TALLINN UNIVERSITY OF TECHNOLOGY
Faculty of Information Technology
Department of Computer Control

Juri Belikov

Synthesis and identification of nonlinear discrete-time models for model based control

Master thesis

Supervisors: Sven Nõmm, PhD
Institute of Cybernetics at TUT
Eduard Petlenkov, PhD
Department of Computer Control,
Tallinn University of Technology

Tallinn 2008

Contents

Abstract

Synthesis and identification of nonlinear discrete-time models for model based control

Present thesis is devoted to symbolical discretization of nonlinear continuous-time systems and application of a class of neural networks based discrete-time models to control of nonlinear systems.

The first part of this thesis concentrates its attention on discretization theory of continuous-time systems and especially on the theory of so called finitely discretizable systems. Besides that, the author considers the implementation of the functions on the basis of the presented theory in Computer Algebra System (CAS) *Mathematica*.

The second part of this thesis is devoted to the application of Neural Networks based Additive Nonlinear Autoregressive eXogenous (NN-based ANARX) structure to identification and control of nonlinear systems. The advantage of using this structure lies in the fact that it is always linearizable by dynamic output feedback as well as representable in a classical state-space form. Different approaches for calculation of control signals by using NN-ANARX based dynamic output linearization algorithm are also considered. The effectiveness of presented techniques is demonstrated on numerical examples.

All calculations and simulations shown in this thesis are performed in *Mathematica* and MATLAB/Simulink environments.

Kokkuvõte

Mittelineaarsete diskreetaja mudelite identifitseerimine ja süntees mudelil põhineva juhtimise jaoks

Käesolev väitekiri käsitleb mittelineaarsete pidevaja süsteemide sümboolset diskretiseerimist ning ühe tehisnärvivõrkudel põhineva diskreetaja mudelite klassi rakendust mittelineaarsete süsteemide juhtimiseks.

Väitekirja esimeses osas on vaadeldud pidevaja süsteemide diskretiseerimise teooria kusjuures erilist tähelepanu on pööratud lõplikult diskretiseeritavatele süsteemidele. Esitatud teooria on rakendatud *Mathematica*-s autori poolt kirjutatud funktsioonide kujul.

Teine osa on pühendatud tehisnärvivõrkudel baseeruva ANARX (Additive Nonlinear Autoregressive eXogenous) struktuuri rakendamisele mittelineaarsete süsteemide identifitseerimiseks ja juhtimiseks. Selle struktuuri eeliseks juhtimissüsteemide projekteerimise seisukohast seisned selles, et ta on alati lineariseeritav dünaamilise tagasiside abil ning on esitatav klassikalise olekuruumi kujul. Töös on ka antud ülevaade erinevatest juhtimisssignaali arvutamise meetodist ANARX mudelil põhinevates juhtimissüsteemides. Lisaks olemasolevatele meetoditele autori poolt on väljatöötatud kaks alternatiivset algoritmi, mis võimaldavad suurendada mittelineaarsete süsteemide klassi, mille ANARX mudelil põhineva juhtimisalgoritm on rakendatav. Töös käsitletud meetodite efektiivsus on demonstreeritud mitmetel numbrilistel näidetel.

Kõik töös esitatud arvutused ja simulatsioonid on teostatud *Mathematica* ja MATLAB/ Simulink keskkondades.

Acknowledgments

This work was partially supported by the Estonian Science Foundation through the grant #6884 and by the Science foundation Archimedes through the personal Kristjan Jaak's scholarship.

Chapter 1

Introduction

1.1 State of the Art

1.1.1 Discretization

During the second half of the 20-th century computers start to grow rapidly and become more and more popular. As a result of the proposed computational power and programmability, they infiltrated into all spheres of human activity. For example, computers may be found in different places ranging from battleships to industrial robots, medical tools, and children's toys. Furthermore, the development of the software has begun. The scientific disciplines do not remain in the party too.

In the latter half of 1980's on a world scene appears a new Computer Algebra System (CAS) *Mathematica*. By the moment of its occurrence in the market of the commercial and free-of-charge software there were already well established software giants such as Maple, Maxima, etc. Thanks to several benefits of *Mathematica* like the symbolic computational power and attractive graphical user interface, it has become widely applied in different research areas and teaching courses. To the present moment this is a one of the most popular programs in its class.

This thesis describes several *Mathematica* functions implemented by the author into **NLControl** package [23] that allow to compute symbolically exact or approximate discrete-time models of continuous-time nonlinear controlled systems. These functions implement the theoretical results of [2], [10] and [18]. The need for discrete-time models comes from the fact that though most physical systems evolve in continuous-time, most model-based control schemes are implemented digitally. In principle, of course, the digital controller can be obtained just by sampling the continuous-time controller. This approach, though still most common, may be unsuitable for slow sampling rates. Moreover, fast sampling intersects with quantification of signals and may yield complex effects that degrade the system performance [27] and numerical errors [3].

1.1.2 Neural Networks

The history of mathematical models of biological neurons begins in 1943, when McCulloch and Pitts formalize the concept of a neural network in the fundamental article about logic calculation of ideas and nervous activity [25]. After that in 1949 Hebb offers the first training algorithm. However, the real beginning of neural networks starts in 1962, when Rosenblatt introduces the concept of a perceptron [41]. It is used for pattern recognition, forecasting of weather, etc. It seemed that the construction of a full-fledged artificial intelligence is just around the corner. But, in 1969 Minsky publishes the formal proof of limitation of the perceptron and shows, that it is unable to solve some problems (the problem of ”parity” and ”one in the block”), connected with invariancy of representations. In a light of these events, and also that neural networks demanded greater computing capacities, interest to them sharply falls. The period of decline was long enough. However, in 1980’s notions of the network minimizing energy and self-organizing map were introduced. Since the middle of 1980’s a new round of development of neural networks begins. Except a new theoretical discoveries, this was promoted also by elimination of one of the main problems, namely a lack of computing power. Modern high technologies allow for reasonable time to create and train a neural network of almost any complexity.

During a long time scientists have been concentrated on approximation capabilities of neural networks. However, at the same time, little attention was paid to the structural aspects. Applications of the specific neural network with so called ANARX structure for identification and control of nonlinear systems are considered in this thesis. ANARX is a subclass of well known NARX models and has all time instances separated. This structure has a number of advantages, which in more detail are considered in Chapter 5.

1.2 Outline of the Thesis

The thesis is organized as follows. Chapter 2 contains a brief summary of mathematical tools used in the next parts of the thesis. It describes notions of input-output and state-space multi-input multi-output systems. While the second type is more used in the first part of the thesis, the first one is required only in Chapter 5. Both continuous- and discrete-time cases are considered.

Chapter 3 discusses the basic discretization theory of continuous-time nonlinear systems. Besides that two techniques, which allow one to obtain the exact or approximate discrete-time models, are presented. Additionally, in this chapter some of the standard facts connected to the finitely discretizable and nilpotent systems are reviewed. For the convenience of the reader the author repeats the relevant material without proofs, thus making this thesis self-contained. At the end of the chapter the general scheme of construction of a discrete-time model is presented. Furthermore, it establishes the relations between programmed functions, which are considered in the next chapter.

Chapter 4 contains a description of programmed functions and instructions for their application. Five different functions are considered. Each of them describes an execution algorithm and application instructions with all necessary information about arguments, their types etc. Additional possibilities and joint work of the functions are drawn in the end of the chapter.

In Chapter 5 the notion of neural networks based ANARX structure is presented. Its advantages over the classical NARX model are also discussed. NN-based ANARX structure is used for identification and control of nonlinear systems. Additionally, the different solutions of the problem of the control signal calculation proposed in the previous researches are summarized. Finally, the practical application of the presented techniques is demonstrated on the basis of numerical examples.

Concluding remarks and subjects for the further research and development are presented in the last chapter.

Chapter 2

Notations and Definitions

The modern control theory operates with many concepts by applying definitions and theorems from various scientific disciplines. Therefore, this chapter will serve as a brief introduction of only basic notations, which will be important throughout the whole thesis.

2.1 Nonlinear Systems

In this section the author has compiled some basic facts of nonlinear continuous- and discrete-time systems. At the beginning, the notion of the input-output system is presented following [35]. After that, the state-space representation of the system is considered according to [26].

2.1.1 Input-Output Systems

In general, the relationships between the input u and the output y signals of a system can be represented by the differential equations

$$f(y, \dot{y}, \ldots, y^{(n)}, u, \dot{u}, \ldots, u^{(n)}) = 0 \tag{2.1.1}$$

in the continuous-time case or by the difference equations

$$f(y(k), y(k-1), \ldots, y(k-n), u(k), u(k-1), \ldots, u(k-n)) = 0 \tag{2.1.2}$$

in the discrete-time case. Both in (2.1.1) and (2.1.2) $u = (u_1, \ldots, u_r) \in \mathbb{R}^r$ is a vector of system inputs, $y = (y_1, \ldots, y_m) \in \mathbb{R}^m$ is a vector of system outputs, $f(\cdot)$ are real analytic functions. Equations (2.1.1) and (2.1.2) are called the input-output form of the system.

2.1.2 State-Space Systems

The majority of nonlinear systems, that can be modeled, are represented by a finite number of the state equations

$$\begin{aligned} \dot{x} &= f(x,u) \\ y &= h(x) \end{aligned} \tag{2.1.3}$$

in the continuous-time case or by

$$\begin{aligned} x(k+1) &= f(x(k),u(k)) \\ y(k) &= h(x(k)) \end{aligned} \tag{2.1.4}$$

in the discrete-time case. Both in (2.1.3) and (2.1.4) $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$ is a vector of state variables, $u = (u_1, \ldots, u_r) \in \mathbb{R}^r$ is a vector of system inputs, $y = (y_1, \ldots, y_m) \in \mathbb{R}^m$ is a vector of system outputs, $f : \mathbb{R}^n \times \mathbb{R}^r \to \mathbb{R}^n$ and $h : \mathbb{R}^n \to \mathbb{R}^m$ are real analytic functions. Equations (2.1.3) and (2.1.4) are called the state-space representation of the system.

Taking into account topics and character of the first part of this thesis, the state equations (2.1.3) and (2.1.4) can be rewritten in the following way

$$\dot{x} = f(x,u) \tag{2.1.5}$$

in the continuous-time case and

$$x(k+1) = f(x(k),u(k)) \tag{2.1.6}$$

in the discrete-time case.

Notice that there are two main classes of systems used in present thesis, namely single-input single-output (SISO) systems in case if $r, m = 1$ and multi-input multi-output (MIMO) systems in case if $r, m > 1$.

Chapter 3

Discretization of Continuous-time Nonlinear Systems

The purpose of this chapter is introduction of some concepts of discretization theory. The chapter is organized as follows. Section 3.1 represents the necessary discretization theory. After that, Sections 3.2 and 3.3 provide a detailed exposition of two discretization methods. Next, in Section 3.4 we introduce the basic definitions of finitely discretizable systems. Besides that we recall notions of dilation and homogeneous degree, and a sufficient condition of finite discretization is also considered. Finally, Section 3.5 represents the basic theory of nilpotent systems. The theory described below is based on [2], [10], [15] and [26].

3.1 Discretization

Often times discrete-time systems originate by ”sampling” of a continuous-time systems. This is an idealized process in which, for the given continuous-time system $\Sigma : u \to x$ (a dynamical system that maps the input u into the state x, as shown in Figure 3.1(a)) we seek a new system $\Sigma_d : u(k) \to x(k)$ (a dynamical system that maps the discrete-time signal $u(k)$ into the discrete-time state $x(k)$, as shown in Figure 3.1(b)).

Both systems Σ and Σ_d are related in the following way. If $u(k)$ is constructed by taking ”samples” every T seconds of the continuous-time signal u, then the output $x(k)$, pre-

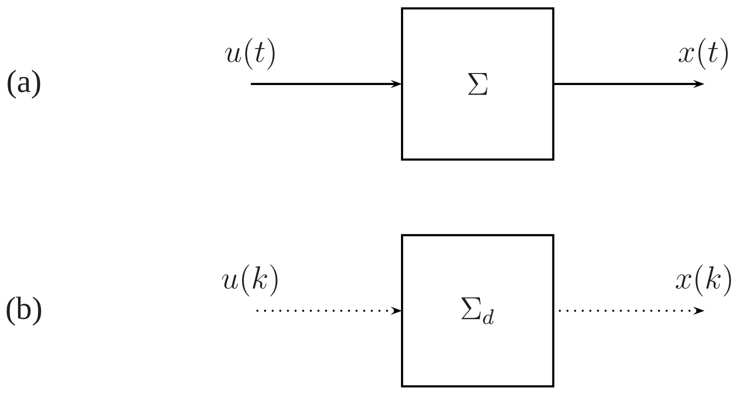


Figure 3.1: (a) Continuous-time system Σ; (b) discrete-time system Σ_d

dicted by the model Σ_d, corresponds to the samples of the continuous-time state $x(t)$ at the same sampling instances.

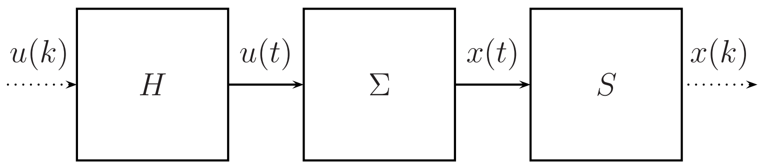


Figure 3.2: Discrete-time system Σ_d

To develop such kind of model, one can use the scheme shown in Figure 3.2, which consists of the cascade combination of the blocks H, Σ and S, where each block in the figure represents the following:

- S represents a *sampler*, i.e. a device that reads the continues variable x every T seconds and produces the discrete-time output $x(k)$, given by $x(k) = x(kT)$. Clearly, this block is an idealization of the operation of an analog-to-digital converter. For easy visualization, continuous and dotted lines were used in Figure 3.2 to represent continuous and discrete-time signal, respectively.

- Σ represents the plant, seen as a mapping from the input u to the state x, the mapping $\Sigma : u \to x$ determines the trajectory $x(t)$ by solving the differential equation

$$\dot{x} = f(x,u) \; . \tag{3.1.1}$$

- H represents a *hold* device that converts the discrete-time signal or sequence $u(k)$, into the continuous-time signal $u(t)$. Clearly, this block is implemented by using a digital-to-analog converter. One can assume an ideal conversion process takes place in which H ”holds” the value of the input sequence between samples (Figure 3.3), given by

$$u(t) = u(k)$$

for $kT \leq t < (k+1)T$.

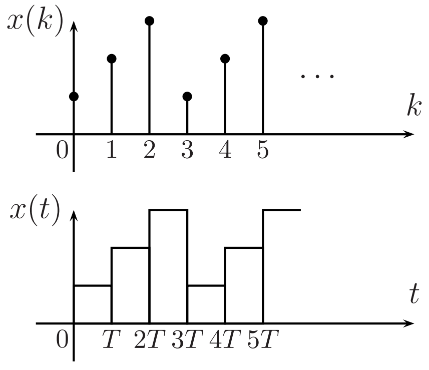


Figure 3.3: Action of the hold device H

In other words, the basis on which discrete-time model is derived from continuous-time system is the assumption that $u(t)$ in (2.1.5) is kept piecewise constant and does not change between the equidistant sampling instants. The discrete-time model relates the sampled state of the continuous-time system at the end of sampling period to the sampled state at the beginning of the sampling period and the sampled input. The relation between the continuous-time system and discrete-time model is given by the fact that the states of the discrete-time model at sampling instants $(k+1)T$ are equal to the solution of the differential equation (2.1.5) at time $t = (k+1)T$, starting at time $t = kT$ in $x(kT)$ and with a constant control $u_k = u(kT)$ applied so that derivatives of control will be zero. Clearly, the sampled-data representation depends on the sampling time T. This model is one of the most popular for implementing discrete-time systems, because of its simplicity.

Usually it is impossible to find the exact discrete-time model of the nonlinear plant Σ given by (2.1.5). The reason is that finding the exact solution requires solving the nonlinear differential equations, what is very difficult or impossible in general. Given this fact, one is usually forced to use an approximate models. There are several different methods of constructing exact or approximate discrete-time models, but we introduce only notations of so called Integration method and Taylor series expansion method, which will be described in Sections 3.2 and 3.3, respectively.

3.2 Direct Integration Method

This section describes how to obtain the exact discrete-time model from a continuous-time system using Integration method. In order to find the discrete-time model, one has to solve the equation (2.1.5) over the sampling half-closed interval $[kT, kT+T)$ under the assumption that during this interval $u(t)$ is kept constant. In principle, one may apply *Mathematica* built-in function `DSolve` to find the solution. Unfortunately, equation (2.1.5), in general, cannot be solved exactly and hence the exact discrete-time model is not available except a few simple cases. Additionally to `DSolve` we have written another function that tries to solve the state equations in the special block triangular form

$$\dot{x}_{[i]} = A_i x_{[i]} + f_{[i]}\left(x_{[1]}, \ldots, x_{[i-1]}, u_{[i]}\right), \quad i = 1, \ldots, r \tag{3.2.1}$$

step by step, starting from the first subsystem. In (3.2.1), $x_{[i]} \in \mathbb{R}^{n_i}$ and $u_{[i]} \in \mathbb{R}^{m_i}$ are the vectors of states and inputs of the i-th subsystem, respectively. Some subsystems may have same, different or none control input. Note that in [2] it has been proved to be possible to solve (3.2.1) under the assumption that $f_{[i]}$'s in (3.2.1) are polynomial functions, though this assumption is not necessary (see example 4.2.2).

An example is given below to illustrate the theory reviewed above.

Example 3.2.1 Consider the system [10]

$$\begin{array}{rcl} \dot{x}_1 & = & u \\ \dot{x}_2 & = & x_1 \\ \dot{x}_3 & = & x_2 + x_1^2 \end{array} . \tag{3.2.2}$$

According to the theory given above, the discrete-time model can be obtained by consistent integration of each state of the system. First of all, we find the definite integral of the first state variable $\dot{x}_1$ as follows

$$x_1 := \int_0^t \dot{x}_1 dt = x_1(0) + \int_0^t u dt = x_1(0) + tu. \tag{3.2.3}$$

Then, we make analogous calculations for the $\dot{x}_2$, but replace x_1 with obtained on the previous step expression (3.2.3)

$$x_2 := \int_0^t \dot{x}_2 dt = x_2(0) + \int_0^t x_1 dt = \\ = x_2(0) + \int_0^t (x_1(0) + tu)dt = x_2(0) + tx_1(0) + \frac{t^2}{2}u. \quad (3.2.4)$$

After that

$$x_3 := \int_0^t \dot{x}_3 dt = x_3(0) + \int_0^t (x_1^2 + x_2)dt = \\ = x_3(0) + \int_0^t \left((x_1(0) + tu)^2 + x_2(0) + tx_1(0) + \frac{t^2}{2}u \right) dt = \\ = x_3(0) + tx_1^2(0) + t^2 x_1(0)u + \frac{t^3}{3}u^2 + tx_2(0) + \frac{t^2}{2}x_1(0) + \frac{t^3}{6}u. \quad (3.2.5)$$

Finally, using equations (3.2.3)-(3.2.5), the exact discrete-time model is described by the following equations

$$\begin{array}{rcl} x_1(kT+T) & = & x_1(kT) + Tu(kT) \\ x_2(kT+T) & = & x_2(kT) + Tx_1(kT) + \frac{T^2}{2}u(kT) \\ x_3(kT+T) & = & x_3(kT) + Tx_1^2(kT) + Tx_2(kT) + \frac{T^2}{2}x_1(kT) + \\ & + & T^2 x_1(kT)u(kT) + \frac{T^3}{6}u(kT) + \frac{T^3}{3}u^2(kT) \end{array}.$$

3.3 Taylor Series Expansion Method

This section explains how to obtain the discrete-time model from the continuous-time nonlinear plant using Taylor series expansion method. As was mentioned above, the nonlinear ordinary differential equations in general, cannot be solved exactly and hence the exact form of the sampled data model is difficult to obtain. However, if the system (2.1.5) has no closed-form solution, one can express the solution in (infinite) Taylor series over the sampling interval $t \in [kT, (k+1)T)$

$$x(kT+t) = x(kT) + t\dot{x}(kT) + \frac{t^2}{2!}\ddot{x}(kT) + \cdots + \frac{t^r}{r!}x^{(r)}(kT) + \cdots,$$

where the higher order derivatives of x can be obtained by repetitive differentiation of the right hand side of (2.1.5)

$$x_i^{(r)} = \sum_{j=1}^{n} \frac{\partial f_i^{(r-2)}}{\partial x_j} f_j := f_i^{(r-1)}, \quad (3.3.1)$$

where $r > 1$. As we are interested in the state x only at the sampling instant $kT+T$, then we obtain for $t = T$

$$x((k+1)T) = x(kT) + \sum_{r\geq 1}^{\infty} \frac{T^r}{r!} x^{(r)}(kT). \tag{3.3.2}$$

Denote the i-th component of x and vector valued function $f(x,u)$ with $u(t) = u_k$ by x_i and $f_{u_k,i}$, respectively. Note that the derivatives of x can be expressed in terms of Lie differential operator

$$L_{f_{u_k}} = \sum_{i=1}^{n} f_{u_k,i}(x) \frac{\partial}{\partial x_i},$$

associated with the vector field $f_{u_k}(x)$, as follow $x^{(r)} = L^r_{f_{u_k}}$ for $r \geq 1$. Then the equation (3.3.2) can be rewritten in the following form

$$x((k+1)T) = x(kT) + \sum_{r\geq 1}^{\infty} \frac{T^r}{r!} L^r_{f_{u_k}} x(kT). \tag{3.3.3}$$

The exact discrete-time model (3.3.3) of a continuous-time system (2.1.5) is, in general, in the form of the infinite series with respect to the sampling period T. In reality to compute the model, one must confine oneself with finite number of terms in this series. In that way we reach the notion of approximate discrete-time models. Computing approximate discrete-time models corresponds to the truncation of the infinite series with respect to the sampling period T at the fixed order τ, which defines the order of approximation of the sampled system.

Two examples are given below to illustrate the Taylor series expansion method.

Example 3.3.1 Consider the system [14], which is depicted in Figure 3.4.

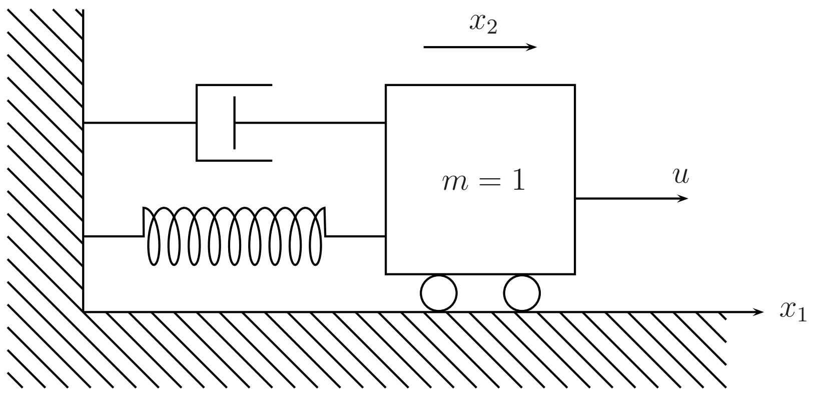


Figure 3.4: Mechanical system. Unit mass with damper and spring

Let x_1 be the position of the cart that moves with velocity x_2. We assume that the spring force is described by a nonlinear function $\phi(x_1)$ of the position and that the damper force is a nonlinear function $\psi(x_2)$ of the velocity. The system is then described by the following equations

$$\begin{array}{rcl} \dot{x}_1 & = & x_2 \\ \dot{x}_2 & = & -\phi(x_1) - \psi(x_2) + u \end{array}. \tag{3.3.4}$$

Next, using the formula (3.3.1), we find the 2-nd

$$\begin{array}{rcl} \ddot{x}_1 & = & -\phi(x_1) - \psi(x_2) + u \\ \ddot{x}_2 & = & -x_2\phi^{'}(x_1) + \psi^{'}(x_2)(\phi(x_1) + \psi(x_2) - u) \end{array}$$

and the 3-rd order derivatives of each state of the system

$$\begin{array}{rcl} x_1^{(3)} & = & -x_2\phi^{'}(x_1) + \psi^{'}(x_2)(\phi(x_1) + \psi(x_2) - u) \\ x_2^{(3)} & = & \phi^{'}(x_1)\left(-u + \phi(x_1) + \psi(x_2) + x_2\psi^{'}(x_2)\right) - x_2^2\phi^{''}(x_1) + \\ & + & \psi^{'2}(x_2)(u - \phi(x_1) - \psi(x_2)) - \psi^{''}(-u + \phi(x_1) + \psi(x_2))^2 \end{array}.$$

Then, it follows from (3.3.2) that the 3-rd order discrete-time model of the continuous-time system (3.3.4) is described by the following equations

$$\begin{array}{rcl} x_1((k+1)T) & = & x_1(kT) + Tx_2(kT) + \frac{T^2}{2}(u(kT) - \phi(x_1(kT)) - \psi(x_2(kT))) + \\ & + & \frac{T^3}{6}\left(-x_2\phi^{'}(x_1(kT)) + \psi^{'}(x_2(kT))\right)(\phi(x_1(kT)) + \psi(x_2(kT)) - \\ & - & u(kT))) \\ x_2((k+1)T) & = & x_2(kT) + T(u(kT) - \phi(x_1(kT)) - \psi(x_2(kT))) + \\ & + & \frac{T^2}{2}\big(-x_2\phi^{'}(x_1(kT)) + \psi^{'}(x_2(kT))(\phi(x_1(kT)) + \psi(x_2(kT)) - \\ & - & u(kT))\big) + \frac{T^3}{6}\big(-\psi^{''}(x_2(kT))(-u(kT) + \phi(x_1(kT)) + \\ & + & \psi(x_2(kT)))^2 + \psi^{'2}(x_2(kT))(u(kT) - \phi(x_1(kT)) - \psi(x_2(kT))) \\ & - & x_2^2(kT)\phi^{''}(x_1(kT)) + \phi^{'}(x_1(kT))\big(-u(kT) + \phi(x_1(kT)) \\ & + & \psi(x_2(kT)) + x_2(kT)\psi^{'}(x_2(kT))\big)\big) \end{array}$$

Example 3.3.2 Consider the system (3.2.2) from Example 3.2.1. Denote

$$\begin{array}{rcl} f_1 & := & u \\ f_2 & := & x_1 \\ f_3 & := & x_2 + x_1^2 \end{array}$$

and using the formula (3.3.1) compute the 2-nd order derivatives

$$\begin{array}{lcl} f_1^{(1)} & := & \ddot{x}_1 = \frac{\partial f_1}{\partial x_1} f_1 + \frac{\partial f_1}{\partial x_2} f_2 + \frac{\partial f_1}{\partial x_3} f_3 = 0 \\ f_2^{(1)} & := & \ddot{x}_2 = \frac{\partial f_2}{\partial x_1} f_1 + \frac{\partial f_2}{\partial x_2} f_2 + \frac{\partial f_2}{\partial x_3} f_3 = u \\ f_3^{(1)} & := & \ddot{x}_3 = \frac{\partial f_3}{\partial x_1} f_1 + \frac{\partial f_3}{\partial x_2} f_2 + \frac{\partial f_3}{\partial x_3} f_3 = 2x_1 u + x_1 \end{array} .$$

After that we compute analogously the 3-rd order derivatives

$$\begin{array}{lcl} x_1^{(3)} & = & 0 \\ x_2^{(3)} & = & \frac{\partial f_2^{(1)}}{\partial x_1} f_1 + \frac{\partial f_2^{(1)}}{\partial x_2} f_2 + \frac{\partial f_2^{(1)}}{\partial x_3} f_3 = 0 \\ x_3^{(3)} & = & \frac{\partial f_3^{(1)}}{\partial x_1} f_1 + \frac{\partial f_3^{(1)}}{\partial x_2} f_2 + \frac{\partial f_3^{(1)}}{\partial x_3} f_3 = 2u^2 + u \end{array} .$$

It is obvious that the 4-th order derivatives of each state will be equal to zero. Then, it follows from (3.3.2) that the exact discrete-time model is described by the following equations

$$\begin{array}{lcl} x_1(kT+T) & = & x_1(kT) + Tu(kT) \\ x_2(kT+T) & = & x_2(kT) + Tx_1(kT) + \frac{T^2}{2} u(kT) \\ x_3(kT+T) & = & x_3(kT) + T(x_2(kT) + x_1^2(kT)) + \\ & + & \frac{T^2}{2}(2x_1(kT)u(kT) + x_1(kT)) + \frac{T^3}{6}(2u^2(kT) + u(kT)) \end{array} .$$

One can easily check that the obtained here model is the same as in Example 3.2.1.

3.4 Finitely Discretizable Systems

There is a subclass of so called finitely discretizable nonlinear systems, which has been introduced in [10]. For a finitely discretizable system a solution is completely determined by a finite number of the state derivatives. In other words, for a finitely discretizable system, for certain positive integer K, $L_f^k x = 0$ for all $k > K$. Note that the property of being finitely discretizable is not invariant under the arbitrary change of local coordinates. However, this property is invariant under a polynomial change of coordinates. In [10], a sufficient condition is given for a polynomial system

$$\dot{x}(t) = X_0(x(t)) + \sum_{j=1}^{m} u_j X_j(x(t)) \tag{3.4.1}$$

to be finitely discretizable. In order to characterize this subclass, we recall the definitions of dilation and homogeneous degree.

Definition 3.4.1 *A dilation is a map*: $\delta_t : \mathbb{R}^+ \times \mathbb{R}^n \to \mathbb{R}^n$ *of the form* $\delta_t(x) = (t^{r_1}x_1, \ldots, t^{r_n}x_n)$, *where the integers* r_i *satisfy the inequality* $1 \leq r_1 \leq r_2 \leq \cdots \leq r_n$.

Definition 3.4.2 *A vector field* $Z(x) = (Z_1(x), \ldots, Z_n(x))^T$ *is said to be homogeneous of degree* $-s$ *with respect to a dilation* δ_t, *if*

$$(T\delta_t(Z)) = t^s(Z \circ \delta_t), \tag{3.4.2}$$

where the components of Z *are the polynomial functions in variables* $x_1, \ldots, x_n$ *and* $T\delta_t$ *is the Jacobian matrix of the map* $\delta_t(x)$, *i.e.*

$$T\delta_t = \begin{pmatrix} t^{r_1} & 0 & \cdots & 0 \\ 0 & t^{r_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & t^{r_n} \end{pmatrix}. \tag{3.4.3}$$

A sufficient condition of finite discretization is formulated in the following theorem.

Theorem 3.4.1 *If* $X_0, \ldots, X_m$ *in* $(3.4.1)$ *are polynomial vector fields, homogeneous of degree* -1, *with respect to the dilation* $\delta_t(x) = (t^{r_1}x_1, \ldots, t^{r_n}x_n)$, *then system* $(3.4.1)$ *is finitely discretizable with* r_n *being the order of the highest nonzero term in Taylor series expansion.*

Using Definition 3.4.2 and putting $s = 1$ in (3.4.2), the assumptions of Theorem 3.4.1 are satisfied iff the system of equations

$$\langle dx_i, T\delta_t(X_j)\rangle = t\langle dx_i, X_j\rangle \circ \delta_t \tag{3.4.4}$$

for $i = 1, \ldots, n$, $j = 0, \ldots, m$ is solvable with respect to $r_1, r_2, \ldots, r_n$.

An example is given below to illustrate the theory presented above.

Example 3.4.1 Consider the chained system [30]:

$$\begin{array}{rcl} \dot{x}_1 & = & u_1 \\ \dot{x}_2 & = & u_2 \\ \dot{x}_3 & = & x_2 u_1 \end{array} . \tag{3.4.5}$$

The vector fields associated with (3.4.5) are $X_1 = \frac{\partial}{\partial x_1} + x_2\frac{\partial}{\partial x_3}$ and $X_2 = \frac{\partial}{\partial x_2}$. In this case, with respect to Definition 3.4.1 the dilation is $\delta_t(x_1, x_2, x_3) = (t^{r_1}x_1, t^{r_2}x_2, t^{r_3}x_3)$ and tangent space is

$$T\delta_t = \begin{pmatrix} t^{r_1} & 0 & 0 \\ 0 & t^{r_2} & 0 \\ 0 & 0 & t^{r_3} \end{pmatrix}. \tag{3.4.6}$$

Then, the vector fields X_1 and X_2, influenced by tangent space (3.4.6), can be rewritten as $T\delta_t(X_1) = t^{r_1}\frac{\partial}{\partial x_1} + t^{r_3}x_2\frac{\partial}{\partial x_3}$ and $T\delta_t(X_2) = t^{r_2}\frac{\partial}{\partial x_2}$. Next, we find scalar products of $\langle dx_i, X_j\rangle$ and $\langle dx_i, T\delta_t(X_j)\rangle$, where $i = 1, 2, 3$ and $j = 1, 2$.

$$\begin{cases} \langle dx_1, X_1\rangle &= 1 \\ \langle dx_2, X_1\rangle &= 0 \\ \langle dx_3, X_1\rangle &= x_2 \end{cases}, \qquad \begin{cases} \langle dx_1, X_2\rangle &= 0 \\ \langle dx_2, X_2\rangle &= 1 \\ \langle dx_3, X_2\rangle &= 0 \end{cases},$$

$$\begin{cases} \langle dx_1, T\delta_t(X_1)\rangle &= t^{r_1} \\ \langle dx_2, T\delta_t(X_1)\rangle &= 0 \\ \langle dx_3, T\delta_t(X_1)\rangle &= t^{r_3}x_2 \end{cases}, \qquad \begin{cases} \langle dx_1, T\delta_t(X_2)\rangle &= 0 \\ \langle dx_2, T\delta_t(X_2)\rangle &= t^{r_2} \\ \langle dx_3, T\delta_t(X_2)\rangle &= 0 \end{cases}.$$

Now, using formula (3.4.4), we try to calculate the values of r_1, r_3 and r_3.
From $\langle dx_1, T\delta_t(X_1)\rangle = t\langle dx_1, X_1\rangle \circ \delta_t$ it follows that $t^{r_1} = t$ and consequently $r_1 = 1$.
From $\langle dx_2, T\delta_t(X_2)\rangle = t\langle dx_2, X_2\rangle \circ \delta_t$ it follows that $t^{r_2} = t$ and consequently $r_2 = 1$.
From $\langle dx_3, T\delta_t(X_1)\rangle = t\langle dx_3, X_1\rangle \circ \delta_t$ it follows that $t^{r_3}x_2 = tx_2t^{r_2}$ and consequently $r_3 = 2$.
Then, the dilation can be rewritten as $\delta_t(x_1, x_2, x_3) = (tx_1, tx_2, t^2x_3)$.
Since it was possible to calculate values of r_1, r_2 and r_3, assumptions of Theorem 3.4.1 are satisfied, and as a result the system (3.4.5) is finitely discretizable in original coordinates at most the order 3.

3.5 Nilpotent Systems

The property of finite discretizability is not invariant under the arbitrary coordinate transformation. However, if the vector fields $X_0, \ldots, X_m$ in (3.4.1) generate the nilpotent Lie algebra $L(X_0, \ldots, X_m)$, then locally there exist the state coordinates in which (3.4.1) is finitely discretizable. The Lie algebra $L(X_0,\dots,X_m)$ is the smallest linear subspace that contains the vector fields $X_0,\dots,X_m$ and is closed under the Lie bracket operation

$$[X_\alpha, X_\beta](x) := \sum_{i=1}^{n}\sum_{j=1}^{n}\left(\frac{\partial X_{\alpha,i}}{\partial x_j}X_{\beta,j} - \frac{\partial X_{\beta,i}}{\partial x_j}X_{\alpha,j}\right)\frac{\partial}{\partial x_i}.$$

The Lie algebra $L(X_0,\dots,X_m)$ is said to be nilpotent if there exists an integer $k > 0$ such that all Lie products of length greater than k vanish, i.e. $[X_0,[X_1,\dots,[X_p,X_{p+1}]]] = 0$ for all $p > k$.

Example 3.5.1 Consider the following system [1]

$$\begin{array}{rcl} \dot{x}_1 &=& u_1 \\ \dot{x}_2 &=& u_1 \tan x_3 \\ \dot{x}_3 &=& u_2 \cos^2 x_3 \end{array} . \qquad (3.5.1)$$

The vector fields associated with (3.5.1) are $X_1 = \frac{\partial}{\partial x_1} + \tan x_3 \frac{\partial}{\partial x_2}$ and $X_2 = \cos^2 x_3 \frac{\partial}{\partial x_3}$. Next, we want to check whether the Lie algebra $L(X_1, X_2)$ is nilpotent or not. Then, the iterated Lie brackets should be calculated.

$$[X_1, X_2] = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -2\cos x_3 \sin x_3 \end{pmatrix}\begin{pmatrix} 1 \\ \tan x_3 \\ 0 \end{pmatrix} - \\ - \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & \frac{1}{\cos^2 x_3} \\ 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \\ \cos^2 x_3 \end{pmatrix} = -\frac{\partial}{\partial x_2},$$

$$[X_1, [X_1, X_2]] = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} 1 \\ \tan x_3 \\ 0 \end{pmatrix} - \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & \frac{1}{\cos^2 x_3} \\ 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} 0 \\ -1 \\ 0 \end{pmatrix} = 0,$$

$$[X_2, [X_1, X_2]] = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \\ \cos^2 x_3 \end{pmatrix} - \\ - \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -2\cos x_3 \sin x_3 \end{pmatrix}\begin{pmatrix} 0 \\ -1 \\ 0 \end{pmatrix} = 0.$$

As we can see the following Lie brackets $[X_1, [X_1, X_2]] = 0$ and $[X_2, [X_1, X_2]] = 0$. It means that the system (3.5.1) is nilpotent, and consequently there exist the state coordinates in which (3.5.1) is finitely discretizable.

Notice that the verifying whether the original system is nilpotent or not is the first step. After that one should to calculate new state coordinates in which the original system obtains the finitely discretizable form. This is a difficult task. Then, we have developed algorithm, which computes new state coordinates. However, at this time this algorithm is at a stage of completion and consequently will not be presented in this thesis. Therefore, this place in Fig. 3.5 is denoted by the dotted arrow.

The general scheme of constructing a discrete-time model from a continuous-time system (2.1.5) is shown in Fig. 3.5.

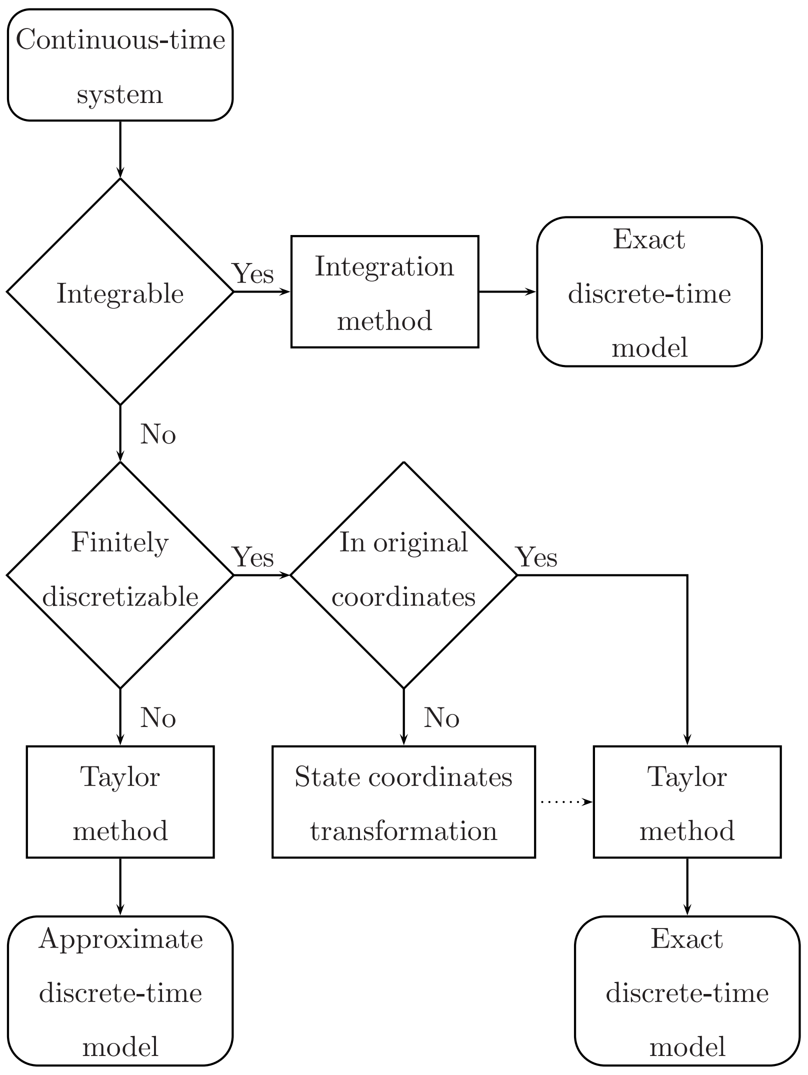


Figure 3.5: The general scheme of constructing a discrete-time model

Chapter 4

Implementation in CAS *Mathematica*

The subject of this chapter is to discuss in detail implementation aspects of different functions. The first function `Discretization` allows one to find the exact or approximate discrete-time models of continuous-time system using two different methods. After that one can plot continuous-time system and discrete-time model by using function `DiscretizationPlot`. Besides that, `Dilation` and `HomogeneousDegree` were created for verifying whether the original continuous-time system is finitely discretizable in original state coordinates or not. Finally, the function `Nilpotency` checks if the original system is nilpotent or not. All those functions were implemented in the form defined by the CAS *Mathematica* and integrated into the package **NLControl**.

The chapter is organized as follows. Four sections of the chapter describe functions listed above. Each section is divided into two subsections. The first of them describes how function works and the second one how to use it. All sections are illustrated by examples. The last section presents some additional examples.

4.1 A Short Guidance on the NLControl package

This section briefly provides necessary information on the **NLControl** package and considers some useful functions. We introduce notions only of those basic functions, which are used throughout this chapter.

In order to make possible using of any function from the package, we should to load it. It is easily can be done by the following command

```
In[1]:= «NLControl‘Master‘
```

We suppose that in the examples used in this chapter, the **NLControl** package is already loaded.

Consider the systems (2.1.5) and (2.1.6). If one wants to perform computations with one of such systems, then it should be entered in the form determined by *Mathematica* and **NLControl** package as shown below. Such form can be obtained using the following function

`StateSpace[`*`f`*`, `*`Xt`*`, `*`Ut`*`, `*`t`*`, `*`Type`*`]`,

where *`f`* is a list of the state functions, *`Xt`* and *`Ut`* define lists of the state and input variables, respectively, and *`t`* is a time argument. Finally, the argument *`Type`* may have one of the following two values. *`TimeDerivatives`* stands for continuous-time case and *`Shift`* for discrete-time case.

Another important function is `BookForm[]`, which displays different objects produced by the package in a standard form pleasant to read. One of the optional arguments of this functions is *`TimeArgument`*. Its value can be `True`, `False` or `Subscripted`. If the value of *`TimeArgument`* is `True`, then the time argument `t` will be printed for each variable in brackets `[]`. If the value is `False`, then the time argument will be left out to make the output result visually as short as possible. `Subscripted` means that `t` will be printed as a subscript. The default value of *`TimeArgument`* is `True`.

The application of the functions presented above is shown by the following example.

Example 4.1.1 Consider the continuous-time system

$$\begin{array}{rcl}\dot{x}_1 & = & u_1x_2^2 \\ \dot{x}_2 & = & x_3+u_2 \\ \dot{x}_3 & = & x_1u_2\end{array}\quad . \qquad (4.1.1)$$

In order to enter and display the system (4.1.1) in the standard form, we should use the following commands

```
In[2]:= f = {u1[t]*x2[t]^2,x3[t]+u2[t],x1[t]*u2[t]};
        Xt = {x1[t],x2[t],x3[t]};
        Ut = {u1[t],u2[t]};
        contSys = StateSpace[f,Xt,Ut,t,TimeDerivative];
        BookForm[contSys]
```

Notice that in the fifth row *Mathematica* creates the object `contSys`. One of the further possibilities on using it consists in calling the function `BookForm[]` as shown above. Then, the output result is

```
        x1[t] = u1[t]x2[t]^2
Out[6]= x2[t] = x3[t]+u2[t]
        x3[t] = x1[t]u2[t]
```

Of course, there are much more useful functions in the **NLControl** package, but here we have described only those that are most oftenly used in the examples of this thesis.

4.2 Discretization

4.2.1 Description of the function

The function `Discretization` implements two discretization methods of continuous-time system (2.1.5), described in Sections 3.2 and 3.3. Its returning value is a discrete-time model of the form (2.1.6). The function has one necessary and three optional arguments. The block-diagram of the function `Discretization` is presented in Fig. 4.1. When called, it performs the following steps:

- **Step 1**. The function `Discretization` initializes the basic variables and on the basis of this information executes the following steps.
- **Step 2**. It defines whether `Taylor` or `Triangular` discretization method was chosen.

- **Step 3a**. If the first method was selected, then firstly function checks whether the original system is finitely discretizable or not, calling function `Dilation`. After that on the basis of obtained information it calculates the exact or approximate discrete-time model using equations (3.3.1) and (3.3.2).

- **Step 3b**. If someone decided to chose the second method, then the exact discrete-time model will be calculated using theory presented in Section 3.2.

- **Step 4**. The function `Discretization` returns the result as follows

$$\left\{ \sum_{r\geq 0}^{p} \frac{T^r}{r!} x_1^{(r)}(kT), \dots, \sum_{r\geq 0}^{p} \frac{T^r}{r!} x_n^{(r)}(kT) \right\},$$

  where p is some positive integer number, in `Taylor` method case, and

$$\{x_1(kT) + F_1(u(kT), T), \dots, x_n(kT) + F_n(x_1(kT), \dots, x_{n-1}(kT), u(kT), T)\}$$

  in `Triangular` method case.

4.2.2 Application

The function `Discretization` has one necessary and three optional arguments:

`Discretization[`*`contSystem`*`, `*`PrintInfo`*`, `*`Method`*`,`
`ApproximationOrder``].`

The *`contSystem`* (continuous-time system) is a necessary argument of the function and has to be given by the state equations in the form, determined by **NLControl** package. The optional argument *`PrintInfo`* is a boolean variable having one of the values `True` or `False`. If its value is `True`, then the result of the function, besides other, will be displayed in traditional form (it can be useful, because the main returned result is in the form, determined by *Mathematica*, and for some people it can be not absolutely clear). The default value of *`PrintInfo`* is `True`. The argument *`Method`* defines one of the following discretization methods `Taylor` or `Triangular`. The first of them stands to Taylor series expansion method and the second to direct Integration method. The default value is `Taylor`. The last optional argument *`ApproximationOrder`* defines

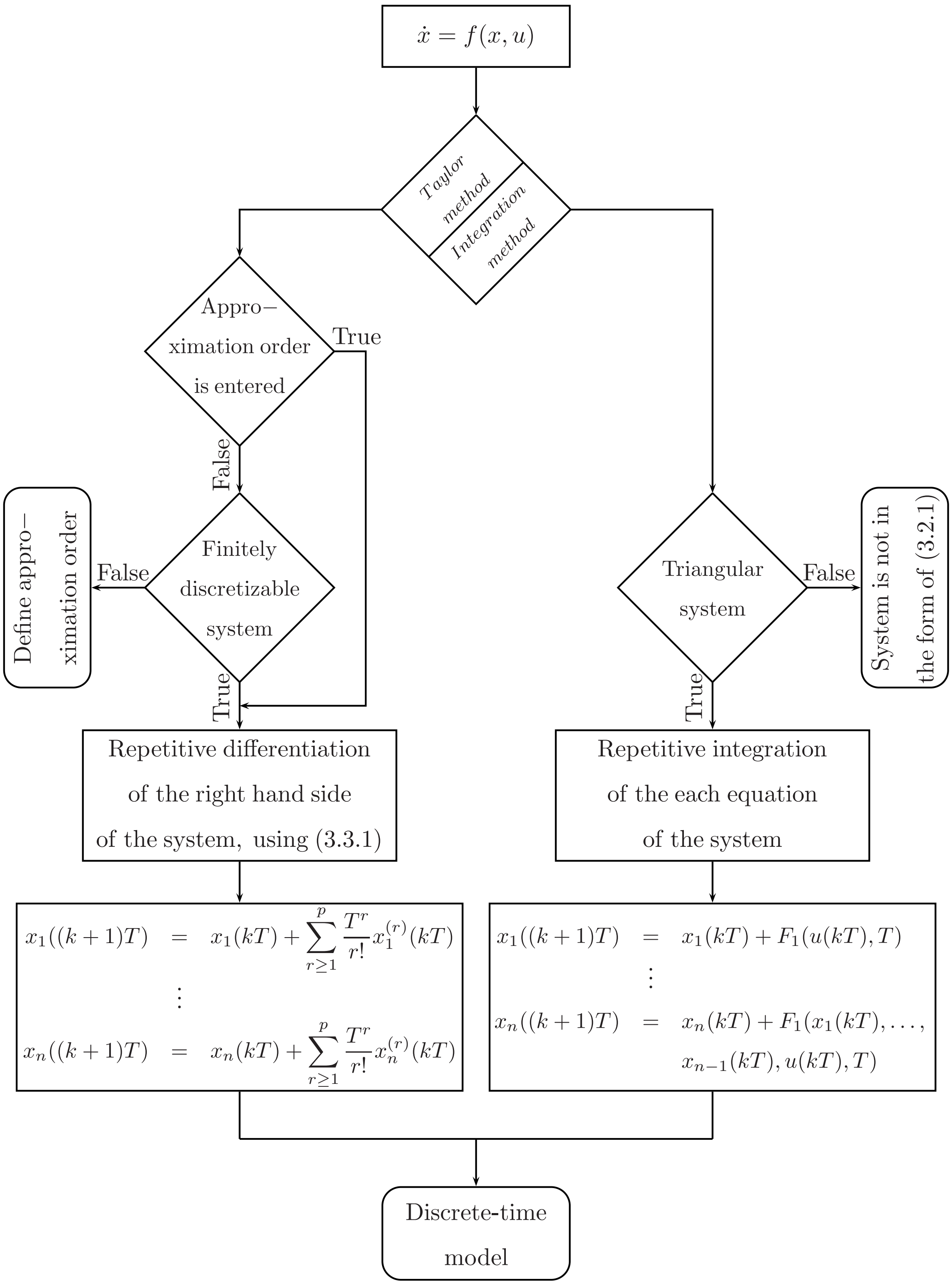

$\dot{x} = f(x,u)$
Taylor method
Integration method
Appro– ximation order is entered
True
False
Finitely discretizable system
False
Define appro– ximation order
True
Triangular system
False
System is not in the form of (3.2.1)
True
Repetitive differentiation of the right hand side of the system, using (3.3.1)
Repetitive integration of the each equation of the system
$x_1((k+1)T) = x_1(kT) + \sum_{r\geq1}^{p} \frac{T^r}{r!} x_1^{(r)}(kT)$
$\vdots$
$x_n((k+1)T) = x_n(kT) + \sum_{r\geq1}^{p} \frac{T^r}{r!} x_n^{(r)}(kT)$
$x_1((k+1)T) = x_1(kT) + F_1(u(kT),T)$
$\vdots$
$x_n((k+1)T) = x_n(kT) + F_1(x_1(kT),\ldots, x_{n-1}(kT),u(kT),T)$
Discrete-time model

Figure 4.1: The block-diagram of the function `Discretization`

how many times the right hand side of a system should be differentiated, when `Taylor` method is used. Its default value is r, which means that if the user did not assign an appropriate numeric value to this argument, then the function tries to compute the exact discrete-time model.

The practical application of the function `Discretization` is demonstrated by the following examples.

Example 4.2.1 Consider the system [24], which describes the dynamics of fish population

$$\dot{x} = Kx(M - x) - u. \tag{4.2.1}$$

Here x is a measure of the size of the fisheries resource and u denotes the harvesting rate.

In order to display the system (4.2.1) in the standard form defined by *Mathematica*, it has to be entered as follows

```
In[2]:= f = {K*x[t]*(M - x[t]) - u[t]};
        Xt = {x[t]};
        Ut = {u[t]};
        contSys = StateSpace[f,Xt,Ut,t,TimeDerivative];
        BookForm[contSys]
```

Then, the output is

```
Out[6]= x′ = -u+K(M-x)x
```

Next, we call the function `Discretization`, using `Taylor` method, with respect to the second order discrete-time model.

```
In[7]:= Discretization[contSys,PrintInfo->True,Method->Taylor,
         ApproximationOrder->2]
```

$$x(k+1) = x(k) + -u(k) + K\,M\,x(k) - K\,x(k)^2 + \frac{1}{2}(-K\,M\,u(k) + K^2 M^2 x(k) + 2K\,u(k)\,x(k) - 3K^2 M\,x(k)^2 + 2K^2 M\,x(k)^3)$$

Out[7]= $\{x[k] + -u[k] + K\,M\,x[k] - K\,x[k]^2 + \frac{1}{2}(-K\,M\,u[k] + K^2 M^2 x[k] + 2K\,u[k]\,x[k] - 3K^2 M\,x[k]^2 + 2K^2 M\,x[k]^3)\}$

Then, the second order discrete-time model is described by the previous equation.

The example below is the kinematic model of the mobile robot.

Example 4.2.2 Consider the following system [9]

$$\begin{aligned}\dot{x}_1 &= u_1 \cos x_3 \\ \dot{x}_2 &= u_1 \sin x_3 \ , \\ \dot{x}_3 &= u_2\end{aligned} \tag{4.2.2}$$

where x_1, x_2 are the vehicle coordinates, and x_3 is the orientation angle of the robot. The input signal u_1 represents the linear velocity of the robot and u_2 its angular velocity.

Next, we enter the system (4.2.2) as follows.

```
In[2]:= f = {u1[t]*Cos[x3[t]],u1[t]*Sin[x3[t]],u2[t]};
        Xt = {x1[t],x2[t],x3[t]};
        Ut = {u1[t],u2[t]};
        contSys = StateSpace[f,Xt,Ut,t,TimeDerivative];
```

After that, we call the function `Discretization`, using `Taylor` method, with respect to the second order discrete-time model.

```
In[6]:= Discretization[contSys,Method->Taylor,
         ApproximationOrder->2]
Out[6]= {x1[kT]+TCos[x3[kT]]u1[kT] - T^2/2 Sin[x3[kT]]u1[kT]u2[kT],
         x2[kT]+TSin[x3[kT]]u1[kT] + T^2/2 Cos[x3[kT]]u1[kT]u2[kT],
         x3[kT]+Tu2[kT]}
```

Then, the second order discrete-time model of the continuous-time system (4.2.2) is described by the previous set of equations.

As we can see, the equations (4.2.2) are not in the form (3.2.1), but after permuting the state and control coordinates as follows $\tilde{x}_1 = x_3$, $\tilde{x}_2 = x_1$, $\tilde{x}_3 = x_2$, $\tilde{u}_1 = u_2$ and $\tilde{u}_2 = u_1$, the system accommodates into the form of (3.2.1), except that the functions in (3.2.1) are not polynomials.

$$\begin{aligned}\tilde{x}_1' &= \tilde{u}_1 \\ \tilde{x}_2' &= \tilde{u}_2 \cos \tilde{x}_1 \\ \tilde{x}_3' &= \tilde{u}_2 \sin \tilde{x}_1\end{aligned} \tag{4.2.3}$$

In spite of trigonometric functions in (4.2.3), we can still apply the direct integration method.

```
In[7]:= f2 = {ũ1[t],ũ2[t]*Cos[x̃1[t]],ũ2[t]*Sin[x̃1[t]]};
        Xt2 = {x̃1[t],x̃2[t],x̃3[t]};
        Ut2 = {ũ1[t],ũ2[t]};
        contSys2 = StateSpace[f2,Xt2,Ut2,t,TimeDerivative];
```

Next, we call the function `Discretization`, using `Triangular` method.

```
In[11]:= discrSys2 = Discretization[contSys2,Method -> Triangular];
         BookForm[discrSys2,{x̃1[k*T],x̃2[k*T],x̃3[k*T],
          {ũ1[k*T],ũ2[k*T]},k]
```

Out[12]=

$$\begin{aligned}\tilde{x}_1[T+kT] &= \tilde{x}_1[kT] + T\,\tilde{u}_1[kT] \\ \tilde{x}_2[T+kT] &= \tilde{x}_2[kT] + \frac{2\,\mathrm{Cos}\left[\frac{1}{2}T\,\tilde{u}_1[kT]\left(\frac{2\tilde{x}_1[kT]}{T\,\tilde{u}_1[kT]}+1\right)\right]\mathrm{Sin}\left[\frac{1}{2}T\,\tilde{u}_1[kT]\right]\tilde{u}_2[kT]}{\tilde{u}_1[kT]} \\ \tilde{x}_3[T+kT] &= \tilde{x}_3[kT] + \frac{2\,\mathrm{Sin}\left[\frac{1}{2}T\,\tilde{u}_1[kT]\left(\frac{2\tilde{x}_1[kT]}{T\,\tilde{u}_1[kT]}+1\right)\right]\mathrm{Sin}\left[\frac{1}{2}T\,\tilde{u}_1[kT]\right]\tilde{u}_2[kT]}{\tilde{u}_1[kT]}\end{aligned}$$

After permuting back the original state and control coordinates, we obtain the exact discrete-time model of the continuous-time system (4.2.2).

4.3 DiscretizationPlot

4.3.1 Description of the function

The function `DiscretizationPlot` was created, since finding out of the quality of obtained discrete-time model is extremely important task for its further use. Obviously, it is more easier to control a model with more simple structure, but at the same time one should not forget about the accuracy of obtained model. Therefore, using visual information about its behavior during some period of time, one can change the model and try to find an equilibrium point between quality and complexity. The function returns figure or figures with drawn changes of states of continuous-time system and discrete-time model during some certain period of time. It has six necessary and one optional arguments. The block-diagram of the function `DiscretizationPlot` is depicted in Figure 4.2. When called, it performs the following steps:

- **Step 1**. The function `DiscretizationPlot` initializes the basic variables and on the basis of this information executes the following steps.
- **Step 2**. The function uses numerical methods (*Mathematica* automatically decides which method or their combination is better to use, relatively to the form of the input system) in order to find the solution of the continuous-time system and collect obtained data.
- **Step 3**. The function collects data from the discrete-time model running time interval from t_{min} up to t_{max}, with respect to the sampling time T.
- **Step 4**. On the basis of collected data `DiscretizationPlot` plots all states of the continuous- and discrete-time systems in one figure or in pairs in different figures.

4.3.2 Application

`DiscretizationPlot` has six necessary and one optional arguments:

```
DiscretizationPlot[contSystem, discrSystem,
initialConditions, controlSignal, timeInterval,
samplingTime, AllInOne].
```

The *`contSystem`* (continuous-time system) and *`discrSystem`* (discrete-time system) are necessary arguments of the function and have to be given by the state equations. In order to get started for the third and the fourth arguments have to be assigned appropriate initial conditions and values of all states of the system and of all control signals,

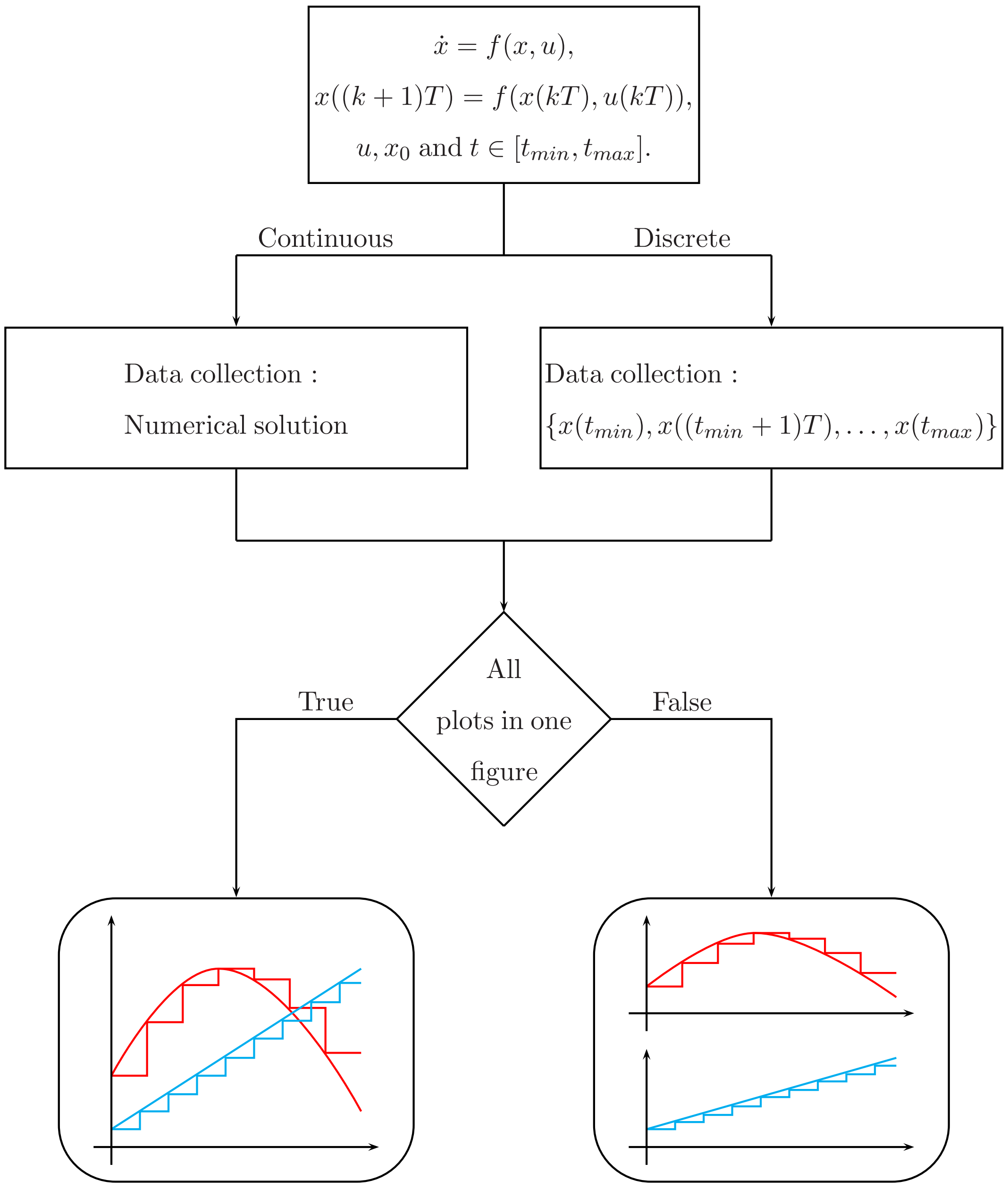

$\dot{x} = f(x, u)$,
$x((k+1)T) = f(x(kT), u(kT))$,
u, x_0 and $t \in [t_{min}, t_{max}]$.
Continuous
Discrete
Data collection :
Numerical solution
Data collection :
$\{x(t_{min}), x((t_{min}+1)T), \ldots, x(t_{max})\}$
True
All plots in one figure
False

Figure 4.2: The block-diagram of the function `DiscretizationPlot`

respectively. The fifth argument is a simulation interval of the form $\{t, t_{min}, t_{max}\}$. The last necessary argument *samplingTime* defines the value of the sampling time T. The optional argument is responsible for showing system states together in one figure or in pairs in different figures (by default its value is equal to *True*). Sometimes it is useful to analyze each state of a system separately.

The action of the function `DiscretizationPlot` is demonstrated by applying it to the following example.

Example 4.3.1 Consider the system from Example 4.2.2. In order to verify the quality of obtained discrete-time model, we should enter the continuous-time system

```
In[2]:= f = {u1[t]*Cos[x3[t]],u1[t]*Sin[x3[t]],u2[t]};
        Xt = {x1[t],x2[t],x3[t]};
        Ut = {u1[t],u2[t]};
        contSys = StateSpace[f,Xt,Ut,t,TimeDerivative];
```

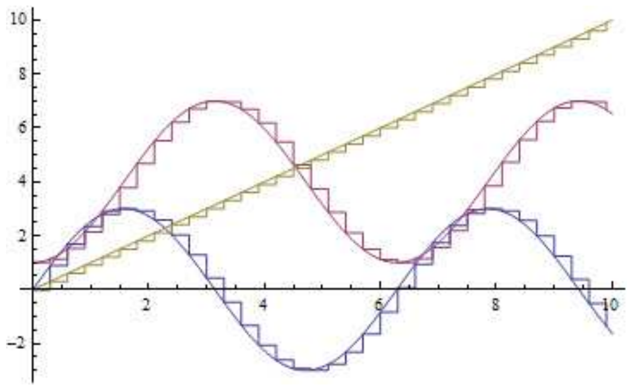
and after that the obtained discrete-time model.

```
In[6]:= f2 = {x1[k*T] - T^2/2 Sin[x3[k*T]]*u1[k*T]*u2[k*T] +
          T*Cos[x3[k*T]]*u1[k*T],x2[k*T] +
          T^2/2 Cos[x3[k*T]]*u1[k*T]*u2[k*T] + T*Sin[x3[k*T]]*u1[k*T],
          x3[k T] + T u2[k T]};
        Xk = {x1[k*T],x2[k*T],x3[k*T]};
        Uk = {u1[k*T],u2[k*T]};
        discrSys = StateSpace[f2,Xk,Uk,k,Shift];
```

Next, we call the function `DiscretizationPlot` with the following arguments.

```
In[10]:= DiscretizationPlot[contSys,discrSys,{0,1,0},{3,1},
          {t,0,10},0.3]
```

Out[10]=



As we can see, the precision of obtained discrete-time model is good enough.

4.4 Dilation

4.4.1 Description of the function

The function `Dilation` finds a dilation of the given continuous-time system. It implements the condition of the Definition 3.4.1 and the equation (3.4.4). The calculation of this function is an important thing in case if you already do not know a dilation and want to check whether the original system is finitely discretizable or not, otherwise you can use the function `HomogeneousDegree`, described in the next subsection. The return value of the current function is a list of the system states. Each of them is multiplied by a time instance in an appropriate power. The function has one necessary and one optional arguments. When called, it performs the following steps:

- **Step 1.** The function `Dilation` initializes the basic variables and on the basis of this information executes the following steps.
- **Step 2.** It calculates the following scalar products $\langle dx_i, X_j\rangle$ and $\langle dx_i, T\delta_t(X_j)\rangle$, where $i = 1, \ldots, n$ and $j = 0, \ldots, m$.
- **Step 3.** Using equation (3.4.4), the function implements the condition of the Definition 3.4.1 to compute integer values $(r_1, r_2, \ldots, r_n)$.
- **Step 4.** `Dilation` returns the result in the following form

$$\{t^{r_1}x_1, t^{r_2}x_2, \ldots, t^{r_n}x_n\}.$$

4.4.2 Application

The function `Dilation` has one necessary and one optional arguments:

`Dilation[`*`contSystem`*`, `*`PrintInfo`*`]`.

The *`contSystem`* (continuous-time system) is a necessary argument of the function and has to be given by the state equations. The optional argument *`PrintInfo`* is a boolean variable with one of the values `True` or `False` (for more information see Section 4.2.2).

The practical application of this function is shown by the following example of the kinematic model of the plate and ball system.

Example 4.4.1 Consider the system [2]

$$\begin{aligned} \dot{x}_1 &= u_1 \\ \dot{x}_2 &= u_2 \\ \dot{x}_3 &= x_1u_2 - x_2u_1 \; . \\ \dot{x}_4 &= x_3u_1 \\ \dot{x}_5 &= x_3u_2 \end{aligned} \tag{4.4.1}$$

After, the system (4.4.1) has been entered

```
In[2]:= f = {u1[t],u2[t],x1[t]*u2[t] - x2[t]*u1[t],x3[t]*u1[t],
          x3[t]*u2[t]};
        Xt = {x1[t],x2[t],x3[t],x4[t],x5[t]};
        Ut = {u1[t],u2[t]};
        contSys = StateSpace[f,Xt,Ut,t,TimeDerivative];
```

the function `Dilation` can be called.

```
In[6]:= Dilation[contSys,PrintInfo -> True]
```

$\delta_t(x_1, x_2, x_3, x_4, x_5) = (t\,x_1, t\,x_2, t^2x_3, t^3x_4, t^3x_5)$

Out[6]= $\{t\,x_1, t\,x_2, t^2x_3, t^3x_4, t^3x_5\}$

Note that it was possible to solve the system (3.4.4) for all r_i, where $i = 1, \ldots, 5$, and therefore the system (4.4.1) is finitely discretizable in original coordinates.

4.5 `HomogeneousDegree`

4.5.1 Description of the function

The function `HomogeneousDegree` calculates homogeneous degrees of the vector fields associated with a continuous-time system with respect to the dilation, according to the Definition 3.4.2. Its calculation is an important step in case if one decided to verify whether the sufficient condition of the Theorem 3.4.1 is satisfied and the considered system is finitely discretizable or not. Of course, for this purpose one can use the function `Dilation`, but `HomogeneousDegree` is preferable, if the dilation of the considered system is already known. The function has two necessary and one optional arguments. When called, it performs the following steps:

- **Step 1.** `HomogeneousDegree` initializes the basic variables and on the basis of this information executes the following steps.
- **Step 2.** The function constructs vector fields of the system and controls whether they are polynomial or not.
- **Step 3.** It calculates the values of s_j with respect to the Definition 3.4.2.
- **Step 4.** The function `HomogeneousDegree` returns the result as a set of homogeneous degrees.

4.5.2 Application

`HomogeneousDegree` has two necessary and one optional arguments:

```
HomogeneousDegree[contSystem, dilation, PrintInfo].
```

The *`contSystem`* is the necessary argument, which defines the continuous-time system of interest given by the state equations. The second argument *`dilation`* is some already known dilation. The optional argument *`PrintInfo`* is a boolean variable with one of the values `True` or `False`. If its value is `True`, then the function returns, besides other, constructed vector fields and the result will be given in the traditional form.

The action of the function `HomogeneousDegree` is demonstrated by applying it to the following example.

Example 4.5.1 Consider the system (3.4.5) from Example 3.4.1 and try to find homogeneous degrees using the function `HomogeneousDegree`. For this purpose we have to enter the system as follows.

```
In[2]:= f = {u₁[t],u₂[t],x₂[t]*u₁[t]};
        Xt = {x₁[t],x₂[t],x₃[t]};
        Ut = {u₁[t],u₂[t]};
        contSys = StateSpace[f,Xt,Ut,t,TimeDerivative];
```

After that, we call the function `HomogeneousDegree` with the following arguments. Although the dilation was obtained, using assumption that homogeneous degrees already equal -1, this example is considered as an illustration of the possibilities of the function.

```
In[6]:= HomogeneousDegree[contSys,{tx₁[t],tx₂[t],t²x₃[t]},
          PrintInfo -> True]
```

Relatively to the dilation $\delta_t(x_1,x_2,x_3) = (t\,x_1, t\,x_2, t^2x_3)$, vector fields: $X_1 = \frac{\partial}{\partial x_1} + \frac{\partial}{\partial x_3}x_2, X_2 = \frac{\partial}{\partial x_2}$ are homogeneous of degrees $-s_1 = -1, -s_2 = -1$, respectively.

```
Out[6]= {-1, -1}
```

As we can see the found values all equal to -1. Then, according to Theorem 3.4.1, it means that the original system is finitely discretizable in original coordinates. This result is identical to obtained in Example 3.4.1.

4.6 Nilpotency

4.6.1 Description of the function

The function `Nilpotency` checks whether the original system is nilpotent or not. This function was programmed on the basis of presented in Section 3.5 theory. Its calculation is an important thing in case if you want to verify whether the original system is nilpotent and as a result is finitely discretizable in some local state coordinates or not. The function returns `True` or `False`. `Nilpotency` has one necessary argument. When called, it performs the following steps:

- **Step 1.** The function `Nilpotency` initializes the basic variables and on the basis of this information executes the following steps.
- **Step 2.** It calculates the iterated Lie brackets until they all become equal to zero or up to the order of the system.
- **Step 3.** `Nilpotency` returns `True` in case if the system is nilpotent and `False` otherwise.

4.6.2 Application

The function `Nilpotency` has one necessary argument:

`Nilpotency[`*`contSystem`*`]`.

It requires the argument *`contSystem`* (continuous-time system) to be given by the state equations.

The practical application of this function is shown by the following example.

Example 4.6.1 Consider the system from Example 4.2.2. In order to verify whether the original system is nilpotent or not we should enter the system as follows.

```
In[2]:= f = {u1[t]*Cos[x3[t]],u1[t]*Sin[x3[t]],u2[t]};
        Xt = {x1[t],x2[t],x3[t]};
        Ut = {u1[t],u2[t]};
        contSys = StateSpace[f,Xt,Ut,t,TimeDerivative];
```

After that, we call the function `Nilpotency`.

```
In[6]:= Nilpotency[contSys]
Out[6]= False
```

Note that during computation process some of Lie brackets do not become equal to zero and as a result the system is not nilpotent. It means that locally there do not exist the state coordinates in which the system (4.4.1) is finitely discretizable.

4.7 Program examples

The purpose of this section is an explanation of additional possibilities of presented above functions by means of examples.

The following simple example illustrates the joint work of almost all programmed functions.

Example 4.7.1 Consider the system [10]

$$\begin{array}{rcl}\dot{x}_1 & = & u \\ \dot{x}_2 & = & \frac{1}{2}x_2x_1^2\end{array}. \tag{4.7.1}$$

First of all we check whether the original system is finitely discretizable or not. Then, the system (4.7.1) should be entered as follows.

```
In[2]:= f = {u[t]], 1/2 x1[t]^2 x2[t]};
        Xt = {x1[t],x2[t]};
        Ut = {u[t]};
        contSys = StateSpace[f,Xt,Ut,t,TimeDerivative];
```

After that, we call the function `Dilation`.

```
In[6]:= Dilation[contSys]
        System is not finitely discretizable in the original
         coordinates
Out[6]= {}
```

Note that it was not possible to solve the system (3.4.4) with respect to the variables r_1, r_2 and therefore the system (4.7.1) is not finitely discretizable in original coordinates. Then, we call the function `Nilpotency`.

```
In[7]:= Nilpotency[contSys]

Out[7]= True
```

It means that the system (4.7.1) is nilpotent, and consequently locally there exist the state coordinates in which it is finitely discretizable. Next we try to calculate those coordinates, calling the function `TransformationToFDF`.

```
In[8]:= transfSys=TransformationToFDF[contSys,{z1[t],z2[t]}];
        BookForm[transfSys]
```

$$\begin{aligned} z_1'[t] &= u[t] \\ z_2'[t] &= \frac{1}{2} z_1[t]^2 \\ z_1[t] &= x_1[t] \\ z_2[t] &= \mathrm{Log}[x_2[t]] \end{aligned} \qquad \text{Out[9]=}$$

The first two rows are equations of the transformed system and last two rows represent the state coordinate transformation. Now one can see that this new system is in the form of (3.2.1) and finitely discretizable. Then, using the function `Discretization` the exact discrete-time model can be obtained.

```
In[10]:= dout=Discretization[transfSys[[1]],Method->Triangular];
         BookForm[DStateSpace[dout,{z1[k*T],z2[k*T]},{u[k*T]},k]]
```

$$\begin{aligned} z_1[T+kT] &= z_1[kT]+Tu[kT] \\ z_2[T+kT] &= z_2[kT]+\frac{1}{2}Tz_1[kT]^2+\frac{1}{2}T^2u[kT]z_1[kT]+\frac{1}{6}T^3u[kT]^2 \end{aligned} \qquad \text{Out[11]=}$$

Example 4.7.2 Consider a single pendulum [4]. Let its mass be m and let the moment of inertia with respect to the pivot point be J. Furthermore let l be the distance from the pivot to the center of mass. The angle between the vertical and the pendulum is θ, where θ is positive in the clockwise direction. The acceleration of gravity is g and the acceleration of the pivot is u. The acceleration u is positive if it is in the direction of the positive x-axis. The equation of motion for the pendulum is

$$J\ddot{\theta} - mgl\sin\theta + mul\cos\theta = 0. \tag{4.7.2}$$

Consider the equation of motion for the inverted pendulum. In this case, after substituting $\theta = 180° - \beta$, (4.7.2) can be rewritten as follows

$$-J\ddot{\beta} - mgl\sin\beta - mul\cos\beta = 0. \tag{4.7.3}$$

Let $x_1 = \beta$, then the state-space representation of the (4.7.3) is

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -\frac{mgl}{J}\sin x_1 - \frac{ml}{J}u\cos x_1 \end{aligned} . \tag{4.7.4}$$

Next, we can find the discrete-time model of the (4.7.4) using functions from the previous sections. Then, we enter the system of state equations.

```
In[2]:= f = {x2[t], - m*g*l/J Sin[x1[t]] - m*l/J u[t]Cos[x1[t]]};
        Xt = {x1[t],x2[t]};
        Ut = {u[t]};
        contSys = StateSpace[f,Xt,Ut,t,TimeDerivative];
```

It is obvious that the system (4.7.4) is not finitely discretizable and there is no possibility to find the exact discrete-time model. Therefore, we call the function `Discretization` using `Taylor` method, with respect to the second order discrete-time model.

```
In[6]:= Discretization[contSys,Method->Taylor,
          ApproximationOrder->2]
Out[6]= {x1[kT] + Tx2[kT] + 1/2 T^2 (- glmSin[x1[t]]/J - lmCos[x1[t]]u[kT]/J),
          x2[kT] + T (- glmSin[x1[t]]/J - lmCos[x1[t]]u[kT]/J) +
          1/2 T^2 (- glmCos[x1[t]]x2[t]/J + lmSin[x1[t]]u[kT]x2[t]/J)}
```

Therefore, the discrete-time model is described by the previous equations. Next, we enter[1] the obtained model.

```
In[7]:= f2 = %;
        Xk = {x1[k*T],x2[k*T]};
        Uk = {u[k*T]};
        discrSys = StateSpace[f2,Xk,Uk,k,Shift];
```

After that we initialize acceleration of gravity (m/s^2), mass (kg), length of the pendulum (m) and moment of inertia $(N * m * s^2)$ with the following values

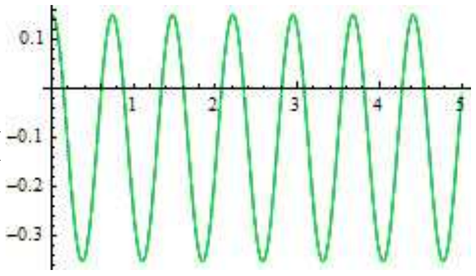
[1]The sign % corresponds to the previous output in the *Mathematica* programming environment.

```
In[11]:= g = 9.8066;
         m = 0.086184;
         l = 0.113;
         J = 0.0013011;
```

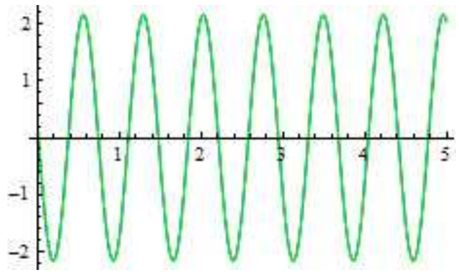
In order to verify the accuracy of obtained model, the function `DiscretizationPlot` should be called.

```
In[15]:= DiscretizationPlot[contSys,discrSys,{0.15,0.1},{1},{t,0,5},
          0.01,AllInOne->False]
```

Out[15]= {



,



}

As we can see from presented above figures, the precision of obtained discrete-time model is good enough and one can use it for the further research or implementation.

Chapter 5

Neural Networks based ANARX Structure for Control of Nonlinear SISO and MIMO Systems

Discrete-time models of nonlinear systems were considered in the previous chapters. In some cases such classical models are not linearizable by dynamic feedback [39]. In order to avoid this obstacle, Artificial Neural Networks can be used to identify a discrete-time model in the form, which is always linearizable.

This chapter briefly describes the identification of nonlinear systems by ANARX models and ANARX models based control. They have several important advantages over classical NARX models. ANARX is a subclass of NARX models and has all time instances separated. Restrictions imposed by this subclass guarantee linearizability by dynamic output feedback as well as state-space representability of the identified model. These advantages are especially important for control applications. That is why this type of the model is a reasonable choice for control of a wide class of nonlinear systems.

The chapter is organized as follows. Section 5.1 contains a brief summary of the theory connected to ANARX structure and establishes the differences from NARX structure. After that in Section 5.2 different control techniques and simulation results of nonlinear plant identification and control using ANARX structure, are presented. Each example is chosen to emphasize a specific point of the theory described above.

5.1 NN-based ANARX Structure

Discrete-time Nonlinear AutoRegressive eXogenous (NARX) models are represented by the high order difference equation

$$y(k) = f(y(k-1), \dots, y(k-n), u(k-1), \dots, u(k-n)).$$

This model can be easily obtained by using a multilayered perceptron. On the one side such structure is capable of modeling a wide class of nonlinear systems with a high precision, but on the other side, from the control system point of view, it has several serious drawbacks. First of them is that in general this structure can not be represented in the classical state-space form. Besides that, there is no possibility to separate different time steps. And finally, this structure is not always linearizable by the dynamic output feedback. Then, due to those facts Additive NARX or shortly ANARX structure was proposed to bridge the gap.

The idea of separating time-instances was firstly proposed in [12]. ANARX model shown in [13], [20] and [21] is a subclass of well known NARX models and has all time instances separated

$$y(k) = f_1(y(k-1), u(k-1)) + f_n(y(k-n), u(k-n)), \qquad (5.1.1)$$

where $f_1(\cdot), \dots, f_n(\cdot)$ are nonlinear functions, $y(k) = (y_1(k), \dots, y_m(k))^T \in \mathbb{R}^m$ is a vector of system outputs and $u(k) = (u_1(k), \dots, u_r(k))^T \in \mathbb{R}^r$ is a vector of system inputs.

Comparing with NARX structure, ANARX model can always be rewritten in the classical state-space form as follows

$$\begin{array}{rcl}
x_1(k+1) & = & x_2(k) + f_1(x_1(k), u(k)) \\
x_2(k+1) & = & x_3(k) + f_2(x_1(k), u(k)) \\
& \vdots & \\
x_{n-1}(k+1) & = & x_n(k) + f_{n-1}(x_1(k), u(k)) \\
x_n(k+1) & = & f_n(x_1(k), u(k)) \\
y(k) & = & x_1(k)
\end{array} .$$

State-space representation is an important property of the control system, which provides a convenient and compact way for its further modeling and analyzing.

Another advantage of ANARX model (5.1.1) is that it can always be linearized by the following dynamic output feedback [39]

$$u(k) = F^{-1}(y(k), \eta_1(k)),$$

where

$$F(y(k), u(k)) = \eta_1(k), \tag{5.1.2}$$

and the dynamics of the feedback linearization are described by the following equations

$$\begin{array}{rcl} \eta_1(k+1) & = & \eta_2(k) - f_2(y(k), u(k)) \\ \eta_2(k+1) & = & \eta_3(k) - f_3(y(k), u(k)) \\ & \vdots & \\ \eta_{n-2}(k+1) & = & \eta_{n-1}(k) - f_{n-1}(y(k), u(k)) \\ \eta_{n-1}(k+1) & = & v(k) - f_n(y(k), u(k)) \end{array} . \tag{5.1.3}$$

Here $v(k)$ is a vector of desired outputs of the system (reference signals).

ANARX structure is very well suited for the representation by using a neural network of the specific structure proposed in [13]. The structure of such neural network is shown in Fig. 5.1.

It can be seen from the figure that the hidden layer consists of n sublayers corresponding to the n-th order of the model. This model is called Neural Networks based ANARX or shortly NN-based ANARX and can be expressed by the following difference equation

$$y(k) = \sum_{i=1}^{n} C_i \phi_i \left(W_i \cdot [y(k-i), u(k-i)]^T \right), \tag{5.1.4}$$

where $\phi_i(\cdot)$ is an activation function of i-th sublayer neurons, C_i is a $m \times l_i$ dimensional matrix of i-th sublayer output synaptic weights and W_i is a $l_i \times (r+m)$ dimensional matrix of i-th sublayer input synaptic weights. Here l_i is the number of hidden neurons in i-th sublayer.

If ANARX model was obtained in the form of neural network (5.1.4), then equations (5.1.2) and (5.1.3) can be rewritten by using parameters of the neural network [21] as follows

$$F := C_1 \phi_1 \left(W_1 \cdot [y(k), u(k)]^T \right) = \eta_1(k) \tag{5.1.5}$$

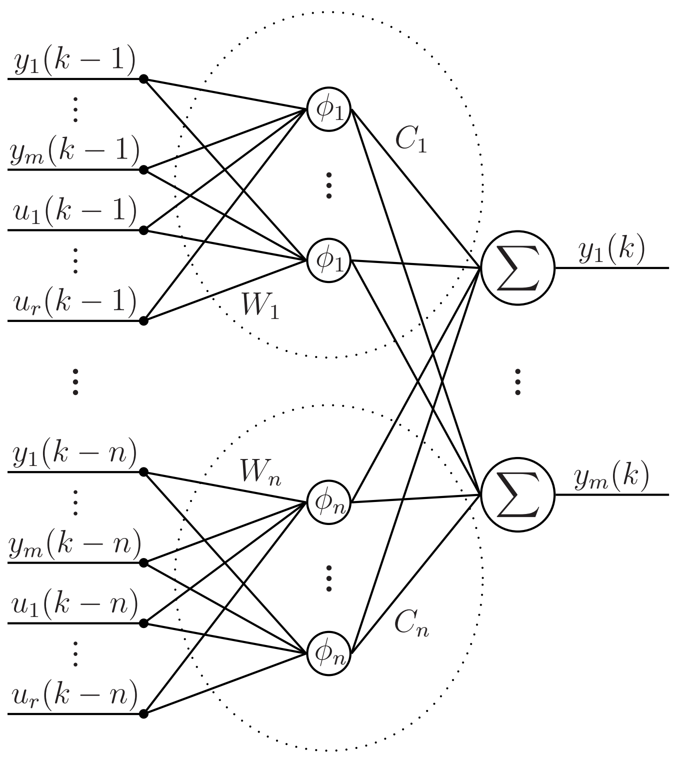


Figure 5.1: Structure of neural network representing MIMO NN-based ANARX model

and

$$\begin{array}{rcl}\eta_1(k+1) & = & \eta_2(k) - C_2\phi_2\left(W_2 \cdot [y(k),u(k)]^T\right)\\ \eta_2(k+1) & = & \eta_3(k) - C_3\phi_3\left(W_3 \cdot [y(k),u(k)]^T\right)\\ & \vdots & \\ \eta_{n-2}(k+1) & = & \eta_{n-1}(k) - C_{n-1}\phi_{n-1}\left(W_{n-1} \cdot [y(k),u(k)]^T\right)\\ \eta_{n-1}(k+1) & = & v(k) - C_n\phi_n\left(W_n \cdot [y(k),u(k)]^T\right)\end{array}. \qquad (5.1.6)$$

The structure of the corresponding control system is depicted in Fig. 5.2.

The theory presented above shows that it is attractive to use NN-based ANARX structure in different real life or academical applications. However, in this case there is a one serious obstacle, namely calculation of the control signals $u(k)$ from (5.1.5). It means that inverse function $F^{-1}(y(k),\eta_1(k))$ of the first sublayer (5.1.5) has to be found, in order to make possible using of dynamic output feedback linearization algorithm (5.1.5) and (5.1.6). A big research was made in this area during last three years and different solutions were offered. Each of proposed techniques has a number of advantages and disadvantages and can be applied in different situations for solving tasks of different complexity.

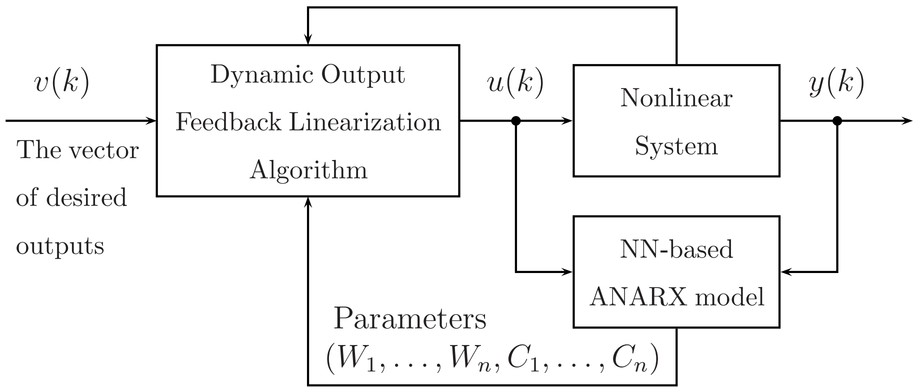


Figure 5.2: Structure of the control system

5.2 Calculation of the Control Signal in Neural Networks based ANARX Models

A number of advantages of the NN-based ANARX structure and reasons why it is more preferable as a control technique were considered in the previous section. However, it was also mentioned that such structure has a one serious drawback, namely calculation of the control signal. Therefore, this section discusses the basic results and solutions of this problem achieved by the author and/or his colleagues during last years. All methods presented below are arranged in the order of their origin.

5.2.1 Newton's method

Newton's method, also called Newton-Raphson method, can be used for solving the equation or system of equations (5.1.5) with respect to the variable(s) $u(k)$ by the methods of the calculus. The practical application of this approach was shown in [37], [38] and [6], where it was applied mainly to SISO systems. Such disproportion can be explained by the fact that numerical solution of (5.1.5), in case of MIMO systems, becomes an extremely difficult task, since the number of variables and equations increase. Besides that practice has shown that the convergence speed of a such classical numerical algorithm may be one of the main obstacles for the calculation of the control signals.

Example 5.2.1 This example gives a brief overview of the results, announced in [6]. For the convenience of the reader we omit some additional details and repeat only the relevant material, thus if someone wants more deeply to familiarize with the control algorithm, then all necessary information can be found in [6].

Nonlinearity and instability make the backing up control of a truck-trailer a difficult task. Existing solutions of a truck-trailer backing up problem are mainly based on employing neural networks [32], fuzzy [40], [43], or neuro-fuzzy [17], [19] controllers and practically nothing is available for more classical control methods. On the one side such disproportion can be explained by the fact that fuzzy and neural controllers very well suite for such tasks. On the other side in many cases models of the truck-trailer are developed from the physical point of view and usually are unsuitable for application of such methods like feedback linearization. Of course, one can linearize the model around a number of operating points, but as it was mentioned in [32] such approach can be computationally complicated and requires considerable design effort.

Neural networks based modeling and such a classical technique like dynamic output feedback linearization were combined in [6] with a purpose to control the backing up truck-trailer. Backward motion of a truck-trailer is modeled by NN-based ANARX model and after that the linearization (5.1.5) and (5.1.6) is applied as a control technique.

Consider the problem of backing up control of the truck-trailer. The problem is to control the steering angle in order to track the desired trajectory of the truck-trailer from any initial position. Following equations were proposed by [17] to model the dynamics of the truck-trailer

$$\begin{array}{rcl} x_1(k+1) & = & x_1(k) + v \cdot \frac{T}{l} \cdot \tan[u(k)] \\ x_2(k+1) & = & x_2(k) + v \cdot \frac{T}{L} \cdot \sin[x_5(k)] \\ x_3(k+1) & = & x_3(k) + v \cdot T \cdot \cos[x_5(k)] \cdot \sin\left[\frac{x_2(k+1)+x_2(k)}{2}\right] \\ x_4(k+1) & = & x_4(k) + v \cdot T \cdot \cos[x_5(k)] \cdot \cos\left[\frac{x_2(k+1)+x_2(k)}{2}\right] \\ x_5(k) & = & x_1(k) - x_2(k) \end{array}, \tag{5.2.1}$$

where $x_1(k)$ is the angle of truck, $x_2(k)$ is the angle of trailer, ($x_4(k)$, $x_3(k)$) is horizontal and vertical positions of the rear end of trailer, respectively, $x_5(k)$ is the angle between truck and trailer, $u(k)$ is the steering angle, l is the length of the truck, L is the length of the trailer, T is the sampling time and v is the constant speed of backing up. Fig. 5.3 shows the model of the truck-trailer.

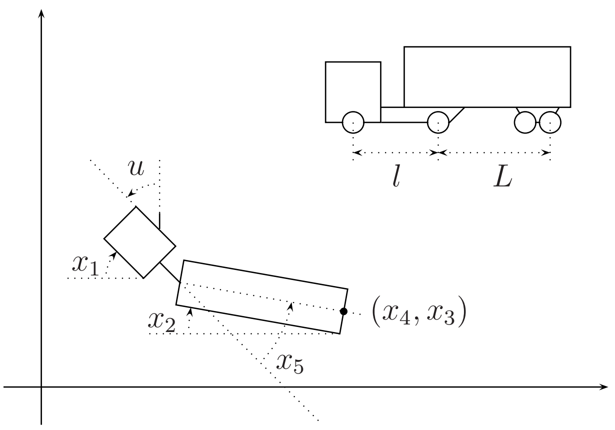


Figure 5.3: Truck-trailer and main parameters of its model

The following parameters of the truck-trailer were used for identification and making experiments:

- $l = 2.8\,(\mathrm{m})$
- $T = 1.0\,(\mathrm{s})$
- $L = 5.5\,(\mathrm{m})$
- $v = -1.0\,(\mathrm{m/s})$

While the system (5.2.1) describes the behavior of the truck-trailer with high level of precision, it is shown in [29] that such system is not linearizable by dynamic output feedback and as a result could not be used directly for the feedback design. Thus, NN-based ANARX structure was chosen as an alternative to model the system (5.2.1).

In order to obtain an input-output data for identification, the plant (5.2.1) was simulated with Uniform Random Number signal $u(k) \in [-\pi/4, \pi/4]$. Steering angle was used as the input of the model and the vertical position of the rear end of the trailer $x_3(k)$ was considered as the output of the model. Then, the neural network based ANARX structure with three sublayers, with respect to the third order model, was trained by the *Levenberg-Marquardt* algorithm. Three neurons and logarithmic sigmoid activation functions were used in each sublayer. After 5000 training epochs the mean square error (MSE) was approximately 0.001. The identified model is represented by the following equation

$$y(k) = \sum_{i=1}^{3} \frac{C_i}{1 + \mathrm{e}^{-W_i \cdot [y(k-i), u(k-i)]^T}}, \tag{5.2.2}$$

where C_i and W_i are matrices of synaptic weights (identified parameters of the model) and $y(k) = x_3(k)$.

After the system (5.2.1) was identified by NN-based ANARX model (5.2.2), the linearization algorithm (5.1.5) and (5.1.6) was applied. Newton's method was used for numerical calculation of the control signal $u(k)$ from (5.1.5). Notice that the aim of the control algorithm is to keep the angle between truck and trailer $x_5(k)$ as close to 0 as possible.

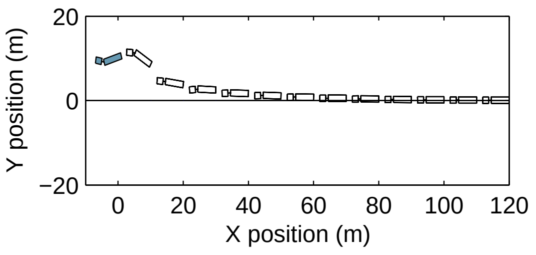
In order to see simulation results of behavior of the truck-trailer, a number of experiments of constructed system have been made. Different initial and desired positions of the truck-trailer were chosen. Table 5.1 presents state parameters.

Table 5.1: Initial and desired parameters of the truck-trailer.

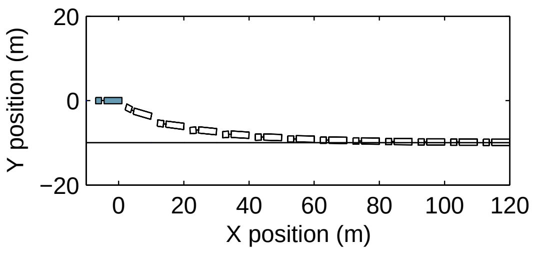
	Initial position			Desired position $v(k)$
Number of experiment	$x_1(0)$	$x_2(0)$	$y(0) = x_3(0)$	$y(k) \to v(k)$
1	$10\,^\circ$	$-20\,^\circ$	$10\,\mathrm{m}$	$0\,\mathrm{m}$
2	$0\,^\circ$	$0\,^\circ$	$0\,\mathrm{m}$	$-10\,\mathrm{m}$
3	$-20\,^\circ$	$45\,^\circ$	$0\,\mathrm{m}$	$5\,\mathrm{m}$
4	$-180\,^\circ$	$-180\,^\circ$	$0\,\mathrm{m}$	$20\sin(0.03t)\,\mathrm{m}$
5	$0\,^\circ$	$0\,^\circ$	$0\,\mathrm{m}$	$55\,\mathrm{m}$

Experiments 1-4 show the movement of the truck-trailer to the desired line. Experiment 5 is harder, in this case the desired trajectory is described by sinusoidal signal. The following Figs. 5.4(a)-5.4(e) show simulation results for each experiment.

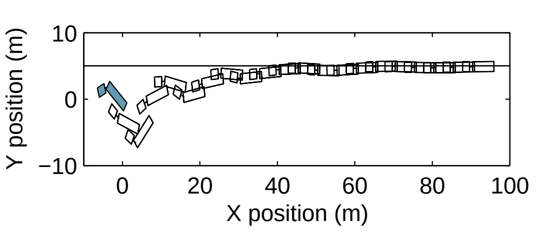
The application of NN-based ANARX structure for identification of the truck-trailer system was shown above. On the one side such approach benefits from high precision of NN-based models and on the other side employment of such structure makes it possible to apply such classical technique like dynamic output feedback linearization. The con-



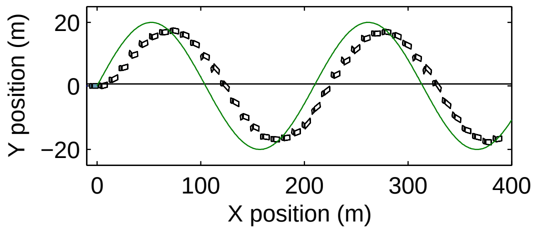
(a) Experiment 1



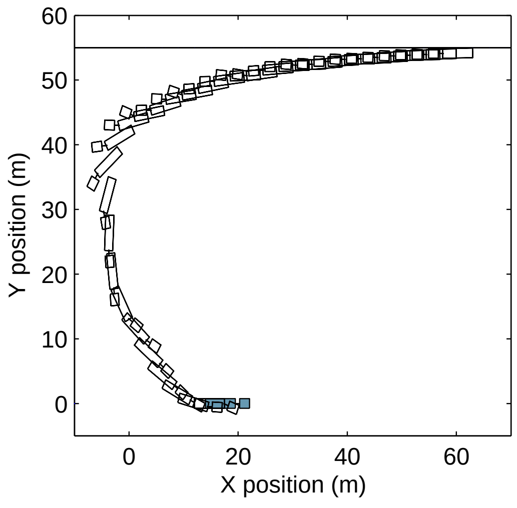
(b) Experiment 2



(c) Experiment 3



(d) Experiment 4



(e) Experiment 5

Figure 5.4: Simulation results of the controlled system (5.2.1)

trol algorithm does not use the model entirely, but only its identified parameters for the calculation of the control signal and dynamics of the controller.

The model of the truck-trailer (5.2.1) imposes a number of physical restrictions such as $u(k) \in [-\pi/4, \pi/4]$ and $x_5(k) \in [-\pi/2, \pi/2]$. Under such conditions it is quite difficult to provide a good enough identification and obtain a model with high degree of accuracy. Another problem is expressed in terms that these restrictions should be included in the control algorithm for the calculation of the control signal. Therefore, all these reasons make this approach a very complex for the solution of this problem. For more details we refer the reader to [6].

5.2.2 NN-based Simplified ANARX Structure

Additionally to Newton's method an alternative technique was proposed in [34]. The problem of calculation of the inverse function $F^{-1}(y(k), \eta_1(k))$ in (5.1.5) can be solved by imposing an additional restriction on NN-based ANARX structure and introducing a new subclass of ANARX models, so called NN-based Simplified ANARX structure or shortly NN-based SANARX, where the first sublayer is linear. It means that the function $\phi_1(\cdot)$ is a linear transfer function. In this case, it follows from equation (5.1.4) that NN-based SANARX model can be described by the following formula

$$y(k) = C_1 \cdot W_1 \cdot [y(k-1), u(k-1)]^T + \sum_{i=2}^{n} C_i \phi_i \left(W_i \cdot [y(k-i), u(k-i)]^T \right). \quad (5.2.3)$$

Such NN-based SANARX models belong to the class of ANARX models. Therefore, ANARX based dynamic output feedback linearization algorithm can be applied to control of the systems identified by NN-based SANARX structure. It means that equations (5.1.5) and (5.1.6) can be rewritten as follows

$$C_1 \cdot W_1 \cdot [y(k), u(k)]^T = \eta_1(k) \quad (5.2.4)$$

and

$$\begin{array}{rcl}\eta_1(k+1) & = & \eta_2(k) - C_2\phi_2\left(W_2 \cdot [y(k), u(k)]^T\right) \\ \eta_2(k+1) & = & \eta_3(k) - C_3\phi_3\left(W_3 \cdot [y(k), u(k)]^T\right) \\ & \vdots & \\ \eta_{n-2}(k+1) & = & \eta_{n-1}(k) - C_{n-1}\phi_{n-1}\left(W_{n-1} \cdot [y(k), u(k)]^T\right) \\ \eta_{n-1}(k+1) & = & v(k) - C_n\phi_n\left(W_n \cdot [y(k), u(k)]^T\right)\end{array} . \tag{5.2.5}$$

Such restriction, imposed by NN-based SANARX structure, guarantees that the control signals $u(k)$ can be easily calculated from a system of linear equations, by using the following expression

$$u(k) = T_2^{-1}(\eta_1(k) - T_1 \cdot y(k)), \tag{5.2.6}$$

where $T = C_1 \cdot W_1$ and $T = [T_1 \;\; T_2]$ and T_2 is a nonsingular square matrix.

It has to be mentioned that in case of SISO systems $T \in \mathbb{R}^2$ is a 2×1 vector and as a result $T_1, T_2 \in \mathbb{R}$ are real numbers. The author refers the reader to [34] for additional details.

The following examples show the application of the control technique described above.

Example 5.2.2 The second order discrete-time system of a Heat Exchanger [8], [37] is described by the following input-output equation

$$\begin{array}{rcl}y(k+2) & = & 2.301 + 0.9173y(k+1) + 0.449u(k+1) + \\ & + & 0.04577u(k) - 0.01889y^2(k+1) - 0.00999u^2(k+1) - \\ & - & 0.002099y^2(k+1)u(k+1) - 0.002434u^3(k+1)\end{array} \tag{5.2.7}$$

and presented in Figure 5.5.

The system (5.2.7) was simulated and the obtained set of the input-output data was used for training of SISO NN-based Simplified ANARX structure with *Levenberg-Marquardt* algorithm and modeling (5.2.7). The network shown in Fig. 5.1 was trained with two sublayers corresponding to the second order of the model. The pure linear activation function was chosen on the first sublayer, with respect to the SANARX structure, and the logarithmic sigmoid activation function on the second sublayer. Identified parameters of

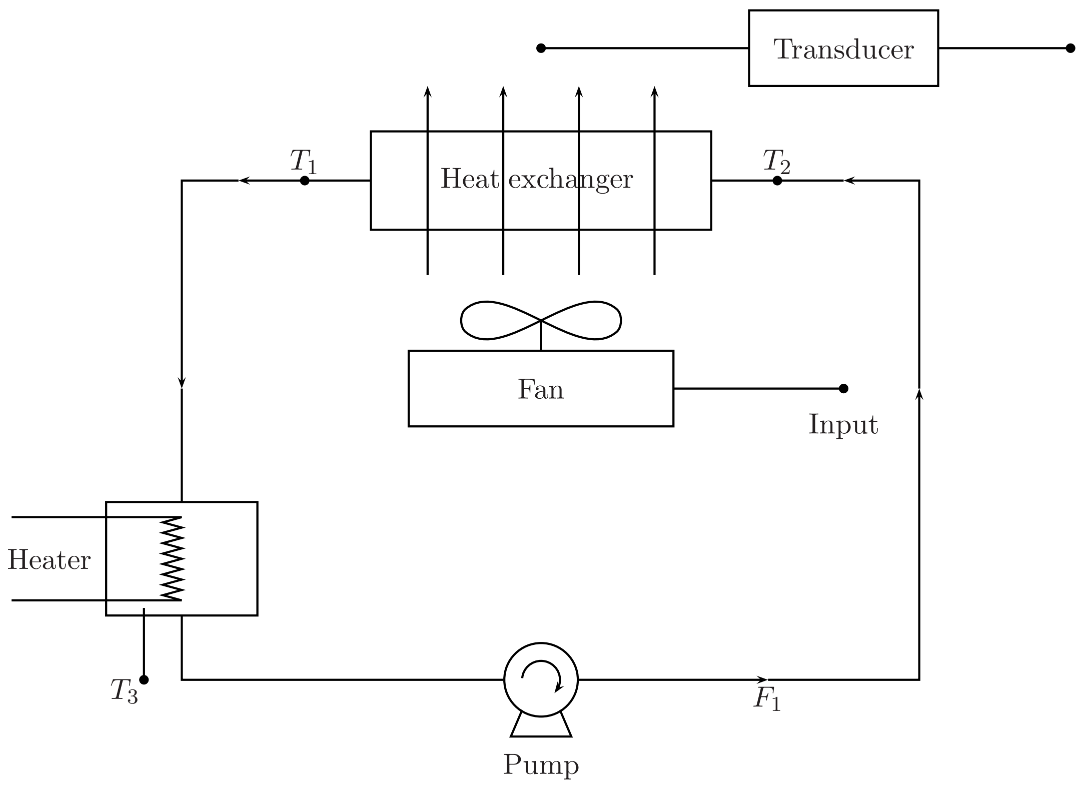


Figure 5.5: The Heat Exchanger

the model have the following values.

$$W_1 = \begin{bmatrix} 0.7359 & -0.0026 \\ 0.8173 & 0.8089 \\ 0.0971 & 0.6378 \end{bmatrix},$$

$$W_2 = \begin{bmatrix} 73.0334 & 241.0942 \\ -14.7492 & -0.5395 \\ -2.3186 & -0.6359 \end{bmatrix},$$

$$C_1 = \begin{bmatrix} 0.2401 & 0.6350 & -0.3261 \end{bmatrix},$$

$$C_2 = \begin{bmatrix} 3.0404 & -161.2081 & 161.3956 \end{bmatrix}.$$

Using identified parameters and equation (5.2.6), the control signal $u(k)$ for the nonlinear system (5.2.7) can be calculated as follows

$$u(k) = 3.2779\eta_1(k) - 2.1765y(k).$$

The dynamic output feedback linearization algorithm (5.2.4) and (5.2.5) was applied to control the system (5.2.7) and the piecewise constant reference signal $v(k)$ was used for its testing. The quality of tracking algorithm is depicted in Fig. 5.6.

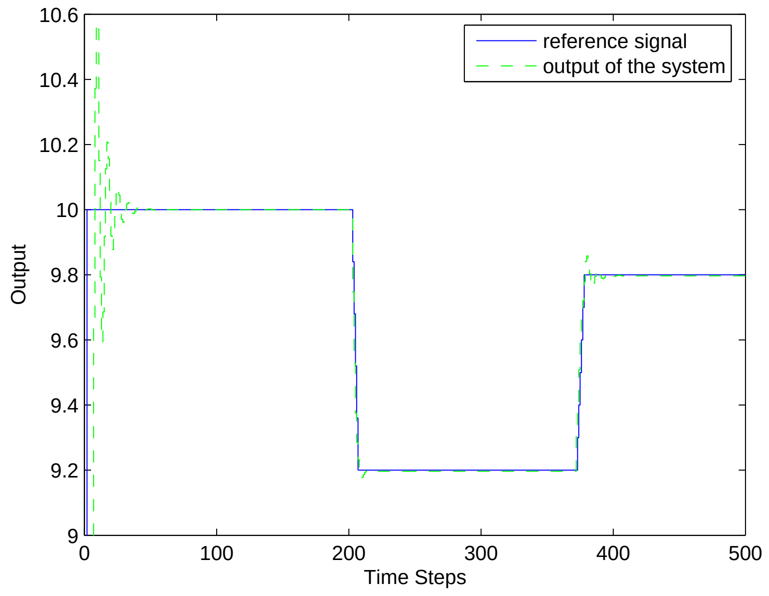


Figure 5.6: Closed loop simulation with piecewise constant reference signal

It can be seen from Fig. 5.6 that the control system is capable of tracking the reference signal $v(k)$, when the NN-based SANARX structure is used and control technique described above applied.

Example 5.2.3 The nonlinear SISO continuous-time system [11] is described by the following state equations

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -\sin x_1 + 0.01u - 0.5x_2 \,. \\ y &= x_1 + x_2 \end{aligned} \tag{5.2.8}$$

Next, we find the discrete-time model of the system (5.2.8), and for this purpose we use the function `Discretization`. Then, the system (5.2.8) can be entered in the following way

```
In[2]:= f = {x₂[t], -Sin[x₁[t]] - 0.5*x₂[t] + 0.01*u[t]};
        Xt = {x₁[t],x₂[t]};
```

```
Ut = {u[t]};
contSys = StateSpace[f,Xt,Ut,t,TimeDerivative];
```

After that, we call the function `Discretization` with the following arguments.

```
In[6]:= Discretization[contSys,PrintInfo->True,Method->Taylor,
         ApproximationOrder->2]
```

$$x_1(T+kT) = x_1(kT)+Tx_2(kT)+\frac{T^2}{2}\left(-\sin(x_1(kT))+\frac{1}{100}u(kT)-\frac{1}{2}x_2(kT)\right)$$

$$x_2(T+kT) = x_2(kT)+T\left(-\sin(x_1(kT))+\frac{1}{100}u(kT)-\frac{1}{2}x_2(kT)\right)+\frac{T^2}{2}\left(\frac{1}{2}\sin(x_1(kT))-\frac{1}{200}u(kT)-\cos(x_1(kT))x_2(kT)+\frac{1}{4}x_2(kT)\right)$$

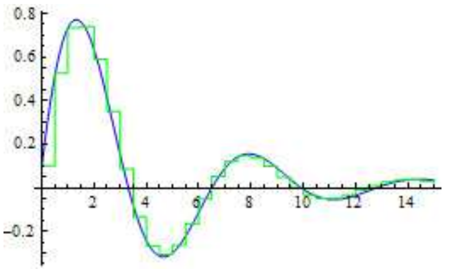
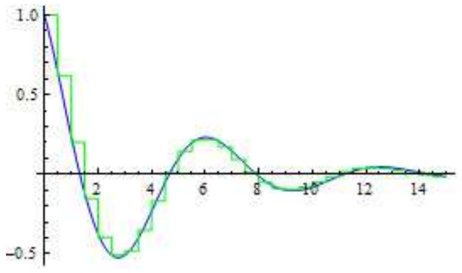
Out[6]=
$$\left\{x_1[kT]+T x_2[kT]+\frac{T^2}{2}\left(-\text{Sin}[x_1[kT]]+\frac{1}{100}u[kT]-\frac{1}{2}x_2[kT]\right),\ x_2[kT]+T\left(-\text{Sin}[x_1[kT]]]+\frac{1}{100}u[kT]-\frac{1}{2}x_2[kT]\right)+\frac{T^2}{2}\left(\frac{1}{2}\text{Sin}[x_1[kT]]-\frac{1}{200}u[kT]-\text{Cos}[x_1[kT]]]x_2[kT]+\frac{1}{4}x_2[kT]\right)\right\}$$

As a result the discrete-time model is described by the previous equations. Next, we enter the obtained model.

```
In[7]:= f2 = %;
        Xk = {x₁[k*T],x₂[k*T]};
        Uk = {u[k*T]};
        discrSys = StateSpace[f2,Xk,Uk,k,Shift];
```

After that, in order to verify the accuracy, we call the function `DiscretizationPlot`.

```
In[11]:= DiscretizationPlot[contSys,discrSys,{0.1,1},{1},{t,0,15},
          0.5,AllInOne->False]
```

Out[11]= {  ,  }

As we can see from presented above figures, the precision of obtained discrete-time model is good enough and we can use it in the further research.

The discrete-time model was simulated and the obtained set of the input-output data was used for training of SISO NN-based Simplified ANARX structure with *Levenberg-Marquardt* algorithm. The network shown in Fig. 5.1 was trained with two sublayers corresponding to the second order of the model. The pure linear activation function was chosen on the first sublayer, with respect to the SANARX structure, and the logarithmic sigmoid activation function on the second sublayer. Identified parameters of the trained model have the following values.

$$W_1 = \begin{bmatrix} 0.5919 & 0.8234 \\ -0.6428 & 0.5386 \\ 0.4342 & -0.1167 \\ -0.3718 & -0.9170 \end{bmatrix},$$

$$W_2 = \begin{bmatrix} 1.2039 & 0.0047 \\ -1.3031 & -0.0037 \\ -0.5962 & -0.3103 \\ -0.6109 & -0.3104 \\ 1.3798 & -2.3620 \\ 3.4453 & -0.0075 \end{bmatrix},$$

$$C_1 = \begin{bmatrix} -0.6075 & -0.5085 & 0.7072 & -0.9477 \end{bmatrix},$$

$$C_2 = \begin{bmatrix} -1.1632 & 1.0746 & -0.5305 & 0.5304 & 0.0000 & 0.0887 \end{bmatrix}.$$

After that, the control technique proposed in [34] was applied to control the model. The following reference signal $v(k) = \sin(0.1k + 0.5) + 0.3\sin(0.35k + 0.1)$ was used for its testing. The simulation result of is represented in Fig. 5.7.

It can be seen from Fig. 5.7 that the output of the model closely follows the reference signal $v(k)$, when the SANARX structure is used and control technique from [34] applied.

5.2.3 Additional static neural network based approach

Further, we describe another technique as a solution of the problem of control signals calculation from (5.1.5), see [36] for details. Because of well known neural networks approximation capabilities, a neural network can be trained to approximate the function

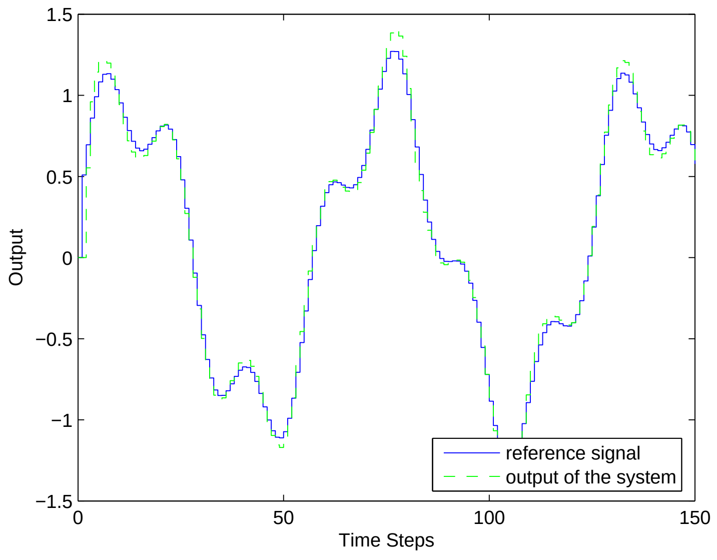


Figure 5.7: Closed loop simulation with sinusoidal reference signal

(5.1.5) and to find the inverse function $F^{-1}(y(k), \eta_1(k))$. This function is a static function of arguments $u(k)$ and $y(k)$ and produces output values $\eta_1(k)$. When ANARX model is obtained, this function can be considered as a separate system and simulated with random inputs to produce a set of input-output data, which can be used as a training set for approximation of the function

$$u(k) = \psi(y(k), \eta_1(k)), \tag{5.2.9}$$

where $\psi(\cdot)$ is a nonlinear map performed by a feed-forward neural network. According to Stone-Weierstrass theorem [16], [42], a two-layer perceptron with nonlinear sigmoid activation functions on its hidden layer is capable of approximating any arbitrary continuous map to within a desired accuracy. It means that function (5.2.9) can be obtained by training a two-layer perceptron

$$u(k) = C_0\phi_0\left(W_0 \cdot [y(k), \eta_1(k)]^T\right), \tag{5.2.10}$$

where W_0 is the matrix of synaptic weights of neurons between input and hidden layers, C_0 is the vector of output layer synaptic weights and $\phi_0(\cdot)$ is a nonlinear sigmoid-type activation function of the hidden layer neurons.

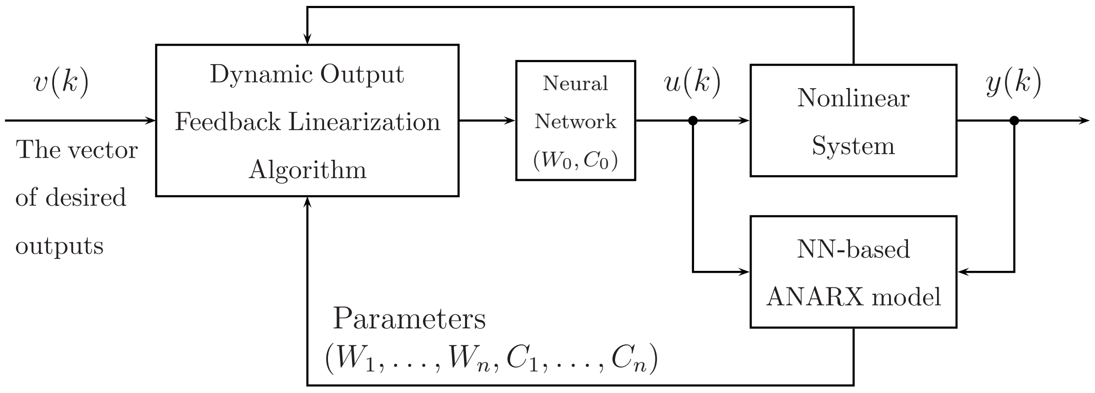
When NN-based ANARX model is used to identify a nonlinear system, the first sublayer

(5.1.5) can be considered as a system for generating a data set for training the neural network (5.2.10).

NN-ANARX based dynamic output feedback linearization control algorithm (5.1.5) and (5.1.6) can now be represented as follows

$$\begin{array}{rcl} u(k) & = & C_0\phi_0\left(W_0\cdot[y(k),\eta_1(k)]^T\right) \\ \eta_1(k+1) & = & \eta_2(k)-C_2\phi_2\left(W_2\cdot[y(k),u(k)]^T\right) \\ \eta_2(k+1) & = & \eta_3(k)-C_3\phi_3\left(W_3\cdot[y(k),u(k)]^T\right) \\ & \vdots & \\ \eta_{n-2}(k+1) & = & \eta_{n-1}(k)-C_{n-1}\phi_{n-1}\left(W_{n-1}\cdot[y(k),u(k)]^T\right) \\ \eta_{n-1}(k+1) & = & v(k)-C_n\phi_n\left(W_n\cdot[y(k),u(k)]^T\right) \end{array} . \qquad (5.2.11)$$

The structure of the corresponding control system is depicted in Fig. 5.8.



$v(k)$
The vector of desired outputs
Dynamic Output Feedback Linearization Algorithm
Neural Network (W_0, C_0)
$u(k)$
Nonlinear System
$y(k)$
NN-based ANARX model
Parameters $(W_1,\ldots,W_n,C_1,\ldots,C_n)$

Figure 5.8: NN-ANARX model based control with additional neural network

The practical application of this approach is shown by the following example.

Example 5.2.4 The nonlinear SISO discrete-time system of a jacketed Continuous Stirred Tank Reactor (CSTR) [33], [37] is described by the following input-output equation

$$\begin{array}{rcl} y(k+2) & = & 0.7653y(k+1)-0.231y(k)+0.4801u(k+1)- \\ & - & 0.6047y^2(k+1)+1.014y(k)y(k+1)-0.3921y^2(k+1)+ \\ & + & 0.592y(k+1)u(k+1)-0.5611y(k)u(k+1) \end{array} \qquad (5.2.12)$$

and presented in Figure 5.9.

The system (5.2.12) was simulated with sinusoidal input signal $u(k) = sin(0.08k) + 0.08e(k)$, where $e(k)$ is normally distributed sequence with zero mean, variance $\sigma^2 = 1$

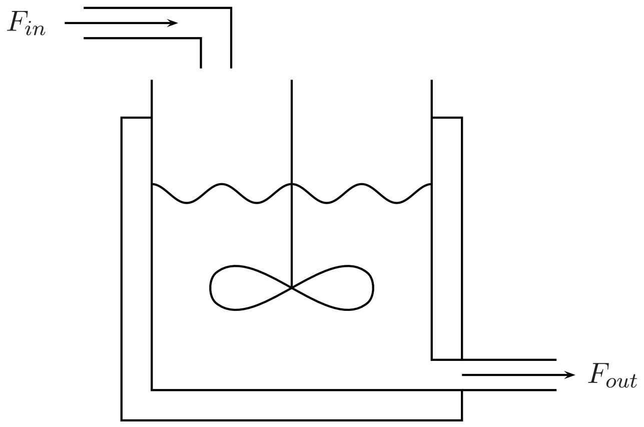


Figure 5.9: The Jacketed Continuous Stirred Tank Reactor

and standard deviation $\sigma = 1$, and a set of training input-output data obtained. This training data set was used to train NN-based ANARX structure (5.1.4) with *Levenberg-Marquardt* algorithm to model the system (5.2.12). The network shown in Fig. 5.1 was trained with three sublayers corresponding to the third order of the model and hyperbolic tangent sigmoid hidden layer activation functions on each sublayer. In this case, identified parameters of the model have the following values.

$$W_1 = \begin{bmatrix} 2.0975 & 0.0897 \\ -0.8384 & 0.7745 \\ 0.5225 & -0.0020 \end{bmatrix},$$

$$W_2 = \begin{bmatrix} -2.7080 & 0.8569 \\ -2.1183 & 2.7942 \\ 2.6811 & -0.8399 \\ -2.1036 & 2.7818 \end{bmatrix},$$

$$W_3 = \begin{bmatrix} -0.0516 & 0.0156 \\ -0.1247 & 0.0377 \\ -7.7123 & 9.6498 \end{bmatrix},$$

$$C_1 = \begin{bmatrix} 0.1430 & 0.6203 & 3.4767 \end{bmatrix},$$

$$C_2 = \begin{bmatrix} -31.8444 & 30.2608 & -32.5052 & -30.5429 \end{bmatrix},$$

$$C_3 = \begin{bmatrix} 821.9564 & -341.1409 & -0.0109 \end{bmatrix}.$$

After training the model, its identified parameters C_1 and W_1 were used for simulating the static function (5.1.5) with random inputs and the obtained data set was used to train a two-layer perceptron (5.2.10) with hyperbolic tangent sigmoid transfer function on its hidden layer. After that, using identified parameters W_0, W_2, W_3, C_0, C_2 and C_3, the controller for the system (5.2.12) can be represented by the following equations

$$\begin{cases} u(k) &= C_0\left(\dfrac{2}{1+\mathrm{e}^{-2\cdot W_0\cdot[y(k),\eta_1(k)]^T}}-1\right) \\ \eta_1(k+1) &= \eta_2(k)-C_2\left(\dfrac{2}{1+\mathrm{e}^{-2\cdot W_2\cdot[y(k),u(k)]^T}}-1\right) \\ \eta_2(k+1) &= v(k)-C_3\left(\dfrac{2}{1+\mathrm{e}^{-2\cdot W_3\cdot[y(k),u(k)]^T}}-1\right) \end{cases} .$$

The control algorithm (5.2.11) was applied to control the system (5.2.12) and the piecewise constant reference signal $v(k)$ was used for its testing. The simulation result of the system is represented in Fig. 5.10.

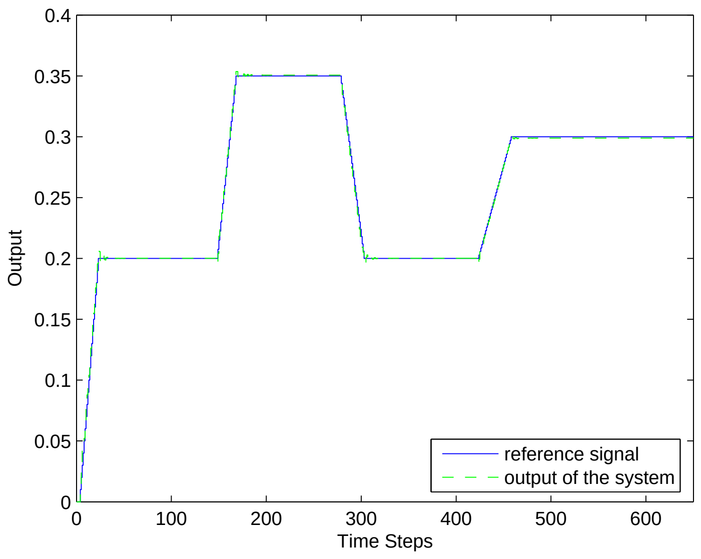


Figure 5.10: Closed loop simulation with piecewise constant reference signal

The corresponding control signal $u(k)$ is depicted in Fig. 5.11.

It can be seen from Fig. 5.10 that the output of the system (5.2.12) and reference signal $v(k)$ are almost indistinguishable, when the ANARX structure and control algorithm, based on additional static neural network, are used.

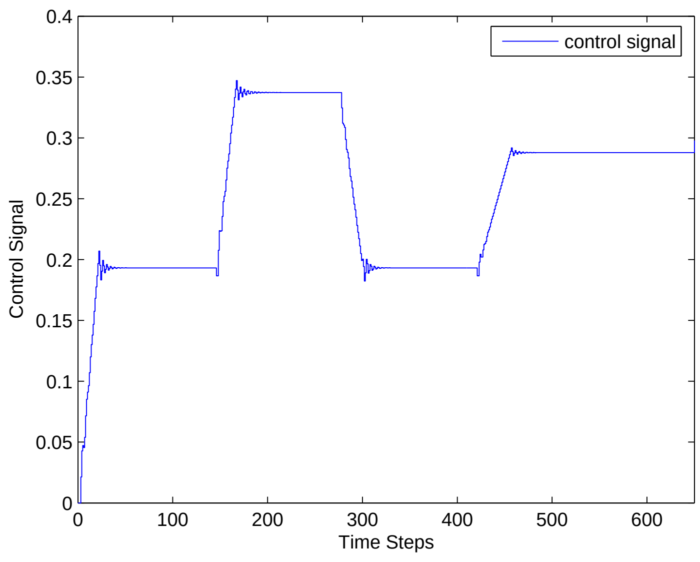


Figure 5.11: Control signal

5.2.4 Taylor Series based approach

Taylor Series based method was proposed in [7] as an alternative technique, which helps to solve the problem of control signals calculation from (5.1.5).

This approach can be divided into two following submethods:

- With multiplying by the *pseudoinverse* matrix C_1^+.
- Without multiplying by the *pseudoinverse* matrix C_1^+.

First of all, for better understanding, the equation (5.1.5) can be rewritten in the following form

$$\begin{bmatrix} c_{11} & \cdots & c_{1l_1} \\ \vdots & \ddots & \vdots \\ c_{m1} & \cdots & c_{ml_1} \end{bmatrix} \cdot \phi_1 \left(\begin{bmatrix} \sum_{i=1}^{m} w_{1i} y_i(k) + \sum_{j=1}^{r} w_{1(m+j)} u_j(k) \\ \vdots \\ \sum_{i=1}^{m} w_{l_1 i} y_i(k) + \sum_{j=1}^{r} w_{l_1(m+j)} u_j(k) \end{bmatrix} \right) = \begin{bmatrix} \eta_{11}(k) \\ \vdots \\ \eta_{m1}(k) \end{bmatrix} . \tag{5.2.13}$$

Case of multiplying by the *pseudoinverse* matrix C_1^+

In order to obtain the vector of control signals $u(k)$, we should to multiply both sides of (5.2.13) by inverse matrix C_1^{-1} from the left. Here we have two possibilities:

- C_1 is not a square matrix. Then the *left pseudoinverse* matrix C_1^+ can be found by using, for example, *Moore-Penrose*[1] or *Singular value decomposition* methods.
- C_1 is a square matrix. Then the inverse matrix C_1^{-1} can be found by standard techniques.[2]

Consider the first case as a more general. It means that the number of outputs in neural network must not be equal to the number of hidden neurons in the first sublayer, i.e. $m \neq l_1$. Denote the result after multiplication matrices C_1^+ and η_1 by $d(k) = [d_1(k), \ldots, d_{l_1}(k)]^T$. Then, the equation (5.2.13) takes the following form

$$\begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix} \cdot \phi_1 \left(\begin{bmatrix} \sum\limits_{i=1}^{m} w_{1i} y_i(k) + \sum\limits_{j=1}^{r} w_{1(m+j)} u_j(k) \\ \vdots \\ \sum\limits_{i=1}^{m} w_{l_1 i} y_i(k) + \sum\limits_{j=1}^{r} w_{l_1(m+j)} u_j(k) \end{bmatrix} \right) = \begin{bmatrix} d_1(k) \\ \vdots \\ d_{l_1}(k) \end{bmatrix}.$$

The previous equation can be rewritten as the system of l_1 equations

$$\begin{cases} \phi_1 \left(\sum\limits_{i=1}^{m} w_{1i} y_i(k) + \sum\limits_{j=1}^{r} w_{1(m+j)} u_j(k) \right) & = & d_1(k) \\ & \vdots & \\ \phi_1 \left(\sum\limits_{i=1}^{m} w_{l_1 i} y_i(k) + \sum\limits_{j=1}^{r} w_{l_1(m+j)} u_j(k) \right) & = & d_{l_1}(k) \end{cases} . \qquad (5.2.14)$$

On the next step we can find Taylor series expansion of the function $\phi_1(\cdot)$ as follows

$$\begin{cases} \phi_1(a) + \phi_1'(a)(x_1(k) - a) + \cdots + R_{s_1}[x_1(k)] & = & d_1(k) \\ & \vdots & \\ \phi_1(a) + \phi_1'(a)(x_{l_1}(k) - a) + \cdots + R_{s_{l_1}}[x_{l_1}(k)] & = & d_{l_1}(k) \end{cases} , \qquad (5.2.15)$$

[1]Using this method a *left pseudoinverse* matrix C_1^+ can be found as follows $C_1^+ = (C_1^T \cdot C_1)^{-1} \cdot C_1^T$ and its dimension will be $(l_1 \times m \cdot m \times l_1)^{-1} \cdot l_1 \times m = l_1 \times m$.

[2]It is obvious that matrix C_1 must be nonsingular.

where

$$x_1(k) = \sum_{i=1}^{m} w_{1i} y_i(k) + \sum_{j=1}^{r} w_{1(m+j)} u_j(k),$$
$$\dots,$$
$$x_{l_1}(k) = \sum_{i=1}^{m} w_{l_1 i} y_i(k) + \sum_{j=1}^{r} w_{l_1(m+j)} u_j(k)$$

and $R_{s_i}[x_i(k)]$ is a remainder term known as the Lagrange remainder.

Taking into account required precision, we can form a system of polynomial equations such that the left hand side of each equation will be formed by first $s_i, i = 1, \dots, l_1$ terms of corresponding equation in (5.2.15). Note that value of s_i depends on required precision. Let b_i be the real root of the corresponding l_i-th equation in (5.2.15). Then, the control vector can be calculated by solving the following system of linear equations with respect to $u(k)$.

$$\left\{ \begin{array}{rcl} w_{1(m+1)} u_1(k) + \dots + w_{1(m+r)} u_r(k) & = & b_1 \\ & \vdots & \\ w_{l_1(m+1)} u_1(k) + \dots + w_{l_1(m+r)} u_r(k) & = & b_{l_1} \end{array} \right. \tag{5.2.16}$$

Case where multiplication by the *pseudoinverse* matrix C_1^+ is not required

By analogy with the previous case we can transform (5.2.13), after deriving Taylor series of the function $\phi_1(\cdot)$ about a point a, into the following form

$$\begin{bmatrix} c_{11} & \cdots & c_{1l_1} \\ \vdots & \ddots & \vdots \\ c_{m1} & \cdots & c_{ml_1} \end{bmatrix} \cdot \begin{bmatrix} \phi_1(a) + \phi_1'(a)(x_1(k) - a) + \dots + R_{s_1}[x_1(k)] \\ \vdots \\ \phi_1(a) + \phi_1'(a)(x_{l_1}(k) - a) + \dots + R_{s_{l_1}}[x_{l_1}(k)] \end{bmatrix} = \begin{bmatrix} \eta_{11}(k) \\ \vdots \\ \eta_{m1}(k) \end{bmatrix}. \tag{5.2.17}$$

Then, after multiplying matrices at the left part of (5.2.17), we obtain the system of m equations and r unknowns.

$$\left\{ \begin{array}{rcl} c_{11}\left(\phi_1(a) + \phi_1'(a)(x_1(k) - a) + \dots + R_{s_1}[x_1(k)]\right) + & & \\ +c_{1l_1}\left(\phi_1(a) + \phi_1'(a)(x_{l_1}(k) - a) + R_{s_{l_1}}[x_{l_1}(k)]\right) & = & \eta_{11}(k) \\ & \vdots & \\ c_{m1}\left(\phi_1(a) + \phi_1'(a)(x_1(k) - a) + \dots + R_{s_1}[x_1(k)]\right) + & & \\ +c_{ml_1}\left(\phi_1(a) + \phi_1'(a)(x_{l_1}(k) - a) + R_{s_{l_1}}[x_{l_1}(k)]\right) & = & \eta_{m1}(k) \end{array} \right. \tag{5.2.18}$$

Let $s_1 = \cdots = s_{l_1} = 1$, then it follows from (5.2.18) that the vector of control signals $u(k)$ can be easily calculated using the following formula

$$u(k) = (C_1 \cdot W_{12})^{-1}(4\eta_1(k) - C_1(2 + W_{11} \cdot y(k))), \tag{5.2.19}$$

where matrices W_{11} and W_{12} are defined as

$$W_1 = W_{11} \begin{bmatrix} y_1(k) \\ \vdots \\ y_m(k) \end{bmatrix} + W_{12} \begin{bmatrix} u_1(k) \\ \vdots \\ u_r(k) \end{bmatrix}.$$

The following two examples illustrate methods described above in this section. The first example considers the second method as a control technique of a single-input single-output system. The second example demonstrates applicability of the first method to control of a nonlinear multi-input multi-output system.

Example 5.2.5 The model of a liquid level system of interconnected tanks [8] is represented by the following input-output equation

$$\begin{aligned} y(k+3) &= 0.43y(k+2) + 0.681y(k+1) - 0.149y(k) + 0.396u(k+2) \\ &+ 0.014u(k+1) - 0.071u(k) - 0.351y(k+2)u(k+2) \\ &- 0.03y^2(k+1) - 0.135y(k+1)u(k+1) - 0.027y^3(k+1) \\ &- 0.108y^2(k+1)u(k+1) - 0.099u^3(k+1). \end{aligned} \tag{5.2.20}$$

The system (5.2.20) was simulated and the obtained set of the input-output data was used for training NN-based ANARX structure with *Levenberg-Marquardt* algorithm and modeling (5.2.20). Identified parameters of the trained model have the following values.

$$W_1 = \begin{bmatrix} -0.0069 & 0.1429 \\ 1.0365 & -0.1062 \\ 1.0732 & -0.3226 \end{bmatrix},$$

$$W_2 = \begin{bmatrix} 1.9410 & 1.2864 \\ -1.3609 & -0.1965 \\ 2.1961 & 0.8532 \\ 2.4046 & 1.4411 \\ 1.6330 & 0.5994 \end{bmatrix},$$

$$W_3 = \begin{bmatrix} 1.1438 & 0.5369 \\ 0.6990 & 0.2296 \\ 0.4645 & -0.8094 \\ 0.4570 & -0.2183 \\ -1.5120 & -0.8273 \end{bmatrix},$$

$$C_1 = \begin{bmatrix} 49.6393 & -21.0410 & 25.4327 \end{bmatrix},$$

$$C_2 = \begin{bmatrix} 16.7279 & -20.1029 & 13.5191 & -10.2899 & -38.5246 \end{bmatrix},$$

$$C_3 = \begin{bmatrix} -22.8377 & 55.3185 & 22.6860 & -67.1972 & -3.3984 \end{bmatrix}.$$

Logarithmic sigmoid hidden layer activation functions $\mathrm{logsig}(x) = \frac{1}{1+e^{-x}}$ were used for identification of the model. Notice that Taylor series expansion of $\mathrm{logsig}(x)$ about the point $a = 0$ is $\frac{1}{1+e^{-x}} = \frac{1}{2} + \frac{x_1}{4} - \frac{x_1^3}{48} + \frac{x_1^5}{480} - \frac{x_1^7}{80640} + \ldots$. Let us choose $s_1 = s_2 = s_3 = 1$, then only two terms of the series should be taken into account. In this case, using identified parameters and the equation (5.2.19), the control signal $u(k)$ can be calculated as follows

$$u(k) = 3.552\eta_1(k) - 100.524y(k).$$

Simulation result is depicted in Figure 5.12 and clearly shows that closed loop system is capable of tracking reference signal.

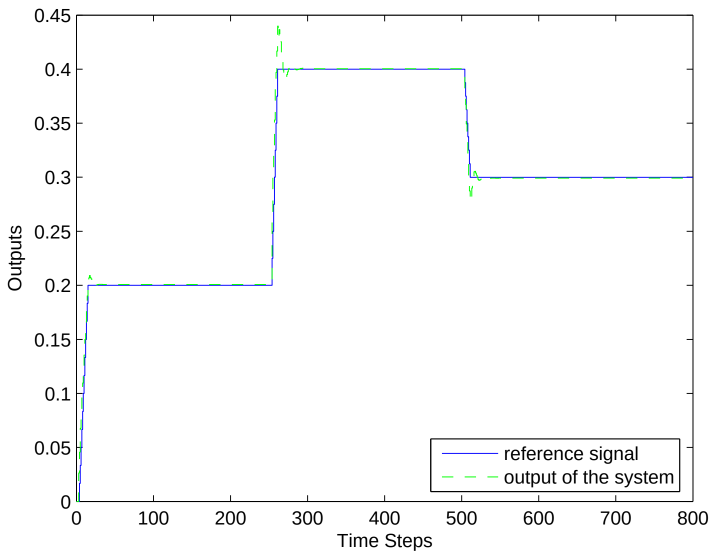


Figure 5.12: Simulation results of the system (5.2.20)

Example 5.2.6 Consider the following MIMO discrete-time system [22]

$$\begin{aligned} y_1(k+1) &= 0.4y_1(k) + \frac{u_1(k)}{1+u_1^2(k)} + 0.2u_1^3(k) + 0.5u_2(k) \\ y_2(k+1) &= 0.2y_2(k) + \frac{u_2(k)}{1+u_2^2(k)} + 0.4u_2^3(k) + 0.2u_1(k) \end{aligned} . \qquad (5.2.21)$$

It was identified by the second order NN-based ANARX model with 2 and 5 neurons on the first and the second sublayers, respectively. Logarithmic sigmoid activation functions were used in each sublayer. Identified parameters of the trained model have the following values.

$$W_1 = \begin{bmatrix} 0.0708 & -0.0398 & 0.0345 & -0.0061 \\ -0.0635 & 0.0022 & -0.0363 & -0.0235 \end{bmatrix},$$

$$W_2 = \begin{bmatrix} -18.8125 & 8.7142 & -22.5216 & -2.5913 \\ -16.5408 & 7.1620 & -20.0015 & -1.9538 \\ -0.0497 & 0.0086 & -0.0824 & -0.0210 \\ 0.0672 & -0.0215 & 0.1016 & -0.0199 \\ 17.5238 & -7.8347 & 21.0865 & 2.2320 \end{bmatrix},$$

$$C_1 = \begin{bmatrix} 4.5297 & -85.2722 \\ -118.8578 & -135.2028 \end{bmatrix},$$

$$C_2 = \begin{bmatrix} 0.9718 & 1.2593 & 61.0178 & 24.3266 & 2.2261 \\ 16.8692 & 22.1391 & 101.6562 & 74.4212 & 38.9746 \end{bmatrix}.$$

Let us choose $s_1 = s_2 = 3$ in (5.2.15), with respect to approximation of the logarithmic sigmoid function with cubic polynomials. After applying (5.2.15) and (5.2.16), the control signals $u_1(k)$ and $u_2(k)$ can be calculated in the following way

$$\begin{cases} 0.0345u_1(k) - 0.0061u_2(k) &= b_1 \\ -0.0363u_1(k) - 0.0235u_2(k) &= b_2 \end{cases},$$

where b_1 and b_2 are defined as follows

$$b_1 = \text{Root}\left[\frac{1}{2} + \frac{x_1(k)}{4} - \frac{x_1^3(k)}{48} = d_1(k)\right] - 0.0708y_1(k) + 0.0398y_2(k)$$

and

$$b_2 = \text{Root}\left[\frac{1}{2} + \frac{x_2(k)}{4} - \frac{x_2^3(k)}{48} = d_2(k)\right] + 0.0635y_1(k) - 0.0022y_2(k).$$

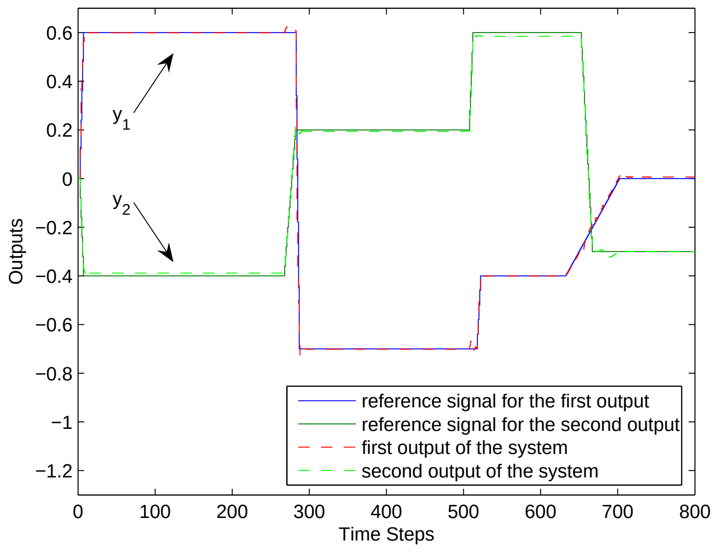


Figure 5.13: Closed loop simulation with piecewise constant reference signal

The system (5.2.21) was simulated with piecewise constant reference signals $v_1(k)$ and $v_2(k)$, and simulation results are depicted in Figure 5.13.

It can be clearly seen that proposed technique provides acceptable tracking results.

5.2.5 Analytical approach

The technique presented below provides an analytical method for calculation of the inverse function $F^{-1}(y(k), \eta_1(k))$ of (5.1.2), in order to make it possible implementation of the NN-ANARX structure based dynamic output feedback linearization algorithm for control of nonlinear systems.

Consider the equation (5.1.5) of the first nonlinear sublayer of the neural network depicted in Fig. 5.1. After multiplying matrices W_1 and $[y(k), u(k)]^T$ we can make a number of transformations and rewrite equation (5.1.5) in the following way

$$\begin{bmatrix} c_{11} & \cdots & c_{1l_1} \\ \vdots & \ddots & \vdots \\ c_{m1} & \cdots & c_{ml_1} \end{bmatrix} \cdot \phi_1 \left(\begin{bmatrix} \sum_{i=1}^{m} w_{1i} y_i(k) + \sum_{j=1}^{r} w_{1(m+j)} u_j(k) \\ \vdots \\ \sum_{i=1}^{m} w_{l_1 i} y_i(k) + \sum_{j=1}^{r} w_{l_1(m+j)} u_j(k) \end{bmatrix} \right) = \begin{bmatrix} \eta_{11}(k) \\ \vdots \\ \eta_{m1}(k) \end{bmatrix}. \tag{5.2.22}$$

In order to continue transformations of (5.2.22), we should multiply its both sides by inverse matrix C_1^{-1} from the left. Analogously with Taylor series based approach described above, here we have the same two possibilities:

- C_1 is not a square matrix.
- C_1 is a square matrix.

Consider the first case as a more general. It means that the number of outputs in neural network should not be equal to the number of hidden neurons in the first sublayer, i.e. $m \neq l_1$. Define the result after multiplication of matrices C_1^+ and $\eta_1(k)$ as $D = [d_1(k), \ldots, d_{l_1}(k)]^T$. Thus, the equation (5.2.22) can be rewritten in the following way

$$\begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix} \cdot \phi_1 \left(\begin{bmatrix} \sum_{i=1}^{m} w_{1i} y_i(k) + \sum_{j=1}^{r} w_{1(m+j)} u_j(k) \\ \vdots \\ \sum_{i=1}^{m} w_{l_1 i} y_i(k) + \sum_{j=1}^{r} w_{l_1(m+j)} u_j(k) \end{bmatrix} \right) = \begin{bmatrix} d_1(k) \\ \vdots \\ d_{l_1}(k) \end{bmatrix}.$$

After multiplying matrices I_{l_1} and $\phi_1(\cdot)$, we obtain the system of the form

$$\begin{cases} \phi_1 \left(\sum_{i=1}^{m} w_{1i} y_i(k) + \sum_{j=1}^{r} w_{1(m+j)} u_j(k) \right) & = & d_1(k) \\ & & \vdots \\ \phi_1 \left(\sum_{i=1}^{m} w_{l_1 i} y_i(k) + \sum_{j=1}^{r} w_{l_1(m+j)} u_j(k) \right) & = & d_{l_1}(k) \end{cases}.$$

Further, after finding the inverse function $\phi_1^{-1}(\cdot)$, we obtain the following system of l_1

equations and r variables

$$\left\{\begin{array}{rcl} w_{1(m+1)}u_1(k) + \cdots + w_{1(m+r)}u_r(k) & = & b_1(k) \\ & \vdots & \\ w_{l_1(m+1)}u_1(k) + \cdots + w_{l_1(m+r)}u_r(k) & = & b_{l_1}(k) \end{array}\right. , \qquad (5.2.23)$$

where

$$b_1(k) = \phi_1^{-1}(d_1(k)) - \sum_{i=1}^{m} w_{1i}y_i(k),$$

$$\ldots,$$

$$b_{l_1}(k) = \phi_1^{-1}(d_{l_1}(k)) - \sum_{i=1}^{m} w_{l_1 i}y_i(k).$$

Remark 5.2.1 *It follows from* $(5.2.23)$*, that the activation function* $\phi_1(\cdot)$ *of the first hidden layer must be invertible.*

Consider the most popular sigmoid type neurons activation functions of the first hidden sublayer:

- Logarithmic sigmoid transfer function $\mathrm{logsig}(x) = \frac{1}{1+e^{-x}}$.
- Hyperbolic tangent sigmoid transfer function $\mathrm{tansig}(x) = \frac{2}{1+e^{-2x}} - 1$.

For these functions the restriction imposed by Remark 5.2.1 is satisfied and as a result the equation (5.2.23) can be rewritten as follows.

In case of logarithmic sigmoid function the system (5.2.23) can be transformed in the following way

$$\left\{\begin{array}{rcl} w_{1(m+1)}u_1(k) + \cdots + w_{1(m+r)}u_r(k) & = & b_1(k) \\ & \vdots & \\ w_{l_1(m+1)}u_1(k) + \cdots + w_{l_1(m+r)}u_r(k) & = & b_{l_1}(k) \end{array}\right. , \qquad (5.2.24)$$

where

$$b_1(k) = \ln\left(\frac{d_1(k)}{1-d_1(k)}\right) - \sum_{i=1}^{m} w_{1i}y_i(k),$$

$$\ldots,$$

$$b_{l_1}(k) = \ln\left(\frac{d_{l_1}(k)}{1-d_{l_1}(k)}\right) - \sum_{i=1}^{m} w_{l_1 i} y_i(k).$$

If we discuss analogously for the second function, then the system (5.2.23) is similar to (5.2.24), but where

$$b_1(k) = \frac{1}{2}\cdot\ln\left(\frac{1+d_1(k)}{1-d_1(k)}\right) - \sum_{i=1}^{m} w_{1i} y_i(k),$$
$$\ldots,$$
$$b_{l_1}(k) = \frac{1}{2}\cdot\ln\left(\frac{1+d_{l_1}(k)}{1-d_{l_1}(k)}\right) - \sum_{i=1}^{m} w_{l_1 i} y_i(k).$$

Proposition 5.2.1 *In order to obtain the **exact** solution of* $(5.2.23)$*, the following conditions must be satisfied* $r = l_1 = m$*, otherwise an **optimal** solution can be found.*

Proof: The proof of that fact follows directly from dimensions of matrices C_1 and W_{12}, where W_{12} is a $l_1 \times r$ submatrix of $l_1 \times (m+r)$ matrix W_1, i.e.

$$W_{12} = \begin{bmatrix} w_{1(m+1)} & \cdots & w_{1(m+r)} \\ \vdots & \ddots & \vdots \\ w_{l_1(m+1)} & \cdots & w_{l_1(m+r)} \end{bmatrix}.$$

The effectiveness of the proposed approach is demonstrated by applying it to the following academical examples.

Example 5.2.7 Consider the nonlinear MIMO discrete-time system from Example 5.2.6. The identification and modeling processes were shown above. Thus, we can simply use the obtained model of the system (5.2.21) and all identified parameters W_1, W_2, C_1 and C_2, in order to check the tracking capabilities of the proposed analytical approach.

Using identified parameters and the system of equations (5.2.24), the control signals $u_1(k)$ and $u_2(k)$ can be calculated as follows

$$\left\{\begin{array}{rcl} 0.0345u_1(k) - 0.0061u_2(k) & = & b_1(k) \\ -0.0363u_1(k) - 0.0235u_2(k) & = & b_2(k) \end{array}\right.,$$

where

$$b_1(k) = \ln\left(\frac{0.0142\eta_{11}(k) - 0.009\eta_{21}(k)}{1 - 0.0142\eta_{11}(k) + 0.009\eta_{21}(k)}\right) - 0.0708y_1(k) + 0.0398y_2(k)$$

and

$$b_2(k) = \ln\left(\frac{-0.0125\eta_{11}(k) + 0.0005\eta_{21}(k)}{1 + 0.0125\eta_{11}(k) - 0.0005\eta_{21}(k)}\right) + 0.0635y_1(k) - 0.0022y_2(k).$$

The system (5.2.21) was simulated with the same piecewise constant reference signals $v_1(k)$ and $v_2(k)$ as in Example 5.2.6, and simulation results are depicted in Figure 5.14.

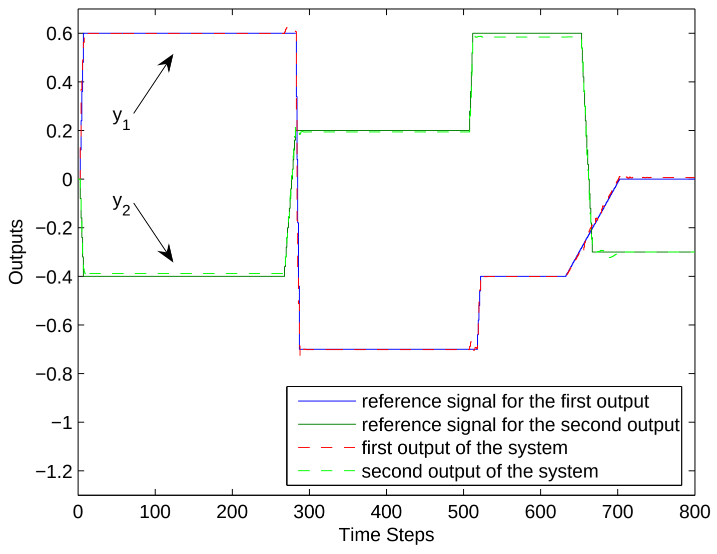


Figure 5.14: Closed loop simulation with piecewise constant reference signals

One can easily see from Fig. 5.14 that simulation results in case of analytical approach are almost the same as in case of Taylor series based approach presented in the previous example.

Example 5.2.8 The nonlinear MIMO discrete-time system [31] is described by the following input-output equations

$$\begin{aligned} y_1(k+1) &= \frac{y_1(k)}{1+y_2^2(k)} + u_1(k) \\ y_2(k+1) &= \frac{y_1(k)y_2(k)}{1+y_2^2(k)} + u_2(k) \end{aligned}. \tag{5.2.25}$$

The system (5.2.25) was simulated and the obtained set of the input-output data was used for training of MIMO NN-based ANARX structure with *Levenberg-Marquardt* algorithm. The network shown in Fig. 5.1 was trained with two sublayers corresponding to the second order of the model and logarithmic sigmoid hidden layer activation functions were used. As we are interested in the exact solution of (5.2.23), then $l_1 = 2$ was chosen.

The system (5.2.25) was simulated with piecewise constant reference signals $v_1(k)$ and $v_2(k)$, and simulation results are depicted in Figure 5.15.

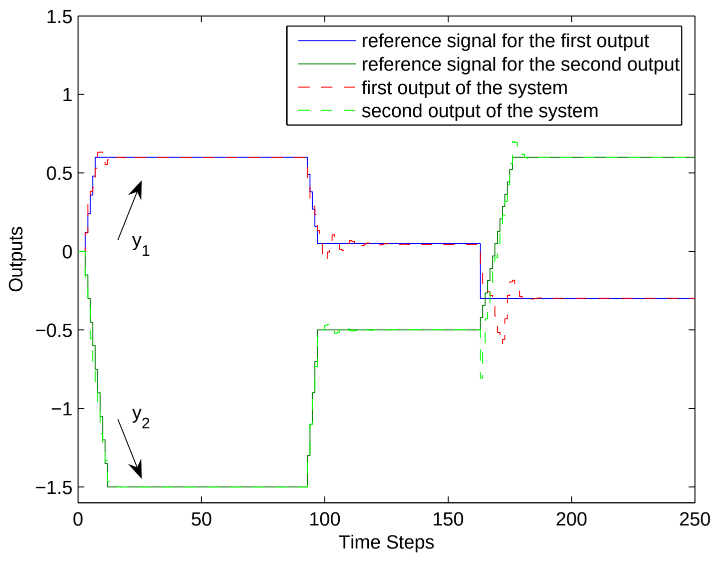


Figure 5.15: Closed loop simulation with piecewise constant reference signals

It can be seen from Fig. 5.15 that the control system is capable of tracking both reference signals $v_1(k)$ and $v_2(k)$ when the analytical approach for the control signals calculation is applied.

5.3 Discussion

In this section the author presents a brief analysis of advantages and drawbacks of control techniques described in the previous section. The main reason why the following methods have been created relies on the fact, that ANARX structure by itself is nothing more than a convenient form of representation of the identified model. Thus, without possibility of calculating control signals, it cannot be used in control application.

Newton's method can be applied to control any nonlinear SISO system identified by an NN-based ANARX model. It can be used for solving the main problem associated with ANARX structure, namely calculation of the control signals $u(k)$. Unfortunately, the convergence speed of a such classical numerical algorithm and high complexity of applying it to MIMO systems make use of this method undesirable for solving hard enough tasks.

In order to overcome problems with application of Newton's method to control of nonlinear systems of different complexity and make calculation of the control signals fast enough, a new class of ANARX structure, so called Simplified ANARX, was introduced in [34]. As was mentioned above, SANARX imposes an additional restrictions on ANARX structure, removing nonlinearity from the first term of (5.1.1) and replacing it with a linear function. In case of neural networks, the structure depicted in Fig. 5.1 with the linear transfer function on the first sublayer can be used for identification of NN-based SANARX model. The practice has shown that this is the most simple and fast enough technique for control signals calculation. Unfortunately, SANARX can be applied only to systems with equal number of inputs and outputs, i.e. $r = m$.

Although Simplified ANARX structure proofed itself as a reliable technique, unfortunately, the linearity of the first term in (5.1.1) reduces the class of functions to be identified. Therefore, another technique for the control signals $u(k)$ calculation was proposed in [36], in order to avoid this restriction. This approach is based on an additional static neural network and does not impose any restrictions on ANARX structure. Unfortunately, the training of the additional neural network for approximation of the inverse function $F^{-1}(y(k), \eta_1(k))$ takes some time. Of course, one can make it offline and after that use this method in control application. But if we talk about online learning of the neural network and especially about adaptive control, then from the computational point of view this technique is undesirable.

Additionally to methods described above, an alternative technique was proposed by the author in [7]. This approach is based on Taylor series and can be divided into the following submethods.

The first of them allows to simplify the process of control vector calculation and reduce it to finding a solution for each of l_1 polynomial equations of the system (5.2.15), separately, and after that solving the system of linear equations (5.2.16). On the one hand, such approach imposes the following rigid restriction $r = l_1 = m$. It should be noticed that such restriction makes possible application of this method only to multi-input multi-output systems with equal number of inputs and outputs and to some single-input single-output systems. In case of SISO systems $l_1 = 1$, but, according to Stone-Weierstrass theorem, the minimal neural network, which can approximate a continuous-time nonlinear function of any complexity, should be represented by a two-layer perceptron. It means that l_1 should at least be equal 2. On the other hand this method allows to approximate almost any hidden layer activation function with an arbitrary accuracy, without dramatic increasing the complexity of computational process. Unfortunately, it will be the vector of approximated control signals. Besides that, the process of finding solutions of each polynomial equation with an arbitrary degree of accuracy in some situations may become costly, even if one uses numerical methods, not mentioning about techniques for symbolical calculations.

The second method allows partially to overcome the restriction $r = l_1 = m$ and to reduce it only to $r = m$. As a result, this method can be applied to MIMO as well as to SISO systems. On the one side, if the approximation order of the hidden layer activation function differs from 1, it means that $s_1 \neq 1, \ldots, s_{l_1} \neq 1$, then the finding of the solution of the system of equations (5.2.18) becomes an extremely difficult task. But on the other side, if $s_i = 1$, then the vector of control signals can be easily calculated using equation (5.2.19). In the last case this method is similar to the technique based on NN-SANARX model, but without linearity, what can sometimes be important for identification process.

Finally, the last described technique is based on analytical calculation of the vector of control signals. This method is similar to the first submethod of Taylor series based approach, but at the same time it does not require the finding solutions of polynomial functions, what significantly improves the speed of control algorithm. The main disadvantage of this method is that it imposes the same unpleasant restriction $r = l_1 = m$, what makes it applicable only to MIMO and some SISO systems.

Summarizing the information presented above, one can see that there is no method, which will be panacea for any situation. In other words, the choice of a method depends on a problem to be solved.

Conclusions

This chapter is devoted to summarizing of what has been studied and some of the results achieved by the author.

Implementation in CAS *Mathematica* of Taylor series expansion based and integration based discretization methods for nonlinear continuous-time systems are discussed in detail in the first part of the present thesis. The main result is a subfamily of five functions for the **NLControl** package, which allows to

- check if a given system is finitely discretizable in the original coordinates by finding a dilation with respect to the homogenous degree -1
- check if the Lie algebra associated with a given system is nilpotent or not. It means, if locally there exist state coordinates in which this system is finitely discretizable or not
- calculate homogeneous degrees of the vector fields associated with a continuous-time system with respect to the dilation
- perform discretization using either Taylor series expansion or direct integration of the differential equations
- plot outputs of original and discretizated systems

All these methods were implemented in the form of Mathematica functions and integrated into the **NLControl** package.

One of the future goals is implementation of the function that finds the state coordinates in which the nilpotent system is finitely discretizable. Now this function works only partially, on some simple examples. Besides that, the author plans to implement the function that finds the nilpotent approximation to the arbitrary control system that opens alternative possibility to find the approximate discrete-time model. Besides above, there are many other ways to extend the present contribution. Based on the theoretical results of [28], it is possible to drop the assumption of the piecewise constant control and implement the more general sampled data models. It is equally important to drop the assumption of equidistant (regular) sampling and allow the sampling rate to change based on the computation resources availability [3].

The application of the specific neural network with so called Additive Nonlinear Autoregressive eXogenous structure for identification and control of nonlinear systems was considered in the second part of the present thesis. This structure, from the control systems point of view, has a number of significant advantages. However, the only problem, which complicates the application of this technique, is complexity of calculation of the control signal from the dynamics of the controller. Therefore, a number of different solutions were proposed, among them Newton's method, NN-based Simplified ANARX and the technique based on additional static neural network. Additionally to presented methods, two more approaches were proposed by the author in order to overcome this obstacle. Each of described techniques has a number of advantages and disadvantages and as a result can be applied in different situations.

- Newton's method

  Advantages:

  - can be used to control any nonlinear SISO system

  Disadvantages:

  - high complexity of applying to control of MIMO system
  - the convergence speed

- The NN-based Simplified ANARX method

  Advantages:

  - pointed to control of SISO as well as MIMO systems

  - control signals can be calculated very easily, what follows from solving the linear equation (in case of SISO system) or system of linear equations (in case of MIMO system)

  Disadvantages:

  - imposes an additional restriction on NN-based ANARX structure, namely linearity of the first sublayer
  - applicable to systems with equal number of inputs and outputs

- Additional static neural network based approach

  Advantages:

  - can be used to control of SISO and MIMO systems
  - does not impose any restrictions on ANARX structure

  Disadvantages:

  - requires time for training of the additional neural network

- Taylor series based approach

  <u>First submethod</u>

  Advantages:

  - applicable to control of MIMO systems
  - does not simplifies ANARX structure
  - allows to approximate almost any hidden layer activation function with an arbitrary accuracy without dramatic increasing the complexity of computational process

  Disadvantages:

  - can be used to control of a very restricted class of SISO systems
  - the number of inputs, first hidden layer neurons and outputs has to be equal
  - the process of finding solutions of each polynomial equation with an arbitrary degree of accuracy in some situations may become costly

Second submethod

Advantages:

- ○ can be used to control of SISO and MIMO systems
- ○ does not impose any restrictions on ANARX structure
- ○ if the approximation order of the hidden layer activation function equals 1, then the vector of control signals can be calculated easily enough

Disadvantages:

- ○ the number of inputs and outputs has to be equal
- ○ in case if the approximation order of the hidden layer activation function greater than 1, then solving the system of polynomial equations becomes an extremely difficult task

- The analytical approach

  Advantages:

  - ○ can be used to control of MIMO systems
  - ○ does not impose any restrictions on ANARX structure

  Disadvantages:

  - ○ can be used to control of a very restricted class of SISO systems
  - ○ the number of inputs, first hidden layer neurons and outputs has to be equal
  - ○ the hidden layer activation function has to be invertible

The effectiveness of each method was demonstrated in present thesis on several numerical examples.

One of the possible directions for the future research is application of NN-based ANARX model and dynamic output feedback linearization algorithm to control of a nonlinear multi-input multi-output systems with unequal numbers of inputs and outputs. Besides that, it is reasonable to adjust introduced techniques for the case of adaptive control.

Bibliography

[1] Agrawal, S. K., Bhattacharya, S. Optimal Control of Driftless Nilpotent Systems: Some New Results. – *Proceedings of the 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems, Kyongju, South Korea*, 1999, 356-361.

[2] Albertos, P. Sampled-data Modeling and Control of Nonlinear Systems. – *Proceedings of the 35th Conference on Decision and Control, Kobe, Japan*, 1996, 925-930.

[3] Albertos, P., Valles, M., Crespo, A. Digital control design and implementation. – *Proceedings of the European Control Conference, Kos, Greece*, 2007, 1159-1166.

[4] Åström, K. J., Furuta, K. Swinging Up a Pendulum by Energy Control. – *Presented at 13th IFAC World Congress, San Francisco, CA*, 1996.

[5] Belikov, J., Petlenkov, E. Calculation of the control signal in MIMO NN-based ANARX models: Analytical approach. – *Submitted for review*, 2008.

[6] Belikov, J., Petlenkov, E., Nõmm, S. Application of Neural Networks based ANARX structure to backing up control of a truck-trailer. – *Proceedinngs of the 6th IFAC Symposium on Intelligent Autonomous Vehicles, Toulouse, France,* September 2007, 1-5.

[7] Belikov, J., Vassiljeva, K., Petlenkov, E., Nõmm S. A Novel Taylor series based Approach for Control Computation in NN-ANARX Structure based Control of Nonlinear Systems. – *Proceedings of the 27th Chinese Control Conference, Kunming, China,* July 2008.

[8] Billings, S. A., Fadzil, M. B. The practical identification of systems with nonlin-

earities. – *Proceedings of the 7th IFAC/IFORS Symp. Identification Syst. Parameter Estimation, York, UK*, 1985, 20-28.

[9] Canudas, C., Siliciano, B., Bastin, G., Brogliat, B., Campion, G., D'Andrea-Novel, B., De Luca, A., Khalil, W., Lozano, R., Ortega, R., Samson, C., Tomei, P. Theory of Robot Control. London : Springer-Verlag, 1996.

[10] Chelouah, A., Petitot, M. Finitely discretizable nonlinear systems: Concepts and definitions. – *Proceedings of the 34th Conference on Decision & Control, New-Orleans, LA*, 1995, 19-24.

[11] Chung-Cheng Chen, Chao-Hsing Hsu, Ying-Jen Chen and Yen-Feng Lin. Disturbance attenuation of nonlinear control systems using an observer-based fuzzy feedback linearization control. – *Chaos, Solitons and Fractals*, 2007, no. 33, 885-900.

[12] Chowdhury, F. N. Input-output modeling of nonlinar systems with time-varying linear models. – *IEEE Transactions on Automatic Control*, 2000, vol. 7, 1355-1358.

[13] Chowdhury, F. N., Kotta, Ü., Nõmm, S. On realizability of neural networks-based input-output models. – *Proceedings of the 3rd Int. Conf. on Differential Equations and Applications, St. Petersburg, Russia*, 2000, vol. 6, 47-51.

[14] Glad, T., Ljung, L. Control theory. Multivariable and Nonlinear Methods. New York: Taylor & Francis, 2000.

[15] Hermes, H. Nilpotent and High-Order Approximations of Vector Field Systems. – *SIAM Review*, 1991, vol. 33, no. 2, 238-264.

[16] Hunt, K. J., Irwin, G. R., Warwick, K. Neural Network Engineering in Dynamic Control Systems: Advances in Industrial Control. London: Springer-Verlag, 1995.

[17] Ichihashi, H., Tokunaga, M. Neuro-Fuzzy Optimal Control of Backing up a Trailer Truck. – *Proceedings of the IEEE International Conference on Neural Networks*, 1993, 306-311.

[18] Kazentzis, N., Kravaris, C. System-theoretic properties of sampled-data representation of nonlinear systems obtained via Taylor-Lie series. – *International Journal of Control*, 1997, vol. 67, 997-1020.

[19] Kong, S., Kosko, B. Adaptive fuzzy systems for backing up a truck-and-trailer. – *IEEE Transactions on Fuzzy Systems*, 1992, vol. 3, no. 2, 211-223.

[20] Kotta, Ü., Chowdhury, F., Nõmm, S. On realizability of neural networks-based input-output models in the classical state space form. – *Automatica*, 2006, vol. 42, no. 7, 1211-1216.

[21] Kotta, Ü., Nõmm, S., Chowdhury, F. On a new type of neural network-based input-output model: The ANARMA structure. – *Proceedings of the 5th IFAC Symposium on nonlinear control systems NOLCOS, St. Petersbourg, Russia*, 2001, 1623-1626.

[22] Li, X., Bai, Y., Yang, L. Neural network online decoupling for a class of nonlinear system. – *Proceedings of the 6th World Congress on Intelligent Control and Automation, Dalian, China*, June 21-23, 2006, 2920-2924.

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

[24] McClamroch, N. H. State Models of Dynamics Systems. A Case Study Approach. New York : Springer-Verlag, 1980.

[25] McCulloch, W. S., Pitts, W. A logical calculus of the ideas immanent in nervous activity. – *Bulletin of Mathematical Biophysics*, 1943, vol. 5, 115-133.

[26] Marquez, H. J. Nonlinear Control Systems. Analysis and Design. New Jersey : John Wiley & Sons, 2003.

[27] Monaco, S., Normand-Cyrot, D. Advanced tools for nonlinear sampled-data systems analysis and control. – *Proceedings of the European Control Conference, Kos, Greece*, 2007, 1155-1158.

[28] Monaco, S., Normand-Cyrot, D., Califano C. From chronological calculus to exponential representations of continuous and discrete-time dynamics: a Lie-algebraic approach. – *IEEE Trans. on Automatic Control*, 2007, vol. 52, no. 12, 2227-2241.

[29] Mullari, T., Kotta, Ü., Tõnso, M. The connection between different static state feedback linearizability conditions of discrete time nonlinear control systems. – *European Control Conference 2007: Final Program and Book of Abstracts, Kos, Greece*, July 2-5, 2007, 4268-4275.

[30] Murray, R. M., Sastry, S. S. Nonholonomic motion planning: Steering using sinusoids. – *IEEE Trans. on Automatic Control*, 1993, vol. 38, 700-716.

[31] Narendra, K. S., Partasarathy, K. Identification and Control of Dynamical Systems Using Neural Networks. – *IEEE Transactions on Neural Networks*, 1990, vol. 1, 4-27.

[32] Nguyen, D., Widrow, B. The truck backer-upper: An example of self-leaming in neural networks. *Proceedings of the International Joint Conference on Neural Networks, Washington, DC*, 1989, 357-363.

[33] Pearson, R. K., Kotta, Ü. Nonlinear discrete-time models: state-space vs. i/o representations. – *Journal of Process Control*, 2004, no. 14, 533-538.

[34] Petlenkov, E. NN-ANARX Structure Based Dynamic Output Feedback Linearization for Control of Nonlinear MIMO Systems. – *Proceedings of the 15th Mediterranean Conference on Control and Automation, Athena, Greece*, 2007, 1-6.

[35] Petlenkov, E. Neural Networks Based Identification and Control of Nonlinear Systems: ANARX Model Based Approach. – *PhD thesis, Tallinn University of Technology, Tallinn*, 2007.

[36] Petlenkov, E., Belikov, J. NN-ANARX Structure for Dynamic Output Feedback Linearization of Nonlinear SISO and MIMO Systems: Neural Networks Based Approach. – *Proceedings of the 26th Chinese Control Conference, Zhangjiajie, China*, 2007, vol. 4, 138-145.

[37] Petlenkov, E., Nõmm, S., Kotta, Ü. NN-based ANARX structure for identification and model-based control. – *Proceedings of the 9th International Conference on Control Automation Robotics & Vision, Singapore*, 2006, 2284-2288.

[38] Petlenkov, E., Nõmm, S., Kotta, Ü. Adaptive Output Feedback Linearization for a Class of NN-based ANARX Models. – *Proceedings of the 6th IEEE International Conference on Control and Automation, Guangzhou, China*, May 30 - June 1, 2007, 3173-3178.

[39] Pothin, R., Kotta, Ü., Moog, C. Output feedback linearization of nonlinear discrete-time systems. – *Proceedings of the IFAC Conf. on Control system design, Bratislava*, 2000, 174-179.

[40] Riid, A., Rüstern, E. Fuzzy hierarchical control of truck and trailer. – *Proceedings of the 8th Biennal Baltic Electronic Conf., Tallinn, Estonia*, 2002, pp. 141-144.

[41] Rosenblatt, F. Principles of Neurodynamics: Perceptron and the Theory of Brain Mechanisms. Washington: Spartan Book, 1961.

[42] Stone, M. H. The generalized Weierstrass approximation theorem. – Mathematics Magazine, 1948, vol. 21, pp. 167-184, 237-254.

[43] Tanaka, K., Sano, M. A robust stabilization problem of fuzzy control systems and its application to backing up control of a truck-trailer. – *IEEE Transactions on Fuzzy Systems*, 1994, vol. 2, no. 2, 119-134.

List of publications

1. J. Belikov, Ü. Kotta, T. Mullari, S. Nõmm and M. Tõnso, ”Discretization of Continuous-time systems with Computer Algebra System Mathematica,” *Proc. of the International Conference Cybernetics and Informatics*, Zhdiar, Slovak Republic, February 2008.

2. J. Belikov and E. Petlenkov, ”Calculation of the control signal in MIMO NN-based ANARX models: Analytical approach,” *Submitted for review*, 2008.

3. J. Belikov, E. Petlenkov and S. Nõmm, ”Application of Neural Networks based ANARX structure to backing up control of a truck-trailer,” *Proc. of the 6th IFAC Symposium on Intelligent Autonomous Vehicles*, Toulouse, France, pp. 1-5, September 2007.

4. J. Belikov, K. Vassiljeva, E. Petlenkov and Sven Nõmm, ”A Novel Taylor series based Approach for Control Computation in NN-ANARX Structure based Control of Nonlinear Systems,” *Proc. of the 27th Chinese Control Conference*, Accepted for presentation, Kunming, China, July 2008.

5. E. Petlenkov and J. Belikov, ”NN-ANARX Structure for Dynamic Output Feedback Linearization of Nonlinear SISO and MIMO Systems: Neural Networks Based Approach,” *Proc. of the 26th Chinese Control Conference*, Zhangjiajie, China, vol. 4, pp. 138-145, July 2007.

6. E. Petlenkov, J. Belikov, Sven Nõmm and Małgorzata Wyrwas, ”Dynamic Output Feedback Linearization Based Adaptive Control of Nonlinear MIMO Systems,” *Proc. of the 27th American Control Conference*, Accepted for presentation, Seattle, USA, June 2008.

7. Sven Nõmm, Eduard Petlenkov, Jüri Vain, Fuijo Miyawaki, Kitaro Yoshimitsu and Juri Belikov, ”Recognition of the Surgeon’s Motions During Endoscopic Operation by Statistics Based Algorithm and Neural Networks Based ANARX Models,” *Proc. of the 17th IFAC World Congress*, Accepted for presentation, Seoul, South Korea, July 2008.