TALLINN UNIVERSITY OF TECHNOLOGY
SCHOOL OF ENGINEERING
Department of Electrical and Power Engineering and Mechatronics

MODELING AND CONTROL WITH NEURAL NETWORKS FOR A MAGNETIC LEVITATION SYSTEM WITH TWO ELECTROMAGNETS

MODELLEERIMINE JA KONTROLL NÄRVIVÕRGU ABIL KAHE ELEKTROMAGNETIGA MAGNETILISE LEVITATSIOONISÜSTEEMI JAOKS

MASTER THESIS

Student: Alexandra Kolosova

Student code: 201639MAHM

Supervisor: Hossein Alimohammadi, Early-Stage Researcher

Co-supervisor: Mart Tamre, Professor

Tallinn 2022

AUTHOR’S DECLARATION

Hereby I declare, that I have written this thesis independently.
No academic degree has been applied for based on this material. All works, major viewpoints and data of the other authors used in this thesis have been referenced.

“18” May 2022

Author: ....
/signature /

Thesis is in accordance with terms and requirements

“.......” .... 20….

Supervisor: ….........................
/signature/

Accepted for defence

“.......”...................20… .

Chairman of theses defence commission: ....
/name and signature/

Non-exclusive licence for reproduction and publication of a graduation thesis[1]

I, Alexandra Kolosova,

1. grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis

“Modeling and Control with Neural Networks for a Magnetic Levitation System with Two Electromagnets”,

supervised by Hossein Alimohammadi,

1.1 to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;

1.2 to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.

2. I am aware that the author also retains the rights specified in clause 1 of the non-exclusive licence.

3. I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

18/05/2022 (date)

[1] *The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.*

Department of Electrical and Power Engineering and Mechatronics

THESIS TASK

Student: Alexandra Kolosova, 201639MAHM
Study programme: MAHM02/18 – Mechatronics,
main speciality: Mechatronics
Supervisor: Early-Stage Researcher Hossein Alimohammadi, +372 57852726
Co-Supervisor: Professor Mart Tamre, +372 5120982

Thesis topic:

(in English) Modeling and Control with Neural Networks for a Magnetic Levitation System with Two Electromagnets

(in Estonian) Modelleerimine ja kontroll närvivõrgu abil kahe elektromagnetiga magnetilise levitatsioonisüsteemi jaoks

Thesis main objectives:

1. Collect and analyse time-series datasets from the Magnetic Levitation System.
2. Develop the Neural Network based control solution for position tracking.
3. Improve existing PID controller's performance and extend its working range.

Thesis tasks and time schedule:

No	Task description	Deadline
1.	Literature analysis. Writing "Introduction" and "Literature overview" chapters of the thesis	October 2021- November 2021
2.	Design of the artificial neural network and related controller	December 2021 – January 2022
3.	Simulations, analysis, and adjustments of the controller	January 2022 – February 2022
4.	Writing thesis work and analysis of the results	March 2022 – April 2022
5.	Thesis submission and defense	May 2022 – June 2022

Language: English **Deadline for submission of thesis:** "18" May 2022

Student: Alexandra Kolosova ……………. "……."…………………2022
/signature/

Supervisor: Hossein Alimohammadi ……………. "……."…………………2022
/signature/

Co-Supervisor: Mart Tamre ……………. "……."…………………2022
/signature/

Head of study

programme: Anton Rassõlkin .... “.......”......................2022

/signature/

TABLE OF CONTENTS

PREFACE

The master thesis topic "Modeling and Control with Neural Networks for a Magnetic Levitation System with Two Electromagnets" was proposed by the Early-Stage Researcher Hossein Alimohammadi. The magnetic levitation system with two electromagnets (MLS2EM), used during the following work, is created by the Polish company INTECO, and currently located in Centre for Intelligent Systems, Tallinn University of Technology.

The main goal of the thesis was to design a better alternative control solution for already existing PD-controller, based on the neural networks. The work was challenging and interesting. Due to certain limitations and obstacles, it required more investigation and effort, which resulted in the significant improvement of the PD-controller, using Reinforcement Learning method.

I would like to express my gratitude towards my main supervisor, Early-Stage Researcher Hossein Alimohammadi, for assistance and support at each stage of experiments and writing the master thesis work. I would also like to thank Professor Mart Tamre from Department of Electrical and Power Engineering and Mechatronics for a right and timely advice, Professor Eduard Petlenkov from Department of Computer Systems, and Early-Stage Researcher Vjatšeslav Škiparev from Department of Software Science, for consultations and fresh eyes on issues appeared during experiments.

Keywords: magnetic levitation system, artificial neural network, reinforcement learning, control design, master thesis.

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

1-DOF	One Degree-of-Freedom
2-DOF	Two Degrees-of-Freedom
ACO	Ant Colony Optimization algorithm
ANN, NN	Artificial Neural Network, or Neural Network; Tehisnärvivõrk (est.)
ASMC	Adaptive Sliding Mode Controller
BSC	Backstepping Control
CCS	Cascade Control System
CDM	Coefficient Diagram Method
DDPG	Deep Deterministic Policy Gradient algorithm
DDPG-agent	Deep Deterministic Policy Gradient Agent
DDQN	Double-Deep Q-Network
DL Toolbox	Deep Learning Toolbox; Süvaõppe Toolbox(est.)
DN Designer	Deep Network Designer App
DPG	Deterministic Policy Gradient algorithm
DQN	Deep Q-Network
DRL	Deep Reinforcement Learning
EM	Electromagnetic Field
EM1	Electromagnet 1 (upper)
EM2	Electromagnet 2 (lower)
EMS	Electromagnetic Suspension
FL	Feedback Linearization
FLC	Fuzzy Logic Control
FOC	Fractional-Order Control
FOPID	Fractional-Order PID controller
FOSMC	Fractional-Order Sliding Mode Controller
Google Colab	Online Jupyter Notebooks environment from Google
GS-C	Gain Scheduling Control
I/O	Input/Output
IAE	Integral Absolute Error
IBS	Integral Backstepping
LCD	Liquid Crystal Display
LMA	Levenberg-Marquardt Algorithm
LQR	Linear Quadratic Regulator
MIT	Massachusetts Institute of Technology
MLFFNN	Multilayer Feedforward Neural Network
MLS	Magnetic Levitation System; Magnetilise Levitatsioonisüsteemi (est.)

MLS1EM	Magnetic Levitation System with one Electromagnet
MLS2EM	Magnetic Levitation System with two Electromagnets, kahe Elektromagnetiga Magnetilise Levitatsioonisüsteemi (est.)
MPC	Model Predictive Control
MRAC-FOPID	Model Reference Adaptive Fractional-Order PID Control
MSE	Mean Squared Error; Ruut Keskmine Viga (est.)
NARX	Nonlinear Auto-Regressive with eXogenous inputs model; Mittelineaarne Autoregressiivne Eksogeensete sisenditega mudel (est.)
NN Toolbox	Neural Network Toolbox; Närvivõrgu Toolbox (est.)
OpenAI Gym	Open Source Python Library
OU	Ornstein-Uhlenbeck noise / process
PD	Proportional Derivative
PD+RL	Proportional Derivative controller with RL-agent in parallel
PI	Proportional Integral
PID	Proportional Integral Derivative
PI-FC	Proportional Integral Fuzzy Control
PLC	Programmable Logic Controller
PWM	Pulse Width Modulation
Q-Network	Neural Network based on the Q-Value, or Q-Function
RBFNN	Radial Basis Function Neural Network
RBFNNSMC	Sliding Mode Controller based on Radial Basis Function Neural Network
ReLU	Rectified Linear Unit
RL Designer	Reinforcement Learning Designer App
RL Toolbox	Reinforcement Learning Toolbox; Stiimulõppe Toolbox (est.)
RL-agent	Reinforcement Learning Agent
RMSE	Root Mean Squared Error
RTW Toolbox	Real Time Windows Toolbox
SMC	Sliding Mode Control
SMD	Sliding Mode Differentiator
TP-C	Tensor-Product based Control
TPU	Tensor Processing Unit

1. INTRODUCTION

Magnetic levitation technology had been studied and developed during the whole 20th century. Nowadays, it is used in various fields of industry and many industrial systems such as magnetic levitation trains, electromagnetic bearing, levitation melting with induction heating, microrobotic systems, vibration isolation, wind turbines, levitation of wind tunnel models, aerospace, etc. The key advantage of the use of magnetic levitation technology is the possibility to almost eliminate contact and, subsequently, friction between surfaces, which guarantees high operation speeds and precision with minimal mechanical wear and low maintenance requirements.

For the efficient applicability of the magnetic levitation technology, the robust and high-performance feedback control should be designed. Magnetic levitation system (MLS) is widely used in the laboratories for this purpose. Magnetic levitation phenomenon is based on the principle of electromagnetic suspension (EMS). The gravity force affecting the levitated ferromagnetic (made of nickel, cobalt, iron, etc.) body is balanced by the electromagnetic field produced by the electromagnet. The feedback loop serves to control the strength of the electromagnetic field and, thus, to control the air gap between the levitated body and the electromagnet.

MLS has a complex nonlinear dynamics, it is open-loop unstable and time-varying dynamical system, which includes parameter uncertainty [1], or parameter variation, for example, the inductance related electromagnetic parameter. Besides this, there is always a discrepancy between actual plant and its mathematical model, which comes from unknown or unpredictable external disturbances and some unmodeled dynamics. Thus, the design of a control law for achieving high control performance along with robustness is a very challenging task, for which the use of classical controllers (e.g., PID-controller) is not the best solution. In order to obtain control gains, it is necessary to perform linearization and long and tedious mathematical calculations. Another issue is that linear controllers can properly operate only in limited region, which depends on the determined operating point. The use of intelligent methods, such as artificial neural networks (ANNs), for acquiring a controller does not require long calculations, and the controller can be extended to a nonlinear region.

In recent decades, ANNs have become more and more popular. They are already broadly and successfully used for finding accurate and robust solutions for complex nonlinear problems in many fields of industry, medicine, security (image processing and recognition), banking and finances (forecasting), military, etc. ANNs have ability to learn and also to model complex nonlinear relationships between inputs and outputs of the system which is essentially important due to high complexity and nonlinearity of the real-world systems.

The following work is focused on the experimental magnetic levitation system with two electromagnets (MLS2EM), provided by INTECO Ltd., Krakow, Poland. The system is fully integrated with MATLAB/Simulink and operates in the real-time in MS Windows. The model consists of two electromagnets (upper and lower), the set of ferromagnetic objects (balls of different diameters and masses) to be levitated, position and current sensors, power interface, RTDAC4/USB measurement and control I/O board, a personal computer (Figure 1.1, [1]). The levitated object is suspended between two electromagnets and the equilibrium stage is maintained by the balance of the electromagnetic and the gravity forces. The lower electromagnet is used to bring the disturbance into the system (an external force excitation) or in addition to the gravity force (for example, to introduce mass disturbance). The ball position is controlled by adjusting the current through the electromagnets applying controlled voltage across the electromagnets' terminals.

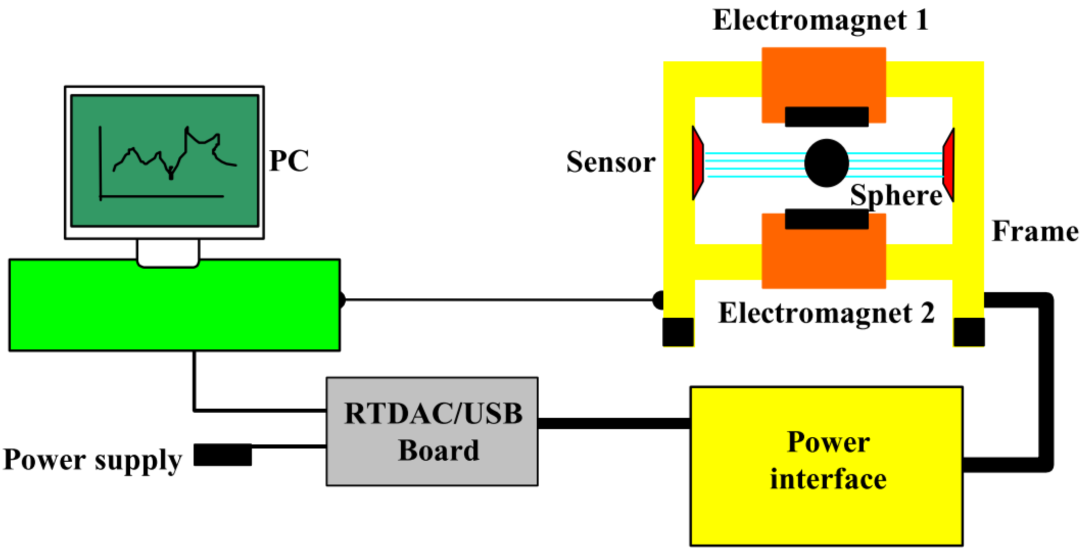


Figure 1.1 MLS2EM laboratory setup [1]

The main goal of the thesis is to design a control solution based on the ANNs for an accurate trajectory tracking of the levitated object and to achieve a better control and stability performance of the system and wider operating range in a presence of unpredictable disturbances and parameter uncertainty. During the design process of the controller, simulations and experiments, MATLAB/Simulink R2017b, R2020b with Deep

Learning (DL), Reinforcement Learning (RL) and Neural Network (NN) Toolboxes will be used.

The time-series datasets including object position will be collected and analysed first from the simulation and from the real-time plant. The ANN structure for the system's model representation will be designed and trained on the obtained datasets. Based on the parameters of ANN, the NN controller will be developed. The NN controller model will be implemented in Simulink model of the MLS2EM system. The various input signals and disturbances will be applied to analyse the control and stability performance. The results of the simulation experiments will be given and analysed for validation of the proposed control solution.

The Chapter 2 of the thesis presents a literature overview of the control solutions, existing to date in the field of the magnetic levitation. The short comparative analysis of the various methods and approaches, including hybrid methods, is given there. The magnetic levitation model is described in the Chapter 3. The NN structure and NARX-based controller are developed in the Chapter 4. The Reinforcement Learning based control solution is developed in the Chapter 5. The results of simulation experiments are discussed in the Chapter 6. Summary gives short conclusion of the thesis and its outcomes. Appendices include algorithms, MATLAB codes and plots of experiments.

2. LITERATURE OVERVIEW

2.1 Necessity of the intelligent control methods

A number of research papers on the problem of magnetic levitation control has been published during few recent years. A wide range of control solutions was proposed for the task of trajectory tracking in the MLS. Since the system is highly nonlinear and unstable, the studies on the robust control methods have become a subject of an intense interest.

The PID-controller, integrated in the MLS Simulink model, is a simple solution for the controlling position, but not the best one for such nonlinear and unstable system as MLS. The controller works using method based on the linearization of the system. The acquiring of the linearized model the MLS is described in details by P. Balko and D. Rosinova in the article [2]. This approach simplifies the process model and leads to the loss of some system's dynamics. This results in a steady-state error, overshoots, and oscillations. The main drawback of the PID-controller's use is that the system can be stabilized only in the controller's operation region, in other words, close to the determined operating point. The performance of the system fully depends on the fortunate selection of the PID-parameters. The instability of the MLS means that inappropriate PID-control unavoidably leads to the situation when the levitated object is either being dropped down or attracted to the upper electromagnet in the first seconds of the experiment. The PID-controllers cannot adapt to the abrupt disturbances, for example, change of the ball's mass. The narrow working range is clearly observed in the real system, but when one expands the movement range of the ball, the PID-controller cannot optimally control the ball anymore.

While the linear controllers are usually valid only around the operating point, the nonlinear controllers need exact knowledge about the plant nonlinearities to guarantee a good performance and stability. The practical implementations of the nonlinear controllers can be difficult due to the modeling and parameter uncertainties in the MLS. Thus, the intelligent control techniques should be used.

Among the proposed linear and nonlinear control strategies both for MLS1EM and MLS2EM (magnetic levitation systems with one and two electromagnets accordingly) are state feedback control and linear quadratic regulator algorithm, gain scheduling approach, backstepping, fuzzy logic control, sliding mode control, fractional-order controllers, predictive control, and neural network control. All these control solutions along with their benefits and drawbacks are discussed in more details in the next section.

2.2 Existing solutions

2.2.1 State feedback control

The state feedback control has been implemented in several research works: [3], [4], [5], [6], [7]. This method is based on the use of the state vector of the system for computation of the control action and the pole placement technique. This technique allows to compensate undesired system's response by placing its closed-loop poles in the complex plane (s-plane) and calculating the feedback matrix.

The state feedback control based on PID-controllers is discussed in [3]. Although, it is a low-cost solution, that can be easily implemented, it requires a lot of effort to derive a nonlinear mathematical model, to identify the parameters of equation through the experiment and to find a proper feedback gain matrix. The authors linearized the MLS model around seven operating points and designed seven PID-controllers. The state feedback control solution was included in the cascade structure with PID-controller in the outer loop to stabilize the system. The real-time experiments showed some oscillations in the beginning of transient response.

In [4] the authors combined PID and state-feedback controller to eliminate the steady-state error with the help of integral component. Two solutions, the state feedback and the cascade control, were proposed, but the stability analysis was not done.

The integral state feedback control has been developed in [5]. As it was done in [4], the integral component was added to eliminate the steady state error. The coefficient diagram method (CDM) was applied instead of trial-error method which is usually used to determine the controller's parameters. Varying the mass, the inductance and the resistance uncertainties during the experiments, the authors found out that the use of the robust parameters of CDM gives faster and more accurate response, while the standard parameters of CDM result in smaller rise time.

Inappropriate choice of PID-control gains may reduce the control performance and even damage the experimental system. The selection process is hard and time-consuming, and the control gains may require the big number of adjustments, done via trial-error method until the best performance is achieved. To simplify this process, B. Bidikli and A. Bayrak proposed a full-state feedback control with self-tuning rules in [6]. The robust controller was designed. Although, it showed effective work, the authors of the paper noticed that the control effort can be decreased with optimal or neural-network methods in future.

The nonlinear state feedback control is designed in [7]. Its main advantage over the linear state feedback control is capability to provide a faster system's response without or with minimum overshoot. The gradually changing feedback gain, introduced by the authors, affected the damping factor of the closed loop. This allowed to drive the overshoot to zero and achieve small rise and settling time in the experiment.

2.2.2 Linear quadratic regulator

The linear quadratic regulator (LQR) algorithm is one of the main results in the theory of optimal control. It serves to reduce optimization efforts on the controller, in other words, it is an automated way of finding state-feedback controller under specified design goals. This method still utilizes the state-space model of the system and requires determination of the cost functions. The difficulty consists in the finding of proper weighting factors, which eventually limits the application of the LQR.

The LQR controller is proposed in [8] as a better solution than the existing PID-controller in terms of better stability and larger work bandwidth. The experiments showed that the ball could follow the sinusoidal reference trajectory but with a lower amplitude and some small error.

Another LQR-PID controller is designed in [9]. Two controllers (LQR and PID) were combined to operate together in two loops for the task of stabilization of the MLS. Additionally, LQR-PID controller solved the overshoot problem, but the authors of the article admitted, that selection of error and control weightage matrices for the LQR control loop was a challenging task.

2.2.3 Gain-scheduling approach

The gain-scheduling approach is one of the most common in the control theory. It requires understanding of the plant's operating regions and determination of the related operating specifications. The procedure includes the linearization of the plant around each operating point and tunning of several controllers, i.e., obtaining a set of control gains for each operating region. It is obvious, that the procedure is time-consuming, since it demands several sets of control gains to be properly defined. The gain-scheduling method can guarantee adequate system's performance by switching between the operating regions and associated with them controllers, but the controllers still need to be robust.

The reference [10] presents three proportional-integral gain-scheduling control (PI-GS-C) solutions, based on the Lagrange interpolating parameter value method, Cauchy kernel distance metric and switching GS between PI linear controllers. During the experiments the authors concluded that all structures guarantee zero steady state error and satisfying trajectory tracking, although some oscillations happen in the beginning of the system's response.

2.2.4 Backstepping

The backstepping (BS) is a recursive technique that uses Lyapunov stability theory to derive a stabilizing control law for nonlinear dynamical systems. The inconvenience here is that there is no auto-tune tool for the nonlinear controllers in MATLAB/Simulink similar to one for PID-controllers, and, thus, researchers had to set and tune control gains manually via trial-error method and the system's response checks.

The backstepping and the integral backstepping (IBS) along with the synergetic control are discussed in [11]. The synergetic control involves macro variables which contain tracking errors of all states of the system. According to the results of the simulations, the IBS controller showed the best performance.

Adapting backstepping control method is proposed in [12]. The adaptation feature of the controller means its capability to adapt to the controlled system with uncertain or time-varying parameters. Usually, backstepping controller is added to reduce a negative effect of the slowly varying parameters of the system.

2.2.5 Fuzzy logic control

Fuzzy control logic is described in [13], [14], [15] and [16]. The parameters of fuzzy controller can be easily tuned, and it has a good potential to stabilize the ball levitation process. The fuzzy controller shows lower overshoot and no steady state error, comparing to the PID-controller, but its response is slower, and the settling time is longer [13]. The fuzzy logic design may include some nonlinearities due to the inference engine and some nonlinear methods applied in defuzzification process (when resulting fuzzy set, being converted to the number, is sent to the process as a control signal).

In [14] the authors utilized the programmable logic controller (PLC) with a liquid crystal display (LCD) panel for data acquisition and user control. They implemented fuzzy PID-controller and compared its performance with that one of the classical PID-controller. It appeared, that fuzzy controller works correctly for different set-points, unlike the classical PID-controller does. The authors point out the advantages of fuzzy controller in that it does not require the explicit model of the process, its control law is simpler and computational complexity is lower in comparison with other nonlinear controllers, for example, neural network-based ones.

Two cascade control system (CCS) structures using tensor-product based control (TP-C) and PI fuzzy logic control (PI-FC) were designed in [15]. The proposed solutions resulted in zero steady state control error and good rejection of the disturbances.

In master thesis work [16] the PD-controller was replaced by the fuzzy logic controller (FLC) for actual plant of MLS2EM. The smooth tracking control was achieved by increase of the number membership functions and rules. The developed FLC showed better steady-state error results comparing to the PD-controller.

2.2.6 Sliding mode control

The sliding mode control (SMC) is a nonlinear variable structure control method that alters the system’s dynamics by applying discontinuous control signal and driving the system’s states to the sliding surface. Among the advantages of the SMC are its robustness and finite-time convergence to the equilibrium point. The main drawback of the SMC consists in undesirable phenomenon of finite-frequency or finite-amplitude oscillations (so called “chattering”) caused by the high-frequency switching of a sliding mode controller due to the presence of some unmodeled dynamics in the closed-loop.

The chattering suppression methods are discussed by V. Utkin and H. Lee in [17]. The chattering problem appears in many SMC implementations, and without proper remedy it can become an obstacle to SMC application in the real-world plants.

The SMC solutions are developed in [18], [19], [20], [21]. Y. Eroğlu in the maser thesis [18] proposed several controllers' design strategies applying cascade control. The SMC was utilized for the inner electrical part of the control loop of MLS in order to obtain its fast response and better compensation of the negative effect of inductance related disturbances. It was found that SMC controllers keep the current error around zero but cannot fully eliminate the position steady-state error.

Simulation and implementation results of the SMC performance are compared in [19]. Robustness of the controller is tested applying external disturbance to the input signal.

The adaptive sliding mode controller (ASMC) with a sliding mode differentiator (SMD) presented in [20] allowed additionally to estimate the ball velocity needed for proper work of the controller and achieve the desired ball position and reference trajectory within only 1 s.

The real-time implementation of SMC–PID controller is done in [21].

2.2.7 Fractional-order controllers

The fractional calculus is applied in the field of control theory, called fractional-order control (FOC). This is a promising direction of the control studies and a good alternative to the classical control methods. FOC utilizes a fractional-order integral operator, that considers the whole history of its input signal, and capable to handle the chaotic behaviour of the complex mathematical models. The fractional-order controllers provide more adjustable system's response and also allow to reduce noise appearing in the control law. The control solution is based on the finding an operating point, linearization of the nonlinear system around operating point, determination of a stability region and stability margins and some parametric optimization.

Fractional-order controllers are discussed in [22], [23], [24], [25], [26], [27]. The stabilizing fractional-order PI$^{\lambda}$D controller, designed by W. Bauer and J. Baranowski in [22], reached the reference point in 1,5 s and managed with the task of disturbance rejection. The authors mentioned that there exists an issue of infinite memory for data storage and computations in the developed solution. For this reason, direct implementation of the fractional-order controllers for the real-time unstable systems may be difficult and require additional approximations.

Four novel fractional-order sliding mode controllers (static, integral static, dynamic and integral dynamic) are presented in [23] as a way to scale down the control effort and achieve robust and energy-efficient performance.

The two degrees-of-freedom (2-DOF) $PI^{\lambda}D^{\mu}$ controller has been designed in [24]. It showed good results in terms of stabilization, trajectory tracking and actuator saturation problem.

The authors of [25] designed 1-DOF and 2-DOF FOPID-controller for MLS and compared their performance with 1-DOF and 2-DOF integer-order PID-controllers (IOPID) in both simulation and real-time experiments. The results depicted the superiority of FOPID-controller over IOPID. The parameters of the controllers were obtained using the dominant pole placement method by optimizing the objective function.

The optimizing fractional PID (FOPID)-controller has been developed in [26]. Applying Ant Colony Optimization (ACO) algorithm and Ziegler Nicholas method, the authors of the article achieved reduction of the settling time and oscillations, but performance of the FOPID-controller with constraints, uncertainties and external disturbances, is to be analysed and improved in future.

A comparative analysis of SMC and FOSMC controllers is conducted by researchers in [27]. FOSMC showed better results, comparing with the SMC, on the basis of all main characteristics: zero overshoot, smaller settling time, the most accurate position tracking, reduction of chattering and lower control effort.

2.2.8 Model predictive control

Model predictive control (MPC) is an advanced method in the control theory. The main benefit of the MPC is ability to predict future response of the system as well as upcoming disturbances using its reference input. The MPC controller can adapt to the changes in the system parameters over the period of time, for example, due to mechanical wear of some parts of the system. The drawback of the MPC is the demand for the proper system’s model, and also the installation of the MPC controllers can be costly, because for optimization purposes it requires a computer.

The MPC is described in the articles [28], [29]. The performance of the model predictive controller was compared with that of classical PI and PID-controllers in [28]. The authors varied PI and PID-controllers’ gains to demonstrate that any change of those can result in oscillations, overshoot, large position error, etc. Meanwhile, the MPC controller easily adapted to the change of weighting matrices and did not show any overshoot or steady state error, and the settling time was minimal.

The real-time implementation of the MPC controller for the MLS with one electromagnet is presented in [29]. The authors compared the performance of the nonlinear feedback linearization (FL), linear MPC and conventional PID-controllers in this research. During the experiment both FL and MPC controllers showed reliable performance, while PID-controller failed to achieve stability when the reference position was changed. The best stability performance and the lowest settling time were reached by FL controller. The MPC controller restored stability with some oscillations.

2.2.9 Neural network control

The key benefit of using the ANNs is the reduction of the design effort related to the linearization of the system and corresponding calculations. The ANNs are capable to capture all system’s dynamics that cannot be formulated analytically, and the system’s stability is no more limited by the controller’s operation region.

The acceptability of the ANNs for the control of the MLS with one electromagnet is investigated in [30]. The multilayer feed forward NN with one input layer, one output layer and two hidden layers (with 20 and 10 neurons) was designed to learn the nonlinear behaviour of the system using the back propagation method. The training of the NN was realized using 30000 samples for inputs and outputs collected from the simulation with PD controller in MATLAB/Simulink environment. During the tests the NN controller did not perform properly during the stable position of the ball and the ball fell. This problem was resolved introducing two gains to amplify the input and error values and to make the NN react on the very small change. Square wave, sine wave and step input were applied during the experiments. The NN controller performed much better than the PID-controller and could efficiently and accurately follow the desired trajectory. For instance, the overshoot in case of the NN controller was almost 10 times lower than that of the PID-controller, and the settling time was around 5 times lower. On the other side, the PID-controller showed better values of rise time, delay time and peak time (Table 2.1). The reason is related to the back propagation method which suffers of slow convergence, being based on the gradient-descent algorithm.

Table 2.1 Transient analysis between PID and ANN controllers [30]

Parameter	PID-controller	ANN controller
Overshoot, %	27	2,8
Settling time, s	5,71	1,12
Rise time, s	0,1	0,46
Delay time, s	0,05	0,36
Peak time, s	0,22	0,48

The reference [31] proposes adaptive sliding mode controller based on the radial basis function neural network (RBFNNSMC) as faster and more robust one, comparing with the SMC and backstepping controllers (Table 2.2). The stability of the closed-loop system was proved by using the Lyapunov stability theory.

Table 2.2 Transient analysis between RBFNNSMC, SMC, BSC controllers [31]

Parameter	RBFNNSMC	SMC	BSC
Overshoot, m	$1{,}5 \times 10^{-6}$	$1{,}45 \times 10^{-5}$	$6{,}9 \times 10^{-5}$
Settling time, s	0,67	>1	>1
Response speed	quick	middle	slow

Comparative study on the performance of radial basis function neural network (RBFNN), multilayer feedforward neural network (MLFFNN) and a recurrent nonlinear auto-regressive with exogenous inputs (NARX) neural network was conducted in [32]. The authors combined back propagation and Lyapunov stability methods to speed up the convergence rate and successfully developed a novel adaptive learning rate for the neural network. Based on the results of simulations, the authors concluded that RBFNN-based controller performed much better than MLFFNN- and NARX-based controllers in regard to the computational time, average mean squared error (MSE), robustness and simplicity (the number of parameters to be trained), which makes the RBFNN an efficient tool for control operations (Table 2.3).

Table 2.3 Average MSE and run-time of MLFFNN-, RBFN- and NARX-based controllers [32]

Parameter	MLFFNN	RBFNN	NARX
Average MSE (10 000 iterations)	$9{,}13 \cdot 10^{-2}$	$2{,}28 \cdot 10^{-2}$	$4{,}91 \cdot 10^{-3}$
Simulation time, s	4,722920	0,335487	4,738064

The NN is utilized in [33] for approximation of the electromagnetic parameter for better compensation by the controller. The nonlinear NN controller with the novel control law was designed and compared with the two stages (or two layers) controller. The two stages controller also utilizes the NN (with the sliding mode) for the trajectory tracking but does not estimate the electromagnetic parameter and does not consider the angular position of the ball. Based on the value of the root mean squared error (RMSE), the researchers concluded that the designed controller had improved the performance of the two stages controller. The authors suggested that further development could be done for approximation of other parameters: non-modelled dynamics and eddy currents (swirls).

In [34] the authors added the adaptive NN controller learning online in a real-time to the inner linearization loop, keeping the PD controller in the outer loop. This resulted in the stable response of the MLS and good adaptation to the abrupt changes of the levitated object's mass, which was realized by excitation of the lower electromagnet of the MLS2EM. The researchers concluded that the ANN is a proper approximator of the MLS nonlinearities.

The article [35] offers a multi-loop Model Reference Adaptive fractional-order proportional integral derivative (MRAC-FOPID) control structures with NARX model as a reference model. The authors noticed that, aside from the presented unpredictable disturbances in the MLS, there is a tradeoff between the position tracking and the disturbance rejection control. To resolve this dilemma, the authors used the FOPID for the improved stability and set-point tracking and the MRAC for improvement of the disturbance rejection. The Massachusetts Institute of Technology (MIT) rule [35] is used in the outer loop for the MRAC process. The comparison of performance parameters of FOPID, MRAC-FOPID and PID with NARX reference model is given in Table 2.4.

Table 2.4 Transient analysis between FOPID, MRAC-FOPID and PID with NARX reference model [35]

Parameter	**FOPID control**	**MRAC-FOPID**	**PID with NARX reference model**
Peak values, m	3,603 × 10^{-3}	2,021 × 10^{-3}	5,552 × 10^{-3}
Settling time after step disturbance, s	1,23	0,43	1,08

For stabilization purpose, in [36] the real experimental data was collected. Using the NN, trained on this data, the authors designed the velocity observer for the MLS.

2.2.10 Deep reinforcement learning

Deep Reinforcement Learning (DRL) is a branch of machine learning. It is applied for the difficult control problems, including highly unstable systems, such as MLS.

Unlike the supervised learning, where input-output datasets are given to the neural network to learn the desired behaviour, the learning process in DRL is based on the direct interaction with the environment. The Reinforcement Learning (RL) agent learns the behaviour, adjusting its parameters throughout the number of episodes, or runs of the experiment. It evaluates its own performance and gets the certain reward after each episode. The RL agent is capable to explore the space of its possible actions and accumulate the experience over the time which makes possible finding the most optimal solution for the specified task.

The following control solutions in RL were proposed by the researchers recently for the MLS.

The stable control for MLS with one electromagnet was realized by T. Huang, Y. Liang and X. Ban in [37] using improved Q-Network method (model-free reinforcement learning method). The proper choice of the reward function plays an important role in RL concept. The authors introduced 3-component reward function (2.1), that includes distance reward, velocity reward and direction reward, using position, velocity and current as observations:

$$\begin{gathered} reward_1 = \frac{d_{max} - |d|}{d + d_0}, \\ reward_2 = \frac{v_{max} - |v|}{v_{max}}, \\ reward_3 = \left|\frac{d}{|d|} - \frac{v}{|v|}\right| - 1, \\ r\ = 3{,}2 \times reward_1 + reward_2 + 0{,}3 \times reward_3, \end{gathered} \tag{2.1}$$

where

d_0 — target position of the ball, m,

d_{max} — maximum allowed deviation from target position, m,

d — real-time distance between actual and target position (error), m,

v_{max} — maximum allowable velocity of the ball, m/s,

v — real-time velocity, m/s,

r — total reward.

The researchers noted, that since RL includes exploration part, this leads to continuous oscillations near the desired position. This means, there is always a large steady-state error, and the performance of Q-network controller suffers from the low fitting accuracy. To resolve this issue, the authors proposed the network retraining algorithm in order to improve the accuracy of the controller. The retraining algorithm is based on the idea of adaptive adjusting of the exploration rate. Namely, the authors adopted ε greedy strategy, where $\varepsilon \in [0,1]$ is the action exploration probability, in such way that exploration rate was decreasing with the progress of the learning process. It is important to note, that the exploration probability allows system to jump out of the local optimal state and continue to explore the action space until the better solution is found.

The training process was split into conditional cycles. After achieving the desired number of "good-steps" (that keep error value within the allowable range), the algorithm sets exploration rate to zero, since there is no need of further exploration. This approach allowed to effectively reduce the oscillations and, subsequently, a steady-state error of the designed Q-network controller by an order of magnitude. The results of simulations proved the reasonableness and significance of use of the proposed retraining algorithm even in presence of the noise.

A self-learning controller for MLS1EM was developed in [38]. The authors designed the Dueling-Double-Deep Q-Network (Dueling-DDQN) and compared it with the Deep Q-network (DQN) controller in simulation. The Dueling-DDQN method has faster convergence, and it was shown that Dueling-DDQN controller can provide more stable control and has a larger attraction domain rather than DQN controller. The benefit of use of Dueling-DDQN algorithm is that it has a double network structure, and the action value function is separated into two parts: a state value function and an advantage function, that is related both to state and action spaces. These improvements help to solve the overestimation issue of the state value that traditional DQN suffers from.

The reward function was chosen to fasten convergence of the training process in the following view (2.2):

$$\begin{cases} r_1 = \tan\left(\frac{\theta_1 - \theta_0}{|e_{max}|}|e| + \theta_0\right), \\ r_2 = \max(sign(ev)|v|, 0), \\ r \ = l_1 r_1 + l_2 r_2 , \end{cases} \tag{2.2}$$

where

e — error between expected and measured position values, mm,

v — velocity of the ball, m/s,

r_1 — reward for the position of the ball, which is more sensitive for smaller $|e|$,

r_2 — reward for the velocity of the ball, $r_2 \geq 0$, (the error and the velocity signs should be the same, otherwise $r_2 = 0$),

r — total reward,

θ_0, θ_1 — angles, $Q_0 = 20°, Q_1 = 89°,$ and

l_1, l_2 — constants, $l_1 = 10, l_2 = 15,$ picked after a serial of experiments.

The training process was split into nine cycles, each of them consists of 300 episodes. The exploration of the action space was organised using ϵ-greedy policy depending on the number of the cycle. The RL-agent was being given the biggest exploration probability in the beginning of the learning. With the growing number of the cycle, the exploration probability was being decreased, and eventually set equal to 0,1 staring from the 5th cycle. This was done for adequate exploration of the action space by the RL-agent with the passing time and for decrease noise after achieving intermediate results. Besides this, the authors introduced stopping criteria for episode, namely, exceeding of the ball's lower and upper position limits and the simulation duration of 4 s.

The most valuable and frequently cited work [39] in the field of DRL gives an adaptation of the ideas of the Q-learning method to the continuous action domain. The authors of the article state that DQN learning methods are not relevant for application to the physical control tasks with complex dynamics, which have continuous and high-dimensional action spaces. Moreover, discretization of the action space could be an obvious solution in adapting DQN learning methods to continuous domains, but it also has its limitations. One of the biggest issues here is that the number of actions grows exponentially with the number of degrees of freedom of the system. The difficulty of efficient manipulation with the large, discretised action spaces and also the fact that discretization leads to the loss of information about the structure of the action space, makes application of the DQN learning methods unreasonable.

In the paper, authors introduced the robust model-free, off-policy actor-critic approach based on the deterministic policy gradient (DPG) algorithm. The concept of actor and critic NNs will be explained and used later in Chapter 5. The authors combined this approach with ideas borrowed from Deep Q Network (DQN) learning method by training the network off-policy using samples from so-called "replay buffer" and introducing target Q-network to provide "consistent targets during temporal backups".

The introduced method, named by the authors as "Deep deterministic policy gradient" (DDPG), was applied to more than 20 simulated physics tasks of various levels of difficulty, including classic problems such as cartpole swing-up, moving gripper, legged locomotion and car driving, etc. The approach benefit consists in its simplicity and easy implementation. The authors showed that DDPG is able to learn good policies and, eventually, provide a robust control. Being applied to the hardest problems, DDPG algorithm even surpassed the performance of the Q-learning, which tends to overestimate values of the Q-function (action-value), using function approximators.

The authors introduced Ornstein-Uhlenbeck (OU) noise process $\mathcal{N}$ for better action space exploration, which will be discussed in Chapter 5, and applied batch normalization at each layer of the NNs. The DDPG approach still has its limitations. One of them is a large number of training episodes required for finding a solution. The DDPG algorithm, developed in [39], is used in the current work, and given in Appendix 2. The authors used Adam optimizer for learning NN parameters. The structure of the NNs and information on the training settings, used in [39], are given in Table 2.5.

Table 2.5 NNs structure and training settings for DDPG agent, [39]

Parameter	**Value**
Learning rate for actor	10^{-4}
Learning rate for critic	10^{-3}
L_2 weight decay	10^{-2}
Discount factor, γ	0,99
Target smooth factor, τ	10^{-3}
Activation functions for all hidden layers of actor and critic	ReLU
Actor output activation function	Tanh
Number of hidden layers	2
Hidden layers size, neurons	400, 300
Weights and biases initialization range at actor and critic final layers, uniform distribution	$[-3\cdot10^{-3}, 3\cdot10^{-3}]$
Minibatch size	64
Replay buffer size / Experience buffer length	10^6
Standard deviation of OU noise, σ	0,2
Mean attraction of OU noise, θ	0,15

In [40] S. Wongsa and N. Kowkasai implemented RL algorithm in continuous control using deep deterministic policy gradient (DDPG) for Magnetic Levitation Model CE 152 with one electromagnet developed by Humusoft company. The authors have focused on the importance of the reward function and its role for the training outcomes and learning the optimal policy. In the article they considered five reward functions and compared the training outcomes.

As a successful result of the training, the controlled variable, which is position of the ball in case of MLS, must follow the desired trajectory. For this, the neural networks parameters are forced to change in a certain way by setting the appropriately chosen reward function. When the process output gets closer to the reference value, or in other words, when the position error remains in the error tolerance range, the more reward is given to RL agent. And on the opposite, when the process output goes beyond the error tolerance range limits, or even falls out of the pre-set system limitations for controlled variables, the more penalty (or negative reward) system receives. It is obviously, that proper reward function can sufficiently improve the training process results and its convergence speed.

The authors considered two types of reward functions $R1$ and $R2$, (2.3), (2.4):

$$R1 = \begin{cases} c, & if\ |e| \leq \varepsilon, \\ -|e|, & otherwise, \end{cases} \tag{2.3}$$

$$R2 = exp(-|e|) - 1. \tag{2.4}$$

where

e — position error, m,

c — constant, $c > 0$,

ε — error tolerance, m, $\varepsilon > 0$.

The authors noted that tuning of the hyper-parameters (constants c and ε) for the reward function of a common view $R1$ can be challenging and tedious. For this reason, they proposed an alternative simple reward function $R2$ which depends only on the calculated error value and does not require manual tuning of the constants.

The RL training was implemented by the authors using open-source OpenAI Gym interface in Python, rendered environment for visualization of behaviour, and Tensor Processing Unit (TPU) available in the Google Colab for parallel training and running of deep RL process. The structure of the networks and the training settings are shown in Table 2.6. The Adam optimizer and batch normalization was used similarly to [39].

Table 2.6 NNs structure and training settings for DDPG agent, [40]

Parameter	**Value**
Learning rate for actor	10^{-4}
Learning rate for critic	10^{-3}
Discount factor, γ	0,99
Target smooth factor, τ	10^{-3}
Activation functions for all hidden layers of actor and critic	ReLU
Number of hidden layers	2

Table 2.6 continued

Parameter	Value
Hidden layers size, neurons	64
Minibatch size	128
Replay buffer size / Experience buffer length	$5 \cdot 10^4$
Standard deviation of OU noise	$1 \cdot 10^{-2}$

For the better exploration of the action space the authors have added OU noise not to the action space, as it was done in [39], but directly to the parameters of the neural network policy. The efficiency of this approach was shown in [41].

The training and tests were conducted using staircase function as a setpoint signal. The outcomes of the trainings were compared for five reward functions (Table 2.7) using integral absolute error (IAE), calculated for the position of the ball (2.5):

$$IAE = \int_0^T |e_t| dt, \tag{2.5}$$

where

e_t — error at time step t, m,

T — experiment length, s.

Table 2.7 Comparison of closed-loop performance of DDPG-based controllers using different reward functions [40]

Reward function	IAE	Settling time, s
R1 ($c = 0{,}1,\ \varepsilon = 0{,}001$)	0,049	0,041
R1 ($c = 0{,}1,\ \varepsilon = 0{,}01$)	0,655	0,095
R1 ($c = 0{,}5,\ \varepsilon = 0{,}001$)	0,121	0,070
R1 ($c = 0{,}5,\ \varepsilon = 0{,}01$)	0,523	0,069
R2	0,086	0,043

The authors concluded that the performance using reward function $R2$ can compete with that using the best tuned reward function $R1$, and the proposed reward function $R2$ can be easily applied to any setpoint tracking problem. Moreover, no obvious correlation between c and ε constants' values and the convergence rate of the learning was found. However, it was found that the smaller value of error tolerance ε requires more episodes for training to get an optimal solution. The experiments showed that DDPG-based controllers can provide stable and accurate setpoint tracking over the full operating points.

One of the most recent works [42] is devoted to another control problem, namely, to the control of nonlinear valves using DDPG algorithm of RL. This article is of great interest because the RL process was implemented by R. Siraskar in MATLAB/Simulink environment with recently launched Reinforcement Learning Toolbox™, while the majority of works utilize Python and OpenAI Gym environments. The Reinforcement Learning (RL) Toolbox for the first time was introduced in MATLAB R2019a version and since that time has been developing and improving. The current thesis work will also exploit RL Toolbox for the control of MLS.

In the Simulink model the author of the article used Reinforcement Learning Agent block, provided within the RL Toolbox, and created an observation vector, reward function and stopping criteria in a similar way, as it was done in the Reinforcement Learning Toolbox™ User's Guide [42, pp.1-19 - 1-26] by MathWorks® for classic Water tank Simulink model. The training of the DDPG agent is realized, by guess, using similar MATLAB code with Reset function, provided in RL User's Guide. The Reset function makes possible to vary the setpoint value (for constant setpoints) in the beginning of each episode during the training and reset the environment. Additionally, the author randomized initial value of the controlled variable (flow value in the case of valve control problem), as it was also done in the given example. This method allows to find more flexible optimal policy, which can provide a better control.

The observation vector consists of the measured signal value (actual flow), error with respect to the reference signal and the integral of the error. The latter one provides a mechanism that computes the total error over the time and drives DDPG agent to decrease it. The reward function was created in a hybrid form based on (2.6) and (2.7):

$$Reward = \begin{cases} 10, & if\ |e| < \varepsilon, \\ -1, & if\ |e| \geq \varepsilon, \\ -100, & if\ (y \leq 0, y > Max\_Flow) \end{cases} \tag{2.6}$$

where ε — allowable error margin, m, $\varepsilon > 0$.

$$Reward = \begin{cases} -100, & if\ (y \leq 0, y > Max\_Flow) \\ \dfrac{1}{e+\lambda}, & otherwise \end{cases} \tag{2.7}$$

where λ — a small constant preventing division by zero-error.

The new reward function included reward and penalty parts (2.8):

$$Reward = \begin{cases} 0{,}1, & if\ |e| \leq 0{,}1, \\ |e|, & if\ |e| > 0{,}1, \\ p, & if\ (y \leq 0, y > Max\_Flow) \end{cases} \tag{2.8}$$

where p — penalty, which value was not specified in [42].

The structure of the NNs and the training parameters are shown in Table 2.8.

Table 2.8 NNs structure and training settings for DDPG agent, [42]

Parameter	Value
Learning rate for actor	10^{-4}
Learning rate for critic	10^{-3}
Discount factor, γ	0,9
Activation functions for all hidden layers of actor and critic	ReLU
Actor output activation function	Tanh
Actor hidden layers number	1
Actor hidden layers size, neurons	25
Critic hidden layers size, neurons	50, 25
Action path size, neurons	25
Minibatch size	64
Standard deviation / Variance of OUP noise	1,5
Variance decay rate	10^{5}

The authors elaborated on "Graded Learning", a progressive coaching method, which is a form of "Curriculum learning" method. The Graded learning helped to avoid long trainings that can consist of thousands of episodes and last many hours. Instead of one long-term training, the RL task was intuitively broken down into levels, where each of them puts forward training criteria with increasing difficulty. The agent was trained for n episodes or until the training criteria is met. Once the level of task is learned, the agent is retrained at the next level and the new experience is built upon. So-called "transfer-learning" technique was used to transfer leaned NN's weights from one task to another throughout 6 stages of increasing difficulty. The analysis of the results showed that Graded Learning is effective way to coach the RL agent.

2.3 Literature overview conclusions

The enormous number of the control methods has been described in the research papers, devoted to the problem of controlling the MLS and the accurate and stable trajectory tracking.

The linear control methods are not suitable for the highly nonlinear and unstable MLS:

- All the assumptions and simplifications, regarding modeling, result in a simplified model and weaker control.
- The state feedback control, the LQR and the gain-scheduling approach require long and tedious calculations in order to linearize the process model and to find a feedback matrix.
- Tuning of one or several PID-controllers, in case of the gain-scheduling approach, is done through the trial-error method, which can be hard and very time-consuming.
- The linear controllers have a very narrow operating region.
- The high control performance, and especially robustness of the controller, are not guaranteed. For this reason, additional methods and hybrid techniques are used.

The nonlinear and advanced control methods require complex mathematical analysis and long and tedious calculations in order to obtain an appropriate control law.

- Cascade control, implemented in the number of research works, can improve simple classical controllers' performance only to some extent, and require various stabilizing controllers in additional loop.
- The backstepping, the fuzzy logic control and the sliding mode control provide better stability of the system than linear control methods. While the backstepping requires manual tuning of the control parameters, the fuzzy controllers are easily tuned, but have slow response and long settling time. The sliding mode controllers have a stabilizing feature and provide fast response but suffer from the chattering problem and additionally require optimization.
- The fractional-order control has a great potential in regard to the accuracy, but not the uncertainties, and there exists a problem of infinite memory for data storage and computations.
- The model predictive control is used for its adaptation ability and prediction of the future system's response, but it also demands a proper system's model.
- While the classical controllers, such as PID, are cheap and affordable, the implementation of the advanced controllers, such as MPC, may be costly and require additional setups.

The amount of effort to be made in order to derive an appropriate control law and develop a controller, depends on the complexity of the model in hand and the design goals and specifications. It is clear, that application of the intelligent control methods, such as ANNs, can significantly reduce the design effort.

- The controller, based on the ANNs, can consider uncertainties that are highly difficult to identify and predict in the mathematical modeling, and thus, guarantee the higher control and stability performance of MLS.
- The NN-based controller's design process does not involve mathematical analysis of the model, thus, requires less effort.
- Additionally, the NN-based controller will have wider operating region, and make various reference signals possible to be applied for the levitated object.

2.4 Aims of the thesis

The main aim of the thesis is to develop a control solution based on the ANNs for an accurate trajectory tracking of the levitated object in the MLS2EM, as well as to achieve a higher control and stability performance of the system in a presence of unpredictable disturbances and parameter uncertainty. The NN-based controller will have wider operating range in comparison to the PID-controller, which operating region is around $2{\cdot}10^{-3}$ m.

To accomplish that, the following sub-goals have been outlined:

- Collect and analyse the time-series datasets including object's position first from the simulation model and then from the actual plant of MLS2EM;
- Develop an appropriate NN structure and train the NN on the collected datasets;
- Design the NN controller;
- Test the NN controller in Simulation and compare it with the existing PD-controller;
- Make the necessary adjustments in the NN structure or control solution based on the tests results;
- Validate the control and stability performance of the MLS2EM applying various reference signals and introducing disturbances;
- Evaluate the results.

3. MAGNETIC LEVITAION SYSTEM AND ITS MODELING

3.1 System description and modeling

3.1.1 Mechanical part and software

Magnetic levitation phenomenon is based on the principle of electromagnetic suspension (EMS). As mentioned in Chapter 1, the magnetic levitation technology makes frictionless motion possible, and therefore can provide operations with high speed and precision and reduce the mechanical wear of the equipment.

The laboratory setup of the magnetic levitation system with two electromagnets (MLS2EM) is schematically shown in Figure 1.1. The mechanical unit of MLS2EM consists of the aluminum frame, two electromagnets attached to it (upper EM1 and lower EM2), an optical detector to sense the object position, and coils current sensors, and three ferromagnetic spheres of different weights and sizes (Table 3.1). The hardware is accompanied with the power supply and interface to a personal computer (PC) and the dedicated RTDAC/USB measurement and control input-output (I/O) board in the Xilinx® technology. The software operates in real time under MS Windows® using MATLAB®/Simulink R2017b and the Real Time Windows (RTW) Toolbox (currently renamed to Simulink Desktop Real-Time Toolbox) for building a real-time model [1]. Additionally, to the control software, the MLS2EM Toolbox is provided by INTECO in order to solve modeling, design and control problems for MLS2EM in MATLAB environment.

Table 3.1 Ferromagnetic objects' parameters

Ball number	Ball size	Ball mass, kg	Ball diameter, m
1	small	$1{,}91\cdot10^{-2}$	$3{,}8\cdot10^{-2}$
2	medium	$3{,}76\cdot10^{-2}$	$5{,}6\cdot10^{-2}$
3	big	$4{,}71\cdot10^{-2}$	$6{,}4\cdot10^{-2}$

The MLS2EM control window is opened by the MATLAB command "mls2em_usb2_main" and shown in Figure 3.1. Through it one can rapidly access all basic functions of the MLS: testing tools, drivers, models and demo applications [1].

Identification of the MLS2EM is done in four steps to verify or modify static and dynamic characteristics of the system. During identification process position sensor characteristics, static and dynamic features of actuators (electromagnets) are identified. The minimal control analysis is conducted to define the minimal control (or minimal applied voltage) required to cause the motion of the levitated object from the lower electromagnet towards the upper electromagnet against the gravity force.

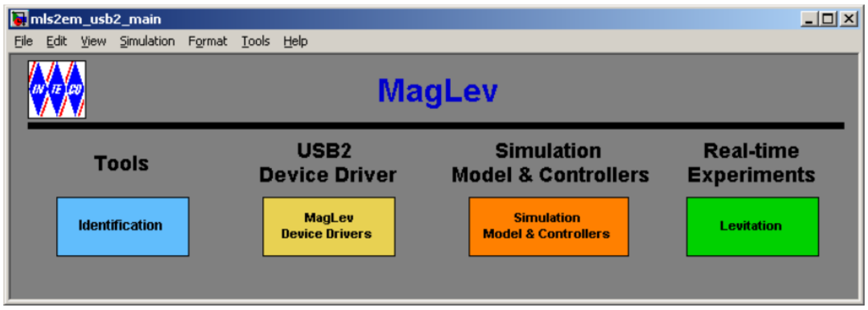


Figure 3.1 The Magnetic Levitation Main window

The MagLev device driver is a software go-between for the real-time MATLAB environment and the RT-DAC4/USB acquisition board [1]. The interior of the driver block is given in [1].

The INTECO provided three simulation models along with pre-tuned controllers (Table 3.2, models 0-2, Figure 3.2), which include Magnetic Levitation Animation block (Figure 3.3), and three pre-tuned experimental controllers (Table 3.2, models 3-5). The controllers will be discussed in Section 3.1.3.

Table 3.2 Existing simulation and experimental models of MLS2EM

Controller/ Model number	Model name	Sample time, s
-/0	Open loop (sim.)	$1\cdot10^{-3}$
1	PD (sim.)	$1\cdot10^{-3}$
2	PD differential mode (sim.)	$1\cdot10^{-3}$
3	PD EM1 (real-time)	$5\cdot10^{-3}$
4	PD EM1, EM2 pulse excitation (real-time)	$5\cdot10^{-3}$
5	PD differential mode (real-time)	$5\cdot10^{-3}$

The "Magnetic Levitation model (MLS2EM)" block has two inputs: EM1 and EM2 control actions, and six outputs, which go to the Scope "MagLev – model Control and States" as four channels:

- Position, m,
- Velocity, m/s,
- EM1 and EM2 currents, A,
- EM1 and EM2 controls, pulse width modulation (PWM) duty 0-1.

The MLS model will be described in more details in the next Section 3.1.2.

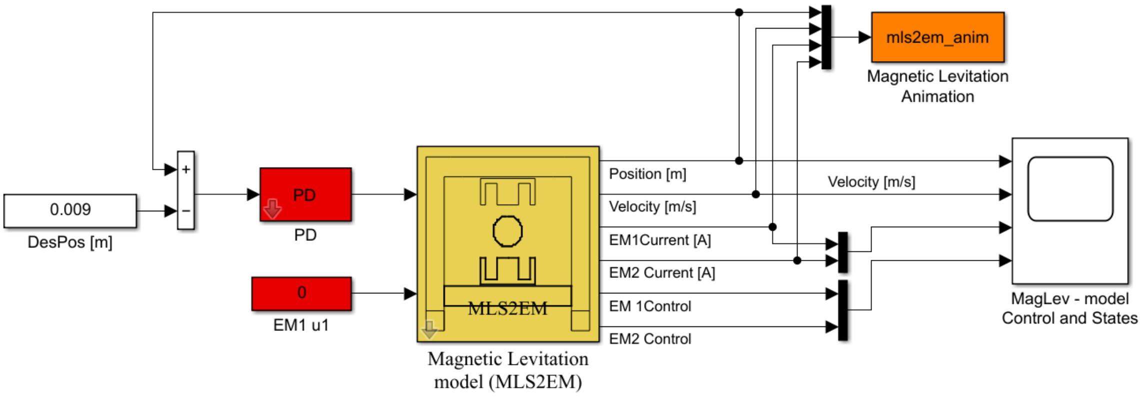


Figure 3.2 Simulation model "PD" (model 1) with Animation block

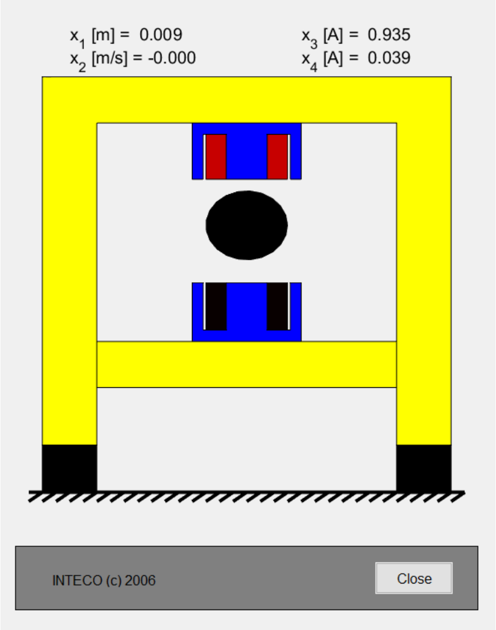


Figure 3.3 Visualization of magnetic levitation using Animation block

3.1.2 System modeling

The nonlinear state space model of the MLS2EM is given by the set of equations (3.1)-(3.2), [1], and represented in the diagram in Figure 3.4:

$$\begin{aligned} \dot{x}_1 &= x_2, \\ \dot{x}_2 &= -\frac{F_{em1}}{m} + g + \frac{F_{em2}}{m}, \\ \dot{x}_3 &= \frac{1}{f_i(x_1)}(k_i u_1 + c_i - x_3), \\ \dot{x}_4 &= \frac{1}{f_i(x_d - x_1)}(k_i u_2 + c_i - x_4), \end{aligned} \qquad (3.1)$$

where

$$F_{em1} = x_3^2 \frac{F_{emP1}}{F_{emP2}} \exp\left(-\frac{x_1}{F_{emP2}}\right),$$
$$F_{em2} = x_4^2 \frac{F_{emP1}}{F_{emP2}} \exp\left(-\frac{x_d - x_1}{F_{emP2}}\right), \qquad (3.2)$$
$$f_i(x_1) = \frac{f_{iP1}}{f_{iP2}} exp\left(-\frac{x_1}{f_{iP2}}\right), \qquad for\ both\ actuators,$$

where

x_1 — position of the ball, m, $x_1 \epsilon [0, 0{,}02]$, measured downwards from EM1 to the highest point of the levitated ball,

x_2 — velocity of the ball, m·s, $x_2 \epsilon \mathbb{R}$,

x_3 — current in the coil EM1, A, $x_3 \in [i_{MIN}, i_{MAX}]$,

x_4 — current in the coil EM2, A, $x_4 \in [i_{MIN}, i_{MAX}]$,

u_1 — control for the EM2, $u_1 \epsilon [u_{MIN}, u_{MAX}]$,

u_2 — control for the EM2, $u_2 \epsilon [u_{MIN}, u_{MAX}]$.

The parameters of the equations (3.1)-(3.2) are described in Table 3.3 below.

Table 3.3 Parameters of the equations (3.1)-(3.2)

Symbol	Description	Value	Unit
m	ball mass	$5{,}71 \cdot 10^{-2}$	kg
d	ball diameter	$6 \cdot 10^{-2}$	m
g	gravity acceleration	9,81	$m \cdot s^{-2}$
F_{em1}, F_{em2}	electromagnetic force	functions of x_1 and x_3	N
F_{emP1}	electromagnetic force parameter	$1{,}7521 \cdot 10^{-2}$	H
F_{emP2}	electromagnetic force parameter	$5{,}8231 \cdot 10^{-3}$	m
$f_i(x_1)$	actuator parameters	function of x_1	$1 \cdot s^{-1}$
f_{iP1}	actuator parameters	$1{,}4142 \cdot 10^{-4}$	m·s
f_{iP2}	actuator parameters	$4{,}5626 \cdot 10^{-3}$	m
c_i	actuator parameters	$2{,}43 \cdot 10^{-2}$	A
k_i	actuator parameters	2,5165	A
x_d	distance between electromagnets minus ball diameter	0,75-d	m
i_{MIN}	minimum current value	$3{,}884 \cdot 10^{-2}$	A
i_{MAX}	maximum current value	2,38	A
u_{MIN}	minimal control for levitation	$4{,}98 \cdot 10^{-3}$	-
u_{MAX}	maximal control	1	-

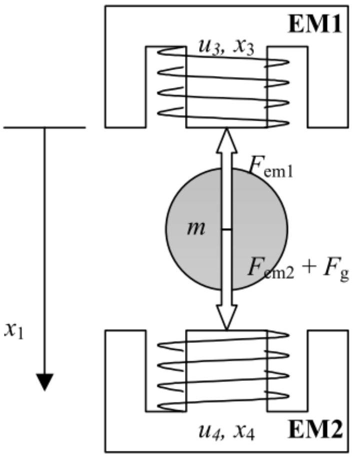


Figure 3.4 MLS2EM diagram [1]

The interior of the “Magnetic Levitation model (MLS2EM)” block is shown in Figure 3.5-Figure 3.7. The distance between the electromagnets EM1 and EM2 is denoted by EMsDistance, and equals to $7{,}5 \cdot 10^{-2}$ m. The result of subtraction $x_b = x_d - x_1$ gives the distance between the lowest point of the levitated ball and the bottom electromagnet EM2.

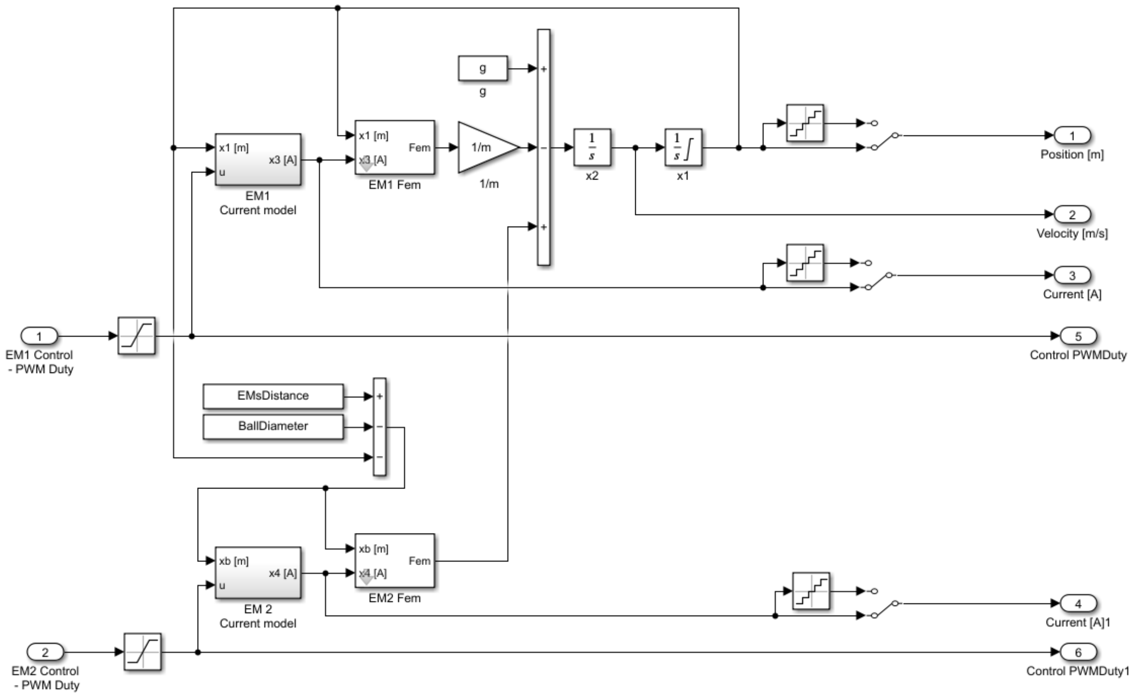


Figure 3.5. Interior of the MLS2EM model

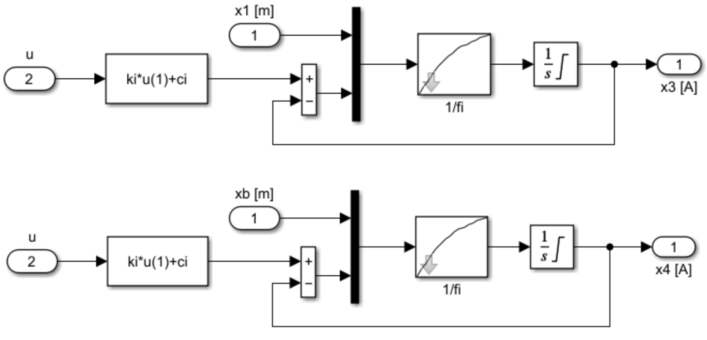


Figure 3.6 Interior of the EM1 and EM2 current models

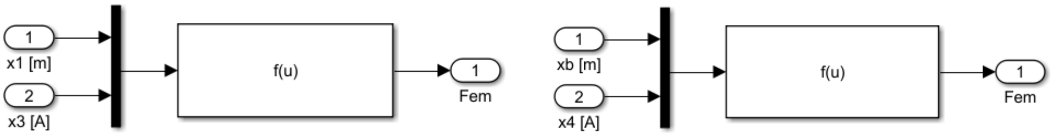


Figure 3.7 Interior of the EM1 Fem and EM2 Fem blocks

The function $f(u)$ corresponds to the expressions for F_{em1}, F_{em2} from the equation (3.2). The system has two inputs: the controls u_1 and u_2, saturated in the limits $[u_{MIN}, 1]$, and six outputs: the position of the ball x_1, the velocity of the ball x_2, the currents x_3 and x_4 in the coils EM1 and EM2, respectively, and the controls PWM duty.

The position of the ball is detected by the position sensor and measured downwards from the bottom of the upper electromagnet EM1. Additionally, control PWM (pulse width modulation) duty cycle is shown for the EM1 and EM2 in the scope (Figure 3.5).

All variables x_1, x_2, x_3, x_4 of the equation (3.1) come together as a state of the system $[x_1, x_2, x_3, x_4]$, which is being changed at every time step t, $t \geq 0$. The initial state of the system is given at time $t = 0$ as follows (3.3):

$$\begin{gathered} x_1(0) = 0{,}009, \\ x_2(0) = 0, \\ x_3(0) = 0{,}9, \\ x_4(0) = 0{,}04. \end{gathered} \tag{3.3}$$

3.1.3 Existing controllers

The control of the ball position in MLS2EM is performed through the feedback control loop using classic PID-controller with zero integral component. The control law of a classic PID-controller includes proportional, integral, and derivative terms, and is given by the well-known equations (3.4)-(3.5):

$$u(t) = K_p e(t) + K_i \int_0^t e(t)\,dt + K_d \frac{de(t)}{dt}, \tag{3.4}$$

$$e(t) = r(t) - x(t), \tag{3.5}$$

where

$r(t)$ — desired process value, or setpoint, SP,

$x(t)$ — measured process value, PV,

$e(t)$ — error value.

As it has been already mentioned in Section 3.1.1, the INTECO provided six Simulink models: three simulation models and three experimental models (Table 3.2). The model 0, "open loop" does not exploit any controller. The rest of the models 1-5 include five pre-tuned PD-controllers.

The interior of the PD and differential mode PD-controllers is shown in Figure 3.8 and Figure 3.9, and described by the equations (3.6) and (3.7), respectively. The integral term $K_i \int_0^t e(t)\,dt$ is replaced by the steady-state control constant u_0, which is tuned for each controller separately. The parameters of the five controllers are given in Table 3.4.

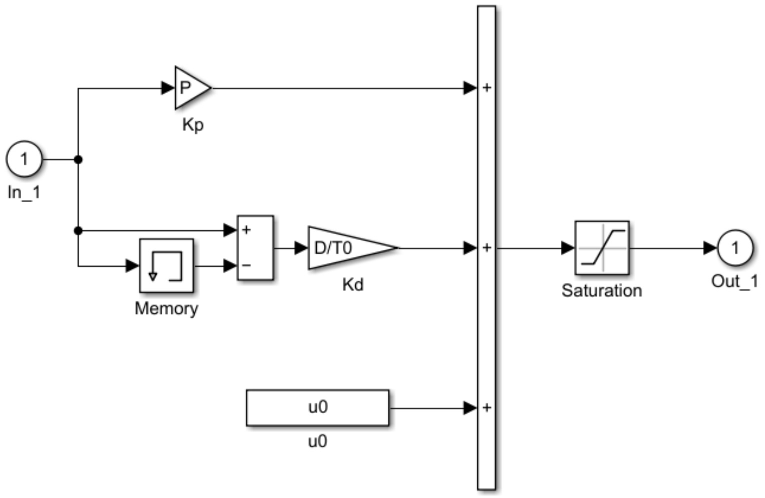


Figure 3.8 Interior of the PD-controller

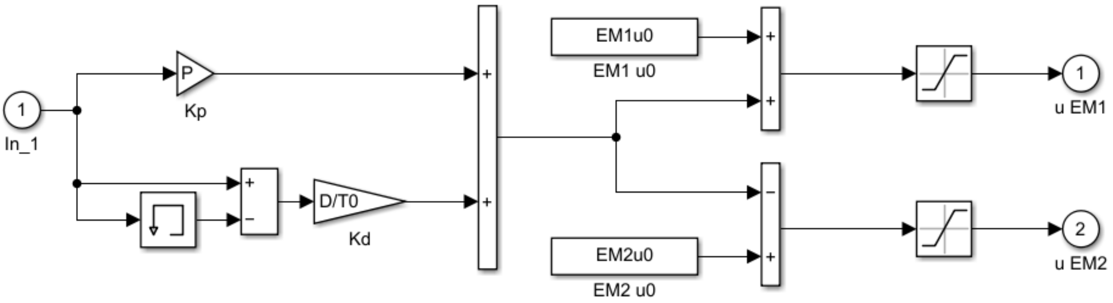


Figure 3.9 Interior of the differential mode PD-controller

The saturation blocks in Figure 3.8 and Figure 3.9 limit PD-controllers outputs $u(t)$ by the range of [0, 1].

$$u(t) = K_p e(t) + K_d \frac{de(t)}{dt} + u_0, \tag{3.6}$$

$$\begin{gathered} u_{EM1}(t) = K_p e(t) + K_d \frac{de(t)}{dt} + u_{0\,EM1}, \\ u_{EM2}(t) = -\left(K_p e(t) + K_d \frac{de(t)}{dt}\right) + u_{0\,EM2}. \end{gathered} \tag{3.7}$$

It is important to note here, that in the given five Simulink models (for example, in Figure 3.2) the position error is calculated differently rather than in common case (3.5), namely:

$$e(t) = x(t) - r(t). \tag{3.8}$$

Another issue is that there is no information on the operating points of the controllers in [1], but empirically it was found equal to 9·10^{-3} m for all the PD-controllers.

Table 3.4 Parameters of the simulation and experimental PD-controllers for the models 1-5

Controller number	**Controller name**	**Proportional gain, K_p**	**Derivative gain, K_d**	**Steady-state control, u_0**	**Controller sample time**
1	PD (sim.)	55	5	0,3617	1·10^{-3}
2	PD differential mode (sim.)	60	4	0,3812 for EM1 0,3812 for EM2	1·10^{-3}
3	PD EM1 (real-time)	30	0,3	0,315	1·10^{-3}
4	PD EM1, EM2 pulse excitation (real-time)	55	5	0,358	1·10^{-3}
5	PD differential mode (real-time)	45	1	0,45 for EM1 0,59 for EM2	5·10^{-4}

The main problem of the PD-controllers used in the simulation and experiments is that they are no more suitable when any change takes place in the MLS2EM. The PD-controllers must be re-tuned for each certain case. For example, controllers 1-5 are tuned for the ball with weight of $5{,}71\cdot10^{-2}$ kg and diameter of $6\cdot10^{-2}$ m, but after changing these parameters to those from Table 3.1, controllers can no more provide accurate and adequate control. The problem becomes even more complicated and time-consuming, since MLS2EM is a highly nonlinear and unstable system, and such MATLAB tool, as transfer function based automated tuning of PID-components (gains), is not available. Any disturbance, introduced in the system, for example, produced by the lower electromagnet EM2, crucially affects the tracking performance of the PD-controller.

The second issue is that the PD-controllers can provide more or less accurate control of the ball position in a very narrow range, which is approximately $2\cdot10^{-3}$ m (2 mm), around the operating point. In the mentioned above conditions regarding the ball weight and diameter, and with the constant reference position of $9\cdot10^{-3}$ m, the PD-control still results in the steady-state error of order 10^{-4} m.

3.1.4 Parameter estimation for MLS2EM

Since there is always a discrepancy between the real plant and its mathematical model, the parameter estimation for the MLS2EM is required. Thus, we can get closer in simulation to the behaviour of the real plant.

The parameter estimation was done using datasets "Setpoint" and "Ball position" (Figure 3.10), collected from the real plant acting on the medium size ball (ball number 2, Table 3.1). The characteristics of the setpoint, or reference signal, are given in Table 3.5.

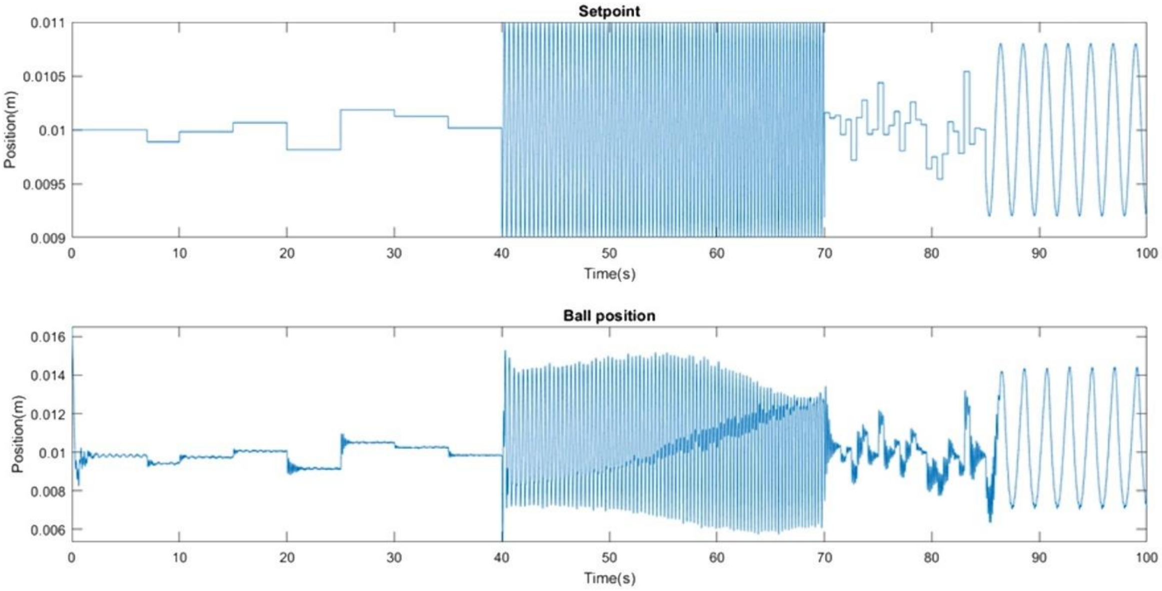


Figure 3.10 Datasets for parameter estimation

Table 3.5 Reference signal characteristics for collecting datasets (setpoint, Figure 3.10)

Time period, s	Singal type	Property	Value
[0, 7)	Constant	Value	$1\cdot10^{-2}$ m
[7, 40)	Uniform Random Number	Minimum Maximum Seed Sample time	$9{,}8\cdot10^{-3}$ m $1{,}02\cdot10^{-2}$ m 7 5 s
[40, 70)	Chirp	Initial frequency Frequency at time 60s Gain Shift along x-axis	1 Hz 6 Hz $1\cdot10^{-3}$ 10^{-2} m
[70, 85)	Band-Limited White Noise	Noise power Sample time Gain Shift along x-axis	0,1 0,5 s $5\cdot10^{-4}$ 10^{-2} m
[85, 100)	Sine Wave	Amplitude Bias Frequency Phase Sample time	$8\cdot10^{-4}$ m $1\cdot10^{-3}$ m 3 rad/sec 0 10^{-3} s

The parameter estimation was performed using Parameter Estimator App in MATLAB R2021b. The Trust-Region-Reflective algorithm and the Nonlinear Least Squares optimization method were used during the parameter estimation (Figure 3.11). The process took 43 iterations. The initial fit and the results of the estimation are depicted in Figure 3.12 and Figure 3.13, and given in Table 3.6. It is seen that the simulated output of MLS2EM is very close to the measured output after parameter estimation.

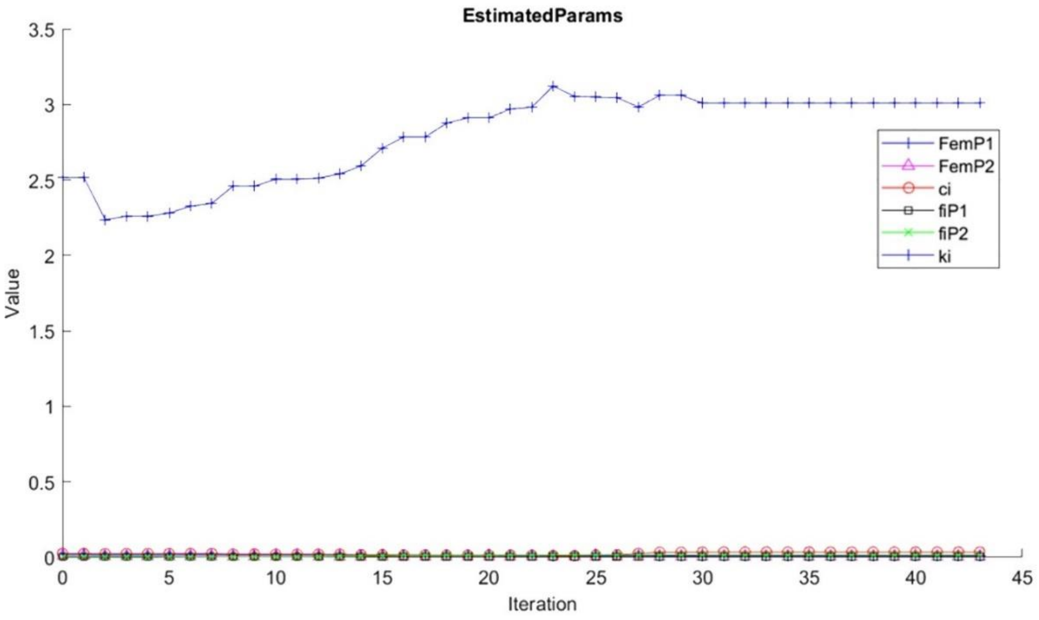


Figure 3.11 Parameter estimation progress

Table 3.6 Initial and estimated parameters' values

Parameter	Initial value	Estimation range	Estimated value	Unit
F_{emP1}	1,7521·10^{-2}	[0, 0,2]	1,1296·10^{-2}	H
F_{emP1}	5,8231·10^{-3}	[0, 0,2]	4,7817·10^{-3}	m
f_{iP1}	1,4142·10^{-4}	[0, 0,1]	2,5326·10^{-3}	m·s
f_{iP2}	4,5626·10^{-3}	[0, 0,1]	1,3875·10^{-2}	m
c_i	2,43·10^{-2}	[0, 0,2]	3,4421·10^{-2}	A
k_i	2,5165	[0, 10]	3,009	A

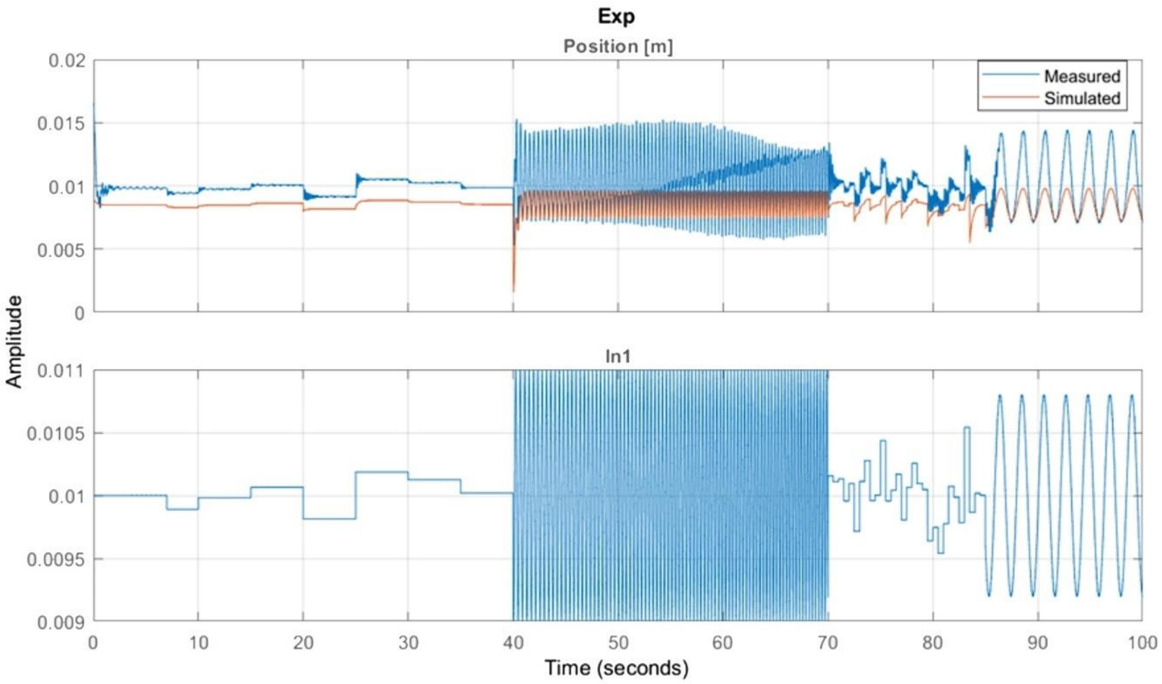


Figure 3.12 Measured and simulated outputs prior parameter estimation (above); reference signal (below)

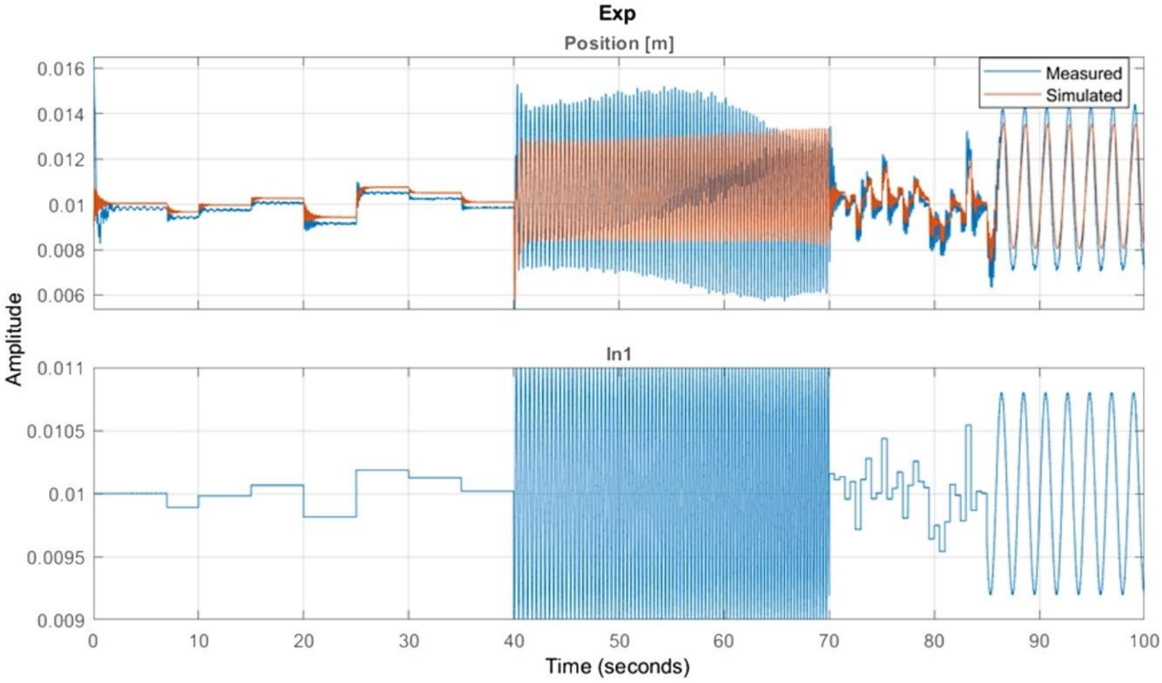


Figure 3.13 Measured and simulated outputs after parameter estimation (above); reference signal (below)

4. NARX-BASED CONTROLLER DESIGN

4.1 NARX-based controller

4.1.1 NARX model

As the first NN-based control solution for MLS2EM the nonlinear autoregressive network with exogenous inputs (NARX) was chosen. The NARX networks have a dynamic recurrent structure with feedback connections between layers. They are commonly used for such problems, as an input signal prediction, nonlinear filtering of the input signal, and also modeling of nonlinear dynamic systems, such as MLS.

The NARX model is defined by the recurrent equation (4.1) [44]:

$$y(t) = f\left(y(t-1), y(t-2), \dots, y(t-n_y), u(t-1), u(t-2), \dots, u(t-n_u)\right), \qquad (4.1)$$

where the output signal $y(t)$ at current time step $t,\ t \geq 0$, depends on its own previous values and the values of the input signal $u(t)$.

There are two configurations of the NARX network commonly used in trainings [44], Figure 4.1:

- Parallel architecture, where the network's estimated output is fed back to the second input of feedforward neural network;
- Series-Parallel architecture, where the true output (available target data) is used as the second input of feedforward network instead of feeding back the network's estimated output.

The Series-Parallel architecture will be used in the current work due to the higher accuracy of the input. After the training of NN is done, the NN structure is transformed into Parallel one with the feedback connection.

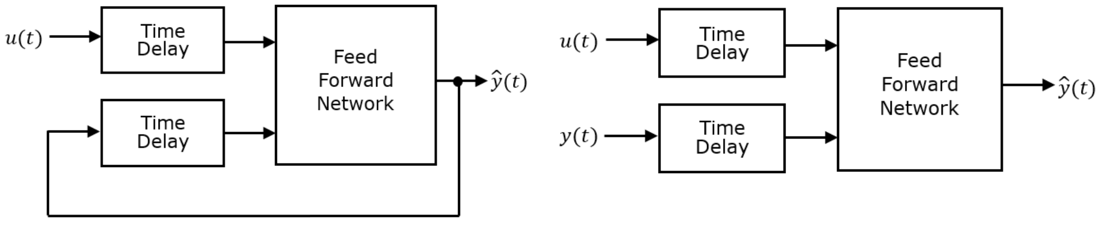


Figure 4.1 Parallel (on the left) and Series-Parallel (on the right) architecture of NARX

4.1.2 Training datasets

To realize the supervised learning for the NARX model, the following datasets has been prepared and collected in open loop as arrays (for convenience) of size 100001×1 with the sample time 10^{-3} s (Figure 4.2, Table 4.1):

- Input data "input1", which is an error generated in appropriate range going as an input to the PD-controller;
- Target data "output1", which is the PD-controller's response to the error, or control action;
- Time vector "t1", collected separately for building plots.

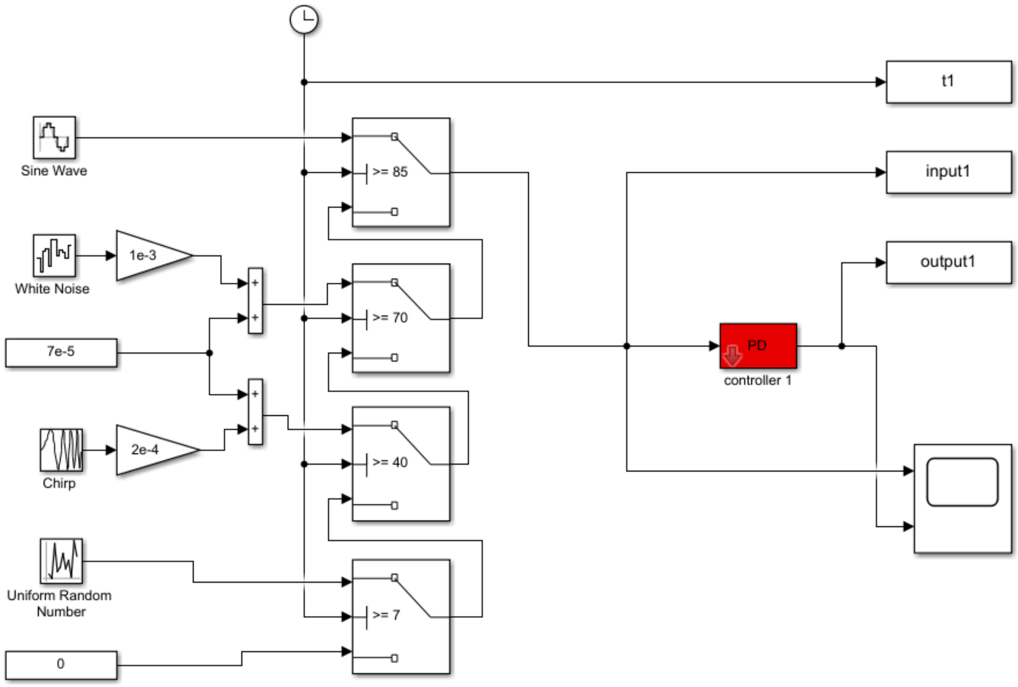


Figure 4.2 Generating of the training dataset, Simulink model

Table 4.1 Characteristics of the input signal

Time period, s	Singal type	Property	Value
[0, 7)	Constant	Value	0
[7, 40)	Uniform Random Number	Minimum Maximum Seed Sample time	$-1{,}1\cdot10^{-3}$ m $1{,}7\cdot10^{-3}$ m 7 3 s
[40, 70)	Chirp	Initial frequency Frequency at time 60s Gain Shift along x-axis	1 Hz 6 Hz $2\cdot10^{-4}$ $7\cdot10^{-5}$
[70, 85)	Band-Limited White Noise	Noise power Sample time Gain Shift along x-axis	0,1 0,5 s $1\cdot10^{-3}$ $7\cdot10^{-5}$

Table 4.1 continued

Time period, s	Singal type	Property	Value
[85, 100)	Sine Wave	Amplitude	10^{-3} m
		Bias	0 m
		Samples per period	10
		Offset	0
		Sample time	10^{-3} s

The training dataset is represented in the plot, Figure 4.3. The input data has been generated to look similar to the error's behaviour in case of closed-loop system.

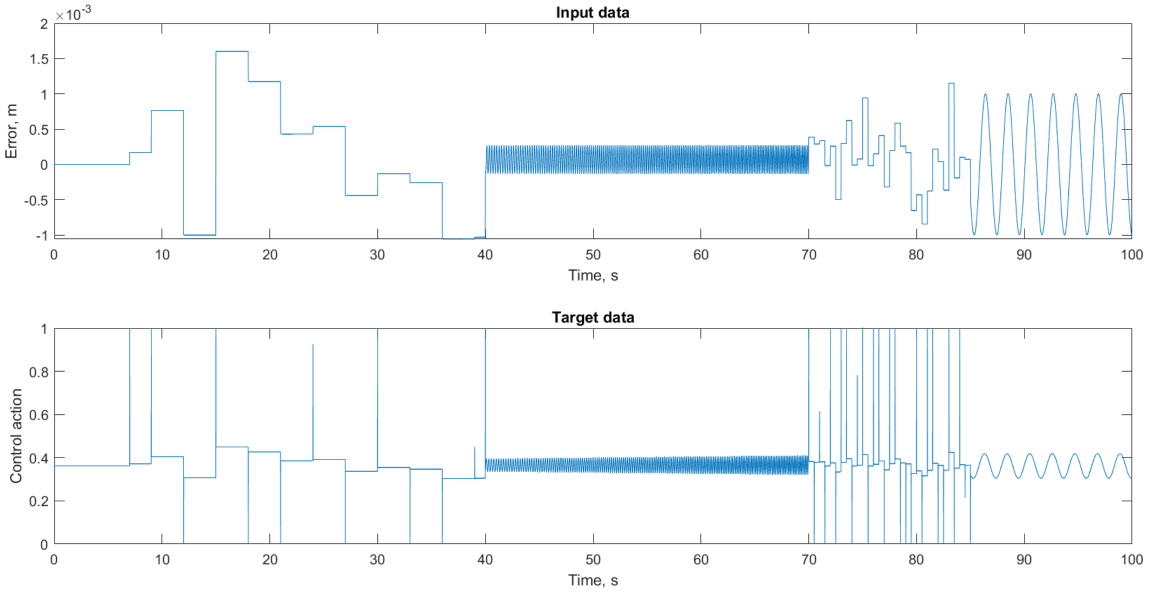


Figure 4.3 Input and target data, collected in simulation

4.1.3 NARX neural network structure

As it was mentioned in Section 4.1.1, the NARX network is trained in Series-Parallel (open-loop) configuration (Figure 4.1), and later transformed into Parallel (closed-loop) configuration. Let's denote:

- $x(t) = \binom{x(t)}{t}$ – input 1 of the NN, time-series dataset, where the first-row element $x(t)$ is a transposed vector "input1";
- $y(t) = \binom{y(t)}{t}$ – input 2 of the NN, time-series dataset, where the first-row element $y(t)$ is a transposed vector "output1";
- $IW\{1,1\}$ – matrix of the weighting coefficients from the input 1, $x(t)$, to the hidden layer 1, where $\{i,j\}$-element of the matrix corresponds to the i-neuron in the hidden layer 1 and j-row of the input vector $x(t)$, $i,j\epsilon\mathbb{N}$;

- $IW\{1{,}2\}$ – matrix of the weighting coefficients from the input 2, $y(t)$, to the hidden layer 1, where $\{i,j\}$-element of the matrix corresponds to the i-neuron in the hidden layer 1 and j-row of the input vector $y(t)$, $i,j\epsilon\mathbb{N}$;
- $b\{1\}$ – bias to the layer 1, of size $1\times i$, where i is a number of neurons in the hidden layer 1;
- $LW\{2{,}1\}$ – matrix of the weighting coefficients from the hidden layer 1 to the output layer 2, of size $1\times i$;
- $b\{2\}$ – bias to the layer 2, of size 1×1;
- F_1 and F_2 – activation functions of the hidden layer 1 and output layer 2, respectively
- $\hat{y}(t)$ – output of the NN.

The output of the NARX network of Series-Parallel structure is given in a matrix form by the equation (4.2):

$$\hat{y}(t) = F_2(LW\{2{,}1\}\cdot F_1(IW\{1{,}1\}\cdot x(t) + IW\{1{,}2\}\cdot y(t) + \mathrm{b}\{1\}) + \mathrm{b}\{2\}), \qquad (4.2)$$

where NN is supposed to have only one hidden layer.

The open-loop and closed-loop NARX network diagrams, interior and structure of its Simulink models are given in Figure 4.4, Figure 4.5, Figure 4.6, Figure 4.7, and Table 4.2. The Simulink models, shown in Figure 4.5 and Figure 4.7, were generated automatically with the command "gensim" from NN Toolbox.

Table 4.2 Open-loop / closed-loop NARX structure

Component number	Name	Type	Size, neurons	Learnables of closed-loop
1	$x(t)$	Input 1	1	-
2	Process input 1	Normalization		-
3	$y(t)$ / $a\{2\}$	Input 2	1	-
4	Layer 1	Hidden layer	10	Weights $IW\{1;1\}$ 10×2 $LW\{1;2\}$ 10×2 Bias $b\{1\}$ 10×1
5	Delays 1	Delay	-	-
6	Delays 2	Delay	-	-
7	Tansig	Activation function	-	-
8	Layer 2	Output layer	1	Weights $LW\{2;1\}$ 1×10 Bias $b\{2\}$ 1×1
9	Purelin	Activation function	-	-
10	Process output 1	Denormalization	-	-
11	$\hat{y}(t)$ / $a\{2\}$	Output	1	-

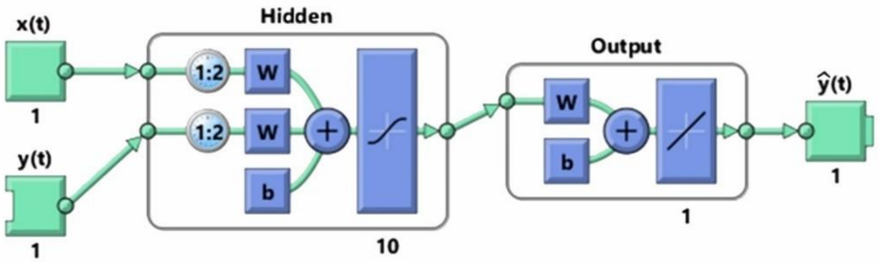
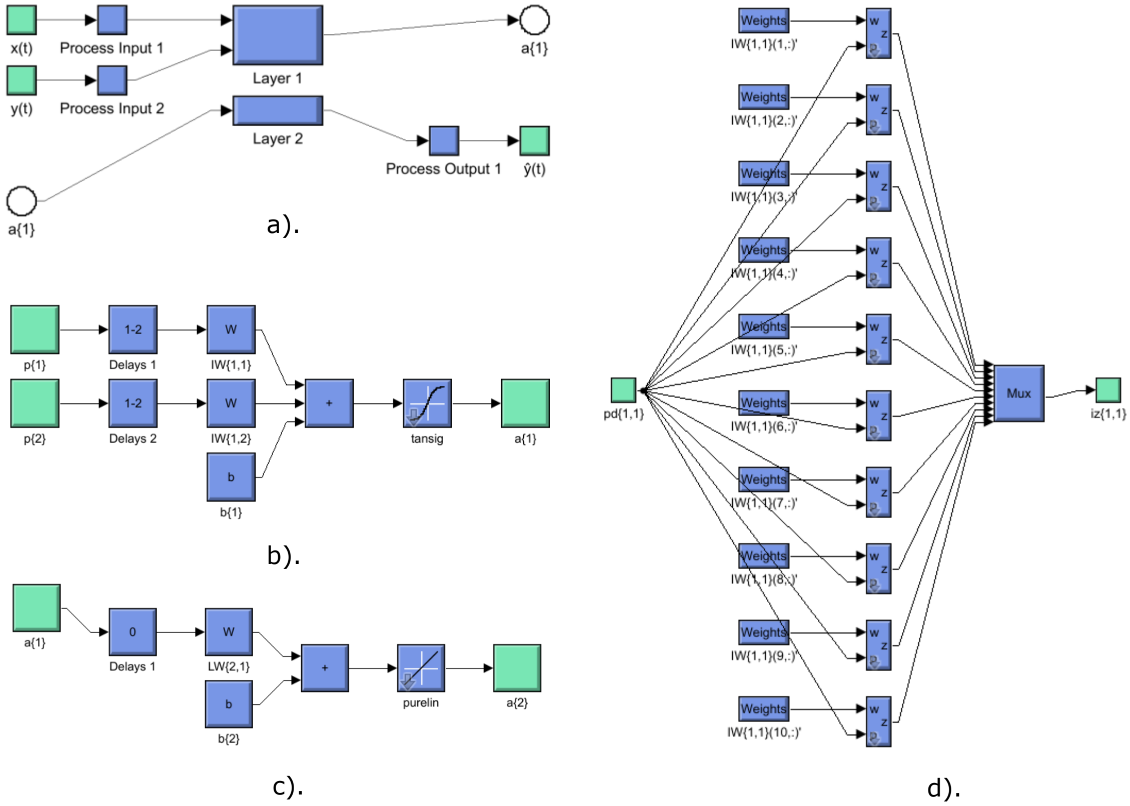


Figure 4.4 NARX series-parallel (open-loop) configuration diagram



a).

b).

c).

d).

Figure 4.5 a).The interior of open-loop NARX model; b). Hidden layer 1; c). Output layer 2; d). Interior of IW{1,1} block

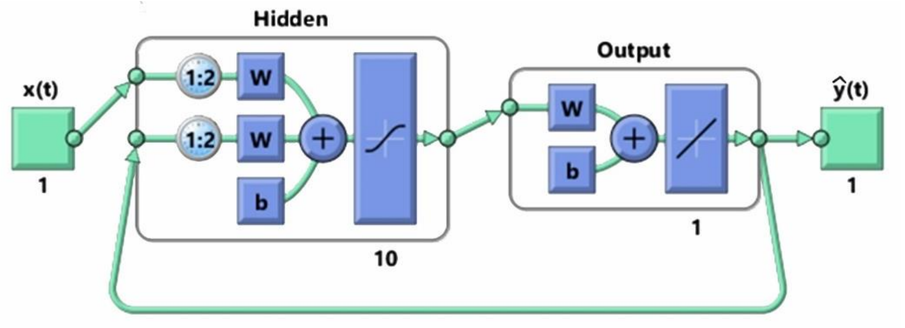


Figure 4.6 NARX parallel (closed-loop) configuration diagram

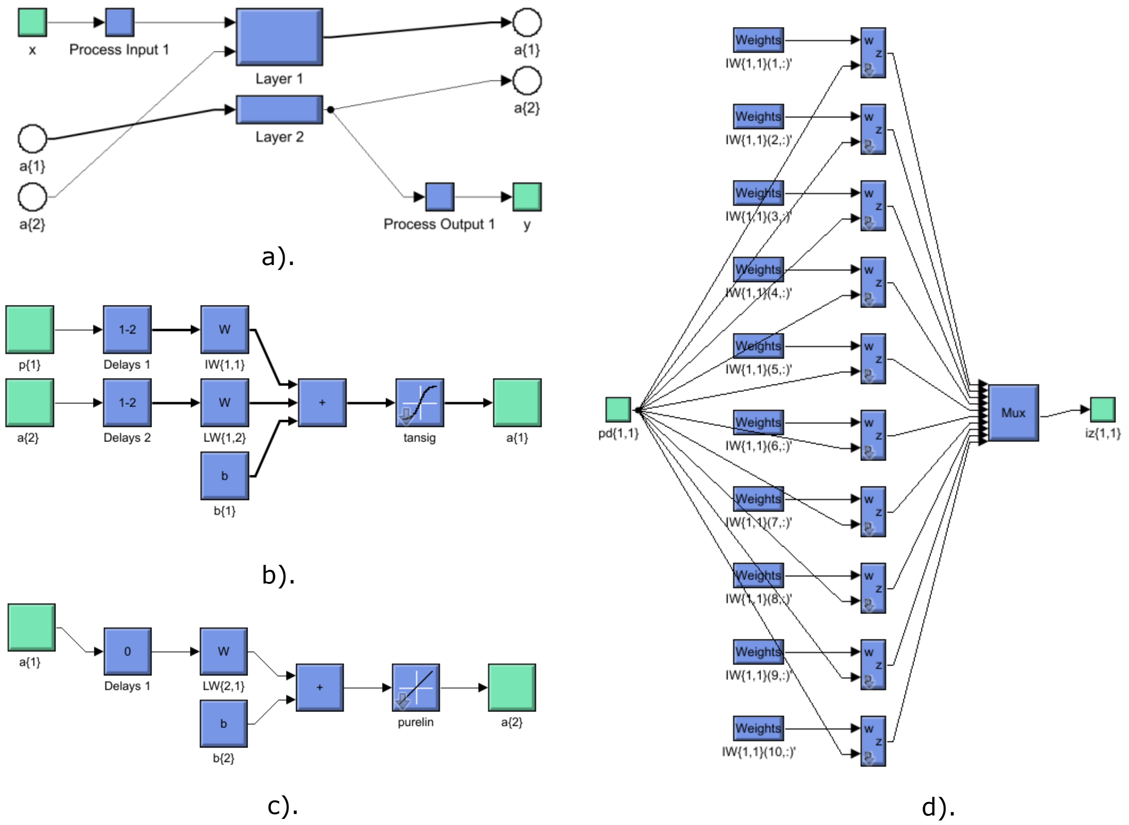


Figure 4.7 a).The interior of closed-loop NARX model; b). Hidden layer 1; c). Output layer 2; d). Interior of IW{1,1} block

The input and target data arrays "input1" and "output1" are transposed and converted into matrices, using command "con2seq", where the second raw is a set of time steps, corresponding to the data samples.

Two "Delays" blocks are set for each input; thus, the training starts from the third sample of datasets.

The "Process input 1" and "Process output 1" blocks conduct normalization of the input and denormalization of the output, respectively.

The input values are normalized according to the formula (4.3), realizing "mapminmax" MATLAB function:

$$\tilde{y} = \frac{(\tilde{x} - \tilde{x}_{min})(\tilde{y}_{max} - \tilde{y}_{min})}{\tilde{x}_{max} - \tilde{x}_{min}} + \tilde{y}_{min}, \qquad (4.3)$$

where $\tilde{x}$ and $\tilde{y}$ are the input and the output of the function.

The output values are denormalized using formula (4.4):

$$\breve{x} = \frac{(\breve{y} - \breve{y}_{min})(\breve{x}_{max} - \breve{x}_{min})}{\breve{y}_{max} - \breve{y}_{min}} + \breve{x}_{min}, \tag{4.4}$$

where $\breve{y}$ and $\breve{x}$ are the input and the output of the reverse function.

The activation functions "Tansig" (symmetric sigmoid transfer function) and "Purelin" (linear transfer function) are described by the equations (4.5) and (4.6):

$$F_1(x) = \frac{2}{1 + \exp(-2x)} - 1, \tag{4.5}$$

$$F_2(x) = x. \tag{4.6}$$

4.1.4 Training and its results

During the training, the NN learns complex associations between input and target data, and evaluates its own performance, or how close the returned output $\hat{y}(t)$ is to the target output $y(t)$.

The Levenberg-Marquardt algorithm (LMA), is used to minimizes the performance function, or loss function $L(t)$, given in a form of sum of squares (4.7):

$$L(t) = \sum_{k=1}^{N} (y_k(t) - \hat{y}_k(t))^2, \tag{4.7}$$

where $k,\ k \epsilon \mathbb{N},$ – number of the sample.

The iterative procedure of LMA, which approximates the Newton's method, is given by the common formula (4.8) [45]:

$$x_{k+1} = x_k [J^T J + \mu I]^{-1} J^T e, \tag{4.8}$$

where

e – vector of network errors,

J – Jacobian matrix of first derivatives of e with respect to network weights and biases,

I – identity matrix,

μ – scalar parameter, which is changed after each iteration step.

The gradient is calculated in a following way:

$$g = J^T e. \tag{4.9}$$

The LMA is used for each case of training. The training is performed in 300 epochs. After the training, the open-loop NARX network is rearranged into original Parallel form, using MATLAB command "closeloop". The NARX controller is generated with the command "gensim" with the sample time of $1 \cdot 10^{-3}$ s.

In the next Sections 4.1.5, 4.1.6, and 4.1.7, three NARX-based controllers will be presented. The following NARX-based controllers were learned from the PD-controllers 1, 2 and 4 (Table 3.4). The related NARX networks were created and trained using open-source MATLAB code [44], see Appendix 1. The full analysis of position tracking and stability performance will be given in Chapter 6.

4.1.5 NARX-based controller 1

The parameters of the designed NARX-based controller "narx_net1_closed", learned from the PD-controller 1, are given in (4.10):

$$\begin{gathered}
IW\{1;1\} = \begin{bmatrix} -1{,}5711 & -1{,}1854 \\ -0{,}9611 & -1{,}4866 \\ 0{,}7872 & -1{,}5972 \\ 0{,}8321 & -0{,}6935 \\ 0{,}7768 & -1{,}0333 \\ -0{,}3205 & 0{,}3598 \\ -1{,}5020 & 1{,}4866 \\ 0{,}8368 & -0{,}4003 \\ 1{,}2098 & -1{,}0431 \\ 0{,}3563 & -1{,}4156 \end{bmatrix} \qquad LW\{2;1\}' = \begin{bmatrix} 0{,}0289 \\ 0{,}0062 \\ 0{,}0564 \\ 1{,}0189 \\ 0{,}7870 \\ 0{,}2254 \\ -1{,}7802 \\ 0{,}3954 \\ -1{,}0288 \\ -0{,}7422 \end{bmatrix} \\
LW\{1;2\} = \begin{bmatrix} -0{,}7365 & -0{,}4897 \\ -0{,}7701 & -0{,}8142 \\ 0{,}8602 & 2{,}3477 \\ 2{,}9899 & -1{,}7177 \\ 7{,}0143 & -4{,}7847 \\ 0{,}1900 & 3{,}4506 \\ 3{,}7893 & -2{,}4252 \\ -0{,}3342 & -0{,}1395 \\ -0{,}3245 & 1{,}9986 \\ -0{,}0014 & -0{,}1234 \end{bmatrix} \qquad b\{1\} = \begin{bmatrix} 2{,}9087 \\ 1{,}4966 \\ 1{,}2087 \\ -0{,}2117 \\ 0{,}2520 \\ 1{,}9699 \\ -0{,}0732 \\ 0{,}4850 \\ 2{,}3823 \\ 2{,}5733 \end{bmatrix} \\
b\{2\} = 1{,}0563
\end{gathered} \tag{4.10}$$

where $LW\{1;2\}$ – matrix of the weighting coefficients replacing corresponding matrix $IW\{1,2\}$ from open-loop case (Figure 4.7).

The simulated output of the NARX network together with the target data, used during the training is shown in Figure 4.8. It is seen, that at some points NARX output goes behind the range of [0,1], which means that the NARX network will work somewhat worse rather than the related PD-controller. Later the control action is saturated at 0 and 1 inside MLS2EM block. The simulated NARX output was compared to the PD-controller's output using the MSE, which equals to $4{,}2925 \cdot 10^{-4}$. The squares of errors for NARX over the time t are depicted in Figure 4.9.

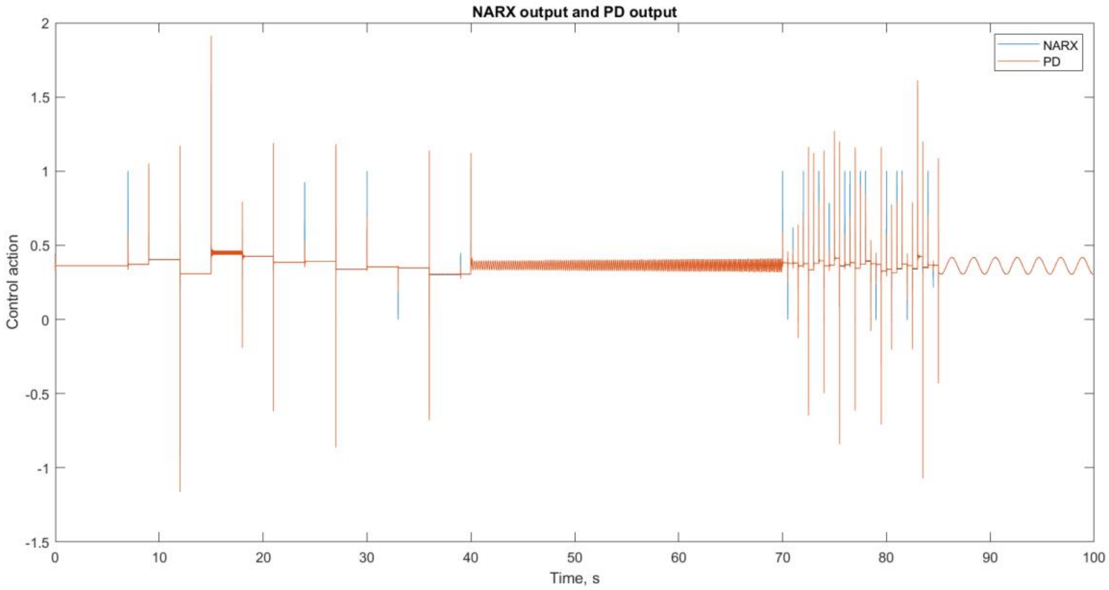


Figure 4.8 NARX and PD output comparison

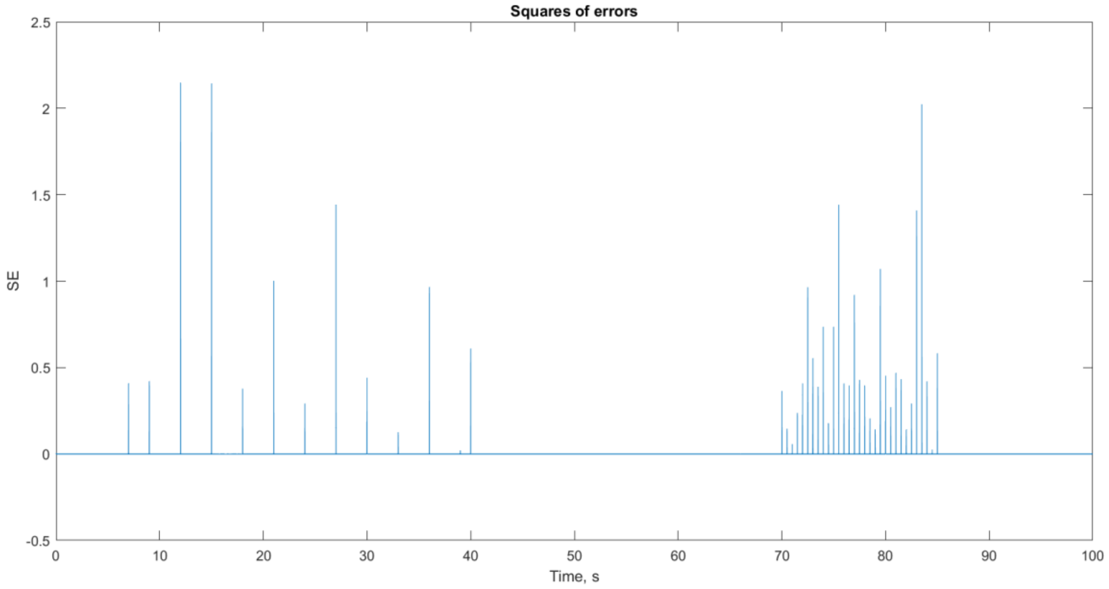


Figure 4.9 Squares of errors for NARX over the time

4.1.6 NARX-based controller 2

The NARX control corresponding to the differential mode PD-controller 2 (Table 3.4) was realized as two NARX-based controllers for EM1 and EM2 separately. Since the PD-controller 2 has two outputs, the training data was collected from each of them. The parameters of the NARX-based controller for EM1 are given in (5.5). And the MSE value, calculated for comparison of NARX and PD-controller's outputs, equals to 1,1387·10^{-4}.

$$IW\{1;1\} = \begin{bmatrix} -0{,}1548 & 1{,}0869 \\ -0{,}2101 & 0{,}0675 \\ -2{,}0148 & -0{,}4340 \\ -1{,}5955 & 1{,}3751 \\ 0{,}5687 & -1{,}3224 \\ -0{,}8278 & -2{,}8071 \\ 0{,}3058 & 1{,}5597 \\ -0{,}6805 & -0{,}3998 \\ 0{,}5860 & -0{,}1964 \\ 1{,}1515 & -0{,}6019 \end{bmatrix} \qquad LW\{2;1\}' = \begin{bmatrix} -0{,}0579 \\ -0{,}5594 \\ -0{,}0087 \\ 0{,}2661 \\ 0{,}7590 \\ 0{,}0259 \\ 0{,}0585 \\ -0{,}1782 \\ 1{,}3428 \\ -0{,}3977 \end{bmatrix}$$

$$LW\{1;2\} = \begin{bmatrix} 1{,}5011 & 1{,}4176 \\ -1{,}9873 & -1{,}2346 \\ -1{,}3850 & -1{,}0671 \\ -2{,}0346 & -0{,}4208 \\ 4{,}4689 & -0{,}9881 \\ -0{,}6768 & -1{,}2108 \\ 3{,}4217 & 0{,}4119 \\ 1{,}3890 & -1{,}2374 \\ -2{,}0713 & 0{,}3746 \\ 0{,}1470 & 2{,}1746 \end{bmatrix} \qquad b\{1\} = \begin{bmatrix} 2{,}4467 \\ -1{,}8737 \\ 1{,}6147 \\ 1{,}0942 \\ 0{,}0545 \\ -0{,}5950 \\ 0{,}3949 \\ -1{,}9347 \\ -0{,}1397 \\ 2{,}7238 \end{bmatrix} \tag{4.11}$$

$$b\{2\} = 0.1027$$

The parameters of the NARX-based controller for EM2 are given in (4.12). And the related MSE equals to 8,2169·10^{-4}.

$$IW\{1;1\} = \begin{bmatrix} -1{,}2599 & -0{,}6166 \\ 2{,}0494 & -0{,}5117 \\ -0{,}2667 & -0{,}7080 \\ -1{,}1055 & 0{,}4494 \\ 0{,}2693 & 0{,}1285 \\ -2{,}0241 & -1{,}1927 \\ 0{,}6604 & 1{,}6507 \\ 0{,}3908 & 2{,}4090 \\ 0{,}2945 & 1{,}4833 \\ -0{,}3247 & -0{,}4166 \end{bmatrix} \qquad LW\{2;1\}' = \begin{bmatrix} 0{,}3820 \\ 0{,}4612 \\ 1{,}2698 \\ 0{,}1860 \\ -0{,}6743 \\ 0{,}2940 \\ 0{,}2520 \\ 0{,}1122 \\ 0{,}8727 \\ -0{,}6959 \end{bmatrix}$$

$$LW\{1;2\} = \begin{bmatrix} -1{,}2935 & 2{,}1896 \\ 0{,}6896 & -0{,}6694 \\ 0{,}6565 & -0{,}6096 \\ 1{,}2389 & 0{,}8733 \\ 3{,}4240 & 0{,}7469 \\ -1{,}6231 & -1{,}2791 \\ 0{,}5122 & -0{,}5876 \\ -1{,}0850 & 1{,}3397 \\ -0{,}1690 & 0{,}2847 \\ -0{,}4476 & -1{,}0349 \end{bmatrix} \qquad b\{1\} = \begin{bmatrix} 1{,}8858 \\ -1{,}9460 \\ -0{,}0858 \\ 1{,}5445 \\ -0{,}6239 \\ -0{,}6272 \\ 0{,}4366 \\ -1{,}0595 \\ 2{,}4012 \\ 1{,}0737 \end{bmatrix} \tag{4.12}$$

$$b\{2\} = 0{,}7505$$

4.1.7 NARX-based controller 4

The parameters of the designed NARX-based controller, learned from the PD-controller 4, are given in (4.13). And the related MSE equals to 2,9628·10^{-4}.

$$IW\{1;1\} = \begin{bmatrix} -1{,}0644 & 0{,}7998 \\ -0{,}6850 & -1{,}9695 \\ 0{,}3325 & -0{,}1619 \\ 1{,}4099 & -0{,}2552 \\ 1{,}3821 & -1{,}5384 \\ -0{,}4540 & -2{,}5546 \\ -0{,}9576 & 0{,}9500 \\ 1{,}0704 & 0{,}2007 \\ 0{,}3932 & -1{,}1143 \\ 0{,}9415 & 0{,}6126 \end{bmatrix} \quad LW\{2;1\}' = \begin{bmatrix} 0{,}0391 \\ 0{,}0103 \\ 0{,}6545 \\ 0{,}0797 \\ 1{,}0609 \\ -0{,}0058 \\ -1{,}5739 \\ 0{,}1115 \\ -0{,}0896 \\ 0{,}0155 \end{bmatrix}$$

$$LW\{1;2\} = \begin{bmatrix} 1{,}1755 & -1{,}4630 \\ 0{,}1278 & -0{,}1900 \\ 1{,}6428 & -1{,}4782 \\ -0{,}7310 & 0{,}4081 \\ 6{,}3106 & -5{,}0281 \\ 0{,}5325 & -0{,}2290 \\ 3{,}8451 & -3{,}4080 \\ -1{,}5176 & -0{,}1900 \\ 1{,}3787 & -1{,}6747 \\ 0{,}7089 & 2{,}0036 \end{bmatrix} \quad b\{1\} = \begin{bmatrix} 2{,}9279 \\ 1{,}8852 \\ 0{,}0382 \\ -0{,}8902 \\ 0{,}3319 \\ -0{,}7329 \\ 0{,}1133 \\ 1{,}6058 \\ 2{,}0165 \\ 2{,}3106 \end{bmatrix} \quad b\{2\} = -0{,}2891 \qquad (4.13)$$

4.2 Obstacles in NARX-based controller design

During the NAXR-controller design and based on the results of the training, the number of obstacles to creating an adequate control has appeared.

- Due to the fact, that the weights and biases of the network are initialized randomly, each certain network training (with same training settings and network structure) ends up with different results. Moreover, the trainings often end up with unpredictable, insufficient, or on opposite, excessive control action, which is frequently shifted away from the expected control action range. This means, that even being simple with respect to the mathematical effort, the design of a proper controller may take many trials and quite a long time.

- The simplicity or the difficulty of the training datasets structure (for example, pure sine wave form "input data") do not result in the design of a better control. The same is true about the network structure. The different number of the hidden layers and different layer sizes were tried out during the network training, but no correlation was found between those and the network's improvement.

- The NARX-based controller can only surpass the PD-controller's performance only at some points in accuracy, but overall performance of the NARX-based controller is weaker (this will be discussed in more details in Chapter 6). The NARX controller cannot interpolate and extend the PD working region. It is less stable to disturbance rather than the PD-controller.

- The training datasets for the experimental NARX-based controller for real-time control cannot be collected from the PD-controller in open-loop since there is no option to run the experiment on the real plant with disconnected PD-controller. The datasets collected in closed-loop include excessive information that comes from MLS2EM output and affects the position error values. These datasets are useless in the NARX-based controller design.

For these reasons, one can conclude that the NARX-based controller is not suitable in the tracking control problem for MLS2EM, or the pure NARX-control is not sufficient. In the next Chapter 5 the Reinforcement Learning based control solution will be presented.

5. RL-BASED CONTROLLER DESIGN

5.1 Reinforcement Learning based controller

5.1.1 Deep Reinforcement Learning approach

Deep Reinforcement Learning (DRL) is a branch of machine learning, which stands separately from supervised and unsupervised learning. Supervised Learning supposes providing input and output behaviour patterns to be achieved during the learning process. Unsupervised Learning realizes self-learning by discovering similar input features and categorizing them into groups with a certain output probability.

Reinforcement Learning does not employ either training datasets as a behaviour sample, or data distribution to categories during the learning process. Instead, RL involves direct interaction between a learner, called "agent", and an environment.

An environment includes everything outside an agent, namely: dynamic model of a plant, reference and measured signals, observations' block, reward generating and termination blocks.

The observations' block contains measured signals to be observed, including calculated error between setpoint and measured values, and its variations. A set of observations is a vector of values which is observed at each moment of time. It is commonly called "state". Thus, the state is being changed and observed throughout the whole learning process at each time step.

The reward block provides the rule to evaluate and regulate the overall performance of the agent's training.

The training is split into episodes, which literally represent separate runs of experiment or simulation. Termination block provides stopping criteria, or certain conditions, that terminate the current episode and immediately launch the next one. The initial state of the system is restored at the beginning of each episode.

The reinforcement learning method includes five main components:

- agent,
- environment E,

- action a_t,
- state s_t,
- reward r_t, where t is a time step, $t \geq 0$.

Let's also introduce the following notations:

$\mathcal{S}$ – set of states, or state space, finite,
$\mathcal{A}$ – set of actions, or action space, finite.

The agent interacts with the environment E in the following way. At each time step t the agent receives a set of observations, or state s_t, selects and performs an action a_t, which drives system to the new state s_{t+1}, and immediately obtains a scalar reward r_t. The immediate reward, that agent receives after performing an action at each time step, is defined by the rule called a "reward function". The reward function plays the crucial role in RL. A nicely defined reward function gives the proper "motivation" to the agent and affects the quality and speed of the learning performance.

Let's denote by $r(s_t, a_t)$ the immediate reward r_t received by the agent after performing an action a_t through the state s_t at time t. By the transition dynamics $p(s_{t+1}|s_t, a_t)$ we will understand probability, that action a_t in a state s_t at time step t will lead to the state s_{t+1} at time step $t+1$.

The agent behavior, or the action choice in the state s_t, is determined by the policy π, or in other words, by the probability of the action a_t at time step t. The policy function, or policy, $\pi: \mathcal{S} \to \mathcal{P}(\mathcal{A})$, realizes mapping between the state space $\mathcal{S}$ and the probability distribution over the action space $\mathcal{A} \subset \mathbb{R}^N$.

The so-called "return from a state", or "cumulative reward", is defined as the sum of discounted future reward R_t given by the formula (5.1) [39]:

$$R_t = \sum_{i=t}^{T} \gamma^{i-t} r(s_i, a_i), \qquad (5.1)$$

where γ – discount factor, $\gamma \in [0,1]$, driving the agent either to take actions immediately or postpone them over time (when $\gamma \to 1$). The discount factor is usually chosen close to 1.

The goal of RL is to find an optimal policy π^*, which maximizes the cumulative reward function (5.1).

5.1.2 Deep Deterministic Policy-Gradient algorithm

Deep Deterministic Policy-Gradient (DDPG) algorithm, described in [39], Appendix 2, represents an off-policy actor-critic algorithm which can learn policies in continuous action spaces. This makes the algorithm applicable for the control of the systems with complex continuous dynamics, such as MLS.

The DDPG algorithm utilizes two artificial neural networks, given by the following functions:

- actor $\mu(s|\,\theta^\mu)$, with parameters θ^μ, which takes the state s_t as an input, and has the specific action a_t at the output, thus, realizing the current policy π,
- critic $Q(s,a|\theta^Q)$, with parameters θ^Q, which takes the state s_t and action a_t as two inputs and returns the corresponding expectation of the discounted cumulative reward R_t.

The DDPG algorithm iteratively solves the recursive Bellman equation (5.2) for critic Q:

$$Q^\mu(s_t,a_t) = \mathbb{E}_{r_t,s_{t+1}\sim E}\left[r(s_t,a_t) + \gamma Q^\mu\big(s_{t+1},\mu(s_{t+1})\big)\right], \qquad (5.2)$$

where $\mathbb{E}_{r_t,s_{t+1}\sim E}$ — an expectation of cumulative reward, depending only on the environment E.

The loss function $L(\theta^Q)$ of critic parameters θ^Q is given by the equations (5.3), (5.4):

$$L(\theta^Q) = \mathbb{E}_{s_t\sim\rho^\beta,a_t\sim\beta,r_t\sim E}\left[(Q(s_t,a_t|\theta^Q) - y_t)^2\right], \qquad (5.3)$$

where

$$\begin{gathered} y_t = r(s_t,a_t) + \gamma Q(s_{t+1},\mu(s_{t+1})|\theta^Q), \\ \mu(s_{t+1}) = a_{t+1}. \end{gathered} \qquad (5.4)$$

β — stochastic behavior policy, used to learn Q off-policy.

The goal of the training is minimization of the loss function $L(\theta^{Q})$ with respect to θ^{Q}.

The actor μ is updated applying the chain rule to the expected return from the start distribution J with respect to the actor parameters θ^{μ} [39]:

$$\begin{aligned} \nabla_{\theta^{\mu}} J &\approx \mathbb{E}_{s_t \sim \rho^{\beta}} \left[\nabla_{\theta^{\mu}} Q(s, a|\theta^{Q})|_{s=s_t, a=\mu(s_t|\, \theta^{\mu})} \right] \\ &= \mathbb{E}_{s_t \sim \rho^{\beta}} \left[\nabla_{a} Q(s, a|\theta^{Q})|_{s=s_t, a=\mu(s_t)} \nabla_{\theta^{\mu}} \mu(s|\theta^{\mu})|_{s=s_t} \right], \end{aligned} \tag{5.5}$$

where $\nabla_{\theta^{\mu}} J$ — gradient of J with respect to θ^{μ}, and J is given by the expression (5.6):

$$J = \mathbb{E}_{r_i, s_i \sim E, a_i \sim \pi} [R_1]. \tag{5.6}$$

During the training, actor and critic update and store their parameters. The training algorithm, given by T.P. Lillicrap and others in [39], considers use of a replay buffer R, which is a finite sized cache, storing the transition tuples (s_t, a_t, r_t, s_{t+1}). The learning process is realized in minibatches. This is done to solve the problem of a uniform distribution of samples for the optimization algorithm in case of continuous time domains.

The "target networks", $\mu'(s|\theta^{\mu'})$ and $Q'(s, a|\theta^{Q'})$, which are the copies of actor and critic networks, with parameters $\theta^{\mu'}$ and $\theta^{Q'}$ respectively, are used to calculate the target values [39]. The weights of target networks μ' and Q' are being updated slowly (5.7), applying target smooth factor τ, $\tau \ll 1$. This prevents instability and divergence issues of the learning process, which is a frequent problem of the Q-learning algorithm. The full DDPG algorithm, described in [39], is given in Appendix 2.

$$\begin{aligned} \theta^{Q'} &\leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'}, \\ \theta^{\mu'} &\leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'}. \end{aligned} \tag{5.7}$$

For effective networks' learning the deep learning technique called "batch normalization" [46] is commonly used in problem of differently scaled input values. A minibatch of data is normalized across all observations using the rule (5.8) from [47]:

$$\begin{aligned} \hat{x}_i &= \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}, \\ y_i &= \gamma \hat{x}_i + \beta, \end{aligned} \tag{5.8}$$

where

x_i — an element of the input to be normalized,

μ_B — mean, and

σ_B^2 — variance, calculated for each channel (or, observation) independently,

ϵ — constant improving numerical stability in case of very small variance value,

γ — scale factor, and

β — offset, updated during the network training.

One of the important challenges of learning process is exploration of the continuous action space. To realize this, the authors of [39] introduced exploration policy μ', where the noise $\mathcal{N}_t$ is added to the actor output, or selected action, at each time step (5.9):

$$\mu'(s_t) = \mu(s_t|\theta^\mu) + \mathcal{N}_t. \tag{5.9}$$

The frequently used action noise model is based on Ornstein-Uhlenbeck process (OUP), which is described by stochastic differential equation and refers to the Wiener process. As it is mentioned in [42], simpler models, such as additive Gaussian noise model, do not suit, since they may cause abrupt unexpected changes over the time and replicate real life actuators' behavior worse that OUP model does.

In RL, OUP action noise model is realized through the notions of standard deviation and mean, provided in [48]. The noise value $v(t+1)$ at each next time step $t+1$ is defined by the recurrent formula (5.10):

$$\begin{gathered} v(t+1) = v(t) + MeanAttractionConstant.*\left(Mean - v(t)\right).*Ts \\ +StandardDeviation(t).*randn\left(size(Mean)\right).*\sqrt{Ts}, \end{gathered} \tag{5.10}$$

where Ts — simulation sampling time.

It is important to appropriately set the noise standard deviation, taking into account the particular environment properties, to encourage good exploration of the action space. The standard deviation is chosen so, that the product of multiplication $StandardDeviation.*\sqrt{Ts}$ equals to 1% to 10% of the expected action range for specific system (5.11), [48]:

$$StandardDeviation.*\sqrt{Ts} = (1\% \ to\ 10\%)\ of\ ActionRange. \tag{5.11}$$

The noise, added to the action at each time step, plays the crucial role in the exploration ability of an agent, and therefore directly impacts the learning performance. For example, increasing the amount of noise, one can overcome the problem of too quick convergence to the local minimum and jamming of the training process.

5.1.3 Environment, observations and reward function

The following control solution suggests training the DDPG agent in parallel with the PD-controller for several cases of the latter one (Table 3.2).

The environment includes the following components (Figure 5.1):

- the plant "Magnetic Levitation model (MLS2EM)";
- the PD-controller to be improved;
- set of reference signals, scope and display blocks;
- "Generate observations" block, which consists of measured position and calculated error at each time step, and also integral error to accumulate the error value over the time, and generates an observations' vector;
- "Stop simulation" block with criteria for immediate termination of simulation in case of falling out of bounds for position;
- "Calculate reward" block, which calculates the scalar reward at each time step.

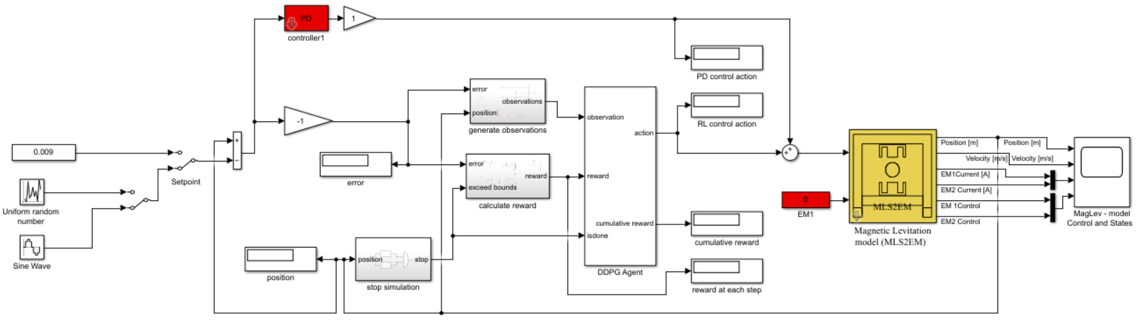


Figure 5.1 Simulink model, PD controller and DDPG agent in parallel

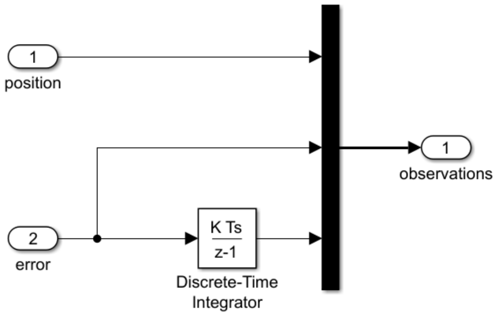


Figure 5.2 "Generate observations" block

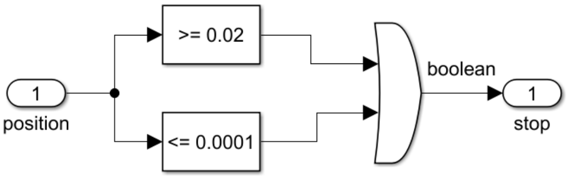


Figure 5.3 "Stop simulation" block

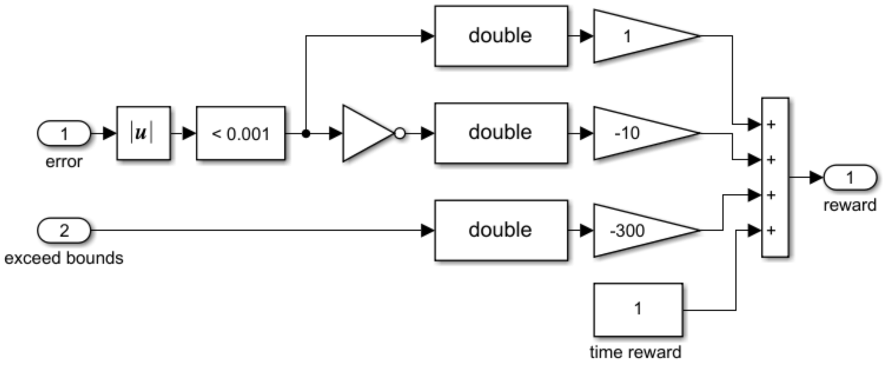


Figure 5.4 "Calculate reward" block

The observations information includes 3x1 vector with elements given in Table 5.1. The position and the error values are observed in meters. The action information includes 1x1 vector called "control", whose values are saturated later in the plant block (Figure 1.1) at the lower and upper actuator limits u_{min} and u_{max} (Table 3.3), namely, at 4,98·10^{-3} and 1.

Table 5.1 List of observations with limits

Observation	Limits
Position	$[0, +\infty) \subset \mathbb{R}^n$
Error	$(-\infty, +\infty) \subset \mathbb{R}^n$
Integral error	$(-\infty, +\infty) \subset \mathbb{R}^n$

The stopping criteria is based on the measured position value. The block sets its output to logic "true" when the agent goes behind the limits of 1·10^{-4} m (lower limit) or 2·10^{-2} m (upper limit) and rises the flag "isdone" inside the "DDPG agent" block.

The measured position value and calculated error go as two inputs into "Calculate reward" block. After many experiments, the most appropriate reward function was chosen of the following view (5.12):

$$
\begin{gathered}
r_1 = \begin{cases} 1, & if \ \ |e| < 1 \cdot 10^{-3}, \\ -10, & otherwise, \end{cases} \\
r_2 = \begin{cases} -300, & if \ \ x_1 \le 1 \cdot 10^{-4}, or \ x_1 \ge 2 \cdot 10^{-2}, \\ 0, & otherwise, \end{cases} \\
r_3 = 1, \qquad \forall t \ge 0, \\
reward = r_1 + r_2 + r_3,
\end{gathered}
\tag{5.12}
$$

where

r_1 — error related (accuracy) reward,

r_2 — position related penalty for exceeding bounds,

r_3 — time (continuation) reward.

The error tolerance was chosen based on the training performance. It is obvious, that the smaller allowable error range, the harder for DDPR agent to find a solution. Taking into account the high instability of MLS, any decrease of the error tolerance to 5·10^{-4}, or even to 8·10^{-4}, appears to be impossible. The experiments in this case end up with constantly growing deviation of action value, being produced by the actor, from the expected control range $[u_{min}, u_{max}]$.

Block "Data type conversion" (

Figure 5.4) was added to convert data from Boolean to Double type. This important point is not mentioned in the similar example for water tank model problem described in RL Toolbox User's Guide [43], pp. 2-46 - 2-49.

The proposed type of reward function (5.12) gives clear understanding of goodness of the training process since the reward is calculated as integer value at each time step. Comparing to the exponential reward functions (2.4) from [42], the reward function (5.12), can give more motivation to the agent, by strictly penalizing it for falling out of the allowable error range. The cumulative reward is one of the "DDPG agent" block's outputs and displayed during the training using "display" block along with single time-step reward (Figure 5.1).

The sine wave was chosen in a role of reference signal (Table 5.2) as the most appropriate pattern for training. To change the setpoint during the training as it was done in [42] and [43], using environment reset function "env.ResetFnc", appeared to be unreasonable due to high instability of MLS. Thus, the training was done, using only one unchangeable reference signal.

The control action of DDPG agent is displayed during training, using "Display" block, as long as cumulative and each time step reward values. The measured position is observed in the "Scope" block.

Table 5.2 Reference signal (sine wave) parameters

Parameter	Value
Amplitude, m	$1 \cdot 10^{-3}$
Bias, m	$9 \cdot 10^{-3}$
Frequency, rad/s	6
Phase, rad	0
Sample time, s	10^{-3}

The MLS2EM parameters set during the training include those given in Table 3.3, where the ball mass is $5{,}71 \cdot 10^{-2}$ kg and ball diameter is $6 \cdot 10^{-2}$ m. The initial state of the system is given by (3.3).

The Simulink environment is built using command "rlSimulinkEnv" from the open-source MATLAB code [43] from RL Toolbox, Appendix 3.

5.1.4 Actor and Critic design

In order to design an appropriate control solution for MLS, enormous number of simulation experiments had been conducted. The different structures of actor and critic NNs were tested out.

The following obstacles were met during the process of actor and critic design.

- The RL Toolbox does not support batch normalization, which was found useful in research works [39] and [40]. The regular normalization applied either at the input layers or at all layers of NNs does not affect the improvement of the training.

- The size of the NNs is limited by the computational ability of the computer (Intel(R) Core(TM) i5-1035G1 CPU, RAM 8,00 GB) used for the RL trainings. It was found that the medium-size NNs (with up to 50 neurons in the hidden layer) work well for the problem of improvement of the PD-controller, and there is no need to exploit large NNs (with 300-400 neurons in the hidden layer), as it was done in [39]. Even regular normalization, applied only to input layers of NNs, could slow down the training process in several times and make the training impossible. For the mentioned reasons, it was decided not to use any normalization.

The actor and critic NNs were created using Deep Network (DN) Designer App from Deep Learning (DL) Toolbox [49] and later imported in Reinforcement Learning (RL) Designer. It is also possible to create NNs directly in RL Designer, but there are limitations for activation functions and number of layers. In fact, the equal number of neurons in each hidden layer of actor and critic is the only option to set in RL Designer.

The selected actor and critic structure is given in Table 5.3 and Table 5.4, and shown in Figure 5.5.

Table 5.3 Actor structure

Layer number	**Name**	**Type**	**Layer size, neurons**	**Learnables**
1	input_1	Feature Input	3	- -
2	fc_1	Fully Connected	25	Weights 25×3 Bias 25×1
3	tanh_1	Tanh	-	- -
4	fc_2	Fully Connected	25	Weights 25×25 Bias 25×1
5	tanh_2	Tanh	-	- -
6	output	Fully Connected	1	Weights 1×25 Bias 1×1

Table 5.4 Critic structure

Layer number	**Name**	**Type**	**Layer size, neurons**	**Learnables**
1	input_1	Feature Input	3	- -
2	st_fc_1	Fully Connected	50	Weights 50×3 Bias 50×1
3	relu_1	ReLU	-	- -
4	st_fc_2	Fully Connected	25	Weights 25×50 Bias 25×1
5	input_2	Feature Input	1	- -
6	act_fc_1	Fully Connected	25	Weights 25×2 Bias 25×1
7	concat	Addition	-	- -
8	relu_output	ReLU	-	- -
9	output_1	Fully Connected	1	Weights 1×25 Bias 1×1

The actor NN, was exported from DN Designer to MATLAB workspace only with "layers" structure, but not "layer graph", as for critic NN. This causes the error while importing actor network in RL Designer. The problem was solved by correcting the code, automatically generated in DN Designer (Appendix 4), namely, by declaring the corresponding layer graph (with command "layerGraph") and adding actor's layers to it (with command "addLayers").

Also, it worth to notice, that there may appear "name mismatch" error in RL Designer, which is solved by keeping the input and output layers' names strictly as "input_1", "input_2", "output", "output_1", etc.

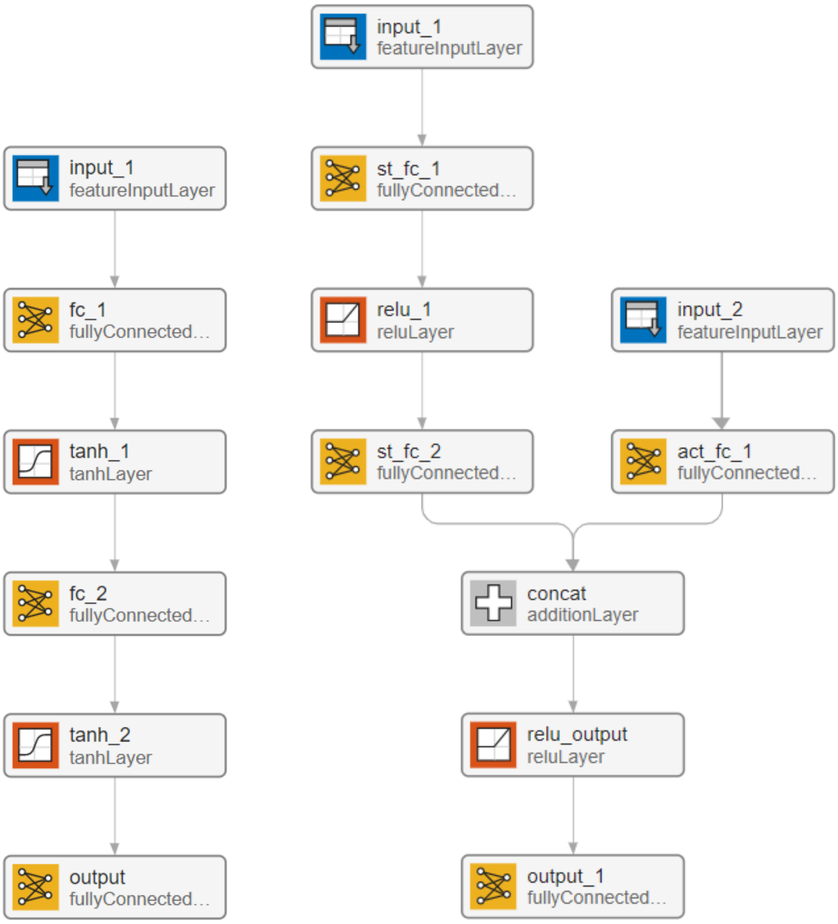


Figure 5.5 Actor (left) and critic (right) structure in Deep Network Designer

The following layer types are used in actor and critic NNs structure:

- Feature Input Layer as an input layer for scalar data set representing features;
- Fully Connected Layer as a hidden layer, that multiplies the input by a weight matrix and adds a bias vector;

The common weights initialization scheme, Glorot initialization, is used at each Fully Connected Layer. The initial weights are sampled independently, from the uniform distribution with zero mean and variance equal to $2/(InputSize + OutputSize)$. The biases are initialized as zeros.

- Tanh Layer, or hyperbolic tangent, as activation layer that applies the tanh function to the input and has bounded output in a range [-1,1]:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}; \tag{5.13}$$

- ReLU Layer, or rectified linear unit, performs the threshold operation to each element of input:

$$f(x) = \begin{cases} x, & x \geq 0, \\ 0, & x < 0. \end{cases} \tag{5.14}$$

- Addition Layer implements concatenation of the layers.

5.1.5 RL-training and its results

The RL-agent training process was organized using Reinforcement Learning Designer from Reinforcement Learning Toolbox [43]. The designed actor and critic NNs along with the built environment was imported prior to the training. Different amounts of noise, actor and critic learning rates and limitations for gradients were tested out. The most suitable properties and training options, that resulted in acceptable solution, are given in Table 5.5, Table 5.6. In role of execution environment CPU was used.

Table 5.5 Agent, actor and critic, and noise properties

Parameter	Value
Sample time, s	10^{-3}
Discount factor	0,99
Batch size	128
Experience buffer length	10^{6}
Target smooth factor	10^{-3}
Actor learn rate	10^{-3}
Actor gradient threshold	1
Critic learn rate	10^{-3}
Critic gradient threshold	1
Standard deviation of OU noise	$2\cdot10^{-2}$
Mean	0

Table 5.6 Agent training options

Parameter	Value
Maximum episodes	10^{4}

Maximum episode length, time steps / s	10^4 / 10
Average window lengths, episodes	5
Stopping criteria: episode reward	$2{\cdot}10^4$
Save agent criteria: episode reward	$2{\cdot}10^4$

The actor and critic networks parameters are updated using Adam optimizer [50], [51], according to the equations (5.15)-(5.16):

$$\theta_{l+1} = \theta_l - \frac{\alpha m_l}{\sqrt{v_l} + \varepsilon}, \tag{5.15}$$

where

$$\begin{aligned} m_l &= \beta_1 m_{l-1} + (1 - \beta_1)\nabla E(\theta_l), \\ v_l &= \beta_2 v_{l-1} + (1 - \beta_2)[\nabla E(\theta_l)]^2, \end{aligned} \tag{5.16}$$

α — step size,

ε — offset,

β_1 and β_1 — decay rates,

$E(\theta_l)$ — expectation of the value θ_l.

The Adam optimization method is suitable for the problems with large datasets and high-dimensional parameter spaces. This robust method required little memory and applied for the wide range of optimization problems in the field of machine learning. The full description of Adam algorithm is given in [50].

The following settings were used for actor and critic parameters updates (Table 5.7).

Table 5.7 Adam optimizer options for actor and critic

Parameter	**Value**
Denominator offset, ε	10^{-8}
Gradient decay, β_1	0,9
Squared gradient decay, β_2	0,999
Gradient threshold method	l2norm
L2 regularization	10^{-4}

The training and validation of the agent can be also done using the MATLAB code from Appendix 5, provided in common view in [43].

Three PD-controllers 1, 2 and 4 (Table 3.4) had been improved with the added in parallel RL(DDPG)-agents. The RL-agents were saved and later used in simulations, being loaded from the workspace. The RL Toolbox, unfortunately, does not provide an option to export actor and critic weights, as array, or function. For this reason, further exploitation of the designed RL agent with the real-plant MLS2EM were impossible, since the plant works on drivers of 2017 and operates in MATLAB/Simulink R2017b, which does not support newer RL Toolbox. The designed RL-agents are provided in supplementary digital materials of the thesis according to the Table 5.8. The performance of the RL-agents will be discussed in Chapter 6.

Table 5.8 RL-agents and related PD-controllers

RL-agent name	Related PD-controller
c1_agent1	controller1
c2_agent1	controller2
c4_agent1	controller4

The training progress and simulation results for the RL agent, named "c1_agent1", trained in parallel with the PD-controller 1, are depicted in Figure 5.6, Figure 5.7. It is seen that training converges to maximum reward at the 17th episode. The simulated RL-agent works well and reaches the maximum reward in 10 of 10 cases.

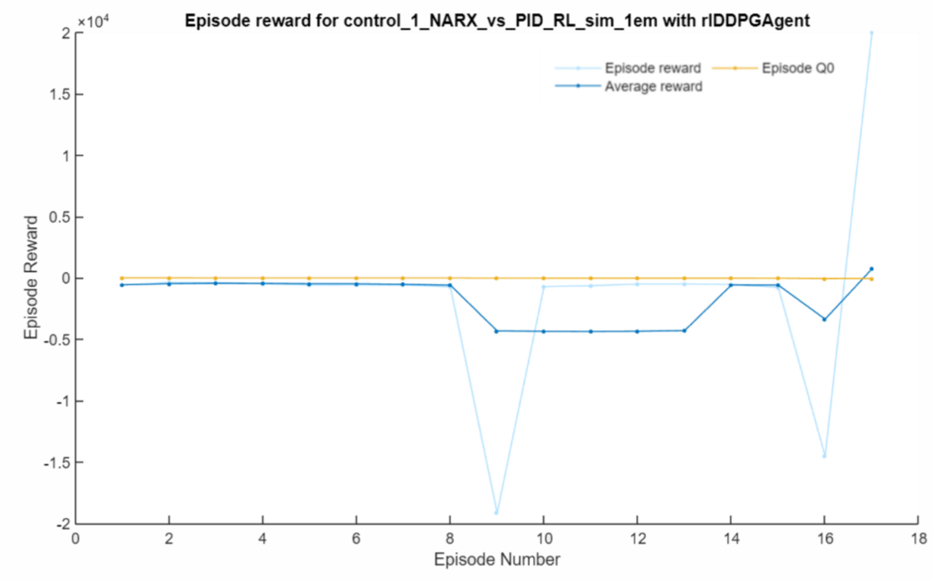


Figure 5.6 Training progress of RL-agent "c1_agent1"

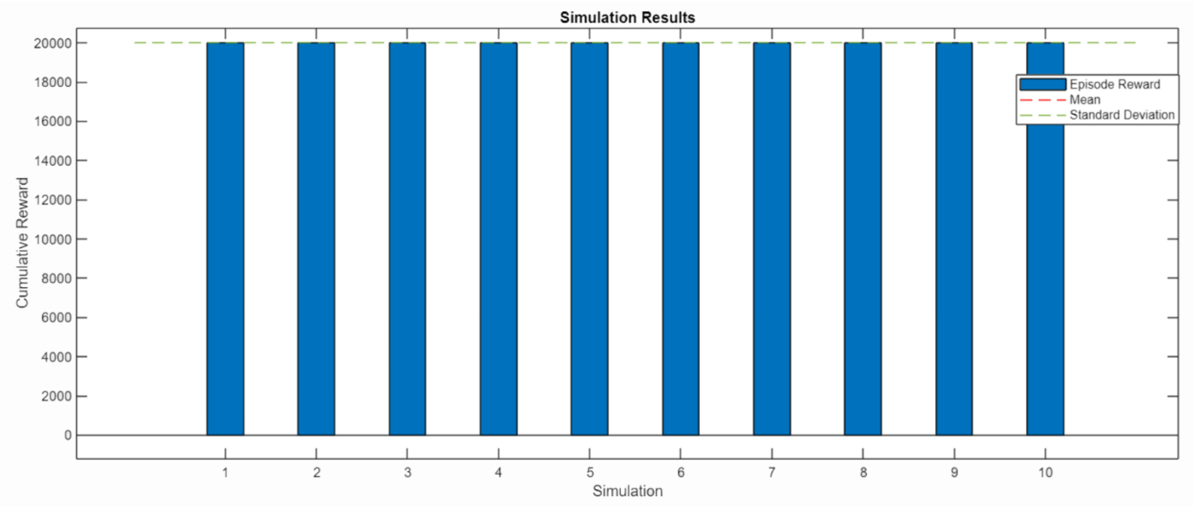


Figure 5.7 Simulation result of the trained RL-agent "c1_agent1"

Here, it is important to note, that the training with same parameters, based on the reward function (5.12) without the term r_3, i.e., without the time reward, results in a longer and less stable training (Figure 5.8). Since the parameters of actor and critic NNs are initialized randomly, each certain training goes differently, requires different number of episodes, and ends up with a little bit different solution. Without time reward the training process tends to diverge (Figure 5.9), the actor's output goes far away from the adequate control range, and the prognosed reward Q_0 continually decreases. The comparison of those three cases depending on the time reward is given in Table 5.9.

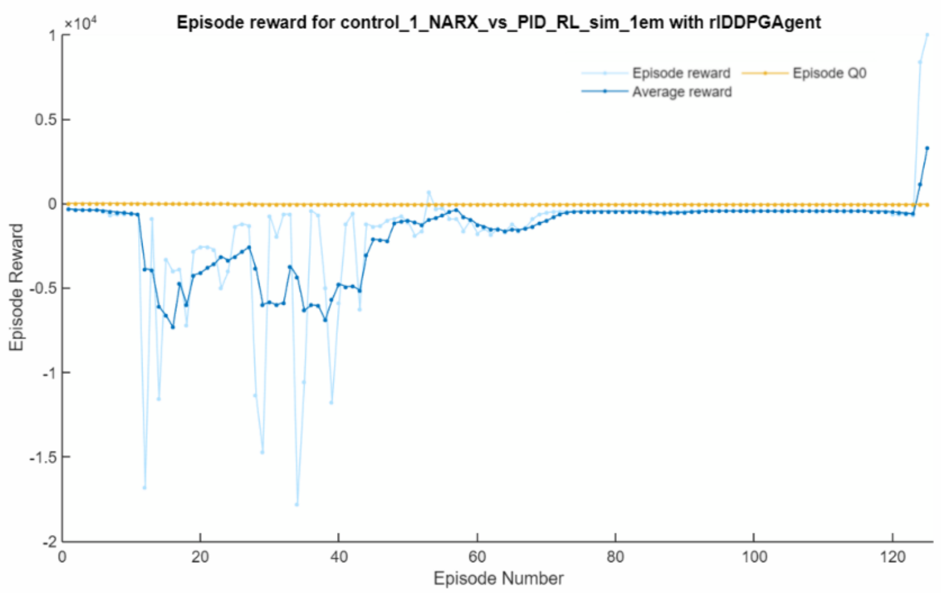


Figure 5.8 Training of "c1_agent1" without time reward

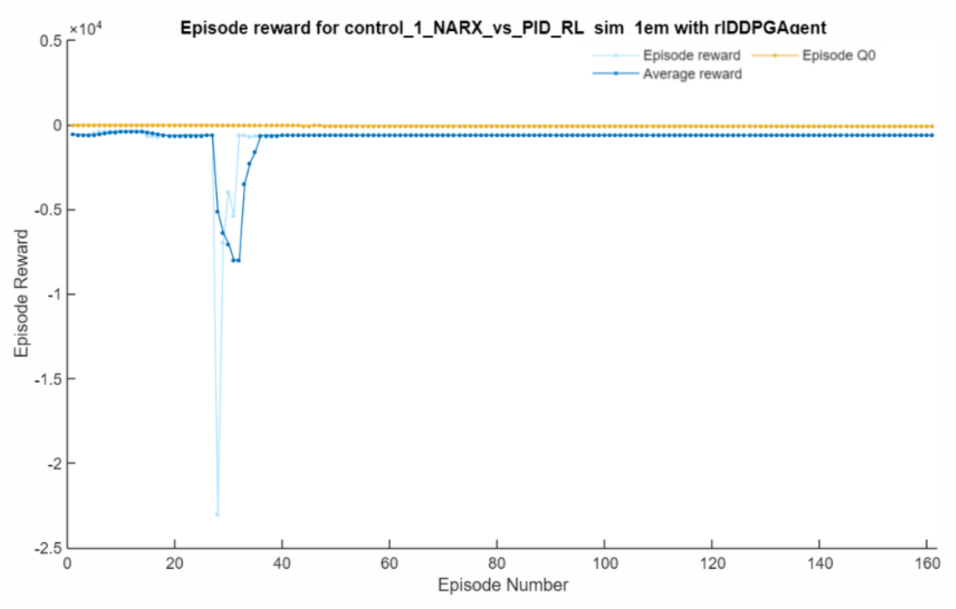


Figure 5.9 Diverging training of "c1_agent1" without time reward

Table 5.9 Training outcomes depending on the time reward

Figure	**Time reward value, r_3**	**Number of episodes**	**Training duration**	**Achieved reward**
Figure 5.7	1	45	33 min	$2{\cdot}10^4$ / $2{\cdot}10^4$
Figure 5.8	0	125	1 h 20 min	$1{\cdot}10^4$ / $1{\cdot}10^4$
Figure 5.9	0	160+	1 h+	-310 / $1{\cdot}10^4$

5.1.6 Graded learning for MLS2EM

The idea of graded learning with learning control tasks by levels of increasing difficulty, introduced in [42], can be applied for the design of the control solution for MLS. This idea will be realized so: the PD-controller output is supposed to be decreased from level to level using gain in a range [0, 1] in series with the PD-controller (Figure 5.1). The RL-agent is supposed to be retrained from one level of difficulty to another, transferring NNs weights and experience buffer from one training to another. The RL Toolbox provides an option to retrain the agent, rebuilding the environment for each training.

This approach allows to extend the working range and increase the stability of the RL-agent, and eventually almost cancel out the role of the PD-controller in position tracking. Besides this, each level trainings does not take much time (up to one hour).

The training schedule is presented in Table 5.10. The trained RL-agents are given in supplementary digital materials of the thesis work.

Table 5.10 Training schedule for “c1_agent1” with decreasing PD-controller output

Training number	Gain for PD-controller output	Trained RL-agent name
1	1	c1_agent1_100
2	0,75	c1_agent1_075
3	0,5	c1_agent1_050
4	0,25	c1_agent1_025
5	0,15	c1_agent1_015
6	0,1	c1_agent1_010

5.1.7 Graded learning on the example of Water Tank model

In this section, the graded learning will be realized on the classic example of the level control in Water Tank model (Figure 5.10), presented in [52] and RL Toolbox User’s Guide [42, pp.1-19 - 1-26]. And then compared to the training results without graded learning.

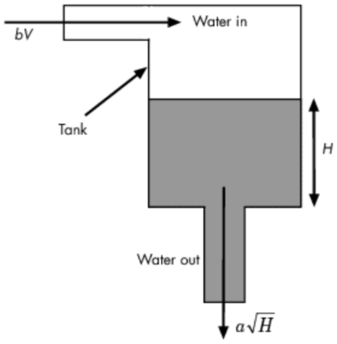


Figure 5.10 Water Tank model diagram [52]

The height of the water in the tank over time $t, t \geq 0$, is described by the equation (5.17):

$$\frac{dH}{dt} = \frac{1}{A}\left(bV - a\sqrt{H}\right), \tag{5.17}$$

where

H — height of the water in the tank, m,

A — the cross-sectional area of the tank, m^2,

b — constant related to the flow rate into the tank,

a — constant related to the flow rate out of the tank,

V — voltage applied to the water pump, V.

The control of the water level is performed in a feedback loop using the PID-controller (Table 5.11). The PID-controller is tuned automatically in Simulink.

Table 5.11 The PID-controller parameters for the water tank model

Gain	Value
Proportional, K_p	1,94
Integral, K_i	0,28
Derivative, K_d	-0,36

The Water Tank Simulink model is given in
Figure 5.11, Figure 5.12.

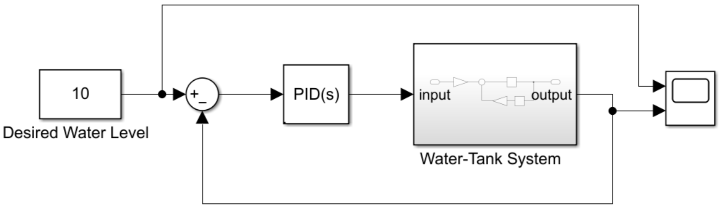


Figure 5.11 Water Tank Simulink model with PID-controller

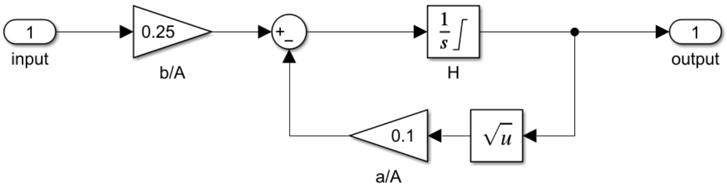


Figure 5.12 Interior of the Water Tank system

The environment and RL/DDPG-agent for the water level control were built according to the example in [43], [53], Figure 5.13, Figure 5.14.

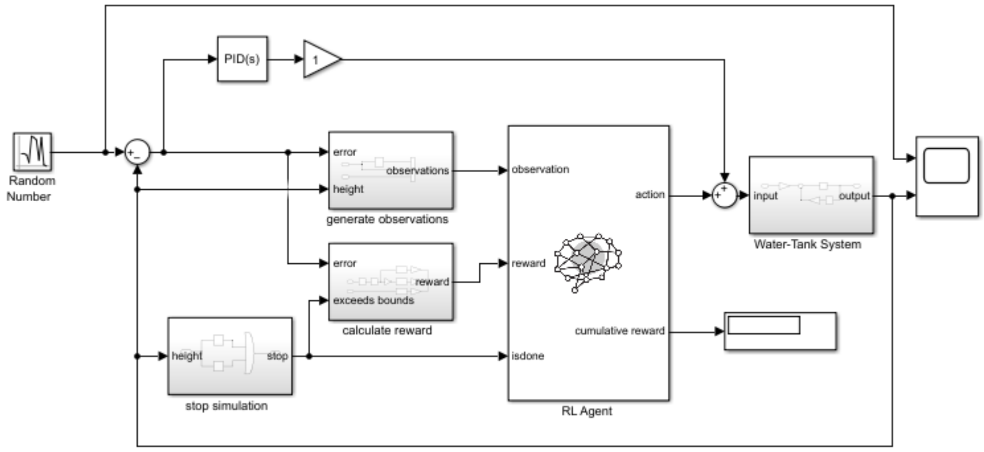


Figure 5.13 Water Tank Simulink model with the RL-agent

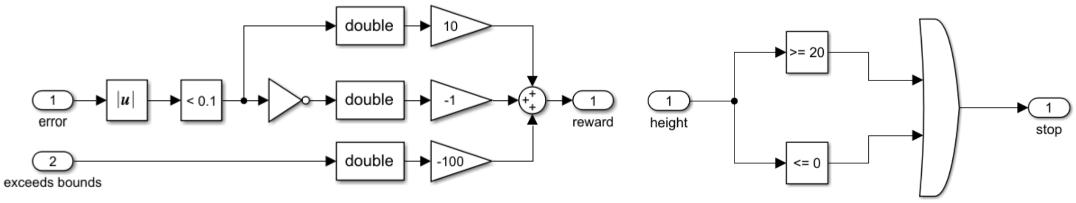


Figure 5.14 The reward block interior (left) and stopping criteria (right), bounds in meters

Also, since for the graded learning it was supposed to use RL Toolbox, but not the full code from [43], [53], it was not possible to vary the reference signal and the initial water level in the tank, using the reset function. Namely, the reason was that RL Toolbox does not suppose generating and reload of the environment in the beginning of each episode. Instead, the Random Number (mean 9, variance 5, sample time 40 s) was used as a reference signal in Water tank model (Figure 5.13). The initial water level was kept as 10 m. The training was performed under the same training conditions and with same actor and critic NNs structure as given in [43], [51], Table 5.12, Table 5.13, with sample time of 1 s and episode length of 200 s.

Table 5.12 Agent, actor and critic, and noise properties for Water Tank problem

Parameter	Value
Sample time, s	1
Discount factor	1
Batch size	64
Experience buffer length	10^6
Target smooth factor	10^{-3}
Actor hidden layers number	1
Actor hidden layers size, neurons	3
Actor output activation function	Tanh
Critic hidden layers size (state path), neurons	50, 25
Critic hidden layers size (action path), neurons	25
Actor learn rate	10^{-4}
Actor gradient threshold	1
Critic learn rate	10^{-3}
Critic gradient threshold	1
Standard deviation / Variance of OU noise	0,3
Variance decay rate	$1·10^{-5}$

Table 5.13 Agent training options for Water Tank problem

Parameter	Value
Maximum episodes	$5·10^3$
Maximum episode length, time steps / s	200 / 200
Average window lengths, episodes	20
Stopping criteria: average reward	800
Save agent criteria: average reward	800

The RL-agent “wt_agent1” was trained by steps with the PID-controller in parallel (trainings 1-5, Table 5.14). The training 6 is performed for “wt_agent2” with no PID-controller in parallel for comparison.

Table 5.14 Training schedule for Water Tank model

Training number	Gain for PID output	RL-agent name	Number of episodes	Training duration	Average reward
1	1	wt_agent1_100	51	10 min 23 s	812 / 800
2	0,75	wt_agent1_075	25	5 min 17 s	803 / 800
3	0,5	wt_agent1_050	23	4 min 58 s	823 / 800
4	0,25	wt_agent1_025	27	5 min 23 s	834 / 800
5	0	wt_agent1_000	110	22 min 6 s	804 / 800
6	No PID	wt_agent2	97	11 min 43 s	816 / 800

As it is seen from Table 5.14, the number of episodes on average is less in case of the graded learning (trainings 1-5). The training progress for trainings 1-6 is given in Appendix 6.

The performance of the trained agents "wt_agent1_000" and "wt_agent2" (both do not use PID-controller in parallel) is shown in Figure 5.15, Figure 5.16, Figure 5.17. During the tests 1-3 the Random Number with different properties was used as a reference signal. Two initial water levels were tested out: 5 m and 10 m. It is seen, that "wt_agent1_000" is more oscillatory but responds faster than "wt_agent2". Both RL-agents are stable to the disturbance, introduced in error signal at time of 30-40 s (tests 2-3), and have less overshoots and undershoots rather than the PID-controller. The MSE values, calculated for the tests, are given in Table 5.15. In all cases, the retrained agent "wt_agent1_000", using the graded learning, has shown lower MSE values.

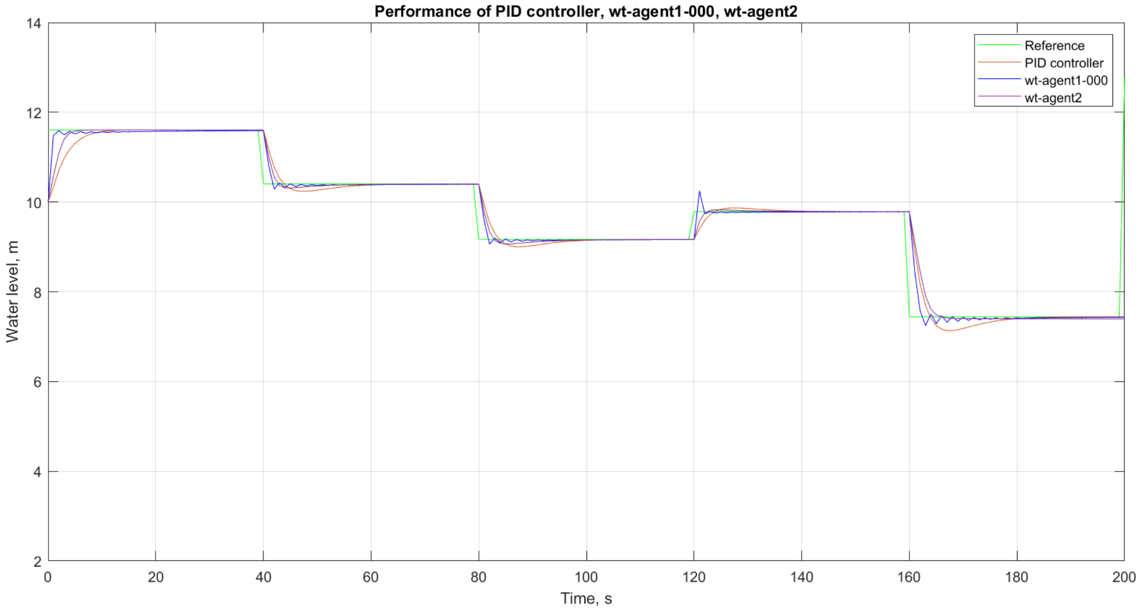


Figure 5.15 Test 1. Performance of the PID-controller, wt_agent1_000 and wt_agent2. Initial water level: 10 m. Reference signal: the Random Number (mean 9, variance 5, sample time 40 s). No disturbance added

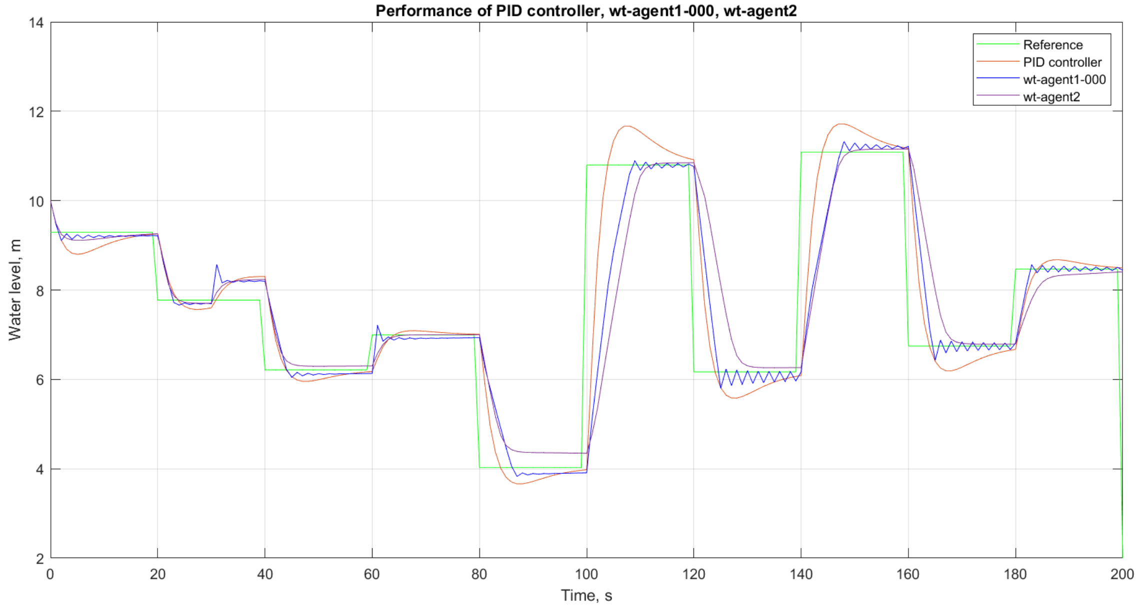


Figure 5.16 Test 2. Performance of the PID-controller, wt_agent1_000 and wt_agent2. Initial water level: 10 m. Reference signal: the Random Number (mean 6, variance 8, sample time 20 s). Disturbance: Pulse (amplitude 0.5, period 200 samples, pulse width 10 samples, phase delay 30 samples)

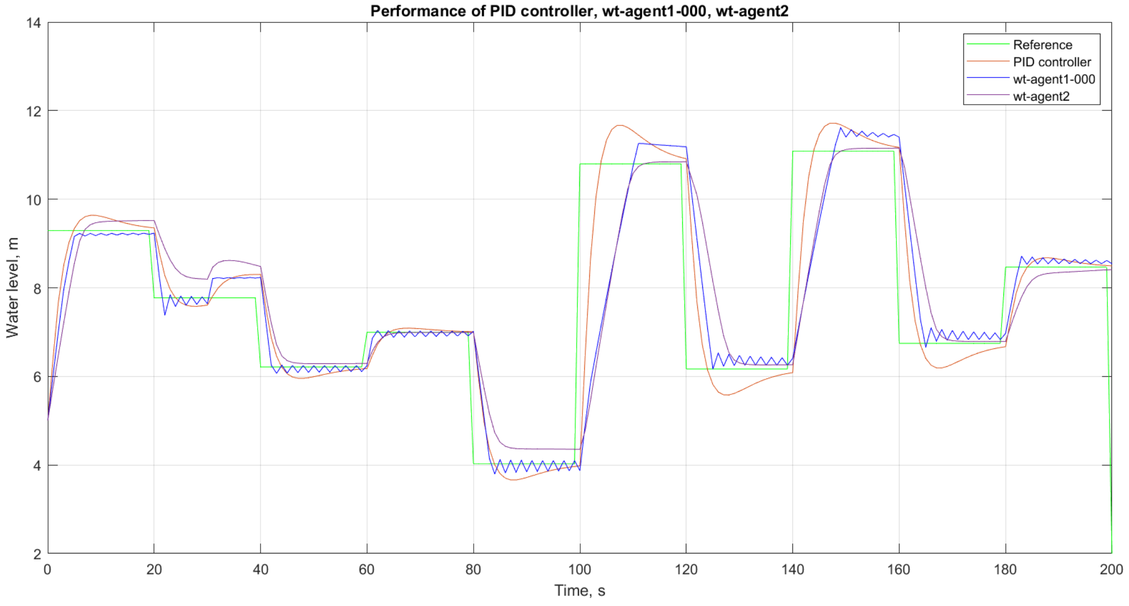


Figure 5.17 Test 3. Performance of the PID-controller, wt_agent1_000 and wt_agent2. Initial water level: 5 m. Reference signal: the Random Number (mean 6, variance 8, sample time 20 s). Disturbance: Pulse (amplitude 0.5, period 200 samples, pulse width 10 samples, phase delay 30 samples)

Table 5.15 Performance of the PID-controller, wt_agent1_000, wt_agent2

Test number	**Figure**	**MSE for PID**	**MSE for wt_agent1_000**	**MSE for wt_agent2**
1	Figure 5.15	0,2422	0,2086	0,2334
2	Figure 5.16	1,3029	1,7535	2,2646

3	Figure 5.17	1,4482	2,1679	2,5256

It worth to add, that, of course, Water Tank and MLS2EM are completely different systems. The Water Tank system has a big inertia, while the MLS2EM is very quickly responding to any change of the setpoint system. The Water Tank is stable system, that can be controlled by the easily tuned PID-controller, while control of the MLS2EM is a challenge due to its high instability.

6. SIMULATION EXPERIMENTS, RESULTS AND DISCUSSIONS

6.1 Simulation experiments results

6.1.1 Experiments for Controller 1

The comparison of performance of the PD-controllers, NARX-based controllers and RL-based controllers is given in the current Section 6.1.1, and the following Sections 6.1.2, 6.1.3. The experiments are divided into groups regarding the PD-controllers 1, 2 and 4 (Table 3.4) and associated with them NARX-based and RL-based controllers.

To begin, the first-three-seconds response of the PD-controller 1, related NARX-based controller and PD-controller 1 with RL-agent is shown in Figure 6.1, Figure 6.2 and Figure 6.3, respectively. The actual position value together with reference, velocity, currents values and controls by the electromagnets EM1 and EM2 are depicted in plots. The control is performed by the upper electromagnet EM1, the electromagnet EM2 is inactive for now.

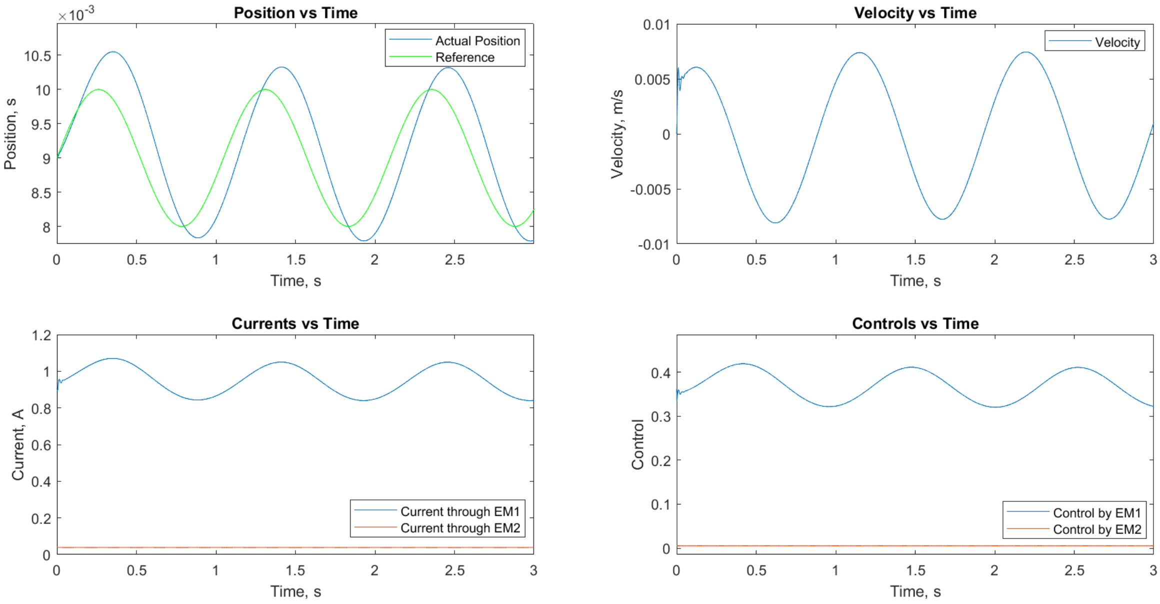


Figure 6.1 First-three-seconds response of the PD-controller 1

It is seen from the position plots, that the ball follows the reference trajectory the most accurately in case of the PD-controller 1, improved with RL-agent (Figure 6.3). The NARX-based controller responds faster, than the PD-controller 1, but the ball deviates from the trajectory along the vertical axis more (Figure 6.1, Figure 6.2). The ball, controlled by the PD+RL controller has bigger velocity in the beginning of the control, that’s why there is a small shift in position (along the vertical axis) in the very beginning of the experiment (Figure 6.3). In case of the NARX-based controller (Figure 6.2), there are small oscillations in the velocity, current and control signals, which may become a problem in overall performance of the controller.

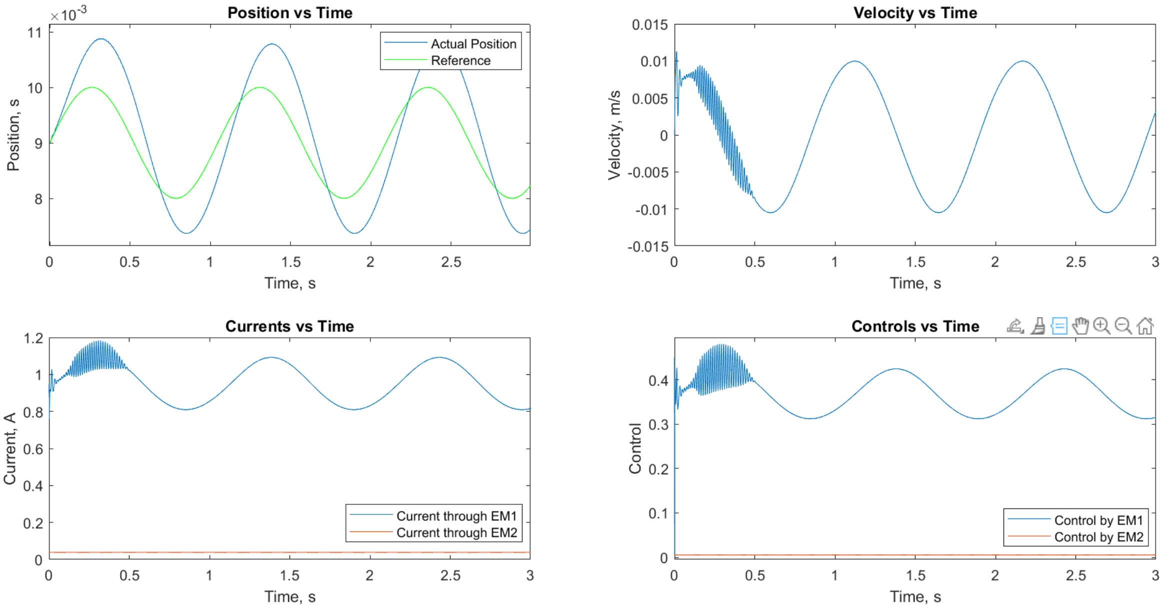


Figure 6.2 First-three-seconds response of the NARX-based controller 1

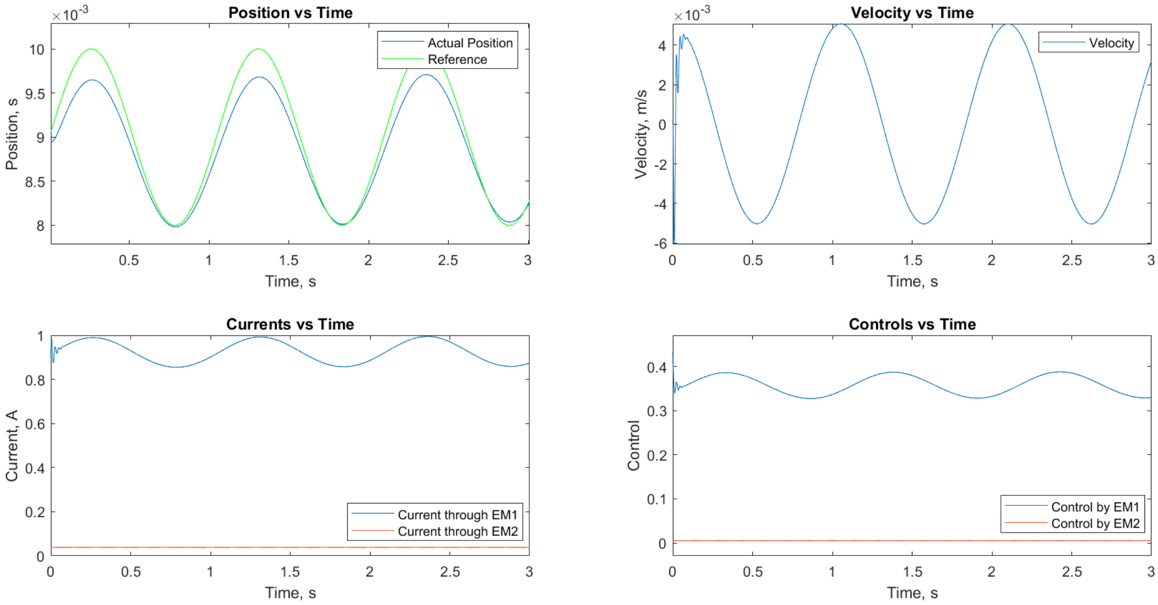


Figure 6.3 First-three-seconds response of the PD-controller 1 with RL-agent in parallel

Further, the controllers will be tested out in a number of experiments. The experiments are split into sets (Table 6.1), for convenience.

The comparison of performance of the PD, PD+RL and NARX-based controllers is done based on calculating of the mean squared error (MSE):

$$MSE = \frac{1}{k}\sum_{k=1}^{n} e^2, \tag{6.1}$$

where

e – position error, m, calculated for each controller separately,

n – number of samples, $n = 10001$.

The MATLAB code for calculating MSE and building plots is given in Appendix 7.

The initial conditions of MLS2EM are not changed (3.3), the initial position of the ball is $9{\cdot}10^{-3}$ m. All the analysis, including calculation of MSE values, is done for the data collected during the simulations of duration of 30 s. The results of the performance of the PD-controller 1, NARX-based controller and PD-controller with RL-agent in parallel are given in Table 6.2 and plots are added into Appendix 8. The cases in Table 6.2, where the tracking control was failed, in other words, the ball was lost by the controller, are pointed out with the orange color.

Table 6.1 Description of the sets of experiments, controller 1

Set number	**Experiment number**	**Description**
1	1-4	Original ball (Table 3.3) with mass of $5{,}71{\cdot}10^{-2}$ kg and diameter of $6{\cdot}10^{-2}$ m. Reference signal: Sine wave (amplitude, m; bias, m; frequency 6 rad/s; sample time $1{\cdot}10^{-2}$ s). No disturbance.
2	5-8	Original ball. Reference signal: Uniform Random Number (URN) (minimum, m; maximum, m; seed 7; sample time 3 s). No disturbance.
3	9-12	Original ball. Reference signal: Constant (value, m). No disturbance.
4	13-15	Small ball. Reference signals: Sine wave, URN, Constant. No disturbance.
5	16-18	Medium ball. Reference signals: Sine wave, URN, Constant. No disturbance.
6	19-21	Big ball. Reference signals: Sine wave, URN, Constant. No disturbance.

Table 6.1 continued

Set number	Experiment number	Description
7	22-24	Original ball. Reference signals: Sine wave, URN, Constant. Disturbance added to error at time 5 s, 15 s, 25 s: Pulse (amplitude $1\cdot10^{-3}$ m, period 10 s, pulse width 1 s, phase delay 5 s)
8	25-27	Original ball. Reference signals: Sine wave, URN, Constant. Disturbance added by EM2: Pulse (amplitude 1, period 2 s, pulse width 0,2 s, phase delay 0 s) with gain 0,4

Table 6.2 Performance of the PD-controller1, NARX-based controller1, PD-controller1+RL-agent

Exp#	Reference signal properties	PD	NARX	PD+RL
1	Sine wave ($1\cdot10^{-3}$, $9\cdot10^{-3}$)	$2,65\cdot10^{-6}$	$4,15\cdot10^{-7}$	$1,71\cdot10^{-8}$
2	Sine wave ($2\cdot10^{-3}$, $9\cdot10^{-3}$)	$1,15\cdot10^{-6}$	$1,59\cdot10^{-6}$	$5,52\cdot10^{-8}$
3	Sine wave ($2\cdot10^{-3}$, $1\cdot10^{-2}$)	$8,22\cdot10^{-6}$	$2,09\cdot10^{-6}$	$6,01\cdot10^{-8}$
4	Sine wave ($2\cdot10^{-3}$, $8\cdot10^{-3}$)	$1,68\cdot10^{-6}$	$6,28\cdot10^{-5}$	$5,53\cdot10^{-8}$
5	URN ($9\cdot10^{-3}$, $1,05\cdot10^{-2}$)	$3,22\cdot10^{-6}$	$1,61\cdot10^{-6}$	$1,47\cdot10^{-8}$
6	URN ($9\cdot10^{-3}$, $1,1\cdot10^{-2}$)	$4,55\cdot10^{-5}$	$2,28\cdot10^{-6}$	$2,10\cdot10^{-8}$
7	URN ($7,5\cdot10^{-3}$, $9,5\cdot10^{-3}$)	$6,34\cdot10^{-7}$	$7,43\cdot10^{-7}$	$1,05\cdot10^{-8}$
8	URN ($7,5\cdot10^{-3}$, $1,05\cdot10^{-2}$)	$1,72\cdot10^{-6}$	$1,13\cdot10^{-6}$	$2,39\cdot10^{-8}$
9	Constant ($9\cdot10^{-3}$)	$6,64\cdot10^{-13}$	$6,44\cdot10^{-10}$	$4,17\cdot10^{-9}$
10	Constant ($1,2\cdot10^{-2}$)	$6,28\cdot10^{-5}$	$4,87\cdot10^{-6}$	$4,48\cdot10^{-8}$
11	Constant ($1,05\cdot10^{-2}$)	$7,22\cdot10^{-5}$	$3,55\cdot10^{-6}$	$1,93\cdot10^{-8}$
12	Constant ($7\cdot10^{-3}$)	$3,69\cdot10^{-6}$	$4,68\cdot10^{-5}$	$7,27\cdot10^{-9}$
13	Sine wave ($1\cdot10^{-3}$, $9\cdot10^{-3}$)	$1,56\cdot10^{-5}$	$7,98\cdot10^{-5}$	$5,96\cdot10^{-8}$
14	URN ($7,5\cdot10^{-3}$, $1,05\cdot10^{-2}$)	$1,56\cdot10^{-5}$	$8,04\cdot10^{-5}$	$6,78\cdot10^{-8}$
15	Constant ($9\cdot10^{-3}$)	$1,57\cdot10^{-5}$	$7,91\cdot10^{-5}$	$4,07\cdot10^{-8}$
16	Sine wave ($1\cdot10^{-3}$, $9\cdot10^{-3}$)	$4,91\cdot10^{-6}$	$7,81\cdot10^{-5}$	$3,15\cdot10^{-8}$
17	URN ($7,5\cdot10^{-3}$, $1,05\cdot10^{-2}$)	$4,88\cdot10^{-6}$	$7,96\cdot10^{-5}$	$3,80\cdot10^{-8}$
18	Constant ($9\cdot10^{-3}$)	$4,90\cdot10^{-6}$	$7,84\cdot10^{-5}$	$1,60\cdot10^{-8}$
19	Sine wave ($1\cdot10^{-3}$, $9\cdot10^{-3}$)	$1,59\cdot10^{-6}$	$7,71\cdot10^{-5}$	$2,31\cdot10^{-8}$
20	URN ($7,5\cdot10^{-3}$, $1,05\cdot10^{-2}$)	$1,79\cdot10^{-6}$	$7,11\cdot10^{-5}$	$2,98\cdot10^{-8}$
21	Constant ($9\cdot10^{-3}$)	$1,51\cdot10^{-6}$	$1,60\cdot10^{-6}$	$9,00\cdot10^{-9}$
22	Sine wave ($1\cdot10^{-3}$, $9\cdot10^{-3}$)	$5,84\cdot10^{-7}$	$6,38\cdot10^{-5}$	$9,64\cdot10^{-8}$
23	URN ($7,5\cdot10^{-3}$, $1,05\cdot10^{-2}$)	$1,75\cdot10^{-6}$	$6,66\cdot10^{-5}$	$1,14\cdot10^{-7}$
24	Constant ($9\cdot10^{-3}$)	$3,43\cdot10^{-7}$	$4,14\cdot10^{-7}$	$8,32\cdot10^{-8}$
25	Sine wave ($1\cdot10^{-3}$, $9\cdot10^{-3}$)	$3,13\cdot10^{-7}$	$5,04\cdot10^{-7}$	$1,69\cdot10^{-8}$
26	URN ($7,5\cdot10^{-3}$, $1,05\cdot10^{-2}$)	$2,63\cdot10^{-6}$	$1,18\cdot10^{-6}$	$2,37\cdot10^{-8}$
27	Constant ($9\cdot10^{-3}$)	$2,34\cdot10^{-8}$	$3,11\cdot10^{-8}$	$4,20\cdot10^{-9}$

The experiments 1-27 showed that:

- PD-controller has a good performance only near its operating point of $9\cdot10^{-3}$ m;
- PD-control is quite slow, and the most suitable for the constant reference signal;
- NARX-based controller can replicate the behavior of the PD-controller, the performance of PD and NARX are comparable, but NARX has faster, more accurate, but more oscillating response;
- NARX-controller worse handles with abrupt changes of the reference signal rather than PD-controller, but at the same time has wider operating range, especially in the region farther from the electromagnet EM1;

- RL agent improves PD-controller performance so that MSE value is decreased on average by two orders for all types of the reference signal;
- PD+RL controller has bigger settling time comparing to the PD-controller working near operating point, but has smaller steady-state error out of the PD-controller operating range;
- PD+RL controller is stable against abrupt changes of the reference signal, and has significantly wider operating range rather than PD- and NARX-based controllers;
- PD+RL tends to eliminate an error over time approximately by the time 20 s.

6.1.2 Experiments for Controller 2

In this Section the differential mode PD-controller 2 (Table 3.4) is compared with the NARX-based controller, consisting of two NARX networks, and improved PD-controller 2 with RL-agent in parallel. The sets of experiments are described in Table 6.3, and the MSE is given in Table 6.4, the plots are given in Appendix 9.

Table 6.3 Description of the sets of experiments, controller 2

Set number	Experiment number	Description
1	28-31	Original ball (Table 3.3) with mass of $5{,}71\cdot10^{-2}$ kg and diameter of $6\cdot10^{-2}$ m. Reference signals: Sine wave (amplitude, m; bias, m; frequency 6 rad/s; sample time $1\cdot10^{-2}$ s), Uniform Random Number (URN) (minimum, m; maximum, m; seed 7; sample time 1 s). Constant (value, m). No disturbance.
2	32-34	Small ball. Reference signals: Sine wave, URN, Constant. No disturbance.
3	35-36	Medium ball (35), big ball (36). Reference signals: Sine wave. No disturbance.
4	37-39	Original ball. Reference signals: Sine wave, URN, Constant. Disturbance added to error at time 5 s, 15 s, 25 s: Pulse (amplitude $1\cdot10^{-3}$ m, period 10 s, pulse width 1 s, phase delay 5 s)
5	40-42	Original ball. Reference signals: Sine wave, URN, Constant. Disturbance added to the control action of EM1: Pulse (amplitude 1, period 4 s, pulse width 0,2 s, phase delay 0 s) with gain $2\cdot10^{-2}$ Disturbance added to the control action of EM2: Pulse (amplitude 1, period 4 s, pulse width 0,2 s, phase delay 2 s) with gain 0,2

Table 6.4 Performance of the PD-controller2, NARX-based controller2, PD-controller2+RL-agent

Exp#	Reference signal properties	PD, MSE	NARX, MSE	PD+RL, MSE
28	Sine wave ($1\cdot10^{-3}$, $9\cdot10^{-3}$)	$3,59\cdot10^{-7}$	$5,46\cdot10^{-7}$	$2,96\cdot10^{-8}$
29	URN ($7,5\cdot10^{-3}$, $1,05\cdot10^{-2}$)	$9,84\cdot10^{-7}$	$1,05\cdot10^{-4}$	$4,55\cdot10^{-8}$
30	URN ($8,5\cdot10^{-3}$, $9,5\cdot10^{-3}$)	$1,14\cdot10^{-7}$	$1,37\cdot10^{-7}$	$2,97\cdot10^{-8}$
31	Constant ($9\cdot10^{-3}$)	$1,47\cdot10^{-10}$	$5,29\cdot10^{-8}$	$2,79\cdot10^{-8}$
32	Sine wave ($1\cdot10^{-3}$, $9\cdot10^{-3}$)	$1,56\cdot10^{-5}$	$4,75\cdot10^{-5}$	$2,29\cdot10^{-9}$
33	URN ($8,5\cdot10^{-3}$, $9,5\cdot10^{-3}$)	$1,58\cdot10^{-5}$	$4,89\cdot10^{-5}$	$1,86\cdot10^{-9}$
34	Constant ($9\cdot10^{-3}$)	$1,56\cdot10^{-5}$	$4,94\cdot10^{-5}$	$6,16\cdot10^{-11}$
35	Sine wave ($1\cdot10^{-3}$, $9\cdot10^{-3}$)	$5,32\cdot10^{-6}$	$1,52\cdot10^{-5}$	$1,04\cdot10^{-8}$
36	Sine wave ($1\cdot10^{-3}$, $9\cdot10^{-3}$)	$7,74\cdot10^{-7}$	$1,75\cdot10^{-6}$	$1,93\cdot10^{-8}$
37	Sine wave ($1\cdot10^{-3}$, $9\cdot10^{-3}$)	$7,17\cdot10^{-7}$	$1,68\cdot10^{-5}$	$1,16\cdot10^{-7}$
38	URN ($8,5\cdot10^{-3}$, $9,5\cdot10^{-3}$)	$4,77\cdot10^{-7}$	$6,58\cdot10^{-7}$	$1,16\cdot10^{-7}$
39	Constant ($9\cdot10^{-3}$)	$3,88\cdot10^{-7}$	$1,62\cdot10^{-5}$	$1,15\cdot10^{-7}$
40	Sine wave ($1\cdot10^{-3}$, $9\cdot10^{-3}$)	$3,84\cdot10^{-7}$	$5,80\cdot10^{-7}$	$2,95\cdot10^{-8}$
41	URN ($8,5\cdot10^{-3}$, $9,5\cdot10^{-3}$)	$1,42\cdot10^{-7}$	$1,75\cdot10^{-7}$	$2,95\cdot10^{-8}$
42	Constant ($9\cdot10^{-3}$)	$2,04\cdot10^{-8}$	$8,88\cdot10^{-8}$	$2,77\cdot10^{-8}$

The experiments 28-42 showed that:

- Differential mode type of control performed by EM1 and EM2 is more difficult task;
- NARX-based control is quite oscillatory, weaker and less stable than PD-control, it does not handle with the change of the ball mass and diameter;
- PD+RL controller handles relatively good as in previous group of experiments, improving the PD controller and decreasing MSE by the one-two orders on average, although PD+RL controller has large settling time, comparing to the pure PD-control;
- PD+RL controller has the fastest response and the smallest deviation from the setpoint under disturbance.

6.1.3 Experiments for Controller 4

Since PD-controllers 1 and 4 are quite similar in structure, in this section only few experiments will be presented. The reference signals are changed. The results of experiments are given in Table 6.5, Table 6.6 and the tracking performance is given in Appendix 10.

Table 6.5 Description of the sets of experiments, controller 4

Set number	Experiment number	Description
1	43-46	Original ball (43) (Table 3.3) with mass of $5,71\cdot10^{-2}$ kg and diameter of $6\cdot10^{-2}$ m. Small (44), medium (45), big (46) balls. Reference signals: Sine wave (amplitude, m; bias, m; frequency 2 rad/s; sample time $1\cdot10^{-2}$ s), No disturbance.

Table 6.5 continued

Set number	Experiment number	Description
2	47	Original ball. Reference signals: URN (minimum, m; maximum, m; seed 7; sample time 3 s). Disturbance added to error at time 5 s, 15 s, 25 s: Pulse (amplitude $1\cdot10^{-3}$ m, period 10 s, pulse width 1 s, phase delay 5 s)
3	48	Original ball. Reference signals: Constant (value, m). Disturbance added by EM2: Pulse (amplitude 1, period 2 s, pulse width 0,2 s, phase delay 0 s) with gain 0,4

Table 6.6 Performance of the PD-controller4, NARX-based controller4, PD-controller4+RL-agent

Exp#	Reference signal properties	PD, MSE	NARX, MSE	PD+RL, MSE
43	Sine wave ($1\cdot10^{-3}$, $9\cdot10^{-3}$)	$9{,}12\cdot10^{-7}$	$1{,}16\cdot10^{-6}$	$3{,}83\cdot10^{-9}$
44	Sine wave ($1\cdot10^{-3}$, $9\cdot10^{-3}$)	$1{,}49\cdot10^{-5}$	$3{,}48\cdot10^{-5}$	$2{,}70\cdot10^{-8}$
45	Sine wave ($1\cdot10^{-3}$, $9\cdot10^{-3}$)	$4{,}39\cdot10^{-6}$	$3{,}43\cdot10^{-6}$	$2{,}74\cdot10^{-9}$
46	Sine wave ($1\cdot10^{-3}$, $9\cdot10^{-3}$)	$1{,}38\cdot10^{-6}$	$1{,}28\cdot10^{-6}$	$8{,}98\cdot10^{-10}$
47	URN ($7{,}5\cdot10^{-3}$, $1{,}05\cdot10^{-2}$)	$1{,}97\cdot10^{-6}$	$5{,}19\cdot10^{-5}$	$1{,}11\cdot10^{-7}$
48	Constant ($9{,}5\cdot10^{-3}$)	$1{,}35\cdot10^{-6}$	$1{,}51\cdot10^{-6}$	$4{,}43\cdot10^{-9}$

The experiments 43-48 showed again that:

- NARX-based controller is less stable to any change in MLS2EM or disturbance, than PD-controller;
- PD-controller, improved by RL-agent in parallel, performs better than pure PD-controller in all the experiments.

It is important to mention here, that the PD-controller 4, which is supposed to be used with the real plant, does not work in simulation with the MLS2EM model with sample time $5\cdot10^{-3}$ s. For this reason, sample time $1\cdot10^{-3}$ s was used. Unfortunately, it was also found, that the PD-controller 4 is not capable to perform a tracking control for the MLS2EM model with estimated parameters (see Subsection 3.1.4) in simulation with sample time either $1\cdot10^{-3}$ s or $5\cdot10^{-3}$ s. Thus, the original MLS2EM model was used. In future research, one can use the MLS2EM model with estimated parameters to design a control solution purely by the RL-agent and, than, utilize the trained RL-agent in the real-time experiment.

6.1.4 Experiments for Controller 1, Graded learning

The results of Section 5.1.6 will be discussed next. The PD-controller 1 will be compared with improved PD+RL controller, while the output of the PD-controller 1 will be decreasing. This is done to see, how much RL-agent can compensate the PD-control. Also, performance of the retrained RL-agents (Table 5.10) will be evaluated. The description of the sets of experiments is given in Table 6.7. The performance of the decreased-output PD controller 1, PD controller 1 with RL-agent in parallel, and PD-controller 1 with consequently retrained RL-agents in parallel is given in Table 6.8. Plots can be found in Appendix 11.

Table 6.7 Description of the sets of experiments, controller 1, graded learning

Set number	Experiment number	Description
1	49-54	Original ball (Table 3.3) with mass of $5{,}71\cdot10^{-2}$ kg and diameter of $6\cdot10^{-2}$ m. Reference signals: Sine wave (amplitude $1\cdot10^{-3}$ m; bias $9\cdot10^{-3}$ m; frequency 2 rad/s; sample time $1\cdot10^{-2}$ s). No disturbance.
2	55-57	Small (55), medium (56), big (57) balls. Reference signal: Sine wave (amplitude $1\cdot10^{-3}$ m; bias $9\cdot10^{-3}$ m; frequency 2 rad/s; sample time $1\cdot10^{-2}$ s). No disturbance.
3	58-59	Original ball. Reference signal: URN (minimum $8\cdot10^{-3}$ m; maximum $1\cdot10^{-2}$ m; seed 7; sample time 2 s). Disturbance added to error at time 5 s, 15 s, 25 s: Pulse (amplitude $1\cdot10^{-3}$ m, period 10 s, pulse width 1 s, phase delay 5 s)
4	60	Original ball. Reference signal: Constant (value, m). Disturbance added by EM2: Pulse (amplitude 1, period 2 s, pulse width 0,2 s, phase delay 0 s) with gain 0,4

Table 6.8 Performance of the decreased-output PD-controller 1, PD controller 1 + RL-agent and PD controller 1 + retrained RL-agent

Exp#	Gain for PD output	PD, MSE	PD+RL, MSE	PD+retrained RL, MSE
49	1	$7{,}79\cdot10^{-7}$	$1{,}69\cdot10^{-8}$	$1{,}69\cdot10^{-8}$
50	0,75	$1{,}19\cdot10^{-4}$	$1{,}33\cdot10^{-8}$	$1{,}16\cdot10^{-8}$
51	0,5	$1{,}20\cdot10^{-4}$	$2{,}38\cdot10^{-8}$	$3{,}19\cdot10^{-9}$
52	0,25	$1{,}21\cdot10^{-4}$	$5{,}07\cdot10^{-8}$	$2{,}91\cdot10^{-10}$
53	0,15	$1{,}21\cdot10^{-4}$	$1{,}20\cdot10^{-4}$	$1{,}79\cdot10^{-8}$
54	0,1	$1{,}21\cdot10^{-4}$	$1{,}20\cdot10^{-4}$	$7{,}13\cdot10^{-7}$
55	0,25	$1{,}20\cdot10^{-4}$	$2{,}37\cdot10^{-8}$	$4{,}01\cdot10^{-7}$
56	0,25	$1{,}21\cdot10^{-4}$	$3{,}24\cdot10^{-8}$	$9{,}10\cdot10^{-8}$
57	0,25	$1{,}21\cdot10^{-4}$	$4{,}05\cdot10^{-8}$	$2{,}43\cdot10^{-8}$
58	0,25	$1{,}21\cdot10^{-4}$	$9{,}77\cdot10^{-5}$	$1{,}05\cdot10^{-7}$
59	0,25	$1{,}24\cdot10^{-4}$	$1{,}11\cdot10^{-4}$	$1{,}09\cdot10^{-7}$
60	0,25	$1{,}21\cdot10^{-4}$	$3{,}87\cdot10^{-8}$	$2{,}37\cdot10^{-9}$

The experiments 49-60 showed that:

- PD-controller is very sensitive to any change in its output, or control action;
- RL-agent can compensate the PD-controller’s output decreased up to 25%, it can handle the change of the ball mass and diameter, although some abrupt changes in error destabilize the RL-agent control;

- Being retrained consequently, RL-agent is capable successfully to compensate the PD-controller’s output decreased up to 15%; the retrained RL-agents provide accurate tracking and handle the disturbance to some extent.

SUMMARY

The main goal of the master thesis was to design a control solution for the Magnetic levitation system with two electromagnets (MLS2EM), provided by INTECO company, which can surpass the performance of the existing in simulations PD-controllers.

It is clear from the Literature Overview, that the control of a highly nonlinear and unstable MLS is a topic of a great incessant interest. Many control solutions had been developed, including hybrid control methods, which allow to improve or surpass the simple PID-controller's performance. But most of them require a big engineering effort and difficult and tedious mathematical calculations. Nowadays, the intelligent methods become more and more popular due to its simplicity in realization and applicability to the various control problems.

In the present master thesis work, two control solutions were found and realized in simulation using MATLAB/Simulink with Deep Learning, Reinforcement Learning and Neural Network Toolboxes.

Originally, the idea was to fully replace the existing PD-controller by the NARX-based controller, which is trained on the datasets collected from the related PD-controller in open-loop. But after tests and simulation experiments it turned out, that the NARX-based controller cannot significantly surpass the PD-controller's performance, especially, in regard to stability (change of the ball mass, or added disturbance). Same can be said regarding the working region of the NARX-based controller. Although, it showed faster and more accurate response rather than the PD-controller to some extent. The ANNs approach was chosen due to its engineering simplicity, although in practice it appeared, that enormous number of trainings is needed to acquire the adequate parameters for the NN. The reason of this is the randomness of parameter initialization, which means that each certain training ends up in a different result. Moreover, unexpected training outcomes are quite often case in the work with NNs.

Since Supervised Learning method is dependent on the training datasets, and subsequently, on the certain PD-controller performance, it was decided to switch to Reinforcement Learning approach. The latter one was applied to develop a RL/DDPG-agent, capable to improve and stabilize the performance of the PD-controller in simulation, and also extend its operating range. Full replacement of the PD-controller by the RL-agent was not possible due to limitations existing in the MATLAB Reinforcement Learning Toolbox, such as impossibility to use batch normalization, or to add exploration noise to the NNs parameters directly. The intention of use of MATLAB was in its multitool environment, convenience, and also for the reason that the MLS2EM simulation models are provided by INTECO in MATLAB/Simulink. Potentially, in future, with the improvements in the RL Toolbox, the task of the full replacement of the PD-controller with the RL-agent may be fulfilled.

The comparison of the performance of the PD-controllers with the related NARX-based controllers and the PD-controllers, improved by RL-agents in parallel, was done for three control cases. Two of them consider control of the upper electromagnet EM1 only, and the third one considers control of both electromagnets EM1 and EM2 in differential mode.

In all cases of 60 simulation experiments, the PD+RL-agent controller showed the decrease of the MSE value one hundred times on average, comparing to that of the pure PD-controller. The good stability of the PD+RL-agent controller was proved adding the disturbance to the error signal, and to the controller's output, using EM2 for the pulse excitation as well. Besides, four balls configurations (mass and diameter) were tested out. While the PD-controller can handle somewhat this change, and the NARX-controller fails, the PD+RL-agent controller still performs an accurate position tracking.

Moreover, the RL-agent is capable to successfully compensate the decrease of the PD-controller's output up to 25%. Being retrained with the presented in the thesis graded learning method, the RL-agent can even compensate it up to 15%. This means, even in case when the PD-controller breaks up, and critically drops its control action, RL-agent can become a solution to uninterrupted system operation. The designed method of the improvement of the PD-controllers can be used in industry, where the PD-controllers are the most common and cheap control method. Since the mechanical wear of the equipment always takes place, the PD-controller, improved by the RL-agent, can continue to perform with no re-tuning.

Having newer hardware drivers for the MLS2EM in laboratory, one can transfer the RL-agent, pretrained in simulation, to control the real plant. The estimated parameters for MLS2EM, found in the thesis, can be utilized to decrease the discrepancy between the simulation model and the real-time system. Having pretrained RL-agent, the retraining can be performed online on the real-time system to achieve the desired performance.

KOKKUVÕTE

Antud magistritöö peamine eesmärk oli projekteerida juhtimislahendus ettevõtte INTECO poolt pakutava kahe elektromagnetiga magnetilise levitatsioonisüsteemi (MLS2EM) jaoks, mis võib ületada olemasolevates simulatsioonides kasutatavate PD-regulaatorite jõudlust.

Kirjanduse ülevaatest selgub, et väga mittelineaarse ja ebastabiilse MLS kontrollimine on teema, mis pakub suurt ja pidevat huvi. Välja on töötatud mitmeid juhtimislahendusi, sealhulgas hübriidjuhtimismeetodid, mis võimaldavad parandada või ületada lihtsa PID-regulaatori jõudlust. Kuid enamik neist nõuab suurt inseneripingutust ning raskeid ja tüütuid matemaatilisi arvutusi. Tänapäeval muutuvad intelligentsed meetodid üha populaarsemaks tänu nende lihtsale realiseerimisele ja rakendatavusele erinevate reguleerimisprobleemide puhul.

Käesolevas magistritöös leiti ja realiseeriti simulatsioonis kaks juhtimislahendust, kasutades MATLAB/Simulinki koos Deep Learning, Reinforcement Learning ja Neural Network Toolboxidega.

Algne idee oli olemasolev PD-regulaator täielikult välja vahetada NARX-põhise regulaatoriga, mis on välja õpetatud seotud PD-regulaatorilt avatud ahelas kogutud andmekogumite põhjal. Kuid pärast teste ja simulatsioonikatsetusi selgus, et NARX-põhine regulaator ei suuda märkimisväärselt ületada PD-regulaatori jõudlust, eriti stabiilsuse osas (palli massi muutus, või lisatud häire). Sama võib öelda ka NARX-põhise regulaatori tööpiirkonna kohta. Kuigi see näitas pigem kiiremat ja täpsemat reageerimist kui PD-regulaator mingil määral. ANN-meetod valiti selle tehnilise lihtsuse tõttu, kuigi praktikas selgus, et NNi jaoks sobivate parameetrite omandamiseks on vaja tohutult palju treeninguid. Selle põhjuseks on parameetrite initsialiseerimise juhuslikkus, mis tähendab, et iga konkreetne treening annab erineva tulemuse. Lisaks sellele on NN-idega tehtavas töös üsna sageli ette tulnud ootamatuid treeningtulemusi.

Kuna superviseeritud õppimise meetod sõltub treeningu andmekogumitest ja seejärel teatud PD-regulaatori jõudlusest, otsustati minna üle stiimulõpe (Reinforcement Learning) meetodile. Viimast rakendati RL/DDPG-agendi väljatöötamiseks, mis on võimeline parandama ja stabiliseerima PD-regulaatori jõudlust simulatsioonis ning laiendama selle tööpiirkonda. PD-regulaatori täielik asendamine RL-agendiga ei olnud võimalik MATLABi Reinforcement Learning Toolboxi piirangute tõttu, näiteks võimatus kasutada partiide normaliseerimist või lisada NNide parameetritele otse uurimismüra.

MATLABi kasutamise eesmärk oli selle multitööriistakeskkond, mugavus ja ka seetõttu, et MLS2EMi simulatsioonimudelid on INTECO poolt pakutud MATLAB/Simulinkis. Võimalik, et tulevikus, koos RL Toolboxi täiustustega, on võimalik täita ülesanne PD-regulaatori täielik asendamine RL-agentidega.

PD-regulaatorite ja nendega seotud NARX-põhiste regulaatorite ning paralleelselt RL-agentide abil täiustatud PD-regulaatorite toimivust võrreldi kolme juhtumi puhul. Kahes neist käsitletakse ainult ülemise elektromagneti EM1 juhtimist ja kolmandas mõlema elektromagneti EM1 ja EM2 juhtimist diferentsiaalrežiimis.

Kõigil 60 simulatsioonikatsete puhul näitas PD+RL-agendi regulaator keskmise ruudu vea (MSE) vähenemist keskmiselt sada korda, võrreldes puhta PD-regulaatoriga. PD+RL-agendi regulaatori head stabiilsust tõestati, kui veasignaalile ja regulaatori väljundile lisati häire, kasutades impulsside ergutamiseks ka EM2. Lisaks katsetati nelja pallikonfiguratsiooni (mass ja läbimõõt). Kuigi PD-regulaator saab selle muudatusega mõnevõrra hakkama ja NARX- põhise regulaator ebaõnnestub, teostab PD+RL-agendi regulaator siiski täpset asukoha jälgimist.

Lisaks sellele on RL-agent võimeline edukalt kompenseerima PD-regulaatori väljundi vähenemist kuni 25%. Kui RL-agent on ümber õpetatud käesolevas töös esitatud astmelise õppimise meetodiga, suudab ta seda isegi kuni 15% ulatuses kompenseerida. See tähendab, et isegi juhul, kui PD-regulaator laguneb ja jätab kriitiliselt oma juhtimismehhanismi välja, võib RL-agent olla lahenduseks süsteemi katkematu toimimise tagamiseks. Kavandatud PD-regulaatorite täiustamise meetodit saab kasutada tööstuses, kus PD-regulaatorid on kõige levinum ja odavam juhtimismeetod. Kuna seadmete mehaaniline kulumine on loomuulik, võib RL-agendi abil täiustatud PD-regulaator jätkata tööd ilma ümberhäälestamiseta.

Kui MLS2EMi uuemad riistvara draiverid on laboris olemas, saab simulatsioonis eelnevalt treenitud RL-agenti üle kanda tegeliku seadme kontrollimiseks. Lõputöös leitud MLS2EMi hinnangulisi parameetreid saab kasutada selleks, et vähendada erinevusi simulatsioonimudeli ja reaalajas toimiva süsteemi vahel. Kui RL-agent on eelnevalt treenitud, saab soovitud jõudluse saavutamiseks reaalajasüsteemis ümber treenida ka veebipõhiselt.

REFERENCES

[1] Inteco, "Magnetic Levitation System 2EM (MLS2EM), User's Manual," Krakow, Poland, 2008.

[2] P. Balko and D. Rosinova, "Modeling of Magnetic Levitation System," in *2017 21st International Conference on Process Control (PC)*, 2017, pp. 252–257, doi: 10.1109/PC.2017.7976222.

[3] C.-A. Bojan-Dragos, S. Preitl, R.-E. Precup, S. Hergane, E. G. Hughiet, and A.-I. Szedlak-Stinean, "State Feedback and Proportional-Integral-Derivative Control of a Magnetic Levitation System," in *2016 IEEE 14th International Symposium on Intelligent Systems and Informatics, Proceedings (SISY)*, 2016, vol. 2, no. 1, pp. 111–116, doi: 10.1109/SISY.2016.7601480.

[4] C.-A. Dragoş, R.-E. Precup, S. Preitl, E. M. Petriu, and M.-B. Rădac, "Control Solutions, Simulation and Experimental Results for a Magnetic Levitation Laboratory System." [Online]. Available: https://www.eurosim.info/fileadmin/user_upload_eurosim/EUROSIM_OA/Congress/2010/data/papers/155.pdf.

[5] I. Iswanto and A. Ma'arif, "Robust Integral State Feedback Using Coefficient Diagram in Magnetic Levitation System," *IEEE Access*, vol. 8, pp. 57003–57011, 2020, doi: 10.1109/ACCESS.2020.2981840.

[6] B. Bidikli and A. Bayrak, "A Self-Tuning Robust Full-State Feedback Control Design for the Magnetic Levitation System," *Control Eng. Pract.*, vol. 78, no. February, pp. 175–185, 2018, doi: 10.1016/j.conengprac.2018.06.017.

[7] D. Khimani, S. Karnik, and M. Patil, "Implementation of High Performance Nonlinear Feedback Control on Magnetic Levitation System," *IFAC-PapersOnLine*, vol. 51, no. 1, pp. 13–18, 2018, doi: 10.1016/j.ifacol.2018.05.003.

[8] D. Maji, M. Biswas, A. Bhattacharya, G. Sarkar, T. K. Mondal, and I. Dey, "MAGLEV System Modeling and LQR Controller Design in Real Time Simulation," in *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, 2016, pp. 1562–1567, doi: 10.1109/WiSPNET.2016.7566399.

[9] K. Anurag and S. Kamlu, "Design of LQR-PID Controller for Linearized Magnetic Levitation System," in *2018 2nd International Conference on Inventive Systems*

and Control (ICISC), 2018, pp. 444–447, doi: 10.1109/ICISC.2018.8399112.

[10] C.-A. Bojan-Dragos, M.-B. Radac, R.-E. Precup, E.-L. Hedrea, and O.-M. Tănăsoiu, "Gain-Scheduling Control Solutions for Magnetic Levitation Systems," *Acta Polytech. Hungarica*, vol. 15, no. 5, pp. 89–108, 2018, doi: 10.12700/APH.15.5.2018.5.6.

[11] A. S. Malik, I. Ahmad, A. U. Rahman, and Y. Islam, "Integral Backstepping and Synergetic Control of Magnetic Levitation System," *IEEE Access*, vol. 7, pp. 173230–173239, 2019, doi: 10.1109/ACCESS.2019.2952551.

[12] F. Adıgüzel, E. Dokumacılar, O. Akbatı, and T. Türker, "Design and Implementation of an Adaptive Backstepping Controller for a Magnetic Levitation System," *Trans. Inst. Meas. Control*, vol. 40, no. 8, pp. 2466–2475, 2018, doi: 10.1177/0142331217725146.

[13] J. R. Middala, "Modeling and Analysis of Magnetic Levitation System Using Fuzzy Logic Control," *Int. J. Sci. Dev. Res.*, vol. 2, no. 6, pp. 318–323, 2017.

[14] K. Czerwiński, A. Wojtulewicz, and M. Ławryńczuk, "Fuzzy Controller for Laboratory Levitation System: Real-time Experiments Using Programmable Logic Controller," *Int. J. Control. Autom. Syst.*, vol. 17, no. 6, pp. 1507–1514, 2019, doi: 10.1007/s12555-018-0394-1.

[15] E.-L. Hedrea, R.-E. Precup, C.-A. Bojan-Dragos, and C. Hedrea, "TP – Based Fuzzy Control Solutions for Magnetic Levitation Systems," in *2019 23rd International Conference on System Theory, Control and Computing (ICSTCC)*, 2019, pp. 809–814, doi: 10.1109/ICSTCC.2019.8886134.

[16] M. B. Unamboowe, "Dynamic Modeling of a Magnetic Levitation System," Tallinn University of Technology, 2021.

[17] V. Utkin and H. Lee, "Chattering Problem in Sliding Mode Control Systems," in *International Workshop on Variable Structure Systems, 2006. VSS'06.*, 2006, p. 1, doi: 10.1016/B978-008044613-4.50002-1.

[18] Y. Eroğlu, "Sliding Mode and PID Based Tracking Control of Magnetic Levitaion Plant and HIL Tests," Abdullah Gul University, 2016.

[19] D. Khimani and R. Rokade, "Implementation of Sliding Mode Control on Magnetic Levitation System," 2018, doi: 10.1109/ICAC3.2017.8318763.

[20] S. A. Al-Samarraie, B. F. Midhat, and R. A. B. Al-Deen, "Adaptive Sliding Mode Control for Magnetic levitation system," *Al-Nahrain J. Eng. Sci.*, vol. 21, no. 2, pp. 266–274, 2018, doi: https://doi.org/10.29194/NJES21020266.

[21] A. V. Starbino and S. Sathiyavathi, "Real-time Implementation of SMC–PID for Magnetic Levitation System," *Sadhana - Acad. Proc. Eng. Sci.*, vol. 44, no. 5, pp. 1–13, 2019, doi: https://doi.org/10.1007/s12046-019-1074-4.

[22] W. Bauer and J. Baranowski, "Fractional PIλD Controller Design for a Magnetic Levitation System," *Electronics*, vol. 9, no. 12, pp. 1–15, 2020, doi: 10.3390/electronics9122135.

[23] S. Pandey, V. Dourla, P. Dwivedi, and A. Junghare, "Introduction and Realization of Four Fractional-Order Sliding Mode Controllers for Nonlinear Open-Loop Unstable System: a Magnetic Levitation Study Case," *Nonlinear Dyn.*, vol. 98, no. 1, pp. 601–621, 2019, doi: 10.1007/s11071-019-05216-x.

[24] S. Pandey, P. Dwivedi, and A. S. Junghare, "A Novel 2-DOF Fractional-Order PIλ-Dμ Controller with Inherent Anti-Windup Capability for a Magnetic Levitation System," *AEU - Int. J. Electron. Commun.*, vol. 79, pp. 158–171, 2017, doi: 10.1016/j.aeue.2017.05.031.

[25] S. K. Swain, D. Sain, S. K. Mishra, and S. Ghosh, "Real Time Implementation of Fractional Order PID Controllers for a Magnetic Levitation Plant," *AEU - Int. J. Electron. Commun.*, vol. 78, pp. 141–156, 2017, doi: 10.1016/j.aeue.2017.05.029.

[26] A. Mughees and S. A. Mohsin, "Design and Control of Magnetic Levitation System by Optimizing Fractional Order PID Controller Using Ant Colony Optimization Algorithm," *IEEE Access*, vol. 8, pp. 116704–116723, 2020, doi: 10.1109/ACCESS.2020.3004025.

[27] P. Roy and B. K. Roy, "Sliding Mode Control Versus Fractional-Order Sliding Mode Control: Applied to a Magnetic Levitation System," *J. Control. Autom. Electr. Syst.*, vol. 31, no. 3, pp. 597–606, 2020, doi: 10.1007/s40313-020-00587-8.

[28] L. Alkurawy and K. G. Mohammed, "Model Predictive Control of Magnetic Levitation System," *Int. J. Electr. Comput. Eng.*, vol. 10, no. 6, pp. 5802–5812, 2020, doi: 10.11591/ijece.v10i6.pp5802-5812.

[29] M. Sherif, D. Victor, and A. El-Badawy, "Real-Time Control of a Magnetic

Levitation System," in *2019 31st International Conference on Microelectronics (ICM)*, 2019, pp. 280–283, doi: 10.1109/ICM48031.2019.9021705.

[30] G. M. K. B. Karunasena, H. D. N. S. Priyankara, and B. G. D. A. Madhusank, "Artificial Neural Network vs PID Controller for Magnetic Levitation System," *Int. J. Innov. Sci. Res. Technol.*, vol. 5, no. 7, pp. 505–511, 2020, doi: 10.38124/ijisrt20jul432.

[31] W. Yang, F. Meng, S. Meng, S. Man, and A. Pang, "Tracking Control of Magnetic Levitation System Using Model-Free RBF Neural Network Design," *IEEE Access*, vol. 8, pp. 204563–204572, 2020, doi: 10.1109/ACCESS.2020.3037352.

[32] R. Kumar, S. Srivastava, and J. R. P. Gupta, "Comparative Study of Neural Networks for Control of Nonlinear Dynamical Systems with Lyapunov Stability-Based Adaptive Learning Rates," *Arab. J. Sci. Eng.*, vol. 43, no. 6, pp. 2971–2993, 2018, doi: 10.1007/s13369-017-3034-9.

[33] J. de Jesús Rubio, L. Zhang, E. Lughofer, P. Cruz, A. Alsaedi, and T. Hayat, "Modeling and Control with Neural Networks for a Magnetic Levitation System," *Neurocomputing*, vol. 227, no. February 2016, pp. 113–121, 2017, doi: 10.1016/j.neucom.2016.09.101.

[34] A. Piłat and A. Turnau, "Neural Adapted Controller Learned On-line in Real-Time," *IFAC Proc. Vol.*, vol. 42, no. 13, pp. 47–52, 2009, doi: 10.3182/20090819-3-pl-3002.00010.

[35] H. Alimohammadi, B. B. Alagoz, A. Tepljakov, K. Vassiljeva, and E. Petlenkov, "A NARX Model Reference Adaptive Control Scheme: Improved Disturbance Rejection Fractional-Order PID Control of an Experimental Magnetic Levitation System," *Algorithms*, vol. 13, no. 8, 2020, doi: 10.3390/A13080201.

[36] A. K. Piłat, J. Źrebiec, and B. Sikora, "Neural Velocity Observer Trained with Experimental Data Supporting Stabilization of Magnetically Levitating Sphere," in *2019 12th Asian Control Conference (ASCC)*, 2019, pp. 214–219.

[37] T. Huang, Y. Liang, X. Ban, J. Zhang, and X. Huang, "The Control of Magnetic Levitation System Based on Improved Q-network," in *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2019, pp. 191–197, doi: 10.1109/SSCI44817.2019.9002980.

[38] Y. Yang, X. Ban, X. Huang, and C. Shan, "A Dueling-Double-Deep Q-Network

Controller for Magnetic Levitation Ball System," in *2020 39th Chinese Control Conference (CCC)*, 2020, vol. 2020-July, pp. 1885–1890, doi: 10.23919/CCC50068.2020.9189157.

[39] T. P. Lillicrap *et al.*, "Continuous Control with Deep Reinforcement Learning," in *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, 2016, p. 14, doi: 10.48550/arXiv.1509.02971.

[40] S. Wongsa and N. Kowkasai, "Deep Deterministic Policy Gradient for Magnetic Levitation Control," in *2020 17th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, 2020, pp. 796–799, doi: 10.1109/ECTI-CON49241.2020.9158096.

[41] "Better Exploration with Parameter Noise." https://openai.com/blog/better-exploration-with-parameter-noise/ (accessed May 01, 2022).

[42] R. Siraskar, "Reinforcement Learning for Control of Valves," *Mach. Learn. with Appl.*, vol. 4, p. 100030, 2021, doi: 10.1016/j.mlwa.2021.100030.

[43] The MathWorks Inc., "Reinforcement Learning Toolbox™ User's Guide," 2020.

[44] "Design Time Series NARX Feedback Neural Networks - MATLAB & Simulink - MathWorks United Kingdom." https://uk.mathworks.com/help/deeplearning/ug/design-time-series-narx-feedback-neural-networks.html (accessed May 01, 2022).

[45] "Levenberg-Marquardt Backpropagation - MATLAB trainlm - MathWorks United Kingdom." https://uk.mathworks.com/help/deeplearning/ref/trainlm.html?s_tid=doc_ta (accessed May 01, 2022).

[46] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *32nd Int. Conf. Mach. Learn. ICML 2015*, vol. 1, pp. 448–456, 2015, doi: 10.48550/arXiv.1502.03167.

[47] "Batch Normalization Layer - MATLAB - MathWorks United Kingdom." https://uk.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.batchnormalizationlayer.html (accessed May 01, 2022).

[48] "Options for DDPG Agent. Noise Model - MATLAB - MathWorks United Kingdom." https://uk.mathworks.com/help/reinforcement-

learning/ref/rlddpgagentoptions.html?s_tid=doc_ta (accessed May 01, 2022).

[49] M. H. Beale, M. T. Hagan, and H. B. Demuth, "Deep Learning Toolbox^{TM} User's Guide," 2020.

[50] D. P. Kingma and J. L. Ba, "Adam: A Method for Stochastic Optimization," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015, pp. 1–15, doi: 10.48550/arXiv.1412.6980.

[51] "Training Options for Adam Optimizer - MATLAB - MathWorks United Kingdom." https://uk.mathworks.com/help/deeplearning/ref/nnet.cnn.trainingoptionsadam .html?searchHighlight=adam (accessed May 01, 2022).

[52] "Watertank Simulink Model - MATLAB & Simulink - MathWorks United Kingdom." https://uk.mathworks.com/help/slcontrol/gs/watertank-simulink-model.html (accessed May 01, 2022).

[53] "Create Simulink Environment and Train Agent - MATLAB & Simulink - MathWorks United Kingdom." https://uk.mathworks.com/help/reinforcement-learning/ug/create-simulink-environment-and-train-agent.html?searchHighlight (accessed May 02, 2022).

APPENDICES

Appendix 1 MATLAB code for training NARX network

```
% Preparing data

u=input1'; % input data
y=output'; % target data

% Data conversion

u = con2seq(u);
y = con2seq(y);

% Time delays

d1 = [1:2];
d2 = [1:2];

% Design of NARX network

narx_net1 = narxnet(d1,d2,10);
narx_net1.divideFcn = '';
narx_net1.trainParam.min_grad = 1e-10;
narx_net1.trainParam.epochs = 300;
[p,Pi,Ai,t] = preparets(narx_net1,u,{},y);

% Training NARX network

narx_net1 = train(narx_net1,p,t,Pi);

% Open-loop to closed-loop modification

narx_net1_closed = closeloop(narx_net1);

%view(narx_net1)
%view(narx_net1_closed)

% Get a controller

gensim(narx_net1_closed, 0.001)

% Simulate closed-loop NARX and compare with PD output

ou_sim1=sim(narx_net1_closed, u);
ou_sim1=cell2mat(ou_sim1);
ou_sim1=ou_sim1';

% Fit

mse = (output1-ou_sim1).^2;
mean_mse=mean(mse);

% Plots

figure('WindowState','maximized','Color',[1 1 1]);
plot(t1, output1'); % pd output
hold on;
plot(t1, ou_sim1); % narx output
title('NARX output and PD output');
```

```
xlabel('Time, s');
ylabel('Control action');
legend('NARX','PD');

figure('WindowState','maximized','Color',[1 1 1]);
plot(t1,mse);
title('Squares of errors');
xlabel('Time, s');
ylabel('SE');
ylim([-0.5 2.5]);
```

Appendix 2 DDPG Algorithm [39]

```
Randomly initialize critic network $Q(s,a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights $\theta^Q$ and $\theta^\mu$
Initialize target network $Q'$ and $\mu'$ with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer $R$
for episode = 1, M do
    Initialize a random process $\mathcal{N}$ for action exploration
    Receive initial observation state $s_1$
    for t = 1, T do
        Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
        Execute action $a_t$ and observe reward $r_t$ and observe new state $s_{t+1}$
        Store transition $(s_t, a_t, r_t, s_{t+1})$ in $R$
        Sample a random minibatch of $N$ transitions $(s_i, a_i, r_i, s_{i+1})$ from $R$
        Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
        Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i \left(y_i - Q(s_i, a_i|\theta^Q)\right)^2$
        Update the actor policy using the sampled policy gradient:
            $\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s,a|\theta^Q)|_{s=s_i,a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i}$
        Update the target networks:
            $\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$
            $\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$
    end for
end for
```

Appendix 3 MATLAB code for building the environment

```matlab
% Create observations

obsInfo = rlNumericSpec([3 1],...
    'LowerLimit',[0 -inf -inf]',...
    'UpperLimit',[inf inf inf]');

obsInfo.Name = 'observations';
obsInfo.Description = 'position, error, integral error';
numObservations = obsInfo.Dimension(1);

% Create actions

actInfo = rlNumericSpec([1 1]);

actInfo.Name = 'control';
numActions = actInfo.Dimension(1);

% Build the environment interface object

env =
rlSimulinkEnv('control_1_NARX_vs_PID_RL_sim_1em','control_1_NARX_vs_PID_RL_sim_
1em/DDPG Agent',...
    obsInfo,actInfo);
```

Appendix 4 MATLAB code for creating actor and critic networks

```
% Actor NN design

lgraph_actor = layerGraph();
layers = [
    featureInputLayer(3,"Name","input_1")
    fullyConnectedLayer(25,"Name","fc_1")
    tanhLayer("Name","tanh_1")
    fullyConnectedLayer(25,"Name","fc_2")
    tanhLayer("Name","tanh_2")
    fullyConnectedLayer(1,"Name","output")
    ];
lgraph_actor = addLayers(lgraph_actor,layers);

% Critic NN design

lgraph_critic = layerGraph();

% state path

tempLayers = [
    featureInputLayer(3,"Name","input_1")
    fullyConnectedLayer(50,"Name","st_fc_1")
    reluLayer("Name","relu_1")
    fullyConnectedLayer(25,"Name","st_fc_2")
    ];
lgraph_critic = addLayers(lgraph_critic,tempLayers);

% action path

tempLayers = [
    featureInputLayer(1,"Name","input_2")
    fullyConnectedLayer(25,"Name","act_fc_1")
    ];
lgraph_critic = addLayers(lgraph_critic,tempLayers);

% add

tempLayers = [
    additionLayer(2,"Name","concat")
    reluLayer("Name","relu_output")
    fullyConnectedLayer(1,"Name","output_1")
    ];
lgraph_critic = addLayers(lgraph_critic,tempLayers);

% clean up helper variable

clear tempLayers;

% state path
lgraph_critic = connectLayers(lgraph_critic,"st_fc_2","concat/in1");

% action path
lgraph_critic = connectLayers(lgraph_critic,"act_fc_1","concat/in2");
```

```
% Plot graphs

figure;
plot(lgraph_actor);

figure;
plot(lgraph_critic);
```

Appendix 5 MATLAB code for setting RL-training options

```
% Sample time and Simulation time

Ts = 0.001;
Tf = 10;

% Specify options for actor and critic representation

actorOpts = rlRepresentationOptions('LearnRate',1e-03,'GradientThreshold',1);

criticOpts = rlRepresentationOptions('LearnRate',1e-03,'GradientThreshold',1);

% Specify the DDPG agent options

agentOpts = rlDDPGAgentOptions(...
    'SampleTime',Ts,...
    'TargetSmoothFactor',1e-3,...
    'DiscountFactor',0.99, ...
    'MiniBatchSize',128, ...
    'ExperienceBufferLength',1e6);

agentOpts.NoiseOptions.Variance = 0.02;


% Create the DDPG agent

agent = rlDDPGAgent(actor,critic,agentOpts);

% Training options

maxepisodes = 10000;
maxsteps = ceil(Tf/Ts);

trainOpts = rlTrainingOptions(...
    'MaxEpisodes',maxepisodes, ...
    'MaxStepsPerEpisode',maxsteps, ...
    'ScoreAveragingWindowLength',5, ...
    'StopTrainingCriteria','EpisodeReward', ...
    'StopTrainingValue',20000, ...
    'SaveAgentCriteria', 'EpisodeReward', ...
    'SaveAgentValue', 20000, ...
    'Verbose',false, ...
    'Plots','training-progress' ...
    );

% Train the DDPG agent

trainingStats = train(agent,env,trainOpts);

% Validate the trained DDPG agent

simOpts = rlSimulationOptions('MaxSteps',maxsteps,'StopOnError','on');
experiences = sim(env,agent,simOpts);
```

Appendix 6 Training progress for the case of graded learning

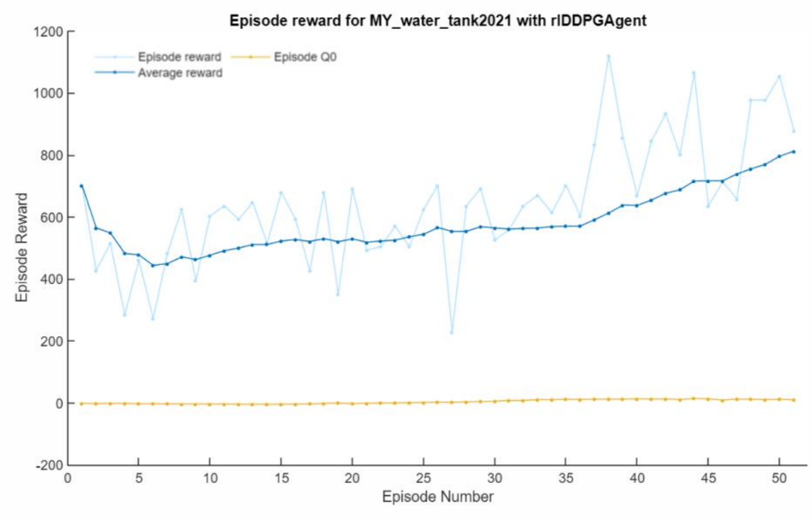


Figure A.1 Training progress of "wt_agent1_100"

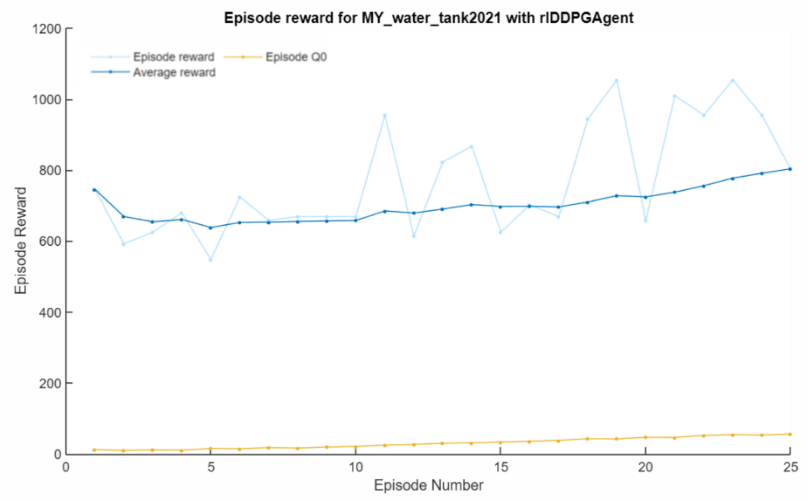


Figure A.2 Training progress of "wt_agent1_075"

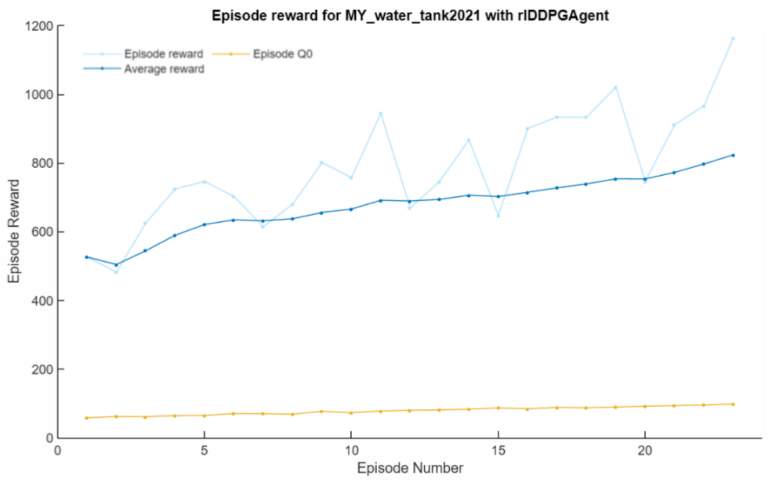


Figure A.3 Training progress of "wt_agent1_050"

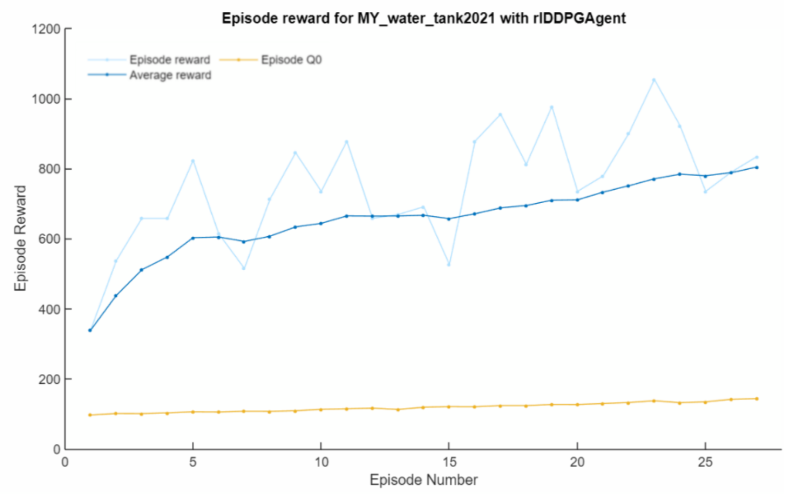


Figure A.4 Training progress of "wt_agent1_025"

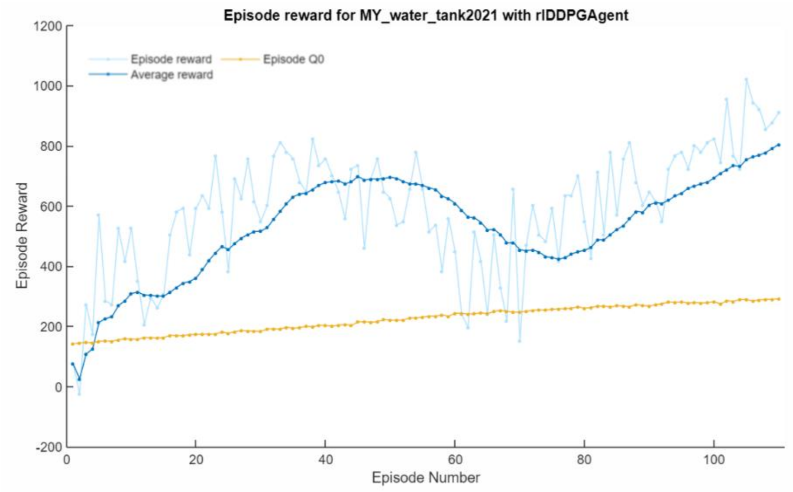


Figure A.5 Training progress of "wt_agent1_000"

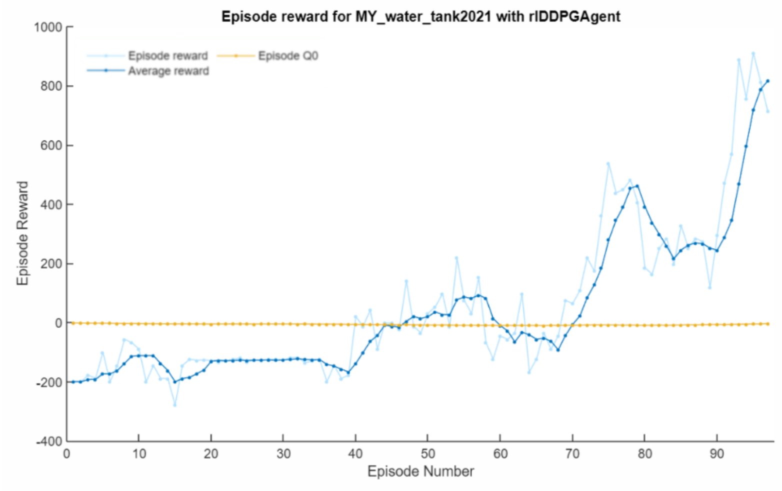
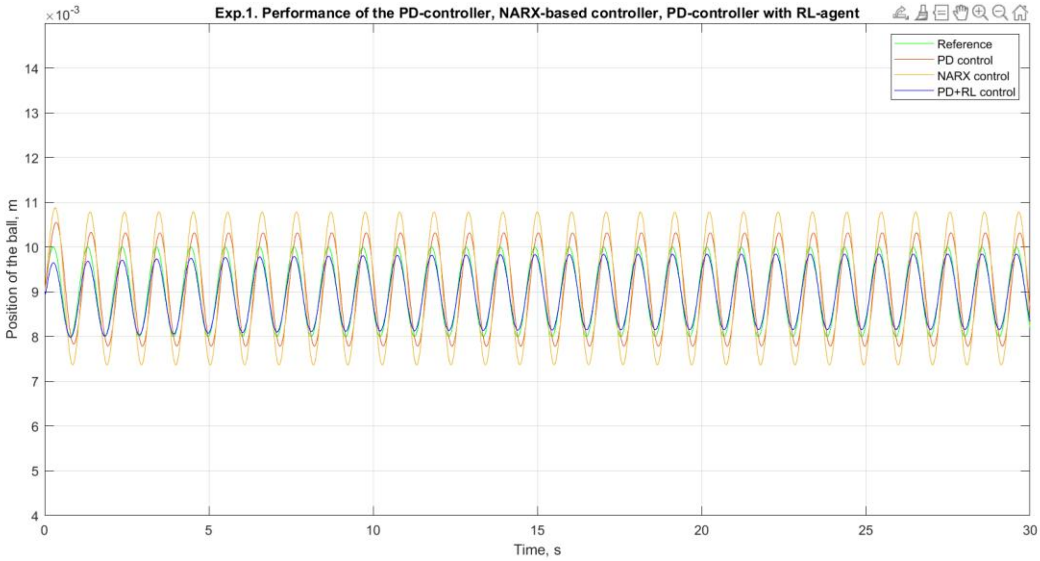


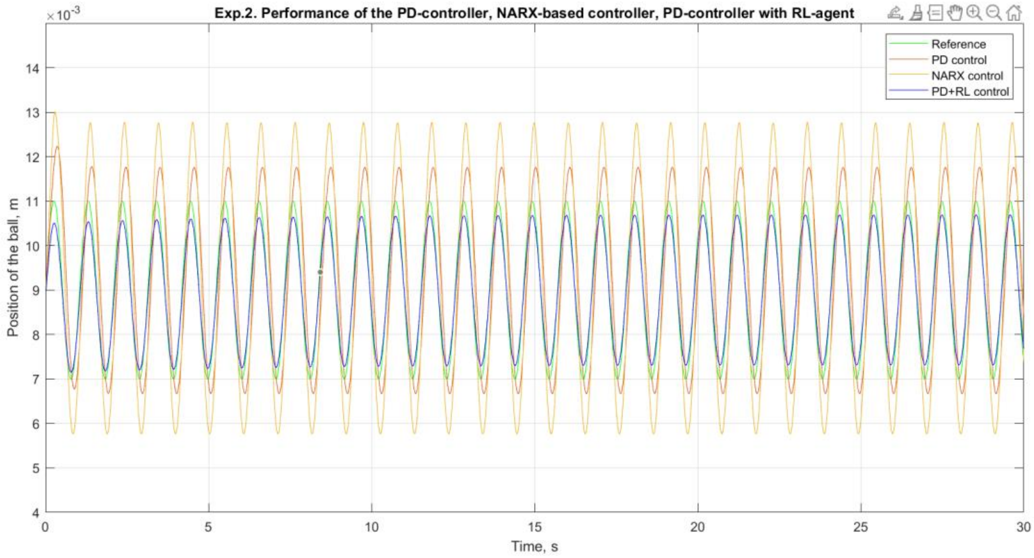
Figure A.6 Training progress of "wt_agent2"

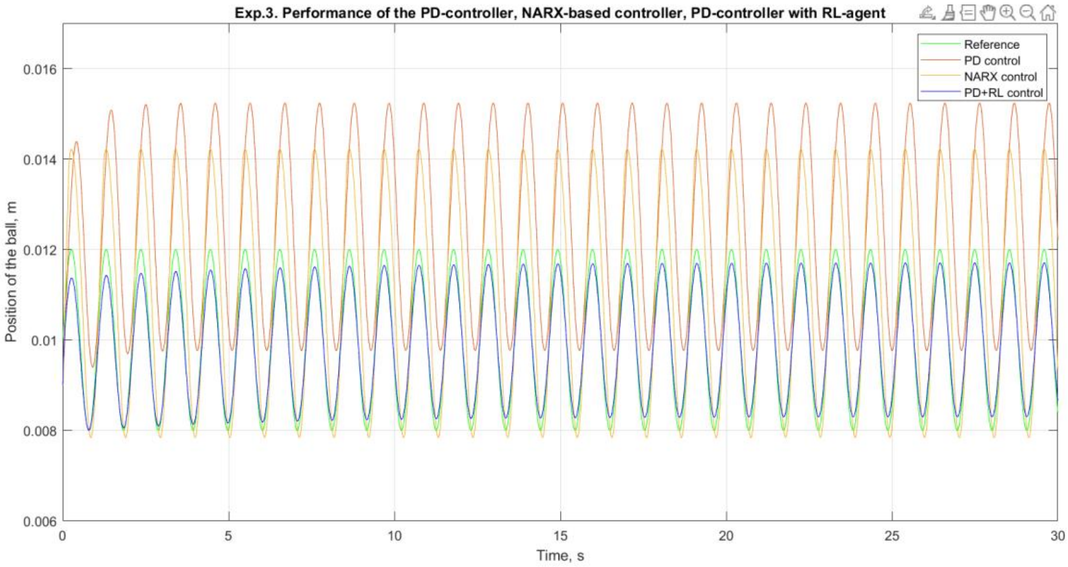
Appendix 7 MATLAB code for calculating MSE and building plots

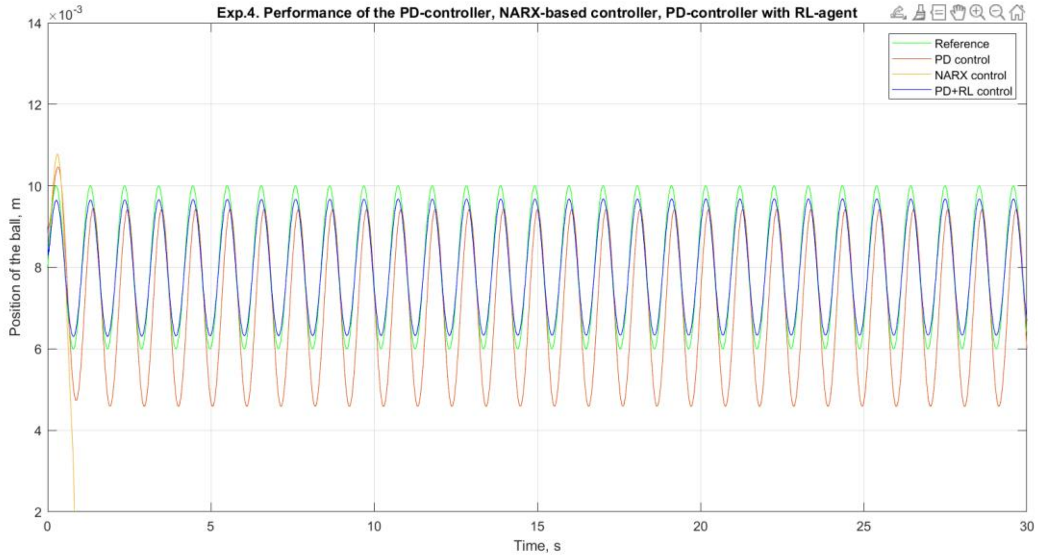
```
% PD

ERR=0;
SSE=0;

ERR = out.refer - out.pid;                           % ERR
SSE = sum(ERR.^2);                        % Sum-Squared Error
MSE_pid = mean(ERR.^2)

% NARX

ERR=0;
SSE=0;

ERR = out.refer - out.narx;                           % ERR
SSE = sum(ERR.^2);                        % Sum-Squared Error
MSE_narx = mean(ERR.^2)

% PD with RL-agent

ERR=0;
SSE=0;

ERR = out.refer - out.pid_rl;                           % ERR
SSE = sum(ERR.^2);                        % Sum-Squared Error
MSE_pid_rl = mean(ERR.^2)


% Building plots

fig1 = figure('WindowState','maximized','Color',[1 1 1]);
plot(out.time',out.refer','green');
hold on;
plot(out.time,out.pid)
title('Exp.1. Performance of the PD-controller, NARX-based controller, PD-
controller with RL-agent');
plot(out.time,out.narx);
plot(out.time,out.pid_rl,'blue');
xlabel('Time, s');
ylabel('Position of the ball, m');
ylim([0.004 0.015]);
legend('Reference', 'PD control', 'NARX control', 'PD+RL control');
grid on;

saveas(fig1,'fig1','fig');
saveas(fig1,'fig1','png');
```

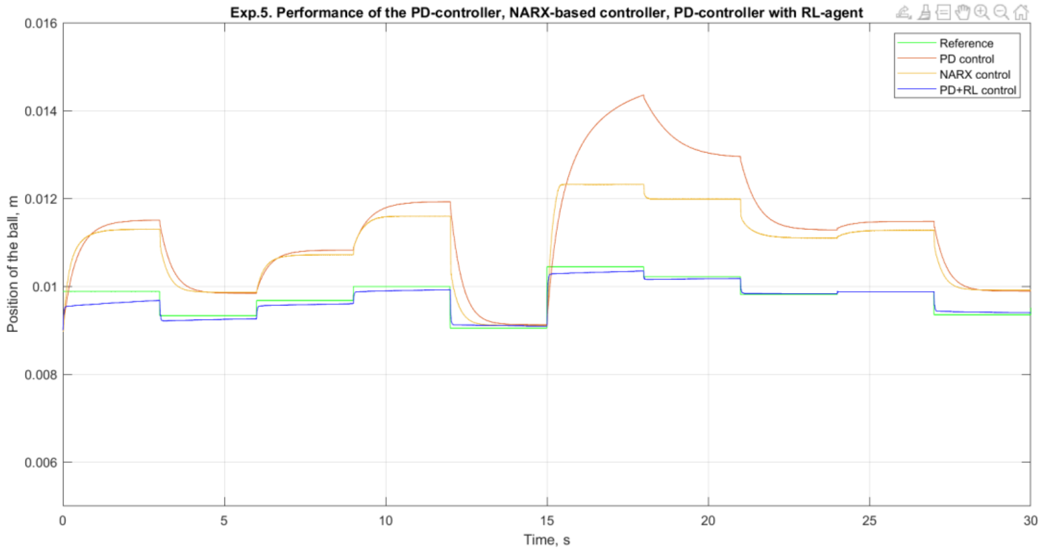
Appendix 8 Plots in experiments for controller 1

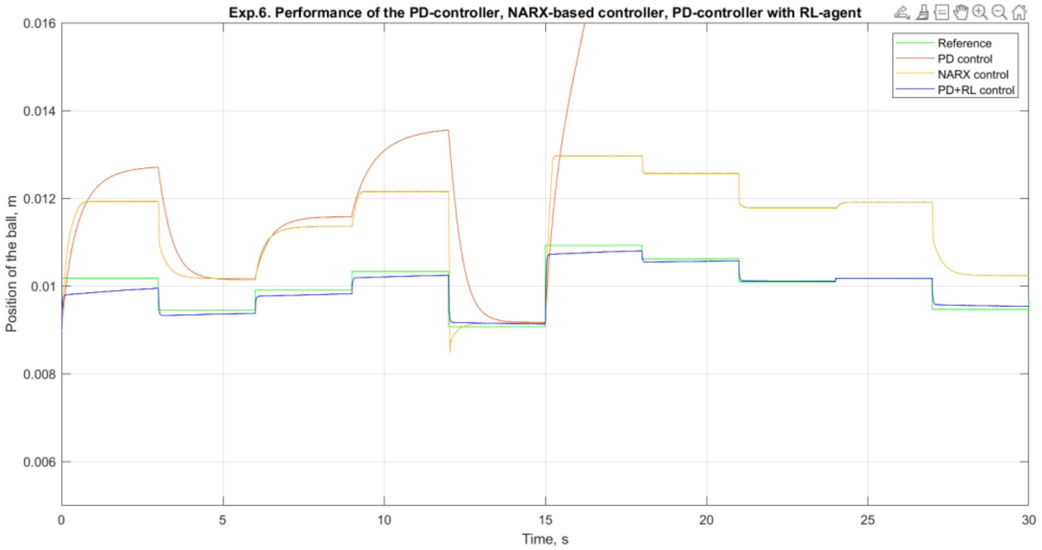


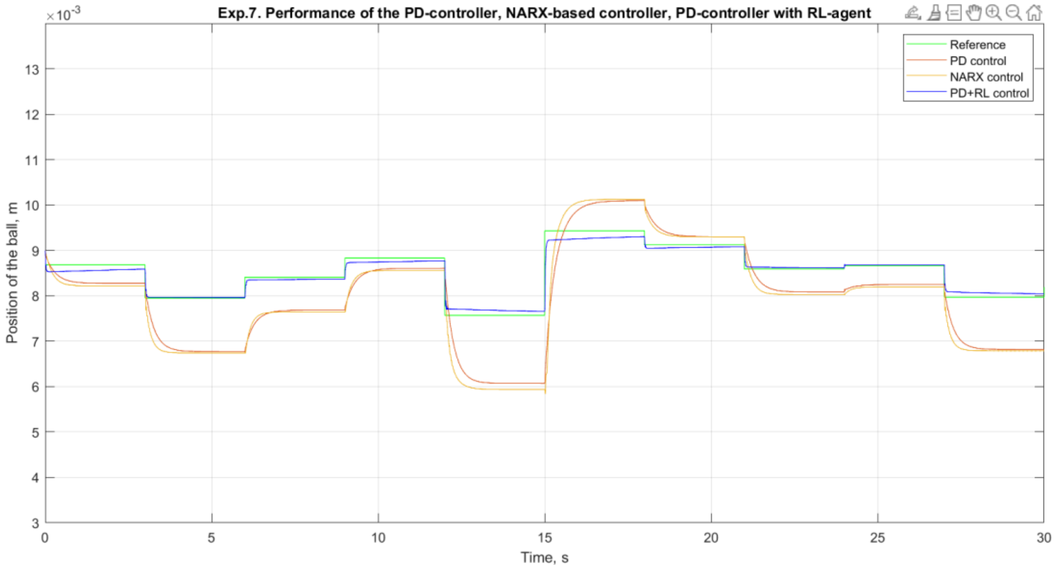


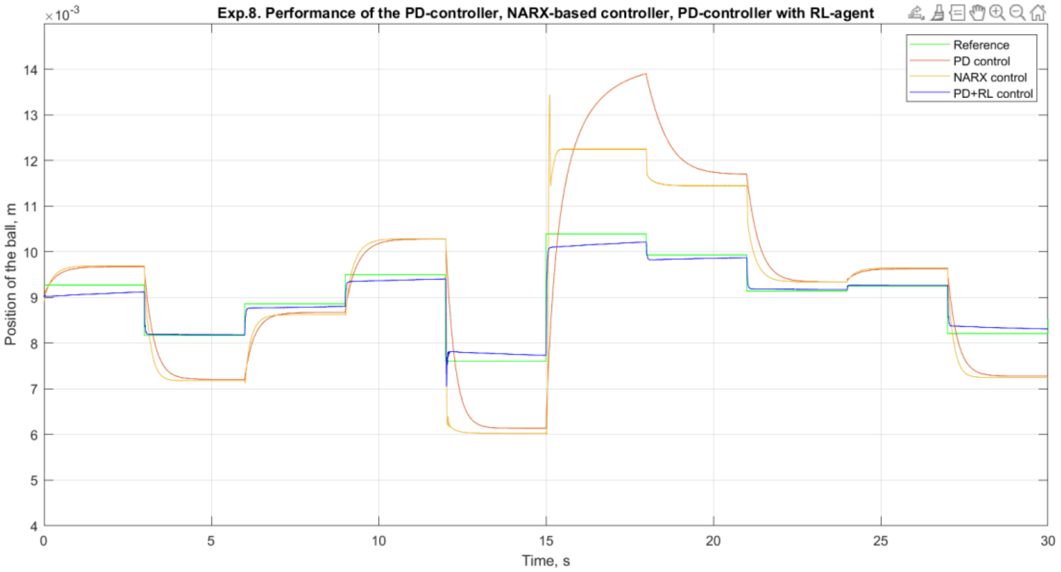


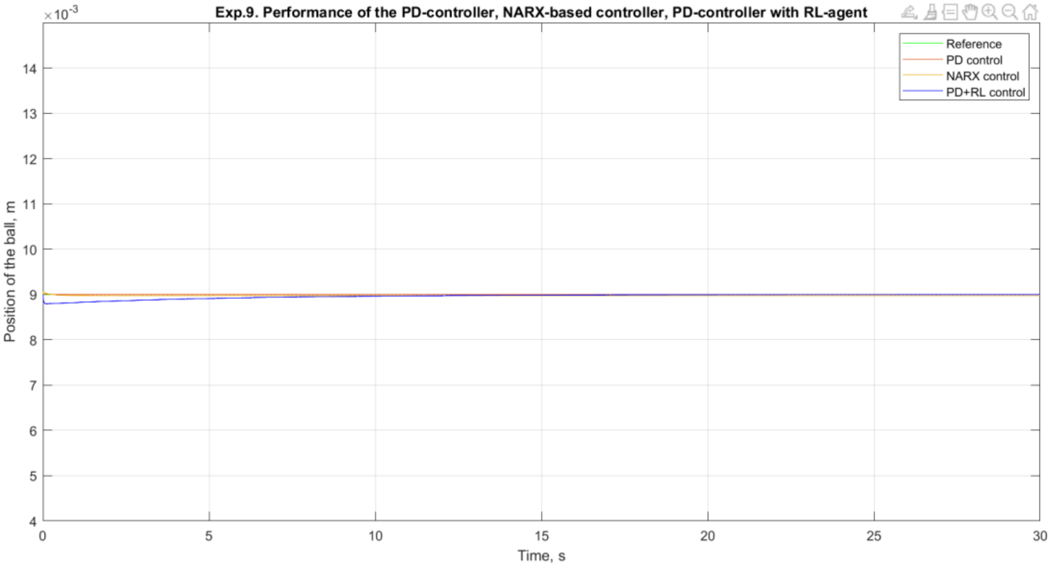

Exp.4. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent
$\times 10^{-3}$
14
12
10
8
6
4
2
Position of the ball, m
0
5
10
15
20
25
30
Time, s
Reference
PD control
NARX control
PD+RL control

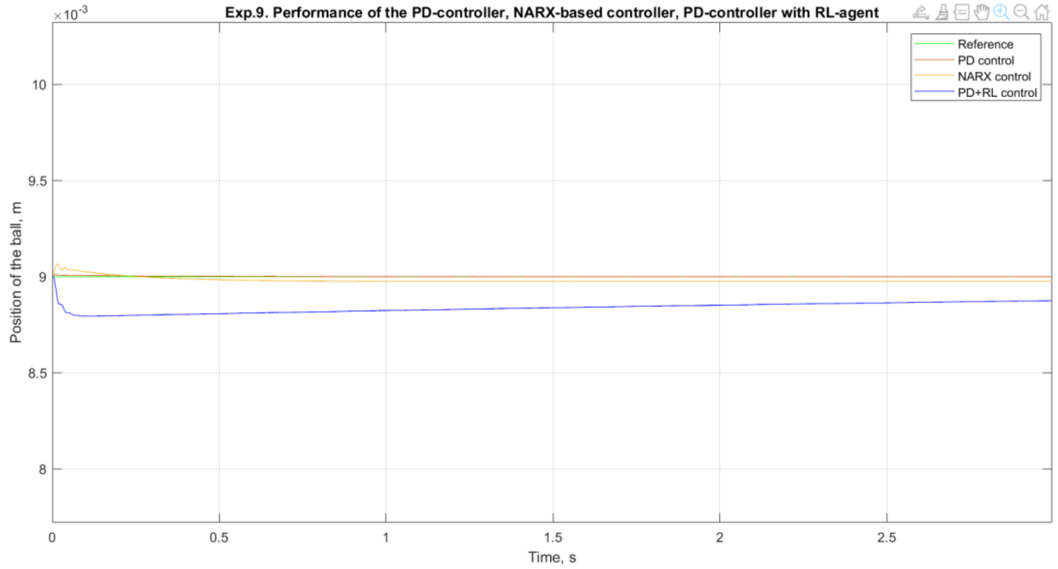

Exp.5. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent
0.016
0.014
0.012
0.01
0.008
0.006
Position of the ball, m
0
5
10
15
20
25
30
Time, s
Reference
PD control
NARX control
PD+RL control


Exp.6. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent
0.016
0.014
0.012
0.01
0.008
0.006
Position of the ball, m
0
5
10
15
20
25
30
Time, s
Reference
PD control
NARX control
PD+RL control

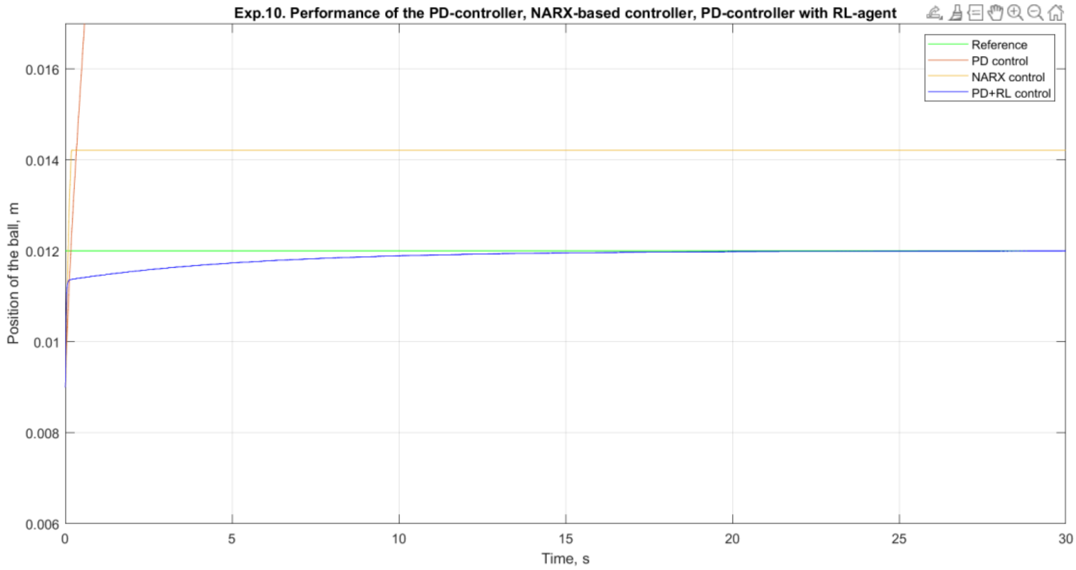

Exp.7. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent
×10^-3
Position of the ball, m
Time, s
Reference
PD control
NARX control
PD+RL control
3
4
5
6
7
8
9
10
11
12
13
0
5
10
15
20
25
30


Exp.8. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent
×10^-3
Position of the ball, m
Time, s
Reference
PD control
NARX control
PD+RL control
4
5
6
7
8
9
10
11
12
13
14
0
5
10
15
20
25
30

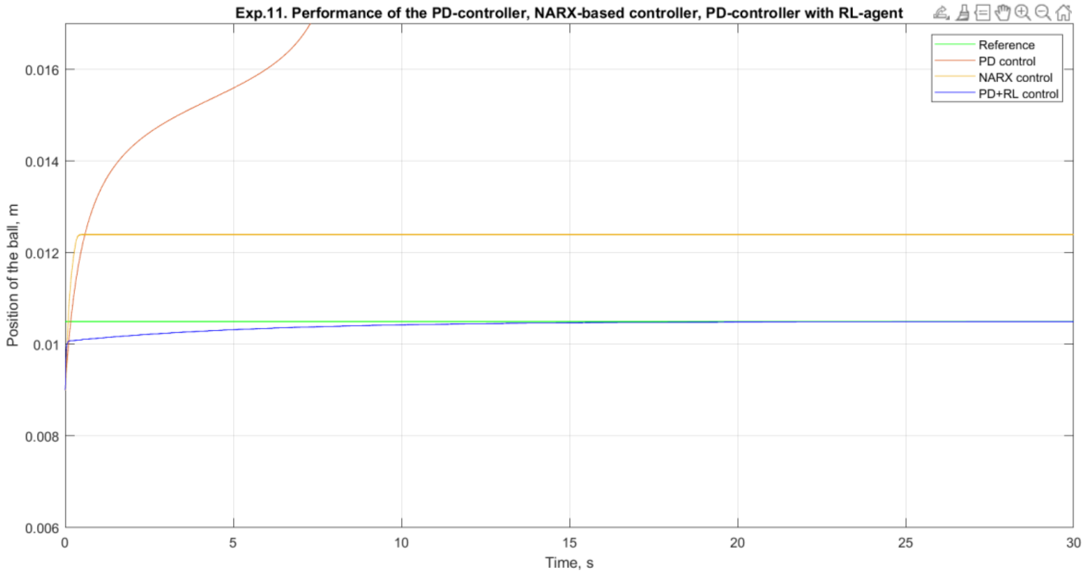

Exp.9. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent
×10^-3
Position of the ball, m
Time, s
Reference
PD control
NARX control
PD+RL control
4
5
6
7
8
9
10
11
12
13
14
0
5
10
15
20
25
30



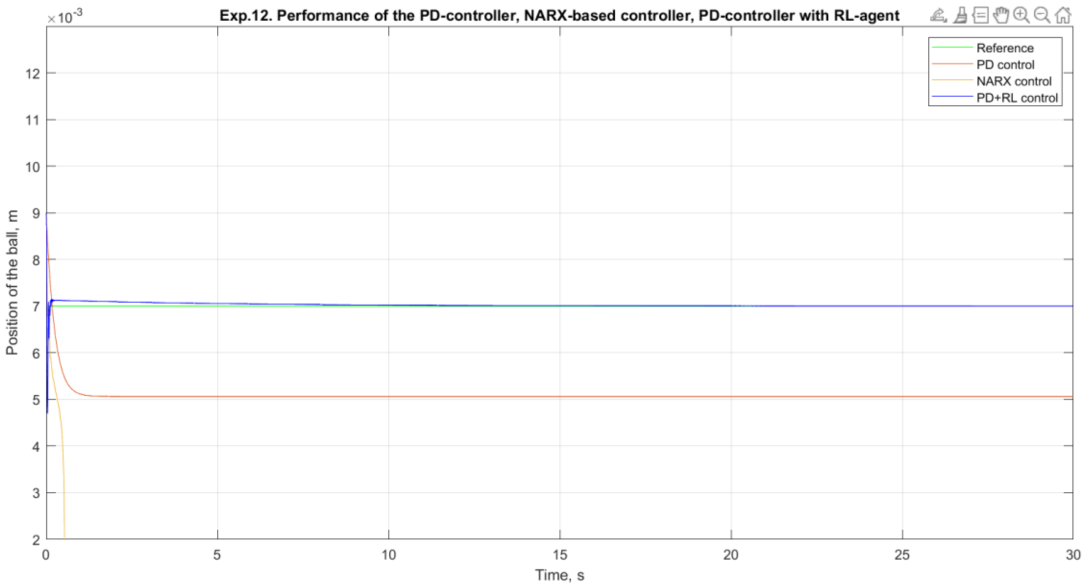
Exp.9. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent

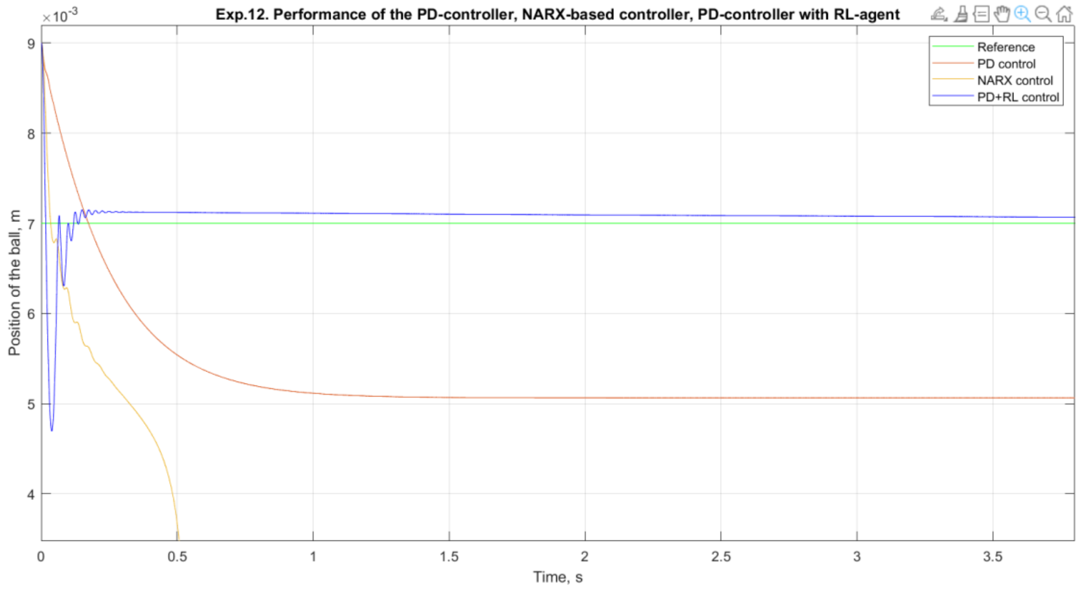


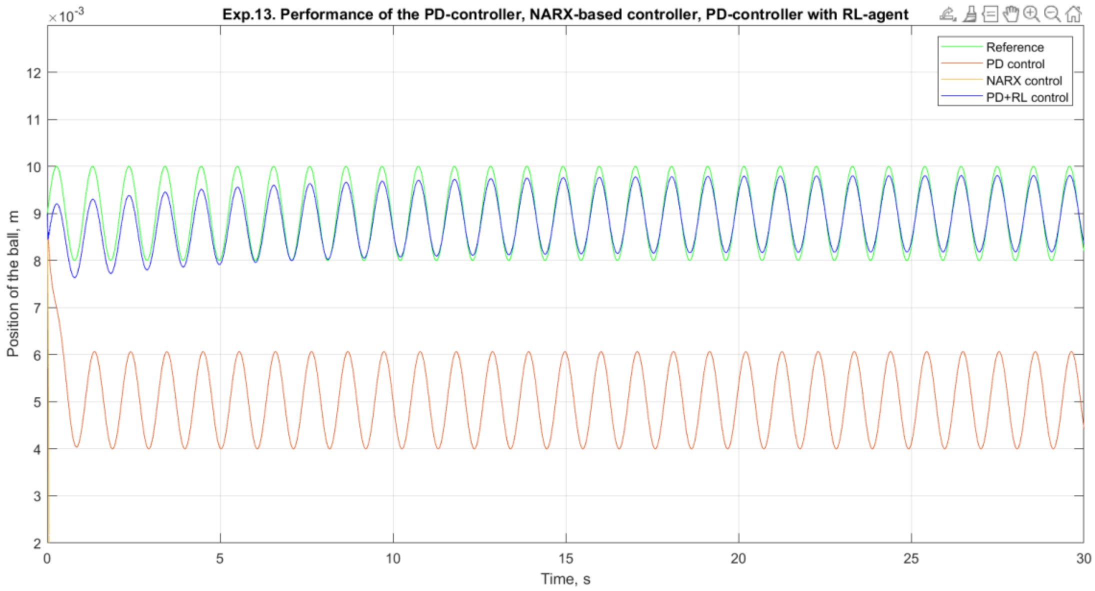
Exp.10. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent

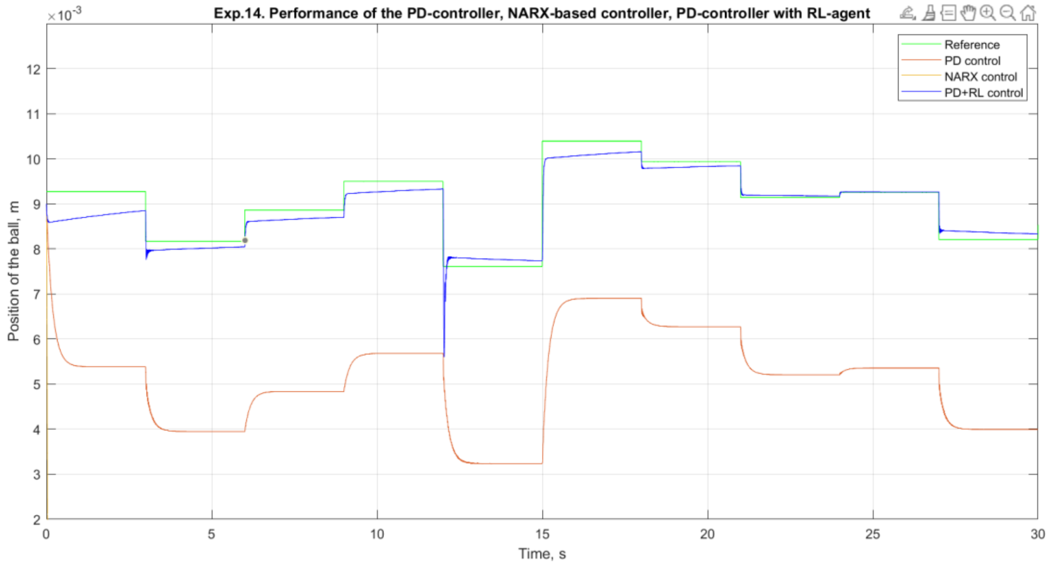


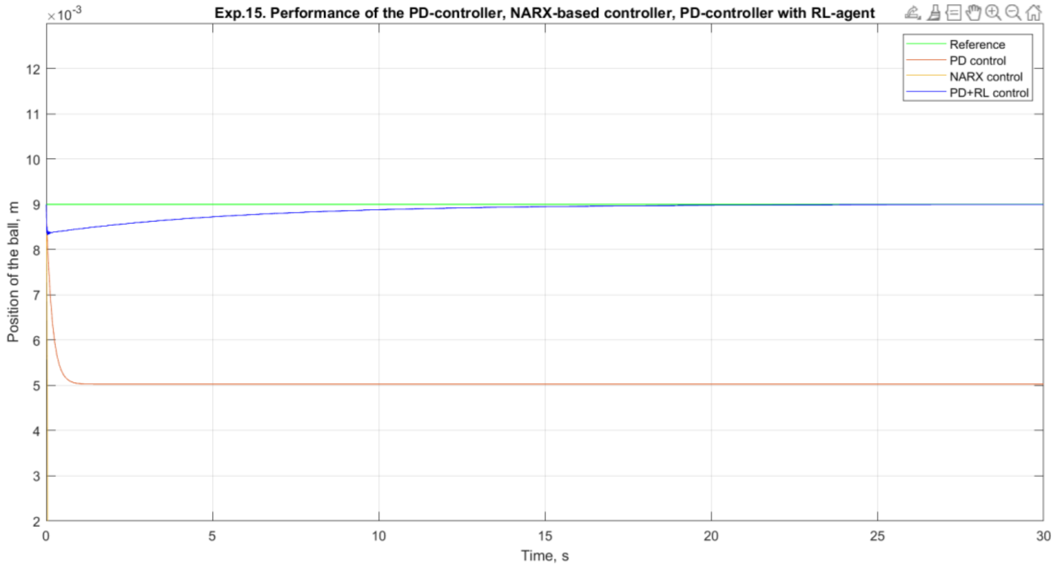
Exp.11. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent

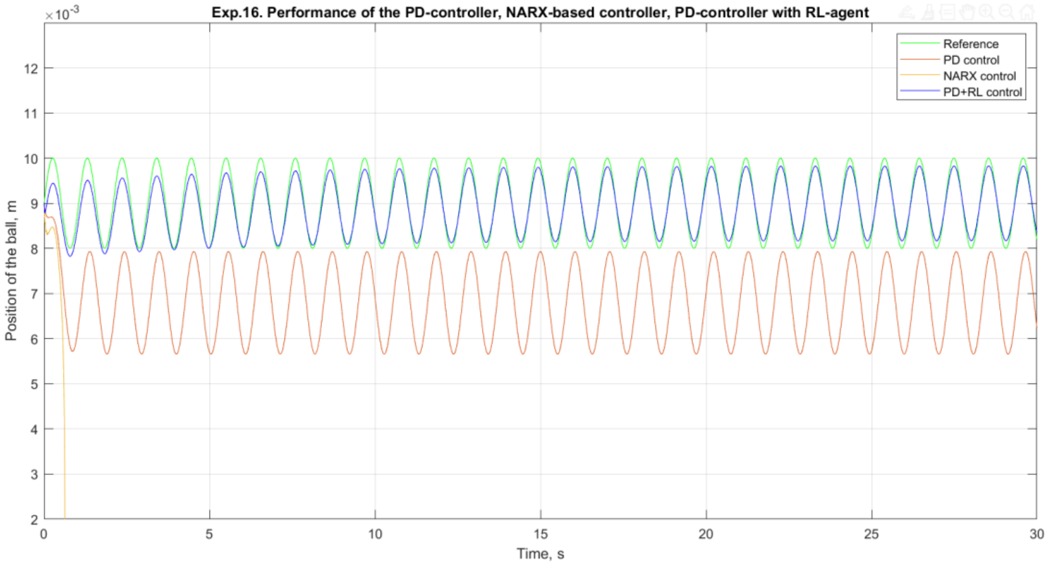

Exp.12. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent
×10^-3
Reference
PD control
NARX control
PD+RL control
Position of the ball, m
12
11
10
9
8
7
6
5
4
3
2
0
5
10
15
20
25
30
Time, s

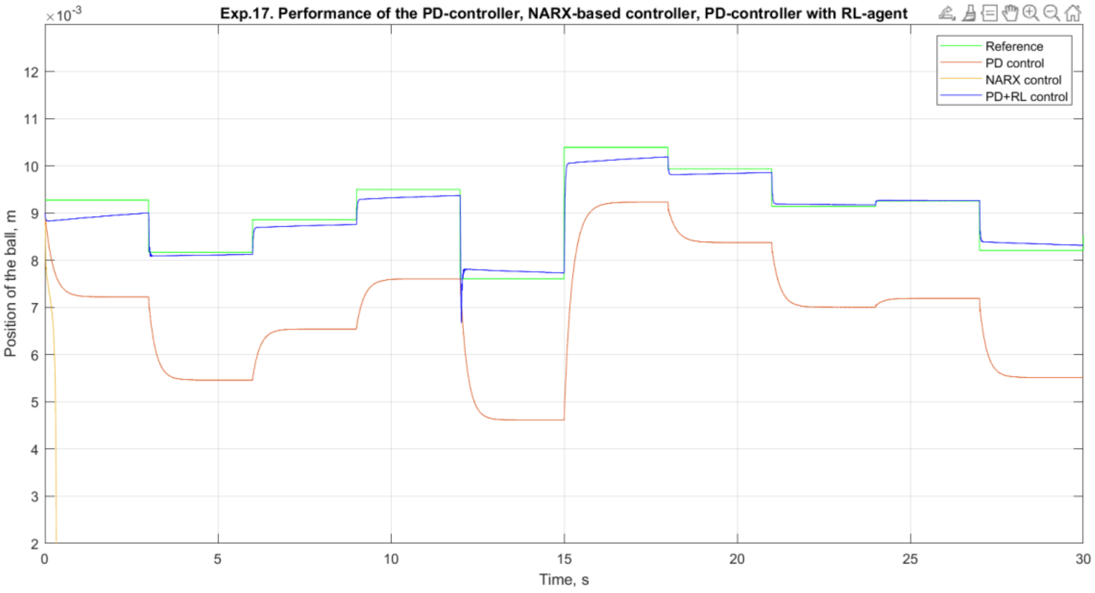

Exp.12. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent
×10^-3
Reference
PD control
NARX control
PD+RL control
Position of the ball, m
9
8
7
6
5
4
0
0.5
1
1.5
2
2.5
3
3.5
Time, s


Exp.13. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent
×10^-3
Reference
PD control
NARX control
PD+RL control
Position of the ball, m
12
11
10
9
8
7
6
5
4
3
2
0
5
10
15
20
25
30
Time, s

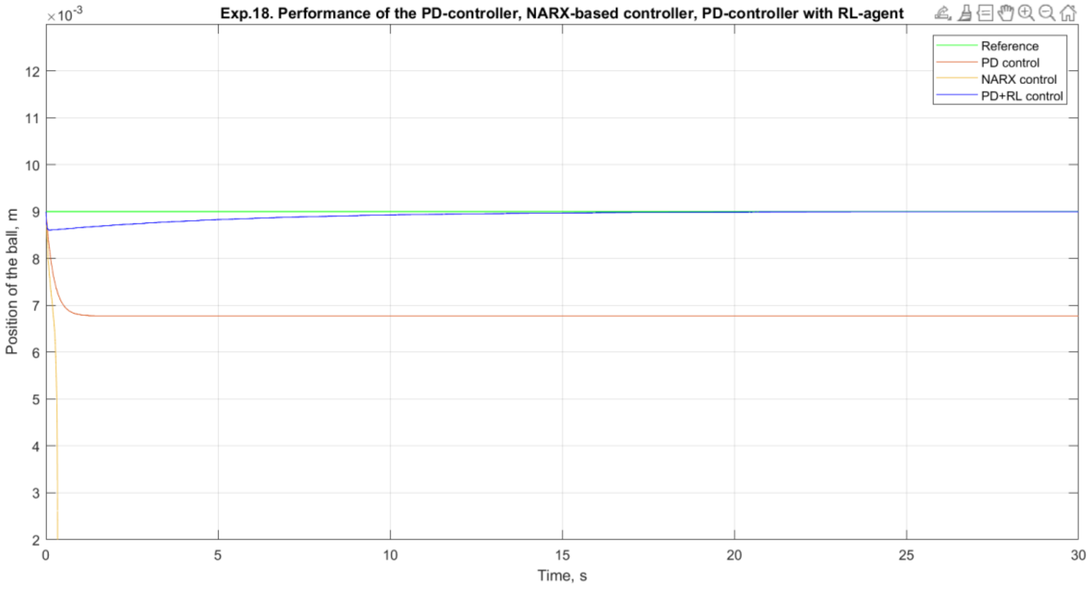

Exp.14. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent
×10^-3
Position of the ball, m
12
11
10
9
8
7
6
5
4
3
2
0
5
10
15
20
25
30
Time, s
Reference
PD control
NARX control
PD+RL control


Exp.15. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent
×10^-3
Position of the ball, m
12
11
10
9
8
7
6
5
4
3
2
0
5
10
15
20
25
30
Time, s
Reference
PD control
NARX control
PD+RL control

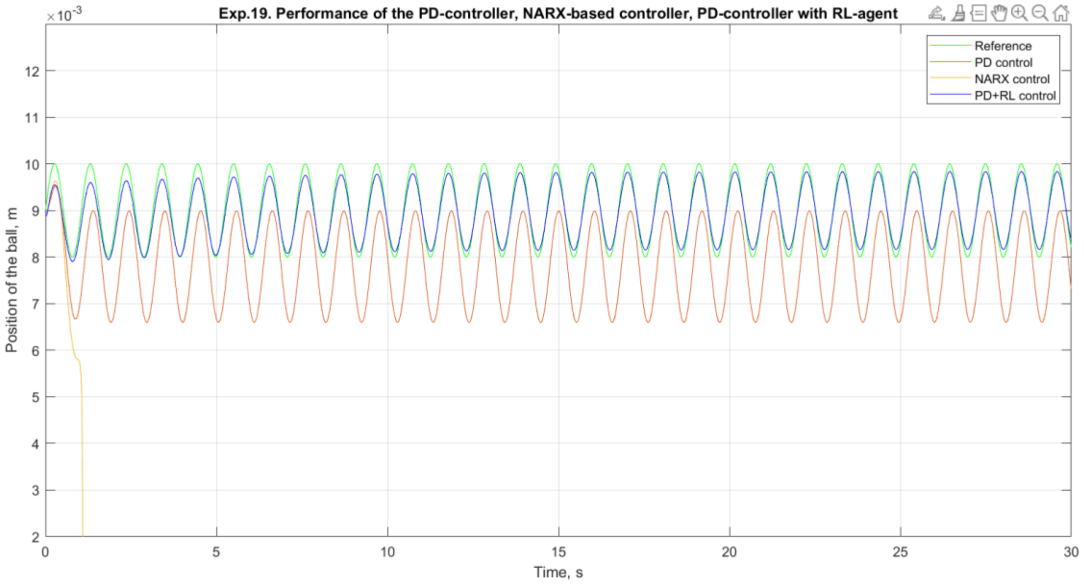

Exp.16. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent
×10^-3
Position of the ball, m
12
11
10
9
8
7
6
5
4
3
2
0
5
10
15
20
25
30
Time, s
Reference
PD control
NARX control
PD+RL control



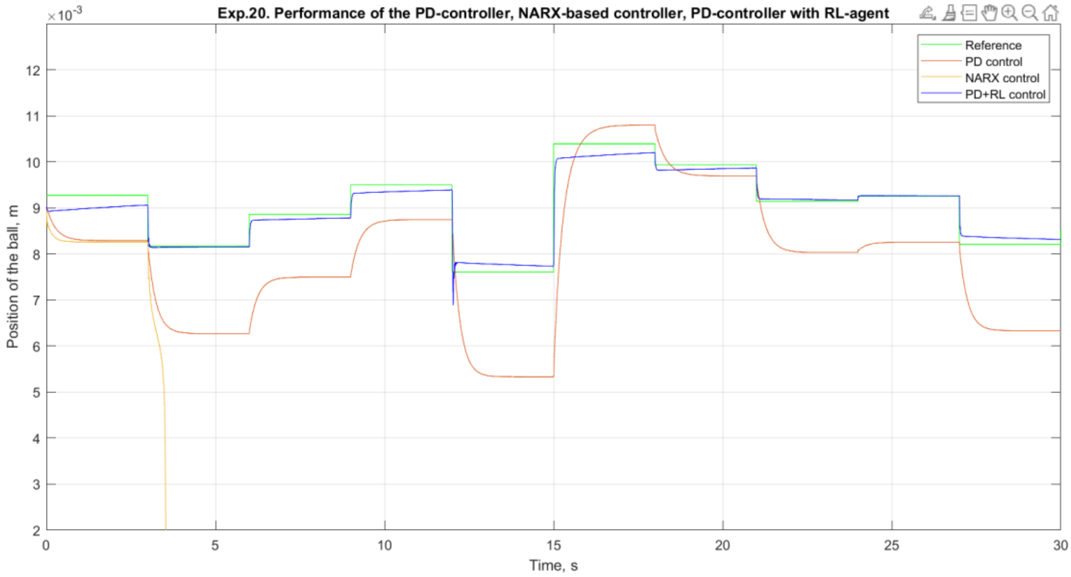
Exp.17. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent



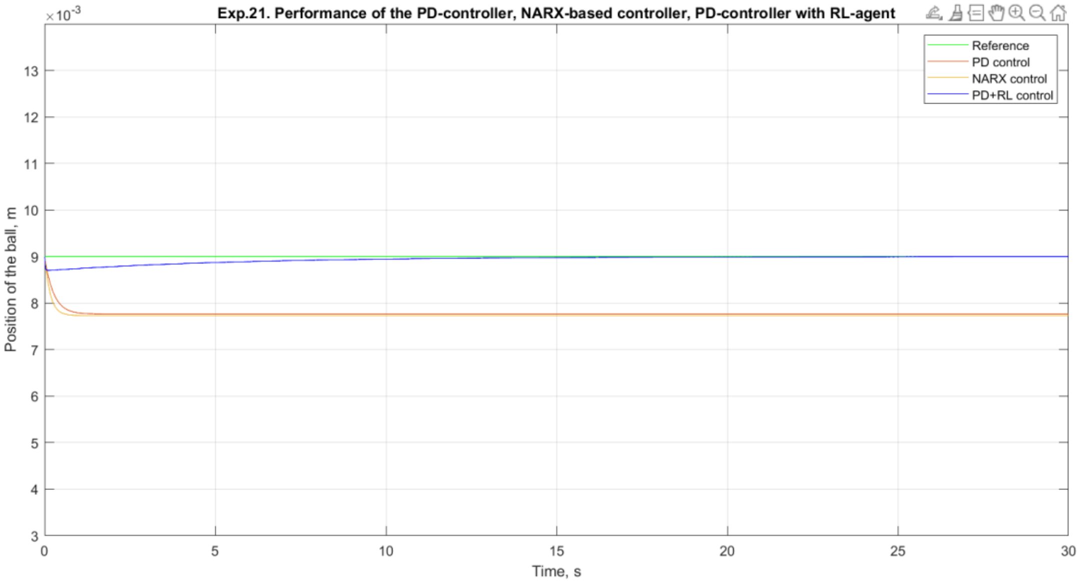
Exp.18. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent



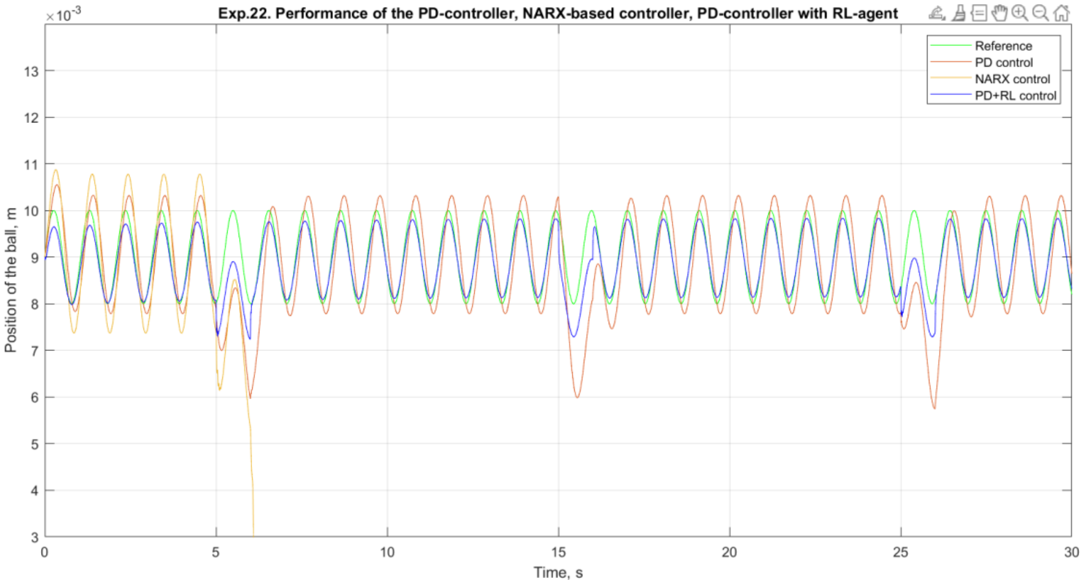
Exp.19. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent



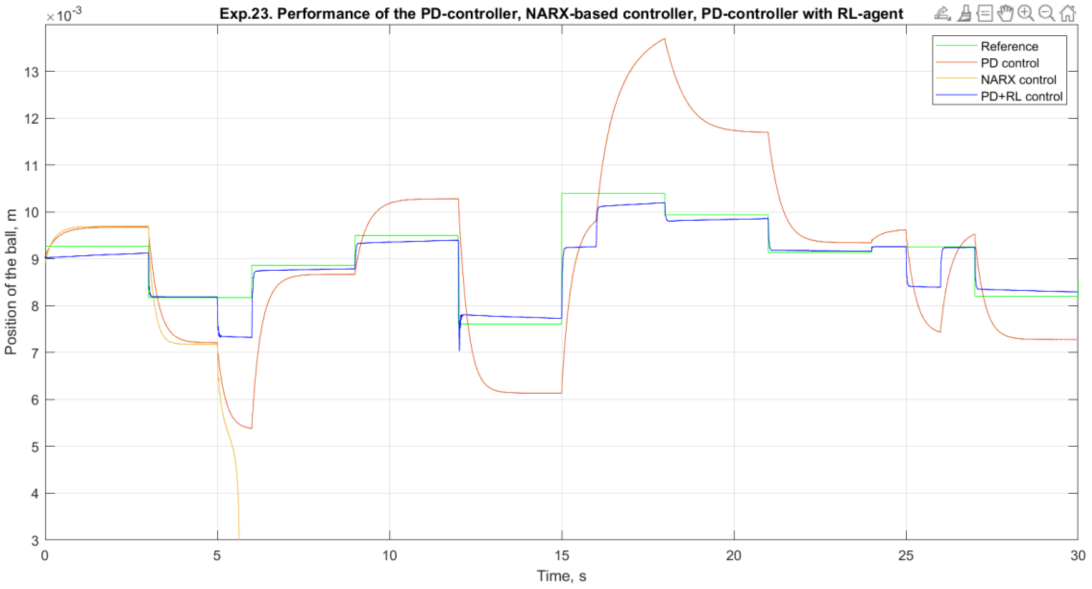
Exp.20. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent

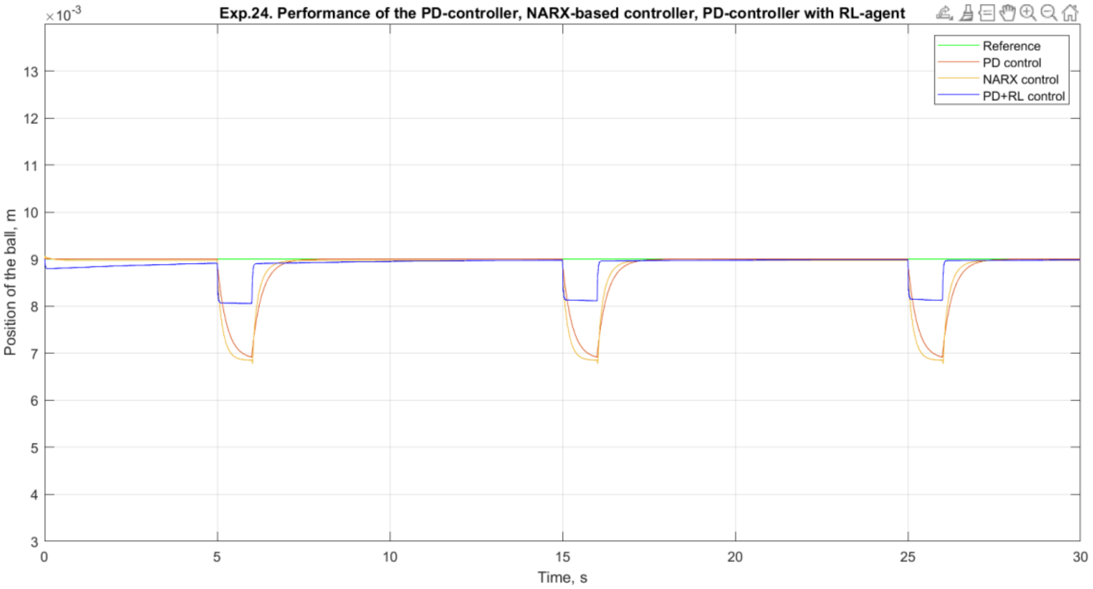


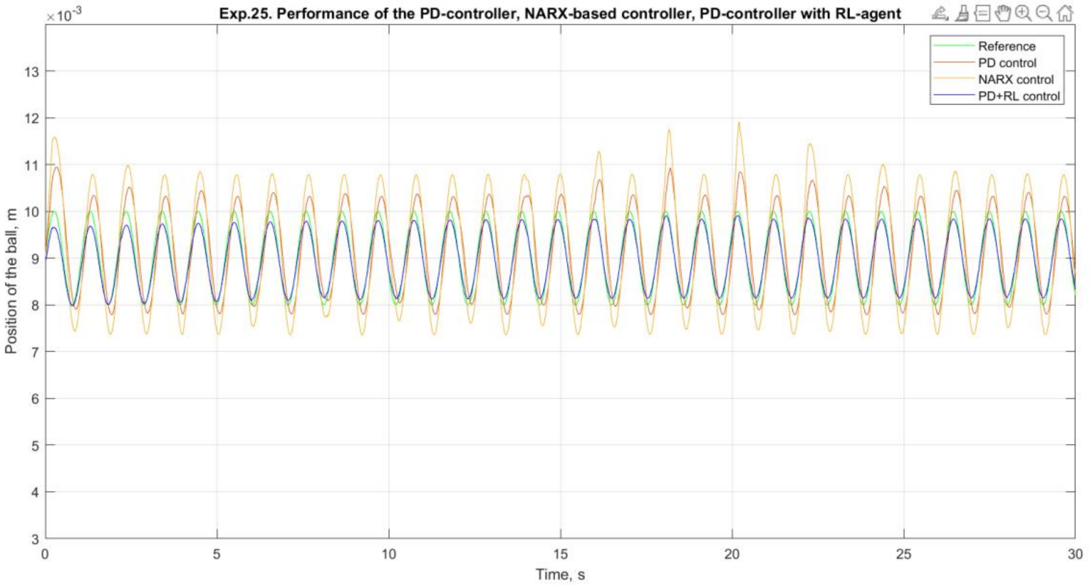
Exp.21. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent

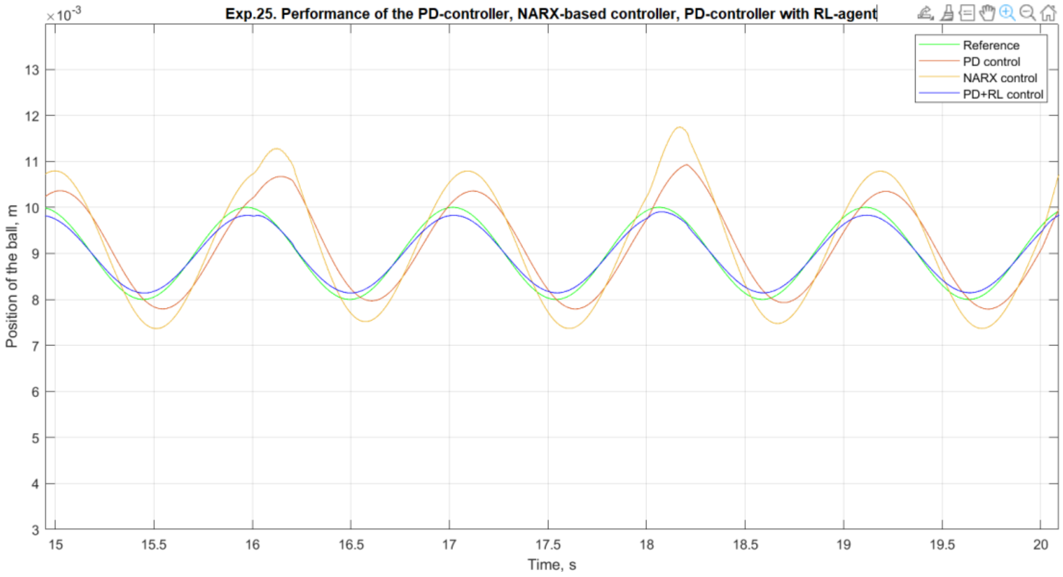


Exp.22. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent

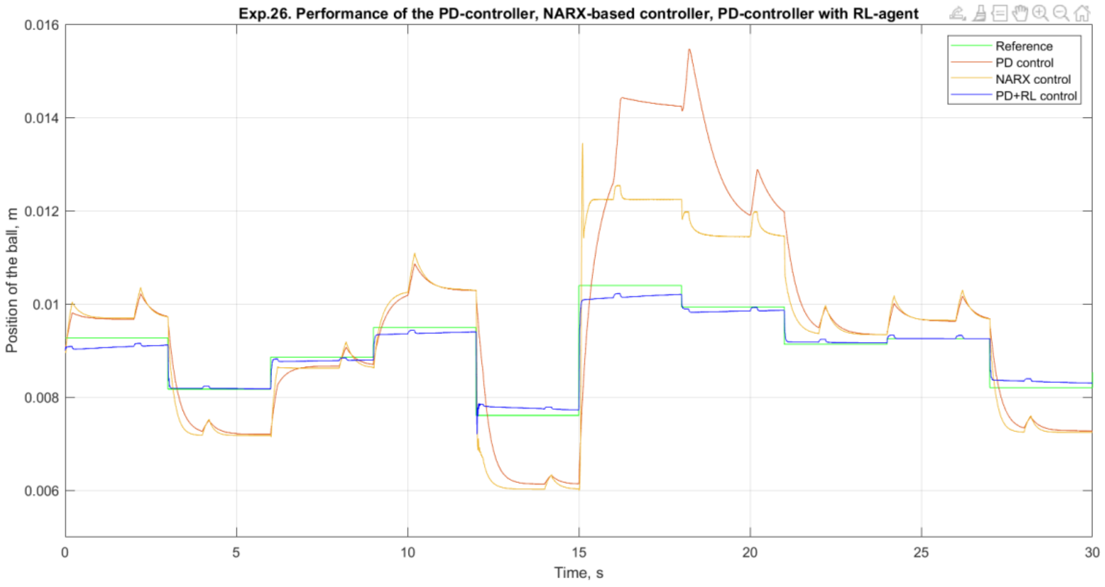

Exp.23. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent


Exp.24. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent

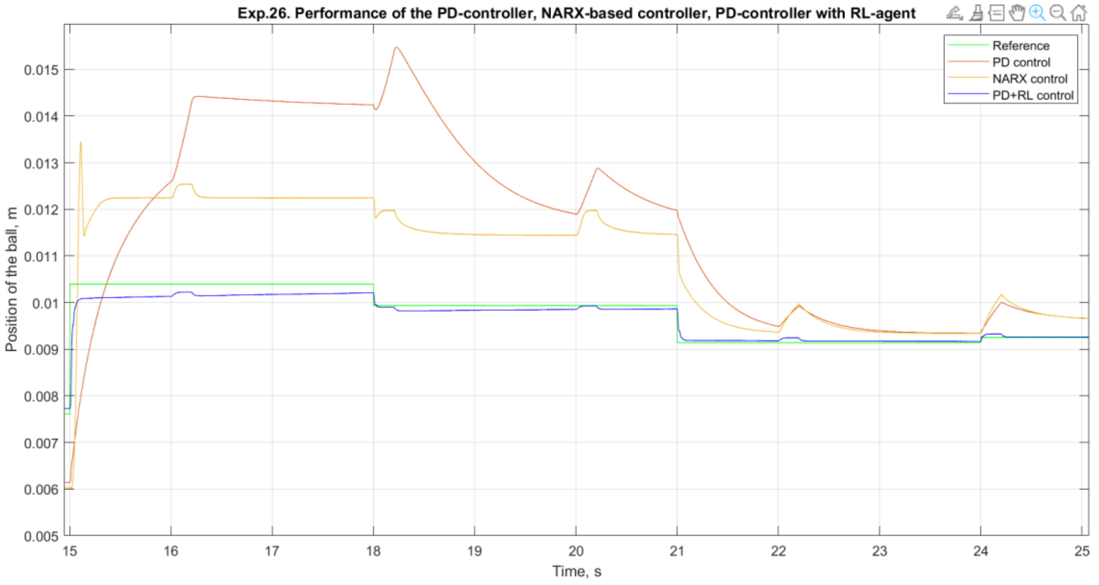

Exp.25. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent



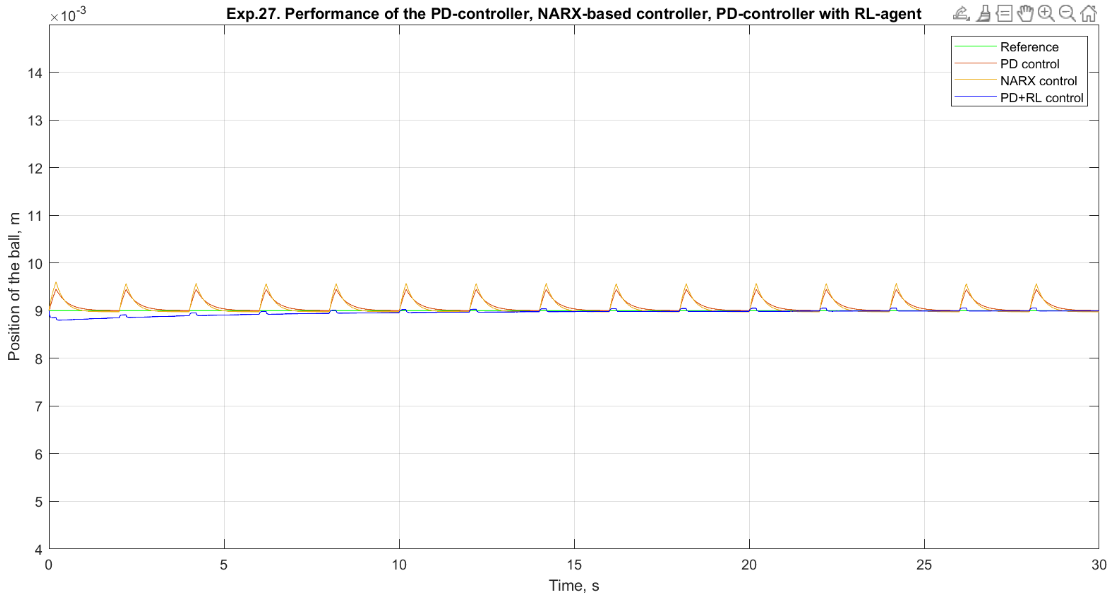
Exp.25. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent

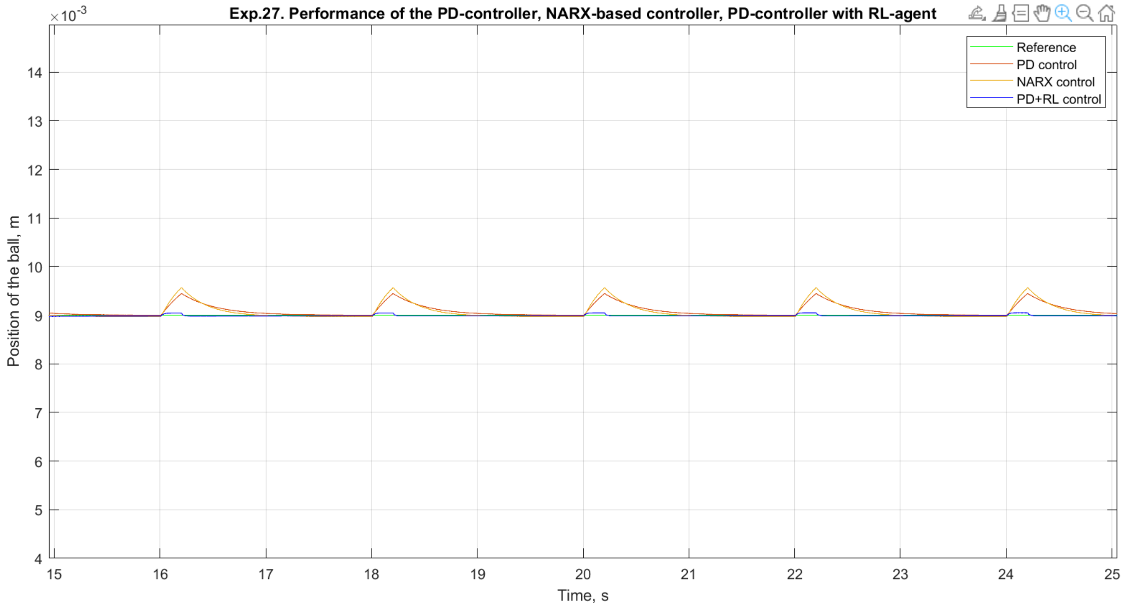


Exp.26. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent

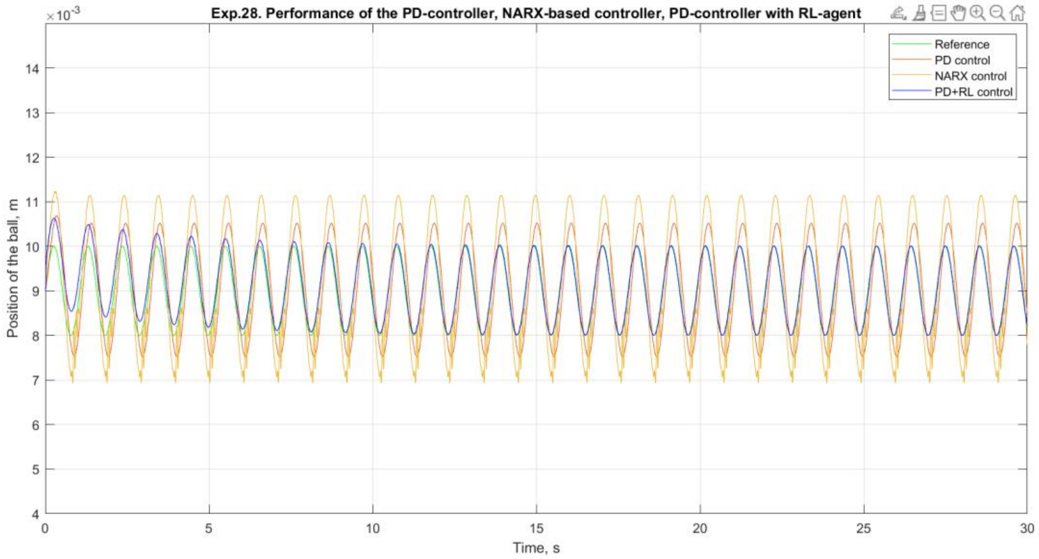


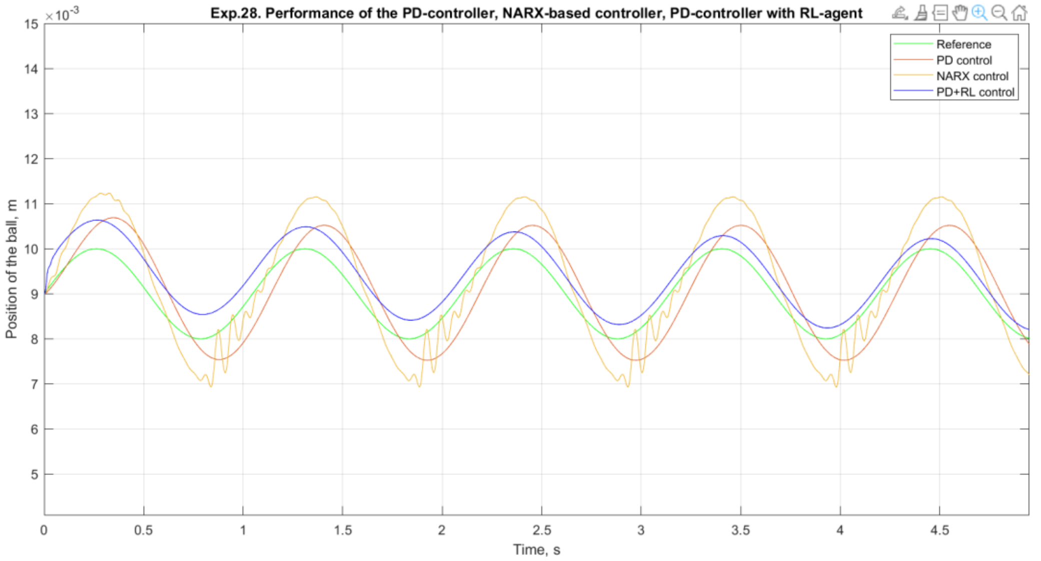
Exp.26. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent

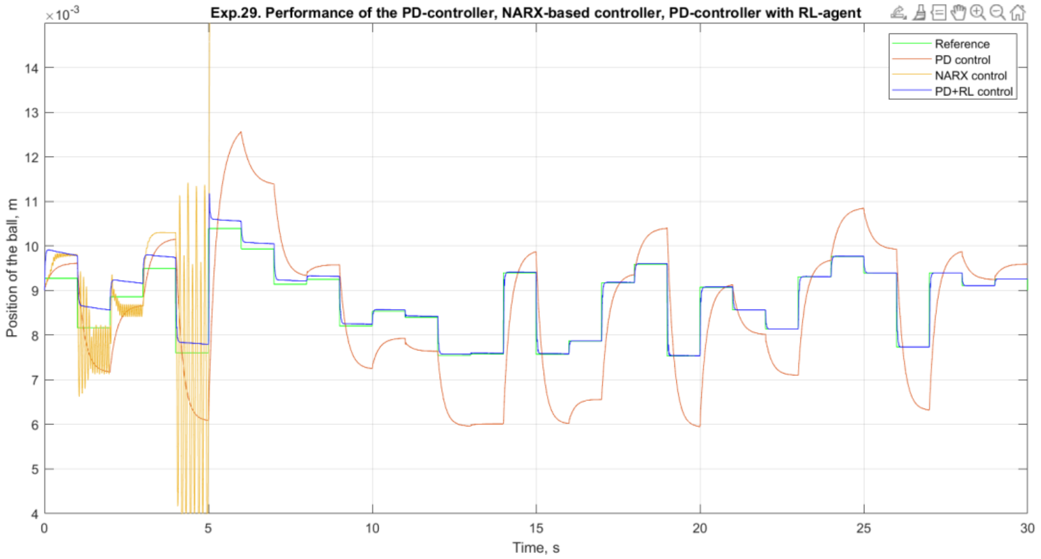

Exp.27. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent
×10^-3
Reference
PD control
NARX control
PD+RL control
Position of the ball, m
4
5
6
7
8
9
10
11
12
13
14
0
5
10
15
20
25
30
Time, s

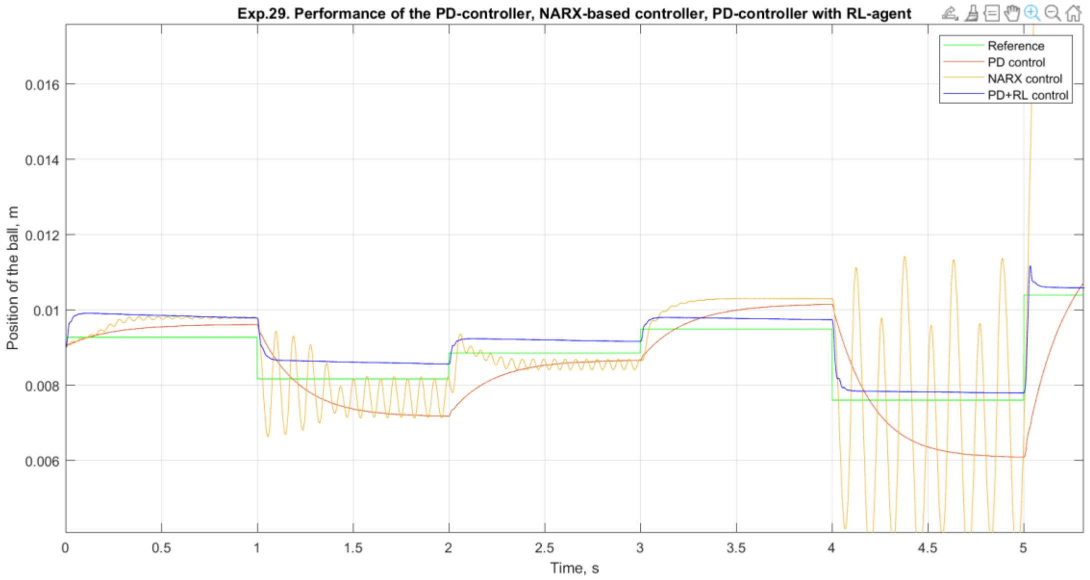

Exp.27. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent
×10^-3
Reference
PD control
NARX control
PD+RL control
Position of the ball, m
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
Time, s

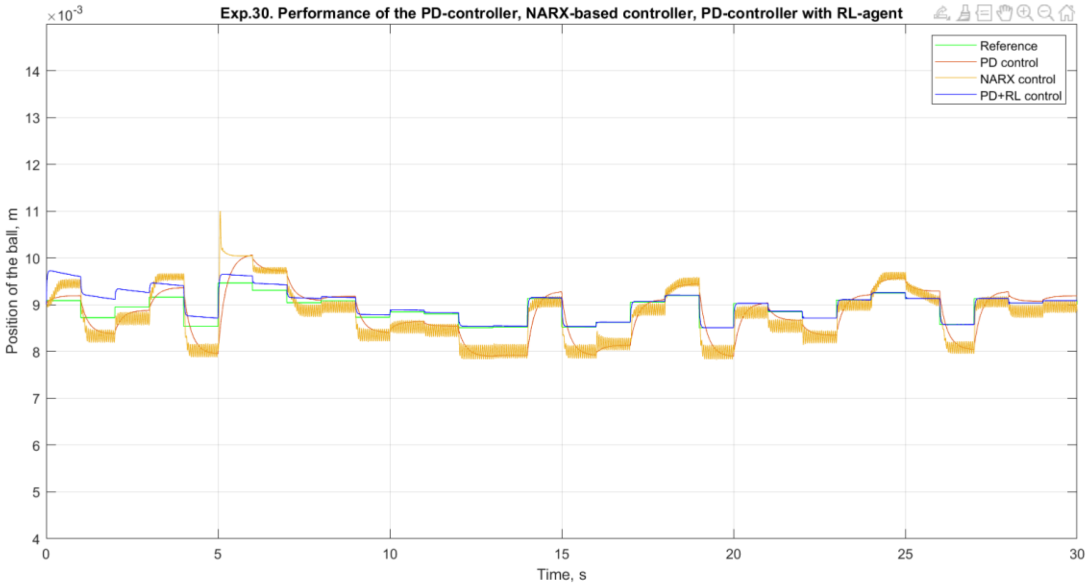
Appendix 9 Plots in experiments for controller 2

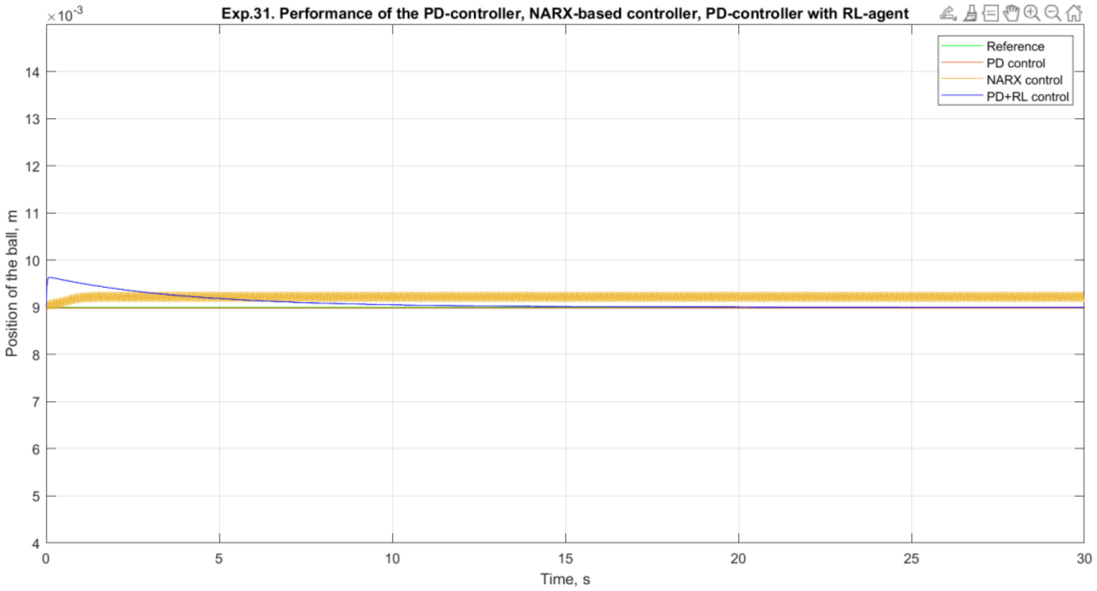


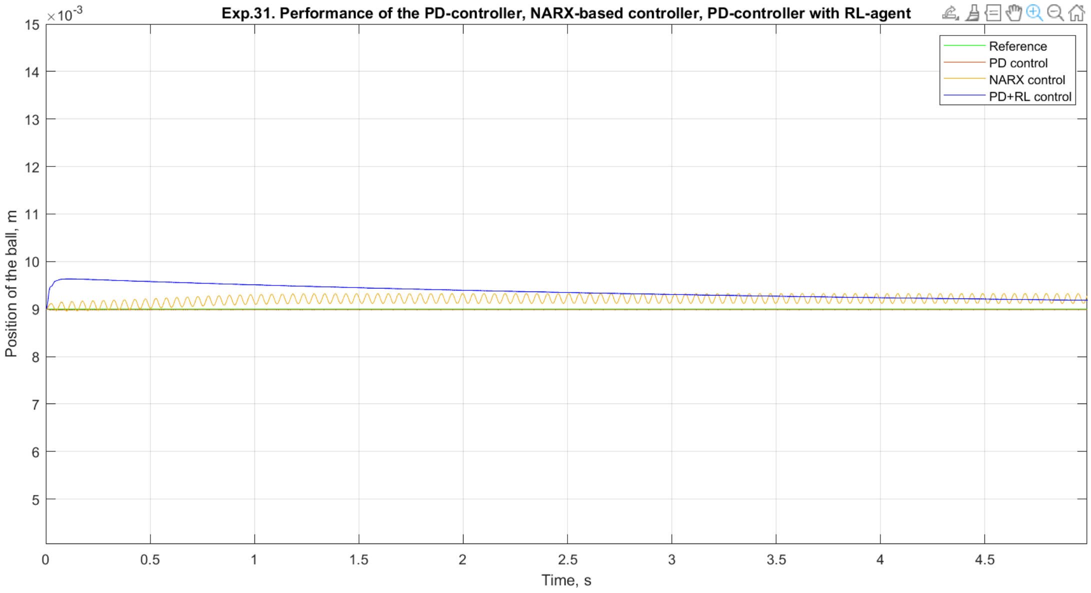


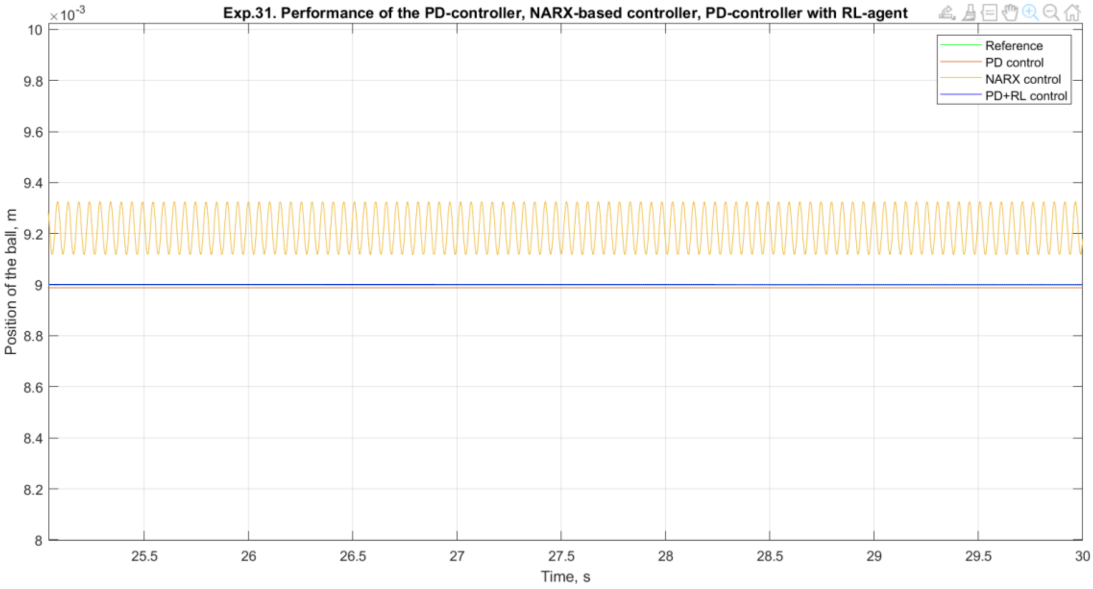


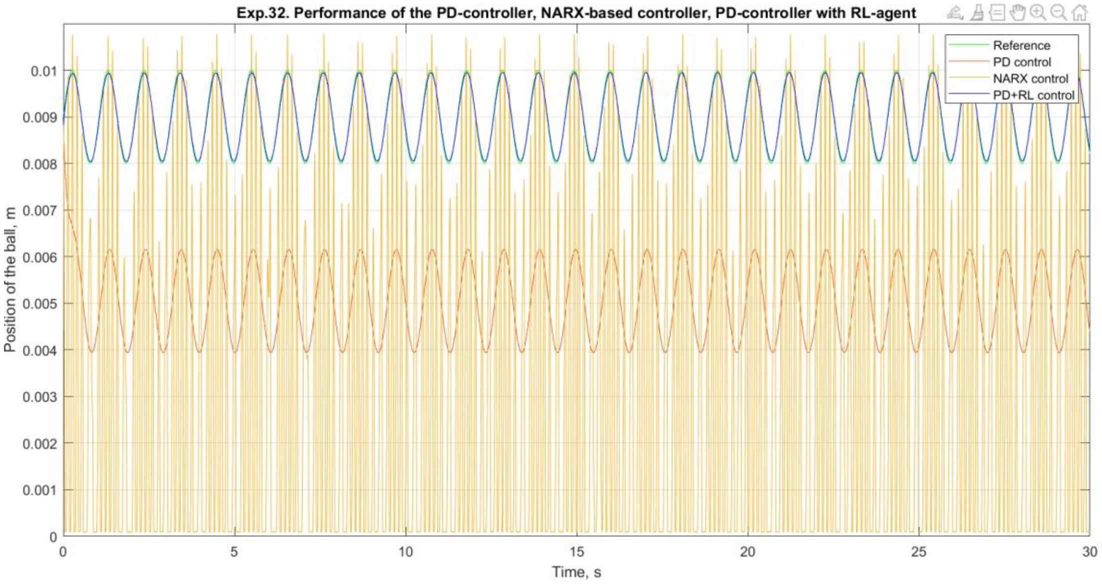

Exp.29. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent
Reference
PD control
NARX control
PD+RL control
Position of the ball, m
0.016
0.014
0.012
0.01
0.008
0.006
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
Time, s

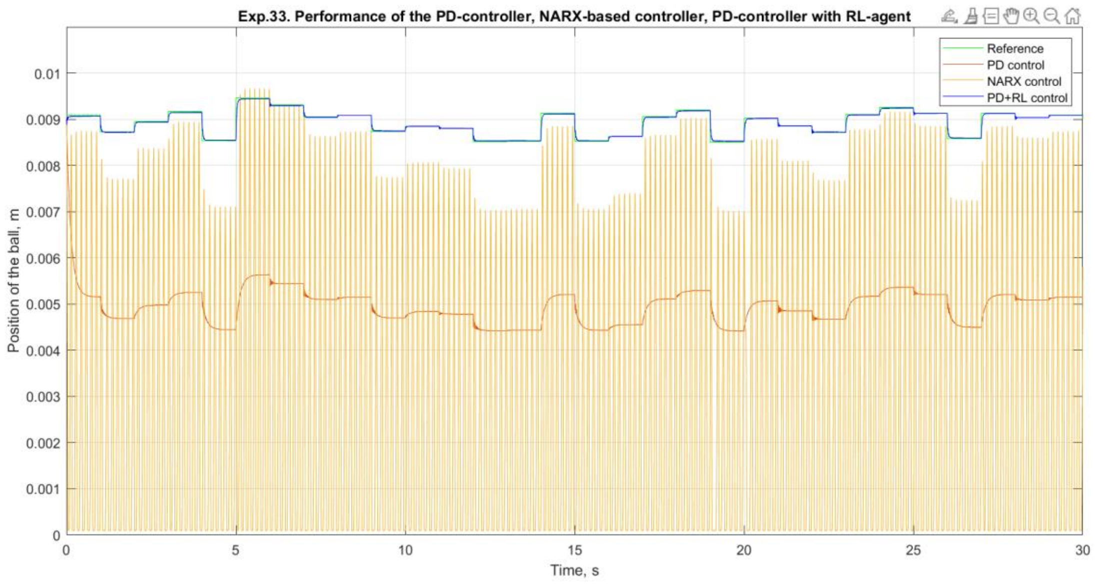

×10-3
Exp.30. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent
Reference
PD control
NARX control
PD+RL control
Position of the ball, m
14
13
12
11
10
9
8
7
6
5
4
0
5
10
15
20
25
30
Time, s

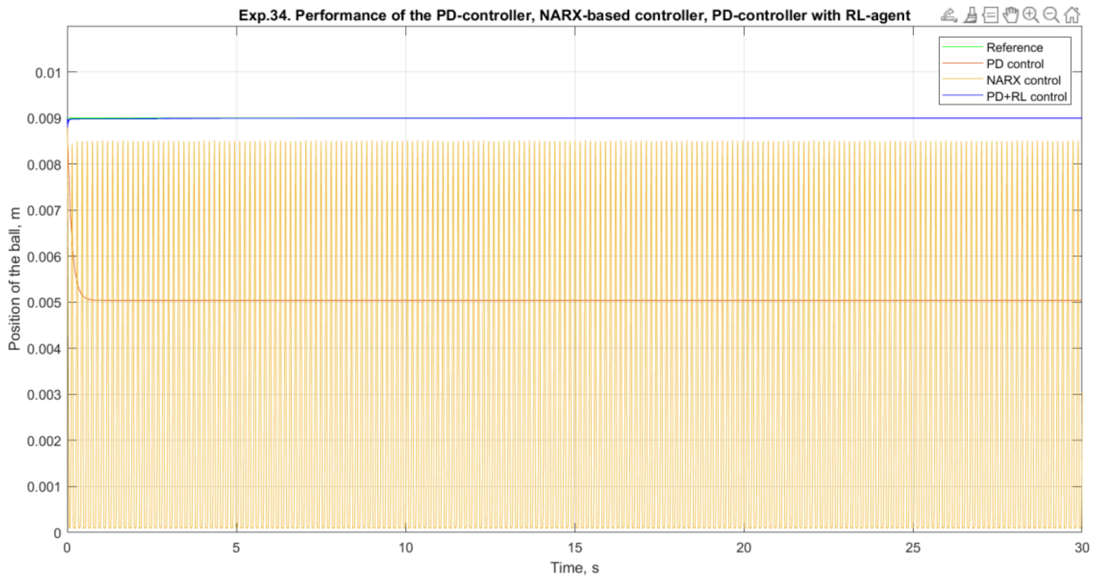

×10-3
Exp.31. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent
Reference
PD control
NARX control
PD+RL control
Position of the ball, m
14
13
12
11
10
9
8
7
6
5
4
0
5
10
15
20
25
30
Time, s

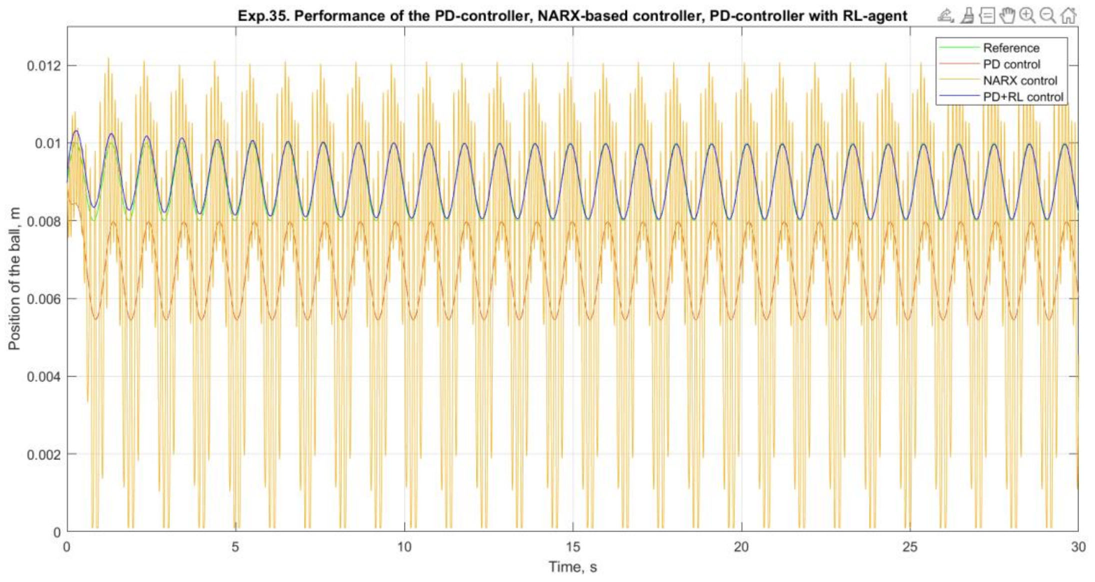

Exp.31. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent
Reference
PD control
NARX control
PD+RL control
×10^-3
15
14
13
12
11
10
9
8
7
6
5
Position of the ball, m
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
Time, s

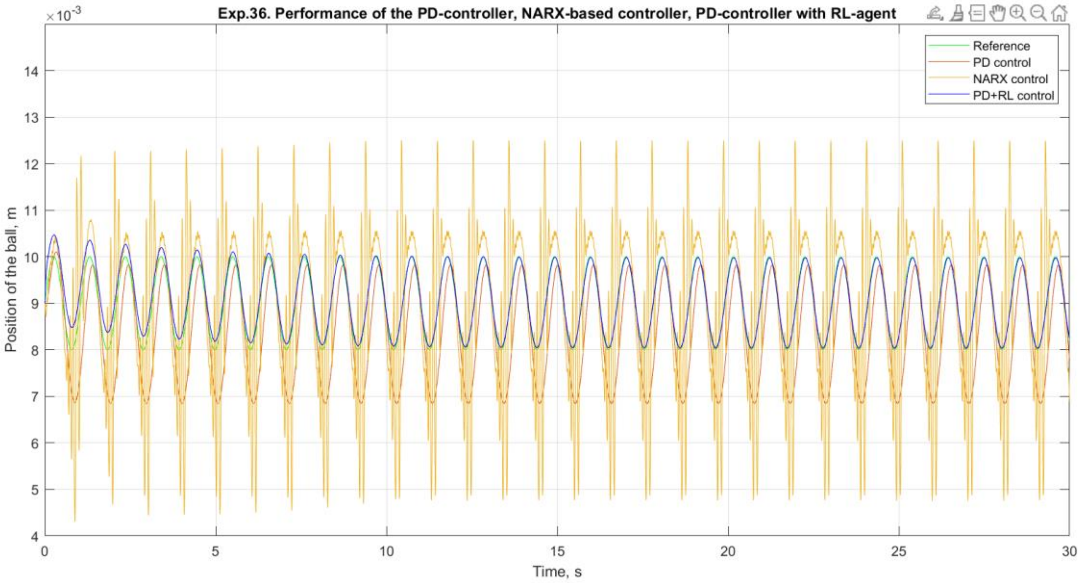

Exp.31. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent
Reference
PD control
NARX control
PD+RL control
×10^-3
10
9.8
9.6
9.4
9.2
9
8.8
8.6
8.4
8.2
8
Position of the ball, m
25.5
26
26.5
27
27.5
28
28.5
29
29.5
30
Time, s

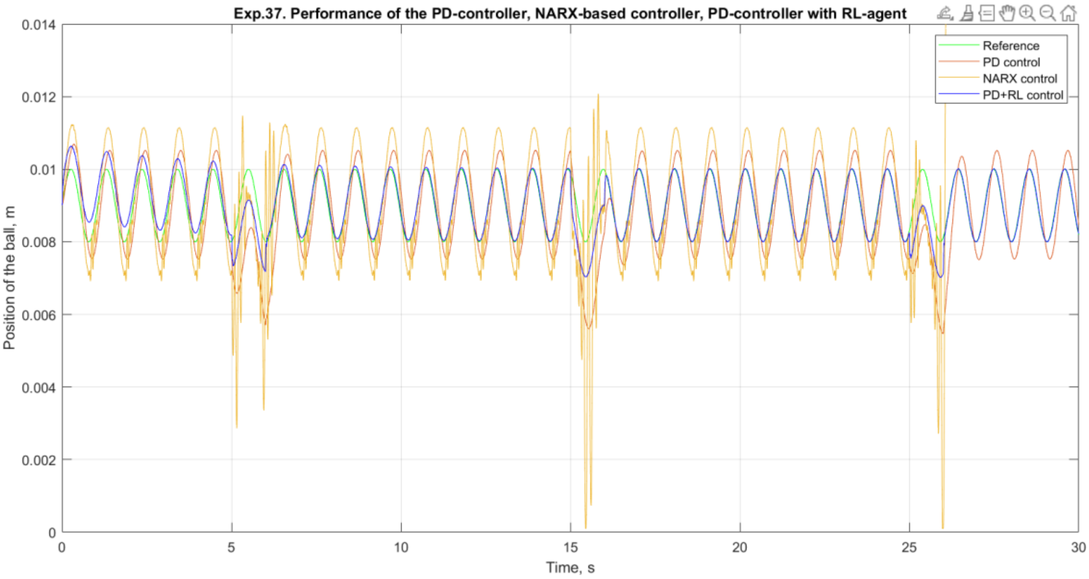

Exp.32. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent
Reference
PD control
NARX control
PD+RL control
0.01
0.009
0.008
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0
Position of the ball, m
0
5
10
15
20
25
30
Time, s

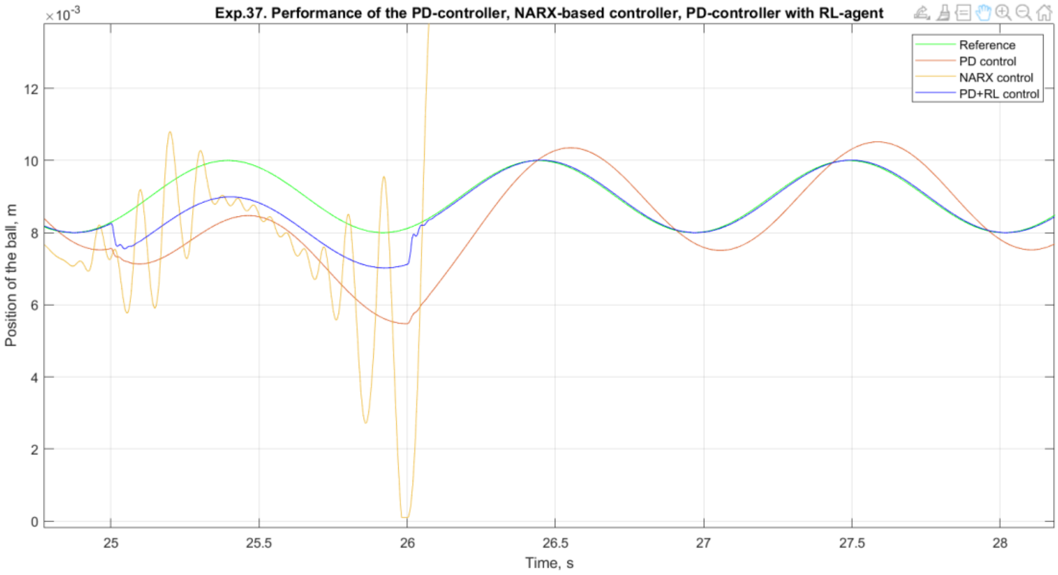

Exp.33. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent
Reference
PD control
NARX control
PD+RL control
Position of the ball, m
0
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
0.009
0.01
Time, s
0
5
10
15
20
25
30

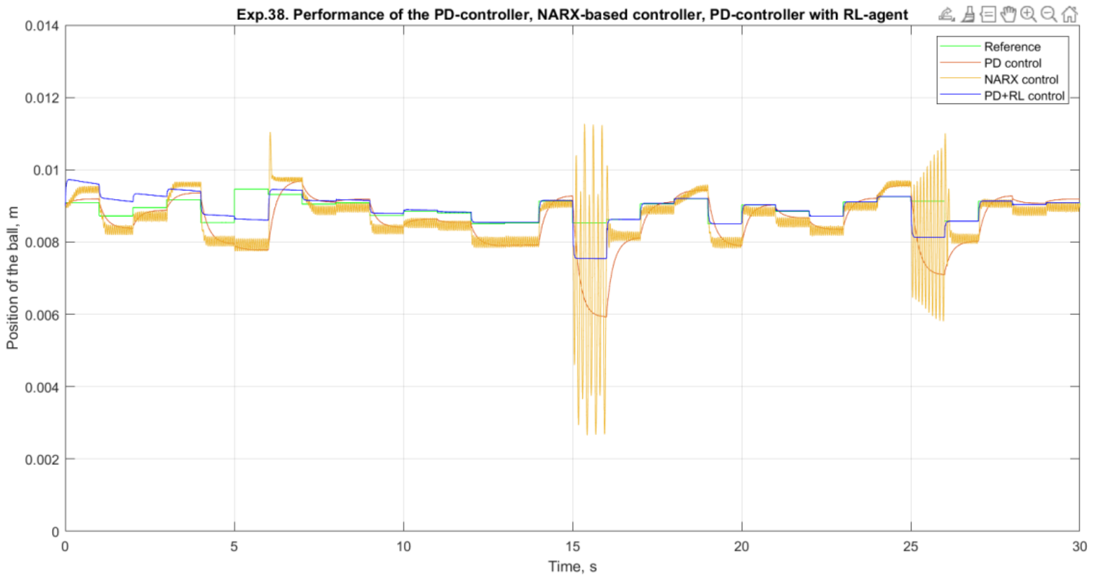

Exp.34. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent
Reference
PD control
NARX control
PD+RL control
Position of the ball, m
0
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
0.009
0.01
Time, s
0
5
10
15
20
25
30

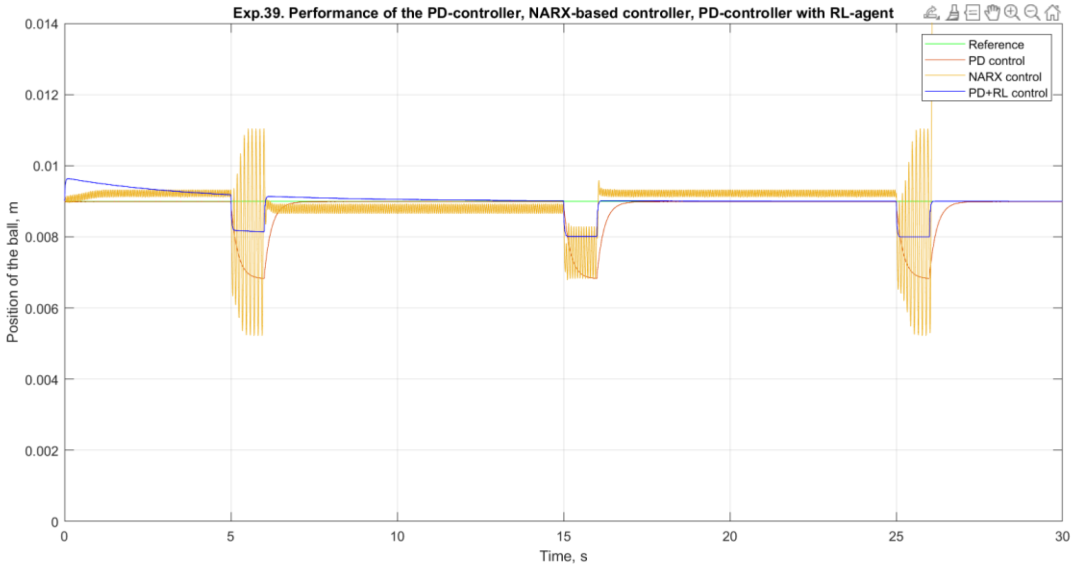

Exp.35. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent
Reference
PD control
NARX control
PD+RL control
Position of the ball, m
0
0.002
0.004
0.006
0.008
0.01
0.012
Time, s
0
5
10
15
20
25
30

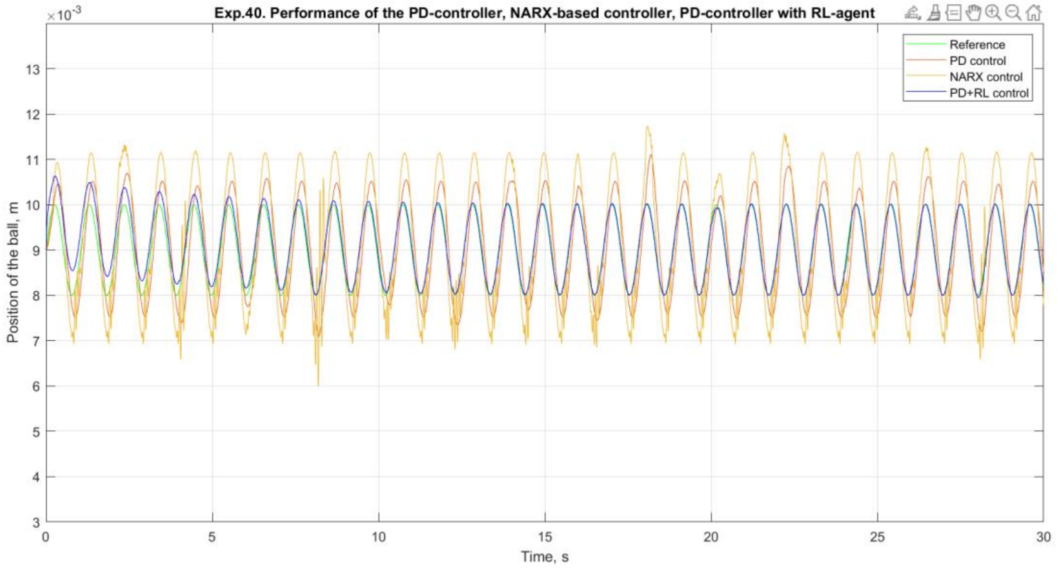

Exp.36. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent
×10⁻³
Reference
PD control
NARX control
PD+RL control
14
13
12
11
10
9
8
7
6
5
4
Position of the ball, m
0
5
10
15
20
25
30
Time, s

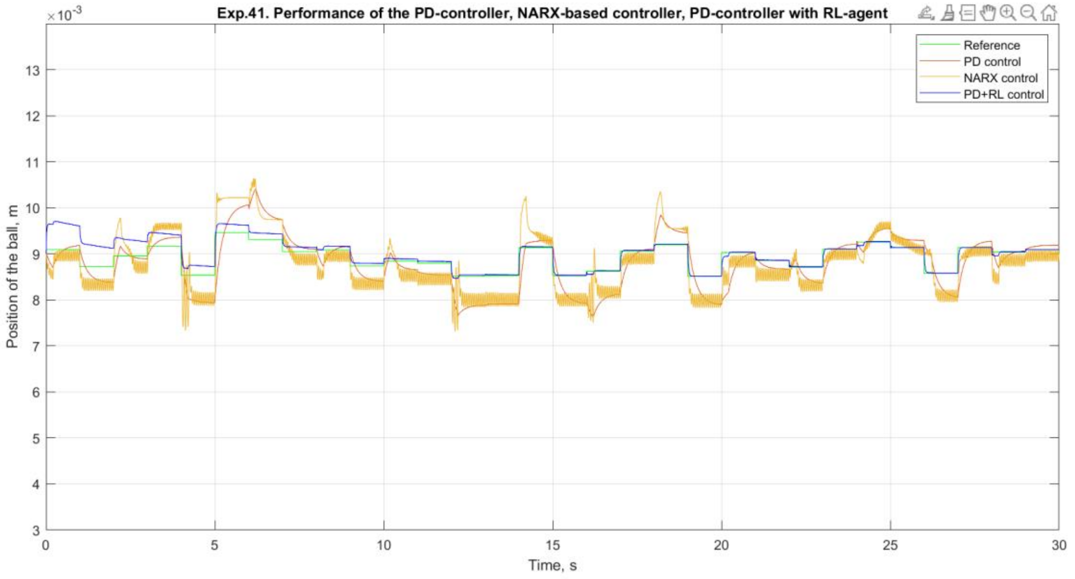

Exp.37. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent
Reference
PD control
NARX control
PD+RL control
0.014
0.012
0.01
0.008
0.006
0.004
0.002
0
Position of the ball, m
0
5
10
15
20
25
30
Time, s

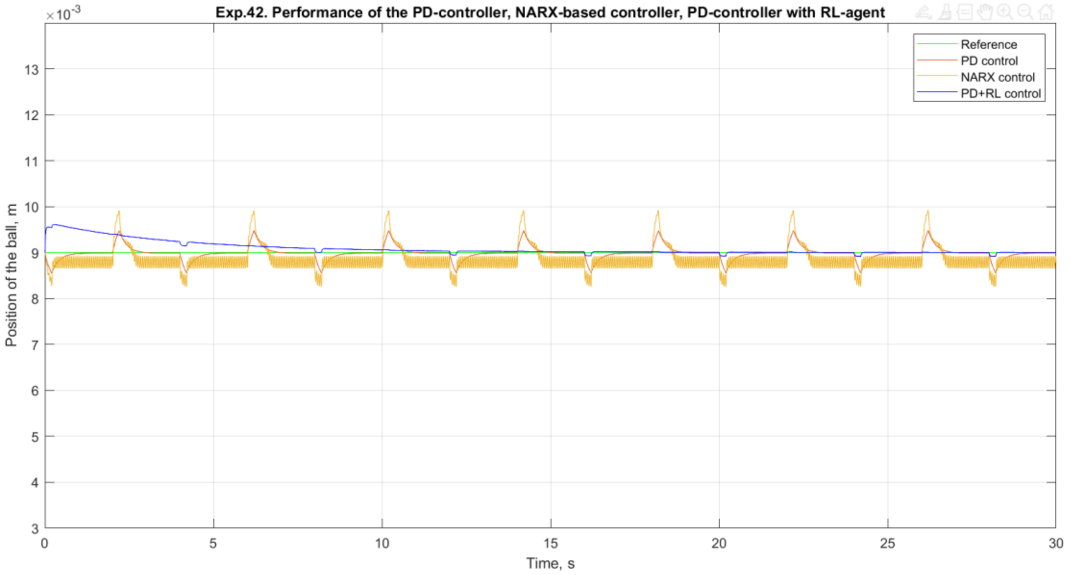

Exp.37. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent
×10⁻³
Reference
PD control
NARX control
PD+RL control
12
10
8
6
4
2
0
Position of the ball, m
25
25.5
26
26.5
27
27.5
28
Time, s

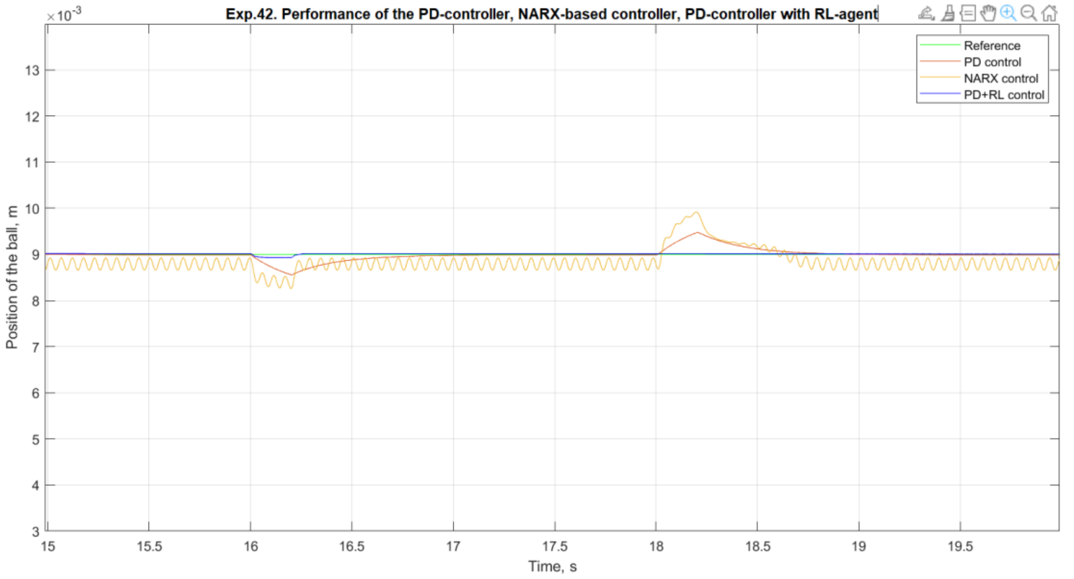

Exp.38. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent
Reference
PD control
NARX control
PD+RL control
Position of the ball, m
0.014
0.012
0.01
0.008
0.006
0.004
0.002
0
0
5
10
15
20
25
30
Time, s


Exp.39. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent
Reference
PD control
NARX control
PD+RL control
Position of the ball, m
0.014
0.012
0.01
0.008
0.006
0.004
0.002
0
0
5
10
15
20
25
30
Time, s

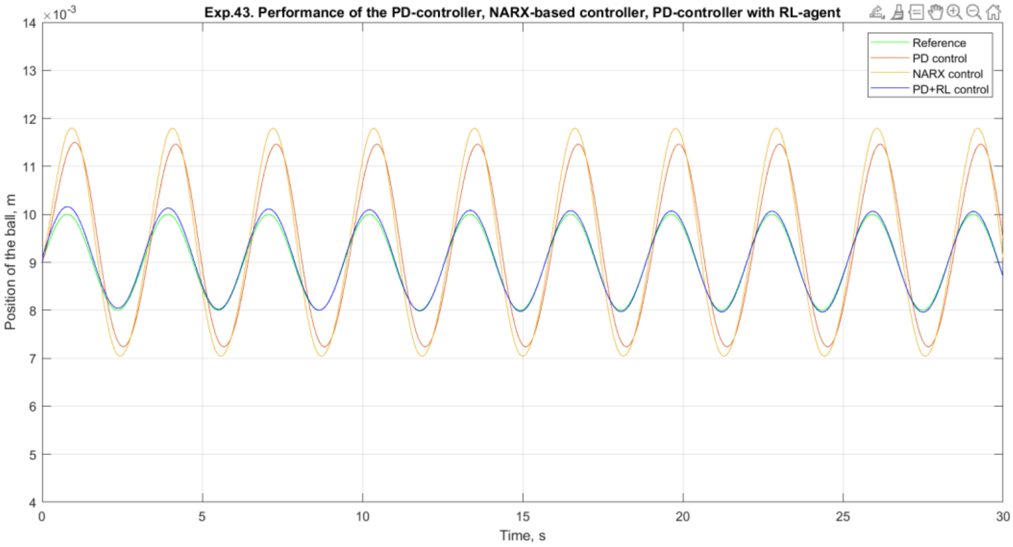

Exp.40. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent
×10-3
Reference
PD control
NARX control
PD+RL control
Position of the ball, m
13
12
11
10
9
8
7
6
5
4
3
0
5
10
15
20
25
30
Time, s

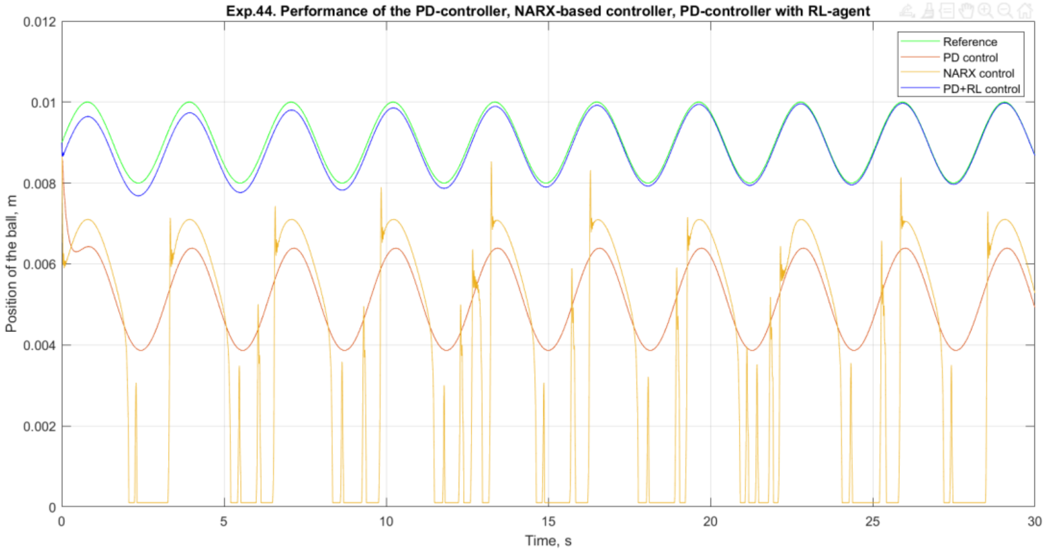

Exp.41. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent
Reference
PD control
NARX control
PD+RL control
$\times 10^{-3}$
Position of the ball, m
3
4
5
6
7
8
9
10
11
12
13
Time, s
0
5
10
15
20
25
30

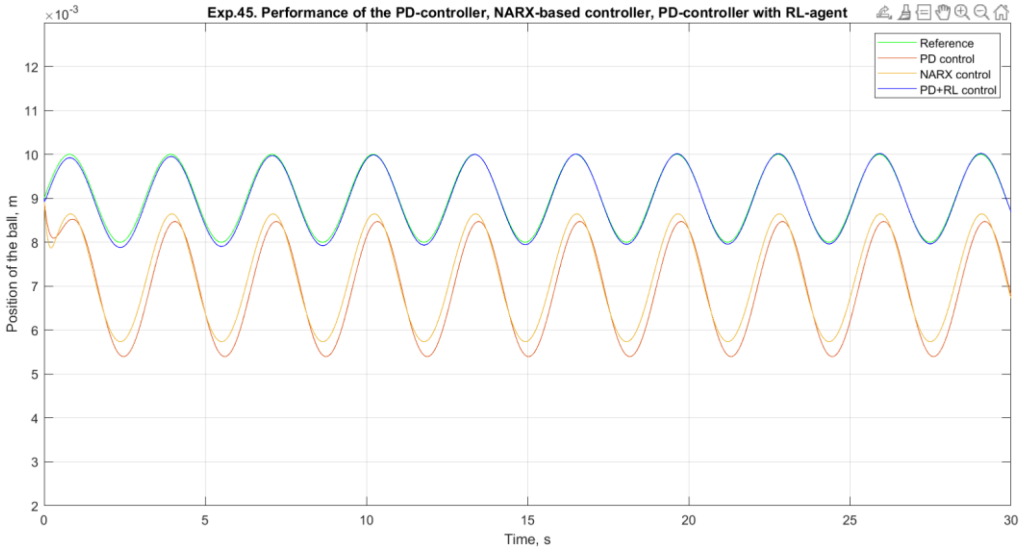

Exp.42. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent
Reference
PD control
NARX control
PD+RL control
$\times 10^{-3}$
Position of the ball, m
3
4
5
6
7
8
9
10
11
12
13
Time, s
0
5
10
15
20
25
30

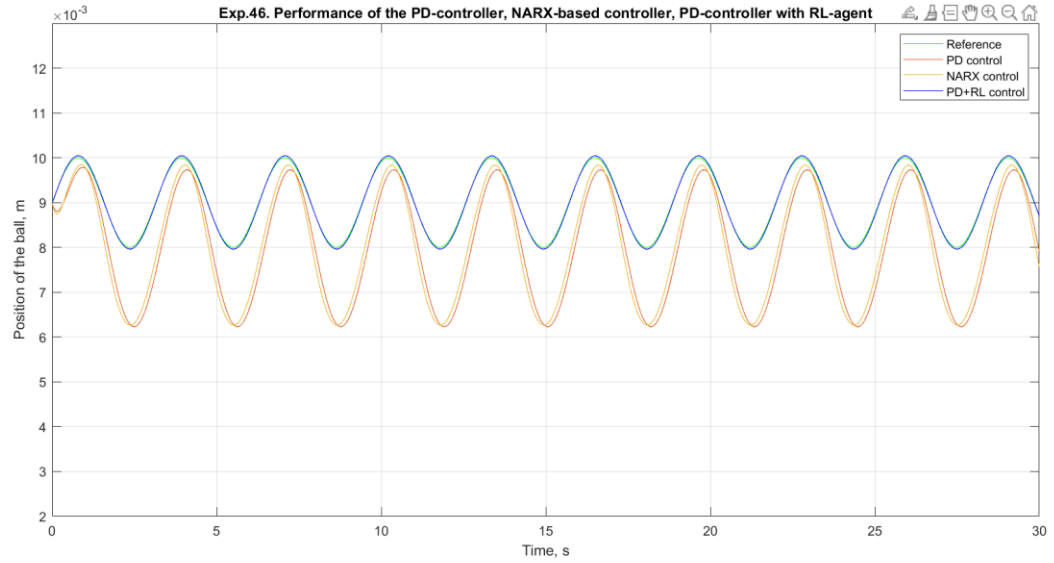

Exp.42. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent
Reference
PD control
NARX control
PD+RL control
$\times 10^{-3}$
Position of the ball, m
3
4
5
6
7
8
9
10
11
12
13
Time, s
15
15.5
16
16.5
17
17.5
18
18.5
19
19.5

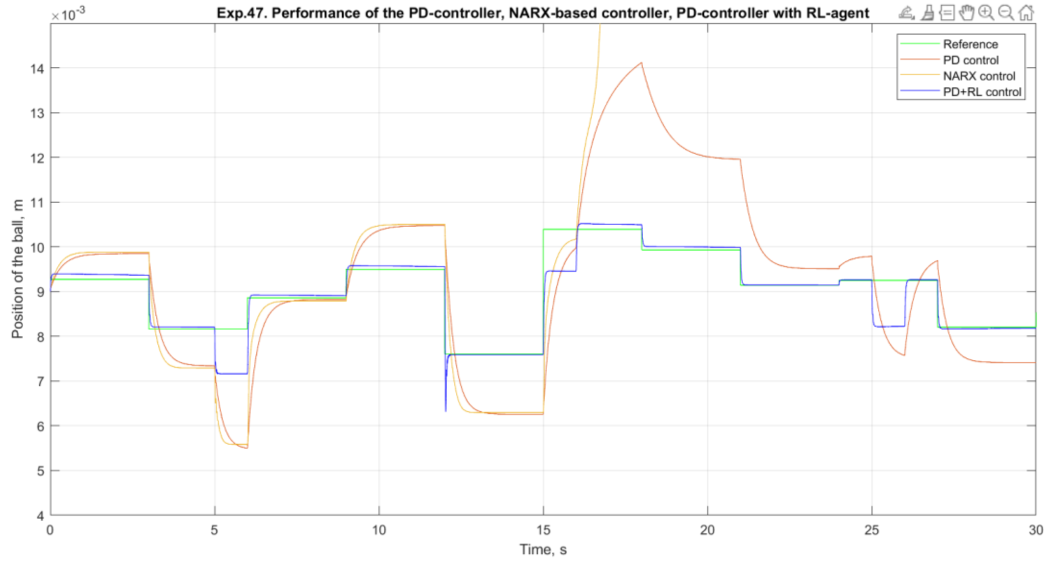
Appendix 10 Plots in experiments for controller 4

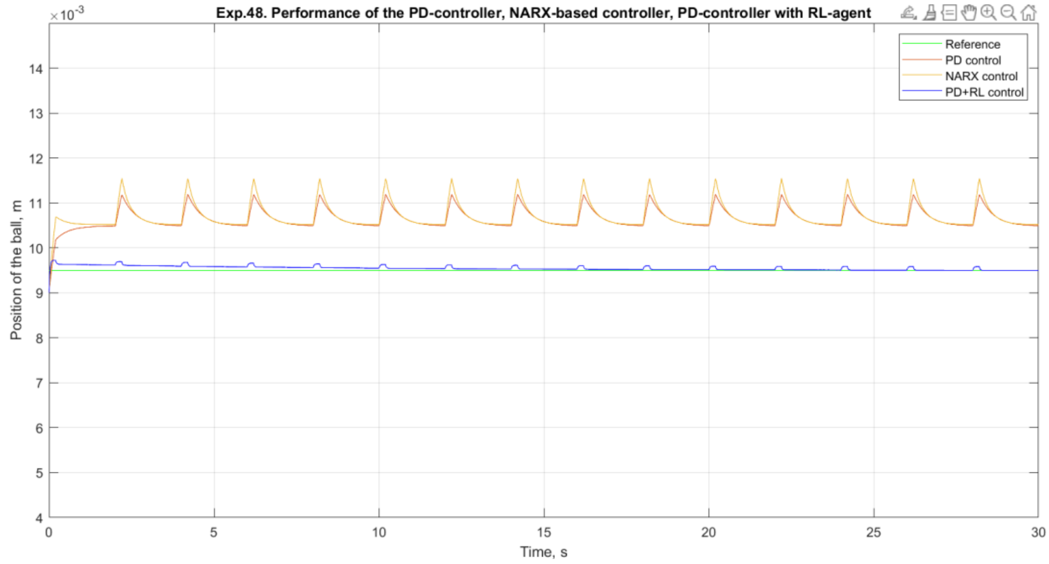




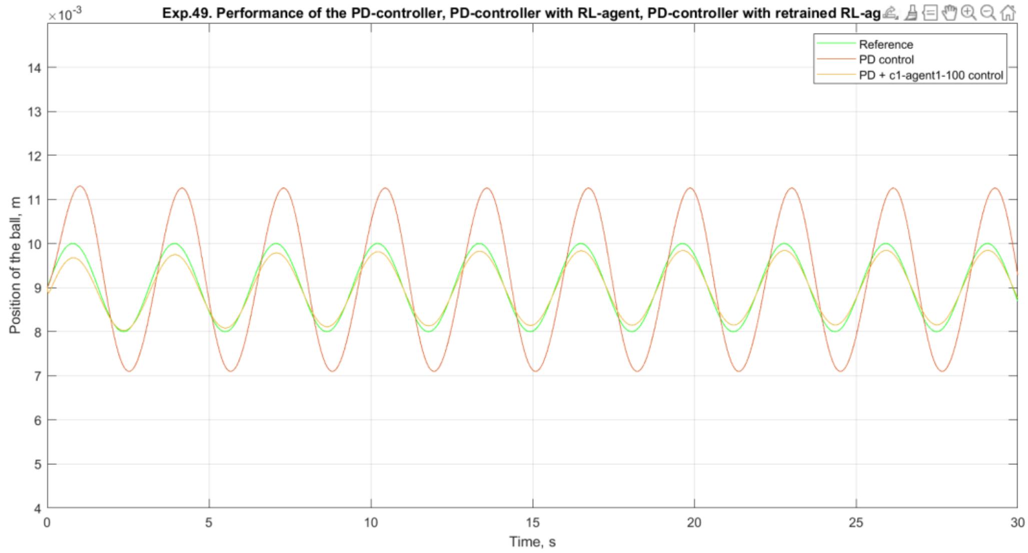


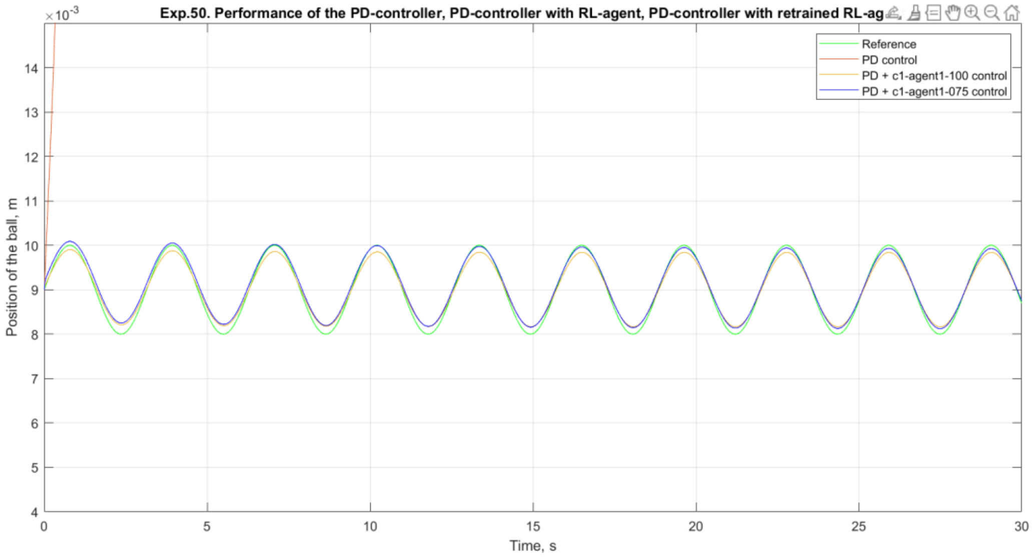

Exp.46. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent
×10-3
Reference
PD control
NARX control
PD+RL control
12
11
10
9
8
7
6
5
4
3
2
Position of the ball, m
0
5
10
15
20
25
30
Time, s

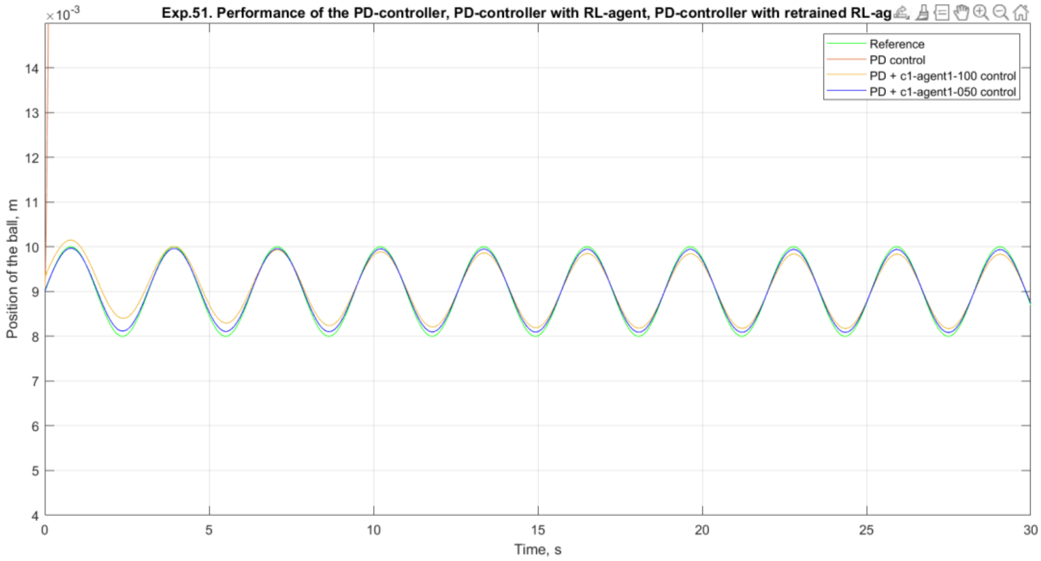

Exp.47. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent
×10-3
Reference
PD control
NARX control
PD+RL control
14
13
12
11
10
9
8
7
6
5
4
Position of the ball, m
0
5
10
15
20
25
30
Time, s

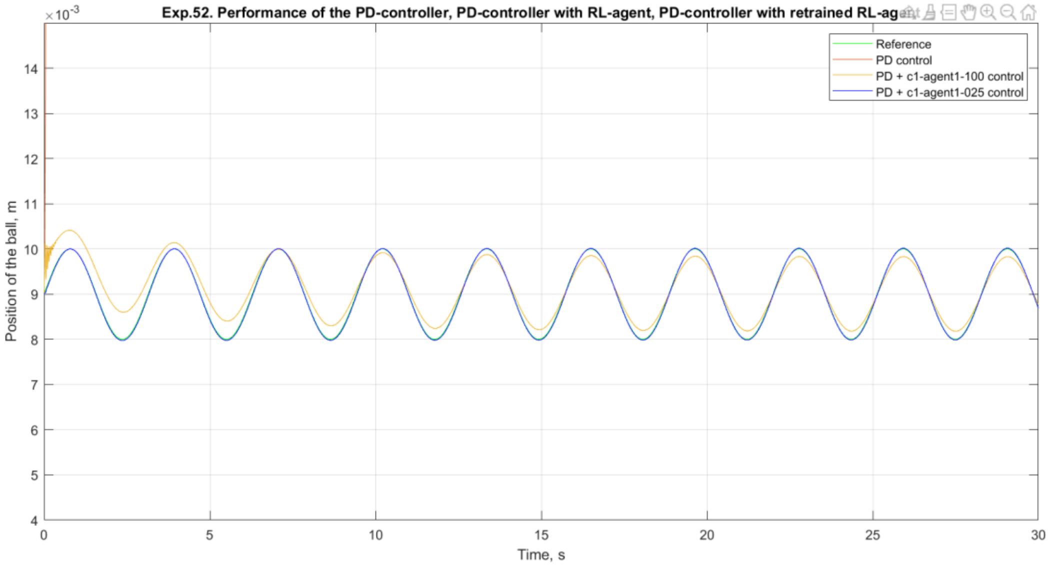

Exp.48. Performance of the PD-controller, NARX-based controller, PD-controller with RL-agent
×10-3
Reference
PD control
NARX control
PD+RL control
14
13
12
11
10
9
8
7
6
5
4
Position of the ball, m
0
5
10
15
20
25
30
Time, s

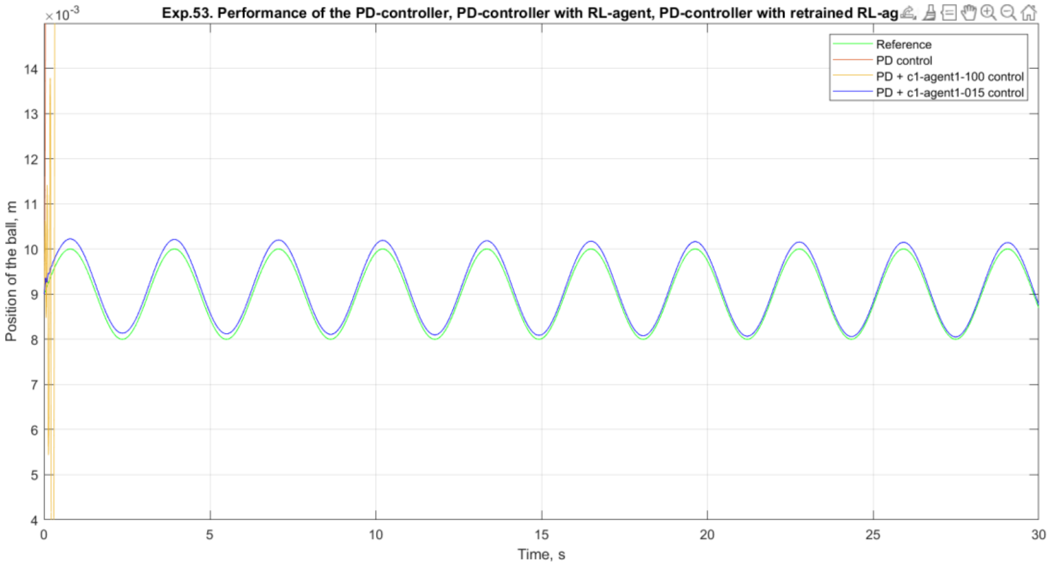
Appendix 11 Plots in experiments for controller 1 with graded learning

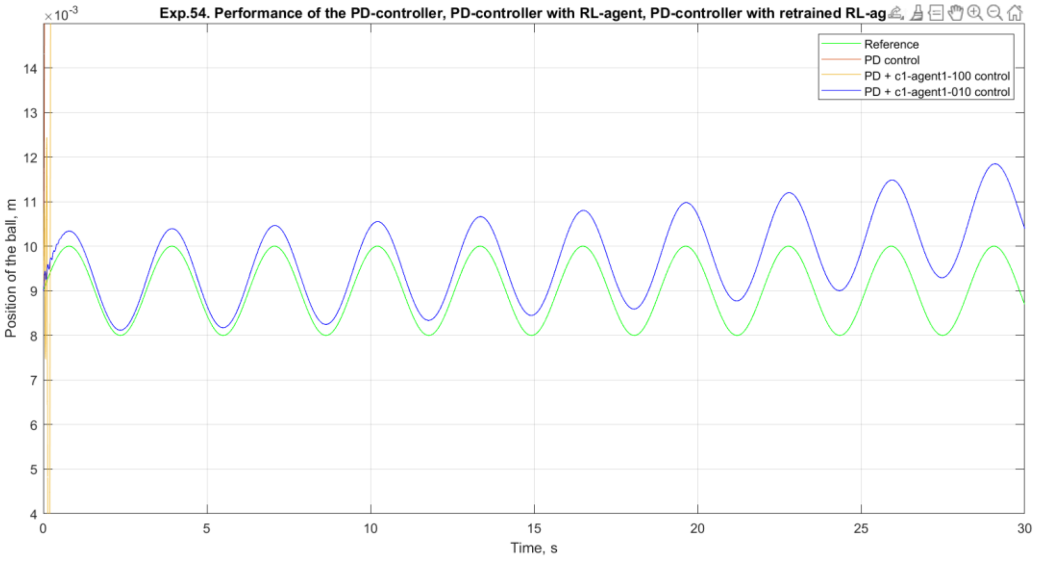


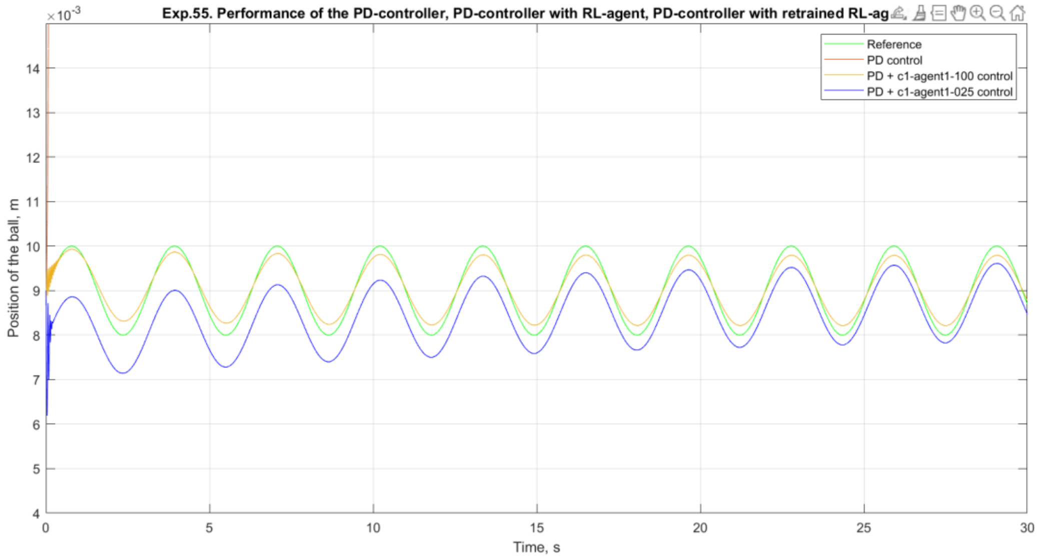


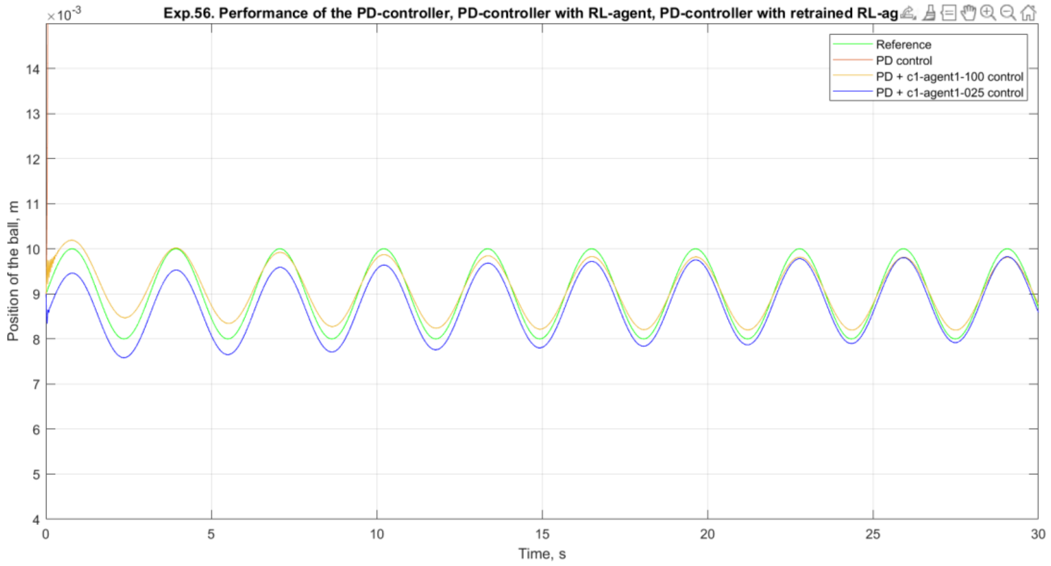


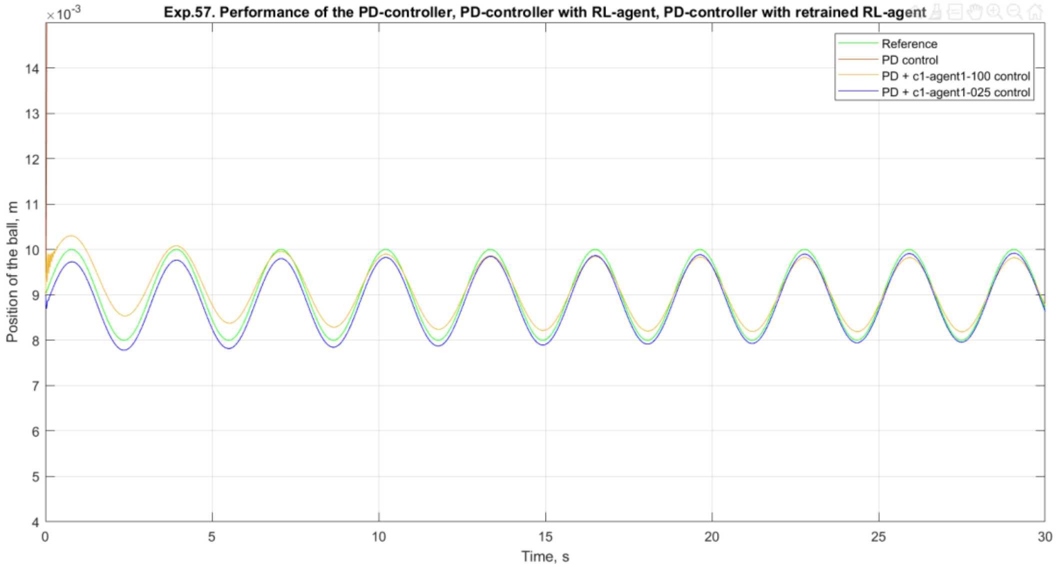

Exp.52. Performance of the PD-controller, PD-controller with RL-agent, PD-controller with retrained RL-ag
Reference
PD control
PD + c1-agent1-100 control
PD + c1-agent1-025 control
×10^-3
14
13
12
11
10
9
8
7
6
5
4
Position of the ball, m
0
5
10
15
20
25
30
Time, s

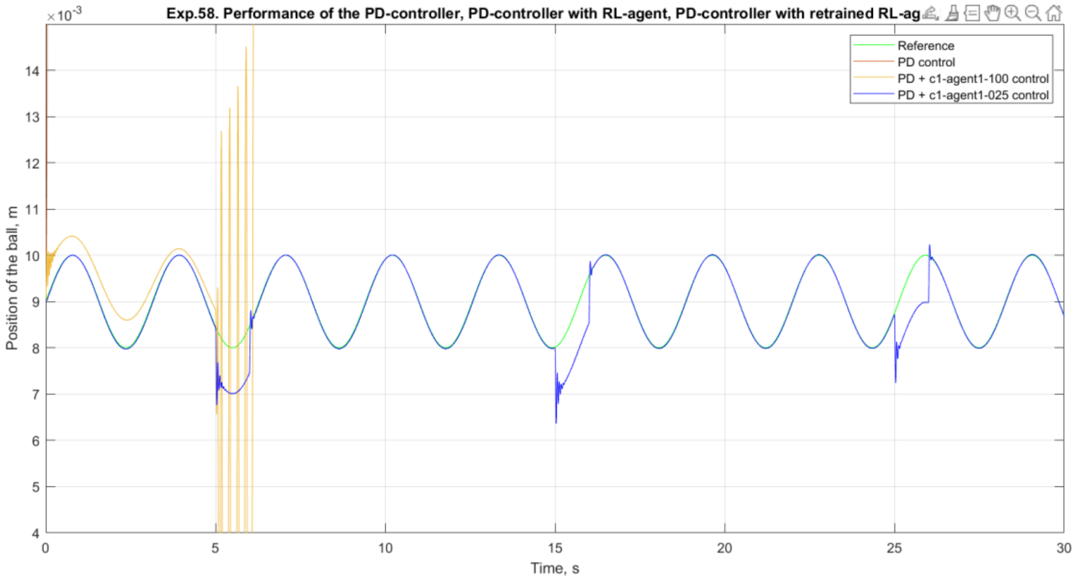

Exp.53. Performance of the PD-controller, PD-controller with RL-agent, PD-controller with retrained RL-ag
Reference
PD control
PD + c1-agent1-100 control
PD + c1-agent1-015 control
×10^-3
14
13
12
11
10
9
8
7
6
5
4
Position of the ball, m
0
5
10
15
20
25
30
Time, s

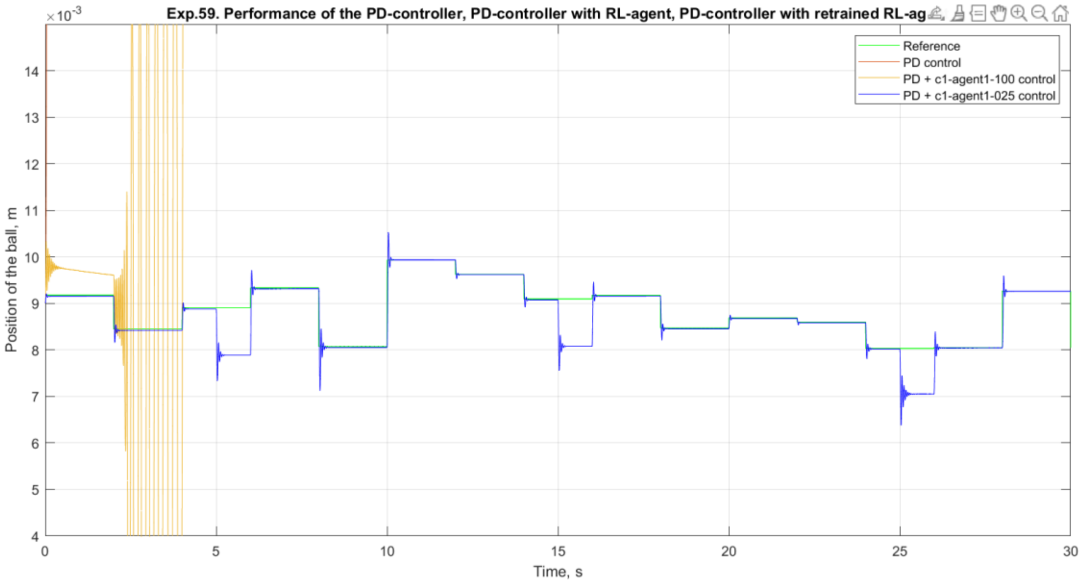

Exp.54. Performance of the PD-controller, PD-controller with RL-agent, PD-controller with retrained RL-ag
Reference
PD control
PD + c1-agent1-100 control
PD + c1-agent1-010 control
×10^-3
14
13
12
11
10
9
8
7
6
5
4
Position of the ball, m
0
5
10
15
20
25
30
Time, s

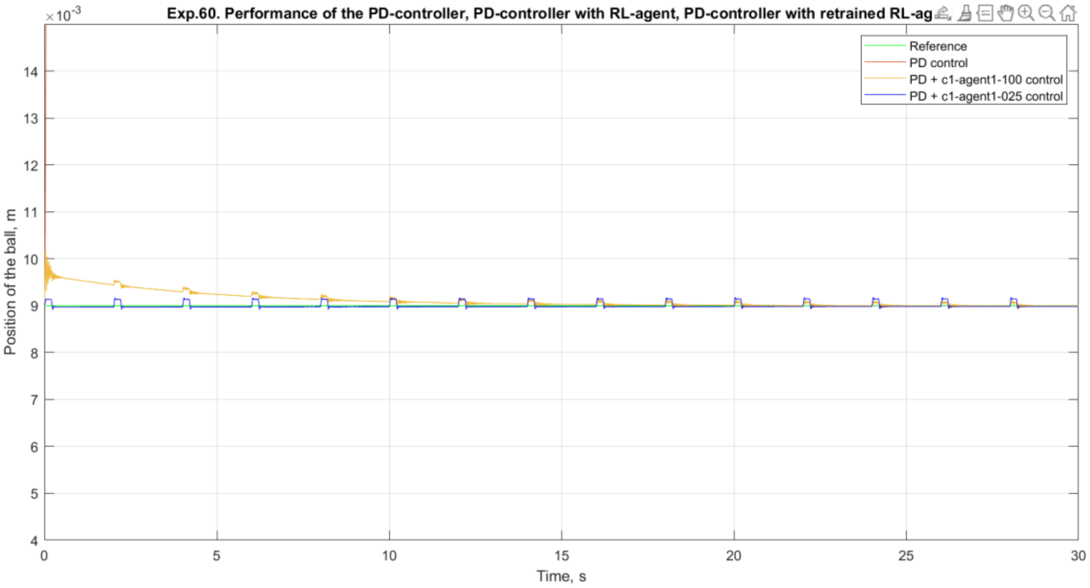

Exp.55. Performance of the PD-controller, PD-controller with RL-agent, PD-controller with retrained RL-ag
×10^-3
14
13
12
11
10
9
8
7
6
5
4
Position of the ball, m
0
5
10
15
20
25
30
Time, s
Reference
PD control
PD + c1-agent1-100 control
PD + c1-agent1-025 control

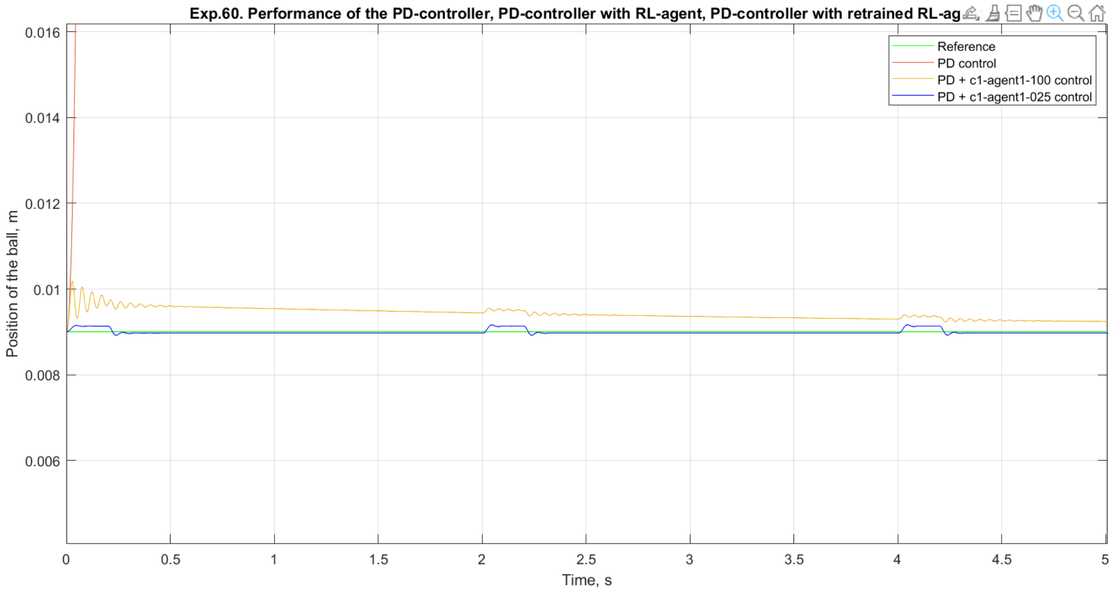

Exp.56. Performance of the PD-controller, PD-controller with RL-agent, PD-controller with retrained RL-ag
×10^-3
14
13
12
11
10
9
8
7
6
5
4
Position of the ball, m
0
5
10
15
20
25
30
Time, s
Reference
PD control
PD + c1-agent1-100 control
PD + c1-agent1-025 control


Exp.57. Performance of the PD-controller, PD-controller with RL-agent, PD-controller with retrained RL-agent
×10^-3
14
13
12
11
10
9
8
7
6
5
4
Position of the ball, m
0
5
10
15
20
25
30
Time, s
Reference
PD control
PD + c1-agent1-100 control
PD + c1-agent1-025 control


Exp.58. Performance of the PD-controller, PD-controller with RL-agent, PD-controller with retrained RL-ag
×10^-3
Position of the ball, m
4
5
6
7
8
9
10
11
12
13
14
0
5
10
15
20
25
30
Time, s
Reference
PD control
PD + c1-agent1-100 control
PD + c1-agent1-025 control


Exp.59. Performance of the PD-controller, PD-controller with RL-agent, PD-controller with retrained RL-ag
×10^-3
Position of the ball, m
4
5
6
7
8
9
10
11
12
13
14
0
5
10
15
20
25
30
Time, s
Reference
PD control
PD + c1-agent1-100 control
PD + c1-agent1-025 control


Exp.60. Performance of the PD-controller, PD-controller with RL-agent, PD-controller with retrained RL-ag
×10^-3
Position of the ball, m
4
5
6
7
8
9
10
11
12
13
14
0
5
10
15
20
25
30
Time, s
Reference
PD control
PD + c1-agent1-100 control
PD + c1-agent1-025 control


Exp.60. Performance of the PD-controller, PD-controller with RL-agent, PD-controller with retrained RL-ag
Reference
PD control
PD + c1-agent1-100 control
PD + c1-agent1-025 control
Position of the ball, m
0.016
0.014
0.012
0.01
0.008
0.006
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
Time, s