A Sage implementation for DD-finite functions

Antonio Jiménez-Pastor*
Doctoral Program Computational Mathematics (JKU)
`ajpastor@risc.uni-linz.ac.at`

Abstract

We present here the Sage package `dd_functions` which provides many features to compute with DD-finite functions, a natural extension of the class of holonomic or D-finite functions. The package, focused on a functional approach, allows the user to compute closure properties, extract coefficient sequences and compute the composition (as formal power series) of functions treated in this package. All these operations reduce the problem to linear algebra computations where classical division-free algorithms are used.

1 Introduction

A formal power series $f(x) = \sum_{n\geq 0} a_n x^n$ is called D-finite, if it satisfies a linear differential equation with polynomial coefficients [7, 11]. Many generating functions of combinatorial sequences are of this type as well as the most commonly used special functions [1, 2, 9]. These objects can be represented in finite terms using a defining differential equation and sufficiently many initial values.

We recently extended this class to a wider set of formal power series that satisfy linear differential equations with D-finite coefficients, and call them DD-finite [3]. In the same way as D-finite functions satisfy several closure properties that have been implemented in different computer algebra systems [10, 8, 6], we present here a Sage implementation for the closure properties and other operations on DD-finite functions following a linear algebra approach [4].

The finiteness of the representation for D-finite functions allows to represent DD-finite functions with a finite amount of data, using a defining differential equation and initial conditions. A difference between our package and other implementations is that our focus is on particular solutions. We include the initial conditions in the structure, not only the annihilating differential operator. This leads to two different steps while manipulating DD-finite functions: computing the final differential equation and computing the initial data required for the specific object. More details on the structure can be found in Section 2.

The package `ore_algebra` [6] implements the computations for the differential operators in the D-finite case and the package presented here uses it by default when computing with D-finite objects. If the user desires otherwise, or `ore_algebra` is not installed, a different implementation [4] is provided. This implementation is used by default with DD-finite functions.

At the time of writing, the package described in this document is still under construction and has not been added to the official Sage distribution. Readers who want to try it are invited to download the current version from

`https://www.dk-compmath.jku.at/Members/antonio/sage-package-dd_functions`

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

and are encouraged to send bug reports, feature request or other comments.

Once downloaded and unpacked the file, the user should add the path to the obtained folder into the package folders in the Sage installation or just run Sage within the directory *diff_defined_functions*. Then the package is ready to be imported into any Sage session. Several tests are provided within the code and can be performed using the command `ajpastor.tests.dd_functions.run()`.

2 Data structure

Although the main goal of our package is to manipulate DD-finite functions, the theory can be stated in a more general setting that is covered by our implementation as well. For further details, see [3].

Definition 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.*

These functions are implemented using the *Parent-Element* model of Sage and our Parent structure is included in the *categories* framework, allowing the user to have an extended coercion system. This allows the user to have a more natural management of the differentially definable functions.

2.1 The Parent structure: *DDRing*

The structure `DDRing` describes the sets of differentially definable functions. Thus, they are built from any ring structure in Sage, R. Following the *Parent-Element* model of Sage, `DDRing` is a parent class that will include as element any differentially definable function over R.

To create a `DDRing`, the user has to provide the base ring R for the coefficients of the differential operators with a derivation. There are several optional inputs that allow more flexibility and a wider use of the structure, such as different derivations or adding parameters. For more information, use `DDRing?`.

This structure is only used to create differentially definable functions and to cast other elements (if possible). It is also used to compute a common parent (`pushout`) for two different elements, which allows the user to compute more easily with these objects.

Two `DDRings` are defined by default in the system when the package is loaded: the object `DFinite` that is the ring of D-finite functions ($R = \mathbb{Q}[x]$), and `DDFinite` which represents the ring of DD-finite functions ($R = \mathrm{D}(\mathbb{Q}[x])$).

2.2 The Element structure: *DDFunction*

The class `DDFunction` is the Element class of `DDRing` and represents a differentially definable function within a particular `DDRing`.

A `DDFunction` is always included in a `DDRing`. Thus the method `element` from the `DDRing` structure must be used. Any `DDFunction` is represented using a linear differential equation and some initial values, so the user must provide two lists to create a `DDFunction`: a list $[r_0, \dots, r_d]$ with coefficients for the differential equation and a list $[a_0, \dots, a_n]$ with initial values for the function:

$$([r_0, \dots, r_d], [a_0, \dots, a_n]) \mapsto \left\{ \begin{array}{c} r_d f^{(d)}(x) + \cdots + r_0 f(x) = 0, \\ f(0) = a_0, \dots, f^{(n)}(0) = a_n. \end{array} \right.$$

The method `element` also checks that the coefficients r_i are in the appropriate ring of coefficients R and that the initial conditions a_i are elements of the field where the function $f(x)$ is considered.

We provide a set of examples of D-finite and DD-finite functions that can be easily called and built from Sage once the package is loaded. Elementary functions as the exponential or trigonometric functions; special functions as the hypergeometric ${}_pF_q$ functions, solutions to the Riccati equation or the Mathieu functions are some of these examples. Use `ddExamples?` for further information.

3 Main features

Any `DDFunction` incorporates several methods that can be used. We split them into two big categories: utility methods and operations. The use of the *magic Python* methods (if they are indicated) is encouraged in order to use appropriately the coercion system of Sage.

3.1 Utility methods

These methods allow the user to extract information from any `DDFunction`.

- `equation`: the differential operator that defines the function.
- `getInitialValue`: the value of the nth derivative of the function at $x = 0$.
- `getSequenceElement`: the value of the nth coefficient of the formal power series.
- `getOrder`: the order of the differential operator defining the function.
- `min_coefficient`: the first non-zero coefficient of the formal power series
- `zero_extraction`: the order of the formal power series and the `DDFunction` defined after factoring the maximal power of x possible.

3.2 Operational methods

Differentially definable functions satisfy several closure properties. They are closed under addition, multiplication, differentiation and integration [3]. The user can also compute the division [3] and composition [5] whenever those operations are defined as formal power series. The implementations are based on linear algebra computations [4].

- **Addition**: `add` or simply `+`
- **Multiplication**: `mult` or simply `*`
- **Difference**: `sub` or simply `-`
- **Division**: `div` or simply `/`
- **Exponentiation**: `pow` or simply `^`
- **Derivation**: `derivative`.
- **Integration**: `integrate`. The value at $x = 0$ can be specified. By default, it is 0.
- **Composition**: `compose` or the magic method `__call__`.

Further algebraic properties that has been proven (see [7, 5]) for `DDFunctions` can also be computed with our package.

- `DAlgebraic`: given an algebraic function $f(x)$ over a field F, represents $f(x) \in \mathrm{D}(F)$.
- `diff_to_diffalg`: given a `DDFunction`, computes a differentially algebraic equation.

4 Conclusion

The Sage package `dd_functions` provides a functional-focused implementation of differentially definable functions for arbitrary differential rings. In particular, it can be used for working with regular D-finite functions and DD-finite functions.

This package can be used to prove symbolically identities between DD-finite functions and to obtain combinatorial sequences that are out of the D-finite scope.

Closure properties, such as addition, multiplication and exponentiation can be performed by the package, as well as other operations such as composition or division. It also provides several methods to convert algebraic functions to `DDFunction` and to compute a non-linear equation with polynomial coefficients for any differentially definable function.

References

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

[2] *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.

[3] Jiménez-Pastor, A., and Pillwein, V. A computable extension for holonomic functions: DD-finite functions. *J. Symbolic Comput.* (2018), 16.

[4] Jiménez-Pastor, A., and Pillwein, V. Algorithmic Arithmetics with DD-Finite Functions. In *Proceedings of the 2018 ACM on International Symposium on Symbolic and Algebraic Computation* (New York, NY, USA, 2018), A. Carlos, Ed., ISSAC '18, ACM, pp. 231–237.

[5] Jiménez-Pastor, A., Pillwein, V., and Singer, M. F. Some structural results on D^n-finite functions. Tech. rep., Doctoral Program Computational Mathematics, Preprint series, 2019. Submitted to journal.

[6] Kauers, M., Jaroschek, M., and Johansson, F. Ore Polynomials in Sage. In *Computer Algebra and Polynomials* (2014), J. Gutierrez, J. Schicho, and M. Weimann, Eds., Lecture Notes in Computer Science, pp. 105–125.

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

[8] Koutschan, C. *Advanced Applications of the Holonomic Systems Approach.* PhD thesis, RISC-Linz, Johannes Kepler University, 9 2009.

[9] Rainville, E. D. *Special Functions*, first ed. Chelsea Publishing Co., Bronx, N.Y., 1971.

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

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