WEBMATHEMATICA BASED TOOLS FOR NONLINEAR CONTROL SYSTEMS

Heli Rennik, Maris Tõnso and Ülle Kotta
Institute of Cybernetics, Tallinn University of Technology, Akadeemia tee 21, Tallinn, 12818, Estonia

Keywords: Web-based education, webmathematica, nonlinear control, linear algebraic framework.

Abstract: Algebraic approach of differential one-forms provides simple theoretical framework for several typical problems of nonlinear control theory that makes it useful for educational purposes. Additional assistance is provided by Mathematica functions, developed by us and made available by creating a web-based application using web-Mathematica. These symbolic computation tools provide solutions for several modelling and anlysis problems like accessibility, identifiability, realizability and realization and require no other software except for an internet browser.

1 INTRODUCTION

Over the many years we have developed a software package NLControl in Mathematica for nonlinear control systems (Kotta and Tõnso, 1999; Kotta and Tõnso, 2003), based on algebraic approach of differential one-forms (Aranda-Bricaire et al., 1996; Kotta et al., 2001; Conte et al., 1999). This package provides basic tools for modelling, analysis and synthesis both of discrete- and continuous-time systems.

The Mathematica functions developed by us and described partly in (Kotta and Tõnso, 1999; Kotta and Tõnso, 2003) cannot be used outside of Mathematica environment. Our task was to make our tools available via the world-wide-web, in such a way that no other software except for an internet browser needs to be installed in a computer to use these tools. Recently, we developed a first set of web-based tools. The reason for this was twofold. First, the computer has become an integral part of the educational process (Moog et al., 2003a). Second, we want to make the tools, developed by us, available to a larger control community. The other web-based tool for nonlinear control system is described in (Ondera and Huba, 2005). Since the intention was to create a web-based application that would re-use the original program code and whose outputs would mimic the original tools as much as possible, we used webMathematica.

The paper is organized as follows. Section 2 describes webMathematica technology both the frontend, which is available to everyone over the web, and web server technology, which includes description of kernels, kernel pools and configuration options. Section 3 provides a description of nonlinear control systems, both discrete- and continuous-time, together with the necessary assumptions and our basic tool to handle the considered problems - the sequence of subspace H_k of one forms, associated to the nonlinear control system. Next, a brief description of modeling and analysis problems, implemented in our webMathematica website is given together with some simple examples and program codes. Finally, the last section concludes the paper by discussing our contribution and providing some future perspectives.

2 WEB-BASED IMPLEMENTATION

This section describes a webMathematica website for nonlinear control systems, both discrete- and continuous-time. We have implemented several Mathematica functions programmed by us to web-Mathematica for public use. WebMathematica is an option to make our functions available for students

Rennik H., Tõnso M. and Kotta Ü. (2007).
WEBMATHEMATICA BASED TOOLS FOR NONLINEAR CONTROL SYSTEMS.
In *Proceedings of the Fourth International Conference on Informatics in Control, Automation and Robotics*, pages 178-183
DOI: 10.5220/0001624501780183
Copyright © SciTePress

and science community without revealing our programming code. Our webMathematica website is simple, including at moment five different function pages for discrete- and continuous-time systems described either by input-output equations or by state space equations.

2.1 About Webmathematica Frontend

WebMathematica frontend is a web based user interface for Mathematica. WebMathematica adds interactive calculations and visualizations to a website by integrating Mathematica with web server technology. WebMathematica makes all Mathematica functionality available over the web and is easy to use even for non-professional programmer. For using webMathematica one has to know HTML basics and use a web browser.

Websites (or in other words user interfaces) can use standard web graphical user interface elements, such as text fields, check boxes, and drop-down lists. The major browsers, like Internet Explorer, Mozilla and Netscape Navigator support webMathematica. WebMathematica allows a site to deliver HTML pages that are enhanced by the addition of Mathematica commands and uses the request/response standard followed by web servers. The process how webMahtematica works is following:

1. Browser sends a request to webMathematica server.
2. WebMathematica server acquires Mathematica kernel from the pool.
3. Mathematica kernel is initialized with input parameters, it carries out calculations, and returns result to server.
4. WebMathematica server returns Mathematica kernel to the pool.
5. WebMathematica server returns result to Browser.

Requests are sent to the server with webMathematica web pages that are based on two standard Java technologies: Java Servlet and JavaServer Pages (JSP) technologies. JavaServer Pages (JSPs) use a special library of tags that work with Mathematica. These tags have the form $\langle msp : tag \rangle$. The $\langle msp : allocate \rangle$ tag causes a Mathematica kernel to be allocated to use for computations. The contents of the $\langle msp : evaluate \rangle$ tags are sent to Mathematica for computation with the result inserted into the final page. The $\langle /msp : allocate \rangle$ tag frees the Mathematica kernel to be used for another computation.The library of tags is called the MSP Taglib. In our site we use also JavaScript and Java. We use JavaScript for opening and closing windows and communicating between windows and Java for calculating random inputs.

2.2 Webmathematica Web Server Technology

There are many different combinations of hardware and operating systems that support webMathematica components, for example Windows, Linux, Solaris, Mac OS X. Before one starts to install webMathematica, one has to install Java and a servlet container. WebMathematica is tested with Apache Tomcat and JRun. We are using Linux operating system and Tomcat as a web container.

WebMathematica server can be configured as necessary. The number of pools of Mathematica kernels can be specified. Also one can set the number of kernels that are launched when the system starts. The parameters, as the number of times that each kernel can be taken from the kernel pool before being shut down, can be specified in order to clean it up from unnecessary processes. It is a good idea to shut down each kernel at a regular time-period. Another parameter specifies the maximum number of milliseconds for processing a page. When this time is exceeded, the kernel is shut down and restarted. This is useful in case one is computing large operations.

We are using only one kernel and one kernel pool and we have got problems with functions that do not work properly. They intend to let Tomcat ending up crashing. We have specified the time 30 seconds in which all responses must be got from the server. When the web site is made available for the students, we will increase the number of kernels.

3 IMPLEMENTED FUNCTIONS

Several different functions programmed by us are gathered into one Mathematica package called NLControl for solving different control problems. At moment we have implemented five functions from this package into webMathematica website. These functions are `Submersivity`, `SequenceHk`, `Realization`, `Accessibility` and `Identifiability`. We have chosen functions that are based on subspaces H_k and cover different modeling and analysis problems.

Consider the continuous-time

$$\dot{x} = f(x,u) \tag{1}$$

or the discrete-time nonlinear control system

$$x^+ = f(x,u), \tag{2}$$

respectively, where $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, f is an analytic function and x^+ denotes the forward shift of x. In order to make computations with these equations in Mathematica, we have created two special objects for NLControl package: `CStateSpace[f,x,u,t]` and `DStateSpace[f,x,u,t]`, that represent systems (1) and (2), respectively. Objects `CIO[f,u,y,t]` and `DIO[f,u,y,t]` are used for representing continuous- and discrete-time input/output equations, respectively.`DStateSpace`, `CStateSpace`, `DIO` and `CIO` are used by all the implemented functions `Submersivity`, `SequenceHk`, `Realization`, `Accessibility` and `Indentifiability`.

3.1 Submersivity

The function f in (2) is said to be a submersion, if generically (i.e. everywhere, except on a set of measure zero)

$$\operatorname{rank}\frac{\partial f}{\partial(x,u)} = n. \tag{3}$$

Submersivity assumption has to be satisfied for discrete-time control systems for the other functions to be applicable. `Submersivity` gives `True` if discrete-time state equations are submersive. If the result is `False` then the other Mathematica functions cannot be used since they may yield wrong results.

For checking `Submersivity` assumption the system has to be given in the state space form with the list of state and input variables.

The Mathematica block sent to the server looks as follows:

```
<msp:evaluate>
   MSPBlock[ {$$f, $$Xt, $$Ut},
      MSPFormat[ Submersivity[
         DStateSpace[$$f, $$Xt, $$Ut, t]],
         OutputForm
      ]
   ]
</msp:evaluate>
```

Example: Consider the bioreactor model, where cells are being grown through the consumption of a substrate (Kazantzis and Kravaris, 2001):

$$\begin{aligned} x_1(t+1) &= x_1(t) + \left[\tfrac{x_1(t)x_2(t)}{x_1(t)+x_2(t)} - 0.08x_1(t)\right]T \\ x_2(t+1) &= x_2(t) + \left[\tfrac{-x_1(t)x_2(t)}{x_1(t)+x_2(t)} - 0.08x_2(t) \right.\\ &\quad \left. +0.008\right]T \end{aligned} \tag{4}$$

The result is `True`.

3.2 SequenceHk

Let $\mathcal{K}$ be the field of meromorphic functions in a finite number of system variables $\{x, u^{(k)}, k \geq 0\}$ or $\{x(0), u(k), k \geq 0\}$ for the continuous- and discrete-time system, respectively. Over the field $\mathcal{K}$ one can define a vector space $\mathcal{E} := \operatorname{span}_{\mathcal{K}}\{d\varphi \mid \varphi \in \mathcal{K}\}$. The elements of $\mathcal{E}$ are called one-forms. The relative degree r of one-form ω in $\mathcal{X} = \operatorname{span}_{\mathcal{K}}\{dx\}$ is given by $r = \min\{k \in \mathbb{N} \mid \operatorname{span}_{\mathcal{K}}\{\omega, \ldots, \omega^{(k)}\} \not\subset \mathcal{X}\}$ or $r = \min\{k \in \mathbb{N} \mid \operatorname{span}_{\mathcal{K}}\{\omega(0), \ldots, \omega(k)\} \not\subset \mathcal{X}\}$.

Let us define a decreasing sequence of subspaces $\mathcal{H}_0 \supset \mathcal{H}_1 \supset \mathcal{H}_2 \supset \ldots$ such that each $\mathcal{H}_k$, for $k > 0$, is the set of all one-forms with relative degree at least k(Aranda-Bricaire et al., 1996; Conte et al., 1999):

$$\begin{aligned} \mathcal{H}_0 &= \operatorname{span}_{\mathcal{K}}\{dx, du\} \\ \mathcal{H}_k &= \{\omega \in \mathcal{H}_{k-1} \mid \dot{\omega} \in \mathcal{H}_{k-1}\} \end{aligned}$$

or

$$\mathcal{H}_k = \{\omega \in \mathcal{H}_{k-1} \mid \omega^+ \in \mathcal{H}_{k-1}\}.$$

There exists an integer $k^* > 0$ such that $\mathcal{H}_k \supset \mathcal{H}_{k+1}$, for $k \leq k^*$, and $\mathcal{H}_{k^*+1} = \mathcal{H}_{k^*+2} = \ldots \mathcal{H}_\infty$, $\mathcal{H}_{k^*} \not\supset \mathcal{H}_\infty$. The existence of k^* comes from the fact that each $\mathcal{H}_k$ is a finite dimensional vector space, so that at each step either the dimension decreases by at least one or $\mathcal{H}_{k+1} = \mathcal{H}_k$.

$\mathcal{H}_\infty$ contains the one-forms with infinite relative degree so that these one-forms will never be influenced by the control.

An one-form $\omega \in \mathcal{E}$ is exact if $d\omega = 0$, and closed if $d\omega \bigwedge \omega = 0$ where by $\bigwedge$ is denoted the wedge product. We say that the subspace is completely integrable if it admits the basis which consists only of closed one-forms.

The function `SequenceHk` computes first N elements in the sequence of subspaces Hk associated to state equations, where N is an positive integer specified by us. For running `SequenceHk`, the system has to be given in the state space form with the list of state and input variables.

The Mathematica block sent to the server looks as follows:

```
<msp:evaluate>
   MSPBlock[ {$$f, $$Xt, $$Ut},
      MSPFormat[ SequenceHk[
         DStateSpace[$$f, $$Xt, $$Ut, t], All],
         OutputForm, TimeArgument-> True
      ]
   ]
</msp:evaluate>
```

Example 1: Consider another bioreactor model, where the microorganisms grow by consuming the substrate (Benamor et al., 1997):

$$\begin{aligned} x_1(t+1) &= x_1(t) + T\tfrac{a_1x_1(t)x_2(t)}{a_2x_1(t)+x_2(t)} - Tu(t)x_1(t) \\ x_2(t+1) &= x_2(t) - T\tfrac{a_3a_1x_1(t)x_2(t)}{a_2x_1(t)+x_2(t)} - Tu(t)x_2(t) \\ &\quad +Ta_4u(t) \\ y(t) &= x_1(t) \end{aligned} \tag{5}$$

The result is the following:

$$\begin{aligned}\mathcal{H}_1 &= Span[dx_1(t), dx_2(t)]\\ \mathcal{H}_2 &= Span[dx_1(t)]\\ \mathcal{H}_3 &= \{0\}\end{aligned} \tag{6}$$

Example 2: The dynamic model of a current-driven induction motor expressing the rotor flux and the stator currents in a reference frame rotating at synchronous speed is given by the continuous-time state equations(Bazanella and Reginatto, 2000):

$$\begin{aligned}\dot{x}_1(t) &= -c_1x_2(t) - u_1(t)x_2(t) + c_2u_3(t)\\ \dot{x}_2(t) &= -c_1x_2(t) + u_1(t)x_1(t) + c_2u_2(t)\\ \dot{x}_3(t) &= -c_3x_3(t) + c_4(c_5(x_2(t)u_3(t) - x_1(t)u_2(t))\\ &\quad -Tm)\end{aligned} \tag{7}$$

The result is the following:

$$\begin{aligned}\mathcal{H}_1 &= Span[dx_1(t), dx_2(t), dx_3(t)]\\ \mathcal{H}_2 &= \{0\}\end{aligned} \tag{8}$$

3.3 Realization

Consider a higher order i/o differential

$$y^{(n)} = \Phi(y, \dots, y^{(n-1)}, u, \dots, u^{(s)}) \tag{9}$$

or difference equation

$$\begin{aligned}y(t+n) &= \Phi(y(t), \dots, y(t+n-1),\\ &\quad u(t), \dots, u(t+s)),\end{aligned} \tag{10}$$

where n and s are nonnegative integers, $s < n$ and Φ is a real analytic function. The realization problem is to construct the state equations of order n,

$$\begin{aligned}\dot{x} &= f(x,u) \qquad & x^{+} &= f(x,u)\\ & & \text{or} & \\ y &= h(x) & y &= h(x)\end{aligned}$$

for the i/o equation (9) and (10), respectively. It is important to stress that the state-space realization does not exist for every i/o equation. In order to find state equations, one has to compute the sequence $\mathcal{H}_k$ for the i/o equations (9) or (10). The detailed procedures can be found in (Kotta et al., 2001) and (Moog et al., 2003b), respectively.

Theorem 1. The i/o equation has a state-space realization iff for $1 \le k \le s+2$ the subspaces $\mathcal{H}_k$ are completely integrable. The state coordinates can be found by integrating the basis functions of $\mathcal{H}_{s+2}$.

Function `Realization` determines whether the nonlinear higher order input-output difference equation can be realized in the classical state-space form and the if the i/o equation is realizable, finds the state equations. For running `Realization` the system has to be given by the input-output equations and the list of input and output variables.

The Mathematica block sent to the server looks as follows:

```
<msp:evaluate>
   MSPBlock[ {$$eqs, $$Ut, $$Yt},
      MSPFormat[ Realization[
         DIO[$$eqs, $$Ut, $$Yt, t], [#][t]&],
         OutputForm, TimeArgument-> True
      ]
   ]
</msp:evaluate>
```

Example 1: The fed-batch bakers' yeast fermentation process 1 is described by the following i/o equations (Keulers et al., 1993).

$$\begin{aligned}y(t+1) &= 0.9106y(t) + 2.072u(t+1) - 1.903u(t)\\ &+120.7y(t)u(t+1) - 107.3y(t)u(t) + 299.6u(t+1)^2\\ &-232.8u(t+1)u(t) - 84.17u(t)^2\end{aligned} \tag{11}$$

The result is that the classical state space form does not exist for system (11).

Example 2: Consider the continuous-time i/o equation:

$$\dddot{y}(t) = \ddot{y}(t)u(t) + \dot{u}(t)y(t) + u(t) \tag{12}$$

The classical state equations for (12) are:

$$\begin{aligned}\dot{x}_1(t) &= x_2(t)\\ \dot{x}_2(t) &= u(t)x_1(t) + x_3(t)\\ \dot{x}_3(t) &= u(t)(1 + u(t)x_1(t) - x_2(t) + x_3(t))\\ y(t) &= x_1(t)\end{aligned} \tag{13}$$

3.4 Accessibility

A function φ_r in $\mathcal{K}$ is said to be an autonomous variable for system (1) or (2), if there exist an integer $\mu \ge 1$ and a non-zero meromorphic function F so that $F(\varphi_r, s\varphi_r, \dots, s^{\mu}\varphi_r) = 0$ or $F(\varphi_r, \delta\varphi_r, \dots, \delta^{\mu}\varphi_r) = 0$ for continuous- and discrete-time system, respectively. The system (1) or (2) is said to be accessible if there does not exist any non-zero autonomous variable in $\mathcal{K}$.

Theorem 2. (Aranda-Bricaire et al., 1996; Conte et al., 1999) The nonlinear system is strongly accessible iff

$$\mathcal{H}_\infty = \{0\}. \tag{14}$$

Note that the *Theorem 1* and 2 holds both in continuous- and discrete-time cases though the rules to calculate the subspaces $\mathcal{H}_k$ are different.

`Accessibility` returns `True` if the nonlinear system is strongly accessible and `False` otherwise.

For computing `Accessibility` the system has to be given in the state space form with the list of state and input variables.

The Mathematica block sent to the server looks as follows:

```
<msp:evaluate>
   MSPBlock[ {$$f, $$Xt, $$Ut},
      MSPFormat[ Accessibility[
         DStateSpace [$$f, $$Xt, $$Ut, t]],
         OutputForm
      ]
   ]
</msp:evaluate>
```

Example 1: Consider a grain drying process (Kotta and Nurges, 1985).

$$\begin{array}{rcl} x_1(t+1) &=& -0.0081u(t)x_1(t)+x_2(t) \\ x_2(t+1) &=& -0.01772892u(t)x_1(t) \\ &+& 1.6332x_2(t)+x_3(t) \\ x_3(t+1) &=& (-0.1751-0.00360073u(t)) \\ && x_1(t)-0.4567x_2(t) \end{array} \tag{15}$$

The result is `True`, that is system (15) is accessible.

Example 2: Consider a simple academic example:

$$\begin{array}{rcl} x_1(t) &=& x_1(t)(x_3^2(t)+1)^2 \\ x_2(t) &=& x_2(t)(x_3^2(t)+1)^3 \\ x_3(t) &=& x_3(t)+u(t) \end{array} \tag{16}$$

The result is `False`. $\mathcal{H}_\infty = span\{3x_2dx_1 - 2x_1dx_2\}$.

3.5 Identifiability

Identifiability property characterizes the possibility to find the unknown parameters of the system from the measured input-output data. Consider the continuous-time nonlinear system

$$\begin{array}{rcl} \dot{x} &=& f(x,u,\theta) \\ y &=& h(x,u,\theta), \end{array} \tag{17}$$

where $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$ and the parameter vector $\theta \in \mathbb{R}^q$. There are two concepts for identifiability - with and without the knowledge of initial conditions. At the moment only identifiability without knowledge of the initial state is implemented on our website. Our website offers two alternative methods to check identifiability of the system. The first is algebraic method, that enables to find the parameters by solving certain algebraic equations depending only on input-output information, see (Xia and Moog, 2003). The second method checks if the parameters can be found by the least squares method (LSM). To check identifiability by the LSM one has to find the i/o representation $E(y,\dot{y},\ldots,y^{(n)},u,\dot{u},\ldots,u^{(s)},\theta)=0$ for system (17). If the above i/o equation admits a separable form $\sum_{i=1}^{n_i} g_i(\theta)E_i(y,\dot{y},\ldots,y^{(n)},u,\dot{u},\ldots,u^{(s)})$ and the condition $g(\theta)=g(\theta^*) \Leftrightarrow \theta=\theta^*$ is satisfied, the least squares method described in (Pearson, 1989), is applicable for parameter identification.

For computing `Identifiability` the system has to be given in the state space form with the list of state, input and output variables.

The Mathematica block sent to the server looks as follows:

```
<msp:evaluate>
   method = Switch[ MSPToExpression[ $$method],
       algebraic, Algebraic, pearson, Pearson];
   MSPBlock[
      {$$f, $$Xt, $$Ut, $$h, $$Yt, $$theta},
      MSPFormat[ Identifiability[
         CStateSpace[f, Xt, Ut, t, h, Yt],
            {theta},Method->method],
         OutputForm
      ]
   ]
</msp:evaluate>
```

Example: Consider the simple academic example

$$\begin{array}{rcl} \dot{x}_1(t) &=& \theta_1 x_1(t)^2+\theta_2 x_1(t)x_2(t)+u(t) \\ \dot{x}_2(t) &=& \theta_3 x_1(t)^2+\theta_4 x_1(t)x_2(t) \\ y(t) &=& x_1 \end{array} \tag{18}$$

The system is identifiable neither with LSM nor with algebraic method.

3.6 Comparative Evaluation

In this section our web-based tools are compared by web-based tools for nonlinear control systems in (Ondera and Huba, 2005). Besides the fact the problems handled are different ((Ondera and Huba, 2005) is dedicated solely to the exact static state feedback linearization problem) there are other points to be mentioned.

Our web-site enables user to specify the system equations and calculate the state transformation and linearizing control law (work in progress, not described in this paper). The web-tools in (Ondera and Huba, 2005) allow the user additionally to submit a desired closed-loop poles of a pole-placement controller and to perform a simulation.

There is also a difference in chosen technology. The web-tools in (Ondera and Huba, 2005) are based on MATLAB and its Symbolic Math Toolbox. This toolbox is a Maple 8 symbolic kernel that was bought from Maplesoft and implemented into MATLAB by The MathWorks. The different platform also implies a different internet implementation. In (Ondera and Huba, 2005) tools are web-accessible via MATLAB Web Server that is based on CGI technology, whereas webMathematica is java and javascript-based.

Both web-tools relieve users from installing Mathematica or MATLAB on their computers and help to make programs available to everyone without seeing program code.

4 CONCLUSION

WebMathematica is a web version of Mathematica that uses a web server technology, HTML and Java Servlet Pages. Calculations entered via web pages are sent to kernel, where the result is calculated and sent back to web pages. Several different functions programmed by us are gathered into one Mathematica package called NLControl for solving different modeling, analysis and synthesis problems. At moment we have implemented five functions from this package into webMathematica website. These functions are `Submersivity`, `SequenceHk`, `Realization`, `Accessibility` and `Identifiability`. In the future we are expanding our website with functions `Linearization` and `PrimeForm`. The function `Linearization` checks if the state equations can be linearized via the static state feedback and coordinate transformation, and finds necessary transformations. The function `PrimeForm` transforms the system into the prime form, whenever possible, using the static state feedback, and the coordinate transformations in the state and output spaces. Besides continuous- and discrete-time systems we are also programming functions for systems, described on homogeneous time scales (Bartosiewicz et al., 2007).

ACKNOWLEDGEMENTS

This work was partially supported by the Estonia Science Foundation Grant No 6922.
The authors thank M. Ondera for discussions on comparative evaluation of web-based symbolic tools presented in this paper and in (Ondera and Huba, 2005).

REFERENCES

Aranda-Bricaire, E., Kotta, Ü., and Moog, C. (1996). Linearization of discrete-time systems. In *SIAM J. Control and Optimization*, volume 34, pages 1999–2023.

Bartosiewicz, Z., Ü. Kotta, Pawluszewicz, E., and Wyrwas, M. (2007). Irreducibility conditions for nonlinear input-output equations on homogeneous time scales. Submitted to IFAC Symp. on Nonlinear Control Systems.

Bazanella, A. and Reginatto, R. (2000). Robustness margins for indirect field oriented control of induction motors. In *IEEE Trans. Automatic Control*, volume 45, pages 1226–1231.

Benamor, S., Hammouri, H., and Couenne, F. (1997). A luenberger-like observer for discrete-time nonlinear systems. In *Proc. European Control Conf.*, Brussels, Belgium.

Conte, G., Moog, C., and Perdon, A. (1999). Nonlinear control systems. In *Lecture Notes in Control and Information Sciences*, volume N242, London. Springer.

Kazantzis, N. and Kravaris, C. (2001). Discrete-time nonlinear observer design using functional equations. In *Systems and Control Letters*, volume 42, pages 81–94.

Keulers, M., Sepp, K., Breur, A., and Reyman, G. (1993). A simulation study of nonlinear structure identification of a fed batch bakers' yeast process. In *Proc. American Control Conference*, pages 2256–2260, San Fransisco.

Kotta, Ü. and Nurges, Ü. (1985). Identification of input-output bilinear systems. In *Proc. 9th IFAC World Congress*, volume 2, pages 723–727. Pergamon Press.

Kotta, Ü. and Tõnso, M. (1999). Transfer equivalence and realization of nonlinear higher order input/output difference equations using mathematica. In *Journal of Circuits, Systems and Computers*, volume 9, pages 23–25.

Kotta, Ü. and Tõnso, M. (2003). Linear algebraic tools for discrete-time nonlinear control systems with mathematica. In *(Lecture Notes in Control and Information Sciences; 281)*, pages 195–205, Nonlinear and Adaptive Control, NCN4 2001 / Eds. A.Zinober, D.Owens. Berlin [etc.]:. Springer.

Kotta, Ü., Zinober, A., and Liu, P. (2001). Transfer equivalence and realization of nonlinear higher order input-output difference equations. In *Automatica*, volume 37, pages 1771–1778.

Moog, C., Kotta, Ü., and Nõmm, S. (2003a). Extensions of linear algebraic methods to linear systems: an educational perspective. In *Proc. of the 6th IFAC Symposium on Advances in Control Education*, pages 179–184, Finland, Oulu.

Moog, C., Zheng, Y.-F., and Liu, P. (2003b). Input-output equivalence of nonlinear systems and their realizations. In *Proc. of the 15th IFAC World Congress: International Federation of Automatic Control*, pages 265–270, Barcelona, Spain, 21-26 July 2002. Plenary, Survey and Milestone Volume / Eds. E. F. Camacho [et al.]. [Oxford]: Pergamon.

Ondera, M. and Huba, M. (2005). Web-based tools for exact linearization control design. In *Proc. of the 16th IFAC World Congress*, Prague, Czech Republic, 4-8 July 2005.

Pearson, A. (1989). Identifiability and well-posedness in nonlinear systems modeling. Tampa, Florida, U.S.A.

Xia, X. and Moog, C. (2003). Identifiability of nonlinear systems with application to hiv/aids models. In *IEEE Transactions on Automatic Control*, volume 48(2), pages 330–335. Pergamon Press.