Solutions Of Difference Equations With Polynomial Coefficients

DIPLOMARBEIT

verfasst von

CHRISTIAN WEIXLBAUMER

RISC Linz, Johannes Kepler Universität

A-4040 Linz, Austria

Christian.Weixlbaumer@risc.uni-linz.ac.at

Abstract

This thesis presents detailed information about the state of art concerning the search for solutions of linear difference equations. It especially tries to fill theoretical gaps and improve some of the existing algorithms.

The first chapter provides some fundamental definitions and theorems that are used throughout the entire thesis.

The second chapter gives an overview about the theory of linear difference operators (with polynomial resp. rational function coefficients). This includes facts about the solution space and a survey about important operations on linear difference operators.

In the subsequent four chapters algorithms for finding polynomial, rational, hypergeometric and d'Alembertian solutions are presented and compared.

Zusammenfassung

Diese Diplomarbeit soll einen Überblick über den derzeitigen Wissensstand bei der Suche nach Lösungen von linearen Differenzengleichungen geben. Dabei gilt besonderes Augenmerk dem Schließen einiger theoretischer Lücken sowie der Verbesserung von existierenden Algorithmen.

Im ersten Kapitel werden einige fundamentale Definitionen und Sätze präsentiert, die für alle folgenden Kapitel benötigt werden.

Das zweite Kapitel soll einen Überblick über die Theorie der linearen Differenzenoperatoren (mit polynomialen bzw. rationalen Funktionen als Koeffizienten) geben. Dies schließt sowohl Untersuchungen über den Lösungsraum eines Differenzenoperators als auch die Definitionen einiger wichtiger Operationen mit derartigen Operatoren mitein.

In den darauffolgenden vier Kapiteln werden diverse Algorithmen zum Finden polynomialer, rationaler, hypergeometrischer und d'Alembert'scher Lösungen präsentiert und verglichen.

Contents

Chapter 1

Introduction

1.1 A Historical Survey

Although linear difference equations are still a current topic of mathematical research they have a long and remarkable history. The first problem concerning linear difference equations (recurrences) actually appeared in 1202 in the treatise *Liber abbaci* by Leonardo da Pisa who might be better known under his nickname Fibonacci:

> "A certain man put a pair of rabbits in a place surrounded on all sides by a wall. How many pairs of rabbits can be produced from that pair in a year if it is supposed that every month each pair begets a new pair which from the second month on becomes productive?"

Obviously, we get the following sequence for the number of rabbits after n month. Note that the new number can be calculated by adding the number of the last month and the number of the last but one month which is just the number of the newborns:

n	rabbit pairs	
$n = 0$	1	
$n = 1$	2	the "old" pair + a "new" pair
$n = 2$	3	two "old" pairs + one "new" pair
$n = 3$	5	three "old" pairs + two "new" pairs
$\vdots$	$\vdots$	
$n = 12$	377	

Thus, 377 is the answer to Fibonacci's rabbit problem. The resulting recurrence is

$$F_n = F_{n-1} + F_{n-2}$$

Later on, an additional 1 was inserted at the beginning of the sequence and the *Fibonacci numbers* were born:

$$\langle F_n \rangle = \langle 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, ... \rangle$$

Another interpretation for the Fibonacci numbers should not be omitted: F_n equals the number of Morse codes of length $n-1$ consisting of dots which have length 1 and dashes which have length 2.

length	Morse codes	number
$n-1=0$	ϵ (empty code)	1
$n-1=1$	.	1
$n-1=2$	.. $\mid$ $-$	2
$n-1=3$	... $\mid$.$-$ $\mid$ $-$.	3
$n-1=4$	.... $\mid$ ..$-$ $\mid$.$-$. $\mid$ $-$.. $\mid$ $--$	5
$n-1=5$	..... $\mid$ ...$-$ $\mid$ ..$-$. $\mid$.$-$.. $\mid$ $-$... $\mid$.$--$ $\mid$ $-$.$-$ $\mid$ $--$.	8

The recursion can be seen by the following consideration: A new code of length n can start either with a dot or a dash. If it starts with a dot, then the rest of the code is a code of length $n-1$, if it starts with a dash, then the rest of the code is a code of length $n-2$.

More than 600 years later (in 1843) Jacques Binet published a formula that allowed to compute the n-th Fibonacci number directly, i.e. without computing all the other (smaller) Fibonacci numbers before:

$$F_n = \frac{1}{\sqrt{5}} \left[\left(\frac{1+\sqrt{5}}{2} \right)^n - \left(\frac{1-\sqrt{5}}{2} \right)^n \right]$$

It is, however, quite certain that Binet was not the first one to find this formula[1] - two other possibilities can be found in the relevant literature:

- Leonard Euler already published the formula in 1765

- Don Knuth in *The Art of Computer Programming, Volume 1 Fundamental Algorithms*, section 12.8, writes that Abraham de Moivre (1667-1754) had written about this formula more than 100 years before Binet, in 1730, and had indeed found a method for finding formulas for any general sequence of numbers formed in a similar way to the Fibonacci numbers.

According to Knuth Moivre was the first one who was able to solve a linear difference equation with constant coefficients methodically. Solving equations with variable (resp. polynomial) coefficients, however, turned out to be a far more complex problem, especially because of the high computing effort. Seemingly, the first useful general[2] methods came from Sergei Abramov (polynomial and rational solutions) and later from Marko Petkovšek (hypergeometric solutions) in the late 1980s and early 1990s. Among others, the search for solutions of linear difference equations was (and is) pushed by problems concerning symbolic summation.

---

[1]Like many results in Mathematics, it is often not the original discoverer who gets the glory of having his name attached to the result, but someone later!

[2]Of course, there existed methods for some "easier" cases, e.g. R. W. Gosper already dealt with inhomogeneous first order difference equations in 1978.

1.2 Summary

We would like to give a short summary of the contents of this thesis here. This summary will not contain explicit definitions or exact proofs, but will rather give an overview of the main results of the thesis.

The main objective of the thesis is to provide detailed information about the state of art concerning the search for solutions of linear difference equations with polynomial coefficients as well as to fill theoretical gaps and to improve some of the existing algorithms. For the software part of the thesis some of the presented algorithms (e.g. Abramov's algorithm for finding rational solutions and van Hoeij's algorithm for finding hypergeometric solutions) have been implemented in *MATHEMATICA*.

The following chapter deals with basic notations and definitions. After introducing the shift- and the difference operator we present a generalized notion of *falling* and *rising* factorials which will enable us to write down the solution of a homogeneous difference equation of order 1 in a very general way. Furthermore, the falling factorial provides a representation of (monic) polynomials, which is the discrete analogue to squarefree factorization. This representation - developed by Peter Paule in the early nineties - is called *greatest factorial factorization* (gff) and will be employed as an eminent method throughout the thesis. We will, inter alia, make use of the gff-concept in a new proof of Sergei Abramov's algorithm for finding rational solutions and we will even offer an explanation why the looping in Abramov's algorithm has to be made from the upper to the lower bound and not vice versa.
The rest of Chapter 2 is concerned with the basic definitions and some fundamental properties of *hypergeometric* terms and the introduction of *linear difference equations*. We decided to introduce difference equations by means of the corresponding difference operator, as the operator notation turned out to be far more effective and convenient.

Chapter 3 leads into the theory of (linear) difference operators. We recall the definition of *polynomial*, *rational* and *hypergeometric* solutions and give a short survey about the solution of linear difference equations with constant coefficients. Afterwards, four important operations on difference operators are introduced:

- the *adjoint operator* which can, for example, be used for factorization of difference operators

- the *symmetric product* which was first introduced by Mark van Hoeij as a discrete analogue to a corresponding operation from the theory of differential equation. This operation will be very useful several times - for instance, it helps to write down the algorithm for finding hypergeometric partial solutions in a very efficient way.

- the *greatest common right divisor* - first introduced by Oystein Ore in 1933 - which can be used to construct a difference equation that has exactly the common solutions of two (or more) given difference equations

- the *least common left multiple* - also first introduced by Oystein Ore in 1933 - which can be used to construct a difference equation that has exactly (the union of) the solutions of two (or more) given difference equations

Beginning with Chapter 4 we present and compare algorithms for finding polynomial, rational, hypergeometric and d'Alembertian solutions. Each chapter contains an example section where we also try to point the differing properties of these algorithms.

In Chapter 4 we first recap the algorithms by Sergei Abramov (the equation given in terms of the *difference operator*) and Marko Petkovšek (the equation given in terms of the *shift operator*), respectively, for finding polynomial solutions of linear difference equations. Later on, we will show that these two algorithms are not only equivalent (the degree polynomials and the degree bounds are the same), but that Abramov's algorithm (as it appeared in the book "A=B") also contains a superfluous condition.
The rest of the chapter deals with the problem of minimizing the number of unknowns in the linear system which has be solved. We present the algorithm of Abramov/Bronstein/Petkovšek which reduces the final linear system to $n + d$ equations with n unknowns (where n denotes the order and d the degree of the difference equation). We will also show that the computed degree bound is equal to the degree bound of the two other algorithms and that the degree polynomials of all three algorithms are connected by a simple formula. We will conclude the chapter with a presentation of two new algorithms for minimizing the number of variables based on interpolation techniques. These methods - using Lagrange interpolation and Newton interpolation, respectively - also yield a linear system with merely n unknowns.

Rational solutions are treated in Chapter 5 in which we will first present the algorithm by Sergei Abramov, together with the already mentioned new correctness proof. Additionally, we analyze the general structure of a possible denominator of a rational solution. We continue with some improvements of Abramov's algorithm, which can be used for other, similar, algorithms as well. Afterwards, the algorithm by Mark van Hoeij, which was developed for systems of linear difference equations, is recalled. We will show how to improve this algorithm for the scalar case, as for this one the general algorithm will not turn out to be optimal. Moreover, this variant of van Hoeij's algorithm does not require root finding (over the complex numbers) anymore.

In Chapter 6 - dealing with hypergeometric solutions - we present the algorithms by Marko Petkovšek and Mark van Hoeij, respectively. Unlike Petkovšek's algorithm, which is relatively easy to understand, van Hoeij's algorithm makes

use of some nontrivial algebraical concepts. The algorithm - which is an analogue to an algorithm for finding (hyperexponential) solutions of linear differential equations - tries to construct first order right hand factors $\tau - r$ of a difference equation (where τ denotes the shift operator and r is a rational function) by finding the roots and poles of r. We will see that van Hoeij's approach is able to avoid splitting field computation as well as to reduce the exponentially growing number of cases to be worked out in Petkovšek's algorithm.

In the second part of Chapter 6 we show how to find particular (hypergeometric) solutions of linear difference equations. All these algorithms will turn out to be generalizations of the well-known Gosper algorithm, which we will analyze first. As mentioned before, the symmetric product operation helps to simplify the resulting algorithms.

In the last chapter we define *d'Alembertian* sequences (which are a natural generalization of hypergeometric sequences) and show how to find solutions of this kind by means of the *reduction of order* method. Although this method goes back to the 18th century the corresponding algorithm for difference equations due to Sergei Abramov and Marko Petkovšek is fairly new. Again, we will show how to simplify this algorithm with the help of the symmetric product.

1.3 Acknowledgments

Chapter 2

Notations and Basic Definitions

2.1 Basics

Notations 2.1.1 *Throughout the entire thesis the following (basic) notations are used:*

- $\mathbb{N} := \{0, 1, 2, ...\}$ denotes the set of nonnegative integers, $\mathbb{Z}$ the set of integers, $\mathbb{Q}$ the set of rational numbers, $\mathbb{R}$ the set of real numbers and $\mathbb{C}$ the set of complex numbers

- $\mathbb{K}$ denotes an arbitrary field (with characteristic zero), $\mathbb{K}[x]$ denotes the ring of polynomials over $\mathbb{K}$, $\mathbb{K}(x)$ denotes the field of rational functions over $\mathbb{K}$, $\mathbb{K}[[x]]$ denotes the field of formal power series over $\mathbb{K}$ and $\mathbb{K}((x))$ denotes the field of fractional power series over $\mathbb{K}$

- $\mathbb{K}^{\mathbb{K}}$ denotes the set of all functions from $\mathbb{K}$ to $\mathbb{K}$

- $\mathrm{GL}_n(\mathbb{K})$ denotes all regular (non-singular) $n \times n$-matrices over the field $\mathbb{K}$

- I denotes the identity matrix

- $\mathcal{L}(A)$ denotes the $\mathbb{K}$-linear hull of a set A

- δ_{ij} denotes the Kronecker Delta, which is 1 for $i = j$ and 0 otherwise

Convention 2.1.2 *Throughout the entire thesis we use the following conventions:*

- $0^0 = 1$

- $\binom{n}{k} = 0$ *when* $k < 0$ *or* $0 \leq n < k$

- $\deg 0 := -\infty$

Moreover we assume that the field $\mathbb{K}$ is computable, *meaning that the elements of $\mathbb{K}$ can be finitely represented and there exists algorithms for carrying out the field operations (e.g. finding integer roots of a polynomial).*

2.2   Sequences, Functions, and Operators

Definition 2.2.1 *Sequences*

- *A map $u : \mathbb{N} \to \mathbb{K}$ is called a* sequence *(over $\mathbb{K}$). We will denote the ring of all sequences over $\mathbb{K}$ (with addition and multiplication defined term-wise) by $\mathbb{K}^{\mathbb{N}}$ and usually write u_k for $u(k)$.*

- *Two sequences u_k and v_k are considered equal if they agree from some point on.*

- *A sequence u_k is called* polynomial *if there exists a polynomial $p(x)$ such that $u_k = p(k)$ for all large enough integers k.*

- *A sequence u_k is called* rational *if there exists a rational function $r(x)$ such that $u_k = r(k)$ for all large enough integers k.*

Since two polynomials which agree on finitely many points are identical, the polynomial defining a polynomial sequence is unique. More exactly, the ring of polynomial sequences is isomorphic to $\mathbb{K}[x]$. The same holds in the rational case: the ring of rational sequences is isomorphic to $\mathbb{K}(x)$. Taking these facts into consideration we will not distinguish between sequences and functions.

Definition 2.2.2 *We define the* Shift-Operator τ *as the $\mathbb{K}$-automorphism of $\mathbb{K}^{\mathbb{K}}$ by*

$$\tau\big(y(x)\big) := y(x + 1) \quad \text{for } y \in \mathbb{K}^{\mathbb{K}}$$

For every integer m we define recursively:

$$\tau^m := \tau(\tau^{m-1})$$

and furthermore

$$\tau^0 := id$$

Definition 2.2.3 *We define the* Delta-Operator *or* Difference-Operator Δ *as the $\mathbb{K}$-automorphism of $\mathbb{K}^{\mathbb{K}}$ by*

$$\Delta\big(y(x)\big) := y(x + 1) - y(x) \quad \text{for } y \in \mathbb{K}^{\mathbb{K}}$$

For every (positive) integer m we define recursively:

$$\Delta^m := \Delta(\Delta^{m-1})$$

and furthermore

$$\Delta^0 := id$$

Proposition 2.2.4 *Some simple properties of τ and Δ:*

(a) $\tau = \Delta + 1$ *and* $\Delta = \tau - 1$

(b) $\tau(c) = c$ *for all* $c \in \mathbb{K}$

(c) $\Delta(c) = 0$ *for all* $c \in \mathbb{K}$

(d) τ *and* Δ *and their powers are linear operators*

(e) $\tau(f \cdot g) = \tau(f) \cdot \tau(g)$ *for* $f, g \in \mathbb{K}^{\mathbb{K}}$

(f) $\tau(\frac{f}{g}) = \frac{\tau(f)}{\tau(g)}$ *for* $f, g \in \mathbb{K}^{\mathbb{K}}$

(g) $\Delta(f \cdot g) = \Delta(f) \cdot g + f \cdot \Delta(g) + \Delta(f) \cdot \Delta(g)$ *for* $f, g \in \mathbb{K}^{\mathbb{K}}$

(h) $\Delta(\frac{1}{f}) = -\frac{\Delta(f)}{\tau(f) \cdot f}$ *for* $f, g \in \mathbb{K}^{\mathbb{K}}$

(i) $\tau^m(y(x)) = y(x + m)$ *for* $m \in \mathbb{Z}$

(j) $\Delta^m(y(x)) = \sum_{i=0}^{m}(-1)^{m-i}\binom{m}{i}y(x+i)$ *for* $m \in \mathbb{N}$

2.2.1 The Falling and Rising Factorial and the GFF

Definition 2.2.5 *For a polynomial* $p \in \mathbb{K}[x]$ *and* $m \in \mathbb{N}$, *we define the m-th falling factorial by*

$$p^{\underline{m}} = p(x)p(x-1)...p(x-m+1) = p \cdot \tau^{-1}(p) \cdot ... \cdot \tau^{-m+1}(p)$$

In particular, we have

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

which is a monic polynomial of degree m. Similarly, we also have the m-th rising factorial

$$p^{\overline{m}} = p(x)p(x+1)...p(x+m-1) = p \cdot \tau(p) \cdot ... \cdot \tau^{m-1}(p)$$

and additionally

$$x^{\overline{m}} = x(x+1)...(x+m-1)$$

For $m = 0$, *we let* $p^{\underline{0}} = p^{\overline{0}} = 1$.

Proposition 2.2.6 *Some simple properties of the falling and rising factorial*

(a) $p^{\overline{m}} = \tau^{m-1}(p^{\underline{m}})$

(b) $\tau^k(p^{\underline{m}}) = (\tau^k p)^{\underline{m}}$ *for* $p \in \mathbb{K}[x]$ *and* $k, m \in \mathbb{N}$

(c) $\tau^k(p^{\overline{m}}) = (\tau^k p)^{\overline{m}}$ *for* $p \in \mathbb{K}[x]$ *and* $k, m \in \mathbb{N}$

(d) $m! = m^{\underline{m}} = 1^{\overline{m}}$ *for* $m \in \mathbb{N}$

(e) $\Delta(x^{\underline{m}}) = m \cdot x^{\underline{m-1}}$ *for* $m \in \mathbb{N}$

(f) $\binom{x}{k} = \frac{x^{\underline{k}}}{k!} = \frac{x^{\underline{k}}}{k^{\underline{k}}} = \left(\frac{x}{k}\right)^{\underline{k}}$

Remark 2.2.7 *Observe that Proposition 2.2.6 (e) is the discrete analogue to* $\partial(x^m) = m \cdot x^{m-1}$ *with the differential operator* ∂.

The following definition due to Peter Paule will turn out to be the discrete analogue to squarefree factorization:

Definition 2.2.8 *Let* $p, p_1, ..., p_m \in \mathbb{K}[x]$ *and* p *monic. Then* $(p_1, ..., p_m)$ *is called a* greatest factorial factorization *(gff) of* p *if the following hold:*

(F1) $p = p_1^{\underline{1}}...p_m^{\underline{m}}$

(F2) $p_1, ..., p_m$ *are monic and* $p_m \neq 1$

(F3) $\gcd(p_i^{\underline{i}}, \tau(p_j)) = 1$ *for* $1 \leq i \leq j \leq m$

(F4) $\gcd(p_i^{\underline{i}}, \tau^{-j}(p_j)) = 1$ *for* $1 \leq i \leq j \leq m$

One can say that the gff is that product of falling factorials which takes care of maximal chains (meaning falling factorials of maximal length). It can be shown that every nonzero monic polynomial has a unique gff. See for example [Pau95] (a preliminary version appeared as [Pau93]) or [vZGe99]. The computation of the gff works analogously to the computation of the squarefree factorization using the following lemma:

Lemma 2.2.9 *Let* $p \in \mathbb{K}[x]$ *monic and nonconstant with* $\mathrm{gff}(p) = (p_1, ..., p_m)$, *then*

$$\mathrm{gff}(\gcd(p, \tau(p))) = (p_2, ..., p_m) \text{ and } p_1 = \frac{p}{p_2^{\underline{2}}...p_m^{\underline{m}}}$$

Proof: See [Pau95] or [vZGe99]

∎

Let's demonstrate the resulting algorithm with an example:

Example 2.2.10 *Let's compute the gff of* $p = x(x-1)^3(x-2)^2(x-4)^2(x-5)$ *(The polynomial* p *is given in factorized form for clarity only).*

We start with computing $q_1 = \gcd(p, \tau(p))$ yielding

$$q_1 = x(x-1)^2(x-4)$$

We continue with q_1 and compute $q_2 = \gcd(q_1, \tau(q_1))$ yielding

$$q_2 = x$$

The next gcd-computation $q_3 = \gcd(q_2, \tau(q_2))$ already yields 1.

Now we can compute $\mathrm{gff}(p)$ starting with a list containing the last nontrivial gcd which is $q_2 = x$, hence

$$\mathrm{gff} = (x)$$

At this point we use our lemma on $p = q_1$ yielding

$$\mathrm{gff} = \left(\frac{x(x-1)^2(x-4)}{x^2}, x \right) = \left(\frac{x(x-1)^2(x-4)}{x(x-1)}, x \right) = ((x-1)(x-4), x)$$

Again using our lemma on p yields

$$\begin{aligned} \mathrm{gff}(p) \;\; &= \;\; \mathrm{gff} = \left(\frac{x(x-1)^3(x-2)^2(x-4)^2(x-5)}{((x-1)(x-4))^2 x^3}, (x-1)(x-4), x \right) = \\ &= \;\; ((x-1)(x-4), (x-1)(x-4), x) \end{aligned}$$

Remark 2.2.11 *Observe the analogue: Let $p = \mathrm{gff}(p) = (p_1, ..., p_m)$, then*

$$\mathrm{gff}(p, \tau(p)) = \mathrm{gff}(p, \tau(p) - p) = \mathrm{gff}(p, \Delta(p)) = (p_2, ..., p_m)$$

and let $p = p_1^1 ... p_m^m$ the squarefree factorization of p, then

$$\gcd(p, p') = \gcd(p, \partial p) = p_2^1 ... p_m^{m-1}$$

2.2.2 Hypergeometric Terms

Definition 2.2.12 *A nonzero sequence y is called hypergeometric iff there exists a rational sequence r such that:*

$$r = \frac{\tau(y)}{y} \Leftrightarrow r_k = \frac{y_{k+1}}{y_k} \quad \text{for all large enough } k$$

We denote the set of all hypergeometric sequences by $\mathbb{H}$. The rational sequence r is sometimes called certificate.

Example 2.2.13 *Some certificates and their corresponding hypergeometric expression:*

(a) $r_k = 2 \Leftrightarrow y_k = c \cdot 2^k$ with $c \in \mathbb{K}$

(b) $r_k = k + 1 \Leftrightarrow y_k = c \cdot k!$ with $c \in \mathbb{K}$

(c) $r_k = k + \sqrt{2} \Leftrightarrow y_k = c_1 \cdot \sqrt{2}^{\overline{k}} = c_2 \cdot (k + \sqrt{2} - 1)^{\underline{k}} = c_3 \cdot \binom{k+\sqrt{2}-1}{k}$ with $c_i \in \mathbb{K}$

Analogously to the polynomial and rational sequences we will not distinguish between hypergeometric sequences and hypergeometric functions (defined analogously). The following table sums up the correspondences and necessary generalizations (a denotes an arbitrary element from $\mathbb{K}$; we omit the possible multiplicative constant):

Table 2.2.2: Certificates and their corresponding sequences/functions

certificate	sequence	function
a	a^k	a^x
k resp. x	$(k-1)!$	$\Gamma(x)$
$k+a$ resp. $x+a$	$a^{\overline{k}} = (k+a-1)^{\underline{k}}$ or $\binom{k+a-1}{k}$	$\Gamma(x+a)$ or $\frac{\Gamma(x+a)}{\Gamma(a)}$
$\frac{r_{k+1}}{r_k}$ resp. $\frac{r(x+1)}{r(x)}$	r_k	$r(x)$

In the sequel we will rather use $\mathbb{H}$ as the set of all hypergeometric functions than use it as the set of all hypergeometric sequences, because it will be a little bit more convenient. However, instead of writing $\Gamma(x)$ we will simply write $(x-1)!$.

Proposition 2.2.14 *Some simple properties of* $\mathbb{H}$*:*

(a) $r(x) = \frac{\tau(y)}{y} = a \in \mathbb{K} \Leftrightarrow y = c \cdot a^x$ *with* $c \in \mathbb{K}$

(b) $r(x) = \frac{\tau(y)}{y} = x - a$ *with* $c \in \mathbb{K} \Leftrightarrow y = c \cdot \Gamma(x-a)$ *with* $c \in \mathbb{K}$

(c) $\mathbb{H}$ *is a group under multiplication*

(d) *Every hypergeometric function* $u(x)$ *can be decomposed in the following manner:*
$$u(x) = c^x \cdot R(x) \cdot \Gamma(x+a_1)^{e_1} \cdot ... \cdot \Gamma(x+a_m)^{e_m}$$
with $c, a_i \in \mathbb{K}, e_i \in \mathbb{Z}$ *and* $R \in \mathbb{K}(x)$

(e) $\mathbb{H}$ *is not closed under addition*

(f) $\mathbb{K}(x)^* \subseteq \mathbb{H}$

(g) $y \in \mathbb{H}$ *with* $\frac{\tau(y)}{y} = r \Rightarrow \frac{\tau^m(y)}{y} = \tau^{m-1}(r) \cdot ... \cdot \tau(r) \cdot r \in \mathbb{K}(x)$ *for all positive integers* m

Proof: Ad (d): Follows from (c) and the corresponding decomposition of the (rational) certificate.
Ad (g): Note that $\frac{\tau^m(y)}{y} = \frac{\tau^m(y)}{\tau^{m-1}(y)} \cdot \frac{\tau^{m-1}(y)}{\tau^{m-2}(y)} \cdot ... \cdot \frac{\tau(y)}{y}$

$\blacksquare$

Note that Proposition 2.2.14 (d) shows that Table 2.2.2 already contains the complete information about certificates and their corresponding hypergeometric sequences/functions.

We know that $\mathbb{H}$ is not closed under addition - the following theorem gives a characterization:

Lemma 2.2.15 *If* $y, z \in \mathbb{H}$ *then* $y + z \in \mathbb{H} \Leftrightarrow y/z \in \mathbb{K}(x)^*$

Proof: First consider $y + z = 0 \Leftrightarrow \frac{y}{z} = -1$. Now let $y \neq -z$:

$\Rightarrow$: Let $y + z = w \in \mathbb{H}$ and let $\frac{\tau(y)}{y} = r_y, \frac{\tau(z)}{z} = r_z, \frac{\tau(w)}{w} = r_w$, then

$$(y + z) \cdot r_w = w \cdot r_w = \tau(w) = \tau(y + z) = \tau(y) + \tau(z) = y \cdot r_y + z \cdot r_z$$

$$\Rightarrow y \cdot (r_y - r_w) + z \cdot (r_z - r_w) = 0$$

- Case 1: $r_y = r_w \lor r_z = r_w \Rightarrow z \cdot (r_z - r_w) = 0 \lor y \cdot (r_y - r_w) = 0 \Rightarrow$
  $r_y = r_z = r_w \Rightarrow \tau(\frac{y}{z}) = \frac{\tau(y)}{\tau(z)} = \frac{y \cdot r_y}{z \cdot r_z} = \frac{y}{z} \Rightarrow \frac{y}{z} \in \mathbb{K} \subset \mathbb{K}(x)^*$
- Case 2: $r_y \neq r_w \land r_z \neq r_w \Rightarrow \frac{y}{z} = -\frac{r_z - r_w}{r_y - r_w} \in \mathbb{K}(x)^*$

$\Leftarrow$: Let $y/z \in \mathbb{K}(x)^*$, then

$$\frac{\tau(y + z)}{y + z} = \frac{\tau(y) + \tau(z)}{y + z} = \frac{\frac{\tau(y)}{\tau(z)} \cdot \frac{\tau(z)}{z} + \frac{\tau(z)}{z}}{\frac{y}{z} + \frac{z}{z}} \in \mathbb{K}(x)^*$$

which shows that $y + z \in \mathbb{H}$

$\blacksquare$

Definition 2.2.16 *We call two hypergeometric functions y and z similar iff $y/z \in \mathbb{K}(x)$. In this case we write $y \sim z$. Otherwise, we write $y \nsim z$ and call y and z dissimilar.*

Proposition 2.2.17 *Let y be a hypergeometric function, then the following assertions hold:*

(a) $\Delta(y)$ is hypergeometric and $\Delta(y) \sim y$

(b) $\Delta^m(y)$ is hypergeometric and $\Delta^m(y) \sim y$ for all positive integers m

Proof: We will prove the first statement, the second statement follows from the first by induction.
Let $r = \frac{\tau(y)}{y}$, then

$$\Delta(y) = y(x + 1) - y(x) = y(x) \cdot \left(\frac{y(x + 1)}{y(x)} - \frac{y(x)}{y(x)} \right) = y(x) \cdot \left(r(x) - 1 \right)$$

Because $r(x) - 1$ is a rational function everything is shown.

$\blacksquare$

Lemma 2.2.18 *Let $f_1, ..., f_k \in \mathbb{H}$ and let $\sum_{i=0}^{k} f_i = 0$ (with $k \geq 2$), then there exists f_i and f_j such that $f_i \sim f_j$.*

Proof: We prove the assertion by induction on k.
$k = 2$: See the beginning of the proof of Lemma 2.2.15.
$(k - 1) \to k$: Because $\sum_{i=0}^{k} f_i = 0$ we have $\sum_{i=0}^{k} \tau(f_i) = \sum_{i=0}^{k} r_i f_i = 0$, too (where $r_i = \frac{\tau(f_i)}{f_i}$). Subtracting this equation from $\sum_{i=0}^{k} f_i = 0$ multiplied by r_k yields

$$\sum_{i=0}^{k-1} (r_k - r_i) f_i = 0 \tag{2.1}$$

Now we have to consider two cases:

- $\exists i$ such that $r_k = r_i$, then $f_k \sim f_i$

- $\forall i$: $r_k \neq r_i$, then all $(k-1)$ terms of (2.1) are hypergeometric and the statement follows from the induction hypothesis.

$\blacksquare$

The following lemma is fundamental:

Lemma 2.2.19 *Up to the order of the terms, the representation of a $\mathbb{K}$-linear combination of hypergeometric terms as sums of pairwise dissimilar hypergeometric terms is unique.*

Proof: Assume that $a_1, ..., a_k$ and $b_1, ..., b_m$ are pairwise dissimilar hypergeometric terms with

$$\sum_{i=1}^{k} a_i = \sum_{j=1}^{m} b_j \neq 0$$

We will use induction on $k + m$ in order to prove that $k = m$ and that each a_i equals some b_j.

$k + m = 0$: Trivial case

$(k + m - 2) \to (k + m)$: By Lemma 2.2.18 it follows that there exists some a_i similar to some b_j. Relabel the terms so that $a_k \sim b_m$, and let $h := a_k - b_m$. If $h \neq 0$, then $\sum_{i=1}^{k-1} a_i + h = \sum_{j=1}^{m-1} b_j$ and $\sum_{i=1}^{k-1} a_i = \sum_{j=1}^{m-1} b_j - h$ and the induction hypothesis (note that h is dissimilar to each arising a_i and b_j) implies $k = m - 1$ and $k - 1 = m$, which is a contradiction. Thus $h = 0$ which means $a_k = b_m$ and we have $\sum_{i=1}^{k-1} a_i = \sum_{j=1}^{m-1} b_j$. Now we can use the induction hypothesis which completes the proof.

$\blacksquare$

2.3   Difference Equations

Definition 2.3.1 *An operator of the form*

$$L = a_n(x)\tau^n + ... + a_0(x)\tau^0 \ \ with \ a_k \in \mathbb{K}(x) \tag{2.2}$$

is called a (linear) difference operator. *That means for a function y*

$$(Ly)(x) = a_n(x)y(x+n) + ... + a_0(x)y(x)$$

Of course Ly is defined for those $x \in \mathbb{K}$ for which $y(x), ..., y(x+n)$ and $a_0(x), ... a_n(x)$ are defined. An equation of the form

$$(Ly)(x) = f(x) \ \ or \ in \ short \ from \ Ly = f$$

where f is an (at the beginning) arbitrary function is called a (linear) difference *equation. The equation is called* homogeneous *if $f = 0$ and* inhomogeneous *otherwise.*

The set of all difference operators is

$$\mathbb{K}(x)[\tau] = \{a_n(x)\tau^n + ... + a_0(x)\tau^0 \mid n \in \mathbb{N}, a_0, ..., a_n \in \mathbb{K}(x)\}$$

Definition 2.3.2 *Let $L = a_n(x)\tau^n + ... + a_0(x)\tau^0 \in \mathbb{K}(x)[\tau]$ and $L \neq 0$. Denote*

$$ord(L) := \max(i \mid a_i \neq 0) - \min(i \mid a_i \neq 0)$$

$$order(L) := \max(i \mid a_i \neq 0)$$

If all $a_i(x) \in \mathbb{K}[x]$, then we define

$$degree(L) := \max_{0 \leq i \leq n} \deg a_i(x)$$

- *L is called* normal *if $a_0 \neq 0$, i.e. if $ord(L) = order(L)$.*

- *The* leading *coefficient of L is the highest non-zero coefficient of L (which is a_n if $a_n \neq 0$).*

- *The* trailing *coefficient of L is the lowest non-zero coefficient of L (which is a_0 if L is normal).*

- *L is called* monic *if the leading coefficient is 1.*

Remark 2.3.3 *Two further notations of difference equations:*

- *Sometimes it is useful to transform a (scalar) difference equation of order n into a difference system of order 1 (w.l.o.g. we have $a_n(x) \equiv 1$):*

$$y(x + n) + a_{n-1}y(x + n - 1) + ... + a_0(x)y(x) = f(x)$$

can be rewritten as

$$\tau(Y) = A \cdot Y + F$$

where

$$Y = (y, \tau(y), ..., \tau^{n-1}(y))^T$$

$$F = (0, ..., 0, f)^T$$

and the so-called companion matrix

$$A = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 1 \\ -a_0 & -a_1 & -a_2 & \dots & -a_{n-1} \end{pmatrix}$$

- *Because $\tau = \Delta + 1$, we can rewrite a difference operator in powers of Δ:*

$$L = \sum_{k=0}^{n} a_k(x)\tau^k = \sum_{k=0}^{n} a_k(x)(\Delta + 1)^k = \sum_{k=0}^{n} b_k(x)\Delta^k$$

where

$$b_k(x) = \sum_{i=k}^{n} \binom{i}{k} a_i(x)$$

Example 2.3.4 *Let's consider the following difference operator of order 2:*

$$L = 3\tau^2 - x\tau + x - 1$$

and the corresponding homogeneous difference equation

$$3 \cdot y(x+2) - x \cdot y(x+1) + (x-1) \cdot y(x) = 0$$

Now, let's transform by hand (remember the powers of Δ)...

$$
\begin{aligned}
Ly &= 3 \cdot y(x+2) - x \cdot y(x+1) + (x-1) \cdot y(x) = \\
&= 3 \cdot \Delta^2 y(x) + 3 \cdot 2 \cdot y(x+1) - 3y(x) - x \cdot y(x+1) + (x-1) \cdot y(x) = \\
&= 3 \cdot \Delta^2 y(x) + (6-x) \cdot y(x+1) + (x-4) \cdot y(x) = \\
&= 3 \cdot \Delta^2 y(x) + (6-x) \cdot \Delta y(x) + (6-x) \cdot y(x) + (x-4) \cdot y(x) = \\
&= 3 \cdot \Delta^2 y(x) + (6-x) \cdot \Delta y(x) + 2 \cdot y(x) \\
&= 3 \cdot \Delta^2 y(x) + (6-x) \cdot \Delta^1 y(x) + 2 \cdot \Delta^0 y(x)
\end{aligned}
$$

Lemma 2.3.5 *Let $L \in \mathbb{K}(x)[\tau]$ and $y \in \mathbb{H}$, then either $Ly = 0$ or $Ly \in \mathbb{H}$ together with $Ly \sim y$.*

Proof: This is an immediate consequence of Proposition 2.2.14 (g) together with the linearity of L.

■

Lemma 2.3.6 *Let $\tau \cdot y := \tau(y)\tau$ for $y \in \mathbb{K}(x)$ the multiplication on $\mathbb{K}(x)[\tau]$, then we have:*

(a) *$\mathbb{K}(x)[\tau]$ is a non-commutative ring (Ore-ring)*

(b) *Every product of difference operators can be uniquely written in the form (2.2), i.e. the coefficients in $\mathbb{K}(x)$ appear only on the left of the τ^k.*

Proof: Obvious!

Example 2.3.7 *Let's consider the following multiplication:*

$$
\begin{aligned}
&(\tau - 3) \cdot (\tau - (x-1)) = \\
&= \tau \cdot \tau - 3 \cdot \tau + \tau \cdot (-(x-1)) + 3 \cdot (x-1) = \\
&= \tau^2 - 3\tau + \tau(-x+1)\tau + 3(x-1) = \\
&= \tau^2 - 3\tau + (-(x+1) + 1)\tau + 3(x-1) = \\
&= \tau^2 - (x+3)\tau + 3(x-1)
\end{aligned}
$$

Remark 2.3.8 *Observe the following difference which will appear in the sequel: Let $L \in \mathbb{K}(x)[\tau]$ and $y \in \mathbb{K}(x)$ then $Ly \in \mathbb{K}(x)$, but $L \cdot y \in \mathbb{K}(x)[\tau]$.*

Definition 2.3.9 *Let $L \in \mathbb{K}(x)[\tau]$. If L can be written as $L = L_1 \cdot L_2$ for $L_1, L_2 \in \mathbb{K}(x)[\tau]$, then we call L_1 a* left hand factor *and L_2 a* right hand factor *of L.*

Chapter 3

Basic Theory About Difference Operators

3.1 Solutions of Difference Equations

Definition 3.1.1 *A (nonzero) sequence $u(0), u(1), ...$ of numbers, that means a function $u : \mathbb{N} \to \mathbb{K}$ is called a solution of a difference operator L iff $L(u) = 0$, i.e.*

$$(Lu)(x) = a_n(x)u(x+n) + ... + a_0(x)u(x) = 0$$

Again, we will use functions instead of sequences, thus, we define the kernel of L by

$$V(L) := \{u \in \mathbb{K}^{\mathbb{K}} \mid L(u) = 0\} \cong \{u \in \mathbb{K}^{\mathbb{N}} \mid L(u) = 0\}$$

Later on, we will be interested in spaces like $V(L) \cap \mathbb{K}[x]$, $V(L) \cap \mathbb{K}(x)$ or $V(L) \cap \mathbb{H}$, rather than in finding only a closed form expression for $u(x)$ resp. $u(k)$ - this would suffice to compute for example $u(1000000)$ without knowing $u(999999)$, $u(999998)$ and so on. In [Mal00] Ranjan Mallik presents such a closed form expression: The (in fact huge) formula is given in terms of the index k and $n = order(L)$ initial values and is even valid for arbitrary (complex) functions $a_i(x)$.

The following theorem sums up some of the most important facts about $V(L)$ - the solution space of a difference equation:

Theorem 3.1.2 *Let $L = a_n(x)\tau^n + ... + a_0(x)\tau^0$ with $a_k \in \mathbb{K}(x)$ and $a_0, a_n \neq 0$*

   (a) $V(L)$ is a vector space over $\mathbb{K}$

   (b) $\dim V(L) = n$

   (c) If u^* is a (particular) solution of $Ly = f$, then all solutions of $Ly = f$ are of the form $u^* + u$, where $u \in V(L)$. In other words: The solution space of $Ly = f$ is the affine space $u^* + V(L)$.

(d) $V(L) = V(r \cdot L)$ for every $r \in \mathbb{K}(x)^*$

Proof:

(a) This follows immediately from the linearity of L: Let $u_i \in V(L)$ and $c_i \in \mathbb{K}$, then
$$L(\sum c_i \cdot u_i) = \sum c_i \cdot L(u_i) = 0$$

(b) See [PWZ96], Theorem 8.2.1

(c) Let $Lu^* = f$ and $Lu = 0$

   - First, we show that $u + u^*$ solves $Lu = f$: $L(u + u^*) = L(u) + L(u^*) = 0 + f = f$
   - Conversely, let v be an arbitrary solution of $Ly = f$, then for $v - u^*$ we get: $L(v - u^*) = L(v) - L(u^*) = f - f = 0$, which implies that $v - u^* \in V(L)$, and that means $v = (v - u^*) + u^*$.

(d) Clear!

$\blacksquare$

Remark 3.1.3 *Remarks on Theorem 3.1.2*

   - *Although the fourth statement of Theorem 3.1.2 seems to be trivial, it enables us to multiply a given difference operator (resp. a homogeneous difference equation) by a rational function without changing the solution space. We will make use of this several times, either for making the difference operator monic, or in order to get polynomial instead of rational coefficients.*

   - *Obviously, there exists a $1 - 1$ correspondence between a hypergeometric function y with its certificate r and the difference equation of order 1 $(\tau - r)y = 0$.*

Example 3.1.4 *Consider $L = \tau - 2(x+1)$ with the solution $y(x) = c \cdot 2^x \cdot x! = c \cdot 2^x \cdot \Gamma(x+1)$.*

The following Theorem deals with the special case "constant coefficients":

Theorem 3.1.5 *Let $L = a_n \tau^n + ... + a_0 \tau^0$ with $a_k \in \mathbb{K}$, where $a_0, a_n \neq 0$ and let $\lambda_1, ..., \lambda_m \in \overline{\mathbb{K}}$ be the roots of the characteristic equation $L = 0$ with multiplicities $r_1, .., r_m$, respectively, then*

$$V(L) = \{\lambda_1^x, x\lambda_1^x, ..., x^{r_1}\lambda_1^x, ..., \lambda_m^x, x\lambda_m^x, ..., x^{r_m}\lambda_m^x\}$$

In other words, the general solution of $Ly = 0$ is given by

$$y(x) = \sum_{i=1}^{m} \lambda_i^x \cdot p_{r_i}(x)$$

where $p_{r_i}(x)$ denotes an arbitrary polynomial of degree r_i.
If all roots are distinct, then

$$y(x) = \sum_{i=1}^{n} c_i \lambda_i^x$$

with arbitrary $c_i \in \mathbb{K}$.

Proof: The special case with all roots are distinct follows directly from the factorization of $L = (\tau - \lambda_1)...(\tau - \lambda_n)$, which commutes as we are computing in the commutative ring $\mathbb{K}[\tau]$. For the general case it is sufficient to check that $y(x) = \lambda^x(c_0 + c_1 x + ... + c_r x^r)$ is the general solution of $(\tau - \lambda)^r y = 0$, which is no problem.

∎

It should be mentioned that using the powers of a companion matrix the general solution of a homogeneous difference equation with constant coefficients can be obtained without finding roots of the characteristic equation:
From Remark 2.3.3 we get

$$\tau(Y) = A \cdot Y \Rightarrow \tau^2(Y) = \tau(A \cdot Y) = \tau(A) \cdot \tau(Y) = A^2 \cdot Y$$

Analogously, we get

$$\tau^k(Y) = A^k \cdot Y \quad resp. \quad Y(k) = A^k \cdot Y(0)$$

where $Y(0)$ contains the initial values. Thus, knowing a formula for A^k suffices to get a closed form expression for the solution Y. Such formulas for A^k can be found in [Mal00] or in [ChLo96].

The next Lemma indicates the importance of right hand factors:

Lemma 3.1.6 *Let L_1 be a right hand factor of L, then $V(L_1) \subset V(L)$*

Proof: As L_1 is a right hand factor of L, there exists an difference operator L_0 such that $L = L_0 \cdot L_1$. Let $y \in V(L_1)$ then $Ly = L_0 \cdot L_1(y) = L_0(0) = 0$, thus $y \in V(L)$.

∎

3.2 Operations on Difference Operators

3.2.1 The Adjoint Operator

Definition 3.2.1 *Let $L = \sum_{k=0}^{n} a_k(x)\tau^k \in \mathbb{K}(x)[\tau]$ be normal, then the* adjoint *operator L^* is defined by*

$$L^* = \sum_{k=0}^{n} \tau^k a_{n-k}(x) = \sum_{k=0}^{n} a_{n-k}(x+k)\tau^k$$

Proposition 3.2.2 *Let $L, M \in \mathbb{K}(x)[\tau]$ and let $n = order(L)$, then*

(a) $(L^*)^* = \tau^n \cdot L \cdot \tau^{-n}$

(b) $(L \cdot M)^* = (\tau^n \cdot M^* \cdot \tau^{-n}) \cdot L^*$

Proof: Let's prove the first statement, thus let $L = \sum_{k=0}^{n} a_k(x)\tau^k$, then

$$
\begin{aligned}
(L^*)^* &= \Big(\sum_{k=0}^{n} a_{n-k}(x+k)\tau^k \Big)^* = \sum_{k=0}^{n} \tau^k a_{n-(n-k)}(x+(n-k)) = \\
&= \sum_{k=0}^{n} \tau^k \big(a_k(x+n-k) \big)\tau^k = \sum_{k=0}^{n} a_k(x+n)\tau^k = \\
&= \sum_{k=0}^{n} \tau^n \big(a_k(x)\tau^{k-n} \big) = \tau^n \sum_{k=0}^{n} a_k(x)\tau^k \tau^{-n} = \tau^n \cdot L \cdot \tau^{-n}
\end{aligned}
$$

∎

Remark 3.2.3 *Let $L = L_1 L_2$, that means that L has the left hand factor L_1 and the right hand factor L_2, then - see Proposition 3.2.2 - $L^* = (\tau^n \cdot L_2^* \cdot \tau^{-n}) \cdot L_1^*$. Thus L_1^* is a righthand factor of L^* and $\tau^n \cdot L_2^* \cdot \tau^{-n}$ is a left hand factor of L^*. We can simply say: Left hand factors of L correspond to right hand factors of L^*, and vice versa.*

Lemma 3.2.4 *Let $L = \tau - r$. If $y \in V(L)$, then $\frac{1}{\tau(y)} \in V(L^*)$*

Proof: Trivial, because by definition $L^* = \tau(r) \cdot \tau - 1$

∎

3.2.2   The Symmetric Product

Definition 3.2.5 *Let $L_1, L_2 \in \mathbb{K}(x)[\tau]$ be normal and at least of order 1. The symmetric product $L = L_1 \circledS L_2$ is defined as the normal monic operator of minimal order such that $y_1 y_2 \in V(L)$ for all $y_1 \in V(L_1)$ and $y_2 \in V(L_2)$.*

Remark 3.2.6 *Using the relation $L_1(y_1) = 0$ we can write any $\tau^i(y_1)$ as a $\mathbb{K}(x)$-linear combination of $\tau^0(y_1), ..., \tau^{n-1}(y_1)$ where n is the order of L_1, and similarly for the $\tau^i(y_2)$. Then L can be obtained by computing a $\mathbb{K}(x)$-linear combination of $y_1 y_2, \tau(y_1)\tau(y_2), ...$ which leads in general to a system of linear equations.*

As a special case we will need the following lemma later on:

Lemma 3.2.7 *If $L_1 = \tau - r$ with $r \in \mathbb{K}(x)^*$, $u \in \mathbb{H}$ with $L_1(u) = 0$, and L_2 is monic and normal then*

$$
L_1 \circledS L_2 = L_2 \circledS L_1 = \tau^{order(L_2)}(u) \cdot L_2 \cdot \frac{1}{u} \in \mathbb{K}(x)[\tau]
$$

Proof: First of all, it should be obvious that $L := \tau^{order(L_2)}(u) \cdot L_2 \cdot \frac{1}{u}$ is monic. Let $y_2 \in V(L_2)$ then $L(u \cdot y_2) = (\tau^{order(L_2)}(u) \cdot L_2)(y_2) = 0$. Therefore L has the same solution space as $L_1 \textcircled{S} L_2$.

It remains to prove that $L \in \mathbb{K}(x)[\tau]$: Note that $L_2 \cdot v \in v \cdot \mathbb{K}(x)[\tau]$ for every $v \in \mathbb{H}$, so $\tau^{order(L_2)}(u) \cdot L_2 \cdot \frac{1}{u} \in \tau^{order(L_2)}(u) \cdot \frac{1}{u} \cdot \mathbb{K}(x)[\tau] = \mathbb{K}(x)[\tau]$, because $\tau^{order(L_2)}(u) \cdot \frac{1}{u} \in \mathbb{K}(x)$.

$\blacksquare$

Remark 3.2.8 *Of course the symmetric product in Lemma 3.2.7 can be computed without the solution u of $\tau - r$: Let $L = \sum_{k=0}^n p_k \tau^k$ with $p_0, p_n \neq 0$ then (remember Proposition 2.2.14 (g))*

$$
\begin{aligned}
L \textcircled{S} (\tau - r) &= \tau^n(u) \cdot L \cdot \frac{1}{u} = \tau^n(u) \cdot \sum_{k=0}^n p_k \tau^k \cdot \frac{1}{u} = \sum_{k=0}^n p_k \tau^n(u) \frac{1}{\tau^k(u)} \tau^k = \\
&= \sum_{k=0}^n p_k \frac{\tau^n(u)}{u} \frac{u}{\tau^k(u)} \tau^k = \sum_{k=0}^n p_k \frac{\tau^{n-1}(r) \cdot \ldots \cdot \tau(r) \cdot r}{\tau^{k-1}(r) \cdot \ldots \cdot \tau(r) \cdot r} \tau^k = \\
&= \sum_{k=0}^n p_k \tau^k(r) \cdot \tau^{k+1}(r) \cdot \ldots \cdot \tau^{n-1}(r) \tau^k = \sum_{k=0}^n p_k \prod_{j=k}^{n-1} \tau^j(r) \tau^k
\end{aligned}
$$

Analogously, we get

$$
L \textcircled{S} \left(\tau - \frac{1}{r} \right) = \sum_{k=0}^n p_k \prod_{j=k}^{n-1} \frac{1}{\tau^j(r)} \tau^k = \frac{1}{\prod_{j=0}^{n-1} \tau^j(r)} \cdot \sum_{k=0}^n p_k \prod_{j=0}^{k-1} \tau^j(r) \tau^k
$$

Proposition 3.2.9 *The set of all normal and monic difference operators with the symmetric product form a (commutative) monoid with the neutral element $\tau - 1$. Difference operators of the form $\tau - r$ with $r \neq 0$ have the inverse element $\tau - \frac{1}{r}$.*

Proof: Just do easy checks!

3.2.3 The Greatest Common Right Divisor

Definition 3.2.10 Right hand division: *Let $F(\tau), G(\tau) \in \mathbb{K}(x)[\tau]$, with degrees n and m, respectively. Then there exists unique polynomials $Q(\tau)$ and $R(\tau)$ with*

$$
F(\tau) = Q(\tau) \cdot G(\tau) + R(\tau)
$$

where the degree of $Q(\tau)$ is $n - m$ and the degree of $R(\tau)$ does not exceed $m - 1$.

Like in the commutative case we can now perform the Euclidian Algorithm which yields the *greatest common right-divisor* (GCRD) of two polynomials:

Definition 3.2.11 *Let $F_1(\tau), F_2(\tau) \in \mathbb{K}(x)[\tau]$, and let*

$$
\begin{aligned}
F_1(\tau) &= Q_1(\tau) \cdot F_2(\tau) + F_3(\tau) \\
F_2(\tau) &= Q_2(\tau) \cdot F_3(\tau) + F_4(\tau) \\
&\quad \ldots \\
F_{n-2}(\tau) &= Q_{n-2}(\tau) \cdot F_{n-1}(\tau) + F_n(\tau) \\
F_{n-1}(\tau) &= Q_{n-1}(\tau) \cdot F_n(\tau)
\end{aligned}
\tag{3.1}
$$

be the Euclidian Algorithm then we define:

$$
\mathrm{GCRD}(F_1, F_2) := F_n
$$

The $\mathrm{GCRD}(F_1, F_2)$ is defined as the polynomial of maximal degree, which is a right hand factor of both, F_1 and F_2. Requiring that the GCRD is monic makes it uniquely defined.

3.2.4   The Lowest Common Left Multiple

Definition 3.2.12 *Let $F(\tau), G(\tau) \in \mathbb{K}(x)[\tau]$. The least common left multiple of F and G $\mathrm{LCLM}(F, G)$ is defined as the polynomial M of lowest degree which is right hand divisible by both, F and G. Requiring that M is monic makes it uniquely defined.*

Theorem 3.2.13 *Let a Euclidian Algorithm of $F_1(\tau)$ and $F_2(\tau)$ of the form (3.1) is given, then*

$$
\mathrm{LCLM}(F_1, F_2) = F_{n-1}(\tau) \cdot \frac{1}{F_n(\tau)} \cdot F_{n-2}(\tau) \cdot \frac{1}{F_{n-1}(\tau)} \cdot \ldots \cdot F_2(\tau) \cdot \frac{1}{F_3(\tau)} \cdot F_1(\tau)
$$

Proof: See [Ore33], Theorem 8

Corollary 3.2.14 *Let $F_1(\tau), F_2(\tau) \in \mathbb{K}(x)[\tau]$, then:*

$$
\deg F_1 + \deg F_2 = \deg \mathrm{GCRD}(F_1, F_2) + \deg \mathrm{LCLM}(F_1, F_2)
$$

Proof: Follows directly from the formula in Theorem 3.2.13

Corollary 3.2.15 *Let $r \neq s$ nonzero rational functions, then:*

$$
\mathrm{LCLM}(\tau - r, \tau - s) = \frac{1}{\tau(s) - \tau(r)} \cdot \tau^2 - \left(\frac{s}{s - r} + \frac{\tau(r)}{\tau(s) - \tau(r)} \right) \cdot \tau + \frac{rs}{s - r}
$$

Proof: Let's do first the GCRD-Computation (of course the GCRD has to be constant with respect to τ as $r \neq s$):

$$
\begin{aligned}
\tau - r &= 1 \cdot (\tau - s) + s - r \\
\tau - s &= \left(\frac{1}{\tau(s) - \tau(r)} \tau - \frac{s}{s - r} \right) \cdot (s - r)
\end{aligned}
$$

This leads to the following LCLM-Computation:

$$\text{LCLM}(\tau - r, \tau - s) = (\tau - s) \cdot \frac{1}{s - r} \cdot (\tau - r) =$$

$$= \ \left(\tau(\frac{1}{s - r}) \cdot \tau - \frac{s}{s - r}\right) \cdot (\tau - r) =$$

$$= \ \tau(\frac{1}{s - r}) \cdot \tau^2 - \frac{s}{s - r} \cdot \tau - \tau(\frac{1}{s - r}) \cdot \tau(r) \cdot \tau + \frac{rs}{s - r} =$$

$$= \ \frac{1}{\tau(s) - \tau(r)} \cdot \tau^2 - (\frac{s}{s - r} + \frac{\tau(r)}{\tau(s) - \tau(r)}) \cdot \tau + \frac{rs}{s - r}$$

■

Corollary 3.2.16 *Let $L \in \mathbb{K}(x)[\tau]$ and let $R \in \mathbb{K}(x)$ be the remainder of the right hand division of L and $\tau - r$ with $r \in \mathbb{K}(x)^*$, then (for $R \neq 0$)*

$$\text{LCLM}(L, \tau - r) = (\tau - r) \cdot \frac{1}{R} \cdot L = \left(\frac{1}{\tau(R)}\tau - \frac{r}{R}\right) \cdot L$$

Proof: Just like before!

■

Remark 3.2.17 *By Lemma 3.1.6 we realize that Corollary 3.2.16 provides an algorithm to construct - step by step - a difference equation (resp. difference operator) which has exactly some given solutions.*

Example 3.2.18 *How does the (normalized) difference operator with the solutions 3^x and $x!$ look like? We only have to compute $\text{LCLM}(\tau - 3, \tau - (x + 1))$ with our LCLM-formula:*

$$\text{LCLM}(\tau - 3, \tau - (x + 1)) = \frac{1}{x - 1} \cdot \tau^2 - (\frac{3}{x - 1} + \frac{x + 1}{x - 2}) \cdot \tau + \frac{3 \cdot (x + 1)}{x - 2}$$

Normalization yields

$$\tau^2 - \frac{x^2 + 3x - 7}{x - 2} \cdot \tau + \frac{3(x + 1)(x - 1)}{x - 2}$$

or without denominators as a polynomial in x and τ:

$$(x - 2) \cdot \tau^2 - (x^2 + 3x - 7) \cdot \tau + 3(x + 1)(x - 1)$$

Summing up all facts of this chapter we obtain (remember Lemma 3.1.6):

Theorem 3.2.19 *Let L_1, L_2 be difference operators, then:*

(a) $V(L_1 \circledS L_2) = V(L_1) \cdot V(L_2)$

(b) $V(\text{GCRD}(L_1, L_2)) = V(L_1) \cap V(L_2)$

(c) $V(\mathrm{LCLM}(L_1, L_2)) = V(L_1) \cup V(L_2)$

(d) L_1 *is a right hand factor of* L_2 *if and only* $V(L_1) \subset V(L_2)$

Proof: ad *(d)*: The implication from the right to the left follows directly from *(b)* and the definition of the GCRD: $\mathrm{GCRD}(L_1, L_2) = L_1$

$\blacksquare$

Remark 3.2.20 *Note also the following special case of Theorem 3.2.19 (d): Let* $(\tau - r)y = 0$, *then* $\tau - r$ *is a right hand factor of a difference operator* L *if and only if* $y \in V(L)$. *Thus, we have a 1-1 correspondence between hypergeometric solutions and monic first order right hand factors.*

Chapter 4

Polynomial Solutions

4.1 The Basic Idea

Problem 4.1.1 *We are given the following problem: Find all (nonzero) polynomial solutions $y \in \mathbb{K}[x]$ of $Ly = f$, where we suppose the following:*

- $L = \sum_{k=0}^{n} p_k(x) \cdot \tau^k$

- $p_k(x) \in \mathbb{K}[x]$

- $p_n \neq 0$, $p_0 \neq 0$

- $f \in \mathbb{K}[x]$

The general idea in order to solve $Ly = f$ is very simple:

1. Find an upper bound N for the possible degree of polynomial solutions of $Ly = f$.

2. Given N, describe all polynomial solutions having degree at most N.

In the following two sections we will present Sergei Abramov's (in [Abr89a]) and Marko Petkovšek's (in [Pet92]) approaches to determine a degree bound N. Comparing both methods we will see later on (section 4.6) that they are equivalent and therefore lead to the same bound.
As the method of undetermined coefficient for the second step is usually used, we have to solve a linear system of equations with $N + 1$ unknowns afterwards. In sections 4.4 and 4.5 we will show how to minimize this number of unknowns.

4.2   Abramov and the Delta-Operator

Abramov's idea was to (re)write the difference operator L in terms of Δ instead of τ:

$$\sum_{k=0}^{n} p_k(x) \cdot \tau^k = \sum_{k=0}^{n} p_k(x) \cdot (\Delta + 1)^k = \sum_{k=0}^{n} q_k(x) \cdot \Delta^k$$

where

$$q_k(x) := \sum_{i=k}^{n} \binom{i}{k} p_i(x) \tag{4.1}$$

We will now show, how to find a degree bound N for a polynomial solution of $Ly = f$: Suppose $y(x) = \sum_{m=0}^{N} a_m x^m$ with $a_m \in \mathbb{K}$ and $a_N \neq 0$ then:

$$
\begin{aligned}
\deg Ly & = \deg\left(\sum_{k=0}^{n} q_k(x) \cdot \Delta^k \left(\sum_{m=0}^{N} a_m x^m \right) \right) \\
& \leq \max_{0 \leq k \leq n} \left(\deg q_k(x) + \deg \Delta^k \left(\sum_{m=0}^{N} a_m x^m \right) \right) \\
& = \max_{0 \leq k \leq n} \left(\deg q_k(x) + (N - k) \right) \\
& = \max_{0 \leq k \leq n} \left(\deg q_k(x) - k \right) + N
\end{aligned}
$$

Let

$$b := \max_{0 \leq k \leq n} \left(\deg q_k(x) - k \right)$$

then we have to consider two cases:

- $f = 0 \Rightarrow b + N < 0 \Leftrightarrow b + N \leq -1 \Leftrightarrow N \leq -b - 1$

- $f \neq 0 \Leftrightarrow \deg f \geq 0$: In this case we consider two subcases:

  - $\deg Ly = N + b \Rightarrow N + b = \deg f \Leftrightarrow N = \deg f - b$
  - $\deg Ly < N + b$: This means that the coefficient of x^{N+b} vanishes, thus

$$a_N \sum_{\substack{k=0 \\ deg(q_k)-k=b}}^{n} lc(q_k(x)) \cdot N^{\underline{k}} = 0$$

  Defining $q_{k,j}$ as the coefficient of x^j of $q_k(x)$, we have

$$a_N \sum_{k=0}^{n} q_{k,b+k} \cdot N^{\underline{k}} = 0$$

  Because $a_N \neq 0$, this means that N is a (integer) root of the *degree polynomial*

$$\alpha(z) := \sum_{j=0}^{n} q_{j,b+j} \cdot z^{\underline{j}} = 0$$

By these considerations we have proven the following theorem:

Theorem 4.2.1 *Let*

$$\sum_{k=0}^{n} q_k(x) \cdot \Delta^k(y) = f$$

where y is a polynomial and let $q_{k,j}$ be the coefficient of x^j in $q_k(x)$. Let

$$b := \max_{0 \le k \le n} \left(\deg q_k(x) - k \right)$$

and

$$\alpha(z) := \sum_{j=0}^{n} q_{j,b+j} \cdot z^{\underline{j}}$$

additionally. Then $\deg y \le N$, where

$$N := \max \left(\{ z \in \mathbb{N} : \alpha(z) = 0 \} \cup \{ \deg f - b, -b - 1 \} \right)$$

The resulting algorithm is as follows:

Algorithm 4.2.2 *by Sergei Abramov*
INPUT: $Ly = f$ with

- $L = \sum_{k=0}^{n} p_k \cdot \tau^k$ *where $p_k \in \mathbb{K}[x]$*

- $p_0, p_n \ne 0$

- $f \in \mathbb{K}[x]$

OUTPUT: The general polynomial solution over $\mathbb{K}$ of $Ly = f$

BEGIN

    FOR $k \in \{0, 1, ..., n\}$ DO $q_k := \sum_{i=k}^{n} \binom{i}{k} p_i$

    $b := \max\{ \deg q_k - k \mid 0 \le k \le n \}$

    $N := \max(\{ z \in \mathbb{N} : \sum_{j=0}^{n} q_{j,b+j} \cdot z^{\underline{j}} = 0 \} \cup \{ \deg f - b, -b - 1 \})$

    IF $N \in \mathbb{N}$

        THEN Use the method of undetermined coefficients to find the general polynomial solution over $\mathbb{K}$ of degree at most N of $Ly = f$

        ELSE Return $\emptyset$

END

$\blacklozenge$

Remark 4.2.3 *Remarks on Algorithm 4.2.2:*

- *It was first shown in [PeWe00] that the $-b-1$ is not necessary at all in order to compute the degree bound N! We will come back to that interesting fact in section 4.6.*

- *Observe that using the method of undetermined coefficients yields a linear system of equations which is already in triangular form.*

- *Of course, it may be that the algorithm computes a degree bound $N \geq 0$ though no nonzero polynomial solution exists, i.e. the final linear system has either no or only the trivial solution. See Example 4.7.3.*

4.3   Petkovšek and the Shift-Operator

Let's follow Petkovšek's idea in [Pet92] which is just *the* straight forward method - however, this approach is not as simple and short as Abramov's one:

Suppose we have (at the beginning we restrict ourselves to the homogeneous case $Ly = 0$):

$$L \;=\; \sum_{i=0}^{n} p_i(x) \cdot \tau^i$$

$$p_i(x) \;=\; \sum_{j=0}^{d} p_{i,j} \cdot x^j \;\; \text{where} \;\; d := degree(L) = \max_{0 \leq i \leq n} \deg p_i$$

and suppose $y(x) = \sum_{k=0}^{N} a_k \cdot x^k$ (with $a_N \neq 0$) is a solution of $Ly = 0$, then:

$$\begin{aligned}
Ly \;&=\; \sum_{i=0}^{n} \sum_{j=0}^{d} p_{i,j} \cdot x^j \cdot \tau^i \Big(\sum_{k=0}^{N} a_k \cdot x^k \Big) \;=\; \\
&=\; \sum_{i=0}^{n} \sum_{j=0}^{d} \sum_{k=0}^{N} p_{i,j} \cdot a_k \cdot (x+i)^k \cdot x^j \;=\; \\
&=\; \sum_{i=0}^{n} \sum_{j=0}^{d} \sum_{k=0}^{N} p_{i,j} a_k \sum_{l=0}^{k} \binom{k}{l} i^{k-l} x^l x^j \;=\; \\
&=\; \sum_{i=0}^{n} \sum_{j=0}^{d} \sum_{k=0}^{N} \sum_{l=0}^{k} p_{i,j} a_k \binom{k}{l} i^{k-l} x^{l+j} \;=\; 0
\end{aligned}$$

Setting $j = r - l$ (observe the bounds for r!) and changing the order of summation yields:

$$\sum_{r=0}^{d+N} x^r \sum_{k=0}^{N} a_k \sum_{l=0}^{k} \binom{k}{l} \sum_{i=0}^{n} i^{k-l} p_{i,r-l} = 0$$

By comparing coefficients and setting $l = k - j$ we get (using also the symmetry-property of the binomial coefficient):

$$\forall r \in \{0, 1, ..., d + N\}: \quad \sum_{k=0}^{N} a_k \sum_{j=0}^{k} \binom{k}{j} \sum_{i=0}^{n} i^j p_{i,r-k+j} = 0 \qquad (4.2)$$

Step 0: First, we look at $r = d + N$ get (note that $p_{i,d+j} = 0$ for $j > 0$)

$$a_N \sum_{j=0}^{N} \binom{N}{j} \sum_{i=0}^{n} i^j p_{i,d+N-N+j} = a_N \binom{N}{0} \sum_{i=0}^{n} p_{i,d} = a_N \sum_{i=0}^{n} p_{i,d}$$

By assumption $a_N \neq 0$, thus we come up with the first necessary condition[1] for the existence of a polynomial solution:

$$\sum_{i=0}^{n} p_{i,d} = 0 \qquad (4.3)$$

Step 1: Let's go back to (4.2) and take a look at $r = d + N - 1$ (again note $p_{i,d+N-k+j} = 0$ for $N - k + j > 0$):

$$\begin{aligned}
0 &= \sum_{k=N-1}^{d} a_k \sum_{j=0}^{1} \binom{k}{j} \sum_{i=0}^{n} i^j p_{i,d+N-1-k+j} = \\
&= a_{N-1} \binom{N-1}{0} \underbrace{\sum_{i=0}^{n} p_{i,d}}_{0 \text{ by } (4.3)} + a_N \sum_{j=0}^{1} \binom{N}{j} \sum_{i=0}^{n} i^j p_{i,d-1+j} = \\
&= a_N \left(N \cdot \sum_{i=0}^{n} i p_{i,d} + \sum_{i=0}^{n} p_{i,d-1} \right)
\end{aligned}$$

The last equation now implies two possibilities: Either N is the (positive integer) root of the linear equation

$$N \cdot \sum_{i=0}^{n} i p_{i,d} + \sum_{i=0}^{n} p_{i,d-1} = 0$$

and we are done or

$$\sum_{i=0}^{n} i p_{i,d} = \sum_{i=0}^{n} p_{i,d-1} = 0 \qquad (4.4)$$

In the latter case we can go back to (4.2) and take a look at $r = d + N - 2$:

---

[1] We will see that this condition can also used in the rational case. See the "Fast negative criterion" on page 64

Step 2:

$$
\begin{aligned}
0 &= \sum_{k=N-2}^{N} a_k \sum_{j=0}^{2} \binom{k}{j} \sum_{i=0}^{n} i^j p_{i,d+N-2-k+j} = \\
&= a_{N-2} \binom{N-2}{0} \underbrace{\sum_{i=0}^{n} p_{i,d}}_{0 \text{ by } (4.3)} + \sum_{k=N-1}^{N} a_k \sum_{j=0}^{2} \binom{k}{j} \sum_{i=0}^{n} i^j p_{i,d+N-2-k+j} = \\
&= a_{N-1} \underbrace{\sum_{j=0}^{1} \binom{N-1}{j} \sum_{i=0}^{n} i^j p_{i,d-1+j}}_{0 \text{ by } (4.4)} + a_N \sum_{j=0}^{2} \binom{N}{j} \sum_{i=0}^{n} i^j p_{i,d-2+j} = \\
&= a_N \left(\binom{N}{2} \sum_{i=0}^{n} i^2 p_{i,d} + \binom{N}{1} \sum_{i=0}^{n} i p_{i,d-1} + \binom{N}{0} \sum_{i=0}^{n} p_{i,d-2} \right)
\end{aligned}
$$

Again, there are two possibilities: Either N is a (positive integer) root of the quadratic equation

$$
\binom{N}{2} \sum_{i=0}^{n} i^2 p_{i,d} + \binom{N}{1} \sum_{i=0}^{n} i p_{i,d-1} + \binom{N}{0} \sum_{i=0}^{n} p_{i,d-2} = 0
$$

and we are done or

$$
\sum_{i=0}^{n} i^2 p_{i,d} = \sum_{i=0}^{n} i p_{i,d-1} = \sum_{i=0}^{n} p_{i,d-2} = 0
$$

Termination Step: This procedure can be done on and on, but - at the latest - after n steps it terminates, that means we show, that not all appearing equations can vanish identically! After s steps we come up with the following equation (we set $p_{i,d-s}$ to zero when $d - s < 0$):

$$
\binom{N}{s} \sum_{i=0}^{n} i^s p_{i,d} + \binom{N}{s-1} \sum_{i=0}^{n} i^{s-1} p_{i,d-1} + \ldots + \binom{N}{0} \sum_{i=0}^{n} p_{i,d-s} = 0 \qquad (4.5)
$$

Obviously, all appearing equations up to the n-th contain the leading coefficients

$$
\sum_{i=0}^{n} i^s p_{i,d} \quad \text{for} \ \ 0 \le s \le n
$$

If all those were zero, then we would get a linear system of equations for $p_{0,d}, \ldots, p_{n,d}$ with the Vandermonde matrix $(i^k)_{i,k=0}^{n}$. Thus, all $p_{i,d}$ would be zero which would be a contradiction to the definition of d.

Inhomogeneous Step: The inhomogeneous case does not make any problems! Suppose we have found (the first) s, such that (4.5) does not vanish identically and we found possible values for N. This means that the coefficient of x^{d+N-s} of Ly is zero. On the other hand this coefficient may be equal to the leading coefficient of f which implies

$$d + N - s = \deg f \Leftrightarrow N = \deg f + s - d$$

Let's formulate the resulting algorithm:

Algorithm 4.3.1 (PolyTau) *by Marko Petkovšek*
INPUT: $Ly = f$ with

- $L = \sum_{i=0}^{n} p_i \cdot \tau^i$ *where* $p_i = \sum_{j=0}^{d} p_{i,j} \cdot x^j$ *and* $d := \max_{0 \leq i \leq n} \deg p_i$

- $p_0, p_n \neq 0$

- $f \in \mathbb{K}[x]$

OUTPUT: The general polynomial solution over $\mathbb{K}$ of $Ly = f$

BEGIN

    s:=-1

    REPEAT

        s:=s+1
        FOR $j \in \{0, 1, ..., s\}$ DO $b_j^{(s)} := \sum_{i=0}^{n} i^j p_{i,d-s+j}$

    UNTIL $\exists j \in \{0, 1, ..., s\}$ such that $b_j^{(s)} \neq 0$

    $N := \max(\{z \in \mathbb{N} : D(z) := \sum_{j=0}^{s} \binom{z}{j} b_j^{(s)} = 0\} \cup \{\deg f + s - d\})$

    IF $d \in \mathbb{N}$

        THEN Use the method of undetermined coefficients to find the general polynomial solution over $\mathbb{K}$ of degree at most N of $Ly = f$
        ELSE Return $\emptyset$

END

                                                                ♦

Remark 4.3.2 *Remarks on Algorithm 4.3.1:*

- *The polynomial $D(z)$ is again called degree polynomial (of $Ly = 0$).*

- *Observe that using the method of undetermined coefficients yields a linear system of equations which is already in triangular form.*

- *Of course, it may again be that the algorithm computes a degree bound $N \geq 0$ though no polynomial solution exists, i.e. the final linear system has either no or only the trivial solution.*

- *A MATHEMATICA-version of Algorithm 4.3.1 (over $\mathbb{C}$) has been implemented by Marko Petkovšek and is part of the package "Hyper" which is available at his homepage (http://www.fmf.uni-lj.si/∼petkovsek/).*

4.4 Minimizing The Number Of Variables

In this section we will follow [ABP95], which describes in a very general way the search for polynomial solutions for differential, difference and q-difference equations.

The critical part of the algorithms of Abramov and Petkovšek is the method of undetermined coefficients, because the number of unknown coefficients, $N+1$, is often much larger than the order of the equation, n. We will show how two reduce the procedure to a system of $n+d$ equations (where d denotes the degree of the difference equation) with only n unknowns.

It will be convenient to write the difference operator L again in terms of Δ. Moreover, in this section we will use "r" instead of "n" for the order of L:

Problem 4.4.1 *We have given the following problem: Find a (polynomial) solution $y \in \mathbb{K}[x]$ of $Ly = f$, where we suppose the following:*

- $L = \sum_{k=0}^{r} q_k(x) \cdot \Delta^k$

- $p_k(x) \in \mathbb{K}[x]$

- $p_r \neq 0$, $p_0 \neq 0$

- $f \in \mathbb{K}[x]$

- $d = degree(L) = \max\{\deg q_k(x) \mid 0 \leq k \leq r\}$

The main idea of Abramov, Bronstein and Petkovšek is the extension from polynomials to formal power series. At the beginning we look for a formal power series solution of $Ly = f$, later we use an upper bound for the degree of a polynomial solution (which will be very similar to the previous algorithms) and cut the formal power series. The second main difference between this approach and the last will be the use of another power basis, not the usual x^n.

We will divide the derivation into three steps.

- Step 1: Define an appropriate power basis and extend it to formal power series

- Step 2: Transform $Ly = f$ into an equivalent recurrence

- Step 3: Find an algorithm to solve the new recurrence

Step 1: Define an appropriate power basis We need an (infinite) sequence of polynomials $(P_n(x))$ from $\mathbb{K}[x]$ such that

(P1) $\deg P_n = n$ for $n \in \mathbb{N}$

(P2) $P_m \mid P_n$ for $0 \le m < n$

(P3) Let $l_n : \mathbb{K}[x] \to \mathbb{K}$ linear functionals such that $l_n(P_m) = \delta_{mn}$.

(P4) Given a difference operator L, there must exist $A, B \in \mathbb{N}$ and elements $\alpha_i(n) \in \mathbb{K}$ for $n \in \mathbb{N}$ and $i \in \{-A, -A+1, ..., B\}$, such that

$$LP_n = \sum_{i=-A}^{B} \alpha_i(n) P_{n+i} \tag{4.6}$$

with the restriction that $\alpha_{-A}(n)$ is not identically zero. We take P_k to be zero when $k < 0$.

Remark 4.4.2 *Because of (P1), (P2) and (P3) the set $\{P_0, P_1, ...\}$ is a basis of the $\mathbb{K}$-linear space $\mathbb{K}[x]$ and every polynomial from $\mathbb{K}[x]$ has a unique expansion in terms of polynomials $P_n(x)$. Furthermore*

$$p(x) = \sum_{n=0}^{\deg p} l_n(p) P_n(x) \ \text{for all } p \in \mathbb{K}[x] \tag{4.7}$$

Example 4.4.3 *We consider two possible bases:*

(a) $P_n(x) = \frac{(x-a)^n}{n!}$ where $a \in \mathbb{K}$ is arbitrary. Then $l_n(p) = p^{(n)}(a)$ and (4.7) is Taylor's formula for $p(x)$. A problem appears only for property (P4) - this power basis does not satisfy it!

(b) $P_n(x) = \binom{x-a}{n}$ where $a \in \mathbb{K}$ is arbitrary. Then $l_n(p) = \Delta^n(p(a))$ and (4.7) is Newton's interpolation formula for $p(x)$. In the following we will show that this power basis satisfies (P4).

We start with a variant of the the so-called Chu-Vandermonde identity:

$$\binom{x}{n} = \sum_{i \in \mathbb{Z}} \binom{x-m}{i-m} \binom{m}{m+n-i}$$

Multiplying by $\binom{x}{m}$ and extracting/revising binomials yields

$$\binom{x}{m} \binom{x}{n} = \sum_{i \in \mathbb{Z}} \frac{(x-m)!}{(i-m)! \cdot (x-i)!} \cdot \binom{m}{i-n} \cdot \frac{x!}{m! \cdot (x-m)!}$$

Canceling $(x-m)!$, extending by $i!$ and reordering yields

$$\binom{x}{m} \binom{x}{n} = \sum_{i \in \mathbb{Z}} \frac{i!}{m! \cdot (i-m)!} \cdot \binom{m}{i-n} \cdot \frac{x!}{i! \cdot (x-i)!}$$

By simplifying to binomials and replacing x by $x - a$, we finally get

$$P_m P_n = \sum_{i \in \mathbb{Z}} \binom{i}{m} \binom{m}{i-n} P_i \tag{4.8}$$

On the other hand we have

$$LP_n = \sum_{k=0}^{r} q_k \Delta^k P_n = \sum_{k=0}^{r} q_k P_{n-k}$$

Using (4.7) on q_k we get

$$LP_n = \sum_{k=0}^{r} \sum_{j=0}^{\deg q_k} l_j(q_k) P_j P_{n-k} = \sum_{k=0}^{r} \sum_{j=0}^{d} l_j(q_k) P_j P_{n-k}$$

Now we use (4.8) which yields

$$
\begin{aligned}
LP_n &= \sum_{k=0}^{r} \sum_{j=0}^{d} l_j(q_k) \sum_{i \in \mathbb{Z}} \binom{i}{j} \binom{j}{i+k-n} P_i = \\
&= \sum_{k=0}^{r} \sum_{j=0}^{d} l_j(q_k) \sum_{i \in \mathbb{Z}} \binom{n+i}{j} \binom{j}{i+k} P_{n+i}
\end{aligned}
$$

Now the second binomial coefficient is zero if $i + k < 0$, which is always the case if $i < -r$. Furthermore this binomial coefficient is also zero for $i + k > j$, which is always the case if $i > d$. Therefore:

$$
\begin{aligned}
LP_n &= \sum_{k=0}^{r} \sum_{j=0}^{d} l_j(q_k) \sum_{i=-r}^{d} \binom{n+i}{j} \binom{j}{i+k} P_{n+i} = \\
&= \sum_{i=-r}^{d} \sum_{k=0}^{r} \sum_{j=0}^{d} \binom{n+i}{j} \binom{j}{i+k} l_j(q_k) P_{n+i}
\end{aligned}
$$

Comparing this with equation (4.6) of (P4) we read off

$$
\begin{aligned}
A &= r \\
B &= d \\
\alpha_i(n) &= \sum_{k=0}^{r} \sum_{j=0}^{d} \binom{n+i}{j} \binom{j}{i+k} l_j(q_k) \tag{4.9}
\end{aligned}
$$

Remark 4.4.4 *For sake of simplicity we will go on writing P_n, l_n and $\alpha_i(n)$ instead of $\binom{x-a}{n}$, $\Delta^n|_a$ and (4.9), respectively. We will also write A instead of r and B instead of d.*

Before we can proceed to step 2, we have to extend (4.7) to formal power series: Let $\mathbb{K}[[P_n]]$ denote the algebra of formal power series of the form

$$y(x) = \sum_{n=0}^{\infty} c_n P_n(x)$$

where $c_n \in \mathbb{K}$ and let $\tilde{l}_n : \mathbb{K}[[P_n]] \to \mathbb{K}$ be linear functionals (the extensions of the l_n) such that $\tilde{l}_n(s(x)) = c_n$.

Step 2: Transformation to an equivalent recurrence We will now suppose that the solution y of $Ly = f$ is a formal power series and we will transform $Ly = f$ to other equivalent equations. Afterwards we will search for a degree bound for y which enables us to show a theorem on which our algorithm will be based.

$$
\begin{aligned}
Ly = f \quad &\Leftrightarrow \quad L\Big(\sum_{n=0}^{\infty} \tilde{l}_n(y) P_n\Big) = f \\
&\Leftrightarrow \quad \sum_{n=0}^{\infty} \tilde{l}_n(y) L P_n = f \\
&\Leftrightarrow \quad \sum_{n=0}^{\infty} \tilde{l}_n(y) \sum_{i=-A}^{B} \alpha_i(n) P_{n+i} = f \\
&\Leftrightarrow \quad \sum_{n=0}^{\infty} \sum_{i=-A}^{B} \alpha_i(n) \tilde{l}_n(y) P_{n+i} = f \\
&\Leftrightarrow \quad \sum_{n=0}^{\infty} \sum_{i=-A}^{B} \alpha_i(n) \tilde{l}_n(y) P_{n+i} + \underbrace{\sum_{n=0}^{B-1} \sum_{i=n}^{B} \alpha_i(n-i) \tilde{l}_{n-i}(y)}_{=0} = f
\end{aligned}
$$

The second double sum is zero by defining $\tilde{l}_k$ to be zero if $k < 0$. Now shifting n by $-i$ and reordering yields

$$Ly = f \Leftrightarrow \sum_{n=0}^{\infty} \Big(\sum_{i=-A}^{B} \alpha_i(n-i) \tilde{l}_{n-i}(y) \Big) P_n = f$$

Now the following step seems to be natural - comparing coefficients, which means (after the substitution $-i$ for i):

$$\sum_{i=-B}^{A} \alpha_{-i}(n+i) \tilde{l}_{n+i}(y) = l_n(f) \text{ for all } n \geq 0 \tag{4.10}$$

Now we gained a recurrence (in n, not in x as we are used) for the unknown sequence $\langle \tilde{l}_n(y) \rangle \in \mathbb{K}$ of order $A + B$ (in fact $A + b$, where $b \leq B$ is considered

in the following Lemma 4.4.5) with leading coefficient $\alpha_{-A}(n + A)$. Note that the recurrence is homogeneous for $n > \deg f$.

Lemma 4.4.5 *Let* $\alpha_i(n)$ *be defined by (4.9), then*

$$b := \max\{\deg q_k - k \mid 0 \le k \le r\} = \max\{i \in \mathbb{Z} \mid \alpha_i(n) \not\equiv 0\}$$

Proof: In the following lines keep the possible values for j, k and n in mind!

$$
\begin{aligned}
&\quad \max\{i \in \mathbb{Z} \mid \alpha_i(n) \not\equiv 0\} = \\
&= \max\{i \in \mathbb{Z} \mid \exists n \ge 0 : \alpha_i(n) \ne 0\} = \\
&= \max\left\{i \in \mathbb{Z} \mid \exists j, k, n : \binom{n+i}{j} \ne 0 \wedge \binom{j}{i+k} \ne 0 \wedge l_j(q_k) \ne 0\right\} = \\
&= \max\left\{i \in \mathbb{Z} \mid \exists j, k : \binom{j}{i+k} \ne 0 \wedge l_j(q_k) \ne 0\right\} = \\
&= \max\{i \in \mathbb{Z} \mid \exists j, k : j \ge i + k \wedge j \ge \deg q_k\} = \\
&= \max\{i \in \mathbb{Z} \mid \exists j, k : j = i + k \wedge j \ge \deg q_k\} = \\
&= \max\{i \in \mathbb{Z} \mid \exists j : j = \deg q_{j-i}\} = \\
&= \max\{i \in \mathbb{Z} \mid \exists k : k + i = \deg q_k\} = \max\{\deg q_k - k \mid 0 \le k \le r\}
\end{aligned}
$$

$\blacksquare$

With this b - which is in fact the same as in Abramov's Algorithm 4.2.2 - and the corresponding polynomial $\alpha_b(n)$ we can now prove the degree bound for a polynomial solution. Note the similarity to Theorem 4.2.1!

Theorem 4.4.6 *Let* $Ly = f$ *where* y *is a polynomial and let* b *as in Lemma 4.4.5. Then* $\deg y \le N$, *where*

$$N = \max(\{z \in \mathbb{N} : \alpha_b(z) = 0\} \cup \{\deg f - b, -b - 1\})$$

Proof: We distinguish two cases:

- Case 1: $N + b < 0 \Rightarrow N < -b \Rightarrow N \le -b - 1$

- Case 2: $N + b \ge 0 \Rightarrow$ equation (4.10) holds for all $n \ge N + b \ge 0$. Let's consider equation (4.10) with $n = N + b$, which becomes

$$\sum_{i=-b}^{A} \alpha_i(N + b + i)\tilde{l}_{N+b+i}(y) = l_{N+b}(f)$$

Now, by the degree bound we know that $l_{N+b+i}(y) = 0$ when $i > -b$, thus

$$\alpha_b(N)l_N(y) = l_{N+b}(f)$$

In this case we consider two subcases:

- Case 2.1: $\deg f \geq N + b \Leftrightarrow N \leq \deg f - b$
- Case 2.2: $\deg f < N + b \Rightarrow l_{N+b}(f) = 0 \Rightarrow \alpha_b(N)l_N(y) = 0 \Rightarrow \alpha_b(N) = 0$, because $l_N(y)$ is a constant different from zero.

■

Remark 4.4.7 *Comments to Theorem 4.2.1 in comparison to Theorem 4.4.6:*

(a) *Note that - in contrast to Theorem 4.2.1 - in the above proof we did not need the exact (explicit) definition of the difference operator L (with all its coefficients). Of course, the entire information of L is implicitly given by the $\alpha_i(n)$.*

(b) *Unlike in Theorem 4.2.1 the $-b-1$ is essential and can not be discarded (see Example 4.7.5). However, we will show in section 4.6, that the degree bounds N are the same.*

Let's turn to the promised Theorem for the algorithm:

Theorem 4.4.8 *Let N and b be defined in Lemma 4.4.5 and Theorem 4.4.6 with $N \geq 0$. Then for every $y \in \mathbb{K}[[P_n]]$ the following two statements are equivalent:*

(a) *y is a polynomial which satisfies $Ly = f$*

(b) *The sequence $\langle \tilde{l}_n(y) \rangle$ satisfies (4.10) for $n \leq N + b$, and $\tilde{l}_n(y) = 0$ for $n > N$.*

Proof:

$(1) \Rightarrow (2)$: We already derived that $\langle \tilde{l}_n(y) \rangle$ satisfies (4.10) even for $n \geq 0$. Moreover $\tilde{l}_n(y) = 0$ for $n > N$, because $\deg y \leq N$.

$(2) \Rightarrow (1)$: We know that $\langle \tilde{l}_n(y) \rangle$ satisfies (4.10) for $n \leq N + b$. For $n > N + b$ the left hand side of (4.10) is zero, because for $n > N + b$ all $\tilde{l}_{n+i}(y)$ are zero. Also, $n > N + b \geq (\deg f - b) + b = \deg f$, thus the right hand side of (4.10) is zero as well. It follows that (4.10) is satisfied for all $n \geq 0$, which implies that y satisfies $Ly = f$. Because of $\tilde{l}_n(y) = 0$ for $n > N$ this y is also a polynomial.

■

Step 3: The Algorithm   Because of Theorem 4.4.8 we can immediately formulate the (rough) algorithm:

Compute polynomial solutions y of $Ly = f$ by:

1. Compute all vectors $\left(\tilde{l}_{-b}(y), \tilde{l}_{-b+1}(y), ..., \tilde{l}_{N+A+b}(y) \right)$, which satisfy (4.10) for $0 \leq n \leq N + b$

2. Select those vectors with $\tilde{l}_n(y) = 0$ for $n < 0$ and for $n > N$

3. Compute y by (4.7): $y = \sum_{n=0}^{N} \tilde{l}_n(y) P_n(x)$

In the following we will show how to do this by using recurrence (4.10) in the forward direction, taking conditions at one end as initial conditions and those at the other end as constraints on the free parameters.

Let's denote $\tilde{l}_n(y)$ by v_n for $n \in \{-b, -b+1, ..., N+A+b\}$. We have to set up a basis of dimension $A + b$ equaling the order of the recurrence. Using the condition $\tilde{l}_n(y) = 0$ for $n < 0$ we already get b initial values, thus we only have to set up a basis of dimension A:
We can set $v_n^{(j)} = 0$ for $n \in \{-b, -b+1, ..., -1\}$ and $j \in \{1, 2, ..., A\}$ and we set $v_n^{(j)} = \delta_{n+1,j}$ for $n \in \{0, 1, ..., A-1\}$ and $j \in \{1, 2, ..., A\}$. Denoting this initializations by $\mathcal{V}$ we have

$$
\mathcal{V} = \left(\begin{pmatrix} v_{-b}^{(1)} \\ \vdots \\ v_{-1}^{(1)} \\ v_0^{(1)} \\ v_1^{(1)} \\ \vdots \\ v_{A-1}^{(1)} \end{pmatrix}, ..., \begin{pmatrix} v_{-b}^{(A)} \\ \vdots \\ v_{-1}^{(A)} \\ v_0^{(A)} \\ v_1^{(A)} \\ \vdots \\ v_{A-1}^{(A)} \end{pmatrix} \right) := \left(\begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, ..., \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} \right)
$$

If the recurrence is inhomogeneous, we also need a vector for the particular solution, which we will denote by g. At the beginning we set

$$
g = (g_0, g_1, ..., g_{A-1})^T := (0, 0, ..., 0)^T.
$$

Now we use recurrence (4.10) in the forward direction and extend $\mathcal{V}$ and g by calculating all other $v_n^{(j)}$ and g_n for $n \in \{A, A+1, ..., N+A+b\}$ and $j \in \{1, 2, ..., A\}$:

$$
v_n^{(j)} = -\frac{1}{\alpha_{-A}(n)} \cdot \sum_{i=1}^{A+b} \alpha_{i-A}(n-i) v_{n-i}^{(j)} \tag{4.11}
$$

$$
g_n = \frac{1}{\alpha_{-A}(n)} \cdot \left(l_{n-A}(f) - \sum_{i=1}^{A+b} \alpha_{i-A}(n-i) v_{n-i}^{(j)} \right) \tag{4.12}
$$

Looking at these equations we should guarantee that the denominator $\alpha_{-A}(n)$ never vanishes - we will do that at the end of the derivation.
After the extensions $\mathcal{V}$ contains A vectors with length $N+A+b+1$ and g is a vector of length $N+A+b+1$.

What remains is the condition $\tilde{l}_n(y) = 0$ for $n > N$, which means that

$$v_n = \sum_{j=1}^{A} \lambda_j v_n^{(j)} = 0 \text{ for all } n \in \{N+1, N+2, ..., N+A+b\}$$

This leads to the following linear systems, for the homogeneous and inhomogenous case, respectively:

$$\begin{pmatrix} v_{N+1}^{(1)} & \cdots & v_{N+1}^{(A)} \\ \vdots & \ddots & \vdots \\ v_{N+A+b}^{(1)} & \cdots & v_{N+A+b}^{(A)} \end{pmatrix} \cdot \begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_A \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \tag{4.13}$$

$$\begin{pmatrix} v_{N+1}^{(1)} & \cdots & v_{N+1}^{(A)} \\ \vdots & \ddots & \vdots \\ v_{N+A+b}^{(1)} & \cdots & v_{N+A+b}^{(A)} \end{pmatrix} \cdot \begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_A \end{pmatrix} = - \begin{pmatrix} g_{N+1} \\ \vdots \\ g_{N+A+b} \end{pmatrix} \tag{4.14}$$

Now the general polynomial solution of $Ly = f$ can be computed as follows:

- Homogeneous case $Ly = 0$: Let $\lambda_1, ..., \lambda_A$ be the general solution of (4.13) then

$$y = \sum_{n=0}^{N} \sum_{j=0}^{A} \lambda_j v_n^{(j)} P_n = \sum_{j=0}^{A} \lambda_j \sum_{n=0}^{N} v_n^{(j)} P_n$$

- Inhomogeneous case $Ly = f$: Let $\lambda_1, ..., \lambda_A$ be the general solution of (4.14) then

$$y = \sum_{n=0}^{N} g_n P_n + \sum_{j=0}^{A} \lambda_j \sum_{n=0}^{N} v_n^{(j)} P_n$$

As mentioned we want to show that during the calculation no vanishing denominator appears, thus, it suffices to show that $\alpha_{-A}(n) \neq 0$ for $n \geq A$. Since

$$\begin{aligned} \alpha_{-A}(n+A) &= \sum_{k=0}^{r} \sum_{j=0}^{d} \binom{(n+A)-A}{j} \binom{j}{k-A} l_j(q_k) = \\ &= \sum_{k=0}^{r} \sum_{j=0}^{d} \binom{n}{j} \binom{j}{k-r} l_j(q_k) = \sum_{j=0}^{d} \binom{n}{j} l_j(q_r) = \\ &= \sum_{j=0}^{d} l_j(q_r) \binom{(n+a)-a}{j} = q_r(n+a) \neq 0 \end{aligned}$$

by the following choice of a (which we have not determined up to now) in the power basis $\binom{x-a}{n}$, no problems come up: Take $a = 0$ if $q_r(x)$ has no nonnegative

integer zero, else take $a = \max\{x \in \mathbb{N} : q_r(x) = 0\} + 1$.

Let's sum up the complete algorithm in

Algorithm 4.4.9 (PolyABP) *by Sergei Abramov, Manuel Bronstein and*
Marko Petkovšek
INPUT: Ly = f with

- $L = \sum_{i=0}^{r} q_i \cdot \Delta^i$ *where* $q_i \in \mathbb{K}[x]$

- $q_0, q_r \neq 0$

- $f \in \mathbb{K}[x]$

OUTPUT: The general polynomial solution over $\mathbb{K}$ *of* $Ly = f$

BEGIN

  IF $q_r(x)$ has no nonnegative integer zero

    THEN $a := 0$
    ELSE $a := \max\{x \in \mathbb{N} : q_r(x) = 0\} + 1$

  $d := \max\{\deg q_k \mid 0 \le k \le r\}$

  $b := \max\{\deg q_k - k \mid 0 \le k \le r\}$

  FOR $i \in \{-d, -d+1, ..., r\}$ DO compute $\alpha_i(n)$ by (4.9)

  $N := \max(\{z \in \mathbb{N} : \alpha_b(z) = 0\} \cup \{deg f - b, -b - 1\})$

  IF $f = 0$ THEN

    Initialize $\mathcal{V}$ and extend it by (4.11)

    Set up the linear system of equations (4.13) and compute the general
    solution $\lambda_1, ..., \lambda_r$

    $y = \sum_{j=0}^{r} \lambda_j \sum_{n=0}^{N} v_n^{(j)} \binom{x-a}{n}$

  ELSE

    Initialize $\mathcal{V}$ and g and extend both by (4.11) and (4.12), respectively

    Set up the linear system of equations (4.14) and compute the general
    solution $\lambda_1, ..., \lambda_r$

    $y = \sum_{n=0}^{N} g_n \binom{x-a}{n} + \sum_{j=0}^{r} \lambda_j \sum_{n=0}^{N} v_n^{(j)} \binom{x-a}{n}$

  Return y

END

$\blacklozenge$

Remark 4.4.10 *Remarks on Algorithm 4.4.9*

- *Unlike to PolyDelta and PolyTau the linear system to solve is not in tri-angular form.*

- *One problem of PolyDelta and PolyTau also appears in PolyABP: we have no guarantee that the linear system has a solution, though a degree bound $N \geq 0$ has been found.*

4.5   Solving by Interpolation Techniques

4.5.1   Using Lagrange Interpolation

In [Bar99] Moulay Barkatou presented a method how to reduce the number of variables in the final linear system using Lagrange interpolation - however, the method is only described for matrix difference equations[2] of order 1: He was able to reduce the number of unknown from $st(N+1)$ to st (where s and t are the dimensions of the matrices and N is the degree bound). Basing ourselves on Barkatou's interpolation idea we were able to develop an analogous algorithm for scalar difference equations of order n. This algorithm needs to solve a linear system of equations with n instead of $N+1$ unknowns which is just the same as in PolyABP (Algorithm 4.4.9).

The main idea comes from Langrange's interpolation formula used on the points $\alpha, \alpha+1, ..., \alpha+N$, where N denotes the (already known) degree bound and α is an (at the beginning) arbitrary element from $\mathbb{K}$, thus we can represent a polynomial solution $y(x)$ (with degree N) in the form

$$y(x) = y(\alpha)L_0(x) + y(\alpha+1)L_1(x) + ... + y(\alpha+N)L_N(x) \qquad (4.15)$$

where

$$L_i(x) = \prod_{j=0, j \neq i}^{N} \frac{x - \alpha - j}{i - j} \quad i \in \{0, 1, ..., N\}$$

are the Lagrange polynomials for the points $\alpha, \alpha+1, ..., \alpha+N$.
Hence, in order to find $y(x)$ it suffices to compute $y(\alpha), y(\alpha+1), ..., y(\alpha+N)$. The key step is now to set up $y(\alpha), ..., y(\alpha+n-1)$ as n unknown coefficients $c_0, ..., c_{n-1}$ and to compute the remaining $y(\alpha+n), ..., y(\alpha+N)$ via the difference equation (used in forward direction). At this point it must be guaranteed that this is always possible meaning that no pole in the leading coefficient appear, but this can achieved by the choice of α (see also the choice of a at the end of the previous section): The leading coefficient may not vanish at the points $\alpha, \alpha+1, ..., \alpha+N-1$.
In this way all $y(\alpha+i)$ become a linear combination of $c_0, ..., c_{n-1}$ and equation (4.15) becomes a polynomial with n instead of $N+1$ unknowns.

---

[2]see for example (5.3) on page 63

The resulting algorithm is straightforward:

Algorithm 4.5.1 (Lagrange interpolation) *by Christian Weixlbaumer resp. Moulay Barkatou*
INPUT: $Ly = f$ with

- $L = \sum_{i=0}^{n} p_k \cdot \tau^k$ *where $p_k \in \mathbb{K}[x]$*

- $p_0, p_n \neq 0$

- $f \in \mathbb{K}[x]$

- *The degree bound N for the polynomial solution*

OUTPUT: The general polynomial solution over $\mathbb{K}$ of $Ly = f$

BEGIN

Choose α such that $p_n(x)$ is not equal to zero for $x = \alpha, \alpha+1, ..., \alpha+N-1$

FOR $i \in \{0, 1, ..., n-1\}$ DO $y(\alpha + i) := c_i$

FOR $i \in \{n, n+1, ..., N\}$ DO compute $y(\alpha + i)$ using the recurrence relation $Ly = f$ (in the forward direction)

FOR $i \in \{0, 1, ..., N\}$ DO $L_i(x) := \prod_{j=0, j\neq i}^{N} \frac{x-\alpha-j}{i-j}$

Plug $y(x) := y(\alpha)L_0(x) + y(\alpha+1)L_1(x) + ... + y(\alpha+N)L_N(x)$ into $Ly = f$ and find the general solution $(c_0, ..., c_{n-1})$ of the resulting linear system

Return $y(x)$

END

♦

Remark 4.5.2 *Remarks on Algorithm 4.5.1*

- *Just like PolyABP Algorithm 4.5.1 yields a linear system with $n = order(L)$ unknowns which is in general not in triangular form. However, the linear system of Algorithm 4.5.1 consists of $N + degree(L)$ equations, whereas PolyABP has to deal with $n + degree(L)$.*

- *The Lagrange-method can be easily used as an alternative to the standard method of undetermined coefficients in PolyDelta and PolyTau if N turns out to be (much) greater than $n = order(L)$. Testing shows that plugging the $y(x)$ into $Ly = f$ is the crucial part of the algorithm, because the necessary computation with the Lagrange polynomials becomes slow with increasing degree (which is N, unfortunately)!*

4.5.2 Using Newton Interpolation

As mentioned one problem of Lagrange polynomials is their degree which is N for *each* polynomial. This motivates the use of Newton's interpolation formula whose $N+1$ polynomials have the degrees $0, 1, ..., N$:

$$y(x) = \sum_{k=0}^{N} \binom{x - \alpha}{k} \Delta^k y(\alpha)$$

Note that $\Delta^k y(\alpha)$ means $\Delta^k y(x)|_\alpha$.

Just like in the previous subsection we only need to know $y(\alpha), \Delta y(\alpha), ..., \Delta^N y(\alpha)$ to get - after solving a system of linear equations - the solution $y(x)$. Again, we set up $y(\alpha), \Delta y(\alpha), ..., \Delta^{n-1} y(\alpha)$ as n unknowns and want to compute the remaining $\Delta^k y(\alpha)$ via the difference equation. This time, however, this is not that straightforward!

We will present three possibilities:

Possibility 1 - Already Done

The first possibility was already described in Section 4.4: With the help of equation (4.6) the difference equation could be rewritten into (4.10) containing $\tilde{l}_k(y)$ which were nothing else than our $\Delta^k y(\alpha)$.

Possibility 2 - Efficiency Problem

Suppose we have the following difference equation of order n with rational functions coefficients:

$$\Delta^n(y) = a_{n-1} \Delta^{n-1}(y) + ... + a_1 \Delta(y) + a_0 y \tag{4.16}$$

Hence, we can immediately compute $\Delta^n y(\alpha)$ by simply plugging α into (4.16). In order to get an analogous equation for $\Delta^{n+1} y(\alpha)$ we apply the Δ-Operator on (4.16), yielding

$$
\begin{aligned}
\Delta^{n+1}(y) &= \Delta\left(a_{n-1} \Delta^{n-1}(y) + ... + a_1 \Delta(y) + a_0 y\right) = \\
&= \Delta\left(a_{n-1} \Delta^{n-1}(y)\right) + ... + \Delta\left(a_1 \Delta(y)\right) + \Delta\left(a_0 y\right)
\end{aligned}
$$

Remember Proposition 2.2.4 (g) where we had the product rule for the Δ-Operator: $\Delta(fg) = \Delta(f)g + f\Delta(g) + \Delta(f)\Delta(g) = \Delta(f)g + \tau(g)\Delta(g)$. Thus, we can simplify the above equation to

$$
\begin{aligned}
\Delta^{n+1}(y) &= \Delta(a_{n-1})\Delta^{n-1}(y) + ... + \Delta(a_1)\Delta(y) + \Delta(a_0)y + \\
&+ \tau(a_{n-1})\Delta^n(y) + ... + \tau(a_1)\Delta^2(y) + \tau(a_0)\Delta(y)
\end{aligned}
$$

In other words:

$$\Delta^{n+1}(y) = b_n \Delta^n(y) + ... + b_1 \Delta(y) + b_0 y \tag{4.17}$$

where $b_n = \tau(a_{n-1})$, $b_0 = \Delta(a_0)$ and all other coefficients are given by $b_k = \Delta(a_k) + \tau(a_{k-1})$. With (4.17) we are now able to compute $\Delta^{n+1}y(\alpha)$; in an analogous way all other $\Delta^k y(\alpha)$ can be computed.

The drawback of this method may be the costly computation of the "new" recurrences, as we have to compute some $\Delta(a_k)$ with rational functions a_k.

Possibility 3 - Optimal

The third possibility for computing all other $\Delta^k y(\alpha)$ uses the following scheme coming from the theory about Newton interpolation. Let's write down the scheme for $N = 4$ and $\alpha = 0$:

$$
\begin{array}{ccccccccc}
y(0) \\
& > & y(1) \\
\Delta y(0) & & & > & y(2) \\
& > & \Delta y(1) & & & > & y(3) \\
\Delta^2 y(0) & & & > & \Delta y(2) & & & > & y(4) \\
& > & \Delta^2 y(1) & & & > & \Delta y(3) \\
\Delta^3 y(0) & & & > & \Delta^2 y(2) \\
& > & \Delta^3 y(1) \\
\Delta^4 y(0)
\end{array}
$$

The construction of this scheme is similar to the construction of the Pascal triangle by using the (general) relation

$$\Delta^k y(\alpha) + \Delta^{k+1} y(\alpha) = \Delta^k y(\alpha + 1) \tag{4.18}$$

This scheme can now be used for the computation of the $\Delta^k y(\alpha)$ which are always contained in the first column of the scheme[3]. We will demonstrate the idea for $N = 4$ and $\alpha = 0$ having a difference equation of order 2: Thus, we have at the beginning $y(0) = c_0$ and $\Delta y(0) = c_1$. By using the difference equation in the form (4.16) we can compute $\Delta^2 y(0)$. Now we can use the scheme to compute $y(1) = y(0) + \Delta y(0)$ and $\Delta y(1) = \Delta y(0) + \Delta^2 y(0)$. With these two values we can again use the difference equation in order to compute $\Delta^2 y(1)$. Using the scheme we get $\Delta^3 y(0) = \Delta^2 y(1) - \Delta^2 y(0)$, and so on ...

The choice of α has to be made as "usual": The denominators of equation (4.16) may not vanish at the points $\alpha, \alpha + 1, ...$

In the following algorithm the choice of α will be the same as the choice of a in PolyABP.

Let's sum up our method in

Algorithm 4.5.3 (Newton interpolation) *by Christian Weixlbaumer*
INPUT: $Ly = f$ with

---

[3]The values needed for Lagrange interpolation $y(0), y(1), ...$ are also contained in the scheme.

- $L = \sum_{i=0}^{n} q_k \cdot \Delta^k$ *where* $q_k \in \mathbb{K}[x]$

- $q_0, q_n \neq 0$

- $f \in \mathbb{K}[x]$

- *The degree bound N for the polynomial solution*

OUTPUT: *The general polynomial solution over $\mathbb{K}$ of $Ly = f$*

BEGIN

IF $q_n(x)$ has no nonnegative integer zero

THEN $\alpha := 0$

ELSE $\alpha := \max\{x \in \mathbb{N} : q_n(x) = 0\} + 1$

FOR $i \in \{0, 1, ..., n - 1\}$ DO $\Delta^i y(\alpha) := c_i$

Compute $\Delta^n y(\alpha)$ using the recurrence relation $Ly = f$ (in the forward direction)

FOR $j \in \{1, 2, ..., N - n\}$ DO

FOR $i \in \{0, 1, ..., n - 1\}$ DO

$\Delta^i y(\alpha + j) = \Delta^i y(\alpha + j - 1) + \Delta^{i+1} y(\alpha + j - 1)$

Compute $\Delta^n y(\alpha + j)$ using the recurrence relation $Ly = f$ (in the forward direction)

FOR $i \in \{1, 2, ..., j\}$ DO

$\Delta^{n+i} y(\alpha + j - i) = \Delta^{n+i-1} y(\alpha + j - i + 1) - \Delta^{n+i-1} y(\alpha + j - i)$

Plug $y(x) := y(\alpha) \binom{x-\alpha}{0} + \Delta y(\alpha) \binom{x-\alpha}{1} + ... + \Delta^N y(\alpha) \binom{x-\alpha}{N}$ into $Ly = f$ and find the general solution $(c_0, ..., c_{n-1})$ of the resulting linear system

Return $y(x)$

END

$\blacklozenge$

Remark 4.5.4 *Remarks on Algorithm 4.5.3*

- *Just like Algorithm 4.5.1 Algorithm 4.5.3 yields a linear system with $n = order(L)$ unknowns and $N + degree(L)$ equations which is in general not in triangular form.*

- *Algorithm 4.5.3 is another alternative to the standard method of undetermined coefficients used in PolyDelta and PolyTau as long as N is (much) greater than $n = order(L)$.*

4.6   Comparing the Bounds

In this section we will compare the degree bounds computed by PolyDelta, PolyTau and PolyABP. Clearly, this includes the comparison of all degree polynomials $\alpha(z)$, $D(z)$ and $\alpha_b(z)$ which has beem done in [PeWe00] for the first time.

Let's shortly recall these bounds and polynomials: We had

$$L = \sum_{k=0}^{n} p_k(x)\tau^k = \sum_{k=0}^{n} q_k(x)\Delta^k$$

$$p_k(x) = \sum_{i=k}^{n}(-1)^{i-k}\binom{i}{k}q_i(x) \quad resp. \quad q_k(x) = \sum_{i=k}^{n}\binom{i}{k}p_i(x).$$

$$b = \max_{0 \le k \le n}(\deg q_k(x) - k)$$

$$d = \max_{0 \le k \le n}\deg p_k(x) = \max_{0 \le k \le n}\deg q_k(x)$$

$$p_k(x) = \sum_{j=0}^{d} p_{k,j}x^j$$

$$q_k(x) = \sum_{j=0}^{d} q_{k,j}x^j$$

(a) PolyDelta: We proved that

$$\deg y \le \max\{M, \deg f - b, -b - 1\}$$

where M is the maximal integer root of the degree polynomial

$$\alpha(z) = \sum_{j=0}^{n} q_{j,b+j}\, z^{\underline{j}}.$$

(b) PolyTau: For $0 \le j \le s$ define

$$b_j^{(s)} = \sum_{i=0}^{n} i^j p_{i,d-s+j}, \quad D_s(z) = \sum_{j=0}^{s}\binom{z}{j}b_j^{(s)},$$

and let
$$s_0 = \min\{s \ge 0: \ D_s(z) \not\equiv 0\}.$$

Then
$$\deg y \le \max\{M, \deg f + s_0 - d\}$$

where M is the maximal integer root of the degree polynomial

$$D(z) = D_{s_0}(z) = \sum_{j=0}^{s_0}\binom{z}{j}b_j^{(s_0)}.$$

(c) PolyABP: We showed that

$$\deg y \leq \max\{M, \deg f - b, -b - 1\}$$

where M is the maximal integer root of the degree polynomial

$$\alpha_b(z) = \sum_{k=0}^{r} \sum_{j=b+k}^{d} \binom{z+b}{j} \binom{j}{b+k} \Delta^j q_k(0).$$

We will make use of the following identity (which is also very useful for some computations in PolyABP):

$$\Delta^n p(x) = \sum_{k=0}^{n} (-1)^{n-k} \binom{n}{k} p(x+k) \quad \text{for } n \in \mathbb{N}$$

where $p(x)$ is an arbitrary polynomial.

Theorem 4.6.1 *Using the notations above, we have:*

(a) $s_0 + b = d$

(b) $\alpha(z) = D(z)$

Proof: Using (4.1) we read off that

$$p_{k,j} = \sum_{i=k}^{n} (-1)^{i-k} \binom{i}{k} q_{i,j}(x)$$

Therefore

$$
\begin{aligned}
b_j^{(s)} &= \sum_{k=0}^{n} k^j p_{k,d-s+j} = \sum_{k=0}^{n} \sum_{i=k}^{n} (-1)^{i-k} \binom{i}{k} q_{i,d-s+j} k^j = \\
&= \sum_{i=0}^{n} q_{i,d-s+j} \sum_{k=0}^{i} (-1)^{i-k} \binom{i}{k} k^j = \sum_{i=0}^{n} q_{i,d-s+j} \Delta^i x^j \big|_{x=0} = \\
&= \sum_{i=0}^{j} q_{i,d-s+j} \Delta^i x^j \big|_{x=0}.
\end{aligned}
$$

Suppose that $s < d - b$, then $d - s + j > b + j \geq b + i \geq \deg q_i$ by definition of b, hence $c_{i,d-s+j} = 0$ for $0 \leq i \leq j$. So $b_j^{(s)} = 0$ for $0 \leq j \leq s$ and $D_s(z) \equiv 0$. It follows that $s_0 \geq d - b$.
On the other hand,

$$b_j^{(d-b)} = \sum_{i=0}^{j} q_{i,b+j} \Delta^i x^j \big|_{x=0}.$$

If $i < j$ then $b + j > b + i \geq \deg q_i$ by definition of b, hence $q_{i,b+j} = 0$ for $0 \leq i < j$ and so

$$b_j^{(d-b)} = q_{j,b+j}\Delta^j x^j|_{x=0} = j!\, q_{j,b+j}.$$

Therefore

$$D_{d-b}(z) = \sum_{j=0}^{d-b}\binom{z}{j}b_j^{(d-b)} = \sum_{j=0}^{d-b}\binom{z}{j}j!\, q_{j,b+j} = \sum_{j=0}^{d-b} q_{j,b+j}z^{\underline{j}} = \alpha(z) \not\equiv 0.$$

It follows that $s_0 = d - b$ and $D(z) = \alpha(z)$ as claimed.

$\blacksquare$

Remark 4.6.2 *Theorem 4.6.1 simply says that Abramov's algorithm PolyDelta and Petkovšek's PolyTau are equivalent up to the additional $-b-1$ in PolyDelta.*

Theorem 4.6.3 *Using the notations above, we have:*

$$\alpha_b(z) = \begin{cases} (z+b)^{\underline{b}} \cdot \alpha(z) & b \geq 0 \\ \frac{1}{z^{\underline{-b}}} \cdot \alpha(z) & b < 0 \end{cases}$$

Proof: If $j > b + k$ then, as $b \geq \deg q_k - k$, we have $j > \deg q_k$ and consequently $\Delta^j q_k(0) = 0$. It follows that

$$
\begin{aligned}
\alpha_b(z) &= \sum_{k=0}^{n}\sum_{j=b+k}^{d}\binom{z+b}{j}\binom{j}{b+k}\Delta^j q_k(0) = \sum_{k=0}^{n}\binom{z+b}{b+k}\Delta^{b+k}q_k(0) = \\
&= \sum_{k=0}^{n}\binom{z+b}{b+k}(b+k)!\, q_{k,b+k} = \sum_{k=0}^{n} q_{k,b+k}(z+b)^{\underline{b+k}}
\end{aligned}
$$

Let's consider both cases:

- If $b \geq 0$ then $(z+b)^{\underline{b+k}} = (z+b)^{\underline{b}}z^{\underline{k}}$ and $\alpha_b(z) = (z+b)^{\underline{b}} \cdot \alpha(z)$

- If $b < 0$ then $(z+b)^{\underline{b+k}} = \frac{1}{z^{\underline{-b}}}z^{\underline{k}}$ and $\alpha_b(z) = \frac{1}{z^{\underline{-b}}} \cdot \alpha(z)$

$\blacksquare$

Corollary 4.6.4 *The degree polynomials $\alpha(z)$, $D(z)$ and $\alpha_b(z)$ coincide if and only if $b = 0$. PolyTau and PolyDelta are equivalent. The degree bounds computed by PolyTau, PolyDelta and PolyABP are the same.*

Proof: Comparing PolyDelta and PolyTau with respect Theorem 4.6.1 to we have to show that the $-b-1$ is not necessary at all and can be discarded. This is clear when $b \geq 0$. When $b < 0$, then Theorem 4.6.3 shows that $\alpha(z)$ is divisible by $z^{\underline{-b}}$ with roots $0, 1, ..., -b-1$, implying that $M \geq -b-1$.
To see that PolyDelta/PolyTau and PolyABP compute the same degree bound, observe that for $b > 0$ the maximum integer root of $(z+b)^{\underline{b}}$ is -1 and for $b < 0$ the maximum integer root of $z^{\underline{-b}}$ is $-b-1$.

■

Corollary 4.6.5 *To have a degree polynomial $P(z)$ of least degree we can take*

$$P(z) = \begin{cases} \alpha(z) & b \geq 0, \\ \frac{1}{z^{-b}} \cdot \alpha(z) & b < 0. \end{cases}$$

Then we have the bound

$$\deg y \leq \begin{cases} \max\{M, \deg f - b\} & b \geq 0 \\ \max\{M, \deg f - b, -b - 1\} & b < 0 \end{cases}$$

where M is the maximal integer root of $P(z)$.

Remark 4.6.6 *Suppose we have the difference equation $Ly = 0$ and computed $b < 0$, then it follows that $q_0 = q_1 = ... = q_{-b-1} = 0$, thus we can write*

$$L = \sum_{k=-b}^{n} q_k \Delta^k$$

This implies that

$$y(x) = c_0 + c_1 x + ... + c_{-b-1} x^{-b-1}$$

is a solution (of dimension $-b$) of $Ly = 0$.

Let's conclude with a comparison of PolyDelta/PolyTau to PolyABP:

- General differences

	PolyDelta	PolyTau	PolyABP
L given in terms of	Δ	τ	Δ
Used power base	x^n	x^n	$\binom{x-a}{n}$

- PolyDelta/PolyTau using the method of undetermined coefficients

	PolyDelta/PolyTau	PolyABP
Preliminary computations	ε	$\alpha_i, \Delta^j(f)$
Unknowns in linear system	$N + 1$	$order(L)$
Equations in linear system	$N + d$	$order(L) + d$
Type of linear system	triangular	arbitrary

- PolyDelta/PolyTau using an interpolation method (Lagrange or Newton)

	PolyDelta/PolyTau	PolyABP
Preliminary computations	expanding polynomials	$\alpha_i, \Delta^j(f)$
Unknowns in linear system	$order(L)$	$order(L)$
Equations in linear system	$N + d$	$order(L) + d$
Type of linear system	arbitrary	arbitrary

Thus, we can state the following: As soon as N is (much) greater than $order(L)$, the method of undetermined coefficients should be avoided. However, a careful implementation concerning the preliminary computations in PolyABP respectively of the interpolation methods is required (and most important).

4.7   Examples and Comparison

Example 4.7.1 *Given $Ly = 0$, with $L = (4x + 33)\tau^2 + 4(x - 16)\tau - (8x - 7)$, thus we have*

$$(4x + 33)y(x + 2) + 4(x - 16)y(x + 1) - (8x - 7)y(x) = 0$$

with the polynomial solution

$$y(x) = c \cdot (x^2 + 2x + 3)$$

Each of the algorithms compute $b = 0$, the degree polynomial $12(z - 2)$ and the (sharp) degree bound 2.

Example 4.7.2 *Given $Ly = f$, with $L = (x + 1)\tau^2 - x^2\tau + x^2$ and $f = x - 1$, thus we have*

$$(x + 1)y(x + 2) - x^2 y(x + 1) + x^2 y(x) = x - 1$$

with the polynomial solution

$$y(x) = 2x - 5$$

Each of the algorithms compute the degree polynomial $z - 1$ and the (sharp) degree bound $N = 1$. Note that the corresponding homogeneous equation has no solution, though the degree bound N is still 1.

Example 4.7.3 *Given $Ly = f$, with $L = (x + 1)\tau^2 - 2\tau - x$ and $f = ax^3 + bx^2 + cx + d$, where a, b, c, d are indeterminates, thus we have*

$$(x + 1)y(x + 2) - 2y(x + 1) - xy(x) = ax^3 + bx^2 + cx + d$$

with the polynomial solution

$$y(x) = \frac{a}{5}x^3 + \frac{-12a + 5b}{15}x^2 + \frac{6a - 20b + 15c}{15}x + \frac{-6a + 10b - 15d}{15}$$

Each of the algorithms compute the degree polynomial $2z - 1$, which has no integer root. The (sharp) degree bound is obtained by $N = \deg f - b = \deg f$.

The next example shows that it is impossible to bound the degree of polynomial solutions by the order and degree of the difference equations:

Example 4.7.4 *Given $Ly = 0$, with $L = x\tau - (x + k)$ and $k \in \mathbb{N}$, thus we have*

$$xy(x + 1) - (x + k)y(x) = 0$$

with the solution

$$y(x) = c \cdot x(x + 1)...(x + k - 1) = c \cdot x^{\overline{k}}$$

Each of the algorithms compute the degree polynomial $z - k$ and the degree bound k. This is in spite of having $order(L) = 1$ and $degree(L) = 1$. Note that using the method of undetermined coefficients we have to solve a system with k equations and $k + 1$ unknowns and using PolyABP the remaining system has dimension 1.

In the next examples we illustrate the possible differences between the degree polynomials $\alpha(z) = D(z)$ and $\alpha_b(z)$ using difference operators of the form $L = x^k \Delta^n + \Delta^m$:

Example 4.7.5 *Given $Ly = f$, with $L = \Delta^n$ with a positive integer n and an arbitrary polynomial f, thus we have e.g. for $n = 3$*

$$y(x + 3) - 3y(x + 2) + 3y(x + 1) - y(x) = f$$

which has the solution

$$y(x) = c_0 + c_1 x + ... + c_{n-1} x^{n-1} + y_p(x)$$

where $y_p(x)$ denotes a partial (polynomial) solution of $Ly = f$ (which exists for every nonzero f) and which has degree $\deg f + n$.

We observe the following (note that $b = -n$):

- $D(z) = z(z - 1)(z - 2)...(z - n + 1)$, which has the maximum integer root $n - 1$.

- $\alpha_b(z) = 1$, which has no roots.

In the inhomogeneous case, the degree bound N is always equal to $\deg f + n$, which is sharp. In the homogeneous case, we have $N = n - 1$, which is also sharp. Note that on the one hand this is equal to the maximum integer root of $\alpha(z) = D(z)$ and on the other hand $n - 1 = -b - 1$.

Example 4.7.6 *Given $Ly = f$, with $L = x^k \Delta^n + 1$ with positive integers k and n and an arbitrary polynomial f, thus we have e.g. for $n = 3$*

$$x^k y(x + 3) - 3x^k y(x + 2) + 3x^k y(x + 1) - (x^k + 1)y(x) = f$$

which has exactly the polynomial solution $y = f$ if $\deg f < n$ (resp. no solution for $f = 0$). If $\deg f \geq n$, then there exists an (in advance unknown) solution if $k \leq n$ and there may be a solution if $k > n$.

We observe the following - note that $b = \max\{k - n, 0\}$:

- If $k > n$, then

$$\begin{aligned} \alpha(z) = D(z) &= (z - n + 1) \cdot \ldots \cdot (z - 1) \cdot z \\ \alpha_b(z) &= (z - n + 1) \cdot \ldots \cdot (z - 1) \cdot z \cdot (z + 1) \cdot \ldots \cdot (z + k - n) \end{aligned}$$

  which have the maximum integer root $n - 1$.

- If $k = n$, then all degree polynomials are equal and of degree n.

- If $k < n$, then all degree polynomials are 1.

Furthermore, the computed degree bounds N are of interest:

- If $\deg f < n$ and $k \leq n$, then $N = \deg f$. In this case the bound N is sharp, because we have the solution $y = f$.

- If $\deg f < n$ and $k > n$, then $N = n - 1$. In this case N is really a bound, as soon as $\deg f \neq n - 1$.

- If $\deg f > n$ and $k \leq n$, then $N = \deg f$. In this case the bound N is also sharp, that means the difference equation has a solution of degree $\deg f$.

- If $\deg f > n$ and $k > n$, then $N = \max\{n - 1, \deg f - n + k\}$. In this case, the bound N is sharp if there exists a solution. However, for an "arbitrary" f we usually have no solution. This is more or less the problem case: It may be that we have to solve a huge system of equations which has at the end no (or only the trivial) solution.

Example 4.7.7 *Given $Ly = f$, with $L = x^n \Delta^n + \Delta^m$ with positive integers n and m and an arbitrary polynomial f, thus we have e.g. for $n = 2$ and $m = 3$:*

$$y(x + 3) + (x^2 - 3)y(x + 2) - (2x^2 - 3)y(x + 1) + (x^2 - 1)y(x) = f$$

We observe the following - note that $b = 0$:

$$\alpha(z) = D(z) = \alpha_b(z) = z(z - 1)(z - 2)\ldots(z - n + 1)$$

which is totally independent from m and that $N = \max\{n - 1, \deg f\}$. Moreover in the homogeneous case we have

- If $m < n$, then $N = n - 1$, however the general polynomial solution is $y(x) = c_0 + c_1 x + \ldots + c_{m-1} x^{m-1}$ which is of lower degree.

- If $m \geq n$, then $N = n - 1$, which is sharp because the general polynomial solution is given by $y(x) = c_0 + c_1 x + \ldots + c_{n-1} x^{n-1}$.

Looking at the inhomogeneous case we have

- If $m > \deg f \geq n$, then a solution $y(x)$ exists which is of the form $y(x) = c_0 + c_1 x + \ldots + c_{n-1} x^{n-1} + y_p(x)$, where $\deg y_p(x) = \deg f$. Thus, in this case N is sharp.

- If $n > m + \deg f$, then a solution $y(x)$ exists which is of the form $y(x) = c_0 + c_1 x + \ldots + c_{m-1} x^{m-1} + y_p(x)$, where $\deg y_p(x) = m + \deg f$. Thus, in this case N is sharp if and only if $n - 1 = m + \deg f$.

- For all other possibilities we have in general (i.e. for "arbitrary" f) no solution, in spite of having $N = \max\{n - 1, \deg f\}$.

Chapter 5

Rational Solutions

In this chapter we will present algorithms which compute all rational solutions of a difference equation. The common idea of all these algorithm can be roughly formulated by:

Given the difference equation $Ly = f$, one can compute all rational solutions y by:

1. Find a polynomial (or rational function) $U(x)$ such such that each rational solution $y(x)$ can be written in a form $y(x) = \frac{N(x)}{U(x)}$ where $N(x)$ is a polynomial.

2. Substitute $y(x) := \frac{z(x)}{U(x)}$ in $Ly = f$

3. Use an algorithm for finding all polynomial solutions $z(x)$ of the resulting system which yields all rational solutions $\frac{z(x)}{U(x)}$.

Obviously, the first step is the crucial one!

5.1 The Universal Denominator By Abramov

Problem 5.1.1 *Given the difference equation $Ly = f$, find a polynomial $u(x)$ (universal denominator) such that $u(x)$ is divisible by the denominator of any (reduced) rational solution of $Ly = f$, where:*

- $L = \sum_{k=0}^{n} p_k(x) \cdot \tau^k$

- $p_k \in \mathbb{K}[x]$ *(if $p_k \in \mathbb{K}(x)$, then we will multiply the entire equation with the denominator of p_k)*

- $p_n \neq 0$, $p_0 \neq 0$

- $f \in \mathbb{K}[x]$ *(if $f \in \mathbb{K}(x)$, then we will multiply the entire equation with the denominator of f)*

Sergei Abramov's *universal denominator* can be computed/implemented very simply, the theoretical background, however, is more challenging:

Algorithm 5.1.2 (Universal Denominator) *by Sergei Abramov*
INPUT: $Ly = f$ with

- *$L = \sum_{k=0}^{n} p_k \cdot \tau^k$ where $p_k \in \mathbb{K}[x]$*

- *$p_0, p_n \neq 0$*

- *$f \in \mathbb{K}[x]$*

OUTPUT: A polynomial $u(x)$ such that every rational solution $y(x)$ of $Ly = f$ can be written in the form $y(x) = \frac{z(x)}{u(x)}$, where $z(x)$ is a polynomial.

BEGIN

$$A(x) := p_n(x - n)$$

$$B(x) := p_0(x)$$

$$u(x) := 1$$

Compute the largest nonnegative integer N such that $A(x)$ and $B(x + N)$ have a nontrivial common divisor. If no such N exists, set $N := -1$

FOR i:=N DOWN TO 0 DO

$$d(x) := \gcd(A(x), B(x + i))$$
$$A(x) := \frac{A(x)}{d(x)}$$
$$B(x) := \frac{B(x)}{d(x-i)}$$
$$u(x) := u(x) \cdot d(x) \cdot \ldots \cdot d(x - i)$$

END

$\blacklozenge$

Remark 5.1.3 *Remarks on Algorithm 5.1.2*

- *For example, N can be computed as the largest nonnegative integer root of $Res_x(A(x), B(x+m))$. A more effective method is presented in [MaWr94].*

- *Note that the loop really goes from N down to 0. Though in most examples also the loop "FOR i:=0 TO N DO" can be used (which often yields universal denominators of lower degree!), there exists examples for which this loop calculates a wrong result. See Example 5.3.5.*

- *An observation from practice: If there exists several integers N_k such that $A(x)$ and $B(x + N_k)$ have a nontrivial common divisor (resp. if $Res_x(A(x), B(x + m))$ has several nonnegative roots), then the algorithm tends to compute - of course - correct but rather rough universal denominators.*

- *A MATHEMATICA-version of Algorithm 5.1.2 (resp. of Algorithm 5.1.11) over $\mathbb{Q}$ and $\mathbb{C}$ is available at the RISC-Homepage.*

In the following we will give a proof of the correctness of the algorithm dealing with the difference equation

$$p_n(x)y(x+n) + ... + p_0(x)y(x) = f(x) \tag{5.1}$$

where $p_i, f \in \mathbb{K}[x]$ and $p_0, p_n \neq 0$.

The proof given is (completely) different to Abramov's proof in [Abr95a] (which is in fact very technical) and uses ideas from [Bar99] and [Khm99] together with Paule's gff-concept (greatest factorial factorization). We will first prove some rather simple but fundamental lemmas which helps to prove a theorem giving a (first) explicit formula for the (universal) denominator of the rational solution of (5.1). Afterwards we show that this denominator is computed by Abramov's algorithm.

Lemma 5.1.4 *Let $y(x)$ be a rational solution of (5.1) and let x_0 be a pole of order μ of $y(x)$, then:*

(a) *If $x_0 + 1, ..., x_0 + n$ are no poles of $y(x)$, then $(x - x_0)^\mu$ divides $p_0(x)$*

(b) *If $x_0 - 1, ..., x_0 - n$ are no poles of $y(x)$, then $(x - x_0)^\mu$ divides $p_n(x - n)$*

(c) *If $x_0 - 1, ..., x_0 - n$ and $x_0 + 1, ..., x_0 + n$ are no poles of $y(x)$, then $(x - x_0)^\mu$ divides $\gcd(p_0(x), p_n(x - n))$*

Proof: In order to prove (a) look at (5.1) in the form

$$p_0(x)y(x) = f(x) - \big(p_1(x)y(x+1) + ... + p_n(x)y(x+n)\big)$$

By assumption there is no pole at x_0 on the right hand side, hence $p_0(x)y(x)$ has no pole at x_0, too, thus $(x - x_0)^\mu$ divides $p_0(x)$. Statement (b) can be proved analogously, (c) follows by combining (a) and (b).

∎

Let's look at the poles of a solution $y(x)$ of (5.1) modulo the integers $\mathbb{Z}$: Obviously $y(x)$ can have (finitely many) poles of the form $x_0 + \mathbb{Z}$, then the previous lemma implies that

- the maximal pole from $x_0 + \mathbb{Z}$ is a root of $p_0(x)$
- the minimal pole from $x_0 + \mathbb{Z}$ is a root of $p_n(x - n)$

In other words, defining

$$\mathcal{R}^p_{q(x)} := \{x \in p + \mathbb{Z} \mid q(x) = 0\} \quad \text{where } p \in \overline{\mathbb{K}}, q \in \mathbb{K}[x]$$

as the set of roots (over $\overline{\mathbb{K}}$) modulo $\mathbb{Z}$ of a polynomial $q(x)$ we have proved the following:

Lemma 5.1.5 *Let $y(x)$ be a rational solution of (5.1) having the denominator $s(x)$, let $p \in \overline{\mathbb{K}}$ (resp. $p \in \overline{\mathbb{K}}/\mathbb{Z}$) and let $\mathcal{R}_{s(x)}^p \neq \emptyset$, then*

(a) $\max \mathcal{R}_{s(x)}^p \in \mathcal{R}_{p_0(x)}^p$

(b) $\min \mathcal{R}_{s(x)}^p \in \mathcal{R}_{p_n(x-n)}^p$

As a consequence we get

Lemma 5.1.6 *Let $y(x)$ be a rational solution of (5.1) having the denominator $s(x)$ and let $p \in \overline{\mathbb{K}}$ (resp. $p \in \overline{\mathbb{K}}/\mathbb{Z}$), then*

(a) *If $\mathcal{R}_{p_0(x)}^p \cap \mathcal{R}_{p_n(x-n)}^p = \emptyset$, then $s(x)$ has no root of the form $p+k$ for $k \in \mathbb{Z}$.*

(b) *If $\mathcal{R}_{p_0(x)}^p \cap \mathcal{R}_{p_n(x-n)}^p \neq \emptyset$ and $\max \mathcal{R}_{p_0(x)}^p < \min \mathcal{R}_{p_n(x-n)}^p$, then $s(x)$ has no root of the form $p + k$ for $k \in \mathbb{Z}$.*

(c) *$s(x)$ can only have roots from the set given by*

$$\bigcup_{p \in \overline{\mathbb{K}}/\mathbb{Z}} \{\min \mathcal{R}_{p_n(x-n)}^p, ..., \max \mathcal{R}_{p_0(x)}^p\}.$$

Proof: Clear!

■

Parts (b) and (c) of the last lemma give rise to the following definition having its roots in [Abr71] respectively in [MaWr94]:

Definition 5.1.7 *The* dispersion set *of a pair of polynomials, $f(x)$ and $g(x)$, is given by*

$$\mathrm{DS}(f(x), g(x)) = \{\alpha \in \mathbb{N} \mid \deg \gcd(f(x + \alpha), g(x)) > 0\}$$

If $\mathrm{DS}(f(x), g(x)) \neq \emptyset$, then the dispersion *of $f(x)$ and $g(x)$ is defined by*

$$\mathrm{dis}(f(x), g(x)) = \max\{\alpha \in \mathbb{N} \mid \deg \gcd(f(x + \alpha), g(x)) > 0\}$$

It should be mentioned that e.g. in [MaWr94] the dispersion is set to 0, if the corresponding dispersion set is empty.

Obviously

$$\mathrm{dis}(p_0(x), p_n(x - n)) = \max_p \{\max \mathcal{R}_{p_0(x)}^p - \min \mathcal{R}_{p_n(x-n)}^p\}$$

Thus we have:

Corollary 5.1.8 *If $\mathrm{DS}(p_0(x), p_n(x - n)) = \emptyset$, then the denominator of a rational solution $y(x)$ of (5.1) is simply 1, i.e. each rational solution is a polynomial solution.*

Now let's look at the case $N := \mathrm{dis}(p_0(x), p_n(x - n)) \geq 0$. The following theorem which can be found in a generalized form in [Bar99] already gives an explicit formula for Abramov's universal denominator:

Theorem 5.1.9 *Let an equation of the form (5.1) be given and let $N :=$ $\mathrm{dis}(p_0(x), p_n(x - n)) \geq 0$, then*

$$u(x) := \gcd\left(\prod_{i=0}^{N} p_0(x + i), \prod_{i=0}^{N} p_n(x - n - i)\right)$$

is multiple of the denominator of any rational solution of (5.1).

Proof: Let's rewrite (5.1) in the form

$$y(x + n) = \frac{f(x) - \big(p_{n-1}(x)y(x + n - 1) + ... + p_0(x)y(x)\big)}{p_n(x)} \qquad (5.2)$$

Carrying out the shifts $x \to x - 1, ..., x \to x - N$ yields

$$y(x + n - 1) \quad = \quad \frac{f(x - 1) - \sum_{j=0}^{n-1} p_j(x - 1)y(x + j - 1)}{p_n(x - 1)}$$

$$\vdots$$

$$y(x + n - N) \quad = \quad \frac{f(x - N) - \sum_{j=0}^{n-1} p_j(x - N)y(x + j - N)}{p_n(x - N)}$$

Now we can substitute the expression for $y(x + n - 1)$ in (5.2), then substitute the expression for $y(x + n - 2)$ and so on. What we get is an expression of the form

$$y(x + n) = \frac{\sum_{j=0}^{n-1} \hat{p}_j(x)y(x + j - N)}{\prod_{i=0}^{N} p_n(x - i)} \quad \text{with } \hat{p}_j(x) \in \mathbb{K}[x]$$

By the definition of N and Lemma 5.1.6 the poles of $y(x + n)$ differ from the poles of $y(x + n - 1 - N), ..., y(x - N)$. Therefore the poles of $y(x)$ must be determined by the product of the $p_n(x - i)$, thus (let $s(x)$ be the denominator of $y(x)$)

$$s(x + n) \mid \prod_{i=0}^{N} p_n(x - i) \Leftrightarrow s(x) \mid \prod_{i=0}^{N} p_n(x - n - i)$$

Analogously, we can prove that

$$s(x) \mid \prod_{i=0}^{N} p_0(x + i)$$

and the statement follows.

∎

Using Paule's gff-concept, we get from the previous theorem (w.l.o.g we may assume that p_0 and p_n are resp. have been made monic; this will merely eliminate the constant factor from $u(x)$)

$$u(x) = \gcd\left(\text{gff}\left(1, ..., 1, p_0(x+N)\right), \text{gff}\left(1, ..., 1, p_n(x-n)\right)\right)$$

where $p_0(x+N)$ and $p_n(x-n)$ are at the (N+1)-th position.

We will now show that this $u(x)$ is computed by Abramov's Algorithm 5.1.2:
Let $A_N(x) := A(x) = p_n(x-n)$ and $B_N(x) := B(x) = p_0(x)$, then

$$u(x) = \gcd\left(\text{gff}\left(1, ..., 1, B_N(x+N)\right), \text{gff}\left(1, ..., 1, A_N(x)\right)\right)$$

Furthermore, let (see Abramov's algorithm)

- $d_i(x) := \gcd(A_i(x), B_i(x+i))$ for $i = N, N-1..., 0$

- $A_{i-1}(x) = \frac{A_i(x)}{d_i(x)}$ for $i = N, N-1..., 1$

- $B_{i-1}(x) = \frac{B_i(x+i)}{d_i(x)}$ for $i = N, N-1..., 1$

then Abramov's algorithm returns

$$u(x) = \prod_{i=0}^{N} \prod_{j=0}^{i} d_i(x-j)$$

or written in gff-form

$$u(x) = \text{gff}\left(d_0(x), ..., d_N(x)\right)$$

Note that in the following lines we will write an extra 1 at the (N-1)-th position in the gff in order to avoid ambiguity:

$$u(x) = \gcd\left(\text{gff}\left(1, ..., 1, 1, B_N(x+N)\right), \text{gff}\left(1, ..., 1, 1, A_N(x)\right)\right) =$$

$$= \gcd\left(\text{gff}\left(1, ..., 1, 1, \frac{B_N(x+N)}{d_N(x)}d_N(x)\right), \text{gff}\left(1, ..., 1, 1, \frac{A_N(x)}{d_N(x)}d_N(x)\right)\right) =$$

$$= \gcd\left(\text{gff}\left(1, ..., 1, 1, B_{N-1}(x+N)\right), \text{gff}\left(1, ..., 1, 1, A_{N-1}(x)\right)\right) \cdot d_N(x)\underline{{}^{N+1}}$$

Because $\gcd(A_{N-1}(x), B_{N-1}(x+N)) = 1$ (by the definition of N), we also have $\gcd(A_{N-1}(x-i), B_{N-1}(x+N)) = 1$ for all $i \in \mathbb{N}$. This implies that $\gcd(A_{N-1}(x)\underline{{}^{N+1}}, B_{N-1}(x+N)) = 1$. Thus, we do not need the $B_{N-1}(x+N)$ and we get

$$\gcd\left(\text{gff}\left(1, ..., 1, 1, B_{N-1}(x+N)\right), \text{gff}\left(1, ..., 1, 1, A_{N-1}(x)\right)\right) =$$

$$= \gcd\Big(\mathrm{gff}\big(1,...,1,B_{N-1}(x+N-1),1\big),\mathrm{gff}\big(1,...,1,1,A_{N-1}(x)\big)\Big)$$

Because $\gcd(A_{N-1}(x-N),B_{N-1}(x)) = 1$ (by the definition of N), we also have $\gcd(A_{N-1}(x-N),B_{N-1}(x+i)) = 1$ for all $i \in \mathbb{N}$. This implies that $\gcd(A_{N-1}(x),B_{N-1}(x+N)\underline{\underline{N}}) = 1$. Thus, we do not need the $A_{N-1}(x)$ and we get

$$\gcd\Big(\mathrm{gff}\big(1,...,1,B_{N-1}(x+N-1),1\big),\mathrm{gff}\big(1,...,1,1,A_{N-1}(x)\big)\Big) =$$

$$= \gcd\Big(\mathrm{gff}\big(1,...,1,B_{N-1}(x+N-1),1\big),\mathrm{gff}\big(1,...,1,A_{N-1}(x),1\big)\Big)$$

All in all, we showed that

$$u(x) = \gcd\Big(\mathrm{gff}\big(1,...,1,1,B_N(x+N)\big),\mathrm{gff}\big(1,...,1,1,A_N(x)\big)\Big) =$$

$$= \gcd\Big(\mathrm{gff}\big(1,...,1,B_{N-1}(x+N-1),d_N(x)\big),\mathrm{gff}\big(1,...,1,A_{N-1}(x),d_N(x)\big)\Big)$$

By induction we finally get

$$\begin{aligned}u(x) &= \gcd\Big(\mathrm{gff}\big(d_0(x),...,d_N(x)\big),\mathrm{gff}\big(d_0(x),...,d_N(x)\big)\Big) =\\ &= \mathrm{gff}\big(d_0(x),...,d_N(x)\big)\end{aligned}$$

which completes the proof.

Remark 5.1.10 *Some remarks on the proof:*

- *Observe that in each reduction step - we explicitly showed the first - we extract the "longest chain" meaning the falling factorial of maximal length. Hence, the* gff*-conditions from Definition 2.2.8 automatically hold.*

- *It would have been possible to do the* gff*-part of the proof without the* gff*-concept using products like in Theorem 5.1.9. However, the* gff *prevents us from beginning from the wrong side: Theoretically, one could also start from 0 instead of N, but an analogous argumentation as we used to throw out the $B_{N-1}(x+N)$ and the $A_{N-1}(x)$ would not be correct in general. In fact, this is the reason why the "alternative" loop in Abramov's algorithm "FOR i:=0 TO N DO" does not always compute the correct result.*

- *A generalization of Theorem 5.1.9 is given in [Bar99] dealing with* linear *matrix difference equations (of order 1)*

$$\tau(Y) \cdot A - B \cdot Y = F \tag{5.3}$$

where $A \in \mathrm{GL}_n(\mathbb{K}[x])$, $B \in \mathrm{GL}_m(\mathbb{K}[x])$ and F and Y are $n \times m$ matrices over $\mathbb{K}[x]$. The resulting algorithm is - as expected - the same as Abramov's one, only the input polynomials $A(x) := \tau^{-1}(\det A)$ and $B(x) := \det B$ are different. It should be mentioned that the case $m = 1$ is already contained in [AbBa98], however, the approach used in [Bar99] is not only more general but also less technical.

- *We will see that the method used for proving Theorem 5.1.9 is also used in Lemma 5.2.7 on page 69 in the next section.*

Improvements

Simplifying the loop   Taking a closer look at Algorithm 5.1.2 one immediately notices that it is not necessary to do the complete loop "FOR i:=N DOWN TO 0 DO": One only has to take each integer i such that $A(x)$ and $B(x+i)$ have a nontrivial common divisor; in other words one has to compute the dispersion set of $B(x)$ and $A(x)$. As mentioned before it is important to use the larger integers first.

gcd-Improvement   Consider the following difference equation:

$$(x + 2) \cdot y(x + 2) - 2(x + 1) \cdot y(x + 1) + x \cdot y(x) = 0$$

Doing the substitution $y(x) := \frac{z(x)}{x}$ looks useful, because (after canceling common factors) we get

$$z(x + 2) - 2z(x + 1) + z(x) = 0$$

as the (simple) difference equation for $z(x)$ which has the (in this case even polynomial) solution $c_1 x + c_2$. Therefore $y(x) = c_1 + \frac{c_2}{x}$.
Obviously, this "substitution-trick" can be transformed into a method, by checking whether $\gcd(p_n(x - n), ..., p_1(x - 1), p_0(x))$ is nontrivial! In section 5.3 we will see that this improvement (which appeared in [AbBa98] for the first time) can really be essential.

After these both improvements for Abramov's algorithm, we will describe a further improvement which can be used *before* the search for the universal denominator. Although this criterion will turn out to be quite simple, it has not appeared in literature up to now:

Fast negative criterion for homogeneous equations   First, remember the general algorithm for finding rational solutions of linear difference equations at the very beginning of the section: In the second step we have to do the substitution $y(x) := \frac{z(x)}{u(x)}$ in $Ly = f = 0$, that means:

$$Ly = p_n(x)y(x + n) + ... + p_0(x)y(x) = 0$$

changes to

$$\tilde{L}z = \tilde{p}_n(x)z(x + n) + ... + \tilde{p}_0(x)z(x) = 0$$

Second, think back to Petkovšek's algorithm for finding polynomial solutions, where we derived the necessary condition (4.3) for the existence of a polynomial solution: The sum of the coefficients of x^d of $p_i(x)$ (where $d = degree(L)$) has

to vanish.

Now, observe that

$$\tilde{p}_i(x) = p_i(x) \cdot \prod_{j=0, j \neq i}^{n} u(x + j)$$

Hence, looking at the leading coefficients we have (note that the leading coefficient of $u(x)$ is the same as the leading coefficient of $u(x + j)$)

$$\tilde{p}_{i,\tilde{d}} = p_{i,d} \cdot \mathrm{lc}(u(x))^{n-1}$$

Consequently,

$$\sum_{i=0}^{n} p_{i,d} = 0 \Leftrightarrow \sum_{i=0}^{n} \tilde{p}_{i,\tilde{d}} = 0$$

where $d = degree(L)$ and $\tilde{d} = degree(\tilde{L})$. Thus, if condition (4.3) is not fulfilled, then we cannot have a rational solution, either.

Let's sum up all facts and write down the improved algorithm:

Algorithm 5.1.11 (Universal Denominator) *by Sergei Abramov*
INPUT: $Ly = f$ with

- $L = \sum_{k=0}^{n} p_k \cdot \tau^k$ *where $p_k \in \mathbb{K}[x]$*

- $p_0, p_n \neq 0$

- $f \in \mathbb{K}[x]$

OUTPUT: A polynomial $u(x)$ such that every rational solution $y(x)$ of $Ly = f$ can be written in the form $y(x) = \frac{z(x)}{u(x)}$, where $z(x)$ is a polynomial.

BEGIN

$\quad g(x) := \gcd(p_n(x - n), ..., p_1(x - 1), p_0(x))$

$\quad A(x) := \frac{p_n(x-n)}{g(x)}$

$\quad B(x) := \frac{p_0(x)}{g(x)}$

$\quad$ Compute the dispersion set $DS := \mathrm{DS}(B(x), A(x))$

$\quad$ FOR $i \in DS$ (in decreasing order) DO

$\qquad d_i(x) := \gcd(A(x), B(x + i))$

$\qquad A(x) := \frac{A(x)}{d_i(x)}$

$\qquad B(x) := \frac{B(x)}{d_i(x - i)}$

$\quad$ Return $u(x) := g(x) \cdot \prod_{i \in I} d_i(x)^{\underline{i+1}}$

END

♦

Let's also write down the algorithm for determining the universal denominator of *linear difference systems* of order 1 developed in [AbBa98]:

Algorithm 5.1.12 (Matrix Universal Denominator) *by Sergei Abramov and Moulay Barkatou*
INPUT: $D\tau(y) + My = f$, *with*

- $D = diag(d_1(x), ..., d_n(x)) \in \mathbb{K}[x]^n$

- $M \in GL_n(\mathbb{K}[x])$,

$$M = \begin{pmatrix} p_{11}(x) & \ldots & p_{1n}(x) \\ \vdots & & \vdots \\ p_{n1}(x) & \ldots & p_{nn}(x) \end{pmatrix}$$

- $f = (f_1, ..., f_n) \in \mathbb{K}[x]^n$

OUTPUT: A polynomial $u(x)$ such that for every rational solution $y(x) = (y_1(x), ..., y_n(x))$ of $D\tau(y) + My = f$ $u(x)$ is divisible by the denominator of any of the rational functions $y_1(x), ..., y_n(x)$.

BEGIN

> FOR $j \in \{1, 2, ..., n\}$ DO
>
> > $g_j(x) := \gcd\left(d_j(x-1), p_{1j}(x), p_{2j}(x)..., p_{nj}(x)\right)$
> >
> > FOR $i \in \{1, 2, ..., n\}$ DO $v_{ij}(x) := \frac{v_{ij}(x)}{g_j(x)}$
>
> $A(x) :=$lcm$\left(\frac{d_1(x-1)}{g_1(x)}, ..., \frac{d_n(x-1)}{g_n(x)}\right)$
>
> Compute $B(x)$ as the least common multiple of the denominators of the elements of M^{-1}
>
> Compute the dispersion set $DS := \mathrm{DS}(B(x), A(x))$
>
> FOR $i \in DS$ (in decreasing order) DO
>
> > $d_i(x) := GCD(A(x), B(x+i))$
> >
> > $A(x) := \frac{A(x)}{d_i(x)}$
> >
> > $B(x) := \frac{B(x)}{d_i(x-i)}$
>
> Return $u(x) :=$lcm$\left(g_1(x), ..., g_n(x)\right) \cdot \prod_{i \in I} d_i(x)^{\underline{i+1}}$

END

♦

5.2 The Denominator Bound By van Hoeij

The denominator bound $D(x)$ by Mark van Hoeij is nothing else than the universal denominator presented before with the only difference that $D(x)$ need not be a polynomial, but is allowed to be a rational function. As a consequence, the degree bound for the numerator may be smaller.

As we are following [vHo98], we will also switch from scalar equations to matrix equations:

Definition 5.2.1 *Given the (homogeneous) matrix difference equation $\tau(Y) = AY$, a vector $D = (D_1, D_2, ..., D_n)^T \in \mathbb{C}(x)^n$ is called a* denominator bound *if for each solution $Y = (Y_1, Y_2, ..., Y_n)^T \in \mathbb{C}(x)^n$ one has $Y_i = \frac{N_i}{D_i}$ for some $N_i \in \mathbb{C}[x]$ where:*

- $A \in GL_n(\mathbb{C}(x))$

- A *is defined like in Remark 2.3.3 for scalar equations (companion matrix)*

Definition 5.2.2 *Let $y \in \mathbb{C}(x)$ and $p \in \mathbb{C} \cup \{\infty\}$, then we define the* valuation *of y at p by $v_p(y) := \inf\{i \mid y_i \neq 0\}$, when y is written as $y = \sum_i y_i (x - p)^i$ for $p \in \mathbb{C}$ and $y = \sum_i y_i \frac{1}{x^i}$ for $p = \infty$.*

Proposition 5.2.3 *Some easy statements concerning the valuation of polynomials and rational functions:*

(a) $v_p(0) = \infty$ for $p \in \mathbb{C} \cup \{\infty\}$

(b) $v_\infty(q) = -\deg(q)$ for $q \in \mathbb{C}[x]$

(c) If $q(x)$ is a polynomial and p is finite, then $v_p(q)$ is the order of $q(x)$ at p, i.e. the highest power of $x - p$ that divides $q(x)$.

(d) If $r = \frac{N}{D}$ with $N, D \in \mathbb{C}[x]$, then $v_p(r) = v_p(N) - v_p(D)$ for $p \in \mathbb{C} \cup \{\infty\}$

(e) If $r = \frac{N}{D}$ with $N \in \mathbb{C}[x]$ and $D \in \mathbb{C}(x)$ then $\deg N = -v_\infty(D) - v_\infty(r)$

(f) $v_p(\sum_i r_i) = \min_i v_p(r_i)$ for $r_i \in \mathbb{C}(x)$

(g) $v_p(\prod_i r_i) = \sum_i v_p(r_i)$ for $r_i \in \mathbb{C}(x)$

(h) $\sum_{p \in \mathbb{C} \cup \{\infty\}} v_p(r) = 0$ for $r \in \mathbb{C}(x)^*$

With the knowledge of valuations, we can solve our problem of finding a denominator bound for a solution $Y = (Y_1, Y_2, ..., Y_n)^T$ by determining lower bounds for the valuations of the Y_i at finite points.

Because rational functions can only have a finite number of poles it is natural to start with the search for a finite set of points, such that for all other points p which are not in this set the valuation is at least zero, i.e. no factor $x - p$ in the denominator. Afterwards, we will take care of this finite set and look for valuations bounds.

Definition 5.2.4 *Let $A \in GL_n(\mathbb{C}(x))$ be the system matrix of the difference equation $\tau(Y) = AY$, then we define*

$$S := \{p \in \mathbb{C} \mid \det A(p) = 0 \vee A \text{ has a pole in } p\}$$

and

$$\overline{S} := \{p \in \mathbb{C} \mid \exists p_1, p_2 \in S : p - p_1 - 1 \in \mathbb{N} \wedge p_2 - p \in \mathbb{N}\}$$

Note: If A is a companion matrix then (after multiplying by the common denominator to get polynomial coefficients p_k)

$$S = \{p \in \mathbb{C} \mid p \text{ is a root of } p_0(x)p_n(x)\}$$

Looking at the definition of $\overline{S}$, it makes sense to divide S and $\overline{S}$ into equivalence classes with the help from the equivalence relation $p \sim q \Leftrightarrow p - q \in \mathbb{Z}$:

$$S = \bigcup_{p \in \mathbb{C}/\mathbb{Z}} S_p \quad \text{and} \quad \overline{S} = \bigcup_{p \in \mathbb{C}/\mathbb{Z}} \overline{S}_p$$

with

$$
\begin{aligned}
S_p &:= \{q \in S \mid q \in p + \mathbb{Z}\} = S \cap \{q \in \mathbb{C} \mid p - q \in \mathbb{Z}\} \\
\overline{S}_p &:= \{q \in p + \mathbb{Z} \mid \exists p_1, p_2 \in S_p : q - p_1 - 1 \in \mathbb{N} \wedge p_2 - q \in \mathbb{N}\}
\end{aligned}
$$

Proposition 5.2.5 *Easy properties concerning S_p, $\overline{S}_p$, S and $\overline{S}$:*

(a) $|S_p| = 1 \Rightarrow \overline{S}_p = \{\}$

(b) $|S_p| > 1 \Rightarrow \overline{S}_p = \{\min S_p + 1, \min S_p + 2, ..., \max S_p\}$

(c) S is finite

(d) $\overline{S}$ is finite

Theorem 5.2.6 *Let A and $\overline{S}$ be defined like in Definition 5.2.4 and let $Y = (Y_1, Y_2, ..., Y_n)^T$ be an arbitrary rational solution, then:*

$$\forall p \in \mathbb{C} \backslash \overline{S} : v_p(Y_i) \geq 0 \text{ for all } i \in \{1, 2, ..., n\}$$

Proof: Let $p \in \mathbb{C} \backslash \overline{S}$ be arbitrary. Suppose, that $v_p(Y_i) < 0 \Leftrightarrow Y_i$ has a pole at p. We consider two cases:

> Case 1: $p > \max \overline{S}$: Because of the difference equation we have $Y(p+1) = A(p)Y(p)$. Thus, Y_i has also a root at $p + 1$, as A(p) is regular. Again, $Y(p+2) = A(p+1)Y(p+1)$. Thus, Y_i has also a root at $p + 2$, as A(p+1) is regular, etc. Therefore Y_i would have infinitely many poles, which is impossible for rational functions.

> Case 2: $p < \min \overline{S}$: Analogously to case 1 with the difference equation in the form $Y(p-1) = A^{-1}(p-1)Y(p)$.

■

Observe the similarity of the above theorem together with part (b) of the previous proposition to Lemma 5.1.6 (c)!

Now, let $p \in \overline{S}$. We will derive bounds $B_i(p)$, such that $v_p(Y_i) \geq B_i(p)$: Take $N \in \mathbb{N}$ such that $p - N \notin \overline{S}$, then

$$
\begin{aligned}
Y(p) &= A(p-1) \cdot Y(p) = A(p-1) \cdot A(p-2) \cdot Y(p-2) = \dots \\
&= A(p-1) \cdot A(p-2) \cdot \dots \cdot A(p-N) \cdot Y(P-N) = \\
&= \tau^{-1}(A) \cdot \tau^{-2}(A) \cdot \dots \cdot \tau^{-N}(A) \cdot \tau^{-N}(Y) =: A_N \cdot \tau^{-N}(Y)
\end{aligned}
$$

Lemma 5.2.7 *Let $p \in \overline{S}$, N and A_N like above, and define $B_i^l(p)$ (the* left hand *bound) as the minimum of the valuations at p of the entries in the i'th row of A_N, then:*

$$
v_p(Y_i) \geq B_i^l(p) \text{ for all solutions } Y = (Y_1, \dots, Y_n) \text{ of } \tau(Y) = AY
$$

Proof: Let Y be an arbitrary solution of $\tau(Y) = AY$, then the entries of Y have no poles at $p - N$, because $p - N \notin \overline{S} \Leftrightarrow$ the entries of $\tau^{-N}(Y)$ have no pole at p $\Leftrightarrow v_p(\tau^{-N}(Y)) \geq 0$. Because $Y = A_N \cdot \tau^{-N}(Y)$, we have that Y_i is nothing else than the inner product of the i'th row of A_N with $\tau^{-N}(Y)$. Now, the statement follows from Proposition 5.2.3 (f) and (g).

■

Analogously, we can define the *righthand bound* $B_i^r(p)$ for a point $p \in \overline{S}$: Take $N \in \mathbb{N}$ such that $p + N \notin \overline{S}$, then

$$
Y(p) = A_{-N}(p) \cdot Y(p+N) \text{ where } A_{-N} := \tau^N(A_N^{-1})
$$

and $B_i^r(p)$ is defined as the minimum of the valuations at p in the i'th row of A_{-N}. Just like in Proposition 5.2.7, we can show that $v_p(Y_i) \geq B_i^r(p)$. We can now define our desired bounds $B_i(p)$ by $B_i(p) := \max\{B_i^l(p), B_i^r(p)\}$ Let's summarize all facts in

Algorithm 5.2.8 (Denominator Bound) *by Mark van Hoeij*
INPUT: $\tau(Y) = AY$ with

- *$A \in GL_n(\mathbb{C}(x))$ ($n \times n$ difference system of order 1)*

- *A is a companion matrix for a n-th order scalar difference equation*

OUTPUT: A vector $D(x) = (D_1, \dots, D_n)$ with rational function entries D_i such that every rational solution $y(x) = (y_1, \dots, y_n)$ of $\tau(Y) = AY$ can be written in the form $y_i(x) = \frac{z_i(x)}{D_i(x)}$, where $z_i(x)$ are polynomials.

BEGIN

$S := \{ p \in \mathbb{C} \mid \det A(p) = 0 \vee A \text{ has a pole in } p \}$

$\overline{S} := \{ p \in \mathbb{C} \mid \exists p_1, p_2 \in S : p - p_1 - 1 \in \mathbb{N} \wedge p_2 - p \in \mathbb{N} \}$

FOR $p \in \overline{S}$ DO

> Take $N_l \in \mathbb{N}$ such that $p - N_l \notin \overline{S}$, i.e. take $N_l \geq 1 + p - \min \overline{S}$
>
> $A_{N_l} := \tau^{-1}(A) \cdot \tau^{-2}(A) \cdot \ldots \cdot \tau^{-N_l}(A)$ (at p)
>
> FOR $i \in \{1, 2, ..., n\}$ DO compute $B_i^l(p)$ as the minimum of the valuations at p of the entries in the i'th row of A_{N_l}
>
> Take $N_r \in \mathbb{N}$ such that $p + N_r \notin \overline{S}$, i.e. take $N_r \geq 1 - p + \max \overline{S}$
>
> $A_{N_r} := A^{-1} \cdot \tau^1(A^{-1}) \cdot \ldots \cdot \tau^{N_r - 1}(A^{-1})$ (at p)
>
> FOR $i \in \{1, 2, ..., n\}$ DO compute $B_i^r(p)$ as the minimum of the valuations at p of the entries in the i'th row of A_{N_r}
>
> FOR $i \in \{1, 2, ..., n\}$ DO $B_i(p) := \max\{B_i^l(p), B_i^r(p)\}$

FOR $i \in \{1, 2, ..., n\}$ DO $D_i := \prod_{p \in \overline{S}} (x - p)^{-B_i(p)}$

Return $(D_1, ..., D_n)$

END

$\blacklozenge$

Improvements

We can speed up the entire algorithm by the following consideration: Suppose we have found $B_i(p) = 0$ for an arbitrary $p \in \overline{S}$, then $B_i(p+1) \geq 0$ if $p+1 \notin S$. This can be seen by $Y(p+1) = A(p)Y(p)$ - the valuation at p of the right hand side is at least 0. Hence, we can set $B_i(p+1) = 0$ and need not do the computation for $p+1$.

We can also use the gcd-Improvement described in section 5.1 on page 64.

Remark 5.2.9 *Remarks on Algorithm 5.2.8*

- *Observe that in case of a companion matrix (if one started with an scalar equation) the computation of A^{-1} is easy, because:*

$$
\begin{pmatrix}
0 & 1 & 0 & \ldots & 0 \\
0 & 0 & 1 & \ldots & 0 \\
\vdots & & & \ddots & \vdots \\
0 & 0 & \ldots & 0 & 1 \\
-a_0 & -a_1 & -a_2 & \ldots & -a_{n-1}
\end{pmatrix}^{-1}
=
\begin{pmatrix}
\mathbf{o} & \mathrm{I} \\
-a_0 & -a
\end{pmatrix}^{-1}
=
$$

$$
=
\begin{pmatrix}
-\frac{a}{a_0} & -\frac{1}{a_0} \\
\mathrm{I} & \mathbf{o}
\end{pmatrix}
=
\begin{pmatrix}
-\frac{a_1}{a_0} & -\frac{a_2}{a_0} & \ldots & -\frac{a_{n-1}}{a_0} & -\frac{1}{a_0} \\
1 & 0 & 0 & \ldots & 0 \\
0 & 1 & 0 & \ldots & 0 \\
\vdots & & & \ddots & \vdots \\
0 & \ldots & 0 & 1 & 0
\end{pmatrix}
$$

- *Of course, it is possible to vary the algorithm concerning the computation of the bounds $B_i(p)$. One can for example compute either the left hand or the right hand bound (take that bound, for which N (N_l or N_r) is smaller, because the matrix product will be smaller).*

- *Algorithm 5.2.8 can be easily extended for inhomogeneous equations: Suppose we have*

$$\tau(Y) = AY + Z \text{ where } A \in GL_n(\mathbb{C}(x)), Z \in \mathbb{C}(x)^n$$

then we can transform this into the following homogeneous difference equation:

$$\tau \begin{pmatrix} Y \\ \tilde{y} \end{pmatrix} = \begin{pmatrix} A & Z \\ \mathbf{o} & 1 \end{pmatrix} \begin{pmatrix} Y \\ \tilde{y} \end{pmatrix}$$

- *A MATHEMATICA-version of Algorithm 5.2.8 for scalar equations over $\mathbb{Q}$ has been implemented by Axel Riese together with Marko Petkovšek and is available at the RISC-Homepage.*

As a conclusion we mention the following remarkable theorem:

Theorem 5.2.10 *Let $p \in \overline{S}$. If all solutions of $\tau(Y) = AY$ are rational, then the left hand bound $B_i^l(p)$ and the right hand bound $B_i^l(p)$ are sharp. So these bounds coincide in this case.*

Proof: See [vHo98]

■

5.2.1   Improving The Scalar Case

Comparing van Hoeij's method to the proof of Abramov's algorithm we observe that in the scalar case the algorithm is not optimal[1] concerning $\overline{S}$. Let $\mathbb{K} = \mathbb{C}$, then by Lemma 5.1.6 (c) and the definition of $\overline{S}$:

$$\{\min \mathcal{R}^p_{p_n(x-n)}, ..., \max \mathcal{R}^p_{p_0(x)}\} = \{\min \mathcal{R}^p_{p_n(x)} + n, ..., \max \mathcal{R}^p_{p_0(x)}\}$$

is a subset of

$$\overline{S_p} = \{\min \mathcal{R}^p_{p_0(x)p_n(x)} + 1, ..., \max \mathcal{R}^p_{p_0(x)p_n(x)}\}$$

Let's shortly explain the difference concerning "$+n$" and "$+1$": In the scalar case we have to deal with a companion matrix and the solution vector is given by $(y(x), y(x+1), ..., y(x+n-1))$. Hence $\overline{S_p}$ also contains the roots of the denominators of $y(x+1), y(x+2), ..., y(x+n-1)$.

It should also be noted that in contrast to Abramov's algorithm van Hoeij's algorithm requires root finding over $\mathbb{C}$.

---

[1] See Example 5.3.9

We will now present a variant of van Hoeij's algorithm for the scalar case: This algorithm (which appeared in [Khm99] for the first time) uses the complete information about the structure of the denominator as well as avoids root finding over $\mathbb{C}$!

Observe that the following derivation is basically the same as the proof of Theorem 5.1.9 - only the conclusion will be different:

Left hand bound   We start with the (homogeneous) difference equation given by

$$\sum_{k=0}^{n} p_k(x)y(x-k) = 0 \Leftrightarrow y(x) = -\sum_{k=1}^{n} \frac{p_k(x)}{p_0(x)}y(x-k)$$

Let's rewrite this in the form

$$y(x) = \sum_{k=1}^{n} a_k^{(0)}(x)y(x-k) \tag{5.4}$$

No we can do the shift $x \to x-1$ in (5.4) to get an equation for $y(x-1)$. This can be plugged into (5.4) an we obtain an equation of the kind

$$y(x) = \sum_{k=2}^{n+1} a_k^{(1)}(x)y(x-k)$$

Carrying out also the shifts $x \to x-2$, $x \to x-3$, ..., $x \to x-N$ in (5.4) where - as "usual" - $N = \mathrm{dis}(p_0(x), p_n(x-n))$ and doing the substitutions yields

$$y(x) = \sum_{k=N+1}^{N+n} a_k^{(N)}(x)y(x-k) \tag{5.5}$$

By the definition of N and Lemma 5.1.6 we know that the roots of the denominator of $y(x)$ are different from the roots of the denominators appearing in $y(x-(N+1)), ..., y(x-(N+n))$. Thus, these roots and their multiplicities are determined by the coefficients $a_k^{(N)}$, namely, the least common multiple of the denominators of the $a_k^{(N)}$ can be used as the left upper bound for the denominator of $y(x)$.

Right hand bound   Starting with the equation

$$\sum_{k=0}^{n} p_k(x)y(x+k) = 0 \Leftrightarrow y(x) = \sum_{k=1}^{n} b_k^{(0)}(x)y(x+k)$$

we obtain by an analogous calculation (in the other direction)

$$y(x) = \sum_{k=N+1}^{N+n} b_k^{(N)}(x)y(x+k) \tag{5.6}$$

and the least common multiple of the denominators of the $b_k^{(N)}$ can be used as the right upper bound for the denominator of $y(x)$.

Taking the greatest common divisor of both bounds yields a (most probably better) bound for the denominator of $y(x)$.

This results in the following algorithm (we also use the gcd-improvement described in section 5.1 on page 64):

Algorithm 5.2.11 (Denominator Bound for Scalar Equations) *by Dennis Khmel'nov resp. Mark van Hoeij*

INPUT: $Ly = f$ with

- $L = \sum_{k=0}^n p_k \cdot \tau^k$ *where $p_k \in \mathbb{K}[x]$*

- $p_0, p_n \neq 0$

- $f \in \mathbb{K}[x]$

OUTPUT: A polynomial $D(x)$ such that every rational solution $y(x)$ of $Ly = f$ can be written in the form $y(x) = \frac{z(x)}{D(x)}$, where $z(x)$ is a polynomial.

BEGIN

$\quad g(x) := \gcd(p_n(x - n), ..., p_1(x - 1), p_0(x))$

$\quad N := \text{dis}(\frac{p_0(x)}{g(x)}, \frac{p_n(x-n)}{g(x)})$

$\quad$ Calculate equation (5.5) resp. the coefficients $a_k^{(N)}$

$\quad D_L(x) := \text{lcm}(\text{den } a_{N+1}^{(N)}, ..., \text{den } a_{N+n}^{(N)})$

$\quad$ Calculate equation (5.6) resp. the coefficients $b_k^{(N)}$

$\quad D_R(x) := \text{lcm}(\text{den } b_{N+1}^{(N)}, ..., \text{den } b_{N+n}^{(N)})$

$\quad D(x) := g(x) \cdot \gcd(D_L(x), D_R(x))$

$\quad$ Return $D(x)$

END

$\blacklozenge$

Remark 5.2.12 *Remarks on Algorithm 5.2.11:*

- *Observe that Algorithm 5.2.8 (in the scalar case) and Algorithm 5.2.11 are not equivalent, as the result of the latter is always a polynomial. Moreover, note that Algorithm 5.2.11 differs from Algorithm 5.2.8 in that it does not consider the possible roots of the denominator one by one, but analyzes all possible roots simultaneously.*

- *Comparing the derivation of Algorithm 5.2.11 to the proof of Theorem 5.1.9 it immediately follows that the denominator polynomial $D(x)$ from Algorithm 5.2.11 divides Abramov's universal denominator $u(x)$. However, van Hoeij's denominator bound D (from Algorithm 5.2.8) does not have this property in general.*

In the following example section we will point out the differences between all three presented algorithms for finding rational solutions of linear difference equations (in the scalar case). Looking at the main ideas of all algorithms it is rather obvious that on the one hand Abramov's algorithm will be faster and on the other hand van Hoeij's approach (at least the improved variant for the scalar case) will be more exact. In this context, it should be observed that Abramov's algorithm only uses the leading and trailing coefficient whereas the algorithms using van Hoeij's method take care of all coefficients!
Of course, the lower exactness of Abramovs's method can cause a difference equation (for the polynomial solution) which is of higher degree than the (often optimal) difference equation resulting from van Hoeij's method. Consequently, the computing time for the following search for the polynomial solution has to be taken into consideration, too.

5.3   Examples and Comparison

Some easy examples which cause no problems for all algorithms (Algorithm 5.1.11, Algorithm 5.2.8) and Algorithm 5.2.11, but already indicate some differences:

Example 5.3.1 *Given $Ly = 0$, with $L = (x + 1)(2x + 1)\tau - (x + 3)(2x - 1)$, thus we have*

$$(x + 1)(2x + 1)y(x + 1) - (x + 3)(2x - 1)y(x) = 0$$

with the general rational solution

$$y(x) = c \cdot \frac{(x + 1)(x + 2)}{2x - 1}$$

The universal denominator by Abramov (and also the denominator polynomial of Algorithm 5.2.11) is equal to $2x - 1$ (already computed by the gcd-improvement), the denominator bound by van Hoeij is equal to $\frac{1}{2}\frac{2x-1}{(x+1)(x+2)}$.

Example 5.3.2 *Given $Ly = 0$, with $L = (4x + 33)\tau^2 + 4(x - 16)\tau - (8x - 7)$, thus we have*

$$(4x + 33)y(x + 2) + 4(x - 16)y(x + 1) - (8x - 7)y(x) = 0$$

with the general rational solution

$$y(x) = c \cdot (x^2 + 2x + 3)$$

All algorithms compute the sharp denominator 1.

Example 5.3.3 *Given $Ly = f$, with $L = (x+4)\tau^3 + (x+3)\tau^2 - x\tau + (x^2-1)$ and $f = \frac{x+2}{x+1}$, thus we have*

$$(x+4)y(x+3) + (x+3)y(x+2) - xy(x+1) + (x^2-1)y(x) = \frac{x+2}{x+1}$$

with the general rational solution

$$y(x) = \frac{1}{x^2-1}$$

In this case the universal denominator is equal to $(x-1)x(x+1)$ and the denominator bound (and also the denominator polynomial of Algorithm 5.2.11) is equal to $(x-1)(x+1)$.

Example 5.3.4 *Given $Ly = 0$, with $L = p_3(x)\tau^3 + p_2(x)\tau^2 + p_1(x)\tau + p_0(x)$, where*

$$\begin{aligned}
p_3(x) &= 2x^3 + 13x^2 + 22x + 8 \\
p_2(x) &= -2x^3 - 11x^2 - 18x - 9 \\
p_1(x) &= 2x^3 + x^2 - 6x \\
p_0(x) &= -2x^3 + x^2 + 2x - 1
\end{aligned}$$

with the general rational solution

$$y(x) = c \cdot \frac{2x-3}{x^2-1}$$

In this case all algorithms compute the same denominator which is $x^2 - 1$. In this example, Algorithm 5.1.2 (without the gcd-improvement) would yield the universal denominator $x(x^2-1)$.

Example 5.3.5 *Given $Ly = 0$ with $L = (x+4)\tau^2 - (x^2+6x+7)\tau + x(x+2)$, thus we have*

$$(x+4)y(x+2) - (x^2+6x+7)y(x+1) + x(x+2)y(x) = 0$$

with the general (rational) solution

$$y(x) = \frac{c}{x(x+1)(x+2)}$$

No problems arise and we immediately get the sharp denominator $x(x+1)(x+2)$ from all algorithms. However, using the loop "FOR i:=0 TO N DO" (resp. taking the nonnegative integers in Algorithm 5.1.11 in increasing order) would yield the wrong universal denominator $u(x) = x + 2$.

The next example shows that the gcd-improvement is really essential:

Example 5.3.6 *Given $Ly = 0$ with $L = \tau^2 - 2\frac{(x-99)(x+101)}{(x-98)(x+102)}\tau + \frac{(x-100)(x+100)}{(x-98)(x+102)}$, thus we have*

$$y(x+2) - 2\frac{(x-99)(x+101)}{(x-98)(x+102)}y(x+1) + \frac{(x-100)(x+100)}{(x-98)(x+102)}y(x) = 0$$

with the general (rational) solution

$$y(x) = \frac{c_1 + c_2 x}{(x-100)(x+100)}$$

Without the gcd-improvement we would get as the universal denominator a polynomial of degree 201 - namely $u(x) = (x-100)(x-99)...(x+99)(x+100)$ - and the following search for the polynomial solution would be very costly. However, using the gcd-improvement we immediately get the sharp denominator $(x-100)(x+100)$. This polynomial is also equal to van Hoeij's denominator bound and to the denominator polynomial of Algorithm 5.2.11.

Example 5.3.7 *Given $Ly = 0$ with $L = p_2(x)\tau^2 + p_1(x)\tau + p_0(x)$ with*

$$
\begin{aligned}
p_2(x) &= (x^2 + 4x + 1)(x^2 + 4x + 6)(x^4 - 28x^3 - 55x^2 - 26x - 9) \\
p_1(x) &= -2(x^2 + 2x - 2)(x^2 + 2x + 3)(x^4 - 26x^3 - 97x^2 - 82x - 18) \\
p_0(x) &= (x^2 - 3)(x^2 + 2)(x^4 - 24x^3 - 133x^2 - 216x - 117)
\end{aligned}
$$

with the general (rational) solution

$$y(x) = c_1 \cdot \frac{3x+1}{x^2 + 2} + c_2 \cdot \frac{x}{x^2 - 3}$$

The universal denominator is equal to $(x^2 + 2)(x^2 - 3)$, which is again already found by the gcd-improvement. In this example an implementation of van Hoeij's algorithm over $\mathbb{Q}$ will compute no solution, because the denominator cannot be factored into linear factors over $\mathbb{Q}$. If the computation is done over $\mathbb{C}$ (resp. over a splitting field of $\mathbb{Q}$), then we will get a sharp denominator bound - however, computation in splitting fields tends to be slow. Thus, the gcd-improvement can also be essential for van Hoeij's algorithm.
Note that an implementation of Algorithm 5.2.11 has no problems to compute the denominator polynomial $(x^2+2)(x^2-3)$ - even without the gcd-improvement.

Example 5.3.8 *Given $Ly = f$ with $L = \tau^2 + \tau + 1$ and*

$$f = \frac{1}{(x - 100)(x + 100)} + \frac{1}{(x - 99)(x + 101)} + \frac{1}{(x - 98)(x + 102)}$$

with the general rational solution

$$y(x) = \frac{1}{(x - 100)(x + 100)}$$

Although, the gcd-improvement already computes $(x - 100)(x + 100)$, the universal denominator $u(x)$ becomes a polynomial of degree 398 - namely

$$u(x) = (x - 100)(x - 99)(x - 98)^2(x - 97)^2...(x + 98)^2(x + 99)(x + 100).$$

Consequently, the following polynomial solution has degree 400 and the computation takes some minutes. Van Hoeij's algorithm and also Algorithm 5.2.11 immediately compute the denominator $(x - 100)(x + 100)$ which is sharp.

The following example illustrates that van Hoeij's algorithm may yield a denominator bound which is of higher degree than Abramov's universal denominator because - as mentioned in the previous section - van Hoeij's algorithm is not optimal concerning $\overline{S}$:

Example 5.3.9 *Given $Ly = f$ with $L = \tau^2 + (x^3 + 15x^2 + 3)\tau - x(x + 5)(x + 10)$ and $f = 4 - 50x$, thus we have*

$$y(x + 2) + (x^3 + 15x^2 + 3)y(x + 1) - x(x + 5)(x + 10)y(x) = 4 - 50x$$

with the general (rational) solution

$$y(x) = 1$$

Abramov's universal denominator and the denominator polynomial of Algorithm 5.2.11 are equal to 1. Van Hoeij's algorithm, however, returns the denominator bound $x(x + 1)(x + 2)(x + 3)(x + 4)$.

Obviously, the computation of the universal denominator is fast in each example - this is not true for van Hoeij's algorithm (and also its variant for the scalar case): The computation becomes slow as soon as the degree of the denominator of the solution grows:

Example 5.3.10 *Given $Ly = 0$ with $L = (x + 100)\tau - x$, thus we have*

$$(x + 100)y(x + 1) - xy(x) = 0$$

with the general (rational) solution

$$y(x) = \frac{c}{x(x + 1)...(x + 99)}$$

Abramov's algorithm immediately computes the sharp universal denominator $x(x+1)...(x+99)$. The same result is also computed by the other algorithms, but it takes quite a long time.

Let's sum up the main differences between all algorithms:

- **Exactness:** The denominator polynomial of Algorithm 5.2.11 always divides Abramov's universal denominator and also (the denominator) of van Hoeij's denominator bound. Usually, Abramov's universal denominator has highest degree.

- **Computation speed:** Abramov's algorithm is fast for all examples. For some special examples van Hoeij's algorithm and its variant may become problems.

- **Full factorization (splitting fields):** Only van Hoeij's algorithm needs full factorization resp. root finding over $\mathbb{C}$.

Chapter 6

Hypergeometric Solutions

6.1 Petkovšek's Hyper

Problem 6.1.1 *We are given the following problem: Find all hypergeometric solutions $y \in \mathbb{H}$ of $Ly = 0$, where we suppose the following:*

- $L = \sum_{k=0}^{n} p_k(x) \cdot \tau^k$

- $p_k(x) \in \mathbb{K}[x]$

- $p_n \neq 0$, $p_0 \neq 0$

Note that in this section we merely look at homogeneous equations - the inhomogenous case will be treated in section 6.4.

Like in [Pet92], we will present the main idea for the general (homogeneous) order-two equation

$$p(x) \cdot y(x+2) + q(x) \cdot y(x+1) + r(x) \cdot y(x) = 0 \qquad (6.1)$$

Using the definition of "hypergeometric" is the first key step for the algorithm: We set $R(x) := \frac{y(x+1)}{y(x)}$ and divide (6.1) by $y(x)$ which yields

$$
\begin{aligned}
p(x) \cdot \frac{y(x+2)}{y(x+1)} \cdot \frac{y(x+1)}{y(x)} + q(x) \cdot \frac{y(x+1)}{y(x)} + r(x) \cdot \frac{y(x)}{y(x)} &= 0 \\
p(x) \cdot R(x+1) \cdot R(x) + q(x) \cdot R(x) + r(x) &= 0 \quad (6.2)
\end{aligned}
$$

By this substitution we gained a recurrence for the rational function $R(x)$ - unfortunately, not a linear one, so we cannot use our knowledge from Chapter 5 directly. We will instead use a similar idea coming from the following Lemma which appeared - without the conditions (6.5) and (6.6) - in [Gos78] for the first time:

Lemma 6.1.2 *Let $R(x)$ be a non-zero rational function, then there exists a nonzero constant $Z \in \mathbb{K}$ and monic polynomials $A(x)$, $B(x)$ and $C(x)$ over $\mathbb{K}$ such that:*

$$R(x) = Z \cdot \frac{A(x)}{B(x)} \cdot \frac{C(x+1)}{C(x)} \qquad (6.3)$$

where

$$\gcd(A(x), B(x+k)) = 1 \text{ for all } k \in \mathbb{N} \qquad (6.4)$$

$$\gcd(A(x), C(x)) = 1 \qquad (6.5)$$

$$\gcd(B(x), C(x+1)) = 1 \qquad (6.6)$$

Proof: Without loss of generality we may assume that $R(x) = \frac{A_1(x)}{B_1(x)}$ where $A_1(x)$ and $B_1(x)$ are monic polynomials with $\gcd(A_1(x), B_1(x)) = 1$ - that means we have already extracted the Z. We will first construct (algorithmically!) a factorization of the type (6.3) which satisfies (6.4). Afterwards we will show that the factorization also satisfies (6.5) and (6.6).
Suppose $\gcd(A_1(x), B_1(x+k)) = 1$ for all $k \in \mathbb{N}$, then we can simply take $A(x) = A_1(x)$, $B(x) = B_1(x)$ and $C(x) = 1$. Now suppose that there exists $h_1 \in \mathbb{N}$ such that $u_1(x) := \gcd(A_1(x), B_1(x+h_1)) \neq 1$. Define $A_2(x) := \frac{A_1(x)}{u_1(x)}$ and $B_2(x) := \frac{B_1(x)}{u_1(x-h_1)}$, then

$$R(x) = \frac{A_1(x)}{B_1(x)} = \frac{A_2(x)}{B_2(x)} \cdot \frac{u_1(x)}{u_1(x-h_1)} = \frac{A_2(x)}{B_2(x)} \cdot \frac{C_2(x+1)}{C_2(x)}$$

with

$$C_2(x) := u_1(x-1) \cdot u_1(x-2) \cdot \ldots \cdot u_1(x-h_1) = \prod_{j=1}^{h_1} u_1(x-j)$$

Repeating this procedure on $\frac{A_2(x)}{B_2(x)}$ an so on yields

$$R(x) = \frac{A_m(x)}{B_m(x)} \cdot \frac{u_{m-1}(x)}{u_{m-1}(x-h_{m-1})} \cdot \ldots \cdot \frac{u_1(x)}{u_1(x-h_1)}$$

with $\gcd(A_m(x), B_m(x+k)) = 1$ for all $k \in \mathbb{N}$. Thus we can take $A(x) := A_m(x)$, $B(x) := B_m(x)$ and

$$C(x) := \prod_{i=1}^{m-1} C_i(x) \quad \text{with} \quad C_i(x) := \prod_{j=1}^{h_i} u_i(x-j)$$

To show (6.5) and (6.6) it is essential to assume that $h_1 < h_2 < \ldots < h_{m-1}$ which causes of course no problems in the computation above (but can change the final result). We first prove that for $i \in \{2, 3, \ldots, m-1\}$: $\gcd(A_i(x), C_i(x)) = 1$. Because of our assumption we have $\gcd(A_i(x), B_i(x+j)) = 1$ for $0 \leq j < h_i$,

therefore $\gcd(A_i(x), u_i(x - h_i + j)) = 1$ for $0 \leq j < h_i$. The statement follows from the definition of $C_i(x)$. In order to see (6.5) observe

$$R(x) = \frac{A_i(x)}{B_i(x)} \cdot \frac{C_i(x+1)}{C_i(x)} \cdot \dots \cdot \frac{C_2(x+1)}{C_2(x)} \quad \text{for } i \in \{2, 3, ..., m\}.$$

Condition (6.6) can be proven analogously.

∎

Lemma 6.1.3 *Let $a, b, c, A, B, C \in \mathbb{K}[x]$ such that*

$$\gcd(a(x), c(x)) = \gcd(b(x), c(x+1)) = \gcd(A(x), B(x+k)) = 1, \forall k \in \mathbb{N}.$$

If

$$\frac{a(x)}{b(x)} \cdot \frac{c(x+1)}{c(x)} = \frac{A(x)}{B(x)} \cdot \frac{C(x+1)}{C(x)}, \tag{6.7}$$

then $c \mid C$. As a consequence the representation of the form (6.3) together with the conditions (6.4), (6.5) and (6.6) is unique.

Proof: Let

$$g(x) = \gcd(c(x), C(x)), \quad d(x) = \frac{c(x)}{g(x)}, \quad D(x) = \frac{C(x)}{g(x)}$$

Then $\gcd(d(x), D(x)) = \gcd(a(x), d(x)) = \gcd(b(x), d(x+1)) = 1$. Substituting $d(x)$ and $D(x)$ in (6.7), canceling $g(x)g(x+1)$ and multiplying by the common denominator yields

$$A(x)b(x)d(x)D(x+1) = a(x)B(x)D(x)d(x+1)$$

Thus - because of the gcd-properties,

$$d(x) \quad | \quad B(x)d(x+1)$$
$$d(x+1) \quad | \quad A(x)d(x).$$

Using this relation repeatedly we obtain for $k \in \mathbb{N}$:

$$d(x) \quad | \quad B(x)B(x+1)...B(x+k-1)d(x+k)$$
$$d(x) \quad | \quad A(x-1)A(x-2)...A(x-k)d(x-k).$$

Since $\mathbb{K}$ has characteristic 0, $\gcd(d(x), d(x+k)) = \gcd(d(x), d(x-k)) = 1$ for all large enough k. It follows that $d(x)$ divides both, $A(x-1)A(x-2)...A(x-k)$ and $B(x)B(x+1)...B(x+k-1)$ for all large enough k. But these two polynomials are by assumption relatively prime, so $d(x)$ is a constant and therefore $\gcd(g(x), c(x)) = c(x)$. Hence $c(x)$ divides $C(x)$.

∎

Remark 6.1.4 *A representation of a rational function in the form (6.3) with condition (6.4) is called* Gosper-form *or* G-form. *Together with the conditions (6.5) and (6.6) the (unique) representation is called* Gosper-Petkovšek-form *or* GP-form.

Let's continue with the derivation of Petkovšek's Hyper: After substituting (6.3) into (6.2) and multiplying by the common denominator we get

$$
\begin{aligned}
& Z^2 \cdot p(x) \cdot A(x+1) \cdot A(x) \cdot C(x+2) + \\
+ \quad & Z \cdot q(x) \cdot A(x) \cdot B(x) \cdot C(x+1) + \\
+ \quad & r(x) \cdot B(x+1) \cdot B(x) \cdot C(x) = 0
\end{aligned}
\tag{6.8}
$$

Now we have gained an recurrence of order two for the polynomial $C(x)$, which enables us to use an algorithm from Chapter 4 - however, we need additional information about $A(x)$, $B(x)$ and Z:

- In (6.8) $A(x)$ is part of the first two summands, therefore $A(x)$ has to divide the third one. Because of (6.4) and (6.5) it follows that $A(x)$ divides $r(x)$ (which is given).

- Analogously $B(x+1)$ is part of the last two summands of (6.8), therefore it has to divide the first one. Because of (6.4) and (6.6) it follows that $B(x+1)$ divides $p(x)$.

- With given $A(x)$ and $B(x)$ we can determine the possible values of Z by considering the leading coefficient of the left hand-side of (6.8) which yields a quadratic equation with known coefficients.

This method for order 2 can easily be generalized for arbitrary order n. Let's summarize the complete algorithm in

Algorithm 6.1.5 (Hyper) *by Marko Petkovšek*
INPUT: $Ly = 0$ with

- $L = \sum_{k=0}^{n} p_k \cdot \tau^k$ *where $p_k \in \mathbb{K}[x]$*

- $p_0, p_n \neq 0$

OUTPUT: A set $\mathcal{B}$ of hypergeometric functions y_i (not necessarily linear independent) such that all y_i are solutions of $Ly = 0$

BEGIN

$\mathcal{B} = \emptyset$

Compute all monic factors $A(x)$ of $p_0(x)$

Compute all monic factors $B(x)$ of $p_n(x - n + 1)$

FOR ALL pairs $A(x)$ and $B(x)$ which satisfy (6.4) DO

FOR $k \in \{0, 1, ..., n\}$ DO

$P_k(x) := p_k(x) \cdot \prod_{j=0}^{k-1} A(x+j) \cdot \prod_{j=k}^{n-1} B(n+j)$

Compute the leading coefficient α_k of $P_k(x)$

Compute all non-zero solutions Z of $\sum_{k=0}^{n} \alpha_k Z^k = 0$

FOR ALL Z DO

Find all non-zero polynomial solutions $C(x)$ of

$$\sum_{k=0}^{n} Z^k P_k(x) C(x+k) = 0 \qquad (6.9)$$

$R(x) := Z \cdot \frac{A(x)}{B(x)} \cdot \frac{C(x+1)}{C(x)}$

Compute $y(x)$ as the of $y(x+1) = R(x) \cdot y(x)$

$\mathcal{B} := \mathcal{B} \cup \{y\}$

Return $\mathcal{B}$

END

♦

Remark 6.1.6 *Remarks on Algorithm 6.1.5*

- *Observe, that it is possible to cancel out the factor $A(x)B(x+n-1)$ from (6.9) which can reduce the degree of the difference equation.*

- *One can generalize Hyper to find m-hypergeometric solutions (these are functions y such that $\tau^m(y) = r \cdot y$ for a positive integer m and a rational function r). For details see [PeSa93].*

- *Note that for the special case "constant coefficients" the equation for Z is nothing else than the characteristic equation given by L itself.*

- *MATHEMATICA-versions of Algorithm 6.1.5 over $\mathbb{Q}$ and $\mathbb{C}$ have been implemented by Marko Petkovšek and are available as "Hyper" at his Homepage (http://www.fmf.uni-lj.si/ petkovsek/). Note that Petkovšek's implementation returns the set of all $R(x)$ - the certificates, not the solutions itself.*

6.2   Van Hoeij's Singularities-Approach

The main idea of van Hoeij's algorithm is the construction of first order right hand factors $\tau - r$ (which corresponds basically to the construction of $R(x)$ in Hyper). The solution of $(\tau - r)y = 0$, however, does not have to be an exact solution of $Ly = 0$, but has to be a solution modulo $\mathbb{C}(x)$. In other words we try to find solutions of $Ly = 0$ in $\mathbb{H}/\mathbb{C}(x)$.

Because the r of the factor $\tau - r$ is a rational function, we try to construct r

by determining all roots and poles of r (together with their orders/valuation). Additionally we will study the singularity of L at the point $p = \infty$.

Let's start with the singularity $p = \infty$:

Definition 6.2.1 *Let $L = \tau - r$ with $r \in \mathbb{C}(x)$. As r can be (uniquely) written as*

$$r = c \cdot x^n \cdot \left(1 + \frac{d}{x} + \mathcal{O}\left(\frac{1}{x^2}\right) \right)$$

for some $c, d \in \mathbb{C}$ and $n \in \mathbb{Z}$, we define

$$g_\infty(L) := (c, n, d + \mathbb{Z}) \in \mathcal{H}_\infty := \mathbb{C}^* \times \mathbb{Z} \times (\mathbb{C}/\mathbb{Z}).$$

Example 6.2.2 *Let $L = \tau - \frac{x^2+1}{2x-6}$, then $g_\infty(L) = (\frac{1}{2}, 1, 3 + \mathbb{Z}) = (\frac{1}{2}, 1, \mathbb{Z})$, because*

$$
\begin{aligned}
r &= \frac{1}{2} \cdot \frac{x^2 + 1}{x - 3} \\
&= \frac{1}{2} \cdot x^1 \cdot \frac{x^2 + 1}{x^2 - 3x} \\
&= \frac{1}{2} \cdot x^1 \cdot \left(1 + \frac{3x + 1}{x^2 - 3x}\right) \\
&= \frac{1}{2} \cdot x^1 \cdot \left(1 + \frac{3}{x} + \frac{10}{x^2 - 3x}\right)
\end{aligned}
$$

Now, the following question appears: In order to compute all first order right hand factors $\tau - r$ of an difference operator L, can we compute the (c, n, d)-information from L, or in other words can we compute the set
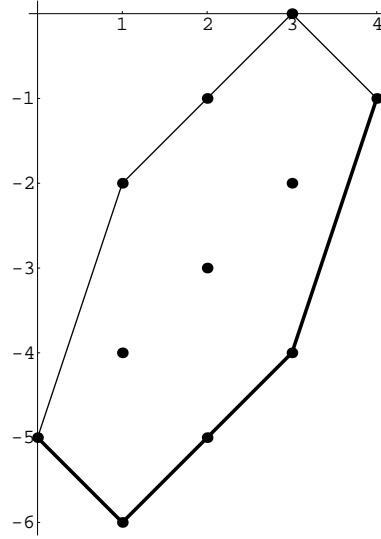
$$\overline{g}_\infty(L) := \{g_\infty(M) \mid M \text{ is a first order right hand factor of } L\}$$

The answer is "yes" - with the help of Puisseux's Theorem (resp. its proof), which shows that $\mathbb{C}((x))$ is algebraically closed, that means every polynomial $f(x, y) \in \mathbb{C}((x))[y]$ has a solution $\bar{y} \in \mathbb{C}((x))$ with $f(x, \bar{y}) = 0$. We will adapt this theorem for the ring $\mathbb{C}(x)[\tau]$ (resp. $\mathbb{C}[x][\tau]$). As a consequence we know in advance:

Lemma 6.2.3 *The number of elements of $\overline{g}_\infty(L)$ is at most $\operatorname{order}(L)$.*

The proof of Puisseux's Theorem is constructive and can be used for the computation of $\overline{g}_\infty(L)$ as showed in the following. The main tools for this proof are the so called Newton polygon and Newton polynomial:

Definition 6.2.4 *Let $L = \sum_{i=0}^{n} \sum_{j=0}^{m} a_{i,j} x^j \tau^i$, then the τ-polygon of Newton $N_\tau(L)$ is defined as the convex hull of all points $(i, -j)$ in a cartesian coordinate system for which $a_{ij} \neq 0$ for $0 \leq i \leq n$ and $0 \leq j \leq m$. Between the leftmost-undermost point P_0 and the rightmost-undermost point P_r (counterclockwise) the τ-polygon possesses a finite number of edges with rational slopes.*

Note that the points and edges between the rightmost-undermost and the leftmost-undermost point (from the right to the left) are of no interest.

Example 6.2.5 *Let's construct the τ-polygon of Newton of*

$$L = x\tau^4 - (5x^4 + 5x^2 - 7)\tau^3 + (25x^5 - 31x^3 - 35x)\tau^2 - (20x^6 - 175x^4 - 28x^2)\tau - 140x^5$$

The polygon consists of the following points $P_0 = (0, -5)$, $P_1 = (1, -6)$, $P_2 = (2, -5)$, $P_3 = (3, -4)$ and $P_4 = P_r = (4, -1)$. (The other points of the polygon are of no interest.) Therefore the polygon possesses the following slopes: -1 between P_0 and P_1, 1 between P_1 and P_3 (P_2 lies on this line), and 3 between P_3 and P_4.

Proposition 6.2.6 *Some easy properties of the τ-polygon of Newton:*

(a) *$N_\tau(L)$ is a finite subset of $\mathbb{R}^+ \times \mathbb{R}^-$*

(b) *If $L \in \mathbb{C}[\tau]$, then the τ-polygon of Newton degenerates to a line which is a finite subset of the $\mathbb{R}^+$-axis.*

(c) *If L is normal, then $P_0 = (0, -d)$, where d is the degree of the coefficient of τ^0.*

(d) *If $order(L) = n$, then $P_r = (n, -d)$, where d is the degree of the coefficient of τ^n.*

(e) *All points between P_0 and P_r are of the form $(k, -d)$ where d is the degree of the coefficient of τ^k.*

Definition 6.2.7 *Let $L = \sum_{i=0}^{n} \sum_{j=0}^{m} a_{i,j} x^j \tau^i$, let $N_\tau(L)$ be the corresponding τ-polygon of Newton and let p be a slope of the polygon between the points $(i_0(p), -j_0(p)), ..., (i_{\eta_p}(p), -j_{\eta_p}(p))$, that means that*

$$\frac{j_k(p) - j_0(p)}{i_k(p) - i_0(p)} = -p \quad \forall k \in \{1, 2, ..., \eta_p\}$$

Then the characteristic polynomial *or* Newton polynomial *(at p) $P_p(T)$ is defined by*

$$P_p(T) := \sum_{k=0}^{\eta_p} a_{i_k(p), j_k(p)} T^{i_k(p) - i_0(p)}$$

Example 6.2.8 *Continuation of Example 6.2.5:*

$$\begin{aligned} P_{-1}(T) &= -140 - 20T = -20(T + 7) \\ P_1(T) &= -20 + 25T - 5T^2 = -5(T^2 - 5T + 4) \\ P_3(T) &= -5 + T = T - 5 \end{aligned}$$

Proposition 6.2.9 *Some easy properties of the Newton polynomial:*

(a) $\deg P_p(T) = i_{\eta_p}(p) - i_0(p)$ *which can be seen as the "length" of the corresponding edge.*

(b) *If $L \in \mathbb{C}[\tau]$, then $P_p(\tau) = L$*

Let's formulate the main result which comes from Puisseux's Theorem in

Theorem 6.2.10 *Let $L \in \mathbb{C}[x][\tau]$, let M be a right hand factor of L and let $g_\infty(M) = (c, n, d)$, then n is a slope of the τ-polygon of Newton and c is a root of the corresponding Newton polynomial.*

Thus, by this theorem we get all possible n by finding all *integer* slopes of the τ-polygon of Newton and c by finding all roots of the corresponding Newton polynomial. What remains is the d:
Suppose we have c and n, then we can calculate the operator $\tilde{L} = L \circledS (\tau - \frac{1}{cx^n})$. By definition of the symmetric product $\tilde{L}$ now contains the right hand factor of the form $\tau - (1 + \frac{d}{x} + \mathcal{O}(\frac{1}{x^2}))$ - that means $c = n = 1$. Now, the possible d's can be computed as the roots of the Newton polynomial for slope 0 of the Δ-polygon of Newton (Definitions below, for more details see [BaDu94], [Duv83], [Tou87]). By definition of the d, it suffices to take the roots modulo the integers.

Definition 6.2.11 *Let $L = \sum_{i=0}^{n} \sum_{j=0}^{m} c_{i,j} x^j \Delta^i$ and let $\mathcal{I} := \{(x,y) \in \mathbb{R}^2 \mid y \geq 0 \land x \leq y\}$. Then the Δ-polygon of Newton $N_\Delta(L)$ is defined as the convex hull of all polygons of the form $\mathcal{I} + (i, i - j)$ for which $c_{ij} \neq 0$ for $0 \leq i \leq n$ and $0 \leq j \leq m$ intersected with $\mathbb{R}^+ \times \mathbb{R}$.*

Proposition 6.2.12 *Some easy properties of the Δ-polygon of Newton:*

(a) $N_\tau(L)$ is an infinite subset of $\mathbb{R}^+ \times \mathbb{R}$

(b) The edge with slope 1 has infinite length.

(c) If $order(L) = n$, then the Δ-polygon of Newton can only have the slopes 0, $\frac{1}{n}$, $\frac{2}{n}$, ..., 1.

Analogously to the newton polynomial for the τ-polygon, one can define the newton polynomial for the Δ-polygon for slopes greater or equal than 0 and less or equal than 1. We will only need the newton polynomial for slope 0, which is defined in a slightly different way:

Definition 6.2.13 Let $L = \sum_{i=0}^{n} \sum_{j=0}^{m} c_{i,j} x^j \Delta^i$, let $N_\Delta(L)$ be the corresponding Δ-polygon of Newton. and let $\mu := \sup\{j - i \mid c_{i,j} \neq 0\}$. Then the characteristic polynomial or Newton polynomial at 0 $P_0(T)$ is defined by

$$P_0(T) := \sum_{j-i=\mu} c_{i,j} (-1)^i T^{\bar{i}} = \sum_{j-i=\mu} c_{i,j} \prod_{k=0}^{i-1} (-T - k)$$

Note that all $c_{i,j}$ with $j - i = \mu$ lie on the edge with slope 0.

Let's formulate the algorithm for computing the set $\overline{g}_\infty(L)$:

Algorithm 6.2.14 ((c,n,d)-Information) *by Mark van Hoeij*
INPUT: $L \in \mathbb{C}[x][\tau]$
OUTPUT: The set $\overline{g}_\infty(L)$

BEGIN

    $\overline{g}_\infty(L) = \emptyset$

    Construct the τ-polygon of Newton of L

    Compute the set of all integer slopes N of the polygon

    FOR EACH $n \in N$ DO

        Compute the Newton polynomial $P_n(T)$

        Find the set of all roots C of $P_n(T) = 0$

        FOR EACH (distinct) root $c \in C$ DO

            $\tilde{L} := L\text{⑤}(\tau - \frac{1}{cx^n})$

            Construct the Δ-Polygon of Newton of $\tilde{L}$

            Compute the Newton polynomial for slope 0 $P_0(T)$

            Find the set of all roots D modulo the integers of $P_0(T)$

            FOR EACH $d \in D$ DO $\overline{g}_\infty(L) = \overline{g}_\infty(L) \cup \{(c, n, d)\}$

    Return $\overline{g}_\infty(L)$

END

<div align="right">♦</div>

Let's turn to the finite singularities:

Definition 6.2.15 *Let the difference operator $L = \sum_{k=0}^{n} p_k(x) \cdot \tau^k \in \mathbb{C}[x, \tau]$ with $\gcd(p_0, ..., p_n) = 1$ be given. Then $q \in \mathbb{C}$ is called a* problem point *if q is a root of the polynomial $p_0(x)p_n(x-n)$ and $p \in \mathbb{C}/\mathbb{Z}$ is called a* finite singularity *if $p_0 p_n$ has a root in p.*

Note: If using the recurrence relation for solutions y of L we cannot determine $u(q)$ from $u(q-1), ..., u(q-n)$, or we cannot determine $u(q)$ from $u(q+1), ..., u(q+n)$, then the point $q \in \mathbb{C}$ is a problem point. The finite singularities in $\mathbb{C}/\mathbb{Z}$ are the problem points modulo $\mathbb{Z}$.

Example 6.2.16 *Suppose we have $L = (x+1)(x+2)(x+3)\tau^2 + (2x+1)\tau - (x^2 - 2)$. Then the set of all problem points is $\{-1, 0, 1, -\sqrt{2}, \sqrt{2}\}$ and the set of all finite singularities is $\{0, -\sqrt{2}, \sqrt{2}\}$.*

Before we can define the analogue to the (c, n, d)-information for finite singularities we have to extend Definition 5.2.2 about the valuation of a rational function at a point $p \in \mathbb{C}/\mathbb{Z}$.

Definition 6.2.17 *Let $r \in \mathbb{C}(x)^*$ and $p \in \mathbb{C}/\mathbb{Z}$, then the* valuation *of r at p is*

$$v_p(r) = \sum_{q \in p + \mathbb{Z}} v_q(r)$$

Example 6.2.18 *Let $r = \frac{x^5(x+1)^2(2x+3)(x^2+2)}{(x+2)(2x+1)^4}$ then have:*

- $v_0(r) = 5 + 2 + (-1) = 6$

- $v_{\frac{1}{2}}(r) = 1 + (-4) = -1$

- $v_{i\sqrt{2}}(r) = v_{-i\sqrt{2}}(r) = 1$

- $v_p(r) = 0$ *for all other $p \in \mathbb{C}/\mathbb{Z}$*

Definition 6.2.19 *Let $L = \tau - r$ with $r \in \mathbb{C}(x)$. For every $p \in \mathbb{C}/\mathbb{Z}$ we define the* valuation growth *of L at p as*

$$g_p(L) := v_p(r) \in \mathbb{Z}$$

The map $p \mapsto g_p(L)$ defines a function from $\mathbb{C}/\mathbb{Z}$ to $\mathbb{Z}$ which has finite support. Let $\mathcal{H}$ be the product of the additive group of all functions $\mathbb{C}/\mathbb{Z} \to \mathbb{Z}$ with finite support and the group $\mathcal{H}_\infty$ from Definition 6.2.1. Then we define

$$g(L) := \big(p \mapsto g_p(L), g_\infty(L)\big) \in \mathcal{H}$$

Hence, g is a map from the set of first order difference operators into $\mathcal{H}$.

Definition 6.2.20 *Let $\mathcal{H}_F \subseteq \mathcal{H}$ be the set of all $(f, (c, n, d + \mathbb{Z})) \in \mathcal{H}$ (where $f : \mathbb{C}/\mathbb{Z} \to \mathbb{Z}$ has finite support, $c, d \in \mathbb{C}$ and $n \in \mathbb{N}$) for which*

$$\sum_{p \in \mathbb{C}/\mathbb{Z}} f(p) = n \quad \text{and} \quad d + \sum_{p \in \mathbb{C}/\mathbb{Z}} f(p)p \equiv 0 \bmod \mathbb{Z} \tag{6.10}$$

The relations (6.10) are called Fuchs' *relations.*

Example 6.2.21 *Continuation of Example 6.2.2: We had $L = \tau - \frac{x^2+1}{2x-6}$ and computed $g_\infty(L) = (\frac{1}{2}, 1, \mathbb{Z})$. Furthermore, we have $g_i(L) = 1$, $g_{-i}(L) = 1$, $g_0(L) = -1$ and $g_p(L) = 0$ for all other $p \in \mathbb{C}/\mathbb{Z}$. Let's check the Fuchs' relations:*

$$\sum_{p \in \mathbb{C}/\mathbb{Z}} f(p) = 1 + 1 + (-1) = 1 \quad = \quad n$$

$$0 + i \cdot 1 + (-i) \cdot 1 + 0 \cdot (-1) = 0 \quad \equiv \quad 0 \bmod \mathbb{Z}$$

Before we can formulate the first main theorem, we have to take a look at the following groups (recall Proposition 3.2.9):

Proposition 6.2.22 *Let $L_H := \{\tau - r \mid r \in \mathbb{C}(x)^*\}$ be the set of all monic first order difference operators which have a nonzero (hypergeometric) solution and let $L_R := \{\tau - \frac{\tau(r)}{r} \mid r \in \mathbb{C}(x)^*\}$ be the set of all monic first order difference operators which have a nonzero rational solution (namely r), then:*

(a) With the operation $\circledS$ both, L_H and L_R, become multiplicative groups.

(b) For all $L \in L_R$ and $p \in \mathbb{C}/\mathbb{Z}$ we have: $g_p(L) = 0$

Theorem 6.2.23 *Let $g : L_H \to \mathcal{H}$ be the group homomorphism between L_H and $\mathcal{H}$ defined in Definition 6.2.19, then:*

(a) The kernel of g is L_R

(b) The image of g is $\mathcal{H}_F$

As a consequence, $g : L_H/L_R \to \mathcal{H}_F$ is an isomorphism.

Proof:

(a) Because of the definition of L_R it is clear that L_R is contained in the kernel of g. On the other hand, an element L can only be L_R if $c = 1$ and $f = 0$, i.e. if $g(L) = (0, (1, 0, \mathbb{Z}))$, which is the identity in $\mathcal{H}_F$. So the kernel of g is contained in L_R and hence equal to L_R.

(b) To show that the image is contained in $\mathcal{H}_F$, it suffices to verify that this is true for the "generators" $\tau - c$ and $\tau - (x - q)$ of the group L_H with $c \in \mathbb{C}^*, q \in \mathbb{C}$ (see Proposition 2.2.14):

- Case $\tau - c$: $g(\tau - c) = (0, (c, 0, \mathbb{Z})) \in \mathcal{H}_F$

- Case $\tau - (x - q)$: $g(\tau - (x - q)) = (f, (1, 1, \mathbb{Z})) \in \mathcal{H}_F$, where $f = 1$ for $q \in \mathbb{C}/\mathbb{Z}$ and 0 otherwise.

Conversely, let $h_F = (f, (c, n, d))$ be an arbitrary element of $\mathcal{H}_F$ with $f(p_i) = e_i \neq 0$ for a finite set of $p_1, ..., p_k \in \mathbb{C}$, and 0 otherwise. Let

$$L = \tau - c \cdot (x - p_1)^{e_1} \cdot ... \cdot (x - p_k)^{e_k}$$

then $L \in L_H$ and $g(L) = h_F$, which shows that g is also surjective.

■

Corollary 6.2.24 *We have the following $1 - 1$ correspondences for a solution y of $Ly = 0$:*

$$y \in \mathbb{H}/\mathbb{C}(x)^* \longleftrightarrow L \in L_H/L_R \longleftrightarrow g(L) \in \mathcal{H}_F$$

In other words this corollary says: If we have the $g(L)$, then we can construct the hypergeometric solution y modulo a rational function. This is the main idea of van Hoeij's algorithm. Thus, we have to look for a method for computing the $g(L)$, respectively the map $p \to g_p(L)$. We will do this in the following subsection.

6.2.1   Computing valuation growths

Looking back to the singularity at infinity, we would like to have something similar to $\bar{g}_\infty(L)$, in other words a set like

$$\{g_p(M) \mid \text{M is a first order right hand factor of } L\}$$

for all $p \in \mathbb{C}/\mathbb{Z}$. Unfortunately, we will see that we cannot always compute this set exactly, but only a superset. This superset will be denoted by $\bar{g}_p(L)$.

We will demonstrate the general idea on the following example:

Example 6.2.25 *Suppose we have the difference operator $L = \tau - x(x + 1)$ with the solution $y = (x - 1)! \cdot x! = \Gamma(x)\Gamma(x + 1)$. Because $order(L) = 1$, we already know in advance that $g_p(L) = 2$ for $p \in \mathbb{Z}$ and $g_p(L) = 0$ for all other p (by Definition 6.2.19). Let's compute this result with other methods:*

- Analytic way: By analytic methods, we know that the solution y has a pole of order 2 (valuation -2) for negative integers, a pole of order 1 in 0 (valuation -1) and no poles in the positive integers (valuation 0). So going through the integers from the left to the right, passing the problem points, the valuation increases exactly by 2. Furthermore, y has no other poles, so the valuation "increases" by 0 when going through $p \notin \mathbb{Z}$.
  The problem with this method is simply that determining pole orders of functions like $x!$ can hardly be done by computers!

- Algebraic way: Let's first extend the field of constants to $\mathbb{C}(\epsilon)$, where ϵ is transcendental over $\mathbb{C}$. Then we can define the "deformed" difference operator $L_\epsilon := \tau - (x + \epsilon)(x + \epsilon + 1) \in \mathbb{C}(\epsilon, x)[\tau]$ with solutions $\tilde{u}$. Now, 0 and -1 are no problem points any more, thus we can use the difference relation without problems:

  - Calculation from the left to the right, starting e.g. with $\tilde{u}(-2) := 1$ yields:

$$
\begin{aligned}
\tilde{u}(-1) &= (-2 + \epsilon)(-1 + \epsilon) \\
\tilde{u}(0) &= (-2 + \epsilon)(-1 + \epsilon)^2 \epsilon \\
\tilde{u}(1) &= (-2 + \epsilon)(-1 + \epsilon)^2 \epsilon^2 (1 + \epsilon)
\end{aligned}
$$

  Now the ϵ-valuation[1] of $\tilde{u}(1)$ (and also of the following $\tilde{u}(k)$) is 2.

  - Calculation from the right to the left, starting e.g. $\tilde{u}(2) := 1$ yields:

$$
\begin{aligned}
\tilde{u}(1) &= \frac{1}{(1 + \epsilon)(2 + \epsilon)} \\
\tilde{u}(0) &= \frac{1}{\epsilon (1 + \epsilon)^2 (1 + \epsilon)} \\
\tilde{u}(-1) &= \frac{1}{(-1 + \epsilon)\epsilon^2 (1 + \epsilon)^2 (2 + \epsilon)}
\end{aligned}
$$

  Now the ϵ-valuation of $\tilde{u}(-1)$ (and also of the following $\tilde{u}(k)$) is -2.

  - It is easy to verify that an analogous calculation starting at a point $p \notin \mathbb{Z}$ yields ϵ-valuations 0.

We will see that the ϵ-valuation 2 of the first calculations yields a lower bound for $g_p(L)$ (for $p \in \mathbb{Z}$) and the ϵ-valuation -2 of the second one, multiplied by -1, yields an upper bound for $g_p(L)$. Thus, in this case, $g_p(L) = 2$ for $p \in \mathbb{Z}$ (and $g_p(L) = 0$, otherwise), as it should be.

Let's turn to the general case:

Definition 6.2.26 *Let ϵ be a new indeterminate; ϵ is transcendental over $\mathbb{C}$. Define the action of τ on $\mathbb{C}(x, \epsilon)$ as $\tau(\epsilon) = \epsilon$ and, as usual, $\tau(x) = x + 1$. This turns $\mathbb{C}(x, \epsilon)$ into a difference field with $\mathbb{C}(\epsilon)$ as the field of constants. For $a \in \mathbb{C}(\epsilon)$ the ϵ-valuation is*

$$
v_\epsilon(a) = \sup\{m \in \mathbb{Z} \mid a \in \epsilon^m \mathbb{C}[[\epsilon]]\} \in \mathbb{Z} \cup \{\infty\}
$$

(which corresponds with Definition 5.2.2, taking $\mathbb{C}(\epsilon)$ as the field of rational functions in ϵ over $\mathbb{C}$).
Let $L \in \mathbb{C}(x)[\tau]$. Define $L_\epsilon \in \mathbb{C}(\epsilon, x)[\tau]$ as the operator one obtains from L by replacing x by $x + \epsilon$. We call L_ϵ the deformation *of L.*

---

[1] The power of ϵ contained in an expression - see Definition 6.2.26

For the next considerations we will use the following definitions/notations:

- $L = p_n \tau^n + ... + p_0 \tau^0 \in \mathbb{C}[x, \tau]$ with $p_n \neq 0$ and $p_0 \neq 0$

- $p \in \mathbb{C}/\mathbb{Z}$

- q_l (resp. q_r) is the smallest (resp. largest) problem point at p, so q_l (resp. q_r) is the smallest (resp. largest) root of $p_0(x)p_n(x - n)$ in p. If p is not a singularity (so then there are no problem points at p) then we take arbitrary elements $q_l, q_r \in p$.

- We will also use the following solution spaces for L (and also analogously for L_ϵ):

  - Denote by $V_p(L) := \{u : p \to \mathbb{C} \mid Lu = 0\}$ the set of all solutions of L which are defined on the line $p + \mathbb{Z}$ and analogously denote $V_p(L_\epsilon) := \{\tilde{u} : p \to \mathbb{C}(\epsilon) \mid L_\epsilon(\tilde{u}) = 0\}$
  - Denote by $V_{p,l}(L) := \{u : q_l - \mathbb{N} \to \mathbb{C} \mid Lu = 0\}$ the set of *left solutions*
  - Denote by $V_{p,r}(L) := \{u : q_r + \mathbb{N} \to \mathbb{C} \mid Lu = 0\}$ the set of *right solutions*

Definition 6.2.27 *Let $\tilde{u} \in V_p(L_\epsilon)$*

- *The* left valuation $v_{\epsilon,l}(\tilde{u})$ *is defined by*

$$v_{\epsilon,l}(\tilde{u}) = \min\{v_\epsilon(\tilde{u}(m)) \mid m \in q_l - 1 - \mathbb{N}\}$$

- *The* right valuation $v_{\epsilon,r}(\tilde{u})$ *is defined by*

$$v_{\epsilon,r}(\tilde{u}) = \min\{v_\epsilon(\tilde{u}(m)) \mid m \in q_r + 1 + \mathbb{N}\}$$

- *The* valuation growth $g_{p,\epsilon}(\tilde{u})$ *is defined by*

$$g_{p,\epsilon}(\tilde{u}) = v_{\epsilon,r}(\tilde{u}) - v_{\epsilon,l}(\tilde{u}) \in \mathbb{Z}$$

- *The* set of valuation growths *of L at p is defined by*

$$\bar{g}_p(L) = \{g_{p,\epsilon}(\tilde{u}) \mid \tilde{u} \in V_p(L_\epsilon), \tilde{u} \neq 0\} \subset \mathbb{Z}$$

- *If $v_{\epsilon,l}(\tilde{u}) \geq 0$, then the* left projection $\rho_l(\tilde{u}) \in V_{p,l}(L)$ *is defined by substituting $\epsilon = 0$ in $\tilde{u}$. Similarly if $v_{\epsilon,r}(\tilde{u}) \geq 0$, then the* right projection $\rho_r(\tilde{u}) \in V_{p,r}(L)$ *is defined.*

First of all, we have to show that the defined $\bar{g}_p(L)$ really has the property of being a superset of $\{g_p(M) \mid$ M is a first order right hand factor of $L\}$. This is an immediate consequence of the following lemma:

Lemma 6.2.28 *Let $L, N, M \in \mathbb{C}(x)[\tau]$ be normal*

(a) *If $order(L) = 1$, then $\bar{g}_p(L)$ contains only one element, $\bar{g}_p(L) = \{g_p(L)\}$*

(b) *If $L = NM$, then $\bar{g}_p(M) \subset \bar{g}_p(L)$.*

Proof: (a) Follows directly from the definition of $g_p(L)$. For (b), observe that M is a right hand factor of L, so M_ϵ is a right hand factor of L_ϵ and therefore $V_p(M_\epsilon) \subset V_p(L_\epsilon)$ by Theorem 3.1.6 (d) and hence (b) follows.

∎

Remark 6.2.29 *It is easy to see that Lemma 6.2.28 holds also for $p = \infty$. Without poof we mention that $\bar{g}_p(N) \subset \bar{g}_p(L)$ as well, and that $\bar{g}_\infty(L) = \bar{g}_\infty(M) \cup \bar{g}_\infty(N)$. However, if p is finite then examples show that in general the set $\bar{g}_p(L)$ is not determined by $\bar{g}_p(M)$ and $\bar{g}_p(N)$.*

We can finally generalize the computation in our Example 6.2.25 with help from

Lemma 6.2.30 *Let $\tilde{u} \in V_p(L_\epsilon)$*

(a) $v_{\epsilon,l}(\tilde{u}) = \min\{v_\epsilon(\tilde{u}(m)) \mid m \in \{q_l - 1, q_l - 2, ..., q_l - n\}\}$

(b) $v_{\epsilon,r}(\tilde{u}) = \min\{v_\epsilon(\tilde{u}(m)) \mid m \in \{q_r + 1, q_r + 2, ..., q_r + n\}\}$

Proof: Note that for all $m \in q_l - 1 - \mathbb{N}$ (resp. $m \in q_r + 1 + \mathbb{N}$) the leading and trailing coefficients of L_ϵ have ϵ-valuation 0 and all other coefficients of L_ϵ have ϵ-valuation ≥ 0. Since the $\tilde{u}(m)$ for $m \in q_l - 1 - \mathbb{N}$ (resp. $m \in q_r + 1 + \mathbb{N}$) are determined by L_ϵ and the $\tilde{u}(m)$ for $m \in \{q_l - 1, q_l - 2, ..., q_l - n\}$ (resp. $m \in \{q_r + 1, q_r + 2, ..., q_r + n\}$), the statements follows.

∎

Algorithm 6.2.31 (Valuation Computation) *by Mark van Hoeij*
INPUT:

- $L = \sum_{k=0}^{n} p_k \cdot \tau^k$ *where $p_k \in \mathbb{C}[x]$*

- $p_n \neq 0$ *and $p_0 \neq 0$*

- $p \in \mathbb{C}/\mathbb{Z}$

OUTPUT: The set $\bar{g}_p(L) \subset \mathbb{Z}$

BEGIN

Determine q_l (resp. q_r) as the smallest (resp. largest) root of the polynomial $p_0(x)p_n(x - n)$ in p.

FOR $i, j \in \{1, 2, ..., n\}$ DO $\tilde{u}_i(q_l - j) = \tilde{v}_i(q_r + j) := \delta_{ij}$

FOR $i \in \{1, 2, ..., n\}$ DO

> FOR $m \in \{q_l, q_l + 1, ..., q_r, ..., q_r + n\}$ DO Compute $\tilde{u}_i(m)$ with the deformation L_ϵ (forward direction)
>
> FOR $m \in \{q_r, q_r - 1, ..., q_l, ..., q_l - n\}$ DO Compute $\tilde{v}_i(m)$ with the deformation L_ϵ (backward direction)

$g_{p,r}(L) := \min\{v_{\epsilon,r}(\tilde{u}_i) \mid 1 \le i \le n\}$

$g_{p,l}(L) := -\min\{v_{\epsilon,l}(\tilde{v}_i) \mid 1 \le i \le n\}$

$\bar{g}_p(L) := \{e \in \mathbb{Z} \mid g_{p,r}(L) \le e \le g_{p,l}(L)\}$

Return $\bar{g}_p(L)$

END

♦

Theorem 6.2.32 *With the notations of Algorithm 6.2.31:*
$g_{p,r}(L) = \min(\bar{g}_p(L))$ *and* $g_{p,l}(L) = \max(\bar{g}_p(L))$ *which proves the correctness of the algorithm.*

Proof: With $\tilde{u}_i$ and $\tilde{v}_i$ defined in the Algorithm, note that $v_{\epsilon,l}(\tilde{u}_i) = v_{\epsilon,r}(\tilde{v}_i) = 0$ and that $v_{\epsilon,r}(\tilde{u}_i)$ and $v_{\epsilon,l}(\tilde{v}_i)$ are computed in the Algorithm. Therefore (see Definition 6.2.27)

$$
\begin{aligned}
g_{p,r}(L) &= \min\{v_{\epsilon,r}(\tilde{u}_i) - 0 \mid 1 \le i \le n\} = \min\{g_{p,\epsilon}(\tilde{u}_i) \mid 1 \le i \le n\} \\
g_{p,l}(L) &= -\min\{v_{\epsilon,l}(\tilde{v}_i) \mid 1 \le i \le n\} = \\
&= \max\{0 - v_{\epsilon,l}(\tilde{v}_i) \mid 1 \le i \le n\} = \max\{g_{p,\epsilon}(\tilde{v}_i) \mid 1 \le i \le n\}
\end{aligned}
$$

Because every non-zero $\tilde{u} \in V_p(L_\epsilon)$ can be written as

$$
\tilde{u} = \epsilon^{v_{\epsilon,l}(\bar{u})} \sum_i a_i \tilde{u}_i = \epsilon^{v_{\epsilon,r}(\bar{u})} \sum_i b_i \tilde{v}_i
$$

for some $a_i, b_i \in \mathbb{C}(\epsilon)$ with $v_\epsilon(a_i) \ge 0$ and $v_\epsilon(b_i) \ge 0$, the statement follows.

∎

Remark 6.2.33 *Remarks on Algorithm 6.2.31:*

- *The computation of the $\tilde{u}_i(m)$ and $\tilde{v}_i(m)$ is the most time consuming part in the algorithm. It should be done modulo a suitable power of ϵ to reduce intermediate expression swell. A possible power of ϵ would be $v_p(p_0 \cdot p_n) + 1$. Moreover, we can combine this with modular arithmetic to eliminate expression swell.*

- *Note that computing the $\bar{g}_p(L)$ is similar to finding the bounds $B_i^l(p)$ and $B_i^r(p)$ of section 5.2. Additionally, we know the following (see theorem 6.2.23): L has only rational solutions if and only if $\bar{g}_p(L) = \{0\}$ for all $p \in \mathbb{C}/\mathbb{Z}$. In this case we also have $g_\infty(L) = \{(1, 0, \mathbb{Z})\}$.*

At this point, at the latest, we know that $\bar{g}_p(L) = \{0\}$ if $p \in \mathbb{C}/\mathbb{Z}$ is not singular - nevertheless, this is not a necessary condition, which leads to the following definition:

Definition 6.2.34 *Let $p \in \mathbb{C}/\mathbb{Z}$ a (finite) singularity. If $\bar{g}_p(L) = \{0\}$ then p is called an* apparent *singularity. If $\bar{g}_p(L)$ has only 1 element then p is called an* semi-apparent *singularity. If $\bar{g}_p(L)$ has more than 1 element then p is called an* essential *singularity.*

Definition 6.2.35 *Let $\tilde{u}_i, \tilde{v}_i, g_{p,r}(L)$ and $g_{p,l}(L)$ be like in Algorithm 6.2.31. Furthermore define the basis $u_1, ..., u_n$ for $V_{p,l}(L)$ by $u_i = \rho_l(\tilde{u}_i)$ and the basis $v_1, ..., v_n$ for $V_{p,r}(L)$ by $v_i = \rho_r(\tilde{v}_i)$. Because we can reconstruct $\tilde{u}_i$ from u_i and $\tilde{v}_i$ from v_i (as $\tilde{u}_i(q_l - j) = u_i(q_l - j)$ and $\tilde{v}_i(q_r + j) = v_i(q_r + j)$ for $j \in \{1, ..., n\}$), we can define the following $\mathbb{C}$-linear maps $E_{p,r,L}$ and $E_{p,l,L}$ by*

$$E_{p,r,L} : V_{p,l}(L) \to V_{p,r}(L) \;\; \text{with} \;\; E_{p,r,L}(u_i) = \rho_r(\tilde{u}_i/\epsilon^{g_{p,r}(L)})$$
$$E_{p,l,L} : V_{p,r}(L) \to V_{p,l}(L) \;\; \text{with} \;\; E_{p,l,L}(v_i) = \rho_l(\tilde{v}_i \cdot \epsilon^{g_{p,l}(L)})$$

It is easy to see that the maps $E_{p,r,L}$ and $E_{p,l,L}$ are $1 - 1$ and each other's inverses if p is not a singularity - one can even identify $V_{p,r}(L)$ and $V_{p,l}(L)$ with $V_p(L)$! Nevertheless, we can show more:

Theorem 6.2.36 *If $g_{p,r}(L) = g_{p,l}(L)$ (i.e. p is a semi-apparent singularity), then $E_{p,r,L}$ and $E_{p,l,L}$ are each other's inverses. If $g_{p,r}(L) \neq g_{p,l}(L)$ (i.e. p is an essential singularity), then $E_{p,r,L} \circ E_{p,l,L} = E_{p,l,L} \circ E_{p,r,L} = 0$, in other words*
$$\text{Im}(E_{p,r,L}) \subset \text{Ker}(E_{p,l,L}) \;\; \text{and} \;\; \text{Im}(E_{p,l,L}) \subset \text{Ker}(E_{p,r,L}).$$

Proof: $E_{p,r,L}(E_{p,l,L})(v_i) = E_{p,r,L}(\rho_l(\tilde{v}_i \cdot \epsilon^{g_{p,l}(L)})) = \rho_r(\tilde{v}_i \cdot \epsilon^{g_{p,l}(L)}/\epsilon^{g_{p,r}(L)}) = \rho_r(\tilde{v}_i \cdot \epsilon^{g_{p,l}(L) - g_{p,r}(L)})$. *Let $s := g_{p,l}(L) - g_{p,r}(L) \geq 0$ as $g_{p,r}(L) \leq g_{p,l}(L)$, then we can distinguish two cases:*

- *Case 1: $s = 0 \Rightarrow E_{p,r,L}(E_{p,l,L})(v_i) = \rho_r(\tilde{v}_i) = \tilde{v}_i$*

- *Case 2: $s > 0 \Rightarrow E_{p,r,L}(E_{p,l,L})(v_i) = \rho_r(\tilde{v}_i \cdot \epsilon^s) = 0$*

The linear map $E_{p,l,L} \circ E_{p,r,L}$ can be computed in the same way.

∎

Corollary 6.2.37 *Let the matrices A and B be defined by*

$$a_{ij} := p_r\left(\tilde{u}_i(q_r + j) \cdot \frac{1}{\epsilon^{g_{p,r}(L)}}\right) \quad \text{and} \quad b_{ij} := p_l\left(\tilde{v}_i(q_l - j) \cdot \epsilon^{g_{p,l}(L)}\right)$$

then
$$B = A^{-1} \Leftrightarrow g_{p,r}(L) = g_{p,l}(L)$$

This corollary can be used to speed up Algorithm 6.2.31: Suppose we computed only the $\tilde{u}_i(m)$ and the $g_{p,r}(L)$ and noticed that the matrix A defined in the corollary is regular. Then we know that p is (semi-)apparent and that $\bar{g}_p(L) = \{g_{p,r}(L)\}$. Thus, we need not compute the $\tilde{v}_i(m)$.

Before we can formulate van Hoeij's algorithm completely we have to consider one remaining problem: "How can we compute an exact solution y when we know a solution $\tilde{y}$ modulo $\mathbb{C}(x)$?"

Lemma 6.2.38 *Let $L \in \mathbb{C}(x)[\tau]$ and let $\tilde{y}$ with $(\tau - r)\tilde{y} = 0$ be a hypergeometric solution of L modulo $\mathbb{C}(x)$, i.e. there exists a rational function R such that $Ly = L(\tilde{y} \cdot R) = 0$. Then R can be computed as the (complete) rational solution of $L\mathⓢ(\tau - \frac{1}{r})$.*

Proof: $y = \tilde{y} \cdot R \Leftrightarrow R = y \cdot \frac{1}{\tilde{y}}$. Now the statement follows from Definition 3.2.5 together with Proposition 3.2.9.

■

Algorithm 6.2.39 (HHyper) *by Mark van Hoeij*
INPUT: $Ly = 0$ with

- *$L = \sum_{k=0}^{n} p_k \cdot \tau^k$ where $p_k \in \mathbb{C}[x]$*

- *$p_0, p_n \neq 0$*

OUTPUT: A basis $\mathcal{B}$ for the space of hypergeometric solutions over $\mathbb{C}$ of $Ly = 0$. More exactly: A set $\mathcal{B} = \{y_1 r_1, ..., y_l r_l\}$ such that $y_i r_i$ are hypergeometric solutions of $Ly = 0$ where y_i is purely hypergeometric and r_i is a linear combination of rational functions.

BEGIN

    $\mathcal{B} = \emptyset$

    Compute all monic factors of the polynomial $p_0(x)p_n(x)$

    Throw out all factors which are integer-shifts from another factor, then the set $S = \{p_1, ..., p_m\}$ of all finite singularities are the roots (modulo the integers) of the remaining factors.

    Compute $\bar{g}_\infty(L)$ with Algorithm 6.2.14

    IF $\bar{g}_\infty(L) = \{\}$ THEN Return $\mathcal{B}$

    FOR $p_i \in S$ DO compute $\bar{g}_{p_i}(L)$ with Algorithm 6.2.31

    FOR ALL tupels $(e_1, ..., e_m, (c, n, d))$ for which $e_i \in \bar{g}_{p_i}(L)$ and $(c, n, d) \in \bar{g}_\infty(L)$ and which satisfy the Fuchs' relations DO

        $r := c \cdot \prod_{i=1}^{m} (x - p_i)^{e_i}$

Compute y such that $\frac{\tau(y)}{y} = r$ (the solution of $(\tau - r)y = 0$)

Compute the general rational solution R of $L\text{Ⓢ}(\tau - \frac{1}{r})(R) = 0$ (if exists)

IF R could be found THEN $\mathcal{B} = \mathcal{B} \cup \{y \cdot R\}$

END

$\blacklozenge$

Remark 6.2.40 *Remarks on Algorithm 6.2.39*

- *Algorithm 6.2.39 can be varied in the following manner: Instead of computing all hypergeometric solutions over $\mathbb{C}$, one can also compute all hypergeometric solutions over a given subfield $C \subset \mathbb{C}$. In this case, we merely have to factor and to find roots over C, and also the elements of $\bar{g}_\infty(L)$ have to be in C. For more details see [vHo99].*

- *A MATHEMATICA-version of Algorithm 6.2.39 over $\mathbb{Q}$ and $\mathbb{C}$ is available at the RISC-Homepage. Note that this implementation does not return the set $\mathcal{B} = \{y_1 r_1, ..., y_l r_l\}$ but the set $\{\text{ToHG}[h_1]r_1, ..., \text{ToHG}[h_l]r_l\}$ where $h_i := \frac{\tau(y_i)}{y_i}$. The "pretty" solution can be achieved by a conversion function.*

6.3   Examples and Comparison

Some easy examples computed with Hyper (Algorithm 6.1.5) and HHyper (Algorithm 6.2.39):

Example 6.3.1 *Given $Ly = 0$, with $L = (x-1)\tau^2 - (x^2 + 3x - 2)\tau + 2x(x+1)$, thus we have*

$$(x-1)y(x+2) - (x^2 + 3x - 2)y(x+1) + 2x(x+1)y(x) = 0$$

with the general solution
$$y(x) = c_1 2^x + c_2 x!$$

Hyper yields $\{2, x+1\}$, HHyper yields $\{c[1]\text{ToHG}[2], c[2]\text{ToHG}[x+1]\}$ or after a conversion $\{c[1]2^x, c[2]x!\}$.

Example 6.3.1 *Given $Ly = 0$, with $L = \tau^2 - (2x+1)\tau + (x^2 - 2)$, thus we have*
$$y(x+2) - (2x+1)y(x+1) + (x^2 - 2)y(x) = 0$$

with the general solution

$$y(x) = c_1(-\sqrt{2})^{\overline{x}} + c_2(\sqrt{2})^{\overline{x}} = c_1\Gamma(x - \sqrt{2}) + c_2\Gamma(x + \sqrt{2})$$

Hyper yields $\{-\sqrt{2}+x, \sqrt{2}+x\}$,
HHyper yields $\{c[1]\text{ToHG}[-\sqrt{2}+x], c[2]\text{ToHG}[\sqrt{2}+x]\}$ or after the conversion $\{c[1]\text{RisingFactorial}[-\sqrt{2}], c[2]\text{RisingFactorial}[\sqrt{2}]\}$.

Example 6.3.2 *Given $Ly = 0$, with $L = (x+2)(2x+3)\tau^2 - 2(4x^2 + 13x + 9)\tau + 3x(2x+5)$, thus we have*

$$(x+2)(2x+3)y(x+2) - 2(4x^2 + 13x + 9)y(x+1) + 3x(2x+5)y(x) = 0$$

with the general solution

$$y(x) = c_1 3^x + c_2 \frac{1}{x}$$

Hyper yields $\{3, \frac{x}{x+1}\}$, HHyper yields $\{\frac{c[1]}{x}, c[2]\text{ToHG}[3]\}$ of after a conversion $\{\frac{c[1]}{x}, c[2]3^x\}$.

Example 6.3.3 *Given $Ly = 0$, with $L = 3\tau^2 - x\tau + x - 1$, thus we have*

$$3y(x+2) - xy(x+1) + (x-1)y(x) = 0$$

with the only hypergeometric (resp. polynomial) solution

$$y(x) = c_1(x^2 - 11x + 27)$$

Hyper yields $\frac{17 - 9x + x^2}{27 - 11x + x^2}$, HHyper yields $\{c[1](x^2 - 11x + 27)\}$.

For most of the "common" examples Hyper works perfectly well. In many cases it is faster than HHyper, as computing the $\bar{g}_p(L)$ in HHyper takes some time. However, Hyper becomes problems with examples of the following kind:

Example 6.3.4 *Given $Ly = 0$, with $L = p_3(x)\tau^3 + p_2(x)\tau^2 + p_1(x)\tau + p_0(x)$ where*

$$
\begin{aligned}
p_3(x) &= (3x+8)(3x+10)(x+3)(x+2)(x+1) \\
p_2(x) &= -3(3x+5)(3x+7)(x+2)(x+1) \\
p_1(x) &= (3x+2)(3x+4)(x+1) \\
p_0(x) &= -(3x-1)(3x+1)
\end{aligned}
$$

with the general solution

$$y(x) = \frac{c_1 + (x-3)c_2 + (x^2 - 3x + 7)c_3}{x!(3x-1)(3x+1)}$$

Hyper yields

$$\{\frac{1}{1+x}, \frac{3x-1}{x(3x+2)}, \frac{3x-1}{(x+1)(3x+2)}, \frac{3x+1}{x(3x+4)}, \frac{3x+1}{x(3x+4)},$$

$$\frac{(3x-1)(3x+1)}{(x-1)(3x+2)(3x+4)}, \frac{(3x-1)(3x+1)}{x(3x+2)(3x+4)},$$

$$\frac{(3x-1)(3x+1)}{(x+1)(3x+2)(3x+4)}, \frac{(x+1)(3x-1)(3x+1)}{x^2(3x+2)(3x+4)}\}$$

Although there seems to be a contradiction to Theorem 3.1.2 (we expected only three solutions), each of these nine functions yields a solution of $Ly = 0$, but they are (of course) not linearly independent! This shows the first (and minor) problem of Hyper. Especially, if L has more (linearly independent) rational solutions, Hyper will come up with some kind of "superset of linearly dependent solutions". Nevertheless, this is a minor problem, because we could modify the algorithm to test for linear dependencies. HHyper has no problems with the example and yields (after conversion) $\{\frac{c[1]+(x-3)c[2]+(x^2-3x+7)c[3]}{x!(3x-1)(3x+1)}\}$.

Taking a closer look at the last example one can even see the real problem of Hyper: The number of cases to work out, which grow exponentially with the degree of the leading and trailing coefficient. Let's illustrate this growth with the next example:

Example 6.3.5 *Given $Ly = 0$, with $L = \tau - \prod_{i=1}^{k}(x+k)$ and $k \in \mathbb{N}$, thus we have*

$$y(x+1) - \prod_{i=1}^{k}(x+i)y(x) = 0$$

with the general solution

$$y(x) = \prod_{i=0}^{k-1}(x+i)! = x!(x+1)!...(x+k-1)!$$

The following table shows the time needed by Hyper (resp. HHyper) to compute the solution:

k	10	11	12	13	14	15	16	17	18
Hyper	0.33	0.58	0.88	1.71	3.29	6.65	13.19	26.75	55.14
HHyper	0.77	0.82	0.93	1.05	1.26	1.37	1.60	1.86	1.92

One easily recognizes the exponential growth of the computing time for Hyper, which is due to the number of cases (2^k which is the number of monic factors of the trailing coefficient) to work out. One the other hand, the computation of $\bar{g}_p(L)$ by HHyper yields $\bar{g}_0(L) = k$, that's why HHyper only has to work out one single case!

Finally, we mention the following example from [vHo99] which was a motivation for Mark van Hoeij to start working on his algorithm.

Example 6.3.6 *Let* $L = p_3(x)\tau^3 + p_2(x)\tau^2 + p_1(x)\tau + p_0(x)$ *with*

$$
\begin{aligned}
p_3(x) &= -4(140x^3 + 731x^2 + 1232x + 678)(2x + 7)^2(x + 3)^2 \\
p_2(x) &= (14560x^7 + 237304x^6 + 1637876x^5 + 6200310x^4 + 13887720x^3 + \\
&\quad +18380306x^2 + 13291032x + 4046652) \\
p_1(x) &= -(x + 2)(23660x^6 + 302879x^5 + 1581604x^4 + 4314577x^3 + \\
&\quad +6487290x^2 + 5099454x + 1638144) \\
p_0(x) &= 18(2x + 3)(x + 2)(141x^3 + 1151x^2 + 3114x + 2781)(x + 1)^2
\end{aligned}
$$

which has no hypergeometric solution.

Obviously, Hyper has to try all $2^3 \cdot 3 \cdot 3 \cdot 2 \cdot 2 \cdot 2^3 \cdot 3 = 6912$ possible combinations of factors of $p_0(x)$ and $p_3(x)$. Moreover, most of these cases involve algebraic numbers (resp. computation in the splitting field of $p_0 p_3$), which makes the computation slow (taking several weeks).

Of course, HHyper has to compute in the splitting field, too, but: We have $S := \{0, \frac{1}{2}, \alpha_1, \alpha_2, \alpha_3\}$ as the set of finite singularities, where $\alpha_i \in \mathbb{C}$ are the roots of $141x^3 + 1151x^2 + 3114x + 2781$. As Hyper computes the following $\bar{g}_p(L)$:

$$
\begin{aligned}
\bar{g}_{\alpha_1}(L) = \bar{g}_{\alpha_2}(L) = \bar{g}_{\alpha_3}(L) &= \{0\} \\
\bar{g}_0(L) &= \{-2, -1, 0, 1, 2\} \\
\bar{g}_{\frac{1}{2}}(L) &= \{-2, -1, 0, 1\} \\
\bar{g}_{\infty}(L) &= \{(\frac{1}{4}, 0, -1), (\frac{9}{4}, 0, -1), (4, 0, -\frac{5}{2})\}
\end{aligned}
$$

there are $5 \cdot 4 = 20$ (without the $\bar{g}_{\infty}(L)$-information) and only 6 (with the $\bar{g}_{\infty}(L)$-information) cases remaining. Furthermore, we need not to compute in a splitting field any more, because all α_i are apparent singularities. All in all, HHyper can prove that $Ly = 0$ has no hypergeometric solution in less than three minutes.

Looking at all examples we come to the conclusion that for examples where the leading and trailing coefficients have small degrees, Hyper works perfectly well. As soon as the degrees grow (and so does the number of cases to check) HHyper is the more efficient algorithm. The main problem however can still occur - exponentially large algebraic extensions. In [vHo99] Mark van Hoeij shows how to avoid this for $order(L) \leq 3$ and promises to do the higher order case later.

Besides, someone is maybe able to improve HHyper with some of the following ideas:

- Finding a simple (fast) method/criterion for determining all apparent singularities (without computing the $\bar{g}_p(L)$). This would for example help to find the solution of Example 6.3.6 in a few seconds.

- Finding a simple (fast) method/criterion for determining all semi-apparent singularities (without computing the $\bar{g}_p(L)$).

- Reducing the number of cases (down to $order(L)$) by finding smaller sub-sets of $\{g_p(M) \mid$ M is a first order right hand factor of $L\}$.

6.4 Inhomogeneous Equations

In this section we will present some methods to compute hypergeometric solutions y of inhomogeneous equations $Ly = f$. As we know from Theorem 3.1.2 it is sufficient to compute one partial solution to get (together with the solution space $V(L)$ of the homogeneous equation) the complete solution of the inhomogeneous one.

Before we start out with the algorithms we note that for polynomial and rational f we can use the algorithms in the previous chapters 4 and 5 in order to compute a partial solution. Therefore it is sufficient to deal with "really hypergeometric" right hand sides (recall Lemma 2.3.5).

6.4.1 Gosper's Algorithm

The Original Approach

We will start with the most famous inhomogeneous equation (Gosper's equation)

$$\Delta y = f \Leftrightarrow \tau(y) - y = f \Leftrightarrow y(x+1) - y(x) = f(x) \text{ with } f \in \mathbb{H} \qquad (6.11)$$

Although this is the simplest possible of all inhomogeneous equations (it has order 1 and both coefficients are 1), it already shows that there exists even first order inhomogeneous difference equations which have no hypergeometric solution (unlike in the homogeneous case).

Let's quickly revisit Gosper, who developed in [Gos78] the algorithm for finding the solution of (6.11), if it exists. Down the next lines observe that Hyper is pretty similar to Gosper's algorithm, as both make use of the Gosper-form (Lemma 6.1.2) and both reduce the problem of finding a hypergeometric solution to a problem of finding a polynomial solution.
Let $r(x) = \frac{f(x+1)}{f(x)}$, then we get from (6.11)

$$y(x+1) - y(x) = f(x) \Leftrightarrow \frac{y(x)}{f(x)} = \frac{1}{\frac{y(x+1)}{y(x)} - 1}$$

It follows that $\frac{y(x)}{f(x)}$ is a rational function, thus let $\frac{y(x)}{f(x)} = R(x)$, then

$$y(x+1) - y(x) = f(x) \Leftrightarrow \frac{y(x+1)}{f(x)} - \frac{y(x)}{f(x)} = 1 \Leftrightarrow r(x) \cdot R(x+1) - R(x) = 1$$

Hence, we have to find a rational function $R(x)$ satisfying

$$r(x) \cdot R(x+1) - R(x) = 1 \qquad (6.12)$$

Because of Lemma 6.1.2 we can write $r(x) = \frac{A(x)}{B(x)} \cdot \frac{C(x+1)}{C(x)}$. Moreover we set $R(x) := \frac{f(x)}{g(x)}$ with $\gcd(f(x), g(x)) = 1$, then we get from (6.12)

$$\frac{A(x)}{B(x)} \cdot \frac{C(x+1)}{C(x)} \cdot \frac{f(x+1)}{g(x+1)} - \frac{f(x)}{g(x)} = 1 \Leftrightarrow$$
$$\Leftrightarrow \frac{A(x)}{B(x)} \cdot \frac{C(x+1)}{C(x)} = \frac{f(x) + g(x)}{f(x+1)} \cdot \frac{g(x+1)}{g(x)}$$

Now Lemma 6.1.3 yields $g \mid C$, thus $R(x) = \frac{q(x)}{C(x)}$ with a unknown polynomial $q(x)$ and we get

$$\frac{A(x)}{B(x)} \cdot \frac{C(x+1)}{C(x)} = \frac{q(x) + C(x)}{q(x+1)} \cdot \frac{C(x+1)}{C(x)} \Leftrightarrow$$
$$\Leftrightarrow A(x) \cdot q(x+1) = (q(x) + C(x)) \cdot B(x)$$

Because $\gcd(A(x), B(x)) = 1$ (Lemma 6.1.2) it follows that $B(x) \mid q(x+1)$ and therefore we can set $q(x) := B(x-1) \cdot p(x)$ and search for the polynomial solution $p(x)$ satisfying

$$A(x) \cdot p(x+1) - B(x-1) \cdot p(x) = C(x) \qquad (6.13)$$

Let's sum up:

Algorithm 6.4.1 (Gosper) *by R. W. Gosper, Jr.*
INPUT: $y(x+1) - y(x) = f$ with $f \in \mathbb{H}$
OUTPUT: The hypergeometric (partial) solution y of $y(x+1) - y(x) = f$ (if it exists)

BEGIN

$\quad$ $r(x) := \frac{f(x+1)}{f(x)}$

$\quad$ Compute $A(x)$, $B(x)$, $C(x)$ such that $r(x) = \frac{A(x)}{B(x)} \cdot \frac{C(x+1)}{C(x)}$

$\quad$ Find the polynomial solution $p(x)$ of (6.13) (if it exists)

$\quad$ IF $p(x)$ could be found THEN

$\qquad$ $R(x) := \frac{B(x-1) \cdot p(x)}{C(x)}$

$\qquad$ $y(x) := R(x) \cdot f(x)$

$\qquad$ Return $y(x)$

$\quad$ ELSE Return $\emptyset$

END

♦

Remark 6.4.2 *Remarks on Algorithm 6.4.1:*

- *The general solution of Gosper's equations $y(x + 1) - y(x) = f(x)$ is given by the result of Algorithm 6.4.1 plus a constant (which is the general solution of $y(x + 1) - y(x) = 0$).*

- *Gosper's algorithm is the key algorithm for the following problem: Given a hypergeometric function (resp. sequence) $f(k)$ find a closed form solution for $\sum_k f(k)$. If we can find the solution $y(k)$ of the difference equation $y(k + 1) - y(k) = f(k)$, then we simply get $\sum_k f(k) = y(M + 1) - y(m)$ where M and m are the summation bounds of the $\sum_k f(k)$.*

The New Approach

We immediately start with equation (6.12) - there is no difference at the beginning: Search for a rational function $R(x)$ satisfying

$$r(x) \cdot R(x + 1) - R(x) = 1$$

where $r(x) = \frac{f(x+1)}{f(x)}$ is a rational function.

At this point we could use Abramov's or van Hoeij's algorithm in order to look for a possible denominator of $R(x)$ and then search for a polynomial solution. We will not do that "directly", but it will turn out that our "new" approach ends up with an algorithm which is very similar to Abramov's one.

We now set $R(x) = \frac{u(x)}{v(x)}$ with coprime polynomials u and v and $r(x) = \frac{a(x)}{b(x)}$ with coprime polynomials a and b (without loss of generality we assume that v and b are monic), thus we have:

$$\frac{a(x)}{b(x)} \cdot \frac{u(x + 1)}{v(x + 1)} - \frac{u(x)}{v(x)} = 1$$

and after multiplying up denominators

$$a(x) \cdot u(x + 1) \cdot v(x) - b(x) \cdot u(x) \cdot v(x + 1) = b(x) \cdot v(x) \cdot v(x + 1) \qquad (6.14)$$

In order to solve (6.14) we try to find a suitable denominator polynomial v (afterwards u can be computed as a polynomial solution of (6.14) - as usual). Let's define the polynomials

$$v_0(x) := \frac{v(x)}{\gcd(v(x), v(x + 1))} \quad \text{and} \quad v_1(x) := \frac{v(x + 1)}{\gcd(v(x), v(x + 1))}$$

and divide (6.14) by $\gcd(v(x), v(x + 1))$, then

$$a(x) \cdot u(x+1) \cdot v_0(x) - b(x) \cdot u(x) \cdot v_1(x+1) = b(x) \cdot v_0(x) \cdot v_1(x+1) \cdot \gcd(v(x), v(x+1))$$

¿From this equation we immediately get that $v_0(x) \mid b(x)$ and that $v_1(x) \mid a(x)$. Using the gff-concept and Lemma 2.2.9 we get that (let $v := \mathrm{gff}(p_1, ..., p_m)$)

$$v_0 = \frac{\mathrm{gff}(p_1, p_2, ..., p_m)}{\mathrm{gff}(p_2, ..., p_m)} = p_1 \cdot \tau^{-1}(p_2) \cdot ... \cdot \tau^{-m+1}(p_m) \quad \mid \quad b \quad (6.15)$$

$$v_1 = \frac{\mathrm{gff}(\tau(p_1), \tau(p_2), ..., \tau(p_m))}{\mathrm{gff}(p_2, ..., p_m)} = \tau(p_1) \cdot \tau(p_2) \cdot ... \cdot \tau(p_m) \quad \mid \quad a \quad (6.16)$$

Consequently,

- **Straightforward conclusion**

$$p_k \mid \gcd\left(\tau^{-1}(a), \tau^{k-1}(b)\right) \quad \forall k \in \{1, ..., m\}$$

Thus, we could take

$$v = \mathrm{gff}(p_1, ..., p_N) \quad \text{with } p_k = \gcd\left(\tau^{-1}(a), \tau^{k-1}(b)\right)$$

where $N := \mathrm{dis}(a, b) = \max\{k \in \mathbb{N} \mid \deg \gcd(a, \tau^k(b)) > 1\}$. If N is not defined then we know that $v \equiv 1$, just like in Abramov's algorithm.

- **Refined conclusions**

$$p_1 \mid \gcd\left(\tau^{-1}(a), b\right)$$

and we take $p_1 := \gcd\left(\tau^{-1}(a), b\right)$, then

$$p_2 \mid \gcd\left(\tau^{-1}\left(\frac{a}{\tau(p_1)}\right), \tau\left(\frac{b}{p_1}\right)\right)$$

and we take $p_2 := \gcd\left(\tau^{-1}\left(\frac{a}{\tau(p_1)}\right), \tau\left(\frac{b}{p_1}\right)\right)$ and so on until we arrive - as above - at a p_N and we may again take $v = \mathrm{gff}(p_1, ..., p_N)$.

Obviously, the "refined conclusions" yield a polynomial which divides the polynomial resulting from the "straightforward conclusion".

Let's write down the complete (Gosper-)algorithm using the "refined conclusions":

Algorithm 6.4.3 (Gosper) *by R. W. Gosper, Jr. resp. Peter Paule*
INPUT: $y(x + 1) - y(x) = f$ with $f \in \mathbb{H}$
OUTPUT: The hypergeometric (partial) solution y of $y(x + 1) - y(x) = f$ (if it exists)

BEGIN

$\quad \frac{a(x)}{b(x)} := \frac{f(x+1)}{f(x)}$

Compute the largest positive integer N such that $a(x)$ and $b(x + N)$ have a nontrivial common divisor.

IF $N > 0$ was found THEN

    FOR i:=1 TO N DO

$$p_i(x) := \gcd(a(x-1), b(x+i-1))$$

$$a(x) := \frac{a(x)}{p(x+1)}$$

$$b(x) := \frac{b(x)}{p(x-i+1)}$$

  $$v(x) := \mathrm{gff}(p_1, ..., p_N)$$

ELSE $v(x) := 1$

Find the polynomial solution $u(x)$ of (6.14) (if it exists)

IF $u(x)$ could be found THEN

$$y(x) := \frac{u(x)}{v(x)} \cdot f(x)$$

  Return $y(x)$

ELSE Return $\emptyset$

END

♦

Remark 6.4.4 *Remarks on Algorithm 6.4.3:*

- *Note the similarity to Abramov's universal denominator algorithm (see Algorithm 5.1.11). However, note also the different loops in both algorithms: Abramov's algorithm may return a wrong result - at least as soon as the order of the difference equation is greater than 1 - using the loop "FOR i:=1 TO N DO" (resp. "FOR i:=0 TO N DO"), Gosper's algorithm would also work correctly with the loop "FOR i:=N DOWNTO 1 DO" (but may return denominators of higher degree).*

- *At this point another summation problem should be outlined: Given a rational function $f(x)$, find a rational function $r(x)$ and a rational function $h(x)$ which has a denominator of lowest possible degree such that*

$$\sum f(x) = r(x) + \sum h(x)$$

This problem - called indefinite rational summation *- is obviously linked to the difference equation $y(x+1) - y(x) = f(x)$ which is Gosper's equation, but with a rational right hand side. The main step is to extract the summable part - the $r(x)$ - which is connected to Algorithm 6.4.3 (and also to Abramov's Algorithm 5.1.11), because the denominator of $r(x)$ is determined in a analogous way. For more details see [Pau95] and [Abr95b].*

Let's shortly explain the connection between Algorithm 6.4.1 (original approach) and Algorithm 6.4.3 (new approach): From equations (6.15) and (6.16) it follows that

$$
\begin{aligned}
a &= \tau(p_1) \cdot \tau(p_2) \cdot \dots \cdot \tau(p_N) \cdot \alpha \quad \text{with } \alpha \in \mathbb{K}[x] \\
b &= p_1 \cdot \tau^{-1}(p_2) \cdot \dots \cdot \tau^{-N+1}(p_N) \cdot \beta \quad \text{with } \beta \in \mathbb{K}[x]
\end{aligned}
$$

Hence,

$$
\begin{aligned}
\frac{f(x+1)}{f(x)} &= \frac{a(x)}{b(x)} = \frac{\alpha}{\beta} \cdot \frac{\tau(p_1) \cdot \tau(p_2) \cdot \dots \cdot \tau(p_N)}{p_1 \cdot \tau^{-1}(p_2) \cdot \dots \cdot \tau^{-N+1}(p_N)} = \\
&= \frac{\alpha}{\beta} \cdot \frac{\tau(p_1)}{p_1} \cdot \frac{\tau(p_2) \cdot p_2}{p_2 \cdot \tau^{-1}(p_2)} \cdot \dots \cdot \frac{\tau(p_N) \cdot p_N \cdot \dots \cdot \tau^{-N+2}(p_N)}{p_N \cdot \tau^{-1}(p_N) \cdot \dots \cdot \tau^{-N+1}(p_N)} = \\
&= \frac{\alpha(x)}{\beta(x)} \cdot \frac{\tau(\mathrm{gff}(p_1, \dots, p_N))}{\mathrm{gff}(p_1, \dots, p_N)} = \frac{\alpha(x)}{\beta(x)} \cdot \frac{v(x+1)}{v(x)} \cong \frac{A(x)}{B(x)} \cdot \frac{C(x+1)}{C(x)}
\end{aligned}
$$

Thus, Algorithm 6.4.3 can be used to compute the Gosper-form of a rational function (used in Algorithm 6.4.1. As it turns out that $\gcd(\alpha(x), v(x)) = \gcd(\beta(x), v(x+1)) = 1$ we even get the Gosper-Petkovšek-form. Consequently, the proof of Lemma 6.1.2 could also be reformulated using the gff-concept.

6.4.2   Generalizations of Gosper's Algorithm

As Gosper's algorithm deals with the simplest possible inhomogeneous equations, it is open for generalizations concerning the order, the coefficients and even the right hand side.

Petkovšek's generalization

In [Pet94] Marko Petkovšek deals with the following problem:

Problem 6.4.5 *Find a hypergeometric solution* y *of the difference equation* $\sum_{k=0}^{n} p_k \cdot \tau^k(y) = f$ *with*

- $f \in \mathbb{H}$

- $p_k \in \mathbb{K}[x]$

- p_0 *and* p_r *are nonzero constants*

Analogously to the derivation of Gosper's algorithm, we may set $R(x) = \frac{y(x)}{f(x)}$ (see also Lemma 2.3.5) and $r(x) = \frac{f(x+1)}{f(x)} = \frac{A(x)}{B(x)} \cdot \frac{C(x+1)}{C(x)}$

$$
\sum_{k=0}^{n} p_k \cdot \tau^k(y) = f \Leftrightarrow \sum_{k=0}^{n} p_k(x) \cdot R(x+k) \cdot \prod_{j=0}^{k-1} r(x+j) = 1 \qquad (6.17)
$$

Looking back to the order 1 (Gosper's) case, we now try $R(x) = \frac{f(x)}{g(x) \cdot C(x)}$, with $f, g \in \mathbb{K}[x]$ and $\gcd(f(x), g(x)) = 1$. We hope that $g(x)$ will turn out to be a constant!

Thus, from (6.17) we get:

$$\sum_{k=0}^{n} p_k(x) \cdot \frac{f(x+k)}{g(x+k) \cdot C(x+k)} \cdot \prod_{j=0}^{k-1} \frac{A(x+j)}{B(x+j)} \cdot \frac{C(x+1+j)}{C(x+j)} = 1$$

$$\sum_{k=0}^{n} p_k(x) \cdot \frac{f(x+k)}{g(x+k)} \cdot \prod_{j=0}^{k-1} \frac{A(x+j)}{B(x+j)} = C(x)$$

$$\sum_{k=0}^{n} p_k(x) \cdot f(x+k) \cdot \frac{\prod_{j=0, j \neq k}^{n} g(x+j)}{\prod_{j=0}^{n} g(x+j)} \cdot \prod_{j=0}^{k-1} A(x+j) \cdot \frac{\prod_{j=k}^{n-1} B(x+j)}{\prod_{j=0}^{n-1} B(x+j)} = C(x)$$

$$\sum_{k=0}^{n} p_k(x) \cdot f(x+k) \cdot \prod_{j=0, j \neq k}^{n} g(x+j) \cdot \prod_{j=0}^{k-1} A(x+j) \cdot \prod_{j=k}^{n-1} B(x+j) =$$

$$= C(x) \cdot \prod_{j=0}^{n-1} B(x+j) \cdot \prod_{j=0}^{n} g(x+j)$$

Obviously, $g(x)$ divides the right hand side and all terms of the sum except the one with $k = 0$ (in which it does not appear). Thus:

$$g(x) \mid p_0(x) \cdot \prod_{j=1}^{n} g(x+j) \cdot \prod_{j=0}^{n-1} B(x+j) \tag{6.18}$$

Analogously, $g(x+n)$ has to divide the summand with $k = n$, which yields (after shifting by $-n$):

$$g(x) \mid p_n(x-n) \cdot \prod_{j=-n}^{-1} g(x+j) \cdot \prod_{j=-n}^{-1} A(x+j) \tag{6.19}$$

Shifting x by 1 in (6.18) yields (note that p_0 is a constant):

$$g(x+1) \mid p_0(x) \cdot \prod_{j=2}^{n+1} g(x+j) \cdot \prod_{j=1}^{n} B(x+j) \tag{6.20}$$

By multiplying (6.18) and (6.20) it follows that

$$g(x) \mid p_0(x)^2 \cdot \prod_{j=2}^{n} g(x+j)^2 \cdot g(x+n+1) \cdot \prod_{j=0}^{n-1} B(x+j) \cdot \prod_{j=1}^{n} B(x+j)$$

and by induction we get for $m \geq 1$:

$$g(x) \mid p_0(x)^{2^{m-1}} \cdot \prod_{j=m}^{n+m-1} g(x+j)^{\gamma_j} \cdot \prod_{j=0}^{n+m-2} B(x+j)^{\beta_j}$$

As the characteristic of $\mathbb{K}$ is zero there is an m such that $\gcd(g(n), g(n+j)) = 1$ for $j \geq m$, therefore:

$$g(x) \mid p_0(n)^{2^{m-1}} \cdot \prod_{j=0}^{n+m-2} B(x+j)^{\beta_j}$$

Analogously, from (6.19) we get

$$g(x) \mid p_n(x-n)^{2^{m-1}} \cdot \prod_{j=-n-m+1}^{-m} A(x+j)^{\alpha_j}$$

From the properties of $A(x)$ and $B(x)$ given by (6.4) it follows that $g(x)$ is a constant. Thus, we can really write $R(x) = \frac{q(x)}{C(x)}$ (just like in the order 1 case) where the polynomial $q(x)$ satisfies:

$$\sum_{k=0}^{n} p_k(x) \cdot q(x+k) \cdot \prod_{j=0}^{k-1} A(x+j) \cdot \prod_{j=k}^{n-1} B(x+j) = C(x) \cdot \prod_{j=0}^{n-1} B(x+j)$$

Again it follows that $q(x)$ is divisible by $B(x-1)$, therefore we finally get $R(x) = \frac{B(x-1) \cdot p(x)}{C(x)}$, where $p(x)$ satisfies:

$$\sum_{k=0}^{n} p_k(x) \cdot p(x+k) \cdot \prod_{j=0}^{k-1} A(x+j) \cdot \prod_{j=k-1}^{n-2} B(x+j) = C(x) \cdot \prod_{j=0}^{n-2} B(x+j) \quad (6.21)$$

Remark 6.4.6 *Suppose that we additionally have $\gcd(A(x), B(x+k)) = 1$ for $k \in \{-n+1, -n+2, ..., -1\}$ - see (6.4) - than equation (6.21) simplifies to*

$$\sum_{k=0}^{n} p_k(x) \cdot p(x+k) \cdot \prod_{j=0}^{k-1} A(x+j) \cdot \prod_{j=-n+k}^{-1} B(x+j) = C(x)$$

and $R(x)$ changes to

$$R(x) = \frac{\prod_{i=1}^{n} B(x-i) \cdot p(x)}{C(x)}$$

Let's sum up Petkovšek's algorithm:

Algorithm 6.4.7 (General Gosper) *by Marko Petkovšek*
INPUT: $Ly = f$ with

- *$L = \sum_{i=0}^{n} p_k \cdot \tau^k$ where $p_k \in \mathbb{K}[x]$*

- p_0 *and* p_r *are nonzero constants*

- $f \in \mathbb{H}$

OUTPUT: The hypergeometric (partial) solution y *of* $Ly = f$ *(if it exists)*

BEGIN

$r(x) := \frac{f(x+1)}{f(x)}$

Compute $A(x)$, $B(x)$, $C(x)$ such that $r(x) = \frac{A(x)}{B(x)} \cdot \frac{C(x+1)}{C(x)}$

Find a polynomial solutions $p(x)$ of (6.21) (if it exists)

IF $p(x)$ could be found THEN

$R(x) := \frac{B(x-1) \cdot p(x)}{C(x)}$

$y(x) := R(x) \cdot f(x)$

Return $y(x)$

ELSE Return $\emptyset$

END

$\blacklozenge$

Note that it would have been possible to stop the above derivation at equation (6.17), because this is (unlike to the derivation of Hyper) a *linear* difference equation which can be solved with the methods presented in Chapter 5. Basically, this has been done in [PWZ96]. Using the symmetric product operation we will now derive two other generalizations of Gosper's algorithm where the second one which will turn out to be the same as in [PWZ96]. However, we think that our algorithm looks a little bit "nicer":

Generalizations using the symmetric product

We have given the following problem:

Problem 6.4.8 *Given a difference operator* $L \in \mathbb{K}(x)[\tau]$, *find - if it exists -* $y \in \mathbb{H}$ *such that* $Ly = f$ *where* $f \in V(\tau - r) \subset \mathbb{H}$ *meaning* $\frac{\tau(f)}{f} = r$.

¿From Lemma 2.3.5 we know that there exists $\tilde{y} \in \mathbb{K}(x)$ such that $Ly = y \cdot \tilde{y}$ and therefore $y = f \cdot \tilde{y}$. In other words: The hypergeometric part of the right hand side will also appear in the (partial) solution and we have again the same method - instead of finding a hypergeometric function, find a rational function

of an equivalent problem. Now observe the following equalities and remember Lemma 3.2.7:

$$f \;\; = \;\; Ly = L(f \cdot \tilde{y}) = (L \cdot f)(\tilde{y}) = \left(L \cdot \frac{1}{\frac{1}{f}}\right)(\tilde{y}) =$$

$$= \;\; \frac{1}{\tau^{order(L)}\left(\frac{1}{f}\right)} \cdot \left(\tau^{order(L)}\left(\frac{1}{f}\right) \cdot L \cdot \frac{1}{\frac{1}{f}}\right)(\tilde{y}) =$$

$$= \;\; \tau^{order(L)}(f) \cdot \left(L\text{\textcircled{S}}\left(\tau - \frac{1}{r}\right)\right)(\tilde{y})$$

Consequently, $\tilde{y}$ is a rational solution of

$$L\text{\textcircled{S}}\left(\tau - \frac{1}{r}\right)\tilde{y} = \frac{f}{\tau^{order(L)}(f)}$$

Thus, we derived the following algorithm:

Algorithm 6.4.9 (HyperPartial) *by Christian Weixlbaumer*
INPUT: Ly = f with

- $L \in \mathbb{K}(x)[\tau]$ *with normal L and order(L) = n*

- $f \in \mathbb{H}$

OUTPUT: The hypergeometric (partial) solution y of Ly = f (if it exists)

BEGIN

> $r := \frac{\tau(f)}{f}$
>
> $M := L\text{\textcircled{S}}\left(\tau - \frac{1}{r}\right)$
>
> Compute a rational solution $\tilde{y}$ of $M\tilde{y} = \frac{f}{\tau^n(f)}$.
>
> IF $\tilde{y}$ could be found THEN
>
> > $y := f \cdot \tilde{y}$
> > Return y
>
> ELSE Return $\emptyset$

END

<div align="right">♦</div>

Remark 6.4.10 *The big surprise: Equation (6.17) and the equation in Algorithm 6.4.9 $M\tilde{y} = \frac{f}{\tau^n(f)}$ are the same ($\tilde{y}$ corresponds to the $R(x)$)! Recall Remark 3.2.8, where we derived*

$$L\text{\textcircled{S}}\left(\tau - \frac{1}{r}\right) = \sum_{k=0}^{n} p_k \prod_{j=k}^{n-1} \frac{1}{\tau^j(r)}\tau^k = \frac{1}{\prod_{j=0}^{n-1}\tau^j(r)} \cdot \sum_{k=0}^{n} p_k \prod_{j=0}^{k-1} \tau^j(r)\tau^k$$

Multiplying both sides by $\frac{\tau^n(f)}{f} = \prod_{j=0}^{n-1} \tau^j(r)$ *yields*

$$\frac{\tau^n(f)}{f} \cdot L\circledS\big(\tau - \frac{1}{r}\big)(\tilde{y}) = \sum_{k=0}^{n} p_k \prod_{j=0}^{k-1} \tau^j(r)\tau^k$$

which is the difference operator corresponding to the left hand side of equation (6.17). This equation is the one given in [PWZ96], too.

Up to now we restricted ourselves on hypergeometric right hand sides f, but the right hand side can also be a $\mathbb{K}$-linear combination of hypergeometric terms. Using Lemma 2.2.19 this causes no problems: Suppose we have $Ly = f$ where $f = \sum_{i=0}^{k} f_i$ with pairwise dissimilar $f_i \in V(\tau - r_i)$. Then we can simply use Algorithm 6.4.9 on each f_i, which yields the following algorithm:

Algorithm 6.4.11 (General HyperPartial) *by Christian Weixlbaumer resp. Marko Petkovšek*
INPUT: $Ly = f$ with

- $L \in \mathbb{K}(x)[\tau]$

- $f \in \mathcal{L}(\mathbb{H})$

OUTPUT: The hypergeometric (partial) solution y of $Ly = f$ (if it exists)

BEGIN

    Write $f = \sum_{i=0}^{k} f_i$ where f_i are pairwise dissimilar hypergeometric terms

    FOR $i \in \{1, ..., k\}$ DO

        $r_i := \frac{\tau(f_i)}{f_i}$

        $M := L\circledS\big(\tau - \frac{1}{r_i}\big)$

        Compute a rational solution $\tilde{y}_i$ of $M\tilde{y}_i = \frac{f_i}{\tau^n(f_i)}$

        IF $\tilde{y}_i$ could not be found THEN Return $\emptyset$ and stop

    $y = \sum_{i=0}^{k} f_i \cdot \tilde{y}_i$

    Return y

END

♦

Remark 6.4.12 *One the one there exists examples where it is not necessary to rearrange the right hand side f into a sum of dissimilar terms, but it suffices to take for example simply each summand of f. However, on the other hand there exists examples where the division into dissimilar terms is the only way to get a solution. See e.g. Example 6.4.14.*

Without going into details we mention a further generalization of Gosper's algorithm described in [AbvH97]: Given the equation $\Delta y = f$ and the operator $L \in \mathbb{K}(x)[\tau]$ of minimal order such that $Lf = 0$, one can find $\tilde{L} \in \mathbb{K}(x)[\tau]$ of minimal order such that $\Delta(V(\tilde{L})) = V(L)$. Moreover, if the orders of L and $\tilde{L}$ are the same then one can also find $M \in \mathbb{K}(x)[\tau]$ such that $M(f)$ is a solution of $\Delta y = f$, i.e. $\Delta(M(f)) = f$.

6.4.3   Increasing the Order by 1

The following Lemma shows how to solve an inhomogeneous equation of order n by solving an homogeneous equation of order $n + 1$ (see also Remark 5.2.9).

Lemma 6.4.13 *Let $L \in \mathbb{K}(x)[\tau]$, let $f \in V(\tau - r)$ and let be y an arbitrary solution of $Ly = f$. Then y is also a solution of $\bar{L}y = 0$ where $\bar{L} := (\tau - r) \cdot L$ and we have*

$$V(\bar{L}) = \mathcal{L}(V(L) \cup \{y\})$$

Proof:

"$\supseteq$": L is a right hand factor of $\bar{L}$, hence we have $V(L) \subseteq V(\bar{L})$. Moreover, $\bar{L}y = ((\tau - r) \cdot L)y = (\tau - r)(Ly) = (\tau - r)(f) = 0$, thus $y \in V(\bar{L})$.

"$\subseteq$": Let $y_i \in V(\bar{L})$, arbitrary, then $0 = \bar{L}y_i = (\tau - r)(Ly_i)$ and hence $Ly_i = c \cdot f$ with $c \in \mathbb{K}$. If $c = 0$ then this implies that $y_i \in V(L)$, if $c \neq 0$ then $y_i \in V(L) + c \cdot y$.

∎

Although the method is simple the resulting algorithm is not very efficient.

6.4.4   Examples

Example 6.4.14 *Let's look for a partial solution of the difference equation*

$$2y(x + 2) - 8y(x + 1) - y(x) = 4 \binom{2x}{x + 2} - 5 \binom{2x}{x}$$

Because the leading and trailing coefficient are constant we can use both, Algorithm 6.4.7 and Algorithm 6.4.9. We do not need Algorithm 6.4.11, because the right hand side is hypergeometric and not a linear combination of hypergeometric terms!

$$\frac{4\binom{2x+2}{x+3} - 5\binom{2x+2}{x+1}}{4\binom{2x}{x+2} - 5\binom{2x}{x}} = \frac{2(2x + 1)(x^2 + 21x + 30)}{(x + 3)(x^2 + 19x + 10)} = \frac{A(x)}{B(x)} \cdot \frac{C(x + 1)}{C(x)}$$

with $A(x) = 4x + 2$, $B(x) = x + 3$ and $C(x) = x^2 + 19x + 10$. Both Algorithms do not have any problems and compute the partial solution $y_p = \frac{(2x)!}{x!^2} = \binom{2x}{x}$. Thus, the general solution y is given by

$$y = c_1 \cdot \left(\frac{4 - 3\sqrt{2}}{2}\right)^x + c_2 \cdot \left(\frac{4 + 3\sqrt{2}}{2}\right)^x + \binom{2x}{x}$$

This example also indicates that in Algorithm 6.4.11 it is essential to divide the right hand side into pairwise dissimilar terms: If we searched for a partial solution for the right hand sides $4\binom{2x+2}{x+3}$ and $-5\binom{2x+2}{x+1}$ individually, we would get no solution (in both cases)!

Example 6.4.15 *Let's consider the difference equation*

$$y(x+2) - 2y(x+1) + y(x) = 2^x + 1$$

The right hand side is not hypergeometric, but consists of the (dissimilar) hypergeometric terms 2^x and 1. For 2^x we get the partial solution 2^x, for 1 we get the partial solution $\frac{x^2}{2}$. Thus, the general solution is given by

$$y = c_1 + c_2 x + 2^x + \frac{x^2}{2}$$

Note: In our algorithms we have to compute only one rational (resp. polynomial) solution. However, if one computes the complete rational (resp. polynomial) solution, it may be that the partial solution is also the general solution. In the last example, we would get the partial solution $c_1 + c_2 x + \frac{x^2}{2}$ instead of $\frac{x^2}{2}$.

Example 6.4.16 *Let's consider the difference equation $Ly = f$ over $\mathbb{C}$ with*

$$
\begin{aligned}
L &= (x+2)\tau^2 - (x^2 + 3x + 1)\tau - (x+1)(x+3) \\
f &= (-2)^x(x+2)(x+3)^2 + 2i^x(x^2 + (4-i)x + 3 - 2i) + x^2 + 3x
\end{aligned}
$$

Using Algorithm 6.4.11 we obtain the partial solutions $x(-2)^x$ for $(-2)^x(x+2)(x+3)^2$ and $(i-1)i^x$ for $2i^x(x^2+(4-i)x+3-2i)$, respectively. Unfortunately, we do not get a partial solution for $x^2 + 3x$. Thus, the given equation has no solution (in $\mathbb{H}$).
If we change f to $f+1$ we will additionally get the partial solution $-\frac{1}{2}$ for $x^2 + 3x + 1$ and the complete partial solution y_p will be given by

$$y_p(x) = x(-2)^x + (i-1)i^x - \frac{1}{2}$$

Using Hyper or HHyper we get the general solution for $Ly = f + 1$

$$y(x) = c_1(-1)^x + c_2 x! + x(-2)^x + (i-1)i^x - \frac{1}{2}$$

Chapter 7

D'Alembertian Solutions

7.1 The Reduction Of Order

Up to now, we are able to compute solutions of linear difference equations which lie in $\mathbb{H}$ - unfortunately, many examples appear which have no hypergeometric solutions or "too few" (less than the order indicates). In the latter case, however, one can have the idea to use a sort of Vietà's Theorem: "If you find a hypergeometric solution, then you will also have the corresponding right hand factor. Thus, the other solutions must be "somewhere" in the remaining left hand factor." The resulting method is due to Jean Baptiste le Rond d'Alembert and known under "the reduction of order".

We will need the following generalization of "hypergeometric" (see Remark 3.1.3) and a further definition which lets future results look more pretty:

Definition 7.1.1 *A function y is called* d'Alembertian *if there exists (monic) first order factors* $\tau - r_1, ..., \tau - r_k$ *with* $r_i \in \mathbb{K}(x)$ *such that*

$$(\tau - r_1) \cdot ... \cdot (\tau - r_k)(y) = 0$$

In other words, y is d'Alembertian if it is a solution of a (monic) difference operator (of order k) which can be factored into first order linear factors. We will denote the set of all d'Alembertian functions by $\mathbb{A}$.

Let's sum up some important properties of $\mathbb{A}$:

Proposition 7.1.2 *Let* $L \in \mathbb{K}(x)[\tau]$

(a) $\mathcal{L}(\mathbb{H}) \subset \mathbb{A}$

(b) $\mathbb{A}$ *is a linear space over* $\mathbb{K}$

(c) $y \in \mathbb{A} \Rightarrow Ly \in \mathbb{A}$

Proof: We will merely prove (a), as (b) and (c) can be found in [AbPe94]. Let $\lambda_1 f_1 + \lambda_2 f_2 \in \mathcal{L}(\mathbb{H})$ and let $(\tau - r_1)f_1 = 0$, then $(\tau - r_1)(\lambda_1 f_1 + \lambda_2 f_2) = \lambda_2(\tau - r_1)f_2$, which is in $\mathbb{H}$ because of Lemma 2.3.5. Thus there exists $r_2 \in \mathbb{K}(x)$ such that $(\tau - r_2)(\lambda_2(\tau - r_1)f_2) = 0$ and therefore $(\tau - r_2)(\tau - r_1)(\lambda_1 f_1 + \lambda_2 f_2) = 0$

$\blacksquare$

Example 7.1.3 *One well known non-hypergeometric but d'Alembertian function (resp. sequence) is*

$$H_n := \sum_{k=1}^{n} \frac{1}{k}$$

the n-th harmonic number, which is (for example) annihilated by the second-order operator $(\tau - \frac{x+1}{x+2}) \cdot (\tau - 1)$.

Definition 7.1.4 *Given the difference equation $\Delta y = f$, we will denote the set of solutions by $y = \sum f$. Obviously, this set is given by*

$$\sum f := \{y_p + c \mid c \in \mathbb{K}\}, \text{ where } y_p \text{ is a particular solution of } \Delta y = f$$

Moreover, let $\mathcal{F}$ be a set of functions, then $\sum \mathcal{F}$ denotes the set of all y such that $\Delta y \in \mathcal{F}$.

Remark 7.1.5 *Remarks on Definition 7.1.4:*

- *This notation is the discrete analogue to the $\int$-symbol: Given the differential equation $y' = f$ (or in terms of the differential operator ∂: $\partial y = f$) the solution is given (resp. denoted) by $y = \int f$.*

- *Note that $\sum f$ can either contain closed-form expressions computed by Gosper's Algorithm 6.4.1 or indefinite sums. Note also the special case $\sum 0$ equals the set of constants.*

With those both, d'Alembertian functions and expressions of the form $y = \sum f$, it is now possible to write down the general solution of an inhomogeneous first order difference equation:

Lemma 7.1.6 *Let $(\tau - r)h = 0$ with $r \in \mathbb{K}(x)^*$, then the general solution $y \in \mathbb{A}$ of $(\tau - r)y = f$ with $f \in \mathbb{H}$ is given by*

$$y = h \cdot \sum \frac{f}{\tau(h)}$$

Proof: $u := \sum \frac{f}{\tau(h)}$ solves $\Delta u = (\tau - 1)u = \frac{f}{\tau(h)} \Leftrightarrow \tau(h) \cdot \tau(u) = f + \tau(h) \cdot u$. Thus,

$$
\begin{aligned}
(\tau - r)(h \cdot u) &= \tau(h) \cdot \tau(u) - r \cdot h \cdot u = f + \tau(h) \cdot u - r \cdot h \cdot u = \\
&= f + (\tau(h) - r \cdot h) \cdot u = f + 0 \cdot u = f
\end{aligned}
$$

The fact that $y \in \mathbb{A}$ follows for example from Lemma 6.4.13.

■

Using this lemma repeatedly we obtain:

Corollary 7.1.7 *Let $(\tau - r_i)h_i = 0$ with $r_i \in \mathbb{K}(x)^*$ for $i \in \{1, 2, ..., n\}$, then the general solution $y \in \mathbb{A}$ of $(\tau - r_n)(\tau - r_{n-1})...(\tau - r_1)y = f$ with $f \in \mathbb{H}$ is given by*

$$y = h_1 \cdot \sum \frac{h_2}{\tau(h_1)} \cdot \sum \frac{h_3}{\tau(h_2)} \cdot ... \cdot \sum \frac{f}{\tau(h_n)} \tag{7.1}$$

Remark 7.1.8 *Remarks on Corollary 7.1.7:*

- *Observe that equation (7.1) is also valid for $f = 0$. Moreover, note that the "beautiful" expression for y equals a linear combination of h_1 and representants of the sets $h_1 \cdot \sum \frac{h_2}{\tau(h_1)}$, $h_1 \cdot \sum \frac{h_2}{\tau(h_1)} \cdot \sum \frac{h_3}{\tau(h_2)}$, etc.*

- *Corollary 7.1.7 yields another characterization of d'Alembertian: We call a function y d'Alembertian if y can be written as*

$$y = \varphi_1 \cdot \sum \varphi_2 \cdot ... \cdot \sum \varphi_k$$

with hypergeometric functions $\varphi_1, ..., \varphi_k$.

At the beginning of the chapter we gave the idea for d'Alembert's reduction of order. It is time to write down the details:

Lemma 7.1.9 *Let $L \in \mathbb{K}(x)[\tau]$. Suppose we have found a factorization of the form $L = L_1 \cdot (\tau - r)$, that means we already know a hypergeometric solution $y_1 \in V(L)$ together with the corresponding right hand factor $\tau - r$.*
If $y_L \in V(L_1)$, then

$$y_2 := y_1 \cdot \sum \frac{y_L}{\tau(y_1)}$$

is a (d'Alembertian) solution of $Ly = 0$.

Proof: Because of Lemma 7.1.6 we know that $(\tau - r)y_2 = y_L$, therefore $Ly_2 = L_1 \cdot (\tau - r)y_2 = L_1 y_L = 0$

■

Before we proceed to the algorithm for computing d'Alembertian solutions of $Ly = f$, we cite an interesting theorem from [AbPe94], which is not hard to believe looking at the last lemma:

Theorem 7.1.10 *Let $L \in \mathbb{K}(x)[\tau]$. If the equation $Ly = 0$ has a nonzero d'Alembertian solution, then it also has a hypergeometric solution. Respectively, if the equation $Ly = 0$ has no hypergeometric solution, then it also has no nonzero d'Alembertian solution.*

Algorithm 7.1.11 (D'Alembert Reduction) *by Sergei Abramov and Marko Petkovšek*
INPUT: $Ly = 0$ with $L \in \mathbb{K}(x)[\tau]$
OUTPUT: A basis for $V(L) \cap \mathbb{A}$

BEGIN

> Find a hypergeometric solution h_1 of $Ly = 0$ (resp. the corresponding first order factor R_1)
>
> If h_1 could not be found THEN Return $\emptyset$ and stop
>
> Compute L_1 such that $L_1 \cdot R_1 = L$ (right hand division)
>
> $i := 1$
>
> WHILE a nonzero hypergeometric solution can be found DO
>
> > Find a hypergeometric solution h_{i+1} of $L_i y = 0$ (resp. the corresponding first order factor R_{i+1})
> >
> > Compute L_{i+1} such that $L_{i+1} \cdot R_{i+1} = L_i$
> >
> > $i := i + 1$
>
> Let $\{h_1, h_2, ..., h_m\}$ be the set of all found hypergeometric solutions
>
> $y_1 := h_1$
>
> FOR i:=2 TO m DO compute a solution y_i of $R_{i-1}R_{i-2}...R_1 y = h_i$ using Corollary 7.1.7 and try to eliminate sums by Gosper's algorithm.
>
> Return $\{y_1, y_2, ..., y_m\}$

END

$\blacklozenge$

Theorem 7.1.12 *Let $L \in \mathbb{K}(x)[\tau]$, then Algorithm 7.1.11 computes a basis for $V(L) \cap \mathbb{A}$.*

Proof: Obviously, Algorithm 7.1.11 finds a factorization $L = L_m R_m R_{m-1}...R_1$ where $order(R_i) = 1$ and the equation $L_m y = 0$ has no hypergeometric solution, hence by Theorem 7.1.10 no nonzero d'Alembertian solution.
(1) We will show that $V(L) \cap \mathbb{A} = V(R_m R_{m-1}...R_1)$: If $y \in V(R_m R_{m-1}...R_1)$, then $Ly = 0$ and y is d'Alembertian, thus $y \in V(L) \cap \mathbb{A}$. Conversely, assume that $y \in V(L) \cap \mathbb{A}$, then $\tilde{y} := R_m R_{m-1}...R_1 y \in \mathbb{A}$ (Proposition 7.1.2 (c)) and $L_m \tilde{y} = Ly = 0$, though we already showed that L_m does not have any nonzero d'Alembertian solution. Thus, $\tilde{y} = 0$ and $y \in V(R_m R_{m-1}...R_1)$.
(2) For $i \in \{1, 2, ..., m\}$ we have $R_{i-1}R_{i-2}...R_1 y_i = h_i$ ($i \neq 1$) and $R_i h_i = 0$, thus $Ly_i = 0 \Leftrightarrow y_i \in V(L) \cap \mathbb{A} = V(R_m R_{m-1}...R_1)$. It remains to prove that the y_i are linear independent: Suppose that $y := \sum_{i=1}^{m} \lambda_i y_i = 0$, then $R_{m-1}R_{m-2}...R_1 y = \lambda_m h_m = 0$ and therefore $\lambda_m = 0$. Analogously we can use $R_{j-1}R_{j-2}...R_1$ (for $j \in \{m-1, m-2..., 1\}$) to show that $\lambda_{m-1} = ... = \lambda_1 = 0$, too.

$\blacksquare$

Remark 7.1.13 *Let's give a short summary of all correspondences concerning Algorithm 7.1.11:*

- *The factorization $L = L_m R_m R_{m-1}...R_1$ with first order right hand factors R_i and L_m has no hypergeometric solution.*

- *The hypergeometric solutions $h_1, h_2, ..., h_m$ of $R_i h_i = 0$*

- *The elements y_i of the solution base given by representants of the sets*

$$h_1 \cdot \sum \frac{h_2}{\tau(h_1)} \cdot \sum \frac{h_3}{\tau(h_2)} \cdot ... \cdot \sum \frac{h_i}{\tau(h_{i-1})}$$

- *The general solution of $Ly = f$ given by*

$$y = \varphi_1 \cdot \sum \varphi_2 \cdot ... \cdot \sum \varphi_m \cdot \sum g$$

where g denotes the general (non d'Alembertian) solution of $L_m y = 0$, $\varphi_1 := h_1$ and $\varphi_i := \frac{h_i}{\tau(h_{i-1})}$ for $i \in \{2, 3, ..., m\}$.

- *$y_i \in \varphi_1 \cdot \sum \varphi_2 \cdot ... \cdot \sum \varphi_i$*

- *$h_i = \tau^{i-1}(\varphi_1) \cdot \tau^{i-2}(\varphi_2) \cdot ... \cdot \varphi_i$, thus $R_i \left(\tau^{i-1}(\varphi_1) \cdot \tau^{i-2}(\varphi_2) \cdot ... \cdot \varphi_i \right) = 0$*

With the help of the symmetric product operation $\circledS$, we can even simplify Algorithm 7.1.11, in that sense that it is not necessary to do a (noncommutative) right hand division (in order to find L_1). Besides, it directly computes the φ_i defined above:

Algorithm 7.1.14 (D'Alembert Reduction $\circledS$) *by Christian Weixlbaumer resp. Sergei Abramov and Marko Petkovšek*
INPUT: $Ly = 0$ with $L \in \mathbb{K}(x)[\tau]$
OUTPUT: The general d'Alembertian solution y of $Ly = 0$

BEGIN

Find a hypergeometric solution φ_1 of $Ly = 0$ (resp. the corresponding first order factor $\tau - r_i$).

If φ_1 could not be found THEN Return $\emptyset$ and stop

$M_1 := L \circledS (\tau - \frac{1}{r_1})$

$L_1 := \frac{M_1}{\tau - 1}$ (division can be performed like in the commutative case!)

$i := 1$

WHILE a nonzero hypergeometric solution can be found DO

Find a hypergeometric solution φ_{i+1} of $L_i y = 0$ (resp. the corresponding first order factor $\tau - r_{i+1}$)

$$M_{i+1} := L_i \circledS (\tau - \tfrac{1}{r_i})$$

$$L_{i+1} := \tfrac{M_{i+1}}{\tau - 1}$$

$$i := i + 1$$

Let $\{\varphi_1, \varphi_2, ..., \varphi_m\}$ be the set of all found hypergeometric solutions

Compute $y := \varphi_1 \cdot \sum \varphi_2 \cdot ... \cdot \sum \varphi_m \cdot \sum g$ (where g denotes the general non-d'Alembertian solution of $L_m y = 0$) as the general d'Alembertian solution. One can try to replace $\sum$-quantifiers with Gosper's algorithm.

END

$\blacklozenge$

It should be mentioned that the main idea of the above algorithm (namely, how to do one reduction step) already appeared in [vHo99].

7.1.1   Inhomogeneous Equations

It is no problem to do - at least formally - the inhomogeneous case with the help of the following theorem:

Theorem 7.1.15 *Let $L \in \mathbb{K}(x)[\tau]$ with order(L)=n and let c be the leading coefficient of L. If $\varphi_1 \cdot \sum \varphi_2 \cdot ... \cdot \sum \varphi_n \cdot \sum 0$ is the general solution of $Ly = 0$, then*

$$\varphi_1 \cdot \sum \varphi_2 \cdot ... \cdot \sum \varphi_n \cdot \sum \frac{f}{c \cdot \tau^n(\varphi_1) \cdot \tau^{n-1}(\varphi_2) \cdot ... \cdot \tau(\varphi_n)}$$

is the general solution of $Ly = f$, where $f \in \mathbb{H}$.

Proof: According to Corollary 7.1.7 (divide the given equation by c; the needed factorization is given by Algorithm 7.1.11) the general solution of $Ly = f$ is given by

$$y = h_1 \cdot \sum \frac{h_2}{\tau(h_1)} \cdot \sum \frac{h_3}{\tau(h_2)} \cdot ... \cdot \sum \frac{f}{c \cdot \tau(h_n)}$$

The statement follows after the substitution $h_i = \tau^{i-1}(\varphi_1) \cdot \tau^{i-2}(\varphi_2) \cdot ... \cdot \varphi_i$.

$\blacksquare$

Ignoring this formal solution given by the theorem we can again use our methods from section 6.4 as long as the right hand side f is hypergeometric (or our methods from chapters 4 and 5, for polynomial and rational right hand sides f, respectively).
New ideas are needed for d'Alembertian right hand sides - without going into details we mention the following:

- The first approach is nothing else than a generalization of Lemma 6.4.13: Suppose we have $Ly = f$ together with $Mf = 0$ (M is the minimal annihilation operator for f), then a partial solution of $Ly = f$ is among the solutions of $MLy = 0$. For more details see [Abr96].

- A more effective method is presented in [AbZi96], which is based upon the generalizations of Gosper's algorithm - reducing the problem to the search for rational solutions. This paper also covers the differential and q-difference case.

7.1.2   A Slight Improvement

Up to now we computed a factorization of the form $L = L_m R_m R_{m-1}...R_1$ with first order right hand factors R_i and L_m has no further first order right hand factor. At this point, one could have the idea to look for first-order left hand factors of L_m which can be computed as adjoints of right hand factors of the adjoint operator L_m^* (remember Remark 3.2.3). This may yield another d'Alembertian solutions:

Theorem 7.1.16 *Let $L \in \mathbb{K}(x)[\tau]$ with order(L)=n and let c be the leading coefficient of L. If $\varphi_1 \sum \varphi_2... \sum \varphi_n \sum 0$ is the general solution of $Ly = 0$, then*

$$\frac{1}{c \cdot \tau^n(\varphi_1) \cdot \tau^{n-1}(\varphi_2) \cdot .... \cdot \tau(\varphi_n)} \sum \tau(\varphi_n) \sum \tau^2(\varphi_{n-1})... \sum \tau^{n-1}(\varphi_2) \sum 0$$

is the general solution of the adjoint equation $L^*y = 0$.

Proof: By some substitutions, but see [AbPe94]

∎

In practice we will use Theorem 7.1.16 in the following manner:

- Compute the adjoint equation and use Algorithm 7.1.11 to find the general solution in the form
$$\varphi_1 \sum \varphi_2... \sum \varphi_k \sum 0$$
where it may be that only the first $j < k$ φ_i are known hypergeometric solutions and the others are only formal solutions.

- By Theorem 7.1.16 the general solution of the second adjoint is given by

$$\frac{1}{c \cdot \tau^k(\varphi_1) \cdot \tau^{k-1}(\varphi_2) \cdot .... \cdot \tau(\varphi_k)} \sum \tau(\varphi_k)... \sum \tau^{k-1}(\varphi_2) \sum 0$$

- Using Proposition 3.2.2 we get

$$\tau^{-k} \left(\frac{1}{c \cdot \tau^k(\varphi_1) \cdot \tau^{k-1}(\varphi_2) \cdot .... \cdot \tau(\varphi_k)} \sum \tau(\varphi_k)... \sum \tau^{k-1}(\varphi_2) \sum 0 \right)$$

as the general solution of the original equation

We conclude with a short remark on factorization:

Remark 7.1.17 *With our knowledge it is now possible to factor every differ-ence operator $L \in \mathbb{K}(x)[\tau]$ with $order(L) \le 3$. More exactly we can factor every difference operator $L \in \mathbb{K}(x)[\tau]$ in the following manner:*

$$L = L_m L_{m-1}...L_1 \tilde{L} R_1 R_2 ... R_n$$

where $\tilde{L}$ contains no first order left/right hand factors. An algorithm for fac-torization of difference (and differential) operators (which finds also nonlinear factors) can be found in [BrPe95]. A preliminary version of this paper appeared as [BrPe94].

7.2 Examples

Example 7.2.1 *Continuation of Example 6.3.3: Compute the second (non-hypergeometric) solution of $Ly = 0$, where $L = 3\tau^2 - x\tau + x - 1$*

We find the following factorization - the right hand factor corresponds to the known polynomial solution:

$$L = \left(3\tau - \frac{(x-1)(x^2 - 11x + 27)}{x^2 - 9x + 17} \right) \cdot \left(\tau - \frac{x^2 - 9x + 17}{x^2 - 11x + 27} \right)$$

Thus, the general solution y of $Ly = 0$ is given by

$$y = (x^2 - 11x + 27) \cdot \sum \frac{(x-2)!}{3^x (x^2 - 9x + 17)(x^2 - 11x + 27)} \sum 0$$

Of course, the sum cannot be simplified with Gosper's algorithm, because oth-erwise Hyper and HHyper would have found already two solutions.

Example 7.2.2 *Let $L = \tau^2 - (x+1)\tau - (x+1)$, thus we have*

$$y(x+2) - (x+1)y(x+1) - (x+1)y(x) = 0$$

We are looking for a general D'Alembertian solution of $Ly = 0$.

Again Hyper and HHyper merely find one solution, namely $x!$. Thus, we know that the second solution must be in $\mathbb{A}\backslash\mathbb{H}$. As we can compute the factorization

$$\tau^2 - (x+1)\tau - (x+1) = (\tau + 1) \cdot (\tau - (x+1))$$

we get the general solution

$$y = x! \sum \frac{(-1)^x}{(x+1)!} \sum 0$$

Example 7.2.3 *Let $L = p_5(x)\tau^5 + p_4(x)\tau^4 + p_3(x)\tau^3 + p_2(x)\tau^2 + p_1(x)\tau + p_0(x)$ with*

$$
\begin{aligned}
p_5(x) &= x(x+1)(x+2)(x+3) \\
p_4(x) &= -x(x+1)(x+2)(x+5)(5x+19) \\
p_3(x) &= -x(x+1)(x+4)^2(x+5)(x^2-4x-17) \\
p_2(x) &= x(x+3)^2(x+4)^2(x+5)(4x^2+6x+1) \\
p_1(x) &= -(x+2)^2(x+3)^2(x+4)^2(x+5)(5x^2+6x+2) \\
p_0(x) &= 2(x+1)^3(x+2)^2(x+3)^2(x+4)^2(x+5)
\end{aligned}
$$

We are looking for a general D'Alembertian solution of $Ly = 0$.

Using Algorithm 7.1.14 we find $\tau - (x+1)$ as a first order right hand factor and $L_1 = q_4(x)\tau^4 + q_3(x)\tau^3 + q_2(x)\tau^2 + q_1(x)\tau + q_0(x)$ given by:

$$
\begin{aligned}
q_4(x) &= x(x+1)(x+2)(x+3) \\
q_3(x) &= -4x(x+1)(x+2)(x+4) \\
q_2(x) &= -(x-3)x(x+1)(x+3)(x+4) \\
q_1(x) &= x(x+2)(x+3)(x+4)(3x+2) \\
q_0(x) &= -2(x+1)^2(x+2)(x+3)(x+4)
\end{aligned}
$$

Repeating the procedure, we get the first order right hand factor $\tau - \frac{x+1}{x}$ and $L_2 = \tau^3 - 3\tau^2 - x\tau + 2(x+1)$, which has no hypergeometric solution. L_2^*, however, given by

$$ L_2^* = 2(x+4)\tau^3 - (x+2)\tau^2 - 3\tau + 1 $$

has the first order right hand factor $\tau - \frac{1}{2}$ and $L_3^* = (x+4)\tau^2 + 2\tau - 4$, which has no first order left/right hand factors. Thus, the general solution of $Ly = 0$ is given by

$$ y = x! \sum x \sum \tau^{-3}\left(\frac{2^x}{(x+4)\cdot\tau(\varphi_1)\cdot\tau^2(\varphi_2)} \sum \tau(\varphi_1) \sum \tau^2(\varphi_2) \sum 0 \right) $$

where $\varphi_1 \sum \varphi_2 \sum 0$ is the general solution of $L_3^* y = 0$.
Note that $Ly = 0$ has two hypergeometric solutions, $x!$ and $(x^2 - x)\cdot x!$.

Bibliography

[Abr71] ABRAMOV, S. A. (1971): On the summation of rational functions. *USSR Computational Mathematics and Mathematical Physics* 11, 324-330.

[Abr89a] ABRAMOV, S. A. (1989): Problems in computer algebra that are connected with a search for polynomial solutions of linear differential and difference equations. *Moscow University Computational Mathematics And Cybernetics* no. 3, 63-68.

[Abr89b] ABRAMOV, S. A. (1989): Rational solutions of linear differential and difference equations with polynomial coefficients. *USSR Computational Mathematics and Mathematical Physics* 29, 7-12.

[Abr93] ABRAMOV, S. A. (1993): On D'Alembert substitution, *Proc. ISSAC '93*, ACM Press, 20-26.

[Abr95a] ABRAMOV, S. A. (1995): Rational solutions of linear difference and q-difference equations with polynomial coefficients, *Proc. ISSAC '95*, ACM Press, 285-289.

[Abr95b] ABRAMOV, S. A. (1995): Indefinite sums of rational functions, *Proc. ISSAC '95*, ACM Press, 303-308.

[Abr96] ABRAMOV, S. A. (1996): Symbolic Search Algorithms for Partial d'Alembertian Solutions of Linear Equations, *Programming and Computer Software*, Vol. 22 No. 1, 26-34.

[AbBa98] ABRAMOV, S. A. AND BARKATOU, M. A. (1998): Rational solutions of first-order difference systems, *Proc. ISSAC '98*, ACM Press, 124-131.

[ABP95] ABRAMOV, S. A., BRONSTEIN, M. AND PETKOVŠEK, M. (1995): On polynomial solutions of linear operator equations, *Proc. ISSAC '95*, ACM Press, 290-296.

[AbPe94] ABRAMOV, S. A. AND PETKOVŠEK, M. (1994): D'Alembertian solutions of linear differential and difference equations, *Proc. ISSAC '94*, ACM Press, 169-174.

[AbvH97] ABRAMOV, S. A. AND VAN HOEIJ, M. (1997): A method for the Integration of Solutions of Ore Equations, *Proc. ISSAC '97*, ACM Press, 172-175.

[AbZi96] ABRAMOV, S. A. AND ZIMA, E. V. (1996): D'Alembertian Solutions of Inhomogenous Linear Equations (differential, difference, and some other), *Proc. ISSAC '96*, ACM Press, 232-240.

[Bar99] BARKATOU, M. A. (1999): Rational Solutions of Matrix Difference Equations: The Problem of Equivalence and Factorization, *Proc. ISSAC '99*, ACM Press, 277-282.

[BaDu94] BARKATOU, M. A. AND DUVAL, A. (1994): Sur les séries formelles solutions d'équations aux différences polynomiales, *Annales de l'Institut Fourier (Grenoble)*, 44(2), 495-524.

[BrPe94] BRONSTEIN, M. AND PETKOVŠEK, M. (1994) On Ore rings, linear operators and factorisation, *Programming and Computer Software* 20, 14-26.

[BrPe95] BRONSTEIN, M. AND PETKOVŠEK, M. (1995): An introduction to pseudo-linear algebra, *Theoretical Computer Science* 157, 3-33.

[ChLo96] CHEN, Y. C. AND LOUCK, J. D. (1996): The combinatorial power of the companion matrix, *Linear Algebra and Its Applications* 232 (1996), 261-278.

[Duv83] DUVAL, A. (1983): Lemmes de Hensel et factorisation formelle pour les opérateurs aux différences, *Funkcialaj Ekvacioj*, 26, 349-468.

[Gos78] GOSPER, R. W. JR. (1978): Decision procedure for indefinite hypergeometric summation. *Proc. National Academy of Sciences USA*, 75(1), 40-42.

[Khm99] KHMEL'NOV D. E. (1999): Improved Algorithms for Solving Difference and q-Difference Equations, *Programming and Computer Software*, Vol. 26. No. 2, 107-115.

[Mal00] MALLIK, R. K. (2000): On the solution of a linear homogeneous difference equation with variable coefficients, *SIAM Journal on Mathematical Analysis*, Vol. 31, No. 2, 375-385.

[MaWr94] MAN, Y.-K. AND WRIGHT, F. J. (1994): Fast Polynomial Dispersion Compuatation and Its Application to Indefinite Summation, *Proc. ISSAC '94*, ACM Press, 175-180.

[Ore33] ORE, O. (1933): The theory of non-commutative polynomials, *Annals Of Mathematics* 34, 480-508.

[Pau93] PAULE, P. (1993): Greatest factorial factorization and symbolic summation I, *RISC-Linz report series 93-02*, Johannes Kepler University, Linz.

[Pau95] PAULE, P. (1995): Greatest Factorial Factorization and Symbolic Summation, *Journal of Symbolic Computation*, 20, 235-268.

[PeSa93] PETKOVŠEK, M. AND SALVY, B. (1993): Finding All Hypergeometric Solutions of Linear Differential Equations, *Proc. ISSAC '93*, ACM Press, 27-33.

[Pet92] PETKOVŠEK, M. (1992): Hypergeometric solutions of linear recurrences with polynomial coefficients, *Journal of Symbolic Computation*, 14, 243-264.

[Pet94] PETKOVŠEK, M. (1994): A generalization of Gosper's algorithm, *Discrete Mathematics* 134, 125-131.

[PeWe00] PETKOVŠEK, M., WEIXLBAUMER, C. (2000): A Comparison of Degree Polynomials (Note), http://www.fmf.uni-lj.si/~petkovsek/

[PWZ96] PETKOVŠEK, M., WILF, H., AND ZEILBERGER, D. (1996): A=B, *A K Peters, Wellesley MA*.

[Tou87] TOURNIER, E. (1987): Solutions formelles d'équations différentielles, *Thèse d'Etat, Faculté des Sciences de Grenoble*, 1987.

[vHo98] VAN HOEIJ, M. (1998): Rational solutions of linear difference equations, *Proc. ISSAC '98*, ACM Press, 120-123.

[vHo99] VAN HOEIJ, M. (1999): Finite singularities and hypergeometric solutions of linear recurrence equations, *Journal of Pure and Applied Algebra*, 139, 109-131.

[vZGe99] VON ZUR GATHEN, J. AND GERHARD, J. (1999): Modern Computer Algebra, *Cambridge University Press*, 1999.