D-MODULES FOR MACAULAY 2

ANTON LEYKIN

D-modules for *Macaulay 2* is a collection of the most recent algorithms that deal with various computational aspects of the theory of *D*-modules. This paper provides a brief guide, which gives examples of using the main functions of this package, as well as an overview of the core algorithms for D-modules and their applications.

1 Introduction

The *Macaulay 2 D*-modules package is an implementation of the Weyl algebra and algorithms related to it in the computer algebra system *Macaulay 2*. Over the last decade there were many advances made in the computational theory of *D*-modules. Several newest algorithmic methods used in the analysis of hypergeometric differential equations are described in the recent book ([11]). Also we worked with a paper by Oaku and Takayama ([8]) providing, in particular, a detailed description of the restriction algorithm. The algorithms for computing localization, *D*-homomorphisms, and global b-functions of polynomials with parameters come from ([10]), ([13]), and ([5]) respectively. As to the applications that we describe here, we compute polynomial and rational solutions according to ([9]), local cohomology via Čech complex comes from a paper of Walther ([14]).

Macaulay 2, a noncommercial computer algebra system crafted by Grayson and Stillman, became one of the favorite tools for specialists in algebraic analysis. You are welcome to join the crowd by downloading the current distribution from the web (see ([4])). *D*-modules package comes as a part of it, for the most recent updates and online documentation see ([6]). If you would like to learn more about *Macaulay 2*, read a recently published book ([1]), which contains a set of very interesting examples of computations in algebraic geometry including a section by Walther featuring *D*-modules package.

At the end of the introduction, let us mention that there are several other systems that are capable of handling *D*-modules. First on the list is Takayama's system Kan ([12]), which is a specialized system for *D*-computations. There is an implementation of Weyl algebra in Maple by Chyzak ([3]). It is also implemented in Singular ([2]), although not included in the main package.

2 Making Weyl algebras

Throughout this paper k is a field of characteristic 0, $R_n(k) = k[x_1, ..., x_n]$ is the ring of polynomials in n variables and $A_n(k) = k\langle x_1, ..., x_n, \partial_1, ..., \partial_n\rangle$ is the corresponding Weyl algebra, i.e. an associative k-algebra generated by x's and ∂'s with the relations $\partial_i x_i = x_i\partial_i + 1$ for all i.

In this paper we refer to M as to a D-module if it is a finitely generated left module over a Weyl algebra $D = A_n$.

Weyl algebras are made in *Macaulay 2* by adding the option `WeylAlgebra` to a polynomial ring. For instance,

```
i1 : D = QQ[x_1,x_2,d_1,d_2,
            WeylAlgebra=>{x_1=>d_1,x_2=>d_2}]
o1 = D
o1 : PolynomialRing
```

makes the Weyl algebra with the commutation rules $d_i x_i = x_i d_i + 1$ for $i = 1, 2$.

Now we may do the usual things in *Macaulay 2* such as forming ideals and computing Gröbner bases:

```
i2 : I = ideal(x_1*d_1+2*x_2*d_2-5, d_1^2-d_2)
                                    2
o2 = ideal (x d  + 2x d  - 5, d  - d )
             1 1      2 2       1    2
o2 : Ideal of D
i3 : gb I
o3 = {0} | d_1^2-d_2 x_1d_1+2x_2d_2-5 x_2d_1d_2+...
o3 : GroebnerBasis
```

Weyl algebra construct belongs to the kernel of the system, to load the D-modules package, however, one has to type the following:

```
i4 : load "D-modules.m2"
```

3 Making D-modules

Construction of a D-module is similar to that of a module over a polynomial ring. For instance, it may be presented as a cokernel of a matrix with entries in a Weyl algebra:

```
i5 : A = matrix {{-x_1^3+x_2, 3*d_2*x_1^2+d_1, 0, 0},
                 {0, 0, -x_1^2+x_2, 2*d_2*x_1+d_1}};
              2        4
o5 : Matrix D  <--- D
```

```
i6 : M = cokernel A;
i7 : isHolonomic M
o7 = true
i8 : makeCyclic A
                                  2 2                    2 2
o8 = HashTable{AnnG => ideal (x_1 d_1  + 5x_1 x_2 d_1 d_2  + 6x_2 d_2  + ...
                                1 1      1 2 1 2       2 2
               Generator => | x_2d_1 |
                            | 1      |
o8 : HashTable
```

Function `isHolonomic` checks whether a D-module is holonomic, more on this in the next section. Every holonomic D-module may be presented as a cyclic one: function `makeCyclic` finds such a presentation.

One can find the module D/I generated by a polynomial (rational function): here I is the annihilator ideal of the polynomial (rational function).

```
i9 : f = x_1^2-x_2^3;
i10 : PolyAnn f
                   3       2              3       2       2
o10 = ideal (- x_2 d_1  + x_1 d_1  - 2x_1 , - x_1 d_1  + x_2 d_2  + 3x_2 ,  ...
o10 : Ideal of D
i11 : g = 2*x_1*x_2;
i12 : RatAnn(g,f)
                     2         1   3 2    2 2
o12 = ideal (x_1 d_1  + -*x_2 d_2  + -, x_1 d_1  - x_2 d_1  - 6x_1 x_2 d_1  + ...
                       3         3     2 2      1 2      1 2 1
o12 : Ideal of D
```

Also there are two functions `gkz` and `AppellF1` that cook up ideals representing GKZ (Gelfand-Kapranov-Zelevinsky) systems and Appell F1 system respectively, which are discussed in ([11]).

4 Basic invariants

Let us now compute some basic invariants associated to a D-module D/I. First, we compute the dimension of I from the line `i2` of the previous section and verify that I is indeed holonomic.

```
i13 : Ddim I
o13 = 2
```

Next, we compute its characteristic ideal, which is the initial ideal with respect to the differential order filtration.

```
i14 : charIdeal I
                  2
o14 = ideal (d , x d  + 2x d )
              1   1 1     2 2
o14 : Ideal of QQ [x , x , d , d ]
                    1   2   1   2
```

Note that the initial ideal lives in a polynomial ring. Now, we may compute its singular locus, which is the projection of the characteristic variety minus the zero section on the cotangent bundle onto the base space.

```
i15 : singLocus I
o15 = ideal x
              2
o15 : Ideal of D
```

Finally, we compute the holonomic rank of the system, which tells us what the dimension of the space of solutions of the system is.

```
i16 : Drank I
o16 = 2
```

5 Main tools

In ([8]), Oaku and Takayama develop fundamental algorithms for functors in the category of D-modules. There are four main tools which are heavily utilized – b-functions, localization, resolutions, and restriction. Using them, one gets algorithms for Tor, Ext, local cohomology, deRham cohomology, and other functors.

5.1 b-functions

Given a weight vector $w = (-u, u)$ corresponding to a Gröbner deformation, we are able compute the b-function of D/I in the direction u as follows:

```
i17 : u = {1,3};
i18 : bFunction(I, u)
o18 = $s - 5
o18 : QQ [$s]
```

These types of b-functions with respect to the appropriate weight vectors are also used in Oaku's algorithm to compute global b-functions, a.k.a. Bernstein-Sato polynomials.

```
i19 : f = x_1^2+x_2^2;
i20 : globalBFunction f
        2
o20 = $s  + 2$s + 1
o20 : QQ [$s]
```

The function `paramBpoly` computes the list of possible global b-functions of a polymonial with parametric coefficients together with the corresponding conditions on the parameters.

```
i2 : D = QQ[a,b,c][x,y,dx,dy,WeylAlgebra=>{x=>dx,y=>dy}];
i3 : bList = paramBpoly(a*x^2+b*x*y+c*y^2,"quadratic");
i4 : bList/factorBFunction
                 2          1
o4 = {($s + 1)^2, ($s + -)($s + 1)}
                    2
o4 : List
```

Here we also use `factorBFunction` to factor the polynomials in the output. Factoring a b-function is a simple business due to the fact that the roots of a b-function are rational: this is why a separate function for this is provided.

If we consider a specification of parameters as a point of Proj $\mathbb{Q}[a, b, c]$ then it is proved in ([5]) the set corresponding to a global b-function in our list is constructible. The file named `quadratic.tex`, which is generated by this script, contains:

- $b(s) = (s + 1)^2$ for $V(0) \setminus V(b^2 - 4 * a * c)$
- $b(s) = (s + 1) * (s + 1/2)$ for $V(b^2 - 4 * a * c)$

5.2 Localization

There is a function that computes the localization of a D-module by inverting a polynomial. We show how to compute $\mathbb{Q}[x, y, (x^2 - y^3)^{-1}]$:

```
i2 : D = QQ[x,y,dx,dy, WeylAlgebra=>{x=>dx,y=>dy}];
i3 : M = cokernel matrix{{dx,dy}};
i4 : f = x^2-y^3;
i5 : Dlocalize(M, f)
o5 = cokernel | xdx+2/3ydy+4 y2dx+2/3xdy y3dy-x2dy+6y2 |
                1
o5 : D-module, quotient of D
```

When calling `Dlocalize`, several variants of localization algorithm could be specified by adding a strategy option `Dlocalize(M,f,Strategy=>OTW)` to apply the algorithm from ([10]), or `Dlocalize(M,f,Strategy=>Oaku)` which is good for f-saturated modules and appears in ([14]). The latter usually works faster than the former.

5.3 Resolutions

To compute resolutions for D-modules, use the usual *Macaulay 2* command `res`:

```
i6 : I = gkz(matrix{{1,1,1},{0,1,3}}, {2,3})
                3    2
o6 = ideal (D  - D D , x D  + x D  + x D  - 2, x D  + ...
            2    1 3   1 1    2 2    3 3       2 2
o6 : Ideal of QQ [x , x , x , D , D , D , WeylAlgebra => ...
                  1   2   3   1   2   3
i7 : D = ring I;
i8 : res I
      1       3       11       9
o8 = D  <-- D  <-- D   <-- D  <-- 0
     0       1       2        3      4
o8 : ChainComplex
```

To find a resolution by Schreyer method, we use the special command `Dres`:

```
i9 : Dres I
      1       8       16       12       3
o9 = D  <-- D  <-- D   <-- D   <-- D  <-- 0
     0       1       2        3        4      5
o9 : ChainComplex
```

Finally, we may compute V-strict resolutions, which are resolutions that respect a weight vector $w = (u, -u)$ associated to a Gröbner deformation. These resolutions are compatible with b-functions and thus become especially useful.

```
i10 : w = {1,3,5,-1,-3,-5};
i11 : Dres(I, w, Strategy => Vhomogenize)
      1       5       8       5       1
o11 = D  <-- D  <-- D  <-- D  <-- D  <-- 0
      0       1       2       3       4      5
o11 : ChainComplex
```

5.4 Restriction

Armed with b-functions and V-strict resolutions, we get an algorithm to compute the restriction functors, which are the Tor groups of a left D-module with the right D-module $D/\{x_1,\dots,x_d\}\cdot D$. The following computes the derived restriction to the origin.

```
i12 : w = {1,3,5};
i13 : Drestrict(I, w, Strategy => Vhomogenize)
                          1
o13 = HashTable{0 => QQ  }
                     2
                1 => QQ
                     1
                2 => QQ
                3 => 0
o13 : HashTable
```

By changing the weight vector, we may compute derived restriction to a subspace such as $\{x_1 = 0\}$.

```
i14 : w = {1,0,0};
i15 : Drestrict(I, w, Strategy => Vhomogenize)
o15 = HashTable{0 => cokernel | x_3D_3-1/2 0         ...
                              | 0          x_3D_3-1 ...
                1 => 0
o15 : HashTable
```

6 Applications

6.1 Solving holonomic systems

Polynomial solutions of I can be computed by duality or by Gröbner deformations.

```
i16 : PolySols I
o16 = {x x }
       1 3
o16 : List
```

More generally, the vector space $\mathrm{Hom}_D(\frac{D}{I}, N)$ corresponds to the N-valued solutions of I.

```
i17 : D = QQ[z,Dz, WeylAlgebra=>{z=>Dz}];
i18 : M = cokernel matrix{{(Dz-1)^2}};
i19 : N = cokernel matrix{{Dz*(Dz-1)}};
```

```
i20 : DHom(M,N)
o20 = {| zDz-2Dz |, | Dz |}
o20 : List
i21 : DHom(N,M)
o21 = {| -zDz+z-2Dz+3 |, | -Dz+1 |}
o21 : List
```

6.2 Local cohomology

One of the first algorithmic applications of D-modules to algebraic geometry was computing local cohomology. Let I be an ideal of $R_n = k[x_1, ..., x_n]$ and let M be an R_n-module, then the i-th local cohomology group $H^i_I(M)$ is defined as the i-th derived functor of the functor

$$\Gamma_Y(M) = \lim_{\rightarrow} Hom_{R_n}(R_n/I^m; M),$$

where the inductive limit is taken as m tends to infinity. We may generalize the definition by letting M be a D-module. D-structure proves to be useful, since whenever M is holonomic, so is $H^i_I(M)$ for every i. Hence, we can pass from viewing local cohomology groups as generally infinite R_n-modules to computing them as holonomic D-modules, which have finite description.

There are two algorithms available in the D-modules package: one due to Oaku and Takayama (uses restriction), another due to Walther (utilizes the Čech complex).

```
i2 : D = QQ[x_1..x_6, dx_1..dx_6,
            WeylAlgebra => toList(1..6)/(i->x_i=>dx_i)];
i3 : I = minors(2, matrix{{x_1, x_2, x_3}, {x_4, x_5, x_6}})
o3 = ideal (- x x  + x x , - x x  + x x , - x x  + x x )
             2 4    1 5     3 4    1 6     3 5    2 6
o3 : Ideal of D
i4 : H = localCohom ({0,1,3}, I,
         D^1/ideal{dx_1,dx_2,dx_3,dx_4,dx_5,dx_6})
o4 = HashTable{0 => subquotient ({0} | dx_6 dx_5 dx_4  ...
               1 => subquotient ({0} | 0     dx_5 0    ...
                                 {0} | 0     0    dx_4 ...
                                 {0} | -dx_6 0    0    ...
               3 => cokernel {0} | x_2^2x_4^2-2x_1x_2x_...
o4 : HashTable
i5 : pruneLocalCohom H
o5 = HashTable{0 => 0                                   ...
```

```
                1 => 0
                3 => {0} | x_4dx_4+x_5dx_5+x_6dx_6+6     ...
o5 : HashTable
```

In practice Walther's approach (option `Strategy=>Walther`) shows to be faster than the Oaku-Takayama method (option `Strategy=>OaTa`).

6.3 DeRham cohomology

Another exciting application of D-modules is for computing the deRham cohomology groups. Using the integration and localization functors, Oaku and Takayama showed how to compute these groups for the complement of an affine complex hypersurface. In ([15]) Walther generalized this algorithm to complements of affine complex varieties, and also showed how to compute the cup product structures.

We have implemented an algorithm for the hypersurface case:

```
i6 : R = QQ[y,z];
i7 : f = y^2-z^3;
i8 : deRham f
                          1
o8 = HashTable{0 => QQ  }
                          1
               1 => QQ
               2 => 0
o8 : HashTable
```

7 Example: rank jumps in A-hypergeometric systems

In this section we shall give a practical example of employing the D-modules package. This example is borrowed from the work of Matusevich ([7]).

For an integer $d \times n$ matrix $A = (a_{ij})$ with the first row entries equal to 1 and a vector $\beta \in \mathbb{C}^d$ we define the GKZ(Gelfand-Karpanov-Zelevinsky) *A-hypergeometric system with parameter β* to be the D-ideal $H_A(\beta)$ generated by

$$\partial^u - \partial^v, \text{ where } u, v \in \mathbb{N}^n \text{ such that } A \cdot u = A \cdot v, \tag{1}$$

$$\sum_{j=1}^{n} a_{ij} x_j \partial_j - \beta_i,\ i = 1, ..., d. \tag{2}$$

The commutative ideal of $\mathbb{C}[\partial_1, ..., \partial_n]$ generated by operators (1) is denoted by I_A and referred to as *underlying toric ideal*.

It is a fact that if I_A is Cohen-Macaulay then the D-rank of $H_A(\beta)$ is equal to $\mathrm{vol}(A)$, the normalized volume of the convex hull of A viewed as an n-point configuration in $\mathbb{Z}^d$.

However, if I_A is non-Cohen-Macaulay, it is possible that $\mathrm{rank}(H_A(\beta)) > \mathrm{vol}(A)$. For codimension 2 case it was proved that the *exceptional set* of parameters β for which the inequality holds is a nonempty constructible set.

Example. Using *Macaulay 2*, for

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & -1 & 0 & 3 \\ 0 & 1 & -2 & 0 & 3 \end{pmatrix}, \ \beta = \begin{pmatrix} 3 \\ 3 \\ 2 \end{pmatrix}$$

we are going to show that β is an exceptional parameter for A.

```
i2 : A = matrix{{1,1,1,1,1},
        {1,0,-1,0,3},
        {0,1,-2,0,3}};
                3      5
o2 : Matrix ZZ  <--- ZZ
i3 : b = {3,3,2};
i4 : H = gkz(A,b)
                2                    3      2
o4 = ideal (D D  - D D D , D D  - D D D , ...
            1 2    3 4 5   1 4    2 3 5

o4 : Ideal of QQ [x , x , x , x , x , D , ...
                  1   2   3   4   5   1
i5 : time Drank H
     -- used 89.75 seconds
o6 = 10
```

Here we used function `gkz` to construct the GKZ A-hypergeometric system with parameter β, and then computed the D-rank of the resulting D-ideal. (Operator `time` put in front of any *Macaulay 2* command prints out the computation time.)

The rank is 10, however, an easy computation shows that the normalized volume of A is only 9. Thus, we conclude that the system experiences a rank jump of 1 for the parameter β.

References

1. *Computations in Algebraic Geometry with Macaulay 2.* Algorithms and Computation in Mathematics, Vol. 8, 2001
2. *Singular: a computer algebra system for polynomial computations.* `http://www.singular.uni-kl.de/`

3. Chyzak, Frédéric. *Mgfun Project.* `http://pauillac.inria.fr/algo/chyzak/mgfun.html`
4. Grayson, Daniel; Stillman, Michael. *Computer algebra system Macaulay 2.* `http://www.math.uiuc.edu/Macaulay2`
5. Leykin, Anton. *Constructibility of the set of polynomials with a fixed Bernstein-Sato polynomial: an algorithmic approach.* Effective methods in rings of differential operators. J. Symbolic Comput. 32 (2001), no. 6, 663–675.
6. Leykin, Anton; Tsai, Harrison. *D-modules for Macaulay 2.* `http://www.math.umn.edu/ leykin/Dmodules`
7. Matusevich, Laura. *Rank jumps in codimension 2 A-hypergeometric systems.* Effective methods in rings of differential operators. J. Symbolic Comput. 32 (2001), no. 6, 663–675.
8. Oaku, Toshinori; Takayama, Nobuki. *Algorithms for D-modules – restriction, tensor product, localization, and local cohomology groups.* J. Pure Appl. Algebra 156 (2001), no. 2-3, 267–308.
9. Oaku, Toshinori; Takayama, Nobuki; Tsai, Harrison. *Polynomial and rational solutions of holonomic systems.* Effective methods in algebraic geometry (Bath, 2000). J. Pure Appl. Algebra 164 (2001), no. 1-2, 199–220.
10. Oaku, Toshinori; Takayama, Nobuki; Walther, Uli. *A localization algorithm for D-modules.* Symbolic computation in algebra, analysis, and geometry (Berkeley, CA, 1998). J. Symbolic Comput. 29 (2000), no. 4-5, 721–728.
11. Saito, Mutsumi; Sturmfels, Bernd; Takayama, Nobuki. *Gröbner deformations of hypergeometric differential equations.* Algorithms and Computation in Mathematics, 6. Springer-Verlag, Berlin, 2000.
12. Takayama, Nobuki. *kan/sm1: a computer algebra system for algebraic analysis.* `http://www.math.sci.kobe-u.ac.jp/KAN/`
13. Tsai, Harrison; Walther, Uli. *Computing homomorphisms between holonomic D-modules.* Effective methods in rings of differential operators. J. Symbolic Comput. 32 (2001), no. 6, 597–617.
14. Walther, Uli. *Algorithmic computation of local cohomology modules and the local cohomological dimension of algebraic varieties.* Effective methods in algebraic geometry (Saint-Malo, 1998). J. Pure Appl. Algebra 139 (1999), no. 1-3, 303–321.
15. Walther, Uli. *Computing the cup product structure for complements of complex affine varieties.* Effective methods in algebraic geometry (Bath, 2000). J. Pure Appl. Algebra 164 (2001), no. 1-2