JOHANNES KEPLER
UNIVERSITÄT LINZ
Netzwerk für Forschung, Lehre und Praxis

Uncoupling Systems of Linear Ore Operator Equations

Diplomarbeit

zur Erlangung des akademischen Grades

Diplomingenieur

in der Studienrichtung

Technische Mathematik

Angefertigt am *Institut für Symbolisches Rechnen (RISC)*

Betreuung:

A.Univ.Prof. Dr. Peter Paule

Eingereicht von:

Stefan Gerhold

Linz, 24.2.2002

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäss entnommenen Stellen als solche kenntlich gemacht habe.

Contents

Chapter 1

Introduction

Systems of linear first order ordinary differential equations play a central role in many areas of applied mathematics. Classical textbooks on this subject show how to solve such systems in the case of constant coefficients and leave the solution for more general coefficients to numerical methods. In an attempt to tackle such systems symbolically, a natural first step is to reduce a system to a higher order equation in a single unknown, which we will call a scalar equation (Note that the inverse operation of transforming a higher order differential equation into a system is easy by introducing one additional variable for each derivative of the unknown). A classical method to achieve this 'uncoupling' is the cyclic vector algorithm, which is the topic of section (3.2). This method is unsatisfactory because it gives uncoupled equations with very large coefficients. Furthermore, it does not generalize to certain other kinds of linear operator equations, such as difference or q-difference equations. These operators are of interest e.g. in the task of proving combinatorial identities automatically. Chyzak [11] describes an algorithm generalizing Zeilberger's 'creative telescoping' summation algorithm [23],[24],[20], that is capable of proving a large class of identities involving summation and integration. It uses the unified framework of Ore operators that encompasses (q-)difference operators, differential operators and many others. One step of this algorithm consists of solving a linear Ore operator system with rational function coefficients for rational function solutions. Uncoupling this system with one of the methods described in this thesis allows to solve it by appealing to one of the algorithms for finding rational function solutions of (higher order) differential-, difference or q-difference equations in one unknown [1],[3],[4],[5]. We formulate all uncoupling algorithms for an arbitrary coefficient field, but our implementations require the coefficients to be rational functions.

Chapter (2) introduces the algebraic ingredients that are the theoretical base of the uncoupling algorithms in the later chapters. It begins with the definition of Ore polynomials. Following [10] we show how many kinds of

higher order linear operators can be interpreted as polynomials with a non-commutative multiplication arising from the composition of operators. These polynomials form rings that are free of zero divisors and allow to divide a polynomial by another one on the right, which leads to a generalization of the (extended) Euclidean Algorithm. Thus we can compute a greatest common right divisor and a least common left multiple of two Ore polynomials. In section (2.6), which is based on [20], we describe in detail how some linear operators that arise in practice fit into the Ore framework. In section (2.7) we deal with the central object of interest of this thesis, that is linear systems of Ore operator equations. It turns out that in the literature on uncoupling [6],[10],[25] two ways of encoding such systems in terms of pseudo-linear maps (= Ore operators) are used. One of them requires the study of pseudo-linear maps an finite dimensional vector spaces, which is somewhat similar to (in fact, generalizes) linear algebra [10]. Theorem (2.28) provides the connection between those two types of Ore operator systems.

Section (3) presents Zürcher's uncoupling algorithm. It proceeds in two stages, a characteristic which is shared with the algorithms in sections (5.1) and (5.2). First the system is transformed into an equivalent system with a 'nice' matrix (in the case of Zürcher's algorithm, a block diagonal matrix where each block is a companion matrix). In the second stage, this normal form is used to derive higher order scalar equations. For readers who wish to implement Zürcher's algorithm (or one of the other three uncoupling algorithms from this thesis) we give a pseudo-code listing. Furthermore we analyze the complexity of the algorithm in the worst case and in a certain 'nondegenerate' case, the latter arising most of the time in practice. Finally we outline the 'uncoupling by cyclic vectors' method and point out its connection with Zürcher's algorithm.

In Chapter (4) the theory of Ore polynomials is employed in a variant of Gaussian elimination based on the least common left multiple. This algorithm is less complicated than the other uncoupling algorithms we present, but it returns scalar equations of rather high order (in general, larger than the dimension of the system) with coefficients of high degree.

Whereas Zürcher's algorithm is based on pseudo-linear algebra and Gaussian Elimination on the extended Euclidean Algorithm in Ore polynomial rings, the uncoupling algorithm by Abramov and Zima (section 5.1) requires no theoretical background except the notion of Ore operator introduced in section (2.5). It proceeds in two stages, like Zürcher's algorithm. First the system matrix is transformed into a block-triangular form, then higher order scalar equations are deduced. We give a detailed description of both stages and prove that the block-triangular system is indeed equivalent to the original one. Theorem (5.3) shows that linearly independent solution vectors for the system can be obtained from linearly independent solutions of the uncoupled equations. Then we analyze the complexity of Abramov and Zima's algorithm in a practically important nondegenerate case.

In section (5.2) we present a new variant of Zürcher's algorithm. Instead of a block-diagonal normal form it transforms the system matrix into a block-triangular matrix (but in a different way than Abramov and Zima's algorithm). The deduction of the scalar equations resembles the second stage of Abramov and Zima's algorithm. The idea of our 'incomplete Zürcher' algorithm is to reduce the amount of computation in the first stage for the price of a more complicated deduction of the scalar equations.

Chapter (6) presents our Mathematica package that implements the four algorithms described in this thesis. Then we give some computational examples and compare the methods. One of the key points is that Gaussian elimination is rather inefficient, as already pointed out above. However, it has to be mentioned that all available uncoupling algorithms lead to a blow up in the coefficients of the equations. Sometimes uncoupling can be avoided. For instance, there are direct methods for finding the rational solutions of differential [8] and difference [2] systems with rational coefficients.

Acknowledgements

I want to thank Frédéric Chyzak and Manuel Bronstein for reasons such as discussions and providing me with copies of papers. Special thanks goes to my thesis advisor Peter Paule for suggesting the topic and guiding me through the work.

Chapter 2

Ore Polynomials and Ore Operators

2.1 Notation

All rings and fields in this thesis have characteristic zero. Fields are commutative. We write $K(x)$ and $K((x))$ for the field of rational functions and formal laurent series with coefficients from a field K, respectively. For any field $\mathbb{K}$, $\mathbb{K}^* = \mathbb{K}\backslash\{0\}$. The letter $\mathbb{N}$ denotes the set of nonnegative integers:

$$\mathbb{N} = \{0, 1, 2, \ldots\},$$

and $\mathbb{Q}$ the set (field) of rational numbers. The ring of $n \times n$ matrices over a ring R is written as $\mathrm{Mat}(n, R)$. A^t is the transpose of a matrix or vector A. I stands for the identity matrix. id_M denotes the identity mapping of a set M. The notation $\mathbb{K}[\partial; \sigma, \delta]$ for skew polynomial rings will be defined in section (2.3). The degree of a (skew) polynomial p is written as $\deg(p)$, with $\deg(0) = -\infty$. For a vector space V, V^* is the dual vector space. An equation in one variable is called 'scalar equation'.

Program listings use a self-explanatory pseudo-code. Comments are enclosed in `(* ... *)`.

2.2 Motivation and Preliminaries

The set of linear differential operators of the form

$$\sum_{k=0}^{n} a_k D^k$$

with rational function coefficients $a_k \in \mathbb{Q}(x)$ forms a ring, if addition is defined pointwise and the product of two operators by composition. If we replace D by an indeterminate over $\mathbb{K} := \mathbb{Q}(x)$, say ∂, we obtain a ring of

polynomials in ∂, where $+$ is defined as usual, and $\cdot$ is associative, distributive, but not commutative and has the following property:

The degree of a product is the sum of the degrees of the factors. (2.1)

Rings of this kind were first studied by Ore in [19]. In the same vein, linear difference operators with rational function coefficients

$$\sum_{k=0}^{n} a_k \Delta^k$$

can be viewed as an algebra of noncommutative polynomials.

We will make the relationship between such polynomials and linear operators such as D or Δ precise in section (2.5); but now let us stick to the polynomial viewpoint and check what the above property implies for the product $\partial \cdot a$ of the indeterminate ∂ and an element $a \in \mathbb{K}^*$. The result must be a polynomial of degree 1:

$$\partial a = \bar{a}\partial + a' \quad \text{for some} \quad \bar{a} \in \mathbb{K}^*, a' \in \mathbb{K}.$$

For $a, b \in \mathbb{K}$ we must have, by distributivity,

$$\partial\,(a+b) = \partial a + \partial b = \bar{a}\partial + a' + \bar{b}\partial + b' = \left(\bar{a}+\bar{b}\right)\partial + a' + b',$$

and

$$\partial ab = \left(\bar{a}\partial + a'\right) b = \bar{a}\partial b + a'b = \bar{a}\left(\bar{b}\partial + b'\right) + a'b = \bar{a}\bar{b}\partial + \bar{a}b' + a'b.$$

Hence the map

$$\begin{array}{rcl} \sigma : \mathbb{K} & \to & \mathbb{K} \\ a & \mapsto & \bar{a} \end{array}$$

is a field monomorphism, and the map

$$\begin{array}{rcl} \delta : \mathbb{K} & \to & \mathbb{K} \\ a & \mapsto & a' \end{array}$$

is additive and satisfies the skew Leibniz rule $\delta(ab) = \sigma(a)\delta b + \delta a\ b$. This gives rise to

Definition 2.1 *Let $\sigma : \mathbb{K} \to \mathbb{K}$ be a monomorphism. An additive map $\delta : \mathbb{K} \to \mathbb{K}$ is called a σ-derivation (or pseudo-derivation) if*

$$\delta(ab) = \sigma(a)\delta b + \delta a\ b \tag{2.2}$$

for all $a, b \in \mathbb{K}$.

It is customary to write δa instead of $\delta(a)$.

Example 2.2 *Some examples of σ-derivations:*

(i) If $\sigma = id_{\mathbb{K}}$, δ is a derivation on $\mathbb{K}$ and the pair $(\mathbb{K}, \delta)$ is called a differential field.

(ii) For any $\alpha \in \mathbb{K}$, the map $\delta_\alpha := \alpha(\sigma - id_{\mathbb{K}})$ is a σ-derivation, called an inner σ-derivation. Proof:

$$\begin{aligned} \delta_\alpha(ab) &= \alpha(\sigma(ab) - ab) = \sigma(a)\alpha(\sigma(b) - b) + \alpha(\sigma(a) - a)b \\ &= \sigma(a)\delta_\alpha b + \delta_\alpha a\ b. \end{aligned}$$

□

Lemma 2.3 *Let σ be a monomorphism on $\mathbb{K}$ and δ be a σ-derivation on $\mathbb{K}$. Then,*

(i) If $\sigma \neq id_{\mathbb{K}}$ then there is an element $\alpha \in \mathbb{K}$ such that $\delta = \delta_\alpha$.

(ii) If $\delta \neq 0$ then there is an element $\beta \in \mathbb{K}$ such that $\sigma = \beta\delta + id_{\mathbb{K}}$.

Proof. Let $a, b \in \mathbb{K}$. Expanding both sides of $\delta(ab) = \delta(ba)$ via (2.2) yields

$$\sigma(a)\delta b + \delta a\ b = \sigma(b)\delta a + \delta b\ a,$$

hence

$$(\sigma(a) - a)\,\delta b = (\sigma(b) - b)\,\delta a. \tag{2.3}$$

(i) We can choose $a \in \mathbb{K}$ with $\sigma(a) \neq a$. Then (2.3) implies $\delta = \delta_\alpha$ with $\alpha = \delta a / (\sigma(a) - a)$.

(ii) If $\delta \neq 0$ we can find $a \in \mathbb{K}$ with $\delta a \neq 0$. Let $\beta = (\sigma(a) - a)/\delta a$. Then it follows from (2.3) that $\sigma = \beta\delta + id_{\mathbb{K}}$.

□

By (i) of the preceding lemma and the fact that inner σ-derivations are trivial if $\sigma = id_{\mathbb{K}}$, we find that we always have one of the following three cases:

(i) $\sigma = id_{\mathbb{K}}$ and $\delta = 0$

(ii) $\sigma = id_{\mathbb{K}}$ and δ is an outer (i.e., non-inner) σ-derivation

(iii) $\sigma \neq id_{\mathbb{K}}$ and δ is an inner σ-derivation

2.3 Univariate Ore Polynomials

Definition 2.4 *Let σ be a monomorphism on $\mathbb{K}$ and δ be a σ-derivation. The* ring of Ore polynomials $\mathbb{K}[\partial;\sigma,\delta]$ *is the ring of polynomials in ∂ with coefficients from $\mathbb{K}$ with the usual polynomial addition, and the multiplication given by*

$$\partial a = \sigma(a)\partial + \delta a \quad \text{for} \quad a \in \mathbb{K} \tag{2.4}$$

and extended by associativity and distributivity.

Using (2.4) and associativity, the product of two monomials is ($a, b \in \mathbb{K}, m, n \in \mathbb{N}, n > 0$):

$$\left(a\partial^n\right)\left(b\partial^m\right) = \left(a\partial^{n-1}\right)\left(\partial b\right)\partial^m = \left(a\partial^{n-1}\right)\left(\sigma(b)\partial^{m+1} + \delta b\ \partial^m\right). \tag{2.5}$$

By induction, this defines the product of two monomials in all cases. For polynomials with arbitrary degrees distributivity yields

$$\left(\sum_{i=0}^{n} a_i\partial^i\right)\left(\sum_{j=0}^{m} b_j\partial^j\right) = \sum_{i=0}^{n}\sum_{j=0}^{n}\left(a_i\partial^i\right)\left(b_j\partial^j\right).$$

Note that we can write any polynomial $A \in \mathbb{K}[\partial;\sigma,\delta]$ in the form $A = \sum_{i=0}^{n} a_i\partial^i$ with $a_i \in \mathbb{K}$, i.e., as the sum of monomials with ∂ on the right, by applying (2.4) iteratively. Thus we can talk about 'coefficients' and 'degree' as for ordinary polynomials.

We argued at the beginning of this chapter that any multiplication of polynomials with property (2.1) must satisfy (2.4) for some σ, δ. Now we will show that the converse holds, too.

Theorem 2.5 *Let $\mathcal{O} = \mathbb{K}[\partial;\sigma,\delta]$ be a ring of skew polynomials. Then for $p, q \in \mathcal{O}$:*

$$\deg(pq) = \deg(p) + \deg(q).$$

Proof: Suppose $p, q \neq 0$ and let $a\partial^n$ and $b\partial^m$ be the leading monomials of p and q, respectively. By applying (2.5) iteratively, we find that the leading monomial of pq is $a\sigma^n(b)\partial^{n+m}$. Now since $ab \neq 0$ and σ is a monomorphism $\deg(pq) = m + n$ follows. □

Clearly, this implies

Corollary 2.6 *Rings of skew polynomials are free of zero-divisors.*

□

Hence we have the cancellation rules

$$\begin{aligned} pq = pr \ &\Rightarrow\ p\,(q-r) = 0 \ \Rightarrow\ q = r, \quad \text{and} \\ qp = rp \ &\Rightarrow\ (q-r)\,p = 0 \ \Rightarrow\ q = r \end{aligned}$$

for any $p, q, r \in \mathbb{K}[\partial; \sigma, \delta],\ p \neq 0$.

Any ring of Ore polynomials $\mathbb{K}[\partial; \sigma, \delta]$ is a $\mathbb{K}$-algebra. Therefore, we will sometimes speak of Ore algebras instead of rings of skew or Ore polynomials. Throughout this thesis, we consider only univariate Ore polynomials. In [11], [12] multivariate Ore algebras are introduced and employed in the task of automatically proving identities involving summations and integrations.

2.4 The Euclidean Algorithm

Let $\mathcal{O} = \mathbb{K}[\partial; \sigma, \delta]$ be a ring of Ore polynomials, $A, B \in \mathcal{O}\backslash\{0\}$, $a\partial^n$ and $b\partial^m$ be their leading monomials. We want to divide A by B on the right, i.e., find $Q, R \in \mathcal{O}$ s.t. $A = QB + R$ and $\deg(R) < \deg(B)$. In the case $n < m$, we simply have

$$A = 0B + A.$$

If $n \geq m$, the right division can be done as follows: With

$$Q_0 := \frac{a}{\sigma^{n-m}(b)}\partial^{n-m},$$

we have

$$\begin{aligned} Q_0 B &= \frac{a}{\sigma^{n-m}(b)}\partial^{n-m} b\partial^m + O(\partial^{n-1}) \\ &= \frac{a}{\sigma^{n-m}(b)}\partial^{n-m-1}\left(\sigma(b)\partial + \delta b\right)\partial^m + O(\partial^{n-1}) \\ &= \frac{a}{\sigma^{n-m}(b)}\partial^{n-m-1}\sigma(b)\partial^{m+1} + O(\partial^{n-1}) = \ldots = \\ &= \frac{a}{\sigma^{n-m}(b)}\sigma^{n-m}(b)\partial^n + O(\partial^{n-1}) = a\partial^n + O(\partial^{n-1}), \end{aligned}$$

where $O(\partial^{n-1})$ stands for any Ore polynomial of degree less than n, hence the leading monomial of $Q_0 B$ is $a\partial^n$. By induction on the degree, we can assume that there are $Q_1, R \in \mathcal{O}$ s.t.

$$A - Q_0 B = Q_1 B + R \quad \text{and} \quad \deg(R) < \deg(B).$$

Then we have

$$A = QB + R \quad \text{and} \quad \deg(R) < \deg(B)$$

with $Q := Q_0 + Q_1$. $R =: \operatorname{rrem}(A, B)$ is called the *right-remainder* of A and B, and $Q =: \operatorname{rquo}(A, B)$ is called their *right quotient*.

In general, we do not require the $\mathbb{K}$-endomorphism σ to be surjective. But if this is the case, that is, σ is an automorphism, there is a similar left division algorithm: With

$$Q_0 := \sigma^{-m}\left(\frac{a}{b}\right)\partial^{n-m}$$

we have

$$\begin{aligned} BQ_0 &= b\partial^m\sigma^{-m}\left(\frac{a}{b}\right)\partial^{n-m} + O(\partial^{n-1}) \\ &= b\partial^{m-1}\left(\sigma^{-m+1}\left(\frac{a}{b}\right)\partial + \delta\sigma^{-m}\left(\frac{a}{b}\right)\right)\partial^{n-m} + O(\partial^{n-1}) \\ &= b\partial^{m-1}\sigma^{-m+1}\left(\frac{a}{b}\right)\partial^{n-m+1} + O(\partial^{n-1}) = \ldots = \\ &= b\sigma^0\left(\frac{a}{b}\right)\partial^n + O(\partial^{n-1}) = a\partial^n + O(\partial^{n-1}), \end{aligned}$$

and we can divide $A - BQ_0$ recursively by B on the left to obtain $Q, R \in \mathcal{O}$ s.t.

$$A = BQ + R \quad \text{and} \quad \deg(R) < \deg(B).$$

Analogously to right division, *left-quotient* and *left-remainder* can be defined by $\mathrm{lquo}(A, B) := Q$ and $\mathrm{lrem}(A, B) := R$, respectively.

Now we can write down the extended (right) Euclidean Algorithm, which, as we will show, yields a greatest common right divisor and a least common left multiple of $A, B \in \mathcal{O}$:

Algorithm 2.7

$R_0 \leftarrow A$, $R_1 \leftarrow B$
$A_0 \leftarrow 1$, $A_1 \leftarrow 0$
$B_0 \leftarrow 0$, $B_1 \leftarrow 1$
$i \leftarrow 1$
while $R_i \neq 0$ *do*
  $i \leftarrow i + 1$
  $Q_{i-1} \leftarrow \mathrm{rquo}(R_{i-2}, R_{i-1})$
  $R_i \leftarrow \mathrm{rrem}(R_{i-2}, R_{i-1})$
  $A_i \leftarrow A_{i-2} - Q_{i-1}A_{i-1}$
  $B_i \leftarrow B_{i-2} - Q_{i-1}B_{i-1}$
$n \leftarrow i$.

This algorithm terminates because $\deg(R_i) < \deg(R_{i-1})$, $1 \leq i \leq n$.

Lemma 2.8 *For* $0 \leq i \leq n$

(i) $R_i = A_iA + B_iB$,

(ii) R_{n-1} *right divides* R_i.

Proof.

(i) Induction on i. The claim holds for $i = 0, 1$, and if it holds for $i-2$ and $i-1$, then

$$\begin{aligned} A_iA + B_iB &= (A_{i-2} - Q_{i-1}A_{i-1})\,A + (B_{i-2} - Q_{i-1}B_{i-1})\,B \\ &= A_{i-2}A + B_{i-2}B - Q_{i-1}\,(A_{i-1}A + B_{i-1}B) \\ &= R_{i-2} - \mathrm{rquo}(R_{i-2}, R_{i-1})R_{i-1} \\ &= \mathrm{rrem}(R_{i-2}, R_{i-1}) = R_i. \end{aligned}$$

(ii) Here we apply induction on i backwards. R_{n-1} right divides both $R_n = 0$ and itself, and from

$$R_{i-2} = Q_{i-1}R_{i-1} + R_i$$

we see that R_{n-1} right divides R_{i-2} if it right divides R_{i-1} and R_i.

□

The following theorem shows that the above algorithm does indeed compute a greatest common right divisor and a least common left multiple of A and B.

Theorem 2.9 (*Correctness of the Euclidean Algorithm)*

(i) $R_{n-1} =: \mathrm{gcrd}(A, B)$ *is a greatest common right divisor of* A *and* B.

(ii) $A_nA = -B_nB =: \mathrm{lclm}(A, B)$ *is a least common left multiple of* A *and* B.

Proof.

(i) By lemma (2.8) (i),

$$R_{n-1} = A_{n-1}A + B_{n-1}B,$$

hence any common right divisor of A and B right divides R_{n-1}. On the other hand, lemma (2.8) (ii) implies that R_{n-1} right divides $A = R_0$ and $B = R_1$.

(ii) Because of lemma (2.8) (i) and the terminating condition of the while-loop, we have

$$A_nA + B_nB = R_n = 0,$$

hence $A_nA = -B_nB$ is a common left multiple of A and B. To see that it is nonzero, first we note

$$\deg(R_i) < \deg(R_{i-1}) \quad \text{and} \quad \deg(Q_{i-1}) = \deg(R_{i-2}) - \deg(R_{i-1})$$

for $2 \leq i \leq n$. By induction on i, we show

$$\deg(A_i) = \deg(B) - \deg(R_{i-1}) \quad \text{and} \quad \deg(B_i) = \deg(A) - \deg(R_{i-1})$$

for $2 \leq i \leq n$. We have $\deg(A_2) = \deg(1) = 0 = \deg(B) - \deg(R_1)$, $\deg(A_3) = \deg(A_1 - Q_2 A_2) = \deg(Q_2) = \deg \operatorname{rquo}(B, R_2) = \deg(B) - \deg(R_2)$, and the induction step is

$$\begin{aligned} \deg(A_i) &= \deg(A_{i-2} - Q_{i-1}A_{i-1}) = \deg(Q_{i-1}) + \deg(A_{i-1}) \\ &= \deg(Q_{i-1}) + \deg(B) - \deg(R_{i-2}) = \deg(B) - \deg(R_{i-1}). \end{aligned}$$

The second assertion is shown analogously. Hence we have $\deg(A_n) = \deg(B) - \deg(R_{n-1}) \geq 0$ and $\deg(B_n) = \deg(A) - \deg(R_{n-1}) \geq 0$, so $A_n \neq 0$ and $B_n \neq 0$. This shows that $A_n A = -B_n B$ is nonzero. In order to show that it is a *least* common left multiple of A and B, suppose $CA = -DB$ is some common left multiple of A and B and define C_i by $C_0 = -D$, $C_1 = C$ and $C_i = C_{i-2} - C_{i-1}Q_{i-1}$ for $2 \leq i \leq n$. We show each of the assertions

$$C_{i-1}R_i - C_i R_{i-1} = 0 \tag{2.6}$$
$$C_{i-1}A_i - C_i A_{i-1} = (-1)^i C \tag{2.7}$$
$$C_{i-1}B_i - C_i B_{i-1} = (-1)^i D \tag{2.8}$$

$(1 \leq i \leq n)$ by induction. The induction bases $i = 1$ are obvious, and the induction steps are

$$\begin{aligned} C_{i-1}R_i - C_i R_{i-1} &= C_{i-1}\operatorname{rrem}(R_{i-2}, R_{i-1}) \\ &\quad - (C_{i-2} - C_{i-1}\operatorname{rquo}(R_{i-2}, R_{i-1}))\, R_{i-1} \\ &= C_{i-1}R_{i-2} - C_{i-2}R_{i-1} = 0, \\ C_{i-1}A_i - C_i A_{i-1} &= C_{i-1}\,(A_{i-2} - Q_{i-1}A_{i-1}) - (C_{i-2} - C_{i-1}Q_{i-1})\, A_{i-1} \\ &= C_{i-1}A_{i-2} - C_{i-2}A_{i-1} = (-1)^i C, \quad \text{and} \\ C_{i-1}B_i - C_i B_{i-1} &= C_{i-1}\,(B_{i-2} - Q_{i-1}B_{i-1}) - (C_{i-2} - C_{i-1}Q_{i-1})\, B_{i-1} \\ &= C_{i-1}B_{i-2} - C_{i-2}B_{i-1} = (-1)^i D. \end{aligned}$$

(2.6) implies that $C_n R_{n-1} = C_{n-1}R_n = 0$, hence $C_n = 0$. (2.7) and (2.8) then show that A_n right divides C, and B_n right divides D. Therefore, $A_n A = -B_n B$ is indeed a (nonzero) least common left multiple of A and B.

□

In the extended Euclidean Algorithm we can save some computation time by omitting the computation of the B_i. By Lemma (2.8)(i), B_{n-1} and B_n can be determined eventually by $B_{n-1} = \operatorname{rquo}(R_{n-1} - A_{n-1}A, B)$ and

$B_n = \mathrm{rquo}(-A_n A, B)$, respectively. This observation is mentioned (for the Euclidean Algorithm for integers) in [16]. For a different algorithmic approach to greatest common left divisors, see [17], where the theory of subresultants is generalized to Ore polynomials.

For an example of a computation with the Euclidean Algorithm, see example (2.25).

More information about the theory of skew polynomials can be found in [13].

2.5 Ore Polynomials as Linear Operators

Let $\mathbb{K}, \sigma, \delta$ be as above and V a $\mathbb{K}$-vector space.

Definition 2.10 *An additive map $\theta : V \to V$ is called* pseudo-linear *(w.r.t. σ and δ) if*

$$\theta(au) = \sigma(a)\theta u + \delta a\ u \quad \textit{for any} \quad a \in \mathbb{K},\ u \in V.$$

Pseudo-linear maps are sometimes called *Ore operators*. This term is used in situations where the pseudo-linear map is supposed to specialize to a linear operator like D or Δ in applications, such as in section (2.6) or at the beginning of section (2.7). We will not use it for the pseudo-linear maps on finite dimensional vector spaces considered later in section (2.7).

Example 2.11

(i) For any $\mathbb{K}$-vectorspace V, every homomorphism $h : V \to V$ is pseudo-linear w.r.t. $\sigma = id_{\mathbb{K}}$ and $\delta = 0$.

(ii) If $\mathbb{K}$, σ, δ are as usual, δ is a pseudo-linear map on $\mathbb{K}$ w.r.t. σ and δ.

(iii) σ is pseudo-linear w.r.t. σ and 0.

In section (2.6) we will describe in detail that the differentiation operator D, the difference operator Δ and several other types of linear operators that arise in practice can be viewed as Ore operators (pseudo-linear maps).

Definition 2.12 *The* constant field *of $\mathbb{K}$ w.r.t. σ and δ is*

$$\mathrm{Const}_{\sigma,\delta} := \{a \in \mathbb{K} \mid \sigma(a) = a \text{ and } \delta a = 0\}.$$

$\mathrm{Const}_{\sigma,\delta}$ is a subfield of $\mathbb{K}$ because it is the intersection of the subfields

$$\mathrm{inv}(\sigma) = \{a \in \mathbb{K} \mid \sigma(a) = a\}$$

and

$$\ker(\delta) = \{a \in \mathbb{K} \mid \delta a = 0\}.$$

Since any subfield of $\mathbb{K}$ must contain (an isomorphic copy of) $\mathbb{Q}$, it follows

Corollary 2.13 *$\sigma|_{\mathbb{Q}} = id_{\mathbb{Q}}$ and $\delta|_{\mathbb{Q}} = 0$.* □

Lemma 2.14 *Let V be a vector space over $\mathbb{K}$ and $\theta : V \to V$ be pseudo-linear (w.r.t. σ and δ). Then θ is $\mathrm{Const}_{\sigma,\delta}$-linear.*

Proof. Let $c \in \mathrm{Const}_{\sigma,\delta}$ and $u, v \in V$. Then,

$$\theta(cu + v) = \theta(cu) + \theta v = (\sigma(c)\theta u + \delta c\ u) + \theta v = c\theta u + \theta v.$$

□

In the case $V = \mathbb{K}$, that is, the vector space of a field over itself, we can characterize all pseudo-linear maps; furthermore, the following lemma shows that in this case there are infinitely many pseudo-linear maps for any pair (σ, δ).

Lemma 2.15 *For any $c \in \mathbb{K}$, the map $\theta_c : \mathbb{K} \to \mathbb{K}$ given by*

$$\theta_c a = c\ \sigma(a) + \delta a$$

is pseudo-linear. Conversely, for any pseudo-linear map $\theta : \mathbb{K} \to \mathbb{K}$ there is an element $c \in \mathbb{K}$ such that $\theta = \theta_c$.

Proof. θ_c is additive because σ and δ are. Furthermore, for $a, b \in \mathbb{K}$ we have

$$\theta_c(ab) = c\ \sigma(ab) + \delta(ab) = c\ \sigma(a)\sigma(b) + \sigma(a)\delta b + \delta a\ b = \sigma(a)\theta_c b + \delta a\ b.$$

To show the converse, we write

$$\theta a = \theta(a\ 1) = \sigma(a)\theta 1 + \delta a,$$

hence $\theta = \theta_c$ with $c = \theta 1$. □

We will encounter several cases where we do not have $\mathbb{K} = V$ as in lemma (2.15), but at least $\mathbb{K} \subset V$; then the proof of lemma (2.15) asserts that the action of θ on $\mathbb{K}$ is determined by σ, δ and $\theta 1$.

So far skew polynomials and pseudo-linear maps are separate concepts. The connection between them is provided by

Definition 2.16 *Given a ring of skew polynomials $\mathcal{O} = \mathbb{K}[\partial; \sigma, \delta]$, a $\mathbb{K}$-vector space V, and a pseudo-linear (w.r.t. σ and δ) map $\theta : V \to V$ the action $*_\theta : \mathcal{O} \times V \to V$ is defined by*

$$\left(\sum_{i=0}^{n} a_i \partial^i\right) *_\theta u = \sum_{i=0}^{n} a_i \theta^i u$$

for any $u \in V$.

By this definition and the following theorem we can view any Ore algebra $\mathbb{K}[\partial;\sigma,\delta]$ as an algebra of linear operators once we find a vector space V and a pseudolinear (w.r.t. σ and δ) map $\theta : V \to V$.

If there is no ambiguity from the context, we will write $*$ instead of $*_\theta$.

Theorem 2.17 *The operation $*_\theta$ turns V into a left $\mathcal{O}$-module.*

Proof. It is clear that we have for $p, q \in \mathcal{O}$, $u, v \in V$:

- $(p+q) * u = p * u + q * u$,
- $p * (u+v) = p * u + p * v$,
- $1 * u = u$.

What remains to show is

- $(pq) * u = p * (q * u)$.

We first prove by induction on n that

$$(a\partial^n b\partial^m) * u = a\partial^n * (b\partial^m * u) \tag{2.9}$$

for any $n, m \geq 0$, $a, b \in \mathbb{K}$ and $u \in V$. If $n = 0$, then

$$(ab\partial^m) * u = ab\theta^m u = a * (b\theta^m u).$$

If (2.9) holds for $n-1$, we obtain

$$\begin{aligned}
(a\partial^n b\partial^m) * u &= \left(\left(a\partial^{n-1}\right)\left(\sigma(b)\partial^{m+1} + \delta b\ \partial^m\right)\right) * u \\
&= \left(a\partial^{n-1}\sigma(b)\partial^{m+1}\right) * u + \left(a\partial^{n-1}\delta b\ \partial^m\right) * u \\
&= a\partial^{n-1} * \left(\sigma(b)\partial^{m+1} * u\right) + a\partial^{n-1} * (\delta b\ \partial^m * u) \\
&= a\partial^{n-1} * \left(\sigma(b)\theta^{m+1}u + \delta b\ \theta^m u\right) = a\partial^{n-1} * \theta\left(b\theta^m u\right) \\
&= a\partial^n * (b\partial^m * u).
\end{aligned}$$

For $p = \sum_{i=0}^n a_i\partial^i, q = \sum_{j=0}^m b_j\partial^j \in \mathcal{O}$ we have, by (2.9),

$$\begin{aligned}
(pq) * u &= \left(\sum_{i=0}^n \sum_{j=0}^m \left(a_i\partial^i\right)\left(b_j\partial^j\right)\right) * u = \sum_{i=0}^n \sum_{j=0}^m \left(\left(a_i\partial^i b_j\partial^j\right) * u\right) \\
&= \sum_{i=0}^n \sum_{j=0}^m = \left(a_i\partial^i * \left(b_j\partial^j * u\right)\right) = p * (q * u).
\end{aligned}$$

□

We conclude this outline of the theory of pseudo-linear maps with the definition of the adjoint of a pseudo-linear map. This concept will be of importance in section (3.2), where cyclic vectors are discussed. First, recall that in linear algebra the adjoint map

$$\begin{aligned} \phi^* : \quad V^* &\to V^* \\ f &\mapsto (v \mapsto f(\phi(v))) \end{aligned}$$

of a vector space homomorphism $\phi : V \to V$ is introduced. This notion can be generalized to pseudo-linear maps, provided that σ is surjective.

Lemma 2.18 *Let V be a vector space over $\mathbb{K}$, $\sigma : \mathbb{K} \to \mathbb{K}$ be an automorphism, $\delta : \mathbb{K} \to \mathbb{K}$ be a σ-derivation and $\theta : V \to V$ be pseudo-linear (w.r.t. σ and δ). Then for all $f \in V^*$, the map ψ_f defined by*

$$\begin{aligned} \psi_f : \quad V &\to \mathbb{K} \\ x &\mapsto \sigma^{-1}(f(\theta x)) - \sigma^{-1}(\delta(f(x))) \end{aligned}$$

is an element of V^*.

Proof. We have to show that ψ_f is linear. Therefore, let $x, y \in V$ and $a \in \mathbb{K}$. Then

$$\begin{aligned} \psi_f(ax+y) &= \sigma^{-1}(f(\theta(ax+y))) - \sigma^{-1}(\delta(f(ax+y))) \\ &= \sigma^{-1}(f(\sigma(a)\theta x + \delta a\ x + \theta y)) - \sigma^{-1}(\delta(af(x) + f(y))) \\ &= \sigma^{-1}(\sigma(a)f(\theta x) + \delta a f(x) + f(\theta y)) - \sigma^{-1}(\sigma(a)\delta f(x) + \delta a\ f(x)) \\ &\quad -\sigma^{-1}(\delta f(y)) \\ &= a\sigma^{-1}(f(\theta x)) + \sigma^{-1}(\delta a)\sigma^{-1}(f(x)) + \sigma^{-1}(f(\theta y)) \\ &\quad -a\sigma^{-1}(\delta f(x)) - \sigma^{-1}(\delta a)\sigma^{-1}(f(x)) - \sigma^{-1}(\delta f(y)) \\ &= a\left(\sigma^{-1}(f(\theta x)) - \sigma^{-1}(\delta f(x))\right) + \sigma^{-1}(f(\theta y)) - \sigma^{-1}(\delta f(y)) \\ &= a\psi_f(x) + \psi_f(y). \end{aligned}$$

□

Definition 2.19 *The* adjoint θ^* *of θ is defined by*

$$\begin{aligned} \theta^* : \quad V^* &\to V^* \\ f &\mapsto \psi_f \end{aligned}$$

with the ψ_f from the preceding lemma.

In the case where θ is a linear map, we have $\sigma = id_{\mathbb{K}}$ and $\delta = 0$, and the definition of the adjoint map from linear algebra is recovered. In general, θ^* is not pseudo-linear w.r.t. σ and δ, but w.r.t. σ^{-1} and $-\sigma^{-1}\delta$, as we will now show.

Theorem 2.20 *Let $\theta : V \to V$ be pseudo-linear w.r.t. σ and δ. Then $-\sigma^{-1}\delta$ is a σ^{-1}-derivation, and $\theta^* : V^* \to V^*$ is pseudo-linear w.r.t. σ^{-1} and $-\sigma^{-1}\delta$.*

Proof. $-\sigma^{-1}\delta$ is a σ^{-1}-derivation because for $a, b \in \mathbb{K}$

$$-\sigma^{-1}\delta(ab) = -\sigma^{-1}\delta(ba) = -\sigma^{-1}\left(\sigma(b)\delta a + \delta b\ a\right) = \\ -\sigma^{-1}(a)\sigma^{-1}(\delta b) - \sigma^{-1}(\delta a)b.$$

Now let $f, g \in V^*$, $x \in V$ and $a \in \mathbb{K}$. We have

$$\begin{aligned}
\theta^*(af+g)(x) &= \sigma^{-1}\left((af+g)(\theta x)\right) - \sigma^{-1}\delta\left((af+g)(x)\right) \\
&= \sigma^{-1}\left(af(\theta x) + g(\theta x)\right) - \sigma^{-1}\left(\sigma(f(x))\delta a + \delta f(x)\ a + \delta g(x)\right) \\
&= \sigma^{-1}(a)\sigma^{-1}(f(\theta x)) + \sigma^{-1}(g(\theta x)) - f(x)\sigma^{-1}(\delta a) \\
&\quad -\sigma^{-1}(a)\sigma^{-1}(\delta f(x)) - \sigma^{-1}(\delta g(x)) \\
&= \sigma^{-1}(a)\left(\sigma^{-1}(f(\theta x)) - \sigma^{-1}(\delta f(x))\right) \\
&\quad -\sigma^{-1}(\delta a)f(x) + \sigma^{-1}(g(\theta x)) - \sigma^{-1}(\delta g(x)) \\
&= \sigma^{-1}(a)\theta^* f(x) - \sigma^{-1}(\delta a)f(x) + \theta^* g(x),
\end{aligned}$$

hence

$$\theta^*(af+g) = \sigma^{-1}(a)\theta^* f - \sigma^{-1}(\delta a)f + \theta^* g.$$

□

2.6 Examples

We saw in example (2.11) that linear maps on vector spaces are special cases of pseudo-linear maps. In this section we give several examples of pseudo-linear maps that are important in applications.

Shift Operator and Difference Operator

For any field K, the set $K^{\mathbb{N}}$ of K-sequences is a commutative ring if addition and multiplication are defined componentwise. The shift operator E is defined by

$$E : K^{\mathbb{N}} \to K^{\mathbb{N}}$$
$$Eu(n) := u(n+1),$$

and the difference operator $\Delta : K^{\mathbb{N}} \to K^{\mathbb{N}}$ is defined by

$$\Delta := E - id.$$

In order to view E and Δ as pseudo-linear maps, we wish to endow $K^{\mathbb{N}}$ with a vector space structure over the field of rational functions $K(x)$. However, the natural scalar multiplication

$$(r(x)u)(n) := r(x)|_{x=n}u(n)$$

($r \in K(x)$, $u \in K^{\mathbb{N}}$) is not well-defined, because r might have poles in $\mathbb{N}$. Therefore, following [20], we introduce the quotient ring

$$\mathcal{S}(K) := K^{\mathbb{N}}/I,$$

where

$$I := \bigcup_{k=0}^{\infty} \ker E^k \subset K^{\mathbb{N}}$$

is the ideal of eventually zero sequences. In other words, we identify sequences that differ only at finitely many places. The elements of $\mathcal{S}(K)$ are called *germs (of sequences)*. Let $\phi : K^{\mathbb{N}} \to \mathcal{S}(K)$ be the canonical epimorphism. Since

$$\ker(\phi E) = (\phi E)^{-1}(0) = E^{-1}\phi^{-1}(0) = E^{-1}(I) = \bigcup_{k=0}^{\infty} \ker E^{k+1} = I,$$

the isomorphism theorem gives rise to an isomorphism

$$\begin{array}{rcl} \tilde{E} : K^{\mathbb{N}}/I & \to & \mathrm{im}\phi E \\ a + I & \mapsto & \phi E(a), \end{array}$$

which is in fact an automorphism of $\mathcal{S}(K)$ and satisfies $\phi E = \tilde{E}\phi$, since

$$\tilde{E}\phi(a) = \tilde{E}(a + I) = \phi E(a)$$

for any $a \in K^{\mathbb{N}}$. $\tilde{E}$ is called the shift operator on $\mathcal{S}(K)$. For simplicity, we will write a for an equivalence class $a + I \in \mathcal{S}(K)$ and E instead of $\tilde{E}$.

To complete the setup, we wish to embed $K(x)$ into $\mathcal{S}(K)$. This is done by

Lemma 2.21 *The map*

$$\begin{array}{rcl} \psi : K(x) & \to & \mathcal{S}(K) \\ r(x) & \mapsto & (r(n))_{n\in\mathbb{N}} \end{array}$$

is a ring monomorphism. (Note that we can ignore the finitely many n for which $r(n)$ is undefined.)

Proof. ψ is apparently additive and multiplicative. To see that it is one-one, let $r_1, r_2 \in K(x)$. If $\psi(r_1) = \psi(r_2)$, there are infinitely many $n \in \mathbb{N}$ s.t. $r_1(n) = r_2(n)$. Writing $r_i = p_i/q_i$ with polynomials $p_i, q_i \in K[x]$, $i = 1, 2$, we get $p_1(n)q_2(n) = p_2(n)q_1(n)$ at infinitely many points, hence $r_1 = r_2$. □

This lemma shows that

$$\mathcal{R}(K) := \psi(K(x))$$

is a subfield of $\mathcal{S}(K)$ isomorphic to $K(x)$, called the *field of rational sequences*. Clearly, $\mathcal{S}(K)$ is a vector space over $\mathcal{R}(K)$, if scalar multiplication is defined componentwise.

Now we are ready to study the behaviour of the shift operator on $\mathcal{S}(K)$. Let $u, v \in \mathcal{S}(K)$, $r \in \mathcal{R}(K)$. We have

$$\begin{aligned} E(u+v)(n) &= (u+v)(n+1) = u(n+1) + v(n+1) = (Eu + Ev)(n), \\ E(ru)(n) &= (ru)(n+1) = r(n+1)u(n+1) = (Er\ Eu)(n), \end{aligned}$$

and

$$\begin{aligned} \Delta(u+v) &= E(u+v) - (u+v) = Eu - u + Ev - v = \Delta u + \Delta v, \\ \Delta(ru) &= E(ru) - ru = Er\ Eu - Er\ u + Er\ u - ru = Er\Delta u + \Delta r\ u. \end{aligned}$$

We read off that

$$E : \mathcal{S}(K) \to \mathcal{S}(K) \text{ is a pseudo-linear map w.r.t. } \sigma = E,\ \delta = 0,$$

and that

$$\Delta : \mathcal{S}(K) \to \mathcal{S}(K) \text{ is a pseudo-linear map w.r.t. } \sigma = E,\ \delta = \Delta.$$

(We do not distinguish between E, Δ and the restrictions $E|_{\mathcal{R}(K)}$, $\Delta|_{\mathcal{R}(K)}$.)

Upon setting $\mathbb{K} = \mathcal{R}(K)$, $\mathcal{S}(K)$ is a module over the ring of Ore polynomials $\mathbb{K}[\partial; E, \Delta]$ due to theorem (2.17), applied with $\theta = \Delta$. By an abuse of notation, we will write $\mathbb{K}[\Delta; E, \Delta]$ instead of $\mathbb{K}[\partial; E, \Delta]$. $\mathbb{K}[\Delta; E, \Delta]$ is the algebra of difference operators with rational function coefficients.

Similarly, $\mathbb{K}[E; E, 0]$ is the algebra of shift operators with rational function coefficients.

Differential Operator

A differential field $(\mathbb{K}, D)$ is a field $\mathbb{K}$ with a derivation $D : \mathbb{K} \to \mathbb{K}$, i.e., an additive mapping that satisfies the Leibniz rule

$$D(ab) = aDb + Da\ b \tag{2.10}$$

for all $a, b \in \mathbb{K}$. For example, let $\mathbb{K} = K((x))$, the field of formal Laurent series over a field K, and D be the usual differentiation operator on $K((x))$. Considering $\mathbb{K}$ as a vector space over itself, we find, looking at (2.10):

$$D : K((x)) \to K((x)) \text{ is a pseudo-linear map w.r.t. } \sigma = id_{\mathbb{K}},\ \delta = D.$$

By theorem (2.17) the ring of Ore polynomials $\mathbb{K}[\partial; id_{\mathbb{K}}, D]$ (which we will write sloppily as $\mathbb{K}[D; id_{\mathbb{K}}, D]$) can be viewed as an algebra of differential operators on $\mathbb{K}$.

q-Shift, q-Difference and q-Differential Operator

Let K be a field and x, q be indeterminates. There is a unique automorphism Q of $\mathbb{K} = K(q)(x)$ that fixes $K(q)$ and satisfies

$$Qx = qx,$$

called the *q-shift* operator. The operator $\Delta_q := Q - id$ is called the *q-difference* operator. Since

$$\begin{aligned} Q(rf)(x) &= r(qx)f(qx) = Qr\ Qf \quad \text{for any } r, f \in \mathbb{K}, \\ \Delta_q(rf) &= Q(rf) - rf = Qr\ Qf - Qr\ f + Qr\ f - rf = Qr\Delta_q f + \Delta_q r\ f \end{aligned}$$

we have:

$$Q : \mathbb{K} \to \mathbb{K} \text{ is a pseudo-linear map w.r.t. } \sigma = Q,\ \delta = 0,$$

and

$$\Delta_q : \mathbb{K} \to \mathbb{K} \text{ is a pseudo-linear map w.r.t. } \sigma = Q,\ \delta = \Delta_q.$$

The *q-differentiation operator* is defined by

$$\begin{aligned} D_q : \mathbb{K} &\to \mathbb{K} \\ f(x) &\mapsto \frac{f(qx) - f(x)}{qx - x} = \frac{\Delta_q f(x)}{\Delta_q x}. \end{aligned}$$

By writing

$$\begin{aligned} D_q(rf)(x) &= \frac{r(qx)f(qx) - r(qx)f(x) + r(qx)f(x) - r(x)f(x)}{qx - x} \\ &= Qr(x)\ D_q f(x) + D_q r(x)\ f(x) \end{aligned}$$

and observing that D_q is additive we find:

$$D_q : \mathbb{K} \to \mathbb{K} \text{ is a pseudo-linear map w.r.t. } \sigma = Q,\ \delta = D_q.$$

The examples of Ore operators and Ore algebras presented in this section are summarized, together with some others, in the following table from [12]. In all these examples, columns one, two, and three give the action of θ, σ

and δ, respectively. The fourth column shows the commutation rule of the corresponding Ore polynomial ring. We set $\mathbb{K} = K(x)$ for some field K, and r denotes an arbitrary element from $K(x)$.

Operator	$\theta f(x)$	$\sigma(r)(x)$	$\delta r(x)$	$(\partial \cdot r)(x)$
Identity	$f(x)$	$r(x)$	0	$r(x)\partial f(x)$
Differentiation	$f'(x)$	$r(x)$	$r'(x)$	$r(x)\partial + r'(x)$
Shift	$f(x+1)$	$r(x+1)$	0	$r(x+1)\partial$
Difference	$\Delta f(x)$	$r(x+1)$	$\Delta r(x)$	$r(x+1)\partial + \Delta r(x)$
q-Shift	$f(qx)$	$r(qx)$	0	$r(qx)\partial$
q-Difference	$\Delta_q f(x)$	$r(qx)$	$\Delta_q r(x)$	$r(qx)\partial + \Delta_q r$
q-Differentiation	$\frac{f(qx)-f(x)}{qx-x}$	$r(qx)$	$\frac{r(qx)-r(x)}{qx-x}$	$r(qx)\partial + D_q r(x)$
Eulerian operator	$xf'(x)$	$r(x)$	$xr'(x)$	$r(x)\partial + xr'(x)$
Mahlerian operator	$f(x^p)$	$r(x^p)$	0	$r(x^p)\partial$
Divided differences	$\frac{f(x)-f(a)}{x-a}$	$r(a)$	$\frac{r(x)-r(a)}{x-a}$	$r(a)\partial + \frac{r(x)-r(a)}{x-a}$

2.7 Pseudo-linear Equations

Let W be a vector space over $\mathbb{K}$, $\vartheta : W \to W$ a pseudo-linear map, $a_{ij} \in \mathbb{K}$, $r_i \in W$ for $1 \leq i, j \leq n$. Consider the system of equations

$$\begin{aligned} \vartheta y_1 &= a_{11}y_1 + \ldots + a_{1n}y_n + r_1 \\ &\vdots \\ \vartheta y_n &= a_{n1}y_1 + \ldots + a_{nn}y_n + r_n \end{aligned}$$

in the unknowns $y = (y_1, \ldots, y_n) \in W^n$, which we will write briefly as

$$\vartheta y = Ay + r \tag{2.11}$$

with

$$\vartheta \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} := \begin{pmatrix} \vartheta y_1 \\ \vdots \\ \vartheta y_n \end{pmatrix},$$

$A = (a_{ij})_{1 \leq i,j \leq n} \in \mathrm{Mat}(n, \mathbb{K})$ and $r = (r_1, \ldots, r_n) \in W^n$. A scalar equation (of higher order) is then an equation of the form

$$\sum_{k=0}^{m} a_k \vartheta^k z = \rho, \tag{2.12}$$

where $a_k \in \mathbb{K}$, $\rho \in W$ and $z \in W$ is unknown. We call such an equation 'scalar' because it contains only one unknown. In chapter (4) and section (5.1) we will present two algorithms that reduce the problem of solving (2.11) to the solution of equations of the form (2.12).

The concept of pseudo-linear maps captures a variety of systems of linear operator equations:

Example 2.22 *Let $W = \mathbb{K} = K((x))$ for some field K and $\vartheta = D$. Then (2.11) becomes a system of first order linear ordinary differential equations*

$$\begin{aligned} Dy_1 &= a_{11}y_1 + \ldots + a_{1n}y_n + r_1 \\ &\vdots \\ Dy_n &= a_{n1}y_1 + \ldots + a_{nn}y_n + r_n. \end{aligned}$$

□

Example 2.23 *Let $\mathbb{K} = \mathcal{R}(K)$ for some field K, $W = \mathcal{S}(K)$ and $\vartheta = E$. Then (2.11) becomes a system of first order linear difference equations*

$$\begin{aligned} Ey_1 &= a_{11}y_1 + \ldots + a_{1n}y_n + r_1 \\ &\vdots \\ Ey_n &= a_{n1}y_1 + \ldots + a_{nn}y_n + r_n. \end{aligned}$$

□

Example 2.24 *Let $\mathbb{K}$ be any field, $W = \mathbb{K}$ and $\vartheta = id_{\mathbb{K}}$. Then (2.11) becomes a system of algebraic linear equations*

$$\begin{aligned} y_1 &= a_{11}y_1 + \ldots + a_{1n}y_n + r_1 \\ &\vdots \\ y_n &= a_{n1}y_1 + \ldots + a_{nn}y_n + r_n. \end{aligned}$$

□

Example 2.25 *This example shows that the Euclidean Algorithm for Ore polynomials is useful for working with pseudo-linear equations. Consider the difference equations*

$$-\frac{1}{x}y(x) + \frac{-1-x-3x^2-x^3}{(1+x)(2+x)}y(x+1) + \frac{(-1+x)(1+x)}{2+x}y(x+2) = 0 \quad (2.13)$$

and

$$-\frac{1}{1+x}y(x) + \frac{2+x^2}{1+x}y(x+1) + (2+2x)y(x+2) = 0. \quad (2.14)$$

Let us apply the Euclidean Algorithm to the operators

$$A = -\frac{1}{x} + \frac{-1-x-3x^2-x^3}{(1+x)(2+x)}E + \frac{(-1+x)(1+x)}{2+x}E^2$$

and

$$B = -\frac{1}{1+x} + \frac{2+x^2}{1+x}E + (2+2x)\,E^2,$$

where E is the forward shift $Ex = x+1$. The loop of algorithm (2.7) is executed two times, and yields the relations

$$\begin{gathered} 0A + 1B = -\tfrac{1}{1+x} + \tfrac{2+x^2}{1+x}E + (2+2x)\,E^2, \\ 1A + \tfrac{1-x}{4+2x}B = \tfrac{1}{x} + xE, \\ \left(\tfrac{2x(2+x)}{4+5x+3x^2} + \tfrac{4(2+x)(3+x)}{12+11x+3x^2}E\right)A + \left(\tfrac{4+6x+2x^2}{4+5x+3x^2} - \tfrac{2x(2+x)}{12+11x+3x^2}E\right)B = 0. \end{gathered}$$

We read off

$$\operatorname{gcrd}(A,B) = \frac{1}{x} + xE$$

and

$$\operatorname{lclm}(A,B) = \left(\frac{2x\,(2+x)}{4+5x+3x^2} + \frac{4\,(2+x)\,(3+x)}{12+11x+3x^2}E\right)A.$$

By dividing A, B by their gcrd on the right, we find that the equations

$$\begin{gathered} \left(-1 + \frac{x-1}{x+2}E\right)\left(\frac{1}{x} + xE\right)y(x) = 0, \\ \left(\frac{x}{1+x} + 2E\right)\left(\frac{1}{x} + xE\right)y(x) = 0 \end{gathered}$$

are equivalent to (2.13) and (2.14), respectively. Furthermore, the difference equation $\operatorname{lclm}(A,B)y = 0$, *or*

$$\begin{gathered} -\tfrac{2(2+x)}{4+5x+3x^2}y(x) - \tfrac{2\left(48+64x+53x^2+37x^3+17x^4+3x^5\right)}{(4+5x+3x^2)(12+11x+3x^2)}y(x+1) \\ -\tfrac{2\left(48+152x+195x^2+119x^3+35x^4+3x^5\right)}{(4+5x+3x^2)(12+11x+3x^2)}y(x+2) + \tfrac{4x(2+x)^2}{12+11x+3x^2}y(x+3) = 0 \end{gathered}$$

is solved by all solutions of (2.13) and all solutions of (2.14).

A more sophisticated application of the Euclidean Algorithm to systems of pseudo-linear equations is the Gaussian Elimination Algorithm of chapter (4). □

There is a less obvious, but elegant way to model linear operator systems such as difference and differential systems using pseudo-linear maps, by considering pseudo-linear maps on finite dimensional vector spaces:

Let V be a vector space over $\mathbb{K}$, $\theta : V \to V$ be pseudo-linear and suppose that $\dim V = n$ is finite. This assumption makes θ amenable to techniques similar to linear algebra. Indeed, the study of pseudo-linear maps on finite

dimensional vector spaces is an area called 'pseudo-linear algebra' with origins in the 1930s ([15]).

Let $\mathcal{B} = (b_1, \ldots, b_n)$ be a basis for V. For a vector $x = \sum_{k=1}^{n} x_k b_k$ we write $(x)_{\mathcal{B}}$ for the coordinates of x w.r.t. $\mathcal{B}$:

$$x = \sum_{k=1}^{n} x_k b_k \quad \Longleftrightarrow \quad (x)_{\mathcal{B}} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \in \mathbb{K}^n.$$

We define $M_{\mathcal{B}}(\theta) = (t_{ik})_{1 \le i,k \le n}$, the matrix of θ w.r.t. $\mathcal{B}$, by

$$\theta b_k = \sum_{i=1}^{n} t_{ik} b_i.$$

Then we have

$$\begin{aligned} \theta x &= \theta \sum_{k=1}^{n} x_k b_k = \sum_{k=1}^{n} (\sigma(x_k)\theta b_k + \delta x_k \ b_k) \\ &= \sum_{k=1}^{n} \sigma(x_k) \sum_{i=1}^{n} t_{ik} b_k + \sum_{i=1}^{n} \delta x_i \ b_i = \sum_{i=1}^{n} \left(\sum_{k=1}^{n} t_{ik} \sigma(x_k) + \delta x_i \right) b_i. \end{aligned}$$

This shows that

$$(\theta x)_{\mathcal{B}} = M_{\mathcal{B}}(\theta) \sigma\left((x)_{\mathcal{B}}\right) + \delta\left((x)_{\mathcal{B}}\right), \tag{2.15}$$

where σ and δ are applied componentwise. Conversely, any n by n matrix over $\mathbb{K}$ gives rise to a pseudo-linear map via (2.15). For $\sigma = id$ and $\delta = 0$, this is familiar from linear algebra.

If $A = (a_{ik}) \in \mathrm{Mat}(n, \mathbb{K})$ is invertible, we can transform

$$\mathcal{B} = \{b_1, \ldots, b_n\}$$

into the new basis

$$A\mathcal{B} = \left\{b'_1, \ldots, b'_n\right\}$$

given by

$$b'_k = \sum_{i=1}^{n} a_{ik} b_k,$$

that is,

$$\left(\left(b'_1\right)_{\mathcal{B}}, \ldots, \left(b'_n\right)_{\mathcal{B}}\right) = A.$$

The matrix of θ associated with this new basis is then

$$T' = A^{-1} T \sigma(A) + A^{-1} \delta(A), \tag{2.16}$$

which can be shown as follows: With the notation $A^{-1} = (a_{ik}^{(-1)})$ we have $b_j = A^{-1}b'_j = \sum_{l=1}^{n} a_{lj}^{(-1)} b'_l$ and thus

$$\begin{aligned}
\theta b'_k &= \theta \sum_{i=1}^{n} a_{ik} b_i \\
&= \sum_{i=1}^{n} \sigma(a_{ik}) \theta b_i + \sum_{j=1}^{n} \delta(a_{jk}) b_j \\
&= \sum_{j=1}^{n} \left(\sum_{i=1}^{n} \sigma(a_{ik}) t_{ji} + \delta(a_{jk}) \right) b_j \\
&= \sum_{l=1}^{n} \sum_{j=1}^{n} a_{lj}^{(-1)} \left(\sum_{i=1}^{n} t_{ji} \sigma(a_{ik}) + \delta a_{jk} \right) b'_l.
\end{aligned}$$

The following example shows how a system of differential equations can be encoded by a pseudo-linear map.

Example 2.26 *Let $(\mathbb{K}, \delta)$ be a differential field (cf. example (2.2)(i))with derivation δ and $T \in \mathrm{Mat}(n, \mathbb{K})$. Then the map $\theta : \mathbb{K}^n \to \mathbb{K}^n$ given by*

$$\theta \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = T \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} + \begin{pmatrix} \delta y_1 \\ \vdots \\ \delta y_n \end{pmatrix}$$

is pseudo-linear w.r.t. $id_{\mathbb{K}}$, δ. Let $r \in \mathbb{K}^n$, then the equation

$$\theta y = r$$

is a system of differential equations. □

In the case of difference equations, some rewriting has to be done before we can assign a pseudo-linear map to a system of equations:

Example 2.27 *Let $\mathbb{K}$ be a field, E be an automorphism of $\mathbb{K}$ (we do not restrict E to the forward shift introduced in section (2.6)), $M \in \mathrm{Mat}(n, \mathbb{K})$ and $r \in \mathbb{K}^n$. We want to write the system of difference equations*

$$Ey = My + r \tag{2.17}$$

in the form

$$\theta y = f$$

for some pseudo-linear map $\theta : \mathbb{K}^n \to \mathbb{K}^n$. Applying $\sigma := E^{-1}$ to both sides of $My - Ey = -r$ gives

$$\sigma(M)\sigma(y) - y = -\sigma(r),$$

which can be rewritten as

$$(\sigma(M) - I)\,\sigma(y) + \sigma(y) - y = -\sigma(r).$$

By (ii) of example (2.2), $\delta := \sigma - id_{\mathbb{K}}$ is a σ-derivation. Summing up, if we define θ to be the mapping pseudo-linear w.r.t. σ and δ whose matrix w.r.t. the canonical basis of $\mathbb{K}^n$ is $\sigma(M) - I$, then (2.17) is equivalent to

$$\theta y = f$$

with $f = -\sigma(r)$. □

These examples show that an equation

$$\theta y = f \tag{2.18}$$

in a finite dimensional vector space encodes a system of pseudo-linear equations. We will use this term for systems of the form (2.11), too and make it clear in each situation which type of system we mean. The relation between them (in the case where σ is an automorphism) is provided by

Theorem 2.28 *Let $\mathbb{K}$ be a field, $\sigma : \mathbb{K} \to \mathbb{K}$ be an automorphism, $\delta : \mathbb{K} \to \mathbb{K}$ a σ-derivation, $\theta : \mathbb{K}^n \to \mathbb{K}^n$ be pseudo-linear w.r.t. σ and δ and $f \in \mathbb{K}^n$. Then there is a pseudo-linear map $\vartheta : \mathbb{K} \to \mathbb{K}$, a matrix $A \in \mathrm{Mat}(n, \mathbb{K})$ and a vector $r \in \mathbb{K}^n$ s.t. for all $y \in \mathbb{K}^n$*

$$\theta y = f \tag{2.19}$$

if and only if

$$\vartheta y = Ay + r.$$

In short, (2.18) is a special case of (2.11).

Proof. Let T be the matrix of θ associated with the canonical basis of $\mathbb{K}^n$. Then equation (2.19) is equivalent to

$$T\sigma(y) + \delta y = f.$$

By applying σ^{-1} on both sides, this is further equivalent to

$$\begin{aligned} & \sigma^{-1}(T)y + \sigma^{-1}\delta(y) = \sigma^{-1}(f) \\ \iff \quad & \sigma^{-1}\delta(y) = -\sigma^{-1}(T)y + \sigma^{-1}(f). \end{aligned}$$

Now for all $a, b \in \mathbb{K}$ we have

$$\delta(ab) = \delta(ba) = \sigma(b)\delta a + \delta b\ a,$$

hence

$$\sigma^{-1}\delta(ab) = \sigma^{-1}(a)\sigma^{-1}\delta(b) + \sigma^{-1}\delta(a)b,$$

i.e., $\sigma^{-1}\delta$ is a σ^{-1}-derivation. Furthermore $\vartheta := \sigma^{-1}\delta$ is a pseudo-linear map w.r.t. σ^{-1} and $\sigma^{-1}\delta$ (cf. example (2.11)), and (2.19) is equivalent to $\vartheta y = Ay + r$ with $A = -\sigma^{-1}(T)$ and $r = \sigma^{-1}(f)$. □

The main idea of this proof is mentioned in [11].

Chapter 3

Block Diagonal Decomposition

In this chapter we consider pseudo-linear equations of the type

$$\theta x = r, \tag{3.1}$$

where θ is a pseudo-linear map w.r.t. σ and δ on a vector space V of finite dimension n. Additionally, we assume that σ is an automorphism. Special cases include systems of difference or differential equations where the unknowns are in a field, as indicated by examples (2.26) and (2.27). On the other hand, Mahlerian operators are examples of pseudo-linear maps where σ is not an automorphism.

In the process of solving (3.1) a natural first step is to find a change of bases that transforms the matrix associated with θ into a simple form, e.g. a diagonal matrix. However, we cannot hope for a normal form of this type in general, because we even cannot always achieve it in the special case where θ is a linear map. In this chapter we will present an algorithm that proceeds in two steps. First it computes a basis w.r.t. which the matrix of θ is block-triangular and each block is a companion matrix. Afterwards this normal form is used to obtain higher order uncoupled equations for some of the unknowns and additional linear algebraic (i.e., without applications of operators to the unknowns) equations to determine the remaining unknowns. The algorithm, which is due to Bruno Zürcher, is a generalization of an algorithm by Danilewski [14]. Our exposition closely follows [25].

3.1 Zürcher's Algorithm

3.1.1 A Normal Form for Pseudo-linear Maps

Matrices of the type

$$\begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & & \vdots \\ \vdots & & \ddots & \ddots & \\ 0 & \dots & \dots & 0 & 1 \\ c_0 & c_1 & \dots & c_{n-2} & c_{n-1} \end{pmatrix}$$

with $c_i \in \mathbb{K}$ are called *companion matrices*. In [25], Zürcher gave an algorithmic proof of

Theorem 3.1 *Let V be a finite dimensional vector space, $\theta : V \to V$ be pseudo-linear w.r.t. σ and δ, where σ is an automorphism. Then there is a basis of V such that the matrix of θ w.r.t. to this basis is of the form*

$$\operatorname{diag}(C_1, \dots C_m),$$

where the C_i, $1 \le i \le m$, are companion matrices.

Suppose we are given a basis $\mathcal{B}$ of V, and let $T = M_{\mathcal{B}}(\theta)$ be the matrix associated with θ. Because of formula (2.16), the problem is equivalent to finding a regular matrix A such that $A^{-1}T\sigma(A) + A^{-1}\delta A$ is of the desired form. We will construct such an A as a product of certain *elementary matrices*. For each of those elementary matrices we describe the effect that the corresponding basis transformation has on T as well as the effect on the basis. For the latter, we can assume (by starting with $B = I$) that our basis is of the form $B\mathcal{B}$.

(i) For $a \in \mathbb{K}\backslash\{0\}$ and $1 \le i \le n$, let

$$D_i(a) := \begin{array}{c} \\ i \to \end{array} \begin{array}{c} \begin{matrix} & & & & \overset{i}{\downarrow} & & \end{matrix} \\ \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & & & a & & \\ & & & & & 1 & \\ & & & & & & \ddots \\ & & & & & & & 1 \end{pmatrix} \end{array}.$$

$D_i(a)$ is an invertible matrix with $D_i(a)^{-1} = D_i(a^{-1})$. If we write E_{ik} for the $n \times n$ matrix with 1 at position (i, k) and 0 elsewhere, we have

$\sigma(D_i(a)) = D_i(\sigma(a))$ and $\delta D_i(a) = \delta a\ E_{ii}$ because of corollary (2.13). By (2.16), the change of bases $\mathcal{B}' = D_i(a)\mathcal{B}$ modifies $T = (t_{ik})$ in the following way:

D1: multiply column i by $\sigma(a)$

D2: multiply line i by a^{-1}

D3: add $a^{-1}\delta a$ to the entry t_{ii}.

The effect of the basis transformation on B is given by (note $D_i(a)\,(B\mathcal{B}) = (BD_i(a))\,\mathcal{B}$):

D1′: multiply column i by a.

(ii) Let $a \in \mathbb{K}$, $1 \le i \ne k \le n$. Then

$$C_{ik}(a) := \begin{array}{c} \\ \\ i \to \end{array} \begin{array}{c} \begin{array}{c} k \\ \downarrow \end{array} \\ \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & & & \\ & a & & \ddots & \\ & & & & 1 \end{pmatrix} \end{array}$$

is an invertible matrix with $C_{ik}(a)^{-1} = C_{ik}(a^{-1})$, $\sigma(C_{ik}(a)) = C_{ik}(\sigma(a))$ and $\delta C_{ik}(a) = \delta a\ E_{ik}$. A change of bases by $C_{ik}(a)$ has the effects

C1: add $\sigma(a)$ times column i to column k

C2: add $-a$ times line k to line i

C3: add δa to the entry t_{ik}.

on T, and concerning B:

C1′: add a times column i to column k.

(iii) For $1 \le i \ne k \le n$

$$P_{ik} = \begin{array}{c} \\ i \to \\ k \to \end{array} \begin{array}{c} \begin{array}{cc} i & k \\ \downarrow & \downarrow \end{array} \\ \begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 0 & 1 & & \\ & & & & & \\ & & 1 & 0 & & \\ & & & & \ddots & \\ & & & & & 1 \end{pmatrix} \end{array}$$

is called a *permutation matrix*. Of course we have $P_{ik}^{-1} = P_{ki}$, $\sigma(P_{ik}) = P_{ik}$ and $\delta P_{ik} = 0$. A change of bases by P_{ik} induces the following operations on T:

P1: exchange column i and column k

P2: exchange line i and line k

And it modifies B by

P2′: exchange column i and column k.

After with these three types of elementary matrices, we introduce the *rotation matrix*

$$R = \begin{pmatrix} 0 & 1 & & & \dots & 0 \\ \vdots & \ddots & \ddots & & & \vdots \\ & & & \ddots & & \\ & & & & \ddots & 1 \\ 1 & & & & \dots & 0 \end{pmatrix}.$$

It satisfies $\sigma(R) = R$ and $\delta(R) = 0$ and its inverse is given by

$$R^{-1} = \begin{pmatrix} 0 & \dots & & & 1 \\ 1 & \ddots & & & \\ & \ddots & & & \\ & & & & \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \dots & & 1 & 0 \end{pmatrix}.$$

The corresponding bases change amounts to

R1: All columns are shifted by 1 to the right (column n becomes the first column)

R2: All lines are shifted by 1 downwards (line n becomes the first line)

And the effect on B is given by

R1′: All columns are shifted by 1 to the right (column n becomes the first column).

In linear algebra it is shown that any change of bases can be expressed as a product of basis changes of the types C, D, and P. $R = P_{1n} \dots P_{n-1,n}$ is introduced to simplify the notation in what follows. The proof of theorem (3.1) requires the following five lemmata:

Lemma 3.2 *Let the matrix of θ be of the form*

$$T = \begin{matrix} \\ i \to \end{matrix} \begin{pmatrix} & & & \overset{i}{\downarrow} & & & & \\ 0 & 1 & & 0 & 0 & \dots & \dots & 0 \\ \vdots & & \ddots & & \vdots & & & \vdots \\ 0 & \dots & & 1 & 0 & \dots & \dots & 0 \\ * & \dots & \dots & * & * & \dots & \dots & * \\ * & \dots & \dots & * & * & \dots & \dots & * \\ \vdots & & & \vdots & \vdots & & & \vdots \\ \vdots & & & \vdots & \vdots & & & \vdots \\ * & \dots & \dots & * & * & \dots & \dots & * \end{pmatrix} \quad \text{with} \quad i < n. \tag{3.2}$$

If there is an element $t_{il} \neq 0$ with $i < l \leq n$, then there is a basis change A s.t.

$$A^{-1}T\sigma(A) + A^{-1}\delta A = \begin{matrix} \\ i+1 \to \end{matrix} \begin{pmatrix} 0 & 1 & & & \overset{i+1}{\downarrow} & & & \\ & & & & 0 & 0 & \dots & 0 \\ \vdots & & \ddots & & & \vdots & & \vdots \\ \vdots & & & \ddots & & \vdots & & \vdots \\ 0 & \dots & & & 1 & 0 & \dots & 0 \\ * & \dots & & \dots & * & * & \dots & * \\ * & \dots & & \dots & * & * & \dots & * \\ \vdots & & & & \vdots & \vdots & & \vdots \\ * & \dots & & \dots & * & * & \dots & * \end{pmatrix},$$

i.e. we can increase the size of the companion block by 1.

Proof. We show how A can be constructed as a product of elementary matrices. To keep notation simple, the associated matrices of θ that occur in the intermediate steps are denoted again by $T = (t_{ik})$. First, by the basis change $P_{i+1,l}$, $t_{i,i+1}$ becomes nonzero. The remaining entries of the affected columns $i+1$ and l are either 0 (rows $1, \dots, i-1$) or not of interest (rows $i+1, \dots, n$). P2 does not change the ordered part of T either.

Now we can perform the basis change $D_{i+1}(\sigma^{-1}(t_{i,i+1}^{-1}))$. D1 sets $t_{i,i+1}$ to 1, and D2 and D3 do not modify lines $1, \dots, i$.

What remains to do is to set t_{ik}, $1 \leq k \leq n$, $k \neq i+1$, to 0. Suppose we have done this up to $k < m$:

$$T = \begin{array}{c} \\ i \rightarrow \end{array} \begin{array}{c} \begin{array}{ccccccccccc} & & & & m & & & i & i+1 & & \\ & & & & \downarrow & & & \downarrow & \downarrow & & \end{array} \\ \begin{pmatrix} 0 & 1 & & & & \dots & 0 & 0 & \dots & \dots & 0 \\ \vdots & \ddots & \ddots & & & & \vdots & \vdots & & & \vdots \\ & & & & & & & & & & \\ & & & & & \ddots & & \vdots & & & \vdots \\ \vdots & & & & & & 1 & 0 & \dots & \dots & 0 \\ 0 & \dots & 0 & * & \dots & * & 1 & * & \dots & \dots & * \\ * & \dots & & & \dots & * & * & \dots & \dots & \dots & * \\ \vdots & & & & & \vdots & \vdots & & & & \vdots \\ \vdots & & & & & \vdots & \vdots & & & & \vdots \\ * & \dots & & & \dots & * & * & \dots & \dots & \dots & * \end{pmatrix} \end{array}.$$

The basis change $C_{i+1,m}(\sigma^{-1}(-t_{im}))$ sets t_{im} to 0 by operation C1. It is easy to check that C1, C2 and C3 do not change the ordered part of T. □

Lemma 3.3 *Let the matrix of θ be of the form*

$$T = \begin{array}{c} \\ i \rightarrow \end{array} \begin{array}{c} \begin{array}{ccccccccc} & & & & i & & & & \\ & & & & \downarrow & & & & \end{array} \\ \begin{pmatrix} 0 & 1 & & & 0 & 0 & \dots & \dots & 0 \\ \vdots & & \ddots & & & \vdots & & & \vdots \\ 0 & \dots & & & 1 & 0 & \dots & \dots & 0 \\ * & \dots & \dots & & * & 0 & \dots & \dots & 0 \\ * & \dots & \dots & & * & * & \dots & \dots & * \\ \vdots & & & & \vdots & \vdots & & & \vdots \\ * & \dots & \dots & & * & * & \dots & \dots & * \end{pmatrix} \end{array}.$$

Then there is a basis change A s.t.

$$A^{-1}T\sigma(A) + A^{-1}\delta A = \begin{array}{c} \\ i \rightarrow \end{array} \begin{array}{c} \begin{array}{ccccccc} & & & & & i & \\ & & & & & \downarrow & \end{array} \\ \begin{pmatrix} 0 & 1 & & 0 & 0 & \dots & 0 \\ \vdots & & \ddots & & \vdots & & \vdots \\ 0 & \dots & & 1 & 0 & \dots & 0 \\ * & \dots & \dots & * & 0 & \dots & 0 \\ * & 0 & \dots & 0 & * & \dots & * \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ * & 0 & \dots & 0 & * & \dots & * \end{pmatrix} \end{array}. \qquad (3.3)$$

Instead of enlarging the companion block, we erase all columns below it except the first one.

Proof. Again, we proceed inductively to delete the specified entries. Let $t_{mu} = 0$ for $i < m \leq N$ and $l < u \leq i$ (where $2 \leq l \leq i$) as well as $t_{ml} = 0$ for $i < m < k$ (where $i < k \leq n$). So T is of the form

$$T = \begin{array}{c} \\ \\ l \to \\ \\ \\ \\ i \to \\ \\ \\ \\ k \to \\ \\ \\ \end{array} \begin{array}{c} \begin{array}{cccccccccccc} & & & & l\downarrow & & & & i\downarrow & & & k\downarrow & & \end{array} \\ \begin{pmatrix} 0 & 1 & & & & \dots & 0 & 0 & \dots & \dots & 0 \\ \vdots & & \ddots & & & & \vdots & \vdots & & & \vdots \\ & & & & & & & & & & \\ & & & & & & & & & & \\ \vdots & & & & & \ddots & & & & & \\ 0 & \dots & & & & & 1 & \vdots & & & \vdots \\ * & \dots & & & & \dots & * & 0 & \dots & \dots & 0 \\ * & \dots & * & 0 & \dots & \dots & 0 & * & \dots & \dots & * \\ \vdots & & \vdots & \vdots & & & \vdots & \vdots & & & \vdots \\ & & \vdots & 0 & 0 & & & & & & \\ & & * & * & \vdots & & & & & & \\ \vdots & & & \vdots & \vdots & & \vdots & \vdots & & & \vdots \\ * & \dots & \dots & * & 0 & \dots & 0 & * & \dots & \dots & * \end{pmatrix} \end{array}.$$

Operation C1 of the basis change $C_{k,l-1}(t_{kl})$ adds a multiple of column k to column $l-1$. The elements with row indices $1, \dots, i$ are not changed, because column k is zero there, and the other elements of column i are irrelevant. C2 sets t_{kl} to zero without destroying the order of T. C3 does no harm. □

The following lemma is needed for the proof of lemma (3.5).

Lemma 3.4 *Let the matrix of θ be of the form*

$$T = \begin{array}{c} \\ \\ k \to \\ k+1 \to \\ \\ \\ i+1 \to \\ \\ \\ \end{array} \begin{array}{c} \begin{array}{cccccccccc} & & & k+1\downarrow & & & & i+1\downarrow & & \end{array} \\ \begin{pmatrix} 0 & 1 & & & & \dots & 0 & 0 & \dots & 0 \\ & & \ddots & & & & \vdots & & & \\ * & \dots & * & 1 & & & & * & \dots & * \\ * & \dots & \dots & * & 1 & & & * & \dots & * \\ 0 & \dots & & \dots & 0 & \ddots & & & & \\ & & & & & & 1 & \vdots & & \vdots \\ * & \dots & & & & \dots & * & 0 & \dots & 0 \\ * & 0 & \dots & & & \dots & 0 & * & \dots & * \\ \vdots & \vdots & & & & & \vdots & \vdots & & \vdots \\ * & 0 & \dots & & & \dots & 0 & * & \dots & * \end{pmatrix} \end{array}.$$

Then there is a basis change A s.t.

$$A^{-1}T\sigma(A)+A^{-1}\delta A = \begin{array}{c} \\ \\ k\to \\ k+1\to \\ \\ \\ i+1\to \\ \\ \\ \end{array} \begin{array}{c} \overset{}{} \\ \overset{}{} \\ \overset{}{} \\ \overset{}{} \\ \overset{}{} \\ \overset{}{} \\ \overset{}{} \\ \overset{}{} \\ \overset{}{} \end{array} \begin{pmatrix} 0 & 1 & & \overset{k+1}{\downarrow} & & \dots & \overset{i+1}{\downarrow} & 0 & \dots & 0 \\ & & \ddots & & & & \vdots & & & \\ 0 & \dots & 0 & 1 & & & & 0 & \dots & 0 \\ * & \dots & \dots & * & 1 & & & * & \dots & * \\ 0 & \dots & & \dots & 0 & \ddots & & & & \\ & & & & & & 1 & \vdots & & \vdots \\ * & \dots & & & & \dots & * & 0 & \dots & 0 \\ * & 0 & \dots & & & \dots & 0 & * & \dots & * \\ \vdots & \vdots & & & & & \vdots & \vdots & & \vdots \\ * & 0 & \dots & & & \dots & 0 & * & \dots & * \end{pmatrix}.$$

That is, the disorder in line k can be shifted to line $k+1$.

Proof. We have to delete t_{kl} for $l = 1, \dots, k+1, i+2, \dots, n$. Let l be one of these row indices. The basis change $C_{k+1,l}(\sigma^{-1}(-t_{kl}))$ does exactly what we want: C1 erases t_{kl} and modifies $t_{k+1,l}$, and C2 and C3 only affect the irrelevant entries of line $k+1$. □

Lemma 3.5 *Let the associated matrix of θ be of the form*

$$T = \begin{array}{c} i\to \end{array} \begin{pmatrix} 0 & 1 & & 0 & \overset{i}{\downarrow} & \dots & 0 \\ \vdots & & \ddots & & \vdots & & \vdots \\ 0 & \dots & & 1 & 0 & \dots & 0 \\ * & \dots & \dots & * & 0 & \dots & 0 \\ * & 0 & \dots & 0 & * & \dots & * \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ * & 0 & \dots & 0 & * & \dots & * \end{pmatrix} \quad \text{where} \quad i < n.$$

If there is an element $t_{k1} \neq 0$ for some $k = i+1, \dots, n$, then there is a basis change A s.t.

$$A^{-1}T\sigma(A) + A^{-1}\delta A = \begin{array}{c} i+1\to \end{array} \begin{pmatrix} 0 & 1 & & & \overset{i+1}{\downarrow} & 0 & \dots & 0 \\ \vdots & & \ddots & & & \vdots & & \vdots \\ \vdots & & & \ddots & & \vdots & & \vdots \\ 0 & \dots & & & 1 & 0 & \dots & 0 \\ * & \dots & & \dots & * & * & \dots & * \\ * & 0 & \dots & \dots & 0 & * & \dots & * \\ \vdots & & & & \vdots & \vdots & & \vdots \\ * & 0 & \dots & \dots & 0 & * & \dots & * \end{pmatrix}.$$

In other words, we can increase the size of the companion block, while maintaining the zero block below it.

Proof. First we show that we may assume $t_{k1} = 0$ for $i < k < n$ and t_{n1}=1. Namely, let $t_{k1} \neq 0$ for some $i < k < n$. The basis change P_{kn} causes $t_{n1} \neq 0$ without destroying the form of T. Now the basis change $D_n(t_{n1})$ is well-defined, D1,D2,D3 do not affect the order of T, and D2 sets $t_{n1} = 1$. If now there is still an element $t_{k1} \neq 0$ for some $i < k < n$, it can be deleted by $C_{kn}(t_{k1})$. Apart from this deletion, C1,C2,C3 just affect the irrelevant lower right block of T. So we have shown that the assumption is allowed.

A basis change by R leads from

$$T = \begin{array}{c} \\ i \rightarrow \end{array} \begin{array}{c} \begin{array}{cccccccc} & & & & \overset{i}{\downarrow} & & & \end{array} \\ \begin{pmatrix} 0 & 1 & & 0 & 0 & \dots & 0 & 0 \\ \vdots & & \ddots & & \vdots & & \vdots & \vdots \\ 0 & \dots & & 1 & \vdots & & \vdots & \vdots \\ * & \dots & \dots & * & 0 & \dots & 0 & 0 \\ 0 & \dots & \dots & 0 & * & \dots & * & * \\ \vdots & & & \vdots & \vdots & & \vdots & \vdots \\ 0 & \dots & \dots & 0 & * & \dots & * & * \\ 1 & 0 & \dots & 0 & * & \dots & * & * \end{pmatrix} \end{array}$$

to

$$R^{-1}TR = \begin{array}{c} \\ i+1 \rightarrow \end{array} \begin{array}{c} \begin{array}{cccccccc} & & & & \overset{i+1}{\downarrow} & & & \end{array} \\ \begin{pmatrix} * & 1 & 0 & \dots & 0 & * & \dots & * \\ 0 & 0 & 1 & & 0 & 0 & \dots & 0 \\ \vdots & \vdots & & \ddots & & \vdots & & \vdots \\ \vdots & 0 & \dots & & 1 & \vdots & & \vdots \\ 0 & * & \dots & \dots & * & 0 & \dots & 0 \\ * & 0 & \dots & \dots & 0 & * & \dots & * \\ \vdots & \vdots & & & \vdots & \vdots & & \vdots \\ * & 0 & \dots & \dots & 0 & * & \dots & * \end{pmatrix} \end{array}.$$

Except the first line the $(i + 1) \times (i + 1)$ upper left block is in companion form. The disorder in the first line is now shifted downwards by applying lemma (3.4) iteratively, until the desired form is attained. □

Before proving theorem (3.1) we make sure that a basis change on a block-diagonal matrix T behaves as expected:

Lemma 3.6 *Let T_1, T_2 be matrices with entries from $\mathbb{K}$ of sizes $n_1 \times n_1$ and $n_2 \times n_2$, respectively. Let further $\theta : \mathbb{K}^{n_1+n_2} \rightarrow \mathbb{K}^{n_1+n_2}$ be the pseudo-linear map whose matrix w.r.t. the canonical basis is $\left(\begin{smallmatrix} T_1 & 0 \\ 0 & T_2 \end{smallmatrix}\right)$ and A be an*

invertible $n_2 \times n_2$ matrix. Then the basis change $\left(\begin{smallmatrix} I & 0 \\ 0 & A \end{smallmatrix}\right)$ turns the matrix of θ into

$$\begin{pmatrix} T_1 & 0 \\ 0 & A^{-1}T_2\sigma(A) + A^{-1}\delta A \end{pmatrix}.$$

Proof. Because of formula (2.16), the matrix that we seek is

$$\begin{pmatrix} I & 0 \\ 0 & A^{-1} \end{pmatrix} \begin{pmatrix} T_1 & 0 \\ 0 & T_2 \end{pmatrix} \sigma \left(\begin{pmatrix} I & 0 \\ 0 & A \end{pmatrix} \right) + \begin{pmatrix} I & 0 \\ 0 & A^{-1} \end{pmatrix} \delta \begin{pmatrix} I & 0 \\ 0 & A \end{pmatrix}$$
$$= \begin{pmatrix} T_1 & 0 \\ 0 & A^{-1}T_2\sigma(A) \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & A^{-1}\delta A \end{pmatrix} = \begin{pmatrix} T_1 & 0 \\ 0 & A^{-1}T_2\sigma(A) + A^{-1}\delta A \end{pmatrix},$$

where $\sigma(I) = I$ and $\delta I = 0$ follow from corollary (2.13). □

Proof of theorem (3.1). Induction on n. For $n = 1$ there is nothing to show, because any 1×1 matrix is in companion form.

Suppose the assertion of the theorem holds for $1, \ldots, n-1$. By taking $i = 1$ if necessary, T is of the form (3.2). With a second induction on i, the size of the companion block, we show that we can either increase this block by 1 or split off a direct factor of size i from T. Therefore, let T be of the form (3.2) for some i. If $i = n$, we are done. If $i < n$ and there is an element $t_{ik} \neq 0$ for some $i < k \leq n$, we can increase the size of the companion block to $i + 1$ by lemma (3.2). If, on the other hand, all these entries are zero, we apply lemma (3.3) to obtain a matrix of the form (3.3).

If now there is a nonzero entry among t_{1k}, $i < k \leq n$, lemma (3.5) increases the companion block by 1. If all those elements are 0, the companion block is a direct factor of T, and we can apply the induction hypothesis (on n) on the lower right block. The basis changes that we need to bring the lower right block into the desired form will not interfere with the upper left companion block because of lemma (3.6). □

This proof gives rise to the following algorithm to find the blockdiagonal normal form:

```
rationalForm(T,σ,δ)
   n := size(T)
   i := 1
   while i<n repeat
     if t_{i,i+1} ≠ 0 or ... or t_{i,n} ≠ 0 then
       expand companion block of T by lemma (3.2)
       i := i + 1
     else
```

```
        apply lemma (3.3) to T
        if t_{i+1,1} ≠ 0 or ... or t_{n,1} ≠ 0 then
          expand companion block of T by lemma (3.5)
            i := i + 1
        else (* T = diag(C, T') where C is a companion matrix *)
          R:=rationalForm(T',σ,δ)
          return diag(C,R)
        end
      end
    end
    return T
end
```

In order to compute the corresponding basis change, the elementary matrices from the proofs of the preceding lemmas are multiplied together (from the right). However, in an efficient implementation, we will not multiply the full matrices T or B, the latter being the current basis change, by elementary matrices, but we will just update the matrix entries which are affected by the current step. Furthermore, we will not call the procedure recursively, but instead introduce a variable `i0` that is 1 plus the size of the left upper part of T that is already in blockdiagonal form. `i0` is initialized with 1 and updated whenever we split off a direct factor of T. Row and column operations are performed only on elements with row and column indices greater than or equal to `i0`. Then the algorithm may take the following explicit form:

Algorithm 3.7 `rationalForm` *by Bruno Zürcher*

```
rationalForm(T, σ, δ)
     n := Size(T)
     i0 := 1; i := 1
     B := identity matrix of size n
     while i < n
         j := i+1
         while j ≤ n and t_{ij} = 0
             j := j + 1
         if j ≤ n then
             transformLemma2(T,i0,i,j,σ,δ,B)
             i := i + 1
         else
             transformLemma3(T,i0,i,σ,δ,B)
             i1 := i + 1
             while i1 ≤ n and t_{i1,i0} = 0
                 i1 := i1 + 1
             if i1 ≤ n then
```

```
                transformLemma5(T,i0,i,i1,σ,δ,B)
                i := i + 1
            else
                i := i + 1
                i0 := i
            end
        end
    end
    return T,B
end


transformP(T,i0,i,k,B)
    n := Size(T)
    for j := i0 to n
        s := t_ji; t_ji := t_jk; t_jk := s
        s := b_ji; b_ji := b_jk; b_jk := s
    end
    for j := i0 to n
        s := t_ij; t_ij := t_kj; t_kj := s
    end
end


transformR(T,i0,B)
    n := Size(T)
    for i := i0 to n
        c_i := t_in
    for i := i0 to n
        for j := n downto i0+1
            t_ij := t_i,j−1
    for i := i0 to n
        t_i,i0 := c_i

    for i := i0 to n
        c_i := t_ni
    for j := i0 to n
        for i := n downto i0+1
            t_ij := t_i−1,j
    for i := i0 to n
        t_i0,i := c_i

    for i := i0 to n
```

```
            c_i := b_in
        for j := n downto i0+1
            for i := i0 to n
                b_ij := b_i,j−1
        for i := i0 to n
            b_i,i0 := c_i
    end


    transformLemma2(T,i0,i,l,σ,δ,B)
        n := Size(T)
        transformP(T,i0,i+1,l,B)

        a := σ^−1(t^−1_i,i+1)
        for j := i to n
            t_j,i+1 := t_j,i+1 * σ(a)      (* D1 *)
        for j := i0 to n
            t_i+1,j := t_i+1,j / a      (* D2 *)
        t_i+1,i+1 := t_i+1,i+1 + δ(a) / a      (* D3 *)
        for j := i0 to n
            b_j,i+1 := a * b_j,i+1      (* basis change *)

        for k := i0 to i
            a := σ^−1(-t_ik)
            for j := i to n
                t_jk := t_jk + σ(a) * t_j,i+1      (* C1 *)
            if k < i
                t_i+1,k+1 := t_i+1,k+1 - a      (* C2 *)
            else
                for j := i to n
                    t_i+1,j := t_i+1,j - a * t_ij
            t_i+1,k := t_i+1,k + δ(a)      (* C3 *)
            for j := i0 to n
                b_jk := b_jk + a * b_j,i+1      (* basis change *)
        end

        for k := i+2 to n
            a := σ^−1(-t_ik)
            for j := i to n
                t_jk := t_jk + σ(a) * t_j,i+1      (* C1 *)
            for j := i0 to n
                t_i+1,j := t_i+1,j - a * t_kj      (* C2 *)
            t_i+1,k := t_i+1,k + δ(a)      (* C3 *)
```

```
            for j := i0 to n
                b_jk := b_jk + a * b_j,i+1      (* basis change *)
        end
    end


    transformLemma3(T,i0,i,σ,δ,B)
        n := Size(T)
        for l := i downto i0+1
            for k := i+1 to n
                a := t_kl
                for j := i+1 to n
                    t_j,l−1 := t_j,l−1 + σ(a) * t_jk     (* C1 *)
                t_kl := 0     (* C2 *)
                t_k,l−1 := t_k,l−1 + δ(a)     (* C3 *)
                for j := i0 to n
                  b_j,l−1 := b_j,l−1 + a * b_jk     (* basis change *)
            end
        end
    end


    transformLemma4(T,i0,i,k,σ,δ,B)
        n := Size(T)
        for l := i0 to k
            a := σ^−1(-t_kl)
            t_kl := 0     (* C1 *)
            t_k+1,l := t_k+1,l + σ(a) * t_k+1,k+1
            if k < i
                t_i+1,l := t_i+1,l + σ(a) * t_i+1,k+1
            if l < k
                t_k+1,l+1 := t_k+1,l+1 - a     (* C2 *)
            else
                for j := i0 to k+1
                    t_k+1,j := t_k+1,j - a * t_kj
                for j := i+2 to n
                    t_k+1,j := t_k+1,j - a * t_kj
            end
            t_k+1,l := t_k+1,l + δ(a)     (* C3 *)
            for j := i0 to n
                b_jl := b_jl + a * b_j,k+1     (* basis change *)
        end
```

```
        for l := i+2 to n
            a := σ^{-1}(-t_{k,l})
            t_{kl} := 0     (* C1 *)
            t_{k+1,l} := t_{k+1,l} + σ(a) * t_{k+1,k+1}
            if k < i
                t_{i+1,l} := t_{i+1,l} + σ(a) * t_{i+1,k+1}
            t_{k+1,i0} := t_{k+1,i0} - a * t_{l,i0}     (* C2 *)
            for j := i+2 to n
                t_{k+1,j} := t_{k+1,j} - a * t_{lj}
            t_{k+1,l} := t_{k+1,l} + δ(a)     (* C3 *)
            for j := i0 to n
                b_{jl} := b_{jl} + a * b_{j,k+1}     (* basis change *)
        end
    end


    transformLemma5(T,i0,i,k,σ,δ,B)
        n := Size(T)
        transformP(T,i0,k,n,B)

        a := t_{n,i0}
        for j := i+1 to n
            t_{jn} := σ(a) * t_{jn}     (* D1 *)
        t_{n,i0} := 1     (* D2 *)
        for j := i+1 to n
            t_{nj} := t_{nj} / a
        t_{nn} := t_{nn} + δ(a) / a     (* D3 *)
        for j := i0 to n
            b_{jn} := a * b_{jn}     (* basis change *)

        for l := i+1 to n
            if t_{l,i0} != 0
                a := t_{l,i0}
                for j := i+1 to n
                    t_{jn} := t_{jn} + σ(a) * t_{jl}     (* C1 *)
                t_{l,i0} := t_{l,i0} - a     (* C2 *)
                for j := i+1 to n
                    t_{lj} := t_{lj} - a * t_{nj}
                t_{ln} := t_{ln} + δ(a)     (* C3 *)
                for j := i0 to n
                    b_{jn} := b_{jn} + a * b_{jl}     (* basis change *)
            end
        end
```

```
        transformR(T,i0,B)
        for j := i0 to i
            transformLemma4(T,i0,i,j,σ,δ,B)
    end
```
□

3.1.2 Deduction of Scalar Equations

After we have transformed a system of Ore operator equations into an equivalent system with a companion matrix, we can deduce higher order scalar equations. In the setting of Zürcher's algorithm it is not immediately clear what we mean by a scalar equation. In the differential case, it is an equation of the form

$$\sum_{i=0}^{m} c_i D^i y = \rho, \quad c_i, \rho \in \mathbb{K},$$

while in the difference case we desire equations of the form

$$\sum_{i=0}^{m} c_i E^i y = \rho, \quad c_i, \rho \in \mathbb{K}.$$

The point is that the operator we are interested in can be either σ or δ. We will discuss the computation of uncoupled scalar equations in the two important special cases listed above. In [25] a more general approach is given. There it is shown how to turn the normal form computed by algorithm (3.7) into uncoupled equations of the form

$$\sum_{i=0}^{m} c_i \vartheta_\kappa^i y = \rho, \quad c_i, \rho \in \mathbb{K},$$

where $\kappa \in \mathbb{K}$ is an arbitrary parameter and $\vartheta_\kappa : \mathbb{K} \to \mathbb{K}$ is the pseudo-linear map w.r.t. σ^{-1}, $-\sigma^{-1}\delta - \kappa\,(\sigma - id)$

$$\vartheta_\kappa = \kappa\sigma^{-1} - \sigma^{-1}\delta - \kappa\left(\sigma^{-1} - id\right).$$

This covers both of the two special cases above: In the differential case $\sigma = id$, $\delta = -D$ (of course, we could also work with $\delta = D$) we set $\kappa = 0$ to obtain $\vartheta_0 = D$. In the difference case we set $\sigma = E^{-1}$, $\delta = E^{-1} - id$ (cf. example (2.27) and 'Difference Equations' below) and $\kappa = 1$, which yields $\vartheta_1 = E$.

Differential Equations

Let $(\mathbb{K}, D)$ be a differential field, T an $n \times n$ matrix with entries in $\mathbb{K}$ and $v \in \mathbb{K}^n$. Consider the system of differential equations

$$Dy = Ty + v. \tag{3.4}$$

We set $\sigma = id_{\mathbb{K}}$, $\delta = -D$ and $\theta : \mathbb{K}^n \to \mathbb{K}^n$ the pseudo-linear map whose matrix w.r.t. the canonical basis is T, that is,

$$\theta y = Ty + \delta y.$$

If we apply Zürcher's algorithm on θ and assume w.l.o.g. that it returns only one companion block, we get a companion matrix C and an invertible matrix A s.t. $\theta z = Cz + \delta z$ with $z = A^{-1}y$. Upon setting $w = A^{-1}v$, $y \in \mathbb{K}^n$ solves (3.4) if and only if $z = A^{-1}y$ solves

$$Dz = Cz + w. \tag{3.5}$$

This system is of the form

$$\begin{aligned} Dz_1 &= z_2 + w_1 \\ &\vdots \\ Dz_{n-1} &= z_n + w_{n-1} \\ Dz_n &= \sum_{i=0}^{n-1} c_i z_{i+1} + w_n. \end{aligned} \tag{3.6}$$

From this we get $z_2 = Dz_1 - w_1$, $z_3 = Dz_2 - w_3 = D^2 z_1 - Dw_1 - w_2$, and, inductively,

$$z_{i+1} = z_1^{(i)} - \sum_{j=1}^{i} w_j^{(i-j)} \quad \text{for} \quad 1 \le i < n.$$

Plugging this into the last equation of (3.6) yields

$$z_1^{(n)} - \sum_{j=1}^{n-1} w_j^{(n-j)} = \sum_{i=0}^{n-1} c_i z_1^{(i)} - \sum_{i=0}^{n-1} c_i \sum_{j=1}^{i} w_j^{(i-j)} + w_n,$$

which is a scalar differential equation for z_1. If this equation can be solved, the other z_i are computed from (3.6), and the original variables y_i by $y = Az$.

Difference Equations

Let

$$Ey = My + v \tag{3.7}$$

be a system of difference equations, where M is an $n \times n$ matrix with entries from $\mathbb{K}$ and $v \in \mathbb{K}^n$. Once again, we do not restrict E to the forward shift introduced in section (2.6). Special cases include linear algebraic systems ($E = id_{\mathbb{K}}$) and ordinary systems of difference equations with rational function coefficients ($\mathbb{K} = \mathcal{R}(K)$, $Ex = x + 1$).

Let $\sigma = E^{-1}$, $\delta = \sigma - id_{\mathbb{K}}$ and $\theta : \mathbb{K}^n \to \mathbb{K}^n$ be the pseudo-linear map whose matrix w.r.t. the canonical basis of $\mathbb{K}^n$ is $\sigma(M) - I$. Then we have

$$\begin{aligned} & Ey = My + v \\ \iff \quad & \theta y = -\sigma(v). \end{aligned}$$

(cf. example (2.27).) Applying Zürcher's algorithm to θ yields, assuming w.l.o.g. that it returns only one companion block, a companion matrix C and an invertible matrix A s.t.

$$\theta z = C\sigma(z) + \delta z, \quad \text{where} \quad z = A^{-1}y.$$

Hence (3.7) is equivalent to

$$\begin{aligned} & C\sigma(z) + \delta z = -A^{-1}\sigma(v) \\ \iff \quad & E(C)z + z - Ez = -E(A^{-1})v \\ \iff \quad & Ez = (E(C) + I)\, z + w, \end{aligned}$$

where $w = E(A^{-1})v$. This system is of the form

$$\begin{aligned} Ez_1 &= z_1 + z_2 + w_1 \\ &\vdots \\ Ez_{n-1} &= z_{n-1} + z_n + w_{n-1} \\ Ez_n &= \sum_{i=0}^{n-1} E(c_i)z_{i+1} + z_n + w_n, \end{aligned} \tag{3.8}$$

which, by using the difference operator $\Delta = E - id_{\mathbb{K}}$, can be written as

$$\begin{aligned} \Delta z_1 &= z_2 + w_1 \\ &\vdots \\ \Delta z_{n-1} &= z_n + w_{n-1} \\ \Delta z_n &= \sum_{i=0}^{n-1} E(c_i)z_{i+1} + w_n. \end{aligned} \tag{3.9}$$

This implies

$$z_{i+1} = \Delta^i z_1 - \sum_{j=1}^{i} \Delta^{i-j} w_j \quad \text{for} \quad 1 \leq i < n. \tag{3.10}$$

In particular, if we take $i = n - 1$ and apply Δ on both sides, we obtain

$$\Delta z_n = \Delta^n z_1 - \sum_{j=1}^{n-1} \Delta^{n-j} w_j. \tag{3.11}$$

Inserting (3.10) and (3.11) into the last equation of (3.9) yields the uncoupled difference equation

$$\Delta^n z_1 - \sum_{j=1}^{n-1} \Delta^{n-j} w_j = \sum_{i=0}^{n-1} E(c_i) \left(\Delta^i z_1 - \sum_{j=1}^{i} \Delta^{i-j} w_j \right) + w_n$$

$$\iff \sum_{i=0}^{n-1} E(c_i) \Delta^i z_1 - \Delta^n z_1 = \sum_{i=0}^{n-1} E(c_i) \sum_{j=1}^{i} \Delta^{i-j} w_j - \sum_{j=1}^{n-1} \Delta^{n-j} w_j - w_n$$

for z_1.

3.1.3 Complexity

Now that we made the computations that Zürcher's uncoupling algorithm performs explicit, it is straightforward to count the number of additions and multiplications in $\mathbb{K}$ that it performs. We just consider algorithm (3.7), because it is readily checked that the second step of Zürcher's algorithm, i.e., the deduction of the scalar equations, requires $O(n^2)$ arithmetical operations in $\mathbb{K}$, which is asymptotically less than the complexity of algorithm (3.7), as we will see.

To begin with, the number of multiplications in the procedure `transformLemma2`, which we will abbreviate by $\mathrm{L2}_*(n, i_0, i)$, is

$$\begin{aligned} \mathrm{L2}_*(n, i_0, i) &= (n-i+1) + (n-i_0+1) + 1 + (n-i_0+1) \\ &\quad + (i-i_0)\left((n-i+1) + (n-i_0+1)\right) \\ &\quad + \left((n-i+1) + (n-i+1) + (n-i_0+1)\right) \\ &\quad + (n-i-1)\left((n-i+1) + (n-i_0+1) + (n-i_0+1)\right). \end{aligned}$$

If we restrict i_0 to the case $i_0 = 1$ and count only the leading terms this simplifies to

$$\begin{aligned} \mathrm{L2}_*(n, 1, i) &= i\,(2n-i) + (n-i)\,(3n-i) + O(n) \\ &= 3n^2 - 2in + O(n). \end{aligned}$$

For the number of additions we find

$$\begin{aligned} \mathrm{L2}_+(n, i_0, i) &= 1 + (i-i_0)\left((n-i+1) + 1 + 1 + (n-i_0+1)\right) \\ &\quad + \left((n-i+1) + (n-i+1) + 1 + (n-i_0+1)\right) \\ &\quad + (n-i-1)\left((n-i+1) + (n-i_0+1) + 1 + (n-i_0+1)\right), \end{aligned}$$

and

$$\begin{aligned} \mathrm{L2}_+(n, 1, i) &= i\,(2n-i) + (n-i)\,(3n-i) + O(n) \\ &= 3n^2 - 2in + O(n). \end{aligned}$$

This allows to analyze the complexity of algorithm (3.7) in an important special case:

Theorem 3.8 *(nondegenerate case) If the first* `if` *condition in algorithm (3.7) is always satisfied throughout the execution, then the algorithm needs*

$$2n^3 + O(n^2)$$

multiplications in $\mathbb{K}$*. The same is true for the number of additions in* $\mathbb{K}$*.*

Proof: The i-th pass of the outer loop requires $\mathrm{L2}_*(n, 1, i)$ multiplications, hence the total number is given by

$$\begin{aligned}\sum_{i=1}^{n-1} \mathrm{L2}_*(n,1,i) &= (n-1)\,3n^2 - 2n\sum_{i=1}^{n-1} i + O(n^2) \\ &= 2n^3 + O(n^2).\end{aligned}$$

We have seen above that the leading (i.e. quadratic) terms of $\mathrm{L2}_+(n, 1, i)$ are the same as those of $\mathrm{L2}_*(n, 1, i)$, so the result holds for plus as well. □

We continue by counting the operations of the remaining procedures, enabling us to analyze the complexity in the worst case. For `transformLemma3`, we have

$$\begin{aligned}\mathrm{L3}_*(n,i_0,i) &= (i-i_0)\,(n-i)\,((n-i)+(n-i_0+1))\,, \\ \mathrm{L3}_*(n,1,i) &= i\,(n-i)\,(2n-i) + O(n^2) \\ &= 2in^2 - 3i^2n + i^3 + O(n^2).\end{aligned}$$

$$\begin{aligned}\mathrm{L3}_+(n,i_0,i) &= (i-i_0)\,(n-i)\,((n-i)+1+(n-i_0+1))\,, \\ \mathrm{L3}_+(n,1,i) &= i\,(n-i)\,(2n-i) + O(n^2) \\ &= 2in^2 - 3i^2n + i^3 + O(n^2).\end{aligned}$$

For `transformLemma4`, we give the number of operations for $k < i$. If $k = i$, there is one more multiplication and addition in each pass of the first `for` loop, which do not contribute to the leading terms.

$$\begin{aligned}\mathrm{L4}_*(n,i_0,i,k) &= (k-i_0)\,(1+1+(n-i_0+1)) \\ &\quad +(1+1+(k-i_0+2)+(n-i-1)+(n-i_0+1)) \\ &\quad +(n-i-1)\,(1+1+1+(n-i-1)+(n-i_0+1))\,, \\ \mathrm{L4}_*(n,1,i,k) &= kn + (n-i)\,(2n-i) + O(n) \\ &= 2n^2 - 3in + kn + i^2 + O(n).\end{aligned}$$

$$\begin{aligned}\mathrm{L4}_+(n,i_0,i,k) &= (k-i_0)\,(1+1+1+1+(n-i_0+1)) \\ &\quad +(1+1+(k-i_0+2)+(n-i-1)+1+(n-i_0+1)) \\ &\quad +(n-i-1)\,(1+1+1+(n-i-1)+1+(n-i_0+1))\,, \\ \mathrm{L4}_+(n,1,i,k) &= kn + (n-i)\,(2n-i) + O(n) \\ &= 2n^2 - 3in + kn + i^2 + O(n).\end{aligned}$$

In the procedure `transformLemma5` some operations are only performed if certain matrix elements are nonzero. We give the number of operations in the worst case.

$$\begin{aligned}
\mathrm{L5}_*(n,i_0,i) &= (n-i)+(n-i)+1+(n-i_0+1) \\
&\quad +(n-i)\left((n-i)+(n-i)+(n-i_0+1)\right) \\
&\quad +\sum_{k=i_0}^{i}\mathrm{L4}_*(n,i_0,i,k), \\
\mathrm{L5}_*(n,1,i) &= \sum_{k=1}^{i}\mathrm{L4}_*(n,1,i,k)+O(n^2) \\
&= 2in^2-\frac{5}{2}i^2n+i^3+O(n^2).
\end{aligned}$$

$$\begin{aligned}
\mathrm{L5}_+(n,i_0,i) &= 1+(n-i)\left((n-i)+1+(n-i)+1+(n-i_0+1)\right) \\
&\quad +\sum_{k=i_0}^{i}\mathrm{L4}_+(n,i_0,i,k), \\
\mathrm{L5}_+(n,1,i) &= 2in^2-\frac{5}{2}i^2n+i^3+O(n^2).
\end{aligned}$$

Putting these results together, we obtain

Theorem 3.9 *(worst case) In the worst case, algorithm (3.7) performs*

$$\frac{2}{3}n^4+O(n^3)$$

multiplications in $\mathbb{K}$*. The same is true for the number of additions in* $\mathbb{K}$*.*

Proof: The worst case obviously arises if we have to increase the companion block with lemma (3.3) and lemma (3.5) in each pass of the outer loop, without ever splitting off a direct factor. The number of multiplications in this case is

$$\begin{aligned}
\sum_{i=1}^{n-1}\left(\mathrm{L3}_*(n,1,i)+\mathrm{L5}_*(n,1,i)\right) &= \sum_{i=1}^{n-1}\left(4in^2-\frac{11}{2}i^2n+2i^3\right)+O(n^3) \\
&= \frac{2}{3}n^4+O(n^3).
\end{aligned}$$

Again, this is true for the number of additions, too, because $\mathrm{L3}_+(n,1,i)$ and $\mathrm{L5}_+(n,1,i)$ have the same leading terms as $\mathrm{L3}_*(n,1,i)$ and $\mathrm{L5}_*(n,1,i)$, respectively. □

In the worst case described in the preceding theorem, all applications of lemma (3.3) except the first one are useless, because the matrix entries that are to be erased are already zero. This could be avoided by testing all those entries and performing the operations from lemma (3.3) only if one of them is nonzero. However, the resulting worst case complexity would still be $O(n^4)$ (we might have to apply lemma (3.3) in every second pass of the outer loop, and the contribution of lemma (3.5) alone accounts for $O(n^4)$ overall time anyways). Furthermore the case where lemma (3.3) and lemma (3.5) have to be applied is rather rare, hence we did not include this refinement in the algorithm. From a practical point of view, theorem (3.8) describes the running time of algorithm (3.7) much better than theorem (3.9).

3.2 Some Remarks on Cyclic Vectors

The use of cyclic vectors is a classical method to uncouple systems of linear ordinary differential equations. We start with the basic definition (for the general pseudo-linear case):

Definition 3.10 *Let V be an n-dimensional vector space and $\theta : V \to V$ be pseudo-linear. A vector $z \in V$ is called* cyclic vector *iff the set*

$$\left\{z, \theta z, \ldots, \theta^{n-1} z\right\}$$

is a basis of V.

If z is a cyclic vector, $\theta^n z$ can be written as

$$\theta^n z = c_0 x + c_1 \theta z + \ldots + c_{n-1} \theta^{n-1} z$$

for some $c_0, \ldots, c_{n-1} \in \mathbb{K}$. The matrix of θ w.r.t. to the basis generated by x is then easily seen to be

$$\begin{pmatrix} 0 & 0 & \ldots & 0 & c_0 \\ 1 & 0 & & \vdots & c_1 \\ 0 & 1 & \ddots & \vdots & \vdots \\ \vdots & & \ddots & 0 & c_{n-2} \\ 0 & \ldots & & 1 & c_{n-1} \end{pmatrix},$$

a transposed companion matrix. Suppose we wish to uncouple the pseudo-linear system $\theta y = r$. We have already seen how to derive an uncoupled higher order equation from a companion matrix, hence it would be better if the above matrix were transposed. Therefore we consider the matrix of the adjoint map θ^*.

Theorem 3.11 *Let T be the matrix of θ w.r.t. a basis $\{b_1, \ldots, b_n\}$ of V. Then the matrix T^* of θ^* w.r.t. the dual basis is given by $T^* = \sigma^{-1}(T^t)$.*

Theorem 3.12 *Upon identifying V and V^{**}, we have $\theta^{**} = \theta$.*

These two theorems, the proofs of which are straightforward verifications (see [25]), imply

Corollary 3.13 *Let $\zeta \in V^*$ be a cyclic vector of θ^*. Then the matrix of θ w.r.t. to the basis of V that is dual to $\{\zeta, \theta^*\zeta, \ldots, \theta^{*n-1}\zeta\}$ is a companion matrix.*

Thus we can uncouple system (3.1) if we can compute a cyclic vector for θ^*.

The drawback of this approach is that it does not always work in the general Ore setting: The adjoint θ^* need not admit a cyclic vector, the simplest counter example being the linear map $\theta = id_V$ (for $n > 1$).

However, cyclic vectors are known to exist in the (ordinary) differential case $\mathbb{K} = \mathbb{R}((x))$, $V = \mathbb{R}((x))^n$, $\delta = D$, $\sigma = id_{\mathbb{K}}$. It is even possible to find a cyclic vector whose components are polynomials of degree less than n, and the probability that a vector chosen at random is cyclic is 1; see [7] and the references given there. A straightforward way to compute a cyclic vector is to test for random candidates whether the set of their pseudo-derivatives is linearly independent.

Even in cases where a cyclic vector can be found, it turns out that the coefficients of the uncoupled equations obtained in this way are very complicated in comparison to other uncoupling methods. Therefore, and because this thesis is concerned with algorithms for the general Ore setting, we deal with the cyclic vector method no further.

We conclude these remarks by citing the following interesting interpretation of Zürcher's algorithm, which is given in more detail and proved in [25]: Zürcher's Algorithm can be used to compute a direct sum decomposition

$$V^* = U_1 \oplus \ldots \oplus U_m,$$

where U_i is a θ^*-invariant subspace of V^* generated by a cyclic vector ζ_i. m is the number of companion blocks in the block-diagonal normal form of the matrix of θ. Hence the uncoupling algorithm by Bruno Zürcher can be viewed as a refined cyclic vector method that works for pseudo-linear systems in general.

Chapter 4

Gaussian Elimination

At first glance it may seem surprising that Gaussian elimination can be applied to systems of linear operator equations. The key point is to consider matrices of Ore operators instead of the coefficient matrix of the system. The resulting algorithm resembles fraction free Gaussian elimination over $\mathbb{Z}$.

Let $\mathcal{O} = \mathbb{K}[\vartheta; \sigma, \delta]$ be an Ore algebra that operates on the left module W, $A = (a_{ij})_{1 \leq i,j \leq n} \in \mathrm{Mat}(n, \mathbb{K})$ and $r = (r_1, \ldots, r_n)^t \in W^n$. The system of equations

$$\begin{aligned} \vartheta y_1 &= a_{11} y_1 + \ldots + a_{1n} y_n + r_1 \\ &\vdots \\ \vartheta y_n &= a_{n1} y_1 + \ldots + a_{nn} y_n + r_n. \end{aligned} \tag{4.1}$$

can be written as

$$My = r, \tag{4.2}$$

where

$$M = \begin{pmatrix} \vartheta - a_{11} & -a_{12} & \ldots & -a_{1n} \\ -a_{21} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & -a_{n-1,n} \\ -a_{n1} & \ldots & -a_{n,n-1} & \vartheta - a_{1n} \end{pmatrix} \in \mathrm{Mat}(n, \mathcal{O}). \tag{4.3}$$

The product My is of course defined by using the outer multiplication of the left $\mathcal{O}$-module W. With this encoding of system (4.1) we can perform Gaussian elimination by using the least common left multiple introduced in section (2.4). Suppose $a_{21} \neq 0$ (otherwise we proceed with a_{31}) and let $a, b \in \mathcal{O}$ be s.t.

$$\mathrm{lclm}(\vartheta - a_{11}, -a_{12}) = a\,(\vartheta - a_{11}) = -b a_{12}.$$

a and b can be computed with algorithm (2.7) (cf. theorem (2.9(ii)). If we multiply the first equation by a on the left, the second one by b and subtract the first equation from the second, we have generated a zero entry at position $(2,1)$. Analogously, we can erase the entries at positions $(i,1)$, $3 \leq i \leq n$ by applying the extended Euclidean Algorithm to each of the pairs $(\vartheta - a_{11}, -a_{i1})$, where $a_{i1} \neq 0$. We then proceed as in the ordinary Gaussian elimination algorithm to triangularize system (4.2). In the general step, the matrix is of the form

$$\begin{pmatrix} m_{11} & \dots & & \dots & m_{1n} \\ 0 & \ddots & & & \vdots \\ & & m_{kk} & \dots & m_{kn} \\ & & m_{k+1,k} & \dots & m_{k+1,n} \\ \vdots & & \vdots & & \vdots \\ 0 & \dots & m_{nk} & \dots & m_{nn} \end{pmatrix}$$

for some $1 \leq k < n$. If now $m_{kk} = 0$, we look for a nonzero entry among m_{ij}, $k \leq i \leq n$, $k \leq j \leq n$. If no such m_{ij} exists, we are done. If, on the other hand, there is such $m_{ij} \neq 0$, we swap lines i and k and columns j and k. Hence we can assume $m_{kk} \neq 0$. We eliminate the nonzero entries among m_{ik}, $k+1 \leq i \leq n$ as described above. Finally the system will be of the form

$$\begin{pmatrix} m_{11} & \dots & & \dots & m_{1n} \\ 0 & \ddots & & & \vdots \\ & & m_{ll} & \dots & m_{ln} \\ & & 0 & \dots & 0 \\ \vdots & & \vdots & & \vdots \\ 0 & \dots & 0 & \dots & 0 \end{pmatrix} \begin{pmatrix} y_1 \\ \vdots \\ \\ \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} s_1 \\ \vdots \\ \\ \\ \vdots \\ s_n \end{pmatrix}$$

for some $1 \leq l \leq n$, $s = (s_1, \dots, s_n)^t \in W^n$. If not all s_i, $l < i \leq n$ are zero, the system has no solution. If they are zero (in particular, if $l = n$) (4.1) is equivalent to the uncoupled system

$$\begin{aligned} m_{11}y_1 + \dots & \dots + m_{1n}y_n = s_1 \\ \ddots & \vdots \\ m_{ll}y_l + \dots + & m_{ln}y_n = s_l. \end{aligned} \tag{4.4}$$

Note that unlike the other uncoupling algorithms presented in this thesis, which return scalar equations whose order sum to the dimension of the initial system, Gaussian elimination in general returns scalar equations of higher order, depending on the degrees of the Ore polynomials in (4.4).
Of course, this algorithm works for any $M \in \mathrm{Mat}(n, \mathcal{O})$, not just those of the special form (4.3) that arises from system (4.1). In the following program listing, the variable `perm` keeps track of the column changes in the system matrix, which induce changes in the order of the unknowns.

Algorithm 4.1

```
perm=(1,...,n) (* identity permutation *)
for k = 2,...,n
    if there is m_pq ≠0, k≤p≤n, k≤q≤n then
        swap rows k and p, columns k and q, r_k and r_p
        perm[p] := q; perm[q] := p
    else stop
    for i = k,...,n
        determine a,b s.t. a m_kk = b m_ik
        for j = k,...,n
            m_ij := a * m_kj - b * m_ij
        r_i := a * r_k - b * r_i
    end
end
```

Apparently, analyzing the complexity of this algorithm would be a rather difficult task. One would have to deal with the complexity of the Euclidean Algorithm in Ore polynomial rings and to keep track of the degrees of the matrix entries during the execution of the elimination algorithm. We confine ourselves with mentioning that empirical evidence shows Gaussian elimination to be less efficient than the other three algorithms in this thesis. It usually takes more time and returns equations of high order and with large coefficients.

Chapter 5

Block Triangular Decomposition

5.1 The Uncoupling Algorithm by Abramov and Zima

5.1.1 The Problem

Let $\mathbb{K}[\vartheta;\sigma,\delta]$ be an Ore algebra that operates on the left module W. Given a linear system of equations

$$\vartheta y = Ay + r \tag{5.1}$$

where $A = (a_{ij})_{1\le i,j\le n} \in \mathrm{Mat}(n,\mathbb{K})$ and $r = (r_1,\ldots,r_n)^t \in W^n$, we want to reduce the problem of finding the solutions to that of solving higher order scalar equations. This can be achieved by an algorithm due to Abramov and Zima [6], which is a generalization of an uncoupling algorithm for differential systems by Murray and Miller [18]. Its goal is to find an equivalent system

$$(y-\text{equations}, z-\text{equations}, T-\text{equations})$$

in $y = (y_1,\ldots,y_n)^t \in W^n$ consisting of the following components:

(i) *y-equations:*

$$\begin{aligned}
\sum_{j=0}^{l_1} \alpha_{1j}\vartheta^j y_{i_1} &= \rho_1 \\
\sum_{j=0}^{l_2} \alpha_{2j}\vartheta^j y_{i_2} &= \sum_{k=1}^{i_2-1}\sum_{j=0}^{l_2-1} \alpha^*_{2kj}\vartheta^j z_k + \rho_2 \\
&\vdots \\
\sum_{j=0}^{l_s} \alpha_{sj}\vartheta^j y_{i_s} &= \sum_{k=1}^{i_s-1}\sum_{j=0}^{l_s-1} \alpha^*_{skj}\vartheta^j z_k + \rho_s,
\end{aligned} \tag{5.2}$$

where $\alpha_{ij}, \alpha^*_{ikj} \in \mathbb{K}$, $\alpha_{kl_k} = 1$ and $\rho_i \in W$. The index set $I = \{i_1, \ldots, i_s\}$ with $1 = i_1 < \ldots < i_s \leq n$ is given by the algorithm. The l_k are defined by $l_k = i_{k+1} - i_k$ for $1 \leq k < s$ and $l_s = n - i_s + 1$, which implies $l_1 + \ldots + l_s = n$.

(ii) *z-equations:* The new variables z_i from (i) satisfy:

$$z_i = y_i \quad \text{for} \quad i \in I \tag{5.3}$$

and

$$z_i = \vartheta z_{i-1} + \sum_{k=1}^{i-1} \beta_{ik} z_k + \tau_i \quad \text{for} \quad 1 < i \leq n,\ i \notin I \tag{5.4}$$

where $\beta_{ik} \in \mathbb{K}$ and $\tau_i \in W$.

(iii) *T-equations:* Finally we get a linear system of equations that do not contain applications of the ϑ-operator:

$$Ty = z, \tag{5.5}$$

where $T \in \mathrm{Mat}(n, \mathbb{K})$ is a non-singular upper triangular matrix. In accordance with (5.3), for $i \in I$ the i-th line of T is e_i, the vector with 1 at position i and zero elsewhere.

It has to be noted that the algorithm might change the order of $y_2, \ldots, y_n$. We did not include this possible renumbering in the above specification in order not to blow up notation.

5.1.2 Solution of the Initial System from the Uncoupled System

Before describing how we can transform (5.1) into the uncoupled system AZ, we will outline how system (5.1) can be solved using AZ, if we assume that we have an algorithm for finding solutions of scalar equations.

First we solve equation 1 of (5.2) for y_1. We use one of these solutions to compute z_j, $2 \leq j < i_2$ by (5.4). Of course, these z_j are uniquely determined by y_1. Then we plug $y_1 (= z_1)$ and z_j, $2 \leq j < i_2$ into the second equation of (5.2), yielding a scalar equation for y_{i_2}, and so on. Afterwards we use (5.5) to compute the remaining y_i from $y_{i_1}, \ldots, y_{i_s}$ and the z_j, $j \in \{1, \ldots, n\} \setminus I$. As already mentioned, we finally might have to restore the order of $y_2, \ldots, y_n$, which might have been permuted by the uncoupling algorithm.

5.1.3 The Algorithm

We begin the presentation of the algorithm by following [6]. There, only the beginning of the computation is described in detail, and it is briefly outlined

how to proceed. We add a detailed description of the general step, both of the first stage of the algorithm (computation of an equivalent system with block triangular matrix) and the second stage (deduction of scalar equations).

Written out in full, system (5.1) reads:

$$\begin{aligned} \vartheta y_1 &= a_{11}y_1 + \ldots + a_{1n}y_n + r_1 \\ &\vdots \\ \vartheta y_n &= a_{n1}y_1 + \ldots + a_{nn}y_n + r_n. \end{aligned} \tag{5.6}$$

Eliminate y_2 from the right hand sides:

If $a_{12} = \ldots = a_{1n} = 0$, we can take the first equation of (5.6) as the first equation of (5.2). Continue reading at 'repeat the whole process' (with $l = 1$).

If, on the other hand, $a_{1j} \neq 0$ for some $j \geq 2$ (w.l.o.g. $j = 2$, since we can reenumerate unknowns), we introduce a new variable

$$z_2 = a_{12}y_2 + \ldots + a_{1n}y_n. \tag{5.7}$$

If we use this relation to eliminate y_2 from the right hand sides of equations 2 to n, we get a system of the form

$$\begin{aligned} \vartheta y_1 &= a_{11}y_1 + z_2 + r_1 \\ \vartheta y_2 &= b_{21}y_1 + b_{22}z_2 + b_{23}y_3 + \ldots + b_{2n}y_n + r_2 \\ &\vdots \\ \vartheta y_n &= b_{n1}y_1 + b_{n2}z_2 + b_{n3}y_3 + \ldots + b_{nn}y_n + r_n. \end{aligned} \tag{5.8}$$

Eliminate y_2 from the left hand side:

In order to get rid of ϑy_2 in equation 2 above, we proceed as follows: Application of ϑ to (5.7) yields

$$\vartheta z_2 = \sigma(a_{12})\vartheta y_2 + \delta(a_{12})y_2 + \ldots + \sigma(a_{1n})\vartheta y_n + \delta(a_{1n})y_n. \tag{5.9}$$

We use (5.7), (5.8) to eliminate $y_2, \vartheta y_2, \ldots, \vartheta y_n$ from (5.9) and take the result as our new second equation. Thus we have arrived at a system of the form

$$\begin{aligned} \vartheta y_1 &= a_{11}y_1 + z_2 + r_1 \\ \vartheta z_2 &= c_{21}y_1 + c_{22}z_2 + c_{23}y_3 + \ldots + c_{2n}y_n + s_2 \\ \vartheta y_3 &= b_{31}y_1 + b_{32}z_2 + b_{33}y_3 + \ldots + b_{3n}y_n + r_3 \\ &\vdots \\ \vartheta y_n &= b_{n1}y_1 + b_{n2}z_2 + b_{n3}y_3 + \ldots + b_{nn}y_n + r_n. \end{aligned} \tag{5.10}$$

By looking at the first equation we see that we have made a first step towards triangularization of system (5.6).

Iteration:

If at least one of $c_{23}, \ldots, c_{2n}$ (w.l.o.g. c_{23}) is nonzero, we introduce the new unknown z_3:

$$z_3 = c_{23}y_3 + \ldots + c_{2n}y_n,$$

and proceed analogously to steps 1 and 2 to eliminate y_3, and so on. If we continue in this fashion, at some point we will have to stop elimination, either because we have eliminated $y_2, \ldots, y_n$ or because the next equation to be considered does not contain any of the remaining y_i. Suppose the latter happens after eliminating $l-1$ variables, then our system will have the form

$$\begin{aligned}
\vartheta y_1 &= d_{11}y_1 + z_2 + r_1 \\
\vartheta z_2 &= d_{21}y_1 + d_{22}z_2 + z_3 + s_2 \\
&\vdots \\
\vartheta z_{l-1} &= d_{l-1,1}y_1 + d_{l-1,2}z_2 + \ldots + d_{l-1,l-1}z_{l-1} + z_l + u_{l-1} \\
\vartheta z_l &= d_{l,1}y_1 + d_{l,2}z_2 + \ldots + d_{l,l-1}z_{l-1} + d_{l,l}z_l + u_l \\
\vartheta y_{l+1} &= d_{l+1,1}y_1 + d_{l+1,2}z_2 + \ldots + d_{l+1,l}z_l + d_{l+1,l+1}y_{l+1} + \ldots + d_{l+1,n}y_n + u_{l+1} \\
&\vdots \\
\vartheta y_n &= d_{n,1}y_1 + d_{n,2}z_2 + \ldots + d_{n,l}z_l + d_{n,l+1}y_{l+1} + \ldots + d_{n,n}y_n + u_n.
\end{aligned} \tag{5.11}$$

We set $l_1 = l$ and $i_2 = l+1$ (Recall $i_1 = 1$). Equations 1 to $l-1$ of (5.11) yield the first $l-1$ equations of (5.4).

Deduce a scalar equation:

We can get an equation of order l in y_1 from the first l equations of (5.11) as follows: The first equation allows to express z_2 via y_1 and ϑy_1. If we apply ϑ to the first equation, we can write ϑz_2 in terms of y_1, ϑy_1 and $\vartheta^2 y_1$. By the second equation, we can now express z_3 and ϑz_3 via y_1, ϑy_1, $\vartheta^2 y_1$ and $\vartheta^3 y_1$, and so on.

Finally, we plug the expressions for z_l and ϑz_l obtained from equation $l-1$ into equation l, which gives the first equation of (5.2).

If we collect the equations by which the variables $z_2, \ldots, z_l$ were introduced, we get a triangular algebraic linear system

$$\begin{aligned}
z_2 &= a_{12}y_2 + a_{13}y_3 + \ldots \quad \ldots + a_{1n}y_n \\
z_3 &= \quad c_{23}y_3 + \ldots \quad \ldots + c_{2n}y_n \\
&\vdots \qquad \ddots \qquad\qquad\quad \vdots \\
z_l &= \qquad\quad e_{l-1,l}y_l + \ldots + e_{l-1,n}y_n.
\end{aligned} \tag{5.12}$$

Repeat the whole process: *(only if $l < n$, which we will call the degenerate case)*

In equations $l+1, \ldots, n$ of (5.11) we consider $y_1, z_2, \ldots, z_l$ as known and perform the same transformations. This yields the second equation of (5.2), the equations for $i_2 < i < i_3$ of (5.4) and so on. Let s be the number of times we repeat the process described so far. Finally, the ϑ-free system that consists of (5.12), its succeding counterparts and the equations $z_j = y_j,\ j \in I$ yield (5.5). (5.3) is just a definition which simplifies notation several times.

Before giving the pseudocode of the algorithm, we will describe the general step of the computation in detail.

By taking $i = \nu = 1$ if necessary, we can assume that our system has the form

$$\begin{pmatrix} \vartheta z_1 \\ \vdots \\ \\ \\ \\ \vdots \\ \vartheta z_{i-1} \\ \vartheta z_i \\ \vdots \\ \vartheta y_n \end{pmatrix} = \begin{pmatrix} A_1 & & & & & & & & & 0 \\ & \ddots & & & & & & & & \\ & & A_{\nu-1} & & & & & & & \\ & & & a_{i_\nu i_\nu} & 1 & 0 & & & \ldots & 0 \\ & & & \vdots & & \ddots & \ddots & & & \vdots \\ & & & a_{i-1,i_\nu} & \ldots & a_{i-1,i-1} & 1 & 0 & \ldots & 0 \\ & & & a_{i,i_\nu} & \ldots & a_{i,i-1} & a_{i,i} & a_{i,i+1} & \ldots & a_{in} \\ & & & \vdots & & & & & & \vdots \\ * & & & a_{n,i_\nu} & \ldots & & & & \ldots & a_{nn} \end{pmatrix} \times$$

$$\times \begin{pmatrix} z_1 \\ \vdots \\ \\ \\ \\ \vdots \\ z_{i-1} \\ z_i \\ \vdots \\ y_n \end{pmatrix} + r, \tag{5.13}$$

where $A_1, \ldots, A_{\nu-1}$ are of the form

$$\begin{pmatrix} * & 1 & 0 & \ldots & 0 \\ \vdots & & \ddots & \ddots & \vdots \\ * & \ldots & * & 1 & 0 \\ * & \ldots & & * & 1 \\ * & \ldots & & & * \end{pmatrix}. \tag{5.14}$$

If now $a_{i,i+1} = \ldots = a_{in} = 0$, we add i_ν to I and by writing

$$A_\nu = \begin{pmatrix} a_{i_\nu i_\nu} & 1 & 0 & \ldots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & 1 & 0 \\ a_{i-1,i_\nu} & \ldots & & a_{i-1,i-1} & 1 \\ a_{i,i_\nu} & \ldots & & a_{i,i-1} & a_{i,i} \end{pmatrix},$$

we are once again in situation (5.13), with ν increased by 1. Now suppose $a_{i,i+1} \neq 0$. (If $a_{i,i+1} = 0$ but $a_{i,j} \neq 0$ for some $i+1 < j \leq n$, we swap lines $i+1$ and j, columns $i+1$ and j of a and the components $i+1$ and j of r. Of course, we have to keep track of these permutations to restore them eventually.)

As described above, we introduce the new variable

$$z_{i+1} = a_{i,i+1} y_{i+1} + \ldots + a_{in} y_n. \tag{5.15}$$

If we express y_{i+1} by this equation, we get

$$y_{i+1} = a_{i,i+1}^{-1} z_{i+1} - \sum_{j=i+2}^{n} b_j y_j \tag{5.16}$$

with $b_j := a_{ij}/a_{i,i+1}$. The b_j are introduced to decrease the amount of computation; we will need them two times in what follows. We use this expression for y_{i+1} to eliminate y_{i+1} from equations $i+1, \ldots, n$. For $i+1 \leq k \leq n$, plugging in (5.16) for y_{i+1} (but not for ϑy_{i+1}) gives the equations

$$\begin{aligned} \vartheta y_k \;=\; & a_{k1} z_1 + \ldots + a_{ki} z_i + \frac{a_{k,i+1}}{a_{i,i+1}} z_{i+1} + (a_{k,i+2} - a_{k,i+1} b_{i+2})\, y_{i+2} + \\ & + \ldots + (a_{kn} - a_{k,i+1} b_n)\, y_n + r_k. \end{aligned} \tag{5.17}$$

We denote by a^*_{kj} for $i+1 \leq k \leq n$, $1 \leq j \leq n$ the matrix entries updated according to (5.17).
Applying ϑ to (5.15) gives

$$\vartheta z_{i+1} = \sigma(a_{i,i+1}) \vartheta y_{i+1} + \delta(a_{i,i+1}) y_{i+1} + \ldots + \sigma(a_{in}) \vartheta y_n + \delta(a_{in}) y_n. \tag{5.18}$$

This will become our new equation $i+1$, once we have eliminated y_{i+1}, $\vartheta y_{i+1}, \ldots, \vartheta y_n$, using (5.16) and (5.17). To do so, we multiply (5.16)by $\delta(a_{i,i+1})$, which yields

$$\delta(a_{i,i+1}) y_{i+1} = \frac{\delta a_{i,i+1}}{a_{i,i+1}} z_{i+1} - \delta(a_{i,i+1}) b_{i+2} y_{i+2} - \ldots - \delta(a_{i,i+1}) b_n y_n, \tag{5.19}$$

and (5.17) by $\sigma(a_{ik})$, giving

$$\begin{aligned} \sigma(a_{ik}) \vartheta y_k = {} & \sigma(a_{ik}) a^*_{k1} z_1 + \ldots + \sigma(a_{ik}) a^*_{ki} z_i + \sigma(a_{ik}) a^*_{k,i+1} z_{i+1} + \\ & + \sigma(a_{ik}) a^*_{k,i+2} y_{i+2} + \ldots + \sigma(a_{ik}) a^*_{kn} y_n + \sigma(a_{ik}) r_k \end{aligned} \tag{5.20}$$

for $i+1 \leq k \leq n$. Using this in (5.18), we get an equation of the form

$$\vartheta z_{i+1} = a^{**}_{i+1,1} z_1 + \ldots a^{**}_{i+1,i+1} z_{i+1} + a^{**}_{i+1,i+2} y_{i+2} + \ldots + a^{**}_{i+1,n} y_n + r^*_{i+1}, \quad (5.21)$$

by which we replace equation $i+1$. r_{i+1} is the only component of r that has to be updated, hence we set $r^*_j = r_j$ for $1 \leq j \leq n$, $j \neq i+1$. We have arrived at the system

$$\begin{pmatrix} \vartheta z_1 \\ \vdots \\ \\ \\ \\ \vdots \\ \vartheta z_{i-1} \\ \vartheta z_i \\ \vartheta z_{i+1} \\ \vdots \\ \vartheta y_n \end{pmatrix} = \begin{pmatrix} A_1 & & & & & & & & & & 0 \\ & \ddots & & & & & & & & & \\ & & A_{\nu-1} & & & & & & & & \\ & & & a_{i_\nu i_\nu} & 1 & 0 & & & & \ldots & 0 \\ & & & \vdots & & \ddots & \ddots & & & & \vdots \\ & & & a_{i-1,i_\nu} & \ldots & a_{i-1,i-1} & 1 & 0 & & \ldots & 0 \\ & & & a_{i,i_\nu} & \ldots & a_{i,i-1} & a_{i,i} & 1 & 0 & \ldots & 0 \\ & & & a^{**}_{i+1,i_\nu} & \ldots & a^{**}_{i+1,i-1} & a^{**}_{i+1,i_\nu} & a^{**}_{i+1,i+1} & & \ldots & a^{**}_{i+1,n} \\ & & & \vdots & & & & & & & \vdots \\ * & & & a^*_{n,i_\nu} & \ldots & & & & & \ldots & a^*_{nn} \end{pmatrix} \times$$

$$\times \begin{pmatrix} z_1 \\ \vdots \\ \\ \\ \\ \vdots \\ z_{i-1} \\ z_i \\ z_{i+1} \\ \vdots \\ y_n \end{pmatrix} + r^*, \quad (5.22)$$

which is of the form (5.13) with i increased by one.
Finally, we discuss the deduction of the y-equations. When we are done with the triangularization, we have transformed the system into the form

$$\vartheta z = \begin{pmatrix} A_1 & & 0 \\ & \ddots & \\ * & & A_s \end{pmatrix} z + r$$

where A_k is an l_k by l_k block of the form (5.14). We have already described how the first equation of (5.2) is obtained. The k-th of these equations is to

be deduced from the l_k equations

$$\begin{aligned}
\vartheta z_{i_k} &= a_{i_k 1} z_1 + \ldots + a_{i_k i_k} z_{i_k} + z_{i_k+1} + r_{i_k} \\
&\vdots \\
\vartheta z_{i_{k+1}-2} &= a_{i_{k+1}-2,1} z_1 + \ldots + a_{i_{k+1}-2,i_{k+1}-2} z_{i_{k+1}-2} + z_{i_{k+1}-1} + r_{i_{k+1}-2} \\
\vartheta z_{i_{k+1}-1} &= a_{i_{k+1}-1,1} z_1 + \ldots + a_{i_{k+1}-1,i_{k+1}-1} z_{i_{k+1}-1} + r_{i_{k+1}-1}
\end{aligned}$$

in the same way, by expressing z_{i_k+1} via the first equation and ϑz_{i_k+1} by applying ϑ to the first equation, plugging this into the remaining equations, and so on. In this process, we apply ϑ $\quad l_k - 1$ times to z_j, $1 \le j < i_k$, and l_k times to z_{i_k}, which explains the order of the equations (5.2).

Let us view this process in more detail. We will encounter equations with higher order pseudo-derivatives of several of the z_i, so we introduce new coefficients $d_{i,t,j}$, where the third index marks the order of the application of ϑ. They are initialized by $d_{i,j,0} = a_{i,j}$, $1 \le i, j \le n$. To keep notation simple, we do not change the names of these coefficients and the r_i, even if they are updated by the steps described below. Now the k-th block of the triangularized system ($1 \le k \le s$) takes the form

$$\begin{aligned}
\vartheta z_{i_k} &= \sum_{t=1}^{i_k} d_{i_k,t,0} z_t + z_{i_k+1} + r_{i_k} \\
&\vdots \\
\vartheta z_{i_{k+1}-1} &= \sum_{t=1}^{i_k} d_{i_{k+1}-1,t,0} z_t + \sum_{t=i_k+1}^{i_{k+1}-1} d_{i_{k+1}-1,t,0} z_t + r_{i_{k+1}-1}.
\end{aligned} \tag{5.23}$$

(We set $i_{s+1} = n + 1$ for convenience of notation.) Let $1 \le m \le l_k - 1$ and suppose we have eliminated $z_{i_k+1}, \ldots, z_{i_k+m-1}$ from (5.23). Then we continue by expressing z_{i_k+m} and ϑz_{i_k+m} by the m-th equation, which will be of the form (with new coefficients $d_{i,t,j}$!)

$$\vartheta^m z_{i_k} = \sum_{t=1}^{i_k} \sum_{j=0}^{m-1} d_{i_k+m-1,t,j} \vartheta^j z_t + z_{i_k+m} + r_{i_k+m-1}, \tag{5.24}$$

(in the first step $m = 1$, this is the first equation of (5.23)) and substituting the result for z_{i_k+m} and ϑz_{i_k+m} in the $m+1$-st equation

$$\begin{aligned}
\vartheta z_{i_k+m} &= \sum_{t=1}^{i_k} \sum_{j=0}^{m-1} d_{i_k+m,t,j} \vartheta^j z_t + d_{i_k+m,i_k+m,0}\ z_{i_k+m} \\
&\quad + [m < l_k - 1]\, z_{i_k+m+1} + r_{i_k+m}.
\end{aligned} \tag{5.25}$$

(For $m = 1$, this is the second equation of (5.23)). Here we made use of the notation [false] $= 0$ and [true] $= 1$. Then we substitute for z_{i_k+m} in the

remaining equations

$$\vartheta z_{i_k+i} = \sum_{t=1}^{i_k}\sum_{j=0}^{m-1} d_{i_k+i,t,j}\vartheta^j z_t + \sum_{t=i_k+m}^{i_k+i} d_{i_k+i,t,0}\ z_t + [i < l_k - 1]\, z_{i_k+i+1} + r_{i_k+i}, \quad m < i < l_k - 1. \tag{5.26}$$

Expressing z_{i_k+m} via (5.24) yields

$$z_{i_k+m} = -\sum_{t=1}^{i_k}\sum_{j=0}^{m-1} d_{i_k+m-1,t,j}\vartheta^j z_t + \vartheta^m z_{i_k} - r_{i_k+m-1} \tag{5.27}$$

and, consequently,

$$\begin{aligned}
\vartheta z_{i_k+m} &= -\sum_{t=1}^{i_k}\sum_{j=0}^{m-1}\big(\sigma(d_{i_k+m-1,t,j})\vartheta^{j+1} z_t + \delta(d_{i_k+m-1,t,j})\vartheta^j z_t\big) \\
&\quad +\vartheta^{m+1} z_{i_k} - \vartheta r_{i_k+m-1} \\
&= -\sum_{t=1}^{i_k}\Bigg(\sum_{j=1}^{m-1}\sigma(d_{i_k+m-1,t,j-1})\vartheta^j z_t + \sigma(d_{i_k+m-1,t,m-1})\vartheta^m z_t \\
&\quad +\delta(d_{i_k+m-1,t,0})z_t) + \sum_{j=1}^{m-1}\delta(d_{i_k+m-1,t,j})\vartheta^j z_t\Bigg) + \vartheta^{m+1} z_{i_k} - \vartheta r_{i_k+m-1} \\
&= -\sum_{t=1}^{i_k}\Bigg(\delta(d_{i_k+m-1,t,0})z_t + \sum_{j=1}^{m-1}\big(\sigma(d_{i_k+m-1,t,j-1}) + \delta d_{i_k+m-1,t,j}\big)\vartheta^j z_t \\
&\quad +\sigma(d_{i_k+m-1,t,m-1})\vartheta^m z_t\Bigg) + \vartheta^{m+1} z_{i_k} - \vartheta r_{i_k+m-1}.
\end{aligned}$$

Plugging this into (5.25), we get

$$\begin{aligned}
&-\sum_{t=1}^{i_k}\Bigg(\delta(d_{i_k+m-1,t,0})z_t + \sum_{j=1}^{m-1}\big(\sigma(d_{i_k+m-1,t,j-1}) + \delta d_{i_k+m-1,t,j}\big)\vartheta^j z_t \\
&+\sigma(d_{i_k+m-1,t,m-1})\vartheta^m z_t\Bigg) + \vartheta^{m+1} z_{i_k} - \vartheta r_{i_k+m-1} \\
= &\sum_{t=1}^{i_k}\sum_{j=0}^{m-1} d_{i_k+m,t,j}\vartheta^j z_t + d_{i_k+m,i_k+m,0}\Bigg(-\sum_{t=1}^{i_k}\sum_{j=0}^{m-1} d_{i_k+m-1,t,j}\vartheta^j z_t \\
&+\vartheta^m z_{i_k} - r_{i_k+m-1}\Bigg) + [m < l_k - 1]\, z_{i_k+m+1} + r_{i_k+m},
\end{aligned}$$

that is,

$$\begin{aligned}\vartheta^{m+1}z_{i_k} &= \sum_{t=1}^{i_k}\Bigg((\delta d_{i_k+m-1,t,0}+d_{i_k+m,t,0}-d_{i_k+m,i_k+m,0}\ d_{i_k+m-1,t,0})\,z_t\\ &\quad+\sum_{j=1}^{m-1}\Bigg(\sigma(d_{i_k+m-1,t,j-1})+\delta d_{i_k+m-1,t,j}+d_{i_k+m,t,j}\\ &\quad-d_{i_k+m,i_k+m,0}\ d_{i_k+m-1,t,j}\Bigg)\vartheta^j z_t+\sigma(d_{i_k+m-1,t,m-1})\vartheta^m z_t\Bigg)\\ &\quad+d_{i_k+m,i_k+m,0}\vartheta^m z_{i_k}+[m<l_k-1]\,z_{i_k+m+1}+\vartheta r_{i_k+m-1}\\ &\quad-d_{i_k+m,i_k+m,0}\ r_{i_k+m-1}+r_{i_k+m}\end{aligned}\tag{5.28}$$

as our new $m+1$-st equation. (In the last step $m=l_k-1$, this is an uncoupled equation for z_{i_k}; note that $z_1,\dots,z_{i_k-1}$ are assumed as known when we are dealing with the k-th block.) What remains to do is to insert (5.27) into (5.26), yielding the equations

$$\begin{aligned}\vartheta z_{i_k+i} &= \sum_{t=1}^{i_k}\sum_{j=0}^{m-1}(d_{i_k+i,t,j}-d_{i_k+i,i_k+m,0}\ d_{i_k+m-1,t,j})\,\vartheta^j z_t+d_{i_k+i,i_k+m,0}\vartheta^m z_{i_k}\\ &\quad+\sum_{t=i_k+m+1}^{i_k+i}d_{i_k+i,t,0}z_t+[i<l_k-1]\,z_{i_k+i+1}+r_{i_k+i}\\ &\quad-d_{i_k+i,i_k+m,0}\ r_{i_k+m-1},\qquad m<i\le l_k-1.\end{aligned}$$

This completes the general step of the deduction of the y-equation for the k-th block. When we have done this for all $1\le k\le s$, $1\le m\le l_k-1$, the last equation (namely, (5.28) for $m=l_k-1$) of each block is an uncoupled higher order equation for z_{i_k}:

$$\vartheta^{l_k}z_{i_k}=\sum_{t=1}^{i_k}\sum_{j=0}^{l_k-1}d_{i_{k+1}-1,t,j}\vartheta^j z_t+r_{i_{k+1}-1},\qquad 1\le k\le s.$$

(Note $i_{s+1}=n+1$) These equations form (5.2).

Algorithm 5.1 *by Abramov and Zima*

```
I := {1}; perm := (1,...,n) (* identity permutation *)
T=0 (* zero matrix; T=t_ij *)
(* first stage: transformation into block triangular shape *)
for i = 1,...,n-1
    (* look for variable with nonzero coefficient *)
    j0 := i + 1
    while j0 <= n and a_i,j = 0
        j0 := j0 + 1

    if j0 <= n then
        (* swap variables i+1 and j0 *)
        perm[i+1] := j0; perm[j0] := i+1
        (* swap equations i+1 and j0 *)
        for j = 1,...,n
            s := a_i+1,j; a_i+1,j := a_j0,j; a_j0,j := s
        (* swap entries i+1 and j0 of the right hand side *)
            s := r_i+1; r_i+1 := r_j0; r_i+1 := s
        (* swap variables i+1 and j0 in equations i,...,n *)
        (* and swap columns i+1 and j0 of T *)
        for j = i,...,n
            s := a_j,i+1; a_j,i+1 := a_j,j0; a_j,j0 := s
            s := t_j,i+1; t_j,i+1 := t_j,j0; t_j,j0 := s

        (* update T *)
        for j = i+1,...,n
            t_i+1,j := a_i,j

        for j = i+2,...,n
            b_j := a_i,j / a_i,i+1

        (* eliminate variable i+1 from the right hand sides *)
        for k = i+1,...,n
            for j = i+2,...,n
                a_k,j := a_k,j - a_k,i+1 * b_j
            a_k,i+1 := a_k,i+1 / a_i,i+1

        (* initialize the coefficients of the new equation i+1 *)
        for j = 1,...,i
            c_j := 0

        (* eliminate variable i+1 from the new equation i+1 *)
```

```
            c_{i+1} := δ(a_{i,i+1}) / a_{i,i+1}
            for j = i+2,...,n
                c_j := -δ(a_{i,i+1}) * b_j

            (* eliminate ϑ of variables i+1,..,n from new equation i+1 *)
            r_{i+1} := 0
            for k = i+1,...,n
                for j = 1,...,n
                    c_j := c_j + σ(a_{i,k}) * a_{k,j}
                r_{i+1} := r_{i+1} + σ(a_{i,k}) * r_k

            (* update the coefficient matrix with the new equation i+1 *)
            for j = 1,...,n
                a_{i+1,j} := c_j


            (* update equation i *)
            a_{i,i+1} := 1;
            for j = i+2,...,n
                a_{ij} := 0
        end
        else        (* there is no variable suitable for elimination *)
            I := I ∪ i+1
            t_{i+1,i+1} := 1
            for j = i+2,...,n
                t_{i+1,j} := 0
        end
    end
```

```
(* second stage: deduction of the y-equations *)
s := |I|
{i_1,...,i_s} := I; i_{s+1} := n+1
for i = 1,...,n
    for j = 1,...,n
        d_{i,j,0} := a_{i,j}

for k = 1,...,s
    for m = 1,...,i_{k+1}-i_k-1
        for t = 1,...,i_k
            d_{i_k+m,t,0} := δ(d_{i_k+m-1,t,0}) + d_{i_k+m,t,0}
                           - d_{i_k+m,i_k+m,0} * d_{i_k+m-1,t,0}
            for j = 1,...,m-1
                d_{i_k+m,t,j} := σ(d_{i_k+m-1,t,j-1}) + δ(d_{i_k+m-1,t,j})
                               + d_{i_k+m,t,j} - d_{i_k+m,i_k+m,0} * d_{i_k+m-1,t,j}
            d_{i_k+m,t,m} := σ(d_{i_k+m-1,t,m-1})
        end
        d_{i_k+m,i_k,m} := d_{i_k+m,i_k,m} + d_{i_k+m,i_k+m,0}
        r_{i_k+m} := r_{i_k+m} + ϑ(r_{i_k+m-1}) - d_{i_k+m,i_k+m,0} * r_{i_k+m-1}
        d_{i_k+m,i_k+m,0} := 0
        for i = m+1,...,i_{k+1}-i_k-1
            for t = 1,...,i_k
                for j = 0,...,m-1
                    d_{i_k+i,t,j} := d_{i_k+i,t,j} - d_{i_k+i,i_k+m,0} * d_{i_k+m-1,t,j}
            d_{i_k+i,i_k,m} := d_{i_k+i,i_k+m,0}
            r_{i_k+i} := r_{i_k+i} - d_{i_k+i,i_k+m,0} * r_{i_k+m-1}
            d_{i_k+i,i_k+m,0} := 0
        end
    end
end
```

5.1.4 Correctness

Theorem 5.2 *The solutions of the uncoupled system AZ are exactly the solutions of (5.1).*

Proof: From the description of the algorithm is is clear that any solution of (5.1) solves the uncoupled system. As for the converse, we take a closer look at the transition from (5.13) to (5.22). Of course, equations 1 to i of (5.22) (which remain unchanged) together with (5.17) for $i+1 \leq k \leq n$ and (5.15) imply (5.13). But in the next step of the algorithm, equation (5.17) for $k = i+1$ is discarded and replaced by (5.21), which becomes the first equation of (5.22). Hence we have to verify that (5.22),(5.15) imply (5.17) for $k = i+1$.

To see it, suppose $z_1, \ldots, z_{i+1}, y_{i+2}, \ldots, y_n$ satisfy (5.22),(5.15). Taking into account how equation (5.21), i.e., the $i+1$-st equation of (5.22) was obtained, it reads

$$\begin{aligned}\vartheta z_{i+1} &= \sigma(a_{i,i+1})\left(\sum_{j=1}^{i+1} a^*_{i+1,j} z_j + \sum_{j=i+2}^{n} a^*_{i+1,j} y_j + r_{i+1}\right) \\ &\quad +\delta(a_{i,i+1})\left(a^{-1}_{i,i+1} z_{i+1} - \sum_{j=i+2}^{n} b_j y_j\right) \\ &\quad + \sum_{t=i+2}^{n}\left(\sigma(a_{it})\left(\sum_{j=1}^{i+1} a^*_{tj} z_j + \sum_{j=i+2}^{n} a^*_{tj} y_j + r_t\right) + \delta(a_{it}) y_t\right).\end{aligned} \tag{5.29}$$

By (5.15) and equations $i+2$ to n of (5.22) the right hand side further equals

$$\begin{aligned}&\sigma(a_{i,i+1})\left(\sum_{j=1}^{i+1} a^*_{i+1,j} z_j + \sum_{j=i+2}^{n} a^*_{i+1,j} y_j + r_{i+1}\right) + \delta(a_{i,i+1}) y_{i+1} \\ &\qquad + \sum_{t=i+2}^{n}\left(\sigma(a_{it})\vartheta y_t + \delta(a_{it}) y_t\right),\end{aligned}$$

and by comparing this to (5.18) (which follows from (5.15)), we have

$$\sigma(a_{i,i+1})\left(\sum_{j=1}^{i+1} a^*_{i+1,j} z_j + \sum_{j=i+2}^{n} a^*_{i+1,j} y_j + r_{i+1}\right) = \sigma(a_{i,i+1})\vartheta y_{i+1}.$$

Now the validity of (5.17) for $k = i+1$ follows from $a_{i,i+1} \neq 0$ and σ being a monomorphism, hence $\sigma(a_{i,i+1}) \neq 0$. □

5.1.5 The Solution Space

The proof of the following theorem shows how we can find a basis for the solution space of the homogeneous system $\vartheta y = Ay$, provided that we can find bases for the solution spaces of the uncoupled equations. The general solution of the inhomogeneous system $\vartheta y = Ay + r$ can then be expressed by one fixed solution plus the solution space of the homogeneous system.

Theorem 5.3 *Let W be a vector space over $\mathbb{K}$, $\vartheta : W \to W$ be pseudo-linear, $A = (a_{ij})_{1 \leq i,j \leq n} \in \mathrm{Mat}(n, \mathbb{K})$ and $K \subseteq \mathbb{K}$ be a subfield of $\mathbb{K}$ (e.g., $K = \mathrm{Const}_{\sigma,\delta}$). If each of the scalar equations*

$$\sum_{j=0}^{l_k} \alpha_{kj} \vartheta^j y_{i_k} = \mu, \qquad k = 1, \ldots, s, \tag{5.30}$$

where α_{kj} as in (5.2) is obtained by uncoupling the system $\vartheta y = Ay$ and $\mu \in W$ is arbitrary, has a solution in W, and if each of the homogeneous scalar equations

$$\sum_{j=0}^{l_k} \alpha_{kj} \vartheta^j y_{i_k} = 0, \qquad k = 1, \ldots, s \tag{5.31}$$

has l_k K-linearly independent solutions in W, then the homogenous pseudo-linear system

$$\theta y = Ay \tag{5.32}$$

has n K-linearly independent solutions in W^n.

Proof. We begin by noting that if $r = 0$ in (5.1), then $\rho_i = 0$ in (5.2) and $\tau_i = 0$ in (5.4), which is easily seen from the description of the algorithm. Furthermore, we can assume w.l.o.g. that the order of the y_i is not changed by the algorithm. In what follows, 'linearly independent' means 'linearly independent over K'.

Now let $z_1^{(1)}, \ldots, z_1^{(i_2-1)}$ be linearly independent solutions of the first equation of (5.2) (recall that we can use the variables y_i and z_i interchangeably for $i \in I$). Define $z_2^{(1)}, \ldots, z_{i_2-1}^{(1)}$ from $z_1^{(1)}$ via (5.4). If we plug $z_1^{(1)}, \ldots, z_{i_2-1}^{(1)}$ into the second equation of (5.2), we can find a solution $z_{i_2}^{(1)}$. We use (5.4) to compute the components $i_2+1, \ldots, i_3-1$, and so on. Thus we have found a vector (throughout this proof, we use bold letters for vectors)

$$\mathbf{z}^{(1)} = \left(z_1^{(1)}, \ldots, z_n^{(1)}\right)^t$$

s.t.

$$\mathbf{y}^{(1)} := T^{-1}\mathbf{z}^{(1)}$$

where T is as in (5.5) solves (5.32). Analogously we construct the vectors $\mathbf{z}^{(2)}, \ldots, \mathbf{z}^{(i_2-1)}$ and $\mathbf{y}^{(2)}, \ldots, \mathbf{y}^{(i_2-1)}$.

Next we set $z_1 = \ldots = z_{i_2-1} = 0$ in the second equation of (5.2) and solve it, obtaining $l_2 = i_3 - i_2$ linearly independent solutions $z_{i_2}^{(i_2)}, \ldots, z_{i_2}^{(i_3-1)}$. We proceed as above to obtain the vectors $\mathbf{z}^{(i_2)}, \ldots, \mathbf{z}^{(i_3-1)}$ and $\mathbf{y}^{(i_2)}, \ldots, \mathbf{y}^{(i_3-1)}$. Analogously we define $\mathbf{z}^{(j)}$ and $\mathbf{y}^{(j)}$, $i_3 \leq j \leq n$. If we let Z be the matrix whose columns are $\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(n)}$, then each of the columns of $T^{-1}Z$ solves

(5.32). Z is of the form

$$\begin{pmatrix}
z_1^{(1)} & \dots & z_1^{(i_2-1)} & 0 & \dots & 0 & \dots & & & \dots & 0 \\
\vdots & & \vdots & \vdots & & \vdots & & & & & \vdots \\
 & & & 0 & & 0 & & & & & \\
z_{i_2}^{(1)} & \dots & z_{i_2}^{(i_2-1)} & z_{i_2}^{(i_2)} & \dots & z_{i_2}^{(i_3-1)} & 0 & \dots & & & \\
\vdots & & \vdots & \vdots & & \vdots & \vdots & & & & \\
 & & & z_{i_3-1}^{(i_2)} & \dots & z_{i_3-1}^{(i_3-1)} & 0 & \dots & & & \\
 & & & \vdots & & \vdots & \ddots & & \vdots & & \vdots \\
 & & & & & & & \ddots & 0 & \dots & 0 \\
 & & & & & & & & z_{i_s}^{(i_s)} & \dots & z_{i_s}^{(n)} \\
\vdots & & & & & & & & \vdots & & \vdots \\
z_n^{(1)} & \dots & & & & & & \dots & z_n^{(i_s)} & \dots & z_n^{(n)}
\end{pmatrix}. \tag{5.33}$$

Suppose that

$$\sum_{k=1}^{n} \lambda_k \mathbf{y}^{(k)} = 0$$

for some $\lambda_k \in K$. Clearly, this implies

$$\sum_{k=1}^{n} \lambda_k \mathbf{z}^{(k)} = 0$$

and hence, by (5.33),

$$\sum_{k=i_t}^{i_{t+1}-1} \lambda_k z_{i_t}^{(k)} = 0, \qquad t = 1, \dots, s.$$

(Once again, we set $i_{s+1} = n+1$.) Since each of the sets $\left\{ z_{i_t}^{(i_t)}, \dots, z_{i_t}^{(i_{t+1}-1)} \right\}$ is linearly independent, we have $\lambda_k = 0$, $1 \le k \le n$, which establishes the linear independency of the $\mathbf{y}^{(k)}$. □

Corollary 5.4 *Let K be a field, $A \in \mathrm{Mat}(n, \mathcal{R}(K))$ be a matrix of rational functions over K and Δ be the forward difference operator on $\mathbb{K} = \mathcal{R}(K)$. Let further α_{kj} as in (5.2) be obtained by applying Abramov and Zima's algorithm to the system of difference equations*

$$\Delta y = Ay, \quad y \in \mathcal{S}(K)^n. \tag{5.34}$$

If $\alpha_{k0} \neq 0$ for $1 \le k \le s$, then (5.34) has n K-linearly independent solution vectors in $\mathcal{S}(K)^n$.

Proof. The fact that (5.30) has a solution is obvious in the difference case, and the second requirement of the preceding theorem is satisfied by appealing to theorem (8.2.1) of [20], which asserts that a difference equation $\sum_{j=0}^{l} a_j E^j y = 0$ has exactly l K-linearly independent solutions in $\mathcal{S}(K)$, if the a_j are polynomials from $K[x]$ and a_0, a_l are nonzero. □

5.1.6 Complexity

In this section we consider the number of operations the uncoupling algorithm by Abramov and Zima performs. First we count the multiplications in $\mathbb{K}$. From algorithm (5.1) we read off that in the first stage, for $i+1 \notin I$ the i-th pass of the outer loop requires

$$\begin{aligned} n-i-1+(n-i)^2+n-i+(n-i)(n+1) \\ = 2n^2-3ni+i^2+3n-3i-1 \end{aligned}$$

multiplications. If $i+1 \in I$, that is, no variable with nonzero coefficient was found, no operations are performed in this pass of the outer loop. Hence we need

$$\sum_{\substack{i=1 \\ i+1 \notin I}}^{n-1} \left(2n^2-3ni+i^2+3n-3i-1\right)$$

multiplications in all. The number of additions in the i-th pass of the outer loop (again we assume $i+1 \notin I$) is

$$\begin{aligned} (n-i)(n-i-1)+(n-i)(n+1) \\ = 2n^2-3in+i^2, \end{aligned}$$

yielding a total of

$$\sum_{\substack{i=1 \\ i+1 \notin I}}^{n-1} \left(2n^2-3in+i^2\right)$$

additions. In the second stage, we need

$$\sum_{k=1}^{s} \sum_{m=1}^{l_k-1} \left(i_k(1+m-1)+1+(l_k-m-1)(i_k m+1)\right)$$

multiplications and

$$\sum_{k=1}^{s} \sum_{m=1}^{l_k-1} \left(i_k(2+3(m-1))+3+(l_k-m-1)(i_k m+1)\right)$$

additions.

These observations lead to

Theorem 5.5 *(nondegenerate case) In the nondegenerate case $I = \{1\}$, algorithm (5.1) performs*

$$n^3 + O(n^2)$$

multiplications in $\mathbb{K}$. The same is true for the number of additions in $\mathbb{K}$.

Proof: As we have seen above, the number of multiplications and additions of the first stage are each of the form

$$\begin{aligned}\sum_{i=1}^{n-1} \left(2n^2 - 3ni + i^2 + O(n)\right) &= 2n^2\,(n-1) - 3n\sum_{i=1}^{n-1} i + \sum_{i=1}^{n-1} i^2 + O(n)^2 \\ &= \frac{5}{6}n^3 + O(n^2).\end{aligned}$$

As for the second stage, we have $s = 1$, $l_1 = n$ and therefore we need

$$\begin{aligned}\sum_{m=1}^{n-1} \left(m + 1 + (n-m-1)\,(m+1)\right) &= (n-1)\,n + (n-1)\sum_{m=1}^{n-1} m - \sum_{m=1}^{n-1} m^2 \\ &= \frac{n^3}{6} + O(n^2)\end{aligned}$$

multiplications and

$$\sum_{m=1}^{n-1} \left(3m + 2 + (n-m-1)\,(m+1)\right) = \frac{n^3}{6} + O(n^2)$$

additions. Adding the complexities of the first and the second stage gives the desired result. □

5.2 A Variant of Zürcher's Algorithm

5.2.1 A Block Triangular Normal Form for Pseudo-Linear Maps

As in section (3.1) we consider the equation

$$\theta y = r$$

or, equivalently,

$$T\sigma(y) + \delta y = r.$$

Zürcher's algorithm reduces the matrix of a pseudo-linear map to a block-diagonal matrix, where the blocks are companion matrices. In this chapter we present a variant, which we will call 'incomplete Zürcher's algorithm', that omits some computations in the process of obtaining the normal form,

thus computing only a blocktriangular matrix. The latter will be of the same form as in the uncoupling algorithm by Abramov and Zima. The price will be a more complicated computation for the scalar equations, which also resembles Abramov and Zima's algorithm.

We start with a 'reduced' version of lemma (3.2):

Lemma 5.6 *Let the matrix of θ be of the form*

$$T = \begin{matrix} \\ i \to \end{matrix} \begin{pmatrix} & & & \overset{i}{\downarrow} & & & & \\ * & 1 & & 0 & 0 & \dots & \dots & 0 \\ \vdots & & \ddots & & \vdots & & & \vdots \\ * & \dots & & 1 & 0 & \dots & \dots & 0 \\ * & \dots & \dots & * & * & \dots & \dots & * \\ * & \dots & \dots & * & * & \dots & \dots & * \\ \vdots & & & \vdots & \vdots & & & \vdots \\ \vdots & & & \vdots & \vdots & & & \vdots \\ * & \dots & \dots & * & * & \dots & \dots & * \end{pmatrix} \quad \text{with} \quad i < n. \tag{5.35}$$

If there is an element $t_{il} \neq 0$ with $i < l \leq n$, then there is a basis change A s.t.

$$A^{-1}T\sigma(A) + A^{-1}\delta A = \begin{matrix} \\ i+1 \to \end{matrix} \begin{pmatrix} & & & & \overset{i+1}{\downarrow} & & & \\ * & 1 & & & 0 & 0 & \dots & 0 \\ \vdots & & \ddots & & & \vdots & & \vdots \\ \vdots & & & \ddots & & \vdots & & \vdots \\ * & \dots & & & 1 & 0 & \dots & 0 \\ * & \dots & & \dots & * & * & \dots & * \\ * & \dots & & \dots & * & * & \dots & * \\ \vdots & & & & \vdots & \vdots & & \vdots \\ * & \dots & & \dots & * & * & \dots & * \end{pmatrix}.$$

Proof. We show how A can be constructed as a product of elementary matrices. To keep notation simple, the associated matrices of θ that occur in the intermediate steps are denoted again by $T = (t_{ik})$. First, by the basis change $P_{i+1,l}$, $t_{i,i+1}$ becomes nonzero. The remaining entries of the affected columns $i+1$ and l are either 0 (rows $1, \dots, i-1$) or not of interest (rows $i+1, \dots, n$). P2 does not change the ordered part of T either.

Now we can perform the basis change $D_{i+1}(\sigma^{-1}(t_{i,i+1}^{-1}))$. D1 sets $t_{i,i+1}$ to 1, and D2 and D3 do not modify lines $1, \dots, i$.

What remains to do is to set t_{ik}, $k = i+2, \dots, n$, to 0. Suppose we have

done this up to $k < m$:

$$T = \begin{array}{c} \\ i \to \end{array} \begin{array}{c} \begin{array}{cccccccccccc} & & & & i & & & & & m & & \\ & & & & \downarrow & & & & & \downarrow & & \end{array} \\ \begin{pmatrix} * & 1 & & & 0 & 0 & \dots & & & & \dots & 0 \\ \vdots & & \ddots & & & \vdots & & & & & & \vdots \\ * & \dots & & & 1 & 0 & \dots & & & & \dots & 0 \\ * & \dots & \dots & & * & 1 & 0 & \dots & 0 & * & \dots & * \\ * & \dots & \dots & & * & * & \dots & & & & \dots & * \\ \vdots & & & & \vdots & \vdots & & & & & & \vdots \\ \vdots & & & & \vdots & \vdots & & & & & & \vdots \\ * & \dots & \dots & & * & * & \dots & & & & \dots & * \end{pmatrix} \end{array}.$$

The basis change $C_{i+1,m}(\sigma^{-1}(-t_{im}))$ sets t_{im} to 0 by operation C1. It is easy to check that C1, C2 and C3 do not change the ordered part of T. □

Next we take a look at how changes of bases modify block triangular matrices.

Lemma 5.7 *Let T_{11}, T_{21}, T_{22} be matrices over $\mathbb{K}$ of sizes $n_1 \times n_1$, $n_2 \times n_1$ and $n_2 \times n_2$, respectively. Let further $\theta : \mathbb{K}^{n_1+n_2} \to \mathbb{K}^{n_1+n_2}$ be the pseudo-linear map whose matrix w.r.t. the canonical basis is $\left(\begin{smallmatrix} T_{11} & 0 \\ T_{12} & T_{22} \end{smallmatrix}\right)$ and A be an invertible $n_2 \times n_2$ matrix. Then the basis change $\left(\begin{smallmatrix} I & 0 \\ 0 & A \end{smallmatrix}\right)$ turns the matrix of θ into*

$$\begin{pmatrix} T_1 & 0 \\ A^{-1}T_{21} & A^{-1}T_{22}\sigma(A) + A^{-1}\delta A \end{pmatrix}.$$

Proof. Because of formula (2.16), the matrix that we seek is

$$\begin{aligned} \begin{pmatrix} I & 0 \\ 0 & A^{-1} \end{pmatrix} \begin{pmatrix} T_{11} & 0 \\ T_{21} & T_{22} \end{pmatrix} \sigma\left(\begin{pmatrix} I & 0 \\ 0 & A \end{pmatrix}\right) + \begin{pmatrix} I & 0 \\ 0 & A^{-1} \end{pmatrix} \delta \begin{pmatrix} I & 0 \\ 0 & A \end{pmatrix} \\ = \begin{pmatrix} T_{11} & 0 \\ A^{-1}T_{21} & A^{-1}T_{22}\sigma(A) \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & A^{-1}\delta A \end{pmatrix} \\ = \begin{pmatrix} T_1 & 0 \\ A^{-1}T_{21} & A^{-1}T_{22}\sigma(A) + A^{-1}\delta A \end{pmatrix}, \end{aligned}$$

where $\sigma(I) = I$ and $\delta I = 0$ follow from corollary (2.13). □

This means that the lower left block of T has to be updated as well whenever we perform a basis change to modify the lower right block.

After these preparations we can formulate and prove

Theorem 5.8 *Let V be an n-dimensional vector space, $\theta : V \to V$ be pseudo-linear w.r.t. σ and δ, where σ is an automorphism. Then there is*

a basis of V such that the matrix of θ w.r.t. to this basis is of the block triangular form

$$\begin{pmatrix} A_1 & & 0 \\ & \ddots & \\ * & & A_m \end{pmatrix},$$

where the A_i, $1 \leq i \leq m$, are of the form

$$\begin{pmatrix} * & 1 & 0 & \dots & 0 \\ \vdots & & \ddots & \ddots & \vdots \\ * & \dots & * & 1 & 0 \\ * & \dots & & * & 1 \\ * & \dots & & & * \end{pmatrix}. \tag{5.36}$$

Proof. Induction on n. For $n = 1$ there is nothing to show.

Suppose the assertion of the theorem holds for $1, \dots, n-1$. By taking $i = 1$ if necessary, T is of the form (5.35). Let i be the size of the upper left block of type (5.36). If $i = n$, we are done. If $i < n$ and there is an element $t_{ik} \neq 0$ for some $i < k \leq n$, we can increase the size of the upper left block to $i + 1$ by lemma (5.6). If, on the other hand, all these entries are zero, we apply the induction hypothesis to the lower right part of T. The basis changes needed to bring it into the desired form do not interfere with the ordered part of T by lemma (5.7). □

This proof gives rise to the following algorithm to find the block triangular normal form. The main procedure is very simple: If we do not find the nonzero matrix entry necessary to increase the upper left block of form (5.36) by lemma (5.6), we just move on to the next line. There are two differences between `transformL5.4` and `transformL2` from algorithm (3.7): Firstly, `transformL5.4` does not erase the elements t_{ik}, $i_0 \leq k \leq i$. Secondly, during the base changes $D_{i+1}(\sigma^{-1}(t_{i,i+1}^{-1}))$ and $C_{i+1,k}(\sigma^{-1}(-t_{ik}))$ the lower left part of T has to be modified according to lemma (5.7). The procedure `transformP` is the same as in algorithm (3.7).

Algorithm 5.9 `triangForm`

```
triangForm(T, σ, δ)
     n := Size(T)
     i0 := 1; i := 1
     B := identity matrix of size n
     while i < n
         j := i+1
         while j ≤ n and t_ij = 0
```

```
            j := j + 1
        if j ≤ n then
            transformLemma5.4(T,i0,i,j,σ,δ,B)
        else
            i0 := i + 1
        i := i + 1
    end
    return T,B
end


transformLemma5.4(T,i0,i,l,σ,δ,B)
    n := Size(T)
    transformP(T,i0,i+1,l,B)

    a := σ^{-1}(t^{-1}_{i,i+1})
    for j := i to n
        t_{j,i+1} := t_{j,i+1} * σ(a)     (* D1 *)
    for j := i0 to n
        t_{i+1,j} := t_{i+1,j} / a     (* D2 *)
    t_{i+1,i+1} := t_{i+1,i+1} + δ(a) / a     (* D3 *)
    for j := 1 to i0-1
        t_{i+1,j} := t_{i+1,j} / a     (* update lower left block of T *)
    for j := i0 to n
        b_{j,i+1} := a * b_{j,i+1}     (* basis change *)

    for k := i+2 to n
        a := σ^{-1}(-t_{ik})
        for j := i to n
            t_{jk} := t_{jk} + σ(a) * t_{j,i+1}     (* C1 *)
        for j := i0 to n
            t_{i+1,j} := t_{i+1,j} - a * t_{kj}     (* C2 *)
        t_{i+1,k} := t_{i+1,k} + δ(a)     (* C3 *)
        for j := 1 to i0-1
            (* update lower left block of T *)
            t_{i+1,j} := t_{i+1,j} - a * t_{k,j}
        for j := i0 to n
            b_{jk} := b_{jk} + a * b_{j,i+1}     (* basis change *)
    end
end                                                              □
```

5.2.2 Deduction of Scalar Equations

Differential Equations

For a differential system

$$Dy = Ty + v$$

(see section (3.1.2) for the relevant definitions) we proceed as in section (3.1.2), but we use algorithm (5.9) instead of Zürcher's algorithm. It returns (if the block triangular matrix has only one block) a system of the form

$$\begin{aligned} Dz_1 &= a_{11}z_1 + z_2 + r_1 \\ &\vdots \\ Dz_{n-1} &= a_{n-1,1}z_1 + \ldots + a_{n-1,n-1}z_{n-1} + z_n + r_{n-1} \\ Dz_n &= a_{n1}z_1 + \ldots + a_{nn}z_n + r_n. \end{aligned}$$

A scalar equation for z_1 can now be obtained in the same way as in the algorithm by Abramov and Zima: The first equation allows to express z_2 and Dz_2 by z_1, Dz_1 and D^2z_1:

$$\begin{aligned} z_2 &= Dz_1 - a_{11}z_1 - r_1, \\ Dz_2 &= D^2z_1 - a_{11}Dz_1 - Da_{11}z_1 - Dr_1. \end{aligned}$$

These expressions are plugged into equations $2, \ldots, n$, and so on, until we arrive at an n-th order uncoupled differential equation for z_1.

If the block-triangular matrix returned by algorithm (5.9) has several blocks, we solve the system corresponding to the first block as described above and continue with the second block, assuming that the variables of the first block are known, i.e., part of the inhomogeneity. This is completely analogous to the second stage of Abramov and Zima's algorithm, see there for details.

Difference Equations

Once again we use the same notation as in section (3.1.2). Using the transformations described there and algorithm (5.9), we can reduce a difference system

$$Ey = My + v$$

to an equivalent system

$$Ez = Pz + w,$$

where $P = (p_{ij})_{1 \le i,j \le n} = E(C) + I$ and C is the block-triangular matrix computed by algorithm (5.9). Assuming w.l.o.g. that C has only one block,

this system is of the form

$$\begin{aligned} Ez_1 &= p_{11}z_1 + z_2 + w_1 \\ &\vdots \\ Ez_{n-1} &= p_{n-1,1}z_1 + \ldots + p_{n-1,n-1}z_{n-1} + z_n + w_{n-1} \\ Ez_n &= p_{n1}z_1 + \ldots + p_{nn}z_n + w_n. \end{aligned}$$

The deduction of an uncoupled higher order equation for z_1 is now analogous to the differential case.

5.2.3 Complexity

The number of multiplications in $\mathbb{K}$ required by `transformL5.4` is

$$\begin{aligned} &(n-i+1) + (n-i_0+1) + 1 + (i_0-1) + (n-i_0+1) + \\ &(n-i+1)\left((n-i+1) + (n-i_0+1) + (i_0-1) + (n-i_0+1)\right). \end{aligned}$$

For the number of additions we find

$$1 + (n-i+1)\left((n-i+1) + (n-i_0+1) + 1 + (i_0-1) + (n-i_0+1)\right).$$

In both cases, if we set $i_0 = 1$ and neglect terms of order n, we obtain

$$(n-i)(3n-i) + O(n) = 3n^2 - 4in + i^2 + O(n),$$

which leads to

Theorem 5.10 *If the first* `if` *condition in algorithm (5.9) is always satisfied throughout the execution, the uncoupling algorithm 'incomplete Zürcher' requires*

$$\frac{3}{2}n^3 + O(n^2)$$

multiplications in $\mathbb{K}$*. The same is true for the number of additions.*

Proof. As seen above, the first stage (i.e., algorithm (5.9)) requires

$$\begin{aligned} &\sum_{i=1}^{n-1} \left(3n^2 - 4in + i^2 + O(n)\right) \\ = \;& (n-1)\,3n^2 - 4n\sum_{i=1}^{n-1} i + \sum_{i=1}^{n-1} i^2 + O(n^2) \\ = \;& \frac{4}{3}n^3 + O(n^2) \end{aligned}$$

multiplications (resp. additions). The second stage is the same as in Abramov/Zima's algorithm, where we found its complexity to be $\frac{n^3}{6} + O(n^2)$. □

Chapter 6

Implementation and Résumé

6.1 The Mathematica Package

We have implemented the four algorithms that we have described in the Mathematica package `OreSys.m`. The main functions are

```
UncoupleDifferentialSystem[equations,variables,helpvariables,x,options]
UncoupleDifferenceSystem[equations,variables,helpvariables,x,options]
UncoupleqDifferenceSystem[equations,variables,helpvariables,x,q,options]
UncoupleGeneralDifferenceSystem[equations,variables,helpvariables,
x,a,b,options]
UncoupleAlgebraicSystem[equations,variables,helpvariables,options].
```

For instance,

```
UncoupleDifferentialSystem[{y1'[x]==x y1[x]-y2[x]+1,
2 y2'[x]-y1[x]==1/(x-1)},{y1[x],y2[x]},{z1[x],z2[x]},x,
Method->AbramovZima]
```

or

```
UncoupleqDifferenceSystem[{y1[q x]==y1[x]/x+y2[x]-x^2,
y2[q x]==y1[x]+(x+1)/(x-1)y2[x]-1},{y1[x],y2[x]},{z1[x],z2[x]},x,q].
```

`helpvariables` are dummy variables that are used by Abramov/Zima, Zürcher and incomplete Zürcher to express the uncoupled system. They are not used (and need not be specified) if Gaussian elimination is applied.

`UncoupleGeneralDifferenceSystem` is for the case $\sigma(x) = ax + b$, $\delta = 0$, which generalizes the ordinary shift and the q-shift (a and b are constants, $a \neq 0$). `UncoupleAlgebraicSystem` uncouples linear algebraic, i.e., operator-free, systems.

By default all these functions use Abramov and Zima's uncoupling algorithm. The uncoupling algorithm can be modified with the option `Method`:

```
Method->AbramovZima
Method->Gauss
Method->Zuercher
Method->IncompleteZuercher.
```

6.2 Examples of Computation

Differential System

We give an example from physics. Consider an object with mass m that moves in a plane influenced by a force $F = (f_1, f_2)$. According to Newton's equation 'force = mass × acceleration', the position $y = y(x)$ of the object at time x satisfies:

$$F(y) = my''.$$

In two dimensions, this reads:

$$\begin{aligned} f_1(y_1(x), y_2(x)) &= my_1''(x) \\ f_2(y_1(x), y_2(x)) &= my_2''(x). \end{aligned}$$

Let us assume that f_1, f_2 are linear and set $u_1 = y_1'$, $u_2 = y_2'$. For the sake of concreteness, set $m = 1$ and

$$\begin{aligned} f_1(y_1, y_2) &= (x^2 - 1)y_1 + \frac{1}{x}y_2, \\ f_2(y_1, y_2) &= -xy_1 + \frac{2}{x-1}y_2. \end{aligned}$$

Now we have arrived at the linear first order differential system

$$\begin{aligned} u_1'(x) &= (x^2 - 1)y_1(x) + \frac{1}{x}y_2(x) \\ u_2'(x) &= -xy_1(x) + \frac{2}{x-1}y_2(x) \\ y_1'(x) &= u_1(x) \\ y_2'(x) &= u_2(x). \end{aligned}$$

We give the output of each of the four algorithms from the preceding chapters applied to this system.

Zürcher:

$$\frac{6-8x+27x^2-12x^3+4x^4}{-3x^2+2x^3}z_1(x)+\frac{-6+4x+22x^2-60x^3+52x^4-14x^5}{-3x+8x^2-7x^3+2x^4}z_1'(x)$$
$$+\frac{-6+14x+x^2-9x^3-x^4+5x^5-2x^6}{3x^2-5x^3+2x^4}z_1''(x)+\frac{-6+2x}{-3x+2x^2}z_1^{(3)}(x)+z_1^{(4)}(x)=0,$$
$$z_2(x)=z_1'(x),$$
$$z_3(x)=z_2'(x),$$
$$z_4(x)=z_3'(x),$$

$$u_1(x)=z_1(x),$$
$$u_2(x)=\frac{-2+9x^2+2x^3-15x^4-2x^5}{x(-3+2x)}z_1(x)+\frac{6-6x-8x^2+4x^3+3x^4-3x^5}{(-1+x)(-3+2x)}z_2(x)$$
$$+\frac{-2+3x^2+2x^3}{x(-3+2x)}z_3(x)+\frac{-1+3x^2}{-3+2x}z_4(x),$$
$$y_1(x)=\frac{2-1+3x^2}{x(-3+2x)}z_1(x)+\frac{3-x-x^2+x^3}{(-1+x)(-3+2x)}z_2(x)-\frac{2}{x(-3+2x)}z_3(x)-\frac{1}{-3+2x}z_4(x),$$
$$y_2(x)=\frac{-2(-1+x)(1+x)\left(-1+3x^2\right)}{-3+2x}z_1(x)-\frac{x\left(6-2x^2+x^4\right)}{-3+2x}z_2(x)$$
$$+\frac{2(-1+x)(1+x)}{-3+2x}z_3(x)+\frac{(-1+x)x(1+x)}{-3+2x}z_4(x).$$

Abramov, Zima:

$$\frac{-2-17x-2x^2+30x^3+26x^4+31x^5-22x^6+4x^7}{x+2x^2+2x^3-4x^4-3x^5+2x^6}z_1(x)$$
$$+\frac{5+2x-29x^2-2x^3+53x^4+60x^5+33x^6-104x^7-54x^8+58x^9+24x^{10}-14x^{11}}{x+2x^2-8x^4-6x^5+12x^6+8x^7-8x^8-3x^9+2x^{10}}z_1'(x)$$
$$+\frac{2+3x-18x^2-24x^3+18x^4+14x^5-8x^6-12x^7+8x^8+3x^9-2x^{10}}{-x-2x^2-x^3+6x^4+5x^5-6x^6-3x^7+2x^8}z_1''(x)$$
$$+\frac{2+2x-4x^2-4x^3-6x^4+2x^5}{x+2x^2+2x^3-4x^4-3x^5+2x^6}z_1^{(3)}(x)+z_1^{(4)}(x)=0,$$

$$z_2(x)=z_1'(x),$$
$$z_3(x)=\left(1-x^2\right)z_1(x)-\frac{2x}{-1+x^2}z_2(x)+z_2'(x),$$
$$z_4(x)=\frac{1}{x^2-1}z_2(x)-\frac{1-3x^2}{x(-1+x^2)}z_3(x)+z_3'(x),$$

$$u_1(x)=z_1(x),$$
$$\left(x^2-1\right)u_2(x)+\frac{1}{x}y_2(x)=z_2(x),$$
$$\frac{1}{x}y_1(x)-\frac{2}{-1+x^2}y_2(x)=z_3(x),$$
$$\frac{-1-2x-3x^2+2x^3}{(-1+x)^2x(1+x)^2}y_2(x)=z_4(x).$$

Gauss:

$$\begin{gathered}
\left(1-x^2\right) y_1(x) - \tfrac{1}{x} y_2(x) + u_1'(x) = 0, \\
x y_1(x) - \tfrac{2}{-1+x} y_2(x) + u_2'(x) = 0, \\
\left(1-x^2\right) y_1(x) - \tfrac{1}{x} y_2(x) + y_1''(x) = 0, \\
\tfrac{4-12x+15x^2-7x^3+3x^4+3x^5-2x^6}{-x^3+3x^4-3x^5+x^6} y_2(x) + \tfrac{4-8x}{x^2-2x^3+x^4} y_2'(x) + \tfrac{2-2x+3x^2-x^3-x^4+x^5}{-x^3+x^4} y_2''(x) \\
+ \tfrac{2}{x^2} y_2^{(3)}(x) - \tfrac{1}{x} y_2^{(4)}(x) = 0.
\end{gathered}$$

Incomplete Zürcher:

$$\begin{gathered}
\tfrac{6-8x+27x^2-12x^3+4x^4}{-3x^2+2x^3} z_1(x) + \tfrac{-6+4x+22x^2-60x^3+52x^4-14x^5}{-3x+8x^2-7x^3+2x^4} z_1'(x) \\
+ \tfrac{-6+14x+x^2-9x^3-x^4+5x^5-2x^6}{3x^2-5x^3+2x^4} z_1''(x) + \tfrac{-6+2x}{-3x+2x^2} z_1^{(3)}(x) + z_1^{(4)}(x) = 0, \\
z_2(x) = z_1'(x), \\
z_3(x) = \left(1-x^2\right) z_1(x) - \tfrac{2x}{-1+x^2} z_2(x) + z_2'(x), \\
z_4(x) = \tfrac{1}{x^2-1} z_2(x) + \tfrac{2\left(-1+2x^2\right)}{x\left(-1+x^2\right)} z_3(x) + z_3'(x),
\end{gathered}$$

$$\begin{gathered}
u_1(x) = z_1(x), \\
u_2(x) = x z_3(x) + \tfrac{-1+3x^2}{-3+2x} z_4(x), \\
y_1(x) = \tfrac{1}{-1+x^2} z_2(x) - \tfrac{1}{-3+2x} z_4(x), \\
y_2(x) = \tfrac{(-1+x)x(1+x)}{-3+2x} z_4(x).
\end{gathered}$$

Difference System

We apply the four uncoupling algorithms to the following (arbitrarily chosen) system of difference equations:

$$\begin{aligned}
y_1(x+1) &= y_1(x) + y_2(x) + y_3(x) - y_4(x) \\
y_2(x+1) &= -x y_1(x) + y_2(x) + y_3(x) + x \\
y_3(x+1) &= y_1(x) + \frac{1}{x} y_4(x) \\
y_4(x+1) &= y_2(x) + y_4(x) - 1.
\end{aligned}$$

Zürcher:

$$\frac{-5-7x-x^2-x^3-x^4}{(-1+x)(1+x)(2+x)}+\frac{-x-2x^2}{-1+x^2}z_1(x)+\frac{-3+x+6x^2+6x^3+2x^4}{-2-x+2x^2+x^3}z_1(x+1)$$
$$+\frac{4+x-7x^2-6x^3-x^4}{-2-x+2x^2+x^3}z_1(x+2)+\frac{-3+3x+3x^2}{-2+x+x^2}z_1(x+3)-z_1(x+4)=0,$$
$$z_2(x)=-z_1(x)+z_1(x+1),$$
$$z_3(x)=-1-x-z_2(x)+z_2(x+1),$$
$$z_4(x)=-z_3(x)+\frac{1+x+x^2+z_3(x+1)+xz_3(x+1)}{1+x},$$

$$y_1(x)=z_1(x),$$
$$y_2(x)=-\frac{x^2z_1(x)}{1+x}+\frac{(-1-x)z_2(x)}{-1+x}+\frac{\left(-1-x-x^2\right)z_3(x)}{(-1+x)(1+x)}+\frac{z_4(x)}{1-x},$$
$$y_3(x)=\frac{\left(-x+x^2\right)z_1(x)}{1+x}-xz_2(x)-\frac{z_3(x)}{1+x}-z_4(x),$$
$$y_4(x)=-\frac{xz_1(x)}{1+x}-\frac{x(1+x)z_2(x)}{-1+x}-\frac{x(2+x)z_3(x)}{(-1+x)(1+x)}-\frac{xz_4(x)}{-1+x}.$$

Gauss:

$$-y_1(x)+y_1(x+1)-y_2(x)-y_3(x)+y_4(x)=0,$$
$$\frac{(-1-x)y_2(x)}{x}+\frac{(1+2x)y_2(x+1)}{x+x^2}+\frac{y_2(x+2)}{-1-x}+\frac{(-1-x)y_3(x)}{x}+\frac{y_3(x+1)}{x+1}+y_4(x)=0,$$
$$y_3(x+1)+\frac{(-2-x)y_3(x+2)}{x}+\frac{(1+3x)y_3(x+3)}{x+x^2}+\frac{y_3(x+4)}{-1-x}+\frac{(-1-2x)y_4(x)}{x^2}$$
$$+\frac{\left(3+7x+3x^2\right)y_4(x+1)}{x+2x^2+x^3}+\frac{\left(-1-5x-x^2\right)y_4(x+2)}{2x+3x^2+x^3}+\frac{y_4(x+3)}{3+4x+x^2}=0,$$
$$\frac{-268-750x-848x^2-487x^3-139x^4-19x^5-x^6}{(2+x)(-10+40x+73x^2+43x^3+11x^4+x^5)(158+364x+278x^2+97x^3+16x^4+x^5)}$$
$$+\frac{1+3x+2x^2}{-10x+40x^2+73x^3+43x^4+11x^5+x^6}y_4(x)$$
$$+\frac{\left(-888-4626x-8338x^2-7877x^3-4466x^4-1574x^5-337x^6-40x^7-2x^8\right)y_4(x+1)}{-3160+3780x+49308x^2+110346x^3+126324x^4+89058x^5+41429x^6+13013x^7+2738x^8+370x^9+29x^{10}+x^{11}}$$
$$+(\text{some terms skipped})+$$
$$+\frac{\left(-10+366x+929x^2+850x^3+367x^4+76x^5+6x^6\right)y_4(x+6)}{-3160+3780x+49308x^2+110346x^3+126324x^4+89058x^5+41429x^6+13013x^7+2738x^8+370x^9+29x^{10}+x^{11}}$$
$$+\frac{(-1-x)y_4(x+7)}{316+886x+920x^2+472x^3+129x^4+18x^5+x^6}=0.$$

Abramov, Zima:

$$\frac{5+7x+x^2+x^3+x^4}{(-1+x)(1+x)(2+x)} + \frac{\left(x+2x^2\right)z_1(x)}{-1+x^2} + \frac{\left(3-x-6x^2-6x^3-2x^4\right)z_1(x+1)}{-2-x+2x^2+x^3}$$
$$+\frac{\left(-4-x+7x^2+6x^3+x^4\right)z_1(x+2)}{-2-x+2x^2+x^3} + \frac{\left(3-3x-3x^2\right)z_1(x+3)}{-2+x+x^2} + z_1(4+x) = 0,$$

$$\begin{aligned} z_2(x) &= -z_1(x) + z_1(x+1), \\ z_3(x) &= -1 - x - (1-x)\, z_1(x) + z_2(x+1), \\ z_4(x) &= -\tfrac{x}{1+x} - z_1(x) + \tfrac{x z_2(x)}{1+x} - \tfrac{x z_3(x)}{1+x} + z_3(x+1), \end{aligned}$$

$$\begin{aligned} y_1(x) &= z_1(x), \\ y_2(x) + y_3(x) - y_4(x) &= z_2(x), \\ y_3(x) + \tfrac{(1-x)y_4(x)}{x} &= z_3(x), \\ \tfrac{(1-x)y_4(x)}{x} &= z_4(x). \end{aligned}$$

Incomplete Zürcher:

$$\frac{5+7x+x^2+x^3+x^4}{(-1+x)(1+x)(2+x)} + \frac{\left(x+2x^2\right)z_1(x)}{-1+x^2} + \frac{\left(3-x-6x^2-6x^3-2x^4\right)z_1(x+1)}{-2-x+2x^2+x^3}$$
$$+\frac{\left(-4-x+7x^2+6x^3+x^4\right)z_1(x+2)}{-2-x+2x^2+x^3} + \frac{\left(3-3x-3x^2\right)z_1(x+3)}{-2+x+x^2} + z_1(x+4) = 0,$$
$$\begin{aligned} z_2(x) &= -z_1(x) + z_1(x+1), \\ z_3(x) &= -1 - x - (1-x)\, z_1(x) + z_2(x+1), \\ z_4(x) &= -\tfrac{x}{1+x} - z_1(x) + \tfrac{x z_2(x)}{1+x} - \tfrac{x}{x+1} z_3(x) + z_3(x+1), \end{aligned}$$

$$\begin{aligned} y_1(x) &= z_1(x), \\ y_2(x) &= z_2(x) - z_3(x) + \tfrac{z_4(x)}{1-x}, \\ y_3(x) &= z_3(x) - z_4(x), \\ y_4(x) &= -\tfrac{x z_4(x)}{-1+x}. \end{aligned}$$

6.3 Comparison of the Methods

It is not easy to give some general hint on what uncoupling algorithm to use. After trying our implementation on many example systems, the best strategy for some particular input seems to be trying several algorithms to figure out which one gives the best result (i.e., the uncoupled equations with smallest order/smallest coefficients, or the shortest running time). Our results on complexity need not be significant for the small dimensional systems where uncoupling is possible; if n is large (say, $n > 15$), none of the available algorithms will uncouple the system in reasonable time and with reasonably sized output.

Gaussian elimination usually gives complicated uncoupled equations and is the only algorithm where the order of the uncoupled equation can be larger than the dimension of the system. However, the differential system from the last section shows that neither of these shortcomings happens always.

Zürcher's algorithm and our 'incomplete' variant have the minor technical restriction that σ must be surjective. Empirical evidence shows that they both return the same uncoupled higher order equation for the first variable z_1, provided that in both algorithms we are in the nondegenerate case, where the block diagonal (resp. block triangular) matrix has only one block. The situation changes when one (or both) of these two algorithms runs into the degenerate case where the system splits into several blocks. This is more likely to occur when incomplete Zürcher is applied than with Zürcher's algorithm, because Zürcher's algorithm tries to increase the current companion block by applying lemma (3.5) if lemma (3.2) is not applicable. On the other hand, incomplete Zürcher immediately proceeds with the next block if lemma (5.6), which corresponds to lemma (3.2), cannot be applied. Consequently, there are systems that are decomposed into several blocks by incomplete Zürcher, but not with Zürcher's algorithm. This is desirable because several scalar equations of small order are easier to handle than one equation of large order.

Bibliography

[1] Abramov, S. A. (1995): Rational solutions of linear difference and q-difference equations with polynomial coefficients, *Proc. ISSAC '95*, ACM Press, 285-289

[2] Abramov, S. A., and Barkatou, M. A. (1998): Rational Solutions of first order difference systems, *Proc. ISSAC '98*, ACM Press, 124-131.

[3] Abramov, S. A., Bronstein, M. and Petkovšek, M. (1995): On polynomial solutions of linear operator equations, *Proc. ISSAC '95*, ACM Press, 290-296.

[4] Abramov, S. A., and Petkovšek, M. (1994): D'Alembertian solutions of linear differential and difference equations, *Proc. ISSAC '94*, ACM Press, 169-174.

[5] Abramov, S. A., Paule, P., and Petkovšek, M. (1998): q-Hypergeometric solutions of q-difference equations, *Discrete Math.* 180, 3-22.

[6] Abramov, S. A., and Zima, E. V. (1996): A universal program to uncouple linear systems, *Proceedings of the International Conference on Computational Modelling and Computing in Physics, Dubna, Russia, September 16-21, 1996*, 16-26.

[7] Barkatou, M. A. (1993): An algorithm for computing a companion block diagonal form for a system of linear differential equations, *Appl. Algebra Eng. Commun. Comput.* 4, 185-195.

[8] Barkatou, M. A. (1997): On rational solutions of systems of linear differential equations, *J. Symbolic Computation* 11, 1-21.

[9] Bronstein, M. and Petkovšek, M. (1994) On Ore rings, linear operators and factorisation, *Programming and Computer Software* 20, 14-26.

[10] Bronstein, M. and Petkovšek, M. (1995): An introduction to pseudo-linear algebra, *Theoretical Computer Science* 157, 3-33.

[11] Chyzak, F. (2000): An extension of Zeilberger's fast algorithm to general holonomic functions, *Discr. Math.* 217, 115-234

[12] Chyzak, F. and Salvy, B. (1998): Non-commutative elimination in Ore algebras proves multivariate holonomic identities, *J. Symbolic Comp.* 26(2), 187-227.

[13] Cohn, P.M. (1971): Free rings and their relations, *London Mathematical Society Monographs, No. 2*, Academic Press, New York.

[14] Danilewski, A. 1937: The numerical solution of the secular equation (in Russian), *Mat. Sbornik* 2, 169-171.

[15] Jacobson, N (1937): Pseudo-linear transformations, *Annals of Mathematics* 38, 484-507.

[16] Knuth, D. E. (1998): The Art of Computer Programming Vol. 2 (Seminumerical Algorithms), *Third Edition*, Addison-Weskey, Reading, Massachusetts.

[17] Li, Zi-Ming (1996): A subresultant theory for linear differential, linear difference and Ore polynomials, with applications, *Ph. D. thesis, RISC-Linz, Johannes Kepler Universität Linz*

[18] Murray, F. J., and Miller, K. S. (1954): Existence theorems for ordinary differential equations, New York Univ. Press, Intersciences, New York.

[19] Ore, O. (1933): The theory of non-commutative polynomials, *Annals Of Mathematics* 34, 480-508.

[20] Petkovšek, M., Wilf, H., and Zeilberger, D. (1996): A=B, *A K Peters, Wellesley MA.*

[21] Weixlbaumer, C. (2001): Solutions of Difference Equations with Polynomial Coefficients, *Diplomarbeit, RISC-Linz, Johannes Kepler Universität Linz*

[22] Zeilberger, D. (1990): A holonomic systems approach to special function identities, *J. Comput. Appl. Math.* 32, 321-368.

[23] Zeilberger, D. (1990): A fast algorithm for proving terminating hypergeometric identities, *Discrete Math.* 80, 207-211.

[24] Zeilberger, D. (1991): The method of creative telescoping, *J. Symbolic Computation* 11, 195-204.

[25] Zürcher, B. (1994): Rationale Normalformen von pseudo-linearen Abbildungen (in German), *Diplomarbeit, Mathematik, ETH Zürich.*

6.4 Curriculum Vitae

29/6/1978: born in Linz, Austria

1988 - 1996: Akademisches Gymnasium (secondary school) in Linz

1996: Matura (undergraduate degree) with distinction

1996 - 2002: diploma study technical mathematics at Johannes Kepler University Linz

2001 - 2002: civil service in Linz

Professional experience:

August, september 2000: Internship at Siemens AG, Vienna (Department for program and system development)

August, september 2001: Internship at eRunway Ltd., Colombo, Sri Lanka (Department for research and development)