Slovak University of Technology in Bratislava
Institute of Information Engineering, Automation, and Mathematics

PROCEEDINGS

17th International Conference on Process Control 2009

Hotel Baník, Štrbské Pleso, Slovakia, June 9 – 12, 2009

ISBN 978-80-227-3081-5

http://www.kirp.chtf.stuba.sk/pc09

Editors: M. Fikar and M. Kvasnica

Tőnso, M., Rennik, H., Belikov, J., Kotta, Ü.: WebMathematica Based Tools for Continuous-Time Nonlinear Control Systems, Editors: Fikar, M., Kvasnica, M., In *Proceedings of the 17th International Conference on Process Control '09*, Štrbské Pleso, Slovakia, 625–633, 2009.

Full paper online: http://www.kirp.chtf.stuba.sk/pc09/data/abstracts/006.html

WEBMATHEMATICA BASED TOOLS FOR CONTINUOUS-TIME NONLINEAR CONTROL SYSTEMS

M. Tõnso *, **H. Rennik** *, **J. Belikov** * and **Ü. Kotta** *

* *Institute of Cybernetics at TUT, Akadeemia tee 21, 12618 Tallinn, Estonia. fax : +372 620 4151*
e-mails : {maris, heli, jbelikov, kotta}@cc.ioc.ee

Abstract: The package NLControl, developed in the Institute of Cybernetics at Tallinn University of Technology within Mathematica environment, has been made partially available over the internet using webMathematica tools. The package consists of functions that assist the solution of different modeling, analysis and synthesis problems for nonlinear control systems, described either by state or by input-output equations. This paper focuses on describing the webMathematica-based tools for continuous-time nonlinear control systems.

Keywords: webmathematica, symbolic computation, nonlinear control systems, linear algebraic framework.

1. INTRODUCTION

The differential algebraic methods, introduced by Fliess (1986) into the studies of nonlinear control theory, provide deep tools to address various nonlinear control problems. These methods have been supplemented by the related approach, based on the vector spaces of differential one-forms over suitable differential fields, associated with nonlinear control systems Conte et al. (2007). The latter approach, working with system tangent linearized equations, allows to simplify the solutions of many problems. The tools based on differential one-forms and the related methods, based on the theory of the skew polynomial rings, are characterized by their inherent simplicity and strong similarity to their linear counterparts. The latter makes these tools to be a natural choice in teaching engineering courses in nonlinear control and in practical applications.

Since the solutions of nonlinear control problems require a huge amount of symbolic computations, additional assistance is provided by the nonlinear control system software package NLControl Kotta and Tõnso (1999 2003), developed in the Institute of Cybernetics at Tallinn University of Technology. The package is based on algebraic methods of differential one-forms and skew polynomials, and is developed within (symbolic) software system Mathematica. This package provides basic tools for modeling, analysis and synthesis both for discrete- and continuous-time nonlinear systems. The reason for developing our own package is that nonlinear control systems, unlike their linear counterparts, practically miss a support of professional software products. For example, both Matlab Control System Toolbox and Mathematica Control System Professional Suite are applicable only for linear systems. Some custom-made packages based on symbolic computations for nonlinear control systems have been developed, see for example Kaddouri et al. (2006); de Jager (1995); Rothfuß and Zeitz (1996); Rodrigues-Millan (2001); Ondera (2008), but their distribution is extremely limited and only the last of them implements the methods based on algebraic tools and skew polynomial rings.

The functions from the package NLControl cannot be used outside of Mathematica environment. The main purpose of this paper is to introduce and describe a webMathematica-based application, developed by us, that allows the most important functions from NLControl make 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. This allows these tools to be applied in graduate courses as well as to make them available to a wider control community and to engineers. The other web-based tool for nonlinear control system is described in Ondera and Huba (2006). The authors are not aware of any other web implementations related to symbolic methods for nonlinear control systems. However, there exist numerous applications of webMathematica in control education, see for example Kujan et al. (2005), but practically all these tools are dedicated to linear systems. The second purpose of the paper is to compare our tools with those described in Ondera and Huba (2006). Because of space limitations, in this paper we focus only on the continuous-time case.

The paper is organized as follows. Section 2 gives a short overview of algebraic approach based on differential forms. Section 3 gives a overview of those aspects of webMathematica, necessary for our application, including technical details about webMathematica server technology. Section 4 describes functions, implemented in our webMathematica website together with numerous examples. In Section 5 we give a short overview of the file structure of our site and make a comparison with the other web-based tools for nonlinear control systems. The next section concludes the paper and after that we have an Appendix, containing a source code sample of the website.

The developed site is available at
`http://webmathematica.cc.ioc.ee/webmathematica/NLControl`.

2. ALGEBRAIC TOOLS BASED ON DIFFERENTIAL FORMS

Nonlinear control systems can be, in general, described in many different ways. In this paper we consider three of them. First, the system can be described by state equations

$$\begin{aligned} \dot{x} &= f(x,u) \\ y &= h(x), \end{aligned} \tag{1}$$

where $u \in U \subset \mathbb{R}^m$ is the input, $y \in Y \subset \mathbb{R}^p$ is the output, $x \in X$, an open subset of $\mathbb{R}^n$, is the state, $f : X \times U \to X$ and $h : X \to Y$ are real analytic functions. Alternatively, the control system may be described by the set of higher order input-output (i/o) differential equations relating the inputs $u_j, j = 1\ldots, m$, the outputs $y_i, i = 1, \ldots, p$, and a finite number of their time derivatives,

$$\begin{aligned} y_i^{(n_i)} = \varphi_i(&y_l, \ldots, y_l^{(n_{il}-1)}, u_k, \ldots, u_k^{(\alpha_{ik})}, \\ &l = 1, \ldots, p, \quad k = 1, \ldots, m), \\ &i = 1, \ldots, p \end{aligned} \tag{2}$$

where $\varphi = [\varphi_1, ..., \varphi_p]^T$ is a real analytic function, α_{ik}, n_{il} and n_i are the integer valued structural parameters of the system, satisfying the conditions $\alpha_{ik} < n_i$, $n_{il} < min(n_i, n_l)$ and $n_1 + ... + n_p = n$.

Below we will give a short overview of the algebraic approach based on differential one-forms. We will use the notations from Conte et al. (2007). One can associate with system (1) the field $\mathcal{K}$ of meromorphic functions in a finite number of independent system variables $\{x, u^{(k)}, k \geq 0\}$. Over the field $\mathcal{K}$ one can define a differential vector space $\mathcal{E} := \mathrm{span}_{\mathcal{K}}\{\mathrm{d}\varphi \mid \varphi \in \mathcal{K}\}$ spanned by the differentials of the elements of $\mathcal{K}$. The elements of $\mathcal{E}$ are called differential one-forms. The derivative operator s in $\mathcal{K}$ induces a derivative operator $s : \mathcal{E} \to \mathcal{E}$ by

$$\textstyle\sum_i a_i \mathrm{d}\varphi_i \to \sum_i s a_i \mathrm{d}(\varphi_i) + \sum_i a_i \mathrm{d}(s\varphi_i),$$

where $a_i, \varphi_i \in \mathcal{K}$. An one-form $\omega \in \mathcal{E}$ is called exact if $\mathrm{d}\omega = 0$, and closed if $\mathrm{d}\omega \wedge \omega = 0$ where by $\wedge$ is denoted the wedge product. The subspace of one-forms in $\mathcal{E}$ is called completely integrable if it admits the basis which consists only of closed one-forms. The relative degree r of a one-form $\omega \in \mathcal{E}$ is defined to be the least integer such that $S^r\omega \notin \mathrm{span}_{\mathcal{K}}\{\mathrm{d}x\}$. If such an integer does not exist, we set $r = \infty$. The relative degree is the number of times one has to apply the derivative operator to make the one-form explicitly dependent on the input.

The differential field $\mathcal{K}$ and the derivative operator s induce a ring of left polynomials in the derivative operator s, denoted by $\mathcal{K}[s]$. A left polynomial $p(s) \in \mathcal{K}[s]$ is written as

$$p(s) = a_k s^k + a_{k-1} s^{k-1} + ... + a_1 s + a_0,$$

where $a_i \in \mathcal{K}$, for $0 \leq i \leq k$. Each polynomial $p(s) \in \mathcal{K}[s]$ is a mapping of $\mathcal{E}$ into itself. An element $a \in \mathcal{K}$ does not commute with the derivative operator s, i. e. $a \cdot s \neq s \cdot a$. Since the multiplication of s and an element $a \in \mathcal{K}$ is not commutative and can be defined by the following rule

$$s \cdot a = a \cdot s + \dot{a},$$

the ring $\mathcal{K}[s]$ thus defined is a non-commutative skew polynomial ring and is proved to be a (left)

Ore ring. That is, the polynomials from $\mathcal{K}[s]$ satisfy the left Ore condition: for all non-zero $a, b \in \mathcal{K}[s]$, there exist non-zero $\alpha, \beta \in \mathcal{K}[s]$ such that $\alpha b = \beta a$.

The i/o equations (2) may be alternatively represented as Kotta et al. (2006) :

$$P(s)\mathrm{d}y = Q(s)\mathrm{d}u \tag{3}$$

where $P(s)$ and $Q(s)$ are $p \times p$ and $p \times m$-dimensional matrices respectively, whose elements $p_{ij}(s), q_{ij}(s) \in \mathcal{K}[s]$:

$$p_{il}(s) = s^{n_i} - \sum_{j=0}^{n_{il}} \frac{\partial \varphi_i}{\partial y_l^{(j)}} s^j$$
$$q_{ik}(s) = \sum_{r=0}^{\alpha_{ik}} \frac{\partial \varphi_i}{\partial u_k^{(r)}} s^r$$

and $\mathrm{d}y = [\mathrm{d}y_1, \ldots, \mathrm{d}y_p]^T$, $\mathrm{d}u = [\mathrm{d}u_1, \ldots, \mathrm{d}u_m]^T$. The equation (3) is obtained from (2) by applying the differential operation to it and using the notations $\mathrm{d}y_l^{(j)} = s^j \mathrm{d}y_l$, $\mathrm{d}u_k^{(r)} = s^r \mathrm{d}u_k$.

3. WEBMATHEMATICA

In this section we will give a short overview of webMathematica technology and features related to our website. In webMathematica server Mathematica kernel is running which is taking requests inserted by users from the webpage, calculating the results and sending them back to the webpage.

Our website (or user interface) uses standard web graphical user interface elements, such as text fields and check boxes. WebMathematica allows a site to deliver HTML pages to which have been added Mathematica commands and uses the request/response standard, followed by web servers. In our website request is entered into text fields and response can be in HTML, image or Mathematica notebook form. The request/response process is the following:

(1) Browser sends a request to webMathematica server. Request includes variables and their symbolic values entered from webpage.
(2) WebMathematica's kernel manager acquires Mathematica kernel. Variables and symbolic values are sent to this kernel.
(3) Mathematica kernel is initialized with input (request) parameters, it carries out the calculations, and returns the result to the server.
(4) The response is sent to the browser.
(5) WebMathematica server returns result to browser.

Requests are sent to the server with webMathematica webpages that are based on two standard Java technologies: Java Servlet and JavaServer Pages (JSP). Servlets are special Java programs that run in a Java-enabled web server, which is typically called a "servlet container". JSPs use a special library of tags that work with Mathematica. This 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 to generate examples.

There are many different combinations of hardware and operating systems that support webMathematica components. Before one starts to install webMathematica, one has to install Java and a servlet container. We are using Linux operating system and Tomcat as a web container.

4. IMPLEMENTED FUNCTIONS

By now, we have implemented 11 different functions for continuous-time nonlinear control systems from NLControl package into webMathematica website. There are 6 and 5 pages for systems, described by state and i/o equations, respectively. Most chosen functions are based on subspaces $\mathcal{H}_k$ Conte et al. (2007) and solve different modeling, analysis and synthesis problems.

Note that the tools of NLControl are not designed for approximate calculations. Therefore, all real (floating-point) numbers are transformed into rational numbers. Before printing the result, the rational numbers are transformed back to real numbers (keeping the accuracy of the input) for better overview. User interface is interconnected with help system, that provides a detailed explanation describing the applied methods behind the functions together with numerous typical examples.

User input data (i. e. system equations, input, output and state variables) validation has been implemented in the user interface. The results of the functions (together with the applied equations) can be given either in form of raster graphics image (gif file), html table or pure Mathematica output. Raster graphics image allows to print the mathematical experssions in traditional form, but textual forms are better for copying and pasting the result into other applications, LaTeX documents for instance. Pure Mathematica output allows to inspect the result generated by NLControl without any formatting functions applied. This option is necessary for developers as the best way to find the reasons of possible errors on the page.

The implemented functions can be divided into several subgroups. First, there are assistant functions, that do not solve any control problems. The first (`SequenceHk`) computes sequence of $\mathcal{H}_k$ subspaces that is a necessary building block of major-

ity of other functions. The other (`OrePolynomials`) is rather a collection functions which allow to perform the basic operations with Ore polynomials. The functions from the second group perform the transformations between different system descriptions (`Realization`, `ClassicStateToIO`, `IOToPolynomials` and `Reduction`). The functions from the third group check the system properties (`Accessibility`) and (`ObservabilityFiltration`). Finally, the last group consists by now only of one function `FeedbackLinearization`.

4.1 *Assistant functions*

Some of the assistant functions are made available on the website to allow the deeper study of the control system. The access to assistant functions may be useful, if the primary function, supposed to solve a control problem, fails on some reason. Assistant functions allow to inquire the reasons of failure and sometimes even solve the problem.

SequenceHk. The subspace $\mathcal{H}_k$ (for $k \geq 0$) contains the one-forms with relative degree equal to at least k and is defined by

$$\begin{aligned} \mathcal{H}_0 &= \mathrm{span}_{\mathcal{K}}\{\mathrm{d}x, \mathrm{d}u\} \\ \mathcal{H}_k &= \{\omega \in \mathcal{H}_{k-1} \mid \dot{\omega} \in \mathcal{H}_{k-1}\}, k \geq 1. \end{aligned} \tag{4}$$

Computing the sequence of subspaces $\mathcal{H}_k$ is necessary for several purposes. First, we need this sequence for checking realizability property and for finding the classical state space realization of the input-output equation (`Realization`). Second, it is also necessary for checking if the system is accessible or not and in case it is not, to find the non-accessible subspace and decompose the system into accessible and non-accessible subsystems (`Accessibility`). Moreover, this sequence is necessary to check if the system is static state feedback linearizable and for finding the state coordinate transformation (`FeedbackLinearization`).

The function `SequenceHk` computes first N elements in the sequence of subspaces $\mathcal{H}_k$, associated either to the state equations (1) or to the input-output equations (2), where N is a positive integer, specified by the user. It is also possible to integrate the computed one-forms by checking a corresponding radio button.

Example 1. Consider the system

$$y^{(3)} = yu + \ddot{y}\ddot{u}. \tag{5}$$

Compute the sequence $\mathcal{H}_k$ and integrate the subspaces. To save the space, limit the task to 4 first elements of the sequence. The function `SequenceHk` returns:

$$\begin{aligned} \mathcal{H}_1 &= \mathrm{span}_{\mathcal{K}}\{\mathrm{d}y, \mathrm{d}\dot{y}, \mathrm{d}\ddot{y}, \mathrm{d}u, \mathrm{d}\dot{u}, \mathrm{d}\ddot{u}\} \\ \mathcal{H}_2 &= \mathrm{span}_{\mathcal{K}}\{\mathrm{d}\dot{u}, \mathrm{d}u, \mathrm{d}\ddot{y}, \mathrm{d}\dot{y}, \mathrm{d}y\} \\ \mathcal{H}_3 &= \mathrm{span}_{\mathcal{K}}\{\mathrm{d}y, \mathrm{d}\dot{y}, \mathrm{d}\ddot{y} + \dddot{y}\mathrm{d}\dot{u}, \mathrm{d}u\} \\ \mathcal{H}_4 &= \mathrm{span}_{\mathcal{K}}\{\mathrm{d}y, \mathrm{d}\dot{y} - \ddot{y}\mathrm{d}u, \mathrm{d}\ddot{y} + uy\mathrm{d}u - \ddot{y}\mathrm{d}\dot{u}\} \end{aligned}$$

The integrated one-forms are:

$$\begin{aligned} \mathcal{H}_1 &: \{y, \dot{y}, \ddot{y}, u, \dot{u}, \ddot{u}\} \\ \mathcal{H}_2 &: \{y, \dot{u}, \ddot{y}, u, \dot{u}\} \\ \mathcal{H}_3 &: \{y, \dot{y}, u, e^{-\dot{u}}\ddot{y}\} \\ \mathcal{H}_4 &: \text{The system of one-forms is not integrable.} \end{aligned}$$

Ore polynomials. In order to implement functions based on the polynomial methods, for example `Reduction`, one needs basic tools for working with Ore polynomials. In Mathematica, unlike Maple, there are no built-in packages dedicated to Ore polynomials. An important point to emphasize is that the basic operations with Ore polynomials have to be performed modulo system equations (2). Whenever the expression $y_i^{(n_i)}$ appears in computations, it should be always replaced by $\varphi_i(\cdot)$ from equations (2). The higher order time derivatives $y_i^{(r_i)}, r_i > n_i$ require repeated replacements, for example $y_i^{(n_i+1)}$ should be at first substituted by $s\varphi_i(\cdot)$ and then each $y_i^{(n_i)}$ in the result given by $\varphi_i(\cdot)$.

The page `Ore polynomials` contains a collection of functions, which operate with polynomials from the Ore ring. It is possible to complete left and right division, find left and right greatest common divisor, left and right common multiple, etc.

Example 2. Consider the polynomials

$$\begin{aligned} p(s) &= us^2 + 1 \\ q(s) &= s + \dot{y}. \end{aligned}$$

The operation of left division finds the left quotient $\gamma(s)$ and remainder $r(s)$ such that $p(s) = q(s)\gamma(s) + r(s)$:

$$\begin{aligned} \gamma(s) &= us + (-\dot{u} - u\dot{y}) \\ r(s) &= (1 + 2\dot{u}\dot{y} + \ddot{u} + u(\dot{y}^2 + \ddot{y})) \end{aligned}$$

4.2 *Transformations between different system descriptions*

In this subsection, the functions are described that allow to transform one system description into another. First, `ClassicStateToIO` finds the i/o equations (2) from the state equations (1). Second, the function `Realization` checks if the i/o equations (2) can be transformed into the state space form, and finds the state equations, if possible. The function `IOToPolynomials` calculates the matrices $P(s)$ and $Q(s)$ in (3), given the input-output equations (2). The function `Reduction` checks if the i/o equations (2) can be reduced and finds the minimal equivalent system description.

Reduction. Function `Reduction` determines whether the system described by i/o equations (2) is irreducible or not, and if not, finds the reduced set of i/o equations. A nonlinear control system described by equations (2) is irreducible iff the greatest common left divisor $G_L(s)$ of polynomial matrices $P(s)$ and $Q(s)$ in (3) is a unimodular matrix (i. e. polynomial matrix with polynomial inverse) Kotta et al. (2006). In case of the non-unimodular $G_L(s)$, equation (3) may be rewritten as

$$G_L(s)[\tilde{P}(s)\mathrm{d}y(t) - \tilde{Q}(s)\mathrm{d}u(t)] = 0$$

and the reduced system equations can be found by integrating the one-forms in the brackets, perhaps after multiplying by the integrating factors.

Example 3. Consider the system

$$\begin{aligned} y_1^{(3)} &= 6u_1u_2 - 4\dot{u}_1\dot{u}_2 + 3\dot{y}_1 - 2u_2\ddot{u}_1 \\ &\quad -2u_1\ddot{u}_2 \\ \ddot{y}_2 &= -3u_2 + 2u_1u_2 - y_1\dot{u}_2 + \dot{y}_1. \end{aligned} \tag{6}$$

Reduction gives

$$\begin{aligned} 2u_1u_2 + \dot{y}_1 &= 0 \\ 3u_2 + y_1\dot{u}_2 + \ddot{y}_2 &= 0. \end{aligned} \tag{7}$$

Realization. The realization problem is to construct the state equations (1) of order $n = n_1 + ...+n_p$ from the set of i/o equations (2), if possible. Note that unlike the linear case, the state-space realization does not exist for every nonlinear i/o model (2). The necessary and sufficient realizability conditions in Conte et al. (2007) require that the subspaces $\mathcal{H}_k$, associated to i/o equation (2), for $1 \leq k \leq \alpha + 2$ are completely integrable. The state coordinates can be found by integrating the basis one-forms of $\mathcal{H}_{\alpha+2}$.

Function `Realization` determines whether the i/o equations (2) can be transformed into the state-space form and in case of the positive answer finds the state equations.

Example 4. Consider the system (7). The function `Realization` returns classical state equations:

$$\begin{aligned} \dot{x}_1 &= -2u_1u_2 \\ \dot{x}_2 &= -u_2x_1 + x_3 \\ \dot{x}_3 &= -u_2 \end{aligned} \tag{8}$$

$$\begin{aligned} x_1 &= y_1 \\ x_2 &= y_2 \\ x_3 &= u_2y_1 + \dot{y}_2 \end{aligned}$$

Example 5. Consider the system (5). The function `Realization` returns the text "Classical state space form does not exist for the system".

ClassicStateToIO. This function transforms classic state equations (1) into i/o equations (2).

Example 6. Transform the system (8) back into form of i/o equations. The result

$$\begin{aligned} \dot{y}_1 &= -2u_1u_2 \\ 3u_2 + y_1\dot{u}_2 + \ddot{y}_2 &= 0 \end{aligned}$$

is the same as equations (7), but generally the transformation from i/o equations to classical state equations and then back to i/o equations does not necessarily lead to the same i/o equations.

IOToPolynomials. This function computes the $p \times p$ and $p \times m$-dimensional matrices $P(s)$ and $Q(s)$ in (3), given the input-output equations (2).

4.3 *Checking the system properties*

In this subsection the functions `Accessibility` and `ObservabilityFiltration` are described that allow to check the accessibility (controllability) and observability properties of the system, respectively.

Accessibility. Accessibility is the structural property of the nonlinear system that in the linear case reduces to the controllability property. The system (1) is said to be accessible if there does not exist any non-zero autonomous variable for (1) in $\mathcal{K}$. Note that the autonomous variable of the system is a variable, not influenced by control, and therefore, cannot be changed by controller. The necessary and sufficient condition for accessibility is $\mathcal{H}_\infty = \{0\}$ Conte et al. (2007).

In case the system is not accessible, it can be decomposed into accessible and non-accessible subsystems. Note that $\mathcal{H}_\infty$ is a non-accessible subspace, and because of its complete integrability, it has an integrable basis $\mathcal{H}_\infty = \mathrm{span}_\mathcal{K}\{\mathrm{d}\zeta_1, ..., \mathrm{d}\zeta_r\}$ Conte et al. (2007). Since $\mathcal{H}_\infty$ is invariant under applying derivative operator,

$$\begin{aligned} \dot{\zeta}_1 &= f_1(\zeta_1, ..., \zeta_r), \\ &\vdots \\ \dot{\zeta}_r &= f_r(\zeta_1, ..., \zeta_r). \end{aligned}$$

The accessible subspace $\mathcal{X}_a := \mathcal{X}/\mathcal{H}_\infty$ such that $\mathcal{X}_a \oplus \mathcal{H}_\infty = \mathcal{X}$ has also an integrable basis $\mathrm{span}_\mathcal{K}\{\mathrm{d}\zeta_{r+1}, ..., \mathrm{d}\zeta_n\}$. Therefore, we have

$$\begin{aligned} \dot{\zeta}_{r+1} &= f_{r+1}(\zeta, u), \\ &\vdots \\ \dot{\zeta}_n &= f_n(\zeta, u). \end{aligned}$$

Observability filtration. Conte et al. (2007) The system (1) is said to be observable, if

$$\operatorname{rank}_{\mathcal{K}} \frac{\partial H_{n-1}}{\partial x} = n, \tag{9}$$

where $H_{n-1} = \big(h(x), sh(x), \ldots, s^{n-1}h(x)\big)^T$. Define the difference vector spaces $\mathcal{Y}^k$, $\mathcal{Y}$ and $\mathcal{U}$ for system (1) as follows

$$\begin{aligned} \mathcal{Y}^k &= \operatorname{span}_{\mathcal{K}}\{\mathrm{d}y,\ 0 \le t \le k\}, \\ \mathcal{Y} &= \operatorname{span}_{\mathcal{K}}\{\mathrm{d}y,\ t \ge 0\}, \\ \mathcal{U} &= \operatorname{span}_{\mathcal{K}}\{\mathrm{d}u,\ t \ge 0\}. \end{aligned}$$

To define the observable subspace, introduce the sequence of subspaces, called the observability filtration

$$0 \subset \mathcal{O}_0 \subset \mathcal{O}_1 \ldots \subset \mathcal{O}_k \subset \ldots, \tag{10}$$

where $\mathcal{O}_k := \mathcal{X} \cap (\mathcal{Y}^k + \mathcal{U})$. The subspace $\mathcal{X} \cap (\mathcal{Y} + \mathcal{U})$ is called the observable space of the system (1) and can be computed as the limit $\mathcal{O}_\infty$ of the observability filtration (10), $\mathcal{O}_\infty = \mathcal{X} \cap (\mathcal{Y} + \mathcal{U})$. The following statements are equivalent: (i) the system is observable, (ii) the condition (9) is satisfied, and (iii) $\mathcal{O}_\infty = \mathcal{X}$. In order to decompose the system (1) into the observable and unobservable subsystems, one has to find an exact basis $\{\mathrm{d}\zeta_1, \ldots, \mathrm{d}\zeta_r\}$. Complete the set $\{\mathrm{d}\zeta_1, \ldots, \mathrm{d}\zeta_r\}$ to a basis $\{\mathrm{d}\zeta_1, \ldots, \mathrm{d}\zeta_r, \mathrm{d}\zeta_{r+1}, \ldots, \mathrm{d}\zeta_n\}$ of $\mathcal{X}$. Then, in the coordinates ζ, the system (1) reads as

$$\begin{aligned} \dot\zeta_1 &= f_1(\zeta_1, \ldots, \zeta_r, u) \\ &\vdots \\ \dot\zeta_r &= f_r(\zeta_1, \ldots, \zeta_r, u) \\ \dot\zeta_{r+1} &= f_{r+1}(\zeta, u) \\ &\vdots \\ \dot\zeta_n &= f_n(\zeta, u) \\ y &= h(\zeta_1, \ldots, \zeta_r). \end{aligned}$$

Example 7. Consider the system

$$\begin{aligned} \dot x_1 &= -ux_1 + x_3 \\ \dot x_2 &= u^2 + x_2 \\ \dot x_3 &= ux_3 + x_1 \\ y &= x_1 + x_2. \end{aligned}$$

Observability filtration gives

$$\begin{aligned} \mathcal{O}_1 &= \operatorname{span}_{\mathcal{K}}\{\mathrm{d}x_1 + \mathrm{d}x_2\} \\ \mathcal{O}_2 &= \operatorname{span}_{\mathcal{K}}\{(-1-u)\mathrm{d}x_1 + \mathrm{d}x_3, \mathrm{d}x_1 + \mathrm{d}x_2)\} \\ \mathcal{O}_3 &= \operatorname{span}_{\mathcal{K}}\{\mathrm{d}x_3, \mathrm{d}x_2, \mathrm{d}x_1\} \\ \mathcal{O}_4 &= \operatorname{span}_{\mathcal{K}}\{\mathrm{d}x_3, \mathrm{d}x_2, \mathrm{d}x_1\}. \end{aligned}$$

4.4 *Feedback linearization*

System (1) is said to be static state feedback linearizable if there exist a state diffeomorphism

$$\xi = \Phi(x)$$

and a regular static state feedback of the form

$$u = \alpha(x, v),$$

with $\operatorname{rank}_{\mathcal{K}}[\partial\alpha(\cdot)/\partial v] = m$, such that in the new coordinates the compensated system equations are in the form

$$\begin{aligned} \dot\xi_{i1} &= \xi_{i2} \\ &\vdots \\ \dot\xi_{ik_i-1} &= \xi_{ik_i} \\ \dot\xi_{ik_i} &= v_i, \quad i = 1, ..., m. \end{aligned}$$

For linearizability conditions we need to define an integer k^* for sequence $\mathcal{H}_k$. There exist an integer $k^* \le n$ such that, for $0 \le k \le k^*$, $\mathcal{H}_{k+1} \subset \mathcal{H}_k$ but $\mathcal{H}_{k+1} \ne \mathcal{H}_k$ and $\mathcal{H}_{k^*+1} = \mathcal{H}_{k^*+2} = \ldots = \mathcal{H}_\infty$. The necessary and sufficient conditions for feedback linearizability are:

(1) $\mathcal{H}_\infty = \{0\}$,
(2) $\mathcal{H}_k$ is completely integrable for $1 \le k \le k^*$.

Not every accessible system can be linearized by static state feedback. The function `Linearization` checks whether this is possible and in the case of affirmative answer finds the state coordinate change, the feedback and the linear closed-loop equations.

Example 8. Consider the system

$$\begin{aligned} \dot x_1 &= -\frac{x_1(u + ux_1x_2 - x_2x_3)}{x_3} \\ \dot x_2 &= \frac{1}{x_1} - x_2^2 \\ \dot x_3 &= u + \frac{x_3}{1 + x_1x_2}. \end{aligned}$$

The function `Linearization` gives

$$\begin{aligned} \dot z_1 &= z_2 \\ \dot z_2 &= z_3 \\ \dot z_3 &= v \end{aligned}$$

$$\begin{aligned} z_1 &= -\frac{1 + z_1z_2}{x_1x_3} \\ z_2 &= \frac{x_2(1 + x_1x_2)}{x_1x_3} \\ z_3 &= \frac{1 - 2x_1^2x_2^3 + x_1x_2 - 2x_1x_2^2}{x_1^2x_3} \\ v &= \frac{uz_1^4 - 6z_2^3 + 6z_1z_2z_3}{z_1^2}. \end{aligned}$$

5. ADDITIONAL COMMENTS

5.1 *Structure of the website*

NLControl website is a project under development, we are adding new functions and reorganizing old pages. What is described here is the state of the art in January, 2009. The structure of the

website is very simple. For every function there is one jsp file which allows to enter the input data using html forms and the other jsp file which shows the result generated by webMathematica. These two files are similar for all functions, with some modifications. The source code of the webpages providing access to the function `Realization` is given in Appendix. The simplicity of the file structure is also its weakness since it yields a large number of almost identical files. Moreover, if one has to make some minor changes, one may need to edit all these files. Our future aim is to reorganize the site in a more sophisticated way, such that it would be easier to administer, for example add new functions, update Mathematica code blocks or change layout.

5.2 *Comparison with other web-based tools*

Besides the fact that the problems handled are different, that is the tools described in Ondera and Huba (2006) are dedicated solely to the exact static state feedback linearization problem, there are other points to be mentioned. As far as linearization problem is concerned, the web-tools in Ondera and Huba (2006) allow more than our function `Linearization`, namely the user may additionally submit the desired closed-loop poles of a pole-placement controller and to perform a simulation of the resulting closed-loop system.

There is also a difference in chosen technology. The web-tools in Ondera and Huba (2006) 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 (2006) 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.

CONCLUSIONS

This paper describes how the symbolic computation package NLControl, developed within Mathematica environment has been made available over the internet using webMathematica programming features. The website has been tested on Microsoft Internet Explorer and Mozilla Firefox web browsers. Our future goal is to implement more NLControl functions into webMathematica website, improve the documentation and example library. Especially, we want to include functions that allow output feedback linearization and/or decoupling, construct the transfer function from the i/o equations (2) or from the state equations (1), find the discrete-time model from the continuous-time system equations and solve the model-matching problem.

APPENDIX

Comments are included in italics. The `<msp:allocateKernel>` tag is used to obtain a Mathematica kernel for computation, `</msp:allocateKernel>` releases the kernel. The tags `<msp:evaluate>` ... `</msp:evaluate>` contain Mathematica/webMathematica blocks between them.

RealizationCont.jsp - input data file

```
<%@ page language="java"  %>
<%@ taglib uri="/webMathematica-taglib"
    prefix="msp" %>
<%@ page import="java.util.Random"%>
```

heading specifying the languages used in the file.

```
<script language="javascript">
function openChild(file,window) {
    childWindow = open(file,window,'resizable=yes,
    scrollbars = yes,
    width = 600,
    height = 600');
    if (childWindow.opener == null)
    childWindow.opener = self;
    }
</script>
```

Javascript function for opening explanation and examples windows.

```
<html>
<head>
<title>Realization of the input-output equation
  </title>
</head>
<BODY bgcolor="#e9e9e9" text="#123456">
<h1>Realization</h1>
Function Realization determines whether the non-
linear higher order input-output differential
equation can be realized in the classical state-
space form and the if the i/o equation is
realizable, finds the state equations.<br>
```

Beginning of the html-part, contains a short explanation about the current function.

```
<A HREF="javascript:openChild(
'RealizationContExplanation.html','vin2')">
read more..</A>
<br><br>
<A HREF="javascript:openChild('
RealizationExamplesCont.html','vin2')">
Examples</A><br>
```

Javascript function is called to open pdf document explaining the mathematical theory behind the function and to open examples file containing more complicated system equations.

```
<msp:allocateKernel>
<form action="RealizationContResult.jsp"
  name="parentForm" method="POST">
```

Beginning of the html-form

```
<%
String examples[][] = { {"{y'''[t] == y''[t] u[t]
    + u'[t] y[t] + u[t]}","",""},
  {"{y1''[t] == y1'[t] u2'[t] + u1[t],\n
    y2''[t] == u1[t] + y2[t]}","",""},
  };
Random random = new Random();
int randInt = random.nextInt(examples.length);

   StringBuffer jexpr_eq = new StringBuffer();
   StringBuffer jexpr_Ut = new StringBuffer();
   StringBuffer jexpr_Yt = new StringBuffer();
   jexpr_eq.append(examples[randInt][0]);
   jexpr_Ut.append(examples[randInt][1]);
   jexpr_Yt.append(examples[randInt][2]);
%>
```

Java code which fills system equations' field with randomly selected simple equations from above defined array.

```
<b>Input-output equation(s): </b>
<br>
<textarea name="eq" rows=4cols=60>
<msp:evaluate>
  MSPValue[$$eq,"<%=jexpr_eq%>"]
</msp:evaluate> </textarea>
<br>
<b>Input variables:</b>
<input type="text" name="Ut"  size="10"
value="<msp:evaluate>MSPValue[$$Ut,""]
</msp:evaluate>"/>
<br>
By default, all variables starting with "u" and
depending on t are considered as input variables,
e.g. u[t], u1[t], u2[t+1].
<br>
To use different symbols, enter the list,
e.g. {v1[t], v2[t]}
<br><br>
<b>Output variables:</b>
<input type="text" name="Yt"  size="10"
value="<msp:evaluate>MSPValue[$$Yt,""]
</msp:evaluate>"/>
<br>
By default, all variables starting with "y" and
depending on t are considered as input variables,
e.g. y[t], y1[t], y2[t+1].
<br>
To use different symbols, enter the list,
e.g. {w1[t], w2[t]}
<br>
```

Text fields for entering i/o equations and lists of input and ouput variables.

```
Choose the output format:
<br>
<input type = "radio" name = "outputformat"
  value = "gif" checked>gif picture<br>
<input type = "radio" name = "outputformat"
  value = "html">html table<br>
<input type = "radio" name = "outputformat"
  value = "mtca">pure Mathematica output<br>
<br>
Leave out argument t to make the result
  visually as short as possible:
<br>
<input type = "radio" name = "argumentt"
  value = "yes">yes<br>
<input type = "radio" name = "argumentt"
  value = "no" checked>no<br>
```

Radio buttons to specify formatting options.

```
<br>
<input type = "hidden" name = "systemtype"
  value = "cont">
<input type = "submit" name = "submitButton"
  value = "Evaluate">
<br><br><br><br>
</form>
</msp:allocateKernel>
</body>
</html>
```

The hidden field `systemtype` *tells to the result file that it is a continuous-time system. This allows to use the same result file for the continuous- and discrete-time systems.*

RealizationResult.jsp - result file

```
<%@ page language="java"  %>
<%@ taglib uri="/webMathematica-taglib"
    prefix="msp" %>
<html>
<BODY bgcolor="#e9e9e9" text="#123456">
The data you entered:<br>
<msp:allocateKernel>
<b>Input-output equation(s):</b>
<br><msp:evaluate>$$eq</msp:evaluate>
<br>
<b>Input variables:</b>
<br><msp:evaluate>$$Ut</msp:evaluate>
<br>
<b>Output variables:</b>
<br><msp:evaluate>$$Yt</msp:evaluate>
```

Prints the equations, entered by the user to allow him to check the input data.

```
<br>
<msp:evaluate>
    Needs["MSP`"];
    Needs["NLControl`Core`"];
    Needs["NLControl`Sequences`"];
    Needs["NLControl`InputOutput`"];
    Needs["NLControl`Ore`"];
    Needs["NLControl`Integration`"];
    Needs["NLControl`StateTransformations`"];
    Needs["NLControl`Identifiability`"];
    Needs["NLControl`Observability`"];
    Needs["NLControl`Web`"];
</msp:evaluate>
```

Loads NLControl package files

```
<msp:evaluate>
  eqs = Null;
  Ut = Null;
  Yt = Null;
  eqs = MSPToExpression[$$eq];
  Yt = MSPToExpression[$$Yt];
  Ut = MSPToExpression[$$Ut];
```

Converts the content of the text fields into Mathematica expressions.

```
  correctdata = eqs=!= Null;
  realizable = False;

  If[ correctdata,
    If[ Ut === Null,
      Ut = DetectVariables[eqs, t, "u*"]  ];
    If[ Yt === Null,
      Yt = DetectVariables[eqs, t, "y*"]  ];
    type = Switch[ $$systemtype,
      "cont", TimeDerivative, "disc", Shift ];
```

```
    data = IO[ eqs, Ut, Yt, t, type];
    result = TimeConstrained[ Realization[ data,
      ToExpression["x"<>ToString[#]][t] &], 30];
    timearg = If[$$argumentt==="yes", False, True];
      "Your system is:",
  (* Else *)
     "Uncorrect input data."
  ]
</msp:evaluate><br><br>
```

If the entered expressions have correct syntax, Realization is applied to them

```
<msp:evaluate>
  If[ correctdata,
    iWebForm[ data, $$outputformat,
        TimeArgument -> timearg]
  ] (* End If *)
</msp:evaluate><br><br>
```

Prints the initial i/o equations to screen.

```
<msp:evaluate>
  If[ correctdata,
    Switch[ result,
      {}, "Classical state space form does not
                exist for the system.",
      _Realization, "Program is unable to find the
            classical state space form for the
            system.",
      $Aborted, "Time limit 30 seconds exceeded.",
      {_StateSpace, _List}, realizable = True;
            "Classical state space equations are:"
    ] (* End Switch *)
  ] (* End If *)
</msp:evaluate><br><br>
```

If the classical state equations do not exist, can not be found or solution takes too much time, the corresponding messages are printed.

```
<msp:evaluate>
    If[realizable, iWebForm[ result,
        $$outputformat, TimeArgument -> timearg]]
</msp:evaluate>
```

If the classical state equations were found, they are printed to screen.

```
</msp:allocateKernel>
</form>
</body>
</html>
```

ACKNOWLEDGMENTS

This work was partially supported by the ETF Grant no 6922.

References

G. Conte, C. Moog, and A. Perdon. *Algebraic Methods for Nonlinear Control Systems*. Springer-Verlag, London, 2007.

B. de Jager. The use of symbolic computations in nonlinear control. is it viable? In *IEEE Transactions on Automatic Control*, volume 40, pages 84–89, 1995.

M. Fliess. Automatique en temps discret et algèbre aux différences. In *Forum Mathematicum*, volume 2, pages 213–232, 1986.

A. Isidori. Nonlinear control systems (3rd ed.). Berlins, 1995. Springer.

A. Kaddouri, S. Blais, M. Ghribi, and O. Akhrif. Nlsoft: An interactive graphical software for desiging nonlinear controllers. In *Mathematics and Computers in Simulation*, volume 71, pages 377–384, 2006.

Ü. Kotta and M. Tõnso. 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.]:, 2003. Springer.

Ü. Kotta and M. Tõnso. 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, 1999.

Ü. Kotta, P. Kotta, S. Nõmm, and M. Tõnso. Irreducibility conditions for continuous-time multi-input multi-output nonlinear systems. In *2006 9th International Conference on Control, Automation, Robotics and Vision*, pages 807–811. Singapore, 2006.

P. Kujan, M. Hromčik, and M. Šebek. Web-based mathematica platform for systems and control education. In *Proc of thr 13th Mediterranean Conference on Control and Automation*, pages 376–381, Cyprus, 2005.

H. Nijmeijer and A. van der Schaft. Nonlinear dynamical control systems. New York, 1990. Springer.

M. Ondera. *Computer-aided design of nonlinear systems and their generalized transfer functions*. PhD thesis, Slovak University of Technology in Bratislava, FEI, 2008.

M. Ondera and M. Huba. Web-based tools for exact linearization control design. In *Proceedings of the 14th IEEE Mediterranean Conference on Control and Automation*, Ancona, Italy, 28-30 June 2006, 2006.

J. Rodrigues-Millan. Integrated symbolic-graphic-numeric analysis and design in nonlinear control through notebooks in mathematica. In *Lecute Notes in Computer Science*, pages 405–420. Springer-Verlag, Berlin, 2001.

R. Rothfuß and M. Zeitz. A toolbox for symbolic nonlinear feedback design. In *Proc. of the 13th IFAC World Congress*, pages 283–288, San Francisco, 1996.