A Sage Package for the Symbolic-Numeric Factorization of Linear Differential Operators

Alexandre Goyer
Inria, Palaiseau, France, 91120
alexandre.goyer@inria.fr

Abstract

We present a SageMath implementation of the symbolic-numeric algorithm introduced by van der Hoeven in 2007 for factoring linear differential operators whose coefficients are rational functions.

1 Introduction

Many functions are solution of a differential equation (E) of the form $a_n(z)f^{(n)}(z) + \cdots + a_1(z)f'(z) + a_0(z)f(z) = 0$ whose coefficients a_i are rational functions. A lot of information concerning these functions can be computed directly from the equations. It is therefore of interest to develop tools for manipulating equations of this form [Sal19].

The left-hand side of (E) can be viewed as the action of the operator $L := a_n\partial^n + \cdots + a_1\partial + a_0$ on the function f where ∂ stands for the differentiation with respect to z. We denote by $\mathbb{K}(z)\langle\partial\rangle$ the algebra of linear differential operators with rational coefficients and we assume that the field of constants $\mathbb{K}$ is a subfield of $\mathbb{C}$. Factoring a linear differential operator consists in writing L as the composition $L = L_1L_2$ of two operators $L_1, L_2 \in \overline{\mathbb{K}}(z)\langle\partial\rangle$ of smaller order, potentially introducing algebraic extensions. The solutions of the right factor L_2 form a subspace of the solutions of L. Note that this fails for left factors because of non-commutativity. In general, it is easier to find an exact symbolic solution or to approximate a numerical solution with equations of smaller order. Hence factoring is useful for manipulating linear differential operators and, consequently, studying their solutions.

This paper presents an implementation[1] (focusing on the Fuchsian case, see below) of the symbolic-numeric factoring algorithm introduced by van der Hoeven in [vdH07]. It mainly provides the command `dfactor(L)` which computes a list of irreducible operators `[L1,...,Lr]` such that `L` is equal to the composition `L1`$\cdots$`Lr`, as illustrated below.

```
sage: from diffop_factorization import z, Dz, dfactor
sage: L = (4*z^2 + 6*z + 2)*Dz^2 + (4*z + 3)*Dz - 1
sage: dfactor(L)
[(4*z + 4)*Dz + 2, (z + 1/2)*Dz - 1/2]
```

Traditionally, algorithms for factoring linear differential operators are purely algebraic (they do not use numerical computation). The first method is due to Beke at the end of the 19th century [Bek94]. Three aspects contribute to the complexity of his method: the so-called "second exterior power method" used to reduce the problem to the task of finding a first order right factor, a combinatorial part (possibly exponential if there are many singular points) and computation over large algebraic extensions. Since the 1990s, improvements were made by many authors (among others, [Sch89], [Bro94], and [Tsa94]). A

[1] The code is available at https://github.com/a-goyer/diffop_factorization under the GNU General Public License.

Algorithm 1 Computing a non-trivial right factor or ensuring that there is none

input: a Fuchsian operator $L \in \overline{\mathbb{Q}}(z)\langle\partial\rangle$
output: a non-trivial right factor of L or `irreducible`

```
function RIGHT_∂FACTOR(L)
    loop
        try
            compute a generating set M of monodromy matrices with rigorous error bounds
            compute a subspace U of V(L) invariant under the action of M
            if U is none then return irreducible
            else
                guess an operator L2 ∈ Q̄(z)⟨∂⟩ which annihilates a nonzero f ∈ U
                check that L2 divides L from the right exactly
                return L2
        catch insufficient precision:
            increase working precision and truncation order
```

detailed complexity analysis of Beke's method can be found in [Gri90] and explicit degree bounds of a right factor in [BRS19]. Singer proposed a different approach (the "eigenring" method) for testing reducibility [Sin96]. Van Hoeij designed a new factoring algorithm (a "local to global" method) which gets around the costly parts of Beke's method in many cases [vH97] and he developed an implementation in Maple. Van Hoeij and Cluzeau gave also a factoring algorithm in positive characteristic [CvH04]. To our knowledge, there is however no efficient algorithm for lifting a modular factorization to characteristic zero.

Van der Hoeven in [vdH07] and later Johansson, Kauers, and Mezzarobba (for first order right factors) in [JKM13] proposed algorithms which involves numerical computation to avoid some tricky parts of the algebraic algorithms but the lack of implementation did not allow one to compare symbolic-numeric approaches with purely algebraic approaches.

2 Van der Hoeven's factoring algorithm

Van der Hoeven's factoring algorithm is sketched in Algorithm 1. The key ingredient is Corollary 3 (page 3 below) which reduces the factorization problem to the task of computing a linear subspace invariant under the action of a finite list of matrices $M_1, \ldots, M_r$. The algorithm starts by computing these matrices with rigorous error bounds then tries to find an invariant subspace. Depending on the reducibility of the input operator, there are two ways to proceed. To certify irreducibility, the error bounds are exploited to ensure that no non-trivial subspace is invariant under the action of $M_1, \ldots, M_r$, in which case a `none` is returned. In the case of reducibility, the algorithm can compute an approximate invariant subspace and consequently an approximate right factor. Two ingredients are used to guess a candidate exact right factor: Hermite–Padé approximants to find rational fractions from truncated power series and the LLL algorithm to find elements of $\overline{\mathbb{Q}}$ from approximate numbers. Finally, the algorithm can prove the exact divisibility by a right Euclidean division in $\overline{\mathbb{Q}}(z)\langle\partial\rangle$. If the working precision is insufficient, the candidate right factor can turn out to be incorrect or some computation steps can fail, in which case an error is raised. We increase the precision until we get either a factorization or an irreducibility certificate.

We shall assume that the reader is familiar with the basic facts of differential Galois theory as presented in [vdPS03]. For the rest of this section, we consider a monic operator $L \in \mathbb{K}(z)\langle\partial\rangle$ of order $n \geq 1$ with $\mathbb{K} \subset \mathbb{C}$ algebraically closed and we fix an ordinary point $z_0 \in \mathbb{K}$ of L. Recall that a point is called ordinary if it is not a pole of a coefficient of L, and it is called singular otherwise.

For each $1 \leq i \leq n$, denote by $h_i = \sum_{p \geq 0} h_{i,p}(z - z_0)^p$ the unique power series solution of L in a

neighborhood of z_0 such that, for $0 \leq j < n$, $h_i^{(j)}(z_0)$ equals 1 if $i = j+1$ and 0 otherwise. It can be proved that $\mathcal{E} := \mathbb{K}(z)(h_1, \ldots, h_n)$ is a Picard–Vessiot extension of $\mathbb{K}(z)$ associated to L (informally, a minimal field extension containing all solutions). The differential Galois group of L is the group (unique up to isomorphism) of all field automorphisms σ of $\mathcal{E}$ fixing $\mathbb{K}(z)$ and satisfying $\sigma(f') = \sigma(f)'$ for any $f \in \mathcal{E}$. The natural left linear action of this group on the solution space $V(L) := \mathrm{Span}_{\mathbb{K}}(h_1, \ldots, h_n)$ of L is closely related to the factorizations of L, as shown in Theorem 1 (see [vdH07], Proposition 5). We identify $V(L)$ and the differential Galois group of L with their matrix representations in the basis $(h_1, \ldots, h_n)$.

Theorem 1 *The map $L_2 \mapsto V(L_2) := \{f \in V(L) \mid L_2 \cdot f = 0\}$ defines a bijection between the monic right factors of L and the subspaces of $V(L)$ invariant under the action of the differential Galois group of L.*

Denote by $\mathrm{Sing}(L)$ the set of singular points of L. Let $(f_1, \ldots, f_n)$ be any basis of $V(L)$. Consider a continuous loop $\gamma \subset \mathbb{C}\backslash\mathrm{Sing}(L)$ with base point z_0. The analytic continuation of $(f_1, \ldots, f_n)$ along γ yields a new basis $(g_1, \ldots, g_n)$ of $V(L)$. One calls monodromy matrix based in z_0 along γ the matrix $\Delta_{z_0,\gamma} \in \mathrm{GL}_n(\mathbb{C})$ such that $\left(g_j^{(i-1)}(z_0)\right)_{1\leq i,j\leq n} = \Delta_{z_0,\gamma}\left(f_j^{(i-1)}(z_0)\right)_{1\leq i,j\leq n}$. It can be proved that $\Delta_{z_0,\gamma}$ only depends on the homotopy class of γ in $\mathbb{C}\backslash\mathrm{Sing}(L)$ and that $\alpha \mapsto \Delta_{z_0,\alpha}$ is a morphism for loop concatenation. The monodromy group of L (in z_0) is the group of all matrices $\Delta_{z_0,\alpha}$ for any loop $\alpha \subset \mathbb{C}\backslash\mathrm{Sing}(L)$ with base point z_0. As expressed by the next result, monodromy provides, under a regularity assumption, an analytic access to the differential Galois group. Recall that an operator is Fuchsian when its solutions have at most polynomial growth in sectors at each singular point. Fuchs' criterion (Corollary 5.5 in [vdPS03]) gives a very simple way to test this.

Theorem 2 *(Density theorem of Schlesinger [Sch97]) If L is Fuchsian, then the differential Galois group of L is the Zariski closure of the monodromy group of L.*

Corollary 3 *Assume that L is a Fuchsian operator and denote by $s_1, \ldots, s_r \in \mathbb{K}$ its finite singular points. For each $1 \leq i \leq r$, let $\gamma_i \subset \mathbb{C}\backslash\{s_1, \ldots, s_r\}$ be a loop with base point z_0 enclosing s_i once and enclosing no s_j for $j \neq i$. The map $L_2 \mapsto V(L_2)$ defines a bijection between the monic right factors of L and the subspaces of $V(L)$ invariant under the action of all monodromy matrices $\Delta_{z_0,\gamma_i} \in \mathrm{GL}_n(\mathbb{C})$.*

3 Implementation

We use ball arithmetic (a form of interval arithmetic implemented in the C library `Arb`, see `https://arblib.org/`) for tracking numerical errors. Our `diffop_factorization` package is mainly based on the `ore_algebra` package for manipulating differential operators, in particular for the rigorous arbitrary-precision computation of monodromy matrices [Mez16].

Computing with balls as "exact" numbers requires deciding what to do with zero-tests. Indeed, let $z \in \mathbb{C}$ be a complex which is only known through a ball $b \subset \mathbb{C}$ containing it. If b does not contain 0, then we are certain that $z \neq 0$, but if b contains 0, we cannot decide whether $z = 0$. However, if $z \neq 0$ and the precision is sufficient, then $0 \notin b$. An optimistic zero-test consists in claiming that $z = 0$ whenever $0 \in b$. Opting for this test is legitimate in Algorithm 1:

1. In the reducible case, this is because candidate factors are validated by an *a posteriori* exact division test (Line 9).

2. In the irreducible case, ensuring that finitely many well-chosen orbits under the action of the monodromy are n-dimensional is sufficient to prove that no non-trivial subspace is invariant. In addition, optimistic zero-tests can do no worse than underestimate the rank in Gaussian reduction, so that if a space is numerically found to be invariant and the full $V(L)$, it is rigorously so.

We use the Sage function `nearby_rational` for guessing rational numbers (and we plan to use the Sage function `algdep` for guessing algebraic numbers). We call the `minimal_approximant_basis` method of the Sage polynomial matrices for the computation of Hermite–Padé approximants.

Computing an invariant subspace is a significant subtask: we implemented a Las Vegas algorithm (adapted from the algorithm presented in [vdH07]) in which the random source is used to avoid the computation of a basis of the algebra generated by $\mathcal{M}$. The function `InvSub` takes a list of matrices with interval coefficients and returns either an approximate non-trivial invariant subspace or the symbol `none` to rigorously indicate that no non-trivial invariant subspace exists. The use of the `ComplexOptimisticField` structure is sufficient to guarantee the correctness of the output `none` in the second case.

To avoid unnecessarily precise computation and the induced cost, precision increases between loops have to be handled with care. In the current implementation, at each failure caused by insufficient precision, we double the working precision and possibly the truncation order, depending on the nature of the error.

We compared our implementation with van Hoeij's implementation (command `DFactor` of the library `DEtools` in the Maple system). The next table shows the results for some operators coming from the theory of random walks, available at `http://koutschan.de/data/fcc1/`. All the tests were done with the same laptop and times (minima over 5 loops) are given in seconds. We observed that our implementation of van der Hoeven's algorithm is faster than van Hoeij's implementation for some instances, in particular for establishing the irreducibility of fcc5 and fcc6. Other results can be found in the github repository of our package (commit: c5f48f34). Monodromy computation represents between 71 and 99 percents of the total time.

operators	fcc3	fcc4	fcc5	fcc6	fcc3^2	fcc4×fcc3	fcc3×fcc4	fcc4^2
orders	3	4	6	8	6	7	7	8
`diffop_factorization`	0.148	1.32	12.9	432.	3.23	31.5	24.8	108.
`DEtools`	0.182	0.630	61.9	$> 10^4$	0.976	1.88	4.59	122.

Consider the operator $S := (z^3-3z^2)\partial^2+(4z^7+z^2+3z-9)\partial+20z^6+12z^5$. Our code takes 3.34 seconds to find the factorization $S = (z^2\partial+3)((z-3)\partial+4z^5)$ while the function `DEtools[DFactor]` does not finish after 9 hours. We suspect that the resolution of a linear system of size 972×972 is the origin of this failure (972 corresponds to the difference between the two exponents at the singular point 3). The symbolic-numeric approach avoids this system. Note that the operator S is not Fuchsian. Our implementation tries anyway to factor it with monodromy matrices and returns a certified right factor if successful. Nevertheless, termination is not guaranteed for general non-Fuchsian operators. In this particular case, both factors have polynomial coefficients. One can thus find this factorization with a factoring algorithm in the so-called first Weyl algebra $\mathbb{K}[z]\langle\partial\rangle$. This task translates into a finite number of polynomial systems (the total degree of a right factor is bounded by L). An implementation of this method is available in the Singular system from the command `facWeyl` of the ncfactor library [LH18]. The `facWeyl` function finds this factorization in 11.4 seconds. Our tests shown that this function is (significantly) slower than our code. Moreover, it is possible that an operator is irreducible in $\mathbb{K}[z]\langle\partial\rangle$ while admitting a factorization with rational coefficients, such as $\partial^2 + (-z^2+2z)\partial + z - 2 = \left(\frac{1}{z}\partial - z + 2\right)(z\partial - 1)$ for example.

4 Conclusion and outlook

First results confirm that the symbolic-numeric approach for factoring linear differential operators can compete with a purely algebraic approach. Other related questions, such as the computation of algebraic solutions, could also benefit from the power of the symbolic-numeric approach.

The development of this algorithm is still a work in progress. We are working to reduce the cost of monodromy compution (trying to guess a factor with only a few of monodromy matrices could be a solution). We notice that the particular solution computed from monodromy is not always the simplest

one (because it involves coefficients with large degree in the corresponding right factor) and it is perhaps a way to avoid that. Clarifying the complexity of Algorithm 1, at least in function of the sufficient precision, would be interesting. We also plan to extend the implementation to the non-Fuchsian case (Theorem 2 admits a generalization for any $L \in \mathbb{K}(z)\langle\partial\rangle$ adding Stokes matrices to monodromy matrices) as also described in [vdH07].

Acknowledgements. This work is done as part of my doctoral research under the joint supervision of Frédéric Chyzak and Marc Mezzarobba, who I thank for all their advice, in particular the latter for helping me with the implementation of the `ComplexOptimisticField` structure. I am also grateful to Bruno Salvy for proposing the operator S above and to the reviewers for their helpful comments.

References

[Bek94] Emanuel Beke. Die Irreducibilität der homogenen linearen Differentialgleichungen. *Mathematische Annalen*, 45(2):278–294, 1894.

[Bro94] Manuel Bronstein. An improved algorithm for factoring linear ordinary differential operators. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation*, pages 336–340, 1994.

[BRS19] Alin Bostan, Tanguy Rivoal, and Bruno Salvy. Explicit degree bounds for right factors of linear differential operators. *Bulletin of the London Mathematical Society*, 2019.

[CvH04] Thomas Cluzeau and Mark van Hoeij. A modular algorithm for computing the exponential solutions of a linear differential operator. *Journal of Symbolic Computation*, 38(3):1043–1076, 2004.

[Gri90] Dmitrii Yur'evich Grigor'ev. Complexity of factoring and calculating the GCD of linear ordinary differential operators. *Journal of Symbolic Computation*, 10(1):7–37, 1990.

[JKM13] Fredrik Johansson, Manuel Kauers, and Marc Mezzarobba. Finding hyperexponential solutions of linear ODEs by numerical evaluation. In *International Symposium on Symbolic and Algebraic Computation*, 2013.

[LH18] Viktor Levandovskyy and Albert Heinle. A factorization algorithm for G-algebras and its applications. *Journal of Symbolic Computation*, 85:188–205, 2018. 41th International Symposium on Symbolic and Algebraic Computation (ISSAC'16).

[Mez16] Marc Mezzarobba. Rigorous multiple-precision evaluation of D-finite functions in SageMath. Technical Report 1607.01967, arXiv, 2016. Extended abstract of a talk at the 5th International Congress on Mathematical Software.

[Sal19] Bruno Salvy. Linear differential equations as a data structure. *Foundations of Computational Mathematics*, 19(5):1071–1112, 2019.

[Sch97] Ludwig Schlesinger. *Handbuch der Theorie der linearen Differentialgleichungen*, volume 2. B. G. Teubner, 1897.

[Sch89] Fritz Schwarz. A factorization algorithm for linear ordinary differential equations. In *Proceedings of the ACM-SIGSAM 1989 International Symposium on Symbolic and Algebraic Computation*, pages 17–25, 1989.

[Sin96] Michael F. Singer. Testing reducibility of linear differential operators: a group theoretic perspective. *Applicable Algebra in Engineering, Communication and Computing*, 7(2):77–104, 1996.

[Tsa94] Sergey Petrovich Tsarev. Problems that appear during factorization of ordinary linear differential operators. *Programming and Computer Software*, 20(1), 1994.

[vdH07] Joris van der Hoeven. Around the numeric-symbolic computation of differential Galois groups. *Journal of Symbolic Computation*, pages 236–264, 2007.

[vdPS03] Marius van der Put and Michael F. Singer. *Galois Theory of Linear Differential Equations*. Springer-Verlag Berlin Heidelberg, 2003.

[vH97] Mark van Hoeij. Factorization of differential operators with rational functions coefficients. *Journal of Symbolic Computation*, 24(5):537–561, 1997.