2011 50th IEEE Conference on Decision and Control and
European Control Conference (CDC-ECC)
Orlando, FL, USA, December 12-15, 2011

A Toolbox for the Analysis of Linear Systems with Delays

Felix Antritter Johannes Middeke

Abstract— **This paper considers linear time varying control systems with delays. Roughly speaking, a so called π-flat output of such a system, if it exists, allows to parameterize the states and inputs of the system using derivatives and delays of the π-flat output. Additionally, also predictions of the π-flat output are allowed, which are characterized by a prediction operator π. We present a toolbox for the computer algebra system *Maple*, which implements the algorithm recently been proposed in [1] for the computation of π-flat outputputs. We also give an equivalent description of hyper-regularity of polynomial matrices using one-sided inverses. This allowed us to replace the transformation to Smith-Jacobson form by the efficient method of row-/column-reduction for checking hyper-regularity. The implementation of the toolbox is illustrated and its application is explained by means of examples.**

I. Introduction

The concept of *differential flatness* [2], [3], [4] has become a very important tool in applied linear and nonlinear control (see e.g. [5], [6], [7] for many interesting applications). Extensions of this concept to time delay systems have been proposed and discussed by many authors (see e.g. [8], [9], [1]; other approaches may be found, e.g., in [10], [11]).

In this contribution we consider linear time varying delay systems like the following example from [1] with the states x_1, x_2, the input u and the time varying coefficient $s(t)$

$$\begin{aligned} \dot{x}_1(t) - s(t)(x_2(t-\tau) - x_2(t-2\tau)) &= 0 \\ \dot{x}_2(t) &= u(t-\tau)\ . \end{aligned}$$

Such systems can be rewritten using matrices which are polynomial in the differential operator $\frac{d}{dt}$ and the delay operator δ. The above system model can be rewritten in the form $Ax = Bu$ with

$$A = \begin{pmatrix} \frac{d}{dt} & -s(t)\delta + s(t)\delta^2 \\ 0 & \frac{d}{dt} \end{pmatrix}, \quad B = \begin{pmatrix} 0 \\ \delta \end{pmatrix}. \tag{1}$$

We basically adopt the ideas of [1] to analyse the properties of such control systems by suitable matrix decompositions, where also fractions in the delay operator are allowed. It has been shown that this method allows to characterize the class of π-flat linear delay systems. An algorithm for the determination of π-flat outputs has been derived using, as a main concept, the hyper-regularity of matrices.

In this paper we present a toolbox for the computer algebra system *Maple*, which implements the algorithm for the determination of π-flat outputs. We discuss here in detail the calculation rules for the used algebra, which are not trivial. A major difference of the implementation is the fact that in [1] hyper-regularity of the polynomial matrices is checked by transformation to Smith-Jacobson form. For the considered algebra, standard algorithms for transformation into Smith-Jacobson form are computationally costly.

In this paper we show that hyper-regularity can be checked by investigation of the row-/column-reduced form of a polynomial matrix. Thus, we can formulate an algorithm for checking π-flatness which is based on row- (resp. column-) reduction. It is rather straight forward to obtain a complexity analysis for checking hyper-regularity using row-reduction. The implementation of the algorithm is rather simple and has therefore been incorporated into the toolbox. This toolbox can be seen as the continuation of our work on the implementation of computer algebra tools for the analysis of linear time varying [12] and nonlinear [13] systems.

The remainder of this paper is organized as follows: in Section II we will introduce the used algebraic framework, which mainly uses Ore polynomials. We will recall the relevant facts about Ore polynomials and the used ground field. Special emphasis is put on the calculation rules, which had to be implemented in the toolbox. Furthermore, we establish the relation between the row-reduced form of the considered polynomial matrices and hyper-regularity. For convenience, we also sketch the algorithm for row-reduction. Then, in Section III it is shown how the considered class of linear time varying systems with delays can be modeled using matrices over Ore polynomials. We recall the concept of differential flatness for systems without delays and the concept of π-flatness for systems with delays and the algorithm for computing flat or π-flat outputs is given. The algorithm also yields the corresponding parameterization of the states and of the input. It is discussed how this parameterization can be used for motion planning. As a next step we discuss, in Section IV, the implementation of the calculation rules for Ore polynomials with the considered ground field in the toolbox. Finally, we apply the toolbox to a well-known example for linear systems with delays in Section V.

II. Algebraic Framework

A. Ore Polynomials

Ore polynomials are a generalization of the usual polynomials with a (not necessarily) commutative multiplication. They have first been studied by Øystein Ore in [14]. As will be shown below they are a good framework for modeling differential as well as delay operators.

Let K be a ring without zero divisors. We consider polynomial expressions in the variable Z with (left-) coefficients

F. Antritter, Computer-Algebra in der Regelungstechnik (EIT 8.3), Universität der Bundeswehr München, 85579 Neubiberg, Germany felix.antritter@unibw.de

J. Middeke was supported by the Austrian Science Foundation (FWF) under the project DIFFOP (P 20 336-N18)

J. Middeke is with Research Institute for Symbolic Computation (RISC), Johannes Kepler University, A-4040 Linz, Austria jmiddeke@risc.uni-linz.ac.at

from K, that is, expressions of the form $f = f_s Z^s + \ldots + f_1 Z + f_0$ where $f_0, \ldots, f_s \in K$. We would like to keep the usual polynomial addition and to retain the *degree rule*, that is, we demand $\deg(fg) = \deg f + \deg g$ for all polynomial expressions f and g. In particular, for $f = Z$ and $g = a \in K$ there must exist elements $\sigma(a)$ and $\vartheta(a)$ such that

$$Za = \sigma(a)Z + \vartheta(a). \tag{2}$$

This equation uniquely defines the maps $\sigma\colon K \to K$ and $\vartheta\colon K \to K$. Assuming now that multiplication is associative and distributive, σ must be an injective endormorphism and ϑ must be a so-called σ*-derivative*, that is, for all $a, b \in K$ the map ϑ fulfills $\vartheta(a+b) = \vartheta(a) + \vartheta(b)$ and the σ*-Leibniz rule* $\vartheta(ab) = \sigma(a)\vartheta(b) + \vartheta(a)b$. (See [14] for a proof.)

Contrarily, for any ring K with (injective) endormorphism σ and σ-derivative ϑ, there exists a ring of polynomial expressions with the multiplication determined by (2). See for example [15, Section 0.10] for a rigid construction. Following the notation of [15] the set of *Ore polynomials* over K with respect to σ and ϑ will be denoted by $K[Z; \sigma, \vartheta]$.

A first example of Ore polynomials are the usual *commutative polynomials*, since for $\sigma = \mathrm{id}$ (the identity function) and $\vartheta = 0$ (the zero-map) the *commutation rule* (2) is just $Za = aZ$ for all $a \in K$, and hence Z is a commutative variable.

Another important example are *differential operators*. Let $\mathscr{K}$ be the field of meromorphic functions and $\frac{d}{dt}$ the usual derivative. For $f \in \mathscr{K}$ we obtain that application of $\frac{d}{dt}f$ to a function g is the same as application of $f \cdot \frac{d}{dt} + \frac{df}{dt}$, since $(\frac{d}{dt}f)g = \frac{d}{dt}(fg) = f\frac{d}{dt}g + \frac{df}{dt}g = (f\frac{d}{dt} + \frac{df}{dt})g$. But this is exactly the commutation rule (2) for $\sigma = \mathrm{id}$ and $\vartheta = \frac{d}{dt}$. Following common custom, we will usually denote $K[Z; \mathrm{id}, \frac{d}{dt}]$ just by $K[\frac{d}{dt}]$.

The last example we give here are the *delay operators* (shift operators). Here, σ is the delay operator δ that is defined as $\delta(f(t)) = f(t-\tau)$ for a fixed $\tau \in \mathbb{R}$. The commutation rule (2) becomes thus $\delta f(t) = f(t-\tau)\delta$. We will simply write $K[\delta] = K[Z; \delta, 0]$ for the ring of delay operators.

Many more examples may be found, for instance, in [16].

In the following we take $Z = \frac{d}{dt}$ or $Z = \delta$. Let $f = f_s Z^s + \ldots + f_1 Z + f_0 \in R$. Multiplying f by Z from the left yields

$$Zf = \sum_{i=0}^{s} Z f_i Z^i = \sum_{i=0}^{s} \sigma(f_i) Z^{i+1} + \sum_{i=0}^{n} \vartheta(f_i) Z^i.$$

If we denote the coefficient-wise application of σ and ϑ to f by $\sigma(f)$ and $\vartheta(f)$, then the above equation may be written more succinctly as $Zf = \sigma(f)Z + \vartheta(f)$.

This formula yields a way to compute products of Ore polynomials. For multiplying f with $g = g_t Z^t + \ldots + g_1 Z + g_0$, we first iteratively compute Zg, $Z^2 g = Z(Zg)$, $\ldots$, $Z^s g = Z(Z^{s-1} g)$ and then we compute the sum $fg = \sum_{i=0}^{s} f_i (Z^i g)$.

We let our operator ring $R = K[Z; \sigma, \vartheta]$ act on a space of functions $\mathscr{F}$: For each $y \in \mathscr{F}$ and $p \in R$ we denote by $p \bullet y \in \mathscr{F}$ the application of p to y.

In the following we assume that $\mathscr{F}$ is a left R-module. That is, we assume that signals may be added and that for all $p, q \in R$ and all $y, z \in \mathscr{F}$ we have:

$$(p+q) \bullet y = p \bullet y + q \bullet y, \quad p \bullet (y+z) = p \bullet y + p \bullet z,$$
$$1 \bullet y = y, \qquad (pq) \bullet y = p \bullet (q \bullet y)$$

(where $1 \in R$ means the identity operator). This application extends naturally to matrices over R and vectors over $\mathscr{F}$. Note that we drop the $\bullet$, when the meaning is unambiguous.

For the description of a linear control system of the form $Ax = Bu$ where $A \in R^{n \times n}$ and $B \in R^{n \times m}$ (see Section III), we can define the *system module* $\mathfrak{M} = R^{1 \times n+m} / R^{1 \times n}(A, -B)$ (see for example [17], [11], [1] and the references therein) as the quotient module of the free R-module $R^{1 \times n+m}$ by the row space of the system matrix $(A, -B)$. This is always a left R-module itself, and one can show that its dual space $\hom_R(\mathfrak{M}, \mathscr{F})$ is isomorphic via the so-called *Malgrange isomorphism* to the behaviour $\{(x,u) \in \mathscr{F}^{1 \times n+m} \mid Ax = Bu\}$ (see [11]). Further examples of signal spaces for various kinds of operators may be found in [18], [11], and [19] and the references therein.

B. Field of Fractions

In order to represent prediction operators, we will allow division by elements in $K[\delta]$. That is, we are considering the total ring of (left-) fractions over $K[\delta]$ which we will denote by $K(\delta)$. A general theory for fractions in non-commutative domains was given by [20]. We use the formulæ from [21], which includes also a section about derivatives of fractions. These are needed to extend the derivation $\frac{d}{dt}$ to $K(\delta)$. This is necessary in order to define the Ore polynomials $K(\delta)[\frac{d}{dt}]$, whose coefficients in $\frac{d}{dt}$ are fractions in δ.

Fractions of elements from $K[\delta]$ can be represented as pairs (b, a) where $a \in K[\delta]$ and $b \in K[\delta] \setminus \{0\}$. The pair (b, a) is thought of representing the *left fraction* $b^{-1}a$. For the computation of sums and products of fractions we will need to compute *least common left multiples (LCLMs)*—denoted by $\mathrm{lclm}(b, d)$—for any two denominators b and d. This can be done using the methods discussed in [22].

In [21], two fractions $b^{-1}a$ and $d^{-1}c$ are called equal if there are $f, g \in K[\delta] \setminus \{0\}$ such that $fb = gd$ and $fa = gc$. Note, that this definition allows simplifying a fraction $b^{-1}a$ by extracting left divisors of a and b, that is, if $a = gs$ and $b = gt$ then $b^{-1}a = t^{-1}s$. Given two fractions $b^{-1}a$ and $d^{-1}c$, the sum is defined as $(rb)^{-1}(ra + sc)$, where $rb = sd = \mathrm{lclm}(b, d)$, and the product is $(rb)^{-1}(sc)$, where $ra = sd = \mathrm{lclm}(a, d)$.

Since δ and $\frac{d}{dt}$ commute by the chain rule, we may extend $\frac{d}{dt}$ from K to $K[\delta]$ setting $\frac{d}{dt}(\delta) = 0$, see for example [23]. By [21, Thm. 13], this extends to a derivation of $K(\delta)$ by $\frac{d}{dt}(b^{-1}a) = (sb)^{-1}(s\frac{da}{dt} - ra)$ where $rb = s\frac{db}{dt}$. Using this, we may finally introduce the ring of *time-varying delay operators* as $K(\delta)[\frac{d}{dt}]$. We discuss the application of such operators for control in Section III.

C. Hyper-Regularity and Row-Reduction

We turn now to matrices of operators. Let $Z = \frac{d}{dt}$ or $Z = \delta$, and let $R = K[Z]$. A matrix $U \in R^{n \times n}$ is called *unimodular* if it has a two-sided inverse in $R^{n \times n}$. We denote the set of all unimodular matrices by $\mathrm{Gl}_n(R)$. For $A \in R^{n \times m}$ we will denote its i^{th} row by $A_{i,*}$.

As outlined in the introduction, we follow [1] by basing our toolbox on the concept of hyper-regularity. In [24],

matrices are called hyper-regular if their Smith-Jacobson form is particularly simple:

Definition 1 ([24, Def. 6]): A matrix $A \in R^{n\times m}$ is *hyper-regular* if there are unimodular matrices $S \in \mathrm{Gl}_n(R)$ and $T \in \mathrm{Gl}_m(R)$ such that

$$SAT = \begin{pmatrix} I_m \\ 0_{n-m,m} \end{pmatrix},$$

if $n \geq m$, or $SAT = (I_n, 0_{n,m-n})$ if $m \geq n$.

If, in the case of $m \geq n$, a matrix $A \in R^{n\times m}$ is hyper-regular, then is possesses a right inverse, since

$$\begin{aligned} SAT = (I_n, 0_{n,m-n}) &\iff AT = (S^{-1}, 0_{n,m-n}) \\ &\iff A\Big(T \begin{pmatrix} S & 0_{n,m-n} \\ 0_{m-n,n} & I_{m-n} \end{pmatrix}\Big) = (I_n, 0_{n,m-n}). \end{aligned}$$

Taking only the first n columns of the right factor, that is, $M = T \begin{pmatrix} S \\ 0_{m-n,n} \end{pmatrix}$ we obtain $AM = I_n$. Contrarily, this implies that column-reducing (see below) A leads to $(I_n, 0_{n,m-n}) = AN$ for $N \in \mathrm{Gl}_m(R)$. Analogously, if $n \geq m$, then hyper-regularity is equivalent to the existence of a left inverse.

For checking the existence of left or right inverses, we use so-called row- or column-reduction. First mentioned in [25] for commutative polynomials, extensions to Ore polynomials may be found in [26]. We will limit our description to the row-reduction. The column-reduction is completely analogous. Abstractly, the goal is to compute low degree generators of the row-space of a matrix using elementary row-operations. Below, we only consider the case $\sigma = \mathrm{id}$. Row-reduction is still possible for other cases, but the formulæ become more complicated. See [26] for details.

We may write any row-vector $v \in R^{1\times n}$ formally as a sum $v = v_s Z^s + \ldots + v_1 Z + v_0$ where $v_0, \ldots, v_s \in K^{1\times n}$ do not contain Z. If $v_s \neq 0$, then we call it the *leading (coefficient) vector* of v and write $v_s = \mathrm{lv}(v)$. In this case, we also say $\deg v = s$. We explicitly define $\mathrm{lv}(0) = 0$ and $\deg 0 = -\infty$.

Let $A \in R^{n\times m}$ and assume that its rows are all non-zero. We will reduce the degrees of the rows of A one by one. The degree of a row can be reduced, if its leading vector is a linear combination of leading vectors of rows of lower degree. Assume for example that we are given a relation $0 = a_1 \mathrm{lv}(A_{1,*}) + \ldots + a_n \mathrm{lv}(A_{n,*})$ where $a_1, \ldots, a_n \in K$. Let j be such that $a_j \neq 0$ and that $\deg A_{j,*} \geq \deg A_{k,*}$ for all k where $a_k \neq 0$. Then the degree of

$$a_j A_{j,*} - \sum_{k\neq j,\, a_k \neq 0} a_k Z^{\deg A_{j,*} - \deg A_{k,*}} A_{k,*}$$

will be strictly smaller than the degree of $A_{j,*}$. Note that the above sum can be realized as a series of elementary row-transformations on A. Iterating this process yields essentially the algorithm that is contained in the proof of [26, Thm. 2.2]. Note, that zero rows that occur during the computations can simply be ignored.

As a tool, we define the *leading (row) coefficient matrix* of A to be that matrix $\mathrm{LC}_{\mathrm{row}}(A) \in K^{n\times m}$ whose i^{th} row is the leading coefficient vector of the i^{th} row of A. That is: $\mathrm{LC}_{\mathrm{row}}(A)_{i,*} = \mathrm{lv}(A_{i,*})$. Row-reduction is only possible if the non-zero rows of $\mathrm{LC}_{\mathrm{row}}(A)$ satisfy a non-trivial relation. That motivates the following definition:

Definition 2: A matrix $A \in R^{n\times m}$ is row-reduced if and only if $\mathrm{LC}_{\mathrm{row}}(A)$ has maximal row-rank.

It is possible to prove, using for example [26, Lem. A.2 (a)], that the rows of a row-reduced matrix are R-linearly independent, that is, they are a basis for the row-space. Also, since row-reducing a basis yields a basis with smaller degrees and since by [26, Lem A.2 (d)] all row-reduced bases have the same degrees, we may conclude that row-reduction indeed yields a basis of lowest possible degree. See [25, Thm. 5] for the analogous result in the commutative case.

The fact that the row space of a left invertible matrix has the canonical basis as minimal degree basis implies now that row-reduction of such a matrix must yield a matrix in $K^{n\times m}$ of degree zero, which must of course still be left invertible and thus of rank n. That means, that we can check hyper-regularity simply by computing a row-reduced form and then checking its rank.

As mentioned in the introduction it is rather straight forward to give a result on the complexity of the used method for checking hyper-regularity, since in every step of the row-reduction, the degree in $\frac{d}{dt}$ is reduced by one. Taking into acount the reduction into row-reduced form and then to the normal form of hyper-regularity we get (based on the results in [26]) that the total amount of necessary operations to check hyper-regulariy of a Matrix $A \in K[\frac{d}{dt}]^{n\times m}$ does not exceed $\mathcal{O}\Big(m^2 n(\deg A)\big(n + (\deg A)^2\big)\Big)$ operations in K. Note that there exist rather few results for determining the complexity of algorithms that transform matrices into Smith-Jacobson form. In future work we will, however, also implement such algorithms and will compare them by means of relevant examples.

III. Application to Linear Control Systems with Delays

With the algebraic background, which has been recalled in Sections II-A, II-B and our results on the properties of row-reduced forms of hyper-regular matrices in Section II-C, we can now analyze a suitable class of control systems. Similar to [1], we consider control systems, which can be modeled in the following form:

$$Ax = Bu, \tag{3}$$

with $A \in K[\frac{d}{dt}]^{n\times n}$, $B \in K[\frac{d}{dt}]^{n\times m}$, $m < n$ and where K is a skew field.[1]

Assumption 1: The rows of $(A, -B)$ are independent (over $K[\frac{d}{dt}]$).

Assumption 2: B is hyper-regular, i.e., there exists a unimodular matrix $\tilde{M} \in \mathrm{Gl}_n(K[\frac{d}{dt}])$ such that

$$\tilde{M}B = \begin{pmatrix} I_m \\ 0_{n-m,m} \end{pmatrix}.$$

Remark 1: Note that both assumptions are natural: Assumption 1 simply assures that there are no superfluous equations in the model. Assumption 2 assures that the inputs

[1] if $n \leq m$ the problem of finding a flat output is useless since x completed by $n-m$ components of u can be chosen as a (π-)flat output. This will become clear in the following.

are really independent. This will become even more clear in the following.

If we take $K=\mathbb{R}$ in (3) this describes the class of linear time invariant systems. For $K=\mathscr{K}$, the field of meromorphic functions in the time t, this describes the class of linear time varying systems. When linear time varying systems with delays are considered, then basically $A\in\mathscr{K}[\delta][\frac{d}{dt}]^{n\times n}$ and $B\in\mathscr{K}[\delta][\frac{d}{dt}]^{n\times m}$. Since we will carry out, in this case, the analysis over $\mathscr{K}(\delta)[\frac{d}{dt}]$, with $\mathscr{K}(\delta)$ the skew field of rational functions in δ with coefficients in $\mathscr{K}$, we will say in the following that, for a unified notation, $A\in K[\frac{d}{dt}]^{n\times n}$ and $B\in K[\frac{d}{dt}]^{n\times m}$ with $K=\mathscr{K}(\delta)$. For examples of the different system classes see, e.g., [6], [1], [27], [28].

If we apply $\tilde{M}$ from Assumption 2 to (3), we obtain the equivalent system model (over $K[\frac{d}{dt}]$)

$$(I_m, 0_{m,n-m})\tilde{M}Ax = u, \quad Fx = 0, \tag{4}$$

where

$$F = (0_{n-m,m}, I_{n-m})\tilde{M}A. \tag{5}$$

Using this reformulated system, we can give an adapted characterization of differential flatness and π-flatness, respectively [1], [8].

Definition 3: System (3) with $K=\mathbb{R}$ or $K=\mathscr{K}$ is differentially flat, if there exist matrices $P\in K[\frac{d}{dt}]^{m\times n}$, $Q\in K[\frac{d}{dt}]^{n\times m}$ and $R\in K[\frac{d}{dt}]^{m\times m}$ such that

$$y = Px, \quad x = Qy, \quad u = Ry. \tag{6}$$

For the case of delay systems we have

Definition 4: System (3) with $K=\mathscr{K}(\delta)$ is π-flat, if there exist matrices $P\in\mathscr{K}(\delta)[\frac{d}{dt}]^{m\times m}$, $Q\in K(\delta)[\frac{d}{dt}]^{n\times m}$ and $R\in\mathscr{K}(\delta)[\frac{d}{dt}]^{m\times m}$ such that

$$y = Px, \quad x = Qy, \quad u = Ry, \tag{7}$$

together with a prediction operator $\pi\in\mathscr{K}[\delta]$ such that $\pi P\in\mathscr{K}[\delta][\frac{d}{dt}]^{m\times m}$, $\pi Q\in\mathscr{K}[\delta][\frac{d}{dt}]^{n\times m}$ and $\pi R\in\mathscr{K}[\delta][\frac{d}{dt}]^{m\times m}$.

It is well known that the differential parameterization (6) has very important applications in control for motion planning and tracking controller design (see e.g. [5], [7], [6], [29]). To mention one aspect: often the flat output is a meaningful quantity and a desired function $y_d(t)$ is supposed to be assigned to that variable. Then, this time function can be planned, e.g., using polynomial interpolation, and inserted into the last equation in (6). This yields the feedforward control signal $u_d(t)=Ry_d(t)$, which, when applied to (3), achieves the desired trajectory y_d for y.

This idea is still valid for the parameterization (7). However, also delays and predictions of y are necessary. This is illustrated by means of the introductory example: We note that, for system (1) it is possible to express x_2 and u using x_1: from the first row in (1) we get $x_2=(s(t)\delta(\delta-1))^{-1}\frac{d}{dt}x_1$ and with the second row we get

$$\begin{aligned} u &= \left((\delta)^{-1}\frac{d}{dt}\right)(s(t)\delta(\delta-1))^{-1}\frac{d}{dt}x_1 \\ &= \left((\delta^2(\delta-1))^{-1}\frac{\dot{s}(t)}{(s(t))^2}\frac{d}{dt} - (\delta^2(\delta-1))^{-1}\frac{1}{(s(t))}\frac{d^2}{dt^2}\right)x_1. \end{aligned}$$

In order to apply the operator in the above formula, we have to compute its Laurent series expansion. In this case we have $(1-\delta)^{-1}=\sum_{i=0}^{\infty}\delta^i$. Thus, given a desired trajectory $x_{1,d}$ for x_1, the necessary feedforward signal is given by

$$u_d(t)=\sum_{i=-2}^{\infty}\left(-\frac{\dot{s}(t-i\tau)}{s^2(t-i\tau)}\dot{x}_{1,d}(t-i\tau) - \frac{1}{s(t-i\tau)}\ddot{s}(t-i\tau)\ddot{x}_{1,d}\right).$$

Although u_d involves an infinite number of delayed terms, only a finite number of terms is non zero at every point of time if $x_{1,d}$ is constant outside of an interval $[t_0,t_1]$. Furthermore, the feedforward signal has to be started only a finite time, namely 2τ, before the transition of x_1 starts at t_0.

A theorem, which is helpful for the computation of flat outputs and π-flat outputs, respectively, is the following [1]:

Theorem 1: The control system (3) with $K=\mathbb{R}$ or $K=\mathscr{K}$ (resp. $K=K(\delta)$) is differentially flat (resp. π-flat), if and only if B and F are hyper-regular over $K[\frac{d}{dt}]$.

Algorithm 1 (Computation of a Parameterization):

Input: Matrices $A\in K[\frac{d}{dt}]^{n\times n}$ and $B\in K[\frac{d}{dt}]^{n\times m}$ with B hyper-regular and $m<n$, representing (3).

Output: If the corresponding system module is free over $K[\frac{d}{dt}]$, a triple (P,Q,R) of matrices $P\in K[\frac{d}{dt}]^{m\times n}$, $Q\in K[\frac{d}{dt}]^{n\times m}$ and $R\in K[\frac{d}{dt}]^{m\times m}$, the defining matrices from Definitions 3/4, together with the prediction operator π.

Else, if the system module corresponding to the system defined by A and B is not free over $K[\frac{d}{dt}]$, then FAIL.

Procedure:

1) Compute (row-reduction) $\tilde{M}\in\mathrm{Gl}_n(K[\frac{d}{dt}])$, s.t. $\tilde{M}B=\begin{pmatrix} I_m \\ 0_{n-m,m}\end{pmatrix}$.
2) Compute $F=(0_{n-m,m},I_{n-m})\tilde{M}A\in K[\frac{d}{dt}]^{n-m\times n}$.
3) If F is hyper-regular with $F\tilde{Q}=(I_{n-m},0_{n-m,m})$ for some $\tilde{Q}\in\mathrm{Gl}_n(K[\frac{d}{dt}])$ (test with column-reduction), then:
   a) Note also $\tilde{Q}^{-1}$, which can be computed in parallel
   b) Set $Q=\tilde{Q}\begin{pmatrix}0_{n-m,m}\\ I_m\end{pmatrix}\in K[\frac{d}{dt}]^{n\times m}$.
   c) Set P to the last m rows of $\tilde{Q}^{-1}$.
   d) Set $R=(I_m,0_{m,n-m})\tilde{M}AQ\in K[\frac{d}{dt}]^{m\times m}$.
   e) Compute π_P, π_Q and π_R such that $\pi_P P$, $\pi_Q Q$ and $\pi_R R$ are polynomial in δ.
   f) Set $\pi=LCLM(\pi_P,\pi_Q,\pi_R)$.
   g) Return (π,P,Q,R).
4) Else, return FAIL.

In [1] it is shown that this algorithm provides a (π-)flat output. We have only adapted the way of checking hyper-regularity. Thus, the proof is also valid for our case.

Remark 2: If $K=\mathbb{R}$ or $K=\mathscr{K}$, then $\pi=1$ in Algorithm 1.

Remark 3: Algorithm 1 can be used to determine flat and π-flat outputs. Since, for the case of π-flatness, the used algebra is much more sophisticated, the implementation with a computer algebra system needs also a lot more of "administrative overhead". We want to emphasize that the implemented toolbox, which is described below, is also capable of computing flat outputs of linear systems without delays. However it is not optimized for that purpose. We have discussed the case without delays in [12].

IV. Implementation of the Toolbox

The provided toolbox is a package for *Maple* and can be obtained at [30]. We give here the chosen data structures for elements of $K(\delta)$ and $K(\delta)[\frac{d}{dt}]$. We restrict to the case where $K=\mathscr{K}$ since all data structures and provided functions can deal with time varying coefficients and thus calculations with $K=\mathbb{R}$ are simply obtained by using constant coefficients.

A. Representation of Elements in $\mathscr{K}(\delta)$

The implemented toolbox uses the *OreTools* package, which is included in *Maple*, in order to define the calculation rules for elements of $\mathscr{K}[\delta]$ by defining a suitable algebra with the SetOreRing command of the *OreTools* package. This definition is included in the toolbox and the command defkd returns the corresponding data structure of that algebra. All functions of the toolbox need this algebra as an argument. An element $a\in\mathscr{K}[\delta]$ is represented by the "not-defined" function[2] OrePoly, whose arguments are the coefficients of $1,\delta,\delta^2,\ldots$. So, e.g., the polynomial $c(t)-\delta^2$ is represented by OrePoly(c(t),0,-1).

Although it is possible to define elements of $\mathscr{K}[\delta]$ using the *OreTools* package, it is not possible to represent fractions. Therefore an extension to this package has been constructed: The elements of $\mathscr{K}(\delta)$ are assumed to be represented by left fractions, i.e., $q\in\mathscr{K}(\delta)$ has the form $q=b^{-1}a$ with $a,b\in\mathscr{K}[\delta]$ (see Section II-B). In order to remain consistent with the *OreTools* package, such q have been represented in the toolbox by the "not-defined" function FRACTION, i.e.

$$q=b^{-1}a: \quad \text{FRACTION(a,b)} \ . \tag{8}$$

The arguments, numerator and denominator of q, can be accessed using the op command of *Maple*. This allows to implement the addition (corresponding function is called addFractions) and multiplication (multiplyFractions) as defined in Section II-B. For the computation of the GCRD or LCLM and the corresponding co-factors, the functionality of the *OreTools* package could be used since these operations have to be applied to the numerator and denominator of q, which both are elements of $\mathscr{K}(\delta)$ (see Section II-B). We implemented also a, rather basic, visualization command (VisualizeOreFrac) which displays $q=b^{-1}a$ in the form $[b,\wedge(-1),*,a]$.

Matrices over $\mathscr{K}(\delta)$ are the usual *Maple* type *Matrix*, whose elements are as shown in (8) and the toolbox provides the corresponding commands MatrixMultFrac and MatrixAddFrac. Additionally, the toolbox comprises the commands gausdel, which performs a Gaussian elimination to a triangular form using only left operations, and redechelonfracM, which transforms a matrix over $\mathscr{K}(\delta)$ into reduced echelon form, again using only left operations. The commands gaussdelrow and redechelonfracMr perform the same operations but using only right operations.

B. Representation of Elements in $\mathscr{K}(\delta)[\frac{d}{dt}]$

The representation of elements of $\mathscr{K}(\delta)[\frac{d}{dt}]$ has been done, again, in a consistent manner with the data structure of the *OreTools* package. We use the "not-defined" function DDT. As an example, the polynomial $c(t)+\delta\frac{d}{dt}+\frac{d^2}{dt^2}$ is represented by DDT(A,B,C), where

A : FRACTION(OrePoly(c(t)),OrePoly(1)) ,
B : FRACTION(OrePoly(0,1),OrePoly(1)) ,
C : FRACTION(OrePoly(1),OrePoly(1)) .

The command VisualizeOreFracddt displays elements of $\mathscr{K}(\delta)[\frac{d}{dt}]$ as polynomials in Dt (which represents $\frac{d}{dt}$) whose coefficients are represented using VisualizeOreFrac (see above). The functions addDDT and mulDDT implement addition and multiplication of elements in $\mathscr{K}(\delta)[\frac{d}{dt}]$. Matrices over $\mathscr{K}(\delta)[\frac{d}{dt}]$ are represented by the *Maple* type Matrix, with the elements being of the above shown data structure. For such matrices the commands MatrixMultFracddt and MatrixAddFracddt are provided by the toolbox. For the analysis of matrices, the toolbox provides the functions rowredkddt and colredkddt, which compute the row- and column-reduced forms of matrices over $\mathscr{K}(\delta)[\frac{d}{dt}]$ according to Section II-C.

Finally, the toolbox provides the function flattest. It performs all steps of Algorithm 1.

V. Examples

The following examples have been computed with the toolbox. The corresponding *Maple* worksheets can be downloaded from [30].

A. Introductory Example Revisited

We consider again the example from the introduction, i.e., $Ax=Bu$ with A and B from (1). We get (using rowredkddt), $\tilde{M}$ such that $\tilde{M}B=\begin{pmatrix}1 & 0\end{pmatrix}^T$ as $\tilde{M}=\begin{pmatrix}0 & \delta^{-1}\\ 1 & 0\end{pmatrix}$ and thus

$$F=\begin{pmatrix}0 & 1\end{pmatrix}\tilde{M}A=\begin{pmatrix}\frac{d}{dt} & -s(t)\delta+s(t)\delta^2\end{pmatrix}.$$

Then, we get (using colredkddt) $\tilde{Q}$ such that $F\tilde{Q}=\begin{pmatrix}1 & 0\end{pmatrix}$ to

$$\tilde{Q}=\begin{pmatrix}\frac{s(t)}{\dot{s}(t)} & \frac{(s(t))^2}{\dot{s}(t)}(\delta-\delta^2)\\ \left(-\delta+\delta^2\right)^{-1}\left(\frac{\ddot{s}(t)}{(\dot{s}(t))^2}-\frac{1}{\dot{s}(t)}\frac{d}{dt}\right) & \tilde{q}_{22}\end{pmatrix},$$

where

$$\begin{aligned}\tilde{q}_{22} &= (1-\delta)^{-1}\left(\frac{-s(t+\tau)\ddot{s}(t+\tau)+2(\dot{s}(t+\tau))^2}{(\dot{s}(t+\tau))^2}(1-\delta)\right)\\ &\quad+(1-\delta)^{-1}\left(\frac{s(t+\tau)}{\dot{s}(t+\tau)}(1-\delta)\left(\frac{d}{dt}\right)\right).\end{aligned}$$

The inverse of $\tilde{Q}$ results to (computed using colredkddt)

$$\tilde{Q}^{-1}=\begin{pmatrix}\frac{d}{dt} & s(t)(\delta^2-\delta)\\ p_{21} & 1\end{pmatrix},$$

where

$$p_{21}=\left(\frac{(s(t))^2}{\dot{s}(t)}(\delta-\delta^2)\right)^{-1}+\left(-\delta+\delta^2\right)^{-1}\frac{1}{s(t)}\frac{d}{dt} \ .$$

Then, P is obtained as the last row of $\tilde{Q}^{-1}$ and Q is obtained as the last column of $\tilde{Q}$. Finally, we get

$$\begin{aligned}R=&\begin{pmatrix}1 & 0\end{pmatrix}\tilde{M}AQ=\\ &\left(-\delta+\delta^2\right)^{-1}\\ &\left(\frac{(\dot{s}(t+\tau))^2\ddot{s}(t+\tau)+\dot{s}(t+\tau)s(t+\tau)s^{(3)}(t+\tau)-2s(t+\tau)(\ddot{s}(t+\tau))^2}{(\dot{s}(t+\tau))^3}(1-\delta)\right)\\ &+\left(-\delta+\delta^2\right)^{-1}\left(\frac{-2s(t+\tau)\ddot{s}(t+\tau)+3(\dot{s}(t+\tau))^2}{(\dot{s}(t+\tau))^2}(\delta-1)\frac{d}{dt}\right)\\ &+\left(-\delta+\delta^2\right)^{-1}\left(\frac{s(t+\tau)}{\dot{s}(t+\tau)}(\delta-1)\frac{d^2}{dt^2}\right)\end{aligned}$$

[2] that means we invoke a function which we do not define, thus, *Maple* keeps the term as it is.

and the prediction operator is $\pi = \delta(\delta - 1)$. The π-flat output is $y = Px = p_{21}x_1 + x_2$. Alternatively, this $\pi-$flat output can be directly obtained using the flattest command.

Note that the parameterization of x_1 with y is given by $x_1 = \frac{(s(t))^2}{(\dot{s}(t))^2}(\delta - \delta^2)y$. Clearly, y is related to x_1 via a unimodular "matrix" (over $\mathscr{K}(\delta)[\frac{d}{dt}]$). This shows again that x_1 is also a π-flat output, as has been shown in Section III.

B. Vibrating String with Interior Mass

We take the model of a vibrating string with two controls, which can be transformed [31] into the time delay system $Ax = Bu$, with $x = (\psi_1, \phi_1, \psi_2, \phi_2)$, $u = (u_1, u_2)$ and

$$A = \begin{pmatrix} 1 & 1 & -1 & -1 \\ \frac{d}{dt} + \eta_1 & \frac{d}{dt} - \eta_1 & \eta_2 & -\eta_2 \\ 1 & \delta^2 & 0 & 0 \\ 0 & 0 & \delta^4 & 1 \end{pmatrix}, \; B = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ \delta & 0 \\ 0 & \delta^2 \end{pmatrix}.$$

Note that, since the toolbox can up to now only deal with a single delay operator δ the original operators δ_1 and δ_2 have been replaced by $\delta_1 = \delta$ and $\delta_2 = \delta^2$.

The flatness algorithm (using the flattest command) directly yields $P = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$, i.e., a π-flat output is given by $y = (\psi_2, \phi_2)$. The parameterization of x and u (represented by Q and R) results to

$$Q = \begin{pmatrix} \left(\frac{\eta_1 - \eta_2}{2\eta_1}\right) - \frac{1}{2\eta_1}\frac{d}{dt} & \left(\frac{\eta_1 + \eta_2}{2\eta_1}\right) - \frac{1}{\eta_1}\frac{d}{dt} \\ \left(\frac{\eta_1 + \eta_2}{2\eta_1}\right) + \frac{1}{2\eta_1}\frac{d}{dt} & \left(\frac{\eta_1 - \eta_2}{2\eta_1}\right) + \frac{1}{2\eta_1}\frac{d}{dt} \end{pmatrix}$$

and

$$R = \begin{pmatrix} R_{1,1} & R_{1,2} \\ \delta^2 & \left(\delta^2\right)^{-1}(1) \end{pmatrix},$$

with

$$\begin{aligned} R_{1,1} &= (\delta)^{-1}\left(\frac{\eta_1 - \eta_2}{2\eta_1} + \frac{(\eta_1 + \eta_2)\delta^2}{2\eta_1}\right) \\ &\quad + (\delta)^{-1}\left(-\frac{1}{2\eta_1} + \frac{\delta^2}{2\eta_1}\right)\frac{d}{dt}, \\ R_{1,2} &= (\delta)^{-1}\left(\frac{\eta_1 + \eta_2}{2\eta_1} + \frac{(\eta_1 - \eta_2)\delta^2}{2\eta_1}\right) \\ &\quad + (\delta)^{-1}\left(-\frac{1}{2\eta_1} + \frac{\delta^2}{2\eta_1}\right)\frac{d}{dt}. \end{aligned}$$

The corresponding prediction operator is $\pi = \delta^2$.

VI. Conclusions

A toolbox for the computer algebra system *Maple* has been presented which performs the computation of differentially flat and π-flat outputs of linear (time varying) systems with and without delays according to the approach in [1]. We discussed all necessary calculation rules for the used algebras since they are rather involved and presented suitable data structures, which allowed us to implement these calculation rules. The application of the toolbox has been illustrated by means of examples.

References

[1] Vincent Morio, Franck Cazaurang, and Jean Lévine, "On the computation of π-flat outputs for linear time-delay systems," *http://www.arxiv.org*, vol. arxiv:math.OC/0910.3619v2, 2010.

[2] Philippe Martin, *Contribution à l'Étude des Systèmes Différentiellement Plats*, Ph.D. thesis, école des Mines de Paris, 1992.

[3] Michel Fliess, Jean Lévine, Philippe Martin, and Pierre Rouchon, "Flatness and defect of nonlinear systems: introductory theory and examples," *Int. J. Control*, vol. 61, no. 6, pp. 1327–1361, 1995.

[4] Michel Fliess, Jean Lévine, Philippe Martin, and Pierre Rouchon, "A Lie-Bäcklund approach to equivalence and flatness of nonlinear systems," *IEEE Trans. Automat. Control*, vol. 44, no. 5, pp. 922–937, 1999.

[5] Philippe Martin, Richard M. Murray, and Pierre Rouchon, "Flat systems," in *Plenary Lectures and Minicourses, Proc. ECC 97, Brussels*, G. Bastin and M. Gevers, Eds., 1997, pp. 211–264.

[6] Hebertt Sira-Ramirez and Sunil K. Agrawal, *Differentially Flat Systems*, Marcel Dekker, New York, 2004.

[7] Jean Lévine, *Analysis and Control of Nonlinear Systems: A Flatness-based Approach*, Mathematical Engineering Series. Springer, 2009.

[8] Hugues Mounier, *Propriétés structurelles des systemes linéaires a retard: aspects théoriques et pratiques*, Ph.D. thesis, University of Paris XI, Paris, France, 1995.

[9] Nicolas Petit, *Systèmes à retards. Platitude en génie des proc édés et contrôle de certaines équations des ondes*, Ph.D. thesis, Ecole des Mines de Paris, Paris, France, 2000.

[10] P. Rocha and Jan C. Willems, "Behavioural controllability of delay-differential systems," *SIAM J. Control Optimiz.*, pp. 254–264, 1997.

[11] Frédéric Chyzak, Alban Quadrat, and Daniel Robertz, "Effective algorithms for parametrizing linear control systems over Ore algebras," *Appl. Algebra Eng., Commun. Comput.*, vol. 16, no. 5, pp. 319–376, 2005.

[12] Felix Antritter and Johannes Middeke, "An effcient algorithm for checking hyper-regularity of matrices (issac poster abstract)," *ACM CCA*, vol. 44(3), 2010.

[13] Felix Antritter and Gregor G. Verhoeven, "On symbolic computation of flat outputs for differentially flat systems," in *Proc. NOLCOS 2010*, 2010.

[14] Øystein Ore, "Theory of non-commutative polynomials," *Annals of Mathematics*, vol. 34, pp. 480 – 508, 1933.

[15] Paul Moritz Cohn, *Free Rings and Their Relations*, Academic Press, London, 1985.

[16] Frédéric Chyzak and Bruno Salvy, "Non-commutative elimination in Ore algebras proves multivariate identities," *Journal of Symbolic Computation*, vol. 26, no. 2, pp. 187–227, 1998.

[17] M. Fliess, "Some basic structural properties of generalized linear systems," *Systems & Control Letters*, vol. 15, pp. 391–396, 1990.

[18] Achim Ilchmann, "Algebraic theory of time-varying linear systems: a survey," in *Proceedings of the 16th IFAC World Congress*, Pavel Piztek, Ed., 2005, pp. 312–318.

[19] Eva Zerz, "Behavioral systems theory: A survey," *Int. J. Appl. Math. Comput. Sci.*, vol. 18, no. 3, pp. 265–270, 2008.

[20] Øystein Ore, "Linear equations in non-commutative fields," *The Annals of Mathematics*, vol. 32, no. 3, pp. 463–477, July 1931.

[21] Jan Ježek, "Non-commutative rings of fractions in algebraical approach to control theory," *Kybernetika*, vol. 32, no. 1, pp. 81–94, 1996.

[22] Manuel Bronstein and Marko Petkovšek, "An introduction to pseudo-linear algebra," *Theoretical Computer Science*, vol. 157, pp. 3–33, 1996.

[23] Michael G. Voskoglou, "Extending derivations and endomorphisms to skew polynomial rings," *Publications de l'Institut Mathématique (Beograd), Nouvelle série*, vol. 39, no. 53, pp. 79–82, 1986.

[24] J. Lévine, "On necessary and sufficient conditions for differential flatness," in *Proc. of IFAC NOLCOS 2004 Conference*, Stuttgart, 2004.

[25] G. David Forney jr., "Minimal bases of rational vector spaces with applications to multivariable linear systems," *SIAM J. Control*, vol. 13, pp. 493–520, 1975.

[26] Bernhard Beckermann, Howard Cheng, and George Labahn, "Fraction-free row reduction of matrices of ore polynomials," *Journal of Symbolic Computation*, vol. 41, pp. 513–543, 2006.

[27] Keqin Gu, Vladimir L. Kharitonov, and Jie Chen, *Stability of Time-Delay Systems*, Birkhäuser, Boston, 2003.

[28] Hugues Mounier, Pierre Rouchon, and Joachim Rudolph, "Some examples of linear systems with delays," *J. Europ. Syst. Autom.*, vol. 31, pp. 911–925, 1997.

[29] Felix Antritter, Bernd Müller, and Joachim Deutscher, "Tracking control for nonlinear flat systems by linear dynamic output feedback," *Proceedings NOLCOS 2004, Stuttgart*, 2004.

[30] www.unibw.de/eit8_1/forschung-en/index_html?set_language=en.

[31] Hugues Mounier, Joachim Rudolph, Michel Fliess, and Pierre Rouchon, "Tracking control of a vibrating string with an interior mass viewed as a delay system," *ESAIM COCV*, vol. 3, pp. 315–321, 1998.