New Algorithms and Implementations for Computational D-Module Theory

Viktor Levandovskyy   Jorge Martín-Morales

Lehrstuhl D für Mathematik
RWTH Aachen (Germany)
Viktor.Levandovskyy@math.rwth-aachen.de

Depart. of Mathematics-I.U.M.A.
University of Zaragoza (Spain)
jorge@unizar.es

Algebraic and Algorithmic Aspects of Differential
and Integral Operators Session (ACA'08)
RISC-Linz, Castle of Hagenberg, Austria, July 27-30, 2008

Introduction

Basic notations

- $\mathbb{K}$ any field of characteristic 0. (In the examples $\mathbb{K} = \mathbb{Q}$).
- $R_n = \mathbb{K}[x_1, \ldots, x_n]$ the ring of polynomials in n variables.
- $D_n = \mathbb{K}[x_1, \ldots, x_n]\langle \partial_1, \ldots, \partial_n \rangle$ the ring of $\mathbb{K}$-linear differential operators on R_n, that is the *n-th Weyl algebra*:

$$\partial_i x_i = x_i \partial_i + 1$$

- $\mathbb{K}[s]$ the ring of polynomials in one variable over $\mathbb{K}$.
- $D_n[s]$ the ring of polynomials in one variable over D_n.

The $D_n[s]$-module $R_n[s, \frac{1}{f}] \cdot f^s$

- Let $f \in R_n$ be a non-constant polynomial.
- By $R_n[s, \frac{1}{f}]$ we denote the ring of rational functions of the form

$$\frac{g(\mathbf{x}, s)}{f^r}$$

  where $g(\mathbf{x}, s) \in R_n[s] = \mathbb{K}[x_1, \ldots, x_n, s]$.

- We denote by $M = R_n[s, \frac{1}{f}] \cdot f^s$ the free $R_n[s, \frac{1}{f}]$-module of rank one generated by the formal symbol f^s.
- M has a natural structure of left $D_n[s]$-module.

$$\partial_i \cdot f^s = s \frac{\partial f}{\partial x_i} \frac{1}{f} \cdot f^s \quad \in \quad R_n\left[s, \frac{1}{f}\right] \cdot f^s = M$$

Theorem (Bernstein)

For every polynomial $f \in R_n$ there exists a non-zero polynomial $b(s) \in \mathbb{K}[s]$ and a differential operator $P(s) \in D_n[s]$ such that

$$P(s)f^{s+1} = b(s)f^s \quad \in \quad R_n\left[s, \frac{1}{f}\right] \cdot f^s = M.$$

Definition

The set of all possible polynomials $b(s)$ satisfying the above equation is an ideal of $\mathbb{K}[s]$. The monic generator of this ideal is denoted by $b_f(s)$ and called the Bernstein-Sato polynomial of f.

Definition

The left ideal of all elements $P(s) \in D_n[s]$ such that $P(s) \cdot f^s = 0$ is called the annihilator of f^s in $D_n[s]$ and denoted by $\mathrm{Ann}_{D_n[s]}(f^s)$.

Computation of $b_f(s)$

$$b(s)f^s = P(s)f^{s+1} \iff \Big(b(s) - P(s)f\Big) \cdot f^s = 0$$

$$\iff b(s) \in \operatorname{Ann}_{D_n[s]}(f^s) + D_n[s] \cdot \langle\, f \,\rangle$$

$$\Big\langle b_f(s) \Big\rangle = \Big(\operatorname{Ann}_{D_n[s]}(f^s) + \langle f \rangle \Big) \cap \mathbb{K}[s]$$

Ingredients of the LOT algorithm

Questions

- The Oaku-Takayama method uses homogenization. Is this homogenization really necessary?
- Are the methods of Oaku-Takayama and Briançon-Maisonobe connected with each other? If so, how?

Ingredients

- generalized product criterion (V. L. and H. Schönemann, ISSAC 2003)
- preimage of a left ideal under a morphism of $\mathbb{K}$-algebras (V. L., ISSAC 2006)
- slim Gröbner basis algorithm (M. Brickenstein, 2005)

Annihilator of a ring element

Suppose R is a commutative ring and $r \in R$.
Let $\mathcal{O} = \mathcal{O}(R)$ be an algebra of linear operators on R (that is R is a $\mathcal{O}(R)$-module). Then, $\text{Ann}_{\mathcal{O}}\, r = \{a \in \mathcal{O} \mid a \bullet r = 0\} \subset \mathcal{O}$ is a left ideal.
Selecting $\mathcal{O}$ to be the algebra of operators we want to deal with (differential, shift, difference, their q-analogues etc.), the annihilator ideal is a system of operator equations, which has r as a solution.

Message

There is an algorithm for computing $\text{Ann}_{\mathcal{O}}\, r$ for a given $r \in R \cong \mathcal{O}/P_R$.
As soon as $P_R \subset \mathcal{O}$ is given, we need one Gröbner basis computation.

In particular, in the `dmodapp.lib` we provide two functions for $\mathcal{O} = D_n$

- `annPoly`(f) for $f \in \mathbb{K}[x_1, \ldots, x_n]$ and
- `annRat`(f, g) for $h = \frac{f}{g} \in \mathbb{K}(x_1, \ldots, x_n)$.

Ann F^s Method: From Kashiwara to Malgrange

Let $\alpha \in \mathbb{K}$. Then $\operatorname{Ann}_{D_n} f^\alpha = \{a \in D_n \mid a \bullet f^\alpha = 0\}$.

Theorem (Kashiwara 1981)

$D_n / \operatorname{Ann}_{D_n} f^\alpha$ is a (regular) holonomic D_n–module for any $\alpha \in \mathbb{K}$.

Malgrange's construction for $f = f_1 \cdot \dots \cdot f_p$: consider the left ideal

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

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

Theorem

The ideal of operators in $D_n[s] := D_n \otimes_{\mathbb{K}} \mathbb{K}[s]$, annihilating f^s equals to the image of the $I_f \cap D[t \cdot \partial t]$ under the substitution $t \cdot \partial t \mapsto -s - 1$.

The OT algorithm

Let $f = f_1 \cdots f_p \in \mathbb{K}[x_1, \ldots, x_n]$. The relations on the variables:

$$[u_j, *] = [v_j, *] = 0, [\partial_i, x_i] = 1, [\partial t_j, t_j] = 1.$$

Oaku-Takayama (OT) method (1999)

$$D_{n+p}[u_j, v_j] \supset \langle \{t_j - u_j f_j, \sum_{j=1}^{p} \frac{\partial f_j}{\partial x_i} u_j \partial t_j + \partial_i, u_j v_j - 1\} \rangle = \mathcal{I}$$

Let $s_j := -t_j \partial t_j - 1$ and $\mathcal{I}' = \mathcal{I} \cap D_{n+p}$. Then,

$$\mathrm{Ann}_{D_n[\mathbf{s}]}(f^{\mathbf{s}}) = \mathcal{I}' \cap D_n[\{s_j\}].$$

The LOT algorithm

Let $f = f_1 \cdots f_p \in \mathbb{K}[x_1, \ldots, x_n]$. The relations on the variables:

$$[\partial_i, x_i] = 1, [\partial t_j, t_j] = 1, [s_j, t_j] = -t_j, [s_j, \partial t_j] = \partial t_j.$$

Levandovskyy's modification of OT method (2008)

$$D_{n+p}\langle\{s_j\}\rangle \supset \langle\{t_j - f_j, \sum_{j=1}^{p} \frac{\partial f_j}{\partial x_i}\partial t_j + \partial_i, f_j\partial t_j + s_j\}\rangle = \mathcal{L}$$

Then we prove, that

$$\mathrm{Ann}_{D_n[\mathbf{s}]}(f^{\mathbf{s}}) = \mathcal{L} \cap D_n[\{s_j\}].$$

The BM algorithm

With the LOT approach we give the new proof of the algorithm of Briançon-Maisonobe. So, OT $\Rightarrow$ LOT $\Rightarrow$ BM.

Briançon-Maisonobe (BM) method (2002)

$$D_n\langle\{\partial t_j, s_j\} \mid [\partial t_j, s_j] = -\partial t_j\rangle \supset \langle\{s_j + f_j\partial t_j, \sum_{k=1}^{p} \frac{\partial f_k}{\partial x_i}\partial t_k + \partial_i\}\rangle = \mathcal{B}$$

Then

$$\mathrm{Ann}_{D_n[\mathbf{s}]}(f^{\mathbf{s}}) = \mathcal{B} \cap D_n[\{s_j\}].$$

LOT shows: the homogenization in the OT is superfluous.

The checkRoot algorithm

Challenge, intractable with classical methods

The following is an intractable example which is a non-isolated singularity.

$$f = (xz + y)(x^4 + y^5 + xy^4)$$

1. Obtain an upper bound for $b_f(s)$: find $B(s) \in \mathbb{K}[s]$ such that $b_f(s)$ divides $B(s)$.

$$B(s) = \prod_{i=1}^{d} (s - \alpha_i)^{m_i}.$$

2. Check whether α_i is a root of the b-function.
3. Compute its multiplicity m_i.

Remark (Kashiwara)

There are some well-known methods to obtain such $B(s)$ once we know, for instance, an embedded resolution of f.

The main trick

- By definition, $(\mathrm{Ann}_{D_n[s]}(f^s) + \langle f \rangle) \cap \mathbb{K}[s] = \langle b_f(s) \rangle$.
- Take $\alpha \in \mathbb{K}$.
- $(\mathrm{Ann}_{D_n[s]}(f^s) + \langle f \rangle) \cap \mathbb{K}[s] + \langle s + \alpha \rangle = \langle b_f(s), s + \alpha \rangle$

Theorem

Let R be a ring whose center contains $\mathbb{K}[s]$ as a subring. Let $q(s) \in \mathbb{K}[s]$ and I be 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 $(\mathrm{Ann}_{D_n[s]}(f^s) + \langle f, q(s) \rangle) \cap \mathbb{K}[s] = \langle b_f(s), q(s) \rangle$.

Consequences

Corollary

The following conditions are equivalent:

1. $\alpha \in \mathbb{Q}$ is a root of $b_f(-s)$.
2. $\mathrm{Ann}_{D_n[s]}(f^s) + \langle f, s+\alpha \rangle \neq D_n[s]$.

- m_α the multiplicity of α as a root of $b_f(-s)$.
- $J_i = \mathrm{Ann}_{D_n[s]}(f^s) + \langle f, (s+\alpha)^{i+1} \rangle \subseteq D_n[s]$.

Corollary

The following conditions are equivalent:

1. $m_\alpha > i$.
2. $(s+\alpha)^i \notin J_i$. $\Rightarrow$ the algoritm CHECKROOT

Resolution of Singularities

Theorem (Varčenko & Saito)

Every root of $b_f(s)$ belongs to the real interval $(-n, 0)$.

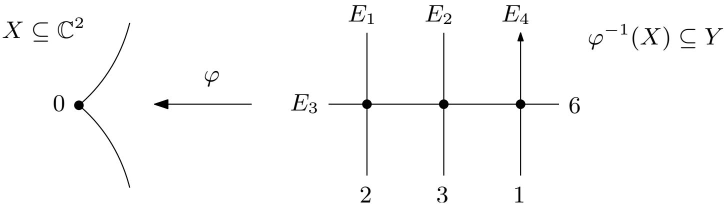
- Let $f \in \mathcal{O}$ be a convergent power series, $f : \Delta \subseteq \mathbb{C}^n \to \mathbb{C}$.
- Assume that $f(0) = 0$, otherwise $b_{f,0}(s) = 1$.
- Let $\varphi : Y \to \Delta$ be an embedded resolution of $\{f = 0\}$.
- Take $F = f \circ \varphi$, then $F^{-1}(0)$ is a normal crossing divisor.

Theorem (Kashiwara)

There exists an integer $k \geq 0$ such that $b_f(s)$ is a divisor of the product $b_F(s)b_F(s+1)\cdots b_F(s+k)$.

Examples

- Let us consider $f = y^2 - x^3 \in \mathbb{C}[x, y]$.



- From Kashiwara, Varcenko (and Saito), the possible roots of $b_f(-s)$ are:

$$\frac{1}{6}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, \frac{5}{6}, 1, \frac{7}{6}, \frac{4}{3}, \frac{3}{2}, \frac{5}{3}, \frac{11}{6}.$$

- Using `checkRoot` algorithms, we have proved that the numbers in red are the roots of $b_f(s)$, all of them with multiplicity one.

Challenge, **which was** intractable with classical methods

Using this method we have computed the b-function of $f = (xz + y)(x^4 + y^5 + xy^4)$ which is a non-isolated singularity.

- Let $B_1(s) = b_{x^5}(s) b_{y^{18}}(s) b_{z^{24}}(s) \in \mathbb{K}[s]$.
- Computing an embedded resolution, we obtain that $B(s) = B_1(s)B_1(s+1)B_1(s+2)$ is an upper bound for $b_f(s)$.

The running time is given in format *[minutes:seconds]*

Root of $B(-s)$	Running time	Root of $b_f(-s)$
$5/4$	29:16	Yes
$31/24$	26:16	Yes
$29/24$	7:51	Yes
$9/8$	0:35	Yes
other roots	$<$ 0:05	Yes/No

Topologically equivalent singularities

- Let f, g be two topologically equivalent singularities.
- Assume that $b_f(s)$ is known.
- Since the set $\{e^{2\pi i \alpha} \mid b_f(\alpha) = 0\}$ is a topological invariant of the singularity $f = 0$ and every root belongs to $(-n, 0)$, one can find an upper bound for $b_g(s)$.
- Then we use algorithms 1 and 2 for computing $b_g(s)$.

Example

- Let $f = x^4 + y^5$ and $g = x^4 + y^5 + xy^4$.
- f and g are topologically equivalent because they have the same Puiseux pairs.
- The following numbers are the roots of $b_f(-s)$, all of them with multiplicity one.

$$\frac{9}{20}, \frac{13}{20}, \frac{7}{10}, \frac{17}{20}, \frac{9}{10}, \frac{19}{20}, \frac{21}{20}, \frac{11}{10}, \frac{23}{20}, \frac{13}{10}, \frac{27}{20}, \frac{31}{20}$$

- The possible roots of $b_g(-s)$ are:

$$\frac{9}{20}, \frac{13}{20}, \frac{7}{10}, \frac{17}{20}, \frac{9}{10}, \frac{19}{20}, \frac{21}{20}, \frac{11}{10}, \frac{23}{20}, \frac{13}{10}, \frac{27}{20}, \frac{31}{20},$$
$$\frac{29}{20}, \frac{33}{20}, \frac{17}{10}, \frac{37}{20}, \frac{19}{10}, \frac{39}{20}, \frac{1}{20}, \frac{1}{10}, \frac{3}{20}, \frac{3}{10}, \frac{7}{20}, \frac{11}{20}.$$

- Using algorithms 1 and 2, we have proved that the numbers in red are the roots of $b_g(-s)$, all of them with multiplicity one.

Another example

Using this method we have computed the Bernstein polynomial for $g = z^4 + x^6 y^5 + x^5 y^4 z$. We chose $f = z^4 + x^6 y^5$ which is topologically equivalent to g.

Comments

The checkRoot algorithm

This algorithm is much faster, than the computation of the whole Bernstein polynomial via Gröbner bases because:

- No elimination ordering is needed.
- The element $(s+\alpha)^{i+1}$ seems to simplify tremendously the computation.

Comments

The checkRoot algorithm

This algorithm is much faster, than the computation of the whole Bernstein polynomial via Gröbner bases because:

- No elimination ordering is needed.
- The element $(s+\alpha)^{i+1}$ seems to simplify tremendously the computation.

Similar approach

One step in Nakayama's algorithm for computing local *b*-function uses a bound for the muliplicity of a given rational root of the global *b*-function. Then his algorithm checks if this multiplicity agrees with the local one. This approach is very similar to our `checkRoot` algorithm.

Structure of the library `dmod.lib`

`annfs`$(F) \Longrightarrow$ `annfsOT`(F), `annfsLOT`(F), `annfsBM`(F)

It computes the Bernstein-Sato polynomial $b_F(s)$ and the ideal $\mathrm{Ann}_D(F^{s_0}) \subset D$ for the D-module structure on $R_n[\frac{1}{F}]$. It calls the `annfsBM` procedure as the default option.

`Sannfs`$(F) \Longrightarrow$ `SannfsOT`(F), `SannfsLOT`(F), `SannfsBM`(F)

It computes $\mathrm{Ann}_{D_n[s]}(F^s) \subset D_n[s]$ and uses the `SannfsBM` procedure as the default option.

`operatorBM`(F)

It computes all the D-modules data relevant to F:

$$\mathrm{Ann}_{D_n[s]}(F^s),\ b_F(s),\ s_0,\ \mathrm{Ann}_{D_n}(F^{s_0}),\ P(s).$$

Structure of the library dmod.lib

annfspecial(I, F, s_0, α), where $I = \text{Ann}_{D_n[s]}(F^s)$

computes the $\text{Ann}_{D_n[s]}(F^\alpha)$ for $\alpha \in \mathbb{K}$.

annfsBMI(F), where $F^s = F_1^{s_1} \cdots F_p^{s_p}$

computes the D-module structure of $R_n[\frac{1}{F}]$ with the multivariate Briançon-Maisonobe algorithm. The procedure outputs $\text{Ann}_{D_n[s_1,\dots,s_p]}(F^s)$ and the Bernstein-Sato ideal $\mathcal{B}_F$.

Flexibility

In all the above procedures it is possible to specify the desired Gröbner basis engine (`std` or `slimgb`) as well as the corresponding algorithm (OT, LOT or BM) via the optional arguments.

Comparison

Example	Polynomial
`cnuN`	$(xz+y)(x^N-y^N)$
`ttNM`	$x^N+y^N+z^N-(xyz)^M$
`ha1`	$xyz(x+y)(x+z)$
`ha2`	$xyzw(x+y)(x+z)(x+w)$
`ha25`	$xyz(x-z)(y-x)(3x+2y+z)(y+z)$
`ha28`	$xyz(x+y)(x+z)(y+z)(y+z-x)$
`ha30`	$xyz(x-y)(y+z)(x-2z)(x-y-z)$
`ha31`	$xyz(x-y)(y+z)(2x-z)(x-y-z)$
`abNM`	$(z^N+w^M)(Nz^{N-1}x+Mw^{M-1}y)$
`xc45`	$(xz+y)(x^4+y^5)$
`reiffen11`	$x^{11}+y^{11}+xy^{10}$
`reiffenNM`	$x^N+y^M+xy^{M-1}$

Comparison of both $\mathrm{Ann}_{D[s]} F^s$ and $b_F(s)$

The running time is given in format [minutes:seconds]

Example	kan/sm1	Macaulay2	Singular
cnu4	0:54	0:11	0:02
cnu5	5:27	1:04	0:18
ha1	21:41	1:13	0:04
reiffen11	19:02	2:22	0:32
reiffen66	0:25	0:03	0:01
reiffen77	0:38	0:06	0:01
reiffen88	1:28	0:14	0:03
reiffen99	4:09	0:30	0:07
tt32	0:25	0:03	0:01
tt42	0:24	0:05	0:01
tt43	3:25	0:21	0:02

Comparison of $\mathrm{Ann}_{D_n[s]}(f^s)$ only

The running time is given in format [minutes:seconds]

Example	Macaulay2	Asir	Singular		
			OT	LOT	BM
ha25	1:31	1:26	1:02	0:46	0:21
ha28	2:00	2:03	1:10	1:17	0:35
ha30	1:58	2:08	1:15	1:36	0:42
ha31	1:23	2:07	1:11	1:11	0:33

Example	Polynomial
ha25	$xyz(x-z)(y-x)(3x+2y+z)(y+z)$
ha28	$xyz(x+y)(x+z)(y+z)(y+z-x)$
ha30	$xyz(x-y)(y+z)(x-2z)(x-y-z)$
ha31	$xyz(x-y)(y+z)(2x-z)(x-y-z)$

Comparison of $\mathrm{Ann}_{D_n[s]}(f^s)$ only

The running time is given in format [hours[h]:minutes:seconds]

Example	Macaulay2	Asir	Singular		
			OT	LOT	BM
reiffen89	3:37	5:20	0:36	0:09	0:03
ab23	4:08	0:25	0:07	0:04	0:02
xc45	16:59	4:09	1:54	1:39	0:59
ha2	6h:25:17	1h:37:08	57:27	52:16	39:20

Example	Polynomial
reiffen89	$x^8 + y^9 + xy^8$
ab23	$(z^2 + w^3)(2zx + 3w^2y)$
xc45	$(xz + y)(x^4 + y^5)$
ha2	$xyzw(x + y)(x + z)(x + w)$

Solved challenges with annfsBMI

In [1][a] there appeared several challenges for the computation of Bernstein-Sato ideals for **products of polynomials**. The following two cases were intractable by any computer algebra system.

1. For $F = (x^2+y^3)(x^3+y^3)$ we spend 68 sec.

$$\mathcal{B} = (s_1+1)\cdot(s_2+2)\cdot\prod_{i\in\Delta}(4s_1+6s_2+i)\cdot(6s_1+4s_2+i)$$

2. For $F = (x^2+y^2+y^3)(x^2+y^3)$ we spend 233 sec.

$$\mathcal{B} = (s_1+1)\cdot(s_2+1)\cdot(s_1+s_2+1)\cdot(s_1+s_2+2)\cdot$$

$$(2s_1+2s_2+3)\cdot(2s_1+2s_2+5)\cdot\prod_{i\in\Delta}(4s_1+6s_2+i)$$

[a] [1] F. Castro-Jiménez and J. Ucha-Enríquez. On the computation of Bernstein-Sato ideals. *J. Symbolic Computation*, 37:629–639, 2004.

Non-principal Bernstei-Sato ideal

$$F = (f_1, f_2) = (z, x^5 + y^5 + zx^2y^3) \qquad \mathcal{B}_F = \langle b_1, b_2, b_3 \rangle$$

$$(s_1+1)(s_2+1)^2(5s_2+2)(5s_2+3)(5s_2+4)(5s_2+6)(s_1+2)(s_1+3)(s_1+4)(s_1+5)$$

$$(s_1 + 1)(s_2 + 1)^2(5s_2 + 2)(5s_2 + 3)(5s_2 + 4)(5s_2 + 6)(5s_2 + 7)(s_1 + 2)$$

$$(s_1 + 1)(s_2 + 1)^2(5s_2 + 2)(5s_2 + 3)(5s_2 + 4)(5s_2 + 6)(5s_2 + 7)(5s_2 + 8)$$

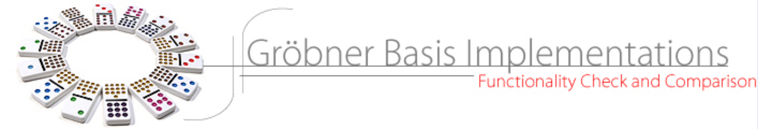
Conclusion

1. Our implementation is very fast in the realm of D-modules algorithms, based on elimination.
2. We confirm that the algorithm of BM for computing $\text{Ann}_{D_n[s]}(f^s)$ is generically the fastest one, while the LOT algorithm is faster than the original algorithm due to OT.

Future work

- Localization, restriction and integration of D-modules.
- Bernstein-Sato polynomial of an arbitrary affine variety.
- De Rham cohomology, local cohomology and Weyl closure.

Visit the Online Database



www.risc.uni-linz.ac.at/Groebner-Bases-Implementations/

Thank you very much!