Automatic Computation of Formal Power Series

SIMO Tao Lee Walter Cedric (cedric.lee@aims-cameroon.org)
African Institute for Mathematical Sciences (AIMS)
Cameroon

Supervised by: Prof. Dr. Wolfram Koepf
Universität Kassel, Germany

June 22, 2016

Submitted in Partial Fulfillment of a Structured Masters Degree at AIMS-Cameroon

Abstract

In this essay, we are implementing in the Computer Algebra System (CAS) Maxima the algorithm for computing formal power series (FPS) of a wide family of functions published in 1992 by Wolfram Koepf [3]. Indeed in the CAS Maxima, the computation of FPS has been rather pattern-based than algorithmic. The algorithm we are implementing in the CAS Maxima has already been implemented by W. Koepf and A. Rennoch in the CAS Mathematica, and by D. Gruntz and T. Sprenger in Maple.

Declaration

I, the undersigned, hereby declare that the work contained in this essay is my original work, and that any work done by others or by myself previously has been acknowledged and referenced accordingly.

[illegible]

SIMO TAO LEE Walter Cedric, June 22, 2016.

Contents

1. Introduction

The expansion $\exp(x) = \sum_{n=0}^{\infty} \frac{x^n}{n!}$ is well known in mathematics and is called the formal power series of the function $\exp(x)$. More generally, in mathematics, formal power series (FPS) which can be defined as an abstraction of the notion of a polynomial, where the number of terms can be infinite, are well known and play a major role in calculus, complex analysis and algebra. Given a function f, it is quite easy to compute its truncated power series and thus this computation is available in each computer algebra system. However the automatic computation of formal power series, i.e. finding a closed formula of the main coefficient a_n in

$$f(x) = \sum_{n=0}^{\infty} a_n x^n,$$

is rather more difficult and is not so far implemented in all Computer Algebra Systems (CASs). In some CAS like Maxima (see Maxima Manual [2]), the computation of FPS relies on a pattern matching based approach whereas in some other CASs like Mathematica (see [6]) and Maple (see Maple Reference Manual [1]), this computation is automatic and based on an algorithmic approach. As illustration, assume we are given the function $f(x) = \sin(x)\exp(x)$ and we wish to have the formal power series of $f(x)$ at the point $x_0 = 0$. In Maple, the command `convert(sin(x)*exp(x),FPS)` does the computation and gives

$$\sum_{k=0}^{\infty} \frac{2^{k/2}\sin(\frac{k\pi}{4})x^k}{k!},$$

whereas in Maxima, the command `powerseries(sin(x)*exp(x),x,0)` rather gives

$$\left(\sum_{i1=0}^{\infty} \frac{x^{i1}}{i1!}\right) \sum_{i1=0}^{\infty} \frac{(-1)^{i1} x^{2i1+1}}{(2i1+1)!},$$

which is no longer a formal power series. The second result obtained with Maxima clearly illustrates the pattern matching based approach used in Maxima for computing FPS contrary to the algorithmic approach implemented in Maple and Mathematica.

In [3] on which we are mainly focused, there are 2 algorithms: one for computing FPS of hypergeometric type functions ([3, Section 3, page 589]) and the other one for computing FPS of rational functions ([3, Section 4, page 593]). In this work, based on those algorithms, we build up a package which will enable automatic computation of formal power series in Maxima like in Maple and Mathematica.

Our work will be organised in five sections. The first three sections are about the implementation of the main algorithm for computing FPSs of hypergeometric type functions. The first section is the starting step in the algorithm which is getting the holonomic differential equation, that is the linear homogeneous differential equation with polynomial coefficients, satisfied by a given function f. The second section is devoted to the conversion of a holonomic differential equation (HDE) to a holonomic recurrence equation (HRE). The third section presents the resolution of recurrence equations of hypergeometric type. Afterwards comes Section 4 where the main point is the implementation of the rational algorithm in order to compute formal power series of rational functions. And finally we end up in section 5 with the combination of all the results of the previous sections to build up a function to compute the FPS of any holonomic or rational function. Also in this step, we are going to present interesting results we got.

2. Holonomic Differential Equation

In the process of getting the FPS of a given function f, the very first step is the computation of a holonomic differential equation (HDE), that is a linear homogeneous differential equation with polynomial coefficients, satisfied by that function f. In this chapter, using some examples, we are going to illustrate how the HDE is computed. Afterwards, we will present the general procedure with its implementation in the CAS Maxima, and finally we will apply this procedure to compute the HDEs of some functions using the function `Holonomic_DE()` from our package.

2.1 Illustration and Examples

In this part, we are going to illustrate the process of getting the HDE of a given function f by taking some functions and computing their HDE (starting from simple cases to more difficult ones). By the way, let us recall that the simplest case is when the function f is constant. Indeed when the function f is constant, the first derivative is 0 and then the HDE is

$$y' = 0, \quad y(0) = f(0).$$

Now let us consider examples where the function f is not constant.

2.1.1 Example. Let the function f be defined by $f(x) = \exp(x^2 + 1)$. Let us assume that f satisfies an HDE of order 1, that means there exist 2 polynomials $P_0(x)$ and $P_1(x)$ such that $P_1(x)$ is not the zero polynomial and

$$P_1(x)f'(x) + P_0(x)f(x) = 0. \tag{2.1.1}$$

Since P_1 is not the zero polynomial, we can define the rational function $Q_0(x) = \frac{P_0(x)}{P_1(x)}$ and then Equation (2.1.1) becomes

$$f'(x) + Q_0(x)f(x) = 0. \tag{2.1.2}$$

This equation leads to

$$Q_0(x) = -\frac{f'(x)}{f(x)}. \tag{2.1.3}$$

Since $f'(x) = 2x\exp(x^2 + 1)$, we deduce that $Q_0(x) = -\frac{2x\exp(x^2+1)}{\exp(x^2+1)} = -2x$. From that, we can consider $P_0(x) = -2x$ and $P_1(x) = 1$, and have the HDE with initial values (IV-HDE) of order 1 satisfied by f as

$$y' - 2x\,y = 0, \quad y(0) = f(0) = e. \tag{2.1.4}$$

2.1.2 Example. Let us consider the function $f(x) = \exp(x^2 + 1)\sin(x)$. We start by assuming that the function f satisfies an HDE of order 1. From Equations (2.1.2) and (2.1.3), we deduce the existence of a rational function $Q_0(x)$ such that $Q_0(x) = -\frac{f'(x)}{f(x)}$. But $f'(x) = 2x\exp(x^2 + 1)\sin(x) + \exp(x^2 + 1)\cos(x)$, which implies that

$$\frac{f'(x)}{f(x)} = \frac{2x\exp(x^2+1)\sin(x) + \exp(x^2+1)\cos(x)}{\exp(x^2+1)\sin(x)} = 2x + \tan(x)^{-1},$$

which is clearly not a rational function. Thus we deduce that f does not satisfy an HDE of order 1. Then we continue by assuming now that f satisfies an HDE of order 2. That means there exist 3 polynomials $P_0(x), P_1(x)$ and $P_2(x)$ such that $P_2(x)$ is not the zero polynomial and

$$P_2(x)f''(x) + P_1(x)f'(x) + P_0(x)f(x) = 0. \tag{2.1.5}$$

And since $P_2(x)$ is not the zero polynomial, we can define 2 rational functions $Q_0(x)$ and $Q_1(x)$ by

$$Q_0(x) = \frac{P_0(x)}{P_2(x)} \quad \text{and} \quad Q_1(x) = \frac{P_1(x)}{P_2(x)}, \tag{2.1.6}$$

and Equation (2.1.5) becomes

$$f''(x) + Q_1(x)f'(x) + Q_0(x)f(x) = 0. \tag{2.1.7}$$

But

$$\begin{aligned} f''(x) &= \exp(x^2+1)\left(2\sin(x) + 4x^2\sin(x) + 2x\cos(x) + 2x\cos(x) - \sin(x)\right) \\ &= \exp(x^2+1)\left((4x^2+1)\sin(x) + 4x\cos(x)\right). \end{aligned}$$

So after dividing by the common factor $\exp(x^2+1)$, Equation (2.1.7) can be written as:

$$(4x^2+1)\sin(x) + 4x\cos(x) + (2x\sin(x) + \cos(x))\,Q_1(x) + \sin(x)Q_0(x) = 0, \tag{2.1.8}$$

or equivalently

$$\left(4x^2 + 1 + 2xQ_1(x) + Q_0(x)\right)\sin(x) + (Q_1(x) + 4x)\cos(x) = 0. \tag{2.1.9}$$

But since the functions $\cos(x)$ and $\sin(x)$ are linearly independent, Equation (2.1.9) leads to the following system of 2 equations with 2 unknown functions $Q_0(x)$ and $Q_1(x)$

$$\begin{cases} 4x^2 + 1 + 2xQ_1(x) + Q_0(x) = 0 \\ Q_1(x) + 4x = 0. \end{cases} \tag{2.1.10}$$

The solution of this system is

$$Q_0(x) = 4x^2 - 1 \quad \text{and} \quad Q_1(x) = -4x.$$

From that, since $Q_0(x) = \frac{P_0(x)}{P_2(x)}$ and $Q_1(x) = \frac{P_1(x)}{P_2(x)}$, we can consider

$$P_0(x) = 4x^2 - 1,\ P_1(x) = -4x \quad \text{and} \quad P_2(x) = 1.$$

Taking also into consideration that $f(0) = 0$ and $f'(0) = e$, we end up with the following IV-HDE of order 2

$$y'' - 4xy' + (4x^2 - 1)y = 0, \quad y(0) = 0, y'(0) = e. \tag{2.1.11}$$

2.1.3 Example. Let us consider the function $f(x) = \sin^2(x)$. As in the first 2 examples above we start by assuming the function f satisfies an HDE of order 1 and from what we did above, that implies that there exists a rational function $Q_0(x)$ satisfying Equation (2.1.3) i.e.

$$Q_0(x) = -\frac{f'(x)}{f(x)}.$$

But $f'(x) = 2\cos(x)\sin(x)$, what implies that $\frac{f'(x)}{f(x)} = 2\frac{\cos(x)}{\sin(x)} = 2\tan(x)^{-1}$ which is clearly a contradiction since $\tan(x)^{-1}$ is not a rational function. Thus we deduce that f does not satisfy an HDE of order 1. Then we continue by assuming now that f satisfies an HDE of order 2. That means there exist 2 rational functions $Q_0(x)$ and $Q_1(x)$ such that

$$f''(x) + Q_1(x)f'(x) + Q_0(x)f = 0. \tag{2.1.12}$$

But $f''(x) = -2\sin^2(x) + 2\cos^2(x) = 2(1 - 2\sin^2(x))$, so Equation (2.1.12) can be written as

$$2(1 - 2\sin^2(x)) + 2\cos(x)\sin(x)Q_1(x) + \sin^2(x)Q_0(x) = 0. \tag{2.1.13}$$

Equation (2.1.13) implies that

$$2\cos(x)\sin(x)Q_1(x) + \sin^2(x)(Q_0(x) - 4) = -2,$$

which clearly does not have a rational solution since the left hand side is not a constant function contrary to the right hand side. From that we deduce that the function f does not satisfy an HDE of order 2. Now we assume that f satisfies an HDE of order 3. That means there exist 4 polynomials $P_0(x), P_1(x), P_2(x)$ and $P_3(x)$ such that $P_3(x)$ is not the zero polynomial and

$$P_3(x)f''' + P_2(x)f'' + P_1(x)f' + P_0(x)f = 0. \tag{2.1.14}$$

Since $P_3(x)$ is not the zero polynomial, we can define 3 rational functions $Q_0(x), Q_1(x)$ and $Q_2(x)$ by

$$Q_0(x) = \frac{P_0(x)}{P_3(x)},\ Q_1(x) = \frac{P_1(x)}{P_3(x)} \text{ and } Q_2(x) = \frac{P_2(x)}{P_3(x)} \tag{2.1.15}$$

such that Equation (2.1.14) becomes

$$f''' + Q_2(x)f'' + Q_1(x)f' + Q_0(x)f = 0. \tag{2.1.16}$$

But $f'''(x) = -8\cos(x)\sin(x)$, hence Equation (2.1.16) can be written as

$$-8\cos(x)\sin(x) + 2(1 - 2\sin^2(x))Q_2(x) + 2\cos(x)\sin(x)Q_1(x) + \sin^2(x)Q_0(x) = 0. \tag{2.1.17}$$

Now we expand completely the expression and gather similar terms, that is linearly dependent terms over the field $\mathbb{Q}(x)$ of rational functions. For instance the terms $-4\sin^2(x)Q_2(x)$ and $\sin^2(x)Q_0(x)$ are similar since their quotient $\frac{-4\sin^2(x)Q_2(x)}{\sin^2(x)Q_0(x)} = \frac{-4Q_2(x)}{Q_0(x)}$ is a rational function. After applying this gathering to Equation (2.1.17), we get the following equation

$$(2Q_1(x) - 8)\cos(x)\sin(x) + (Q_0(x) - 4Q_2(x))\sin^2(x) + 2Q_2(x) = 0. \tag{2.1.18}$$

And since the functions $\cos(x)\sin(x)$, $\sin^2(x)$ and the identity function 1 are linearly independent, Equation (2.1.18) brings us to the following system of 3 equations with 3 unknown functions $Q_0(x), Q_1(x)$ and $Q_2(x)$

$$\begin{cases} 2Q_1(x) - 8 = 0 \\ Q_0(x) - 4Q_2(x) = 0 \\ Q_2(x) = 0. \end{cases} \tag{2.1.19}$$

The solution of this system is

$$Q_0(x) = 0, \quad Q_1(x) = 4 \quad \text{and} \quad Q_2(x) = 0. \tag{2.1.20}$$

From those rational functions $Q_0(x), Q_1(x)$ and $Q_2(x)$, we can define $P_0(x)$, $P_1(x)$, $P_2(x)$ and $P_3(x)$ as follows:

$$P_0(x) = 0, \quad P_1(x) = 4, \quad P_2(x) = 0 \quad \text{and} \quad P_3(x) = 1. \tag{2.1.21}$$

Furthermore, since $f(0) = 0$, $f'(0) = 0$, $f''(0) = 2$, we can then deduce that $\sin^2(x)$ satisfies the HDE of order 3

$$y''' + 4y' = 0, \quad y(0) = 0,\ y'(0) = 0,\ y''(0) = 2. \tag{2.1.22}$$

From the above examples, one can actually have a good picture of the procedure of computing an HDE of a given holonomic function. Thus we can now move to the general procedure and its implementation in the CAS Maxima.

2.2 General Procedure and Implementation in Maxima

Following the examples given in Section 2.1, let us present the general procedure to compute an HDE of a given holonomic function. Before going to the procedure, let us remind that if the given function f is not holonomic, then following the process used in the examples in Section 2.1, we will never stop. This is why at the beginning, we first of all need to fix a maximal order of the HDE we are searching. In our procedure we consider `MAX_ORDER_DERIVATIVE` to be that maximal order.
Furthermore before getting into the algorithm, let us remark that if f satisfies an HDE of order k, then finding $k+1$ polynomials $P_j(x),\ j \in \{0, ..., k\}$ such that

$$P_k(x)f^{(k)}(x) + P_{k-1}(x)f^{(k-1)}(x) + \cdots + P_1(x)f^{(1)}(x) + P_0(x)f(x) = 0 \tag{2.2.1}$$

is equivalent to finding k rational functions $Q_j(x),\ j \in \{0, ..., k-1\}$ such that

$$f^{(k)}(x) + Q_{k-1}(x)f^{(k-1)}(x) + \cdots + Q_1(x)f^{(1)}(x) + Q_0(x)f(x) = 0. \tag{2.2.2}$$

Indeed, if we have $k+1$ polynomials $P_j(x),\ j \in \{0, ..., k\}$ satisfying (2.2.1) then since $P_k(x)$ is not the zero polynomial (because the order of the HDE is k), dividing Equation (2.2.1) by $P_k(x)$ and defining

$$Q_j(x) = \frac{P_j(x)}{P_k(x)},\ j \in \{0, ..., k-1\}$$

we have k rational functions $Q_j(x),\ j \in \{0, ..., k-1\}$ satisfying (2.2.2).

Conversely let us assume that we have k rational functions $Q_j(x),\ j \in \{0, ..., k-1\}$ satisfying (2.2.2). Defining the polynomial $D(x) = lcm(denominator(Q_0(x)), ..., denominator(Q_{k-1}(x)))$ and multiplying Equation (2.2.2) by $D(x)$, we have

$$D(x)f^{(k)}(x) + D(x)Q_{k-1}(x)f^{(k-1)}(x) + \cdots + D(x)Q_1(x)f^{(1)}(x) + D(x)Q_0(x)f(x) = 0. \tag{2.2.3}$$

By the way, let us remark that $D(x)Q_j(x),\ j \in \{0, ..., k-1\}$ are now polynomial functions since $D(x)$ by definition is a polynomial multiple of $Denominator(Q_j(x))$ for all $j \in \{0, ..., k-1\}$. Dividing Equation (2.2.3) by

$$W(x) = \gcd\left(D(x), D(x)Q_{k-1}(x), ..., D(x)Q_1(x), D(x)Q_0(x)\right),$$

and taking $P_j(x) = \frac{D(x)Q_j(x)}{W(x)},\ j \in \{0, ..., k-1\}$ and $P_k(x) = \frac{D(x)}{W(x)}$, we end up with $k+1$ polynomials $P_j(x),\ j \in \{0, ..., k\}$ satisfying Equation (2.2.1).

Having clarified the latter point, let us move now to the general algorithm for the computation of an HDE of a given function f.

Algorithm for computing a holonomic differential equation of a given function f

1. We fix the upper bound of the order of the HDE we are looking for (in our case we fix it to `MAX_ORDER_DERIVATIVE`).
2. If $f'(x) = 0$ then the HDE is $y' = 0,\ y(0) = f(0)$ and we stop.
3. Otherwise we set a variable k to 1.
4. We compute $f^{(k)}(x)$, the k-th derivative of $f(x)$.

5. We write Equation $f^{(k)}(x) + \sum_{j=0}^{k-1} f^{(j)}(x)Q_j(x) = 0$.
6. We rewrite this equation in the form of a system of n_k equations with k unknown rational functions $Q_j(x), j \in \{0, ..., k-1\}$ after identification of similar terms, $n_k \in \mathbb{N}$.
7. We solve this system to find $Q_j(x), j \in \{0, ..., k-1\}$.
8. If the system does not have a solution and $k <$ `MAX_ORDER_DERIVATIVE`, we increment k by 1 (i.e. $k := k+1$) and we go to step 4.
9. If the system does not have a solution and $k =$`MAX_ORDER_DERIVATIVE`, we either reset `MAX_ORDER_DERIVATIVE` to a higher value and go to step 4 or exit with the message "No HDE satisfied by f of order less than or equal to `MAX_ORDER_DERIVATIVE`".
10. Otherwise if we found a solution $Q_j(x), j \in \{0, ..., k-1\}$, we set $D(x)$ to be the least common multiple of denominators of the rational functions $Q_j(x), j \in \{0, ..., k-1\}$ i.e.
$$D(x) := lcm(denominator(Q_0(x)), ..., denominator(Q_{k-1}(x))).$$
Afterwards we set $k+1$ polynomials $L_j(x) = D(x)Q_j(x),\ j \in \{0, ..., k-1\}$ and $L_k(x) := D(x)$ and finally we define
$$P_j(x) = \frac{L_j(x)}{\gcd(L_0(x), ..., L_k(x))}$$
in order to eliminate all the common factors. This leads to the simplest possible form of HDE satisfied by f as
$$P_k(x)y^{(k)} + \sum_{j=0}^{k-1} P_j(x)y^{(j)} = 0,\ y^{(j)}(0) = f^{(j)}(0) \text{ for } j \in \{0, ..., k-1\}. \tag{2.2.4}$$

In this algorithm, what is not really explicit is how to obtain the system of n_k equations with k unknown functions $Q_j(x),\ j \in \{0, ..., k-1\}$ from Equation
$$f^{(k)}(x) + \sum_{j=0}^{k-1} f^{(j)}(x)Q_j(x) = 0. \tag{2.2.5}$$
Let us give some clarifications on how this is done. First of all, we expand completely Equation (2.2.5). Afterwards we gather together summands that are similar to get an equation of the form
$$R_{n_k-1}(x)F_{n_k-1}(x) + ... + R_1(x)F_1(x) + R_0(x)F_0(x) = 0 \quad \text{with} \quad n_k \in \mathbb{N}, \tag{2.2.6}$$
where $R_l(x),\ l \in \{0, ..., n_k-1\}$ are rational functions (and contain also $Q_j(x), j \in \{0, 1, ..., k-1\}$) and $F_l(x),\ l \in \{0, ..., n_k-1\}$ are free of $Q_j(x)$ ($j \in \{0, 1, ..., k-1\}$) and are linearly independent functions over the field of rational functions $\mathbb{Q}(x)$. We can find illustrations of such rewriting of equation in the second example, Equation (2.1.9) and the third example, Equation (2.1.18). Since the functions $F_l(x),\ l \in \{0, ..., n_k-1\}$ are linearly independent, equating the coefficients of $F_l(x),\ l \in \{0, ..., n_k-1\}$ in Equation (2.2.6) yields the following system of n_k equations with k unknown functions $Q_j,\ j \in \{0, ..., k-1\}$
$$\begin{cases} R_{n_k-1}(x) = 0 \\ R_{n_k-2}(x) = 0 \\ \cdots \\ R_1(x) = 0 \\ R_0(x) = 0. \end{cases}$$

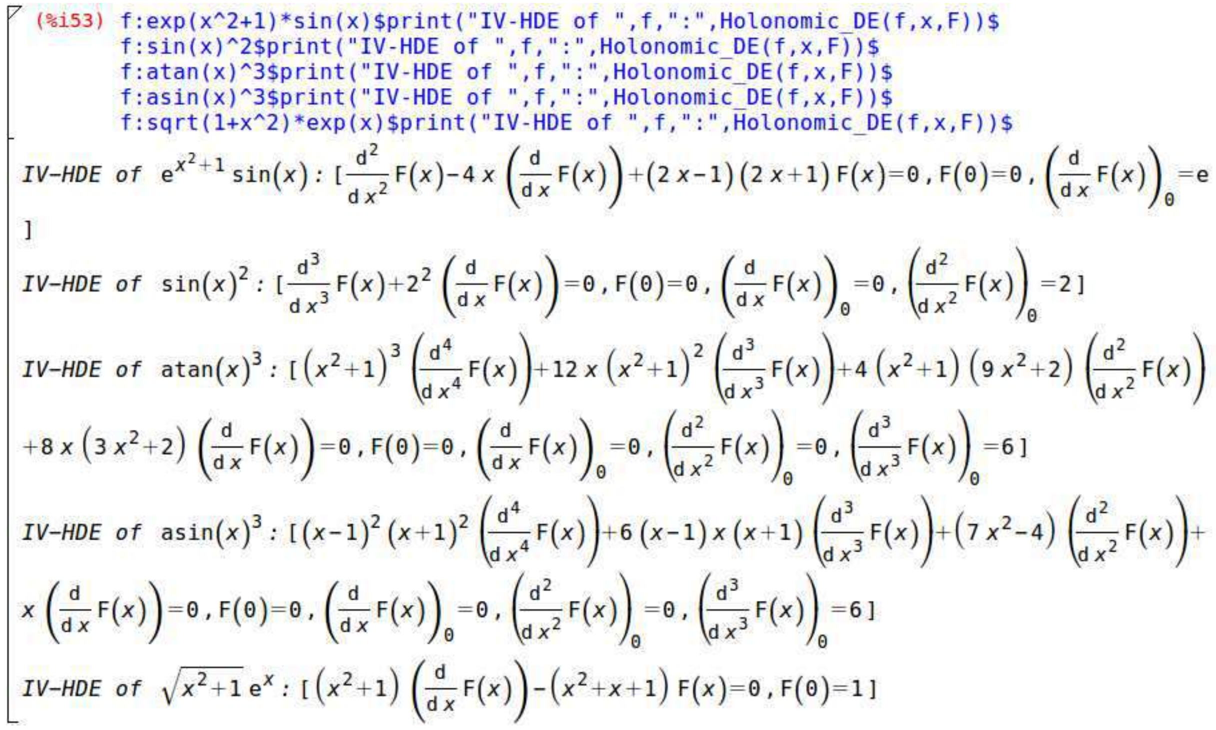
In order to achieve every step presented above and build a function `Holonomic_DE(f,x)` to compute a IV-HDE of a given function `f` with respect to the variable `x`, we define many other functions having different aims. Here are the functions we defined in our package with a brief description of their purpose:

- `Polynomial_P(Exp,x)`: this function checks whether `Exp` is a polynomial with respect to the variable `x` or not. Let us remind that there is an embedded function `polynomialp(p,L)` in Maxima which checks whether the expression `p` is a polynomial with respect to the given list of variables `L`. But this embedded function failed to recognise for instance that $\exp(y)x+\tan(y)x^2$ is a polynomial with respect to x, reason why we decided to define the function `Polynomial_P(Exp,x)`;
- `Rational_P(Exp,x)`: this function uses the function `Polynomial_P` to check whether the expression `Exp` is a rational function with respect to `x` or not;
- `Terms_functions(Fct)`: this function takes an expression `Fct` which is a sum of terms and returns the list of terms occurring in the expression `Fct`.
  As illustration: `Terms_function(`$\cos(x)\exp(x)+\tan(x)+\sin(x)$`)` gives $[\cos(x)\exp(x), \tan(x), \sin(x)]$;
- `Group_terms(Fct,x)`: this function takes a function `Fct`, uses the function `Terms_functions` and returns a list as output. In this list, each element is a sum of linearly dependent terms over $\mathbb{Q}(x)$ occurring in `Fct`.
  As illustration: `Group_terms(`$\cos(x)+a\sin(x)+bx^2\cos(x), x$`)` gives $[[\cos(x)+bx^2\cos(x)], [a\sin(x)]]$;
- `Find_p_k(G,L)`: this function takes a list `G` of functions and a list `L` of unknown coefficients and returns the solution of the system of equations obtained by assuming that each function in the list `G` is equal to zero. As illustration: Find_p_k($[a\cos(x) - x^2\cos(x), b\sin(x) - \sin(x)], [a, b]$) gives $[x^2, 1]$;
- `Linear_formal_eq(f,x,n)` takes a function `f` of the variable `x` and returns a linear combination of `f` and its `n` derivatives and the list of the unknown coefficients of this linear combination.
  As illustration: `Linear_formal_eq(`$\cos(x), x, 2$`)` gives $[-\cos(x) - p_1\sin(x) + p_0\cos(x), [p_0, p_1]]$;
- `Coefficient_diff_eq(f,x)`: this function takes a function `f` of the variable `x` and returns the list of the polynomial coefficients in the Holonomic DE satisfied by `f` in this way: $[P_0(x), P_1(x), ..., P_k(x)]$ where k here represents the order of the HDE.
  As illustration: `Coefficient_diff_eq(`$\exp(x^2+1), x$`)` gives $[-2x, 1]$;
- `Reduce_denominator(L)`: this function takes a list `L` of rational functions and returns the simplest list of polynomials proportional to `L`.
  As illustration: `Reduce_denominator(`$[x/(1-x), 3x^2/(x^2-1)]$`)` gives $[x+1, -3x]$ (in this example the coefficient of proportionality is $\frac{x}{x^2-1}$);
- `Holonomic_DE(f,x,F)`: this function takes a function `f`, its main variable `x` and returns an HDE satisfied by `f` with `F` as unknown function in the HDE.

Let us recall that the maximal order `MAX_ORDER_DERIVATIVE` of the HDE in the algorithm above is initialised to 4 and can be changed any time the user wants.

2.3 Some Applications

In this part, we are going to present some results we obtained with our function `Holonomic_DE()` defined for the computation of the HDE of a given function `f`. Below is a snapshot of the computation of the HDE of some functions using our function `Holonomic_DE()`.



```
(%i53) f:exp(x^2+1)*sin(x)$print("IV-HDE of ",f,":",Holonomic_DE(f,x,F))$
       f:sin(x)^2$print("IV-HDE of ",f,":",Holonomic_DE(f,x,F))$
       f:atan(x)^3$print("IV-HDE of ",f,":",Holonomic_DE(f,x,F))$
       f:asin(x)^3$print("IV-HDE of ",f,":",Holonomic_DE(f,x,F))$
       f:sqrt(1+x^2)*exp(x)$print("IV-HDE of ",f,":",Holonomic_DE(f,x,F))$
```

IV-HDE of $e^{x^2+1}\sin(x) : [\frac{d^2}{d x^2}F(x)-4x\left(\frac{d}{d x}F(x)\right)+(2x-1)(2x+1)F(x)=0, F(0)=0, \left(\frac{d}{d x}F(x)\right)_0=e]$

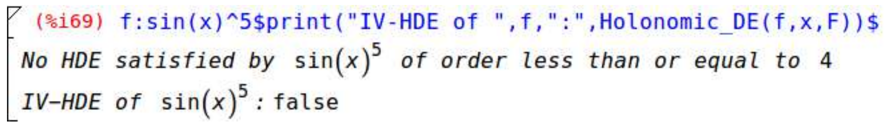
IV-HDE of $\sin(x)^2 : [\frac{d^3}{d x^3}F(x)+2^2\left(\frac{d}{d x}F(x)\right)=0, F(0)=0, \left(\frac{d}{d x}F(x)\right)_0=0, \left(\frac{d^2}{d x^2}F(x)\right)_0=2]$

IV-HDE of $\operatorname{atan}(x)^3 : [\left(x^2+1\right)^3\left(\frac{d^4}{d x^4}F(x)\right)+12x\left(x^2+1\right)^2\left(\frac{d^3}{d x^3}F(x)\right)+4\left(x^2+1\right)\left(9x^2+2\right)\left(\frac{d^2}{d x^2}F(x)\right)$
$+8x\left(3x^2+2\right)\left(\frac{d}{d x}F(x)\right)=0, F(0)=0, \left(\frac{d}{d x}F(x)\right)_0=0, \left(\frac{d^2}{d x^2}F(x)\right)_0=0, \left(\frac{d^3}{d x^3}F(x)\right)_0=6]$

IV-HDE of $\operatorname{asin}(x)^3 : [(x-1)^2(x+1)^2\left(\frac{d^4}{d x^4}F(x)\right)+6(x-1)x(x+1)\left(\frac{d^3}{d x^3}F(x)\right)+\left(7x^2-4\right)\left(\frac{d^2}{d x^2}F(x)\right)+$
$x\left(\frac{d}{d x}F(x)\right)=0, F(0)=0, \left(\frac{d}{d x}F(x)\right)_0=0, \left(\frac{d^2}{d x^2}F(x)\right)_0=0, \left(\frac{d^3}{d x^3}F(x)\right)_0=6]$

IV-HDE of $\sqrt{x^2+1}\,e^x : [\left(x^2+1\right)\left(\frac{d}{d x}F(x)\right)-\left(x^2+x+1\right)F(x)=0, F(0)=1]$

In the above cases, `MAX_ORDER_DERIVATIVE` is 4 and we cannot obtain the HDE of $\sin^5(x)$ (since the order of the HDE of $\sin^5(x)$ is 6). And if we keep trying we will have:

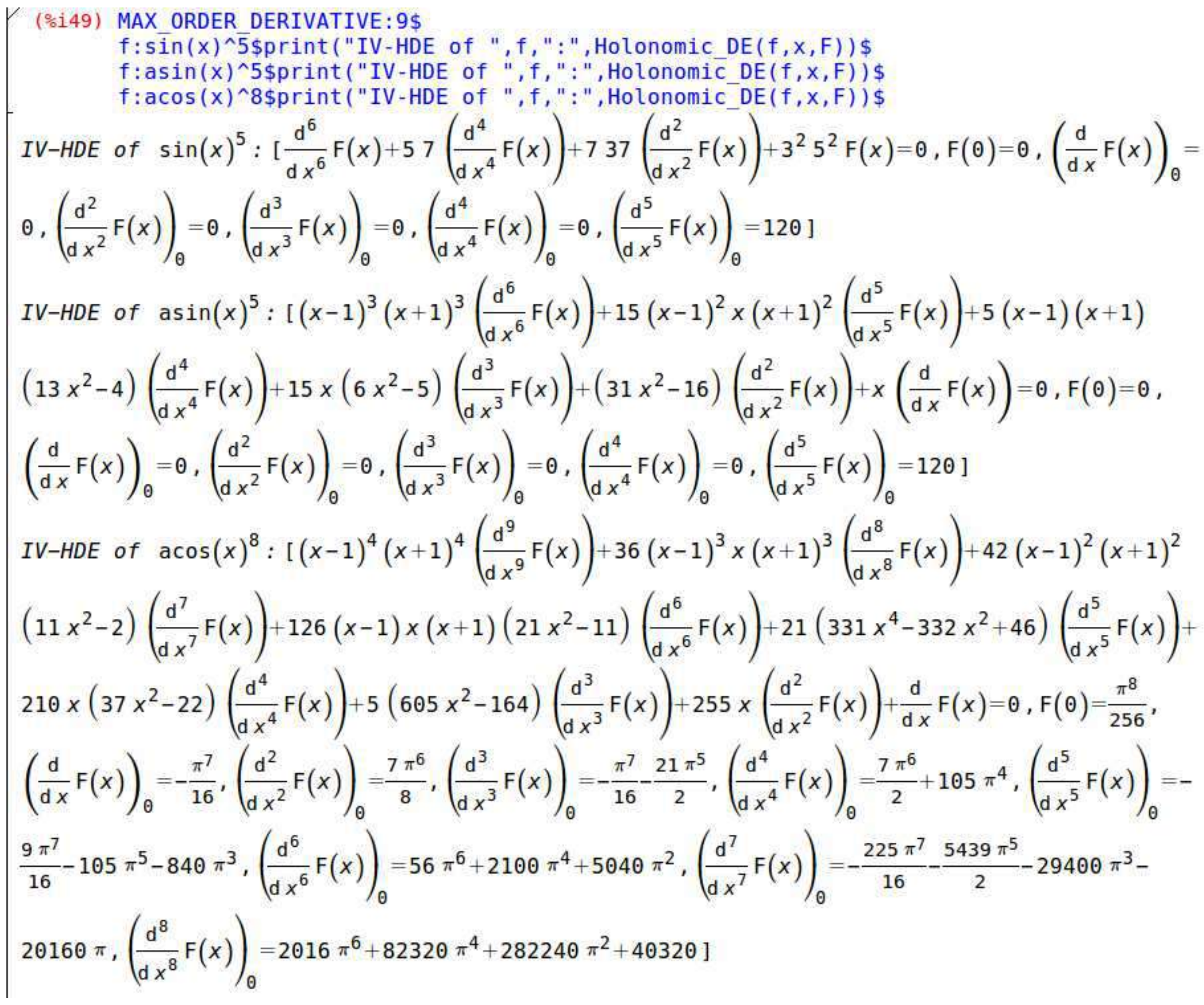


```
(%i69) f:sin(x)^5$print("IV-HDE of ",f,":",Holonomic_DE(f,x,F))$
```

No HDE satisfied by $\sin(x)^5$ *of order less than or equal to* 4

IV-HDE of $\sin(x)^5$: false

We increase the value of `MAX_ORDER_DERIVATIVE` in order to compute HDE of order greater than 4 as shown in the examples below.



```
(%i49) MAX_ORDER_DERIVATIVE:9$
       f:sin(x)^5$print("IV-HDE of ",f,":",Holonomic_DE(f,x,F))$
       f:asin(x)^5$print("IV-HDE of ",f,":",Holonomic_DE(f,x,F))$
       f:acos(x)^8$print("IV-HDE of ",f,":",Holonomic_DE(f,x,F))$
```

IV-HDE of $\sin(x)^5$: $[\frac{d^6}{dx^6}F(x)+5\,7\left(\frac{d^4}{dx^4}F(x)\right)+7\,37\left(\frac{d^2}{dx^2}F(x)\right)+3^2\,5^2\,F(x)=0, F(0)=0, \left(\frac{d}{dx}F(x)\right)_0=0, \left(\frac{d^2}{dx^2}F(x)\right)_0=0, \left(\frac{d^3}{dx^3}F(x)\right)_0=0, \left(\frac{d^4}{dx^4}F(x)\right)_0=0, \left(\frac{d^5}{dx^5}F(x)\right)_0=120]$

IV-HDE of $\operatorname{asin}(x)^5$: $[(x-1)^3(x+1)^3\left(\frac{d^6}{dx^6}F(x)\right)+15(x-1)^2x(x+1)^2\left(\frac{d^5}{dx^5}F(x)\right)+5(x-1)(x+1)(13x^2-4)\left(\frac{d^4}{dx^4}F(x)\right)+15x(6x^2-5)\left(\frac{d^3}{dx^3}F(x)\right)+(31x^2-16)\left(\frac{d^2}{dx^2}F(x)\right)+x\left(\frac{d}{dx}F(x)\right)=0, F(0)=0, \left(\frac{d}{dx}F(x)\right)_0=0, \left(\frac{d^2}{dx^2}F(x)\right)_0=0, \left(\frac{d^3}{dx^3}F(x)\right)_0=0, \left(\frac{d^4}{dx^4}F(x)\right)_0=0, \left(\frac{d^5}{dx^5}F(x)\right)_0=120]$

IV-HDE of $\operatorname{acos}(x)^8$: $[(x-1)^4(x+1)^4\left(\frac{d^9}{dx^9}F(x)\right)+36(x-1)^3x(x+1)^3\left(\frac{d^8}{dx^8}F(x)\right)+42(x-1)^2(x+1)^2(11x^2-2)\left(\frac{d^7}{dx^7}F(x)\right)+126(x-1)x(x+1)(21x^2-11)\left(\frac{d^6}{dx^6}F(x)\right)+21(331x^4-332x^2+46)\left(\frac{d^5}{dx^5}F(x)\right)+210x(37x^2-22)\left(\frac{d^4}{dx^4}F(x)\right)+5(605x^2-164)\left(\frac{d^3}{dx^3}F(x)\right)+255x\left(\frac{d^2}{dx^2}F(x)\right)+\frac{d}{dx}F(x)=0, F(0)=\frac{\pi^8}{256}, \left(\frac{d}{dx}F(x)\right)_0=-\frac{\pi^7}{16}, \left(\frac{d^2}{dx^2}F(x)\right)_0=\frac{7\pi^6}{8}, \left(\frac{d^3}{dx^3}F(x)\right)_0=-\frac{\pi^7}{16}-\frac{21\pi^5}{2}, \left(\frac{d^4}{dx^4}F(x)\right)_0=\frac{7\pi^6}{2}+105\pi^4, \left(\frac{d^5}{dx^5}F(x)\right)_0=-\frac{9\pi^7}{16}-105\pi^5-840\pi^3, \left(\frac{d^6}{dx^6}F(x)\right)_0=56\pi^6+2100\pi^4+5040\pi^2, \left(\frac{d^7}{dx^7}F(x)\right)_0=-\frac{225\pi^7}{16}-\frac{5439\pi^5}{2}-29400\pi^3-20160\pi, \left(\frac{d^8}{dx^8}F(x)\right)_0=2016\pi^6+82320\pi^4+282240\pi^2+40320]$

3. Conversion of a Holonomic Differential Equation to a Recurrence Equation

In the previous chapter, we saw how to get the IV-HDE of a given holonomic function f. After getting this IV-HDE, on the way of computing the FPS of the function f, the next step is to convert the IV-HDE into a recurrence equation with initial values (IV-RE) for the coefficients a_n of the power series $f(x) = \sum_{n=0}^{\infty} a_n x^n$. Indeed we assume that the solution of the HDE can be written as a power series of the form $\sum_{n=0}^{\infty} a_n x^n$ and we substitute this into the differential equation in order to end with a recurrence relation in a_n with initial values. In this chapter, the main point is the conversion of an IV-HDE for $f(x)$ to an IV-RE for a_n. The first part presents some illustrative examples of how the conversion is done, the second part is the general procedure with its implementation in the CAS Maxima, and in the last part we show some IV-REs computed using this algorithm.

3.1 Illustrative Examples

Let us start by giving some examples of conversion of an IV-HDE into an IV-RE starting from the simplest case to more difficult ones. By the way, let us recall that the simplest conversion is when the IV-HDE is

$$y'(x) = 0, \quad y(0) = a. \tag{3.1.1}$$

In this case, we end up obviously with the IV-RE

$$a_n = 0, \quad \forall n > 0 \quad \text{and} \quad a_0 = a.$$

Now let us consider the cases where the IV-HDE is different from (3.1.1).

3.1.1 Example. Let us consider the IV-HDE

$$y'(x) - y(x) = 0, \quad y(0) = a.$$

The first thing is to write the solution $y(x)$ in the form $y(x) = \sum_{n=0}^{\infty} a_n x^n$. Afterwards, we formally differentiate the series $y(x)$ and plug it into the IV-HDE to get

$$\sum_{n=1}^{\infty} n a_n x^{n-1} - \sum_{n=0}^{\infty} a_n x^n = 0, \quad a_0 = a. \tag{3.1.2}$$

Next, we make a change of indices such that we have the same power (n) of x in each series in Equation (3.1.2). This leads to

$$\sum_{n=0}^{\infty} (n+1) a_{n+1} x^n - \sum_{n=0}^{\infty} a_n x^n = 0, \quad a_0 = a, \tag{3.1.3}$$

which is equivalent to

$$\sum_{n=0}^{\infty} \left((n+1) a_{n+1} - a_n\right) x^n = 0, \quad a_0 = a.$$

Thus we deduce the IV-RE

$$(n+1) a_{n+1} - a_n = 0, \; \forall n \geq 0, \; a_0 = a. \tag{3.1.4}$$

3.1.2 Example. Let us consider the IV-HDE

$$(x^3+1)y'(x) - xy(x) = 0, \quad y(0) = a.$$

As said previously, we assume that the solution of this IV-HDE can be written as a series $y(x) = \sum_0^\infty a_n x^n$, we formally differentiate that series and we plug its formal derivatives into the IV-HDE. Having done that, we get

$$(x^3+1)\sum_{n=1}^{\infty} n a_n x^{n-1} - x\sum_{n=0}^{\infty} a_n x^n = 0, \quad y(0) = a. \tag{3.1.5}$$

We expand Equation (3.1.5) to get

$$\sum_{n=1}^{\infty} n a_n x^{n+2} + \sum_{n=1}^{\infty} n a_n x^{n-1} - \sum_{n=0}^{\infty} a_n x^{n+1} = 0, \quad y(0) = a. \tag{3.1.6}$$

Now applying change of indices in order to have the common exponent n in each series yields

$$\sum_{n=3}^{\infty}(n-2)a_{n-2}x^n + \sum_{n=0}^{\infty}(n+1)a_{n+1}x^n - \sum_{n=1}^{\infty} a_{n-1}x^n = 0, \quad y(0) = a,$$

which is equivalent to

$$\sum_{n=2}^{\infty}(n-2)a_{n-2}x^n + \sum_{n=0}^{\infty}(n+1)a_{n+1}x^n - \sum_{n=1}^{\infty} a_{n-1}x^n = 0, \quad y(0) = a. \tag{3.1.7}$$

In Equation (3.1.7) contrary to Equation (3.1.3), the sum starts at 2 for the first series, 0 for the second one and 1 for the third one. Now we rewrite each series in order to have all the indices starting at the same value. Let us remark that the final starting value of indices is the highest one in series occurring in (3.1.7) and is 2. The first series will remain unchanged, the second and third series will be rewritten respectively as

$$a_1 + 2a_2x + \sum_{n=2}^{\infty}(n+1)a_{n+1}x^n, \quad -a_0x - \sum_{n=2}^{\infty} a_{n-1}x^n.$$

Equation (3.1.7) becomes

$$\sum_{n=2}^{\infty}(n-2)a_{n-2}x^n + a_1 + 2a_2x + \sum_{n=2}^{\infty}(n+1)a_{n+1}x^n - a_0x - \sum_{n=2}^{\infty} a_{n-1}x^n = 0, \quad a_0 = a,$$

which is equivalent to

$$a_1 + (2a_2 - a_0)x + \sum_{n=2}^{\infty}\left((n-2)a_{n-2} + (n+1)a_{n+1} - a_{n-1}\right)x^n = 0, \quad a_0 = a. \tag{3.1.8}$$

Equating the coefficients of x^n in Equation (3.1.8) yields the system:

$$\begin{cases} a_1 = 0 \\ 2a_2 - a_0 = 0 \\ (n-2)a_{n-2} + (n+1)a_{n+1} - a_{n-1} = 0 \quad \forall n \geq 2. \end{cases} \tag{3.1.9}$$

The first 2 equations in (3.1.9) imply that $a_2 = a_0/2 = a/2, a_1 = 0$ and we get the IV-RE

$$(n-2)a_{n-2} + (n+1)a_{n+1} - a_{n-1} = 0, \quad \forall n \geq 2 \quad a_0 = a,\ a_1 = 0,\ a_2 = a/2. \tag{3.1.10}$$

Since we have indices in the recurrence equation in (3.1.10) strictly less than n, we shift the index n (we replace n by $n+2$) in order to have the lowest index n and we finally obtain the IV-RE

$$\begin{cases} (n+3)a_{n+3} - a_{n+1} + na_n = 0 \quad \forall n \geq 0 \\ a_0 = a,\ a_1 = 0,\ a_2 = a/2. \end{cases} \tag{3.1.11}$$

3.1.3 Example. Let us consider the IV-HDE of the function $f(x) = \log(1-x^2)$ given by

$$\left(x^3 - x\right) y''(x) + \left(x^2+1\right) y'(x) = 0, \quad y(0) = 0,\ y'(0) = 0. \tag{3.1.12}$$

Assuming that $y(x)$ can be written as $y(x) = \sum_{n=0}^{\infty} a_n x^n$, differentiate $y(x)$ and substitute in (3.1.12) to get

$$\left(x^3 - x\right) \sum_{n=2}^{\infty} n(n-1)a_n x^{n-2} + \left(x^2+1\right) \sum_{n=1}^{\infty} n a_n x^{n-1} = 0, \quad a_0 = 0,\ a_1 = 0. \tag{3.1.13}$$

After expanding Equation (3.1.13) we have

$$\sum_{n=2}^{\infty} n(n-1)a_n x^{n+1} - \sum_{n=2}^{\infty} n(n-1)a_n x^{n-1} + \sum_{n=1}^{\infty} n a_n x^{n+1} + \sum_{n=1}^{\infty} n a_n x^{n-1} = 0,\ a_0 = 0,\ a_1 = 0. \tag{3.1.14}$$

We do a change of indices in Equation (3.1.14) in order to get the same exponent n in each series and obtain

$$\sum_{n=2}^{\infty}(n-1)(n-2)a_{n-1}x^n - \sum_{n=1}^{\infty}(n+1)(n)a_{n+1}x^n + \sum_{n=2}^{\infty}(n-1)a_{n-1}x^n + \sum_{n=0}^{\infty}(n+1)a_{n+1}x^n = 0,\ a_0 = 0,\ a_1 = 0. \tag{3.1.15}$$

Now, we rewrite Equation (3.1.15) as follows in order to have index in each series starting at the same value 2

$$\sum_{n=2}^{\infty}(n-1)(n-2)a_{n-1}x^n - 2a_2x - \sum_{n=2}^{\infty}(n+1)(n)a_{n+1}x^n +$$
$$\sum_{n=2}^{\infty}(n-1)a_{n-1}x^n + a_1 + 2a_2x + \sum_{n=2}^{\infty}(n+1)a_{n+1}x^n = 0,\ a_0 = 0,\ a_1 = 0.$$

This is equivalent to

$$\sum_{n=2}^{\infty}\left((n-1)(n-2)a_{n-1} - n(n+1)a_{n+1} + (n-1)a_{n-1} + (n+1)a_{n+1}\right)x^n = 0,\ a_0 = 0,\ a_1 = 0.$$

This leads to the system

$$\begin{cases} (n-1)^2 a_{n-1} - (n+1)(n-1))a_{n+1} = 0 \quad \forall n \geq 2 \\ a_0 = 0,\ a_1 = 0. \end{cases} \tag{3.1.16}$$

By shifting the index n in (3.1.16), we finally obtain the IV-RE

$$\begin{cases} n(n+2)a_{n+2} - n^2 a_n = 0 \quad \forall n \geq 1 \\ a_0 = 0,\, a_1 = 0. \end{cases} \tag{3.1.17}$$

Let us remark that the IV-RE (3.1.17) is incomplete since the value of a_2 is not given. And without its value, we cannot compute $a_{2n},\ n \geq 1$, however this issue will be resolved in the next section.

From the given examples above, we can actually see how the conversion of an IV-HDE to an IV-RE is done manually. The implementation of this conversion into programming follows the same steps.

3.2 General Procedure of Conversion and Implementation

Having given the above illustrative examples of the conversion of an IV-HDE into an IV-RE, let us now present the general procedure of the conversion.
Before getting into the procedure, let us do some remarks. Suppose we want to convert the terms $x^j y^{(k)}(x)$ into their corresponding terms in the recurrence equation (with $y(x) = f(x) = \sum_{n=0}^{\infty} a_n x^n$). The term $x^j y^{(k)}(x)$ from the differential equation corresponds to

$$x^j y^{(k)}(x) = x^j \sum_{n=0}^{\infty} n(n-1)\cdots(n-k+1)a_n x^{n-k} = \sum_{n=0}^{\infty} n(n-1)\cdots(n-k+1)a_n x^{n+j-k}.$$

The change of index $n \leftarrow n+k-j$ in order to have the exponent n gives

$$x^j y^{(k)}(x) = \sum_{n=j-k}^{\infty} (n+k-j)(n+k-j-1)\cdots(n-j+1)a_{n+k-j}x^n,$$

which can be written as

$$\sum_{n=j-k}^{\infty} (n-j+1)_k a_{n+k-j} x^n,$$

where the Pochhammer symbol is defined by:

$$(a)_k = \begin{cases} 1 & \text{if} \quad k=0 \\ a(a+1)\cdots(a+k-1) & \text{if} \quad k \in \mathbb{N}. \end{cases}$$

From that observation, we actually see that the corresponding term of $x^j y^{(k)}(x)$ in the recurrence equation is $(n-j+1)_k a_{n+k-j}$ i.e.

$$\text{HDE} \dashrightarrow \text{RE}$$
$$x^j y^{(k)}(x) \mapsto (n-j+1)_k a_{n+k-j}. \tag{3.2.1}$$

But since any HDE can be written as linear combination of terms of the form $x^j y^{(k)}(x)$ i.e. in the form

$$\sum_{k=0}^{N} \left(\sum_{j=0}^{n_k} \alpha_{j,k} x^j \right) y^{(k)}(x) \quad \text{with} \quad \alpha_{j,k} \quad \text{constant} \quad \text{and} \quad N,\, n_k \in \mathbb{N},$$

where N is the order the HDE and n_k is the degree of the polynomial coefficient of $y^{(k)}(x)$, we can deduce that using (3.2.1), any HDE can be converted into a holonomic recurrence equation (that means linear homogeneous recurrence equation with polynomial coefficients). This brings a light on how we convert an HDE to a recurrence equation. But how do we derive the initial values of the recurrence equation? First of all, we transform any single initial condition in the IV-HDE into one initial value in the recurrence equation using the formula

$$a_j = \frac{y^{(j)}(0)}{j!}. \tag{3.2.2}$$

But from the result of the conversion of the IV-HDE in Example (3.1.2), Equation (3.1.11), we can actually realise that the initial values coming from the direct conversion of initial conditions in the IV-HDE are not always enough. So we need to find a way to complete initial values in the IV-RE if necessary. By the way let us recall that in most cases the number of initial values in an IV-RE corresponds to the order of the recurrence equation. For instance this is the case in the result of conversion in Example (3.1.1), Equation (3.1.4) and in Example (3.1.2), Equation (3.1.11), but not in Example (3.1.3), Equation (3.1.17). The reason why in Equation (3.1.17), the number of initial values is greater than the order of the recurrence equation is that the recurrence equation $n(n+2)a_{n+2}-n^2a_n = 0$ of (3.1.17) is valid only for $n \geq 1$, which means the value of a_2 is undefined reason why we need to add a_2 to the initial values to get all the terms with even indices (a_{2n}). More generally, if we cannot compute the value of some a_q using the recurrence equation, we need to add a_q to the initial values. In order to manage the number of initial values in an IV-RE, we use the function f which generated that IV-HDE and proceed as follows

- We convert the initial conditions of the HDE into the initial values of the IV-RE using the transformation $a_k = y^{(k)}(0)/k!$ where $y(x) = f(x)$;
- We compute the order M of the RE, if M is greater than the number P of initial conditions in the IV-HDE, we add $M - P$ initial values to the IV-RE obtained from the conversion stated above using the function f given;
- We look for the highest positive root R of the term having the highest order in the RE and we also add $R + 1$ initial values to the IV-RE obtained above using the function f given.

Having made those remarks, let us now give the general procedure we apply to get an IV-RE from an IV-HDE satisfied by a holonomic function f.

1. We first expand the HDE of the IV-HDE into the form
$$\sum_{k=0}^{N}\sum_{j=0}^{n_k} \alpha_{j,k} x^j y^{(k)}(x).$$
2. We use the correspondence
$$x^j y^{(k)}(x) \mapsto (n-j+1)_k a_{n+k-j}$$
to transform the HDE into a holonomic recurrence equation REq.
3. We convert the initial conditions of the HDE into initial values of the IV-RE using the formula $a_k = y^{(k)}(0)/k!$.

4. We compute the order M of the recurrence equation REq.
5. We compute the highest positive root R of the term having the highest index in REq.
6. If R does not exist, then we set it to -1.
7. If $M+R+1$ is greater than the number N of initial conditions in the IV-HDE, then we add the following $M+R+1-N$ initial values to the IV-RE obtained in step 3 using the function f

$$a_q = f^{(q)}(0)/q!,\ q \in \{N, ..., M+R\}.$$

8. Otherwise the IV-RE obtained in step 3 is complete.

In order to achieve every step presented above and build a function `Holonomic_RE(f,x)` to compute an IV-RE of a given function `f` of the variable `x`, we define many other functions having different aims. Here are the main functions we use for this purpose in our package with their brief description:

- `Pol_DE_to_RE_Exp(pol,x,k,a,n)`: This function transforms `pol`$\times y^{(k)}(x)$ into its corresponding term in the recurrence equation using the correspondence $x^j y^{(k)}(x) \longrightarrow (n-j+1)_k a_{n+k-j}$, where `pol` is a polynomial of the variable `x`. For instance if `pol` $= x^2+1$, and `k`=1, the ouput of this function is the corresponding term of $(x^2+1)y'(x)$ in the recurrence equation.
  Illustration: `Pol_DE_to_RE_Exp(x`^{2}`+1,x,1,a,n)` returns (n+1)*a[n+1]+(n-1)*a[n-1];

- `List_pol_DE_to_RE(List_pol,x,a,n)`: The role of this function is to compute a recurrence equation of a list of polynomials `List_pol` of the variable `x` using the function `Pol_DE_to_RE_Exp()`. Here we suppose that if `List_pol` is $[p_0(x), p_1(x), ..., p_n(x)]$, then the output is the recurrence equation of the differential equation $p_0(x)y(x) + p_1(x)y'(x) + \cdots + p_n(x)y^{(n)}(x) = 0$.
  For instance, if we consider `List_pol` = [x^2,1,-1], then the result obtained from the function `List_pol_DE_to_RE(List_pol,x,a,n)` is the recurrence equation corresponding to the differential equation $x^2y(x) + y'(x) - y''(x) = 0$.
  Illustration: `List_pol_DE_to_RE([ x`^{2}`,-1,1],x,a,n)` returns (n+1)*(n+2)*a[n+2]-(n+1)*a[n+1]+a[n-2]=0;

- `Order_RE_of_DE(DE,x,F)`: This function takes an HDE `DE` and returns the integer K such that the RE obtained from the HDE `DE` can be written as $\sum_{j=0}^{K} c_j(n)a_{n+j}$ with $c_K(n) \neq 0$.
  For instance if we consider `DE` to be: `diff(F(x),x,2) - diff(F(x),x,1) + x`^{2}`*F(x)=0`, from the illustration above the corresponding RE is (n+1)*(n+2)*a[n+2]-(n+1)*a[n+1]+a[n-2]=0 which is equivalent to (n+3)*(n+4)*a[n+4]-(n+3)*a[n+3]+a[n]=0. From that we can deduce the result $K=4$ of the function `Order_RE_of_DE(DE,x,F)`. By the way let us remark that this function does not compute the RE but uses once more the correspondence (3.2.1) to find out the order K.
  Illustration: `Order_RE_of_DE(diff(F(x),x,2) - diff(F(x),x,1) + x`^{2}`*F(x)=0,x,F)` returns 4;

- `Shift_RE(Req,n,a)`: This function takes a RE `Req` in a_n and shifts the indices of `Req` if there are terms of the form $a_{n-k}, k = 1, 2, ...$ in order to have only terms of the form $a_{n+k}, k = 0, 1, 2, ...$
  Illustration: `Shift_RE((n+1)*(n+2)*a[n+2]-(n+1)*a[n+1]+a[n-2]=0,n,a)` returns (n+3)*(n+4)*a[n+4]-(n+3)*a[n+3]+a[n]=0;

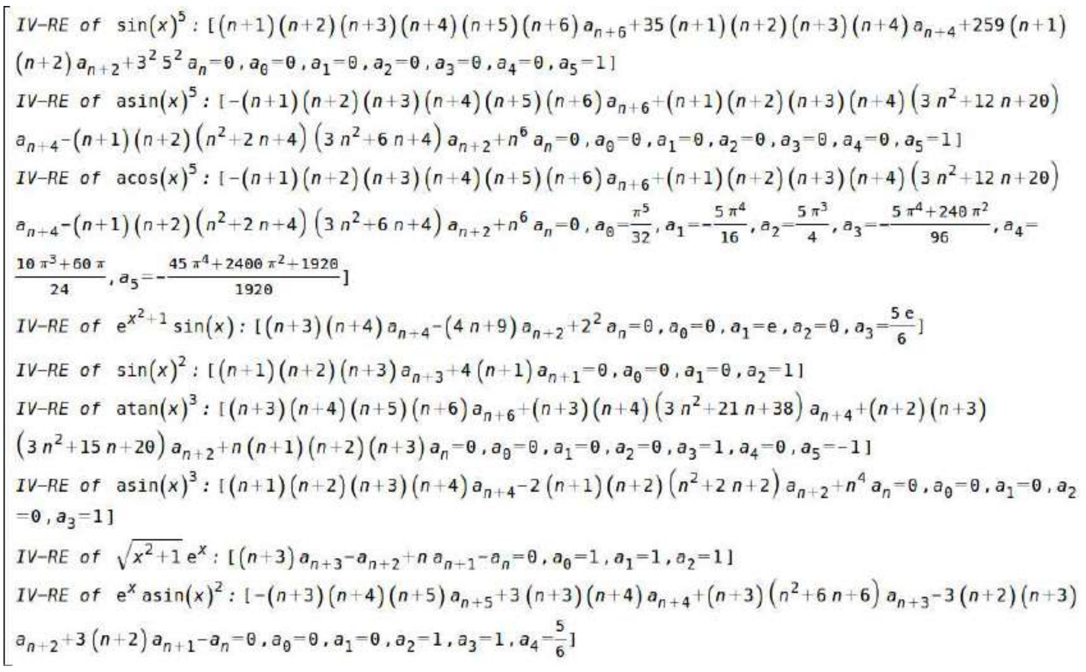
- `DE_to_RE(DE,F,x,a,n)`: This function takes a differential equation `DE` of unknown `F(x)`, uses the functions List_pol_DE_to_RE() and `Pol_DE_to_RE_Exp()` and returns the RE (in a_n) resulting from the conversion of the HDE `DE`.
  Illustration: `DE_to_RE(diff(F(x),x,1)-F(x)=0,F,x,a,n)` returns [(n+1)*a[n+1]-a[n]=0] ;

- `Highest_factor_root(RE,a,n)`: This function takes a recurrence equation `RE` and returns the highest positive integer root of the factor of a_{n+M} where M is the highest index k of the a_{n+k} appearing in the recurrence equation.
  Illustration: `Highest_factor_root((n+1)*(n-2)*(n-1)*a[n+3]-(n+2)*a[n]=0,a,n)` returns 2;

- `Holonomic_RE(f,x,a,n)`: This function takes a function `f`, of the variable `x`, and gives back the IV-RE satisfied by the coefficients a_n in the FPS of `f`. Indeed this function proceeds as follows: it first computes the IV-HDE satisfied by `f` using the function `Holonomic_DE()`, next the HDE is transformed using the function `DE_to_RE` into a RE and initial conditions of the IV-HDE are converted into initial values of the IV-RE. Afterwards the values of `M` and `R` defined in the above algorithm are computed through the functions `Order_RE_of_DE()` and `Highest_factor_root()`. Finally, using `M`, `R` and the function `f`, we complete if necessary the initial values of the IV-RE to get the full IV-RE.
  Illustrations: `Holonomic_RE(exp(x),x,a,n)` returns [(n+1)*a[n+1]-a[n]=0,a[0]=1].

3.3 Some Applications

In this part, we present some results we obtain with our function `Holonomic_RE()` defined for computating an IV-RE of a given function `f`. Below is a snapshot of the computation of the IV-RE of some functions using our function `Holonomic_RE()`.



$$\textit{IV-RE of } \sin(x)^5 : [(n+1)(n+2)(n+3)(n+4)(n+5)(n+6)\,a_{n+6}+35\,(n+1)(n+2)(n+3)(n+4)\,a_{n+4}+259\,(n+1)(n+2)\,a_{n+2}+3^2\,5^2\,a_n=0,\ a_0=0,\ a_1=0,\ a_2=0,\ a_3=0,\ a_4=0,\ a_5=1]$$

$$\textit{IV-RE of } \mathrm{asin}(x)^5 : [-(n+1)(n+2)(n+3)(n+4)(n+5)(n+6)\,a_{n+6}+(n+1)(n+2)(n+3)(n+4)\left(3\,n^2+12\,n+20\right)a_{n+4}-(n+1)(n+2)\left(n^2+2\,n+4\right)\left(3\,n^2+6\,n+4\right)a_{n+2}+n^6\,a_n=0,\ a_0=0,\ a_1=0,\ a_2=0,\ a_3=0,\ a_4=0,\ a_5=1]$$

$$\textit{IV-RE of } \mathrm{acos}(x)^5 : [-(n+1)(n+2)(n+3)(n+4)(n+5)(n+6)\,a_{n+6}+(n+1)(n+2)(n+3)(n+4)\left(3\,n^2+12\,n+20\right)a_{n+4}-(n+1)(n+2)\left(n^2+2\,n+4\right)\left(3\,n^2+6\,n+4\right)a_{n+2}+n^6\,a_n=0,\ a_0=\frac{\pi^5}{32},\ a_1=-\frac{5\,\pi^4}{16},\ a_2=\frac{5\,\pi^3}{4},\ a_3=-\frac{5\,\pi^4+240\,\pi^2}{96},\ a_4=\frac{10\,\pi^3+60\,\pi}{24},\ a_5=-\frac{45\,\pi^4+2400\,\pi^2+1920}{1920}]$$

$$\textit{IV-RE of } \mathrm{e}^{x^2+1}\sin(x) : [(n+3)(n+4)\,a_{n+4}-(4\,n+9)\,a_{n+2}+2^2\,a_n=0,\ a_0=0,\ a_1=\mathrm{e},\ a_2=0,\ a_3=\frac{5\,\mathrm{e}}{6}]$$

$$\textit{IV-RE of } \sin(x)^2 : [(n+1)(n+2)(n+3)\,a_{n+3}+4\,(n+1)\,a_{n+1}=0,\ a_0=0,\ a_1=0,\ a_2=1]$$

$$\textit{IV-RE of } \mathrm{atan}(x)^3 : [(n+3)(n+4)(n+5)(n+6)\,a_{n+6}+(n+3)(n+4)\left(3\,n^2+21\,n+38\right)a_{n+4}+(n+2)(n+3)\left(3\,n^2+15\,n+20\right)a_{n+2}+n\,(n+1)(n+2)(n+3)\,a_n=0,\ a_0=0,\ a_1=0,\ a_2=0,\ a_3=1,\ a_4=0,\ a_5=-1]$$

$$\textit{IV-RE of } \mathrm{asin}(x)^3 : [(n+1)(n+2)(n+3)(n+4)\,a_{n+4}-2\,(n+1)(n+2)\left(n^2+2\,n+2\right)a_{n+2}+n^4\,a_n=0,\ a_0=0,\ a_1=0,\ a_2=0,\ a_3=1]$$

$$\textit{IV-RE of } \sqrt{x^2+1}\,\mathrm{e}^x : [(n+3)\,a_{n+3}-a_{n+2}+n\,a_{n+1}-a_n=0,\ a_0=1,\ a_1=1,\ a_2=1]$$

$$\textit{IV-RE of } \mathrm{e}^x\,\mathrm{asin}(x)^2 : [-(n+3)(n+4)(n+5)\,a_{n+5}+3\,(n+3)(n+4)\,a_{n+4}+(n+3)\left(n^2+6\,n+6\right)a_{n+3}-3\,(n+2)(n+3)\,a_{n+2}+3\,(n+2)\,a_{n+1}-a_n=0,\ a_0=0,\ a_1=0,\ a_2=1,\ a_3=1,\ a_4=\frac{5}{6}]$$

4. Resolution of Recurrence Equations of Hypergeometric Type

The resolution of an IV-RE obtained after conversion of an IV-HDE generated from a function f is one of the crucial and more tricky step in the computation of the FPS of f. Indeed among the family of REs, only REs of hypergeometric type (i.e. REs in a_n such that the quotient $\frac{a_{n+m}}{a_n}$ is a rational function in n for some natural number m) can be handled easily. However in the process of finding the FPS of holonomic functions, we will not always get REs of hypergeometric type. In fact, even for hypergeometric functions (i.e. functions having an FPS $\sum_{n=0}^{\infty} a_n x^n$ such that $\frac{a_{n+1}}{a_n} = \frac{(n+\alpha_1)\cdots(n+\alpha_p)}{(n+\beta_1)\cdots(n+\beta_q)}$ with p, q positive integers and α_i and β_j complex numbers $\forall i \in \{1, ..., p\}$ and $\forall j \in \{1, ..., q\}$), our procedure for getting the HDE does not guarantee a DE leading to a RE of hypergeometric type. For instance the function $f(x) = \exp(x)\sin(x)$ is hypergeometric, it satisfies the HDE $f^{(4)}(x) + 4f(x) = 0$, which yields the RE of hypergeometric type $(n+4)(n+3)(n+2)(n+1)a_{n+4} + 4a_n = 0$. But our procedure gives the HDE $f''(x) - 2f'(x) + 2f(x) = 0$ which yields the RE $(n+2)(n+1)a_{n+2} - 2(n+1)a_{n+1} + 2a_n = 0$ which is not of hypergeometric type. But the Petkovsek-van-Hoeij algorithm [5, Chapter 9] which deals with non-hypergeometric type recurrence equations is not available in the CAS Maxima. In order to solve the IV-RE obtained after conversion of an IV-HDE, in the case we have a RE of hypergeometric type, we use the procedure that we will present in this chapter, otherwise we use the embedded solver `solve_rec()` of REs of Maxima. In this chapter, using some examples, we will show how we solve REs of hypergeometric type, next we will present the general procedure of resolution of such types of REs with some applications.

4.1 Illustrative Examples

In this part, we illustrate the process of solving REs of hypergeometric type by practical resolution going from simple cases to more difficult ones.

4.1.1 Example. Let us consider the IV-RE satisfied by the coefficients a_n in the power series representation $\sum_{n=0}^{\infty} a_n x^n$ of the function $f(x) = \exp(x) + 1$ given by

$$(n+2)a_{n+2} - a_{n+1} = 0, \quad \forall n \geq 0, \quad a_0 = 2,\, a_1 = 1.$$

From that RE, we can get a general formula of the terms a_n with $n \geq 1$. This suggests to define a new sequence $b_n = a_{n+1}$ which satisfies the IV-RE

$$(n+2)b_{n+1} - b_n = 0, \quad \forall n \geq 0, \quad b_0 = 1.$$

From this RE, we deduce that

$$b_n = \frac{b_{n-1}}{n+1} \Rightarrow b_n = \frac{b_0}{(n+1)!} = \frac{1}{(n+1)!}.$$

Hence we deduce the solution of our initial IV-RE given by

$$a_{n+1} = \frac{1}{(n+1)!}, \quad \forall n \geq 0, \text{ or } a_n = \frac{1}{n!}, \quad \forall n \geq 1, \quad \text{and} \quad a_0 = 2. \tag{4.1.1}$$

4.1.2 Example. Let us consider the following IV-RE satisfied by the coefficients a_n in the power series representation $\sum_{n=0}^{\infty} a_n x^n$ of the function $f(x) = \log(1-x^2)$:

$$n^2 a_n - n\,(n+2)\,a_{n+2} = 0, \quad \forall n \geq 1, \quad a_0 = 0,\, a_1 = 0,\, a_2 = -1. \tag{4.1.2}$$

Since $n \geq 1$ in the RE above, we can divide the RE by the common factor n to get

$$n a_n - (n+2)\,a_{n+2} = 0 \quad \forall n \geq 1, \quad a_0 = 0,\, a_1 = 0,\, a_2 = -1. \tag{4.1.3}$$

Since the recurrence equation has only terms a_n and a_{n+2}, then in order to compute a_k we use a_{k-2}, $\forall k \geq 3$. This implies that we compute a_3 from a_1, a_4 from a_2 and in general, we compute $a_{2(n+1)}$ from a_{2n} and $a_{2(n+1)+1}$ from a_{2n+1}. This suggests the definition of the 2 following sequences:

$$a_{n,1} = a_{2n+1} \quad \text{and} \quad a_{n,2} = a_{2n+2}.$$

Note that we do not define $a_{n,2} = a_{2n}$ to keep $n \geq 0$ and not $n \geq 1$. Indeed if we define $a_{n,2} = a_{2n}$, then the recurrence equation of $a_{n,2}$ will be valid only for $n \geq 1$, what we do not want. Substituting n by $2n+1$ (respectively $2n+2$) in (4.1.3) we get the IV-REs satisfied by $a_{n,1}$ (respectively $a_{n,2}$) as follows:

$$\begin{aligned} (2n+3)a_{n+1,1} &= (2n+1)a_{n,1}, \quad \forall n \geq 0, \quad a_{0,1} = 0 \\ (2n+4)a_{n+1,2} &= (2n+2)a_{n,2}, \quad \forall n \geq 0, \quad a_{0,2} = -1, \end{aligned}$$

which leads to

$$\begin{aligned} a_{n,1} &= 0, \quad \forall n \geq 0, \\ a_{n,2} &= \frac{2n}{2n+2} \times \frac{2n-2}{2n} \times \cdots \times \frac{4}{6} \times \frac{2}{4} \times a_{0,2} = -\frac{2}{2n+2} = -\frac{1}{n+1}, \quad \forall n \geq 0. \end{aligned}$$

We deduce the solution of our initial IV-RE given by

$$a_{2n+1} = 0, \quad a_{2n+2} = -\frac{1}{n+1}, \quad \forall n \geq 0, \quad a_0 = 0. \tag{4.1.4}$$

4.1.3 Example. Let us consider the IV-REs satisfied by the coefficients a_n in the FPS of the function $f(x) = \arctan(x^2)$ given by

$$(n+2)\,(n+4)\,a_{n+4} + n\,(n+2)\,a_n = 0, \quad \forall n \geq 0 \quad a_0 = 0,\, a_1 = 0,\, a_2 = 1,\, a_3 = 0,$$

which is equivalent to

$$(n+4)\,a_{n+4} + n a_n = 0, \quad \forall n \geq 0 \quad a_0 = 0,\, a_1 = 0,\, a_2 = 1,\, a_3 = 0. \tag{4.1.5}$$

From the RE above, we compute a_4 from a_0, a_5 from a_1, a_6 from a_2, a_7 from a_3 and in general to compute the term a_k we use the fourth preceding term a_{k-4}, $\forall k \geq 4$. This leads us to the definition of the 4 following sequences:

$$a_{n,0} = a_{4n}, \quad a_{n,1} = a_{4n+1}, \quad a_{n,2} = a_{4n+2} \quad \text{and} \quad a_{n,3} = a_{4n+3}.$$

Substituting n by $4n$ (respectively $4n+1$, $4n+2$ and $4n+3$) in (4.1.5), we obtain the 4 following IV-REs of first order satisfied by $a_{n,0}$ (respectively $a_{n,1}$, $a_{n,2}$ and $a_{n,3}$):

$$\begin{aligned} (4n+4)a_{n+1,0} + 4n a_{n,0} &= 0, \quad \forall n \geq 0, \quad a_{0,0} = 0, \\ (4n+5)a_{n+1,1} + (4n+1)a_{n,1} &= 0, \quad \forall n \geq 0, \quad a_{0,1} = 0, \\ (4n+6)a_{n+1,2} + (4n+2)a_{n,2} &= 0, \quad \forall n \geq 0, \quad a_{0,2} = 1, \\ (4n+7)a_{n+1,3} + (4n+3)a_{n,3} &= 0, \quad \forall n \geq 0, \quad a_{0,3} = 0. \end{aligned}$$

This implies that

$$\begin{aligned}
a_{n,0} &= 0, \quad \forall n \geq 0,\\
a_{n,1} &= 0, \quad \forall n \geq 0,\\
a_{n,2} &= \left(-\frac{2n-1}{2n+1}\right) \times \left(-\frac{2n-3}{2n-1}\right) \times \cdots \times \left(-\frac{3}{5}\right) \times \left(-\frac{1}{3}\right) a_{0,2} = \frac{(-1)^n}{2n+1}, \quad \forall n \geq 0,\\
a_{n,3} &= 0, \quad \forall n \geq 0.
\end{aligned}$$

Hence we deduce that the general solution of our initial IV-RE is given $\forall n \geq 0$ by

$$a_{4n+k} = \begin{cases} \dfrac{(-1)^n}{2n+1} & \text{if} \quad k = 2,\\ 0 & \text{if} \quad k = 0, 1, 3. \end{cases} \tag{4.1.6}$$

4.1.4 Example. Let us consider the IV-RE

$$(3n+8)a_{n+5} - (2n+7)a_{n+2} = 0, \quad \forall n \geq 0, \quad a_0 = a_1 = 1,\, a_2 = a_3 = a_4 = 2. \tag{4.1.7}$$

In the recurrence equation (4.1.7), the lowest index is $n+2$, which suggests that we may consider the sequence $b_n = a_{n+2}$ and look for its IV-RE which is

$$(3n+8)b_{n+3} - (2n+7)b_n = 0, \quad \forall n \geq 0,\, b_0 = b_1 = b_2 = 2. \tag{4.1.8}$$

Next, we define 3 sequences $b_{n,0}$, $b_{n,1}$ and $b_{n,2}$ as

$$b_{n,k} = b_{3n+k}, \quad k = 0, 1, 2.$$

Substituting n by $3n$ (respectively $3n+1$ and $3n+2$) we get the IV-REs satisfied by $b_{n,0}$ (respectively $b_{n,1}$ and $b_{n,2}$)

$$\begin{aligned}
(9n+8)b_{n+1,0} - (6n+7)b_{n,0} = 0 \quad \forall n \geq 0, \quad b_{0,0} = 2,\\
(9n+11)b_{n+1,1} - (6n+9)b_{n,1} = 0 \quad \forall n \geq 0, \quad b_{0,1} = 2,\\
(9n+14)b_{n+1,2} - (6n+11)b_{n,2} = 0 \quad \forall n \geq 0, \quad b_{0,2} = 2.
\end{aligned}$$

This leads to the following equations:

$$\frac{b_{n+1,0}}{b_{n,0}} = \frac{6(n+7/6)}{9(n+8/9)}, \quad \frac{b_{n+1,1}}{b_{n,1}} = \frac{6(n+3/2)}{9(n+11/9)}, \quad \frac{b_{n+1,2}}{b_{n,2}} = \frac{6(n+11/6)}{9(n+14/9)}, \; \forall n \geq 0,\, b_{0,2} = 2,\, b_{0,1} = 2,\, b_{0,2} = 2,$$

which yield

$$\begin{aligned}
b_{n,0} &= 2\frac{2(n-1+7/6)}{3(n-1+8/9)} \times \frac{2(n-2+7/6)}{3(n-2+8/9)} \times \cdots \times \frac{2(1+7/6)}{3(1+8/9)} \times \frac{2(7/6)}{3(8/9)}\\
&= 2\,\frac{2^n}{3^n}\frac{(n-1+7/6)(n-2+7/6)\cdots(1+7/6)(7/6)}{(n-1+8/9)(n-2+8/9)\cdots(1+8/9)(8/9)},\\
b_{n,1} &= 2\frac{2(n-1+3/2)}{3(n-1+11/9)} \times \frac{2(n-2+3/2)}{3(n-2+11/9)} \times \cdots \times \frac{2(1+3/2)}{3(1+11/9)} \times \frac{2(3/2)}{3(11/9)}\\
&= 2\,\frac{2^n}{3^n}\frac{(n-1+3/2)(n-2+3/2)\cdots(1+3/2)(3/2)}{(n-1+11/9)(n-2+11/9)\cdots(1+11/9)(11/9)},\\
b_{n,2} &= 2\frac{2(n-1+11/6)}{3(n-1+14/9)} \times \frac{2(n-2+11/6)}{3(n-2+14/9)} \times \cdots \times \frac{2(1+11/6)}{3(1+14/9)} \times \frac{2(11/6)}{3(14/9)}\\
&= 2\,\frac{2^n}{3^n}\frac{(n-1+11/6)(n-2+11/6)\cdots(1+11/6)(11/6)}{(n-1+14/9)(n-2+14/9)\cdots(1+14/9)(14/9)}.
\end{aligned}$$

Using the Pochhammer symbol $(a)_k$, we deduce that

$$b_{n,0} = 2\frac{2^n(7/6)_n}{3^n(8/9)_n}, \quad b_{n,1} = 2\frac{2^n(3/2)_n}{3^n(11/9)_n}, \quad b_{n,2} = 2\frac{2^n(11/6)_n}{3^n(14/9)_n}.$$

Thus we can deduce that the general solution of our initial IV-RE is given by

$$a_{3n+2} = \frac{2^{n+1}(7/6)_n}{3^n(8/9)_n}, \quad a_{3n+3} = \frac{2^{n+1}(3/2)_n}{3^n(11/9)_n}, \quad a_{3n+4} = \frac{2^{n+1}(11/6)_n}{3^n(14/9)_n}, \quad \forall n \geq 0, \quad a_0 = a_1 = 1. \tag{4.1.9}$$

From the examples above, we can actually see how we handle the resolution of IV-REs of hypergeometric type. Let us move to the general procedure and its implementation in Maxima.

4.2 General Procedure of Resolution of IV-REs of Hypergeometric Type

Before getting into the general procedure, let us give the general form of an IV-RE of hypergeometric type:

$$Q(n)a_{n+m+p} + P(n)a_{n+p} = 0, \;\; a_k = \alpha_k, \;\; k \in \{0, 1, ..., p+m-1\} \text{ with } \alpha_k \text{ given}, \tag{4.2.1}$$

where m and p are positive integers and $Q(n)$, $P(n)$ are polynomials in the variable n. We remark that by substituting n by $mn + l$ in the RE (4.2.1), we obtain

$$Q(mn+l)a_{m(n+1)+p+l} + P(mn+l)a_{mn+p+l} = 0. \tag{4.2.2}$$

Defining $b_{n,l} = a_{mn+p+l}$, with $b_{0,l} = a_{p+l}$, $Q_l(n) = Q(mn+l)$, $P_l(n) = P(mn+l)$, Equation (4.2.2) becomes

$$Q_l(n)b_{n+1,l} + P_l(n)b_{n,l} = 0, \quad \forall n \geq 0, \quad b_{0,l} = a_{p+l}. \tag{4.2.3}$$

For l from 0 to $m-1$, we have m IV-REs of first order

$$Q_l(n)b_{n+1,l} + P_l(n)b_{n,l} = 0, \quad \forall n \geq 0, \quad b_{0,l} = a_{p+l}, \quad l \in \{0, 1, ..., m-1\}. \tag{4.2.4}$$

We can solve all the above IV-REs using the embedded REs solver `solve_rec()` of the CAS Maxima. Let us now move to the algorithm.

Algorithm of resolution of an IV-RE of hypergeometric type on the form (4.2.1)

- We define an index l.
- For l from 0 to $m-1$ we do the following:
  1. We define the sequence $b_{n,l} = a_{mn+p+l}$,
  2. We define polynomials $Q_l(n) = Q(mn+l)$ and $P_l(n) = P(mn+l)$,
  3. Using the function `solve_rec()` of the CAS Maxima, we solve the IV-RE

$$Q_l(n)b_{n+1,l} + P_l(n)b_{n,l} = 0 \quad \forall n \geq 0, \quad b_{0,l} = a_{p+l},$$

4. We set $a_{mn+p+l} = b_{n,l}$,

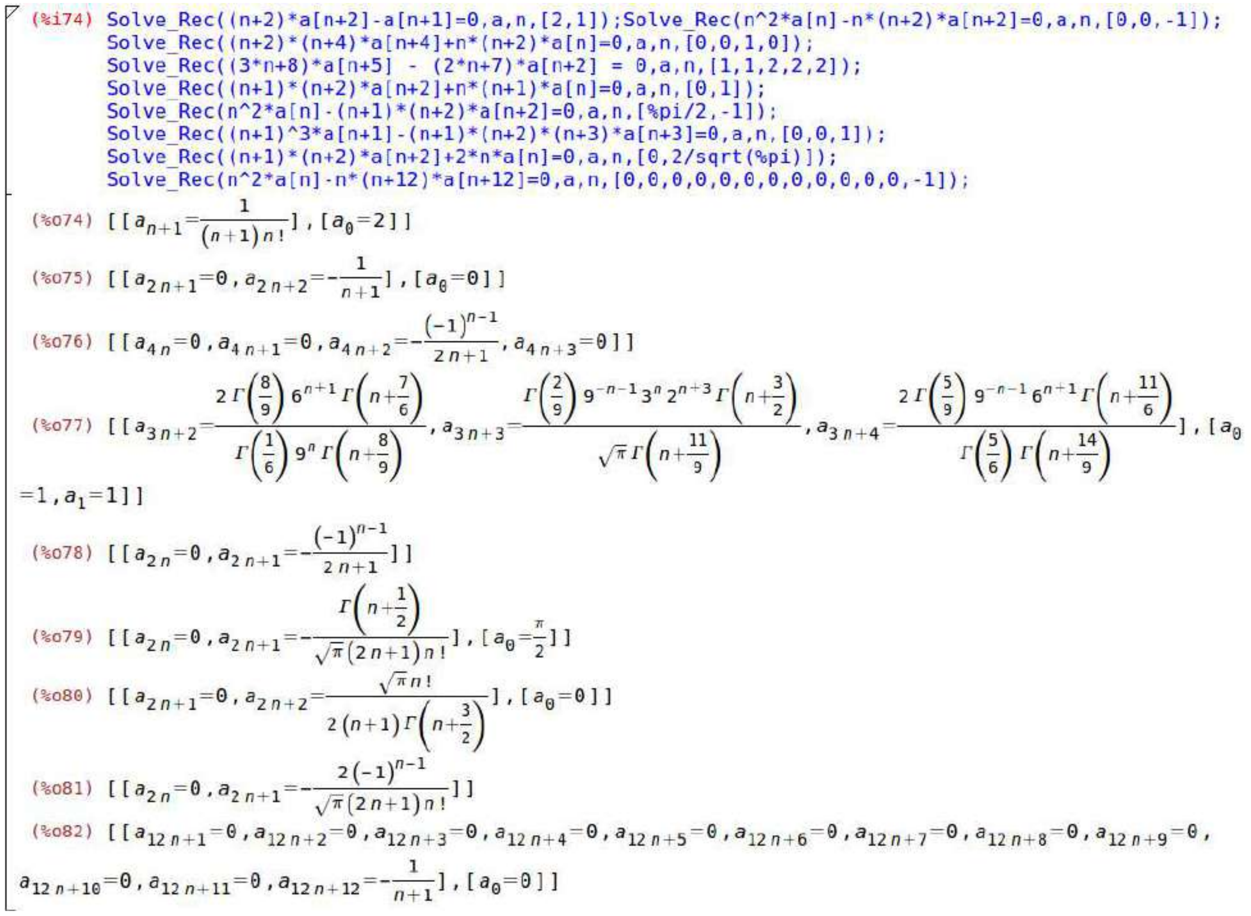
- Finally the general solution of the IV-RE is given by

$$a_{mn+p+l} = b_{n,l}, \quad \forall n \geq 0, \quad l \in \{0, 1, ..., m-1\}, \quad a_k = \alpha_k, \quad k \in \{0, ..., p-1\}. \tag{4.2.5}$$

In our package, in order to solve IV-REs, we defined a function `Solve_Rec()`. This function takes 4 arguments: the RE, the variables a and n occurring in the RE, and a list `L` containing the initial values $a_0, a_1, ..., a_{p+m-1}$. Firstly the function shifts indices if indices of the form $n-k$ with $k = 0, 1, ...$ occur in the RE. Secondly it computes m and p appearing in (4.2.1) and checks whether the number of initial values is enough (i.e. at least $m+p$). Thirdly it checks whether the given RE is of hypergeometric type or not. If the RE is of hypergeometric type then it applies the algorithm above to compute the general solution, otherwise it tries to solve the IV-RE satisfied by $b_n = a_{n+p}$ using the function `solve_rec()` of Maxima. If the `solve_rec()` function is not able to solve the IV-RE, the function returns false otherwise it gives the general solution found adding the p initial values a_k, $k \in \{0, 1, ..., p-1\}$. Let us move now to some applications.

4.3 Some Applications

Using our algorithm, we solve some IV-REs as shown below:



```
(%i74) Solve_Rec((n+2)*a[n+2]-a[n+1]=0,a,n,[2,1]);Solve_Rec(n^2*a[n]-n*(n+2)*a[n+2]=0,a,n,[0,0,-1]);
       Solve_Rec((n+2)*(n+4)*a[n+4]+n*(n+2)*a[n]=0,a,n,[0,0,1,0]);
       Solve_Rec((3*n+8)*a[n+5] - (2*n+7)*a[n+2] = 0,a,n,[1,1,2,2,2]);
       Solve_Rec((n+1)*(n+2)*a[n+2]+n*(n+1)*a[n]=0,a,n,[0,1]);
       Solve_Rec(n^2*a[n]-(n+1)*(n+2)*a[n+2]=0,a,n,[%pi/2,-1]);
       Solve_Rec((n+1)^3*a[n+1]-(n+1)*(n+2)*(n+3)*a[n+3]=0,a,n,[0,0,1]);
       Solve_Rec((n+1)*(n+2)*a[n+2]+2*n*a[n]=0,a,n,[0,2/sqrt(%pi)]);
       Solve_Rec(n^2*a[n]-n*(n+12)*a[n+12]=0,a,n,[0,0,0,0,0,0,0,0,0,0,0,0,-1]);
```

(%o74) $[[a_{n+1}=\frac{1}{(n+1)\,n!}],[a_0=2]]$

(%o75) $[[a_{2n+1}=0,a_{2n+2}=-\frac{1}{n+1}],[a_0=0]]$

(%o76) $[[a_{4n}=0,a_{4n+1}=0,a_{4n+2}=-\frac{(-1)^{n-1}}{2n+1},a_{4n+3}=0]]$

(%o77) $[[a_{3n+2}=\frac{2\,\Gamma\left(\frac{8}{9}\right)6^{n+1}\,\Gamma\left(n+\frac{7}{6}\right)}{\Gamma\left(\frac{1}{6}\right)9^n\,\Gamma\left(n+\frac{8}{9}\right)},a_{3n+3}=\frac{\Gamma\left(\frac{2}{9}\right)9^{-n-1}\,3^n\,2^{n+3}\,\Gamma\left(n+\frac{3}{2}\right)}{\sqrt{\pi}\,\Gamma\left(n+\frac{11}{9}\right)},a_{3n+4}=\frac{2\,\Gamma\left(\frac{5}{9}\right)9^{-n-1}\,6^{n+1}\,\Gamma\left(n+\frac{11}{6}\right)}{\Gamma\left(\frac{5}{6}\right)\Gamma\left(n+\frac{14}{9}\right)}],[a_0$
$=1,a_1=1]]$

(%o78) $[[a_{2n}=0,a_{2n+1}=-\frac{(-1)^{n-1}}{2n+1}]]$

(%o79) $[[a_{2n}=0,a_{2n+1}=-\frac{\Gamma\left(n+\frac{1}{2}\right)}{\sqrt{\pi}\,(2n+1)\,n!}],[a_0=\frac{\pi}{2}]]$

(%o80) $[[a_{2n+1}=0,a_{2n+2}=\frac{\sqrt{\pi}\,n!}{2\,(n+1)\,\Gamma\left(n+\frac{3}{2}\right)}],[a_0=0]]$

(%o81) $[[a_{2n}=0,a_{2n+1}=-\frac{2\,(-1)^{n-1}}{\sqrt{\pi}\,(2n+1)\,n!}]]$

(%o82) $[[a_{12n+1}=0,a_{12n+2}=0,a_{12n+3}=0,a_{12n+4}=0,a_{12n+5}=0,a_{12n+6}=0,a_{12n+7}=0,a_{12n+8}=0,a_{12n+9}=0,$
$a_{12n+10}=0,a_{12n+11}=0,a_{12n+12}=-\frac{1}{n+1}],[a_0=0]]$

5. Formal Power Series of Rational Functions

This chapter presents the procedure for computing the FPS of rational functions. Indeed, for rational functions, the procedure presented in the three first chapters will not always be satisfactory since there are rational functions for which that procedure gives a recurrence equation not of hypergeometric type and which also fails to be solved by the recurrence equation solver of Maxima, like the function $f(x) = (1+x^2)/(1-x^2)$. For this reason, we use another procedure to deal with rational functions. The first section presents some illustrative examples on how we get the FPS of rational functions, the second section gives the general procedure used to compute the FPS of rational functions and the third section shows some applications of this procedure.

5.1 Illustrative Examples

Let us start by giving some examples of conversion of a rational function into an FPS starting from the simplest cases to more difficult ones. By the way let us recall that the simplest conversion occurs when the rational function is a polynomial function in which case it is itself its FPS. Now let us consider cases where the rational function is not a polynomial.

5.1.1 Example. Let us consider the rational function $f(x) = \frac{1}{1-x}$.
If we compute the $n-$th derivative of $f(x)$ at the point $x = 0$, we will get $n!$, meaning that the FPS of the function f is $\sum_{n=0}^{\infty} x^n$. But we could have also used the following formula

$$FPS\left(\frac{1}{(1-x)^k}\right) = \sum_{n=0}^{\infty} \binom{n+k-1}{n} x^n, \quad k = 1, 2, ..., \tag{5.1.1}$$

from which we deduce the FPS of $f(x)$

$$\frac{1}{1-x} = \sum_{n=0}^{\infty} x^n.$$

5.1.2 Example. Let us consider the rational function $f(x) = \frac{2\,(x+3)}{(x+1)^2}$.
We compute the partial fraction decomposition (PFD) of the function f, i.e. we decompose the function f on the form

$$\sum_{n=1}^{m} \left(\frac{d_{n1}}{x-r_n} + \frac{d_{n2}}{(x-r_n)^2} + \cdots + \frac{d_{np_n}}{(x-r_n)^{p_n}} \right), \; d_{ni},\, r_n \in \mathbb{C}, \; i \in \{1, 2, ...p_n\}, \; p_n \in \mathbb{N}.$$

Since the denominator of $f(x)$ is $(x+1)^2$, this suggests a decomposition of the form

$$f(x) = \frac{a}{x+1} + \frac{b}{(x+1)^2}.$$

This equation leads to

$$\frac{2x+6}{(1+x)^2} = \frac{ax+a+b}{(1+x)^2},$$

which yields $a = 2$ and $b = 4$. Thus we get

$$f(x) = \frac{2}{x+1} + \frac{4}{(x+1)^2}.$$

Next we use the formula (5.1.1) to get the FPS of each summand in the above expression of $f(x)$ as

$$\frac{2}{x+1} = 2\sum_{n=0}^{\infty}\binom{n}{n}(-x)^n \quad \text{and} \quad \frac{4}{(x+1)^2} = 4\sum_{n=0}^{\infty}\binom{n+1}{n}(-x)^n.$$

From those expressions, we deduce the FPS of $f(x)$ as

$$\frac{2\ (x+3)}{(x+1)^2} = \sum_{n=0}^{\infty}(-1)^n(4n+6)x^n.$$

5.1.3 Example. Let us consider the rational function $f(x) = \frac{x+1}{x^2+x+1}$.
We start by factorising the denominator of $f(x)$ over the field $\mathbb{C}$ as follows:

$$x^2+x+1 = \left(x - \frac{\sqrt{3}i-1}{2}\right)\left(x + \frac{\sqrt{3}i+1}{2}\right).$$

Next we look for the partial fraction decomposition of $f(x)$ of the form

$$f(x) = \frac{a}{x - \frac{\sqrt{3}i-1}{2}} + \frac{b}{x + \frac{\sqrt{3}i+1}{2}}.$$

This equation leads us to

$$\frac{x+1}{x^2+x+1} = \frac{(2b+2a)x + (1-\sqrt{3}i)b + (\sqrt{3}i+1)a}{2(x^2+x+1)},$$

which is equivalent to the following system of equations

$$\begin{cases} b+a=1 \\ (1-\sqrt{3}i)b + (\sqrt{3}i+1)a = 2. \end{cases}$$

The solution of this system is $a = \frac{3-\sqrt{3}i}{6}$ and $b = \frac{\sqrt{3}i+3}{6}$, hence we deduce the PFD of $f(x)$ as

$$f(x) = \frac{3-\sqrt{3}i}{3(2x-(\sqrt{3}i-1))} + \frac{3+\sqrt{3}i}{3(2x+\sqrt{3}i+1)}.$$

Now we put each summand in the form $\frac{\alpha}{1-\beta x}$ as follows:

$$\frac{3-\sqrt{3}i}{3(2x-(\sqrt{3}i-1))} = \frac{\alpha_1}{1-\beta_1 x}, \quad \text{with} \quad \alpha_1 = -\frac{3-\sqrt{3}i}{3(\sqrt{3}i-1)},\ \beta_1 = \frac{2}{\sqrt{3}i-1},$$
$$\frac{3+\sqrt{3}i}{3(2x+\sqrt{3}i+1)} = \frac{\alpha_2}{1-\beta_2 x}, \quad \text{with} \quad \alpha_2 = \frac{3+\sqrt{3}i}{3(\sqrt{3}i+1)},\ \beta_2 = -\frac{2}{\sqrt{3}i+1}.$$

With $f(x)$ written in the form

$$f(x) = \frac{\alpha_1}{1-\beta_1 x} + \frac{\alpha_2}{1-\beta_2 x},$$

we use the formula (5.1.1) and deduce the FPS of $f(x)$ as

$$f(x) = \alpha_1\sum_{n=0}^{\infty}\beta_1^n x^n + \alpha_2\sum_{n=0}^{\infty}\beta_2^n x^n,$$

with

$$\alpha_1 = -\frac{3-\sqrt{3}i}{3(\sqrt{3}i-1)},\ \beta_1 = \frac{2}{\sqrt{3}i-1},\ \alpha_2 = \frac{3+\sqrt{3}i}{3(\sqrt{3}i+1)},\ \beta_2 = -\frac{2}{\sqrt{3}i+1}.$$

5.1.4 Example. Let us consider the rational function $f(x) = \frac{x^5+x^2+1}{-x^4+5x^2-4}$.
Since the degree of the numerator is greater than the degree of the denominator, we start by doing the polynomial division with remainder of the numerator by the denominator of $f(x)$ and rewrite $f(x)$ as

$$f(x) = -x - \frac{5x^3+x^2-4x+1}{x^4-5x^2+4}.$$

Next we factorise the denominator of the fraction in $f(x)$ into a product of polynomials of degree 1 and rewrite $f(x)$ as:

$$f(x) = -x - \frac{5x^3+x^2-4x+1}{(x-2)(x+2)(x-1)(1+x)}.$$

Now we compute the partial fraction decomposition of the above fraction in $f(x)$. The complete linear factorisation of the denominator in this fraction induces that its PFD is of the form

$$\frac{5x^3+x^2-4x+1}{(x-2)(x+2)(x-1)(1+x)} = \frac{a}{x-2} + \frac{b}{x+2} + \frac{c}{x-1} + \frac{d}{x+1}.$$

The complete expansion of both sides of this equation leads to

$$\frac{5x^3+x^2-4x+1}{x^4-5x^2+4} = \frac{1}{x^4-5x^2+4}[(d+c+b+a)\,x^3 + (-d+c-2b+2a)\,x^2 \\ + (-4d-4c-b-a)\,x + 4d - 4c + 2b - 2a],$$

which yields the system of equations

$$\begin{cases} a+b+c+d=5 \\ 2a-2b+c-d=1 \\ a+b+4c+4d=4 \\ 2a-2b+4c-4d=-1. \end{cases}$$

The solution of this system is $a = 37/12$, $b = 9/4$, $c = -1/2$ and $d = 1/6$. Hence we obtain the full PFD of $f(x)$ as

$$f(x) = -x - \frac{9}{4\,(x+2)} - \frac{1}{6\,(x+1)} + \frac{1}{2\,(x-1)} - \frac{37}{12\,(x-2)},$$

which can be rewritten as

$$f(x) = -x - \frac{9/8}{1-(-x/2)} - \frac{1/6}{1-(-x)} - \frac{1/2}{1-x} + \frac{37/24}{1-(x/2)}.$$

Using the formula (5.1.1), we deduce the FPS of $f(x)$ as

$$\frac{x^5+x^2+1}{-x^4+5x^2-4} = -x - \frac{9}{8}\sum_{n=0}^{\infty}\frac{(-1)^n x^n}{2^n} + \frac{37}{24}\sum_{n=0}^{\infty}\frac{x^n}{2^n} - \frac{1}{6}\sum_{n=0}^{\infty}(-1)^n x^n - \frac{1}{2}\sum_{n=0}^{\infty}x^n,$$

which can be rewritten as

$$\frac{x^5+x^2+1}{-x^4+5x^2-4} = -x - \sum_{n=0}^{\infty}\left(\frac{9(-1)^n}{2^{n+3}} - \frac{37}{3(2^{n+3})} + \frac{(-1)^n}{6} + \frac{1}{2}\right)x^n.$$

Having given those examples, we can move to the general procedure for computing the FPS of rational functions.

5.2 General Procedure for Computing Formal Power Series of Rational Functions

The general procedure for converting a rational function $f(x) = \frac{N(x)}{Q(x)}$ (where $N(x)$ and $Q(x)$ are polynomials) into an FPS is as follows:

- We first do the polynomial division with remainder of $N(x)$ by $Q(x)$ and we call $P(x)$ the quotient and $R(x)$ the remainder such that
$$f(x) = P(x) + \frac{R(x)}{Q(x)} \quad \text{with} \quad \deg(R(x)) < \deg(Q(x));$$
- We do the complete linear factorisation of $Q(x)$ over the field $\mathbb{C}$, such that
$$Q(x) = (x-\alpha_1)^{p_1}(x-\alpha_2)^{p_2}\cdots(x-\alpha_m)^{p_m} \quad \text{with } \alpha_i \in \mathbb{C}, \text{ and } p_i \in \mathbb{N}, \ \forall i \in \{1, ..., m\};$$
- We write $\frac{R(x)}{Q(x)}$ as
$$\frac{R(x)}{Q(x)} = \sum_{k=1}^{m}\left(\frac{d_{k1}}{x-\alpha_k} + \frac{d_{k2}}{(x-\alpha_k)^2} + \cdots + \frac{d_{kp_k}}{(x-\alpha_k)^{p_k}}\right) \tag{5.2.1}$$
where d_{ki}, $i \in \{1, 2, ..., p_k\}$ are unknown real or complex numbers;
- Writing the right-hand side of Equation (5.2.1) in normal form and equating the numerators, we get
$$R(x) = G_0 + G_1 x + \cdots + G_l x^l \tag{5.2.2}$$
where $l \in \mathbb{N}$, and G_i, $\forall i \in \{0, 1..., l\}$ are functions of d_{ki}, $k \in \{1, 2, ..., m\}$ and $i \in \{1, 2, ..., p_k\}$;
- We equate coefficients of x in Equation (5.2.2) and obtain a system of equations of the form
$$\begin{cases} G_0 = r_0 \\ G_1 = r_1 \\ \ldots \\ G_{l-1} = r_{l-1} \\ G_l = r_l, \end{cases} \tag{5.2.3}$$
where r_i, i from 0 to l is the coefficient of x^i in the polynomial $R(x)$;
- We solve the system (5.2.3) and get the values of d_{ki}, i from 1 to p_k and k from 1 to m, this yields the PFD of $f(x)$ as
$$f(x) = P(x) + \sum_{k=1}^{m}\left(\frac{d_{k1}}{x-\alpha_k} + \frac{d_{k2}}{(x-\alpha_k)^2} + \cdots + \frac{d_{kp_k}}{(x-\alpha_k)^{p_k}}\right);$$
- We rewrite $f(x)$ as
$$f(x) = P(x) + \sum_{k=1}^{m}\left(-\frac{d_{k1}/\alpha_k}{1-(x/\alpha_k)} + \frac{d_{k2}/(\alpha_k)^2}{(1-(x/\alpha_k))^2} + \cdots + (-1)^{p_k}\frac{d_{kp_k}/(\alpha_k)^{p_k}}{(1-(x/\alpha_k))^{p_k}}\right);$$

- We deduce using the formula (5.1.1) the FPS of $f(x)$ as

$$P(x)+\sum_{n=0}^{\infty}\left(\sum_{k=1}^{m}\left(\frac{1}{\alpha_k^n}\right)\left(-\frac{d_{k1}}{\alpha_k}+(n+1)\frac{d_{k2}}{(\alpha_k)^2}+\cdots+(-1)^{p_k}\binom{n+p_k-1}{n}\frac{d_{kp_k}}{(\alpha_k)^{p_k}}\right)\right)x^n.$$

In order to achieve all the steps presented above and obtain the FPS of a given rational function f, we define the following functions:

1. `Complex_Fact(Exp,var)`: this function takes a polynomial `Exp` and returns its full linear factorisation with respect to `var` over the field $\mathbb{C}$ of complex numbers if possible otherwise returns false.
   Illustration: `Complex_Fact`$(x^6-11\,x^5+46\,x^4-92\,x^3+99\,x^2-81\,x+54,x)$ returns $(x-3)^3\;(x-2)\;(x-i)\;(x+i)$.

2. `Part_Frac(f,var)`: this function takes a rational function `f` and returns its full PFD in the form of a list of 2 elements. The first element is the polynomial part and the second element is the purely rational part in the PFD of `f`.
   Illustration: `Part_Frac`$(\frac{x^4+2\,x^2+5\,x+6}{x^2+2\,x+1},x)$ returns $[x^2-2\,x+5,\frac{4}{(x+1)^2}-\frac{3}{x+1}]$.

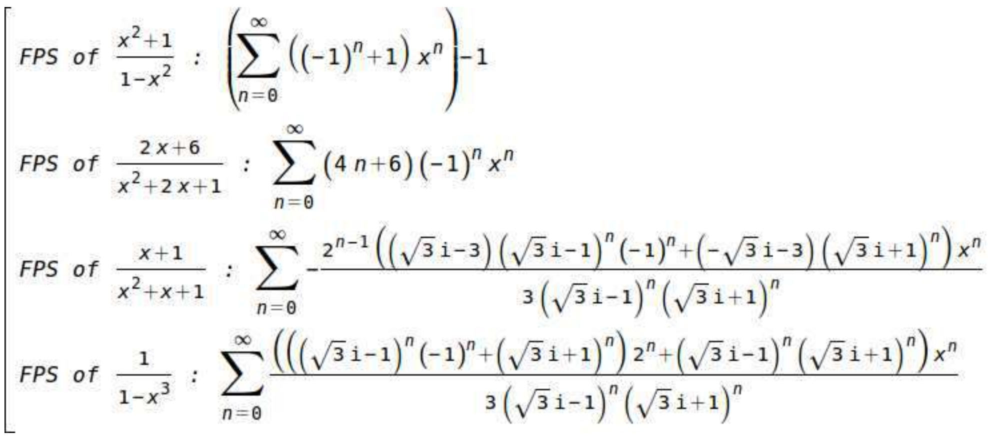
3. `Frac_to_FPS_coef(Fract,var,m)`: this function takes a rational function `Fract` on the form $\frac{a}{(cx+d)^k}$ and returns the formula of the coefficient a_n appearing in the FPS $\sum_{n=0}^{\infty}a_nx^n$ of `Fract`.
   Illustration: `Frac_to_FPS_coef`$(\frac{1}{(1+x)^2},x,n)$ returns $(n+1)\;(-1)^{n+2}$.

4. `Fract_FPS(g,var)`: this function takes a function g of the main variable `var` and returns its FPS if g is rational and false otherwise.

Having presented the above general procedure for computing FPS of rational functions, let us give some applications.

5.3 Some Applications

In the snapshot below, we present FPS of some rational functions obtained using the function `Fract_FPS()` of our package.



FPS of $\frac{x^2+1}{1-x^2}$: $\left(\sum_{n=0}^{\infty}\left((-1)^n+1\right)x^n\right)-1$

FPS of $\frac{2\,x+6}{x^2+2\,x+1}$: $\sum_{n=0}^{\infty}(4\,n+6)\,(-1)^n\,x^n$

FPS of $\frac{x+1}{x^2+x+1}$: $\sum_{n=0}^{\infty}-\frac{2^{n-1}\left(\left(\sqrt{3}\,\mathrm{i}-3\right)\left(\sqrt{3}\,\mathrm{i}-1\right)^n(-1)^n+\left(-\sqrt{3}\,\mathrm{i}-3\right)\left(\sqrt{3}\,\mathrm{i}+1\right)^n\right)x^n}{3\left(\sqrt{3}\,\mathrm{i}-1\right)^n\left(\sqrt{3}\,\mathrm{i}+1\right)^n}$

FPS of $\frac{1}{1-x^3}$: $\sum_{n=0}^{\infty}\frac{\left(\left(\left(\sqrt{3}\,\mathrm{i}-1\right)^n(-1)^n+\left(\sqrt{3}\,\mathrm{i}+1\right)^n\right)2^n+\left(\sqrt{3}\,\mathrm{i}-1\right)^n\left(\sqrt{3}\,\mathrm{i}+1\right)^n\right)x^n}{3\left(\sqrt{3}\,\mathrm{i}-1\right)^n\left(\sqrt{3}\,\mathrm{i}+1\right)^n}$

6. The Formal Power Series Algorithm

In the previous chapters, we focused on the presentation and implementation of the different steps of the general procedure for computing FPS of holonomic functions or rational functions. In this chapter, we present the general algorithm presented in [3] (see also [4]) for computing the FPS of a rational function or a holonomic function and its implementation in Maxima. We also present applications, add-ons, and finish with some general remarks about the package we built in Maxima.

6.1 General Procedure for Computing Formal Power Series

The general algorithm for computing FPS of a given function $f(x)$ that we implemented in Maxima can be found in [3, Section 3, Algorithm 3.1, Page 589] and is given as follows:

1. We set 2 variables: N_1 = `MAX_ORDER_DERIVATIVE`, the maximal order of the HDE generated for f, N_2=`MAX_ORDER_DERIVATIVE_RAT`, the number of derivatives we compute to get a rational function.
2. If $f(x)$ is rational then we compute its FPS using the algorithm in Chapter 5. Otherwise, we compute the first $N_2 - 1$ derivatives $f'(x), f''(x), ..., f^{(N_2-1)}(x)$ of the function $f(x)$. If there exists $k \in \{1, ..., N_2 - 1\}$ such that the $k-$th derivative of $f(x)$ is a rational function, then we proceed as follows:
    - We compute the FPS of $f^{(k)}(x)$ using the algorithm presented in Chapter 5;
    - We integrate the FPS of $f^{(k)}(x)$ k times and add $f(0) + f'(0)x + \cdots + \frac{f^{(k-1)}(0)}{(k-1)!}x^{k-1}$ to get the FPS of $f(x)$;
3. Otherwise if $f(x)$ is not rational and none of it first $N_2 - 1$ derivatives is rational, then we search an HDE satisfied by the function $f(x)$ using the algorithm presented in Chapter 2.
4. We convert the HDE obtained in step 3 into an IV-RE using the algorithm presented in Chapter 3.
5. We try to solve the IV-RE obtained in step 4 using the algorithm presented in Chapter 4.
6. If the resolution is successful, we have the coefficients of our FPS. Otherwise, we print out the IV-RE.

In our package, there are 3 main functions to compute the FPS of a given function.

- The function `HOLO_FPS(f,x)`: this function takes a function `f` and its main variable `x` and computes its FPS using the following steps: it computes the IV-HDE satisfied by `f`, then converts it to an IV-RE (using the function `Holonomic_RE()`), solves this IV-RE (using the function `Solve_Rec()`), and returns the FPS of `f`.
- The function `RAT_FPS(f,x)`: this function takes a function `f` and its main variable `x`. Using the procedure for rational functions presented in Chapter 5, it computes the FPS of `f` if `f` or one of the first `MAX_ORDER_DERIVATIVE_RAT`-1 derivatives of `f` is a rational function with respect to `x`, or it returns `f` otherwise.

- The function `FPS(f,x)`: this function applies the algorithm presented above to compute the FPS of a function `f`. If the given function `f` does not have a rational derivative and is a sum of functions, then this function computes the FPS of each summand in the expression of `f(x)` and adds those FPS to get the FPS of the function `f`.

We also extended this work and implemented the computation of formal Laurent series (series of the form $\sum_{n=n_0}^{\infty} a_n x^n$ with $n_0 \in \mathbb{Z}$) and Puiseux series (series of the form $\sum_{n=n_0}^{\infty} a_n x^{n/q}$ for some $q \in \mathbb{N}$ and $n_0 \in \mathbb{Z}$). And since this is beyond the scope of this essay topic, we will just mention results of those forms without presenting the full algorithm of how those computations of formal Laurent and Puiseux series are done. The computation of those series (Laurent and Puiseux) is done using the function `HLP_FPS(f,x)` which takes 2 arguments: the function `f` and its main variable `x`.
In order to allow the users to have some flexibility and easily adapt the outputs according to their need, we defined many global variables accessible and modifiable by the user. Those variables are listed below with their descriptions.

- `MAX_ORDER_DERIVATIVE`: this variable represents the maximal order of an HDE generated from a given function `f`, its default value is 4.
- `MAX_ORDER_DERIVATIVE_RAT`: this variable (equal to 4 by default) represents the maximum number of derivatives of the function `f` we have to compute to get one of them rational. This variable is used in the function `FPS()` and also in the function `RAT_FPS()`.
- `COMPLEX_COEFF`: this is a boolean variable (equal to false by default) which gives the user the possibility to have FPS with complex coefficients or FPS with real coefficients. Indeed, if this variable is false, then the coefficients in the FPS are replaced by their real parts. This is justified by the fact that for a real function $f(x)$, the FPS must have real coefficients (since the coefficient a_n appearing in the FPS is nothing but $\frac{f^{(n)}(0)}{n!}$). If this variable is set to true, no conversion of complex to real coefficients is done.
- `Initial_values_DE`: this boolean variable (equal to true by default) influences the output of the function `Holonomic_DE()`. In fact, if this variable is true, then the function `Holonomic_DE()` returns an IV-HDE, otherwise it returns an HDE without initial conditions.
- `Initial_values_RE`: this boolean variable (equal to true by default) influences the output of the function `Holonomic_RE()`. In fact, if this variable is true, then the function `Holonomic_RE()` returns an IV-RE, otherwise it returns a holonomic recurrence equation without initial values.
- `Factoring_Coef_DE`: this boolean variable (equal to true by default) influences the output of the function `Holonomic_DE()`. If this variable is true, then the polynomial coefficients in the HDE are factorised, otherwise those coefficients are not factorised. Indeed, it is always nice to see the HDE with factorised coefficients. However for HDE with high degree polynomial coefficients, with this factorisation, the function `Holonomic_DE()` may need a lot of time before returning the HDE reason why we define such a variable.
- `Factoring_Coef_RE`: this boolean variable (equal to true by default) influences the output of the function `Holonomic_RE()`. If this variable is true, then the polynomial coefficients in the holonomic recurrence equation are factorised, otherwise those coefficients are not factorised. And this variable is defined for the same reason as the variable `Factoring_Coef_DE`.

- `Laurent_Puiseux_Series`: this is a boolean variable (equal to false by default) allowing the user to decide whether he or she wants also Laurent series and Puiseux series or not. Indeed, if this variable is true, then the series obtained as output can be a Laurent or Puiseux series, otherwise outputs are formal power series.
- `Info_Level_FPS`: This is an integer variable (equal to 0 by default) representing the level of information printed out during the execution of functions in the package. This variable can take values from $0, 1, 2, ..., 5$. The larger that variable is, the more informations are printed out during the execution of functions.

Let us present some applications of our algorithm.

6.2 Some Applications

In this part, we present FPS of some functions we got using functions in our package.
Let us start by presenting FPS of some holonomic functions we got using the function `HOLO_FPS()`.

$$\textit{FPS of } \cos(x)^3 : \sum_{n=0}^{\infty} \frac{(-1)^{n/2}\left((-1)^n+1\right)\left(3^n+3\right)x^n}{8\,n!}$$

$$\textit{FPS of } \operatorname{atan}(x) : \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{2n+1}$$

$$\textit{FPS of } \operatorname{erf}(x) : \sum_{n=0}^{\infty} \frac{2(-1)^n x^{2n+1}}{\sqrt{\pi}(2n+1)n!}$$

$$\textit{FPS of } \sqrt{x^2+1} : \sum_{n=0}^{\infty} -\frac{(-1)^n (2n)!\,x^{2n}}{(2n-1)2^{2n}n!^2}$$

$$\textit{FPS of } \sin\left(\sqrt{x+1}\right) : \left[\left(4n^2+12n+8\right)a_{n+2}+\left(4n^2+6n+2\right)a_{n+1}+a_n=0, a_0=\sin(1), a_1=\frac{\cos(1)}{2}\right]$$

$$\textit{FPS of } e^x\sin(x) : \sum_{n=0}^{\infty} \frac{2^{n/2}\sin\left(\frac{\pi n}{4}\right)x^n}{n!}$$

$$\textit{FPS of } e^x-2e^{-\frac{x}{2}}\cos\left(\frac{\sqrt{3}x}{2}+\frac{\pi}{3}\right) : \sum_{n=0}^{\infty} \frac{3^{1-3n}x^{3n+1}}{\left(\frac{1}{3}\right)_n\left(\frac{2}{3}\right)_n(3n+1)n!}$$

$$\textit{FPS of } \frac{\log\left(\frac{x+1}{1-x}\right)}{2}-\operatorname{atan}(x) : \sum_{n=0}^{\infty} \frac{2x^{4n+3}}{4n+3}$$

$$\textit{FPS of } \sin(x)+\operatorname{atan}(x) : \left[\left(-n^4-26n^3-251n^2-1066n-1680\right)a_{n+8}+\left(7n^4+104n^3+502n^2+755n-150\right)a_{n+6}+\left(9n^4+118n^3+574n^2+1205n+884\right)a_{n+4}+\left(n^4+4n^3+14n^2+57n+74\right)a_{n+2}+\left(n^2+n\right)a_n=0, a_0=0, a_1=2, a_2=0, a_3=-\frac{1}{2}, a_4=0, a_5=\frac{5}{24}, a_6=0, a_7=-\frac{103}{720}\right]$$

Next, we present FPS of some functions having a rational derivative like $\arctan(x)$, $x^2\log(x^2+2)$ computed using the function `RAT_FPS()`.

$$\textit{FPS of } \operatorname{atan}(x): \sum_{n=0}^{\infty} \frac{\left((-1)^n+1\right)x^{n+1}}{2\,(n+1)\,(-1)^{\frac{3n}{2}}}$$

$$\textit{FPS of } x\log\left(x^2+2\right): \left(\sum_{n=0}^{\infty} \frac{2^{-\frac{n}{2}-\frac{1}{2}}\left((-1)^n-1\right)(-1)^n \sin\left(\frac{\pi n}{2}\right)x^{n+2}}{n+1}\right)+\log(2)\,x$$

$$\textit{FPS of } \left(x^2+1\right)\log\left(\frac{x+1}{1-x}\right)+2: \left(\sum_{n=0}^{\infty} \frac{\left(\left(2n^2+8n+8\right)(-1)^n+2n^2+8n+8\right)x^{n+3}}{(n+1)(n+2)(n+3)}\right)+2x+2$$

$$\textit{FPS of } \log(x+1)+\operatorname{atan}(x): \sum_{n=0}^{\infty} \frac{\left(2\,(-1)^{\frac{5n}{2}}+(-1)^n+1\right)x^{n+1}}{2\,(n+1)\,(-1)^{\frac{3n}{2}}}$$

$$\textit{FPS of } \frac{x^2+1}{x^3-2x^2+1}: \sum_{n=0}^{\infty} \frac{\left(\left(\left(\sqrt{5}-1\right)\left(\sqrt{5}+1\right)^n(-1)^n-\left(\sqrt{5}-1\right)^n\sqrt{5}-\left(\sqrt{5}-1\right)^n\right)2^n+4\left(\sqrt{5}-1\right)^n\left(\sqrt{5}+1\right)^n\right)x^n}{2\left(\sqrt{5}-1\right)^n\left(\sqrt{5}+1\right)^n}$$

We also present FPS of some functions using the general function `FPS()`. In this case, since we want to have only formal power series, not Laurent or Puiseux series, the variable `Laurent_Puiseux_Series` is set to false. Below are some results we obtain in this case.

$$\textit{FPS of } \operatorname{asin}(x)^2+\frac{x+2}{x^3+2x^2-1}: \left(\sum_{n=0}^{\infty} \frac{2^{2n}\,n!^2\,x^{2n+2}}{(n+1)(2n+1)(2n)!}\right)+$$

$$\sum_{n=0}^{\infty} \frac{\left(\left(\left(3\left(\sqrt{5}-1\right)^n\sqrt{5}-5\left(\sqrt{5}-1\right)^n\right)(-1)^n+\left(-3\sqrt{5}-5\right)\left(\sqrt{5}+1\right)^n\right)2^n-10\left(\sqrt{5}-1\right)^n\left(\sqrt{5}+1\right)^n(-1)^n\right)x^n}{10\left(\sqrt{5}-1\right)^n\left(\sqrt{5}+1\right)^n}$$

$$\textit{FPS of } e^x-2e^{-\frac{x}{2}}\cos\left(\frac{\sqrt{3}\,x}{2}+\frac{\pi}{3}\right): \left(\sum_{n=0}^{\infty} -\frac{(-1)^n\left(\sqrt{3}\sin\left(\frac{\pi n}{3}\right)+\cos\left(\frac{\pi n}{3}\right)\right)x^n}{n!}\right)+\sum_{n=0}^{\infty}\frac{x^n}{n!}$$

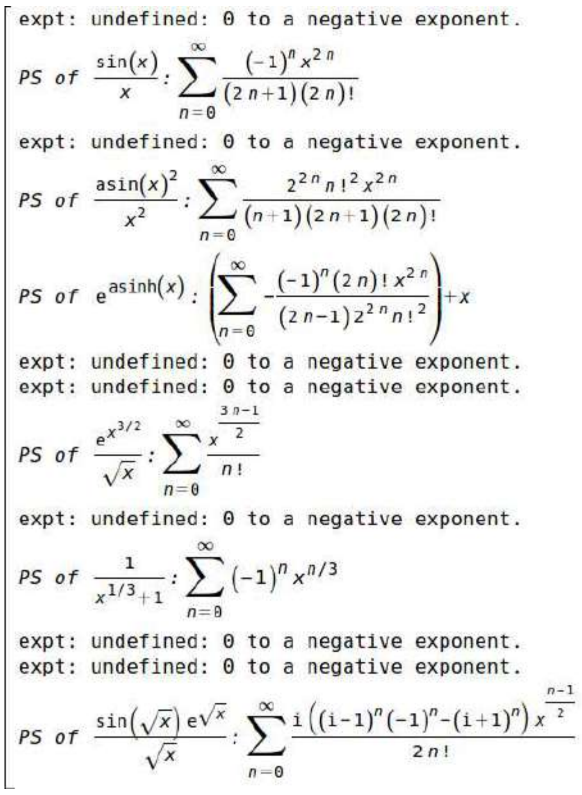
$$\textit{FPS of } \frac{\log\left(\frac{x+1}{1-x}\right)}{2}-\operatorname{atan}(x): \sum_{n=0}^{\infty} \frac{\left((-1)^{\frac{5n}{2}}+(-1)^{\frac{3n}{2}}-(-1)^n-1\right)x^{n+1}}{2\,(n+1)\,(-1)^{\frac{3n}{2}}}$$

$$\textit{FPS of } \sin(x)+\operatorname{atan}(x): \left(\sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{(2n+1)(2n)!}\right)+\sum_{n=0}^{\infty} \frac{\left((-1)^n+1\right)x^{n+1}}{2\,(n+1)\,(-1)^{\frac{3n}{2}}}$$

$$\textit{FPS of } \left(x^2+1\right)\left(\sin(x)+\operatorname{atan}(x)\right): \left(\sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+3}}{(2n+1)(2n)!}\right)+\left(\sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{(2n+1)(2n)!}\right)+\left(\sum_{n=0}^{\infty} \frac{\left((-1)^n+1\right)x^{n+3}}{2\,(n+1)\,(-1)^{\frac{3n}{2}}}\right)+$$

$$\sum_{n=0}^{\infty} \frac{\left((-1)^n+1\right)x^{n+1}}{2\,(n+1)\,(-1)^{\frac{3n}{2}}}$$

Let us now show some results of computation of Laurent and Puiseux series using the function `HLP_FPS()` and `FPS()`. First, using the function `HLP_FPS()`, we get the outputs below.



```
expt: undefined: 0 to a negative exponent.
```
PS of $\frac{\sin(x)}{x}: \sum_{n=0}^{\infty}\frac{(-1)^n x^{2n}}{(2n+1)(2n)!}$

```
expt: undefined: 0 to a negative exponent.
```
PS of $\frac{\operatorname{asin}(x)^2}{x^2}: \sum_{n=0}^{\infty}\frac{2^{2n} n!^2 x^{2n}}{(n+1)(2n+1)(2n)!}$

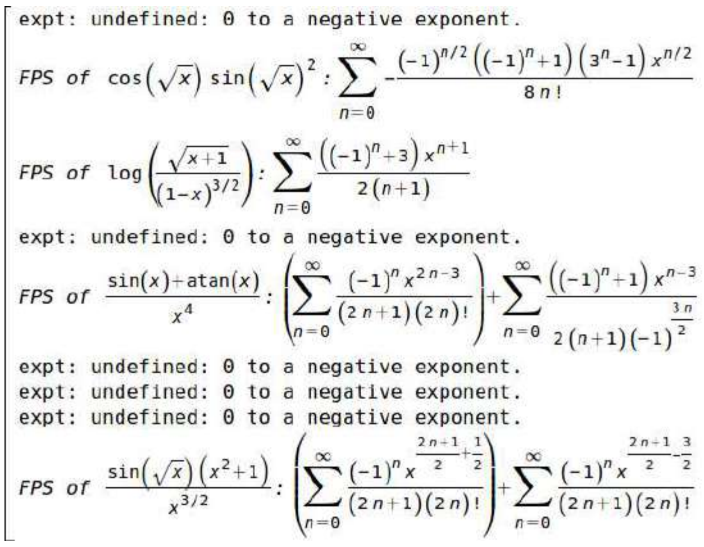
PS of $e^{\operatorname{asinh}(x)}: \left(\sum_{n=0}^{\infty}-\frac{(-1)^n (2n)!\, x^{2n}}{(2n-1)2^{2n} n!^2}\right)+x$

```
expt: undefined: 0 to a negative exponent.
expt: undefined: 0 to a negative exponent.
```
PS of $\frac{e^{x^{3/2}}}{\sqrt{x}}: \sum_{n=0}^{\infty}\frac{x^{\frac{3n-1}{2}}}{n!}$

```
expt: undefined: 0 to a negative exponent.
```
PS of $\frac{1}{x^{1/3}+1}: \sum_{n=0}^{\infty}(-1)^n x^{n/3}$

```
expt: undefined: 0 to a negative exponent.
expt: undefined: 0 to a negative exponent.
```
PS of $\frac{\sin(\sqrt{x})\, e^{\sqrt{x}}}{\sqrt{x}}: \sum_{n=0}^{\infty}\frac{i\left((i-1)^n(-1)^n-(i+1)^n\right)x^{\frac{n-1}{2}}}{2n!}$

Now setting the variable `Laurent_Puiseux_Series` to true, we can compute Laurent series and Puiseux series using the function `FPS()` which in this case is able to compute power series of more functions compared to the function `HLP_FPS()`. Let us show some results we obtain in this case.



```
expt: undefined: 0 to a negative exponent.
```
FPS of $\cos(\sqrt{x})\sin(\sqrt{x})^2: \sum_{n=0}^{\infty}-\frac{(-1)^{n/2}\left((-1)^n+1\right)\left(3^n-1\right)x^{n/2}}{8n!}$

FPS of $\log\left(\frac{\sqrt{x+1}}{(1-x)^{3/2}}\right): \sum_{n=0}^{\infty}\frac{\left((-1)^n+3\right)x^{n+1}}{2(n+1)}$

```
expt: undefined: 0 to a negative exponent.
```
FPS of $\frac{\sin(x)+\operatorname{atan}(x)}{x^4}: \left(\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n-3}}{(2n+1)(2n)!}\right)+\sum_{n=0}^{\infty}\frac{\left((-1)^n+1\right)x^{n-3}}{2(n+1)(-1)^{\frac{3n}{2}}}$

```
expt: undefined: 0 to a negative exponent.
expt: undefined: 0 to a negative exponent.
expt: undefined: 0 to a negative exponent.
```
FPS of $\frac{\sin(\sqrt{x})\left(x^2+1\right)}{x^{3/2}}: \left(\sum_{n=0}^{\infty}\frac{(-1)^n x^{\frac{2n+1}{2}+\frac{1}{2}}}{(2n+1)(2n)!}\right)+\sum_{n=0}^{\infty}\frac{(-1)^n x^{\frac{2n+1}{2}-\frac{3}{2}}}{(2n+1)(2n)!}$

Having given numerous results we can get, using our package, let us move to some general remarks.

6.3 General Remarks about our Package

In this part, we make some useful remarks for the user about functions in the package.

6.3.1 Remark. When the function `HOLO_FPS()` is able to compute the FPS, then the expression it returns is simpler than all the other expressions one gets using other functions like `FPS()` or `RAT_FPS()`. For instance $\texttt{FPS}(\log(1-x^5),x)$ gives a very large expression of the FPS whereas $\texttt{HOLO\_FPS}(\log(1-x^5),x)$ gives the following simple expression $\sum_{n=0}^{\infty} -\frac{x^{5\,n+5}}{n+1}$.

6.3.2 Remark. For many rational functions, the procedure for hypergeometric type functions works, nevertheless there are also exceptions like the following functions $(1+x^2)/(1-x^2)$, $(1+x^4)/(1-x^3)$, $(1+x^4)/(1+x^3)$, for which this procedure fails. Of course those functions are managed well by the procedure presented in Chapter 5 for rational functions.

6.3.3 Remark. Using the function `HOLO_FPS()`, one can fail to get the FPS of a function `f` which can be obtained using the function `FPS()` due to the fact that the function `FPS()` computes the FPS of each summand in the expression and returns their sum. Further for a function $f(x)$ of the form $f(x) = p(x)g(x)$ where $p(x)$ is a polynomial, the function `HOLO_FPS()` might be able to compute only the FPS of $g(x)$ but not the FPS of $f(x)$, in this case the function `FPS()` will compute the FPS of $f(x)$ by doing the product of $p(x)$ and the FPS of $g(x)$.

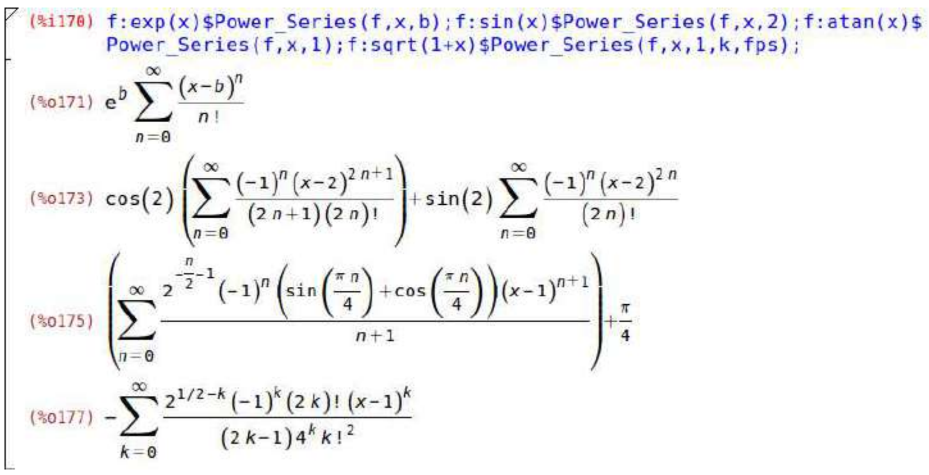
6.3.4 Remark. In order to compute FPS of functions like $f(x) = \frac{\sin(x)}{x}$ which are not defined at 0 but which can be extended at 0, the user needs to enable the computation of Laurent and Puiseux series by setting the variable `Laurent_Puiseux_Series` to true. Indeed, in our implementation we did not consider the limits of a function f and its derivatives at 0 since in some cases, the computation of such limits can take a long time and slows down the program.

6.3.5 Remark. The function `Holonomic_DE()` apart from its 3 main arguments can also take an optional fourth argument. Indeed, the fourth argument should be a positive integer and represents the lowest order of the HDE searched. If the fourth argument is less than `MAX_ORDER_DERIVATIVE`, then the function `Holonomic_DE()` looks for a HDE of order greater than the value of that argument and less than `MAX_ORDER_DERIVATIVE`, otherwise the function looks only for a HDE of order exactly the value of this argument. This optional argument can be very useful sometimes and can reduce the time of computation. For instance if one guesses that the order of the HDE is greater than 10, then setting the fourth argument to 10 will considerably reduce the time of computation. As illustration, the computation of the IV-HDE of the function $f(x) = \sin(x)^6 \arcsin(x)$ without specification of the optional argument takes around 8 minutes and 43 seconds to find a HDE of order 14 whereas by specifying the optional argument to 14, we get the IV-HDE only after 1 minute and 48 seconds.

6.3.6 Remark. If the user wants to find the FPS of a complex function $f(x)$, he or she must first set the variable `COMPLEX_COEFF` to true, otherwise the output will be only the real part of the solution and therefore is incorrect.

6.3.7 Remark. In order to compute the FPS of a function $f(x)$ at a point a, we define the function $\texttt{Power\_Series}(f,x,a)$ which takes the function f, its main variable x, the point a and returns the FPS of $f(x)$ at the point a. Indeed, knowing that the FPS of $f(x)$ at a point a is equivalent to the FPS of the function $g(x) = f(x+a)$ at the point 0, this function computes the FPS of $g(x)$ at 0 (through the functions `HOLO_FPS()`, `HLP_FPS()` or `FPS()`) and replaces x by $x-a$. Note also that this function takes 2 optional arguments: the first one is the index of summation, and the second one

indicates the function to use to compute the FPS of $g(x)$. If the second optional argument is `holo` (`hlp`) then, the function `Power_Series()` uses the function `HOLO_FPS()` (`HLP_FPS()` respectively). If the second argument is `fps_hlp` then the variable `Laurent_Puiseux_Series` is set to true and the function `Power_Series()` uses the function `FPS()`, otherwise the variable `Laurent_Puiseux_Series` is set to false and the function `Power_Series()` uses the function `FPS()`.
Remark that, for a function $f(x)$, the function `Power_Series()` can be able to compute the FPS at a point a, but might not be able to compute the FPS at another point b different from a. The reason is that a function $f(x)$ can be of hypergeometric type at one point of development but fails to be of hypergeometric type at another point.
Below is a snapshot of some results we got using the function `Power_Series()`.



```
(%i170) f:exp(x)$Power_Series(f,x,b);f:sin(x)$Power_Series(f,x,2);f:atan(x)$
        Power_Series(f,x,1);f:sqrt(1+x)$Power_Series(f,x,1,k,fps);
```

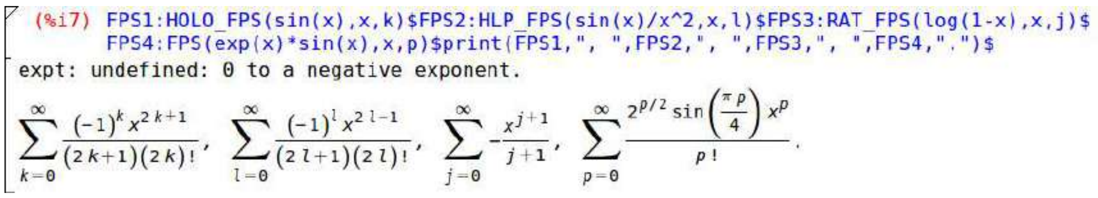
(%o171) $$e^b\sum_{n=0}^{\infty}\frac{(x-b)^n}{n!}$$

(%o173) $$\cos(2)\left(\sum_{n=0}^{\infty}\frac{(-1)^n(x-2)^{2n+1}}{(2n+1)(2n)!}\right)+\sin(2)\sum_{n=0}^{\infty}\frac{(-1)^n(x-2)^{2n}}{(2n)!}$$

(%o175) $$\left(\sum_{n=0}^{\infty}\frac{2^{-\frac{n}{2}-1}(-1)^n\left(\sin\left(\frac{\pi n}{4}\right)+\cos\left(\frac{\pi n}{4}\right)\right)(x-1)^{n+1}}{n+1}\right)+\frac{\pi}{4}$$

(%o177) $$-\sum_{k=0}^{\infty}\frac{2^{1/2-k}(-1)^k(2k)!\,(x-1)^k}{(2k-1)4^k k!^2}$$

6.3.8 Remark. The functions `HOLO_FPS()`, `HLP_FPS()`, `RAT_FPS()` and `FPS()` can take an optional third argument representing the index of summation in the FPS as shown in the snapshot below.



```
(%i7) FPS1:HOLO_FPS(sin(x),x,k)$FPS2:HLP_FPS(sin(x)/x^2,x,l)$FPS3:RAT_FPS(log(1-x),x,j)$
      FPS4:FPS(exp(x)*sin(x),x,p)$print(FPS1,", ",FPS2,", ",FPS3,", ",FPS4,".")$
expt: undefined: 0 to a negative exponent.
```

$$\sum_{k=0}^{\infty}\frac{(-1)^k x^{2k+1}}{(2k+1)(2k)!},\ \sum_{l=0}^{\infty}\frac{(-1)^l x^{2l-1}}{(2l+1)(2l)!},\ \sum_{j=0}^{\infty}-\frac{x^{j+1}}{j+1},\ \sum_{p=0}^{\infty}\frac{2^{p/2}\sin\left(\frac{\pi p}{4}\right)x^p}{p!}.$$

7. Conclusion and Perspectives

In this essay, the aim was the implementation in the CAS Maxima of the algorithm for computing formal power series of hypergeometric functions and rational functions. This algorithm was presented in [3] by Wolfram Koepf in 1992. We have presented and implemented those algorithmic procedures in the CAS Maxima, and also considered the computation of Laurent series and Puiseux series. Based on those procedures, we got nice results and were able to compute formal power series of a large family of functions. However, each of those procedures is not fully satisfactory. Indeed, the procedure applied for hypergeometric functions has a constraint, it might happen that the recurrence equation obtained in the process is not of hypergeometric type and cannot be solved by the methods implemented in the recurrence equation solver `solve_rec()` of the computer algebra system Maxima. In this case, the procedure fails and ends up with the recurrence equation instead of the formal power series. The procedure used for rational functions has also a constraint. In fact, in this process, we need to compute the full factorisation of the polynomial representing the denominator of the rational function over the field of complex numbers $\mathbb{C}$, though such full factorisation theoretically exists, there is no general procedure to get it. Hence in some cases, the computation of such full factorisation is impossible in a computer algebra system. This can be illustrated by the polynomial $x^7 + x^5 + 4x^2 + 67$ for which such a full factorisation is not available in the CAS Maxima. In those cases, the procedure for rational functions fails. For further researchs, in order to improve the work we did, one may consider the implementation in the CAS Maxima of the Petkovsek-van-Hoeij algorithm to solve our recurrence equations in order to deal with recurrence equations which are not of hypergeometric type.

Acknowledgements

Firstly, I would like to thank the Lord to have given me the great opportunity to be part of the 2016/2017 Aims-Cameroon students batch.

Secondly, I am very grateful to Pr. Dr. Wolfram Koepf for having accepted to be my supervisor and for all the help he provides me during my research period. I also thank Dr. Daniel Duviol Tcheutia for all the assistance I received from him. Also I would like to specially thank Dr. Ferdinand Ngakeu who has always believed in me and encouraged me.

Thirdly, I am very grateful to the founder of AIMS Pr. Neil Turok, to the Academic Director Pr. Mama Foupouagnigni and the staff of Aims-Cameroon.

Finally a special thank to my father Mr. Simo Rene and my mother Mme. Tchuensu Josephine and my brother and sisters for all the supports I have always received from them.

I want also to thank everyone who helped me in my study process.

References

[1] Bruce W Char. *MAPLE Reference Manual*. Watcom Publications, 1988.

[2] Robert Dodier. Maxima Manual. http://maxima.sourceforge.net/docs/manual/maxima.html, 2015. Accessed: May 10, 2016.

[3] Wolfram Koepf. Power series in computer algebra. *Journal of Symbolic Computation*, 13(6):581–603, 1992.

[4] Wolfram Koepf. Symbolic computation of formal power series with Macsyma. *Proceedings of the Workshop on Functional-Analytic Methods in Complex Analysis and Applications to Partial Differential Equations, Trieste, Italy, January 1993, Eds. W. Tutschke, A. S. Mshimba, World Scientific Publishing Co.*, pages 306–328, 1995.

[5] Wolfram Koepf. *Hypergeometric Summation. An Algorithmic Approach to Summation and Special Function Identities*. Springer Universitext. Springer, London, Second edition, 2014.

[6] Stephen Wolfram. *The Mathematica Book, 5th Ed.* 2003.