ÖAW Austrian Academy of Sciences

Doctoral Program
Computational Mathematics
Numerical Analysis and Symbolic Computation

JKU JOHANNES KEPLER UNIVERSITY LINZ

Algorithmic Arithmetics with DD-Finite Functions

Antonio Jiménez-Pastor Veronika Pillwein

DK-Report No. 2018-03 02 2018

A–4040 LINZ, ALTENBERGERSTRASSE 69, AUSTRIA

Supported by

Austrian Science Fund (FWF) Upper Austria

FWF Der Wissenschaftsfonds.

LAND OBERÖSTERREICH

DK sponsors:

- **Johannes Kepler University Linz** (JKU)
- **Austrian Science Fund** (FWF)
- **Upper Austria**

Algorithmic Arithmetics with DD-Finite Functions

Antonio Jiménez-Pastor[a], Veronika Pillwein[b]

[a] *Doctoral Program Computational Mathematics, JKU, Linz*
[b] *Research Institute for Symbolic Computation, JKU, Linz*

Abstract

Many special functions as well as generating functions of combinatorial sequences that arise in applications are D-finite, i.e., they satisfy a linear differential equation with polynomial coefficients. These functions have been studied for centuries and over the past decades various computer algebra methods have been developed and implemented for D-finite functions. Recently, we have extended this notion to DD-finite functions (functions satisfying linear differential equations with D-finite functions coefficients). Numerous identities for D-finite functions can be proven automatically using closure properties. These closure properties can be shown to hold for DD-finite functions as well. In this paper, we present the algorithmic aspect of these closure properties, discuss issues related to implementation and give several examples.

1. Introduction

During the past decades, algorithms for functions satisfying ordinary linear differential equations have been studied in the case where the coefficients of the differential equation were polynomials. Those functions were called *differentiably finite* or in short D-finite [13, 19, 20]. The D-finite (formal) power series are interesting for many reasons, e.g., many generating functions for combinatorial sequences arising in applications are of this type as well as many special functions [3, 18].

It is well known that D-finite functions are closed under operations such as addition, multiplication, algebraic substitution, etc.; and also implementations in various computer algebra systems exist to compute effectively these closure properties [6, 12, 15, 16]. The finiteness in the representation of D-finite functions is critical in those algorithms. In order to specify a D-finite function only the order of the equation, the polynomial coefficients, and sufficiently many initial values need to be stored. Given functions in this form, a defining differential equation of the same type (plus initial values) for their sum, product, etc., can be computed automatically. These algorithms can be used to prove identities [11].

Even though many interesting objects are D-finite, there are more special functions out there. Recently, we have extended the notion of D-finite functions to DD-finite functions [10]. This new class of formal power series satisfies linear differential equations with D-finite functions as coefficients. We extended most of the closure properties for D-finite functions to the DD-finite case. Moreover, we carried out these closure properties for power series that satisfy linear differential equations with coefficients in an arbitrary differential ring. This setting covers cases already studied by Van Hoeij [23] in the context of factorization and formal solutions, and Abramov et al. [1, 2] for linear differential systems. Also classic non D-finite functions as $\exp(\exp(x))$, $\tan(x)$, and Mathieu's functions [7] are in this new class.

It turns out that, not only the closure properties, but also the corresponding algorithms for D-finite functions can be extended to the more general case. We have implemented those algorithms in SAGE [21, 9] using as input structure the defining differential equation and initial values for each function. In this paper, we describe and explain our implementation of these closure properties and how we maintain an acceptable performance.

In section 2 we introduce some mathematical background that eases the description in the algorithms. Then in section 3 we recall the formal definition of a differentially definable function and the relation with the theory mentioned above. In sections 4 and 5 we detail the algorithms for computing the addition

Email addresses: antonio.jimenez-pastor@dk-compmath.jku.at (Antonio Jiménez-Pastor), vpillwei@risc.jku.at (Veronika Pillwein)

This research was funded by the Austrian Science Fund (FWF): W1214-N15, project DK15.

and product in this class of functions. In particular, in section 4 we give the general structure of the algorithm and in section 5 the issues of the implementation we provide. We close with two concrete examples carried out in some detail.

2. Differential Linear Algebra

In the present section we study a general framework that will be useful later during the description of the algorithms: *differential linear algebra*. In the same spirit as differential rings and fields are defined [5, 22] we want to work with differential vector spaces adding a derivation operator over a vector space.

Definition 2.1. [14, 22] Let (K, ∂) be a differential field of characteristic zero and V a K-vector space. We say that $\vec{\partial} : V \to V$ is a *derivation over V w.r.t. ∂* if it satisfies

1. $\vec{\partial}(\mathbf{v} + \mathbf{w}) = \vec{\partial}(\mathbf{v}) + \vec{\partial}(\mathbf{w})$ for all $\mathbf{v}, \mathbf{w} \in V$.
2. $\vec{\partial}(c\mathbf{v}) = \partial(c)\mathbf{v} + c\vec{\partial}(\mathbf{v})$ for all $c \in K$ and $\mathbf{v} \in V$.

We say that $(V, \vec{\partial})$ is a *differential vector space over* (K, ∂). We denote by $\Delta_\partial(V)$ the set of all derivations over V w.r.t. ∂. We say that $\mathbf{v} \in V$ is a *constant* if $\vec{\partial}(\mathbf{v}) = 0$. We denote by $C_{\vec{\partial}}(V)$ the set of constants of V w.r.t. the derivation $\vec{\partial}$.

In this definition we see that a differential vector space relies on a previously given differential field. Hence the set $\Delta_\partial(V)$ depends directly on the derivation ∂ of the ground field.

It is well-known [22] that a derivation over V w.r.t. ∂ is uniquely defined by the values it takes over a basis $B \subset V$. In particular, we denote by κ_∂^B the unique derivation s.t. $B \subset C_{\kappa_\partial^B}(V)$ and call it the *coefficient-wise derivation w.r.t. a basis B*. Usually we omit ∂ and B and write simply κ if the derivative and basis are clear from the context. If $\mathbf{v}$ has coordinates v_b in the basis B, then $\kappa(\mathbf{v})$ has coordinates $\partial(v_b)$, hence the name.

Derivations over V w.r.t. ∂ can be related with linear endomorphisms on V.

Lemma 2.2. *Let $(V, \vec{\partial})$ be a differential vector space over (K, ∂). Let B be a basis of V and $f : V \to V$ the endomorphism such that $f(\mathbf{b}) = \vec{\partial}(\mathbf{b})$ for any $\mathbf{b} \in B$. Then, for any $\mathbf{v} \in V$:*

$$\vec{\partial}(\mathbf{v}) = \kappa(\mathbf{v}) + f(\mathbf{v}).$$

Proof. Let $\mathbf{v} \in V$. As B is a basis of V, we can write $\mathbf{v}$ as a linear combination of its elements. We apply the basic properties of a derivation and the definition of κ and f and obtain;

$$\begin{aligned}\vec{\partial}(\mathbf{v}) &= \vec{\partial}\left(\sum_{\mathbf{b}\in B} v_b\mathbf{b}\right) = \sum_{\mathbf{b}\in B} \vec{\partial}(v_b\mathbf{b}) = \sum_{\mathbf{b}\in B} (\partial(v_b)\mathbf{b} + v_b\vec{\partial}(\mathbf{b})) \\ &= \sum_{\mathbf{b}\in B} (\partial(v_b)\mathbf{b}) + \sum_{\mathbf{b}\in B} (v_b f(\mathbf{b})) = \kappa(\mathbf{v}) + f(\mathbf{v}).\end{aligned}$$

□

Lemma 2.2 yields that we can associate a derivation $\vec{\partial} \in \Delta_\partial(V)$ with the matrix representation of the linear mapping f. Then, due to the definition of f, this associated matrix is the matrix whose columns represent the derivatives of the basis elements. This idea can be generalized to any list of generators of the vector space instead of having a basis.

Definition 2.3. Let (K, ∂) be a differential field, $(V, \vec{\partial})$ be a differential vector space over (K, ∂), and $\Phi = \{\phi_1, ..., \phi_n\}$ be generators of V. We say a matrix $M = (m_{ij})_{i,j=1}^n$ is a *derivation matrix of $\vec{\partial}$ w.r.t. Φ* if it satisfies

$$\vec{\partial}(\phi_j) = m_{1j}\phi_1 + \cdots + m_{nj}\phi_n, \quad \text{for all } j = 1, \ldots, n.$$

In this setting, mimicking the proof of Lemma 2.2, for any $\mathbf{v} = v_1\phi_1 + \cdots + v_n\phi_n \in V$ we have

$$\vec{\partial}(\mathbf{v}) = \partial(v_1)\phi_1 + v_1\vec{\partial}(\phi_1) + \cdots + \partial(v_n)\phi_n + v_n\vec{\partial}(\phi_n) = \sum_{j=1}^{n}\left(\sum_{i=1}^{m} m_{ij}v_i + \partial(v_j)\right)\phi_j.$$

In matrix-vector notation, a representation of $\vec{\partial}(\mathbf{v}) = \hat{v}_1\phi_1 + \cdots + \hat{v}_n\phi_n$ can thus be computed as

$$\begin{pmatrix}\hat{v}_1\\ \hat{v}_2\\ \vdots\\ \hat{v}_n\end{pmatrix} = M\begin{pmatrix}v_1\\ v_2\\ \vdots\\ v_n\end{pmatrix} + \begin{pmatrix}\partial(v_1)\\ \partial(v_2)\\ \vdots\\ \partial(v_n)\end{pmatrix}.$$

Conversely, if a matrix M satisfy the previous equality (namely, a representation for the derivatives in V can be computed with that formula), then M is a derivation matrix.

For any set of generators Φ a derivation matrix can be computed and it is unique if Φ is a basis of V. With these notations, we can compute the derivative in a differential vector space using matrix operations. This is computationally interesting, since it allows us to implement derivatives in any vector space only using the knowledge of the generators. We consider now how these derivation matrices behave when we operate with the vector spaces, for instance, with the direct sum or the tensor product.

Proposition 2.4 (Direct sum)**.** *Let $(V, \vec{\partial}_V), (W, \vec{\partial}_W)$ be two differential vector spaces over (K, ∂) of dimensions n and m, with basis B_V and B_W. Let M_V and M_W be the derivation matrices of V and W, respectively. Then:*

(1) The map $\vec{\partial} : V \oplus W \to V \oplus W$ defined as

$$\vec{\partial}(\mathbf{v} \oplus \mathbf{w}) = \vec{\partial}_V(\mathbf{v}) \oplus \vec{\partial}_W(\mathbf{w})$$

is the unique derivation over $V \oplus W$ w.r.t. ∂ that extends $\vec{\partial}_V$ and $\vec{\partial}_W$.

(2) The derivation matrix associated with the basis $B_V \oplus B_W$ is $M_V \oplus M_W$.

Proof. We build $V \oplus W$ as $V \times W$ with the following operations:

- $(v_1, w_1) + (v_2, w_2) = (v_1 + v_2, w_1 + w_2)$,
- $c(v, w) = (cv, cw)$,

and represent $(\mathbf{v}, \mathbf{w})$ by $\mathbf{v} \oplus \mathbf{w}$. Obviously $\vec{\partial}$ is a derivation on $V \oplus W$. Consider now a derivation $\hat{\partial}$ over $V \oplus W$ w.r.t. ∂ that extends $\vec{\partial}_V$ and $\vec{\partial}_W$ (i.e., $\hat{\partial}(\mathbf{v} \oplus 0) = \vec{\partial}_V(\mathbf{v}) \oplus 0$ and $\hat{\partial}(0 \oplus \mathbf{w}) = 0 \oplus \vec{\partial}_W(\mathbf{w})$). Then we have that:

$$\begin{aligned}\hat{\partial}(\mathbf{v} \oplus \mathbf{w}) &= \hat{\partial}((\mathbf{v} \oplus 0) + (0 \oplus \mathbf{w})) = (\vec{\partial}_V(\mathbf{v}) \oplus 0) + (0 \oplus \vec{\partial}_W(\mathbf{w}))\\ &= \vec{\partial}_V(\mathbf{v}) \oplus \vec{\partial}_W(\mathbf{w}) = \vec{\partial}(\mathbf{v} \oplus \mathbf{w}),\end{aligned}$$

hence $\vec{\partial}$ is unique extending $\vec{\partial}_V$ and $\vec{\partial}_W$. Since $B_V \oplus B_W = \{\mathbf{b} \oplus 0 \ : \ \mathbf{b} \in B_V\} \cup \{0 \oplus \mathbf{b} \ : \ \mathbf{b} \in B_W\}$, it is a basis of $V \oplus W$. Clearly, the derivation matrix is the direct sum of the derivation matrices. □

Proposition 2.5 (Tensor product)**.** *Let $(V, \vec{\partial}_V), (W, \vec{\partial}_W)$ be two differential vector spaces over (K, ∂) of dimensions n and m and with bases B_V and B_W. Let M_V and M_W be the derivation matrices of V and W, respectively. Then:*

(1) The map $\vec{\partial} : V \otimes W \to V \otimes W$ defined as

$$\vec{\partial}(\mathbf{v} \otimes \mathbf{w}) = \vec{\partial}_V(\mathbf{v}) \otimes \mathbf{w} + \mathbf{v} \otimes \vec{\partial}_W(\mathbf{w})$$

is a derivation over $V \otimes W$ w.r.t. ∂.

(2) The derivation matrix associated with the basis $B_V \otimes B_W$ is

$$M_V \otimes \mathcal{I}_m + \mathcal{I}_n \otimes M_W.$$

Proof. We build $V \otimes W$ as 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 element $(\mathbf{v}, \mathbf{w})$ by $\mathbf{v} \otimes \mathbf{w}$. To prove that $\vec{\partial}$ is a derivation over $V \otimes W$, we first need to check that $\vec{\partial}$ is well-defined with respect to the three relations, which is a straightforward computation. Then the additive property and the Leibniz rule are granted by definition.

To prove that the derivation matrix has the appropriate shape, consider an element $\mathbf{v} \otimes \mathbf{w}$ in the basis $B_V \otimes B_W$, and let f_V, f_W be the linear mappings associated with $\vec{\partial}_V$ and $\vec{\partial}_W$ as we saw in Lemma 2.2. Then:

$$\begin{aligned}\vec{\partial}(\mathbf{v} \otimes \mathbf{w}) &= \vec{\partial}_V(\mathbf{v}) \otimes \mathbf{w} + \mathbf{v} \otimes \vec{\partial}_W(\mathbf{w}) = f_V(\mathbf{v}) \otimes \mathbf{w} + \mathbf{v} \otimes f_W(\mathbf{w}) \\ &= (f_V \otimes id_W)(\mathbf{v} \otimes \mathbf{w}) + (id_V \otimes f_W)(\mathbf{v} \otimes \mathbf{w}) = (f_V \otimes id_W + id_V \otimes f_W)(\mathbf{v} \otimes \mathbf{w})\end{aligned}$$

Hence, as this formula holds for all elements in the basis $B_V \otimes B_W$, the derivation matrix is the matrix representation of the linear map $(f_V \otimes id_W + id_V \otimes f_W)$, which is precisely:

$$M_V \otimes \mathcal{I}_m + \mathcal{I}_n \otimes M_W.$$

□

This Leibniz rule on the matrix level is also called *Kronecker sum of two matrices* [8].

Proposition 2.6 (Quotient space)**.** *Let $(V, \vec{\partial})$ be a differential vector space over (K, ∂) and consider a subspace $N \subset V$ closed under $\vec{\partial}$. Then:*

(1) There is a unique derivation $\overline{\partial}$ over the quotient space V/N such that $\overline{\partial} \circ \pi = \pi \circ \vec{\partial}$ where π is the canonical projection $\pi : V \to V/N$.

(2) If $\Phi = \{\phi_1, ..., \phi_n\}$ generates V and M is a derivation matrix of $\vec{\partial}$ w.r.t. Φ, then M is a derivation matrix of $\overline{\partial}$ w.r.t. $\pi(G)$.

Proof.

(1) Let $\overline{\partial} : V/N \to V/N$ be defined as $\overline{\partial}(\mathbf{v} + N) = \vec{\partial}(\mathbf{v}) + N$. This function $\overline{\partial}$ is well defined because if $\mathbf{v} + N = \mathbf{w} + N$ then $\mathbf{v} - \mathbf{w} \in N$ and its derivative is again in N so $\vec{\partial}(\mathbf{v}) + N = \vec{\partial}(\mathbf{w}) + N$.

A straightforward computation using the definition of $\overline{\partial}$ and the linearity of π proves that $\overline{\partial}$ is a derivation. Now suppose there is another derivation $\hat{\partial}$ over V/N such that $\hat{\partial} \circ \pi = \pi \circ \vec{\partial}$. Then we have that for any $\mathbf{v} + N \in V/N$,

$$\hat{\partial}(\mathbf{v} + N) = \hat{\partial}(\pi(\mathbf{v})) = \pi(\vec{\partial}(\mathbf{v})) = \vec{\partial}(\mathbf{v}) + N = \overline{\partial}(\mathbf{v} + N),$$

so $\hat{\partial} = \overline{\partial}$.

(2) Let $\mathbf{v} = c_1\phi_1 + ... + c_n\phi_n$. Then, using that $\overline{\partial}(\mathbf{v}+N) = \vec{\partial}(\mathbf{v}) + N$ and the linearity of the canonical projection, we have that, if $(\hat{v}_1, ..., \hat{v}_n)$ is a representation of $\vec{\partial}(\mathbf{v})$ w.r.t. Φ, then it is a representation of $\overline{\partial}(\mathbf{v} + N)$ w.r.t. $\pi(\Phi)$.

As M is a derivation matrix of $\vec{\partial}$ w.r.t. Φ, we have that

$$M \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} + \begin{pmatrix} \partial(v_1) \\ \partial(v_2) \\ \vdots \\ \partial(v_n) \end{pmatrix}$$

is a representation of $\vec{\partial}(\mathbf{v})$ w.r.t Φ and, as we already saw, a representation of $\overline{\partial}(\mathbf{v} + N)$ w.r.t. $\pi(\Phi)$. Hence M is also a derivation matrix of $\overline{\partial}$ w.r.t. $\pi(\Phi)$. □

Lemma 2.7 (Differential linear mappings)**.** *Let $(V, \vec{\partial}_V)$ and $(W, \vec{\partial}_W)$ be two differential vector spaces over (K, ∂) and let $f : V \to W$ be a linear map that commutes with the derivatives of V and W, 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 $f(\vec{\partial}_V(\mathbf{v})) = \vec{\partial}_W(f(\mathbf{v})) = \vec{\partial}_W(0) = 0$, so $\vec{\partial}_V(\mathbf{v}) \in \ker(f)$. □

These four results are tools for computing derivation matrices in more complicated vector spaces starting from simple ones. We use this later when we explain the main method for computing the closure properties for differentially definable functions.

3. Differentially definable functions

Now we recall the concepts we developed in our previous work [10] and we relate those concepts with the derivations we defined in the previous section. From this section on, we fix the following notation: K is a field of characteristic zero, $K[[x]]$ denotes the ring of formal power series over K, ∂ the standard derivation in $K[[x]]$ and $\langle S \rangle_K$ the K-vector space generated by the set S.

Definition 3.1. Let R be a non-trivial differential subring of $K[[x]]$ and $R[\partial]$ the ring of linear differential operators over R. We call $f \in K[[x]]$ *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 $\mathrm{D}(R)$ we denote the set of all $f \in K[[x]]$ that are differentially definable over R. 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 $\mathcal{A} \cdot f = 0$).

Note that $R \subset R[\partial]$ and hence for non-trivial subrings of $K[[x]]$ the set of differentially definable functions is never empty. Classical D-finite functions are in our notation just $\mathrm{D}(K[x])$. It is well known [13] that $\mathrm{D}(K[x])$ is closed under derivation, addition, and multiplication, i.e., they form a differential subring of $K[[x]]$. Hence the set of *DD-finite functions* can be defined as $\mathrm{D}(\mathrm{D}(K[x]))$.

Example 3.2. $f_0(x) = \exp(x) \in K[[x]]$ is D-finite satisfying $f_0'(x) - f_0(x) = 0,\ f_0(0) = 1$, and so is the constant function $f_1(x) = 1 \in K[[x]]$. Hence $g(x) = \exp(\exp(x) - 1) \in K[[x]]$ is DD-finite as solution to $f_1(x)g'(x) - f_0(x)g(x) = 0,\ g(0) = 1$. The coefficients in the defining differential equation for $g(x)$ can be represented in turn using their respective defining (in)homogeneous differential equations.

Example 3.3. Mathieu's equation in its standard form is given by [7, 17]

$$w'' + (a - 2q\cos(2x))w = 0, \tag{1}$$

for some parameters a and q. This differential equation has a pair of fundamental solutions (w_1, w_2) with initial values

$$\begin{aligned} &w_1(0; a, q) = 1,\ w_1'(0; a, q) = 0, \text{ and} \\ &w_2(0; a, q) = 0,\ w_2'(0; a, q) = 1. \end{aligned}$$

$w_1(z; a, q)$ is even and $w_2(z; a, q)$ is odd and both are DD-finite functions. Mathieu functions are related to several problems in applied mathematics and were introduced by Mathieu in the context of vibrating elliptical drumheads [17].

Analogously to D-finite functions, differentially definable functions can be characterized equivalently by an inhomogeneous differential equation or as a finite dimensional vector space.

Theorem 3.4. *Let R be a differential subring of $K[[x]]$, $R[\partial]$ the ring of linear differential operators over R, and $F = Q(R)$ be the field of fractions of R. Let $f \in K[[x]]$. Then the following are equivalent:*

(1) $f \in \mathrm{D}(R)$

(2) $\exists\, \mathcal{A} \in R[\partial]\, \exists\, g \in \mathrm{D}(R) \colon \mathcal{A} \cdot f = g$

(3) $\dim \langle f^{(i)} \mid i \in \mathbb{N} \rangle_F < \infty$

Proof. See [10]. □

It is well known that D-finite functions satisfy various closure properties [13] and that these closure properties can be executed automatically [6, 15, 12]. Many of these closure properties can be carried over to differentially definable functions and also the algorithmic aspect can be kept [10]. We recall some of these closure properties and, for the sake of being self-contained, repeat part of the proof.

Theorem 3.5. *Let R be a non-trivial differential subring of $K[[x]]$ and $f(x), g(x) \in \mathrm{D}(R)$ with orders d_1 and d_2, respectively, and $r(x) \in R$. Then:*

(1) $f'(x) \in \mathrm{D}(R)$ with order at most d_1.

(2) Any antiderivative of $f(x)$ is in $\mathrm{D}(R)$ with order at most $d_1 + 1$.

(3) $f(x) + g(x) \in \mathrm{D}(R)$ with order at most $d_1 + d_2$.

(4) $f(x)g(x) \in \mathrm{D}(R)$ with order at most $d_1 d_2$.

(5) If $r(0) \neq 0$, then its multiplicative inverse $1/r(x)$ in $K[[x]]$ is in $\mathrm{D}(R)$ with order at most 1.

Proof. Given an annihilating operator $\mathcal{A} = r_{d_1}\partial^{d_1} + \cdots + r_1\partial + r_0$ with $\mathcal{A} \cdot f = 0$, we have that $r_d\partial^{d_1-1} + \cdots + r_1$ annihilates f' if $r_0 = 0$. In the case $r_0 \neq 0$, the operator $(r_0\partial - r_0')\mathcal{A}$ has constant coefficient equal to zero, hence is an annihilating operator for f' of order at most d_1. This yields (1). The annihilating operators for (2) and (5) are immediate.

For the addition of two differentially definable functions let F be the field of fractions of R and given $f \in K[[x]]$ define $V_F(f) = \langle f^{(i)} \mid i \in \mathbb{N} \rangle_F$. By Theorem 3.4 we have that $\dim(V_F(f)) = d_1 < \infty$ and $\dim(V_F(g)) = d_2 < \infty$. Since $V_F(f+g) \subset V_F(f) + V_F(g)$, we have

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

This gives (3) and the closure property (4) follows analogously using the tensor product. □

Because of (1), (3), and (4) in Theorem 3.5, given a differential subring R of $K[[x]]$, $\mathrm{D}(R)$ is again a differential subring of $K[[x]]$. Hence the construction can be iterated with closure properties holding at each level. In this sense, we have that D-finite functions are the same as $\mathrm{D}(K[x])$, DD-finite are $\mathrm{D}^2(K[x])$, and we refer to $\mathrm{D}^k(K[x])$ also as D^k-finite functions.

In order to compute the (DD-finite) annihilating operator of the derivative, antiderivative, or multiplicative inverse it suffices to use a precomputed formula. In this case at most closure properties on the coefficient level need to be applied. We illustrate this with the following example.

Example 3.6. Let us return to Mathieu's function and let $w(x)$ be defined by (1), i.e., the annihilating operator is given by $\mathcal{A} = r_0 + r_2\partial^2$ with $r_2(x) = 1$ and $r_0(x) = a - 2q\cos(2x)$. As stated in the proof, we compute

$$(r_0\partial - r_0') \cdot \mathcal{A} = (r_0\partial^2 - r_0'\partial + r_0^2) \cdot \partial.$$

Hence, we have that $w'(x)$ is a solution to

$$\begin{aligned}(a - 2q\cos(2x))y''(x) - 4q\sin(2x)y'(x) \\ +(a - 2q\cos(2x))^2 y(x) = 0.\end{aligned} \tag{2}$$

Note that the coefficient $r_0(x)$ is actually represented in the computer as the D-finite function satisfying

$$y'''(x) + 4y'(x) = 0, \quad y(0) = a - 2q, y'(0) = 0, y''(0) = 8q.$$

Analogously the coefficients in (2) are not given explicitly, but represented in terms of their defining D-finite differential equations plus initial values. These representations are in turn computed using closure properties on the level of D-finite functions.

For computing the closure properties of addition and multiplication of two differentially definable functions, we use the bound for the dimension of the vector space $V_F(f+g)$ and $V_F(fg)$ obtained in Theorem 3.5. The details of the algorithm are in section 4.

Before we turn to this description, let us remark on the relation between differential linear algebra, see Section 2, and differentially definable functions. In the proof of Theorem 3.4 (see [10]) it is shown that if $f \in \mathrm{D}(R)$ is annihilated by an order d operator $\mathcal{A} \in R[\partial]$, then $\{f, \dots f^{(d-1)}\}$ generates $V_F(g)$. Moreover, both F and $V_F(f)$ are subsets of $K((x))$, the Laurent series over K, which has a unique extension of ∂. Both sets are closed under this derivation, which makes (F, ∂) a differential field and $V_F(f)$ a differential vector space over (F, ∂) in the sense defined in Section 2. In fact, we can compute a derivation matrix of ∂ in $V_F(f)$.

Lemma 3.7. *Let $f(x) \in \mathrm{D}(R)$ for some differential subring $R \subset K[[x]]$ and $\mathcal{A} = r_d\partial^d + ... + r_0 \in R[\partial]$ be such that $\mathcal{A} \cdot f = 0$. Then the companion matrix of $\mathcal{A}$*

$$\mathcal{C}_f = \begin{pmatrix} 0 & \dots & 0 & -r_0/r_d \\ 1 & \dots & 0 & -r_1/r_d \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & 1 & -r_{d-1}/r_d \end{pmatrix}.$$

is a derivation matrix for the derivation ∂ over the vector space $V_F(f)$ w.r.t. $(f(x), \dots, f^{(d-1)}(x))$.

Proof. Define the column vectors $\mathbf{v} = (v_i)_{i=0}^{d-1}$ and let $g = v_0 f + \cdots + v_{d-1}f^{(d-1)} \in V_F(f)$. Then

$$\begin{aligned} \vec{\partial} g = &\left(\partial(v_0)f + \cdots + \partial(v_{d-1})f^{(d-1)}\right) \\ &+ \left(v_0 f' + \cdots + v_{d-1}f^{(d)}\right), \end{aligned}$$

and $f^{(d)}$ can be expressed in terms of $f, \dots, f^{(d-1)}$,

$$\begin{aligned} \vec{\partial} g = &\left(\partial(v_0)f - \frac{r_0}{r_d}v_{d-1}\right) f + \left(\partial(v_1) + v_0 - \frac{r_1}{r_d}v_{d-1}\right) f' + \\ &\cdots + \left(\partial(v_{d-1}) + v_{d-2} - \frac{r_{d-1}}{r_d}v_{d-1}\right) f^{(d-1)}. \end{aligned}$$

Thus $\vec{\partial} g = \hat{v}_0 f + \cdots + \hat{v}_{d-1}f^{(d-1)}$ for $\hat{\mathbf{v}} = \mathcal{C}_f\mathbf{v} + \kappa(\mathbf{v})$. □

Given a differentially definable function f of order d, let $\mathbf{f}$ denote the column vector $(f^{(i)})_{i=0}^{d-1}$ and let the derivative act component-wise on the vector. Then for the companion matrix we have that $\kappa\mathbf{f} = \mathcal{C}_f^T\mathbf{f}$ and for g and $\hat{\mathbf{v}}$ as above, $\vec{\partial} g = \mathbf{f}^T\hat{\mathbf{v}}$.

4. Overview of the general method

Now that we have the basic concepts, we give a general structure that the algorithms for addition and multiplication of differentially definable functions have. The main structure of these algorithms is usually called an *ansatz* method.

Let $f(x), g(x)$ be differentially definable functions over R and consider either $h(x) = f(x) + g(x)$ or $h(x) = f(x)g(x)$. In either case the function $h(x)$ is differentially definable over the same fixed differential ring R. In order to prove this, we showed that the vector space $V_F(h) = \langle h, h', ...\rangle_F$ is contained in some vector space $W \subset K[[x]]$ of finite dimension d, getting then an upper bound for the dimension of $V_F(h)$. There must be a non-trivial linear combination of $h(x), h'(x), \dots, h^{(d)}(x)$ equal to zero.

Hence, we build an ansatz for that homogeneous equation where the unknown variables are the coefficients of the linear combination (i.e. elements of F). More precisely, the method proceeds as follows:

(i) Compute a vector of generators $\Phi = (\phi_0, \phi_2, \dots, \phi_{d-1})$ of the vector space W.

(ii) Compute representations of $h^{(i)}$ in terms of Φ, i.e., vectors $\mathbf{v}_i = (v_{i,0}, \dots, v_{i,d-1})$ such that for all $i = 0, ..., n$:

$$h^{(i)} = v_{i,0}\phi_0 + ... + v_{i,d-1}\phi_{d-1}.$$

(iii) Set up an ansatz with $d+1$ variables $\alpha_0, ..., \alpha_d$:

$$\alpha_0 h(x) + ... + \alpha_d h^{(d)}(x) = 0.$$

(iv) Using the expressions of step (ii), compute a matrix $\mathcal{S}$ with coefficients in R such that:

$$\mathcal{S} \begin{pmatrix} \alpha_0 \\ \vdots \\ \alpha_d \end{pmatrix} = 0.$$

(v) Compute a particular element $\hat{\alpha} = (\hat{\alpha}_0, ..., \hat{\alpha}_d)$ in the nullspace of $\mathcal{S}$. Then h is annihilated by:

$$\mathcal{A} = \hat{\alpha}_n \partial^n + \cdots + \hat{\alpha}_0.$$

(vi) Compute as many initial values of h as necessary to characterize a unique power series by $\mathcal{A}$.

Let us analyze the algorithm step by step. In step (i) we compute generators of W. This must be done independently for each operation. Theorem 3.5 contains the key idea how to compute W and the vector of generators $(\phi_0, \ldots, \phi_{d-1})$ for either case.

For step (ii) we compute the vectors $\mathbf{v}_i$. The vector space W together with ∂ is a differential vector space in the sense defined in Section 2. If we had a derivation matrix M of W w.r.t. $(\phi_0, \ldots, \phi_{d-1})$ then we could compute the representation of $h'(x), \ldots, h^{(d)}$ (i.e., the vectors $\mathbf{v}_1, \ldots, \mathbf{v}_d$) once we have the first vector $\mathbf{v}_0$ representing $h(x)$. More precisely, we can compute:

$$\mathbf{v}_{i+1} = M\mathbf{v}_i + \kappa(\mathbf{v}_i). \tag{3}$$

In step (iii) we set up the ansatz with new variables and then in step (iv) we compute the system matrix $\mathcal{S}$. Using the ansatz equation and the vector representation of the derivatives of h, we have that:

$$\begin{aligned} \alpha_0 h(x) + ... + \alpha_d h^{(d)}(x) &= (\phi_0, ..., \phi_{d-1})(\alpha_0 \mathbf{v}_0 + ... + \alpha_n \mathbf{v}_d) \\ &= (\phi_0, \ldots, \phi_{d-1}) \begin{pmatrix} v_{0,0} & v_{1,0} & ... & v_{d,0} \\ v_{0,1} & v_{1,1} & ... & v_{d,1} \\ \vdots & \vdots & \ddots & \vdots \\ v_{0,d-1} & v_{1,d-1} & ... & v_{d,d-1} \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_d \end{pmatrix} = \Phi^T \tilde{\mathcal{S}} \alpha, \end{aligned}$$

where $\tilde{\mathcal{S}} \in F^{d \times (d+1)}$ is built from the column vectors $\mathbf{v}_i$ computed in step (ii). In order to obtain the desired system $\mathcal{S}$ with coefficients in R we only need to clear denominators.

For step (v) we compute a vector in the right-nullspace of $\mathcal{S}$ using any linear algebra algorithm. We are sure that it is not trivial because the dimension of the matrix guarantees at least a nullspace of dimension 1. For a discussion on how many and which initial values need to be computed see [10].

Summarizing, for each particular operation we need to:

- Compute the dimension bound d and generators $\phi_0, ..., \phi_{d-1}$.
- Compute the derivation matrix M.
- Compute the initial vector $\mathbf{v}_0$.
- Compute the required initial values $h(0), h'(0), \ldots$.

Once we have these four details fixed, we have fully described the ansatz algorithm. Next we give the details for both addition and multiplications.

4.1. Addition

Given $f, g \in \mathrm{D}(R)$ and $\mathcal{A}_f, \mathcal{A}_g \in R[\partial]$ of orders d_1 and d_2 such that $\mathcal{A}_f \cdot f = 0 = \mathcal{A}_g \cdot g$, let $h(x) = f(x) + g(x)$. We know (see Theorem 3.5) that h is differentially definable over R and that $V_F(h)$ is contained in the finite dimensional F-vector space $W = V_F(f) + V_F(g)$.

To complete the algorithm we need the following steps:

- **Computing the generators of W:** as it is the sum of two vector spaces, its generators are built from the generators of each of the summands. We have that:

$$V_F(f) = \langle f, ..., f^{(d_1-1)} \rangle_F \quad \text{and}$$
$$V_F(g) = \langle g, ..., g^{(d_2-1)} \rangle_F,$$

  Hence the direct sum of the generators of $V_F(f)$ and $V_F(g)$ (i.e. the union) generates W, obtaining a bound of $d_1 + d_2$ for the dimension of W.

$$\mathbf{f} \oplus \mathbf{g} = (f, ..., f^{(d_1-1)}, g, ..., g^{(d_2-1)}).$$

- **Computing the derivation matrix w.r.t. $\mathbf{f} \oplus \mathbf{g}$:** by Propositions 2.4 and 2.6 and Lemma 2.7, we know that a derivation matrix w.r.t. $\mathbf{f} \oplus \mathbf{g}$ is the direct sum of derivation matrices on $V_F(f)$ and $V_F(g)$. As we saw in Lemma 3.7, those derivation matrices are the companion matrices of $\mathcal{A}_f$ and $\mathcal{A}_g$, respectively. Hence, the derivation matrix M_+ is:

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

- **Computing the initial vector $\mathbf{v}_{0+}$:** as $h = f + g$, a representation of h is the direct sum (i.e., concatenation) of representations of f and g in their generators,

$$\mathbf{v}_{0+}^T = (1, 0, ..., 0, 1, 0, ..., 0) = \mathbf{e}_{d_1,1} \oplus \mathbf{e}_{d_2,1},$$

  where $\mathbf{e}_{d,i}$ is ith unit vector of length d.

- **Computing the initial values of h:** for any $k \in \mathbb{N}$, the kth initial value of h is the sum of the kth initial values of f and g:

$$h^{(k)}(0) = f^{(k)}(0) + g^{(k)}(0),$$

With these details fixed, the addition algorithm is complete. We can easily apply it to compute a precise representation of the sum of two differentially definable functions.

Example 4.1. Let $f(x) = \exp(\sin(x))$ and $g(x) = \tan(x)$. Both are DD-finite functions with annihilating operators:

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

If we want to compute the differential operator that $h(x) = f(x) + g(x)$ satisfies we need to use the companion matrices of $\mathcal{A}_f$ and $\mathcal{A}_g$:

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

Hence, if we put together both matrices to build the matrix M_+ we have:

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

The initial vector is $\mathbf{v}_{0+} = (1, 1, 0)$ and we can compute the whole system using the matrix M_+:

$$\mathbf{v}_{1+} = M_+ \mathbf{v}_{0+} + \kappa(\mathbf{v}_{0+}) = \begin{pmatrix} \cos(x) \\ 0 \\ 1 \end{pmatrix}.$$

If we iterate this procedure and clear denominators, we obtain the system matrix:

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

where we abbreviate $c = \cos(x)$ and $s = \sin(x)$.

Then we can apply any linear algebra algorithm to obtain that $(\alpha_0, \alpha_1, \alpha_2, \alpha_3)^T$ is in the right-nullspace of $\mathcal{S}_+$, where:

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

Now, let $\mathcal{A} = \alpha_3\partial^3 + \alpha_2\partial^2 + \alpha_1\partial + \alpha_0$. We apply the methodology explained in [10] and conclude that any function annihilated by $\mathcal{A}$ is characterized by its first three initial values:

$$\begin{aligned} h(0) =& f(0) + g(0) = 1 + 0 = 1 \\ h'(0) =& f'(0) + g'(0) = 1 + 1 = 2 \\ h''(0) =& f''(0) + g''(0) = 1 + 0 = 1 \end{aligned}$$

So we have that $h(x) = e^{\sin(x)} + \tan(x)$ is the unique formal power series satisfying:

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

4.2. Multiplication

We proceed now similarly as we did in Subsection 4.1. Let $f, g \in \mathrm{D}(R)$, $\mathcal{A}_f, \mathcal{A}_g \in R[\partial]$ of orders d_1 and d_2 such that $\mathcal{A}_f \cdot f = 0 = \mathcal{A}_g \cdot g$, and $h(x) = f(x)g(x)$. We know that h is differentially definable over R and that $V_F(h)$ is contained in the finite dimensional F-vector space $W = V_F(f)V_F(g)$ (see Theorem 3.5).

To complete the algorithm we need the following steps:

- **Computing the generators of W:** now W is the product of $V_F(f)$ and $V_F(g)$. Then, it is generated by the tensor product of the generators of each vector space. Namely:

$$\mathbf{f} \otimes \mathbf{g} = (\begin{matrix} fg, & fg', & ..., & fg^{(d_2-1)}, \\ f'g, & f'g', & ..., & f'g^{(d_2-1)}, \\ \vdots & \vdots & \ddots & \vdots \\ f^{(d_1-1)}g, & f^{(d_1-1)}g', & ..., & f^{(d_1-1)}g^{(d_2-1)}). \end{matrix}$$

- **Computing the derivation matrix w.r.t. $\mathbf{f} \otimes \mathbf{g}$:** using the results from Propositions 2.5 and 2.6 and Lemma 2.7, a derivation matrix w.r.t. $\mathbf{f} \otimes \mathbf{g}$ is the Kronecker sum of derivation matrices on $V_F(f)$ and $V_F(g)$. Again, by Lemma 3.7, those derivation matrices are the companion matrices of $\mathcal{A}_f$ and $\mathcal{A}_g$. Hence, the derivation matrix M_* is:

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

- **Computing the initial vector $\mathbf{v}_{0*}$:** now $h = fg$, then a representation of h is the tensor product of the representations of f, g. Namely:

$$\mathbf{v}_{0*} = (1, 0, ..., 0)^T = \mathbf{e}_{d_1,1} \otimes \mathbf{e}_{d_2,1}.$$

- **Computing the initial values of h:** if we apply the iterated Leibniz rule to $h(x) = f(x)g(x)$ we get:

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

for any $k \in \mathbb{N}$, so the kth initial value of $h(x)$ depends on the first k initial values of $f(x)$ and $g(x)$ and can be computed using the formula above.

We have adapted the ansatz algorithm, and now the multiplication algorithm is complete. We can use it to compute a precise representation of the product of two differentially definable functions. Let us see how it works over an example:

Example 4.2. Let $f(x) = \cos(x)$ and $g(x) = \tan(x)$. Both are DD-finite functions with the following annihilating operators:

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

If we want to compute the differential operator that $h(x) = \cos(x)\tan(x)$ satisfies we need to use the companion matrices of $\mathcal{A}_f$ and $\mathcal{A}_g$:

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

Hence, if we compute the Kronecker sum of both matrices to build the matrix M_* we have:

$$M_* = \begin{pmatrix} -\frac{\sin(x)}{\cos(x)} & \frac{1}{\cos(x)^2} \\ 1 & -\frac{\sin(x)}{\cos(x)} \end{pmatrix}.$$

In this case, the initial vector is $\mathbf{v}_0 = (1, 0, 0)$ and we can compute the whole system using the matrix M_*:

$$\mathbf{v}_{1*} = M_* \mathbf{v}_{0*} + \kappa_\partial(\mathbf{v}_{0*}) = \begin{pmatrix} -\frac{\sin(x)}{\cos(x)} \\ 1 \end{pmatrix}.$$

If we iterate this procedure and clear denominators, we obtain the system matrix:

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

Then we can apply any linear algebra algorithm to obtain that the $(\alpha_0, \alpha_1, \alpha_2)^T$ is in the right-nullspace of $\mathcal{S}_*$, where:

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

Now, let $\mathcal{A} = \alpha_2\partial^2 + \alpha_1\partial + \alpha_0$. We apply the methodology explained in [10] and conclude that any function annihilated by $\mathcal{A}$ is characterized by its first two initial values:

$$\begin{aligned} h(0) &= f(0)g(0) = 0 \\ h'(0) &= f'(0)g(0) + f(0)g'(0) = 0 + 1 = 1 \end{aligned}$$

So we have that $h(x) = \cos(x)\tan(x)$ is the unique formal power series satisfying:

$$\begin{cases} \cos(x)\,h''(x) + 2\sin(x)\,h'(x) - \cos(x)\,h(x) = 0, \\ h(0) = 0, \quad h'(0) = 1. \end{cases}$$

We can easily check that $\sin(x)$ satisfies those properties, proving (with this automated approach) that $\frac{\sin(x)}{\cos(x)} = \tan(x)$.

In the examples through this section we simply carried out all computations with the actual coefficient functions, without taking into account any relations between sine and cosine that might help to keep the coefficients smaller. In the general case, the coefficient functions are given in terms of their defining equations over some differential subring R. This means in particular that this construction may be nested. All operations for the computation of the nullspace can be carried out division free using closure properties of the underlying subring. These computations quickly become very heavy. In the next section, we describe how our current implementation handles these difficulties.

5. Operations on the coefficients

For executing the closure properties of addition and multiplication as described in the previous section, all operations need to be carried out in the differential subring R. E.g., for setting up the system matrix $\mathcal{S}$, addition, multiplication, and derivation of elements in R need to be computed. Finally, solving the system requires more arithmetic operations.

In the case of D-finite functions (where $R = K[x]$), these operations can be carried out fast in virtually all available computer algebra systems. However, if we move on to DD-finite functions (or an even deeper layer of D^k-finite functions), addition, multiplication, and derivation of elements in R have to be computed recursively using closure properties. If directly implemented in this recursive way, the calculations quickly come to a halt because of memory consumption. Every execution of a closure property potentially increases the order of the given recurrence and the size of its coefficients.

We have implemented all algorithms described in this paper in the mathematical software system SAGE [21] in our package dd_functions [9]. In this section we describe how the actual implementation currently handles the closure properties addition and multiplication (recall that the others merely require plugging into precomputed formulas). As mentioned above, there are two major steps where computations on the coefficient-level are needed: the construction of the system matrix $\mathcal{S}$ and the computation of the nullspace.

5.1. Obtaining the linear system

In either case (addition or multiplication) we have two functions $f(x), g(x) \in \mathrm{D}(R)$ that are defined with the following differential operators:

$$\mathcal{A}_f = r_n\partial^n + ... + r_0 \quad \text{and} \quad \mathcal{A}_g = s_m\partial^m + ... + s_0$$

of orders n and m respectively. During computations any non-trivial denominator in the companion matrices $\mathcal{C}_f$ and $\mathcal{C}_g$ (and thus in the derivation matrix M) can only be a product of powers of the leading coefficients r_n and s_m. Then, while computing the vectors $\mathbf{v}_i$ using (3), if the coefficients of $\mathbf{v}_i$ have a denominator bound $r_n^p s_m^q$ then $r_n^{p+1} s_m^{q+1}$ is a denominator bound for $\mathbf{v}_{i+1}$. Hence, we can clear denominators at the end of the iteration using this (possibly coarse) denominator bound.

At this stage of setting up the system matrix $\mathcal{S}$, the coefficients of $\mathcal{A}_f$ and $\mathcal{A}_g$ and their derivatives are kept as indeterminates $r_i, s_j, r_k^{(\gamma)}, s_l^{(\delta)}$. In the current implementation, those variables are added to the system on the fly. This means that we start with a polynomial ring $K[r_0, \ldots, r_n, s_0, \ldots, s_m]$ and when we compute the derivatives, we add more variables following the rules $\partial(r_k^{(\gamma)}) = r_k^{(\gamma+1)}$ and $\partial(s_l^{(\delta)}) = s_l^{(\delta+1)}$.

5.2. Solving the linear system

At this stage, we have a matrix $\mathcal{S}$ with variables $r_i, s_j, r_k^{(\gamma)}, s_l^{(\delta)}$. In order to get the nullspace of $\mathcal{S}$ we keep this polynomial behavior and follow the division-free algorithm described by Bareiss [4]. During this algorithm, some pivots have to be chosen. The condition for an element to be a pivot is that it is not zero. Here, we have polynomials on the variables $r_i, s_j, r_k^{(\gamma)}, s_l^{(\delta)}$ that can be non-zero in the polynomial sense but, since this variables represents precise functions, it may be that the functional equivalent to the polynomial is zero.

Thus, we modify Bareiss' algorithm such that each time a pivot is going to be picked, we check that the equivalent power series is not zero. This zero testing must be done using the operations on the differential ring R and, if necessary, *applying recursively* these closure properties.

A defining differential equation for h can be obtained from any element in the nullspace. For various reasons one may wish to have an operator of low order. Hence, we compute a normal basis to obtain an appropriate vector in the nullspace [4, 8].

When returning the annihilating operator for $h = f + g$ or $h = fg$, the elements of the nullspace have to be given explicitly *applying recursively* the closure properties.

5.3. Managing the variables

The main issue in this approach is the number of variables the algorithm must handle. As the performance of polynomials algorithms usually depends on the number of variables, we would like to exploit the relations between our variables and keep their number as low as possible without increasing the computational cost.

We present here two means we apply in our implementation after the system $\mathcal{S}$ is set up:

- *Reducing linear relations:* after building the system matrix $\mathcal{S}$ we have a list of variables representing $r_i, s_j, r_k^{(\gamma)}$, and $s_l^{(\delta)}$. These coefficient functions in turn are given by defining differential equations plus initial values. From the initial values it is cheap to check pairwise if there are possibly linear relations of the form $Y = cX + d$ among them for some constants c, d.

  Suppose we pick two of the variables X and Y and they satisfy the relation $Y = cX + d$. Then, if we consider $\hat{Y} = Y - Y(0)$ and $\hat{X} = X - X(0)$, we have that $p = \mathrm{ord}_x(\hat{Y}) = \mathrm{ord}_x(\hat{X})$ and

  $$Y^{(p)}(0) = \hat{Y}^{(p)}(0) = c\hat{X}^{(p)}(0) = cX^{(p)}(0).$$

  On the other hand, it is clear that $d = Y(0) - cX(0)$.

  Then, we can compute the unique pair of candidates c, d and check if the equality holds *using closure properties in R*. Checking this type of identities for all the variables we have has the same computational cost as checking only for repeated variables.

- *Reducing algebraic relations:* this is done when choosing a new pivot element. At this step closure properties need to be applied in order to verify that we are not computing with a zero pivot. Else, we have found a non-trivial algebraic relation between the variables.

  Every time a new relation is found, it is added to a set of known relations that is used to simplify the system using Gröbner basis computations [24]. This set is kept during a session of computations in SAGE and the relations are also used if they appear in another problem. If there is a priori knowledge about algebraic relations between the coefficients, this information may be added by the user to improve the performance.

6. A complete example. Why reductions are useful.

Now, we present a complete example for the method proposed in this document through a particular example with DD-finite functions. We consider two functions $f(x), g(x)$ that are defined with the following differential equations:

$$\begin{cases} f''(x) + b(x)f'(x) + a(x)f(x) = 0, \\ f(0) = 1, \quad f'(0) = 0, \end{cases}$$

$$\begin{cases} g''(x) + g'(x) = 0, \\ g(0) = 0, \quad g'(0) = 1. \end{cases}$$

We see that $g(x)$ is a D-finite function (so in particular is DD-finite) and $f(x)$ depends directly on two coefficients $a(x)$ and $b(x)$. Let them be defined as follows:

$$\begin{cases} a'(x) - a(x) = 0, \\ a(0) = 1, \end{cases}$$

$$\begin{cases} b'''(x) - 3b''(x) + 2b'(x) = 0, \\ b(0) = 1, \quad b'(0) = -1, \quad b''(0) = -3. \end{cases}$$

So both are D-finite functions making $f(x)$ DD-finite. Let $h(x) = f(x) + g(x)$. We want to compute the data-structure that defines $h(x)$, so we first compute an operator $\mathcal{A}$ that annihilates $h(x)$ and then we compute initial values of $h(x)$ to represent it as a particular solution of the differential equation given by $\mathcal{A}$. For doing so, we apply step by step the method analyzed in this paper.

6.1. Computing the linear system

Following the steps described in section 4, we first need to compute the derivation matrix for the space $V_F(f) + V_F(g)$ and, as we saw in subsection 4.1, we need to compute the matrix:

$$M_+ = \mathcal{C}_f \oplus \mathcal{C}_g.$$

The companion matrices for the defining differential equations for $f(x)$ and $g(x)$ are

$$\mathcal{C}_f = \begin{pmatrix} 0 & -a \\ 1 & -b \end{pmatrix}, \qquad \mathcal{C}_g = \begin{pmatrix} 0 & 0 \\ 1 & -1 \end{pmatrix}.$$

Now that we have the companion matrices we need to compute the vectors $\mathbf{v}_i$ for $i = 0, \ldots, 4$ following the formula

$$\mathbf{v}_{i+1} = M_+ \mathbf{v}_i + \kappa(\mathbf{v}_i),$$

where we have $\mathbf{v}_0 = \mathbf{e}_{2,1} \oplus \mathbf{e}_{2,1} = (1, 0, 1, 0)$.

We perform those computations considering a, b and their derivatives as variables (as we said in subsection 5.1), obtaining the following system to solve:

$$\mathcal{S} = \begin{pmatrix} 1 & 0 & -a & ab - a' & -ab^2 + 2ab' + a^2 + a'b - a'' \\ 0 & 1 & -b & b^2 - b' - a & -b^3 + 3bb' - 2ba - 2a' - b'' \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 1 & -1 \end{pmatrix}.$$

6.2. Solving the linear system

For solving the system $\mathcal{S}$ and computing its nullspace, we will follow the steps described in subsection 5.2, but using the optimizations described in subsection 5.3. First of all, we need to translate the system $\mathcal{S}$ to a polynomial system choosing the appropriate variables. While computing $\mathcal{S}$ we realized that at most $a''(x)$ and $b''(x)$ appear. After looking for linear relations between the 6 initial variables (a, a', a'', b, b', b''), we find the relations $a = a' = a''$ (as can be read directly from the defining differential equation). Hence we have a total amount of 4 variables:

$$x_1 \leftarrow a(x), \quad x_2 \leftarrow b(x), \quad x_3 \leftarrow b'(x), \quad x_4 \leftarrow b''(x),$$

and we can compute the polynomial matrix $\hat{\mathcal{S}}$:

$$\hat{\mathcal{S}} = \begin{pmatrix} 1 & 0 & -x_1 & x_1x_2 - x_1 & -x_1x_2^2 + x_1^2 + x_1x_0 + 2x_1x_3 - x_1 \\ 0 & 1 & -x_2 & x_2^2 - x_1 - x_3 & -x_0^3 + 2x_1x_2 + 3x_2x_3 - x_4 - 2x_1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 1 & -1 \end{pmatrix}.$$

Now we apply the modified Bareiss algorithm to get a normal form of $\hat{\mathcal{S}}$ and compute the nullspace. During this algorithm we need to perform some zero-recognition steps for the polynomials:

- First iteration: we check 1. It is clearly not zero. We go on with the usual algorithm.
- Second iteration: we check 1. Again the algorithm goes on as usual.
- Third iteration: we check x_1. As $a(x) \neq 0$, the algorithm may go on as usual.
- Fourth iteration: we check the polynomial $x_3x_1 + x_1^2 - 2x_1x_2 + 2x_1$. We need to perform the closure properties in the D-finite level, obtaining a function defined as

$$\begin{cases} \alpha'''(x) - 6\alpha''(x) + 11\alpha'(x) - 6\alpha(x) = 0, \\ \alpha(0) = 0, \ \alpha'(0) = 0, \ \alpha''(0) = 0 \end{cases}$$

  which is equal to zero (as all the initial values are zero). Hence we found an algebraic relation and we reduce the matrix with it.

We keep looking for a pivot in the last position left, and now we need to check the polynomial $3x_1^2 - 4x_1x_2 + x_1x_4 + 4x_1$. Again, we have to apply the closure properties in the D-finite level obtaining a function defined by:

$$\begin{cases} \beta'''(x) - 6\beta''(x) + 11\beta'(x) - 6\beta(x) = 0, \\ \beta(0) = 0,\ \beta'(0) = 0,\ \beta''(0) = 0 \end{cases}$$

This is again zero, so a new relation was found.

This is the last step of the algorithm, because no more elements can be picked as pivots. Then we already have the normal form

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -x_2+2 & x_2^2-3x_2-x_3+2 \\ 0 & 0 & 1 & -x_2+1 & x_2^2-3x_2-x_3+3 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

We compute there the nullspace obtaining the following generating vectors:

$$(0, x_2 - 2, x_2 - 1, 1, 0),$$

$$(0, -x_2^2 + 3x_2 + x_3 - 2, -x_2^2 + 3x_2 + x_3 - 3, 0, 1).$$

The lower order operator we can get arises from the first generator, so the final differential operator is:

$$\mathcal{A} = \partial^3 + (b(x) - 1)\partial^2 + (b(x) - 2)\partial,$$

which needs the first three initial values to determine a unique solution. Hence, the function $h(x)$ is defined by:

$$\begin{cases} h'''(x) + (b(x) - 1)h''(x) + (b(x) - 2)h'(x) = 0, \\ h(0) = 1, h'(0) = 1, h''(0) = -2 \end{cases}$$

In this last representation, the coefficients are computed using closure properties on the D-finite level, so each of them is represented (as always) by a differential equation and initial values. The initial values for $h(x)$ are computed using the defining structures for $f(x)$ and $g(x)$.

7. A second example. The Mathieu's Wronskian

In this section we present some computations to prove a property of Mathieu functions. Recall (see Example 3.3) that these functions are defined as solutions of the differential equation

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

where $a, q \in \mathbb{Q}$.

It is well known that Mathieu functions have a closed form only when $q = 0$. Then we have that any solution of the equation is of the form

$$w_{A,B}(x) = A\cos\left(\sqrt{a}x\right) + B\sin\left(\sqrt{a}x,\right)$$

for some constants $A, B \in \mathbb{Q}$. For the other case, $q \neq 0$, no closed form can be computed.

In this section, we are going to prove the following statement:

Theorem 7.1. *Let $w_1(x; a, q)$ and $w_2(x; a, q)$ be two solutions to the Mathieu differential equation with parameters a, q and initial values*

$$w_1(0; a, q) = 1, \qquad w_1'(0; a, q) = 0,$$

$$w_2(0; a, q) = 0, \qquad w_2'(0; a, q) = 1.$$

Then the Wronskian of w_1 and w_2 is exactly one, i.e.:

$$w_1(x; a, q)w_2'(x; a, q) - w_1'(x; a, q)w_2(x; a, q) = 1.$$

In the case when $q = 0$, the functions w_1 and w_2 have the simple closed form

$$w_1(x; a, 0) = \cos(\sqrt{a}x), \qquad w_2(x; a, 0) = \frac{1}{\sqrt{a}} \sin(\sqrt{a}x),$$

and the result is just the well-known identity of Pytagoras. For the case $q \neq 0$, we are going to apply closure properties of DD-finite functions to verify that the Wronskian is identically one.

7.1. Representing the elements of the equality

As we can see in the Mathieu equation, we have that for any choice of a, q, the solutions w_1 and w_2 are DD-finite functions. Hence we can represent them using the following structures:

$$\begin{cases} w_1''(x) + \alpha_{a,q} w_1(x) = 0, \\ w_1(0) = 1, \quad w_1'(0) = 0 \end{cases}$$

$$\begin{cases} w_2''(x) + \alpha_{a,q} w_2(x) = 0, \\ w_2(0) = 0, \quad w_2'(0) = 1 \end{cases}$$

where the element $\alpha_{a,q}$, as D-finite function, is defined by the following differential equation:

$$\begin{cases} \alpha_{a,q}'''(x) + 4\alpha_{a,q}'(x) = 0, \\ \alpha_{a,q}(0) = a - 2q, \quad \alpha_{a,q}'(0) = 0, \quad \alpha_{a,q}''(0) = 8q. \end{cases}$$

Now, we apply several closure properties in the DD-finite level to get, in the end, that the desired Wronskian is equal to 1. The first step is compute the derivative of $w_1(x)$ and $w_2(x)$ using the method described in section 3. We already did that in example 3.6 obtaining:

$$\alpha_{a,q}(x)(w')''(x) - \alpha_{a,q}'(x)(w')'(x) + \alpha_{a,q}^2(x)(w')(x) = 0$$

As long as $a \neq 2q$, we have that the leading coefficient of the latter equation does not vanish when $x = 0$. Hence we can define the derivatives of w_1 and w_2 with 2 initial values, namely:

$$\begin{array}{ll} w_1'(0) = 0, & w_1''(0) = 2q - a, \\ w_2'(0) = 1, & w_2''(0) = 0. \end{array}$$

Otherwise, if we study the equation using the methods described in [10], we get that 4 initial values are needed, so $w_1'(x; 2q, q)$ and $w_2'(x; 2q, q)$ can be defined with the initial values:

$$\begin{array}{llll} w_1'(0) = 0, & w_1''(0) = 0, & w_1'''(0) = 0, & w_1^{(4)}(0) = -8q. \\ w_2'(0) = 1, & w_2''(0) = 0, & w_2'''(0) = 0, & w_2^{(4)}(0) = 0 \end{array}$$

Now, we have to compute the following products: $w_1(x)w_2'(x)$ and $w_1'(x)w_2(x)$. As in both cases we have the same defining differential equations, we will obtain the same differential equation when we apply the closure properties. Hence, we do not need to compute a new differential equation for the difference of those functions, and only some initial values must be computed to obtain the representation of the Wronskian.

In the following subsections, we compute the differential operator for those products and the final initial values of the Wronskian.

7.2. Getting the linear system

We start by computing the companion matrices for the differential operators of $w(x)$ and $w'(x)$:

$$\mathcal{C}_w = \begin{pmatrix} 0 & -\alpha_{a,q} \\ 1 & 0 \end{pmatrix}, \quad \mathcal{C}_{w'} = \begin{pmatrix} 0 & -\alpha_{a,q} \\ 1 & \frac{\alpha_{a,q}'}{\alpha_{a,q}} \end{pmatrix}$$

Then, we compute the Kronecker sum of those matrices, so we have the derivation matrix in the product space

$$M = \mathcal{C}_w \otimes \mathcal{I}_2 + \mathcal{I}_2 \otimes \mathcal{C}_{w'}.$$

Now, starting with $\mathbf{v}_0 = (1,0,0,0)^T$ and using the derivation rule described in section 4 we compute the system:

$$\mathcal{S} = \begin{pmatrix} \alpha_{a,q} & 0 & -2\alpha_{a,q}^2 & -3\alpha'_{a,q}\alpha_{a,q} & 8\alpha_{a,q}^3 - 4\alpha''_{a,q}\alpha_{a,q} \\ 0 & 1 & 0 & -4\alpha_{a,q} & -10\alpha'_{a,q} \\ 0 & \alpha_{a,q}^2 & \alpha'_{a,q}\alpha_{a,q} & \alpha''_{a,q}\alpha_{a,q} - 4\alpha_{a,q}^3 & -14\alpha'_{a,q}\alpha_{a,q}^2 - \alpha''_{a,q}\alpha'_{a,q} + \alpha'''_{a,q}\alpha_{a,q} + \alpha''_{a,q}\alpha_{a,q} \\ 0 & 0 & 2\alpha_{a,q} & 3\alpha'_{a,q} & -8\alpha_{a,q}^3 + 4\alpha''_{a,q}\alpha_{a,q} \end{pmatrix}$$

7.3. Solving the linear system

Next we have to set up some variables to change the system S into a polynomial matrix where we can apply Bareiss' algorithm. We see that only $\alpha_{a,q}$, $\alpha'_{a,q}$, $\alpha''_{a,q}$ and $\alpha'''_{a,q}$ appear in the system. After looking for linear relations between these functions, it turns out that we find two particular relations:

$$\alpha''_{a,q}(x) = -4\alpha_{a,q} + 4a, \qquad \alpha'''_{a,q} = -4\alpha'_{a,q},$$

se we end with two variables:

$$x_0 \leftarrow \alpha_{a,q}(x), \qquad x_1 \leftarrow \alpha'_{a,q}(x).$$

Using these relations and new variables in $\mathcal{S}$ we can compute the final polynomial system obtaining:

$$\hat{\mathcal{S}} = \begin{pmatrix} 1 & 0 & -2x_0 & -3x_1 & 8x_0^2 + 16x_0 - 16a \\ 0 & x_0 & x_1 & -4x_0^2 - 4x_0 + 4a & -14x_0x_1 - 4x_1 \\ 0 & 1 & 0 & -4x_0 & 10x_1 \\ 0 & 0 & 2x_0 & 3x_1 & -8x_0^2 - 16x_0 + 16a \end{pmatrix}$$

Now we apply the modified Bareiss algorithm to get a normal form of $\hat{\mathcal{S}}$ and compute the nullspace. During this algorithm we need to perform some zero-recognition steps for the polynomials:

- First iteration: we check 1. It is clearly not zero. We go on with the usual algorithm.
- Second iteration: we check x_0. Because $q \neq 0$, so is $\alpha_{a,q}(x)$. The algorithm goes on as usual.
- Third iteration: we check $-x_1$. Again, $\alpha'_{a,q}(x) \neq 0$ because $q \neq 0$, so the algorithm may go on as usual.
- Fourth iteration: we check the polynomial $-3x_1^2 - 8x_0^2 + 8x_0a$. We apply here closure properties in the D-finite level to check that this is, indeed, not zero for any choice of a and q where $q \neq 0$. Hence, the algorithm goes on as usual.

Now that we have the normal form computed we can get a generator of the nullspace, obtaining a vector of the form:

$$(0, \beta_1(x_0,x_1), \beta_2(x_0,x_1), \beta_3(x_0,x_1), -3x_1^2 - 8x_0^2 + 8x_0a),$$

where $\beta_1, \beta_2, \beta_3$ are some polynomials (not important for this proof). Hence the differential operator

$$\mathcal{A} = (-3(\alpha'_{a,q})^2 - 8\alpha_{a,q}^2 + 8\alpha_{a,q}a)\partial^4 + \beta_3(\alpha_{a,q}, \alpha'_{a,q})\partial^3 + \beta_2(\alpha_{a,q}, \alpha'_{a,q})\partial^2 + \beta_1(\alpha_{a,q}, \alpha'_{a,q})\partial,$$

annihilates both $w_1(x)w_2'(x)$ and $w_1'(x)w_2(x)$.

7.4. Finishing the proof

Now we have to compute initial values of the products $w_1(x)w_2'(x)$ and $w_1'(x)w_2(x)$ in order to have the representation for W using the operator $\mathcal{A}$. If we look the leading coefficient of $\mathcal{A}$, we see that when $x = 0$ it evaluates to $16q(a - 2q)$ which is zero *if and only if* $a = 2q$. Suppose that is not the case, i.e., $a \neq 2q$. Then we need exactly 4 initial values to define a function using $\mathcal{A}$. Otherwise, when $a = 2q$, we apply the methodology described in our previous work [10] and we realize that we need only the first 3 initial values.

Using the iterated Leibniz rule to get the initial values of the products, as shown in subsection 4.2, we obtain the following initial values:

$f(x)$	$f(0)$	$f'(0)$	$f''(0)$	$f'''(0)$
$w_1(x)w_2'(x)$	1	0	$-2(a-2q)$	0
$w_1'(x)w_2(x)$	0	0	$-2(a-2q)$	0

Hence, the final Wronskian $W(x;a,q)$ is defined by:

$$\begin{cases} \mathcal{A}\cdot(W(x;a,q))=0, \\ W(0;a,q)=1, \quad W'(0;a,q)=0, \quad W''(0;a,q)=0, \quad W'''(0;a,q)=0 \end{cases}$$

or, when $a=2q$:

$$\begin{cases} \mathcal{A}\cdot(W(x;2q,q))=0, \\ W(0;2q,q)=1, \quad W'(0;2q,q)=0, \quad W''(0;2q,q)=0 \end{cases}$$

If we look at the operator $\mathcal{A}$ we can wee that ∂ can be factored to the right, obtaining then that any constant is annihilated by $\mathcal{A}$ (in particular, $\mathcal{A}\cdot 1=0$). On the other hand, we have that the function $f(x)=1$ has the same 4 first initial values as W (or the 3 first initial values in the case $a=2q$), and both are annihilated by $\mathcal{A}$. Hence $W(x;a,q)=1$, finishing the proof.

8. Conclusions

We have introduced a computable extension of D-finite functions to differentially definable functions. Starting from any differential subring of the formal power series, this construction can be iterated. For these classes of functions essentially the same closure properties hold as for classical D-finite functions. With the example of Mathieu's functions, we have illustrated how the closure properties can be used to prove identities of DD-finite functions. The results presented here have been implemented in the open source mathematical software SAGE [21] and can be freely accessed from the RISC webpage [9].

The algorithms described are valid for any differential subring $R\subset K[[x]]$. But some considerations were made in the particular case we have a recursive structure (i.e., $R=\mathrm{D}(S)$). When that is not the case, we directly rely on the implementation of the arithmetic over R.

The key in the algorithms for the closure properties of addition and multiplication is the bound for the dimension. Hence it can be easily adapted to other closure properties going along the same lines. The composition with formal power series or algebraic functions seems natural extensions to be studied following this methodology.

For future work it will be interesting to study analogues on the sequence level. This concerns on the one hand defining the equivalent construction extending P-finite sequences (sequences satisfying linear recurrences with polynomial coefficients). On the other hand the structure of coefficient sequences in the formal power series expansion $\sum_{n\geq 0} f_n x^n$ of DD-finite functions deserves further study. For D-finite functions, closure properties are often used in a combined guess-and-prove approach. Also for DD-finite functions, it would be interesting to have a guessing routine at hand. For this it might be better to start first with guessing on the level of coefficient sequences. Last, but not least, further improvements of the code are required to cover bigger examples and also include more parameters.

[1] S. Abramov, M. Barkatou, D. Khmelnov, On full rank differential systems with power series coefficients, J. Symbolic Computation 68 (2015) 120–137.

[2] S. Abramov, D. Khmelnov, Regular solutions of linear differential systems with power series coefficients, Programming and Computer Software 40 (2) (2014) 98–106.

[3] G. Andrews, R. Askey, R. Roy, Special Functions, Encyclopedia of Mathematics and its Applications, Cambridge University Press, 1999.

[4] E. Bareiss, Sylvester's identity and multistep integer-preserving gaussian elimination, Mathematics of Computation 22 (1967) 565–578.

[5] M. Bronstein, Symbolic Integration I, Algorithms and Computation in Mathematics, 1st ed., Springer, 1997.

[6] F. Chyzak, Gröbner bases, symbolic summation and symbolic integration, in: Gröbner bases and applications (Linz, 1998), vol. 251 of London Math. Soc. Lecture Note Ser., Cambridge Univ. Press, Cambridge, 1998, pp. 32–60.

[7] *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.

[8] R. Horn, C. Johnson, Matrix Analysis, Cambridge University Press, New York, NY, USA, 1986.

[9] A. Jiménez-Pastor, dd_functions: SAGE package, http://www.risc.jku.at/research/combinat/software/.

[10] A. Jiménez-Pastor, V. Pillwein, A computable extension for holonomic functions: DD-finite functions, Tech. Rep. 2017-10, DK Computational Mathematics (12 2017).

[11] M. Kauers, The Holonomic Toolkit, in: J. Blümlein, C. Schneider (eds.), Computer Algebra in Quantum Field Theory: Integration, Summation and Special Functions, Springer, 2013, pp. 119–144.

[12] M. Kauers, M. Jaroschek, F. Johansson, Ore Polynomials in Sage, in: J. Gutierrez, J. Schicho, M. Weimann (eds.), Computer Algebra and Polynomials, Lecture Notes in Computer Science, 2014.

[13] M. Kauers, P. Paule, The Concrete Tetrahedron: Symbolic Sums, Recurrence Equations, Generating Functions, Asymptotic Estimates, 1st ed., Springer Publishing Company, Incorporated, 2011.

[14] E. Kolchin, Differential algebra and algebraic groups, Academic Press New York, 1973.

[15] C. Koutschan, Advanced Applications of the Holonomic Systems Approach, Ph.D. thesis, RISC-Linz, Johannes Kepler University (September 2009).

[16] C. Mallinger, Algorithmic Manipulations and Transformations of Univariate Holonomic Functions and Sequences, Master's thesis, RISC, J. Kepler University (August 1996).

[17] N. W. McLachlan, Theory and application of Mathieu functions, Dover Publications, Inc., New York, 1964.

[18] E. Rainville, Special Functions, 1st ed., Chelsea Publishing Co., Bronx, N.Y., 1971.

[19] R. Stanley, Differentiably finite power series, European Journal of Combinatorics 1 (2) (1980) 175–188.

[20] R. Stanley, Enumerative Combinatorics, vol. 2, Cambridge University Press, Cambridge, 1999.

[21] W. Stein, et al., Sage Mathematics Software (Version 8.1), The Sage Development Team, `http://www.sagemath.org` (2017).

[22] M. van der Put, M. Singer, Galois Theory of Linear Differential Equations, Grundlehren der mathematischen Wissenschaften, Springer Berlin Heidelberg, 2003.

[23] M. van Hoeij, Formal solutions and factorization of differential operators with power series coefficients, J. Symbolic Comput. 24 (1) (1997) 1–30.

[24] F. Winkler, Polynomial Algorithms in Computer Algebra, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1996.

Technical Reports of the Doctoral Program "Computational Mathematics"

2018

2018-01 D. Dominici: *Laguerre-Freud equations for Generalized Hahn polynomials of type I* Jan 2018. Eds.: P. Paule, M. Kauers

2018-02 C. Hofer, U. Langer, M. Neumüller: *Robust Preconditioning for Space-Time Isogeometric Analysis of Parabolic Evolution Problems* Feb 2018. Eds.: U. Langer, B. Jüttler

2018-03 A. Jiménez-Pastor, V. Pillwein: *Algorithmic Arithmetics with DD-Finite Functions* Feb 2018. Eds.: P. Paule, M. Kauers

2017

2017-01 E. Buckwar, A. Thalhammer: *Importance Sampling Techniques for Stochastic Partial Differential Equations* January 2017. Eds.: U. Langer, R. Ramlau

2017-02 C. Hofer, I. Toulopoulos: *Discontinuous Galerkin Isogeometric Analysis for parametrizations with overlapping regions* June 2017. Eds.: U. Langer, V. Pillwein

2017-03 C. Hofer, S. Takacs: *Inexact Dual-Primal Isogeometric Tearing and Interconnecting Methods* June 2017. Eds.: B. Jüttler, V. Pillwein

2017-04 M. Neumüller, A. Thalhammer: *Combining Space-Time Multigrid Techniques with Multilevel Monte Carlo Methods for SDEs* June 2017. Eds.: U. Langer, E. Buckwar

2017-05 C. Hofer, U. Langer, M. Neumüller: *Time-Multipatch Discontinuous Galerkin Space-Time Isogeometric Analysis of Parabolic Evolution Problems* August 2017. Eds.: V. Pillwein, B. Jüttler

2017-06 M. Neumüller, A. Thalhammer: *A Fully Parallelizable Space-Time Multilevel Monte Carlo Method for Stochastic Differential Equations with Additive Noise* September 2017. Eds.: U. Langer, E. Buckwar

2017-07 A. Schafelner: *Space-time Finite Element Methods for Parabolic Initial-Boundary Problems with Variable Coefficients* September 2017. Eds.: U. Langer, B. Jüttler

2017-08 R. Wagner, C. Hofer, R. Ramlau: *Point Spread Function Reconstruction for Single-Conjugate Adaptive Optics* December 2017. Eds.: U. Langer, V. Pillwein

2017-09 M. Hauer, B. Jüttler, J. Schicho: *Projective and Affine Symmetries and Equivalences of Rational and Polynomial Surfaces* December 2017. Eds.: U. Langer, P. Paule

2017-10 A. Jiménez-Pastor, V. Pillwein: *A Computable Extension for Holonomic Functions: DD-Finite Functions* December 2017. Eds.: P. Paule, M. Kauers

2017-11 D. Dominici: *Mehler-Heine Type Formulas for Charlier and Meixner Polynomials II. Higher Order Terms* December 2017. Eds.: P. Paule, M. Kauers

2017-12 D. Dominici: *Orthogonality of the Dickson Polynomials of the* $(k+1)$*-th Kind* December 2017. Eds.: P. Paule, M. Kauers

The complete list since 2009 can be found at
`https://www.dk-compmath.jku.at/publications/`

Doctoral Program
"Computational Mathematics"

Director:

Dr. Veronika Pillwein
Research Institute for Symbolic Computation

Deputy Director:

Prof. Dr. Bert Jüttler
Institute of Applied Geometry

Address:

Johannes Kepler University Linz
Doctoral Program "Computational Mathematics"
Altenbergerstr. 69
A-4040 Linz
Austria
Tel.: ++43 732-2468-6840

E-Mail:

office@dk-compmath.jku.at

Homepage:

http://www.dk-compmath.jku.at

Submissions to the DK-Report Series are sent to two members of the Editorial Board who communicate their decision to the Managing Editor.