

Submitted by
Antonio Jiménez Pastor

Submitted at
Doktoratskolleg
"Computational
Mathematics"

Supervisor and
First Examiner
Assoc. Prof. Dr.
Veronika Pillwein

Second Examiner
Researcher Frédéric
Chyzak

November 2020

A computable extension for holonomic functions: DD-finite functions



Doctoral Thesis
to obtain the academic degree of
Doktor der technischen Wissenschaften
in the Doctoral Program
Technische Wissenschaften

Abstract

D-finite or holonomic functions are solutions to linear differential equations with polynomial coefficients. They have been in the focus of attention in symbolic computation during the last decades due to their computability properties. They can be represented with a finite amount of data (coefficients of the equation and initial values) and several algorithms have been developed to work with this representation.

In the current thesis we extend the definition of D-finite functions to a more general setting and we focus on extending some of the existing algorithms to this wider class of functions: those that satisfy a linear differential equation with D-finite coefficients. We call them DD-finite functions.

Further we show that, by the same principle, those algorithms can be extended to the wider setting of solutions to linear differential equations with coefficients in some computable differential ring. This allows us to iterate the construction, yielding the set of D^n -finite function. We also study their properties and their relation with the differentially algebraic functions (which are an even larger class of functions).

All algorithms developed and presented in this thesis have been implemented in Sage (an Open Source system for computer algebra based on Python). We include in the thesis a technical user guide for our implementation together with several examples on how to use the package.

Zusammenfassung

D-finite oder holonome Funktionen sind Lösungen von linearen Differentialgleichungen mit polynomiellen Koeffizienten. Da für diese Funktionen viele Eigenschaften algorithmisch berechnet werden können, standen sie in den letzten Jahrzehnten im Mittelpunkt der Aufmerksamkeit im Gebiet des Symbolischen Rechnens. Sie können mit einer endlichen Datenmenge (die Koeffizienten der Gleichung sowie die Anfangswerte) dargestellt werden und mehrere Algorithmen, die mit dieser Darstellung arbeiten, wurden entwickelt.

In dieser Arbeit verallgemeinern wir die Definition von D-finiten Funktionen und erweitern einige der existierenden Algorithmen auf diese größere Klasse von Funktionen: Funktionen, die eine lineare Differentialgleichung mit D-finiten Koeffizienten erfüllen. Wir nennen diese Funktionen DD-finit.

Außerdem zeigen wir, dass diese Algorithmen in derselben Weise in das allgemeinere Setting von Lösungen von linearen Differentialgleichungen mit Koeffizienten in einem berechenbaren Differentialring erweitert werden können. Das erlaubt uns, diese Konstruktion zu iterieren und die Menge der D^n -finiten Funktionen zu erhalten. Wir untersuchen auch ihre Eigenschaften und ihre Beziehung zu Lösungen von Differential-Algebraischen Gleichungen (die eine noch größere Klasse von Funktionen ist).

Alle Algorithmen, die wir entwickelt haben und in dieser Arbeit präsentieren, wurden in Sage (einem auf Python basierenden Open Source Computeralgebrasystem) implementiert. Die vorliegende Arbeit umfasst ein Benutzerhandbuch unserer Implementierung sowie mehrere Beispiele, wie dieses Programmpaket verwendet werden kann.

Acknowledgments

Fulfilling a thesis is usually not the work of one individual, but the collaboration and enthusiasm between experts and a student to explore new areas of knowledge. That is why I would like to thank my advisor, Veronika Pillwein, for providing the opportunity to embark in this journey that culminates with this thesis. She suggested the topic of the thesis to me and kept me on the right track throughout all these years.

I would also like to thank all the other experts that have actively spent time with me. In particular, I thank Frédéric Chyzak, who facilitated my two visits to INRIA Saclay, reserved much of his time for our discussions and also reviewed this thesis. I also thank Moulay Barkatou for allowing me to join his team in the University of Limoges on two different occasions.

I would like to express my gratitude also to all the colleagues in Linz. In particular, I thank the RISC institute, which provided me with equipment and material to perform my work in the best environment. I thank the Algorithmic Combinatorics group and its two leaders Peter Paule and Carsten Schneider. The energy and discussions in that seminar will be impossible to forget. I also want to thank Jakob Ablinger, who did not only share office with me during most part of my time at RISC, but also helped me to feel integrated in Linz.

I also want to specially mention the Doctoral Program Computational Mathematics (or DK). The people and the seminar brought to light a wide variety of topics in mathematics, not only on the field of computer algebra and symbolic computation, that I hope I can use in the future. My full recognition to Gabriela Danter, our beloved secretary, who helped me with all the bureaucracy and saved me lots of time and effort. I also thank the Austrian Science Fund FWF, in particular the grant W1214-N15, project DK 15, for the financial support.

I would like to show my gratitude to those that helped me with the implementation of the package detailed in the thesis: Ralf Hemmecke, who helped me with Sage and git, Manuel Kauers, whose package I used intensively, and Marc Mezzarobba, who always had time to discuss about the proper way of coding in Sage.

Last but not least, a special thanks to my family and friends for the unconditional emotional support. Even if there is so much distance between us, I felt their strength that helped me moving forward and, successfully, complete the thesis.

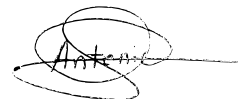
Statutory Declaration

I hereby declare that the thesis submitted is my own unaided work, that I have not used other than the sources indicated, and that all direct and indirect sources are acknowledged as references.

This printed thesis is identical with the electronic version submitted.

Linz, November 2020

Antonio Jiménez Pastor

A handwritten signature in black ink, appearing to read 'Antonio', with a large, loopy flourish extending to the right.

Contents

1	Introduction	1
2	Preliminaries	5
2.1	Formal power series	6
2.2	D-finite functions	9
2.3	P-recursive sequences	11
2.4	Beyond holonomicity	14
2.5	Differential linear operators	15
3	Differentially definable functions	17
3.1	Differentially definable functions	17
3.2	Closure properties	21
3.3	DD-finite power series solutions	25
4	Implementation of closure properties	35
4.1	Closure properties by direct proof	36
4.1.1	Annihilator for the derivative	36
4.1.2	Annihilator for the integral	37
4.2	Closure properties by linear algebra	38
4.2.1	Description of the ansatz method	38
4.2.2	Specifications for the sum	42
4.2.3	Specifications for the product	45
4.2.4	Specifications for the algebraic inclusion	49
4.3	Performance considerations	53
5	Structural results	61
5.1	D^n -finite functions	61
5.2	Increasing structure of the D^n -finite chain	65
5.3	Relation with D-algebraic functions	68
5.3.1	The Riccati differential equation	73
5.3.2	Separable first order equations	78

5.3.3	A minimal counterexample	82
6	Implementation of structural results	85
6.1	Implementation of composition	86
6.2	Implementation of algebraic substitution	91
6.3	Implementation of Theorem 5.17	96
7	The package <code>dd_functions</code>	103
7.1	Sage: an Open Source system	104
7.2	Installing the package	105
7.3	The main user interface	106
7.3.1	Parent class: <code>DDRing</code>	107
7.3.2	Element class: <code>DDFunction</code>	109
7.4	Computing closure properties	112
7.4.1	Arithmetic and differential closure properties	112
7.4.2	Composition of <code>DDFunction</code>	114
7.5	Built-in examples	115
7.5.1	Exponential and trigonometric functions	115
7.5.2	Special functions: parametric functions	117
7.5.3	Special functions: non-parametric functions	120
7.6	Relation to D-algebraic functions	121
8	Ongoing and future work	125
8.1	Deeper study of the D^∞ domain	125
8.2	Characterize composition	129
8.3	Combinatorial examples	130
8.4	Walks in the quarter plane	136
8.5	Numerical approximation	140
A	Linear algebra	151
A.1	Fraction-free system solving	151
A.1.1	Diagonalizing matrices	152
A.1.2	Computing a non-trivial element	155
A.2	Differential linear algebra	157
B	Differential Galois theory	165
B.1	An introduction to differential algebra	165
B.1.1	Differential rings and fields	165
B.1.2	Differential extensions	168
B.1.3	Monomial extensions	172
B.1.4	Primitive and hyperexponential extensions	174

B.2	Linear differential equations	177
B.3	Picard-Vessiot rings	179
B.3.1	Picard-Vessiot closure	184
B.4	Galois groups of PV-extensions	186

Chapter 1

Introduction

State of the art

In the last decades, the symbolic treatment of functions has been a wide area of research. Being able to manipulate functions symbolically (i.e., without approximation errors) is a very useful technique that allows to check identities and to perform computations in real life problems without accumulating these approximation errors.

The theoretical manipulation of these functions is important, but nowadays, being able to perform this manipulation algorithmically is essential. If those algorithms exist, then we can implement them into our computer systems and, hence, we are able to manipulate functions automatically.

We do not only need the existence of the algorithms, but also the possibility of representing functions with a finite amount of data. This finite representation of an object must contain all the information of the function, i.e., it is an exact and complete representation of the function.

One class of functions that satisfies both conditions, i.e., a class of functions that can be finitely represented and can be algorithmically manipulated, are the so called *holonomic* or D-finite functions [44, 66, 67]. These functions are formal power series that satisfy a linear differential equation with polynomial coefficients. These equations together with some initial conditions describe these functions exactly. Storing the coefficients (polynomials) and the initial conditions (numbers) can be done with a finite amount of data. Moreover, in various computer algebra systems [42, 49, 65] there are algorithms developed to perform operations such as addition, multiplication, derivation, etc. over D-finite functions.

There are many examples of special functions that are D-finite [5, 63], such as the exponential and trigonometric functions, the Bessel functions, the Airy functions, all the families of orthogonal polynomials, hypergeometric functions,

etc. Also, the generating function of many sequences arising in combinatorics are D-finite, such as the generating functions for the Fibonacci numbers, the Catalan numbers, the harmonic numbers, etc.

The tools provided for D-finite functions allow the mathematical community to manipulate these functions automatically. This manipulation starts with the defining differential equations for the operands (which we said can be finitely represented) and computes a linear differential equation (again, with polynomial coefficients) for the resulting function. These algorithms can be used to prove identities for special functions or sequences in enumerative combinatorics [41, 62].

However, there are also many examples that do not fit into the D-finite framework. Simple functions such as the tangent $\tan(x)$, which is the quotient of two D-finite functions (namely, $\sin(x)$ and $\cos(x)$), are not D-finite. The double exponential function e^{e^x} , or the Mathieu differential functions (see [26, Chapter 28]) are not D-finite either. Recently, more examples were found [1, 3, 11] such that they satisfy a differential equations whose coefficients are not simply polynomials.

Being able to manipulate all these non D-finite objects will be useful to increase the representation capacity of our computers, allowing mathematicians from different fields to compute with these objects automatically and prove new identities.

Objectives of the thesis

The main goal of this thesis is to extend the algorithms for D-finite functions to a wider class of functions. We started from the fact that D-finite functions can be finitely represented and all the operations with D-finite functions are already implemented. What stops us from using these objects as our main building blocks?

The fact that a differential equation together with some initial conditions represent exactly one formal power series solutions is not particular for polynomial coefficients. In fact, that applies for any type of coefficient.

In the case of D-finite functions, the coefficients were polynomials and they can be finitely represented. That implied that D-finite functions can be finitely generated, meaning that we can use them as coefficients of linear differential equations and obtain, again, a class of functions that have a finite representation. This is the motivation for defining the class of DD-finite functions: formal power series that satisfy a linear differential equation whose coefficients are D-finite [38]. The algorithms for manipulating these DD-finite objects are developed during this thesis as well [37].

Once we prove the main closure properties for DD-finite functions (such as addition, multiplication, derivation and integration), we generalize this construction for any generic type of coefficient. We establish the main conditions required for the coefficients in order to apply the same algorithms for manipulating these

formal power series. In particular, if we iterate this construction, we can build the D^n -finite functions. In this thesis we also study the main properties of this wider class of formal power series [39].

Since having a theoretical framework is not enough for us, another goal of the thesis is to provide new software implementing these classes of formal power series and the algorithms to manipulate them [35, 36].

Outline

In Chapter 2 we introduce the main concepts required for understanding the thesis. Namely, we describe the ring of formal power series and study the basic properties of D -finite functions. We also provide a brief description about linear differential operators and their algebraic structure as a ring.

Chapter 3 contains the main theoretical results of the thesis. Here we define the class of *differentially definable* functions and show how D -finite and DD -finite functions are particular cases of this class. We then proceed to prove that these functions are closed under several operations.

In Chapter 4 we describe algorithmically how the closure properties for differentially definable functions can be performed. The idea is to use an *ansatz* method to transform the problem of finding a linear differential equation into a problem of linear algebra.

Once all closure properties can be automatized, we extend the theory of differentially definable functions in Chapter 5 defining the class of D^n -finite functions. These functions are formal power series that satisfy a linear differential equation whose coefficients are D^{n-1} -finite. This iterative construction allows us to prove certain relations with the composition of two D^n -finite functions. In addition, this iterative chain of rings always include more functions.

In Chapter 6 we describe the algorithmic aspects of the operations described in the previous Chapter. Here we adapt the method used in Chapter 4 to fit this new context. All the structures and algorithms developed in this thesis are compiled in a software package for Sage, called `dd_functions`. We provide a user manual of this package in Chapter 7.

Finally, in Chapter 8, we provide several open questions about differentially definable functions and connect our thesis with other research areas where our techniques or ideas may be useful to extend the current results.

Chapter 2

Preliminaries

In this Chapter we present the basic definitions and objects that are studied in further chapters of this thesis. These definitions includes the set of formal power series, D-finite functions and linear differential operators.

Formal power series, closely related with analytic functions and sequences, are the main object of study on this thesis and we present here several results that will be assumed in later chapters of the thesis.

A subclass of the formal power series are the D-finite functions. These functions satisfy linear differential equations with polynomial coefficients and, as shown in many previous works [32, 44, 67, 74], this allows us to finitely represent this object into the computer. This lead to several implementations in various computer algebra system [49, 54, 41] of these objects.

On the other hand, considering these power series as sequences, we have the equivalent class P-recursive sequences. These sequences satisfy linear differential recurrences with polynomial coefficients. It was shown [44, 67] the these two classes were equivalent and this link helped proving extra properties for the functions.

One common tool used for studying these two classes of functions are linear operators. Namely, linear differential operators for D-finite functions and liner difference operators for P-recursive sequences. These linear operators behave as polynomial rings with the particularity of being a non-commutative ring.

We introduce now all these concepts and present the basic results that are used later in the thesis. Most of the content in this Chapter can be found in the literature [44, Chapter 2] and [67]. We refer to those sources for the proofs that are missing in this Chapter.

2.1 Formal power series

Throughout this chapter and this thesis, let \mathbb{K} be a field of characteristic zero (i.e., $\mathbb{Q} \subset \mathbb{K}$) and $\mathbb{K}[x]$ the ring of polynomials with variable x . We denote by $\mathbb{K}[[x]]$ the ring of *formal power series with coefficients in \mathbb{K}* , whose elements are of the form

$$f(x) = \sum_{n \geq 0} f_n x^n.$$

For the coefficient functional, we use the notation $f_n = [x^n]f(x)$.

The two operations that we consider over this set $\mathbb{K}[[x]]$ are:

- Termwise addition:

$$f(x) + g(x) = \sum_{n \geq 0} (f_n + g_n) x^n.$$

- Cauchy product:

$$f(x)g(x) = \sum_{n \geq 0} \left(\sum_{k=0}^n f_{n-k} g_k \right) x^n.$$

Then $(\mathbb{K}[[x]], +, \cdot)$ is an *integral domain* (i.e., $f(x)g(x) = 0$ implies either $f(x) = 0$ or $g(x) = 0$) [44, Theorem 2.2].

Definition 2.1. Let $f(x) \in \mathbb{K}[[x]]$. We call *the order of $f(x)$* the minimal $n \in \mathbb{N}$ such that $f_n \neq 0$, i.e.,

$$\text{ord}(f) = \min_{n \geq 0} \{n : f_n \neq 0\}.$$

In particular, $\text{ord}(0) = \infty$.

The following lemma shows that $(\mathbb{K}[[x]], \text{ord})$ is a valuation ring:

Lemma 2.2. *Let $f(x), g(x) \in \mathbb{K}[[x]]$. Then:*

- $\text{ord}(f + g) \geq \min\{\text{ord}(f), \text{ord}(g)\}$, with equality when $\text{ord}(f) \neq \text{ord}(g)$.
- $\text{ord}(fg) = \text{ord}(f) + \text{ord}(g)$
- If $f(x) \neq 0$, there is a unique $\tilde{f}(x) \in \mathbb{K}[[x]]$ such that $\text{ord}(\tilde{f}) = 0$ and

$$f(x) = x^{\text{ord}(f)} \tilde{f}(x).$$

Proof. Let $n = \text{ord}(f)$ and $m = \text{ord}(g)$. Without loss of generality, we assume that $n \leq m$.

By definition, $[x^k]f(x) = [x^k]g(x) = 0$ for all $k < n$. Hence $[x^k](f + g) = 0$ for all $k < n$ yielding $\text{ord}(f + g) \geq n$. On the other hand, if $n < m$ then $[x^n]g(x) = 0$. Hence, $[x^n](f + g) = [x^n]f(x) \neq 0$, and thus $\text{ord}(f + g) = n$.

For the Cauchy product, we have

$$[x^r](f(x)g(x)) = \sum_{k=0}^r f_{r-k}g_k.$$

Obviously, for $s < n + m$, $[x^s](f(x)g(x)) = 0$. On the other hand:

$$[x^{n+m}](f(x)g(x)) = \sum_{k=0}^{n+m} f_{n+m-k}g_k = f_n g_m \neq 0.$$

Finally, consider $\tilde{f}(x) = \sum_{n \geq 0} f_{n+\text{ord}(f)}x^n$. It is clear that $f(x) = x^{\text{ord}(f)}\tilde{f}(x)$ and, since $\mathbb{K}[[x]]$ is an integral domain, $\tilde{f}(x)$ is unique. \square

This concept of order allows us to define precisely a notion of convergence on the ring of formal power series:

Definition 2.3. Let $\{f_n(x)\}_n \subset \mathbb{K}[[x]]$ be a sequence of formal power series. We say that it converges to $f(x) \in \mathbb{K}[[x]]$ if the sequence of natural numbers $(\text{ord}(f_n - f))_n$ diverges (i.e., $\text{ord}(f_n - f) \rightarrow \infty$).

Lemma 2.4. Let $f(x) \in \mathbb{K}[[x]]$. Then there is a sequence of polynomials $\{p_m(x) \in \mathbb{K}[x]\}_m$ such that $p_m(x) \rightarrow f(x)$.

Proof. Let $(f_n)_n$ be the sequence in \mathbb{K} such that $f(x) = \sum_{n \geq 0} f_n x^n$ and consider the polynomials $p_m(x) = \sum_{n=0}^m f_n x^n$. Obviously,

$$\text{ord}(p_m(x) - f(x)) = \text{ord}\left(\sum_{n \geq m+1} f_n x^n\right) > m.$$

Hence, $\text{ord}(p_m(x) - f(x)) \rightarrow \infty$. \square

With this, the composition of formal power series can be defined as follows:

Definition 2.5. Let $f(x), g(x) \in \mathbb{K}[[x]]$. We define the composition of $f(x)$ with $g(x)$ (and denoted $f(g(x))$) to be the limit (if it exists) of the sequence:

$$h_n(x) = \sum_{k=0}^n f_k g(x)^k.$$

Lemma 2.6 ([44, Theorems 2.4 and 2.6]). *Let $f(x), g(x) \in \mathbb{K}[[x]]$.*

- *(Multiplicative inverse) There is $h(x) \in \mathbb{K}[[x]]$ such that $f(x)h(x) = 1$ if and only if $\text{ord}(f) = 0$.*
- *(Existence of composition) If $\text{ord}(g) > 0$, then $f(g(x))$ exists.*

In the polynomial ring $\mathbb{K}[x]$ we have the standard derivation ∂_x w.r.t. x such that $\partial_x(\alpha) = 0$ for all $\alpha \in \mathbb{K}$ and $\partial_x(x) = 1$. In the ring of formal power series we can extend this derivation. Similarly, we can extend the integration of polynomials to the whole ring of formal power series.

Definition 2.7. Let $f(x) \in \mathbb{K}[[x]]$. We define the derivation ∂_x of $f(x)$ with the formula

$$\partial_x \left(\sum_{n \geq 0} f_n x^n \right) = \sum_{n \geq 0} f_n \partial_x(x^n) = \sum_{n \geq 0} n f_n x^{n-1}.$$

We define the formal integral operator \int_x of $f(x)$ with the formula

$$\int_x \sum_{n \geq 0} f_n x^n = \sum_{n \geq 0} \frac{f_n}{n+1} x^{n+1}.$$

These two operators satisfy several properties:

Lemma 2.8 ([44, Theorem 2.3]). *Let ∂_x and \int_x defined as before. Then:*

- *($\mathbb{K}[[x]], \partial_x$) is a differential ring (i.e., $\partial_x(f+g) = \partial_x(f) + \partial_x(g)$ and $\partial_x(fg) = \partial_x(f)g + \partial_x(g)f$).*
- *(Fundamental Theorem of Calculus I) For all $f(x) \in \mathbb{K}[[x]]$,*

$$\partial_x \left(\int_x f(x) \right) = f(x).$$

- *(Fundamental Theorem of Calculus II) For all $f(x) \in \mathbb{K}[[x]]$,*

$$\int_x \partial_x(f(x)) = f(x) - f(0).$$

- *(Taylor's formula) For all $f(x) \in \mathbb{K}[[x]]$,*

$$[x^n]f(x) = \frac{1}{n!} \partial_x^n(f(x))(0).$$

During this thesis, when working with a formal power series $f(x)$, we use the notation $f'(x)$, $f''(x)$, $f^{(3)}(x)$, etc. for representing the derivatives $\partial_x(f(x))$, $\partial_x^2(f(x))$, etc.

2.2 D-finite functions

Formal power series, as we have defined in the previous Section, are very interesting objects for several fields of mathematics. In analysis, the Taylor expansion of a function is just a way to rewrite the function as a formal power series. However, a formal power series is defined by the sequence of its coefficients and those are usually an infinite amount of elements.

Then, in order to implement a subclass of formal power series into the computer, we need extra information. For example, with *rational* power series we can represent them as the quotient of two polynomials (which have a finite amount of data). We could also consider *algebraic* power series, those that satisfy a polynomial equation. In this case only the minimal polynomial is needed to represent the formal power series. But there is a wider class of formal power series that also allows us to represent them with a finite amount of data and also to implement several operations to work with them. This class are the D-finite functions (see [44, Chapter 7] and [67]).

Definition 2.9. Let \mathbb{K} be a field of characteristic zero. We say that $f(x) \in \mathbb{K}[[x]]$ is *D-finite* or *holonomic* if there is $d \in \mathbb{N}$ and $p_0(x), \dots, p_d(x) \in \mathbb{K}[x]$ with $p_d(x) \neq 0$ such that:

$$p_d(x)f^{(d)}(x) + \dots + p_1(x)f'(x) + p_0(x)f(x) = 0.$$

We say that *the order* of $f(x)$ is the minimal d such that the previous equation exists.

D-finite functions are those formal power series that satisfies a linear differential equation with polynomial coefficients. Some well known functions fit naturally in this definition:

Example 2.10 (Exponential function). Let $c \in \mathbb{K}$. Then $\exp(cx) = e^{cx}$ is D-finite, since it satisfies the differential equation

$$(\exp(cx))' - c(\exp(cx)) = 0.$$

Example 2.11 (Trigonometric functions). The trigonometric functions $\sin(x)$ and $\cos(x)$ are D-finite since they satisfy $f''(x) + f(x) = 0$.

In order to be able to find this linear differential equations, the following characterization theorem provide us a great tool:

Theorem 2.12 ([67, Theorem 1.2]). *Let \mathbb{K} be a field of characteristic zero and $f(x) \in \mathbb{K}[[x]]$. The following are equivalent:*

1. $f(x)$ is D-finite with order d .

2. Inhomogeneous: there are $p_0(x), \dots, p_k(x)$ with $p_k(x) \neq 0$ and $g(x)$ a D -finite function of order $d - k$ such that

$$p_k(x)f^{(k)}(x) + \dots p_0(x)f(x) = g(x).$$

3. The vector space $\langle f(x), f'(x), f''(x), \dots \rangle_{\mathbb{K}(x)}$ is d .

For the proof of this theorem we refer to the proof by Stanley [67].

This characterization, and more precisely, the finite dimension of the vector space generated by a function and its derivatives is the key to prove the following closure properties:

Theorem 2.13 ([44, Theorem 7.2]). *Let \mathbb{K} be a field of characteristic zero, $f(x) \in \mathbb{K}[[x]]$ D -finite of order r and $g(x)$ D -finite of order s . Then:*

- (i) *For any $\alpha, \beta \in \mathbb{K}$, $\alpha f(x) + \beta g(x)$ is D -finite with order at most $r + s$.*
- (ii) *$f(x)g(x)$ is D -finite with order at most rs .*
- (iii) *$f'(x)$ is D -finite with order at most r .*
- (iv) *$\int f$ is D -finite with order at most $r + 1$.*

And, moreover, we can prove now easily the inclusion of other classes of formal power series into the D -finite class:

Theorem 2.14 ([67, Theorem 2.1 and 2.7]). *Let \mathbb{K} be a field of characteristic zero, $f(x) \in \mathbb{K}[[x]]$ D -finite of order r and $a(x) \in \mathbb{K}[[x]]$ algebraic with minimal polynomial $m(x, y) \in \mathbb{K}[x, y]$ of degree p (i.e., $m(x, a(x)) = 0$). Then:*

- *$a(x)$ is D -finite with order at most p .*
- *If $a(0) = 0$, then $f(a(x))$ is D -finite with order at most rp .*

Example 2.15. The following examples are all D -finite:

$$e^{\sqrt{x^2+1}-1}, \quad \frac{\sin(x)}{1-x}, \quad \log(x+1), \quad \frac{\sqrt[7]{x^6+x-1} - \cos\left(x^3-1+\sqrt{x+1}\right)}{x^2-x+1}$$

2.3 P-recursive sequences

At this point we have all the tools to go to Chapter 3. But before extending the class of D-finite functions, we present an equivalent representation for this power series. Now we focus on the sequences associated with particular formal power series. We study in this Section how previous definitions fits into the framework of sequences. We finish this section presenting the exact relation with the D-finite functions.

Let \mathbb{K} be a field of characteristic zero, the set of sequences over \mathbb{K} is the set $\mathbb{K}^{\mathbb{N}}$, namely, the functions $\varphi : \mathbb{N} \rightarrow \mathbb{K}$. We denote by $(a_n)_n$ the sequence where $a_n = \varphi(n)$. There is a trivial relation between the sequences over \mathbb{K} and the ring of formal power series via the map:

$$(a_n)_n \leftrightarrow \sum_{n \geq 0} a_n x^n.$$

However, the Cauchy product in the ring of formal power series seems unnatural for this set. The natural option would be the so called Hadamard product (i.e., the termwise product):

$$(a_n)_n * (b_n)_n = (a_n b_n)_n.$$

This product, together with the termwise addition, makes the set $\mathbb{K}^{\mathbb{N}}$ a ring but not an integral domain.

For example, consider the sequences $(a_n)_n$ with $a_{2n} = 0$ and $a_{2n+1} = 1$, and $(b_n)_n$ with $b_{2n} = 1$ and $b_{2n+1} = 0$. Clearly this two sequences are not the zero sequence $(0)_n$. On the other hand:

$$(a_n)_n * (b_n)_n = (0 \cdot 1, 1 \cdot 0, 0 \cdot 1, \dots) = (0, 0, 0, \dots) = (0)_n.$$

Nevertheless, we can still define an analogous class of sequences to the D-finite functions:

Definition 2.16. Let \mathbb{K} be a field of characteristic zero. We say that a sequence $(a_n)_n$ is a *P-recursive* (or *holonomic*) if there is $d \in \mathbb{N}$ and polynomials $p_0(x), \dots, p_d(x) \in \mathbb{K}[x]$ with $p_d(x) \neq 0$ such that, for all $n \in \mathbb{N}$:

$$p_d(n)a_{n+d} + \dots + p_1(n)a_{n+1} + p_0(n)a_n = 0.$$

Similarly to the D-finite functions, P-recursive sequences together with the termwise addition and the Hadamard product, form a subring of the sequences:

Theorem 2.17 ([44, Theorem 7.2]). *Let $(a_n)_n$ and $(b_n)_n$ be two P-recursive sequences of order d_1 and d_2 respectively. Then*

- (i) $(a_n)_n + (b_n)_n = (a_n + b_n)_n$ is P-recursive with order at most $d_1 + d_2$.

(ii) $(a_n)_n * (b_n)_n = (a_n b_n)_n$ is P -recursive with order at most $d_1 d_2$.

When studying P -recursive sequences, it is useful to remark that the polynomial ring $\mathbb{K}[x]$ have different basis. We say that $(p_n(x))_n \in \mathbb{K}[n]^\mathbb{N}$ is a polynomial basis if $p_n(x) \in \mathbb{K}[x]$ has degree n for all $n \in \mathbb{N}$.

Usually, when working with polynomials, we use the monomial basis. This basis is built with the powers of the variable of the polynomial ring $(x^n)_n$. However, in the study of P -recursive sequences, it is very useful to consider two other different basis:

- *Falling factorial basis*: in any ring we can define the falling factorial operations by

$$(a)^{\underline{n}} = a(a-1) \cdots (a-n+1).$$

With this notation, we can consider the sequence of polynomials $((x)^{\underline{n}})_n$. Obviously, this sequence is a polynomial basis that satisfies:

$$(x+1)^{\underline{n}} = (x+1)(x)^{\underline{n-1}}.$$

- *Raising factorial basis*: in a similar way, we can in any ring define the raising factorial by

$$(a)_n = a(a+1) \cdots (a+n-1).$$

We consider the polynomial basis $((x)_n)_n$.

These three polynomial basis are related in a very interesting way, by the Stirling numbers of first and second kind [44]:

$$(x)^{\underline{n}} = \sum_{k=0}^n S_1(n, k) x^k, \quad x^n = \sum_{k=0}^n S_2(n, k) (x)^{\underline{k}},$$

$$(x)^{\underline{n}} = \sum_{k=0}^n \binom{n}{k} (n-1)_{n-k} (x)_k, \quad (x)_n = \sum_{k=1}^n \binom{n-1}{k-1} \frac{n!}{k!} (x)^{\underline{k}}.$$

Theorem 2.18 ([67, Theorem 1.4] and [44, Theorem 7.1]). *Let \mathbb{K} be a field of characteristic zero and $(a_n)_n$ a sequence over \mathbb{K} . Then $(a_n)_n$ is P -recursive if and only if the associated formal power series $f(x) = \sum_{n \geq 0} a_n x^n$ is D -finite.*

Proof. Let us assume first that $(a_n)_n$ is a P -recursive sequence with an equation of the form:

$$p_d(n)a_{n+d} + \cdots + p_1(n)a_{n+1} + p_0(n)a_n = 0. \quad (2.19)$$

Each $p_i(x)$ can be written uniquely as a linear combination of $(x+i)^{\underline{j}}$ for $j = 0$ up to $\deg(p_i)$. It is easy to see that

$$\sum_{n \geq 0} (n+i)^{\underline{j}} a_{n+i} x^n = R_{i,j}(x) + x^{j-i} f^{(i)}(x),$$

for some polynomial $R_{i,j}(x)$ of degree i . In fact, we can write down explicitly this polynomial as the i first terms of that sum:

$$R_{i,j}(x) = \sum_{n=0}^{i-1} (n+i)^j a_{n+i} x^n$$

Then multiplying equation (2.19) by x^n and summing for $n \geq 0$, we obtain a linear combination of derivatives of $f(x)$ and polynomials $R_{i,j}(x)$ that provides exactly a linear differential equation showing that $f(x)$ is D-finite.

Now, assume that $f(x)$ is D-finite with a differential equation of the type:

$$p_d(x)f^{(d)}(x) + \dots + p_1(x)f'(x) + p_0(x)f(x) = 0. \quad (2.20)$$

In this case it is easy to show that for any i and j :

$$x^i f^{(j)}(x) = \sum_{n \geq i} (n-i+j)^j a_{n+j-i} x^n,$$

then applying this identity in (2.20), we obtain an equation of the form:

$$\sum_{n \geq 0} \left(\sum_{i=0}^d \sum_{j=0}^r p_{j,i} (n-i+j)^j a_{n+j-i} \right) x^n = 0,$$

where r is the maximal degree of the polynomials $p_0(x), \dots, p_d(x)$. We conclude that the two inner sums have to be zero for all $n \in \mathbb{N}$, and substituting $n \mapsto n+d$, we find an equation

$$q_{d+r}(n)a_{n+d+r} + \dots + q_1(n)a_{n+1} + q_0(n)a_n = 0,$$

proving that $(a_n)_n$ is P-recursive. \square

This equivalence between D-finite functions and P-recursive sequences is truly important for relating the framework of D-finite functions, and hence, much of our work in this thesis, to the combinatorial work. However, as we will see in Chapter 5 this relation disappears once we extend slightly the class of D-finite functions.

Example 2.21 (Combinatorial sequences). The generating functions of the Fibonacci numbers ($F(x)$) and the Catalan numbers ($C(x)$) are both D-finite.

The sequences of Fibonacci numbers and Catalan numbers are P-recursive with recurrences

$$F_{n+2} - F_{n+1} - F_n = 0, \quad (n+2)C_{n+1} = 2(2n+1)C_n,$$

so Theorem 2.18 guarantees that both generating functions are D-finite.

Example 2.22 (Generalized hypergeometric functions). Consider $a_1, \dots, a_p \in \mathbb{K}$, $b_1, \dots, b_q \in \mathbb{K}$ and the corresponding sequence:

$$f_n = \frac{(a_1)_n \cdots (a_p)_n}{n! (b_1)_n \cdots (b_q)_n}.$$

It is easy to see that the sequence f_n is hypergeometric, i.e., the quotient of two consecutive elements is in $\mathbb{K}(n)$:

$$\frac{f_{n+1}}{f_n} = \frac{(a_1 + n) \cdots (a_p + n)}{n(b_1 + n) \cdots (b_q + n)},$$

which makes the sequence f_n P-recursive and then, its generating function (called *generalized hypergeometric function*) is D-finite:

$${}_pF_q \left(\begin{matrix} a_1, \dots, a_p \\ b_1, \dots, b_q \end{matrix} ; x \right) = \sum_{n \geq 0} \frac{(a_1)_n \cdots (a_p)_n}{n! (b_1)_n \cdots (b_q)_n} x^n$$

2.4 Beyond holonomicity

Note that in the previous sections we only presented examples of D-finite functions and P-recursive sequences. In this Section we study in this section the two examples that are not included in this class of formal power series. These two examples rely on previous results that are widely known:

Lemma 2.23. *The exponential function $e^x = \sum_{n \geq 0} \frac{x^n}{n!}$ is transcendental over $\mathbb{K}(x)$.*

Corollary 2.23.1. *The formal power series $\exp(\exp(x) - 1)$ is not D-finite.*

Proof. Assume that it was indeed D-finite. Since $(e^{e^x-1})' = e^x e^{e^x-1}$, it is easy to show that $(e^{e^x-1})^{(k)} = e^{e^x-1} p_k(e^x)$ for some polynomial $p_k(x) \in \mathbb{K}$. Putting this into the linear differential equation of e^{e^x-1} will lead to:

$$e^{e^x-1} P(x, e^x) = 0,$$

and since $e^{e^x-1} \neq 0$, we have that e^x is algebraic over $\mathbb{K}(x)$ which contradicts Lemma 2.23. Hence e^{e^x-1} is not D-finite. \square

Lemma 2.24. *Let $f(x)$ be analytic around 0 and satisfying a linear differential equation*

$$r_d(x) f^{(d)}(x) + \dots r_1(x) f'(x) + r_0(x) f(x) = 0,$$

where r_i are also analytic around 0. Then the poles of any analytic continuation of $f(x)$ must be a pole of $r_i(x)$ for some $i = 0, \dots, d-1$ or a zero of $r_d(x)$.

Corollary 2.24.1. *Let $f(x)$ be a D -finite function. Then $f(x)$ has a finite number of poles.*

Proof. Using Lemma 2.24, the poles of $f(x)$ should be on the pole set of the first coefficients or in the zeros of the leading coefficients. However, since the coefficients are polynomials, they have no poles.

Then the poles of $f(x)$ are between the zeros of the leading polynomial of the defining equation, which is always a finite set. \square

Corollary 2.24.2. *The function $\tan(x)$ is not D -finite.*

Proof. We know that $\tan(x) = \sin(x)/\cos(x)$, so it has a pole for every $x = \frac{(2k+1)\pi}{2}$. This contradicts Corollary 2.24.1, so $\tan(x)$ is not D -finite. \square

2.5 Differential linear operators

In the previous sections we considered linear differential and difference equations. In fact, the equations we considered were always homogeneous equations. In this Section we present the definitions and notations of linear differential operators. These definitions can be extended to the difference case and beyond (leading to the so called *Ore algebras* [22, 23]).

Definition 2.25. Let R be a \mathbb{K} -algebra (i.e., a ring that is also a \mathbb{K} -vector space). We say that $L : R \rightarrow R$ is a \mathbb{K} -linear operator if, for all $r, s \in R$ and $\alpha \in \mathbb{K}$:

$$L(r + s) = L(r) + L(s), \quad L(\alpha r) = \alpha L(r).$$

We denote by $\mathcal{L}_{\mathbb{K}}(R)$ the set of all \mathbb{K} -linear operators over R .

It is clear that if $L_1, L_2 \in \mathcal{L}_{\mathbb{K}}(R)$, then $L_1 + L_2$ and $L_1 \circ L_2$ are also \mathbb{K} -linear operators. This makes $(\mathcal{L}_{\mathbb{K}}(R), +, \circ)$ a non-commutative ring.

Example 2.26 (Multiplication operators). Let R be a \mathbb{K} -algebra and $s \in R$. The operator $\mu_s : R \rightarrow R$ defined by $\mu_s(r) = sr$ is a \mathbb{K} -linear operator.

The linearity of the addition is clear from the distribution property of multiplication and addition in a ring. Moreover, if R is commutative, then $\mu_s(\alpha r) = s\alpha r = \alpha \mu_s(r)$.

Example 2.27 (Derivation operators). Let R be a \mathbb{K} -algebra and $\partial : R \rightarrow R$ a derivation over R (i.e., it satisfies the Leibniz rule) such that $\partial(\mathbb{K}) = 0$. Then ∂ is a \mathbb{K} -linear operator.

In this case, we have that for any $\alpha \in \mathbb{K}$:

$$\partial(\alpha r) = \partial(\alpha)r + \alpha\partial(r) = \alpha\partial(r).$$

Example 2.28 (Difference operators). Let R be a \mathbb{K} -algebra and $\sigma : R \rightarrow R$ and endomorphism (i.e., $\sigma(rs) = \sigma(r)\sigma(s)$) such that $\sigma(\alpha) = \alpha$ for all $\alpha \in \mathbb{K}$. Then σ is a \mathbb{K} -linear operator.

Definition 2.29. Let R be a \mathbb{K} -algebra and L a \mathbb{K} -linear operator. We denote by $R[L]$ the set of linear operators of the form:

$$\mu_{r_0} + \dots + \mu_{r_d} \circ L^d.$$

We usually write this operator as a polynomial, replacing \circ by a product notation and μ_{r_i} by r_i :

$$\mu_{r_0} + \dots + \mu_{r_d} \circ L^d \simeq r_0 + r_1 L + \dots + r_d L^d.$$

Lemma 2.30. Let R be a \mathbb{K} -algebra and L a derivation or an endomorphism over R . Then $R[L] \subset \mathcal{L}_{\mathbb{K}}(R)$ is a subring.

Proof. It is enough to show that for any $r \in R$, the operator $L \circ \mu_r$ can be written as an element of $R[L]$.

In the case of L being a derivation, we have that for any element $s \in R$:

$$(L \circ \mu_r)(s) = L(sr) = L(r)s + rL(s),$$

so we have $(L \circ \mu_r) \equiv \mu_{L(r)} + \mu_r \circ L \in R[L]$.

In the case of L being an endomorphism, we have that for any element $s \in R$:

$$(L \circ \mu_r)(s) = L(rs) = L(r)L(s),$$

so we have that $(L \circ \mu_r) \equiv \mu_{L(r)} \circ L \in R[L]$. □

If we look to the definition of D-finite functions, we can rephrase it in the following way: $f(x) \in \mathbb{K}[[x]]$ is D-finite if there is an operator $\mathcal{A} \in K[x][\partial_x]$ such that $\mathcal{A}(f(x)) = 0$.

Similarly, a sequence $(a_n)_n$ is P-recursive if there is an operator $\mathcal{A} \in K[n][\sigma_n]$, where $\sigma_n((a_n)_n) = (a_{n+1})_n$, such that $\mathcal{A}((a_n)_n) = 0$.

In this thesis we use the ring of linear operators generated by a derivation, $R[\partial]$. Note that the commutativity rules described in Lemma 2.30 showed that, in general, if we have $L_1, L_2 \in R[\partial]$ it is not true that $L_1 \circ L_2 = L_2 \circ L_1$.

Chapter 3

Differentially definable functions

This is the main theoretical chapter of the thesis. In the following sections we present an extension for D-finite functions iterating its construction (namely, DD-finite functions) and we prove all the results in Section 2.2 adapted to this new class.

We will also consider the problem of initial values for these type of equations, meaning which initial conditions are required for a given differential operator to characterize one of its solutions. We focus on formal power series solutions.

All the results in this Chapter were published in [38].

3.1 Differentially definable functions

In the present section we develop the mathematical framework for DD-finite functions, i.e., functions satisfying linear differential equations with D-finite coefficients. We present here a more generic definition that includes as particular cases the D-finite and DD-finite functions. We use the concept of a derivation (i.e., an additive map that satisfies the Leibniz rule). For further details and results about this derivations, see Appendix B.

Definition 3.1. Let (R, ∂) be a differential integral domain and $(S, \partial) \supset (R, \partial)$ a differential extension. We say that $f \in S$ is *differentially definable over R* if there is a non-zero operator $\mathcal{A} \in R[\partial]$ that annihilates f , i.e., $\mathcal{A} \cdot f = 0$.

By $D_S(R)$ we denote the set of all $f \in S$ that are differentially definable over R . We omit S , writing simply $D(R)$, when the extension is clear from context.

We define the *order of f w.r.t. R* as the minimal order of the operators that annihilate f (i.e., the minimal ∂ -degree of $\mathcal{A} \in R[\partial]$ such that $\mathcal{A} \cdot f = 0$).

This definition is highly related with the definition of Picard-Vessiot extensions (see Appendix B) when R is a field. In all our particular examples, we will fix $S = \mathbb{K}[[x]]$ and $\partial = \partial_x$ unless stated otherwise.

The condition for R being non-trivial is important because it guarantees that $D(R)$ is never empty. In fact, the set $D(R)$ includes all the elements of the base ring R .

Lemma 3.2. *Let (R, ∂) be a non-trivial differential integral domain and (S, ∂) a differential extension. Then*

$$R \subset D_S(R) \subset S.$$

Proof. The second inclusion is trivial by definition. Let then $f \in R$ and $\mathcal{A} = f\partial - \partial(f)$. Clearly, $\mathcal{A} \in R[\partial]$, and

$$\mathcal{A} \cdot f = f\partial(f) - \partial(f)f = 0,$$

proving that $f \in D_S(R)$. □

For $R = \mathbb{K}$, Definition 3.1 yields the ring of formal power series satisfying linear differential equations with *constant* coefficients, also referred to as C-finite functions. If R is the ring of polynomials in x over \mathbb{K} , $\mathbb{K}[x]$, then Definition 3.1 yields the classical D-finite functions.

We know by Theorem 2.13 that the set $D(\mathbb{K}[x])$ is a differential subring of $\mathbb{K}[[x]]$. Hence, we can consider the set of differentially definable functions over $D(\mathbb{K}[x])$. We call these functions *DD-finite functions*. We denote that set with $D^2(\mathbb{K}[x])$.

The class of DD-finite functions is the main topic of this thesis. Lemma 3.2 shows that this class is bigger than class of D-finite functions. Moreover we can see that there are new functions that falls into this new class:

Example 3.3 (Double exponential). The double exponential function from Corollary 2.23.1 is DD-finite with order one since it is annihilated by the following operator:

$$\mathcal{A}_e = \partial_x - e^x.$$

Example 3.4 (Tangent). The tangent function $\tan(x) = \sin(x)/\cos(x)$ is known to have as derivative the fraction $1/\cos(x)^2$. If we compute its derivative again we have

$$\tan''(x) = \frac{2\sin(x)}{\cos(x)^3} = \tan(x) \frac{2}{\cos(x)^2}.$$

Since $\cos(x)$ is D-finite, we have that $\tan(x)$ is DD-finite annihilated by the differential operator

$$\mathcal{A}_t = \cos(x)^2 \partial_x^2 - 2.$$

Example 3.5 (Mathieu functions [26]). Mathieu functions are the solutions to the Mathieu differential equation:

$$w''(x) + (a - 2q \cos(2x))w(x) = 0,$$

where the parameters a and q are assumed to be constants. This is a generalization of the sine and cosine functions (which we obtain when $a = 1$ and $q = 0$).

The linear differential operator is $\mathcal{A}_m = \partial_x^2 + (a - 2q \cos(2x))$. Since $\cos(x)$ is D-finite annihilated by $\partial_x^2 + 1$, it is easy to see that $\cos(2x)$ is D-finite with annihilating operator $\partial_x^2 + 4$. And, since a and q are constants, we have that all coefficients of \mathcal{A}_m are D-finite. Hence, any Mathieu function $w(x)$ (i.e., solution to $\mathcal{A}_m \cdot w(x) = 0$) is DD-finite for any constant value of a and q .

For classical D-finite functions there exist equivalent characterizations that help prove certain properties. These characterizations (see Theorem 2.12) can be carried over immediately to differentially definable functions.

Theorem 3.6. *Let (R, ∂) be a non-trivial differential integral domain, (S, ∂) a differential extension, $R[\partial]$ the ring of linear differential operators over R , and $F = \text{Fr}(R)$ be the fraction field of R . Let $f \in S$. Then the following are equivalent:*

- (i) f is differentially definable over R ($f \in D_S(R)$).
- (ii) (Inhomogeneous) There are $\mathcal{A} \in R[\partial]$ and $g \in D_S(R)$ such that $\mathcal{A} \cdot f = g$.
- (iii) (Finite dimension) The F -vector space generated by the set $\{\partial^i(f) : i \in \mathbb{N}\}$ has finite dimension.

In fact, f is differentially definable over R with order d if and only if the F -dimension of the vector space in (iii) is exactly d .

Proof. First suppose that (ii) holds, i.e.,

$$\mathcal{A} \cdot f = g.$$

Then, there is a $\mathcal{B} \in R[\partial]$ that annihilates g since $g \in D_S(R)$. Hence,

$$\mathcal{B} \cdot (\mathcal{A} \cdot f - g) = (\mathcal{B}\mathcal{A}) \cdot f = 0.$$

Since $(\mathcal{B}\mathcal{A}) \in R[\partial]$, f is differentially definable over R , proving, (ii) \Rightarrow (i).

Now suppose (i) holds. Then there is $\mathcal{B} \in R[\partial]$ with $\mathcal{A} \cdot f = 0$, so taking $\mathcal{A} = \mathcal{B}$ and $g = 0$, we have proven (ii).

Suppose now that (i) holds for some operator \mathcal{A} of order d . We are going to prove that, for all $i \geq d$, $f^{(i)}(x)$ is an F -linear combination of the previous derivatives.

Let $\mathcal{A} = r_0 + \cdots + r_d \partial^d$, for $r_i \in R$ and $r_d \neq 0$, and consider $\mathcal{A}_i = \partial^i \cdot \mathcal{A}$. Then for each i , $\mathcal{A}_i \in R[\partial]$ has order $d + i$ and leading coefficient r_d . Moreover,

$$\mathcal{A}_i \cdot f = (\partial^i \cdot \mathcal{A}) \cdot f = \partial^i \cdot (\mathcal{A} \cdot f) = \partial^i(0) = 0.$$

Thus, for any $i \in \mathbb{N}$, there are elements $s_0, \dots, s_{d+i-1} \in R$ such that

$$s_0 f + \cdots + s_{d+i-1} \partial^{d+i-1}(f) + r_d \partial^{d+i}(f) = 0,$$

and then the following identity holds:

$$\partial^{d+1}(f) = -\frac{s_0}{r_d} f - \cdots - \frac{s_{d+i-1}}{r_d} \partial^{d+i-1}(f).$$

Since all $-\frac{s_i}{r_d} \in F$, we have that $f^{(d+i)}$ is an F -linear combination of the first $d + i - 1$ derivatives of f for all $i \in \mathbb{N}$. Hence,

$$\langle \partial^i(f) : i \in \mathbb{N} \rangle_F = \langle f, \partial(f), \dots, \partial^{d-1}(f) \rangle_F.$$

Finally, suppose now that (iii) holds and the F -vector space generated by the derivatives of f has finite dimension d . Then there is $n \leq d$ such that the set $\{f, \partial(f), \dots, \partial^n(f)\}$ is linearly dependent and the set $\{f, \partial(f), \dots, \partial^{n-1}(f)\}$ is linearly independent. Take then for $i = 0, \dots, n$ fractions $\frac{r_i}{s_i} \in F$, such that $r_n \neq 0$ and

$$\frac{r_0}{s_0} f + \cdots + \frac{r_n}{s_n} \partial^n(f) = 0.$$

Let s be a common multiple of $\{s_0, \dots, s_n\}$ and $p_i = \frac{sr_i}{s_i} \in R$. Then

$$p_0 f + \cdots + p_n \partial^n(f) = 0.$$

For $\mathcal{A} = p_0 + \cdots + p_n \partial^n \in R[\partial]$ we have that $\mathcal{A} \cdot f = 0$, i.e., $f \in D(R)$. \square

Example 3.7 (Tangent). Following Example 3.4, we know that

$$\tan'(x) = \frac{1}{\cos(x)^2},$$

which leads to the inhomogeneous equation $\cos(x)^2 \tan'(x) = 1$, showing that $\tan(x)$ is DD-finite with order at most two. If we apply the Theorem 3.6, we obtain that $\tan(x)$ is annihilated by the differential operator

$$\tilde{\mathcal{A}}_t = \cos(x) \partial_x^2 - 2 \sin(x) \partial_x,$$

which is a different operator than the obtained in Example 3.4.

3.2 Closure properties

After properly defining the set of DD-finite functions and studying several characterizations for differentially definable functions, we are going now to prove several closure properties that D-finite functions already satisfied (see Theorem 2.13). All the results included in this Section are a direct extension for those that we stated in Section 2.2.

These closure properties were the key to proving identities for D-finite functions, and having them proven for differentially definable functions will allow us to prove identities for DD-finite functions too. Furthermore, as we will study deeper in Chapter 5, these closure properties show that we can iterate even further this construction, building new rings $D(D^2(\mathbb{K}[x])) = D^3(\mathbb{K}[x])$ and so on. The closure properties that we derive here hold in every layer of this hierarchy.

We start first with the two differential closure properties: derivation and integration.

Proposition 3.8. *Let (R, ∂) be a differential integral domain, F its field of fractions and $f \in D(R)$ with order d . Then $\int f \in D(R)$ with order at most $d + 1$.*

Proof. There are two ways of proving this result. First, we can use Theorem 3.6 and see that for $g = \int f$, the vector space $\langle \partial^i(g) : i \in \mathbb{N} \rangle_F$ is exactly the vector space generated by f and its derivatives (since $\partial(g) = f$) and g itself, having at most dimension $d + 1$.

On the other hand, let $\mathcal{A} = r_0 + \dots + r_d \partial^d \in R[\partial]$ such that $\mathcal{A} \cdot f = 0$. Using $\partial(g) = f$, we then have

$$\mathcal{A} \cdot \partial(g) = (\mathcal{A}\partial) \cdot g = 0.$$

□

The differences between these two approaches is that in the second case we already have the differential operator that annihilates $\int f$ directly, while the first approach guarantees the existence of this operator, but never computes it. The same two approaches can be use for proving the closure of the derivation:

Proposition 3.9. *Let (R, ∂) be a differential integral domain, F its field of fractions and $f \in D(R)$ with order d . Then $\partial(f) \in D(R)$ with order at most d .*

Proof. We can prove this using Theorem 3.6 again and remarking that

$$\langle \partial(f), \partial^2(f), \dots \rangle_F \subset \langle f, \partial(f), \partial^2(f), \dots \rangle_F.$$

However, we also provide a constructive approach for this proof. Let $\mathcal{A} = r_0 + \dots + r_d \partial^d$ be a differential operator that annihilates f . If $r_0 = 0$, then we can write $\mathcal{A} = (r_1 + \dots + r_d \partial^{d-1})\partial$, so we have

$$(r_1 + \dots + r_d \partial^{d-1}) \cdot (\partial(f)) = 0,$$

showing that $\partial(f) \in D(R)$ with order at most $d - 1$.

On the other hand, if $r_0 \neq 0$, we can build $\mathcal{B} = r_0(\partial\mathcal{A}) - \partial(r_0)\mathcal{A}$. It is clear that $\mathcal{B} \cdot f = 0$, $\deg_{\partial}(\mathcal{B}) = d + 1$ and that:

$$[\partial^0](\mathcal{B}) = r_0[\partial^0](\partial\mathcal{A}) - \partial(r_0)[\partial^0](\mathcal{A}) = r_0\partial(r_0) - \partial(r_0)r_0 = 0$$

Then, applying the construction when $r_0 = 0$, we have $\partial(f) \in D(R)$ with order at most d . \square

Note that in the proof above only basic ring operations in $R[\partial]$ were needed. Now, we prove the closure properties for addition and Cauchy product. For these two operations, only the non constructive approach is used, based in Theorem 3.6. We will spend Chapter 4 explaining how to make these closure properties constructive and, then, building a full computable class of functions.

Proposition 3.10 (Arithmetic closure properties). *Let (R, ∂) be a differential integral domain and (S, ∂) a differential extension. Let $f, g \in D_S(R)$ with orders d_1, d_2 , respectively. Then:*

- $f + g$ is in $D_S(R)$ with order at most $d_1 + d_2$.
- fg is in $D_S(R)$ with order at most $d_1 d_2$.

Proof. Let F be the field of fractions of R . Define for $f \in S$ the F -vector space

$$V_F(f) = \langle f, \partial(f), \dots \rangle_F = \langle \partial^i(f) : i \in \mathbb{N} \rangle_F \subset S.$$

Then, using Theorem 3.6, we obtain

$$\dim(V_F(f)) = d_1, \dim(V_F(g)) = d_2.$$

On the other hand, if we compute the k th derivative of the addition and product of f and g we have:

- $\partial^k(f + g) = \partial^k(f) + \partial^k(g) \in V_F(f) + V_F(g),$
- $\partial^k(fg) = \sum_{i=0}^k \binom{k}{i} \partial^i(f) \partial^{k-i}(g) \in V_F(f)V_F(g),$

where $V_F(f) + V_F(g), V_F(f)V_F(g) \subset S$ are the vector spaces generated by the sum (resp., the product) of generators of $V_F(f)$ and $V_F(g)$.

Thus we have that $V_F(f + g) \subset V_F(f) + V_F(g)$ and that $V_F(fg) \subset V_F(f)V_F(g)$, and so:

$$\dim(V_F(f + g)) \leq \dim(V_F(f)) + \dim(V_F(g)) \leq d_1 + d_2 < \infty,$$

$$\dim(V_F(fg)) \leq \dim(V_F(f)) \dim(V_F(g)) \leq d_1 d_2 < \infty.$$

Hence, both $f + g$ and fg are in $D_S(R)$ with orders at most $d_1 + d_2$ and $d_1 d_2$, respectively. \square

As we will detail in Chapter 4, this finite dimension of a vector space will lead to an ansatz algorithm to build explicitly the operators that annihilate both the addition and product of two differentially definable functions.

An immediate consequence of the closure properties of Propositions 3.9 and 3.10 is that $D(R)$ is a differential subring of $\mathbb{K}[[x]]$.

Theorem 3.11. *Let (R, ∂) be a differential integral domain and (S, ∂) a differential extension. Then $(D_S(R), \partial)$ is again a differential integral domain.*

Hence, starting from any differential integral domain (R, ∂) and an ambient space S , the process of constructing the differentially definable functions over R can be iterated and, for each layer of $D^k(R)$, the closure properties hold. In particular, for $S = \mathbb{K}[[x]]$ and $R = \mathbb{K}[x]$, we say that the n th iteration, $D^n(\mathbb{K}[x])$, is the class of D^n -finite functions, including then the well known class of D -finite functions, and the class of DD -functions that we focus on this thesis.

Now that we have proven the arithmetic and differential closure properties for differentially definable functions (which are a direct extension of the results from Theorem 2.13), we extend also the result from Theorem 2.14 concerning algebraic functions. The proof of the following Proposition is an adaptation from the general setting from Stanley's proof [67, Theorem 2.1].

Proposition 3.12. *Let (R, ∂) be a differential integral domain and (S, ∂) be a differential extension. Let F be the field of fractions of R . If $f \in S$ is algebraic over F then $f \in D_S(R)$.*

Proof. Suppose that f is algebraic over F and let $m(y) \in F[y]$ be its minimal polynomial. Let κ_∂ denote the coefficient-wise derivation on $F[y]$ induced by the given derivation ∂ on F and ∂_y the usual derivation on $F[y]$ (i.e., $\partial_y y = 1$ and $\partial_y F = 0$). Then differentiating the equation $m(f) = 0$, we get that

$$(\partial_y m(f))\partial(f) + \kappa_\partial(m)(f) = 0. \quad (3.13)$$

As $m(y)$ is irreducible and $\deg_y(m) > \deg_y(\partial_y m)$, they are coprime polynomials in $F[y]$. Hence, there are $s(y), r(y) \in F[y]$ such that

$$r(y)m(y) + s(y)\partial_y m(y) = 1,$$

and substituting $y = f$ we obtain

$$s(f)\partial_y m(f) = 1.$$

Using this equality in (3.13), we have that

$$\partial(f) = -s(f)\kappa_\partial(m)(f) = \tilde{m}(f).$$

Now for any g where $g = p(f)$ for some $p(y) \in F[y]$,

$$\partial(g) = \partial_y p(f(x)) \partial(f) + \kappa_{\partial} p(f) = (\partial_y p(y) \tilde{m}(y) + \kappa_{\partial}(p)(y))(f).$$

In other words, all the derivatives of any g that can be written as a polynomial in f with coefficients in F can be expressed in the same way. Summarizing we have

$$\langle \partial^i(f) : i \in \mathbb{N} \rangle_F \subset \langle f^i : i \in \mathbb{N} \rangle_F.$$

Since f is algebraic with minimal polynomial $m(y)$, the F -vector space generated by the powers of f has dimension $\deg_y(m) < \infty$, and then by Theorem 3.6, $f \in D_S(R)$ with order at most $\deg_y(m)$. \square

There are other closure properties regarding the multiplicative inverse of functions:

Lemma 3.14. *Let (R, ∂) be a differential integral domain and (S, ∂) a differential extension. Consider $r \in R$ and $f \in D_S(R)$.*

- (i) *If $s \in S$ is the multiplicative inverse of r (i.e., $rs = 1$), then $s \in D_S(R)$ with order 1.*
- (ii) *If $g \in S$ is the multiplicative inverse of f (i.e., $fg = 1$) and f has order 1 then $g \in D_S(R)$ also with order 1.*
- (iii) *If there is $g \in S$, $r \in R$ and $n \in \mathbb{N}$ such that $f = gr^n$, then $g \in D(R)$ with order at most the order of f .*

Proof. Since, by Lemma 3.2, $R \subset D_S(R)$ and all elements of R are of order 1, (i) is a consequence of (ii). For proving (ii), consider $\mathcal{A} \in R[\partial]$ that annihilates f of the form

$$\mathcal{A} = a\partial + b.$$

Let $\tilde{\mathcal{A}} = a\partial - b$. Then, using basic properties of derivations

$$\tilde{\mathcal{A}} \cdot g = -\frac{a\partial(f)}{f^2} - \frac{b}{f} = -\frac{a\partial(f) + bf}{f^2} = -\frac{\mathcal{A} \cdot f}{f^2} = 0,$$

hence $g \in D_S(R)$ with order 1.

Finally, for proving (iii) is enough to plug the formula,

$$\partial^k(f) = \partial^k(gr^n) = \sum_{i=0}^k \binom{k}{i} \partial^i(g) \partial^{k-i}(r^n),$$

into the equation $\mathcal{A} \cdot f = 0$ and rearrange terms to obtain the operator annihilating g . The resulting operator only depends on the coefficients of \mathcal{A} , binomial coefficients, and powers of derivatives of r . As R is a differential integral domain, provided that $f \in D_S(R)$, this operator is in $R[\partial]$. These calculations do not affect the order of the operator, hence g has at most the same order as f . \square

Example 3.15 (Tangent). Let take a look again to Examples 3.4 and 3.7. In both cases we had that $\tan(x)$ was annihilated by a differential operator of order two with D-finite coefficients. But now, using the results in Proposition 3.10 and Lemma 3.14 we can show that $\tan(x)$ is DD-finite with order 1.

We start from its original formula: $\tan(x) = \sin(x)/\cos(x)$. Since $\cos(0) = 1 \neq 0$, its multiplicative inverse is again a formal power series and, from Lemma (i), such inverse is DD-finite of order 1. On the other hand, $\sin(x)$ is also DD-finite of order 1 by Lemma 3.2. Now, using Proposition 3.10, we have that the product of $\sin(x)$ and $1/\cos(x)$ is DD-finite with order 1. In fact, it is easy to check that the operator

$$\overline{\mathcal{A}}_t = \sin(x) \cos(x) \partial - 1$$

annihilates $\tan(x)$.

3.3 DD-finite power series solutions

In this Section we study how to distinguish one particular solution from a linear differential equation. Until now, we have left the ambient ring S free, but in this particular section we fix $S = \mathbb{K}[[x]]$, and we look for solutions in the ring of formal power series.

Naturally, when working with differential equations, initial values need to be given to characterize one solution to the given equation. Under certain regularity assumptions on the coefficients, existence and uniqueness of a solution can be proven [63]. This solution is formed as a linear combination of d linearly independent solutions, where d is the order of the differential equation. The coefficients in this linear combination are given by d initial conditions.

In the case of formal power series $f(x) = \sum_{n \geq 0} f_n x^n$, the only evaluation possible is at the origin with $f(0)$ equal to the constant coefficient in this series expansion. Moreover, as we saw in Lemma 2.8, further evaluations at $x = 0$ of the derivatives yield to the elements f_n , namely

$$n! f_n = f^{(n)}(0).$$

Lemma 3.16. *Let R be a differential subring of $\mathbb{K}[[x]]$ and $\mathcal{A} \in R[\partial_x]$. Let $\mathcal{S}_{\mathbb{K}[[x]]}(\mathcal{A})$ (or simply $\mathcal{S}(\mathcal{A})$) be the set of formal power series $f(x) \in \mathbb{K}[[x]]$ such that $\mathcal{A} \cdot f(x) = 0$. Then $\mathcal{S}(\mathcal{A})$ is a \mathbb{K} -vector space.*

Proof. Trivial since $\partial_x(cf(x) + g(x)) = c\partial_x(f(x)) + \partial_x(g(x))$ for any $c \in \mathbb{K}$ and $f(x), g(x) \in \mathbb{K}[[x]]$ \square

The following results intend to find an appropriate basis of this vector space relating it with the initial values $f^{(n)}(0)$ that are needed to fix one particular

solution in $\mathcal{S}(\mathcal{A})$. Consider the differential operator we are studying:

$$\mathcal{A} = r_0(x) + r_1(x)\partial_x + \dots + r_d(x)\partial_x^d,$$

where all the coefficients $r_i(x)$ are formal power series in the differential subring R :

$$r_i(x) = \sum_{n \geq 0} r_{i,n} x^n.$$

Usually (when $r_d(0) \neq 0$), it is easy to see that all solutions are characterized by the first d initial conditions $f(0), \dots, f^{(d-1)}(0)$, since

$$\partial_x^k \mathcal{A} = s_0(x) + s_1(x)\partial_x + \dots + s_{d+k-1}(x)\partial_x^{d+k-1} + r_d(x)\partial_x^{d+k},$$

so using that $\mathcal{A} \cdot f(x) = 0$ we obtain that $\partial_x^k \mathcal{A} \cdot f(x) = 0$. In this identity, we can evaluate at the value $x = 0$ and obtain:

$$f^{(k+d)}(0) = -\frac{s_0(0)f(0) + s_1(0)f'(0) + \dots + s_{k+d-1}(0)f^{(k+d-1)}(0)}{r_d(0)}.$$

This means that, fixed a vector $\mathbf{v} = (v_0, \dots, v_{d-1})$ there is a unique power series $f(x) \in \mathbb{K}[[x]]$ such that $\mathcal{A} \cdot f(x) = 0$ and $f^{(i)}(0) = v_i$ for $i = 0, \dots, d-1$. Moreover, if we consider $f_{(i)}(x) \in \mathbb{K}[[x]]$ as the solution associated with \mathbf{e}_i (the vector with 0 in all components except the i th coordinate), it is clear that all $f_{(i)}(x)$ are \mathbb{K} -linearly independent, showing that they form a basis of $\mathcal{S}(\mathcal{A})$.

However, there are plenty of cases where this condition ($r_d(0) \neq 0$) does not hold. Intuitively, we would think that this condition may only reduce the dimension of the vector space $\mathcal{S}(\mathcal{A})$, as it happens with the particular case of $\mathcal{B} = x\partial_x^2 + \partial_x + x \in \mathbb{K}[x][\partial_x]$. It is well known [26] that this differential equation has two solutions: the Bessel function of first kind $J_0(x)$ and the Bessel function of second kind $Y_0(x)$. This latter function has a pole at $x = 0$ and it is not a formal power series. Hence the solution space $\mathcal{S}(\mathcal{B})$ can only be 1-dimensional (instead of the expected 2-dimensional space given by the order of the equation).

But even so, this is not complete. As a motivating example, consider the simple differential operators $\mathcal{C}_m = x\partial_x - m$ for $m \in \mathbb{N} \setminus \{0\}$. The solutions are of the form Cx^m , for some $C \in \mathbb{K}$ (i.e, they form \mathbb{K} -vector spaces of dimension 1). It is easy to see that these equations fix the first m initial values to be 0 and the $(m+1)$ th initial value determines C (indeed, $f^{(m)}(0) = m!C$).

These examples show that we do not only need to compute how many initial conditions are needed to characterize one particular solution of \mathcal{A} , but also which initial conditions are required.

Similar to the case of D-finite functions we transfer the problem of computing initial values to the level of the coefficient sequences [44]. We provide a method to compute the dimension of the solution space for a linear differential equation and divide this process into three steps:

1. Set up an infinite linear system for the coefficient sequence from the differential equation.
2. Examine the structure of this linear system.
3. Relate the dimension of the solution space to the rank of a finite matrix.

We also reduce the problem to a particular type of differential operators where x is not a left factor of the operator (which means that there is a coefficients with a non-zero value at $x = 0$).

Definition 3.17. Let R be a differential subring of $\mathbb{K}[[x]]$ and $\mathcal{A} \in R[\partial_x]$, written as

$$\mathcal{A} = r_d(x)\partial_x^d + \dots + r_0(x).$$

We say that \mathcal{A} is *zero-reduced* if there is $i = 0, \dots, d$ such that $r_i(0) \neq 0$.

Since this condition is equivalent to $\gcd(r_0(x), \dots, r_d(x)) = 1$ in $\mathbb{K}[[x]]$, then it is clear that being differentially definable w.r.t. $R \subset \mathbb{K}[[x]]$ is equivalent to be annihilated by a zero-reduced operator in $R[\partial_x]$:

Lemma 3.18. *Let R be a differential subring of $\mathbb{K}[[x]]$. Then $f(x) \in D(R)$ if and only if there exists a zero-reduced operator $\tilde{\mathcal{A}} \in R[\partial_x]$ such that $\mathcal{A} \cdot f(x) = 0$*

Proof. Let $f(x) \in D(R)$ with annihilating operator $\mathcal{A} = r_d(x)\partial_x^d + \dots + r_0(x) \in R[\partial_x]$. Assume that $r_j(0) = 0$ for all $j = 0, \dots, d$ (else there is nothing to prove). Then there exists a maximal index k such that $r_{j;l} = 0$ for all $j = 0, \dots, d$ and for all $l = 0, \dots, k$. Thus the factor x^k can be pulled out of each coefficient $r_j(x)$. The operator $\tilde{\mathcal{A}} = x^{-k}\mathcal{A}$ is in $R[\partial_x]$, annihilates f , and not all its coefficients vanish at $x = 0$. \square

In the following, we denote the coefficient sequence of any arbitrary power series $f(x) = \sum_{n \geq 0} f_n x^n$ by $\mathbf{f} = (f_n)_{n \geq 0}$. It is clear that $\mathcal{A} \cdot f = 0$ if and only if all its coefficients vanish, i.e.,

$$[x^n](\mathcal{A} \cdot f(x)) = \sum_{i=0}^d \sum_{k=0}^n r_{i;n-k} (k+i)^{\underline{i}} f_{k+i} = 0, \quad \text{for all } n \in \mathbb{N},$$

where, as we used in Section 2.3, $(a)^{\underline{m}} = a \cdot (a-1) \cdots (a-m+1)$ denotes the falling factorial. Reorganizing this sum, one can extract the coefficients f_k from the innermost sum and obtain

$$\sum_{k=0}^{n+d} \left(\sum_{l=\max\{0, k-n\}}^{\min\{d, k\}} (k)^{\underline{l}} r_{l;n-k+l} \right) f_k = 0, \quad \text{for all } n \in \mathbb{N},$$

as the condition for \mathcal{A} annihilating f on the sequence level. Now let's define the infinite dimensional matrix $M^{\mathcal{A}} = (m_{n,k})_{n,k \geq 0}$ with

$$m_{n,k} = \begin{cases} \sum_{l=\max\{0,k-n\}}^{\min\{d,k\}} (k)^l r_{l;n-k+l}, & k \leq n+d, \\ 0, & k > n+d. \end{cases}$$

With this definition we have that $\mathcal{A} \cdot f = 0$ if and only if $M^{\mathcal{A}} \mathbf{f} = \mathbf{0}$. This matrix $M^{\mathcal{A}}$ has almost a lower left triangular structure since $m_{n,k} = 0$ for $k > n+d$.

The values of the entries of $M^{\mathcal{A}}$ does not seem easy to compute, in the sense that we need to compute all the expansions of the coefficients $r_i(x)$ for getting the whole matrix. However, there are areas of the matrix where the entries of the matrix, if written in proper way, can be easily computed.

Let us consider the strip $n \leq k \leq n+d$, these matrix entries can be computed as follows for each $n \geq d$:

$$m_{n,k} = \sum_{l=\max\{0,k-n\}}^{\min\{d,k\}} (k)^l r_{l;n-k+l} = \sum_{l=k-n}^d (k)^l r_{l;n-k+l},$$

and rewriting $k = n + i$, for $i = 0, \dots, d$, we obtain for each i the following polynomials in n of degree at most d :

$$m_{n,n+i} = \sum_{l=i}^d (n+i)^l r_{l;l-i} =: \mathfrak{f}_i(n).$$

We single out these polynomials as they play an important role in understanding the matrix $M^{\mathcal{A}}$.

As we saw at the beginning of this section, for computing the number of initial values necessary to define a formal power series uniquely with a given operator, it plays an essential role whether or not the leading coefficient vanishes at $x = 0$. Now we see a generalization of that fact that applies to zero-reduced operators and gives the key information to understand the matrix $M^{\mathcal{A}}$.

Lemma 3.19. *Let R be a differential subring of $\mathbb{K}[[x]]$ and $\mathcal{A} = r_d(x)\partial_x^d + \dots + r_0(x) \in R[\partial_x]$. If $r_i(0) \neq 0$ then $\mathfrak{f}_i(n) \neq 0$.*

Proof. Let i be such that $r_i(0) = r_{i;0} \neq 0$. Now $\mathfrak{f}_i(n) = 0$, if and only if all its coefficients vanish. But, by definition,

$$\mathfrak{f}_i(n) = \sum_{l=i}^d (n+i)^l r_{l;l-i} = (n+i)^i r_{i;0} + \sum_{l=i+1}^d (n+i)^l r_{l;l-i},$$

so at least one coefficient (the one with less degree) is non-zero. □

We denote by $\nu(\mathcal{A})$ now the minimal index i such that $\mathfrak{f}_i(n) \neq 0$, and we set it to -1 when they are all identically zero. It is clear that \mathcal{A} is zero-reduced if and only if $\nu(\mathcal{A}) \geq 0$. Moreover, for zero-reduced operators the entries of the matrix $M^{\mathcal{A}}$ vanishes after the diagonal $k = n + \nu(\mathcal{A})$:

Corollary 3.19.1. *Let R be a differential subring of $\mathbb{K}[[x]]$ and \mathcal{A} a zero-reduced operator of order d . Then for $M^{\mathcal{A}} = (m_{n,k})_{n,k \geq 0}$ and $n \geq d - \nu(\mathcal{A})$,*

$$\begin{cases} m_{n,k} = 0 & \text{for all } k > n + \nu(\mathcal{A}), \\ m_{n,n+\nu(\mathcal{A})} = \mathfrak{f}_{\nu(\mathcal{A})}(n) \end{cases}$$

Proof. Once again, writing $k = n + i$, $i \geq \nu(\mathcal{A})$, we have $\mathfrak{f}_i(n) = m_{n,n+i}$ for $n \geq d - \nu(\mathcal{A})$. Then the statement follows for $n + \nu(\mathcal{A}) \leq k \leq n + d$ by definition of $\nu(\mathcal{A})$ and for $k > n + d$ by definition of $M^{\mathcal{A}}$. \square

Then, the diagonal given by $k = n + \nu(\mathcal{A})$ will only vanish in a finite set of rows (since it is the evaluation of a polynomial on n) and after one particular point, this diagonal will never be zero again, providing the conditions for the required initial conditions that we were looking for. The following definitions and results are the formalism of this idea.

Let start by analyzing the finite submatrices that are truly interesting for our study:

Definition 3.20. Let R be a differential subring of $\mathbb{K}[[x]]$ and $\mathcal{A} \in R[\partial_x]$ a zero-reduced operator of order d .

- We define the p th recursion matrix as $M_p^{\mathcal{A}} = (m_{n,k})_{0 \leq n \leq p}^{0 \leq k \leq p + \nu(\mathcal{A})}$, i.e., the upper left submatrix of $M^{\mathcal{A}}$ of dimension $(p + 1) \times (p + \nu(\mathcal{A}) + 1)$.
- Let $\mathfrak{f}(n) = \mathfrak{f}_{\nu(\mathcal{A})}(n)$ (also known as *indicial polynomial*) and μ be the maximal integer root of $\mathfrak{f}(n)$ or $-\infty$, if there are no integer roots. Then we define $\lambda(\mathcal{A}) := \max\{d - \nu(\mathcal{A}), \mu\}$.

With these notations at hand, we are in the position to state our result about the dimension of the solution space of a zero-reduced operator. First note that for any $p \geq \lambda(\mathcal{A})$, the dimension of the nullspace of $M_p^{\mathcal{A}}$ does not change any more.

Proposition 3.21. *Let R be a differential subring of $\mathbb{K}[[x]]$ and $\mathcal{A} \in R[\partial_x]$ a zero-reduced operator of order d . Let $p > \lambda(\mathcal{A})$. Then:*

$$\text{rk}(M_p^{\mathcal{A}}) = 1 + \text{rk}(M_{p-1}^{\mathcal{A}}).$$

Proof. Using Corollary 3.19.1 we have, for $p \geq d - \nu(\mathcal{A})$ that the matrix $M_p^{\mathcal{A}}$ has the following recursive shape:

$$M_p^{\mathcal{A}} = \begin{pmatrix} M_{p-1}^{\mathcal{A}} & 0 \\ * & \mathfrak{f}(p) \end{pmatrix}.$$

Now, as we have $p > \lambda(\mathcal{A})$, we know that $p > d - \nu(\mathcal{A})$ and also that $\mathfrak{f}(p) \neq 0$. Hence, the last row of the matrix is always linearly independent of the others, which makes that

$$\text{rk}(M_p^{\mathcal{A}}) = 1 + \text{rk}(M_{p-1}^{\mathcal{A}}).$$

□

Proposition 3.22. *Let R be a differential subring of $\mathbb{K}[[x]]$ and $\mathcal{A} \in R[\partial_x]$ a zero-reduced operator of order d . Let $\mathbf{v} = (v_0, \dots, v_{\lambda(\mathcal{A})+\nu(\mathcal{A})})$ be a vector in the right nullspace of $M_{\lambda(\mathcal{A})}^{\mathcal{A}}$. Then there is a unique sequence $\mathbf{f} = (f_n)_{n \geq 0}$ such that $f_i = v_i$ for $i = 0, \dots, \lambda(\mathcal{A}) + \nu(\mathcal{A})$ and, for all $p \in \mathbb{N}$,*

$$M_p^{\mathcal{A}} \mathbf{f}_p = 0,$$

where $\mathbf{f}_p = (f_0, \dots, f_{p+\nu(\mathcal{A})})$.

Proof. We build the sequence \mathbf{f} inductively, i.e., we build the element f_p using the values of f_i for $0 \leq i < p$. We start with $f_i = v_i$ for $i = 0, \dots, \lambda(\mathcal{A}) + \nu(\mathcal{A})$. Since \mathbf{v} is in the right nullspace of $M_{\lambda(\mathcal{A})}^{\mathcal{A}}$, and because of the structure of the matrix (see definition 3.20), for $0 \leq p \leq \lambda(\mathcal{A})$,

$$M_p^{\mathcal{A}} \mathbf{f}_p = 0.$$

Now suppose we have constructed \mathbf{f}_p such that $M_q^{\mathcal{A}} \mathbf{f}_q = 0$ for all $q = 0, \dots, p$. Using again the block structure of the matrix it follows immediately that

$$M_{p+1}^{\mathcal{A}} \mathbf{f}_{p+1} = \begin{pmatrix} M_p^{\mathcal{A}} & \mathbf{0} \\ m_{p+1,0} \dots m_{p+1,p+\nu(\mathcal{A})} & \mathfrak{f}(p+1) \end{pmatrix} \begin{pmatrix} \mathbf{f}_p \\ f_{p+\nu(\mathcal{A})+1} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ * \end{pmatrix}.$$

Since we want that \mathbf{f}_{p+1} to be in the nullspace of $M_{p+1}^{\mathcal{A}}$, and noting that $\mathfrak{f}(p+1) \neq 0$, we need to set

$$f_{p+\nu(\mathcal{A})+1} = - \left(m_{p+1,0} f_0 + \dots + m_{p+1,p+\nu(\mathcal{A})} f_{p+\nu(\mathcal{A})} \right) / \mathfrak{f}(p+1)$$

getting a unique definition for $f_{p+\nu(\mathcal{A})+1}$. □

The sequence \mathbf{f} constructed in the previous proposition yields the formal power series annihilated by \mathcal{A} . Hence we can conclude with the explicit formula for the dimension of the solution space, which is equivalent to the number of initial values needed to define a unique solution of \mathcal{A} .

Theorem 3.23 (Dimension of solution space). *Let R be a differential subring of $\mathbb{K}[[x]]$ and $\mathcal{A} \in R[\partial_x]$ a zero-reduced operator. Then the dimension of the solution space of \mathcal{A} within $\mathbb{K}[[x]]$ is the dimension of the right nullspace of $M_{\lambda(\mathcal{A})}^{\mathcal{A}}$, i.e.:*

$$\lambda(\mathcal{A}) + \nu(\mathcal{A}) - \text{rk}(M_{\lambda(\mathcal{A})}^{\mathcal{A}}) + 1.$$

Let $\{v_1, \dots, v_r\}$ be a basis for the nullspace of $M_{\lambda(\mathcal{A})}^{\mathcal{A}}$, i.e., the dimension of the solution space of the operator \mathcal{A} is r . Then the first $\lambda(\mathcal{A}) + \nu(\mathcal{A}) + 1$ values of the coefficient sequence \mathbf{f} are given as $\mathbf{f}_{\lambda(\mathcal{A})} = \gamma_1 v_1 + \dots + \gamma_r v_r$. Obviously the elements f_m for which all components $v_{j;m} = 0$, $j = 1, \dots, r$, are fixed to be zero. Hence, we know how many and which initial values are necessary to define a unique solution in the ring of formal power series.

This is a generalization of the classical result where the leading coefficient r_d does not vanish at zero. In that case, we have $\mathbf{f}_d(n) = r_d(0)(n+d)^d$, hence $\nu(\mathcal{A}) = d$ and $\lambda(\mathcal{A}) = 0$. Then the dimension of the solution space is determined by the nullspace of the matrix $M_0^{\mathcal{A}}$ which is a row vector with $d+1$ entries with the last being $\varphi(0) \neq 0$. Hence the dimension is d and the initial values $f(0), \dots, f^{(d-1)}(0)$ are required to characterize a solution of \mathcal{A} .

We now apply this results to the motivating examples for the Bessel functions and the power functions Cx^n described at the beginning of the Section and see how this methodology provides the full information for identifying solutions for differential operators:

Example 3.24 (Bessel differential operators). Let \mathcal{B}_m be the differential operator for the m th Bessel function:

$$\mathcal{B}_m = x^2 \partial_x^2 + x \partial_x + (x^2 - m^2).$$

If $m \neq 0$, we have the following \mathbf{f} polynomials:

$$\mathbf{f}_0(n) = n^2 - m^2 = (n-m)(n+m), \quad \mathbf{f}_1(n) = 0, \quad \mathbf{f}_2(n) = 0.$$

Then, we have directly that $\nu(\mathcal{B}_m) = 0$. If $m \notin \mathbb{N}$ then $\lambda(\mathcal{B}_m) = 2$ and then we need to study the matrix $M_2^{\mathcal{B}_m}$:

$$M_2^{\mathcal{B}_m} = \begin{pmatrix} -m^2 & 0 & 0 \\ 0 & 1 - m^2 & 0 \\ 1 & 0 & 4 - m^2 \end{pmatrix}.$$

It is easy to see (since $m \notin \mathbb{N}$) that the nullspace has dimension 0, hence there is only one formal power series solution to this equation: the constant function zero.

Now, suppose that $m \in \mathbb{N}$ and $m > 0$. In this case we have $\lambda(\mathcal{B}_m) = \max\{2, m\}$. Since $\mathbf{f}_0(n)$ has only as positive root m and $\nu(\mathcal{B}_m) = 0$, then it is the m th initial value the one that fixes the solution to \mathcal{B}_m .

In the case of $m = 0$, \mathcal{B}_0 is not zero-reduced. But we can see that $\widetilde{\mathcal{B}}_0 = x\partial_x^2 + \partial_x + x$ is the corresponding zero-reduced operator. For this new operator we have the following \mathfrak{f} polynomials:

$$\mathfrak{f}_0(n) = 0, \quad \mathfrak{f}_1(n) = (n+1)^2, \quad \mathfrak{f}_2(n) = 0.$$

From this is easily obtained that $\nu(\widetilde{\mathcal{B}}_0) = 1$ and $\lambda(\widetilde{\mathcal{B}}_0) \max\{-1, (2-1)\} = 1$. Hence, to determine the dimension and structure of the solution space of \mathcal{B}_0 within $\mathbb{K}[[x]]$ we need to consider the matrix

$$M_1^{\widetilde{\mathcal{B}}_0} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 4 \end{pmatrix}.$$

This matrix has a one-dimensional right nullspace with basis $\{(1, 0, -1/4)^T\}$. Hence, to define a unique solution in $\mathbb{K}[[x]]$ of the equation given by \mathcal{C} one initial value needs to be specified (not two as the order of the differential operator would indicate). Furthermore we have that f_0 or f_2 need to be specified and that f_1 is fixed to be zero.

Example 3.25. Now consider the linear differential operator $\mathcal{C}_m = x\partial_x - m$ for some $m \in \mathbb{N} \setminus \{0\}$ with general solution Cx^m . This operator is also already zero-reduced and its order is one. We have that $\mathfrak{f}_0(n) = n - m$ and $\mathfrak{f}_1(n) = 0$. Hence, $\nu(\mathcal{C}_m) = 0$ for all $m \in \mathbb{N}$ and $\lambda(\mathcal{C}_m) = m$. Now we can set up the matrix $M_m^{\mathcal{C}_m}$ which has the following structure

$$M_m^{\mathcal{B}_m} = \begin{pmatrix} -m & 0 & 0 & \cdots & 0 & 0 \\ 0 & -m+1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & -m+2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \\ 0 & 0 & 0 & \cdots & -1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 \end{pmatrix}.$$

For any m , the right nullspace of this matrix has dimension one and is generated by the vector $(0, \dots, 0, 1)^T$. Consequently f_m needs to be specified and all $f_j = 0$ for $j = 0, \dots, m-1$. This was the same phenomena that we saw in the case of Example 3.24 where $m \in \mathbb{N} \setminus \{0\}$.

We now apply this ideas to the operators we got for the tangent and compare its results.

Example 3.26 (Continuation from Example 3.15). Let $\overline{\mathcal{A}}_t = \sin(x) \cos(x) \partial - 1$, be the annihilating operator for $\tan(x)$. This operator is zero-reduced of order 1. Computing the \mathfrak{f} polynomials yield:

$$\mathfrak{f}_0(n) = 1 - n, \quad \mathfrak{f}_1(n) = 0,$$

so we have $\nu(\tilde{\mathcal{C}}) = 0$ and, since $\mathfrak{f}_0(1) = 0$, $\lambda(\tilde{\mathcal{C}}) = \max\{1, 1 - 0\} = 1$. Now we study the first recursion matrix:

$$M_1^{\overline{\mathcal{A}_t}} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix},$$

which clearly have a 1-dimensional nullspace generated by the vector $(0, 1)^T$. This means that solutions to $\overline{\mathcal{A}_t}$ are defined by its first derivative and have the value at $x = 0$ equal to 0.

This means that $\tan(x)$ is the only solution to $\overline{\mathcal{A}_t}$ with $f'(0) = 1$.

Example 3.27 (Continuation from Example 3.7). Now consider $\tilde{\mathcal{A}}_t = \cos(x)\partial_x^2 - 2\sin(x)\partial_x$. In this case we have that the leading coefficient does not vanish at $x = 0$, so we need exactly the value at 0 of the formal power series and its derivative to characterize a particular solution of $\tilde{\mathcal{A}}_t$.

This means that $\tan(x)$ is the only solution to $\tilde{\mathcal{A}}_t$ with $f(0) = 0$ and $f'(0) = 1$.

Chapter 4

Implementation of closure properties

In this Chapter we describe precisely all the algorithms for computing the closure properties described in Chapter 3. We also include some examples on how these algorithms perform in some particular cases. All these results are based on the published article [37].

Throughout this Chapter we assume all operations over the coefficients are implemented. Namely, addition, product and derivation of the coefficients of the differential operators and the differential operators themselves. Moreover, when working with formal power series, we assume we have an algorithm for computing any element of the power series sequence. This operation is represented in the pseudocode by the function `init`:

$$\text{init}(f, i) \longrightarrow f^{(n)}(0).$$

We provide in this Chapter just pseudocode for all algorithms. All the functions that are not included in this Chapter are described (at least the input/output) in the Appendix A.

Note that all algorithms are divided into two main procedures:

- *Obtaining the differential operator*: here we compute a differential operator that annihilates the resulting function.
- *Computing initial conditions*: in the case of working with formal power series, obtain we compute enough initial conditions for the resulting function. Namely, after obtaining the differential operator, we apply Proposition 3.22 and Theorem 3.23 to know exactly how many and which initial conditions we need to compute. Then we use the computations showed in each particular case for computing those initial conditions.

4.1 Closure properties by direct proof

In this first section we include the algorithms related with those operations in Chapter 3 that did not required the use of the characterization Theorem 3.6.

We already showed briefly during the proofs of Propositions 3.9 and 3.8 the direct formulas for the linear equation annihilating the derivative and the integral of a differentially definable function.

4.1.1 Annihilator for the derivative

Let f be a differentially definable function satisfying the differential equation

$$\mathcal{A} \cdot f = r_d \partial^d(f) + \dots + r_1 \partial(f) + r_0 f = 0,$$

and $g = \partial(f)$. In the proof of Proposition 3.9 we distinguished the two cases:

- If $r_0(x) \neq 0$: we build a new linear operator, $\tilde{\mathcal{A}}$, defined by:

$$\tilde{\mathcal{A}} = r_0 \partial \mathcal{A} - \partial(r_0) \mathcal{A},$$

for which we know that $\tilde{\mathcal{A}} \cdot f = 0$ and $[\partial^0] \tilde{\mathcal{A}} = 0$.

- If $r_0(x) = 0$: then the linear operator $\mathcal{B} = r_d \partial^{d-1} + \dots + r_1$ annihilates g .

The algorithm `annihilator_derivative` includes the pseudocode for computing the differential operator annihilating $g = \partial(f)$.

Algorithm 1: `annihilator_derivative`

Input : operator \mathcal{A} annihilating a function f
Output: operator annihilating the function $\partial(f)$
 $r_0 \leftarrow [\partial_0] \mathcal{A};$
if $r_0 = 0$ **then**
 $\mathcal{A} \leftarrow r_0 \partial \mathcal{A} - \partial(r_0) \mathcal{A};$
return $\mathcal{A} // \partial;$

In the particular case of working with formal power series, to compute the corresponding initial conditions for the derivative $g(x) = \partial_x(f(x))$, we only need to compute initial conditions for the function $f(x)$:

$$g^{(d)}(0) = f^{(d+1)}(0) \text{ for all } d \in \mathbb{N}.$$

4.1.2 Annihilator for the integral

Let f be a differentially definable function satisfying the differential equation

$$\mathcal{A} \cdot f = r_d \partial^d(f) + \dots + r_1 \partial(f) + r_0 f = 0,$$

and take $g = \int_x(f)$.

In the proof of Proposition 3.8 we showed that any antiderivative of f is annihilated by the differential operator $\mathcal{A}\partial$. The pseudocode for this operation is included in [annihilator_integral](#).

Algorithm 2: `annihilator_integral`

Input : operator \mathcal{A} annihilating a function f
Output: operator annihilating the function $\int f$
return $\mathcal{A}\partial$;

In the particular case of working with formal power series, we know that the function $g(x)$ is not uniquely defined. We need to specify the first initial value for $g(x)$, i.e., the integration constant. Once we have fixed such constant $c \in \mathbb{K}$, we can create the sequence of initial condition for $g(x) = \int_x f(x)$ using only the values from $f(x)$:

$$g(0) = c, \quad g^{(d)}(0) = f^{(d-1)}(0) \text{ for all } d \geq 1.$$

Example 4.1 (Derivation of the double exponential). Let $g(x) = \partial_x(e^{e^x-1})$. In Example 3.3 we saw that e^{e^x-1} is a formal power series annihilated by the differential operator

$$\mathcal{A} = \partial_x - e^x.$$

We can see that $[\partial_x^0]\mathcal{A} \neq 0$, then [annihilator_derivative](#) computes the operator $\tilde{A} = e^x \partial_x \mathcal{A} - \partial_x(e^x)\mathcal{A}$. More precisely:

$$\begin{aligned} \tilde{A} &= (e^x \partial_x^2 - e^{2x} \partial_x - e^x) - (e^x \partial_x + e^x) \\ &= e^x \partial_x^2 - (e^{2x} + e^x) \partial_x \end{aligned}$$

Hence, $\partial_x(e^{e^x-1})$ is annihilated by the operator \tilde{A}/∂ :

$$\mathcal{B} = \tilde{A}/\partial = e^x \partial_x - (e^{2x} + e^x).$$

Applying Theorem 3.23 over \mathcal{B} guarantees that it is enough to provide the value at zero for determining a particular function annihilated by \mathcal{B} . In the case of $g(x)$, we know that

$$g(0) = (e^{e^x-1})'(0) = (e^x e^{e^x-1})(0) = 1.$$

Thus $g(x)$ is the unique power series annihilated by \mathcal{B} such that $g(0) = 1$.

Example 4.2 (Integral of the tangent). Let $g(x) = \int_0^x \tan(x)dx$. From Example 3.4 we know that $\tan(x)$ is annihilated by the differential operator

$$\mathcal{A}_t = \cos(x)^2 \partial_x^2 - 2.$$

Hence, using `annihilator_integral` we obtain that $g(x)$ is annihilated by the differential operator

$$\mathcal{B} = \mathcal{A}_t \partial = \cos(x)^2 \partial_x^3 - 2\partial_x.$$

Applying Theorem 3.23 over \mathcal{B} we obtain that $g(0)$, $g'(0)$ and $g''(0)$ are required to characterize $g(x)$ as a solution of $\mathcal{B} \cdot g(x) = 0$. Since $g(x) = \int_0^x \tan(x)dx$, we obtain that the integration constant is $c = 0$. For $g'(0)$ and $g''(0)$ we need to compute the first two initial values for the tangent:

$$g'(0) = \tan(0) = 0, \quad g''(0) = \tan'(0) = 1.$$

Thus $g(x)$ is the unique power series annihilated by \mathcal{B} such that $g(0) = 0$, $g'(0) = 0$ and $g''(0) = 1$.

4.2 Closure properties by linear algebra

In this Section we focus on the operations for which we do not have a direct formula for the annihilating operator, but for which we used the characterization Theorem 3.6. Since all these operations rely on the same Theorem, the procedure is very similar for all of them. In fact, most of the actual work is making the implication (iii) \Rightarrow (i) constructive and concrete.

This Section uses extensively the basic results of fraction-free diagonalization of matrices using Bareiss' algorithm [7] and differential linear algebra [48, 70]. All these results are summarized in the Appendix A.

The algorithm described in this Section is a natural extension of the algorithms that are currently used in other implementations of D-finite functions throughout the literature [42, 50]. In this thesis we focus on describing the algorithm for an arbitrary differential integral domain.

This algorithm is also known as an *ansatz method*. In this method, we set up an equation of the desired shape where the coefficients are the unknowns. Then, using all the available knowledge, we translate the differential ansatz equation to a linear system of equations for the unknowns. This allows us to solve our problem using just linear algebra computations.

4.2.1 Description of the ansatz method

Throughout this subsection, let R be a differential integral domain and F its field of fractions. We also consider h to be the function we are interested in. This h

could be the addition of two functions f and g in $D(R)$ or it could be their product. It can also be an algebraic function over R .

The exact meaning of h does not matter at this moment. But, in some way, we have showed that

$$\langle h, \partial(h), \partial^2(h), \dots \rangle \subset W,$$

for some F -vector space W finitely generated. This proves that:

$$\dim_F(\langle h, \partial(h), \dots \rangle) < \infty,$$

and, using Theorem 3.6, $h \in D(R)$.

Assume that $\dim_F(W) \leq d$, then we know that the set $\{h, \partial(h), \dots, \partial^d(h)\}$ is linearly dependent, so we set up the following ansatz:

$$\alpha_0 h + \alpha_1 \partial(h) + \dots + \alpha_d \partial^d(h) = 0, \quad (4.3)$$

where $\alpha_0, \dots, \alpha_d \in R$. The goal of our algorithm is to find a non-trivial solution to this equation. We know the solution exists from the fact that the dimension of W is bounded by d .

Now we need to transform that linear differential equation into a F -linear system for the coefficients α_i and compute an actual non-trivial solution.

Getting a linear system

Assume that $\dim_F(W) \leq d$ and that we know a tuple $\Phi = (\phi_1, \dots, \phi_d)$ of generators of W . This means that $\langle \phi_1, \dots, \phi_d \rangle_F = W$.

Since $\partial^i(h) \in W$ for all $i \in \mathbb{N}$, there are representations in Φ for those derivatives:

$$\partial^i(h) = c_{1,i}\phi_1 + c_{2,i}\phi_2 + \dots + c_{d,i}\phi_d, \quad i = 0, \dots, d. \quad (4.4)$$

Once we have this representation, we can plug it into our ansatz equation:

$$\sum_{i=0}^d \alpha_i \partial^i(h) = \sum_{i=0}^d \sum_{j=1}^d \alpha_i c_{i,j} \phi_j.$$

This double sum can be written with a matrix-vector multiplication:

$$\sum_{i=0}^d \sum_{j=1}^d \alpha_i c_{i,j} \phi_j = \Phi \begin{pmatrix} c_{1,0} & c_{1,1} & \dots & c_{1,d} \\ c_{2,0} & c_{2,1} & \dots & c_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ c_{d,0} & c_{d,1} & \dots & c_{d,d} \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_d \end{pmatrix}.$$

Hence, solving the equation (4.3) is equivalent to solve the linear system

$$\begin{pmatrix} c_{1,0} & c_{1,1} & \dots & c_{1,d} \\ c_{2,0} & c_{2,1} & \dots & c_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ c_{d,0} & c_{d,1} & \dots & c_{d,d} \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_d \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}. \quad (4.5)$$

However, we still need to compute the representations for $\partial^i(h)$. For doing so, we use the results in Appendix A about differential linear algebra.

Assume that W is closed under ∂ , making (W, ∂) a differential vector space over (F, ∂) (see definition A.3). Also assume that M is a *derivation matrix* of ∂ w.r.t. Φ . As shown in equation (A.8) we can now compute derivatives in W using only matrix-vector multiplication and the derivation on F :

$$\partial(\mathbf{v}) = M\mathbf{v} + \kappa_\partial(\mathbf{v}),$$

where $\kappa_\partial((v_1, \dots, v_n)) = (\partial(v_1), \dots, \partial(v_n))$.

Then we would only need a representation of h in terms of Φ and we would be able to compute the linear system (4.5) by a repeated computation of the derivatives of h using the derivation matrix.

Algorithm 3: get_linear_system

Input : vector \mathbf{v}_0 representing h and a derivation M on W w.r.t. Φ
Output: matrix of the linear system (4.5)
 $d \leftarrow \text{ncols}(M);$
for $i = 1, \dots, d$ **do**
 $\mathbf{v}_i = M\mathbf{v}_{i-1} + \kappa(\mathbf{v}_{i-1});$
return $(\mathbf{v}_0 | \dots | \mathbf{v}_d);$

Note that this part of the algorithm is based on the knowledge of W , Φ and the derivation matrix M . All this data is directly dependent on the operation we are computing and is described in detail later.

Solving the system

At this point, we have transformed the ansatz equation (4.3) into the linear system (4.5). Solving this system in R means to compute an element of the right nullspace where all the coefficients are in R . Since the system has more unknowns than equations, we know that there are non-trivial solutions.

One possibility for computing one solution is to work on the field F , compute a vector in the nullspace and then clear denominators, obtaining a vector in the nullspace with all the coefficients in R .

However, the performance of implementing the field of fractions could be costly if the simplification routines in R are not efficient. This is why we focus in a division-free algorithm: Bareiss' algorithm. This algorithm diagonalize the matrix M , making easy the computation of its right nullspace. The subroutines of `non_trivial_solution` are detailed in Appendix A.

Algorithm 4: non_trivial_solution

Input : matrix C of the system (4.5)
Output: a non-trivial vector α solution to the system (4.5)
 $\tilde{C}, (r_1, \dots, r_k), \sigma \leftarrow \text{bareiss}(C);$
 $\mathbf{v}_1, \mathbf{v}_{d-k+1} \leftarrow \text{right_nullspace}(\tilde{C});$
 // At this point, the vectors \mathbf{v}_i need to be rearranged
for $i = 1, \dots, d - k + 1$ **do**
 $\mathbf{v}_i \leftarrow (v_{i,\sigma^{-1}(1)}, \dots, v_{i,\sigma^{-1}(d+1)});$
return $\mathbf{v}_1;$

About initial values

When we look for formal power series solutions, we need to specify some initial values to characterize the particular function $h(x)$ that we want to represent. Once we have computed the linear differential equation for $h(x)$, we apply Proposition 3.22 and Theorem 3.23 to know exactly how many initial conditions for $h(x)$ are needed.

Then, depending on how $h(x)$ was computed, we need to compute these values. This is completely dependent on the operation we are performing.

Summary of the ansatz method

The ansatz method allows us to compute very generically the new differential equation for a function $h \in D(R)$. However, there are some missing elements that depend directly on how h was built. This missing components are:

- A vector space W where $\langle h, \partial(h), \partial^2(h), \dots \rangle_F \subset W$, where $\partial(W) \subset W$.
- A tuple of generators $\Phi = (\phi_1, \dots, \phi_d)$ for W .
- A derivation matrix M of ∂ w.r.t. Φ .

- A vector $\mathbf{v}_0 = (c_{1,0}, \dots, c_{d,0})$ such that

$$h = c_{1,0}\phi_1 + \dots + c_{d,0}\phi_d.$$

- In the case of working with formal power series, a method to compute initial values $h^{(k)}(0)$ for any $k \in \mathbb{N}$ required.

4.2.2 Specifications for the sum

In this subsection we focus on the addition closure property. We assume that $h = f + g$ for two elements $f, g \in D(R)$ where we know differential operators of order d_1 and d_2 respectively. We denote by $V_F(f)$ the F -vector space generated by f and all its derivatives.

The ambient vector space $W_+ = V_F(f) + V_F(g)$ and its generators

In the proof of Proposition 3.10, we saw that $V_F(h) \subset V_F(f) + V_F(g)$. Since the dimensions of $V_F(f)$ and $V_F(g)$ are bounded by d_1 and d_2 , this vector space has finite dimension bounded by $d_1 + d_2$. Also, by the construction of $V_F(f)$ and $V_F(g)$, it is clear that W_+ is closed under ∂ .

Moreover, since $V_F(f)$ is generated by $(f, \partial(f), \dots, \partial^{d_1-1}(f))$ and $V_F(g)$ is generated by $(g, \partial(g), \dots, \partial^{d_2-1}(g))$, then it is also trivial that

$$W_+ = \langle f, \dots, \partial^{d_1-1}(f), g, \dots, \partial^{d_2-1}(g) \rangle_F.$$

Hence the generators Φ_+ of W_+ is just the concatenation of the generators from $V_F(f)$ and $V_F(g)$.

A derivation matrix M_+ w.r.t. Φ_+

First, we start with a very natural lemma relating differentially definable functions, derivation matrices and companion matrices [34]:

Lemma 4.6. *Let (R, ∂) be a differential integral domain, S a differential extension and $f \in D_S(R)$ satisfying the differential equation:*

$$r_0 f + r_1 \partial(f) + \dots + r_d \partial^d(f) = 0.$$

Then the companion matrix of the differential operator, defined as

$$C_f = \begin{pmatrix} 0 & 0 & \dots & 0 & -\frac{r_0}{r_d} \\ 1 & 0 & \dots & 0 & -\frac{r_1}{r_d} \\ 0 & 1 & \dots & 0 & -\frac{r_2}{r_d} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & -\frac{r_{d-1}}{r_d} \end{pmatrix}$$

is a derivation matrix of ∂ w.r.t. $\{f, \partial(f), \dots, \partial^{d-1}(f)\}$.

Proof. By Definition A.7, we only need to check that the derivations of our generators are represented by the columns of \mathcal{C}_f . For the first $d - 2$ columns is trivial, since $\partial(\partial^i(f)) = \partial^{i+1}(f)$. The only generator that we need to check is $\partial^{d-1}(f)$. For this, we use the differential equation of f to conclude:

$$\partial(\partial^{d-1}(f)) = \partial^d(f) = \sum_{i=0}^{d-1} -\frac{r_i}{r_0} \partial^i(f),$$

which are precisely the coefficients we have in the last column of \mathcal{C}_f . □

Using this lemma, we can compute the derivation matrix of ∂ w.r.t. Φ_+ in a very straightforward way. First remark that $W_+ = V_F(f) + V_F(g)$ can be expressed as a quotient

$$V_F(f) + V_F(g) \cong (V_F(f) \oplus V_F(g))/N,$$

where N is the kernel of the map $\varphi : V_F(f) \oplus V_F(g) \rightarrow V_F(f) + V_F(g)$ that maps $(v, w) \mapsto v + w$. Then we can use Lemmas A.12 and A.6 and Proposition A.9 to conclude that

$$M_+ = \mathcal{C}_f \oplus \mathcal{C}_g = \begin{pmatrix} \mathcal{C}_f & 0 \\ 0 & \mathcal{C}_g \end{pmatrix}$$

is a derivation matrix of W_+ w.r.t. Φ_+ .

Intuitively, the derivation matrix is a matrix that provides how to compute derivatives in a vector space. It is natural then that the derivation matrix on the sum of two vector spaces is the sum of the two derivation matrices.

Representation of $h = f + g$ w.r.t. Φ_+

In this case the representation of h is very simple since the generators Φ include both f and g as the first and $(d_1 + 1)$ th element. Hence

$$h = f + g \quad \rightarrow \quad \mathbf{v}_0 = (1, 0, \dots, 0, 1, 0, \dots, 0).$$

Putting together all the procedures and specifications that we have just described, we can provide a complete algorithm (see Algorithm [add_operator](#)) for computing the differential operator that annihilates h .

Initial conditions for formal power series

In the case we are working with formal power series (i.e., $S = \mathbb{K}[[x]]$) we also need to provide a method to compute initial conditions for our function $h(x) = f(x) + g(x)$. Since we know that $f(x), g(x) \in D(R)$, we can compute with the method `init`

Algorithm 5: add_operator

Input : elements $f, g, \in D(R)$ represented by their operators \mathcal{A} and \mathcal{B}
Output: operator \mathcal{L} such that $\mathcal{L} \cdot (f + g) = 0$
 // Getting the specific for addition
 $d_1 \leftarrow \text{order}(\mathcal{A}); d_2 \leftarrow \text{order}(\mathcal{B});$
 $\mathcal{C}_f \leftarrow \text{companion}(\mathcal{A}); \mathcal{C}_g \leftarrow \text{companion}(\mathcal{B});$
 $M_+ \leftarrow \mathcal{C}_f \oplus \mathcal{C}_g;$
 $\mathbf{v}_0 \leftarrow (\delta_{1,i} + \delta_{d_1+1,i} \text{ for } i = 1, \dots, d_1 + d_2);$
 // Getting the solution for the ansatz
 $\alpha \leftarrow \text{non_trivial_solution}(\text{get_linear_system}(\mathbf{v}_0, M_+));$
return $\alpha_0 + \alpha_1 \partial + \dots + \alpha_{d_1+d_2} \partial^{d_1+d_2};$

any initial value for both $f(x)$ and $g(x)$. Recall that the method `init` takes two arguments, namely, the function and the index for the initial value:

$$\text{init}(f, i) \longrightarrow f^{(i)}(0).$$

Computing the value $h^{(n)}(0)$ is trivial due to the linearity of ∂_x with respect to the addition:

$$h^{(n)}(0) = (f(x) + g(x))^{(n)}(0) = f^{(n)}(0) + g^{(n)}(0).$$

This computation is provided in the algorithm `add_init_values`.

Algorithm 6: add_init_values

Input : elements $f(x), g(x), \in D(R)$ and a bound $m \in \mathbb{N}$
Output: first m initial values of $f(x) + g(x)$
return $(\text{init}(f, i) + \text{init}(g, i) \text{ for } i = 0, \dots, m - 1);$

Example 4.7 ([37, Example 3.6]). Consider the functions $f(x) = \exp(\sin(x))$ and $g(x) = \tan(x)$. Both are DD-finite functions annihilated by the following operators:

$$\mathcal{A}_f = \partial_x - \cos(x), \quad \mathcal{A}_g = \cos(x)^2 \partial_x^2 - 2.$$

Now, consider the function $h(x) = f(x) + g(x)$. We know by Proposition 3.10 that $h(x)$ is DD-finite. For applying the method `add_operator`, we first compute the two companion matrices for $f(x)$ and $g(x)$, yielding

$$\mathcal{C}_f = \begin{pmatrix} \cos(x) \end{pmatrix}, \quad \mathcal{C}_g = \begin{pmatrix} 0 & \frac{2}{\cos(x)^2} \\ 1 & 0 \end{pmatrix}.$$

Putting these matrix together for building M_+ we obtain:

$$M_+ = \begin{pmatrix} \cos(x) & 0 & 0 \\ 0 & 0 & \frac{2}{\cos(x)^2} \\ 0 & 1 & 0 \end{pmatrix}.$$

Using this matrix and the vector $\mathbf{v}_0 = (1, 1, 0)$, we can build the linear system computing consecutive derivatives of \mathbf{v}_0 , yielding:

$$\mathcal{S} = \begin{pmatrix} 1 & c & c^2 - s & c^3 - 3sc - c \\ c^3 & 0 & 2c & 4s \\ 0 & c^2 & 0 & 2 \end{pmatrix},$$

where we abbreviate $c = \cos(x)$ and $s = \sin(x)$. We can compute an element in the right nullspace of \mathcal{S} using Bareiss' algorithm and we obtain the vector $\boldsymbol{\alpha} = (\alpha_0, \alpha_1, \alpha_2, \alpha_3)$ where:

$$\begin{cases} \alpha_0 &= 2 \cos(x)^5 - 10 \cos(x)^3 \sin(x) - 6 \cos(x)^3, \\ \alpha_1 &= -2 \cos(x)^4 + 2 \cos(x)^2 \sin(x) + 4, \\ \alpha_2 &= -\cos(x)^7 + \cos(x)^5 + 2 \cos(x)^3 + (3 \cos(x)^5 + 4 \cos(x)) \sin(x), \\ \alpha_3 &= \cos(x)^6 - \cos(x)^4 \sin(x) - 2 \cos(x)^2. \end{cases}$$

Finally, using `add_init_values`, we obtain that the first initial values of $h(x)$ are:

$$h(0) = 1, \quad h'(0) = 2, \quad h''(0) = 1.$$

Hence, we have that $h(x)$ is the unique power series such that

$$\begin{cases} \alpha_0(x)h(x) + \alpha_1(x)h'(x) + \alpha_2(x)h''(x) + \alpha_3(x)h'''(x) = 0 \\ h(0) = 1, \quad h'(0) = 2, \quad h''(0) = 1. \end{cases}$$

4.2.3 Specifications for the product

In this subsection we focus on the product closure property. We assume that $h = fg$ for two elements $f, g \in D(R)$ where we know differential operators of order d_1 and d_2 respectively. We denote by $V_F(f)$ the F -vector space generated by f and all its derivatives.

The ambient vector space $W_* = V_F(f)V_F(g)$, and its generators

In the proof of Proposition 3.10, we saw that $V_F(h) \subset V_F(f)V_F(g)$, which is the vector space generated by the products of elements in $V_F(f)$ and $V_F(g)$. Since the dimensions of $V_F(f)$ and $V_F(g)$ are bounded by d_1 and d_2 , this vector space has finite dimension bounded by $d_1 d_2$. Also, by the construction of $V_F(f)$ and $V_F(g)$ it is clear that W_* is closed under ∂ .

Moreover, since $V_F(f)$ is generated by $(f, \partial(f), \dots, \partial^{d_1-1}(f))$ and $V_F(g)$ is generated by $(g, \partial(g), \dots, \partial^{d_2-1}(g))$, then it is also trivial that

$$W_* = \langle \begin{array}{cccc} fg, & f\partial(g), & \dots, & f\partial^{d_2-1}(g) \\ \partial(f)g, & \partial(f)\partial(g), & \dots, & \partial(f)\partial^{d_2-1}(g) \\ \vdots & \vdots & \ddots & \vdots \\ \partial^{d_1-1}(f)g, & \partial^{d_1-1}(f)\partial(g), & \dots, & \partial^{d_1-1}(f)\partial^{d_2-1}(g) \end{array} \rangle_F$$

Hence the generators Φ_* of W_* is the tensor product of the generators from $V_F(f)$ and $V_F(g)$.

A derivation matrix M_* w.r.t. Φ_*

We use here again Lemma 4.6 to get that the derivation matrices in $V_F(f)$ and $V_F(g)$ are the corresponding companion matrices from the annihilating operators of f and g .

Now, remark that $W_* = V_F(f)V_F(g)$ can be expressed as a quotient

$$V_F(f)V_F(g) \cong (V_F(f) \otimes V_F(g))/N,$$

where N is the kernel of the map $\varphi : V_F(f) \otimes V_F(g) \rightarrow V_F(f)V_F(g)$ that maps $v \oplus w \mapsto vw$. Then we can use Lemmas A.12 and A.6 and Proposition A.10 to conclude that

$$M_* = \mathcal{C}_f \boxplus \mathcal{C}_g = \mathcal{C}_f \otimes \mathcal{I}_{d_2} + \mathcal{I}_{d_1} \otimes \mathcal{C}_g$$

is a derivation matrix of W_* w.r.t. Φ_* .

Intuitively, here we have that the derivation on the product space can be written as the derivation on the first space times the identity on the second space plus the identity on the first space times the derivation on the second. This reminds us to the Leibniz rule but in vector spaces.

Representation of $h = fg$

In this case the representation of h is even simpler than in the addition case: $h = fg$ is precisely the first element in the generators of W_* . Hence,

$$h = fga \rightarrow \mathbf{v}_0 = (1, 0, 0, \dots, 0).$$

Putting together all the procedures and specifications that we have just described, we can provide a complete algorithm (see Algorithm [mul_operator](#)) for computing the differential operator that annihilates h .

Algorithm 7: `mul_operator`

Input : elements $f, g, \in D(R)$ represented by their operators \mathcal{A} and \mathcal{B}
Output: operator \mathcal{L} such that $\mathcal{L} \cdot (fg) = 0$
 // Getting the specific for multiplication
 $d_1 \leftarrow \text{order}(\mathcal{A}); d_2 \leftarrow \text{order}(\mathcal{B});$
 $\mathcal{C}_f \leftarrow \text{companion}(\mathcal{A}); \mathcal{C}_g \leftarrow \text{companion}(\mathcal{B});$
 $M_* \leftarrow \mathcal{C}_f \boxplus \mathcal{C}_g;$
 $\mathbf{v}_0 \leftarrow (\delta_{1,i} \text{ for } i = 1, \dots, d_1 d_2);$
 // Getting the solution for the ansatz
 $\alpha \leftarrow \text{non_trivial_solution}(\text{get_linear_system}(\mathbf{v}_0, M_a));$
return $\alpha_0 + \alpha_1 \partial + \dots + \alpha_{d_1 d_2} \partial^{d_1 d_2};$

Initial conditions for formal power series

In the case we are working with formal power series (i.e., $S = \mathbb{K}[[x]]$) we also need to provide a method to compute initial conditions for our function $h(x) = f(x)g(x)$. Since we know that $f(x), g(x) \in D(R)$, we can compute with the method `init` any initial value for both $f(x)$ and $g(x)$.

Using the Leibniz rule iteratively, computing the value $h^{(n)}(0)$ is just a simple computation on the coefficient level that involve the first coefficients of both $f(x)$ and $g(x)$:

$$h^{(n)}(0) = (f(x)g(x))^{(n)}(0) = \sum_{k=0}^n \binom{n}{k} f^{(n-k)}(0) g^{(k)}(0).$$

This computation is provided in the algorithm [mul_init_values](#).

Algorithm 8: `mul_init_values`

Input : elements $f(x), g(x), \in D(R)$ and a bound $m \in \mathbb{N}$
Output: first m initial values of $f(x) + g(x)$
return $\left(\sum_{k=0}^i \binom{i}{k} \text{init}(f, i-k) \text{init}(g, k) \text{ for } i = 0, \dots, m-1 \right);$

Example 4.8. Let $f(x) = \cos(x)$ and $g(x) = \tan(x)$. The function $f(x)$ is D-finite, so it is in particular DD-finite. On the other hand, $g(x)$ is DD-finite (see Example 3.4). These functions are annihilated by the following linear differential operators:

$$\mathcal{A}_f = \cos(x)\partial_x + \sin(x), \quad \mathcal{A}_g = \cos(x)^2\partial_x^2 - 2.$$

Now, consider the function $h(x) = f(x)g(x)$. We know by Proposition 3.10 that $h(x)$ is DD-finite. For applying the method `add_operator`, we first compute the two companion matrices for $f(x)$ and $g(x)$, yielding

$$\mathcal{C}_f = \begin{pmatrix} -\frac{\sin(x)}{\cos(x)} \end{pmatrix}, \quad \mathcal{C}_g = \begin{pmatrix} 0 & \frac{2}{\cos(x)^2} \\ 1 & 0 \end{pmatrix}.$$

Putting these matrix together for building M_* we obtain:

$$\begin{aligned} M_* &= \mathcal{C}_f \boxplus \mathcal{C}_g = \mathcal{C}_f \otimes \mathcal{I}_2 + \mathcal{I}_1 \otimes \mathcal{C}_g \\ &= \begin{pmatrix} \cos(x) & 0 & 0 \\ 0 & 0 & \frac{2}{\cos(x)^2} \\ 0 & 1 & 0 \end{pmatrix}. \end{aligned}$$

Using this matrix and the vector $\mathbf{v}_0 = (1, 0)$, we can build the linear system computing consecutive derivatives of \mathbf{v}_0 , yielding:

$$\mathcal{S} = \begin{pmatrix} c^2 & -sc & s^2 + 1 \\ 0 & c & -2s \end{pmatrix},$$

where we abbreviate $c = \cos(x)$ and $s = \sin(x)$. We can compute an element in the right nullspace of \mathcal{S} using Bareiss' algorithm and we obtain the vector $\boldsymbol{\alpha} = (\alpha_0, \alpha_1, \alpha_2)$ where:

$$\begin{cases} \alpha_0 &= -\cos(x)^2, \\ \alpha_1 &= 2 \cos(x) \sin(x), \\ \alpha_2 &= \cos(x)^2. \end{cases}$$

Finally, using `mul_init_values`, we obtain that the first initial values of $h(x)$ are:

$$h(0) = 1, \quad h'(0) = 2.$$

Hence, we have that $h(x)$ is the unique power series such that

$$\begin{cases} \alpha_0(x)h(x) + \alpha_1(x)h'(x) + \alpha_2(x)h''(x) = 0 \\ h(0) = 1, \quad h'(0) = 2 \end{cases}$$

4.2.4 Specifications for the algebraic inclusion

In this subsection we focus on Proposition 3.12. We assume that h is algebraic over F with a minimal polynomial:

$$m(h) = r_0 + r_1h + \dots + r_{p-1}h^{p-1} + r_ph^p,$$

where all the coefficients $r_i \in R$.

The ambient vector space W_a and its generators

The key point of the proof in Proposition 3.12 is that we show how $\partial(h) = q(h)$ for some polynomial $q(y) \in F[y]$. Iterating this, we can show that for all $k \in \mathbb{N}$, $\partial^k(h) = q_k(h)$, where

$$q_1(y) = q(y), \quad q_{i+1}(y) = \kappa_y(q_i(y)) + q(y)\partial_y(q_i(y)).$$

Since h is algebraic of degree p , that means that the vector space generated by the power of h has dimension p and, in particular:

$$V_F(h) \subset \langle 1, h, h^2, \dots, h^{p-1} \rangle_F.$$

So the ambient space W_a is the vector space generated by the powers of h and the generators $\Phi_a = (1, h, \dots, h^{p-1})$.

A derivation matrix M_a w.r.t. Φ_a

Computing the derivation matrix of ∂ w.r.t. Φ_a is slightly more complicated than for the addition of multiplication. However, the proof of Proposition 3.12 give the key to do so.

First of all, remark that $\partial(h^k) = kh^{k-1}\partial(h)$. This means that, in order to compute the k th column of M_a , we need to compute $\partial(h)$ as a polynomial in h , then multiply by h^{k-1} and reduce it using the minimal polynomial $m(h)$ to get the final polynomial in h with bounded degree.

Let us begin computing $\partial(h)$. Consider the polynomials:

$$\kappa(m)(y) = \partial(r_0) + \partial(r_1)y + \dots + \partial(r_p)y^p,$$

$$\partial_y(m)(y) = r_1 + 2r_2y + \dots + pr_py^{p-1}.$$

Then it is clear that for any $f \in S$:

$$\partial(m(f)) = \kappa(m)(f) + \partial(f)\partial_y(m)(f).$$

In particular for $f = h$, since $m(h) = 0$, we obtain:

$$\kappa(m)(h) + \partial(h)\partial_y(m)(h) = 0. \quad (4.9)$$

By definition, m is irreducible, so m and $\partial_y(m)$ are coprime, meaning that there are $r(y), s(y) \in F[y]$ such that

$$r(y)m(y) + s(y)\partial_y(m)(y) = 1.$$

These three polynomials can be computed using the *extended Euclidean algorithm*. Now if we evaluate at $y = h$, we obtain:

$$s(h)\partial_y(m)(h) = 1.$$

Hence, if we multiply (4.9) by $s(h)$ then we obtain:

$$\partial(h) = -s(h)\kappa(m)(h).$$

Then we can reduce this polynomial $-s(y)\kappa(m)(y)$ using $m(y)$ to compute the representation of $\partial(h)$ only with the generators Φ_a .

Once we have this representation, computing the derivatives of the other elements in Φ_a can be easily done in an iterative way:

$$\partial(h^n) = \frac{n}{n-1}h\partial(h^{n-1}).$$

This particular formulation is very interesting for reducing the resulting polynomial. Since we are only multiplying each time by h , the degree can not grow further than p , making necessary only one trivial substitution to compute the reduction.

More precisely, assume that $v = \alpha_0 + \dots + \alpha_{p-1}h^{p-1}$. If we want to compute hv with respect to the set Φ_a , we have:

$$hv = \alpha_0h + \dots + \alpha_{p-2}h^{p-1} - \sum_{i=0}^{p-1} \frac{\alpha_{p-1}r_i}{r_d}h^i = -\frac{\alpha_{p-1}r_0}{r_d} + \sum_{i=1}^{p-1} \alpha_{i-1} - \frac{\alpha_{p-1}r_i}{r_d}h^i.$$

This process can be written again with a companion matrix. In fact, given $v = (\alpha_0, \dots, \alpha_{p-1})$, we can compute the representation of hv with the matrix-vector multiplication:

$$\begin{pmatrix} 0 & 0 & \dots & 0 & -\frac{r_0}{r_p} \\ 1 & 0 & \dots & 0 & -\frac{r_1}{r_p} \\ 0 & 1 & \dots & 0 & -\frac{r_1}{r_p} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & -\frac{r_{p-1}}{r_p} \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{p-1} \end{pmatrix} = \mathcal{C}_m v^T$$

We can see in [algebraic_derivation_matrix](#) the pseudocode for building the derivation matrix M_a following the ideas we have just described.

Algorithm 9: algebraic_derivation_matrix

Input : irreducible polynomial $m \in F[y]$ such that $m(h) = 0$
Output: derivation matrix of ∂ w.r.t. Φ_a
 $r, s \leftarrow \text{extended_euclidean}(m, \partial_y(m));$
 $\text{der} \leftarrow (s\kappa(m))\%m;$
 $\mathbf{v}_1 \leftarrow ([y^i](\text{der}) \text{ for } i = 0, \dots, p-1);$
 $\mathcal{C}_m \leftarrow \text{companion}(m)$
for $i = 2, \dots, p-1$ **do**
 $\mathbf{v}_i = \frac{i}{i-1} \mathcal{C}_m \mathbf{v}_{i-1};$
return $(0|\mathbf{v}_1| \dots |\mathbf{v}_{p-1});$

Representation of h

Once again, h belong to the generator set. In this case h is precisely the second element in the generators of W_* . Hence,

$$h \rightarrow \mathbf{v}_0 = (0, 1, 0, \dots, 0).$$

Putting together all the procedures that we have just described, we can provide a complete algorithm (see Algorithm [algebraic_operator](#)) for computing the differential operator that annihilates h .

Algorithm 10: algebraic_operator

Input : irreducible polynomial $m \in F[y]$ such that $m(h) = 0$
Output: operator \mathcal{L} such that $\mathcal{L} \cdot (h) = 0$
// Getting the specific for algebric function
 $M_a \leftarrow \text{algebraic_derivation_matrix}(m);$
 $\mathbf{v}_0 \leftarrow (\delta_{2,i} \text{ for } i = 1, \dots, p);$
// Getting the solution for the ansatz
 $\alpha \leftarrow \text{non_trivial_solution}(\text{get_linear_system}(\mathbf{v}_0, M_a));$
return $\alpha_0 + \alpha_1 \partial + \dots + \alpha_p \partial^p;$

Initial conditions for formal power series

Computing the initial conditions for algebraic power series is equivalent to compute the Taylor expansion at $x = 0$. Note that only usually only $h(0)$ is needed to determine the whole power series. But this is not always the case.

In this thesis we propose a simple approach that work in many cases but it is not complete. Recall from the description of algorithm 9 that we can write

$$h'(x) = -s(x, h(x))\kappa(m)(h(x)) = P(x, h(x)),$$

where $P(x, y) \in \mathbb{K}(x)[y]$. Assume that all the coefficients are defined at $x = 0$. Then we only need a value for $h(0)$ to define any derivative of $h(x)$ at $x = 0$.

In case that the value for $h(0)$ is not compatible with the algebraic equation or any of these coefficients are not well defined at $x = 0$, we do not provide a method for computing the initial values of $h(x)$.

Example 4.10 ([39, Example 7]). Let $h(x)$ be given by the algebraic equation

$$\frac{1}{2} \cos(x)h(x)^2 - 2h(x) + \cos(x)^2 = 0,$$

where all coefficients are D-finite. We know that $h(x)$ is DD-finite.

In order to build the derivation matrix M_a , we need to apply first algorithm `algebraic_derivation_matrix`. Here we consider the two polynomials

$$m(x, y) = \frac{1}{2} \cos(x)y^2 - 2y + \cos(x)^2, \quad m_y(x, y) = \cos(x)y - 2.$$

Using the extended Euclidean algorithm, we find $r(x, y), s(x, y), t(x, y)$ such that

$$r(x, y)m(x, y) + s(x, y)m_y(x, y) = t(x, y).$$

These three polynomials are:

$$r(x, y) = 2 \cos(x), \quad s(x, y) = 2 - y \cos(x), \quad t(x, y) = 2 \cos(x)^3 - 4.$$

In this case, this leads to the non-homogeneous equation for $h(x)$:

$$2 \cos(x) (\cos(x)^3 - 2) h'(x) + \sin(x) (\cos(x)^3 + 4) h(x) = 6 \sin(x) \cos(x)^2.$$

From here it is easy to get the DD-finite equation for $h(x)$:

$$\alpha_2 h''(x) + \alpha_1 h'(x) - \alpha_0 h(x) = 0,$$

where the values of the coefficients are:

$$\begin{cases} \alpha_2 = 2 \sin(x) \cos(x)^2 (\cos(x)^3 - 2) \\ \alpha_1 = \cos(x) (3 \cos(x)^5 - 5 \cos(x)^3 + 4) \\ \alpha_0 = 2 \sin(x)^3 (\cos(x)^3 - 2) \end{cases}$$

4.3 Performance considerations

In the previous Sections, we described the algorithms that implements the addition and multiplication for differentially definable functions, i.e., given differential equations for the operants, these algorithms provide a differential equation for the resulting function. We also saw how to transform an algebraic equation into a linear differential equation.

These operations rely on Theorem 3.6 and the order bounds found in Propositions 3.10 and 3.12. In our approach, we never try to get the smallest differential equation. This means that the equation obtained has usually the actual bound as order.

This would not be a huge problem in general if operations on the ring R were fast. However, when we consider DD-finite functions, we have that $R = D(\mathbb{K}[x])$. The operations performance over D-finite functions depends directly on the order of the differential equations that represent the operands. Hence, this approach can lead to a huge computational cost in small computations.

As an illustrative example, consider we want to add an order 2 DD-finite function and an order 1 DD-finite function. Following the algorithm `add_operator`, we would have:

$$\mathcal{C}_f = \begin{pmatrix} 0 & -\frac{r_0}{r_2} \\ 1 & -\frac{r_1}{r_2} \end{pmatrix}, \quad \mathcal{C}_g = \begin{pmatrix} -\frac{s_0}{s_1} \end{pmatrix},$$

and the matrix of which we need an element in the nullspace is:

$$\begin{pmatrix} 1 & 0 & -\frac{r_0}{r_2} & \frac{r_0 r_1}{r_2^2} - \left(\frac{r_0}{r_2}\right)' \\ 0 & 1 & -\frac{r_1}{r_2} & -\frac{r_0}{r_2} + \frac{r_1^2}{r_2^2} - \left(\frac{r_1}{r_2}\right)' \\ 1 & -\frac{s_0}{s_1} & \frac{s_0^2}{s_1^2} - \left(\frac{s_0}{s_1}\right)' & -\frac{s_0^3}{s_1^3} + 3\frac{s_0}{s_1} \left(\frac{s_0}{s_1}\right)' - \left(\frac{s_0}{s_1}\right)'' \end{pmatrix}. \quad (4.11)$$

Let assume *the simplest* situation where r_0, r_1, r_2, s_0 and s_1 have all order 1 as D-finite functions. In this case, after clearing denominators in matrix (4.11) we end up with a matrix of D-finite functions with the following orders:

$$\begin{pmatrix} 1 & 0 & 1 & 3 \\ 0 & 1 & 1 & 4 \\ 1 & 1 & 3 & 7 \end{pmatrix}.$$

We do more multiplications and additions while computing the right nullspace of matrix 4.11 (see method `non_trivial_solution` for further information). In the end, we obtain a matrix with the following orders:

$$\begin{pmatrix} 5 & 0 & 0 & 29 \\ 0 & 5 & 0 & 34 \\ 0 & 0 & 5 & 14 \end{pmatrix},$$

which leads to a solution vector may have orders $(145, 170, 70, 5)$. This is a very important issue that we need to consider in order to make operations with DD-finite functions effective.

There are two ways of keeping the order smaller:

- Simplify intermediate results: every time we do an operation that may increase the order of an elements (i.e., each addition and product) we try to simplify the result. Since the order we usually obtain is the actual bound, this idea can produce an important reduction of the size on the output.
- Perform lazy computations: we work with the coefficients as symbols, postponing the computations until they are required. This helps to keep track on the relations between coefficients (as some divisibility properties).

Of course, these two approaches do not always succeed in keeping the order smaller. The bounds for addition and multiplication are sharp, meaning that there are particular examples where the bound is reached.

Simplification of intermediate results

There are two main ways for simplify the results of the operations:

- Compute smaller annihilating operators: in our algorithms we never considered looking for smaller equations when computing the closure properties. Since the bounds provided by Proposition 3.10 are the worst possible case, we can tweak the algorithm to look for smaller operators first.

In order to do so, we can simply consider ansatz of smaller size and look for solutions. Independently of the size of the ansatz, the problem is translated into linear algebra in the same way we showed in our algorithms. However, if the ansatz is too small we have no guarantee that a solution exists.

This approach increases the cost of one computation, but it reduces future operations with its result. Moreover, the consecutive ansatz are closely related, so we can reuse computations from onw system to the following.

For D-finite functions, this was already done in the package `ore_algebra` [42] for Sage. Since the operations here are based on polynomials, the cost of looking for solutions of linear systems is not high in comparison with the actual operations we are performing. We will make use of this in our implementation.

- Reducing the order of the final solutions: once we have an annihilating operator for the result of our operation, we have no guarantee that operator is the smallest annihilating that function. If we can compute a factorization of

the annihilating operator, we may be able to simplify the equation for one particular function even further.

In the case of D-finite functions this is possible in some cases, but is topic of current research [71, 72]. This is not a simple operation anyway and it may have a high cost. This is why we did not consider it in our implementation.

Lazy computations

Performing lazy computations means to delay the actual computations as much as possible and only perform these operations when they are strictly needed. In the meantime, we keep track of the expressions using variables for distinguishing between different functions.

This allows us to realize some relations between the elements we are considering. For example, in the matrix (4.11), we see that all components of the matrix are just polynomial expressions of the coefficients and their derivatives. Let us work out that particular example a bit more.

Example 4.12. Let $f(x), g(x) \in D^2(\mathbb{K}[x])$ with differential equations

$$f''(x) - \cos(x)f(x) = 0, \quad g'(x) - (\sin(x) - 1)g(x) = 0.$$

In this example we have two coefficients, $\cos(x)$ and $(\sin(x) - 1)$, that are D-finite and are annihilated by the differential operators

$$\partial_x^2 + 1, \quad \partial_x^3 + \partial_x,$$

respectively. We know that these functions are closely related, but during a normal execution of our code, the system will only know these differential operators. Let $r(x)$ denote $\cos(x)$ and $s(x)$ denote $\sin(x) - 1$. Hence, the companion matrices for $f(x)$ and $g(x)$ are

$$\begin{pmatrix} 0 & r \\ 1 & 0 \end{pmatrix}, \quad \begin{pmatrix} s \end{pmatrix}$$

and the matrix for which we need an element in the right nullspace is:

$$\begin{pmatrix} 1 & 0 & r & r' \\ 0 & 1 & 0 & r \\ 1 & s & s' + s^2 & s'' + 3ss' + s^3 \end{pmatrix}.$$

After applying Bareiss' algorithm to this matrix blindly, we have:

$$\begin{pmatrix} t & 0 & 0 & tr' - (s'' + 3ss' + s^3 - r' - sr)r \\ 0 & t & 0 & r \\ 0 & 0 & t & (s'' + 3ss' + s^3 - r' - sr) \end{pmatrix},$$

where $t = s^2 + s' - r$, and then we can find a non-trivial element in the right nullspace:

$$\left(tr' - (s'' + 3ss' + s^3 - r' - sr)r, r, s'' + 3ss' + s^3 - r' - sr, t \right).$$

Computing these elements leads to a very high order in the coefficients. These elements may have order up to $(90, 2, 47, 14)$. But we can do better with a simple observation:

$$s'(x) = r(x), \quad s''(x) = r'(x). \quad (4.13)$$

This leads to the following vector solution:

$$\left(s(sr' - 2r + s^2), r, s(s^2 - 2r), -s^2 \right),$$

where the orders are clearly smaller: $(63, 2, 33, 9)$.

We can do even better with the identities:

$$s(x) = -r'(x) - 1, \quad s'(x) = -r''(x) = r(x). \quad (4.14)$$

Hence, the following vector is also in the desired right nullspace:

$$\left(-(r' + 1)(r' - 2r + 1), r, -(r' + 1)(r'^2 + 1), -r'^2 - 2r - 1 \right)$$

which now leads to functions of order up to $(15, 2, 15, 7)$.

This example shows how recognizing simple relations between the coefficients can lead to easier expressions, reducing not only the order of the output but also the time needed to the actual computation.

This example was very particular and we use *ad hoc* properties to simplify the computations. In order to implement a system with lazy computations, we need to consider the following questions:

- What new variables do we add? It seems clear that we need to add more variables when we find *new* coefficients on the differential equations. We also their derivatives.
- How many simplifications do we make? This question has several answers that will impact directly the performance of the system:
 - We can do no simplifications: we will have as many variables as possible, making the inclusion of new variables very fast. However, the results will have bigger order. This implies a slower computation in the end.

- We can do all possible simplifications: if we compute all possible relations between our variables, we can simplify the final output as much as possible. This takes a lot of time. In fact, finding algebraic relations between D-finite objects is a very difficult task [45]. But the result will be as small as possible.
- We can check that we do not add repeated variables. This implies just checking identity between the coefficients and their derivatives. In Example 4.12 this checking is precisely observation (4.13).
- We can check whether two variables are linearly related. This means checking for two functions r and s if there are constants α, β such that

$$r = \alpha s + \beta.$$

In case that equality holds, α and β are easy to compute. This means that checking this relation is (computationally speaking) as expensive as simply checking $r = s$. In Example 4.12, this is precisely observation (4.14).

Moreover, while applying Bareiss' algorithm we need to check that the pivot we chose are not zero. We can use this extra computation in our favor to reduce even more our output: if we find that an expression has value zero, we add it to a list of elements that vanishes. Then, after every operation, we reduce all polynomial expressions using the Gröbner basis of the ideal generated by this list.

Example 4.15. Let $f(x), g(x) \in D^2(\mathbb{K}[x])$ with differential equations

$$f''(x) - (1 - \sin(x) - \sin(x)^2)f(x) = 0, \quad g'(x) - (\cos(x))g(x) = 0.$$

In this example we have two coefficients that are D-finite annihilated by the differential operators

$$\partial_x^5 + 5\partial_x^3 + 4\partial_x, \quad \partial_x^2 + \partial_x.$$

Let $r(x) = (1 - \sin(x) - \sin(x)^2)$ and $s(x) = \cos(x)$. When we look for a pivot in the third iteration of Bareiss' algorithm, we find that

$$s' + s^2 - r = -\sin(x) + \cos(x)^2 - 1 + \sin(x) + \sin(x)^2 = 0.$$

Using this relation to reduce the last entry of the matrix, we get:

$$s'' + 3ss' + s^3 - r' - sr = -s + 2ss' - r'.$$

This by itself reduces the size of the entries of the matrix. Moreover, in this particular example:

$$-s + 2ss' - r' = -\cos(x) - 2\sin(x)\cos(x) + \cos(x) + 2\sin(x)\cos(x) = 0,$$

which makes that after Bareiss' algorithm we end up with the matrix:

$$\begin{pmatrix} 1 & 0 & r & r' \\ 0 & 1 & 0 & r \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

and we find that the vector $(-r, 0, 1, 0)$ is in the right nullspace showing that:

$$(f(x) + g(x))'' - r(x)(f(x) + g(x)) = 0,$$

where the orders of the coefficients are now $(5, 0, 1)$.

Final implementation

In the end, we have described several ideas to keep the result as small as possible. In the following paragraphs we describe precisely the decisions we took in the implementation of DD-finite functions.

In our system we use the approach of lazy computations. For this operations, we use a set of variables x_1, x_2 , etc. where we add more variables when they are needed. Let $f_i(x) \in \mathbb{K}[[x]]$ be the real function represented by x_i . Assume that we have used n variables. We have a list of algebraic relations $(q_1(x_1, \dots, x_n), \dots, q_l(x_1, \dots, x_n))$ that we know from previous computations that $q_i(f_1(x), \dots, f_n(x)) = 0$. Let I be the ideal generated by these polynomials and G a Gröbner basis for this ideal.

Then, we start the construction of the matrix M for which we need to compute an element in the right nullspace:

- If we consider a new coefficient $f(x)$ of a DD-finite function, we **check for each variable** x_i if $f(x) = \alpha f_i(x) + \beta$ for some constants α and β . If no relation is found, then we add a new variable x_{n+1} and we set $f_{n+1}(x) = f(x)$. Otherwise we substitute $f(x)$ by $\alpha f_i(x) + \beta$ whenever $f(x)$ occurs.
- If we need to compute a derivative for $f_k(x)$, **we check for each variable** x_i if $\partial_x(f_k(x)) = \alpha f_i(x) + \beta$. If no relation is found, we add a new variable x_{n+1} and we set $f_{n+1}(x) = \partial_x(f_k(x))$. Moreover, we define $\partial(x_k) = x_{n+1}$. In the case a relation is found, then we set $\partial(x_k) = \alpha x_i + \beta$.
- For every polynomial that we compute, we reduce it using G .

At this stage, we need to compute a vector in the right nullspace of a matrix M where all entries are polynomials in $\mathbb{K}[x_1, \dots, x_n]$ reduced by G . We use the algorithm `non_trivial_solution` with the following tweaks:

- While looking for a pivot in `bareiss`, we check if $p(x_1, \dots, x_n)$ is zero or not. In that case, we compute $P = p(f_1(x), \dots, f_n(x))$ **performing the required closure properties** on the D-finite level. If $P = 0$, we add $p(x_1, \dots, x_n)$ to the list of algebraic relations and recompute I and G . We reduce again all entries of M . Otherwise, we have found a pivot and we can move to the next step of Bareiss' algorithm.
- Once we have the element α in the right nullspace of M , we **apply the closure properties** on the D-finite level for the entries of α .

In this process we have **marked with bold font** all the moments that we actually compute with D-finite elements. As we can see, we avoid these computations as much as possible.

We also keep the set of variables through one session of our code, meaning that if we find nice relations between two functions in one operation, we will use those simplifications again if we find those functions again.

Chapter 5

Structural results

In Chapter 3 we studied the main properties of DD-finite functions, showing that this larger class of formal power series satisfy similar closure properties that the classical D-finite functions. However, we kept sure that we stated the theory using a generic context, using differential integral domains instead of classes of formal power series (like polynomials or D-finite functions).

In this Chapter we see how can we use this generality to study and understand further the properties of differentially definable functions. We use different tools (like differential Galois theory) and relate these objects to an even bigger (but more complicated to use on the computer) class of functions: the differentially algebraic class. This Chapter is inspired in the results published on [8] and [39].

5.1 D^n -finite functions

In Section 3.1 we defined the set of *differentially definable* functions over a differential integral domain R within an ambient domain S . Putting together the results in Propositions 3.9 and 3.10 we can easily see that $D_S(R)$ is always a differential integral domain. This allows us (in the same way we did with D-finite functions) to iterate the process.

From Lemma 3.2, we obtain a chain of differential rings:

$$R \subset D_S(R) \subset D_S(D_S(R)) = D_S^2(R) \subset D_S^3(R) \subset \dots \subset S$$

Definition 5.1. Let (R, ∂) be a differential integral domain and (S, ∂) a differential extension. We call *differentially definable closure of R over S* (and denote it by $D_S^\infty(R)$) to the direct limit of the differential integral domains $D_S^n(R)$, i.e.,

$$D_S^\infty(R) = \bigcup_{n \in \mathbb{N}} D_S^n(R).$$

This new differential integral domain, closely related to the Picard-Vessiot closure (see Appendix B) is the biggest class of functions within S that can be represented using linear differential equations starting with coefficients in R .

In the particular case where $S = \mathbb{K}[[x]]$ and $R = \mathbb{K}[x]$, we call $D^n(\mathbb{K}[x])$ the D^n -finite functions. These classes and their limit, $D^\infty(\mathbb{K}[x])$, are the main objects of study in this Chapter.

Probably one of the most interesting results concerning these new classes is that, now that we consider the whole chain of functions, we can compute the composition of formal power series within this chain:

Theorem 5.2 ([39, Theorem 10]). *Let $f(x) \in D^n(\mathbb{K}[x])$ and $g(x) \in D^m(\mathbb{K}[x])$ with $g(0) = 0$ for some $n, m \in \mathbb{N}$. Then $f(g(x)) \in D^{n+m}(\mathbb{K}[x])$ with order at most d .*

Proof. Using Lemma 2.6, we know that $h(x) = f(g(x)) = (f \circ g)(x)$ is a formal power series since $g(0) = 0$. We now proceed by induction on n : let $n = 0$, then $f(x)$ is a polynomial and $f(g(x))$ is an polynomial expression of D^m -finite functions. Using the corresponding closure properties from Proposition 3.10, we conclude that $h(x) \in D^m(\mathbb{K}[x])$.

Let now be $n > 0$ and assume that, for all $k < n$ and all $r(x) \in D^k(\mathbb{K}[x])$, the composition $r(g(x)) \in D^{k+m}(\mathbb{K}[x])$. For proving that $h(x) = f(g(x))$ is D^{n+m} -finite, we are going to use Theorem 3.6 and study the dimension of the vector space $V(h) = \langle h, h'(x), h''(x), \dots \rangle_{F_{n+m-1}(x)}$, where $F_n(x)$ will denote the field of fractions of the ring $D^n(\mathbb{K}[x])$.

Using the chain rule iteratively, we can find that

$$h^{(p)}(x) = \sum_{l=1}^p f^{(p)}(g(x)) B_{p,l}(g'(x), \dots, g^{(p-l+1)}(x)),$$

where $B_{p,l}(y_1, \dots, y_{p-l+1})$ are the Bell polynomials. Since $g(x) \in D^m(\mathbb{K}[x])$ and $n > 0$, we know by Lemma 3.2 and Proposition 3.9 that

$$B_{p,l}(g'(x), \dots, g^{(p-l+1)}(x)) \in F_{n+m-1}(x).$$

Hence, it is clear that

$$V(h) \subset \langle f(g(x)), f'(g(x)), \dots \rangle_{F_{n+m-1}(x)}.$$

Now, since $f(x) \in D^n(\mathbb{K}[x])$ with order d , there are elements $r_0(x), \dots, r_d(x) \in D^n(\mathbb{K}[x])$ such that

$$r_d(x)f^{(d)}(x) + \dots + r_1(x)f'(x) + r_0(x)f(x) = 0.$$

If we compose both sides with $g(x)$ we obtain

$$f^{(d)}(g(x)) = \frac{-1}{r_d(g(x))} \left(r_0(g(x))f(g(x)) + \dots + r_{d-1}(g(x))f^{(d-1)}(g(x)) \right),$$

and by induction hypothesis, we know that $r_i(g(x)) \in D^{n+m-1}(\mathbb{K}[x])$, so we have

$$f^{(d)}(g(x)) \in \langle f(g(x)), f'(g(x)), \dots, f^{(d-1)}(g(x)) \rangle_{F_{n+m-1}(x)}.$$

Similarly, since $f^{(p)}(x) \in \langle f(x), \dots, f^{(d-1)}(x) \rangle_{F_{n-1}(x)}$, we have that for all $p \geq d$ we have:

$$f^{(p)}(g(x)) \in \langle f(g(x)), f'(g(x)), \dots, f^{(d-1)}(g(x)) \rangle_{F_{n+m-1}(x)},$$

showing that $\dim(V(h)) \leq d$. By Theorem 3.6, $f(g(x)) \in D^{n+m}(\mathbb{K}[x])$ with order at most d . \square

Example 5.3. The following functions are DD-finite:

- $\exp(\exp(x) - 1)$,
- $\sin(\cos(x) - \exp(x))$,
- ${}_2F_1 \left(\begin{matrix} a, -a \\ 1/2 \end{matrix} ; \frac{1}{2}(1 - \cos(x)) \right)$, for any $a \in \mathbb{C}$.

Since the proof of Theorem 5.2 uses the Characterization Theorem 3.6 (in the same way that we proved Proposition 3.10), the algorithm to actually compute the differential equation goes in a very similar way (see Chapter 4).

Note that Theorem 5.2 only provides an upper bound for where the composition $f(g(x))$ will belong. However, it is pretty clear that there are cases where this bound is not reached: we know that $\exp(x) - 1$ and $\ln(x + 1)$ are both D-finite functions, so its composition is, by this result, DD-finite. But it is also clear (since both functions are compositional inverses) that the composition was indeed a polynomial (i.e., it belongs to $D^0(\mathbb{K}[x])$).

Knowing more precisely where a composition belongs is very important regarding performance of possible operations. Since the D^n -finite functions are built recursively, all algorithms will work recursively too. This means that knowing the ring where a composition belongs can reduce almost exponentially the performance for further operations with this function.

This is why we present here an extension of Theorem 2.14, where we can improve the bound for the composition of two functions $f(x), g(x)$ if we have extra algebraic information:

Theorem 5.4 ([39, Theorem 8]). *Let $f(x) \in D^n(\mathbb{K}[x])$ some $n \geq 1$ and $a(x)$ be algebraic over $F_m(x)$ (the field of fractions of $D^m(\mathbb{K}[x])$) for some $m \geq 0$ with $a(0) = 0$. Then $h(x) = f(a(x))$ is in $D^{n+m}(\mathbb{K}[x])$.*

Before jumping into the proof of this Theorem, let remark that this is indeed a generalization of Theorem 2.14 and Proposition 3.12: for the first, the classical result have a D-finite function $f(x)$ and an algebraic function (i.e., algebraic over $D^0(\mathbb{K}[x])$). Then Theorem 5.4 gives directly the classical result. In the case of Proposition 3.12, if we consider an algebraic function $a(x)$ over $F_{n-1}(x)$, we can express it as x composed with $a(x)$. Since x is D-finite, then Theorem 5.4 confirms that $a(x) \in D^n(\mathbb{K}[x])$.

Proof of Theorem 5.4. Let $f(x) \in D^n(\mathbb{K}[x])$ be of order d and $a(x)$ be algebraic of degree p over $F_m(x)$.

We proceed by induction on n . The base case $n = 1$ (i.e., when $f(x)$ is D-finite) can be proven using the same ideas used in Theorem 2.14. As we did in Theorem 5.2, we can write $h^{(k)}(x)$ as a linear combination of $f^{(l)}(a(x))$ with coefficients in $\mathbb{K}[a(x), a'(x), \dots]$. Since $a(x)$ is algebraic over $F_m(x)$, we have that all the derivatives of $a(x)$ are in the vector space $\langle a(x), \dots, a(x)^{p-1} \rangle_{F_m(x)}$, so we can write

$$h^{(k)}(x) = \sum_{l=1}^k Q_{k,l}(x, a(x)) f^{(l)}(a(x)),$$

where $Q_{k,l}(x, a(x)) \in F_m(x)(a) = F_m(x)[a, a^2, \dots, a^{p-1}]$.

On the other hand, since $f(x)$ is D-finite, there are polynomials $p_0(x), \dots, p_d(x)$ such that

$$p_d(x) f^{(d)}(x) + \dots + p_0(x) f(x) = 0.$$

If we compose here with $a(x)$, we obtain right away that

$$\dim(\langle f(a(x)), f'(a(x)), \dots \rangle_{F_m(x)(a)}) \leq d.$$

Altogether, we have

$$\langle h(x), h'(x), \dots \rangle_{F_m(x)} \subset \langle f(a(x)), f'(a(x)), \dots, f^{(d-1)}(a(x)) \rangle_{F_m(x)(a)}.$$

Since $F_m(x)(a)$ is finite over $F_m(x)$, the vector space spanned by $h(x)$ and its derivatives over $F_m(x)$ will have then a finite dimension (at most dp). Hence $h(x) \in D^{m+1}(\mathbb{K}[x])$.

Now, let $n > 1$. This part of the proof is very similar to the induction case on Theorem 5.2. Since $f(x) \in D^n(\mathbb{K}[x])$, there are elements $r_0(x), \dots, r_d(x)$ such that

$$r_d(x) f^{(d)}(x) + \dots + r_0(x) f(x) = 0.$$

Using the induction hypothesis, we can compose this identity with $a(x)$ obtaining a linear relation for $f^{(l)}(a(x))$ with coefficients in $D^{n+m-1}(\mathbb{K}[x])$. Moreover, since $a(x)$ is algebraic over $F_m(x)$, all its derivatives can be written as a linear combination of $a(x), a(x)^2, \dots, a(x)^{p-1}$, so all the derivatives of $h(x)$ can be written as a linear combination over $F_{n+m-1}(x)$ of products of $f^{(l)}(a(x))$ and $a(x)^t$

$$h^{(k)}(x) \in \langle f(a(x)), f'(a(x)), \dots, f^{(d-1)}(a(x)) \rangle \otimes \langle 1, a(x), \dots, a(x)^{p-1} \rangle,$$

concluding that

$$\dim(\langle h(x), h'(x), \dots \rangle_{F_{n+m-1}(x)}) \leq dp,$$

yielding $h(x) \in D^{n+m}(\mathbb{K}[x])$. \square

5.2 Increasing structure of the D^n -finite chain

In the previous Section we have introduced the chain of D^n -finite functions starting from the ring of polynomials $\mathbb{K}[x]$ and applying iteratively the construction of differentially definable functions over it. We also studied the behavior of the composition of formal power series over this chain.

But in Theorem 5.4 we have shown that we do not always need to reach the bound granted by Theorem 5.2. In fact, we have not provided any example of a formal power series that is, for sure, D^3 -finite but not DD-finite.

In this Section, we use differential Galois theory [25, 70] to show that, starting with the polynomial ring $\mathbb{K}[x]$ the chain of D^n -finite functions is strictly increasing, i.e., $D^n(\mathbb{K}[x]) \subsetneq D^{n+1}(\mathbb{K}[x])$ for all $n \in \mathbb{N}$.

If we look to the examples that we have currently shown, we have two main examples that are DD-finite and not D-finite: the double exponential function e^{e^x-1} (see Corollary 2.23.1) and the tangent function $\tan(x)$ (see Corollary 2.24.2). The argument for the tangent does not seem easy to extend, since it already has an infinite amount of singularities.

However, we may try to extend the argument for the double exponential and generalize for a more general class of functions: the iterated exponentials.

Definition 5.5. We defined the *iterated exponentials* in a recursive manner:

- We set $e_0(x) = 1$.
- Given $i \in \mathbb{N}$, we define $\hat{e}_i = \int_0^x e_i(x) dx$.
- We define $e_{i+1}(x) = \exp(\hat{e}_i(x))$.

If we look to the first three iterated exponentials we can see that we get

$$e_0(x) = 1, \quad e_1(x) = e^x, \quad e_2(x) = e^{e^x-1},$$

which are the two exponentials that we know are, respectively, D-finite and DD-finite. It is pretty simple to prove that $e_n(x) \in D^n(\mathbb{K}[x])$:

Lemma 5.6. *For all $n \in \mathbb{N}$, the n th iterated exponential is D^n -finite.*

Proof. The case $n = 0$ is trivial since $1 \in \mathbb{K} \subset \mathbb{K}[x]$. Now, suppose that $e_n(x) \in D^n(\mathbb{K}[x])$. By Proposition 3.8, we know that $\hat{e}_n(x) \in D^n(\mathbb{K}[x])$ as well.

Then we can apply Theorem 5.2 to $\exp(x)$ and $\hat{e}_n(x)$, showing that $e_{n+1}(x) \in D^{n+1}(\mathbb{K}[x])$. In fact, we can check that

$$e'_{n+1}(x) - e_n(x)e_{n+1}(x) = 0.$$

□

We show now that, indeed, $e_n(x) \notin D^{n-1}(\mathbb{K}[x])$ for any $n \in \mathbb{N}$. For doing so, we need differential Galois theory (see Appendix B) and also to consider \mathbb{K} a algebraically closed field (which we will denote with C). This extension is not a problem since our rings $\mathbb{K}[x], D(\mathbb{K}[x]), \dots$ will be directly contained in these extensions $C[x], D(C[x]), \dots$, so we obtain the same conclusion.

We need to consider a bigger chain of fields to apply the corresponding Galois theory, and the construction of these fields is detailed in the literature and in the Appendix B. But a brief intuitive explanation is provided now:

We start with the ring $C[x]$ and consider its field of fractions $F_0 = C(x)$. In the construction of the *Picard-Vessiot closure* (see Definition B.39) of F_0 , we have intermediate fields K_i that are minimal fields with all the possible solutions to linear differential equations with coefficients in K_{i-1} , starting with $K_0 = F_0$. Due to this construction, it is clear that, letting F_i being the field of fractions of $D^i(C[x])$ we have the containment diagram shown below. With this setting, we can prove a stronger statement.

$$\begin{array}{ccccccc} C[x] & \subset & D(C[x]) & \subset & \dots & \subset & D^n(C[x]) & \subset & \dots & \subset & C[[x]] \\ \cap & & \cap & & \ddots & & \cap & & \ddots & & \\ F_0 & \subset & F_1 & \subset & \dots & \subset & F_n & \subset & \dots & \subset & C((x)) \\ \cap & & \cap & & \ddots & & \cap & & \ddots & & \\ K_0 & \subset & K_1 & \subset & \dots & \subset & K_n & \subset & \dots & \subset & K_{PV} \end{array}$$

Lemma 5.7. *Let $c \in C^*$ (i.e., a non-zero constant element) and $n \in \mathbb{N} \setminus \{0\}$. Then $e_n(x)^c = \exp(c\hat{e}_{n-1}(x)) \notin K_{n-1}$.*

Before detailing the proof of Lemma 5.7, we want to remark that this directly proves the result we were looking for: if we specialize $c = 1$, we get that $e_n(x) \notin K_{n-1}$. Then, looking into the containment diagram above, we can see that $e_n(x) \notin F_{n-1}$, meaning that, in particular, $e_n(x) \notin D^{n-1}(C[x])$.

Corollary 5.7.1. *For any $n \in \mathbb{N} \setminus \{0\}$, $e_n(x) \notin D^{n-1}(\mathbb{K}[x])$.*

Proof of Lemma 5.7. This proof proceeds by induction on n . For the base case $n = 1$, we need to show that for any $c \in C^*$, the function $e^{cx} \notin C(x)$. However, we already know by Lemma 2.23 that e^x is transcendental over $C(x)$ so, in particular, it is not a rational function.

Now, let $n > 1$ and assume the hypothesis holds for all $1 \leq i < n$. Suppose there is a $c \in C^*$ such that $e_n(x)^c \in K_{n-1}$. By the construction of the field K_{n-1} (see Appendix B) and since $e_{n-1}(x) \in D^{n-1}(C[x])$, there is a PV-extension E of K_{n-2} that contains both $e_{n-1}(x)$ and $e_n(x)^c$.

Let $u = ce_{n-1}(x)$ and $v = e_n(x)^c$. We can apply now Proposition B.43 and obtain that $ce_{n-1}(x)$ is algebraic over K_{n-2} . By Lemma B.42 we have that there is $m \in \mathbb{N} \setminus \{0\}$ such that $c^m e_{n-1}(x)^m \in K_{n-2}$. But this would mean that $e_{n-1}(x)^m \in K_{n-2}$ contradicting the induction hypothesis (as $m \in \mathbb{N} \subset C$). Thus $e_n(x)^c \notin K_{n-1}$ for any $c \in C^*$. \square

We have proven that the chain of D^n -finite functions does not stabilize:

$$\mathbb{K}[x] \subsetneq D(\mathbb{K}[x]) \subsetneq D^2(\mathbb{K}[x]) \subsetneq \dots \subsetneq D^n(\mathbb{K}[x]) \subsetneq \dots$$

and then the ring $D^\infty(\mathbb{K}[x])$ that we defined at the beginning of this Chapter can not be directly instantiated in the computer. However this limit ring has extra properties that are of interesting consideration:

Lemma 5.8. *Let (R, ∂) be a differential integral domain and S a differential extension. Then:*

- $D_S^\infty(R) \subset S$.
- $D_S^\infty(R)$ is an integral domain.
- $D_S(D_S^\infty(R)) = D_S^\infty(R)$.
- Let S^* denote the units of S . For all $f \in D_S^\infty(R) \cap S^*$, $1/f \in D_S^\infty(R)$.

Moreover, in the case $R = \mathbb{K}[x]$ and $S = \mathbb{K}[[x]]$, for all $f(x), g(x) \in D^\infty(\mathbb{K}[x])$ with $g(0) = 0$, $f(g(x)) \in D^\infty(\mathbb{K}[x])$.

Several questions remain unanswered about this limit ring D^∞ :

Open Question 5.9 (Same limit). It is clear that if we start to construct the limit from different steps, we may reach different rings. However, there are instances where we reach the same limit:

$$D^\infty(D(R)) = D^\infty(R), \quad D^\infty(\mathbb{C}) = D^\infty(\mathbb{C}[x]).$$

Given two different integral domains R_1 and R_2 , and an ambient integral domain S , can we decide whether or not $D_S^\infty(R_1) = D_S^\infty(R_2)$?

More precisely, fixed $S = \mathbb{K}[[x]]$, can we characterize which subrings $R \subset \mathbb{K}[[x]]$ satisfy $D^\infty(R) = D^\infty(\mathbb{K}[x])$? Can we do the same when $S = \mathbb{K}((x))$ or even the Puiseux series?

Open Question 5.10 (Stable limit). As we have shown in this section, if our starting integral domain R is contained in $\mathbb{C}[x]$ and the ambient integral domain S is big enough, the chain of $D_S^n(R)$ never stabilizes. Given an integral domain R and an ambient integral domain S , can we decide whether or not the chain $D_S^n(R)$ stabilizes?

More precisely, if we fix $S = \mathbb{K}[[x]]$, can we decide for which rings this chains stabilizes?

5.3 Relation with D-algebraic functions

Looking for a deeper understanding of the limit ring $D^\infty(\mathbb{K}[x])$ we study in this section the relation between D^n -finite functions and another wider class of functions: the differentially algebraic functions.

Definition 5.11. Let (R, ∂) be a differential integral domain. We define the ring of differential polynomials $R\{y\}$ as the polynomial ring $R[y_0, y_1, \dots]$ with infinitely many variables such that $\partial(y_i) = y_{i+1}$.

Let (S, ∂) be a differential extension of (R, ∂) . We say that $f \in S$ is differentially algebraic over R if there is $P(y) \in R\{y\}$ such that $P(f) = 0$. This means, if $P(y) = P(y_0, \dots, y_n)$, that

$$P(f, \partial(f), \dots, \partial^n(f)) = 0.$$

We denote by $DA_S(R)$ the set of all differentially algebraic functions of S over R .

The differentially algebraic functions, also called D-algebraic functions, are defined satisfying non-linear differential equations. This allows algebraic relations between the derivatives of the functions. For example, the double exponential $e_2(x)$ is D-algebraic over \mathbb{K} , since it satisfies the differential equation

$$e_2''(x)e_x(x) - e_2'(x)^2 - e_2'(x)e_2(x) = 0, \quad (5.12)$$

and also the tangent is D-algebraic over \mathbb{K} since it satisfies

$$\tan'(x) - \tan(x)^2 - 1 = 0.$$

These functions are well studied and have nice properties [13, 69], similar to those of D-finite functions. However, the algebraic structure behind them is more complex and difficult to manipulate on the computer, as can be seen from the following result:

Theorem 5.13. *Let (R, ∂) be a differential integral domain, (S, ∂) a differential extension and $f \in S$. It is equivalent:*

- (i) $f \in \text{DA}_S(R)$.
- (ii) (Inhom.) *There is $P(y) \in R\{y\}$ and $g \in \text{DA}_S(R)$ such that $P(f) = g$.*
- (iii) (Finite transcendence) *Let F be the field of fractions of R and consider the field $F_\partial(f) = F(f, \partial(f), \partial^2(f), \dots)$. Then*

$$\text{trdeg}(F_\partial(f) : F) < \infty.$$

Proof. Proving (i) \Rightarrow (ii) is trivial setting $g = 0$. For the converse, since $g \in \text{DA}_S(R)$, there is another differential polynomial $Q(y) \in R\{y\}$ such that $Q(g) = 0$. From $P(f) = g$ we can compute other polynomials $P_i(y) = \partial^i(P(y))$ obtaining then

$$Q(P_0(f), P_1(f), \dots) = Q(g) = 0,$$

and it is clear then that $Q(P_0(y), P_1(y), \dots) \in R\{y\}$.

Let prove now (i) \Rightarrow (iii). Consider $P(y) \in R\{y\}$ such that $P(f) = 0$ and assume $P(y)$ has order n (i.e., n is the maximal number $m \in \mathbb{N}$ such that $\deg_{y_m}(P) \neq 0$). We can write

$$P(y) = \sum_{k=0}^d P_i(y_0, \dots, y_{n-1}) y_n^k,$$

showing then that $\partial^n(f)$ is algebraic over $F(f, \dots, \partial^{n-1}(f))$. Moreover,

$$\partial(P(y)) = \sum_{k=0}^d \partial(P_i(y_0, \dots, y_{n-1})) y_n^k + y_{n+1} \sum_{k=0}^d k P_i(y_0, \dots, y_{n-1}) y_n^{k-1},$$

from which, once we evaluate with f and its derivatives, we obtain

$$\partial^{n+1}(f) = - \frac{\sum_{k=0}^d \partial(P_i(f, \dots, \partial^{n-1}(f))) \partial^n(f)^k}{\sum_{k=0}^d k P_i(f, \dots, \partial^{n-1}(f)) \partial^n(f)^{k-1}} = \frac{N_1(f, \dots, \partial^n(f))}{D_1(f, \dots, \partial^n(f))}.$$

Inductively, we can show that for all $p \in \mathbb{N} \setminus \{0\}$, there are two polynomials $N_p(y_0, \dots, y_{n+p-1}), D_p(y_0, \dots, y_{n+p-1}) \in R\{y\}$ such that

$$\partial^{n+p}(f) = \frac{N_p(f, \dots, \partial^{n+p-1}(f))}{D_p(f, \dots, \partial^{n+p-1}(f))},$$

showing that $\partial^{n+p}(f)$ is algebraic over $F(f, \dots, \partial^{n+p-1}(f))$ and, consequently, $\partial^{n+p}(f)$ is algebraic over $F(f, \dots, \partial^{n-1}(f))$. Hence,

$$\text{trdeg}(F_{\partial}(f) : F) \leq \text{trdeg}(F(f, \dots, \partial^{n-1}(f)) : F) \leq n.$$

Finally, to show (iii) \Rightarrow (i) we consider $n = \text{trdeg}(F_{\partial}(f) : F)$ and the elements $f, \partial(f), \dots, \partial^n(f)$. By definition of transcendence degree, we know these $n + 1$ elements are algebraically dependent, so there is a polynomial $\tilde{P}(y_0, \dots, y_n) \in F[y_0, \dots, y_n]$ such that

$$\tilde{P}(f, \dots, \partial^n(f)) = 0.$$

Now, since F is the field of fractions of R , we can clear denominators in the coefficients of \tilde{P} to obtain a polynomial $P(y_0, \dots, y_n) \in R[y_0, \dots, y_n]$ such that $P(f, \dots, \partial^n(f)) = 0$, proving that $f \in \text{DA}_S(R)$. \square

Clearly, all D-finite functions are D-algebraic (since linear differential equations are a particular case of differential polynomials), and we have seen that our two main DD-finite examples falls into this new class. We can easily extend this remark to the following Lemma:

Lemma 5.14. *Let $f(x)$ be a DD-finite function. Then $f(x) \in \text{DA}(\mathbb{K})$.*

Proof. Consider a linear differential equation for $f(x)$ with D-finite coefficients

$$r_0(x)f(x) + r_1(x)f'(x) + \dots r_d(x)f^{(d)}(x) = 0. \quad (5.15)$$

Where $r_i(x)$ is a D-finite function with order s_i . Consider $S = \sum_{i=0}^d s_i$. We are going to prove that there is $P(y_0, \dots, y_{S+d-1}) \in \mathbb{K}[x]\{y\}$ of total degree at most S such that $P(f(x)) = 0$.

First, observe that any derivative of the left hand side of (5.15) also vanishes, yielding

$$\partial_x^n \left(\sum_{i=0}^d r_i(x) f^{(i)}(x) \right) = \sum_{i=0}^d \sum_{k=0}^n \binom{n}{k} r_i^{(k)}(x) f^{(n-k+i)}(x) = 0, \quad n \geq 0.$$

Using the fact that all the $r_i(x)$ are D-finite functions, we can reduce all the derivatives of each $r_i(x)$ of order higher than $s_i - 1$ to obtain a expression of the following shape:

$$\sum_{i=0}^d \sum_{k=0}^{s_i-1} \left(\sum_{j=0}^n p_{i,k,j}(x) f^{(j+i)}(x) \right) = 0,$$

for some polynomials $p_{i,k,j}(x)$.

Since this happens for all $n \geq 0$, we can write these identities as a matrix-vector multiplication $M \cdot (r_0(x), \dots, r_0^{(s_0-1)}(x), \dots, r_d^{(s_d-1)}(x))^T$, where the columns

of M are indexed by the elements in the vector and the rows are indexed by the corresponding n .

It is clear that M has a non-trivial right nullspace and, as M is a square matrix of size S , we have that $\det(M) = 0$. This determinant is a polynomial expression in $f(x)$ and its derivatives and have order at most $S + d$ and a total degree at most S with coefficients in $\mathbb{K}(x)$. It is then well known that $f(x) \in \text{DA}(\mathbb{K}[x])$ implies that $f(x) \in \text{DA}(\mathbb{K})$. \square

Example 5.16 (Double exponential). Let now use Lemma 5.14 with $e_2(x) = e^{e^x - 1}$ and show how we get the same equation that we showed in (5.12). We start from the DD-finite equation for $e_2(x)$:

$$e_2'(x) - e^x e_2(x) = 0.$$

Here we have two coefficients: 1 and e^x that are D-finite with order 1 and the following differential equations:

$$(1)' = 0, \quad (e^x)' - e^x = 0.$$

This will mean that the matrix M has the columns indexed by 1 and e^x . We then compute the first derivative of the equation for $e_2(x)$, yielding:

$$e_2''(x) - e^x e_2'(x) - e^x e_2(x) = 1(e_2''(x)) + e^x(-e_2'(x) - e_2(x)) = 0,$$

and rearranging both equations we have:

$$\begin{pmatrix} e_2'(x) & e_2(x) \\ -e_2''(x) & -e_2'(x) - e_2(x) \end{pmatrix} \begin{pmatrix} 1 \\ e^x \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

Hence, the determinant of the matrix has to be zero, and we obtain:

$$\det \begin{pmatrix} e_2'(x) & -e_2(x) \\ e_2''(x) & -e_2'(x) - e_2(x) \end{pmatrix} = -e_2'(x)^2 - e_2'(x)e_2(x) + e_2(x)e_2''(x) = 0.$$

The proof of the previous Lemma is mainly the walk-through of an elimination process of the coefficients $r_i(x)$ using their differential equations. This leads to a non-linear equation for $f(x)$ but now with simpler coefficients. We can extend this result event further:

Theorem 5.17. *Let $f(x) \in \mathbb{K}[[x]]$ be a differentially algebraic function over $D^n(\mathbb{K}[x])$. Then $f(x)$ is differentially algebraic over $D^{n-1}(\mathbb{K}[x])$, or equivalently, $\text{DA}(D^n) = \text{DA}(D^{n-1})$.*

Before the proof of Theorem 5.17, let analyze its consequences. We already mentioned that D-finite functions are trivially D-algebraic over \mathbb{K} . With the same reasoning, we have also that $D(R) \subset DA(R)$. Using now inductively Theorem 5.17, we have that

$$D^n(\mathbb{K}[x]) \subset DA(D^{n-1}(\mathbb{K}[x])) = DA(\mathbb{K}[x]) = DA(\mathbb{K}),$$

so all the D^n -finite functions that we can obtain with linear differential equations are already contained in the D-algebraic functions using only constant coefficients

$$D^\infty(\mathbb{K}) \subset DA(\mathbb{K}) \quad (5.18)$$

This is a very interesting trade-off: we can have *difficult* coefficients for very simple differential equations (the case of D^n -finite functions) or very simple coefficients with more difficult differential equations (the case of D-algebraic functions).

Proof of Theorem 5.17. Let $f(x)$ be D-algebraic over $D^n(\mathbb{K}[x])$. There there exists a differential polynomial

$$P(y_0, \dots, y_m) = \sum_{\alpha \in \Lambda} p_\alpha(x) (y_0, \dots, y_m)^\alpha,$$

where the elements $p_\alpha(x) \in D^n(\mathbb{K}[x])$, and $\alpha \in \Lambda \subset \mathbb{N}^{m+1}$ represents the corresponding monomials by $(y_0, \dots, y_m)^\alpha = y_0^{\alpha_0} \cdots y_m^{\alpha_m}$; such that

$$P(f(x), f'(x), \dots, f^{(m)}(x)) = 0. \quad (5.19)$$

Also any derivative of the left hand side of (5.19) vanishes. Hence, using the notation $\mathbf{f}(x) = (f(x), \dots, f^{(m)}(x))$, we have:

$$\partial_x^t P(\mathbf{f}(x)) = \sum_{\alpha \in \Lambda} \sum_{k=0}^t \binom{t}{k} p_\alpha^{(k)} \partial_x^{t-k} \mathbf{f}(x)^\alpha = 0.$$

Since the coefficients $p_\alpha(x)$ are in $D^n(\mathbb{K}[x])$, each of them satisfies a linear differential equation of order d_α with coefficients in $D^{n-1}(\mathbb{K}[x])$. These can be used to reduce the equation above to

$$\sum_{\alpha \in \Lambda} \sum_{j=0}^{d_\alpha-1} \left(\sum_{k=0}^t r_{\alpha,k,j} \partial_x^k \mathbf{f}(x)^\alpha \right) p_\alpha^{(j)}(x) = 0,$$

for some $r_{\alpha,k,j}(x) \in D^{n-1}(\mathbb{K}[x])$. Let $S = \sum_{\alpha \in \Lambda} d_\alpha$. Let us denote by $\mathbf{p}(x) = (p_\alpha^{(j)}(x) \mid \alpha \in \Lambda \wedge 0 \leq j \leq d_\alpha)$, then these equation for $t = 0, \dots, S-1$ can be rewritten as a matrix-vector multiplication $M\mathbf{p}(x)^T = \mathbf{0}$, where M is an $S \times S$

matrix where the columns are indexed by the elements of $\mathbf{p}(x)$ and the rows by the index t .

As we remarked in Lemma 5.14, this matrix-vector equation means that the determinant of M is identically zero. This determinant is a polynomial in $f(x)$ and its derivatives up to order $m + S$ with coefficients in $D^{n-1}(\mathbb{K}[x])$. It also has a maximal degree $S \cdot \deg(P)$. This concludes the proof that $f(x)$ is D-algebraic over $D^{n-1}(\mathbb{K}[x])$. \square

This proof uses the same ideas from Lemma 5.14 but it may seem different due to the technicalities involved to express appropriately the matrix-vector multiplication. However the proof is interesting by itself because all the details given provide a constructive approach that will help in the implementation of the Theorem into a computational system. This implementation will be given in Chapter 6.

In the following subsections we analyze the equation (5.18) and study if that containment is strict and when we can transform a D-algebraic equation into a D^n -finite equation.

5.3.1 The Riccati differential equation

While studying D-algebraic equations there are two main ways to increase the complexity of the equation: increase the order of the equation and increase the degree of the equation. The simplest equations that only increase the order of the equation are the linear differential equations. This has been the main object of study during the thesis.

On the other hand, the simplest differential equations that we can find without increasing the order are those of order 1. More precisely, those that are called *monomial differential equations* [20], that are equations of the form $f'(x) = P(f(x))$ for a polynomial $P(y) \in F[y]$ for a differential field F . These monomial equations are the key objects to study for solving problems related with symbolic integration.

In this Subsection we study a particular type of monomial equation: the quadratic case, also known as the Riccati differential equation, which is an equation of the form

$$f'(x) = c(x)f(x)^2 + b(x)f(x) + a(x).$$

We already know that, for some instances of this equation, the solutions can be put inside the DD-finite ring. Namely, the tangent (which has $a(x) = 1$, $b(x) = 0$ and $c(x) = 1$). Can we do something similar for other instances of the Riccati differential equation? Can we know the conditions over the coefficients $a(x)$, $b(x)$ and $c(x)$ to know when a solution will be D^n -finite?

To answer this question, we need to recall a classical method for solving the Riccati differential equation that linearize the differential equation to a second order differential equation.

Lemma 5.20 (Linearization of Riccati's equation). *Let (F, ∂) be a differential field and (E, ∂) a differential extension. Let $y \in E$ such that*

$$\partial(y) = cy^2 + by + a,$$

for some $a, b, c \in F$. Then for any element $v \in E$ such that $y = -\partial(v)/cv$, we have $v \in D_E(F)$ satisfying the following differential equation:

$$\partial^2(v) - \left(b - \frac{\partial(c)}{c}\right) \partial(v) + (ac)v = 0.$$

Proof. From the identity $y = \partial(v)/cv$ we have that

$$\partial(y) = -\frac{c\partial^2(v)v - \partial(c)\partial(v)v - c\partial(v)^2}{c^2v^2},$$

and plugging these two identities into the Riccati differential equation for y we obtain:

$$\frac{\partial(c)\partial(v)v + c\partial(v)^2 - c\partial^2(v)v}{c^2v^2} = \frac{\partial(v)^2}{cv^2} + \frac{b\partial(v)}{cv} + a(x).$$

After clearing denominators, the term with $\partial(v)^2$ cancels out and we obtain the equation for v :

$$-c\partial^2(v)v + \partial(c)\partial(v)v = cb\partial(v)v + ac^2v^2,$$

and after dividing by cv and rearranging the identity, we obtain:

$$\partial^2(v) + \left(b - \frac{\partial(c)}{c}\right) \partial(v) + ca(v) = 0.$$

Since F is a field, all these coefficients are in F , so $v \in D_E(F)$. □

Since the function v is in $D(F)$, and we have the formula $y = -\frac{\partial(v)}{cv}$, we then will have that $y \in D^2(F)$. In the particular context of formal power series, we have the following result:

Proposition 5.21. *Let $a(x), b(x) \in D^n(\mathbb{K}[x])$ and $c(x) \in D^{n-1}(\mathbb{K}[x])$ with $c(0) \neq 0$. Then any D -algebraic formal power series $f(x)$ satisfying*

$$f'(x) - c(x)f(x)^2 - b(x)f(x) - a(x) = 0,$$

is D^{n+2} -finite. More precisely, it is the quotient of two D^{n+1} -finite functions.

Proof. First of all, since $c(x) \in D^{n-1}$ and $c(0) \neq 0$ we know by Lemma 2.6 and Lemma 3.14 that $1/c(x) \in D^n(\mathbb{K}[x])$. Let $v(x)$ be defined by $f(x) = -v'(x)/c(x)v(x)$. Lemma 5.20 shows that $v(x) \in D^{n+1}(\mathbb{K}[x])$.

Using once more Lemma 3.14, we have that $1/v(x) \in D^{n+2}$, and then, again by the closure properties in Proposition 3.10, $f(x) \in D^{n+2}(\mathbb{K}[x])$.

Now that we know $f(x)$ satisfies a differential equation, we need to know we can fix $v(x)$ as a power series. Since formal power series solutions to the Riccati differential equation are fixed by the initial value at $x = 0$, we may choose any value $\alpha \in \mathbb{K}^*$ for $v(0)$ and then take $v'(0)$ as the corresponding value:

$$v'(0) = f(0)c(0)v(0).$$

It is easy to check that we get the same function $f(x)$ independently on which α we take. So we can fix $v(0) = 1$ to have a unique formal power series possible for $v(x)$. \square

Example 5.22 (Tangent as Riccati solution). Let go back with one of our favorite examples: the Tangent function $\tan(x)$. We know that this function satisfies a non-linear differential equation of the following shape:

$$\tan'(x) = \tan(x)^2 + 1.$$

We can see here that this equation is a Riccati differential equation for coefficients

$$a(x) = 1, \quad b(x) = 0, \quad c(x) = 1.,$$

which are all D^0 -finite. Hence, by Proposition 5.21, we have that $\tan(x)$ is DD-finite and it is the quotient of a D-finite function $v(x)$ and its derivative where

$$v''(x) + v(x) = 0.$$

Fixing the value $v(0) = 1$ as we have done in the proof of the Proposition 5.21, we have then that $\tan(x)$ is, as we already knew, the quotient between the sine function ($v'(x)$) and the cosine ($v(x)$).

Corollary 5.22.1. *Let $t(x)$ be a monomial over $\mathbb{K}(x)$ of degree at most 2. Then $t \in D^4(\mathbb{K}[x])$.*

Proof. By definition of a monomial (see Appendix B), we have that $t(x)$ is a monomial over $\mathbb{K}[x]$ of degree at most 2 if there are elements $a(x), b(x), c(x) \in \mathbb{K}(x)$ such that

$$t'(x) = c(x)t(x)^2 + b(x)t(x) + a(x).$$

Since $a(x), b(x), c(x) \in D(\mathbb{K}[x])$, we have by Proposition 5.21 that $t(x) \in D^4(\mathbb{K}[x])$. Moreover, if $c(x) \in \mathbb{K}[x]$, we have that $t(x) \in D^3(\mathbb{K}[x])$. Finally, if $a(x), b(x) \in \mathbb{K}[x]$ and $c(x) \in \mathbb{K}$, then $t(x) \in D^2(\mathbb{K}[x])$. \square

There is a similar linearization applied to other differential equations, called higher order Riccati differential equations that allows us to go from a D-algebraic function to a D^n -finite function.

Definition 5.23. Let (F, ∂) be a differential field and $c \in F^*$. Let $L_y = \partial + cy$ be considered as a differential operator. We define as *nth order Riccati differential equation* to any differential equation that can be written as:

$$R_n(\alpha; y) = (L_y^n \cdot y) + \sum_{i=0}^{n-1} \alpha_i (L_y^i \cdot y) + \alpha_{-1} = 0.$$

We can see that if $n = 1$ we obtain the Riccati differential equation

$$R_1(\alpha_0, \alpha_{-1}; y) = L_y \cdot y + \alpha_0 y + \alpha_{-1} = y' + cy^2 + \alpha_0 y + \alpha_{-1} = 0,$$

where now $b = \alpha_0$ and $a = \alpha_{-1}$. Let us compute now what shape has the second order Riccati differential equation:

- $L_y \cdot y = y' + cy^2$.
- $L_y^2 \cdot y = y'' + 3cyy' + c'y^2 + c^2y^3$.

$$R_2(\alpha; y) = y'' + c^2y^3 + c'y^2 + 3cyy' + c\alpha_1y^2 + \alpha_1y' + \alpha_0y + \alpha_{-1}.$$

We can see here that the second order Riccati differential equation is not as generic as the original Riccati differential equation, and it not covers as many second order differential equations as it happened with the first order case. However, we can apply the same change of variables to linearize the differential equation. First, we need to prove a technical Lemma that will make easier future computations:

Lemma 5.24. Let (F, ∂) be a differential field and (E, ∂) a differential extension. Let $y \in E$, $c \in F^*$ and consider the differential operator $L_y = \partial + cy$. Let $v \in E$ be defined by the equation $y = \partial(v)/cv$. Then, for all $n \in \mathbb{N}$, there is $M_n \in F[\partial]$ such that

$$L_y^n \cdot y = \frac{M_n \cdot v}{cv}.$$

Proof. We proceed by induction on n . Let start with $n = 0$. This case is trivial since $L_y^0 = 1$ and then we have that for $M_0 = \partial$ we have the result.

Now, consider the result true for $n \geq 0$. Then we have that:

$$L_y^{n+1} \cdot y = L_y \cdot (L_y^n \cdot y) = L_y \cdot \frac{M_n \cdot v}{cv}.$$

Using the fact that $y = \partial(v)/vc$ we can compute the final application of L_y :

$$\begin{aligned}
L_y^{n+1} \cdot y &= \partial \left(\frac{M_n \cdot v}{cv} \right) + \frac{\partial v}{v} \frac{M_n \cdot v}{cv} \\
&= \frac{cv((\partial M_n) \cdot v) - (\partial cv + c\partial v)(M_n \cdot v)}{c^2v^2} + \frac{\partial v M_n \cdot v}{cv^2} \\
&= \frac{cv(\partial M_n) \cdot v - \partial(c)v M_n \cdot v - c\partial(v)M_n \cdot v + c\partial(v)M_n \cdot v}{c^2v^2} \\
&= \frac{cv(\partial M_n) \cdot v - \partial(c)v M_n \cdot v}{c^2v^2} = \frac{\left(\partial M_n - \frac{\partial(c)}{c} M_n \right) \cdot v}{cv},
\end{aligned}$$

So we have that $M_{n+1} = \partial M_n - (\partial(c)/c)M_n$. Since $c \in F$, and F is a differential field and $M_n \in F[\partial]$, then $M_{n+1} \in F[\partial]$. \square

We see in Lemma 5.24 that the linear operators M_n are defined recursively and only depend on the element $c \in F^*$ and its derivatives. Here we show the first operators of this sequence:

$$M_0 = \partial, \quad M_1 = \partial^2 - \frac{\partial(c)}{c}\partial, \quad M_2 = \partial^3 - \frac{2\partial(c)}{c}\partial^2 + \frac{2\partial(c)^2 - \partial^2(c)c}{c^2}\partial.$$

Proposition 5.25. *Let (F, ∂) be a differential field and (E, ∂) a differential extension. Let $c \in F$ and $y \in E$ a D-algebraic element over F such that $R_n(\alpha; y) = 0$. Then for any $v \in E$ such that $y = \partial(v)/cv$, we have that $v \in D_E(F)$.*

Proof. Let $y \in E$ be such $R_n(\alpha; y) = 0$. This means that

$$L_y^n \cdot y + \sum_{i=0}^{n-1} \alpha_i L_y^i \cdot y + \alpha_{-1} = 0,$$

and applying Lemma 5.24 we obtain that, for $v \in E$ defined by $y = \partial(v)/cv$ we have:

$$\frac{M_n \cdot v}{cv} + \sum_{i=0}^{n-1} \alpha_i \frac{M_i \cdot v}{cv} + \alpha_{-1} = 0,$$

so we can clean the denominator cv everywhere and we will obtain:

$$(M_n + \alpha_{n-1}M_{n-1} + \dots + \alpha_0\partial + c\alpha_{-1}) \cdot v = 0,$$

which is a linear differential equation with coefficients in F , i.e., $v \in D_E(F)$. \square

Note that when c is a constant (i.e., $\partial(c) = 0$) we obtain that $M_n = \partial_n$ and the linear differential equation for v ends up being:

$$\partial^n(v) + \alpha_{n-1}\partial^{n-1}(v) + \dots + \alpha_0\partial(v) + \alpha_{-1}v = 0.$$

Proposition 5.25 allows us to prove a similar result to Proposition 5.21 but with the general case of the higher order Riccati differential equation:

Corollary 5.25.1. *Let $c(x) \in D^{n-2}(\mathbb{K}[x])$ such that $c(0) \neq 0$ and some elements $\alpha_{-1}(x), \dots, \alpha_{n-1}(x) \in D^{n-1}(\mathbb{K}[x])$. Let $y(x) \in \mathbb{K}[[x]]$ be a solution to the n th order Riccati differential equation $R_n(\alpha(x); y(x)) = 0$. Then $y(x) \in D^{n+1}(\mathbb{K}[x])$.*

Proof. Taking a formal power series $v(x)$ such that $y(x) = \frac{v'(x)}{c(x)v(x)}$ (i.e., we need that $v(0) \neq 0$) we know by Proposition 5.25 that $v(x) \in D^n(\mathbb{K}[x])$. Hence, by Lemma 2.6 and Lemma 3.14, $1/v(x) \in D^{n+1}(\mathbb{K}[x])$ and using the closure properties from Proposition 3.10 we conclude that $y(x) \in D^{n+1}(\mathbb{K}[x])$.

The existence of $v(x) \in \mathbb{K}[[x]]$ satisfying the equation $y(x) = v'(x)/c(x)v(x)$ is easy to see and for any value $\beta \in \mathbb{K}^*$ there is a unique formal power series $v(x)$ with that property. \square

In this subsection we have seen how we can transform some non-linear differential equations into a linear differential equation using a change of variables. In the case for the Riccati differential equation this change of variables was always of the type $y(x) = v'(x)/c(x)v(x)$ for some fixed $c(x)$ given by the differential equation.

It was this change of variables that always guaranteed that if $v(x)$ satisfies a linear differential equation, then $y(x)$ will be D^n -finite for some particular n (in this case, $y(x)$ is always in the next iteration that $v(x)$). We can apply similar tricks to linearize D-algebraic differential equations and obtain similar results. However, which change of variables allows us to obtain this type of classification is not clear. We can state it in the following way:

Open Question 5.26. For which type of functional \mathcal{F} can we allow the change of variables $y(x) = \mathcal{F}(v)$ in such a way that if $v(x) \in D^n(\mathbb{K}[x])$ then $y(x) \in D^m(\mathbb{K}[x])$?

Open Question 5.27. Given a functional \mathcal{F} that is valid according to the previous question, which differential operators (linear or non-linear) are compatible with this change of variables?

5.3.2 Separable first order equations

In the following Subsection we study another way of ensuring that a D-algebraic is D^n -finite for some particular n , namely, we study sufficient conditions for solutions of first order separable equations to be D^n -finite.

Definition 5.28. We say a differential equation is *separable* if it can be written in the following way:

$$y'(x) = g(x)f(y(x)),$$

for some functions $g(x)$ and $f(x)$.

In our context, we consider only formal power series $g(x)$ and $f(x)$ where $f(0) \neq 0$. We can formally solve this type of equations:

$$\frac{y'(x)}{f(y(x))} = g(x) \Rightarrow \int_0^x \frac{y'(t)dt}{f(y(t))} = \int_0^x g(t)dt,$$

so if we consider $F(x) = \int dx/f(x)$, we have

$$F(y(x)) = \int_0^x g(t)dt,$$

and then composing with $F^{-1}(x)$ we can obtain an expression for $y(x)$:

$$y(x) = \int_0^{F^{-1}(x)} g(t)dt.$$

This computations are valid for formal power series when $f(0) \neq 0$, and thanks to Theorem 5.2, we can prove the following result:

Proposition 5.29. *Let $f(x) \in \mathbb{K}[[x]]$ such that $f(0) \neq 0$ and $g(x) \in D^n(\mathbb{K}[x])$. Consider*

$$F(x) = \int_x \frac{dx}{f(x)},$$

and assume $F^{-1}(x) \in D^m(\mathbb{K}[x])$. Then any formal power series $y(x)$ solution to the separable equation $y'(x) = g(x)f(y(x))$ is D^{n+m} -finite.

Proof. The previous computations show that $y(x) = F^{-1}(G(x))$ where $G(x)$ is the integral of $g(x)$ with $G(0) = 0$. Hence, by Proposition 3.8 we have that $G(x) \in D^n(\mathbb{K}[x])$, so using now Theorem 5.2 we obtain that $y(x) \in D^{n+m}(\mathbb{K}[x])$. \square

The conditions for Proposition 5.29 could be considered too artificial. In fact, knowing if the functional inverse of a formal power series is D^n -finite or not is a very difficult and unsolved problem.

Lemma 5.30. *Let $f(x) \in \mathbb{K}[[x]]$ such that $f(0) = 0$. For all $n \in \mathbb{N} \setminus \{0\}$ we have*

$$f^{(n)}(f^{-1}(x)) = F_n(f^{-1}),$$

where $F_k(y)$ is a fraction of differential polynomials in \mathbb{K} .

Proof. We proceed by induction. The case $n = 1$ is clear using the chain rule:

$$1 = (x)' = (f \circ f^{-1})'(x) = f'(f^{-1}(x))(f^{-1})'(x),$$

so we have $F_1(y) = 1/y_1$.

For $n > 1$, using again the chain rule we have that:

$$(f^{-1})'(x)f^{(n+1)}(f^{-1}(x)) = (f^{(k)}(f^{-1}(x)))' = F_n(f^{-1})',$$

so we have that $F_{n+1}(y) = F_n(y)'/y_1$. \square

If we compute the first quotients $F_n(y)$ we can easily see:

$$F_1(y) = \frac{1}{y_1}, \quad F_2(y) = \frac{-y_2}{y_1^3}, \quad F_3(y) = \frac{3y_2^2 - y_3y_1}{y_1^5}.$$

This Lemma will help us prove a more interesting result about the inverse of a D-algebraic function:

Lemma 5.31. *Let $f(x) \in \mathbb{K}[[x]]$ be a D-algebraic function over \mathbb{K} such that $f(0) = 0$. Then $f^{-1}(x)$ is also D-algebraic over \mathbb{K} .*

Proof. Since $f(x)$ is D-algebraic over \mathbb{K} , there is a differential polynomial $P(y) \in \mathbb{K}\{y\}$ such that $P(f) = 0$. Now, assume the order of this polynomial is n , then

$$P(f) = P(f(x), f'(x), \dots, f^{(n)}(x)) = 0.$$

If we compose this equality with $f^{-1}(x)$, which exists since $f(0) = 0$, we have, using Lemma 5.30, that

$$P(x, F_1(f^{-1}(x)), \dots, F_n(f^{-1}(x))) = 0,$$

which is a D-algebraic equation (after clearing denominators) for $f^{-1}(x)$ with coefficients in $\mathbb{K}[x]$. Now using Theorem 5.17, we have then that $f^{-1}(x)$ is D-algebraic over \mathbb{K} . \square

However, knowing whether an inverse of a D-algebraic function is D^n -finite is not so simple. We can give several examples where both $f(x)$ and $f^{-1}(x)$ are D-finite:

Example 5.32. The following functions are D-finite formal power series:

- $e^x - 1$ and $\log(x + 1)$.
- $\sin(x)$ and $\arcsin(x)$.
- $\cos(x)$ and $\arccos(x)$.
- Any *algebraic* function and its functional inverse.

Lemma 5.33. *Let $t(x) \in \mathbb{K}[[x]]$ be a monomial over \mathbb{K} , i.e., it satisfies a differential equation of the shape $t'(x) = p(t(x))$ for some polynomial $p(y) \in \mathbb{K}[y]$. If $t^{-1}(x) \in \mathbb{K}[[x]]$, then $t^{-1}(x) \in D(\mathbb{K}[x])$.*

Proof. We start from the equation for $t(x)$:

$$t'(x) = \alpha_n t(x)^n + \dots + \alpha_1 t(x) + \alpha_0,$$

where all $\alpha_i \in \mathbb{K}$. Then we compose this equation with $t^{-1}(x)$, and we obtain:

$$\frac{1}{(t^{-1})'(x)} = \alpha_n x^n + \dots + \alpha_1 x + \alpha_0,$$

or, rearranging the previous equation, we have that $t^{-1}(x)$ satisfies:

$$(\alpha_n x^n + \dots + \alpha_0)(t^{-1})'(x) = 1,$$

which is a inhomogeneous equation leading to $t^{-1}(x)$ being D-finite. \square

After studying briefly the behavior of inverses for some formal power series, we will proceed now to use Proposition 5.29 in several particular cases:

Example 5.34. Let $p(x) = \prod_{i=0}^n (x - \alpha_i)$ for some elements $\alpha_i \in \mathbb{K}$, such that they are pairwise different (i.e., $\alpha_i \neq \alpha_j$ if $i \neq j$). Then all the formal power series solutions to the non-linear differential equation $y'(x) = p(y(x))$ are DD-finite.

Proof. Using the notation in Proposition 5.29, we have here that $g(x) = 1$ and $f(x) = p(x)$. Then, the function $F(x)$ can be easily computed using Partial Fraction Decomposition:

$$\begin{aligned} F(x) &= \int_x \frac{dx}{p(x)} = \int_x \sum_{i=1}^n \frac{a_i}{x - \alpha_i} \\ &= \sum_{i=1}^n a_i \log(x - \alpha_i) = \log \left(\prod_{i=1}^n (x - \alpha_i)^{a_i} \right), \end{aligned}$$

where the values of a_i are integers. Let $B(x)$ the argument of such logarithm. It is a polynomial and, in particular, an algebraic formal power series. Hence we have:

$$F^{-1}(x) = B^{-1}(e^x),$$

so it is a composition of an algebraic function with the exponential function, both being D-finite. This means that $F^{-1}(x)$ is DD-finite and, by Proposition 5.29, we conclude that $y(x)$ is DD-finite. \square

Example 5.35. Let $p(x) = (x - \alpha)^n$ for some algebraic element α over \mathbb{K} and $n \in \mathbb{Q} \setminus \{1\}$. Then all solutions to the non-linear equation $y'(x) = p(y(x))$ are D-finite. In fact, they are algebraic over $\mathbb{K}[x]$.

Proof. In this case, we have that $g(x) = 1$ and $f(x) = (x - \alpha)^n$. We can compute then the function $F(x)$:

$$F(x) = \int_x \frac{dx}{(x - \alpha)^n} = \frac{1}{(-n + 1)(x - \alpha)^{n-1}}.$$

We have in this case that $F(x)$ is an algebraic function (since we allowed $n \in \mathbb{Q}$) and Then $F^{-1}(x)$ is also algebraic. Using now Proposition 5.29, we conclude that $y(x) = F^{-1}(x + C)$ is algebraic. So in particular, $y(x)$ is D-finite. \square

These two examples are the extreme cases for equations of the type $y'(x) = p(y(x))$ for some polynomial $p(x) \in \mathbb{K}[x]$. Note that these two examples cover all cases for the Riccati differential equation: in a polynomial of degree two, either all roots are the same or all roots are pairwise different.

We would also like to remark that we can formally remove in the previous examples the conditions that the roots α_i are constants. This will allow us to use the previous result for equations $y'(x) = p(y(x))$ where $p(t) \in F[t]$ for any differential field F . However, in doing so, we would need to give a precise definition of the expressions $(x - \alpha_i)^{a_i}$ where the exponent a_i is also a function.

Before going to the next subsection, we propose several open questions that remained unsolved until now.

Open Question 5.36. Is there a computable criteria to characterize those D-algebraic functions that are D^n -finite?

Open Question 5.37. It is a well known result for D-finite functions that, given a differential field (F, ∂) , f and $1/f$ are in $D(F)$ if and only if f'/f is algebraic over F [33]. Is there any similar computable criteria to characterize when $f(x)$ and $f^{-1}(x)$ are both D^n -finite functions for a fixed n ?

5.3.3 A minimal counterexample

Until now, we have focused on studying when we can decide whether a D-algebraic function is D^n -finite for some particular n . The results have been limited but, so far, we could not show that any of the D-algebraic examples were not D^n -finite.

In this Subsection we show (using the results in [58]), that the solutions of the differential equation

$$y'(x) - y(x)^3 + y(x)^2 = 0 \tag{5.38}$$

are not D^n -finite for any $n \in \mathbb{N}$. This example is very interesting because it is the minimal example that does not fall into the results shown in the previous subsections. This equation is separable with $g(x) = 1$ and $f(x)$ a polynomial of degree 3, so it is a Riccati differential equation, and the roots of the polynomial

$y(x)^3 - y(x)^2$ are only two, one multiple root and one simple root, which is the only case we did not covered in the previous subsection.

However, showing this result requires some knowledge of differential Galois theory, that can be read in [25, 70] and we have detailed in Appendix B.

The main result that allows us to prove that the solutions to (5.38) are not D^n -finite for any n was proven by Rosenlicht[64] and generalized recently by Noordman, can der Put and Top [58]:

Theorem 5.39 ([64, Collorary of Proposition 1] and [58, Theorem 2.1]). *Consider any non constant solutions $y_1(x), \dots, y_n(x)$ to the differential equation (5.38). Then they are algebraically independent.*

In the case of Rosenlicht, the results is stated for two different solutions and in the paper by Noordman et al. we can fin this result generalized for any number of solutions and for a wider class of equation (that are called of *general type*).

Lemma 5.40 ([58, Proposition 7.1]). *Let \mathcal{S} be the solutions of a D-algebraic equation over \mathbb{K} such that any finite subset of non constant elements are algebraically independent. Then for all $y(x) \in \mathcal{S}$ there exists none iterated Picard-Vessiot extension that contains $y(x)$.*

Proof. Let $y(x) \in \mathcal{S}$ and assume there is a chain of Picard-Vessiot extensions $\mathbb{K}(x) \subset \mathbb{K}_1 \subset \dots \subset \mathbb{K}_n = F$ such that $y(x) \in F$. Let G be the differential Galois group if $\mathbb{K}_n \supset \mathbb{K}_{n-1}$. For any $\sigma \in G$ we have that $\sigma(y)$ is also a solution of the same D-algebraic equations over $\mathbb{K}(x)$ that $y(x)$, so $\sigma(y) \in \mathcal{S}$. Since $\text{trdeg}(\mathbb{K}_n : \mathbb{K}_{n-1})$ is finite, there are only finitely many possibilities for $\sigma(y)$ by the assumption of algebraic independence in \mathcal{S} .

This means that $y(x)$ lies on the fixed field $\mathbb{K}_n^{G^0}$, where G^0 is the connected component of the identity element in G . But this implies that $y(x)$ is algebraic over \mathbb{K}_{n-1} . If $n = 1$ this yield a contradiction, since we know that $y(x)$ is transcendent over $\mathbb{K}(x)$.

For $n > 1$, consider $P(T) = T^m + a_{m-1}T^{m-1} + \dots + a_0$ be the minimal polynomial of $y(x)$ over \mathbb{K}_{n-1} . Let now σ be in the differential Galois group of \mathbb{K}_{n-1} over \mathbb{K}_{n-2} . Since $y(x)$ is algebraic over \mathbb{K}_{n-1} there is a unique option for defining $\sigma(y)$. This $\sigma(y)$ is also algebraic over \mathbb{K}_{n-1} and it is a zero of the polynomial $\sigma(P) = T^m + \sigma(a_{m-1})T^{m-1} + \dots + \sigma(a_0)$.

Since there are only many options for $\sigma(y)$, then there are also a finitely many possibilities for $\sigma(P)$. This means, as we had before for y , that all a_i are algebraic over \mathbb{K}_{n-2} . This implies that, in particular, $y(x)$ is algebraic also over \mathbb{K}_{n-2} . Repeating this procedure, we obtain that $y(x)$ is algebraic over $\mathbb{K}(x)$ yielding a contradiction. \square

Corollary 5.40.1. *Let $y(x)$ be a solution to the differential equation $y'(x) = y(x)^3 - y(x)^2$. Then $y(x)$ is not D^n -finite for any $n \in \mathbb{N}$.*

Proof. It is clear that if $y(x) \in D^n(\mathbb{K}[x])$, then $y(x)$ would be in an iterated Picard-Vessiot extension of $\mathbb{K}[x]$. By Theorem 5.39 and Lemma 5.40 we know that this is impossible for $y(x)$. Hence, $y(x) \notin D^\infty(\mathbb{K}[x])$. \square

Chapter 6

Implementation of structural results

In the same spirit of Chapter 4, we describe here precisely all the algorithms for computing the operations described in Chapter 5. We will also include some examples on how these algorithms perform in some particular cases. These algorithms are extracted directly from the proofs in [39].

If we recall Chapter 4, when using Theorem 3.6 to prove that a function is differentially definable, the ansatz method comes naturally into place. However, in the algorithms described in Chapter 4, the integral domain R for the coefficients remained fixed.

In the composition operations, before starting the ansatz method we have to decide the exact coefficient ring that the ansatz work with. This is an important step computing compositions and determine the final performance of the execution: using a coefficient ring more complex will increase the execution time in a significant way.

As in Chapter 4, let $h(x)$ be the function we want to represent. To have a complete implementation of the ansatz method for the composition and the algebraic substitution, we need to specify the following components:

- The coefficient ring R for the resulting differential operator.
- An ambient vector space W with finite dimension that contains the vector space spanned by $h(x)$ and its derivatives.
- A list of generators Φ for W .
- A derivation matrix M of ∂_x w.r.t. Φ (see Definition A.7).
- A representation of $h(x)$ in terms of Φ .

- A procedure to compute initial conditions of $h(x)$: $h(0), h'(0), \dots$

Once we have all these components, we can apply methods `get_linear_system` and `non_trivial_solution` as done before to compute the differential operator that annihilates $h(x)$. Computing enough initial conditions we can characterize the particular solution of that operator to represent the function $h(x)$ exactly.

6.1 Implementation of composition

In this Section, we consider $h(x) = f(g(x))$ where $f(x)$ is a D^n -finite function, for $n \geq 1$, satisfying the following differential equation

$$r_0(x)f(x) + r_1(x)f'(x) + \dots r_d(x)f^{(d)}(x) = 0, \quad (6.1)$$

and $g(x)$ is a D^m -finite function, for $m \geq 0$, with $g(0) = 0$, so the composition is well defined as a formal power series. We refer to the proof of Theorem 5.2 for the deep details of this Section.

The coefficient ring

Theorem 5.2 guarantees that $f(g(x))$ belongs to $D^{n+m}(\mathbb{K}[x])$. This means that the coefficients of the final differential operator will be D^{n+m-1} -finite functions.

In particular, this implies that $g(x)$, as a function, can be used as a coefficient of the differential equation.

The ambient vector space W_\circ and its generators

During the proof of Theorem 5.2, we showed that the vector space

$$V_{F_{n+m-1}(x)}(h(x)) = \langle h(x), h'(x), h''(x), \dots \rangle_{F_{n+m-1}(x)},$$

was contained in the following vector space:

$$W_\circ = \langle f(g(x)), f'(g(x)), f''(g(x)), \dots \rangle_{F_{n+m-1}(x)},$$

and that vector space had a finite dimension since the substitution of x by $g(x)$ in the equation 6.1 leads to linear relations between $f^{(k)}(g(x))$ and the previous derivatives for all $k \leq d$.

This means that $\Phi_\circ = (f(g(x)), f'(g(x)), \dots, f^{(d-1)}(g(x)))$ is the list of generators that we are looking for.

A derivation matrix M_o w.r.t. Φ_o

We need to define a derivation matrix w.r.t. Φ_o , i.e., a matrix M such we can compute derivatives of elements represented w.r.t. Φ_o with the formula

$$\partial(\mathbf{v}) = M\mathbf{v} + \kappa_\partial(\mathbf{v}),$$

where $\kappa_\partial((v_1, \dots, v_k)) = (\partial(v_1), \dots, \partial(v_k))$ (see Definition A.7 for further information). In order to do so, we only need to represent the derivatives of the elements of Φ_o with respect to themselves.

For the first $d - 1$ elements, this is simply applying the chain rule:

$$\partial_x(f^{(k)}(g(x))) = g'(x)f^{(k+1)}(g(x)),$$

and, since $n \geq 1$ and $g(x) \in D^m(\mathbb{K}[x])$ we have guaranteed that $g(x)$ and $g'(x)$ belong to $D^{n+m-1}(\mathbb{K}[x])$.

The last element $f^{(d-1)}(g(x))$ is more interesting, since $f^{(d)}(g(x))$ is not an element in Φ_o anymore. But in this case we can use equation (6.1) and, after the substitution $x \mapsto g(x)$, we get:

$$f^{(d)}(g(x)) = -\frac{r_0(g(x))}{r_d(g(x))}f(g(x)) - \dots - \frac{r_{d-1}(g(x))}{r_d(g(x))}f^{(d-1)}(g(x))$$

The resulting derivation matrix, where the k th column is the representation of the derivative of $f^{(k)}(g(x))$ w.r.t. Φ_o , is:

$$M_o = \begin{pmatrix} 0 & 0 & \dots & 0 & -g'(x)\frac{r_0(g(x))}{r_d(g(x))} \\ g'(x) & 0 & \dots & 0 & -g'(x)\frac{r_1(g(x))}{r_d(g(x))} \\ 0 & g'(x) & \dots & 0 & -g'(x)\frac{r_2(g(x))}{r_d(g(x))} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & g'(x) & -g'(x)\frac{r_{d-1}(g(x))}{r_d(g(x))} \end{pmatrix}.$$

Note that M_o is closely related with the companion matrix of $f(x)$. In fact, if we compose every entry of the companion matrix \mathcal{C}_f with $g(x)$ and multiply then by $g'(x)$, we obtain exactly M_o . This means that computing the composition between $f(x)$ and $g(x)$ *requires a recursive call* for each $r_i(g(x))$. This is important for determining the performance of the whole algorithm. For simplicity, we will write $M_o = g'(x)\mathcal{C}_f(g(x))$.

Intuitively, the derivation matrix represent how to compute derivatives in a vector space. Here we see that the derivation matrix for the composition looks like the chain rule. This similarity with the functional case also happened before with the addition and multiplication.

Representation of $h(x) = f(g(x))$

As it happened in Chapter 4 during the specification of the product and the algebraic inclusion, the function $h(x)$ is precisely one of the elements of our generators. In particular for the composition, it is the first. Hence,

$$h(x) = f(g(x)) \quad \rightarrow \quad \mathbf{v}_0 = (1, 0, 0, \dots, 0).$$

Putting together all the procedures and specifications that we have just described, we can provide a complete algorithm (see Algorithm `comp_operator`) for computing the differential operator that annihilates h .

Algorithm 11: `comp_operator`

Input : elements $f \in D^n(\mathbb{K}[x])$ and $g \in D^m(\mathbb{K}[x])$
Output: operator $\mathcal{L} \in D^{n+m-1}(\mathbb{K}[x])[\partial]$ such that $\mathcal{L} \cdot (f(g(x))) = 0$
 // Getting the specific for composition
 $\mathcal{C}_f \leftarrow \text{companion}(f);$
 $d \leftarrow \text{order}(f);$
 $M_o \leftarrow g'(x)\mathcal{C}_f(g(x));$ // **Recursive step**
 $\mathbf{v}_0 \leftarrow (\delta_{1,i} \text{ for } i = 1, \dots, d);$
 // Getting the solution for the ansatz
 $\alpha \leftarrow \text{non_trivial_solution}(\text{get_linear_system}(\mathbf{v}_0, M_o));$
return $\alpha_0 + \alpha_1\partial + \dots + \alpha_d\partial^d;$

Initial conditions for $h(x)$

As we remarked during the proof of Theorem 5.2, iteratively applying the chain rule, we obtain the following formula:

$$h^{(p)}(x) = \sum_{l=1}^p f^{(p)}(g(x)) B_{p,l}(g'(x), \dots, g^{(p-l+1)}(x)),$$

where $B_{p,l}(y_1, \dots, y_{p-l+1})$ are the Bell polynomials. If we evaluate this formula at $x = 0$, we have that, in order to get the p th initial value for $h(x) = f(g(x))$ we only need the first initial conditions for $f(x)$ and $g(x)$. These values can be obtained by the method `init` for both $f(x)$ and $g(x)$:

$$\text{init}(f, i) \rightarrow f^{(i)}(0), \quad \text{init}(g, i) \rightarrow g^{(i)}(0).$$

The case when $p = 0$ is not included in the previous formula. However, since $h(x) = f(g(x))$ and $g(0) = 0$, we can see that $h(0) = f(0)$.

These computations are provided in the algorithm `comp_init_values`.

Algorithm 12: comp_init_values

Input : elements $f(x) \in D^n(\mathbb{K}[x])$ and $g(x) \in D^m(\mathbb{K}[x])$ and a bound $p \in \mathbb{N}$

Output: first p initial values of $f(g(x))$

for $j = 0, \dots, p-1$ **do**

$f_j \leftarrow \text{init}(f, j);$

$g_j \leftarrow \text{init}(g, j);$

$h_0 \leftarrow \text{init}(f, 0);$

for $i = 1, \dots, p-1$ **do**

$h_i \leftarrow \sum_{l=1}^i f_l \text{bell_polynomial}(i, l)(g_1, \dots, g_{i+l-1});$

return $(h_0, \dots, h_{p-1});$

Example 6.2 ([39, Example 11]). Let $f(x) = \exp(x)$ and $g(x) = \sin(x)$. Consider the function $h(x)$. We know that both $f(x)$ and $g(x)$ are D-finite, so we have that $h(x)$ is DD-finite. Note that the following differential operator annihilates $f(x)$:

$$\mathcal{A} = \partial_x - 1,$$

so computing the matrix M_o becomes trivial:

$$M_o = g'(x)\mathcal{C}_f(g(x)) = (\cos(x)).$$

On the other hand, $\mathbf{v}_0 = (1)$, so the final system we need to solve is

$$\mathcal{S} = \begin{pmatrix} 1 & \cos(x) \end{pmatrix}.$$

Clearly, $(-\cos(x), 1)^T$ is an element in the right nullspace of \mathcal{S} implying that $h(x)$ satisfies the following differential equation:

$$h'(x) - \cos(x)h(x) = 0.$$

Lastly, we need to compute $h(0)$. In this case this is simply $f(0) = e^0 = 1$.

Example 6.3 ([39, Example 12]). Let $f(x) = \log(x+1)$ and $g(x) = \exp(x) - 1$. They are both D-finite with annihilating operators

$$\mathcal{A}_f = (x+1)\partial_x^2 + \partial_x, \quad \mathcal{A}_g = \partial_x^2 - \partial_x.$$

We are going to show using `comp_operator` that $h(x) = f(g(x)) = x$, i.e., that the logarithm and the exponential are functional inverses.

First of all, we need to compute the derivation matrix for $f(g(x))$. In this case we obtain:

$$\mathcal{C}_\circ = g'(x)\mathcal{C}_f(g(x)) = e^x \begin{pmatrix} 0 & 0 \\ 1 & -\frac{1}{e^x} \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ e^x & -1 \end{pmatrix}.$$

Starting now with $\mathbf{v}_0 = (1, 0)^T$ we compute the system to be solved by iteratively computing the derivatives of \mathbf{v}_0 , yielding:

$$\mathcal{S} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & e^x & 0 \end{pmatrix}.$$

In this case, it is clear that $(0, 0, 1)^T$ is in the right nullspace of \mathcal{S} , proving that

$$h''(x) = 0.$$

Moreover, using algorithm `comp_init_values` to compute the initial conditions of $h(x)$, we obtain:

$$h(0) = 0, \quad h'(0) = 1.$$

The differential equation we obtained means that $h(x)$ is a polynomial of degree at most 1 and the initial conditions show that $h(x) = x$.

Example 6.4 ([39, Example 13]). Let $f(x) = g(x) = \sin(x)$ and $h(x) = f(g(x))$. We know that $f(x)$ and $g(x)$ are D-finite annihilated by $\mathcal{A} = \partial^2 + 1$. Hence $h(x)$ is DD-finite.

In this particular case, the derivation matrix \mathcal{C}_\circ is

$$\mathcal{C}_\circ = g'(x)\mathcal{C}_f(g(x)) = \begin{pmatrix} 0 & \cos(x) \\ -\cos(x) & 0 \end{pmatrix}.$$

Starting from $\mathbf{v}_0 = (1, 0)^T$, we construct the following system:

$$\mathcal{S} = \begin{pmatrix} 1 & 0 & -\cos(x)^2 \\ 0 & \cos(x) & -\sin(x) \end{pmatrix}.$$

We can easily check that $(\cos(x)^3, \sin(x), \cos(x))^T$ is in the right nullspace of \mathcal{S} , proving that

$$\cos(x)h''(x) + \sin(x)h'(x) + \cos(x)^3h(x) = 0.$$

Using now algorithm `comp_init_values`, we compute the initial values for $h(x)$, yielding:

$$h(0) = 0, h'(0) = 1.$$

In fact, it is easy to see that for any $g(x)$ with $g(0) = 0$ and $g'(0) \neq 0$, the function $\sin(g(x))$ satisfies the differential equation:

$$g'(x)h''(x) - g''(x)h'(x) + g'(x)h(x) = 0,$$

and have initial conditions $h(0) = 0$ and $h'(0) = g'(0)$.

Example 6.5 ([39, Example 14]). Let $h(x) = \sin(\sin(\sin(x)))$. We can represent $h(x)$ as a composition of $f(x) = \sin(x)$ and $g(x) = \sin(\sin(x))$. Using Example 6.4, we obtain directly that:

$$g'(x)h''(x) - g''(x)h'(x) + g'(x)h(x) = 0,$$

$$h(0) = 0 \text{ and } h'(0) = 1.$$

However, we could also write $h(x) = f(g(x))$ where $f(x) = \sin(\sin(x))$ and $g(x) = \sin(x)$. In this case, the derivation matrix required is:

$$\mathcal{C}_o = g'(x)\mathcal{C}_f(g(x)) = \begin{pmatrix} 0 & -\cos(\sin(x))^3 \\ \cos(x) & -\frac{\sin(\sin(x))}{\cos(\sin(x))} \end{pmatrix}.$$

Starting with $\mathbf{v}_0 = (1, 0)^T$ we obtain the following linear system:

$$\mathcal{S} = \begin{pmatrix} 1 & 0 & -\cos(x)\cos(\sin(x))^3 \\ 0 & \cos(x)\cos(\sin(x)) & -\cos(x)\sin(\sin(x)) - \sin(x)\cos(\sin(x)) \end{pmatrix}.$$

We can check that the vector $(\alpha_0, \alpha_1, \alpha_2)^T$ is in the right nullspace of \mathcal{S} where

$$\begin{cases} \alpha_0 = \cos(x)^2 \cos(\sin(x))^4 \\ \alpha_1 = \cos(x) \sin(\sin(x)) + \sin(x) \cos(\sin(x)) \\ \alpha_2 = \cos(x) \cos(\sin(x)) \end{cases}$$

This example shows how important the *recursive step* is. This recursion, performed when computing the derivation matrix, change drastically the system \mathcal{S} . Since the composition *only depends* on the equation for $f(x)$, we should always take that into consideration when computing three or more compositions in a row. Here the order of the compositions may change the performance of the algorithm.

6.2 Implementation of algebraic substitution

In this Section, we consider $h(x) = f(a(x))$ where $f(x)$ is a D^n -finite function for some $n \geq 1$ satisfying the following differential equation

$$r_0(x)f(x) + r_1(x)f'(x) + \dots r_d(x)f^{(d)}(x) = 0,$$

and $a(x)$ is an algebraic function of degree p over $F_m(x)$ where $m \geq 0$ with $a(0) = 0$, so the composition is well defined as a formal power series. We refer to the proof of Theorem 5.4 for the details of this Section.

The ring for coefficients

Theorem 5.4 guarantees that $f(a(x))$ belongs to $D^{n+m}(\mathbb{K}[x])$. This means that the coefficients of the final differential operator will be D^{n+m-1} -finite functions.

In particular, this implies that $a(x)$, as a function, can be used as a coefficient for differential equation whenever $n \geq 2$. In the case $n = 1$ the coefficient ring is $F_m(x)$, and here $a(x) \notin F_m(x)$ (since it was algebraic over it).

The ambient vector space $W_{\circ,a}$ and its generators

During the proof of Theorem 5.4, we showed that the vector space

$$V_{F_{n+m-1}(x)}(h(x)) = \langle h(x), h'(x), h''(x), \dots \rangle_{F_{n+m-1}(x)},$$

was contained in the following vector space:

$$W_{\circ,a} = \langle f(a(x)), f'(a(x)), \dots, f^{(d-1)}(a(x)) \rangle_{F_{n+m-1}(x)} \otimes \langle 1, a(x), \dots, a(x)^{p-1} \rangle_{F_{n+m-1}(x)}$$

.

This means we can compute $\Phi_{\circ,a}$ as the tensor product of the generators of each vector space:

$$\Phi_{\circ,a} = \{f(a(x)), f'(a(x)), \dots, f^{(d-1)}(a(x))\} \otimes \{1, a(x), \dots, a(x)^{p-1}\}$$

A derivation matrix $M_{\circ,a}$ w.r.t. $\Phi_{\circ,a}$

Recall that a derivation matrix M is a matrix that allows us to compute derivatives within a vector space with the formula

$$\partial(\mathbf{v}) = M\mathbf{v} + \kappa_{\partial}(\mathbf{v}),$$

where $\kappa_{\partial}((v_1, \dots, v_k)) = (\partial(v_1), \dots, \partial(v_k))$. Once we know the generator set $\Phi_{\circ,a}$, we can simply compute the derivatives of its elements and represent them w.r.t themselves.

In this particular case, we can use the results we have from the implementation of the algebraic inclusion (see Subsection 4.2.4), the previous Section about the composition and similar arguments from Subsection 4.2.3 about how to compute the derivation matrix of a tensor product of two vector spaces.

On one hand, if we focus on the first vector space

$$\langle f(a(x)), f'(a(x)), \dots, f^{(d-1)}(a(x)) \rangle_{F_{n+m-1}(x)},$$

we have the same situation as in Section 6.1, so we get a derivation matrix of the shape:

$$a'(x)\mathcal{C}_f(a(x)),$$

which requires a recursive computation with each of the coefficients $r_i(x)$.

On the other hand, if we focus on the vector space

$$\langle 1, a(x), \dots, a(x)^{p-1} \rangle_{F_{n+m-1}(x)},$$

we have the same situation as in Subsection 4.2.4, reaching a derivation matrix using the method `algebraic_derivation_matrix`. However, in the case $n \geq 2$ we can also have a different approach.

Since we are considering the vector space over $F_{n+m-1}(x)$, and $n+m-1 \geq m+1$, we can use $a(x)$ as a coefficient. In this particular case (recall that this is not valid for D-finite functions $f(x)$), we have that:

$$\partial_x(a(x)^k) = ka'(x)a(x)^{k-1},$$

leading to the following derivation matrix for $\langle 1, a(x), \dots, a(x)^{p-1} \rangle_{F_{n+m-1}(x)}$:

$$\begin{pmatrix} 0 & a'(x) & 0 & \dots & 0 \\ 0 & 0 & 2a'(x) & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & (p-1)a'(x) \\ 0 & 0 & 0 & \dots & 0 \end{pmatrix}. \quad (6.6)$$

It is not important which derivation matrix we use. The result of the algorithm will be correct in both cases. In any case, we denote this derivation matrix by M_a .

As it happened with the product (see Subsection 4.2.3, using Lemmas A.12 and A.6 and Proposition A.10, we can conclude now that the derivation matrix w.r.t. $\Phi_{\circ,a}$ is

$$M_{\circ,a} = (a'(x)\mathcal{C}_f(a(x))) \boxplus M_a.$$

In the description of algorithm 13, we used the first approach (using the algebraic information of $a(x)$) since it works in all cases of n . However, using matrix (6.6) may lead to simpler computations since $a'(x)$ appears as a common factor of both matrices before the Kronecker sum (\boxplus).

Representation of $h(x) = f(a(x))$

As it happened in the previous section with the composition, the function $h(x)$ that we are interested in is precisely the first element of our generators. Hence,

$$h(x) = f(a(x)) \quad \rightarrow \quad \mathbf{v}_0 = (1, 0, 0, \dots, 0).$$

Putting together all the procedures that we have just described, we can provide a complete algorithm (see Algorithm [alg_substitution_operator](#)) for computing the differential operator that annihilates h .

Algorithm 13: [alg_substitution_operator](#)

Input : elements $f \in D^n(\mathbb{K}[x])$ and a algebraic over $F_m(x)$
Output: operator $\mathcal{L} \in D^{n+m-1}(\mathbb{K}[x])[\partial]$ such that $\mathcal{L} \cdot (f(g(x))) = 0$
 // Getting the specific for composition
 $\mathcal{C}_f \leftarrow \text{companion}(f)$;
 $d \leftarrow \text{order}(f)$; $p \leftarrow \text{degree}(a)$;
 $M_1 \leftarrow g'(x)\mathcal{C}_f(g(x))$; // **Recursive step**
 $M_2 \leftarrow \text{algebraic_derivation_matrix}(a)$;
 $M_{o,a} \leftarrow M_1 \boxplus M_2$;
 $\mathbf{v}_0 \leftarrow (\delta_{1,i} \text{ for } i = 1, \dots, dp)$;
 // Getting the solution for the ansatz
 $\alpha \leftarrow \text{non_trivial_solution}(\text{get_linear_system}(\mathbf{v}_0, M_{o,a}))$;
return $\alpha_0 + \alpha_1\partial + \dots + \alpha_{dp}\partial^{dp}$;

Initial conditions for $h(x)$

For the initial conditions on the algebraic substitution, we can use the same formula for the iterative chain rule that we used for the composition:

$$h^{(p)}(x) = \sum_{l=1}^p f^{(p)}(a(x)) B_{p,l}(a'(x), \dots, a^{(p-l+1)}(x)),$$

where $B_{p,l}(y_1, \dots, y_{p-l+1})$ are the Bell polynomials. If we evaluate this formula at $x = 0$, we have that, in order to get the p th initial value for $h(x) = f(a(x))$ we only need the first initial conditions for $f(x)$ and $a(x)$.

The computation of the initial values of $a(x)$ can be done in a similar way as we did in the algebraic inclusion (see Subsection 4.2.4). In both cases, we represent this computation on the pseudocode with the method `init`:

$$\text{init}(f, i) \rightarrow f^{(i)}(0), \quad \text{init}(a, i) \rightarrow a^{(i)}(0).$$

The case when $p = 0$ is not included in the previous formula. However, since $h(x) = f(a(x))$ and $a(0) = 0$, we can see that $h(0) = f(0)$.

These computations are provided in the algorithm [comp_init_values](#).

Algorithm 14: comp_init_values

Input : elements $f(x) \in D^n(\mathbb{K}[x])$, $a(x)$ algebraic over $F_m(x)$ and a bound $p \in \mathbb{N}$

Output: first p initial values of $f(a(x))$

for $j = 0, \dots, p-1$ **do**

$f_j \leftarrow \text{init}(f, j);$

$a_j \leftarrow \text{init}(a, j);$

$h_0 \leftarrow \text{init}(f, 0);$

for $i = 1, \dots, p-1$ **do**

$h_i \leftarrow \sum_{l=1}^i f_l \text{bell_polynomial}(i, l)(a_1, \dots, a_{i+l-1});$

return $(h_0, \dots, h_{p-1});$

Example 6.7. Let $f(x) = \sin(\sin(x))$ (see Example 6.4 for the differential equation) and $a(x)$ algebraic over the D-finite functions with minimal polynomial

$$\frac{1}{2} \cos(x) a(x)^2 - 2a(x) + \cos(x)^2 = 0.$$

See Example 4.10 for a prove of $a(x)$ being DD-finite.

Let $h(x) = f(a(x))$. Using the algorithm `comp_operator`, we would get that $h(x)$ is D^4 -finite. However, applying algorithm `alg_substitution_operator` we can show that $h(x) \in D^3(\mathbb{K}[x])$.

In this case, we have that $W_{\circ, a}$ is generated by the tensor product:

$$\{f(a(x)), f'(a(x))\} \otimes \{1, a(x)\},$$

which leads to the following derivation matrix:

$$\begin{aligned} \mathcal{C}_{\circ, a} &= \left(a'(x) \begin{pmatrix} 0 & -\cos(a(x))^2 \\ 1 & \frac{-\sin(a(x))}{\cos(a(x))} \end{pmatrix} \right) \boxplus \left(a'(x) \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \right) \\ &= a'(x) \left(\begin{pmatrix} 0 & -\cos(a(x))^2 \\ 1 & \frac{-\sin(a(x))}{\cos(a(x))} \end{pmatrix} \boxplus \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \right) \\ &= \begin{pmatrix} 0 & a'(x) & -a'(x) \cos(a(x))^2 & 0 \\ 0 & 0 & 0 & -a'(x) \cos(a(x))^2 \\ a'(x) & 0 & -a'(x) \frac{\sin(a(x))}{\cos(a(x))} & a'(x) \\ 0 & a'(x) & 0 & -a'(x) \frac{\sin(a(x))}{\cos(a(x))} \end{pmatrix}. \end{aligned}$$

starting now with $\mathbf{v}_0 = (1, 0, 0, 0)^T$ we can see that computing 2 derivatives is

enough to find a vector in the right nullspace:

$$\mathcal{S} = \begin{pmatrix} a'(x) \cos(a(x)) & 0 & -a'(x)^3 \cos(a(x))^3 \\ 0 & 0 & 0 \\ 0 & a'(x) \cos(a(x)) & a''(x) \cos(a(x)) - a'(x)^2 \sin(a(x)) \\ 0 & 0 & 0 \end{pmatrix},$$

obtaining the vector $(\alpha_0, \alpha_1, \alpha_2)^T$ in the right nullspace of \mathcal{S} where:

$$\begin{cases} \alpha_0 = a'(x) \cos(a(x))^3 \\ \alpha_1 = a'(x)^2 \sin(a(x)) - a''(x) \cos(a(x)) \\ \alpha_2 = a'(x) \cos(a(x)) \end{cases}$$

Hence we obtain that $h(x)$ satisfies the following linear differential equation:

$$\alpha_2 h''(x) + \alpha_1 h'(x) + \alpha_0 h(x) = 0.$$

It is clear, applying the algorithm recursively, that $\sin(a(x))$ and $\cos(a(x))$ are DD-finite functions, so $h(x)$ is D^3 -finite.

6.3 Implementation of Theorem 5.17

In this last Section we focus on Theorem 5.17. This Theorem showed that any function $f \in \text{DA}(D(R))$ is also differentially algebraic over R . The proof was constructive, showing all the steps necessary for computing the non-linear differential equation.

Our interest here is to focus on those steps, provide an algorithmic description of them and also consider some performance improvements to obtain smaller differential equations. In this Section we will consider (R, ∂) to be a differential integral domain, F its field of fractions, S a differential extension and $f \in \text{DA}(D(R))$ satisfying the non-linear differential equation

$$\sum_{\alpha \in \Lambda} p_\alpha \cdot (f, \partial(f), \dots, \partial^m(f))^\alpha, \quad (6.8)$$

where $p_\alpha \in R$, $\Lambda \subset \mathbb{N}^{m+1}$ and

$$(a_0, \dots, a_m)^\alpha = \prod_{i=0}^m a_i^{\alpha_i}.$$

The main idea of the proof of Theorem 5.17 was switch the focus between the function f and the coefficients p_α . Here, since all the coefficients belong to $D(R)$, we know that the F -vector space generated by:

$$\{\partial^k(p_\alpha) : \alpha \in \Lambda \text{ and } k \in \mathbb{N}\}$$

has a finite dimension.

In fact, this dimension can only become smaller when considering bigger ground fields. Hence, we focus on the vector space

$$W = \left\langle \partial^k(p_\alpha) : \alpha \in \Lambda \text{ and } k \in \mathbb{N} \right\rangle_{F\{y\}}.$$

In the proof of Theorem 5.17, we computed several derivatives of the differential equation. Since f is solution to the equation (6.8), all the derivatives of the equation also vanish.

If we represent those derivatives as vectors in W and build a matrix, we have that the matrix has a non-trivial nullspace (since the generators that W are in that nullspace). Then the determinant of the matrix has to be zero.

Algorithmically speaking, this procedure looks exactly like the ansatz method we have been using for the different operations: we start with a vector of a differential vector space and then we compute as many derivatives of the vector to obtain a square matrix. Then we compute the determinant.

The last step (computing the determinant) is trivial. But the computation of the derivatives relies again in the theory we described in Appendix A: after computing a list of generators Φ of W , we get a derivation matrix of ∂ w.r.t. Φ . This allows us to compute derivatives in W with the formula

$$\partial(\mathbf{v}) = M\mathbf{v} + \kappa(\mathbf{v}),$$

where $\kappa((v_1, \dots, v_k)) = (\partial(v_1), \dots, \partial(v_k))$. The vector that represents the equation (6.8) in W can be used simple matrix-vector to compute all the derivatives we need.

The generators of W

The choice of the generators of W is key for this algorithm, since it determines how many derivatives we need to compute in order to reach a square matrix. However, determining the relations between the coefficients of equation (6.8) could cost time.

Let, for each $\alpha \in \Lambda$, consider d_α the order of p_α as a differentially definable function over R . The simplest choice for Φ would be the list including all the possible options:

$$\Phi = \left(\partial^k(p_\alpha) : \alpha \in \Lambda \text{ and } k = 0, \dots, d_\alpha - 1 \right).$$

This is the worst case scenario in terms of length of the generators set. On the other side, if we are able to compute a basis $\{e_1, \dots, e_k\}$ of W , we would have the minimal possible size. But computing these F -linear relations are usually not possible. This is why we focus here in a intermediate approach. We look for some relations that can be checked fast.

Namely, given two elements $g, h \in D(R)$, we can quickly check if there are constants α, β such that:

$$g = \alpha h + \beta.$$

In fact, $\alpha = g'(0)/h'(0)$ and $\beta = g(0) - \alpha h(0)$ are the only candidates. Then we only check the equality of g and $\alpha h + \beta$. Since multiplication by constants and addition of constants are two of the cheapest operations in $D(R)$ (there is no linear algebra involved), we can check this equality with the same cost of simply checking the equality $g = h$. In this way, we are able to find more relations between two function at the same cost.

Algorithm 15: linear_relation

Input : elements $g, h \in D(R)$
Output: a pair (α, β) such that $g = \alpha h + \beta$ or *Not possible*
if $h'(0) = 0$ **then**
 return *Not possible*;
 $\alpha \leftarrow g'(0)/h'(0)$;
 $\beta \leftarrow g(0) - \alpha h(0)$;
if $g = \alpha h + \beta$ **then**
 return (α, β) ;
return *Not possible*;

Then, we can extend this algorithm to not only compare two functions g and h , but also check whether this type of linear relation holds between g and any derivative of h . The algorithms `linear_relation` and `linear_relation_derivatives` perform this linear checking.

Algorithm 16: linear_relation_derivatives

Input : elements $g, h \in D(R)$ and a bound $m \in \mathbb{N}$
Output: a triplet (j, α, β) such that $g = \alpha \partial^j(h) + \beta$ or *Not found*
for $j = 0, \dots, m$ **do**
 $\text{res} \leftarrow \text{linear_relation}(g, \partial^j(h))$;
 if *not* res *is* *Not possible* **then**
 $(\alpha, \beta) \leftarrow \text{res}$;
 return (j, α, β) ;
return *Not found*;

We can now describe our simplification method:

1. For each pair $(\alpha, \beta) \in \Lambda^2$ we compute using `linear_relation_derivatives` the values of $j^{(\alpha, \beta)}$, $c_1^{(\alpha, \beta)}$ and $c_2^{(\alpha, \beta)}$ such that

$$p_\alpha = c_1^{(\alpha, \beta)} \partial^{j^{(\alpha, \beta)}}(p_\beta) + c_2^{(\alpha, \beta)}.$$

Every time a relation of this type is found, we replace p_α by it in equation (6.8), and we remove α from Λ .

2. For each α in the remaining set Λ , we check if $p_\alpha \in R$. If those coefficients exist, we do the following substitution:

$$\Lambda_1 = \Lambda \setminus \{\alpha \in \Lambda : p_\alpha \in R\} \cup \{(-1, 0, \dots, 0)\}.$$

We set $p_{(-1, 0, \dots, 0)} = 1$ and $d_{(-1, 0, \dots, 0)} = 1$.

At this step, we have Λ_1 as a reduced set from Λ (i.e., $|\Lambda_1| \leq |\Lambda|$). Moreover, we have that

$$\Phi_1 = \left(\partial^k(p_\alpha) : \alpha \in \Lambda_1 \text{ and } k = 0, \dots, d_\alpha - 1 \right)$$

is a list of generators of W . The algorithm `reduce_generators` includes this simplification method.

Derivation matrix of ∂ w.r.t. Φ_1

At this stage, we have a list of generator Φ_1 that can be written as

$$\Phi_1 = \bigoplus_{\alpha \in \Lambda_1} (p_\alpha, \dots, \partial^{d_\alpha - 1}(p_\alpha)).$$

For each summand, the derivation matrix is obtained with the companion matrix of the differential equation satisfied by the coefficient p_α (recall they are in $D(R)$). We can then put everything together with the direct sum of the matrices to build the derivation matrix with respect to Φ_1 (see Propositions A.9 and A.11 and Lemma A.12 for further information).

This shows that

$$M = \bigoplus_{\alpha \in \Lambda_1} \mathcal{C}_{p_\alpha},$$

is a derivation matrix of W w.r.t. Φ_1 .

Putting everything together, we can write down the algorithm that, given the non-linear equation (6.8) with coefficients in $D(R)$, computes another non-linear equation with coefficients in R that is satisfied by all the solutions of equation (6.8). This algorithm is `dalg_reduction`.

Example 6.9 (Non-linear equation for Mathieu functions). Recall from Example 3.5 that a Mathieu function is a DD-finite function $w(x)$ that satisfies the following differential equation:

$$w''(x) + (a - 2q \cos(2x))w(x) = 0,$$

for some constant parameters a and q . If we translate this differential equation into a differential polynomial (namely mapping $w(x) \mapsto y_0$ and $w''(x) \mapsto y_2$) we have that

$$P(y_0, y_1, y_2) = y_2 + (a - 2q \cos(2x))y_0,$$

vanishes at $y = w(x)$. The coefficient $a - 2q \cos(2x)$ is a D-finite function annihilated by the differential operator

$$\mathcal{A} = \partial^3 + 4\partial.$$

Hence, applying algorithm `reduce_generators`, we obtain 4 generators:

$$\Phi_1 = (a - 2q \cos(2x), \quad (a - 2q \cos(2x))', \quad (a - 2q \cos(2x))'', \quad 1),$$

leading to the derivation matrix

$$M = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & -4 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

and an initial vector

$$\mathbf{v}_0 = (y_0, 0, 0, y_2).$$

We can now compute three derivatives of \mathbf{v}_0 obtaining that the following matrix has determinant zero when evaluated at $w(x)$:

$$\begin{pmatrix} y_0 & y_1 & y_2 & y_3 \\ 0 & y_0 & 2y_1 & 3y_2 - 4y_0 \\ 0 & 0 & y_0 & 3y_1 \\ y_2 & y_3 & y_4 & y_5 \end{pmatrix},$$

which is a differential polynomial with constant coefficients.

Algorithm 17: reduce_generators

Input : a differential polynomial $P \in D(R)\{y\}$ **Output:** reduced list of generators Λ_1 and a list of coefficients for each generatormons \leftarrow monomials(P);**for** $p_\alpha \in$ mons **do** **for** $p_\beta \in$ mons **do** res \leftarrow linear_relation_derivatives($p_\alpha, p_\beta, \text{order}(p_\beta)$); **if** res \neq Not found **then** $j, a, b \leftarrow$ res; **if** $a \neq 1$ or $b \neq 0$ **then** **return** reduce_generators($P(p_\alpha \mapsto a\partial^j(p_\beta) + b)$); **else** mons \leftarrow remove(mons, p_β); res \leftarrow linear_relation_derivatives($p_\beta, p_\alpha, \text{order}(p_\alpha)$); **if** res \neq Not found **then** $j, a, b \leftarrow$ res; **if** $a \neq 1$ or $b \neq 0$ **then** **return** reduce_generators($P(p_\beta \mapsto a\partial^j(p_\alpha) + b)$); **else** mons \leftarrow remove(mons, p_α);simpler $\leftarrow 0$;**for** $p_\alpha \in$ mons **do** **if** $p_\alpha \in R$ **then** simpler \leftarrow simpler + coefficient(P, p_α); mons \leftarrow remove(mons, p_α); $\Phi_1 \leftarrow [\partial^i(p_\alpha) \text{ for } p_\alpha \in \text{mons and } i = 0, \dots, \text{order}(p_\alpha)]$;coefficients \leftarrow [coefficient(P, gen) for $\text{gen} \in$ generators];**if** simpler $\neq 0$ **then** $\Phi_1 \leftarrow \Phi_1 + [1]$; coefficients \leftarrow coefficients + [simpler];**return** Φ_1 , coefficients;

Algorithm 18: dalg_reduction

Input : a differential polynomial $P \in D(R)\{y\}$
Output: a differential polynomial $Q \in R\{y\}$ such that any $P(f) = 0$
 implies $Q(f) = 0$
 // Getting the reduced list of generators
 $\Phi_1, \text{coeffs} \leftarrow \text{reduce_generators}(P);$
 // Getting the basic functions
 $n \leftarrow 0;$
 $m \leftarrow \text{length}(\Phi_1);$
 $i \leftarrow 1;$
while $i < m$ **do**
 $n \leftarrow n + 1;$
 $f_n \leftarrow \Phi_1[i];$
 $i \leftarrow i + \text{order}(f_n);$
 // Computing the derivation matrix
for $i = 1, \dots, n$ **do**
 $\mathcal{C}_i \leftarrow \text{companion}(\Phi_{1,i});$
 $M \leftarrow \oplus_{i=1}^n \mathcal{C}_i;$
 // Computing the final matrix
 $\mathbf{v}_1 \leftarrow (\text{coeffs}[i] \text{ for } i = 1, \dots, m);$
for $i = 2, \dots, n$ **do**
 $\mathbf{v}_i \leftarrow M\mathbf{v}_{i-1} + \kappa(\mathbf{v}_{i-1});$
return $\det(\mathbf{v}_1 | \dots | \mathbf{v}_n);$

Chapter 7

The package `dd_functions`

In Chapters 3 and 5 we have defined and studied the concepts and properties of differentially definable functions. In Chapters 4 and 6 we also provide the algorithms to execute the closure properties automatically.

All these structures and theoretical results have been implemented as part of the PhD work in the computer algebra system Sage [68]. The package was presented in several conferences [35, 36], once the main features were fully implemented. This chapter contains the description of that implementation in the spirit of a user manual, to help the reader to get, use and even extend the package.

The particular case of D-finite functions (i.e., the differentially definable functions over the polynomials) was implemented before in several computer algebra systems. In Maple, we can find the package `gfun`[65]. In Mathematica we can use the package `HolonomicFunctions`[49].

In Sage we can find the package `ore_algebra`[42]. In this package the authors provide a full and efficient implementation of closure properties for D-finite functions. More precisely, they provide the algorithms for handling the differential operators that annihilate D-finite objects.

In our package, we have a different approach: our main object is the function itself. This function will include its defining differential equation and some initial conditions. This is important, for example, to implement properly the composition closure property (see Theorem 5.2).

The algorithms described in Chapters 4 and 5 also work for D-finite functions. However, we use the performance boost of the package `ore_algebra` in all our D-finite computations and our algorithms in any other differentially definable ring.

7.1 Sage: an Open Source system

Before detailing the implementation of our package, let us see what Sage is and why we chose it over other computer algebra systems.

Sage or SageMath is a computer algebra system developed on top of Python. It is a free open-source software that combines many other open-source packages as NumPy, SciPy, Maxima, Flint, R and others.

All these packages and Sage itself are under the GNU license. This license encourages the sharing and distribution of its content. No money is gained under this licence and it is not possible to make profit using third party GNU software. Since the main aim of the package `dd_functions` is research, we found this setting to be the best choice.

Since using Sage is already free, providing our package with an Open Source licence is convenient to have a nicer, fast and simple distribution of the ideas. Moreover, it allows everyone to get a grasp of the concepts of the thesis through its documentation and also allows performing experiments or computations in educational centers without any expense.

The fact that Sage is based on Python allows the user to use Python functionalities within Sage. Moreover, this also allows Python standalone users to use the packages from Sage. We refer to the SageMath documentation [68] for further details about its functionality.

The last versions of Sage (9.0 and later) are based on Python 3. This made a big difference on syntax and functionalities for basic Python. Since the change is recent, there are some packages designed for previous versions of Sage that do not work properly in the latest version.

In fact, this update was done while we were developing of our package. That is why in the git repository (see more information below) we offer an old version of the package compatible with Sage 8.8 (i.e., still based on Python 2). For getting the last and best implementation of the package, the user needs to have Sage 9.0 and Python3 (or later versions) installed.

We consider Sage as a good computer algebra system for the development of the package `dd_functions` not only because it is open source. Sage is developed under the paradigm *Parent-Element*. This is based on the Object Oriented nature of Python and it distinguishes between two types of objects: *parent* structures (like rings, fields, vector spaces, groups, etc.) and their actual *elements*.

In this paradigm, the *parent* structures have information about the type of elements that they contain. For example, if we consider the ring of polynomials $\mathbb{Q}[y]$, the object in Sage that represents it will contain information about the coefficients (namely \mathbb{Q}) and the name of the variable (in this case y). These objects always have a method to check whether an object is an element in that structure or not.

On the other hand, the *element* objects are the particular elements in a particular *parent*. There can not be an *element* without a *parent*. Then, the *element* class has all the methods that are intrinsic for these elements. For example, $3y + 1$ will be an element of $\mathbb{Q}[y]$. Inside this object, we may find all the information about this polynomial (in this case a list of coefficients) and all the methods to compute with this polynomial (such as a method for the sum or product of polynomials).

The coercion system on Sage is the culmination of this paradigm. The coercion system allows the system to mix elements from different parent structures if such parents are related. As an example, we may see $1/7$ as an element of \mathbb{Q} or as a polynomial in $\mathbb{Q}[y]$. If properly implemented, when we try to perform an operation over elements from different parents, the system asks their parents if they can be viewed as elements of the same parent structure. In that case, the system performs these conversions automatically and then makes the computation the user was looking for.

This coercion system is an excellent feature of Sage. Since a polynomial can be viewed as an element in so many rings, it allows the user to perform operations without taking care of the exact ring where the operation actually takes place. Implementing new structures using this framework can be difficult. However, it is a feasible and fair price to pay.

7.2 Installing the package

In order to use the package `dd_functions`, the user needs to install it in the Sage system. We assume the user has already installed Sage 9.0 or a later version.

The package `dd_functions` is managed using a *git* repository hosted on the platform GitHub. This platform offers free open access to many repositories and provides all the functionality for a *git* system for version control.

In this repository we may find all the files to install the package. We can also find the documentation (either in its web version or the actual files) and a demo that allows the reader to try out the package even without installing any program on his system. Only internet connection is required.

The reader can find the repository at the following link:

https://github.com/Antonio-JP/dd_functions

On the main page of the GitHub repository, a README file can be found detailing the main steps for installing the packages. However, we recommend the automatic installation using PiPy.

In a terminal where the Sage installation is available via the command `sage`, the user can type the following line to automatically install the package and all its dependencies:

```
sage -pip install [--user] git+https://github.com/Antonio-JP/dd_functions.git
```

The optional argument `--user` is only needed if you prefer to have a local installation of the package (only valid for your user). Otherwise, it will be installed in the main folder of your Sage installation with all the other packages.

A different option for installing the package is to get a clone of the repository and use PiPy or the *Makefile* command `install`. For cloning the package, the reader can use any *git* client to clone the following URL:

https://github.com/Antonio-JP/dd_functions.git

Once installed, the package can be loaded in any Sage session with the following command

```
sage: from ajpastor.dd_functions import *
```

In order to read the documentation of the package, the reader can clone the repository as we have just mentioned and run the *Makefile* command `doc` to generate their local copy of the documentation. The user can also access it directly using his web browser from the URL:

https://antonio-jp.github.io/dd_functions/docs

The repository also make use of the functionality of Binder ¹ to offer an interactive demo available for everyone through their web browser. In this case, the link for accessing that demo is:

https://mybinder.org/v2/gh/Antonio-JP/dd_functions.git/v0.7-sage8.8?filepath=dd_functions_demo.ipynb

7.3 The main user interface

In Chapter 3 we introduced the concept of *differentially definable* functions and discussed the formal power series solutions that are in this class. In the current implementation of the package `dd_functions`, we focused exclusively on this particular type of solutions, although the system is general enough to accept and represent differentially definable functions that are not formal power series. However, we do not guarantee full functionality for those objects.

¹<https://mybinder.org/>

7.3.1 Parent class: `DDRing`

In order to fit into the *parent-element* paradigm, we need to provide a class for representing the Parent structure. In this case, we are working with differentially definable functions over a ring R within an ambient integral domain S . We denoted this ring with $D_S(R)$ and that is precisely what the class `DDRing` represents.

In order to create a `DDRing` the user must provide the integral domain R . This ring must have a derivation operation (in order to have a differential integral domain). If C is the ring of constants of R , we always set the ambient integral domain S to be $C[[x]]$.

For example, if we want to create the ring of differentially definable functions over \mathbb{Q} (i.e., $D_{\mathbb{Q}[[x]]}(\mathbb{Q})$), we can use the following code:

```
sage: DDRing(QQ)
DD-Ring over (Rational Field)
```

If we put $\mathbb{Q}[x]$ as an input, we create the rings of D-finite functions. Since all objects of the class `DDRing` are by definition differential integral domains (as proven in Chapter 3), we can use that object to create the ring of DD-finite functions.

```
sage: DQ = DDRing(QQ[x]); DQ
DD-Ring over (Univariate Polynomial Ring in x
over Rational Field)
sage: DDQ = DDRing(DQ); DDQ
DD-Ring over (DD-Ring over (Univariate Polynomial
Ring in x over Rational Field))
```

These two rings are the main object of study of the thesis. We provide two default variables (`DFinite` and `DDFinite`) for representing the ring of D-finite and DD-finite functions respectively.

```
sage: DQ == DFinite
True
sage: DDQ == DDFinite
True
```

To simplify notation and help the user to construct differentially definable rings, we allow more arguments to the class `DDRing`:

- **depth**: natural number n to build $D^n(R)$ directly.
- **derivation**: if the user wants to use a ring R in Sage that has no derivation method, this argument allows to indicate a particular derivation over R .

In the following example, we can see a comparison between different constructions that lead to the same ring:

```

sage: DDRing(QQ[x],depth=2) == DDFinite
True
sage: DDRing(DDRing(QQ[x], depth=2), depth=4) ==
....: DDRing(QQ[x],depth=6)
True

```

Finally, we also allow to extend the constants with some parameters. Since these parameters may be evaluated at some point, we provide a different class that includes these methods for evaluating the parameters: `ParametrizedDDRing`.

In order to build a differentially definable ring with parameters, we first need to have its corresponding `DDRing` without parameters and then indicate the names that the parameters will have.

These parameters are always constants with respect to the derivation and, internally, these parameters are considered algebraically independent. Arbitrary evaluation of these parameters may lead to unexpected results.

In the following example, we create the DD-finite functions with two parameters a and q :

```

sage: DDFiniteAQ = ParametrizedDDRing(DDFinite, ["a","q"])
sage: DDFiniteAQ
DD-Ring over (DD-Ring over (Univariate Polynomial Ring in x
over Rational Field)) with parameters (a, q)
sage: DDFiniteAQ.parameters()
[a, q]
sage: DDFiniteAQ.base_field()
Fraction Field of Multivariate Polynomial Ring in a, q over
Rational Field

```

We can also create the ring of D-finite functions with parameters a and q and then extend it using the class `DDRing` to build the same ring:

```

sage: DDRing(ParametrizedDDRing(DDFinite, ["a","q"])) ==
....: DDFiniteAQ
True

```

The class `DDRing` also provides a method `to_depth` that allows the user to create the ring $D^n(R)$ for any depth from the `DDRing` itself.

```

sage: DFinite.to_depth(2) == DDFinite
True
sage: DFinite.to_depth(5) == DDFinite.to_depth(5)
True

```

7.3.2 Element class: DDFunction

The class we are specially interested in is the *element* class of a *DDRing*. This element class is called *DDFunction*.

In Chapter 3 we saw that, in order to characterize one particular formal power series solution of a linear differential equation, we only need to know the differential equation and some initial conditions. This is precisely the data structure behind our *DDFunction*.

Assume we want to create $f(x) \in D(R)$. From the corresponding *DDRing* object, we need to use the method `element`. We provide as arguments the list of the coefficients of the differential equation that $f(x)$ satisfies (which are elements in R) and a list with initial conditions of the form $[f(0), f'(0), \dots, f^{(n)}(0)]$.

$$\begin{array}{c} (r_0, \dots, r_n), \\ (\alpha_0, \dots, \alpha_d) \end{array} \xrightarrow{\text{element}} f(x) : \begin{cases} r_0(x)f(x) + \dots + r_n(x)f^{(n)}(x) = 0, \\ f(0) = \alpha_0, f'(0) = \alpha_1, \dots, f^{(n)}(0) = \alpha_n \end{cases}$$

Then, creating functions such as e^x , $\sin(x)$ or $\cos(x)$ is a straightforward process:

```
sage: sine = DFinite.element([1,0,1],[0,1]) # sin(x)
sage: cosine = DFinite.element([1,0,1],[1,0]) # cos(x)
sage: expo = DFinite.element([-1,1],[1]) # exp(x)
```

And, once we have defined D-finite objects, we can use them to build DD-finite objects, like $\tan(x)$. In the example below, we create three different tangents using the three equations we got in Examples 3.4, 3.7 and 3.15:

```
sage: tan1=DDFinite.element([-2,0,cosine*cosine],[0,1])# (3.4)
sage: tan2=DDFinite.element([0,-2*sine, cosine],[0,1])# (3.7)
sage: tan3=DDFinite.element([-1, sine*cosine],[0,1])# (3.15)
```

As an illustration on how we can work with parameters, we show in the next example how to create the Mathieu functions 3.5. Recall that the Mathieu function $w(x)$ satisfies the following differential equation

$$w''(x) + (a - 2q \cos(2x))w(x) = 0.$$

We need first to create the object $\cos(2x)$. Since $\cos''(x) = -\cos(x)$, it is easy to see that $\cos''(2x) = -4\cos(2x)$. We can then build $w(x)$ in our package:

```
sage: cos2x = DFinite.element([4, 0, 1], [1,0]) # cos(2x)
sage: a,q = DDFiniteAQ.parameters() # getting parameters
sage: w = DDFiniteAQ.element([a - 2*q*cos2x,0,1])
```

We can see in this example that the function $w(x)$ has no input for initial conditions. When this happens, the package considers the object as the set of all solutions to the differential equation. We can still perform operations with this function symbolically, but whenever we need to extract initial conditions an error is raised.

Moreover, we saw that the object for $\cos(2x)$ was created as a D-finite function without parameters and then we mixed it with the parameters a and q . This is an example of how the coercion system of Sage helps the user. We know theoretically that D-finite functions are included in the D-finite functions with two parameters. This is detected automatically within the code and the casting of the structure to fit the new parent structure is performed without concern of the user. If at any point the coercion system fails, then an error is raised.

The class `DDFunction` includes several utility methods to read the structure that defines a particular functions:

- **equation:** returns the object that defines the linear differential equation satisfied by the function. This object can be explored further to get each of the coefficients of the differential equation and recover the list that was used as input of `DDFunction`.
- **getInitialValue:** given an integer n , it returns the n th initial condition for the function, i.e., $f^{(n)}(0)$.
- **getSequenceElement:** since `DDFunction` is always a formal power series, given an integer n , this method returns its n th coefficient, i.e., $[x^n]f(x)$. Note that this output is related to the method `getInitialValue` with the formula

$$[x^n]f(x) = \frac{1}{n!}f^{(n)}(0).$$

- **getOrder:** returns the order of the differential equation that defines the `DDFunction`.
- **min_coefficient:** returns the trailing coefficient of $f(x)$, i.e., the first non-zero coefficient of $f(x)$ as a formal power series. If the object represents the zero function, it returns 0.
- **zero_extraction:** computes the order of $f(x)$ as a formal power series and returns it together with the associated formal power series of order 0. See Lemma 3.14, point (iii) for the proof that this associated formal power series is in the same `DDRing`.
- **change_init_values:** returns a new `DDFunction` that satisfies the same linear differential equation but has different initial conditions.

In the next example we can see how we extract information for the function $\tan(x)$ from Example 3.4:

```
sage: tan1.equation
FullLazyOperator([-2, 0, (3:3:3)DD-Function in (DD-Ring over
(Univariate Polynomial Ring in x over Rational Field))])
```

Here we see that the differential operator is represented with a class called `FullLazyOperator`. This class implements all the performance considerations explained in Section 4.3. With the following syntax we can extract the coefficients:

```
sage: tan1.equation[0] # First coefficient
-2
sage: tan1.equation[1] # Second coefficient
0
sage: tan1.equation[2] # Third coefficient
(3:3:3)DD-Function in (DD-Ring over (Univariate Polynomial
Ring in x over Rational Field))
sage: tan1.equation[2] == cosine*cosine
True
```

The following example shows the relation between the two methods for extracting coefficients of the power series:

```
sage: all(
.....:     tan1.getInitialValue(n) ==
.....:     tan1.getSequenceElement(n)*factorial(n)
.....: for n in range(20))
True
```

And also, we can check that all three representations of the tangent provide the same sequence up to the coefficient 50:

```
sage: l1 = tan1.getInitialValueList(50)
sage: l2 = tan2.getInitialValueList(50)
sage: l3 = tan3.getInitialValueList(50)
sage: l1 == l2 and l1 == l3
True
```

The method `change_init_values` is particularly useful for extending the information of an incomplete `DDFunction`. Recall how we have already defined the Mathieu function $w(x)$ but without any information concerning the initial conditions. This method will allow us to provide this information after the creation of the object:

```
sage: wSine = w.change_init_values([0,1])
sage: wSine.equation == w.equation
True
sage: wSine.getInitialValueList(4)
[0, 1, 0, -a + 2*q]
```

We can also check that with this method we can move from $\sin(x)$ to $\cos(x)$:

```
sage: sine.change_init_values([1,0]) == cosine
True
```

7.4 Computing closure properties

In Chapters 3 and 5 we have proved that differentially definable functions are closed under several operations and, then, we can compute a corresponding differential equation and initial values for the resulting functions. In the code examples of Section 7.3 we have already used some of these operations. We include here a comprehensive description of all the implemented closure properties and how they can be computed with the package `dd_functions`.

7.4.1 Arithmetic and differential closure properties

In Chapter 3 we proved the set of differentially definable functions is closed under addition and multiplication. We also explained the algorithms to compute the new differential equations in Chapter 4.

The package `dd_functions` allow the automatic computation of these closure properties and more with the following methods:

- **Addition:** the user can call the method `add` of a `DDFunction` and use the other `DDFunction` as an argument. The user can also use the usual syntax in Python for the addition (+).
- **Scalar multiplication:** since the multiplication by a constant is easier to perform than the usual multiplication for a `DDFunction`, we provide a quicker method `scalar` to perform this operation. The user can also use the usual syntax in Python for the multiplication (*) and the package will detect if one of the objects is constant or not.
- **Subtraction:** this is a shortcut for the addition of the two objects where the second is multiplied by -1 . Here we can use the method `sub` or the usual syntax in Python for the subtraction (-).
- **Multiplication:** the user can call the method `mult` of a `DDFunction` and use the other `DDFunction` as an argument. The user can also use the usual syntax in Python for the multiplication (*).

- **Division:** the division is not always well defined in the ring of formal power series. It requires that the order of the numerator (as a formal power series) is greater than the order of the denominator. Moreover, the multiplicative inverse of a `DDFunction` may not be in the same `DDRing`. This method (called `div`) checks all these cases and extends the `DDRing` if needed in order to compute the division of two `DDFunction`. The user can also use the usual syntax in Python for the division (`/`).
- **Exponentiation:** we can compute $f(x)^n$ for any $n \in \mathbb{Z}$ using the method `pow` of any `DDFunction` or the usual syntax in Python for exponentiation (`^`).
- **Equality:** being able to check that two objects represent the same function is critical in this package. The main feature (proving identities) is based in this operation. This equality can be checked with the method `equals` or the usual Python syntax (`==`).

Checking equality is equivalent to checking if an object represent the zero function. This zero recognition process can be easily done thanks to the work on Section 3.3. Using the results in that section, we only need to check a finite amount of initial conditions.

For this zero recognition we provide a method in the class `DDFunction` called `is_zero`. We also provide a method `is_constant` to check whether a `DDFunction` is constant or not.

Also in Chapter 3, we saw that the derivation is closed in a differentially definable ring. So is the indefinite integral. Moreover, there were closed formulas for the differential equation that these functions satisfied. These operations are implemented in the following methods of `DDFunction`:

- **derivative:** this method accepts an optional argument `times` that allows to compute higher order derivatives of a `DDFunction`. Given `times=n`, this method returns the n th derivative.
- **integrate:** this method returns the indefinite integral. Since the value at zero (i.e., the integration constant) is not fixed, this method accepts a value for that integration constant. If not provided, zero is taken as a default value.

```
sage: all(
....:     expo.derivative(times=i) == expo
....:     for i in range(10))
True
```

```

sage: sine.derivative() == cosine
True
sage: sine.derivative(times=2) == -sine
True
sage: cosine.derivative() == -sine
True
sage: tan1.integral(10) == tan1.integral() + 10
True
sage: tan1.derivative()*cosine^2 == 1
True
sage: w.derivative(times=2) == -w.equation[0]*w
True

```

7.4.2 Composition of `DDFunction`

In Chapter 5 we studied how the composition of two differentially definable functions behave. In that case, we saw that if $f(x) \in D^n(R)$ and $g(x) \in D^m(R)$, then $f(g(x))$, if it is well-defined, belongs to $D^{n+m}(R)$.

With the structures `DDRing` and `DDFunction`, it is now easy to implement this property and compute the differential equation of $f(g(x))$ as an element of $D^{n+m}(R)$ using the algorithm described in Chapter 6.

There are two ways to perform the composition in the package `dd_functions`: either use the method `compose` of the class `DDFunction`, or use the magic method `__call__` of Python and its short notation (similar to calling a function).

```

sage: cosine.compose(2*x) # method 'compose'
(2:2:2)DD-Function in (DD-Ring over (Univariate Polynomial
Ring in x over Rational Field))
sage: cosine.compose(2*x) in DFinite
True
sage: cosine(2*x) # magic syntax
(2:2:2)DD-Function in (DD-Ring over (Univariate Polynomial
Ring in x over Rational Field))
sage: cosine(2*x) == cos2x
True

```

The composition checks the condition for performing the operation (i.e., checks whether $g(0) = 0$) and computes the final ring to which the composition will belong.

Due to performance issues, each time a computation reaches $D^3(\mathbb{Q}[x])$, we print a warning. This warning does not stop any computations, but allows the user to know that these computations may take a very long time.

7.5 Built-in examples

In previous sections we have explained how to create elements in our `DDRing` structure starting from the differential equation. However, in most of the cases the user has a closed form expression for the D-finite coefficients or even for the DD-finite functions they want to create.

In the subpackage `ddExamples` we provide a list of special functions that are D-finite or DD-finite and give the user a simpler syntax for creating examples.

We encourage the reader to check the documentation of this subpackage:

https://antonio-jp.github.io/doc/dd_functions/dd_functions/ddExamples.html

where we provide links to several sources explaining each function in detail and providing some identities as tests of the whole package. Moreover, we are planning to extend the list of examples in the near future. The reader can find the updated list in the previous documentation web page.

7.5.1 Exponential and trigonometric functions

Exponential, logarithmic and trigonometric functions are basic examples in the set of D-finite and DD-finite functions. We provide several methods to create these particular functions with the biggest flexibility we could allow.

All these methods receive an argument that must be a function $f(x)$ such that $f(0) = 0$. This function $f(x)$ can have any of the following structures:

- A symbolic expression where x is a factor. All variables in the expression will be interpreted as parameters.
- A polynomial $p(x_1, \dots, x_m)$. Here we require that x_1 divides the polynomial and then x_1 will take the role of x as main variable. All other variables will be treated as parameters.
- A `DDFunction` object. Then these methods return the composition of the corresponding special function with that `DDFunction`.

The methods to create these exponentials and trigonometric functions are the following:

- `Exp`: it creates the exponential function e^x .
- `Sin`: it creates the sine function $\sin(x)$.
- `Cos`: it creates the cosine function $\cos(x)$.

- **Tan**: it creates the tangent function $\tan(x)$. This function, as we have seen throughout the thesis, is DD-finite, so we recommend caution when composing this function with other D-finite objects.
- **Sinh**: it creates the hyperbolic sine function $\sinh(x)$.
- **Cosh**: it creates the hyperbolic cosine function $\cosh(x)$.
- **Tanh**: it creates the hyperbolic tangent function $\tanh(x)$. Similarly to the tangent, the hyperbolic tangent is DD-finite, so we recommend caution when composing this function with other D-finite objects.

Example 7.1 (Relation between $\sin(x)$ and e^x). It is well known that $\sin(x)$ and $\cos(x)$ can be expressed in terms of the exponential function using the imaginary number i in the exponent. In this example we show how we can check that identity with our package.

We first include i as an element of our base field (extending \mathbb{Q}). Then, we create the ring $D(\mathbb{Q}[i])$ and the exponential functions e^{ix} and e^{-ix} . Using the arithmetic properties, we can easily check the identity:

```
sage: Qi = NumberField(QQ['i']('i^2+1'), 'i'); i = Qi('i')
sage: Qix = Qi[x]; x = Qix(x)
sage: DFiniteI = DDRing(Qix)
sage: Sin(x) == (Exp(i*x) - Exp(-i*x))/(2*i)
True
```

Example 7.2 (Trigonometric relation with different argument). Another well known identity involving trigonometric functions is that

$$\sin(x)^2 + \cos(x)^2 = 1, \text{ for all } x.$$

And also the following identity with the hyperbolic trigonometric functions

$$\sinh(2x) = 2 \sinh(x) \cosh(x), \text{ for all } x.$$

Then, we can prove with our package the following trivial identity:

$$\sin(\sinh(2x))^2 + \cos(2 \sinh(x) \cosh(x))^2 = 1,$$

using the following code:

```
sage: Sin(Sinh(2*x))^2 + Cos(2*Sinh(x)*Cosh(x))^2 == 1
True
```

It is interesting to note that all the functional inverses of these functions (i.e., those functions such the composition yields the identity function) are also D-finite. We provide similar methods for these inverses that accept the same input as the previous methods:

- **Log**: creates the logarithmic function. Since $\log(x)$ has no formal power series expansion at $x = 0$, this method requires as input a function that takes value 1 in $x = 0$.
- **Log1**: builds the function $\log(\bullet + 1)$ and accepts as argument a function that evaluates zero at $x = 0$.
- **Arcsin**: the functional inverse of the sine ($\arcsin(x)$).
- **Arccos**: the functional inverse of the cosine ($\arcsin(x)$).
- **Arctan**: the functional inverse of the tangent ($\arctan(x)$).
- **Arcsinh**: the functional inverse of the hyperbolic sine ($\operatorname{arcsinh}(x)$).
- **Arccosh**: the functional inverse of the hyperbolic cosine ($\operatorname{arccosh}(x)$).
- **Arctanh**: the functional inverse of the hyperbolic tangent ($\operatorname{arctanh}(x)$).

Using the composition of `DDFunction` we can easily check that these functions are, in fact, the compositional inverses:

```
sage: Sin(Arcsin(x)) == x
True
sage: Arcsinh(Sinh(x)) == x
True
sage: Exp(Log(x+1)) == x+1
True
sage: Log(Exp(x)) == x
True
```

7.5.2 Special functions: parametric functions

There is a wide class of special functions that are D-finite [4, 5, 26]. We include some of them in the package so the user does not need to look up the differential equations these special functions satisfy. We can split them into two main categories:

1. *Parametric families of functions*: these are families $f_n(x)$ such that for each n we obtain a different function. We could use the class `ParametrizedDDRing` to represent the differential equation but in these cases the initial conditions are not *appropriate*, meaning that either they are not rational expressions in n or that the initial conditions required change for each $f_n(x)$. These methods require a fixed value of $n \in \mathbb{N}$ as argument.
 - `BesselD`: method that returns the Bessel function of first kind ($J_n(x)$).
 - `LegendreD`: method that returns the Legendre polynomial of first kind (which are a family of orthogonal polynomials) $P_n(x)$.
 - `ChebyshevD`: method that returns the Chebyshev polynomial of first kind (which are a family of orthogonal polynomials) $T_n(x)$.
2. *Functions with particular parameters*: in this case the dependency with respect with the parameters is simpler: they are constants and the initial conditions can be expressed as polynomial in those parameters. In the code, these parameters must be provided as constants (i.e., they should not depend on x) or strings that will be interpreted as the name of the parameter. In the latter case, we create the corresponding `ParametrizedDDRing` before returning the function. In this category we may find the following special functions
 - `MathieuD`: returns a Mathieu function (see Example 3.5). It takes three arguments: the value for a , the value for q and the initial conditions. The user can use the shorter notation `MathieuSin` and `MathieuCos` if the initial conditions are $(0, 1)$ and $(1, 0)$ respectively.
 - `GenericHypergeometricFunction`: returns the hypergeometric function ${}_pF_q$ (see Example 2.22) given the corresponding parameters and initial conditions.
For the particular case of the Gaussian hypergeometric function (i.e., ${}_2F_1$), we provide also the method `HypergeometricFunction`.

Example 7.3 (Simplifying DD-finite functions). Recall the hyperexponential function described in Example 5.3:

$$f_a(x) = {}_2F_1 \left(\begin{matrix} a, -a \\ 1/2 \end{matrix} ; \frac{1}{2}(1 - \cos(x)) \right),$$

where a was a parameter. We can build this function easily with our package and we can see what the initial values are:

```
sage: a = var('a')
sage: arg = (1-Cos(x))/2
sage: fa = HypergeometricFunction(a, -a, 1/2)(arg)
sage: fa.getInitialValueList(6)
[1, 0, -a^2, 0, a^4, 0]
```

This list of initial values should look familiar. In fact, if we compute the function $\cos(ax)$ we can see that the initial conditions are exactly the same:

```
sage: cosA = Cos(a*x)
sage: cosA.getInitialValueList(6)
[1, 0, -a^2, 0, a^4, 0]
```

With our package we can check that, indeed, they are the same function:

```
sage: fa == cosA
True
```

which proves that $f_a(x)$ is not only DD-finite, but even D-finite for all possible values of $a \in \mathbb{C}$.

Example 7.4 (Mathieu Wronskian). Consider the two fundamental solutions $w_1(x)$ and $w_2(x)$ of the Mathieu differential equation with initial conditions $(1, 0)$ and $(0, 1)$ (what we have called the Mathieu sine and the Mathieu cosine respectively). We can prove that for all values of a and q , the Wronskian is identically 1.

Recall (see Definition B.25 for a precise definition) that the Wronskian of $w_1(x)$ and $w_2(x)$ is the following determinant:

$$\begin{vmatrix} w_1(x) & w_2(x) \\ w_1'(x) & w_2'(x) \end{vmatrix} = w_1(x)w_2'(x) - w_2(x)w_1'(x)$$

We can do this computation ourselves using the following code:

```
sage: w1 = MathieuCos()
sage: w2 = MathieuSin()
sage: w1*w2.derivative() - w2*w1.derivative() == 1
True
```

Equivalently, we can use the code in Sage for the Wronskian to obtain exactly the same result:

```
sage: wronskian(MathieuCos(), MathieuSin()) == 1
True
```

7.5.3 Special functions: non-parametric functions

There are other classes of examples provided in the package that does not depend on any parameters, only the initial conditions or other type of arguments. These examples are:

- **RiccatiD**: in Subsection 5.3.1 we showed that for any three functions $a(x)$, $b(x)$, $c(x)$ in some $D^n(R)$, the solutions to the differential Riccati equation

$$f'(x) = c(x)f(x)^2 + b(x)f(x) + a(x),$$

are elements in $D^{n+3}(R)$. This method computes the **DDFunction** representation of a particular solution (provided an initial condition), deciding the ambient space automatically.

- **AiryD**: the Airy functions are defined as solutions to the D-finite differential equation

$$f''(x) - xf(x) = 0.$$

This method, provided the initial conditions, returns the corresponding Airy function.

- **PolylogarithmD**: the s -polylogarithm is defined as the generating function of the sequence n^{-s} . This method receives an integer s and return the s -polylogarithm.

Example 7.5 (Riccati generalizes the Tangent). The Riccati differential equation is one of the few cases where we have proven that a non-linear differential equation can be transformed into a linear differential equation, proving its solutions are D^n -finite functions for some n .

It is a classical result that the tangent satisfies a Riccati type differential equation:

$$\tan(x)' = \tan(x)^2 + 1.$$

We can check that the method **RiccatiD** returns the tangent function when the arguments are the appropriate:

```
sage: f = RiccatiD(1,0,1,0) # f'(x) = f(x)^2 + 1; f(0) = 0
sage: f == Tan(x)
True
```

Finally, we also provide more examples that come from combinatorial sequences:

- **FactorialD**: returns the D-finite representation of the generating function of the sequence $(n!)_n$.
- **CatalanD**: returns the D-finite representation of the generating function of the Catalan numbers [66].
- **FibonacciD**: given the values for F_0 and F_1 , returns the D-finite representation of the generating function for the Fibonacci numbers starting with F_0 and F_1 .
- **BellD**: returns the DD-finite representation of the exponential generating function for the Bell numbers [32, Chapter II.3]. For this particular sequence, it was proven by Klazar [46] that the ordinary generating function is not D^n -finite.
- **BernoulliD**: returns the DD-finite representation of the exponential generating function for the Bernoulli numbers [4, Chapter 23]. In this case, it is well known that this exponential generating function is equal to $x/(e^x - 1)$.

```
sage: FibonacciD((0,1)).getSequenceList(10)# Fibonacci numbers
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
sage: B = BernoulliD()
sage: B == x/(Exp(x)-1)
True
sage: B.getInitialValueList(10)# Bernoulli numbers
[1, -1/2, 1/6, 0, -1/30, 0, 1/42, 0, -1/30, 0]
sage: BellD().getInitialValueList(10)# Bell numbers
[1, 1, 2, 5, 15, 52, 203, 877, 4140, 21147]
```

7.6 Relation to D-algebraic functions

In this last section we show the features included in the package concerning the D-algebraic functions and their relation to the differentially definable functions.

Recall that in Theorem 5.17 we showed that differentially definable functions are always D-algebraic with coefficients in the base domain. In fact, the proof of that theorem, as shown in Chapter 6, is constructive. The method that put together all the algorithms described in that Chapter is called `diff_to_diffalg`.

This method has three different arguments:

- **func**: the `DDFunction` for which we want to compute a D-algebraic equation.
- **varname**: an optional argument (the default value is "y"). This is the name of the variable for the ring of differential polynomials. The method takes care of checking if it is a valid name.

- **constant**: an optional argument (the default value is **True**). While reducing the coefficients, we can either stop with polynomial coefficients (when this argument is **False**) or guarantee constant coefficients in the D-algebraic equation (when the argument is **True**).
- **infinite**: an optional argument (the default value is **False**). If set to **True**, the output of the method will be a polynomial in the ring of differential polynomials (which is an infinite variable polynomial ring). Otherwise, the output is casted to a particular ring with finitely many variables.
- **debug**: an optional argument (the default value is **False**). If set to **True**, the method will print useful information, such as the the matrix whose determinant is computed.

```
sage: diff_to_diffalg(Exp(Exp(x)-1))
-y_2*y_0 + y_1^2 + y_1*y_0
sage: diff_to_diffalg(AiryD(1,0))
y_3*y_0 - y_2*y_1 - y_0^2
sage: diff_to_diffalg(AiryD(1,0), constant=False)
y_2 + (-x)*y_0
```

The differential equation returned by this method does not depend on the initial conditions of the function we use as argument. This means that the set of solutions to the non-linear differential equation has to include all possible combinations of initial values of the linear differential equation.

Example 7.6 (Non-linear equations for the Tangent). We have three different DD-finite representations for the tangent, meaning that we can obtain three different non-linear differential equations for it:

```
sage: diff_to_diffalg(tan1)
2*y_5*y_2^2*y_0 + (-12)*y_4*y_3*y_2*y_0 + 6*y_4*y_2^2*y_1 +
12*y_3^3*y_0 + (-12)*y_3^2*y_2*y_1 + 4*y_3*y_2^3 +
8*y_3*y_2^2*y_0 + (-8)*y_2^3*y_1
sage: diff_to_diffalg(tan2)
(-2)*y_5*y_3*y_1^2 + 2*y_5*y_2^2*y_1 + 3*y_4^2*y_1^2 +
(-10)*y_4*y_3*y_2*y_1 + 4*y_4*y_2^3 + 6*y_3^3*y_1 +
(-3)*y_3^2*y_2^2 + 4*y_3^2*y_1^2 + (-8)*y_3*y_2^2*y_1 +
4*y_2^4
sage: diff_to_diffalg(tan3)
y_4*y_1^2*y_0 + (-6)*y_3*y_2*y_1*y_0 + 2*y_3*y_1^3 +
6*y_2^3*y_0 + (-3)*y_2^2*y_1^2 + 4*y_2*y_1^2*y_0 +
(-4)*y_1^4
```

The difference between these three non-linear equations is significant. Just summarizing the order, the degree and the number of terms of the differential equations we obtain:

- For the first example (see Example 3.4): order 5, degree 4, 8 terms
- For the second example (see Example 3.7): order 5, degree 4, 10 terms.
- For the third example (see Example 3.15): order 4, degree 4, 7 terms.

None of these equations is close to the *minimal* equation for the tangent (namely, its Riccati differential equation $y'(x) = 1 + y(x)^2$).

Example 7.7 (Non-linear equation for Mathieu functions). Consider a solution $w(x)$ to the Mathieu differential equation. Recall that this function satisfied the differential equation

$$w''(x) + (a - 2q \cos(2x))w(x) = 0.$$

We can apply our code to get a non-linear differential equation that is satisfied by it:

```
sage: diff_to_diffalg(w)
-y_5*y_0^3 + 3*y_4*y_1*y_0^2 + 4*y_3*y_2*y_0^2 +
(-6)*y_3*y_1^2*y_0 + (-4)*y_3*y_0^3 +
(-6)*y_2^2*y_1*y_0 + 6*y_2*y_1^3 + 4*y_2*y_1*y_0^2
```

Note that in this equation the parameters a and q do not show up, even if they appeared in the differential equation of $w(x)$. This means that all Mathieu functions, no matter which values for a and q we choose satisfy exactly the same non-linear differential equation.

Chapter 8

Ongoing and future work

During this thesis we have worked with the definition of differentially definable functions (see Chapter 3). They have the possibility of a finite representation allowing us to get a package where we can automatically work with them and prove symbolically identities (see Chapter 7).

We also looked into new related problems in the world of combinatorics that exceed the holonomic framework. Guided with the structural results in Chapter 5, we started a study of several non-holonomic problems. We present here the recent progress in these problems.

Moreover, during this thesis (see Chapter 5) we saw that there are plenty of open theoretical questions. In this Chapter, we collect and organize all the open questions stated in the thesis and provide some thoughts about them.

8.1 Deeper study of the D^∞ domain

In Chapter 5 we defined the integral domain $D_S^\infty(R)$ as the union of the sequence $D_S^n(R)$ for $n \in \mathbb{N}$. We can see, from Appendix B, that this limit is closely related to the *Picard-Vessiot closure* of a field. In our case, instead of considering the existence of a theoretical field, we consider this closure within an ambient space.

This last remark is important when studying the domain $D_S^\infty(R)$. A Picard-Vessiot closure always have all the linearly independent solutions to linear differential equations. In the case of the limit ring $D_S^\infty(R)$, this may not be the case.

All the constructions and algorithms described in Chapters 4 and 6 show that if R is a computable differential integral domain, then $D_S^\infty(R)$ is also computable. However, understanding the structure of this integral domain is a difficult problem. Related with this domain, we propose three different questions:

- How does the limit ring change w.r.t. the domain R ?

- How does the limit ring change w.r.t. to the ambient domain S ?
- When is the limit ring?

Changing the base domain

For this problem, we consider an ambient space S and two differential integral domains R_1 and R_2 . The question is

Under which conditions do we have $D_S^\infty(R_1) = D_S^\infty(R_2)$?

The main example we can show here are the limit rings for C-finite functions and D-finite functions. Since $\mathbb{K}[x] \subset D(\mathbb{K})$, then it is clear that

$$D^\infty(\mathbb{K}[x]) = D^\infty(\mathbb{K}).$$

So it is very simple to check the following sufficient condition:

Lemma 8.1. *Let R_1, R_2 be two differential integral domains with a common differential extension S . If there is $n, m \in \mathbb{N}$ such that $R_2 \subset D_S^n(R_1)$ and $R_1 \subset D_S^m(R_2)$, then*

$$D_S^\infty(R_1) = D_S^\infty(R_2).$$

Proof. Using the fact that $R_2 \in D_S^n(R_1)$, we have that $D_S^k(R_2) \subset D_S^{n+k}(R_1)$ for all k . Hence, we conclude that $D_S^\infty(R_2) \subset D_S^\infty(R_1)$.

The other direction follows analogously. □

These conditions can be seen as an interlacement of the differentially definable rings defined over the different integral domains R_1 and R_2 . However, is all of this necessary for having the same limit?

Open Question 8.2 (Same limit with same ambient). Let R_1, R_2 be two differential integral domains with a common differential extension S .

1. Is there a necessary condition (*and computable*) for deciding if the limit domains, $D_S^\infty(R_1)$ and $D_S^\infty(R_2)$, coincide?
2. If $R_1 \subset R_2$, can we conclude something about the inclusions $R_2 \subseteq D_S^n(R_1)$?

Changing the ambient domain

For this problem, we consider R a fixed differential integral domain and S_1, S_2 two differential extensions of R . The question that arise is

Under which conditions over S_1 and S_2 do we have $D_{S_1}^\infty(R) = D_{S_2}^\infty(R)$.

It is obvious that $S_1 \subset S_2$ implies that $D_{S_1}^\infty(R) \subset D_{S_2}^\infty(R)$. Moreover, we always have the equality

$$D_{S_1}^\infty(R) = D_{D_{S_1}^\infty(R)}^\infty(R),$$

so the following lemma is trivial to prove:

Lemma 8.3. *Let R be a differential integral domain and S_1, S_2 two differential extensions. If $D_{S_1}^\infty(R) \subset S_2$ and $D_{S_2}^\infty(R) \subset S_1$, then $D_{S_1}^\infty(R) = D_{S_2}^\infty(R)$.*

This Lemma, however, is not helpful understanding the limit domains from their basic elements (namely, R, S_1 and S_2). We do not know any better conditions for this equality to hold.

Open Question 8.4 (Same limit with same base). Let R be a differential integral domain and S_1, S_2 two differential extensions.

1. Is there a necessary condition (*and computable*) for deciding if the limit domains, $D_{S_1}^\infty(R)$ and $D_{S_2}^\infty(R)$, coincide?
2. If S_2 is the fraction field of S_1 , can we decide the relation between $D_{S_1}^\infty(R)$ and $D_{S_2}^\infty(R)$?
3. If S_2 is an algebraic extension of S_1 , can we decide the relation between $D_{S_1}^\infty(R)$ and $D_{S_2}^\infty(R)$?
4. If S_2 is a monomial extension of S_1 (see definition in Appendix B), can we decide the relation between $D_{S_1}^\infty(R)$ and $D_{S_2}^\infty(R)$?

Reaching the limit

The limit domain $D_S^\infty(R)$ is defined as the union of all $D_S^n(R)$ for $n \in \mathbb{N}$. There are two possibilities: either $D_S^n(R) \subsetneq D_S^{n+1}(R)$ (as we have shown in Corollary 5.7.1 for the case $S = \mathbb{K}[[x]]$ and $R = \mathbb{K}[x]$), or $D_S^\infty(R) = D_S^n(R)$ for some fixed n .

In the first case, we can study the structure of this limit as we have proposed in the open problems 8.2 and 8.4. In the second case, we can try to understand the reasons for reaching the limit ring:

- *The base domain is too big:* we know from Lemma 5.8, that if $\tilde{R} = D_S^\infty(R)$, then $D_S(\tilde{R}) = \tilde{R}$.

- *The ambient domain is too small:* the trivial case here is to consider $S = R$. Then, since $R \subset D_S(R)$ from Lemma 3.2, $D_R(R) = R$. But we can find a non-trivial example: consider $S = \mathbb{K}(x, \tan(x))$ and $R = \mathbb{K}$, then we have that:

$$D_{\mathbb{K}(x, \tan(x))}^\infty(\mathbb{K}) = D_{\mathbb{K}(x, \tan(x))}^3(\mathbb{K}).$$

It is easy to check that

$$\cos(x)^2 = \frac{1}{1 + \tan(x)^2}, \quad 2 \sin(x) \cos(x) = \frac{2 \tan(x)}{(1 + \tan(x)^2)^2},$$

are both in $D_{\mathbb{K}(x, \tan(x))}(\mathbb{K})$. Then we have

$$\tan(x) = \frac{2 \sin(x) \cos(x)}{2 \cos(x)^2},$$

will be in the second iteration $D_{\mathbb{K}(x, \tan(x))}^2(\mathbb{K})$. Finally we complete the whole field $\mathbb{K}(x, \tan(x))$, once we introduce $1/\tan(x)$ in the third iteration.

These two situations show that the possibilities concerning reaching the limit domain are full of possibilities. A deeper study of these possibilities could be an interesting question to study in the future.

Adding extra functions

In our research, we focused on D^n -finite functions, i.e., formal power series that are solutions to linear differential equations with coefficients in $D^{n-1}(\mathbb{K}[x])$. It was shown by Noordman, van der Put and Top (see Corollary 5.40.1) that there are some formal power series that are D -algebraic but are not D^n -finite.

We may consider, for some computations, adding some of these functions to our base domains and start the same construction. Theoretically, this is not a problem and the whole Chapter 3 can be applied. But, in order to use our package described in Chapter 7, we need to construct the Sage structures to handle these bigger rings of functions.

One particular example that may be interesting is the \wp -Weierstrass function [73]. This function is D -algebraic, satisfying the non-linear differential equation

$$\wp'^2 = \wp^3 - g_2\wp - g_3,$$

where g_2 and g_3 are constants. Adding this function to our system can be done straightforward, considering as a base domain the ring

$$R_\wp = \mathbb{K}[x, u, v] / (v^2 - u^3 + g_2u + g_3),$$

where the derivation is defined by

$$\partial(u) = v, \quad \partial(v) = \left(\frac{3}{2}u^2 - \frac{g_2}{2} \right).$$

Computationally, working with this ring is possible and provides no further issue. However, the questions concerning the structure of the chain $D^n(R_\varphi)$ have to be studied again.

We state now one particular question that seems interesting: what is the difference between adding φ to the initial domain and then computing the limit $D^\infty(R_\varphi)$ and computing the limit twice but adding φ in between. Namely, we would like to know the relation between the rings:

$$D^\infty(R_\varphi), \quad D^\infty \left(D^\infty(\mathbb{K}[x])[u, v] \Big/ (v^2 - u^3 + g_2u + g_3) \right).$$

Open Question 8.5. Let R be a differential integral domain and S a differential extension. Let $F \in S \setminus D_S^\infty(R)$. Is there a relation between $D_S^\infty(R)(F)$ and $D_S^\infty(R(F))$?

8.2 Characterize composition

In Chapter 5 we worked with the Riccati differential equation and showed nice conditions to include its solutions as D^n -finite functions. One of the main parts of the process was to do the change of variables $y(x) = v'(x)/c(x)v(x)$. This transformed the non-linear equation for y into a linear equation for v .

In the open problem 5.26 we asked which change of variables $y(x) = \mathcal{F}(v)$ guarantee that, if $v(x) \in D^n(\mathbb{K}[x])$, then $y(x) \in D^m(\mathbb{K}[x])$.

A particular instance of these change of variables is fixing $g(x) \in \mathbb{K}[[x]]$ and consider of change of coordinates is $y = v(g(x))$. In this case, if $g(x) \in D^n(\mathbb{K}[x])$ we can prove using 5.2 that

$$v(x) \in D^m(\mathbb{K}[x]) \implies y(x) \in D^{n+m}(\mathbb{K}[x]).$$

Consider the Mathieu function (see Example 3.5), which satisfies a linear differential equation of the type

$$w''(x) + (a - 2q \cos(2x))w(x) = 0.$$

By definition, $w(x)$ is DD-finite. However, we can do a change of variables $w(x) = f(\sin(x))$:

$$w'(x) = f'(\sin(x)) \cos(x), \quad w''(x) = f''(\sin(x)) \cos(x)^2 - f'(\sin(x)) \sin(x).$$

We can now plug these expressions into the equation of $w(x)$ and change every $\cos(x)^2$ with $1 - \sin(x)^2$, obtaining:

$$(1 - \sin(x)^2)f''(\sin(x)) - \sin(x)f'(\sin(x)) + (a - 2q(1 - 2\sin(x)^2))f(\sin(x)) = 0.$$

We have now an equation in $\sin(x)$. Since its functional inverse is still a formal power series, we can conclude

$$(1 - x^2)f''(x) - xf'(x) + (a - 2q(1 - 2x^2))f(x) = 0,$$

showing that $f(x)$ is D-finite. This means that all Mathieu functions are the composition of two D-finite functions.

The following structural question arises naturally:

Open Question 8.6. Given $f(x) \in D^2(\mathbb{K}[x])$, are there $g(x)$ and $h(x) \in D(\mathbb{K}[x])$ such that $f(x) = g(h(x))$?

We know by Theorem 5.2 that $g(h(x))$ is always DD-finite. Answering this question will emphasize the difference between the ring of D-finite functions and the ring of DD-finite functions.

Moreover, being able to explicitly answer this question (i.e., given $f(x) \in D^2(\mathbb{K}[x])$ being able to compute the possible values for $g(x)$ and $h(x)$) could be a performance booster for many closure properties. For example, if $f_i(x) = g_i(h(x))$ for $i = 1, 2$, then $f_1(x) + f_2(x) = (g_1(x) + g_2(x)) \circ h(x)$. This can be done even further, reducing operations in the DD-functions to operations on D-finite functions and one final composition.

8.3 Combinatorial examples

Throughout this thesis we provided several examples of DD-finite functions that were not D-finite. We saw that the double exponential e^{e^x-1} (see Example 3.3), the tangent $\tan(x)$ (see Example 3.4) were DD-finite.

We also saw that the solutions of the Riccati differential equation are, in the simplest case, always DD-finite (see Proposition 5.21). Also the solutions to the Mathieu differential equation are DD-finite (see Example 3.5).

However, all of these examples are *elementary*, in the sense that they are built directly from D-finite objects:

- The double exponential is the composition (see Theorem 5.2) of the exponential function with itself.
- The tangent is the quotient (see Lemma 3.14) of the sine and cosine functions, both simple D-finite functions.

- The solutions to the Riccati differential equation are closely related with the tangent. In fact, in Proposition 5.21, we proved that they are the quotient of D-finite functions.
- The solutions to the Mathieu differential equation, as shown in Section 8.2, are the composition of a D-finite function with the sine function.

In this section we present the main examples of DD-finite functions that were not built from D-finite objects in a straightforward manner. These examples come from the combinatorial realm, having some combinatorial classes fitting into the DD-finite framework.

As a quick summary (see [32, 61]) of combinatorial classes, we define a univariate combinatorial class as a set of objects \mathcal{S} together with a size map $\nu : \mathcal{S} \rightarrow \mathbb{N}$ such that the amount of elements in \mathcal{S} with a fixed size n is always finite, denoted by s_n .

Given a combinatorial class \mathcal{S} , we consider two types of formal power series:

- The *ordinary generating function* (or o.g.f.): we multiply the variable x^n by s_n and collect all the terms

$$\text{ogf}(\mathcal{S}) = \text{ogf}((s_n)_n) = \sum_{n \geq 0} s_n x^n.$$

- The *exponential generating function* (or e.g.f.): we multiply the variable x^n by $s_n/n!$ and collect all the terms

$$\text{egf}(\mathcal{S}) = \text{egf}((s_n)_n) = \sum_{n \geq 0} \frac{s_n}{n!} x^n.$$

We say that \mathcal{S} is a D-finite (or holonomic) combinatorial class if the ordinary generating function or the exponential generating function is D-finite. Since holonomic functions are closed under Hadamard product (see Theorem 2.17), it follows that $\text{ogf}(\mathcal{S})$ is D-finite if and only if $\text{egf}(\mathcal{S})$ is D-finite.

However, that is not the case anymore for DD-finite functions. We will see that thanks to all the results in this thesis together with result proved by M. Klazar [46].

Let $[n] = \{1, \dots, n\}$ and $\mathcal{P}([n])$ the subsets of $[n]$. We say that $P \subset \mathcal{P}([n])$ is a partition of $[n]$ if:

- The elements of P are pairwise disjoint.
- The union of elements of P yield the whole set $[n]$.

For example, $P = \{\{1, 2\}, \{3\}\}$ is a partition of $N(3)$. On the other hand,

$$\{\{1, 2, 3\}, \{2, 3, 4\}\}, \quad \{\{1, 2\}, \{4\}\},$$

are not partitions of $[4]$. We denote by $P(n)$ the set of partitions of $[n]$.

Let \mathcal{S} be the set of all the partitions of $[n]$ for all n and $\nu : \mathcal{S} \rightarrow \mathbb{N}$ mapping each partition to the corresponding n . This means that

$$\mathcal{S} = \bigcup_{n \geq 0} P(n), \quad \nu(\sigma) = n \Leftrightarrow \sigma \in P(n).$$

We can see that \mathcal{S} is a combinatorial class and the sequence s_n takes the values:

$$1, 1, 2, 5, 15, 52, 203, 877, 4140, 21147, 115975, \dots$$

This sequence of numbers is called *Bell numbers*, and they satisfy the following recursive formula:

$$s_{n+1} = \sum_{k=0}^n \binom{n}{k} s_k.$$

Consider now $B(x) = \text{egf}(\mathcal{S})$. If we compute the derivative of this function, we obtain:

$$B'(x) = \sum_{n \geq 1} \frac{s_n}{(n-1)!} x^{n-1} = \sum_{n \geq 0} \frac{s_{n+1}}{n!} x^n,$$

and using the recursive formula for s_{n+1} , we get

$$\begin{aligned} B'(x) &= \sum_{n \geq 0} \left(\sum_{k=0}^n \binom{n}{k} \frac{s_k}{n!} \right) x^n \\ &= \sum_{n \geq 0} \left(\sum_{k=0}^n \frac{s_k}{k!} \frac{1}{n-k!} \right) x^n \\ &= \left(\sum_{n \geq 0} \frac{s_n}{n!} x^n \right) \left(\sum_{n \geq 0} \frac{x^n}{n!} \right) \\ &= e^x B(x). \end{aligned}$$

This means that $B(x)$ is DD-finite. In fact, that is the linear differential equation satisfied for the function e^{e^x-1} . Since the initial conditions for this function are precisely the first Bell numbers, we can conclude that $B(x) = e^{e^x-1}$.

It is at this point where we use the result by Klazar. In his article [46], he showed that the ordinary generating function of the Bell numbers is not differentially algebraic. This, using Theorem 5.17, proves that $\text{ogf}(\mathcal{S})$ is not D^n -finite for any $n \in \mathbb{N}$. This provides an example for which the Hadamard product is not closed for DD-finite functions.

This property raises a question about the structure of the sequences for DD-finite functions. We know, as was seen in Section 3.3, that the sequences of DD-finite functions satisfy recurrence equations that, like in the case of Bell numbers, goes over the full history, meaning that the element f_n of the sequence depends on all the f_i with $i = 0, \dots, n-1$.

Open Question 8.7. Let $(s_n)_n$ be a sequence that satisfies a recurrence equation of the type:

$$s_n = f_n(s_0, \dots, s_{n-1}),$$

where f_n is a function that combines in some way the values of s_0, \dots, s_{n-1} . Is there any condition in the function f such that $\text{ogf}((s_n)_n)$ or $\text{egf}((s_n)_n)$ is DD-finite?

In the joint work with A. Bostan [15], we studied another combinatorial: the class of labelled trees. Let \mathcal{T} be the set of all labelled trees, and consider the size function that, given a tree, return the number of vertices. Let t_n be the number of labelled trees of size n . It was proven in [21] that

$$t_n = n^{n-1}.$$

If we name $T(x) = \text{egf}(\mathcal{T})$, we can see that $T(x)$ satisfies a functional equation:

$$T(x) = xe^{T(x)}, \quad (8.8)$$

using the combinatorial construction of the labelled trees. This equation leads to a D-algebraic equation computing the derivative of both sides:

$$T'(x) = e^{T(x)}(1 + T'(x)) \quad \Rightarrow \quad xT'(x) = T(x)(1 + T'(x)).$$

Theorem 8.9 ([15, Theorem 1]). *Let $T(x)$ be the generating function of the labelled trees. Then*

$$T(x) \notin D^\infty(\mathbb{K}[x]).$$

Proof. The proof of this Theorem is similar to the proof of Proposition 5.21: we do several changes of variables that are *valid* within the D^n -finite chain and arrive at a function that is not in $D^\infty(\mathbb{K}[x])$.

Currently, our only proved example of a function that is not in $D^\infty(\mathbb{K}[x])$ are the solutions to the differential equation $y'(x) = y(x)^3 - y(x)^2$ (see Corollary 5.40.1). Reaching this equation is our primary goal.

First of all, consider the formal power series $W(x) = -T(-x)$. This formal power series is known as the main branch of the Lambert W function [24]. The first coefficients of its Taylor expansion are:

$$0, 1, -1, \frac{3}{2}, -\frac{8}{3}, \frac{125}{24}, \dots$$

From equation (8.8), we obtain the following functional equation for $W(x)$:

$$-T(-x) = xe^{T(-x)} \Rightarrow W(x)e^{W(x)} = x.$$

Now, we consider the function $Y(x) = W(e^{x+1})$. This change of variables $x \mapsto e^{x+1}$, allows us to have an exponential in the right hand side of the equation. Namely:

$$Y(x)e^{Y(x)-1} = e^x,$$

and this makes the computation of a differential equation for $Y(x)$ very natural:

$$Y'(x)e^{Y(x)-1}(1+Y(x)) = Y(x)e^{Y(x)-1}, \Rightarrow Y'(x)(1+Y(x)) = Y(x).$$

Finally, consider the function $U(x) = \frac{1}{1+Y(x)}$. Using the differential equation for $Y(x)$ we can easily see that:

$$\begin{aligned} U'(x) &= \frac{-Y'(x)}{(1+Y(x))^2} = \frac{-Y(x)}{(1+Y(x))^3} \\ &= \frac{1}{(1+Y(x))^3} - \frac{1}{(1+Y(x))^2} = U(x)^3 - U(x)^2. \end{aligned}$$

Now we can use Corollary 5.40.1 to conclude that $U(x) \notin D^\infty(\mathbb{K}[x])$ and, in consequence, $T(x) \notin D^\infty(\mathbb{K}[x])$. \square

We can see at this point that deciding whether a function is in $D^\infty(\mathbb{K}[x])$ or even if a function is D-algebraic are difficult questions. However, a structural result was conjectured by Pak and Yeliussizov that helps solving this dilemma in many cases:

Conjecture 8.10 (Open Problem 2.4 [60]). Let $(a_n)_n$ a sequence such that both $\text{ogf}((a_n)_n)$ and $\text{egf}((a_n)_n)$ are D-algebraic. Then they are both D-finite.

Since D-finite functions are closed under Hadamard product, it was clear that these should be on the exception list for the conjecture. However, the conjecture goes further guaranteeing that D-finite functions are the only formal power series with such property.

As an intermediate step toward proving the conjecture, we propose here as an open question a weaker version of Pak's conjecture. Namely, we restrict to the class of D^n -finite functions.

Open Question 8.11. Let $(a_n)_n$ be a sequence such that both $\text{ogf}((a_n)_n)$ and $\text{egf}((a_n)_n)$ are D^n -finite for some $m \in \mathbb{N}$. Then they are both D-finite.

In a recent paper by Bodini, Genitrini, Gittenberger and Wagner [11], the authors showed that a whole class of combinatorial classes are closely related to DD-finite functions. In that paper, the authors were interested mainly in getting information about the asymptotics of these classes and, in the process, they showed that the exponential generating functions for these combinatorial classes were DD-finite multiplied by some unknown functions.

They start with the set of *weakly increasing trees* (see Definition 1 in [11]). These are binary trees that may not be completed, i.e., some inner vertices may have arity less than 2, and have labels on their vertices such that:

- If a vertex has the label k then all labels $1, \dots, k-1$ appear on the tree.
- Along each branch of the tree, the labels are strictly increasing.
- The labels may be repeated in different branches.

Let \mathcal{B} be the set of weakly increasing trees and B_n the number of them with size n . The authors then consider the scaled version of the sequence, defining

$$b_n = \frac{\log(2)^n B_n}{(n-1)!}, \quad b_0 = b_1 = 0. \quad (8.12)$$

In fact, they show in the paper that the b_n can be written with the following formula:

$$b_n = \sum_{k=1}^n \frac{\alpha^k}{k!} \frac{(n-k) \cdots (n-2k+1)}{(n-1) \cdots (n-k)},$$

where $\alpha = 1/\log(2)$. They define an auxiliary generating function $a(x) = \sum_n a_n x^n$ in such a way that allows them to write:

$$b_n = a_n + \sum_{k=1}^n \frac{\alpha^{-k}}{k!} \left(1 - \frac{k(k-1)}{n}\right) b_{n-k}.$$

On the right hand side of that identity, they recognized some Cauchy product of sequences, and when they translated this equality to the generating functions, they obtained [11, Lemma 7]:

$$(2 - e^{x/\alpha}) b'(x) + \left(\frac{x}{\alpha^2} - \frac{1}{\alpha}\right) b(x) = a'(x).$$

Let $g(x)$ be the DD-finite function defined by:

$$(2 - e^{x/\alpha}) g'(x) + \left(\frac{x}{\alpha^2} - \frac{1}{\alpha}\right) g(x) = 0, \quad g(0) = 1.$$

Then we can see that there is $C(x)$ with $C(0) = 0$ such that $C'(x)(2 - e^{x/\alpha})g(x) = a'(x)$ that satisfies

$$b(x) = C(x)g(x).$$

This formulation allowed the authors of the paper to study the asymptotic behavior of b_n and, in consequence, the behavior of B_n . This is one motivating example for the interesting open problem of studying the asymptotic behavior of DD-finite functions.

8.4 Walks in the quarter plane

Another great combinatorial example are the counting problems of walks in the quarter plane. We define a *walk in the plane* as a sequence of points $P_0, \dots, P_n \in \mathbb{Z}^2$. We say that P_0 is the *origin*, P_n is the *ending point* and n is the *length* of the walk.

Given a walk, we define its *sequence of steps* as the sequence $S_i = P_i - P_{i-1}$ for $i = 1, \dots, n$. Given $\mathcal{S} \subset \{-1, 0, 1\}^2 \setminus \{(0, 0)\}$, we say that $W = (P_0, \dots, P_n)$ is a *walk with steps in \mathcal{S}* if $S_i \in \mathcal{S}$ for $i = 1, \dots, n$.

Consider the following counting problem: given a set \mathcal{S} of valid steps, how many walks in the plane with steps in \mathcal{S} start at $(0, 0)$ and end at (i, j) in exactly n steps? This is a classical problem in combinatorics [31, 17, 18, 52]. Let $g_{i,j,n}$ be this number and consider the generating function:

$$G(x, y, t) = \sum_{\substack{i, j \in \mathbb{Z} \\ n \in \mathbb{N}}} g_{i,j,n} x^i y^j t^n.$$

The multivariate setting

Note that at this stage, we have a multivariate generating function. These objects are useful in combinatorics to simplify counting problems. However, these objects are slightly outside the scope of this thesis.

D-finite functions were also considered only as univariate objects. But these definitions were extended to the multivariate case. The main difference to the univariate setting is that being *holonomic* (i.e., satisfy some linear differential restrictions) and being D-finite (i.e., the set of derivatives form a finite dimensional vector space) do not coincide [22].

In this setting, we consider formal power series in several variables x_1 up to x_n : $\mathbb{K}[[x_1, \dots, x_n]]$. We can relate its elements with sequences with several indices:

$$(a_{i_1, \dots, i_n})_{i_j \in \mathbb{N}} \longleftrightarrow \sum_{i_1, \dots, i_n \geq 0} a_{i_1, \dots, i_n} x_1^{i_1} \cdots x_n^{i_n}.$$

And instead of taking only one derivation ∂_x , we consider all the standard derivations for each of the variables:

$$\partial_i(x_j) = \delta_{i,j}, \quad \partial_i(\alpha) = 0 \text{ for all } \alpha \in \mathbb{K}.$$

It is very easy to see that $\partial_i \circ \partial_j = \partial_j \circ \partial_i$, i.e., these derivations commute. We can then represent the ring of linear differential operators over $\mathbb{K}[[x_1, \dots, x_n]]$ as a non-commutative ring where the derivations ∂_i act like variables.

In our approach, when studying differentially definable functions, we have always made use of the finiteness of a vector space and then used linear algebra to compute the closure properties and all related operations. This is fitting more with the concept of *D-finite* multivariate functions. However, there are several issues that would need to be tackled in order to create a full implementation of the closure properties.

We can find an implementation of multivariate D-finite functions in the Maple package `Mgfun` [23] and in the Mathematica package `HolonomicFunctions` (developed by C. Koutschan [49]). In the latter, the author showed how to combine the two concepts of *holonomic* and *D-finite* functions to provide an extensive package for manipulating multivariate formal power series.

Extending our work to the multivariate case is a possibility that we have not yet considered. However, the mentioned software packages are a perfect starting point for extending the theory in this thesis to the multivariate framework.

A kernel equation for counting walks

Going back to the counting problem of walks in the plane, we can prove with a very simple combinatorial argument, that the sequence $g_{i,j,n}$ satisfies the following recurrence:

$$g_{i,j,n+1} = \sum_{(a,b) \in \mathcal{S}} g_{i-a,j-b,n},$$

which translates into the functional equation:

$$G(x, y, t) = \left(t \sum_{(a,b) \in \mathcal{S}} x^a y^b \right) G(x, y, t).$$

The function $\sum_{(a,b) \in \mathcal{S}} x^a y^b$ is called *step function of \mathcal{S}* and it is denoted by $S(x, y)$. Clearly, $S(x, y) \in \mathbb{Q}(x, y)$, so $G(x, y, t)$ is always a rational function.

But something really interesting happens when we add restrictions to the walks we are counting. Consider now the counting sequence $q_{i,j,n}$ for the number of walks with steps in \mathcal{S} that start at the origin and end at (i, j) in exactly n steps but always stay in the first quadrant (i.e., $P_i \in \mathbb{N}^2$ for $i = 0, \dots, n$). Let $Q(x, y, t) \in \mathbb{Q}[[x, y, t]]$ be the corresponding multivariate generating function.

The functional equation for $Q(x, y, t)$ changes and involves not only the step function, but also particular evaluations of $Q(x, y, t)$ and, the so called, *kernel polynomial* of \mathcal{S} :

$$K(x, y, t) = xy(1 - tS(x, y)).$$

Since all steps in \mathcal{S} are small (i.e., the coordinates are $-1, 0$ or 1), $K(x, y, t)$ is clearly a polynomial. Again, with a very simple combinatorial argument, we can show that $Q(x, y, t)$ satisfies the following functional equation:

$$\begin{aligned} Q(x, y, t)K(x, y, t) = & xy - Q(0, 0, t)K(0, 0, t) \\ & + Q(x, 0, t)K(x, 0, t) + Q(0, y, t)K(0, y, t). \end{aligned} \quad (8.13)$$

The study of this generating function $Q(x, y, t)$ has been the focus of many studies over the years [10, 14, 17, 18, 27, 30, 31, 52, 53] and, finally, in 2020, the classification of the generating functions for different step sets \mathcal{S} according to their algebraic and differential nature was completed [28, 29].

After removing trivial and symmetric cases, only 79 different models (i.e., options for \mathcal{S}) need to be considered. The classification states that the generating function $Q(x, y, t)$ fits into one of the following categories:

- **Algebraic models:** 4 models have an algebraic generating function. The minimal polynomial has been computed [16, 43, 17].
- **D-finite models:** there are 19 models that have a D-finite generating function. In this multivariate framework, this means that the $Q(x, y, t)$ satisfy a linear differential equation with respect to ∂_x , ∂_y and ∂_t . These equations have been all computed in previous works.
- **D-algebraic models:** there are 9 models that are D-algebraic. More precisely, some non-linear equations with respect to ∂_x and ∂_y were found for these models.
- **D-transcendental models:** the 47 remaining models satisfy no differential equation.

In our framework of DD-finite functions, the models that are interesting for us are the D-algebraic models. If we manage to extend the multivariate framework to include DD-finite functions, we can study if these D-algebraic models are DD-finite.

Note that the non-linear differential equations for the D-algebraic models were not computed in an automatic fashion. This means, for each of the models a different study was made to compute the concrete differential equation. The joint work with A. Bostan, F. Chyzak and P. Lairez [12] was oriented on building algorithms to compute these differential equations.

If these algorithms were completed, then we could prove that other counting generating functions (i.e., adding weights to the steps or allowing bigger/different steps) are also D-algebraic.

Elliptic curves and walks in the quarter plane

The starting point for any method to understand the nature of $Q(x, y, t)$ is the kernel functional equation (8.13). It turns out that the behavior of the kernel polynomial $K(x, y)$ is enough to extract algebraic and differential information of $Q(x, y, t)$.

If we consider t as a parameter, we have that $K(x, y, t)$ is a polynomial with two variables. This defines a curve in the projective space $\mathbb{P}^2(\mathbb{C})$. However, instead of having 1 homogenization variable, we may add two (one for x and one for y):

$$K(x, y, t) \longrightarrow \bar{K}_t(x_0, x_1, y_0, y_1) = \text{num} \left(K\left(\frac{x_0}{x_1}, \frac{y_0}{y_1}, t\right) \right),$$

where **num** takes the numerator of the evaluated kernel polynomial. This new polynomial defines a curve in $\mathbb{P}(\mathbb{C}) \times \mathbb{P}(\mathbb{C})$:

$$E_t = \{(\alpha_0 : \alpha_1, \beta_0 : \beta_1) \in \mathbb{P}(\mathbb{C}) \times \mathbb{P}(\mathbb{C}) : \bar{K}_t(\alpha_0, \alpha_1, \beta_0, \beta_1) = 0\}.$$

Since all the steps are small, \bar{K}_t is a biquadratic polynomial, and using an extended version of Bezout's theorem, we only have two possibilities:

- The curve E_t is smooth. Then it has genus 1, i.e., is elliptic.
- The curve E_t has a singular point. Then it has genus 0, i.e., is a rational curve.

It was recently shown that the singular cases always lead to transcendental generating functions [29]. But in the case where E_t is an elliptic curve, different things can happen depending on the nature of the elliptic curve.

More precisely, it depends on the behavior of an automorphism τ of the curve. In order to define τ we first need to introduce the two involutions ι_x and ι_y .

Consider $(\alpha_0 : \alpha_1) \in \mathbb{P}(\mathbb{C})$ and the evaluated polynomial

$$K_x(y_0, y_1) = \bar{K}_t(\alpha_0, \alpha_1, y_0, y_1).$$

This polynomial has degree 2 and, in consequence, will have at most two different roots. Let $(\beta_0 : \beta_1)$ and $(\tilde{\beta}_0 : \tilde{\beta}_1)$ be these two roots. It is clear (by definition of E_t) that both

$$(\alpha_0 : \alpha_1, \beta_0 : \beta_1), \quad (\alpha_0 : \alpha_1, \tilde{\beta}_0 : \tilde{\beta}_1),$$

are in E_t . We define ι_x as the involution that swap these two points. We can define ι_y symmetrically as the involution that fixes the y coordinate and swaps between the two roots of $K_y(x_0, x_1)$.

Now, define $\tau = \iota_y \circ \iota_x$. It was shown in [52] that

$$\begin{aligned} (\tau - 1)(Q(x, 0, t)K(x, 0, t)) &= \tau(y)(\tau(x) - x) =: b_1 \in \mathbb{C}(E_t) \\ (\tau - 1)(Q(0, y, t)K(0, y, t)) &= x(\tau(y) - y) =: b_2 \in \mathbb{C}(E_t) \end{aligned}$$

The telescoping problem over elliptic curves

The study of τ was critical in a first part of the classification. Namely, if the order of τ is finite (i.e., there is $n \in \mathbb{N}$ such that $\tau^n = id_{E_t}$) then the generating function $G(x, y, t)$ is D-finite.

A further study of the functions b_2 and b_1 is then the key to distinguish between the remaining cases. In fact, it was shown [28] that $G(x, y, t)$ is D-algebraic if b_1 and b_2 telescope in $\mathbb{C}(E_t)$.

This means that there is a linear differential operator $L_i \in \mathbb{C}[\partial]$ and a rational function $g_i \in \mathbb{C}(E_t)$ such that

$$L_i \cdot b_i = \tau(g) - g,$$

which is similar to other creative telescoping problems [51, 75] but in the context of rational functions over elliptic curves.

In [12], we presented a package to perform the operations described above for arbitrary walk models. In that package, we provide the user explicitly with all the important objects (like the kernel function $K(x, y, t)$, the automorphism τ or the rational functions b_1 and b_2).

We also provide methods for computing the telescoping equation. This requires an analysis of the poles over E_t of the functions b_1 and b_2 following the ideas detailed in [28].

8.5 Numerical approximation

Computing numerical approximations for solutions to linear differential equations is part of the analytic study of differential equations. There are plenty of methods for computing approximate solutions to (partial) differential equations such as *finite element methods* [19].

In this section we focus on the problem of computing numerical approximations of solutions to linear differential equations. In particular, let $\varepsilon > 0$, $\alpha \in \mathbb{C}$ and $f(x) \in \mathbb{K}[[x]]$ given by a linear differential equation of the shape

$$r_d(x)f^{(d)}(x) + \dots + r_0(x)f(x) = 0, \quad f(0) = \alpha_0, \dots, f^{(d-1)}(0) = \alpha_{d-1}.$$

Can we compute an β such that $|f(\alpha) - \beta| < \varepsilon$?

In our context of D^n -finite functions, we always work with formal power series, and approximating the value anywhere except at $x = 0$ may be difficult. One approach, followed by M. Mezzarobba in [56] requires to define properly an analytic continuation from $x = 0$ to the point $x = \alpha$, because that influences directly the possible value of the function.

Moreover, this analytic continuation requires to know where the (possible) poles of the function $f(x)$ are located. This is a well known result.

Lemma 8.14. *Let $f(x)$ satisfying a linear differential equation of the shape*

$$r_d(x)f^{(d)}(x) + \dots + r_0(x)f(x) = 0,$$

and let $P(f)$ and $Z(f)$ denote the poles and zeros of f respectively. Then

$$P(f) \subset \bigcup_{i=0}^{d-1} P(r_i) \cup Z(r_d).$$

There are several works that studied these singularities, since they are important to understand the asymptotic behavior of the solutions to those equations [9]. Understanding these poles is so important that similar studies have been done in the difference case [2].

In the case of D -finite functions, since all the coefficients are polynomials, they have no singularities and the number of zeros are a finite set of algebraic elements. Hence, applying Lemma 8.14 is very useful.

This was used by M. Mezzarobba [55, 56] during his thesis to develop algorithms for a rigorous numerical approximation where he provides an arbitrary precision numerical value for D -finite functions and also a rigorous error bound, making it a validated solver. These algorithms were included later in the package `ore_algebra` [42, 57].

Extending these algorithms to DD -finite functions is an interesting problem because it would allow the user of our packages to compute symbolically with these objects and then perform a numerical evaluation knowing precisely the error they are making with that evaluation. However, once we change the coefficients into D -finite functions, applying Lemma 8.14 becomes extremely difficult since computing the zeros of D -finite functions is an open problem.

Open Question 8.15. Let $f(x) \in D^n(\mathbb{K}[x])$. Can we decide a set $Z \subset \mathbb{C}$ such that $Z(f) \subset Z$?

A possible idea would be to apply the method of Mezzarobba to a special type of DD-finite functions. Namely, those where the leading coefficient is a polynomial and, hence, we can locate the zeros properly. This lead, for instance, to the following open problems:

Open Question 8.16. Given $f(x)$ a D^n -finite function, i.e., only knowing a linear differential equation and several initial conditions, can we decide whether or not $f(x) \in \mathbb{K}[x]$?

Open Question 8.17. Consider $S \subset D^2(\mathbb{K}[x])$ the set of all DD-finite function that satisfy a linear differential equation with a polynomial leading coefficient. Has S any algebraic structure?

In case this set is a ring, are the closure properties of addition, multiplication and derivation constructive?

Bibliography

- [1] S. Abramov, M. Barkatou, and D. Khmelnov. On full rank differential systems with power series coefficients. *Journal of Symbolic Computation*, 68:120 – 137, 2015.
- [2] S. Abramov and M. Hoeij. Set of poles of solutions of linear difference equations with polynomial coefficients. *Computational Mathematics and Mathematical Physics*, 43:57, 01 2003.
- [3] S. A. Abramov and D. E. Khmelnov. Regular solutions of linear differential systems with power series coefficients. *Programming and Computer Software*, 40(2):98–106, 2014.
- [4] M. Abramowitz and I. A. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Dover, New York, 9th dover printing, tenth gpo printing edition, 1964.
- [5] G. E. Andrews, R. Askey, and R. Roy. *Special Functions*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1999.
- [6] M. F. Atiyah and I. G. MacDonald. *Introduction to commutative algebra*. Addison-Wesley-Longman, 1969.
- [7] E. H. Bareiss. Sylvester’s identity and multistep integer-preserving gaussian elimination. *Mathematics of Computation*, 22:565–578, 1967.
- [8] M. Barkatou and A. Jiménez-Pastor. Linearizing Differential Equations: Riccati Solutions as D^n -Finite Functions. DK Report 2019-07, Doctoral Program Computational Mathematics, Johannes Kepler University Linz, 4040 Linz, Austria, June 2019.
- [9] M. A. Barkatou and S. S. Maddah. Removing apparent singularities of systems of linear differential equations with rational function coefficients. In *Proceedings of the 2015 ACM on International Symposium on Symbolic and*

- Algebraic Computation*, ISSAC '15, page 53–60, New York, NY, USA, 2015. Association for Computing Machinery.
- [10] O. Bernardi, M. Bousquet-Mélou, and K. Raschel. Counting quadrant walks via Tutte’s invariant method. Preprint, <https://hal.archives-ouvertes.fr/hal-01577762>. 54 pages, 10 figures, 10 tables, Sept. 2017.
 - [11] O. Bodini, A. Genitrini, B. Gittenberger, and S. Wagner. On the number of increasing trees with label repetitions. *Discrete Mathematics*, 343(8):111722, 2020.
 - [12] A. Bostan, F. Chyzak, A. Jiménez-Pastor, and P. Lairez. The sage package `comb_walks` for walks in the quarter plane. *ACM Communications in Computer Algebra*, 54(2):30–38, Sept. 2020.
 - [13] A. Bostan, F. Chyzak, F. Ollivier, B. Salvy, É. Schost, and A. Sedoglavic. Fast computation of power series solutions of systems of differential equations. In *18th ACM-SIAM Symposium on Discrete Algorithms*, pages 1012–1021, 2007. New Orleans, January 2007.
 - [14] 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.
 - [15] A. Bostan and A. Jiménez-Pastor. On the exponential generating function of labelled trees. *Comptes Rendus. Mathématique*, page 5, 2020. Accepted.
 - [16] A. Bostan and M. Kauers. The complete generating function for gessel walks is algebraic, 2009. <https://arxiv.org/abs/0909.1965>.
 - [17] M. Bousquet-Mélou. Walks in the quarter plane: Kreweras’ algebraic model. *The Annals of Applied Probability*, 15(2):1451–1491, 2005.
 - [18] M. Bousquet-Mélou and M. Mishna. Walks with small steps in the quarter plane. In *Algorithmic probability and combinatorics*, volume 520 of *Contemp. Math.*, pages 1–39. Amer. Math. Soc., Providence, RI, 2010.
 - [19] D. Braess. *Finite Elements: Theory, Fast Solvers, and Applications in Solid Mechanics*. Cambridge University Press, 2007.
 - [20] M. Bronstein. *Symbolic Integration I*. Springer Berlin Heidelberg, 1997.
 - [21] A. Cayley. A theorem on trees. *Q. J. Math.*, 23:376–378, 1889.

- [22] F. Chyzak. *Fonctions holonomes en Calcul formel*. PhD thesis, École polytechnique, 1998. TU 0531. 227 pages.
- [23] F. Chyzak and B. Salvy. Non-commutative elimination in ore algebras proves multivariate identities. *Journal of Symbolic Computation*, 26(2):187 – 227, 1998.
- [24] R. M. Corless, G. H. Gonnet, D. E. G. Hare, D. J. Jeffrey, and D. E. Knuth. On the Lambert W function. *Adv. Comput. Math.*, 5(4):329–359, 1996.
- [25] T. Crespo and Z. Hajto. *Algebraic Groups and Differential Galois Theory*. Graduate Studies in Mathematics. American Mathematical Soc., 2011.
- [26] *NIST Digital Library of Mathematical Functions*. <http://dlmf.nist.gov/>, Release 1.0.16 of 2017-09-18. F. W. J. Olver, A. B. Olde Daalhuis, D. W. Lozier, B. I. Schneider, R. F. Boisvert, C. W. Clark, B. R. Miller and B. V. Saunders, eds.
- [27] T. Dreyfus and C. Hardouin. Length derivative of the generating series of walks confined in the quarter plane, 2019. Preprint, <https://arxiv.org/abs/1902.10558>.
- [28] T. Dreyfus, C. Hardouin, J. Roques, and M. F. Singer. On the nature of the generating series of walks in the quarter plane. *Inventiones Mathematicae*, 213(1):139–203, 2018.
- [29] T. Dreyfus, C. Hardouin, J. Roques, and M. F. Singer. Walks in the quarter plane: Genus zero case. *Journal of Combinatorial Theory, Series A*, 174, 2020.
- [30] T. Dreyfus and K. Raschel. Differential transcendence & algebraicity criteria for the series counting weighted quadrant walks. *Publications Mathématiques de Besançon*, (1):41–80, 2019.
- [31] G. Fayolle, R. Iasnogorodski, and V. Malyshev. *Random walks in the quarter plane*, volume 40 of *Probability Theory and Stochastic Modelling*. Springer, 2nd edition, 2017.
- [32] P. Flajolet and R. Sedgewick. *Analytic Combinatorics*. Cambridge University Press, USA, 1 edition, 2009.
- [33] W. A. Harris and Y. Sibuya. The reciprocals of solutions of linear ordinary differential equations. *Advances in Mathematics*, 58(2):119 – 132, 1985.

- [34] R. A. Horn and C. R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, 1991.
- [35] A. Jiménez-Pastor. DD-finite functions implemented in Sage. In D. Slamanig, E. Tsigaridas, and Z. Zafeirakopoulos, editors, *Mathematical Aspects of Computer and Information Sciences*, pages 457–462, Cham, 2019. Springer International Publishing.
- [36] A. Jiménez-Pastor. A Sage implementation for DD-finite functions. *ACM Communnications in Computer Algebra*, 53(2):53–56, Nov. 2019.
- [37] A. Jiménez-Pastor and V. Pillwein. Algorithmic arithmetics with DD-finite functions. In *Proceedings of the 2018 ACM International Symposium on Symbolic and Algebraic Computation*, ISSAC '18, pages 231–237, New York, NY, USA, 2018. Association for Computing Machinery.
- [38] A. Jiménez-Pastor and V. Pillwein. A computable extension for D-finite functions: DD-finite functions. *Journal of Symbolic Computation*, 94:90 – 104, 2019.
- [39] A. Jiménez-Pastor, V. Pillwein, and M. F. Singer. Some structural results on D^n -finite functions. *Advances in Applied Mathematics*, 117:102027, 2020.
- [40] M. Kauers. Guessing Handbook. RISC Report Series 09-07, Research Institute for Symbolic Computation (RISC), Johannes Kepler University Linz, Schloss Hagenberg, 4232 Hagenberg, Austria, 2009.
- [41] M. Kauers. The holonomic toolkit. In *Texts & Monographs in Symbolic Computation*, pages 119–144. Springer Vienna, 2013.
- [42] M. Kauers, M. Jaroschek, and F. Johansson. Ore polynomials in Sage. In *Lecture Notes in Computer Science*, pages 105–125. Springer International Publishing, 2015.
- [43] M. Kauers, C. Koutschan, and D. Zeilberger. Proof of Ira Gessel’s lattice path conjecture. *Proceedings of the National Academy of Sciences*, 106(28):11502–11505, 2009.
- [44] M. Kauers and P. Paule. *The Concrete Tetrahedron: Symbolic Sums, Recurrence Equations, Generating Functions, Asymptotic Estimates*. Springer Publishing Company, Incorporated, 1st edition, 2011.
- [45] M. Kauers and B. Zimmermann. Computing the algebraic relations of c-finite sequences and multisequences. *Journal of Symbolic Computation*, 2007, 2006.

- [46] M. Klazar. Bell numbers, their relatives, and algebraic differential equations. *J. Combin. Theory Ser. A*, 102(1):63–87, 2003.
- [47] E. R. Kolchin. Algebraic groups and algebraic dependence. *American Journal of Mathematics*, 90(4):1151–1164, 1968.
- [48] E. R. Kolchin. *Differential Algebra & Algebraic Groups*, volume 54 of *Pure and Applied Mathematics*. Academic Press, 1973.
- [49] C. Koutschan. *Advanced Applications of the Holonomic Systems Approach*. PhD thesis, RISC-Linz, Johannes Kepler University, September 2009.
- [50] C. Koutschan. Advanced applications of the holonomic systems approach. *ACM Communications in Computer Algebra*, 43(3/4):119–119, 2009.
- [51] C. Koutschan. A fast approach to creative telescoping. *Mathematics in Computer Science*, 4(2-3):259–266, 2010.
- [52] I. Kurkova and K. Raschel. On the functions counting walks with small steps in the quarter plane. *Publications Mathématiques. Institut de Hautes Études Scientifiques*, 116:69–114, 2012.
- [53] I. Kurkova and K. Raschel. New steps in walks with small steps in the quarter plane: series expressions for the generating functions. *Annals of Combinatorics*, 19(3):461–511, 2015.
- [54] C. Mallinger. A Mathematica package for manipulations of univariate holonomic functions and sequences. Master’s thesis, Research Institute for Symbolic Computation, Aug. 1996.
- [55] M. Mezzarobba. NumGfun: a package for numerical and analytic computation with D-finite functions. In S. M. Watt, editor, *ISSAC ’10: Proceedings of the 2010 International Symposium on Symbolic and Algebraic Computation*, page 139–146. ACM, 2010.
- [56] M. Mezzarobba. *Autour de l’évaluation numérique des fonctions D-finies*. Thèse de doctorat, École polytechnique, Nov. 2011.
- [57] M. 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.
- [58] M. P. Noordman, M. van der Put, and J. Top. Autonomous first order differential equations, 2019.

- [59] A. Ostrowski. Sur les relations algébriques entre les intégrales indéfinies. *Acta Math.*, 78:315–318, 1946.
- [60] I. Pak. Complexity problems in enumerative combinatorics. In *Proceedings of the International Congress of Mathematicians—Rio de Janeiro 2018. Vol. IV. Invited lectures*, pages 3153–3180. World Sci. Publ., Hackensack, NJ, 2018.
- [61] M. Petkovsek, H. Wilf, and D. Zeilberger. *A = B*. CRC Press, 1996.
- [62] V. Pillwein. *Computer Algebra Tools for Special Functions in High Order Finite Element Methods*. PhD thesis, Johannes Kepler University Linz, 2008.
- [63] E. Rainville and P. Bedient. *Elementary Differential Equations*. The Macmillan Company, New York. Collier-Macmillan Limited, London, 4 edition, 1969.
- [64] M. Rosenlicht. The nonminimality of the differential closure. *Pacific J. Math.*, 52(2):529–537, 1974.
- [65] B. Salvy and P. Zimmermann. Gfun: A maple package for the manipulation of generating and holonomic functions in one variable. *ACM Trans. Math. Softw.*, 20(2):163–177, June 1994.
- [66] R. Stanley and S. Fomin. *Enumerative Combinatorics: Volume 2*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 1999.
- [67] R. P. Stanley. Differentiably finite power series. *European J. Combin.*, 1(2):175–188, 1980.
- [68] W. Stein et al. *Sage Mathematics Software (Version 9.1)*. The Sage Development Team, 2020. <http://www.sagemath.org>.
- [69] J. van der Hoeven. Computing with D-algebraic power series. *Applicable Algebra in Engineering, Communication and Computing*, 30(1):17–49, 2019.
- [70] M. van der Put and M. F. Singer. *Galois Theory of Linear Differential Equations*, volume 328 of *Grundlehren der mathematischen Wissenschaften*. Springer Berlin Heidelberg, 2003.
- [71] M. van Hoeij. Factorization of differential operators with rational functions coefficients. *Journal of Symbolic Computation*, 24(5):537 – 561, 1997.
- [72] M. van Hoeij. Formal solutions and factorization of differential operators with power series coefficients. *Journal of Symbolic Computation*, 24(1):1 – 30, 1997.
- [73] E. T. Whittaker and G. N. Watson. *A Course of Modern Analysis*. Cambridge Mathematical Library. Cambridge University Press, 4 edition, 1996.

- [74] D. Zeilberger. A holonomic systems approach to special functions identities. *Journal of Computational and Applied Mathematics*, 32(3):321–368, Dec. 1990.
- [75] D. Zeilberger. The method of creative telescoping. *Journal of Symbolic Computation*, 11(3):195 – 204, 1991.

Appendix A

Linear algebra

Throughout this thesis, we have used several results concerning linear algebra. This field of Mathematics is a very well studied area. Most of the basic algorithms are already studied by undergraduate students. Moreover, nowadays it is still an important research area with results concerning performance of matrix multiplication. In this thesis, we focus on two main areas of linear algebra.

In Chapter 4 we described several methods that required computing a nullspace of a matrix over an arbitrary integral domain. The resulting vector must have its coefficients in the same integral domain. That is why we considered a fraction-free algorithm, namely, Bareiss' algorithm [7]. We describe this algorithm with detail here.

But, in order to obtain those matrices, we use results concerning differential operators. This can be seen as part of the theory of differential modules [70]. In this thesis, we provide an introduction to this topic focusing in the particular case of vector spaces.

A.1 Fraction-free system solving

Through the whole thesis, our main object of study was the class of DD-finite functions (see Chapter 3 for all definitions). This set is a subring of the ring of formal power series, $\mathbb{K}[[x]]$. However, as we saw in Theorem 3.6 and its use in Chapter 4, we need to perform linear algebra operations in a field of fractions.

Computationally, given an integral domain R it is very simple to have a field of fractions implemented just by taking the ring $R \times (R \setminus \{0\})$ with the following operations:

- $(a, b) + (c, d) = (ad + bc, bd)$,
- $(a, b)(c, d) = (ac, bd)$,

and also taking the relation $(a, b) \sim (c, d)$ if and only if $ac = bd$. Doing this only requires the operations of addition, multiplication and zero recognition on R . We then represent the element (a, b) with the notation a/b .

If $R = \mathbb{K}[x]$, we know we can compute the greatest common divisor of the numerator and denominator of the fraction a/b and, after removing it, obtain a canonical representation for the fraction where the numerator and denominator are coprime. However, this is not possible with D-finite functions.

On one hand, if we consider the D-finite functions as formal power series, we can only factor an appropriate power of x . On the other hand, if we expect to do further simplification we would need to know the algebraic relations between D-finite objects. Computing these relations is not simple. Even for C-finite functions (i.e., formal power series defined by a linear differential equation with constant coefficients) we can see, as it is shown by Kauers and Zimmermann [45], that computing these relation is a non-trivial task.

Moreover, reducing a differential equation is not a simple problem either. This means, having $f(x)$ defined by a linear differential equation $\mathcal{A} \cdot f(x) = 0$ for $\mathcal{A} \in R[\partial_x]$ for some ring R , find another operator \mathcal{B} that annihilates $f(x)$ with minimal order, is a difficult problem. This would require either a guessing and proving tool [40] or a procedure to factorize linear differential operators [72].

Since this is not feasible to run every single time, the operations on the field of fractions of D-finite function has a critical performance issue: the orders of the operators grow extremely fast while doing operations with fractions. This was one of the main reasons to avoid the use of fractions and, instead, perform linear algebra operations only on the integral domain R of our coefficients.

A.1.1 Diagonalizing matrices

The main linear algebra operation that we need to perform is to compute one element in the nullspace of a fixed matrix. Let $\mathbb{M}_{n \times m}(R)$ be the ring of matrices with n rows and m columns with coefficients in the integral domain R . Let $M \in \mathbb{M}_{n \times m}(R)$ be a matrix. The easiest way to compute the nullspace of M is to perform a Gauss-Jordan elimination on the matrix and check afterwards if the nullspace contains a non-trivial element.

But this Gauss-Jordan elimination is not fraction-free. This is where Bareiss' algorithm [7] is so useful for us. In his paper, Bareiss was focused in integer-preserving elimination on matrices in such a way that the size of the coefficients during the algorithm remained as small as possible. This was intended in order to compute determinants of matrices.

In our case, we focus more on the elimination problem in order to compute the right nullspace of a matrix with coefficients over an integral domain. The method

has two main steps: first the choice of a valid pivot and then the proper elimination step.

Let us consider a matrix $M \in \mathbb{M}_{n \times m}(\widetilde{R})$ with the following shape:

$$M = \begin{pmatrix} D & A \\ 0 & B \end{pmatrix} \quad (\text{A.1})$$

where A and B are arbitrary submatrices with coefficients $a_{i,j}$ and $b_{i,j}$ respectively, and D is a diagonal matrix of size p where each successive element is divisible by the next (i.e., $d_{i+1} \mid d_i$). Assume we know all the quotients $r_i = d_i/d_{i+1} \in R$. From this situation, we build a matrix \widetilde{M} equivalent to M (i.e., with the same right nullspace up to a permutation of the coordinates) with exactly the same shape but with \widetilde{D} having size $(p+1)$. We will also provide the new values for the quotients r_i .

Choosing a pivot

Choosing a pivot should be, in general, a simple task: any element that is not a zero divisor is a valid pivot. In our case, since we are working with integral domains, this is equivalent to have a non-zero element. In order to use this method over an integral domain R , the zero recognition algorithm must be implemented as well. After finding a pivot, we must rearrange the matrix swapping rows and columns in order to put the new pivot in the appropriate position, namely, the element $b_{1,1} \neq 0$.

We look for the pivot in the matrix B . The exact position in B for searching the pivot is theoretically irrelevant. However, since we are concerned in the computation of the right nullspace of M , swapping rows does not change the nullspace. This will avoid operations after the whole Bareiss' algorithm is completed. Hence, we look for a pivot in the first column of B and, if that fails, we move on to other columns.

It could happen that we do not find any valid pivot. Since we are working over an integral domain, this implies that $B = 0$ and, hence, Bareiss' algorithm is completed.

These searching procedure can be found in the algorithm [find_pivot](#).

Elimination step

Once we have a valid pivot in the appropriate position (i.e., $b_{1,1} \neq 0$), we can perform an elimination step that will create zeros above and below the pivot only using operations in the integral domain.

Consider the row $(p+i)$ for $i > 1$. In order to create a zero in the position $(p+i, p+1)$, which is below our pivot, we multiply the row $(p+1)$ by $b_{i,1}$ and

Algorithm 19: find_pivot

Input : matrix M like in (A.1) and the size p of the diagonalized part**Output:** position of the new pivot or *Not found*

```

for  $j = p + 1, \dots, m$  do
    for  $i = p + 1, \dots, n$  do
        if  $m_{i,j} \neq 0$  then
            return  $(i, j)$ ;
return Not found;

```

the row $(p + i)$ by $b_{1,1}$ and substitute the row $(p + i)$ by the result of subtracting these two new rows. This means that in the position $(p + i, p + j)$ we obtain the following element:

$$b_{1,1}b_{i,j} - b_{1,j}b_{i,1} = \begin{vmatrix} b_{1,1} & b_{1,j} \\ b_{i,1} & b_{i,j} \end{vmatrix}.$$

It is clear that the new element below the pivot is now zero and that the other columns in B are simply the corresponding 2 by 2 minors of the matrix B .

This would be enough if we are looking for a triangularization of the matrix M . But after these operations we do not have zeros above the pivot nor does the diagonal in D have the desired divisibility property. Fortunately for us, repeating what we did below the pivot solves both problems. Again, for any row $i \in \{1, \dots, p\}$, we substitute the i th row for the difference between the row i times $b_{1,1}$ and the row $(p + 1)$ times $a_{i,1}$. At the position $(i, p + j)$ we now have

$$a_{i,1}b_{1,j} - b_{1,1}a_{i,j} = \begin{vmatrix} a_{i,1} & a_{i,j} \\ b_{1,1} & b_{1,j} \end{vmatrix}.$$

It is clear that above the pivot we obtain all zeros and, moreover, if we look to the positions (i, i) for $i \in \{1, \dots, p\}$, these minors are simply $d_i b_{1,1}$.

Note that in this elimination step the row $(p + 1)$ does not change. Moreover, by construction, it is easy to see that $r_p = d_{p-1}/d_p = b_{1,1}$. This elimination step is encoded in the algorithm [bareiss_step](#).

Combining all intermediate steps

Until now we have described what to do for one elimination step and we have discussed all the details that we need to keep in order to get the final result. Summarizing, for each elimination step:

Algorithm 20: bareiss_step**Input** : matrix M like in (A.1) and the size p of the diagonalized part**Output**: matrix \widetilde{M} after the elimination step**for** $j = 1, \dots, m$ **do** **for** $i = 1, \dots, p-1, p+1, \dots, n$ **do** $\widetilde{m}_{i,j} \leftarrow \begin{vmatrix} m_{i,j} & m_{i,p+1} \\ m_{p+1,j} & m_{p+1,p+1} \end{vmatrix};$ $\widetilde{m}_{p+1,j} \leftarrow m_{p+1,j}$ **return** $(\widetilde{m}_{i,j})_{i=1,\dots,n}^{j=1,\dots,m};$

- We localize a valid pivot. We keep track of its value since it will be the new quotient on the diagonal matrix. If there is no valid pivot, the algorithm terminates successfully.
- We put the pivot in the appropriate position. We have to keep track of the changes in the columns since we are interested in the right nullspace of the original matrix M .
- We perform the elimination step by computing several 2 by 2 minors of the original matrix.

In order to apply the algorithm we need to have full control of the operations over R . Namely, addition, multiplication and zero recognition. As we have remarked, the output of the algorithm will be a matrix $\widetilde{M} \in \mathbb{M}_{n \times m}(R)$ equivalent to M with the shape

$$\widetilde{M} = \begin{pmatrix} D & A \\ 0 & 0 \end{pmatrix},$$

where D is a diagonal matrix of size p such that each element of the diagonal is divisible by the next (i.e., $d_{i+1} \mid d_i$). We also return the successive quotients between the elements in this diagonal part $r_i = d_{i+1}/d_i$ and a permutation $\sigma \in S_m$ representing the changes of columns done during the algorithm.

The complete Bareiss' algorithm is encoded in the algorithm [bareiss](#).

A.1.2 Computing a non-trivial element

At this point, we can assume that the matrix $M \in \mathbb{M}_{n \times m}(R)$ with the following shape:

$$\begin{pmatrix} D & A \\ 0 & 0 \end{pmatrix}, \tag{A.2}$$

Algorithm 21: bareiss

Input : matrix M
Output: matrix \widetilde{M} with the appropriate shape, the sequence (r_1, \dots, r_p)
and a permutation σ for the change of columns
 $\widetilde{M} \leftarrow M$; // we make a copy of M
 $\sigma \leftarrow id$;
for $i = 1, \dots, n$ **do**
 pivot \leftarrow **find_pivot**(M, i);
 if pivot **is** *Not found* **then**
 | break;
 else
 | $(P_r, P_c) \leftarrow$ pivot;
 $r_i \leftarrow m_{P_r, P_c}$; // storing the new value for r_i
 if $P_c \neq i$ **then**
 | $\sigma \leftarrow \sigma \cdot (i \ P_c)$; // update the change of columns
 | $\widetilde{M} \leftarrow$ **swap_column**(\widetilde{M}, i, P_c)
 if $P_r \neq i$ **then**
 | $\widetilde{M} \leftarrow$ **swap_row**(\widetilde{M}, i, P_r)
 $\widetilde{M} \leftarrow$ **bareiss_step**(\widetilde{M}, i);
return $\widetilde{M}, (r_1, \dots, r_i), \sigma$;

where D is a diagonal matrix of size $p < m$ and each element in the diagonal is divisible by the next, $A \in \mathbb{M}_{p \times (m-p)}(R)$ and the zero rows may not be present. From this point on, we omit these zero rows since they provide no extra information about the right nullspace. We have that the right nullspace of M has dimension $m - p$.

Let the diagonal of D be the elements d_1, d_2, \dots, d_p and consider the quotients $s_i = d_1/d_i \in R$. Then, if we multiply the i th row of M by s_i we obtain:

$$\begin{pmatrix} d_1 & 0 & \dots & 0 & s_1 a_{1,1} & s_1 a_{1,2} & \dots & s_1 a_{1,m-p} \\ 0 & d_1 & \dots & 0 & s_2 a_{2,1} & s_2 a_{2,2} & \dots & s_2 a_{2,m-p} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & d_1 & s_p a_{p,1} & s_p a_{p,2} & \dots & s_p a_{p,m-p} \end{pmatrix}$$

For any $j = 1, \dots, m - p$, there is one vector in the nullspace with $v_{p+i} = 0$ for $i \neq j$ and $v_{p+j} = d_1$. In fact:

$$\mathbf{v}_j = (-s_1 a_{1,j}, -s_2 a_{2,j}, \dots, -s_p a_{p,j}, 0, \dots, d_1, \dots, 0) \quad \text{for } j = 1, \dots, m - p$$

are those vectors. They are clearly linearly independent, all the coefficients are in R and they are all in the nullspace of M . These computations of the right nullspace can be found as pseudocode in the algorithm `right_nullspace`.

Algorithm 22: `right_nullspace`

Input : matrix M like described in (A.2)
Output: basis of the right nullspace of M with coefficients in R
for $i = 1, \dots, p$ **do**
 $s_i \leftarrow m_{p,p} // m_{i,i}$; // computing the quotients of the diagonal
for $j = 1, \dots, m - p$ **do**
 $\mathbf{v}_j = (s_1 m_{1,p+j}, \dots, s_p m_{p,p+j}, m_{1,1} \delta_{1,j}, \dots, m_{1,1} \delta_{m-p,j})$;
return $\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_{m-p}\}$;

Note that, in our case after the elimination using `bareiss`, the computation of the quotients s_i can be done without any division since that information was gathered in the output (r_1, \dots, r_p) . Namely:

$$s_i = \prod_{j=1}^{i-1} r_j,$$

so when we use the algorithm `right_nullspace` after `bareiss` we can avoid the divisions.

A.2 Differential linear algebra

In this Section we present a basic theory about vector spaces when we add a differential operator in a similar way that adding a derivation to rings and fields (see Appendix B for further details). This topic has been studied in the general case of differential modules (see [48] and [70]).

Definition A.3. Let (F, ∂) be a differential field (see Appendix B for a precise definition) of characteristic zero and V an F -vector space. We say that $\vec{\partial} : V \rightarrow V$ is a *derivation over V w.r.t. ∂* if it satisfies:

- $\vec{\partial}(\mathbf{v} + \mathbf{w}) = \vec{\partial}(\mathbf{v}) + \vec{\partial}(\mathbf{w})$ for all $\mathbf{v}, \mathbf{w} \in V$.
- $\vec{\partial}(c\mathbf{v}) = \partial(c)\mathbf{v} + c\vec{\partial}(\mathbf{v})$ for all $c \in F$ and $\mathbf{v} \in V$.

We say that $(V, \vec{\partial})$ is a *differential vector space over (F, ∂)* . We denote by $\Delta_{\partial}(V)$ the set of all derivations over V with respect to ∂ .

Fixed a derivation $\vec{\partial}$ over V , we say that a vector \mathbf{v} is a constant if $\vec{\partial}(\mathbf{v}) = 0$. We will denote the set of all constants by $C_{\vec{\partial}}(V)$.

Lemma A.4. *Let (F, ∂) be a differential field and V an F -vector space. Let $B = \{\mathbf{v}_\alpha\}_\alpha$ be a basis of V and $I = \{\mathbf{w}_\alpha\}_\alpha$ another fixed set of vectors. Then there is a unique derivation $\vec{\partial}$ over V w.r.t. ∂ such that*

$$\vec{\partial}(\mathbf{v}_\alpha) = \mathbf{w}_\alpha \text{ for all } \alpha.$$

Proof. The uniqueness is clear since each element \mathbf{x} of V can uniquely be represented by the basis

$$\mathbf{x} = \sum_{\alpha} c_{\alpha} \mathbf{v}_{\alpha},$$

and if we had two derivations $\vec{\partial}_1$ and $\vec{\partial}_2$ with the desired property, we can check that:

$$\begin{aligned} \vec{\partial}_i(\mathbf{x}) &= \vec{\partial}_i \left(\sum_{\alpha} c_{\alpha} \mathbf{v}_{\alpha} \right) = \sum_{\alpha} \vec{\partial}_i(c_{\alpha} \mathbf{v}_{\alpha}) \\ &= \sum_{\alpha} \partial(c_{\alpha}) \mathbf{v}_{\alpha} + c_{\alpha} \vec{\partial}_i(\mathbf{v}_{\alpha}) \\ &= \sum_{\alpha} \partial(c_{\alpha}) \mathbf{v}_{\alpha} + c_{\alpha} \mathbf{w}_{\alpha}, \end{aligned}$$

so $\vec{\partial}_1(\mathbf{x}) = \vec{\partial}_2(\mathbf{x})$ for all \mathbf{x} , showing that $\vec{\partial}_1 \equiv \vec{\partial}_2$.

Moreover, we can check that the map $\vec{\partial}$ defined by

$$\vec{\partial}(\mathbf{x}) = \sum_{\alpha} \partial(c_{\alpha}) \mathbf{v}_{\alpha} + c_{\alpha} \mathbf{w}_{\alpha}$$

is well defined (since the representation of \mathbf{x} in the basis is unique), it is a derivation over V w.r.t. ∂ and, for any α :

$$\vec{\partial}(\mathbf{v}_{\alpha}) = \partial(1) \mathbf{v}_{\alpha} + \mathbf{w}_{\alpha}.$$

□

This Lemma showed that we only need to know the derivatives of the basis elements in order to know the derivation over the whole vector space V . In addition, it highlights the underlying structure of the derivations of a vector space.

Definition A.5. Let (F, ∂) be a differential field, V an F -vector space and the set $B = \{\mathbf{v}_{\alpha}\}_{\alpha}$ a basis of V . We define the *coefficient-wise derivation map w.r.t. B* (and denote it by κ_B) to the map that takes a vector $\mathbf{x} = \sum_{\alpha} c_{\alpha} \mathbf{v}_{\alpha}$ and returns the same vector but changing the coefficients with their derivatives:

$$\kappa_B(\mathbf{x}) = \sum_{\alpha} \partial(c_{\alpha}) \mathbf{v}_{\alpha}.$$

We may write only κ when the basis is clear from the context.

Lemma A.6. *Let (F, ∂) be a differential field, V an F -vector space and $B = \{\mathbf{v}_\alpha\}_\alpha$ a basis of V . Let $\vec{\partial}$ be a derivation over V w.r.t. ∂ . Then there is a unique F -linear map f such that*

$$\vec{\partial}(\mathbf{x}) = \kappa(\mathbf{x}) + f(\mathbf{x}),$$

for all $\mathbf{x} \in V$.

Proof. As we saw in the proof of Lemma A.4, if we denote $\mathbf{w}_\alpha = \vec{\partial}(\mathbf{v}_\alpha)$, then we have that

$$\begin{aligned} \vec{\partial}(\mathbf{x}) &= \sum_{\alpha} \partial(c_\alpha) \mathbf{v}_\alpha + c_\alpha \mathbf{w}_\alpha \\ &= \kappa(\mathbf{x}) + \sum_{\alpha} c_\alpha \mathbf{w}_\alpha. \end{aligned}$$

Let $f : V \rightarrow V$ be the unique F -linear map with $f(\mathbf{v}_\alpha) = \mathbf{w}_\alpha$. Then,

$$\vec{\partial} \equiv \kappa + f.$$

□

Lemma A.6 shows that there is a bijection between the set of derivations over V w.r.t. ∂ and the set of F -linear maps over V . In fact, we have one bijection for each basis of V .

In the case of a finite dimensional vector space, linear maps are related with matrices. Once we have fixed a basis of the vector space V , there is then a matrix that allows us to perform derivations over the vector space V only using the derivation over the ground field, ∂ , and a matrix-vector multiplication. This concept can be extended to *generator sets* instead of only basis:

Definition A.7. Let (F, ∂) be a differential field, $(V, \vec{\partial})$ a differential vector space over (F, ∂) of finite dimension and $\Phi = (\phi_1, \dots, \phi_n)$ a tuple of generators of V (i.e., $\langle \phi_1, \dots, \phi_n \rangle_F = V$). We say that a matrix $M = (m_{i,j})_{i,j=1}^n$ is a *derivation matrix* of $\vec{\partial}$ w.r.t. Φ if it satisfies

$$\vec{\partial}(\phi_j) = m_{1,j}\phi_1 + \dots + m_{n,j}\phi_n, \quad \text{for all } j = 1, \dots, n.$$

From this definition, and mimicking the proof of Lemma A.4, for any $\mathbf{x} = \sum_{i=1}^n c_i \phi_i \in V$ we have

$$\vec{\partial}(\mathbf{x}) = \partial(c_1)\phi_1 + c_1\vec{\partial}(\phi_1) + \dots + \partial(c_n)\phi_n + c_n\vec{\partial}(\phi_n) = \sum_{j=1}^n \left(\sum_{i=1}^n m_{i,j}c_i + \partial(c_j) \right) \phi_j.$$

If we write this equality in a matrix-vector notation, a representation of $\vec{\partial}(\mathbf{x}) = (\hat{c}_1\phi_1 + \dots + \hat{c}_n\phi_n)$ can thus be computed as

$$\begin{pmatrix} \hat{c}_1 \\ \hat{c}_2 \\ \vdots \\ \hat{c}_n \end{pmatrix} = M \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix} + \begin{pmatrix} \partial(c_1) \\ \partial(c_2) \\ \vdots \\ \partial(c_n) \end{pmatrix}. \quad (\text{A.8})$$

Conversely, if a matrix M satisfies the equality (A.8) (namely, computes the derivatives of elements in V by a matrix-vector multiplication), then M is a derivation matrix of $\vec{\partial}$ w.r.t. Φ .

For any tuple of generators Φ a derivation matrix can always be computed, being the extreme case when Φ is a basis and the matrix is unique. This is computationally interesting, since we can use information about the generators of a vector space V to compute the derivatives of all the elements of V . As we see now, this is particularly interesting because we can understand and compute derivatives in vector spaces which are built from smaller spaces.

Proposition A.9 (Direct sum). *Let $(V, \vec{\partial}_V)$ and $(W, \vec{\partial}_W)$ be two differential vector spaces over (F, ∂) of dimensions n and m and with basis B_V and B_W . Let M_V and M_W be the derivation matrices of V and W respectively. Then:*

(i) *The map $\vec{\partial} : V \oplus W \rightarrow V \oplus W$ defined as*

$$\vec{\partial}(\mathbf{v} \oplus \mathbf{w}) = \partial_V(\mathbf{v}) + \partial_W(\mathbf{w}),$$

is the unique derivation over $V \oplus W$ w.r.t. ∂ that extends $\vec{\partial}_V$ and $\vec{\partial}_W$.

(ii) *The derivation matrix associated with the basis $B_V \oplus B_W$ is $M_V \oplus M_W$.*

Proof. Recall first that the vector space $V \oplus W$ is build as the set $V \times W$ with the operations

- Termwise addition: $(\mathbf{v}_1, \mathbf{w}_1) + (\mathbf{v}_2, \mathbf{w}_2) = (\mathbf{v}_1 + \mathbf{v}_2, \mathbf{w}_1 + \mathbf{w}_2)$.
- Scalar product: $c(\mathbf{v}, \mathbf{w}) = (c\mathbf{v}, c\mathbf{w})$.

We write $(\mathbf{v}, \mathbf{w}) = \mathbf{v} \oplus \mathbf{w}$. Obviously, $\vec{\partial}$ is a derivation over $V \oplus W$ w.r.t. ∂ and it clearly extends both $\vec{\partial}_V$ and $\vec{\partial}_W$. Now, assume there is a derivation $\hat{\partial}$ over $V \oplus W$ with the same property. Then we have:

$$\begin{aligned} \hat{\partial}(\mathbf{v} \oplus \mathbf{w}) &= \hat{\partial}((\mathbf{v} \oplus 0) + (0 \oplus \mathbf{w})) = \hat{\partial}(\mathbf{v} \oplus 0) + \hat{\partial}(0 \oplus \mathbf{w}) \\ &= (\vec{\partial}_V(\mathbf{v}) \oplus 0) + (0 \oplus \vec{\partial}_W(\mathbf{w})) = \vec{\partial}_V(\mathbf{v}) \oplus \vec{\partial}_W(\mathbf{w}) = \vec{\partial}(\mathbf{v} \oplus \mathbf{w}), \end{aligned}$$

proving that $\hat{\partial} \equiv \vec{\partial}$. It is known that the set

$$B_V \oplus B_W = \{\mathbf{v} \oplus 0 : \mathbf{v} \in B_V\} \cup \{0 \oplus \mathbf{w} : \mathbf{w} \in B_W\},$$

is a basis of $V \oplus W$ and a simple calculation shows that the matrix $M_V \oplus M_W$ is the derivation matrix for that matrix. Recall that the direct sum of two matrices is defined as:

$$M_V \oplus M_W = \begin{pmatrix} M_V & 0 \\ 0 & M_W \end{pmatrix}.$$

□

Proposition A.10 (Tensor product). *Let $(V, \vec{\partial}_V)$ and $(W, \vec{\partial}_W)$ be two differential vector spaces over (F, ∂) of dimensions n and m and with basis B_V and B_W . Let M_V and M_W be the derivation matrices of V and W respectively. Then:*

(i) *The map $\vec{\partial} : V \otimes W \rightarrow V \otimes W$ defined as*

$$\vec{\partial}(\mathbf{v} \otimes \mathbf{w}) = \partial_V(\mathbf{v}) \otimes \mathbf{w} + \mathbf{v} \otimes \partial_W(\mathbf{w}),$$

is a derivation over $V \otimes W$ w.r.t. ∂ .

(ii) *The derivation matrix associated with the basis $B_V \otimes B_W$ is $M_V \boxplus M_W$, where \boxplus denotes the Kronecker sum of two matrices (see [34]) and can be written using the tensor product of matrices:*

$$M_V \boxplus M_W = M_V \otimes \mathcal{I}_m + \mathcal{I}_n \otimes M_W.$$

Proof. Recall first how the tensor product of two vector spaces is generated: $V \otimes W$ is the free additive group generated by $V \times W$ with the following relations:

- $(\mathbf{v}_1 + \mathbf{v}_2, \mathbf{w}) \sim (\mathbf{v}_1, \mathbf{w}) + (\mathbf{v}_2, \mathbf{w})$.
- $(\mathbf{v}, \mathbf{w}_1 + \mathbf{w}_2) \sim (\mathbf{v}, \mathbf{w}_1) + (\mathbf{v}, \mathbf{w}_2)$.
- $(c\mathbf{v}, \mathbf{w}) \sim (\mathbf{v}, c\mathbf{w})$.

And we represent the generators elements (\mathbf{v}, \mathbf{w}) by $\mathbf{v} \otimes \mathbf{w}$. It is interesting to remark that, given the basis B_V and B_W , the set

$$B_V \otimes B_W = \{\mathbf{v} \otimes \mathbf{w} : \mathbf{v} \in B_V, \mathbf{w} \in B_W\}$$

is a basis of $V \otimes W$.

In order to check now that the map $\vec{\partial}$ defined above is a derivation over $V \otimes W$ w.r.t. ∂ , we first need to check that the definition we gave is well-defined with

respect to the three relations that define the tensor product, which is a straightforward computation. Then the additive property and the Leibniz rule are granted by the definition of $\vec{\partial}$.

To see that the derivation matrix associated with the basis $B_V \otimes B_W$ is the Kronecker sum of the derivation matrices for V and W , consider $\mathbf{v} \otimes \mathbf{w} \in B_V \otimes B_W$ and the linear mappings $f_V : V \rightarrow V$ and $f_W : W \rightarrow W$ defined in Lemma A.6. Recall that, since $\mathbf{v} \in B_V$ and $\mathbf{w} \in B_W$, we have that $\vec{\partial}_V(\mathbf{v}) = f_V(\mathbf{v})$ and $\vec{\partial}_W(\mathbf{w}) = f_W(\mathbf{w})$. Then:

$$\begin{aligned} \vec{\partial}(\mathbf{v} \otimes \mathbf{w}) &= \partial_V(\mathbf{v}) \otimes \mathbf{w} + \mathbf{v} \otimes \partial_W(\mathbf{w}) = f_V(\mathbf{v}) \otimes \mathbf{w} + \mathbf{v} \otimes f_W(\mathbf{w}) \\ &= (f_V \otimes id_W)(\mathbf{v} \otimes \mathbf{w}) + (id_V \otimes f_W)(\mathbf{v} \otimes \mathbf{w}) \\ &= (f_V \otimes id_W + id_V \otimes f_W)(\mathbf{v} \otimes \mathbf{w}) \end{aligned}$$

Hence, the derivation matrix associated to $B_V \otimes B_W$ is precisely the matrix associated with the linear map $(f_V \otimes id_W + id_V \otimes f_W)$, namely:

$$M_V \otimes \mathcal{I}_m + \mathcal{I}_n \otimes M_W = M_V \boxplus M_W.$$

□

This Kronecker sum can be seen exactly as a Leibniz rule for the derivation matrices: the derivation matrix for the product of two vector spaces V and W is the derivation matrix of V times the identity plus the identity times the derivation matrix of W .

Proposition A.11 (Quotient space). *Let $(V, \vec{\partial})$ be a differential vector space over (F, ∂) and consider a subspace $N \subset V$ closed under $\vec{\partial}$. Then:*

- (i) *There is a unique derivation $\bar{\partial}$ over V/N that commutes with the canonical projection $\pi : V \rightarrow V/N$, i.e., $\bar{\partial} \circ \pi = \pi \circ \vec{\partial}$.*
- (ii) *If $\Phi = (\phi_1, \dots, \phi_n)$ is a tuple of generators of V and M is a derivation matrix of $\vec{\partial}$ w.r.t. Φ , then $\pi(M)$ is a derivation matrix of $\bar{\partial}$ w.r.t. $\pi(\Phi)$.*

Proof. Recall that V/N was built with the equivalence classes of the relation defined by

$$\mathbf{v} \sim \mathbf{w} \Leftrightarrow \mathbf{v} - \mathbf{w} \in N.$$

We represent the elements of V/N by $\mathbf{v} + N$ and we have that

$$\mathbf{v} \sim \mathbf{w} \Leftrightarrow \mathbf{v} + N = \mathbf{w} + N.$$

Then, the canonical projection $\pi : V \rightarrow V/N$ is defined as the linear map $\pi(\mathbf{v}) = \mathbf{v} + N$.

Let now $\bar{\partial} : V/N \rightarrow V/N$ be defined by $\bar{\partial}(\mathbf{v} + N) = \vec{\partial}(\mathbf{v}) + N$. This map is well defined because if $\mathbf{v} \sim \mathbf{w}$, then we have that

$$\vec{\partial}(\mathbf{v}) - \vec{\partial}(\mathbf{w}) = \vec{\partial}(\mathbf{v} - \mathbf{w}) \in N,$$

because $\vec{\partial}$ is closed in N . Hence $\vec{\partial}(\mathbf{v}) \sim \vec{\partial}(\mathbf{w})$.

A straightforward computation using the definition of $\bar{\partial}$ and the linearity of π shows that $\bar{\partial}$ is a derivation and that $\bar{\partial} \circ \pi = \pi \circ \vec{\partial}$. Let $\tilde{\partial}$ be another derivation over V/N with the same property, i.e., $\tilde{\partial} \circ \pi = \pi \circ \vec{\partial}$. Then for any $\mathbf{v} + N \in V/N$:

$$\tilde{\partial}(\mathbf{v} + N) = \vec{\partial}(\mathbf{v}) + N = \bar{\partial}(\mathbf{v} + N),$$

showing that $\tilde{\partial} \equiv \bar{\partial}$.

On the other hand, it is clear that if Φ is a tuple of generators for V , the tuple $\pi(\Phi) = (\pi(\phi_1), \dots, \pi(\phi_n))$ generates V/N . Let $\mathbf{v} = c_1\phi_1 + \dots + c_n\phi_n$. By Definition A.7, we know that $\vec{\partial}\mathbf{v}$ can be represented by the vector obtained with:

$$\begin{pmatrix} \hat{c}_1 \\ \vdots \\ \hat{c}_n \end{pmatrix} = M \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix} + \begin{pmatrix} \partial(c_1) \\ \vdots \\ \partial(c_n) \end{pmatrix}.$$

Now, we also know (by the linearity of π and the definition of $\bar{\partial}$) that:

$$\bar{\partial}(\mathbf{v} + N) = \pi(\hat{c}_1)\pi(\phi_1) + \dots + \pi(\hat{c}_n)\pi(\phi_n).$$

Hence, $\pi(M)$ is a derivation matrix of $\bar{\partial}$ w.r.t. $\pi(\Phi)$. □

Lemma A.12 (Differential linear mappings). *Let $(V, \vec{\partial}_V)$ and $(W, \vec{\partial}_W)$ be two differential vector spaces over (F, ∂) and let $f : V \rightarrow W$ be a linear map that commutes with the derivations, i.e., $f \circ \vec{\partial}_V = \vec{\partial}_W \circ f$. Then $\ker(f)$ is closed under $\vec{\partial}_V$.*

Proof. Let $\mathbf{v} \in \ker(f)$. Then we have that

$$f(\vec{\partial}_V(\mathbf{v})) = \vec{\partial}_W(f(\mathbf{v})) = \vec{\partial}_W(0) = 0,$$

showing that $\vec{\partial}_V(\mathbf{v}) \in \ker(f)$. □

All these tools are used to compute derivations on vector spaces starting from the knowledge of derivations in smaller vector spaces. Moreover, applying Proposition A.11 and Lemma A.12, allows us to work not only with basis (as we did in Propositions A.9 and A.10), but also with lists of generators of the vector spaces. This is interesting computationally because we can avoid the computation of all linear relations between the generators and simply compute the desired derivations.

Appendix B

Differential Galois theory

Differential Galois theory is a branch of differential algebra. This theory studies the differential properties of several objects in a very algebraic fashion. We do not consider here the exact meaning of the derivation (as it happens with analysis). The elements we consider are inside an algebraic structure and the derivation is a function with basic properties (namely, it satisfies the Leibniz rule) that make it behave like the usual derivation studied in analysis.

These definitions were used in several parts of the thesis (see Chapter 3 and Appendix A), and it provides a generic framework to study all the properties of differentially definable functions. This leads in the end to a very generic implementation of their closure properties (see Chapters 4 and 6).

In order to obtain a deeper understanding of the structure of these functions (see Chapter 5), we need to use some results from differential Galois theory, a similar theory to the standard Galois theory for fields, that study the symmetries of solutions to linear differential equations.

In this appendix, we introduce some basic definitions and results about differential algebra following classical literature [20, 48]. Then we proceed to introduce all the known results about differential Galois theory that are required through the thesis. All the results displayed here can be found in the literature [25, 70] but we include them here for completion.

B.1 An introduction to differential algebra

B.1.1 Differential rings and fields

In differential algebra, we consider an algebraic structure (like a ring or field) and add a operator that behaves like the derivation that is studied in analysis.

Definition B.1. Let R be a ring. We say that $\partial : R \rightarrow R$ is a *derivation* if it satisfies, for all $r, s \in R$:

- $\partial(r + s) = \partial(r) + \partial(s)$.
- (*Leibniz rule*) $\partial(rs) = \partial(r)s + r\partial(s)$.

We say that the pair (R, ∂) is a *differential ring*. If R is a field, we call the pair *differential field*.

There is always a derivation for all rings. Namely, the map $0_R : R \rightarrow R$, with $0_R(r) = 0$ for all $r \in R$, is a derivation. If we consider all the possible derivations we can see that they have a richer structure.

Lemma B.2. Let R be a ring and Ω_R the set of all derivations of R . Then Ω_R is an R left-module.

Proof. Let $\partial_1, \partial_2 \in \Omega_R$ and $r \in R$. Showing that $(\partial_1 + \partial_2)$ is a derivation is just a simple computation and apply the two main properties of a derivation.

For $\partial = r\partial_1$ is a very similar computation: take two elements $s, t \in R$. We can see that:

$$\partial(s + t) = r\partial_1(s + t) = r\partial_1(s) + r\partial_1(t) = \partial(s) + \partial(t).$$

$$\begin{aligned} \partial(st) &= r\partial_1(st) = r(\partial_1(s)t + s\partial_1(t)) \\ &= r\partial_1(s)t + sr\partial_1(t) = \partial(s)t + s\partial(t) \end{aligned}$$

□

Note that the natural right multiplication is not closed on the derivations. Let $r \in R$ and $\partial \in \Omega_R$. Then we can easily check that $\partial r = \partial(r) + r\partial$:

$$(\partial r)(s) = \partial(rs) = \partial(r)s + r\partial(s) = (\partial(r) + r\partial)(s).$$

Using Lemma B.2, we have that (∂r) is a derivation if and only if the multiplication map by $\partial(r)$ is a derivation. We can see easily that is only the case when $\partial(r) = 0$:

$$\partial(r) = \partial(r)(1) = \partial(r)(1 \cdot 1) = \partial(r) \cdot 1 + 1 \cdot \partial(r) = 2\partial(r).$$

Hence, in general, the map (∂r) is not a derivation.

A similar reasoning leads naturally to the conclusion that the derivation of 1 is always zero:

$$\partial(1) = \partial(1 \cdot 1) = \partial(1) + \partial(1) = 2\partial(1) \quad \Rightarrow \quad \partial(1) = 0.$$

The set of elements whose derivation is zero is a very important set in differential algebra and is called the set of constants.

Definition B.3. Let (R, ∂) be a differential ring. We say that an element $r \in R$ is a *constant* if $\partial(r) = 0$. We denote by $C_\partial(R)$ the set of all constants of (R, ∂) . We may omit R and ∂ where they are clear from the context.

The set of constants have indeed a nice structure inside the whole ring.

Lemma B.4. *Let (R, ∂) be a differential ring. Then $C_\partial(R)$ is a subring of R . Moreover, if R is a field, then $C_\partial(R)$ is a subfield of R .*

Proof. That $C_\partial(R)$ is closed under addition and product is a direct application of the two main properties of a derivation. Let $r \in R^*$ be a constant unit of R and let s be its multiplicative inverse. Then:

$$0 = \partial(1) = \partial(rs) = \partial(r)s + r\partial(s) = r\partial(s).$$

Since r is a unit, we know it is not a zero divisor and then $\partial(s) = 0$, showing that $s \in C_\partial(R)$. \square

At this point we can prove several classical properties for derivations that will help in future computations.

Lemma B.5. *Let (R, ∂) be a differential ring. Then:*

- (Derivation of powers) *For all $r \in R$ and $n \in \mathbb{N}$, $\partial(r^n) = n\partial(r)r^{n-1}$.*
- (Derivation of quotient) *Let $r \in R^*$ be a unit and s its multiplicative inverse. Then $\partial(s) = -\partial(r)s^2$.*
- (Generalized Leibniz Rule) *For all $r_1, \dots, r_n \in R$*

$$\partial(r_1 \cdots r_n) = \sum_{i=1}^n \left(\prod_{j \neq i} r_j \right) \partial(r_i).$$

- (Logarithmic derivatives) *For all $r_1, \dots, r_n \in R^*$ and $e_1, \dots, e_n \in \mathbb{Z}$*

$$\frac{\partial(r_1^{e_1} \cdots r_n^{e_n})}{r_1^{e_1} \cdots r_n^{e_n}} = \sum_{i=1}^n e_i \frac{\partial(r_i)}{r_i}.$$

Proof. We can prove the first result by induction on n . The case $n = 1$ is trivial, since it states $\partial(r) = \partial(r)$. Now, let $n > 1$:

$$\begin{aligned} \partial(r^n) &= \partial(r^{n-1}r) = \partial(r^{n-1})r + r^{n-1}\partial(r) \\ &= (n-1)\partial(r)r^{n-2}r + r^{n-1}\partial(r) = n\partial(r)r^{n-1}. \end{aligned}$$

For proving the second result, we use the identity $rs = 1$ and compute the derivation:

$$0 = \partial(1) = \partial(rs) = \partial(r)s + r\partial(s).$$

It is clear, since $rs = 1$ that we get $\partial(s) = \partial(r)s^2$. As a natural corollary, we have that for any $r \in R$ and $s \in R^*$:

$$\partial\left(\frac{r}{s}\right) = \frac{\partial(r)s - \partial(s)r}{s^2}.$$

For the third property, we use induction and the Leibniz rule from Definition B.1. Finally, the fourth property is a direct application of the first, second and third result. \square

The following examples will be used during this Appendix and will come handy to write other properties easier:

- $(F, 0_F)$ is a differential field with constants F .
- $(F[x], \partial_x)$ where ∂_x is the standard derivation w.r.t. x is a differential ring.
- If (F, ∂) is a differential field, then $\kappa_\partial : F[x] \rightarrow F[x]$ defined by

$$\kappa_\partial(a_0 + \dots a_n x^n) = \partial(a_0) + \dots + \partial(a_n)x^n,$$

is a derivation over $F[x]$.

B.1.2 Differential extensions

Now that we have studied the basic structures and definitions, we will study how we can extend a derivation from a given ring and the relation between these derivations.

Definition B.6. Let $(R, \partial_1), (S, \partial_2)$ be two differential rings. We say that (S, ∂_2) is a *differential extension* of (R, ∂_1) if $R \subset S$ and $\partial_2|_R \equiv \partial_1$.

Although studying the extensions for rings is interesting, from this point on we are going to focus in the case of differential fields and their extensions. We will see that, when we extend a field (either adding an algebraic or transcendental element) we can control the type of differential extension we may get.

Proposition B.7. Let (R, ∂) be a differential ring and $M \subset R$ a multiplicatively closed set of R . Let $S = R_M$ be the localization of R with M . Then there is a unique derivation $\tilde{\partial}$ on S that extends ∂ .

Proof. Recall that M is multiplicatively closed if $0 \notin M$, $1 \in M$ and, for all $m, n \in M$, we have $mn \in M$ too. Then S is built as the elements in $R \times S$ where we set that $(r, m) = (s, n)$ if and only if there is a unit $p \in M$ such that $p(rn - sm) = 0$. We denote the equivalence class of (r, m) by the fraction r/m .

It is well known that the units of S are the units of R together with the fractions $m/1$ and $1/m$ for $m \in M$. Then, if we define

$$\tilde{\partial}(r/m) = \frac{\partial(r)m - \partial(m)r}{m^2},$$

we can proof that:

- If $r/m = s/n$ then this definition yield the same derivative: from the identity $p(rn - sm) = 0$, we get that $p^2 \partial(rn - sm) = 0$. Manipulating that expression we get exactly the condition for proving

$$\frac{\partial(r)m - \partial(m)r}{m^2} = \frac{\partial(s)n - \partial(n)s}{n^2}.$$

- $\tilde{\partial}$ extends ∂ : for all $r \in R$, we have:

$$\tilde{\partial}(r/1) = \frac{\partial(r)1 - \partial(1)r}{1^2} = \partial(r)/1.$$

Then we need to prove that this derivation is unique. This is easy using Lemma B.5. For any derivation D on S and any element r/m :

$$D(r/m) = D(r)/m + rD(1/m) = \partial(r)/m - r\partial(m)/m^2 = \tilde{\partial}(r/m).$$

□

Corollary B.7.1. *Let (R, ∂) be a differential integral domain and F its field of fractions. Then there is a unique derivation $\tilde{\partial}$ over F that extends ∂ .*

Proof. The field of fractions is the localization of R with $R^* = R \setminus \{0\}$. Then we apply Proposition B.7. □

Proposition B.8. *Let (F, ∂) be a differential field and $E = F(\alpha)$ for some algebraic element α . Then there is a unique derivation $\tilde{\partial}$ over E that extends ∂ .*

Proof. Let $m(y) \in F[y]$ be the minimal polynomial of α . For any derivation D over E that extends ∂ we have that:

$$0 = D(m(\alpha)) = \kappa_{\partial}(m)(\alpha) + D(\alpha)\partial_y(m)(\alpha),$$

hence we can conclude that

$$D(\alpha) = -\frac{\kappa_{\partial}(m)(\alpha)}{\partial_y(m)(\alpha)}.$$

Since the right hand side of that identity does not depend on D , we conclude that the derivative of α is always the same for any derivation D on E that extends ∂ .

Moreover, for any $p(y) \in F[y]$ we can also show that

$$D(p(\alpha)) = \kappa_{\partial}(p)(\alpha) + D(\alpha)\partial_y(p)(\alpha).$$

This, together with the fact that all elements in E are polynomials in α with coefficients in F (since $E = F(\alpha)$), shows that D is completely defined by the value of $D(\alpha)$. Since this value is unique, then there can be only one derivation that extends ∂ .

We now define $\tilde{\partial}$ by the following formula: let $e = p(\alpha) \in E$,

$$\tilde{\partial}(e) = \kappa_{\partial}(p)(\alpha) - \frac{\kappa_{\partial}(m)(\alpha)}{\partial_y(m)(\alpha)}\partial_y(p)(\alpha).$$

We can proof that $\tilde{\partial}$ is well defined (i.e., it does not depend on the polynomials representation of its elements), that is a derivation over E and it extends ∂ . \square

Proposition B.9. *Let (F, ∂) a differential field and t_1, \dots, t_p transcendental elements over F algebraically independent between them. Then for every vector $(\alpha_1, \dots, \alpha_p) \in F(t_1, \dots, t_p)^p$ there is a unique derivation $\tilde{\partial}$ over $F(t_1, \dots, t_p)$ such that*

$$\tilde{\partial}(t_i) = \alpha_i.$$

Proof. We first show the uniqueness property. Let D be a derivation over the field $F(t_1, \dots, t_n)$ such that it extends ∂ and $D(t_j) = \alpha_j$. Let $m = t_1^{n_1} \cdots t_p^{n_p}$. Using Lemma B.5 we know that:

$$D(m) = \sum_{i=1}^p \left(\prod_{j \neq i} t_j^{e_j} \right) n_i \alpha_i t_i^{n_i-1}.$$

We can observe that the right hand side does not depend on D , so all derivations coincide on the monomials.

Then, all derivations coincide on $F[t_1, \dots, t_n]$ and, by Lemma B.5, also on $F(t_1, \dots, t_n)$. This proves the uniqueness of the derivation.

For proving the existence, we define the derivation $\tilde{\partial}$ in the expected way and then we can show that it is well defined, that it is indeed a derivation and, finally, that it extends ∂ . \square

These three Propositions remark the fact that if a field is built from a smaller differential field, the derivations that extends the original derivation easy to describe. For example, we know that for all field extensions $F \subset E$, there is an intermediate field \tilde{F} such that $F \subset \tilde{F}$ is algebraic and $\tilde{F} \subset E$ is purely transcendental. Hence, for extending a derivation over F to E , we only need to know the transcendental elements of E over \tilde{F} and set the values for those elements.

Before moving to the next subsection, we present here a result about the constants on these extensions. Having an explicit control over the constants is important since many algorithms compute results up to constants.

Lemma B.10. *Let (R, ∂) a differential ring and M a multiplicatively closed set. Let (S, ∂) be the differential extension where S is the localization of R by M . Then $C(R)_{C(M)} \subset C(S)$.*

Proof. Let r/m where $r, m \in C(R)$. Then it is obvious that

$$\partial(r/m) = \frac{\partial(r)m - \partial(m)r}{m^2} = \frac{0}{m^2} = 0.$$

Hence $r/m \in C(S)$. □

Lemma B.11. *Let (F, ∂) be a differential field and (E, ∂) a differential extension. Let $\alpha \in E$ be algebraic over F . Then $\alpha \in C(E)$ if and only if α is algebraic over $C(F)$.*

Proof. Let $\alpha \in E$ be algebraic over $C(F)$ and $m(y) \in C(F)[y]$ its minimal polynomial. Then, using the proof of Proposition B.8, we know that

$$0 = \kappa_{\partial}(m)(\alpha) + \partial(\alpha)\partial_y(m)(\alpha).$$

Since $m(y) \in C(F)[y]$, we get that $\kappa_{\partial}(m) = 0$, so we conclude that

$$0 = \partial(\alpha)\partial_y(m)(\alpha).$$

Using that $m(y)$ is the minimal polynomial over $C(F)[y]$ and $\partial_y(m)$ has smaller degree than $m(y)$, we obtain that $\partial_y(m)(\alpha) \neq 0$, thus $\alpha \in C(E)$.

To prove the reciprocal, assume now that α is algebraic over F with minimal polynomial $m(y) \in F[y]$ and that $\alpha \in C(E)$. Again, using the argument from the proof of Proposition B.8, we know that

$$0 = \kappa_y(m)(\alpha) + \partial(\alpha)\partial_y(m)(\alpha).$$

In this case, $\partial(\alpha) = 0$, so we conclude that

$$\kappa_y(m)(\alpha) = 0.$$

If $m(y) \notin C(F)[y]$, then $\deg_y(\kappa_{\partial}(m)) \geq 0$. In particular, since $m(y)$ is minimal, the degree of $\kappa_{\partial}(m)$ has to be exactly $\deg_y(m)$. But this is impossible since $m(y)$ is monic (because it is the minimal polynomial). Hence, $m(y) \in C(F)[y]$, showing that α is algebraic over $C(F)$. □

B.1.3 Monomial extensions

From Proposition B.8 and Lemma B.11, it seems that algebraic extensions are easy to handle in the differential framework. On the other side, transcendental extensions are very difficult to treat mostly because the constants that we obtain are not easy to characterize.

In this subsection we study a particular type of transcendental differential extensions, called monomial extensions.

Definition B.12. Let (F, ∂) be a differential field. Let (E, ∂) be a differential extension and $t \in E$. We say that t is a *monomial over (F, ∂)* if it is transcendental over F and $\partial(t) \in F[t]$.

Usually, when we consider monomial extensions over a differential field, we do not specify the ambient field E . We usually only consider the field $F(t)$. The transcendency requirement is important by itself to guarantee that we are adding something new to the field.

Being a monomial can also be seen as a condition on the derivation over $F[t]$.

Lemma B.13. *Let (F, ∂) be a differential field and t a transcendental element. Then t is a monomial if and only if $\partial(F[t]) \subset F[t]$.*

Proof. Let $p(t) \in F[t]$. We have shown several times that:

$$\partial(p(t)) = \kappa_{\partial}(p)(t) + \partial(t)\partial_t(p)(t).$$

This means that if $\partial(t) \in F[t]$ (i.e., if t is a monomial) then $\partial(p(t)) \in F[t]$. On the other hand, if $\partial(F[t]) \subset F[t]$, in particular we have that $t \in F[t]$, so $\partial(t) \in F[t]$ and t is a monomial. \square

We can see that these monomials include some of our favorite examples of D-finite and DD-finite functions:

- e^x is a monomial over \mathbb{K} , with $\partial(t) = t$.
- e^{e^x} is a monomial over $\mathbb{K}(e^x)$, with $\partial(t) = e^x t$.
- $\log(x)$ is a monomial over $\mathbb{K}(x)$, with $\partial(t) = 1/x$.
- $\tan(x)$ is a monomial over \mathbb{K} , with $\partial(t) = 1 + t^2$.

In monomial extensions, most of the problems over $F(t)$ can be transformed into problems over $F[t]$. Then, it is interesting to study the polynomials on a monomial extension. There are two type of polynomials that stand out.

Definition B.14. Let t be a monomial over a differential field (F, ∂) and let $p(t) \in F[t]$.

- We say that $p(t)$ is *normal* if $\gcd(p(t), \partial(p(t))) = 1$.
- We say that $p(t)$ is *special* if $p \mid \partial(p(t))$.

Example B.15. Consider $\mathbb{C}(x)$ and $t = e^x$. We know that t is a monomial over $\mathbb{C}(x)$. Here the polynomial $t^2 + t$ is not special nor normal:

$$\partial(t^2 + t) = 2t^2 + t \Rightarrow \gcd(t^2 + t, \partial(t^2 + t)) = t.$$

On the other, hand, since $\partial(t^k) = kt^k$ for all $k \in \mathbb{Z}$, these polynomials are always special.

Example B.16. Let (F, ∂) be a differential field and t a monomial over F . The unique elements in $F[t]$ that are both normal and special are the elements in F , since the derivation is closed in F .

Special and normal polynomials has nice structural properties:

Lemma B.17. Let t be a monomial over a differential field (F, ∂) and $p(t), q(t) \in F[t]$ two non-zero polynomials. Then:

- $p(t)q(t)$ is normal if and only if $p(t)$ and $q(t)$ are normal and $\gcd(p, q) = 1$.
- $p(t)q(t)$ is special if and only if $p(t)$ and $q(t)$ are special.

Proof. First, we prove the direct implication of the first property. Without lost of generality, assume that $p(t)$ is not normal. Then there is $r(t) \in F[t]$ that divides both $p(t)$ and $\partial(p(t))$. On one hand, it is clear that $r(t)$ divides $p(t)q(t)$. On the other hand, since $\partial(p(t)q(t))$ is the sum of two terms, one involving $p(t)$ and other involving $\partial(p(t))$, then $r(t)$ also divides $\partial(p(t)q(t))$, showing that $p(t)q(t)$ is not normal. On the other hand, if $\gcd(p(t), q(t)) = s(t) \neq 1$, then $s(t)$ divides both $p(t)q(t)$ and $\partial(p(t)q(t))$.

For proving the reciprocal, assume $p(t)q(t)$ is not normal but $p(t)$ and $q(t)$ are both normal. Let $r(t) \in F[t]$ be an irreducible polynomial such $r(t)$ divides both $p(t)q(t)$ and $\partial(p(t)q(t))$. If $r(t)$ divides both $p(t)$ and $q(t)$, then we are done. On the other case, assume without lost of generality that $r(t)$ divides $p(t)$ and do not divide $q(t)$. Since $r(t)$ divides $\partial(p(t)q(t))$, we have that $r(t)$ divides $\partial(p(t))$, which is a contradiction with $p(t)$ being normal.

For proving the second property, assume that both $p(t)$ and $q(t)$ are special. Then $p(t)$ divides $\partial(p(t))$ and $q(t)$ divides $\partial(q(t))$ and

$$p(t)q(t) \mid \partial(p(t))q(t), \quad p(t)q(t) \mid p(t)\partial(q(t)),$$

so $p(t)q(t)$ is special.

Now, assume that $p(t)^n$ is special for an irreducible polynomial $p(t)$. Then we have $p(t)^n$ divides $np(t)^{n-1}\partial(p(t))$, so $p(t)$ is special.

Secondly, assume that $\gcd(p, q) = 1$ and $p(t)q(t)$ divides $\partial(p(t)q(t))$. Then $p(t)$ divides also $\partial(p(t)q(t))$, which implies that $p(t)$ divides $\partial(p(t))q(t)$. Since $p(t)$ and $q(t)$ are coprime, then $p(t)$ divides $\partial(p(t))$. A symmetric argument shows that also $q(t)$ divides $\partial(q(t))$.

Now, consider $p(t)q(t)$ being special. All irreducible factors of $p(t)$ and $q(t)$, using the last two paragraphs, are special. Then $p(t)$ and $q(t)$ are special using the other direction of the implication. \square

B.1.4 Primitive and hyperexponential extensions

Study the general case of monomial extensions is difficult. For example, the problem of finding the set of special or normal polynomials is not solved in the general case. We focus on special cases of monomial extensions. Let t be a monomial over a differential field (F, ∂) .

- We say that t is *primitive* if $\partial(t) \in F$.
- We say that t is *hyperexponential* if there is $f \in F$ such that $\partial(t) = ft$.

These are the easiest cases and are the basic elements for building the so called Liouvillian extensions[20, Chapter 5.1]. In these two cases we can know exactly the set of special polynomials and the set of constants in the monomial extension:

Proposition B.18. *Let t be primitive over a differential field (F, ∂) . If $\partial(t) = \alpha \in F$ is not a derivative of an element of F , then t is a monomial (i.e., transcendental), $C(F(t)) = C(F)$ and there are no special polynomials.*

Conversely, if t is transcendental and $C(F) = C(F(t))$, then $\partial(t)$ is not a derivative of an element of F .

Proof. Assume first that t is algebraic but still a primitive element. Then there is $m(y) = a_0 + \dots + a_{n-1}y^{n-1} + y^n \in F[y]$ such that $m(t) = 0$ and has minimal degree. Hence,

$$\alpha m_y(t) + \kappa_{\partial}(m)(t) = 0,$$

and since $m(y)$ was monic, this equation implies that, as polynomials,

$$\alpha m_y(y) = -\kappa_{\partial}(m)(y).$$

Looking to the coefficient of t^{n-1} we obtain $n\alpha = -\partial(a_{n-1})$, and, since $n \in C(F)$, we have $\alpha = \partial(-a_{n-1}/n)$, showing that there is an element in F with the same derivative as t .

On the other hand, assume that t is transcendental and $C(F(t)) \neq C(F)$. Let $f(t) \in C(F(t))$ be written on the form $f(t) = p(t)/q(t)$ where $p(t)$ and $q(t)$ are two coprime and polynomials. Since $\partial(f(t)) = 0$, we have that

$$\partial(p(t))q(t) - p(t)\partial(q(t)) = 0.$$

Using the fact that $p(t)$ and $q(t)$ are coprime, $p(t)$ divides $\partial(p(t))$, i.e., $p(t)$ is a special polynomial. In particular, $p(t)/\text{lc}(p(t))$ is again a special polynomial that is monic. Since $\partial(t) \in F$, we have that all monic special polynomials are constants.

Let $u(t) = p(t)/\text{lc}(p(t)) = a_0 + \dots + a_{d-1}t^{d-1} + t^d$. We have that:

$$0 = \partial(u(t)) = \alpha u_t(t) + \kappa_{\partial}(u)(t),$$

so, if we take the coefficient of t^{d-1} , we obtain:

$$d\alpha = -\partial(a_{d-1}),$$

so in particular $\alpha = \partial(-a_{d-1}/d)$ showing that there is an element in F with the same derivative as t .

Finally, assume that t is transcendental and $C(F(t)) = C(F)$. If there is $\beta \in F$ such that $\partial(\beta) = \alpha = \partial(t)$, then we have that $t - \beta$ is a constant. But $t - \beta \notin C(F)$, leading to a contradiction. Hence, if t is transcendental and $C(F(t)) = C(F)$, then there is no element in F with the same derivative as t . \square

Proposition B.19. *Let t be hyperexponential over a differential field (F, ∂) . If $\partial(t)/t = \alpha$ is not a logarithmic derivative of a F -radical then t is a monomial (i.e., transcendental), $C(F(t)) = C(F)$ and all special polynomials are powers of t .*

Conversely, if t is transcendental and $C(F(t)) = C(F)$ then $\partial(t)/t$ is not the logarithmic derivative of a F -radical.

Proof. This proof follows the same ideas from the proof of Proposition B.18, but adapted to the hyperexponential case. First, assume that t is algebraic over F and let $m(y) \in F[y]$ be the monic minimal polynomial of t with degree n . Since $m(t) = 0$, we have that

$$0 = \partial(m(t)) = \alpha t m_y(t) + \kappa_{\partial}(m)(t).$$

Since $\deg_y(\alpha y m_y(y)) = \deg_y(m(y))$, we have that, as polynomials,

$$\alpha y m_y(y) + \kappa_{\partial}(m)(y) = \alpha n m(y).$$

If we look to the constant coefficient of both sides and compare, we obtain:

$$\alpha n a_0 = \partial(a_0),$$

which means that α is the logarithmic derivative of a F -radical.

Now, assume that t is transcendental and that $C(F(t)) \neq C(F)$. We can see again that if $f(t) = p(t)/q(t)$ is a constant with $p(t), q(t)$ two coprime polynomials and $p(t)$ monic, then $p(t)$ is special. We see now that $p(t)/t^{\deg(p(t))}$ is a constant. First, let $n = \deg_t(p(t))$ and note that, since t is hyperexponential, we have that $\deg_t(\partial(p(t))) = \deg_t(p(t))$. Since $p(t)$ is special and monic, we obtain the following identity

$$\partial(p(t)) = n\alpha p(t).$$

If we now compute the derivative of $p(t)/t^n$ we obtain:

$$\partial\left(\frac{p(t)}{t^n}\right) = \frac{\partial(t) - n\alpha p(t)}{t^n} = 0,$$

showing that it is a constant and, in particular, taking the constant term of the numerator:

$$\partial(a_0) = n\alpha a_0,$$

where a_0 is the constant coefficient of $p(t)$. Hence α is the logarithmic derivative of a F -radical.

Finally, assume that t is transcendental and $C(F(t)) = C(t)$. If α is the logarithmic derivative of a F -radical, we have an element $\beta \in F$ and an integer $n \in \mathbb{N}$ such that

$$\frac{\partial(\beta)}{\beta} = n\alpha = n\frac{\partial(t)}{t}.$$

Consider now the element t^n/β . If we compute its derivative, we obtain

$$\partial\left(\frac{t^n}{\beta}\right) = \frac{t^{n-1}(n\partial(t)\beta - \partial(\beta)t)}{\beta^2} = 0,$$

so t^n/β is a constant. But this is a contradiction with the assumption on the field of constants. Hence, α is not the logarithmic derivative of a F -radical. \square

These results allows us to give transcendency criteria for elements when we know the field of constants:

Corollary B.19.1. *Let (F, ∂) be a differential field and (E, ∂) a differential extension with $C(E) = C(F)$. Let $\alpha \in E$.*

- *If α is primitive over F , then $\alpha \in F$ or α is transcendental.*
- *If α is hyperexponential over F , then either there is $n \in \mathbb{N}$ such that $\alpha^n \in F$ or α is transcendental.*

Proof. Let us start with the case of α primitive and let $h = \partial(\alpha) \in F$. There are two cases: if there is $f \in F$ with $\partial(f) = h$, then $\partial(\alpha) = \partial(f)$. This implies that $\partial(\alpha - f) = 0$ and then $\alpha \in F$. On the other hand, assume f does not exist. Then, by Proposition B.18, α is transcendental.

Now, in the case of α being hyperexponential, let $h = \partial(\alpha)/\alpha$. There are again two cases: if h is not a logarithmic derivative of a F -radical, then we get by Proposition B.19 that α is transcendental. On the other hand, assume there is $f \in F$ and $n \in \mathbb{N}$ such that

$$n \frac{\partial(\alpha)}{\alpha} = nh = \frac{\partial(f)}{f}.$$

Using Lemma B.5, we have that the previous identity implies that

$$\partial(\alpha^n f) = 0,$$

so $\alpha^n f \in F$ and, in particular, $\alpha^n \in F$. □

B.2 Linear differential equations

In this Section we study the main algebraic properties of the solutions to linear differential equations over differential fields. Given a differential ring (R, ∂) we denote by $R[\partial]$ the ring of linear differential operators over R (see Definition 2.29 for a precise definition).

It is common, given a differential operator, to consider the solutions to the corresponding differential equation $L \cdot y = 0$.

Definition B.20 (Solution space). Let (R, ∂) be a differential ring, (S, ∂) a differential extension and $L \in R[\partial]$ a linear differential operator. We define *the solution space of L in S* to be the set

$$V_S(L) = \{r \in S : L \cdot r = 0\}.$$

This set of solutions has a very precise structure with respect to the constants:

Lemma B.21. *Let (R, ∂) be a differential ring, (S, ∂) a differential extension, $C(S)$ its ring of constants and $L \in R[\partial]$ a linear differential operator. Then $V_S(L)$ is a left $C(S)$ -module.*

Proof. Using the fact that L is a linear differential operator it is easy to see that if $r, s \in V_S(L)$, then its sum is also a solution.

Now, if $c \in C(S)$, then we have that $cL = Lc$, meaning that if $r \in V_S(L)$ then $L \cdot (cr) = c(L \cdot r) = 0$. □

Moreover, when we are working with fields, we obtain vector spaces. The following result gives an upper bound to the dimension of the solution space:

Lemma B.22. *Let (F, ∂) be a differential field, C its field of constants and $L \in F[\partial]$ a monic linear differential operator of order n . Let $f_1, \dots, f_r \in F$ be such $L \cdot f_i = 0$ and the vectors*

$$\mathbf{v}_i = (f_i, \partial(f_i), \dots, \partial^{n-1}(f_i)),$$

are C -linearly independent. Then the vectors are F -linearly independent.

Proof. We proceed by induction on r . The case $r = 1$ is trivial since a vector is always linearly independent. Let $r > 1$. By induction hypothesis, we have that all proper subset of $\{\mathbf{v}_1, \dots, \mathbf{v}_r\}$ are F -linearly independent. Assume that these vectors are F -linearly dependent. Then there is a unique set of $\alpha_1, \dots, \alpha_{r-1} \in F$ such that for all $j = 0, \dots, n-1$

$$\alpha_1 \partial^j(f_1) + \dots + \alpha_{r-1} \partial^j(f_{r-1}) = \partial^j(f_r). \quad (\text{B.23})$$

Using the fact that $L \cdot f_i = 0$ for all $i = 1, \dots, n$, we can show that

$$\alpha_1 \partial^n(f_1) + \dots + \alpha_{r-1} \partial^n(f_{r-1}) = \partial^n(f_r). \quad (\text{B.24})$$

If we apply ∂ to equation B.23, we obtain another expression for $\partial^{j+1}(f_r)$. We can combine that with equation B.24 to obtain for all $j = 0, \dots, n-1$:

$$\partial(\alpha_1) \partial^j(f_1) + \dots + \partial(\alpha_{r-1}) \partial^j(f_{r-1}) = 0.$$

Since the vectors $\{\mathbf{v}_1, \dots, \mathbf{v}_{r-1}\}$ are F -linearly independent, we get $\partial(\alpha_i) = 0$. Hence $\alpha_i \in C$ for all $i = 1, \dots, r-1$ showing that the set $\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_r\}$ is C -linearly dependent. \square

Corollary B.24.1. *Let (F, ∂) be a differential field, (E, ∂) a differential extension, $C(E)$ its field of constants and $L \in F[\partial]$ a linear differential operator of order n . Then $V_E(L)$ has dimension at most n as a $C(E)$ -vector space.*

Proof. First, if $L \in F[\partial]$ and $F \subset E$, then $L \in E[\partial]$. Now, let f_1, \dots, f_m be a basis of $V_E(L)$ and consider the vectors $\mathbf{v}_i = (f_i, \partial(f_i), \dots, \partial^{n-1}(f_i))$ as we did in Lemma B.22. If $m > n$, then the vectors are E -linearly dependent and, by Lemma B.22, they are $C(E)$ -linearly dependent. In particular, this shows that f_1, \dots, f_m are $C(E)$ -linearly dependent and they could not be a basis of $V_E(L)$. Hence $m \leq n$. \square

This can be extended even further and have a general criteria for linear independence of elements in a differential field only looking to a determinant:

Definition B.25 (Wronskian). Let (R, ∂) be a differential ring and $r_1, \dots, r_n \in R$. We define the *Wronskian matrix* of r_1, \dots, r_n as the matrix

$$W(r_1, \dots, r_n) = \begin{pmatrix} r_1 & r_2 & \dots & r_n \\ \partial(r_1) & \partial(r_2) & \dots & \partial(r_n) \\ \vdots & \vdots & \ddots & \vdots \\ \partial^{n-1}(r_1) & \partial^{n-1}(r_2) & \dots & \partial^{n-1}(r_n) \end{pmatrix}.$$

We also call *the Wronskian* of r_1, \dots, r_n , and denote it by $w(r_1, \dots, r_n)$, to be the determinant of their Wronskian matrix.

Lemma B.26. *Let (F, ∂) be a differential field and C its field of constants. Then f_1, \dots, f_n are C -linearly dependent if and only if their Wronskian is zero.*

Proof. For any set $f_1, \dots, f_n \in F$ we can always build a linear differential operator $L \in F[\partial]$ of order at most n such that $L \cdot f_i = 0$ for all $i = 1, \dots, n$. We can build it by induction on n , building a sequence of operators L_j such that $L_j \cdot f_i = 0$ for all $1 \leq i \leq j$.

We start with $L_1 = \partial - \frac{\partial(f_1)}{f_1}$. It clearly annihilates f_1 . Now assume we have built L_j . Define

$$L_{j+1} = (\partial L_j) - \frac{(\partial L_j) \cdot f_{j+1}}{L_j \cdot f_{j+1}} L_j.$$

It is clear that $L_{j+1} \cdot f_i = 0$ for all $i = 1, \dots, j$. By construction, it also annihilates f_{j+1} and it has order at most $j + 1$.

Now that we have all f_i solutions to the same linear differential equation, we can apply Lemma B.22. In fact, the vectors mentioned in that Lemma are precisely the columns of the Wronskian matrix. Meaning that they are F -linearly dependent if and only if the Wronskian vanishes. This proves the result. \square

B.3 Picard-Vessiot rings

In field theory, algebraic extensions were the main type of extension that was considered. But in differential algebra we also have a derivation that is key to the underlying structure of our fields.

So far we have studied how the classical extensions (adding algebraic or transcendental elements) behave with the derivations, but now we are interested in how we can extend a differential field with solutions to linear differential equations. This is where Picard-Vessiot extensions come into place.

But before we introduce these extensions, we need to provide some definitions and properties of differential fields and rings.

Definition B.27. Let (R, ∂) be a differential ring. We say that an ideal $I \subset R$ is a *differential ideal* if $\partial(I) \subset I$. We say that R is *simple* if the only differential ideals are (0) and R .

Note that all fields are simple, since the only two ideals of a field are the zero ideal and the whole field. We can see that being a simple differential ring is a strong property:

Lemma B.28. *Let (R, ∂) be a simple differential ring. Then R is an integral domain.*

Proof. First, we are going to show that if $r \in R$ is not nilpotent, then r can not be a zero divisor. Assume that r is not nilpotent, i.e., $r^n \neq 0$ for all $n \in \mathbb{N}$. Consider the set

$$I_r = \{s \in R : \text{there is } n \in \mathbb{N} \text{ with } r^n s = 0\}.$$

It is easy to check that I_r is an ideal with $1 \notin I_r$. Moreover, if $s \in I_r$, then we have that $nr^{n-1}\partial(r)s + r^n\partial(s) = 0$. Multiplying that identity by r , and using the fact that $r^n s = 0$, we obtain:

$$r^{n+1}\partial(s) = 0,$$

meaning that $\partial(s) \in I_r$. Hence I_r is a differential ideal. Since R is simple, then $I_r = (0)$. This implies that r is not a zero divisor.

Consider now the ideal J of nilpotent elements. It is clear that $1 \notin J$. Let $r \in J$. Then there is a minimal $n \in \mathbb{N}$ such that $r^n = 0$. Computing the derivative of that identity yields

$$nr^{n-1}\partial(r) = 0,$$

which means that, since $nr^{n-1} \neq 0$ that $\partial(r)$ is a zero divisor and, in particular, a nilpotent element (i.e., $\partial(r) \in J$). Hence J is a differential ideal. Since R is simple, we have $J = (0)$. These two points together show that there are no zero divisors on R , meaning that R is an integral domain. \square

This Lemma is interesting by itself because it shows the rigidity of the differential rings. If all the non-trivial ideals are not closed under derivation, then there are no zero divisors. This can be used for example to show that $(\mathbb{K}[[x]], \partial_x)$ is an integral domain. There, the ideals are of the form (x^n) for $n \in \mathbb{N}$ and, since $\partial_x(x^n) = nx^{n-1} \notin (x^n)$, they are not differential ideals.

Lemma B.29. *Let (F, ∂) be a differential field with an algebraically closed field of constants C . Suppose that (R, ∂) is a simple differential ring extending F and R is finitely generated. Then its field of fractions E has C as field of constants.*

Proof. First of all, we are going to show that if $c \in C(E)$, then $c \in R$. This is an interesting property of simple differential rings. Let $c \in C(E)$ and consider $I_c = \{r \in R : cr \in R\}$. This set is clearly an ideal of R . Moreover, since c is a constant, we have that

$$\partial(r)c = \partial(rc) - \partial(c)r = \partial(rc) \in R,$$

showing that I_c is a differential ideal. Using now that $c \neq 0$, then $I_c \neq (0)$ and, by the simplicity of R , $I_c = R$. In particular, $1 \in I_c$, showing that $c \in R$.

For the rest of the proof (showing that $c \in C$), we refer to [70, Lemma 17]. \square

Lemma B.30. *Let (R, ∂) be a differential ring and $I \subsetneq R$ a differential ideal. There exists a unique derivation $\bar{\partial}$ on R/I that satisfies*

$$\pi \circ \partial = \bar{\partial} \circ \pi,$$

where π is the canonical projection $\pi : R \rightarrow R/I$.

Proof. Consider the map $\bar{\partial}(r + I) = \partial(r) + I$. First we check that it is well defined. If $r - s \in I$, then $\partial(r) - \partial(s) \in I$ since I is a differential ideal. Hence, $\partial(r) + I = \partial(s) + I$.

Checking that $\bar{\partial}$ is a derivation is a straightforward computation using the fact that ∂ is a derivation. Moreover, it is clear by definition that:

$$\pi(\partial(r)) = \partial(r) + I = \bar{\partial}(r + I) = \bar{\partial}(\pi(r)).$$

Now, consider D another derivation on R/I that satisfies $\pi \circ \partial = D \circ \pi$. Let $r + I \in R/I$. We have that

$$D(r + I) = D(\pi(r)) = \pi(\partial(r)) = \partial(r) + I = \bar{\partial}(r + I),$$

proving that $D \equiv \bar{\partial}$. \square

Definition B.31. Let (F, ∂) be a differential ring and $L \in F[\partial]$ a linear differential operator of order n . A *Picard-Vessiot ring for the equation $L \cdot y = \alpha$* is a differential ring extension of (F, ∂) such that:

1. R is a simple differential ring.
2. There exist $f_1, \dots, f_n \in R$ solutions to $L \cdot y = \alpha$ that are $C(F)$ -linearly independent.
3. R is generated by the set

$$\left\{ \partial^j(f_i) : i = 1, \dots, n, \quad j = 0, \dots, n-1 \right\} \cup \left\{ \frac{1}{w(f_1, \dots, f_n)} \right\}.$$

The previous Lemmas of this Appendix are helpful to prove directly that if (R, ∂) is a Picard-Vessiot ring for a differential equation then:

- R is an integral domain (using Lemma B.28)
- R has all possible independent solutions to $L \cdot y = 0$ (see Lemma B.22).
- $C(R) = C(F)$ (see Lemma B.29)
- The derivation on R extending ∂ is unique (combination of Proposition B.9 and Lemma B.30).

The next step is to prove that for any $L \in F[\partial]$ and $\alpha \in F$, a Picard-Vessiot ring always exists and, in some sense, is unique. This may seem a very technical result but allows us to talk about *the* Picard-Vessiot extension of a differential field.

Proposition B.32. *Let (F, ∂) be a differential field, $L \in F[\partial]$ and $\alpha \in F$. Then:*

1. *There exists a Picard-Vessiot ring for the equation $L \cdot y = \alpha$.*
2. *If R_1, R_2 are Picard-Vessiot rings for the equation $L \cdot y = \alpha$, then they are isomorphic.*

Proof. Proving the existence is a simple game using Proposition B.9. Assume that $L = p_0 + \dots + p_n \partial^n$ and consider the variables $X_{i,j}$ for $i = 1, \dots, n$ and $j = 0, \dots, n-1$. Take the differential ring $F[X_{i,j}]$ where:

- $\partial(X_{i,j}) = X_{i,j+1}$ for $i = 1, \dots, n$ and $j = 0, \dots, n-2$.
- $\partial(X_{i,n-1}) = -\frac{1}{p_n} (p_0 X_{i,0} + \dots + p_{n-1} X_{i,n-1})$.

In this new ring, consider the element $W = \det(X_{i,j})$. It is clear that the set $M = \{W^k : k \in \mathbb{N}\}$ is multiplicatively closed. Hence we can build the ring $F[X_{i,j}, \frac{1}{W}]$ as the localization of $F[X_{i,j}]$ by M . Here, by Proposition B.7, the derivation is uniquely determined.

At this point, we have a ring with all the possible solutions to the differential equation, that is finitely generated by these solutions, their derivatives and their wronskian. The only missing condition for this ring to be a Picard-Vessiot ring of L is the simplicity condition. If we take now $I \subset F[X_{i,j}, \frac{1}{W}]$ a maximal differential ideal then $R = F[X_{i,j}, \frac{1}{W}]/I$ is now a Picard-Vessiot ring for L .

Showing the *uniqueness* of the Picard-Vessiot ring is a more technical proof. The main idea is, given two Picard-Vessiot rings, we build a mixed ring where the two rings coexist and then show that in this ring we have *too many* solutions to L , leading to a simple connection between these solutions.

Consider R_1, R_2 two Picard-Vessiot rings for L and let $f_1, \dots, f_n \in R_1$ and $g_1, \dots, g_n \in R_2$ be the $C(F)$ -linearly independent solutions of L .

Consider now the differential ring $R_1 \otimes_F R_2$ where the derivation is defined by

$$\partial(r_1 \otimes r_2) = \partial(r_1) \otimes r_2 + r_1 \otimes \partial(r_2).$$

Now take a maximal differential ideal $I \subset R_1 \otimes_F R_2$. Let R_3 be the quotient $(R_1 \otimes_F R_2)/I$. By construction, there are obvious maps $\phi_i : R_i \rightarrow R_3$ defined by

$$\phi_1(r_1) = \pi(r_1 \otimes 1), \quad \phi_2(r_2) = \pi(1 \otimes r_2).$$

It is well known that the kernel of ϕ_i is an ideal in R_i . Moreover, we can check that these kernels are differential ideals. Using the fact that both R_1 and R_2 are simple, we get that these kernels are the zero ideal, i.e., ϕ_1 and ϕ_2 are injective maps.

On the other hand, $\{\phi_1(f_i) : i = 1, \dots, n\}$ and $\{\phi_2(g_i) : i = 1, \dots, n\}$ are basis of $V_{R_3}(L)$. Hence, there is a change of basis $\psi : \phi_1(R_1) \rightarrow \phi_2(R_2)$ that is a $C(F)$ -linear map.

It is now easy to check that $\phi : R_1 \rightarrow R_2$ defined by $\phi = \phi_2^{-1} \circ \psi \circ \phi_1$ is a differential isomorphism between the two Picard-Vessiot rings. \square

Definition B.33. Let (F, ∂) be a differential field, $L \in F[\partial]$ and $\alpha \in F$. We call *Picard-Vessiot field of the equation $L \cdot y = \alpha$* (also PV-field or PV-extension) to the field of fractions of the Picard-Vessiot ring of $L \cdot y = \alpha$ and we denote it with $PV_F(L \cdot y = \alpha)$. We also write $PV(L)$ if the field F is known from the context and $\alpha = 0$.

PV-fields of a linear differential equation can be seen as the *minimal* fields preserving the constants that have all possible solutions to a differential equation.

Example B.34 (PV-extension for primitive equations). Let (F, ∂) be and consider the differential equation $\partial(y) = \alpha$ for some $\alpha \in F$.

If $V_F(L) \neq \{0\}$, then $PV(L) = F$ since we have already all possible solutions.

Otherwise, we know by Proposition B.18 that the solution $t \in PV(L)$ will be transcendental. Moreover, we know that $PV(L \cdot y = \alpha)$ is the field of fractions of a ring generated by t and $1/t$ (which is exactly the wronskian appearing in the definition of Picard-Vessiot rings). It is now easy to check that such field of fractions is exactly the field $F(t)$, which was the monomial extension of F that we studied in Section B.1.

Example B.35 (PV-extension for hyperexponential equations). Let (F, ∂) be and consider the operator $L = \partial - \alpha \in F[\partial]$.

In this case we consider the homogeneous equation $L \cdot y = 0$. Any solution will be, according to the definitions in Section B.1, a hyperexponential element. Let $t \in PV(L)$ be the solution to $L \cdot t = 0$. We know, by definition of PV-field, that $PV(L)$ is generated by t and $1/t$.

Assume first that t is transcendental. Then we have $PV(L) = F(t)$, i.e., the Picard-Vessiot field is exactly the monomial extension described in Section B.1.

There is only one other option: t is algebraic. Then by Proposition B.19, we have that $t^n = \beta \in F$ for some minimal $n \in \mathbb{N}$. We claim that $PV(L) = F[T]/(T^n - \beta)$.

There, $t = \bar{T}$ is a solution to $L \cdot y = 0$. So we need to show it is simple. Let I be a non-trivial differential ideal. Then there is a monic $p(T)$ with minimal $\deg(p) = d < n$ such that $p(t) \in I$. We know that $\partial(p(t)) \in I$ because I is a differential ideal, so $q(t) = \partial(p(t)) - d\alpha p(t) \in I$. If we look here, we have:

$$\partial(p(t)) = \kappa_{\partial}(p)(t) + \alpha t \partial_t(p)(t).$$

so if we look to the coefficient t^d in $q(t)$ we have:

$$[t^d]q(t) = d\alpha - d\alpha = 0.$$

This contradicts the minimality of d . Hence I is trivial and the algebraic extension $F[T]/(T^n - \beta)$ (where $\partial(\beta)/\beta = n\alpha$) is the Picard-Vessiot field.

B.3.1 Picard-Vessiot closure

Until now, we have defined what a Picard-Vessiot extension is. It is similar to the separable extensions in field theory. There, we added all the possible solutions to a polynomial equation making the polynomial fully factorable. In a PV-extension we add all possible solutions to a linear differential equation.

If we look into field theory, we have then the concept of algebraic closure of a field. That means that all polynomials can be factored into linear components or, in other words, we can find all the possible solutions to polynomial equations. In this subsection we study the analogue of this algebraic closure in the case of linear differential equations.

First of all, we need a similar result to the primitive element theorem, that guarantees that for any algebraic extension there is one equation that describes the extension.

Lemma B.36. *Let (F, ∂) be a differential field and $L_1, L_2 \in F[\partial]$. There is $L \in F[\partial]$ such that for any differential extension (E, ∂) and any element $f \in E$*

$$L_1 \cdot f = 0 \text{ or } L_2 \cdot f = 0 \implies L \cdot f = 0.$$

Proof. In the non-commutative ring $F[\partial]$, there is an Euclidean algorithm such that for all $L_1, L_2 \in F[\partial]$, there are unique $R, S \in F[\partial]$ with $\text{ord}(S) < \text{ord}(L_2)$ and

$$L_1 = RL_2 + S. \quad (\text{B.37})$$

Assume w.l.o.g. that $\text{ord}(L_1) \geq \text{ord}(L_2)$. We are going to prove by induction on $\text{ord}(L_2)$ that there are $L, A, B \in F[\partial]$ such that $L = AL_1 = BL_2$.

First, note that if $L_1 = RL_2$, then we can take $L = L_1$. When $\text{ord}(L_2) = 0$, that is always the case. Now, for $\text{ord}(L_2) > 0$ we have that the corresponding euclidean coefficients R and S satisfy:

- $R \neq 0$ (since $\text{ord}(L_2) \leq \text{ord}(L_1)$) and
- $\text{ord}(S) < \text{ord}(L_2)$ or $S = 0$.

If $S = 0$ we have $L = L_1$, $A = 1$ and $B = R$. Otherwise, by induction hypothesis, there are $\tilde{L}, C, D \in F[\partial]$ such that $\tilde{L} = CL_1 = DS$. Multiplying (B.37) by D we obtain:

$$DL_1 = DRL_2 + DS \implies (D - C)L_1 = DRL_2,$$

so $L = (D - C)L_1$, $A = D - C$ and $B = DR$ have the desired property.

Once we have a common left multiple L of L_1 and L_2 it is easy to check that for any f in any differential extension (E, ∂) ,

$$L_1 \cdot f = 0 \text{ or } L_2 \cdot f = 0 \implies L \cdot f = 0.$$

□

We call any operator with minimal order and that property a least common left multiple of L_1 and L_2 , and denote it by $\text{lclm}(L_1, L_2)$. It is now clear that if $L_1 = DL_2$, then the corresponding PV-field for the equations $L_1 \cdot y = 0$ and $L_2 \cdot y = 0$ are closely related:

Lemma B.38. *Let (F, ∂) be a differential field and $L_1, L_2 \in F[\partial]$ such that there is $D \in F[\partial]$ with $L_1 = DL_2$. Then*

$$PV(L_2) \subset PV(L_1).$$

Proof. Since $L_2 \cdot f = 0$ implies $L_1 \cdot f = 0$, then there are solutions to L_2 in $PV(L_1)$, since in this field we have all possible solutions to L_1 . In particular, if f_1, \dots, f_n are $C(F)$ -linearly independent solutions of L_2 in $PV(L_1)$, then we have a simple inclusion map defined from $PV(L_2)$, just mapping each of the solutions in $PV(L_2)$ to one of the f_i . □

Moreover, we can consider only homogeneous differential equations without lost of generality, since if $L \cdot f = \alpha$, then it is clear that $(\partial L - \frac{\partial(\alpha)}{\alpha}) \cdot f = 0$. With all these properties we can easily prove that the set of all Picard-Vessiot extensions of (F, ∂) form a direct system with respect to the inclusion. This means that for each two PV-fields E_1, E_2 , there is another PV-field E_3 such that $E_1, E_2 \subset E_3$.

We can now build a direct limit [6, Chapter 2] of differential fields, and obtain again a differential field \bar{F}^∂ such that

- The field of constants is $C(F)$.
- For every $L \in F[\partial]$ of order n there are $f_1, \dots, f_n \in \bar{F}^\partial$ such that $L \cdot f_i = 0$ for $i = 1, \dots, n$ and are $C(F)$ -linearly independent.

However, this is not *complete* yet. We know that in the algebraic closure of a field, not only the polynomials over the original field can be decompose into linear factors, but also all polynomials with coefficients in the algebraic closure have that property. This is not the case with the direct limit of Picard-Vessiot fields. In order to get this extra property we need to do one last construction.

Definition B.39. Let (F, ∂) be a differential field. Let $F_0 = F$ and $F_{i+1} = \bar{F}_i^\partial$. This yields a chain of differential extensions $F_n \subset F_{n+1}$. We define the *Picard-Vessiot closure of F* as the limit of those fields, i.e.,

$$F_{PV} = \bigcup_{n \in \mathbb{N}} F_n.$$

On one hand, it is easy to see that the set of constants of the PV-closure is still $C(F)$. On the other hand, this PV-closure has now the property that for any $L \in F_{PV}[\partial]$ of order n , its solution space within F_{PV} is a $C(F)$ -vector space of dimension exactly n .

B.4 Galois groups of PV-extensions

In this section, we focus on the definition and main properties of the differential Galois groups. In the classical theory, the Galois group was the group of automorphisms on a field that permuted the solutions for a particular polynomial equation.

In the differential case, we look for a similar property. But the elements in the group has to be restrained to commute with the derivation.

Definition B.40. Let (F, ∂) be a differential field and (E, ∂) a differential extension. We call *the differential Galois group of E over F* to the set of F -automorphisms that commute with ∂ . Namely,

$$\text{Gal}_\partial(E/F) = \{\sigma : E \rightarrow E : \sigma|_F = \text{id}_F \text{ and } \sigma \circ \partial = \partial \circ \sigma\}$$

This group is particularly interesting in PV-extensions because its elements permute the solutions to linear differential equations:

Lemma B.41. *Let (F, ∂) be a differential field, (E, ∂) a differential extension of F , $L \in F[\partial]$ and $\alpha \in F$. Then for all $f \in E$ such that $L \cdot f = \alpha$, and any $\sigma \in \text{Gal}_\partial(E/F)$:*

$$L \cdot \sigma(f) = \alpha.$$

In particular, if E is a PV-extension of F for the differential equation $L \cdot y = 0$, then the elements in $\text{Gal}_\partial(E/F)$ permutes the solutions of the defining differential equation.

Proof. Let $f \in E$ such that $L \cdot f = \alpha$. Since $L \in F[\partial]$ and σ commutes with ∂ , it is clear that $L \cdot \sigma(f) = \sigma(L \cdot f)$. On the other hand, since σ fixes all the elements in F , we have:

$$L \cdot \sigma(f) = \sigma(L \cdot f) = \sigma(\alpha) = \alpha.$$

□

At this stage, the differential Galois group on a PV-extension can be seen as automorphisms that act over the vector space of solutions. This action is, in fact, very particular. Since $\sigma \in \text{Gal}_\partial$ is an automorphism that fixes the set $\sigma(V_E(L \cdot y = \alpha))$, then $\sigma|_{V_E(L \cdot y = \alpha)}$ is a $C(F)$ -linear bijection. Hence, we have a relation between $\text{Gal}_\partial(E/F)$ and matrices in $GL_n(C(F))$, where n is the order of L .

We have seen in Examples B.35 that the PV-extension of a hyperexponential equation $\partial(y) = \alpha y$ are always $F(t)$ if t is transcendental or $F[T]/(T^n - \beta)$ for some $\beta \in F$. Now we see how we can also obtain that using the differential Galois group.

Lemma B.42. *Let (F, ∂) be a differential field with an algebraically closed field of constants C , and $L = \partial - \alpha$. Let $E = \text{PV}(L)$ of the form $F(t)$, where t is a solution to $L \cdot y = 0$ and $G = \text{Gal}_\partial(E/F)$. Then*

- $t \in F$ if and only if $G = \{id_F\}$.
- t is algebraic over F if and only if G is a cyclic group.
- t is transcendental over F if and only if $G \simeq C^*$.

Proof. Since the order of L is 1, we can associate each element of G with a constant c such that $\sigma_c(t) = ct$.

We have only three options for t . First, $t \in F$. Then $E = F$ and there is only one automorphism that fix F , namely, id_F . Hence $G = \{\sigma_1 = id_F\}$.

Second, t is algebraic over F . As we saw in Example B.35, we know then that $E = F[T]/(T^n - \beta)$ for some $\beta \in F$. In particular, if $\sigma_c \in G$, we have that:

$$0 = \sigma_c(0) = \sigma_c(t^n - \beta) = c^n t^n - \beta = \beta(c^n - 1).$$

So we need that $c^n = 1$. This implies that G is a cyclic group of order n .

Third, t is transcendental. Then for any $c \in C^*$, the isomorphism σ_c is well defined. Hence $G \simeq C^*$. \square

Before concluding this Appendix, we see how these results about the differential Galois group can help to prove algebraic properties of some elements in PV-extensions.

Proposition B.43 ([39, Proposition 24]). *Let (F, ∂) be a differential field with algebraically closed field of constants C . Let E be a PV-extension of F . Let $u, v \in E \setminus \{0\}$ such that*

$$\frac{\partial(u)}{u} = \alpha \in F, \quad \frac{\partial(v)}{v} = u.$$

Then u is algebraic over F .

Proof. Assume that u is transcendental over F . As $\partial(u)/u \in F$, we have that $F(u)$ is the PV-extension of F for the equation $\partial(y) = \alpha y$. Using Lemma B.42, we have that $G = \text{Gal}_\partial(F(u)/u) \simeq C^*$. Let $\sigma_c \in G$ such that $\sigma_c(u) = cu$.

Since E is an extension of $F(u)$, we can extend any $\sigma_c \in G$ to a $\tilde{\sigma}_c \in \text{Gal}_\partial(E/F)$. In fact, for any extension that we consider, we have:

$$\frac{\partial(\tilde{\sigma}_c(v))}{\tilde{\sigma}_c(v)} = \frac{\tilde{\sigma}_c(\partial(v))}{\tilde{\sigma}_c(v)} = \sigma_c(u) = cu.$$

Using the fact that E is a PV-extension of F , we know that E is finitely generated and, in particular, it has a finite transcendence degree m . Let $c_0, \dots, c_m \in C$ be $(m+1)$ \mathbb{Q} -linearly independent constants and consider the elements $v_i = \tilde{\sigma}_{c_i}(v)$.

As a consequence of the Kolchin-Ostrowsky Theorem [47, 59], we have that there are integers n_0, \dots, n_m (not all zero) such that

$$z := \prod_{i=0}^m v_i^{n_i} \in F(u).$$

On one hand, we have that $\partial(z) = \beta z$ for $\beta = u(\sum_{i=0}^m n_i c_i)$. On the other hand, z has to be transcendental over F , otherwise u will be algebraic. If we now factor the numerator and denominator of z as

$$z = \gamma u^{r_0} \prod_{j=1}^t (u - \alpha_j)^{r_j},$$

where $\gamma, \alpha_j \in \bar{F}$, $\alpha_j \neq 0$ and the $r_j \in \mathbb{Z}$. Using Lemma B.5, we have that the logarithmic derivative of z is

$$\frac{\partial(z)}{z} = \frac{\partial(\gamma)}{\gamma} + r_0\alpha + \sum_{j=1}^t r_j \frac{\partial(u) - \partial(\alpha_j)}{u - \alpha_j}. \quad (\text{B.44})$$

The last quotients can be written removing $\partial(u)$ from the numerator using the fact that $\partial(u) = \alpha u$, so

$$\frac{\partial(u) - \partial(\alpha_j)}{u - \alpha_j} = \frac{\alpha u - \partial(\alpha_j)}{u - \alpha_j} = \alpha + \frac{\alpha\alpha_j - \partial(\alpha_j)}{u - \alpha_j}.$$

We know plug this final expression into (B.44) and, using the fact that $\partial(z) = \beta z$:

$$\left(\sum_{i=0}^m n_i c_i \right) u = \beta = \frac{\partial(\gamma)}{\gamma} + \alpha \sum_{j=0}^t r_j + \sum_{j=1}^m r_j \frac{\alpha\alpha_j - \partial(\alpha_j)}{u - \alpha_j}. \quad (\text{B.45})$$

If we analyze the previous equation, we see that in the left hand side we have u only appearing with degree 1 and in the right hand side, only appearing in the denominator of the last summands.

If, for any j , $(\alpha\alpha_j - \partial(\alpha_j)) = 0$, then we would have that $\partial(\alpha_j/u) = 0$, so $u \in \bar{F}$. This would contradict the assumption of u being transcendental.

Then we need that all $r_i = 0$ for $i = 1, \dots, m$. We can rewrite equation (B.45) into

$$\left(\sum_{i=0}^m n_i c_i \right) u = \frac{\partial(\gamma)}{\gamma} + r_0\alpha,$$

which implies that u is algebraic over F . This is a contradiction, proving finally that u can not be transcendental. \square