Multivariate Ore Polynomials in SageMath

Manuel Kauers* and Marc Mezzarobba†

Institute for Algebra · Johannes Kepler University · Linz, Austria
Sorbonne Université, CNRS, Laboratoire d'informatique de Paris 6, LIP6, F-75005 Paris, France

manuel@kauers.de · marc@mezzarobba.net

Abstract

We present the latest update of the ore_algebra package for SageMath. The main new feature in this release is the support of operators in several variables.

1 Introduction

Ore polynomials are important features of computer algebra systems because they provide functionality for manipulating differential and recurrence operators that describe so-called D-finite functions and sequences [13, 9, 7]. Packages for working with Ore polynomials include, among others, OreTools [1] and gfun [12] for Maple, the HolonomicFunctions package of Koutschan [10] for Mathematica, and ore_algebra package by Kauers, Jaroschek and Johansson [8] for SageMath (Sage) [5]. This software demo abstract is about the most recent version of the latter package, which works with Sage 8.7.

Earlier versions of the ore_algebra package already provided functionality for basic arithmetic and actions; GCRD and LCLM; D-finite closure properties; natural transformations between related algebras; guessing; desingularization; solvers for polynomials, rational functions and (generalized) power series. Mezzarobba [11] added a subpackage ore_algebra.analytic for symbolic-numeric computations with D-finite analytic functions, which is now fully integrated into the main version of the package. Hofstadler [6] contributed a set of convenience functions which support a better integration with symbolic expressions.

While basic arithmetic of Ore polynomials in several variables has been supported since the beginning, everything else so far was limited to the univariate case. The main new feature in versions 0.3–0.4 of the package is an extension of the core functionality to multivariate operators, with efficient support of D-finite closure properties and creative telescoping as well as a rudimentary implementation of Gröbner bases. Other novelties include substantially improved numerical evaluation code compared to the status described in [11], and preliminary support for Python 3.

2 Basic Examples

With a typical Sage setup, ore_algebra v0.4 can be installed using the shell command

```
$ sage -pip install git+https://github.com/mkauers/ore_algebra.git@0.4
```

(Omit the `@0.4` suffix for the latest development version.)

After loading the package and creating the base ring $\mathbb{Z}[x, y]$, the Ore algebra of differential operators for x and y can be created as follows.

```
sage: from ore_algebra import *
```

*Supported by the Austrian FWF grants P31571-N32, F5004
†Supported in part by ANR grant ANR-14-CE25-0018-01 (FastRelax).

```
sage: R.<x,y> = PolynomialRing(ZZ)
sage: A.<Dx,Dy> = OreAlgebra(R)
sage: A
```
$\mathbb{Z}[x,y]\langle Dx, Dy\rangle$

The usual arithmetic respects the required commutation rules.
```
sage: (Dx + Dy)*(y + x)
```
$(x+y)\,Dx + (x+y)\,Dy + 2$

```
sage: (x + y)*(Dx + Dy)
```
$(x+y)\,Dx + (x+y)\,Dy$

Differential operators can be applied to objects which Sage knows to differentiate and to multiply by base ring elements:
```
sage: (Dx + Dy)(sin(x*y))
```
$x\cos(xy) + y\cos(xy)$

Using Sx, Sy instead of Dx, Dy creates an Ore algebra with shift operators instead of differential operators. Of course, combinations like $\mathbb{Z}[x,y]\langle Sx, Dy\rangle$ and other generators besides shift and derivation are also possible. The available operators are the same as in the univariate case, see [8] for more details.

In the univariate case, methods for executing closure properties were directly attached to the annihilating operators. In the multivariate case, where several annihilating operators are needed to describe a D-finite object, we must work with left ideals of Ore algebras. Ideals can be constructed as follows.
```
sage: I = A.ideal([Dy^2, Dx*Dy, Dx^2 - y*Dy + 1])
sage: I
```
$\big(Dy^2, Dx\,Dy, Dx^2 + (-y)\,Dy + 1\big)\,\mathrm{Frac}(\mathbb{Z}[x,y])\langle Dx, Dy\rangle$

Note that the base ring is always converted to a field.

Various ideal theoretic operations are available as methods.
```
sage: I.eliminate([Dy])
```
$\big(Dx^3 + Dx\big)\,\mathrm{Frac}(\mathbb{Z}[x,y])\langle Dx\rangle$

```
sage: Dx^3 + Dx in I # (membership)
```
True

```
sage: J = A.ideal([-Dy^2 + x*Dx - 1, Dx*Dy, Dx^2])
sage: I == J
```
False

```
sage: K = I.intersection(J) # closure property ‘addition‘
sage: K <= J # (inclusion)
```
True

```
sage: K = I.symmetric_product(I) # closure property ‘multiplication‘
sage: K.vector_space_basis() # terms under the staircase
```
$\big[1, Dy, Dx, Dy^2, Dx\,Dy, Dx^2\big]$

```
sage: I.annihilator_of_associate(Dx + Dy) # closure property ‘ore action‘
```
$\big(Dy, Dx^3 + Dx\big)\,\mathrm{Frac}(\mathbb{Z}[x,y])\langle Dx, Dy\rangle$

```
sage: I.annihilator_of_composition(x=x^2,y=1-x^2) # closure property ‘composition‘
```

$$\left(Dy, \left(-x^4+x^2\right) Dx^3 + \left(5x^3-3x\right) Dx^2 + \left(-4x^6+4x^4-5x^2+3\right) Dx + 8x^5\right) \mathrm{Frac}(\mathbb{Z}[x,y])\langle Dx, Dy\rangle$$

Most of these operations rely on a generic FGLM implementation which uses the packages own linear system solvers. There is also a direct implementation of Buchberger's algorithm for computing Gröbner bases, which however is not highly optimized.

```
sage: I.groebner_basis()
```
$$\left[Dy^2, DxDy, Dx^2 + (-y)\, Dy + 1\right]$$

All operations discussed so far are generalizations of the univariate counterparts that already existed in earlier versions. A feature that is inherently multivariate is creative telescoping [14, 3, 4]. Here, the input is a left ideal $I \subseteq A$, say $A = \mathbb{Q}(x,y)\langle Dx, Dy\rangle$ and the task is to find the ideal consisting of all operators $P \in \mathbb{Q}(x)\langle Dx\rangle$ such that there exists $Q \in A$ with $P - D_y Q \in I$. The operators P are called *telescopers,* and the corresponding operators Q are called their *certificates.* Our new version of ore_algebra includes an implementation of Chyzak's algorithm [3] for creative telescoping in the general D-finite setting.

Creative telescoping is a central operation for evaluating definite sums and integrals [4]. We cannot explain here in detail the relevance of creative telescoping for these applications, but the use of the package is easily demonstrated:

```
sage: A.ideal([Dx - 2*x*y^2, Dy - 2*x^2*y]).ct(Dy)
```
$$\left([-xDx - 1], [-y]\right)$$

The output is a pair $([P_1, P_2, \dots], [Q_1, Q_2, \dots])$ whose first component is a basis of the ideal of telescopers and whose second component contains the certificates corresponding to the basis elements. The example above originates from the integration problem $\int_y \exp(x^2y^2)dy$. The following example is the essence of a proof of the binomial theorem $\sum_k \binom{n}{k} = 2^n$:

```
sage: B.<Sx,Sy> = OreAlgebra(R)
sage: B.ideal([(y+1)*Sy + (y-x), (x-y+1)*Sx - (x+1)]).ct(Sy - 1)
```
$$\left([Sx - 2], \left[\frac{y}{-x+y-1}\right]\right)$$

Finally, also the guessing features have been extended to the multivariate setting. For example, annihilating operators for the binomial coefficients can be found as follows.

```
sage: data = [[binomial(n,k) for k in range(20)] for n in range(20)]
sage: guess(data, B, point_filter = lambda n,k : k<=n).groebner_basis()
```
$$\left[(y+1)\, Sy - x + y, (x-y+1)\, Sx - x - 1\right]$$

The option `point_filter` informs the guessing engine which part of the input array contains interesting data. A number of further options is specified in the documentation.

3 Some Slightly Larger Examples

The examples shown above need virtually no computation time. In order to get some idea about the performance of our implementation, we consider a set of 19 creative telescoping problems that appeared in a study of restricted lattice walks [2]. Each of the 19 problems starts from a certain rational function $r \in \mathbb{Q}(u,v,t)$ and the task consists of two applications of creative telescoping. Writing I for the ideal of annihilating operators of r in $\mathbb{Q}(u,v,t)\langle Du, Dv, Dt\rangle$, we first have to compute the ideal T_v in $\mathbb{Q}(u,t)\langle Du, Dt\rangle$ of all telescopers of I with respect to Dv and then the ideal $T_{u,v}$ in $\mathbb{Q}(t)\langle Dt\rangle$ of telescopers of T_v with respect to Du. The smallest rational function in the collection is

$$\frac{-u^2v^2+u^2+v^2-1}{tu^3v^2-tu^3v+tu^2v^3-2tu^2v^2+2tu^2v-tu^2-tuv^3+2tuv^2-2tuv+tu-tv^2+tv-u^2v^2+u^2v+uv^2-uv},$$

and the largest is about twice as long. The complete data is available in ore_algebra.examples.ssw.

In the table below, we compare our performance to that of Koutschan's package [10]. We do not claim that a performance comparison will look similarly for any creative telescoping problem, but we think the experiment is at least an indication that our implementation is not too bad. Note in passing that reaching an acceptable performance level required a number of improvements to the implementation of multivariate polynomials and rational functions in Sage itself. Although ore_algebra is developed as an external Sage package, a significant part of the development effort reported here will thus directly benefit other Sage users.

case	1	2	3	4	5	6	7	8	9	10
ore_algebra	601ms	668ms	705ms	862ms	4.82s	10.3s	5.22s	10.7s	11.1s	11.9s
HolonomicFunctions.m	1.55s	1.59s	1.86s	2.09s	25.2s	51.9s	26.8s	55.1s	51.5s	63.4s
case	11	12	13	14	15	16	17	18	19	
ore_algebra	4.46s	8.57s	11.1s	11.7s	4.21s	8.27s	18.9s	36.2s	2.05s	
HolonomicFunctions.m	19.4s	35.4s	51.3s	62.3s	19.1s	34.7s	198s	445s	5.77s	

Acknowledgements. We thank Bruno Grenet, Ben Hutz, Julian Rüth, and Travis Scrimshaw for reviewing the Sage patches mentioned above.

References

[1] S. A. Abramov, H. Q. Le, and Z. Li. OreTools: a computer algebra library for univariate ore polynomial rings. Technical Report CS-2003-12, University of Waterloo, 2003.

[2] A. Bostan, F. Chyzak, M. van Hoeij, M. Kauers, and L. Pech. Hypergeometric expressions for generating functions of walks with small steps in the quarter plane. *European Journal of Combinatorics*, 61:242–275, 2017.

[3] F. Chyzak. An extension of Zeilberger's fast algorithm to general holonomic functions. *Discrete Mathematics*, 217:115–134, 2000.

[4] F. Chyzak. *The ABC of Creative Telescoping – Algorithms, Bounds, Complexity.* Habilitation à diriger des recherches. University Paris-Sud 11, 2014.

[5] P. Zimmermann et al. *Computational Mathematics with SageMath.* SIAM, 2018.

[6] C. Hofstadler. D-finite sequences and functions in Sage. Bachelor's thesis, J. Kepler University, Linz, 2019.

[7] M. Kauers. The holonomic toolkit. In Johannes Blümlein and Carsten Schneider, editors, *Computer Algebra in Quantum Field Theory: Integration, Summation and Special Functions.* Springer, 2013.

[8] M. Kauers, M. Jaroschek, and F. Johansson. Ore polynomials in Sage. In *Computer Algebra and Polynomials*, LNCS 8942, pages 105–125. Springer, 2014.

[9] M. Kauers and P. Paule. *The Concrete Tetrahedron.* Springer, 2011.

[10] C. Koutschan. HolonomicFunctions (User's Guide). Technical Report 10-01, RISC Report Series, University of Linz, Austria, January 2010.

[11] M. Mezzarobba. Rigorous multiple-precision evaluation of D-finite functions in SageMath. Technical Report 1607.01967, ArXiv, 2016. Presented at ICMS'16.

[12] B. Salvy and P. Zimmermann. Gfun: a Maple package for the manipulation of generating and holonomic functions in one variable. *ACM Transactions on Mathematical Software*, 20(2):163–177, 1994.

[13] R. P. Stanley. Differentiably finite power series. *European Journal of Combinatorics*, 1:175–188, 1980.

[14] D. Zeilberger. The method of creative telescoping. *Journal of Symbolic Computation*, 11:195–204, 1991.