_p)$. The corresponding procedure in `dmod.lib` is called `annfsBMI`. It computes both $\mathrm{Ann}_{D_n[s]} f^s \subset D_n[s]$ and the Bernstein-Sato ideal in $\mathbb{K}[s]$, which is defined as

$$\mathcal{B}(f) = (\mathrm{Ann}_{D_n[s_1,\ldots,s_p]}(f_1^{s_1} \cdots f_p^{s_p}) + \langle f_1 \cdots f_p \rangle) \cap \mathbb{K}[s_1, \ldots, s_p].$$

In contrary to the case $f = f_1$, in general the ideal $\mathcal{B}(f)$ need not be principal. However, it is an open question to give a criterion for the principality of $\mathcal{B}(f)$. Armed with such a criterion, one can apply a generalization of the method of Principal Intersection 5.11 to multivariate subalgebras [2] and thus replace expensive elimination above by the computation of a minimal polynomial. Otherwise we still can apply the Principal Intersection, which, however, will deliver only one polynomial to us. As in the case $f = f_1$ it is an open question, which strategy and which orderings should one use in the computation of the annihilator and of the Bernstein-Sato ideal in order to achieve better performance.

We reported in [20] on several challenges, which have been solved with the help of our implementation. Namely, the products $(x^3 + y^2)(x^2 + y^3)$ and $(x^2 + y^2 + y^3)(x^3 + y^2)$ give rise to principal Bernstein-Sato ideals.

Example 9.1. Let us consider the following example from [3], which is quite challenging to compute indeed.

```
LIB "dmod.lib"; ring r = 0,(x,y,z),dp;
ideal F = z, x^5 + y^5 + x^2*y^3*z;
def A = annfsBMI(F); setring A;
LD; // prints the annihilator in D[s1,s2]
BS; // prints the Bernstein-Sato ideal
```

We do not show the output here because of its size. But from the output one can see, the Bernstein-Sato ideal is of dimension 1 in $D_3[s_1, s_2]$ since its Gröbner basis consists of three elements $\{15625 s_1 s_2^8 + 17 \text{ l.o.t.}, 3125 s_1^2 s_2^7 + 24 \text{ l.o.t.}, 625 s_1^5 s_2^6 + 42 \text{ l.o.t.}\}$. Notably, every generator factorizes into linear factors and each factor involves either s_1 or s_2, which happens quite seldom in general.

In general, quite a little is known about Bernstein-Sato ideals. Their dimensions, principality, factorization of generators and primary decomposition constitute open problems from the theoretical side.

10 Bernstein-Sato polynomial for a variety

Now we proceed to the construction of the Bernstein-Sato polynomial of an affine algebraic variety. We refer to [7] for the details of the complete construction for arbitrary varieties and to [1] for the details about annihilator-driven algorithms.

Given two positive integers n and r, for the rest of this section we fix the indices i, j, k, l ranging between 1 and r and an index m ranging between 1 and n.Let $f = (f_1, \dots, f_r)$ be an r-tuple in $\mathbb{K}[x]^r$. Here, $s = (s_1, \dots, s_r)$, $\frac{1}{f} = \frac{1}{f_1 \cdots f_r}$ and $f^s = f_1^{s_1} \cdots f_r^{s_r}$. Let us denote by $\mathbb{K}\langle S\rangle$ the universal enveloping algebra $U(\mathfrak{gl}_r)$, generated by the set of variables $S = (s_{ij})$, $i, j = 1, \dots, r$, with $s_{ii} = s_i$, subject to relations $[s_{ij}, s_{kl}] = \delta_{jk}s_{il} - \delta_{il}s_{kj}$. We denote by $D_n\langle S\rangle := D_n \otimes_{\mathbb{K}} \mathbb{K}\langle S\rangle$, which is a G-algebra of Lie type by e. g. [22]. Then the free $\mathbb{K}[x, s, \frac{1}{f}]$-module of rank one generated by the formal symbol f^s has a natural structure of a left $D_n\langle S\rangle$-module:

$$s_{ij} \bullet (G(s) \cdot f^s) = s_i \cdot G(s + \epsilon_j - \epsilon_i)\frac{f_j}{f_i} \cdot f^s \in \mathbb{K}[x, s, \frac{1}{f}] \cdot f^s,$$

where $G(s) \in \mathbb{K}[x, s, \frac{1}{f}]$ and ϵ_j stands for the j-th standard basis vector.

Theorem 10.1 (Budur et al. [7])**.** *For every r-tuple $f = (f_1, \dots, f_r) \in \mathbb{K}[x]^r$ there exists a non-zero polynomial in one variable $b(\sigma) \in \mathbb{K}[\sigma]$ and r differential operators $P_1(S), \dots, P_r(S) \in D_n\langle S\rangle$ such that*

$$\sum_{k=1}^{r} P_k(S)f_k \cdot f^s = b(s_1 + \dots + s_r) \cdot f^s \in \mathbb{K}[x, s, \frac{1}{f}] \cdot f^s. \tag{2}$$

The *Bernstein-Sato polynomial* $b_f(\sigma)$ of $f = (f_1, \dots, f_r)$ is defined to be the monic polynomial of lowest degree in the variable σ satisfying the equation (2). It turns out, that every root of the Bernstein-Sato polynomial is rational, as in the case of a hypersurface. Let I be the ideal generated by $f_1, \dots, f_r$ and Z the affine algebraic variety associated with I in $\mathbb{K}^n$. Then it can be verified, that $b_f(\sigma)$ is independent of the choice of a system of generators of I, and moreover that $b_Z(\sigma) = b_f(\sigma - \operatorname{codim} Z + 1)$ depends only on Z.

Now, let us denote by $\operatorname{Ann}_{D_n\langle S\rangle}(f^s)$ the left ideal of all elements $P(S) \in D_n\langle S\rangle$ such that $P(S) \bullet f^s = 0$. We call this ideal the *annihilator* of f^s in $D_n\langle S\rangle$. From the definition of the Bernstein-Sato polynomial it becomes clear that

$$(\operatorname{Ann}_{D_n\langle S\rangle}(f^s) + \langle f_1, \dots, f_r\rangle) \cap \mathbb{K}[s_1 + \dots + s_r] = \langle b_f(s_1 + \ldots + s_r)\rangle.$$

Since the final intersection can be computed with Principal Intersection 5.11, the above formula provides an algorithm for computing the Bernstein-Sato polynomial of affine algebraic varieties, once we know a Gröbner basis of the annihilator of f^s in $D_n\langle S\rangle$.

Theorem 10.2. *Let $f = (f_1, \dots, f_r)$ be an r-tuple in $\mathbb{K}[x]^r$ and $D_n\langle \partial t, S\rangle$ the $\mathbb{K}$-algebra generated by D_n, ∂t and S with the non-commutative relations of $D_n\langle S\rangle$, described above and additional relations $[s_{ij}, \partial t_k] = \delta_{jk}\partial t_i$ (∂t_i commute mutually with the subalgebra D_n). Then the annihilator of f^s in $D_n\langle S\rangle$ can be expressed as follows:*

$$\left[D_n\langle \partial t, S\rangle \Big(s_{ij} + \partial t_i f_j \, , \; \partial_m + \sum_{k=1}^{r} \frac{\partial f_k}{\partial x_m}\partial t_k \;\Big|\; \begin{matrix} 1 \le i, j \le r \\ 1 \le m \le n \end{matrix} \Big) \right] \cap D_n\langle S\rangle.$$

Note, that this result and its proof [1] can be presented as natural generalization of the algorithm for computing $\operatorname{Ann}_{D_n[s]}(f^s)$ with the method of Briançon-Maisonobe (cf. Section 3).

As Budur et al. point out [7, p. 794], the Bernstein-Sato polynomial for varieties coincides, up to shift of variables, with the b-function in [34, p. 194], if the weight vector is chosen appropriately, see also [37]. Algorithms for computing the b-function have been already discussed in Section 5, so the procedure `bfctIdeal` can be immediately applied to this situation. Hence, like for the case of a hypersurface, we have two essentially different ways to compute Bernstein-Sato polynomials for varieties. The comparison of these two methods is the subject of further research.

In the new Singular library `dmodvar.lib`[1], we present the implementations of the following algorithms

`SannfsVar`, which computes $\operatorname{Ann}_{D_n\langle S\rangle}(f^s)$ according to the Theorem 10.2,

`bfctVarIn`, which computes $b_f(s_1+\ldots+s_r)$ using initial ideal approach,

`bfctVarAnn`, which computes $b_f(s_1+\ldots+s_r)$ using annihilator-driven approach.

Example 10.3. Let $TX = V(x_0^2+y_0^3, 2x_0x_1+3y_0^2y_1) \subset \mathbb{C}^4$ the tangent bundle of $X = V(x^2+y^3) \subset \mathbb{C}^2$. Then the Bernstein-Sato polynomial of TX can be computed as follows:

```
LIB "dmodvar.lib";
ring R = 0,(x0,x1,y0,y1),Dp;
ideal F = x0^2+y0^3, 2*x0*x1+3*y0^2*y1;
bfctVarAnn(F); // annihilator-driven approach
// alternatiely, one can run
bfctVarIn(F); // approach via initial ideal
```

In both cases we obtain the polynomial

$$b_{TX}(\sigma) = (\sigma+1)^2(\sigma+1/3)^2(\sigma+2/3)^2(\sigma+1/2)(\sigma+5/6)(\sigma+7/6).$$

The annihilator ideal can be computed via executing, in addition to the first 3 lines of the above code, the following code:

```
def S = SannfsVar(F); // returns a ring
setring S; // in this ring, ideal LD is the annihilator
option(redSB); LD = groebner(LD); // reduced GB of LD
```

There are 15 generators in the Gröbner basis of $\operatorname{Ann} F^s$:

$3y_0^2\partial x_1 - 2x_0\partial y_1,\quad 3y_0^2\partial x_0 + 6y_0y_1\partial x_1 - 2x_0\partial y_0 - 2x_1\partial y_1,\quad x_0y_0\partial x_1\partial y_0 - x_0y_0\partial x_0\partial y_1 + x_1y_0\partial x_1\partial y_1 - 2x_0y_1\partial x_1\partial y_1,\quad 3y_0y_1\partial x_1^2 - x_0\partial x_1\partial y_0 + x_0\partial x_0\partial y_1 - x_1\partial x_1\partial y_1,\quad 3x_0y_0y_1\partial x_0\partial x_1\partial y_1 + 6x_0y_1^2\partial x_1^2\partial y_1 - x_0^2\partial x_1\partial y_0^2 + x_0^2\partial x_0\partial y_0\partial y_1 - 2x_0x_1\partial x_1\partial y_0\partial y_1 + x_0x_1\partial x_0\partial y_1^2 - x_1^2\partial x_1\partial y_1^2 + 3x_1y_0\partial x_1^2 + 3x_0y_1\partial x_1^2 - 3y_0y_1\partial x_1\partial y_1,\quad 6x_0y_1^2\partial x_1^2\partial y_0\partial y_1 + 3x_0y_0y_1\partial x_0^2\partial y_1^2 - 3x_1y_0y_1\partial x_0\partial x_1\partial y_1^2 + 6x_0y_1^2\partial x_0\partial x_1\partial y_1^2 - x_0^2\partial x_1\partial y_0^3 + x_0^2\partial x_0\partial y_0^2\partial y_1 - 2x_0x_1\partial x_1\partial y_0^2\partial y_1 + x_0x_1\partial x_0\partial y_0\partial y_1^2 - x_1^2\partial x_1\partial y_0\partial y_1^2 + 3x_1y_0\partial x_1^2\partial y_0 + 3x_0y_1\partial x_1^2\partial y_0 + 9x_0y_1\partial x_0\partial x_1\partial y_1 - 6y_0y_1\partial x_1\partial y_0\partial y_1 + 3y_0y_1\partial x_0\partial y_1^2 + 6y_1^2\partial x_1\partial y_1^2 +$

[1] it will be distributed with the next release of Singular

$3x_1\partial x_1^2+3y_1\partial x_1\partial y_1,\quad 6x_0y_1^2\partial x_1^3\partial y_1-x_0^2\partial x_1^2\partial y_0^2+2x_0^2\partial x_0\partial x_1\partial y_0\partial y_1-2x_0x_1\partial x_1^2\partial y_0\partial y_1-x_0^2\partial x_0^2\partial y_1^2+2x_0x_1\partial x_0\partial x_1\partial y_1^2-x_1^2\partial x_1^2\partial y_1^2-3x_0y_0\partial x_0\partial x_1^2+3x_1y_0\partial x_1^3+3x_0y_1\partial x_1^3-2x_0\partial x_1\partial y_0\partial y_1+x_0\partial x_0\partial y_1^2-3x_1\partial x_1\partial y_1^2+6y_0\partial x_1^2,\quad s_{22}-x_1\partial x_1-y_1\partial y_1,\quad 6s_{21}-3x_0\partial x_1-2y_0\partial y_1,\quad 6s_{11}-3x_0\partial x_0+3x_1\partial x_1-2y_0\partial y_0+4y_1\partial y_1,\quad s_{12}x_0+3y_0y_1^2\partial x_1-x_0x_1\partial x_0+x_1^2\partial x_1-x_0y_1\partial y_0,\quad s_{12}y_0\partial y_1-x_1y_0\partial x_1\partial y_0+2x_1y_1\partial x_1\partial y_1-y_0y_1\partial y_0\partial y_1+2y_1^2\partial y_1^2-y_0\partial y_0+4y_1\partial y_1,\quad 3s_{12}y_0^2+6x_1y_0y_1\partial x_1-3y_0^2y_1\partial y_0+6y_0y_1^2\partial y_1-2x_0x_1\partial y_0,\quad s_{12}x_1\partial y_1^2-3s_{12}y_0\partial x_1+3x_1y_0y_1\partial x_0\partial x_1\partial y_1+6x_1y_1^2\partial x_1^2\partial y_1-3y_0y_1^2\partial x_1\partial y_0\partial y_1+3y_0y_1^2\partial x_0\partial y_1^2+6y_1^3\partial x_1\partial y_1^2-x_0x_1\partial x_1\partial y_0^2+x_0x_1\partial x_0\partial y_0\partial y_1-2x_1^2\partial x_1\partial y_0\partial y_1-x_1y_1\partial y_0\partial y_1^2+3x_1y_0\partial x_0\partial x_1+3x_1y_1\partial x_1^2-6y_0y_1\partial x_1\partial y_0+9y_0y_1\partial x_0\partial y_1+18y_1^2\partial x_1\partial y_1-2x_1\partial y_0\partial y_1+3y_0\partial x_0,\quad 3s_{12}y_0y_1\partial x_1-s_{12}x_1\partial y_1-3x_1y_0y_1\partial x_0\partial x_1-3y_0y_1^2\partial x_0\partial y_1+x_1^2\partial x_1\partial y_0+x_1y_1\partial y_0\partial y_1-3y_0y_1\partial x_0+x_1\partial y_0.$

This ideal belongs to the $\mathbb{K}$-algebra $D_4\langle S\rangle$ in 12 variables, as defined in the beginning of this section. By executing

```
GKdim(LD);
```

we obtain, that the Gel'fand-Kirillov dimension of $D_4\langle S\rangle/\operatorname{Ann} F^s$ is 6, the half of the Gel'fand-Kirillov dimension of $D_4\langle S\rangle$. However, $D_4\langle S\rangle/\operatorname{Ann} F^s$ is not a generalized holonomic $D_4\langle S\rangle$-module, since the annihilator of this module contains a central element $s_{12}s_{21}-s_{11}s_{22}-s_{11}$ and hence is not zero.

References

[1] D. Andres, V. Levandovskyy, and J. Martín-Morales. Principal intersection and Bernstein-Sato polynomial of an affine variety. In J. P. May, editor, *Proc. of the International Symposium on Symbolic and Algebraic Computation (ISSAC'09)*, pages 231–238. ACM Press, 2009.

[2] D. Andres, V. Levandovskyy, and J. Martín-Morales. Effective methods for the computation of Bernstein-Sato polynomials for hypersurfaces and affine varieties. `http://arxiv.org/abs/1002.3644`, 2010.

[3] R. Bahloul and T. Oaku. Local Bernstein-Sato ideals: algorithm and examples. *J. Symbolic Computation*, 45(1):46–59, 2010.

[4] I. N. Bernstein. Modules over a ring of differential operators. An investigation of the fundamental solutions of equations with constant coefficients. *Functional Anal. Appl.*, 5(2):89–101, 1971.

[5] J. Briançon and P. Maisonobe. Remarques sur l'idéal de Bernstein associé à des polynômes. *Preprint no. 650, Univ. Nice Sophia-Antipolis*, 2002.

[6] M. Brickenstein. Slimgb: Gröbner Bases with Slim Polynomials. In *Rhine Workshop on Computer Algebra*, pages 55–66, 2006. Proceedings of RWCA'06, Basel, March 2006.

[7] N. Budur, M. Mustaţă, and M. Saito. Bernstein-Sato polynomials of arbitrary varieties. *Compos. Math.*, 142(3):779–797, 2006.

[8] J. Bueso, J. Gómez-Torrecillas, and A. Verschoren. *Algorithmic methods in non-commutative algebra. Applications to quantum groups.* Kluwer Academic Publishers, 2003.

[9] F. Castro-Jiménez and L. Narváez-Macarro. Homogenising differential operators. *Prepublicación no. 36, Universidad de Sevilla*, 1997.

[10] S. Coutinho. *A primer of algebraic $\mathcal{D}$-modules.* Cambridge Univ. Press., 1995.

[11] W. Decker, G.-M. Greuel, G. Pfister, and H. Schönemann. Singular 3-1-1. A Computer Algebra System for Polynomial Computations. Centre for Computer Algebra, University of Kaiserslautern. `http://www.singular.uni-kl.de`, 2010.

[12] J. C. Faugere, P. Gianni, D. Lazard, and T. Mora. Efficient computation of zero-dimensional Gröbner bases by change of ordering. *J. Symbolic Computation*, 16(4):329–344, 1993.

[13] J. Gago-Vargas, M. Hartillo-Hermoso, and J. Ucha-Enríquez. Comparison of theoretical complexities of two methods for computing annihilating ideals of polynomials. *J. Symbolic Computation*, 40(3):1076–1086, 2005.

[14] G.-M. Greuel, V. Levandovskyy, and H. Schönemann. Plural. A Singular 3-1-0 Subsystem for Computations with Non-commutative Polynomial Algebras. Centre for Computer Algebra, University of Kaiserslautern. `http://www.singular.uni-kl.de`, 2006.

[15] G.-M. Greuel and G. Pfister. *A SINGULAR Introduction to Commutative Algebra.* Springer, 2nd edition, 2008. With contributions by O. Bachmann, C. Lossen and H. Schönemann.

[16] M. I. Hartillo-Hermoso. About an algorithm of T. Oaku. In *Ring theory and algebraic geometry (León, 1999)*, volume 221 of *Lecture Notes in Pure and Appl. Math.*, pages 241–250. Dekker, New York, 2001.

[17] A. Kandri-Rody and V. Weispfenning. Non-commutative Gröbner bases in algebras of solvable type. *J. Symbolic Computation*, 9(1):1–26, 1990.

[18] M. Kashiwara. B-functions and holonomic systems. Rationality of roots of B-functions. *Invent. Math.*, 38(1):33–53, 1976/77.

[19] V. Levandovskyy. On preimages of ideals in certain non–commutative algebras. In G. Pfister, S. Cojocaru, and V. Ufnarovski, editors, *Computational Commutative and Non-Commutative Algebraic Geometry.* IOS Press, 2005.

[20] V. Levandovskyy and J. Martín-Morales. Computational D-module theory with singular, comparison with other systems and two new algorithms. In *Proc. of the International Symposium on Symbolic and Algebraic Computation (ISSAC'08).* ACM Press, 2008.

[21] V. Levandovskyy and J. Martín-Morales. Algorithms for checking rational roots of b-functions and their applications. `http://arxiv.org/abs/1003.3478`, 2010.

[22] V. Levandovskyy and H. Schönemann. Plural — a computer algebra system for noncommutative polynomial algebras. In *Proc. of the International Symposium on Symbolic and Algebraic Computation (ISSAC'03)*, pages 176 – 183. ACM Press, 2003.

[23] H. Li. *Noncommutative Gröbner bases and filtered-graded transfer.* Springer, 2002.

[24] B. Malgrange. Le polynôme de Bernstein d'une singularité isolée. In *Fourier integral operators and partial differential equations (Colloq. Internat., Univ. Nice, Nice, 1974)*, pages 98–119. Lecture Notes in Math., Vol. 459. Springer, Berlin, 1975.

[25] Z. Mebkhout. Sur le théorème de finitude de la cohomologie p-adique d'une variété affine non singulière. *Amer. J. Math.*, 119(5):1027–1081, 1997.

[26] Z. Mebkhout and L. Narváez-Macarro. Le théorème de continuité de la division dans les anneaux d'opérateurs différentiels. *J. Reine Angew. Math.*, 503:193–236, 1998.

[27] H. Nakayama. Algorithm computing the local b function by an approximate division algorithm in $\widehat{\mathcal{D}}$. *J. Symbolic Computation*, 44(5):449–462, 2009. Spanish National Conference on Computer Algebra.

[28] L. Narváez-Macarro. Linearity conditions on the Jacobian ideal and logarithmic-meromorphic comparison for free divisors. In *Singularities I, Algebraic and Analytic Aspects*, pages 245–269. Contemporary Mathematics, 474, AMS, 2008.

[29] K. Nishiyama and M. Noro. Stratification associated with local b-functions. *J. Symbolic Computation*, 45(4):462 – 480, 2010.

[30] M. Noro. An efficient modular algorithm for computing the global b-function. In *Mathematical software (Beijing, 2002)*, pages 147–157. World Sci. Publ., River Edge, NJ, 2002.

[31] M. Noro, T. Shimoyama, and T. Takeshima. Risa/Asir, an open source general computer algebra system. `http://www.math.kobe-u.ac.jp/Asir`, 2006.

[32] T. Oaku. Algorithms for the b-function and D-modules associated with a polynomial. *J. Pure Appl. Algebra*, 117/118:495–518, 1997.

[33] M. Saito. On microlocal b-function. *Bull. Soc. Math. France*, 122(2):163–184, 1994.

[34] M. Saito, B. Sturmfels, and N. Takayama. *Gröbner deformations of hypergeometric differential equations*, volume 6 of *Algorithms and Computation in Mathematics.* Springer-Verlag, Berlin, 2000.

[35] K. Schindelar, V. Levandovskyy, and E. Zerz. Exact linear modeling using Ore algebras. *J. Symbolic Computation*, 2010. To appear.

[36] M. Schulze. A normal form algorithm for the Brieskorn lattice. *J. Symbolic Computation*, 38(4):1207–1225, 2004.

[37] T. Shibuta. An algorithm for computing multiplier ideals. `http://arxiv.org/abs/math/0807.4302`, 2008.

[38] N. Takayama. KAN/SM1, a Gröbner engine for the ring of differential and difference operators. `http://www.math.kobe-u.ac.jp/KAN/index.html`, 2003.

[39] T. Torrelli. Logarithmic comparison theorem and D-modules: an overview. In *Singularity theory*, pages 995–1009. World Sci. Publ., Hackensack, NJ, 2007.

[40] H. Tsai and A. Leykin. *D*-modules package for MACAULAY 2 – algorithms for D–modules. `http://www.ima.umn.edu/~leykin/Dmodules`, 2006.

[41] A. N. Varchenko. Asymptotic Hodge structure on vanishing cohomology. *Izv. Akad. Nauk SSSR Ser. Mat.*, 45(3):540–591, 1981.

Acknowledgements

We would like to thank Francisco Castro-Jiménez, José-María Ucha, Gert-Martin Greuel, Enrique Artal, José-Ignacio Cogolludo and Luis Narváez for fruitful discussions and insightful remarks concerning our work.

Daniel Andres
Lehrstuhl D für Mathematik, RWTH Aachen, Templergraben 64, 52062 Aachen, Germany
e-mail: `Daniel.Andres@math.rwth-aachen.de`

Michael Brickenstein
Mathematisches Forschungsinstitut Oberwolfach, Schwarzwaldstr. 9-11, 77709 Oberwolfach-Walke, Germany
e-mail: `brickenstein@mfo.de`

Viktor Levandovskyy
Lehrstuhl D für Mathematik, RWTH Aachen, Templergraben 64, 52062 Aachen, Germany
e-mail: `Viktor.Levandovskyy@math.rwth-aachen.de`

Jorge Martín-Morales
Department of Mathematics-I.U.M.A., University of Zaragoza, C/ Pedro Cerbuna, 12 - 50009, Zaragoza, Spain
e-mail: `jorge@unizar.es`

Hans Schönemann
Fachbereich Mathematik, TU Kaiserslautern, Erwin-Schrödinger-Str. 48, 67632 Kaiserslautern, Germany
e-mail: `hannes@mathematik.uni-kl.de`