BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF MECHANICAL ENGINEERING
FAKULTA STROJNÍHO INŽENÝRSTVÍ

INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS
ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY

CONTROL OF NONLINEAR SYSTEMS USING LOCAL APPROXIMATION METHODS
ŘÍZENÍ NELINEÁRNÍCH SYSTÉMŮ S VYUŽITÍM LOKÁLNÍCH APROXIMAČNÍCH METOD

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR
AUTOR PRÁCE

Bc. Martin Brablc

SUPERVISOR
VEDOUCÍ PRÁCE

doc. Ing. Robert Grepl, Ph.D.

BRNO 2016

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF MECHANICAL ENGINEERING

Master's Thesis Assignment

Institut:	Institute of Solid Mechanics, Mechatronics and Biomechanics
Student:	**Bc. Martin Brablc**
Degree programm:	Applied Sciences in Engineering
Branch:	Mechatronics
Supervisor:	**doc. Ing. Robert Grepl, Ph.D.**
Academic year:	2015/16

As provided for by the Act No. 111/98 Coll. on higher education institutions and the BUT Study and Examination Regulations, the director of the Institute hereby assigns the following topic of Master's Thesis:

Control of Nonlinear Systems using Local Approximation Methods

Brief description:

This thesis deals with the application and development of methods for automatic design of inverse dynamic model of the controlled system. Dynamic model is used as feedforward controller. The thesis is focused on a specific class of systems – electromechanical actuators with nonlinear spring and significant dry friction. The work is aimed at the design and simulation and experimental verification of the methods based on local linear models.

Master's Thesis goals:

1) Perform a search study in the field of modelling of nonlinear systems inverse dynamics and using these model to control nonlinear systems.
2) Evaluate the application of the chosen methods under real conditions, for example liability to noise, computational complexity or the ability to approximate the behaviour of real systems.
3) Perform simulation verification of various methods for modelling the inverse dynamics of the controlled system. Focus on approximation methods which use local linear models ("lazy learning" category, RFWR).
4) Implement the chosen method for modelling the inverse dynamics on a microcontroller. The focus is mainly on the versatility of the final control unit.
5) Validate the developed control algorithm on real systems, for example on an electronic throttle.

Bibliography:

Nelles, O.: Nonlinear System Identification, Springer 2001

Jung, L.: System Identification, 2009

Valášek, M.: Mechatronika, Vydavatelství ČVUT 1995

Noskievič, P.: Modelování a identifikace systémů, 1999

Students are required to submit the thesis within the deadlines stated in the schedule of the academic year 2015/16.

In Brno, 30. 11. 2015

L. S.

prof. Ing. Jindřich Petruška, CSc.
Director of the Institute

doc. Ing. Jaroslav Katolický, Ph.D.
FME Dean

Abstrakt

Tato práce se zabývá návrhem adaptivního řídícího algortitmu pro konkrétní třídu elektromechanických aktuátorů, založeného na principu dopředného řízení pomocí inverzního dynamického modelu. Adaptibilita řízení spočívá v mechanismu získání inverzního dynamického modelu. Tato práce se zaměřuje na jeho online aproximaci pomocí lokálních aproximačních metod.

Výstupem práce je shrnutí analýzy, simulačního testování a reálných experimentů, které testovaly možnosti praktického využití lokálných aproximačních metod pro účely adaptivního řízení v reálném prostředí.

Summary

This thesis deals with the development of an adaptive control algorithm for a specific class of electromechanical actuators, based on the feedforward compensation principle using an inverse dynamic model. The control algorithm's adaptability originates in a mechanism of learning the inverse dynamic model. This thesis focuses on using local approximation methods for an online inverse model learning.

The outcome of this thesis is a summary of the analysis, simulations and actual experiments, which tested the possibilities of using the local approximation methods for adaptive control purposes in real environment.

Klíčová slova

Lokální aproximace, adaptivní řízení, feedforward kompenzace, kompozitní řízení, metoda nejmenších čtverců, LS, RLS, LWL, RFWR, LOLIMOT, lazy learning, řízení nelineárních systémů

Keywords

Local approximation, adaptive control, feedforward compensation, composite control, the Least Squares method, LS, RLS, LWL, RFWR, LOLIMOT, lazy learning, nonlinear systems control

Bibliographic citation

BRABLC, M. *Control of Nonlinear Systems using Local Approximation Methods*. Brno: Brno University of Technology, Faculty of mechanical engineering, 2016. 75 pages, Master's thesis supervisor: doc. Ing. Robert Grepl, PhD..

I affirm that the presented master's thesis is my genuine work and that it was created with the support of the stated literature, under the supervision of my tutor.

Martin Brablc

Brno . . . .

I would like to thank my supervisor, doc. Ing. Robert Grepl, PhD., for his guidance and support during the development of this thesis. Also, I would like to express my gratitude to Ing.Václav Sova for many insightful conversations which helped me formulate many ideas presented in this thesis, and all members of the Mechatronics Laboratory staff for creating such a work-supportive atmosphere.

Martin Brablc

Contents

1 Introduction

During the last several decades, the performance on microcontrollers was continuously improving. Nowadays, it is coming to a point, when even a very cheap microcontroller can provide enough computational power that, when used to control a dynamic system, allow the use of much more complicated control algorithms than a common feedback PID controller.

One way fo improving the classical control algorithms is a feedforward compensation, which uses an inverse dynamic model of the controlled system as an open-loop controller. Thanks to the inverse model, the feedforward compensator calculates the input of the system that has to be applied to follow a specified trajectory on the output.

This kind of control algorithm is often combined with a classical feedback controller, because unlike the feedforward, any feedback control algorithm can react to the error between the desired and measured output of the system and thus ensure higher precision. The combination of a feedforward and a feedback controller is sometimes called the composite control, for example in [11].

The crucial task, that needs to be dealt with when using such a control algorithm is obtaining the inverse dynamic model of the system. Many different approximation methods can by used for that. Generally, an approximation method is used to replace an unknown nonlinear function, that represents the dynamics of a system, by another (usually simpler) known function in an explicit form, based on the measured input-output data. These methods can be divided into two groups, global and local methods.

Global methods, like polynomial models or sigmoidal neural networks, tend to replace the unknown function by one extensive model. On the other hand, local methods, like look-up tables or radial basis neural networks, use a system of simple basis functions. Each of the basis functions is valid only in the area where it matches the approximated function with sufficient accuracy. [6]

This thesis deals primarily with the local approximation methods. According to [3], the main advantages of this approach are general learning stability, lower computational demands and that the results can be reviewed and understood by a human being.

The basis functions can have various structures. However, it has to be possible to determine, or at least estimate, their parameters according to the measured input-output data. Usually, the parameters of a basis function are estimated using one of the Least Squares method variants [2]. This means, that the simple function, that defines the structure of a basis function has to be linear in parameters [1]. The most common structure of a basis function is a linear combination of the unknown functions input variables, which in the case of inverse dynamics have the meaning of the system states [6].

The inverse dynamic model of a controlled system acquired by local approximation can serve as a very effective feedforward compensator. To be an adaptive controller, the specific approximation method must be able to continuously learn from any newly delivered data.

This brings the second purpose of the feedback controller in the composite control algorithm. Before an approximation of the inverse dynamic model is created, it serves as a pattern for the local approximation algorithm. The feedback controller does not have to be precise, but it has to be stable at least in some small initial area.

In this thesis, some local approximation methods are used to develop an adaptive control algorithm for several automotive actuators, but I assume that the same approach could be used to deal with many other systems.

Besides, it is important to note that the inverse dynamic model is usually a function of several time derivatives of one measured signal, so it can approximate the dynamics of the analysed system. If distance is the measured signal, the other variables could obviously be velocity and acceleration. It brings along a practical difficulties because it is necessary to calculate several time derivatives of a noisy signal.

The purpose of this thesis is not to develop a control algorithm for one specific system, but to explore the possibilities and limitations of the described approaches when controlling various electromechanical systems.

1.1 Possible Applications

The above-described principles of an adaptive control algorithm could find a practical use in situations where the controlled system changes its parameters in time, when individual products have different parameters, or when the controlled system is severely nonlinear.

Due to the fact, that this approach provides the ability to control nonlinear systems, it would not be necessary to emphasise linearity when developing a new product.

Further, this kind of control algorithm could be used in laboratory environment when developing new devices. It is often necessary to develop a control algorithm for various electromechanical systems, which parts and parameters can change during the development process.

Finally, similar approach could be used as an interesting tool to analyse dynamic systems, because it is possible to acquire a relatively precise representation of the analysed system state space without exactly knowing its mathematical model or all physical effects that take place in the system.

2 Theoretical Survey

This chapter contains a short review of the most important algorithms and approximation methods used in this thesis. Its purpose is not to describe and explain every detail of the specific topic and therefore substitute the original articles, but to briefly introduce the reader into the idea of local approximation and its use in control of dynamic systems.

The section 2.1 explains how we can control dynamic systems with the feedforward compensation and describes some problems that come with practical realisation of such an adaptive control algorithm (for example trajectory planning). Especially in cases where we require the algorithm to be able to learn.

The section 2.2 deals with general aspects of local approximation methods, motivation for their use and also describes the most important mathematical algorithms, which are common amongst various approximation methods.

Also, there is a brief description of specific local approximation methods used in this thesis in the section 2.3.

Finally, section 2.4 summarises the most important literature and academic articles, which are related to the content of this thesis and also specifically analyses the official thesis assignment.

2.1 Composite control

A combination of a classical feedback controller (usually a PID or a state space controller) and a feedforward compensation is often known as the composite control algorithm. The compensator can have various structures, for example, a dry or viscous friction compensation, but this thesis focuses solely on dynamic compensation with an inverse dynamic model.

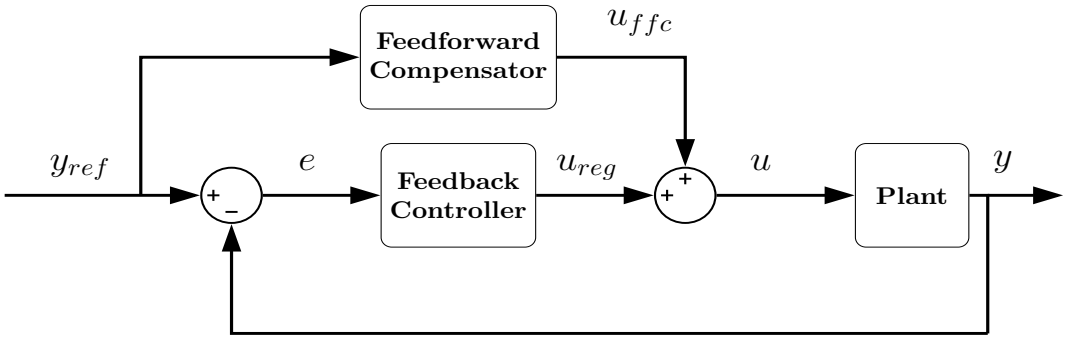

Figure 2.1: Composite control block diagram

The figure 2.1 shows a functional block diagram of the composite control algorithm. The signal $\boldsymbol{y_{ref}}$ is the reference value of the plant output, $\boldsymbol{e}$ is a control error, $\boldsymbol{u_{reg}}$ is the feedback controller output, $\boldsymbol{u_{ffc}}$, $\boldsymbol{u}$ is the combined input of the plant and $\boldsymbol{y}$ is the measured output of the plant.

2.1.1 Feedforward Compensation

The principle of a feedforward dynamic system control is using an approximated inverse dynamic model. As an inverse model, we mean an explicit relation between the outputs and input of the controlled system. The feedforward compensator does not calculate the system input according to the control error, as for example, a PID controlled does, but only according to the reference input.

The theory says that if the inverse model exactly matches the systems behaviour, the control is perfectly accurate [3]. In practice, that is not possible. That is the main reason, why a feedback controller (usually an P controller is sufficient) is still present, because it can deal with the model inaccuracies and ensure the long-term precision and stability.

2.1.2 Inverse dynamics model

When working with SISO systems, the inverse model defines, as was mentioned before, what input should be applied to the system so that its output (and possibly other internal states) follows the reference trajectory. Of course, the model does not have to contain every single aspect of the system's dynamics, just the most important parts.

Since this thesis deals mainly with position control of electromechanical actuators, the inverse model is usually considered in the form of the equation 2.1. The number of independent variables that appear in the model determines the number of dimensions of the controlled system state space. The dependent variable $\boldsymbol{u(t)}$ represents the inverse model output, which is the input of the control system. In a case of electromechanical actuators usually voltage. The input variables $\boldsymbol{x(t), \dot{x}(t), \ddot{x}(t), ...}$ represent the measured output of the system and series of its time derivatives up to the specified order.

$$u(t) = f\left(x(t), \dot{x}(t), \ddot{x}(t), ...\right) \tag{2.1}$$

Sometimes, the inverse model can also be represented in a discrete form, like equation 2.2. Instead of time derivatives, it is a function of the measured system output in different time step. The sequence of the measured samples is represented by the lower index.

$$u_k(t) = f\left(x_k(t), x_{k-1}(t), x_{k-2}, ...\right) \tag{2.2}$$

The function $\boldsymbol{f}$ is generally nonlinear, but often the controlled system behaves in a way, that the nonlinearity concerns just certain system states. The rest, like the inertia, can behave in a linear way and thus be represented in the model by a linear combination. Thanks to this fact, the model can sometimes be represented in the form of the equation 2.3, which can also be written in a discrete form.[1]

$$u(t) = f\left(x(t), \dot{x}(t)\right) + b\ddot{x}(t) + ... \tag{2.3}$$

Finally, when the model is supposed to be linear, it can be represented only as a linear combination of the considered input variables. In the continuous form, it has the form of the equation 2.4.

[1]This simplification has a great practical significance because it can be used to lower the computation demands of a specific (local) approximation method. It is described in detail in the section 3.2.3.

$$u(t) = b_1 x(t) + b_2 \dot{x}(t) + b_3 \ddot{x}(t) + ... \tag{2.4}$$

The question is how to practically realise the function $\boldsymbol{f}$ - what should its model structure look like and how should the model parameters be acquired? As was explained before, local approximation methods are used in this thesis to create the model. The description of these methods is summarised in the following sections 2.2 and 2.3.

It is important to note, that both the continuous and discrete form of the inverse model are practically realised with a discrete period. The difference between them is the representation of the controlled system dynamics.

In the discrete case, it is represented by a series of samples of the same physical quantity, measured in different time steps. In the continuous case, the series of time derivatives is acquired in the same time step. From the theoretical point of view, there should be no difference between the dynamics representation in both cases, but a very different tolerance for a noise in the measured signal is expected, especially its influence on the parameters estimation.

The continuous case also brings a problem with determining the derivatives of a noisy signal. Especially when determining derivatives of higher order (second or third), the noise could cause the parameter estimation to fail.

For a specific implementation of an inverse dynamic model for control purposes, a so-called **query point** has to be defined. It is a point in the system state space, where the system input should be calculated. During the control process, usually the current state of the planned trajectory is taken as the query point. If an inverse dynamic model is defined as equation 2.1, the query point would be defined as a specific point in a space of input variables, and substituted into the model, according to the equation 2.5.[3]

$$\boldsymbol{X}_q = [x_q, \dot{x}_q, \ddot{x}_q, ...] \quad \Rightarrow \quad u_q = f\left(x_q, \dot{x}_q, \ddot{x}_q, ...\right) \tag{2.5}$$

2.1.3 Trajectory Planning

An important feature, that has to be used when implementing a feedforward compensation is the trajectory planning. In this case, the trajectory is defined as a path through the system state space. If an inverse model is represented by the equation 2.4, it is not possible to use a reference signal in the form of discrete steps, as is possible with a PID controller. The discontinuities in the reference signal would cause its derivatives to be infinite or undefined. These cannot be realised by an actual actuator.

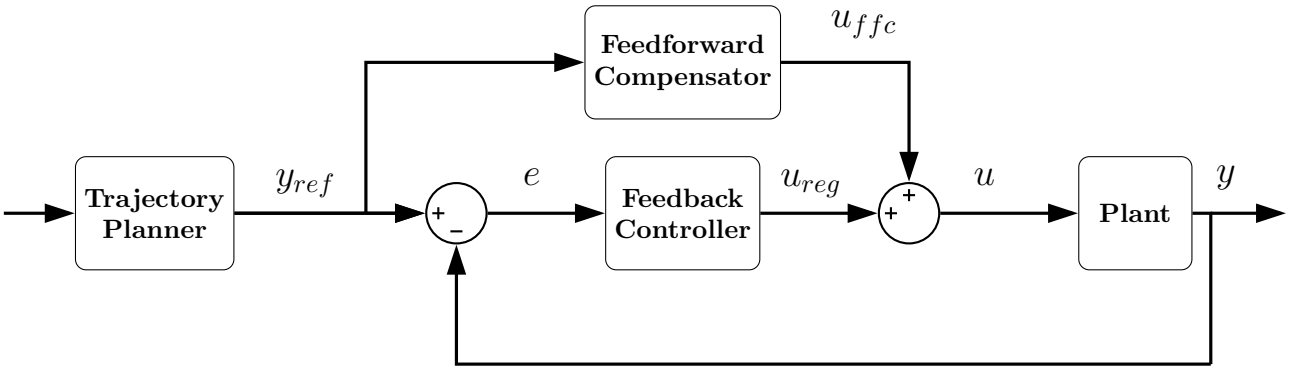

Figure 2.2: Composite control with trajectory planning block diagram

The reference signal has to follow two conditions: the transient of the signal has to be feasible for the real system, and it has to be possible to calculate a time derivative of the signal up to the order specified by the model structure. The figure 2.2 shows the block diagram of a complete composite control algorithm with trajectory planning.

An easy way to meet both conditions is to use a lowpass filter of the order specified by the highest order derivative present in the inverse model. [14]

2.2 General Aspects of Local Approximation

As mentioned in the section 2.1.2, the local approximation is based on dividing a nonlinear function into smaller areas, where it can be replaced by a simpler (possibly linear) basis function. Every specific local approximation method deals with two main, partially separated problems. The first is estimating parameters of the basis functions and the second is determining the area of validity for each of them.

The problem of estimating the parameters of a local model is usually solved by one of the Least Squares method variants, which are described in the section 2.2.1. However, the problem of local models distribution is is much more complicated and every specific approximation method deals with it in a different way. These methods are summarised in the section 2.3.

An example of local approximation used to control a dynamic system can be a case where an inverse dynamic model of the system can be written in the form of the equation 2.1. In this case, the model is divided into smaller areas where it is replaced by local model with a form of the equation 2.4.

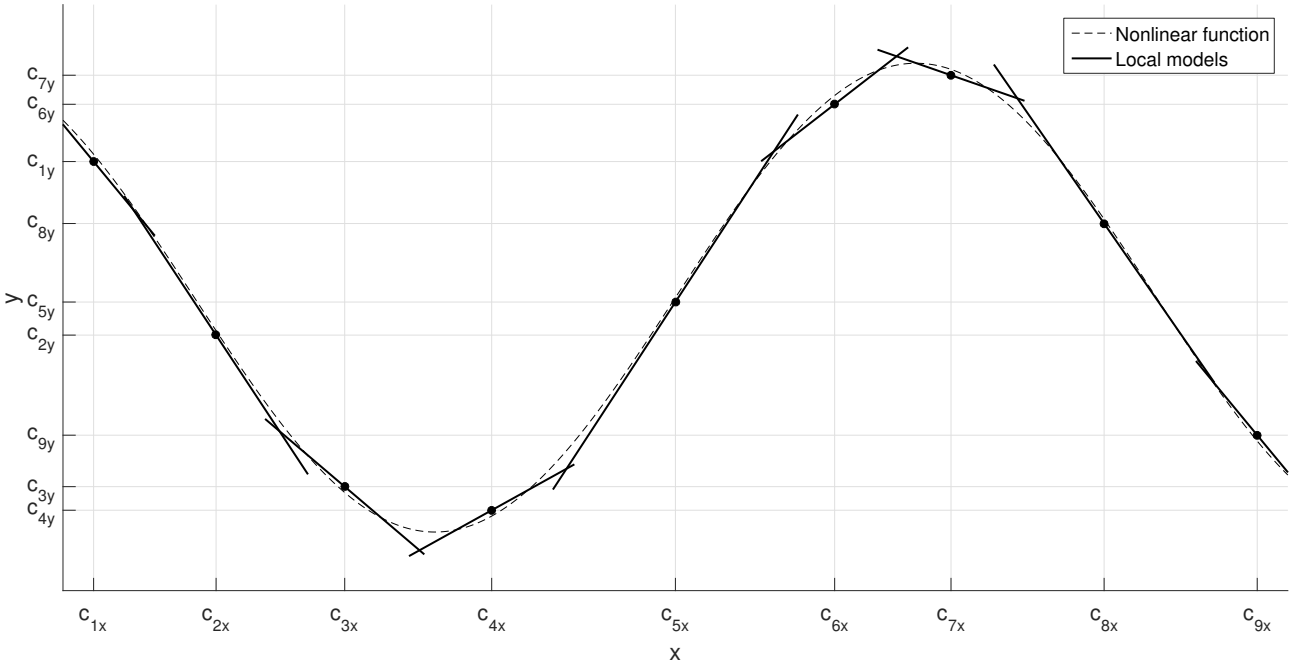

Figure 2.3: Example of local linear approximation

The figure 2.3 shows an example of local approximation of a nonlinear function by linear models. This is a one-dimensional case, but the whole idea can be easily scaled to an arbitrary number of dimensions.

Because a real-time implementation of these methods is expected, it is important to mention,

that the computational power needed to process them grows exponentially with the number of dimensions, which is determined by the number of the input variables of an inverse model. [5]. Even though the local approximation methods are defined in any number of dimensions, in the case of electromechanical actuators it is expected that the number of the input variables will not be higher than 5.

2.2.1 Least Squares Optimisation

The least squares method (in other chapters referred as LS) is one of the most common tools for linear parameter estimation, detailed description and inference can be found in [1]. In this thesis, it serves as a method for estimating parameters of the local models. If we have a model according the equation 2.6, where $\hat{\boldsymbol{y}}$ je is estimated model output, $\boldsymbol{b} = [\boldsymbol{b_1}, \boldsymbol{b_2}, ..., \boldsymbol{b_n}]^{\boldsymbol{T}}$ are model parameters and $\boldsymbol{X} = [\boldsymbol{x_1}, \boldsymbol{x_2}, ..., \boldsymbol{x_n}]$ are model inputs,

$$\hat{y} = \boldsymbol{X}\boldsymbol{b} \tag{2.6}$$

the best estimation of the parameters, based on a series of measured input-output data, can be calculated according to the equation 2.7. [1]

$$\boldsymbol{b} = \left(\underline{\boldsymbol{X}}\boldsymbol{X}^T\right)^{-1}\underline{\boldsymbol{X}}\boldsymbol{Y} \tag{2.7}$$

Where $\boldsymbol{Y} = [\boldsymbol{y_1}, \boldsymbol{y_2}, ..., \boldsymbol{y_m}]^{\boldsymbol{T}}$ and $\underline{\boldsymbol{X}} = \left[\boldsymbol{X_1^T}, \boldsymbol{X_2^T}, ..., \boldsymbol{X_m^T}\right]^{\boldsymbol{T}}$. Every row of $\boldsymbol{Y}$ and $\underline{\boldsymbol{X}}$ corresponds to an input-output pattern. $\boldsymbol{m}$ is a number of patterns $\boldsymbol{n}$ is a number of model parameters. The best estimate is such that has the least mean squared error considering the input-output patterns.

Sometimes, it is useful to assign each pattern a weight, according to its importance. The equation with weighted patterns looks like 2.8. The diagonal elements of the matrix $\boldsymbol{W} = \boldsymbol{diag}\,[\boldsymbol{w_1}, \boldsymbol{w_2}, ...\boldsymbol{w_m}]$, that appears in the equation, correspond to the weight of each pattern.

$$\boldsymbol{b} = \left(\underline{\boldsymbol{X}}\boldsymbol{W}\underline{\boldsymbol{X}}^T\right)^{-1}\underline{\boldsymbol{X}}\boldsymbol{W}\boldsymbol{Y} \tag{2.8}$$

The least squares method according to the equations 2.7 or 2.8 serves to the parameter estimation in the case where there is a batch of measured patterns available. It is usually used for off-line estimation. The computational power demands of this method depend on the number of patterns ans thanks to the matrix inversion present in both equations, it grows very quickly. [2]

There is also an incremental version of this method, which is called the Recursive Least Squares method. Detailed description and inference can be found in [1] and [6]. The estimate of the parameters is updated step by step with new patterns, which allows the method to be used for on-line estimation. When presented with the same patterns, both methods should end up with similar results.

This recursive approach demands that except the best estimate acquired so far, an estimate of a covariance matrix also has to be saved for further estimation.[3] Both estimates are incrementally updated with every new pattern. The equation 2.9 describes the update of the

[2]Sometimes, the matrix inversion can be replaced by the Moore-Penrose pseudoinverse, which can lower the computational complexity. Sometimes at the cost of precision. [1]

[3]More precisely, it is the inversion of a covariance matrix as described in [4] and [5].

covariance matrix. The estimated parameters are then updated according to the equation 2.10, where e_{cv} is the mode deviation from the actual pattern, which is acquired from the equation 2.11.

$$\boldsymbol{P}^{n+1} = \frac{1}{\lambda}\left(\boldsymbol{P}^n - \frac{\boldsymbol{P}^n\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{P}^n}{\frac{\lambda}{w} + \boldsymbol{X}\boldsymbol{P}^n\boldsymbol{X}^T}\right) \tag{2.9}$$

$$\boldsymbol{b}^{n+1} = \boldsymbol{b}^n + w\boldsymbol{P}^{n+1}\boldsymbol{X}^T e_{cv} \tag{2.10}$$

$$e_{cv} = y - \boldsymbol{X}\boldsymbol{b} \tag{2.11}$$

There are several other variables in the equations: the parameter $\boldsymbol{\lambda} \in \langle \mathbf{0;1} \rangle$, which represents the forgetting rate of the estimated covariance matrix [4], and the pattern weight $\boldsymbol{w} \in \langle \mathbf{0;1} \rangle$, which has the same meaning as the diagonal matrix $\boldsymbol{W}$ elements in the equation 2.8.

2.2.2 Gaussian and Local Approximation

To determine the weight of each pattern for a local model parameter estimation, a so-called kernel function is used, as described in [5],[3] and [2]. A pattern weight is calculated according to the distance between the local model validity area centre and the pattern in the space of the inverse dynamic model input variables, as a value of the kernel function for this distance.

Several different functions can be used as a kernel function (see [1], [5] or [15]), but the most common is the Gaussian function according to the equation 2.12 which shows the one-dimensional Gaussian function. For general number of input dimensions the Gaussian function is defined by the equation 2.13,

$$f(x) = ae^{-\frac{(\boldsymbol{x}-\boldsymbol{c})^2}{2\boldsymbol{C}}} \tag{2.12}$$

$$f(\boldsymbol{x}) = ae^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{c})\boldsymbol{C}^{-1}(\boldsymbol{x}-\boldsymbol{c})^T} \tag{2.13}$$

where $\boldsymbol{c}$ is the Gaussian function centre position (in the case of calculating the weight, it is the position of the local model validity area centre), $\boldsymbol{x}$ is independent variable (in the case of calculating the weight, it is the position of a pattern), $\boldsymbol{C}$ is a distance matrix, and $\boldsymbol{a}$ is the maximum value of a Gaussian function when $\boldsymbol{x} = \boldsymbol{c}$.

The reason for using the Gaussian function is, that it is basically an exponential function, which means it can be easily analytically differentiated and has an infinite domain.

The figure 2.4 shows the shape of a Gaussian function in one and to dimensions. In one-dimensional case, the parameter $\boldsymbol{C}$ is just a scalar, that represents the square of the distance from the function centre, where it has the value of approximately $\mathbf{0.6065}$.

In a two-dimensional case, the parameter $\boldsymbol{C}$ represents the same distance in every direction. The curve, that meets the criteria $(\boldsymbol{x}-\boldsymbol{c})\,\boldsymbol{C}^{-1}\,(\boldsymbol{x}-\boldsymbol{c})^{\boldsymbol{T}} = \mathbf{1}$ has a shape of an ellipse, which is highlighted in the figure. In a three-dimensional case, it would be an ellipsoid and so on.

The equation 2.14 suggests the conversion between the elements of the distance matrix $\boldsymbol{C}$

[4] Usually $\boldsymbol{\lambda} = \mathbf{0.99..}$ according to [5].

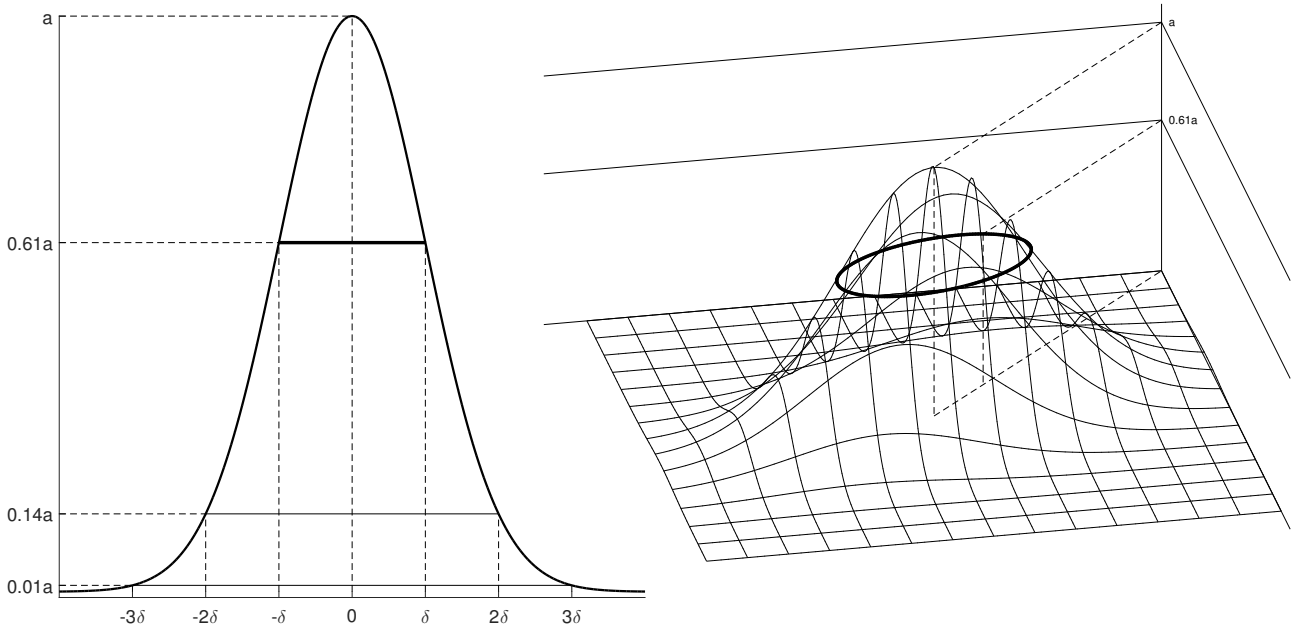

Figure 2.4: Example of 1D and 2D Gaussian function

and a standard ellipse equation, in a case, when $\boldsymbol{c}$ is a zero vector. [16]

The representation of a distance matrix (or a whole Gaussian function) by an ellipse is very effective for 2D visualisation of several Gaussian function in a plain and it is a useful tool for working with them.

$$\boldsymbol{x}\boldsymbol{C}\boldsymbol{x}^T = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} \delta_1^2 & \delta_{12} \\ \delta_{12} & \delta_2^2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \delta_1^2 x_1^2 + 2\delta_{12} x_1 x_2 + \delta_2^2 x_2^2 = 1 \tag{2.14}$$

It is also known, that a one-dimensional Gaussian function for $\boldsymbol{a} = \frac{\mathbf{1}}{\sqrt{\mathbf{2\pi c}}}$ represents a probability density function of the Normal distribution. In that case, the matrix $\boldsymbol{C}$ is called the covariance matrix, and its corresponding ellipse the error or covariance ellipse. In a one-dimensional case, the $\boldsymbol{C}$ scalar represents the variance of the random process.

Figure 2.5 shows the visualisation of several different Gaussian functions. If the distance matrix is diagonal, the ellipse axis of symmetry are parallel with the space axis. Nonzero elements outside the main diagonal cause the ellipse to rotate and stretch. In both cases, the distance matrix has to be positive definite.

2.3 Specific Approximation Methods

As was mentioned in the section 2.2, the basic local approximation methods are very similar, when it comes to the parameter estimation of each local model. However, they use very different approaches, when it comes to the model placement problem. That is optimising the area of validity for each local model, adding new models or deleting the redundant ones, when necessary. Some of the methods also keep in their memory a list of the patterns and create a local model just for a specific query point, others use new patterns just to optimise a system of local models.

The table 2.1 contains a list of some selected methods and their most important features. It is also important to note, that all of these methods can be used for real-time control as well as perform their learning simultaneously.

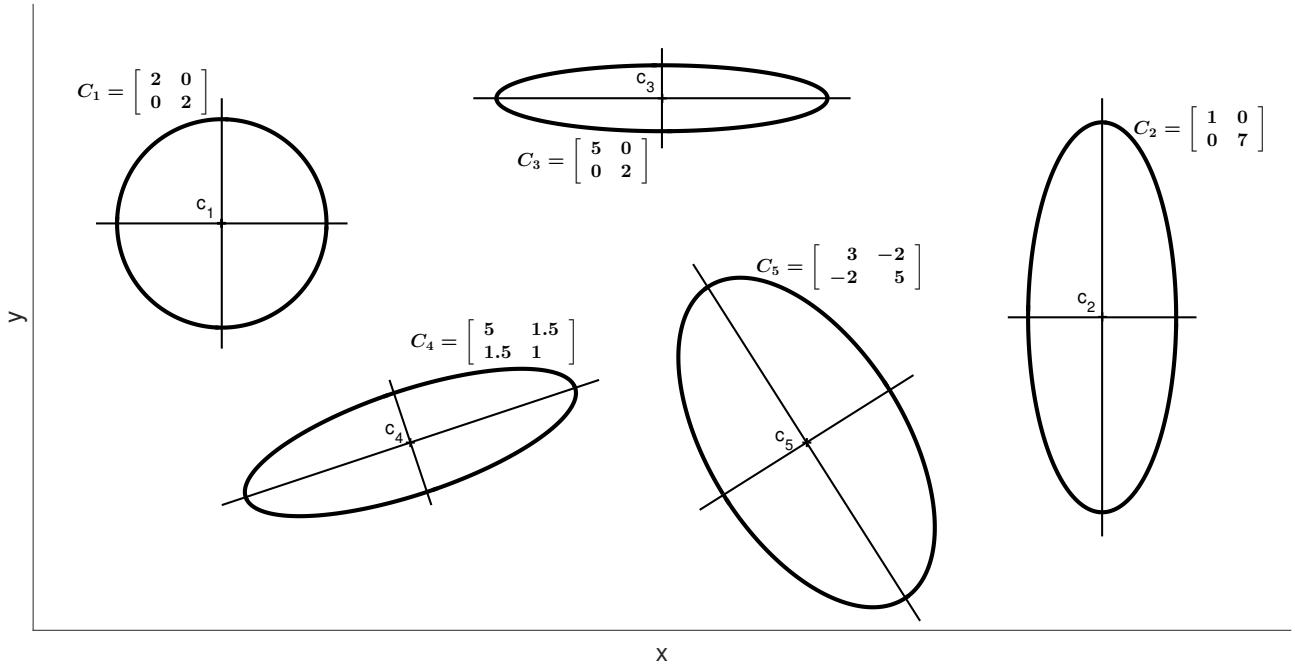

Figure 2.5: Visualisation of different Gaussian functions using ellipses

Except the LWPR[5] method (Locally weighted Projection Regression), which is adapted for dealing with high-dimensional problems, these methods are implemented, analysed and subjected to experiments.

Method Name	**Learning algorithm**	**Memory type**	**Model Placement algorithm**
LUT	weighted average	model based	various
LWL	LS	memory based	none
RFWR	RLS	model based	gradient search
LOLIMOT	RLS	model based	heuristic
LWPR	PLS	model based	gradient search

Table 2.1: Summary of local approximation methods

2.3.1 Grid based Look-Up Table (LUT)

A look-up table is the simplest method, because the local models degenerate to single points. Its advantage is, that the values of these points can be optimised by much simpler and more stable methods. For example, a weighted average of a table value and a new pattern can be used. Also, new points can be added to the table grid, if necessary.

If a query point does not exactly match the specified table point position, its value can be calculated as linear (or higher polynomial) interpolation. The same approach can be used when

[5]This method is based on the RFWR algorithm, but instead of the RLS, it uses the Partial Least Square Method (PLS), which is able to determine and ignore input dimensions with no useful information. [13]

a new pattern lies between several table points. It can be matched with an interpolated value based on those table points, and then the deviation is used to optimise the table values.

2.3.2 Locally Weighted Learning (LWL)

The LWL method is a memory based method. That means it stores a list of the measured patterns (unimportant ones can be forgotten). The output for any query point is determined by a local linear model which centre of the area of validity matches the query point. Parameters of the model are estimated by the weighted LS method (equation 2.8). Weights for each pattern are calculated according to a kernel function (usually Gaussian) as described in the section 2.2.2.

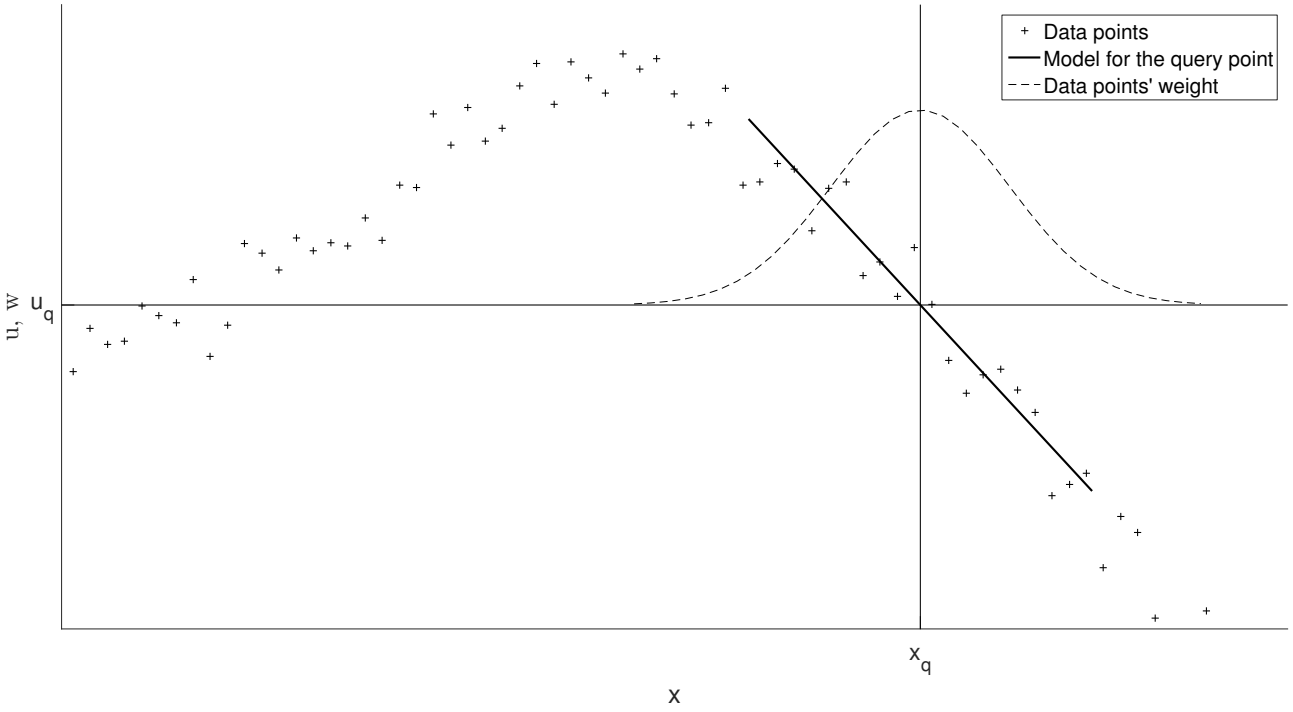

Figure 2.6: Example of Locally Weighted Learning

Tuning of this method is based on determining the right distance matrix of the kernel function. It has to cover enough points for estimating the model parameters correctly, but must not contain regions that do not apply to the linear model. An example of local linear approximation by a cloud of points is in the figure 2.6. Further description of the LWL method can be found in [2] and [3].

2.3.3 Receptive Field Weighted Regression (RFWR)

The RFWR method was created as an improvement of the LWL aiming to decrease the computational complexity. The complete inference can be found in [6].

Unlike LWL, this method saves a number of local models that approximates the measured data. New patterns are used just to search for the best areas of validity for each model using a gradient search method, and to optimise the models parameters using the RLS method described in the section 2.2.1. Specifically according to the equations 2.9, 2.10 and 2.11.

A new pattern weight is also determined using a Gaussian kernel, which is unique for

each local model, which also represents its area of validity. Together, the local model and its kernel function form a so-called receptive field. The distance matrix of the kernel function is subject to the gradient optimization aiming to find the largest area of validity possible that still approximates the measured data precisely enough.

Due to simplification, instead of the distance matrix itself, its inversion $\boldsymbol{D}$ is saved and optimized. This leads to a Gaussian kernel function according to the equation 2.15,

$$f(x) = e^{-0.5\boldsymbol{v_e}\boldsymbol{D}\boldsymbol{v_e}^T} \tag{2.15}$$

where

$$\boldsymbol{v_e} = \boldsymbol{x} - \boldsymbol{c} \tag{2.16}$$

Since the distance matrix $\boldsymbol{C}$ has to be symmetrical according to its main diagonal and positive definite, its inversion $\boldsymbol{D}$ is, too. Therefore, the matrix $\boldsymbol{D}$ can be decomposed according to the equation 2.17 with a matrix $\boldsymbol{M}$, which is also an upper triangular matrix. [6]

$$\boldsymbol{C}^{-1} = \boldsymbol{D} = \boldsymbol{M}^T\boldsymbol{M} \tag{2.17}$$

With every new pattern, the matrix M is updated by its gradient in search of the maximum area of validity that meets the required precision, according to the equation 2.18, where J represents a so-called cost function, which statistically enumerates the precision criteria for the model deviation from the measured data. So basically, it is a gradient search for the J function minimum.

$$\boldsymbol{M}^{n+1} = \boldsymbol{M}^n - \alpha\frac{\partial J}{\partial \boldsymbol{M}} \tag{2.18}$$

In a case where a new pattern has lower weight, that a set limit (usually $w_{gen} = 0.01$) for every local model, a new model is created on the specified place. On the contrary, when two local models share an area with the weight larger than a set limit (usually $w_{prune} = 0.7$), the smaller one is pruned. This approach iteratively leads to an improved distribution of the local models.

The figure 2.7 shows an example of local linear approximation with the RFWR method. As expected, the models are distributed in a way, that there are smaller models in the area with an important nonlinearity and the rest is covered with a lower number of wider models.

$$u = \frac{\sum_{i=1}^{N} w_i u_i}{\sum_{i=1}^{N} w_i} \tag{2.19}$$

When calculating the output for a specified query point, it is determined according to the equation 2.19, where $\boldsymbol{u_i}$ is the i-th local model output for the specified query point and $\boldsymbol{w_i}$ is its weight, using the same kernel function as for the patterns weights. This equation represents a weighted average of all local models. For simplification, only the most important local models are usually activated and used for the output calculation. Those who's weight is smaller than $\boldsymbol{w_{act}}$ are ignored.

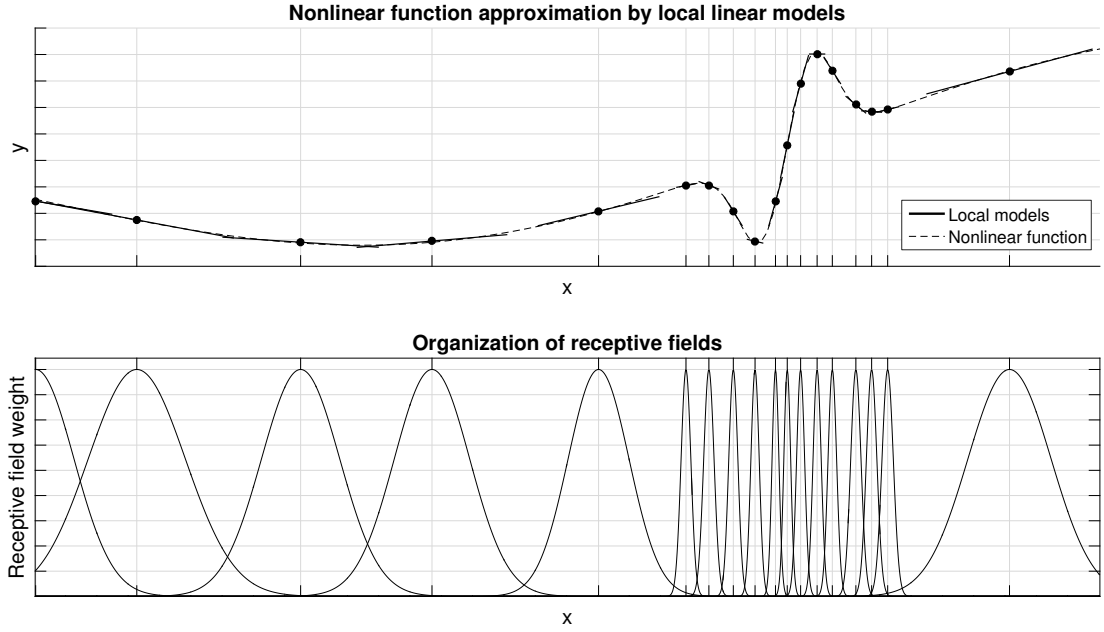

Figure 2.7: Example of Receptive Field Weighted Regression

2.3.4 Local Linear Model Tree (LOLIMOT)

The LOLIMOT method is very similar to RFWR, but they use a different approach for adding new models. At the beginning, there is only one local model covering the state space. As new patterns are acquired, its parameters are adjusted with the RLS method and it can be divided into halves along one of its axis of symmetry. The algorithm chooses such an axis of symmetry, which creates a pair of local models with lower deviation from the measured data.

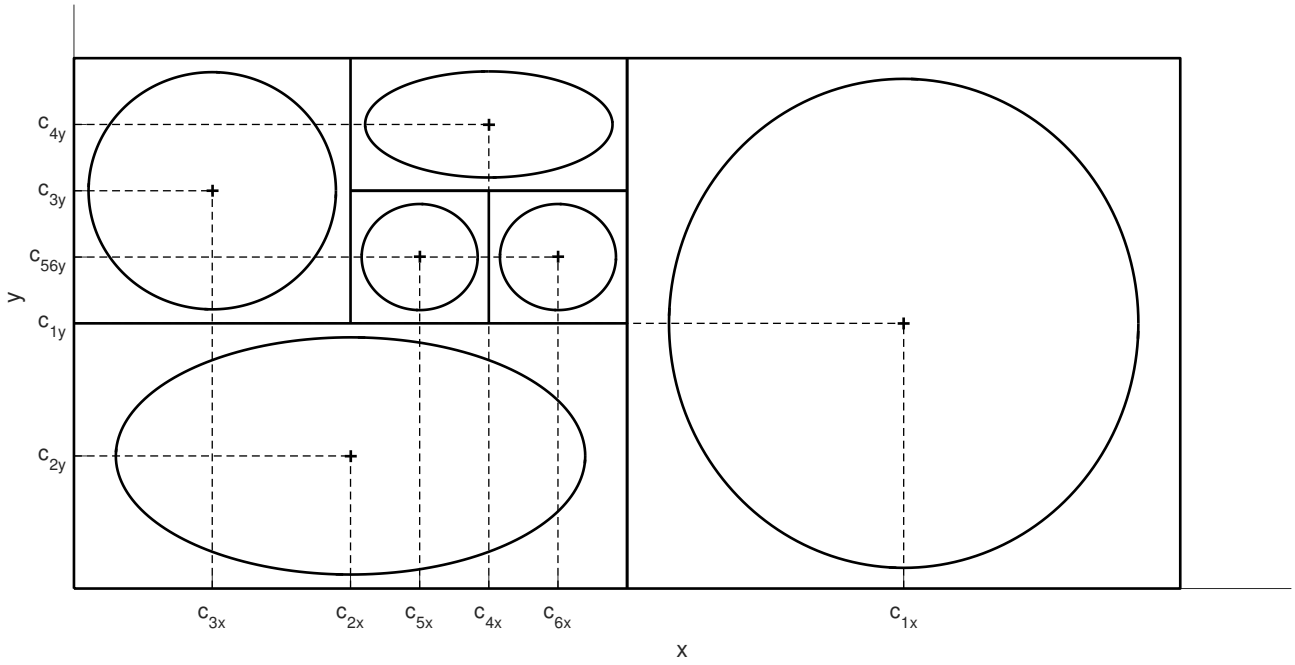

Figure 2.8: Example of Local Linear Model Tree

The models are not pruned nor are new local models are added. The number of models increases simply by dividing the already existing ones. This means that a local models area of validity is represented also by an orthogonal grid of borders, not only by its Gaussian kernel function. Figure 2.8 shows an example of several divided models in a two-dimensional case. Further details about this method and its different variants can be found in [1].

2.4 Important Literature

The following list of brief descriptions covers the most important articles that explain in detail all theoretical aspects concerning local approximation methods. Especially those from C.G. Atkeson, S. Schaal. This thesis tries to follow their terminology, sign and symbols, so that it would be easier to find a further description of some specific parts of this thesis in the original articles.

- **Nelles O. : Nonlinear system identification** [1]
  An extensive book covering both linear and nonlinear system identification. It contains detailed description and inference of the Least Squares methods, look-up table interpolations, the LOLIMOT method, fuzzy systems, artificial neural networks, and much more.

- **Atkeson C.G. : Locally Weighted Learning** [2]
  The article covers Local weighted regression and learning, and its use for nonlinear systems approximation.

- **Atkeson C.G. : Locally Weighted Learning in Control** [3]
  This article builds on the previous one and covers using the LWL method for dynamic system control by approximating both inverse and forward dynamics.

- **Schaal S. : Constructive Incremental Learning from Only Local Information** [6]
  Uses experience with the LWL method and introduces the new, incremental RFWR method along with some simulations and experiments.

- **Vijayakumar S. : Incremental Online Learning in High Dimensions** [13]
  This article contains an improvement of the RFWR method for high-dimensional problems using PLS instead of RLS, introducing the LWPR method.

- **Birattari M. : The Lazy Learning Toolbox** [7]
  The article contains many detailed descriptions of local approximation methods as well as documentation for a lazy learning toolbox for Matlab.

2.4.1 Thesis Objectives

The goal of this thesis is to examine possibilities of local approximation methods, especially in control of dynamic systems. Both general algorithms used for local approximation described in the section 2.2 and the specific methods described in the section 2.3 will be subjected to simulations and experiments with position control of automotive actuators. The particular objectives of this thesis can be summarised as follows:

- Choose methods that have the potential for real implementation in on-line learning and control. Not precision but stability is the key feature that is required because the precision of the position control can be significantly improved by a regular PID or State Space controller.
- Analyse and simulate the behaviour of the selected methods under real conditions, especially signal noise.
- Test the most important principles which are general to various local approximation methods, mainly the parameter estimation, during an experiment.
- Develop an adaptive control algorithm for nonlinear systems using the local approximation methods, capable of managing a position control of specified automotive actuators, described in the section 4.1. After connection to the controlled system, the developed algorithm must be able to learn the inverse model approximation in a relatively short period of time (about tens to lower hundreds of seconds) and use it to control the system.
- Experimentally test the stability and overall behaviour of the developed system with the specified actuators using a real-time control hardware available in the Mechatronics Laboratory (I/O card Humusoft MF624, a dSpace board).
- Try to implement the algorithm, at least in some simplified form, at a dsPIC microcontroller board, using automatic code generation from the Matlab/Simulink environment.

3 Analysis of Local Approximators

The content of this chapter is an analysis, adjustments and simulations of the approximation methods summarized in chapter 2. The adjustments in the methods were made mainly to improve the computational complexity and to make on-line learning possible.

Sections 3.1, 3.2 and 3.3 describe the specific algorithms and methods implemented in the Matlab environments, along with their simulations.

The section 3.4 deals with practical problems when using the recursive variant of the LS method described in the section 2.2.1 in real experiments. It also contains a description of an adaptive composite controller for a DC motor as a result of the final experiment.

3.1 LWL

The basic principles of the LWL method (Locally Weighted Learning) are described in the section 2.3.2. This method is much simpler than the following ones, because it saves only the measured points for later model creation. The method is very effective, although there is a problem. We do not know, how many patterns are needed to create a correct model, which means we do not know how to set the distance matrix of the kernel function.

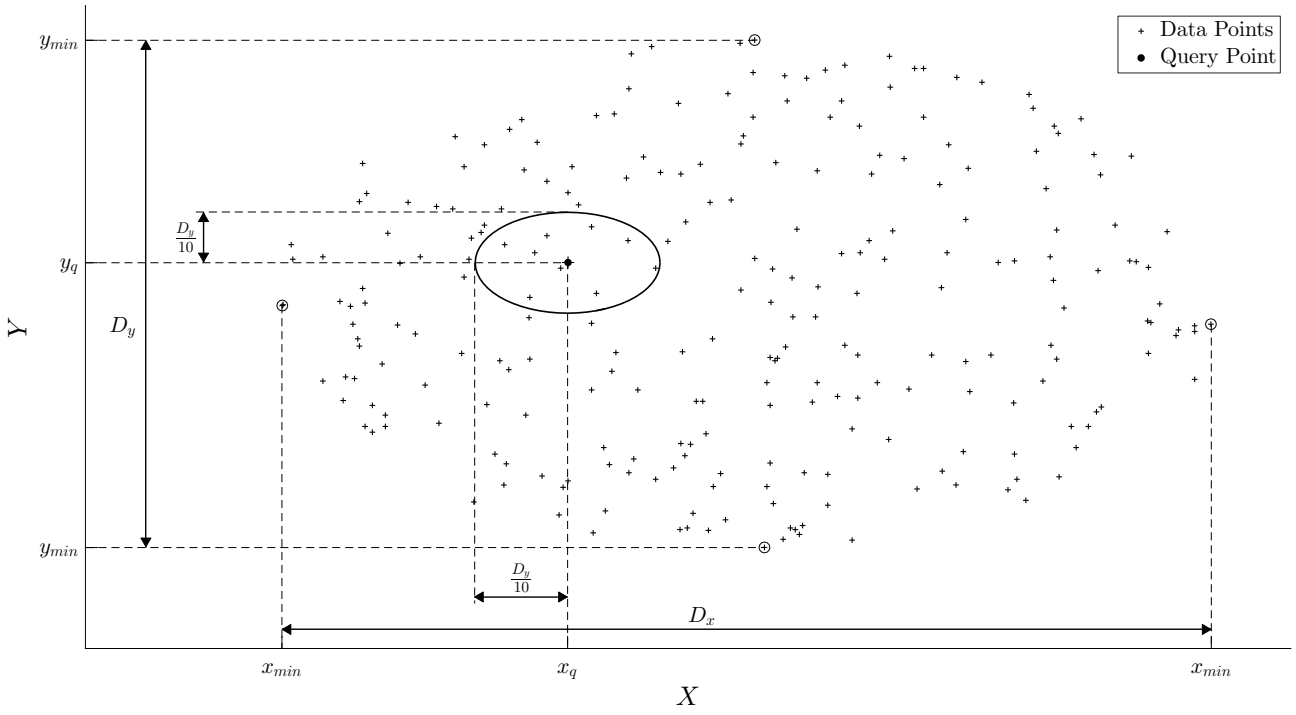

Figure 3.1: LWL distance matrics solution example

Also, it is advantageous to reduce the computational complexity of the algorithm by decreas-

ing the number of patterns saved in memory. For the actual implementation, a straightforward algorithm based on combining close or identical patterns was used.

Another question is, how to determine the correct distance matrix of the kernel function. There are some iterative approaches to find it characterized in [2] and [3], but they are not suitable for real-time computation, since they are not deterministic.

A different, much simpler solution can be used assuming the Gaussian kernel function shape is represented by an ellipse, whose axes are orthogonal to the state space axes. That means the distance matrix is diagonal.

The length of each ellipse's axis, which is represented by the corresponding distance matrix diagonal element, can be determined as a specified fraction of a maximal distance between two pattern coordinates on the state space axis. An example of this situation is expressed in the figure 3.1 assuming the specified fraction to be a tenth of the maximal distance.

The figure 3.2 shows a simulation of a composite control algorithm applied to control a nonlinear system (see section 2.1 for further details). The system is described by the equation 3.1, which represents a torque driven rotation of an inertia $\boldsymbol{J}$ with a viscous friction $\boldsymbol{b(\omega(t))}$ described by a nonlinear function 3.2. The system was simulated using the Simulink environment and a white noise was added to both input torque $\boldsymbol{m(t)}$ and output angular speed $\boldsymbol{\omega}$ of the system.

$$J\frac{d\omega(t)}{dt} + b(\omega(t)) = m(t) \tag{3.1}$$

$$b(\omega(t)) = 20\sin\left(\frac{\omega(t)}{200}\right) + 15\sin\left(\frac{\omega(t)}{20}\right) + 3 \tag{3.2}$$

The nonlinear function 3.2 does not represent any real friction behaviour, it was used solely to test the systems control precision and if the noise introduced to the output signal would influence the system stability.

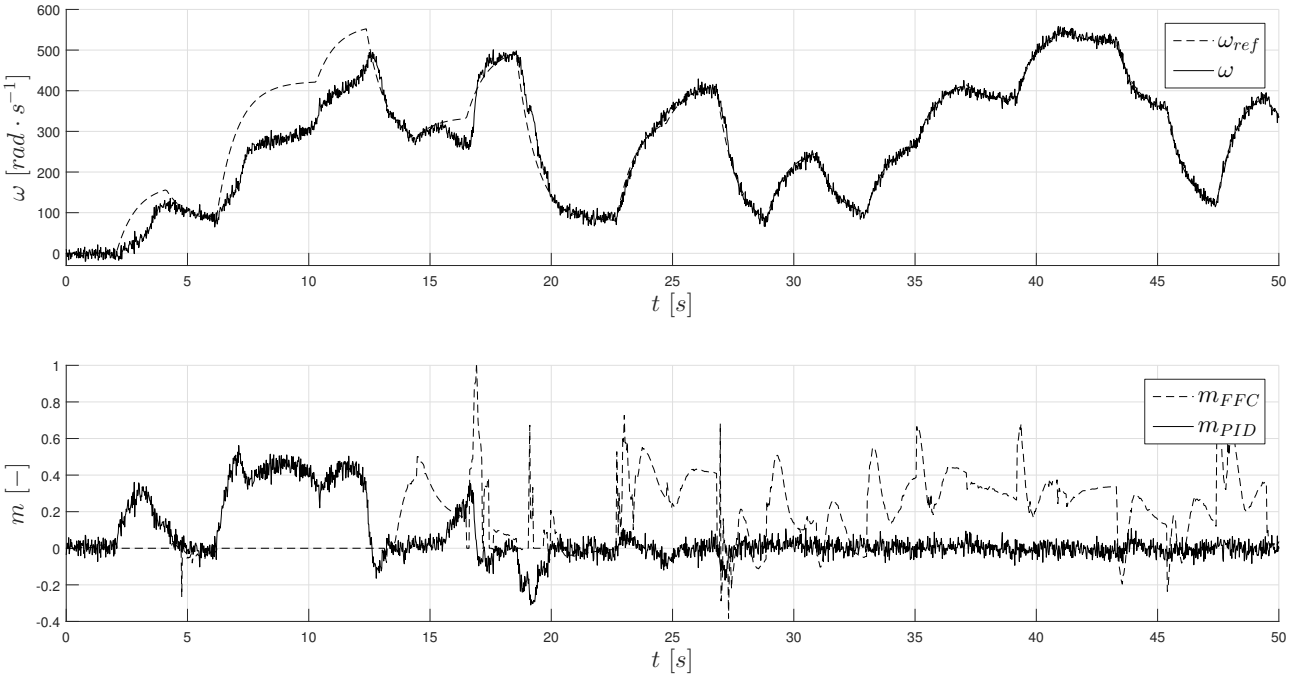

Figure 3.2: Dynamic system control with LWL

The local model used by the LWL method had the structure of the equation 3.3 and its parameters were estimated by the LS method.

$$m(t) = b_1\omega(t) + b_2\frac{d\omega(t)}{dt} + b_3 \tag{3.3}$$

The number of patterns saved for the LWL approximation was limited to 5000, and the kernel function distance matrix was calculated according to the equation 3.4, where $\boldsymbol{d_{wmax}}$ is the maximum distance between two patterns along the $\boldsymbol{w}$ axis and $\boldsymbol{d_{dwmax}}$ is the maximum distance between two patterns along the $\boldsymbol{dw}$ axis.

$$C = \begin{bmatrix} \frac{d_{wmax}}{50} & 0 \\ 0 & \frac{d_{wmax}}{5} \end{bmatrix} \tag{3.4}$$

According to the results shown in the figure 3.2, the feedforward approximation took about 20 seconds to measure enough data to precisely control the system. After that, the feedback P controller is almost unnecessary.

3.2 RFWR

The RFWR method (Receptive Field Weighted Regression) is more complex than LWL, as described in the section 3.2. This approximation method saves a system of local models that best matches earlier measured pattern data.

For a specific implementation of this method, four basic functions must be ensured:

1. Optimisation of local modes parameters.

2. Optimisation of distance matrices of each local model.

3. Adding new local models.

4. Deleting unnecessary local models.

Function number 1 is most easily solved by the RLS method.

Function number 2 can be dealt with by various approaches described, for example, in [6] and [13], which are based on gradient optimisation. The section 3.2.2 describes a partially different approach to find the best possible distribution of local models, that uses a set of heuristic rules instead of a cost function. Especially in low-dimensional cases, it showed to be a very effective, stable and also less computationally complex.

Function number 3, that means adding a new local model, is used when no other model has weight higher than $\boldsymbol{w_{gen}}$ for a new pattern. In such a case, a new local models validity area centre is placed exactly on the patterns position and its parameters are initialised as zeros or random values. The parameter estimation covariance matrix is initialised with large variances.

Determining the kernel function distance matrix, on the other hand, is much more complicated. The section 3.2.1 describes how to calculate the new kernel size according to the nearby existing local models. Eventually, the function number 4 is solved in a the same way as in the original algorithm. If two local models are weighted higher than a set limit at a single point it the state space, the smaller one is deleted.

3.2.1 Adding New Local Models

In the original RFWR variant described in [6], the distance matrix of a new local model is initialised as a diagonal matrix with a user defined values as its diagonal elements. This section tries to describe a new approach, where the diagonal elements of the matrix are estimated according to the distance and shape of the nearest local model to the new local models position. Since the new local model centre is exactly determined by the new pattern, its distance matrix can be acquired as the minimal distance between the new centre and another point in the state space, which has the weight with a local model equal to a user set value $\boldsymbol{w_{max}} = \mathbf{0.3}$. The parameter $\boldsymbol{w_{max}}$ determines the maximal overlapse between the new and any already existing model.

Given the maximal overlapse value and the new local model centre, the size of the new area of validity can be exactly calculated if we assume that it will not be an ellipse, but a circle.

The figure 3.3 shows a scheme of the situation in a two-dimensional case. The vector $\boldsymbol{v_e} = \boldsymbol{c_i} - \boldsymbol{c_I}$ represents a direction vector from the new centre $\boldsymbol{c_I}$ to a nearby local model centre $\boldsymbol{c_i}$, $V = |\boldsymbol{v_e}|$ is the distance between both centres and $\boldsymbol{v_n} = \frac{\boldsymbol{v_e}}{V}$ is a unity vector of the vector $\boldsymbol{v_e}$.

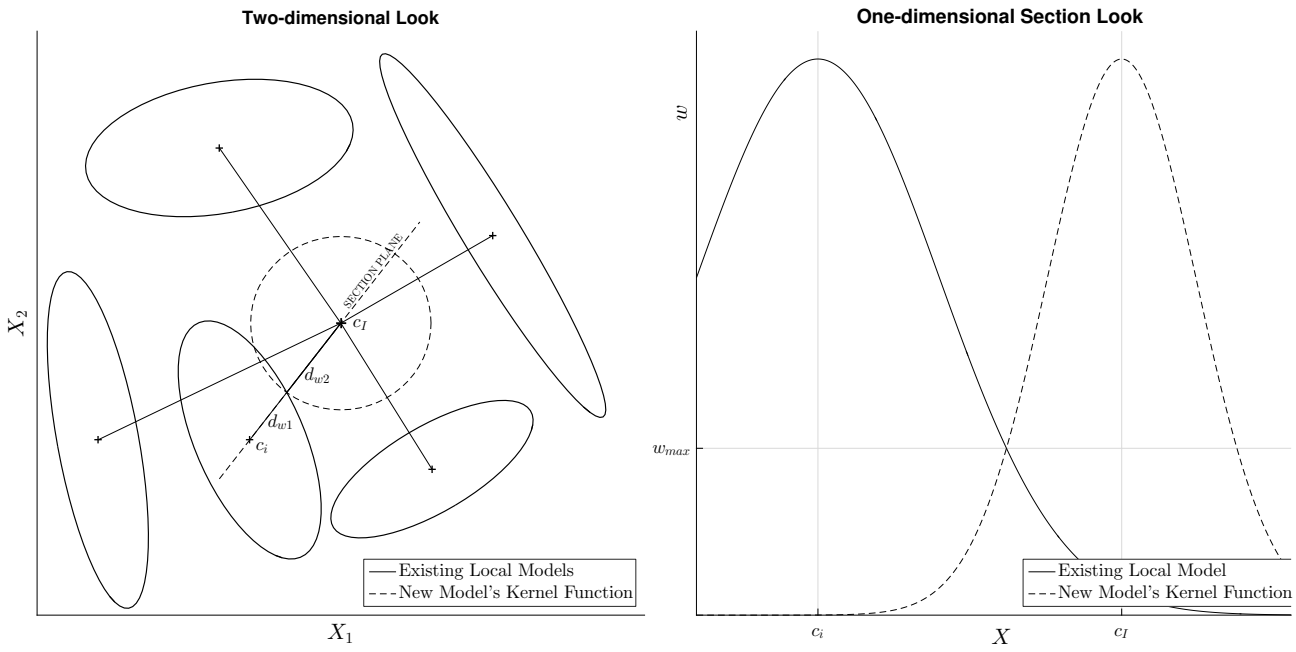

Figure 3.3: Adding a new local model

In the second part of the figure, the whole situation is visualised as simplifying view, which is created as a section through the original figure according to the vector $\boldsymbol{v_e}$. The existing models two-dimensional Gaussian kernel function is represented as a Gaussian curve with a size parameter d, which is an intersection of the original figure the section plain.

The parameter d can be calculated according to the equation 3.5, where $\boldsymbol{D}$ is the original kernel functions distance matrix.

$$d = \boldsymbol{v_n} \boldsymbol{D} \boldsymbol{v_n}^T \tag{3.5}$$

Then, according to the equation 3.6, we can determine the distance d_{w1} in the direction of

the vector $\boldsymbol{v_e}$ from the existing model centre $\boldsymbol{c_i}$, where its weight is exactly w_{max},

$$d_{w1} = \sqrt{-\frac{2}{d}\ln\left(w_{max}\right)} \tag{3.6}$$

and according to the equation 3.7 the distance d_{w2} between the same point and the new model centre $\boldsymbol{c_I}$.

$$d_{w2} = V - d_{w1} \tag{3.7}$$

Finally, the diameter of the initial area of validity can be calculated according to the equation 3.8, so that the new model has the weight w_{max} at the same point as the existing model. After checking all the nearby models, the least distance d_I found is used as the are of validity diameter.[1]

$$d_I = -\frac{2}{d_{w2}}\ln\left(w_{max}\right) \tag{3.8}$$

$$\boldsymbol{D}_I = I d_I \tag{3.9}$$

The distance matrix inverse is then created according to the equation 2.12, where $\boldsymbol{I}$ is a unity matrix of the specified order.[2]

3.2.2 Receptive Field Learning Methods

Another adjustment of the RFWR method is a simplification of the optimisation algorithm, which searches for the best possible distribution of the local models' validity areas, because the original method is computationally too complex.

Instead of a cost function, a set of four logical rules based on the model weight and its precision is used:

1. The model is not accurate enough and the weight is too large. ⇒ The validity area has to be smaller.

2. The model is not accurate enough and the weight is small. ⇒ There is no need to change the validity area.

3. The model is accurate and the weight is high. ⇒ There is no need to change the validity area.

4. The model is accurate and the weight is too low. ⇒ The validity area has to be larger.

Thanks to these rules, the distance matrices are not optimised according to the equation 2.18, but according to the equation 3.10, where α represents a learning step length parameter, p represents a quantitative evaluation of the optimisation rules and $\frac{\partial w}{\partial \boldsymbol{M}}$ is a weight Jacobian.

[1] Numerically, it is the maximum value, because parameter d_I is the inverse of the actual diameter for the same reason, why the matrix $\boldsymbol{D}$ is the inverse of the actual distance matrix.

[2] The equations 3.6 and 3.8 can be derived using a natural logarithm of the original equation of the Gaussian function 2.12

$$\boldsymbol{M}^{n+1} = \boldsymbol{M}^n + \alpha p \frac{\partial w}{\partial \boldsymbol{M}} \tag{3.10}$$

To evaluate the p parameter, we need to enumerate the variables that appear in the optimisation rules. The model accuracy is determined by the absolute deviation e between the expected and the measured value, according to the equation 3.11, where e_{cv} is the same as in the equation 2.11. Another option would be to use the relative deviation, but there is a problem with its calculation when the values are close to zero.

$$e = |e_{cv}| \tag{3.11}$$

The border between the accurate and the inaccurate model is a user-defined parameter e_{lim}, which represents the statistically maximal allowed deviation from the measured data and usually can be determined from a basic knowledge about the analysed system and demands for the precision of approximation.

The second variable that appears in the rules is the weight value. The border between the ,,low" and the ,,high" weight is a user defined parameter w_{lim}, which is usually set about 0.5.

A detailed representation of the optimisation rules are in the figure 3.4, which shows the value of the parameter p as a function of the weight value and the deviation value $p = f(w, e)$.

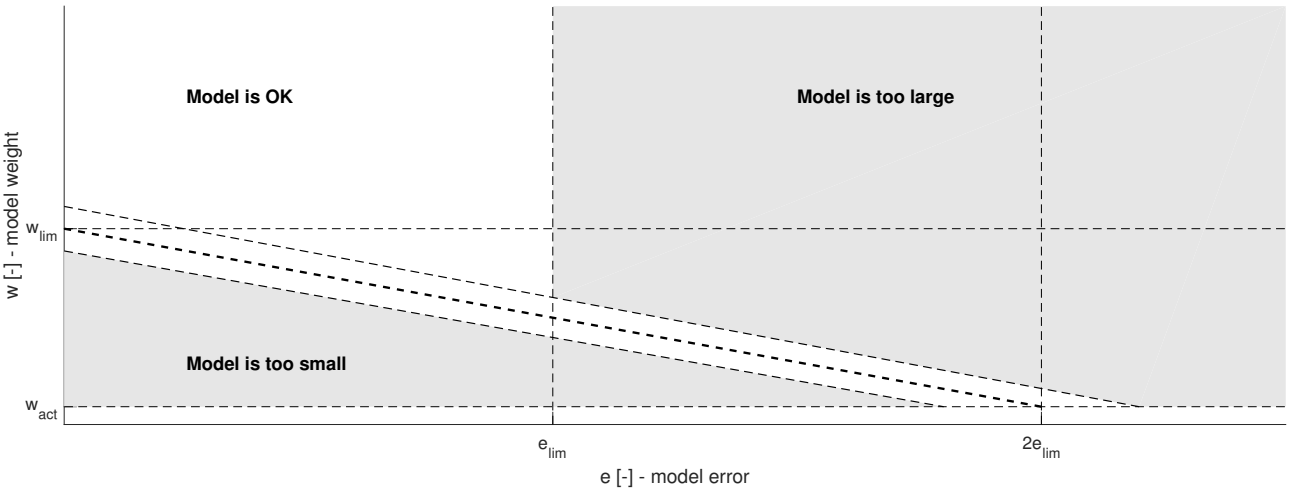

Figure 3.4: Reprezentation of optimization rules

Mathematically, the parameter p can be calculated according to the equation 3.12, where W is a border in the w-e plain between too small and too large area of validity. It can be calculated according to the equation 3.13. w_{act} is the model activation parameter described in the previous section.

$$p(w,e) = \begin{cases} min([0.9 * W - w, w - w_{act}]), & \text{if } w < 0.9W \\ -min([e - e_{lim}, w - 1.1 * W, w - w_{act}]), & \text{if } w > 1.1W \wedge e > e_{lim} \\ 0, & \text{otherwise} \end{cases} \tag{3.12}$$

$$W = w_{lim} - \frac{w_{lim} - w_{act}}{2e_{lim}} e \tag{3.13}$$

3.2.3 Lowering Number of Learning Dimensions

Another important finding, which can be used to significantly lower the computational complexity of the RFWR algorithm is the fact that most nonlinear systems are partially linear. For example, the nonlinearity of the electromechanical actuator used in this thesis concerns just their position and velocity. The inverse model can be therefore described by the equation 2.3. The acceleration (and possibly other variables) appear in the model as a linear combination.

It is desirable, that the local models structure contains a complete model of the systems dynamic, for example as the equation 2.4. That means, it is a function of all relevant states and quantities, that describe the dynamics.

On the other hand, the distribution of the local model does not have to be solved along the dimensions, which represent the linear states.

As a result of this simplification, the kernel function does not concern the distance along the linear dimensions, just those which are connected to a nonlinearity, which significantly lower the number of local models needed to cover the whole space.

As an example of this simplification we can use a torque driven pendulum, which has an inverse model described by the equation 3.14.

$$m = b_1 \sin\varphi + b_2\dot{\varphi} + b_3\ddot{\varphi} \tag{3.14}$$

This inverse model can be substituted by several local linear models with the structure of the equation 3.15. Every local model has four parameters $b_{i1} - b_{i4}$, therefore its covariance matrix (when using RLS for parameter estimation) will also be of fourth order.

$$m_i = b_{i1}\varphi + b_{i2}\dot{\varphi} + b_{i3}\ddot{\varphi} + b_{i4} \tag{3.15}$$

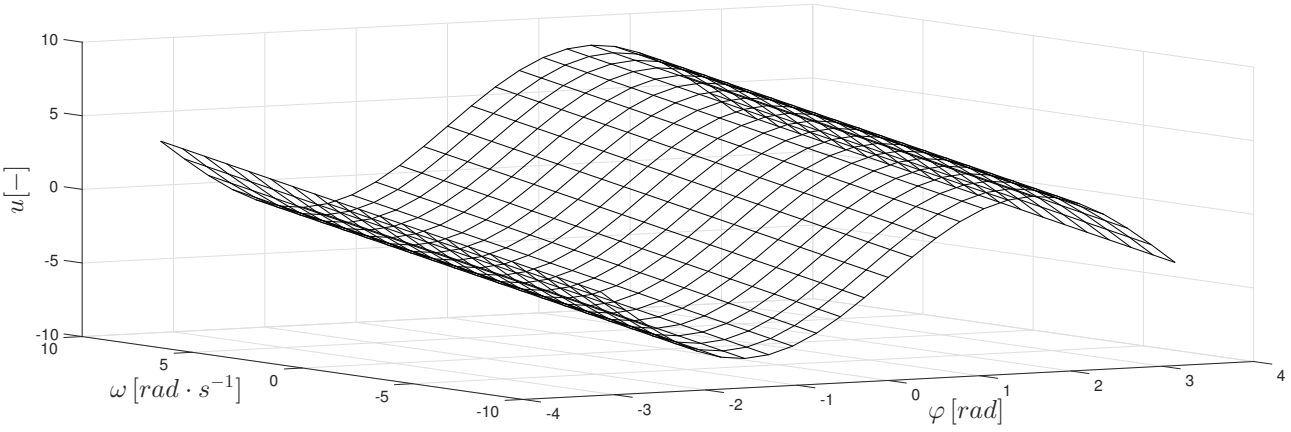

Figure 3.5: Inverse model with a lower number of learning dimensions

However, the nonlinearity concerns only the pendulums rotation $\boldsymbol{\varphi}$ so that it is sufficient that the local model distribution is optimised only in one-dimension. That means the distance matrices of each local model's kernel function will be scalars. Figure 3.5 shows the visualisation of the pendulum inverse model in two dimensions $\boldsymbol{\varphi}$ and $\dot{\boldsymbol{\varphi}}$, assuming $\ddot{\boldsymbol{\varphi}} = \mathbf{0}\,\boldsymbol{rad\cdot s^{-2}}$.

3.2.4 Simulations

To test the RFWR method and its adjustments it was implemented in the Matlab environment and used in several simulations. Those simulations were devoted to test the methods approximation capabilities, that mean if the parameter estimation works, if the models are added and removed correctly and mainly if the local models validity area optimisation converges to a satisfactory result.

The first simulation was based on approximating a known nonlinear function of one variable according to the equation 3.16, with a domain $\langle -\pi; \pi \rangle$. The patterns were generated as value of the approximated function with white noise at a random position on the functions domain.

The local models' had structure according to the equation 3.17.

$$y = \sin(x) + \frac{1}{3}\cos(3x) + 25\arctan(x + 0.5)\, e^{-250(x+0.5)^2} \tag{3.16}$$

$$y = b_1 x + b_2 \tag{3.17}$$

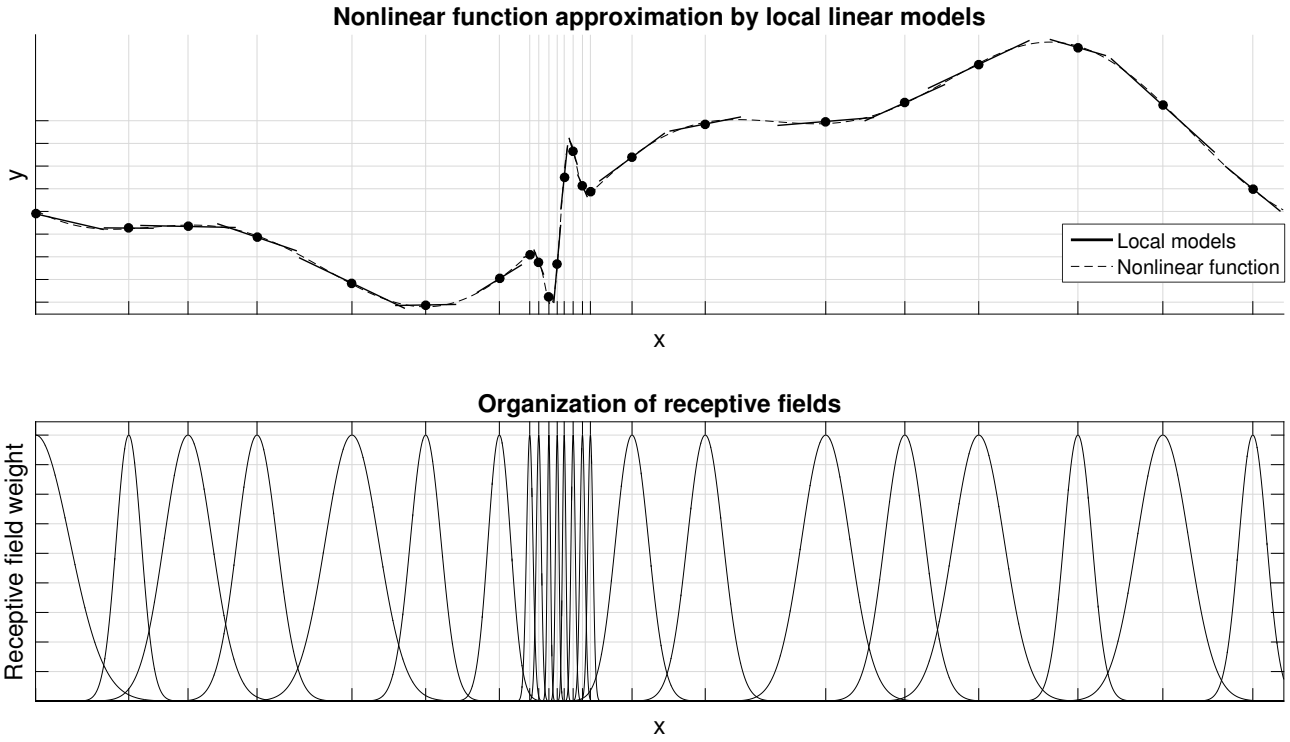

Figure 3.6: RFWR approximation of a one-dimensional nonlinear function

The resulting approximation with the areas of validity is shown in the figure 3.6. It is clear, that the RFWR method with above described adjustments works very well in a one-dimensional case. The local models are distributed as planned, with more models in the area of a significant nonlinearity.

The second simulation was similar to the first, but it tried to approximate a two-dimensional function described by the equation 3.18. The local models' had structure according to the equation 3.19.

$$y = 0.05x_1 + 1.5 \cdot 10^{-4} x_1^3 + 10^{-3} x_1 x_2 + 25 \arctan(x_1)\, e^{-10^{-2} x_1^2} \tag{3.18}$$

$$y = b_1 x_1 + b_2 x_2 + b_3 \tag{3.19}$$

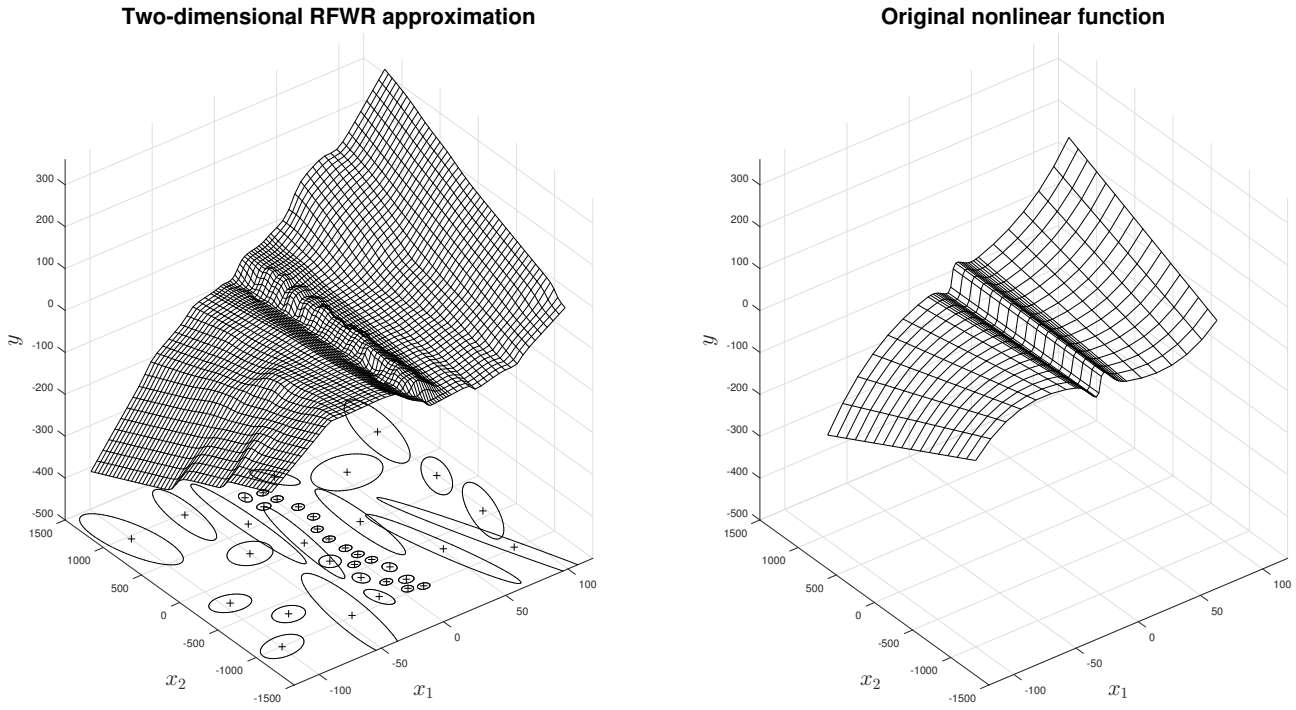

Figure 3.7: RFWR approximation of a two-dimensional nonlinear function

The figure 3.7 shows the results of the approximation together with the distribution of local modes expressed using the ellipses. We can see, that the RFWR algorithm was still able to approximate the original nonlinear function, even though there was white noise added to the patterns.

Both simulations proved that the adjusted algorithms work as intended. Of course, it was necessary to set correct user parameters, especially those which are not general but are connected to the actual nonlinear function. A list of all parameters that need to be set by the user with their default values and values used in each simulation is in the table 3.1.

3.3 LOLIMOT

The LOLIMOT method (Local Linear Model Tree), as further described in the section 2.3.4, differs from RFWR mainly in the approach of placing new local models. The original algorithm described in [1] is suitable for off-line usage.

Since new models are created by dividing the existing ones, the algorithm needs to decide along which axis the specific model will be divided. If working off-line, it can be done simply by trying all possible divisions and selecting the best one, but this is not a deterministic approach and it cannot be used for real-time implementation.

An on-line adjustment of this algorithm firstly chooses which model is to be divided. This

Parameter Name	**Sign**	**Default value**	**Interval**	**1D Test**	**2D Test**
Learning speed	α	1	$(0;\infty)$	25	5
Standard precision	e_{lim}	-	$(0;\infty)$	50	3
Model activation weight limit	w_{act}	0.001	$\langle 0;1\rangle$	0.001	0.001
Model generation weight limit	w_{gen}	0.01	$\langle 0;1\rangle$	0.01	0.01
Model pruning weight limit	w_{prun}	0.7	$\langle 0;1\rangle$	0.7	0.7
Distance matrices optimisation weight limit	w_{lim}	0.7	$\langle 0;1\rangle$	0.7	0.7
Number of distance matrices learning dimensions	N	2	$\langle 1;\infty)$	1	2

Table 3.1: List of RFWR user parameters

is done according to the model statistical estimate of its mean square error, which is updated with every new measurement according to the equation 3.20, where $\boldsymbol{\lambda}$ is the same forgetting parameter as in case of the RLS method used for the parameter estimation, $\boldsymbol{w}$ is the model weight given the new pattern and $\boldsymbol{e_{cv}}$ is the local model deviation.

$$MSE_i^{n+1} = MSE_i^n \left(1 - (1-\lambda)\, w\right) + w e_{cv}^2 \tag{3.20}$$

For a simple decision, along which axis the model should be divided, we can use the covariance matrix of the corresponding models estimated parameters. The model is divided along the axis, whose corresponding parameter estimate has the largest variance.

This approach gives slightly worse results than the original algorithm and ends up with a few more local models than necessary, but that is the price for lower computational complexity and deterministic response.

Also note that the whole problem is much simpler in a one-dimensional case, like the one described in the end of the section 3.2.3, because there is only one possibility for the model division and thus no decision algorithm is needed. In a two-dimensional case, the distribution of local models can look like 2.8.

3.3.1 Simulations

The LOLIMOT method was simulated using the same nonlinear functions, as RFWR in the previous sections. The only difference was that LOLIMOT had to be initialised by the first local model covering the whole domain of the nonlinear function.

Also to make the method converge and stop adding new models to increase the approximation precision even further, the model deviation e_{cv} used for the mean square error calculation according to the equation 3.20 was considered zero if lower than a set limit e_{lim}. These limits were 250 for the one-dimension case and 5 for the two-dimensional case.

The result of the first simulation, approximating a one-dimensional nonlinear function according to the equation 3.16 is shown in the figure 3.8. It used the same model structure according to the equation 3.17.

The LOLIMOT method also performs well in a one-dimensional case, placing more models

into the area with the most significant nonlinearity.

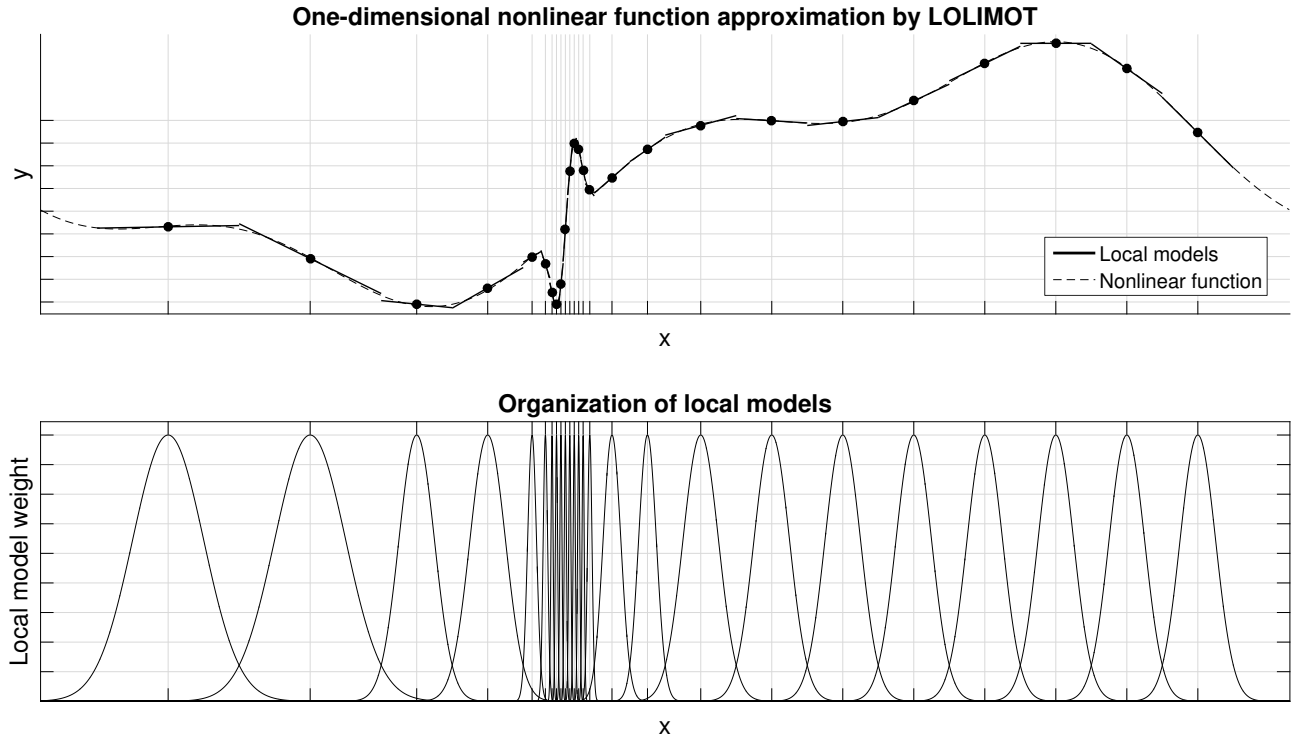

Figure 3.8: RFWR approximation of a two-dimensional nonlinear function

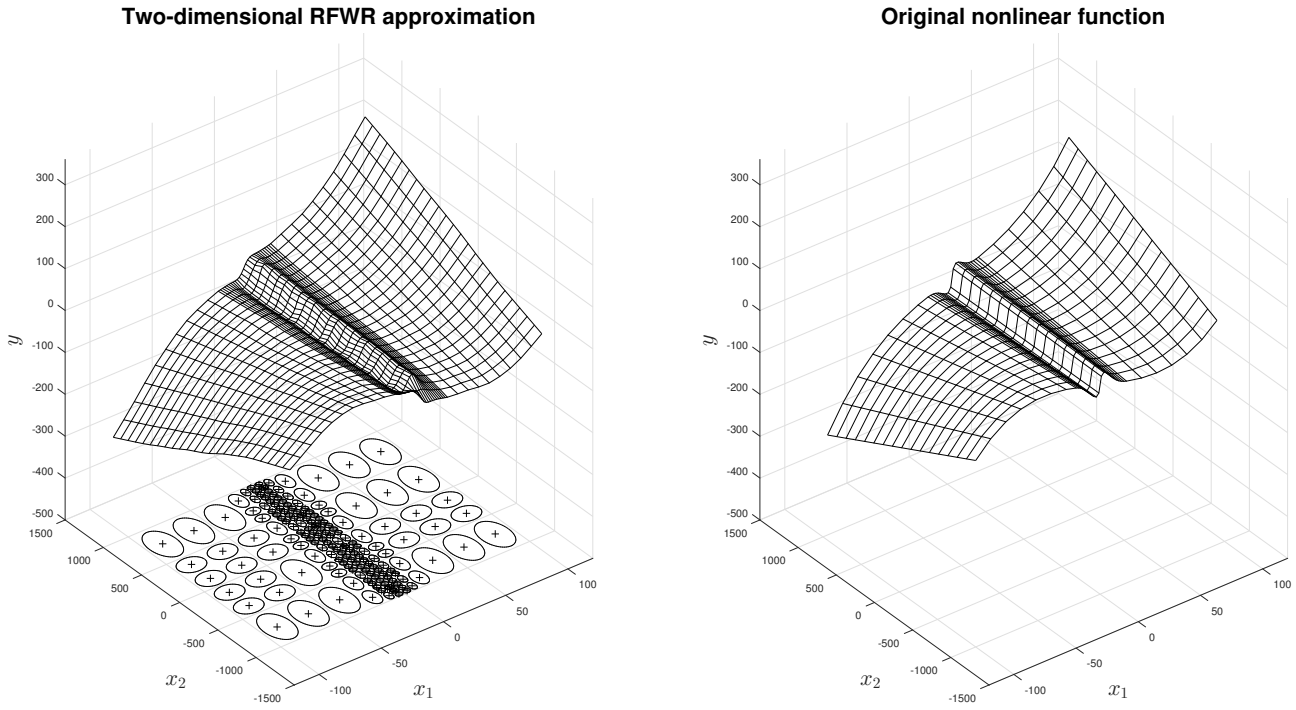

Figure 3.9: LOLIMOT approximation of a two-dimensional nonlinear function

The result of the second simulation, approximating a two-dimensional function according to the equation 3.16, is shown in the figure 3.9. Is also used the same model structure as the

RFWR algorithm, according to the equation 3.19.

In the two-dimensional case, the LOLIMOT function is also able to create a working approximation, but it needs about four times more models than RFWR. The explanation is that LOLIMOT does not prune the models when they become redundant.

3.4 Learning with RLS

This section is devoted to simulation and experimental analysis of the parameter estimation using the RLS method described in the section 2.2. As mentioned before, the use of this estimation method requires the structure of the local models to be linear in parameters.

It was decided that a simpler system - a position controlled DC motor will be used for both simulations and experiments. The reason is that it is a simple, almost linear, and well described system and its mathematical model can be used with the RLS method. From the local approximation point of view, it means that only one local model is needed to approximate the inverse model of the system.

In fact it is a global approximation, but it allows for testing of the behaviour of the RLS estimation and also the composite control with the inverse dynamic model without the need to deal with the distribution of local models.

3.4.1 DC Motor Inverse Dynamics Model

A DC motor model, which also contains a dry friction Coulomb model is described by a voltage equation 3.21 and a torque equation 3.22, which are interdependent on the motor current.

The parameters and variables in the equation are:

Variables:

$u(t)$ $[V]$	.....	*voltage*
$\omega(t)$ $[rad \cdot s^{-1}]$	.....	*angular velocity*
$i(t)$ $[A]$	.....	*current*

Parameters:

R $[\Omega]$	.....	*winding resistance*
L $[H]$	.....	*winding inductance*
C_ϕ $[V \cdots]$	.....	*motor induction constant*
J $[kg \cdot m^2]$	.....	*shaft inertia*
b $[N \cdot m \cdot s]$	.....	*viscous friction coefficient*
T $[N \cdot m \cdot s]$	.....	*dry friction coefficient*

It is important to remark that the model does not cover a general load torque, because it cannot be compensated with a feedforward compensation without its precise measurement or estimation. This experiment is meant to cover all cases, where the load torque can be mathematically defined as a function of any motor state, which could be easily added to the torque equation as another function term.

$$u(t) = Ri(t) + L\frac{di(t)}{dt} + C_\phi \omega(t) \tag{3.21}$$

$$J\frac{d\omega}{dt} = C_\phi i(t) - b\omega(t) - Tsgn(\omega(t)) \tag{3.22}$$

To infer the structure of the DC motor's inverse dynamic model as the equation 2.1, we express the current $i(t)$ from the equation 3.22, acquiring the equation 3.23, and substitute it in the equation 3.21. We also substitute its time derivative from the equation 3.24[3].

$$i(t) = \frac{J}{C_\phi}\frac{d\omega(t)}{dt} + \frac{b}{C_\phi}\omega(t) + \frac{T}{C_\phi}sgn(\omega(t)) \tag{3.23}$$

$$\frac{di(t)}{dt} = \frac{J}{C_\phi}\frac{d^2\omega(t)}{dt^2} + \frac{b}{C_\phi}\frac{d\omega(t)}{dt} \tag{3.24}$$

After the substitutions, we get the equation 3.25, which represents an inverse dynamic model of a DC motor. When we substitute the motor parameters by the parameters $\boldsymbol{b_1}$ - $\boldsymbol{b_4}$ we get the equation 3.26, whose parameters can be estimated by the RLS method and which can be used as a feedforward compensator of the DC motor.

$$u(t) = \left(\frac{Rb}{C_\phi} + C_\phi\right)\omega(t) + \frac{RJ + Lb}{C_\phi}\frac{d\omega(t)}{dt} + \frac{LJ}{C_\phi}\frac{d^2\omega(t)}{dt^2} + \frac{RT}{C_\phi}sgn(\omega(t)) \tag{3.25}$$

$$u(t) = b_1\omega(t) + b_2\frac{d\omega(t)}{dt} + b_3\frac{d^2\omega(t)}{dt^2} + b_4 sgn(\omega(t)) \tag{3.26}$$

The equation 3.26 describes the inverse dynamics by a series of time derivatives. Another possibility is to convert the model to a discreet form by substituting the derivatives with simple differences. The discreet form is described by the equation 3.27.

$$u(t) = b_1\varphi_k + b_2\varphi_{k-1} + b_3\varphi_{k-2} + b_4\varphi_{k-3} + b_5 sgn(\varphi_k - \varphi_{k-1}) \tag{3.27}$$

An advantage of using the discreet form over the continuous is a fact, that there is no need to calculate or estimate the original signal (in this case a motor shaft rotation φ) derivatives, if they are not measured, which is a very problematic task.

However, a significant disadvantage of the discreet form is, that a nonlinearity can become a function of two or more variables. When we compare the equations 3.26 and 3.27, we can see, that in the discreet form the signum term is a function of both φ_k and φ_{k-1}.

If such a local model structure was used in a more complex case, it would prevent use of the simplification described in the section 3.2.3 and thus significantly increasing the computational complexity.

The difference between the discreet and the continuous model form can be also expressed graphically. The figure 3.10 shows the continuous and the discrete form of an inverse dynamic model described by equations 3.28 and 3.29. Where $u(t)$ is the input force of the system, and $x(t)$ and $v(t)$ are position and velocity.

[3]The time derivatives of equation 3.23 term $Tsgn(\omega(t))$ would be a Dirac impulse, which is neglected, because in real implementation with discrete time period it is impracticable. Also the signum function is not a precise representation of the dry friction effect.[17]

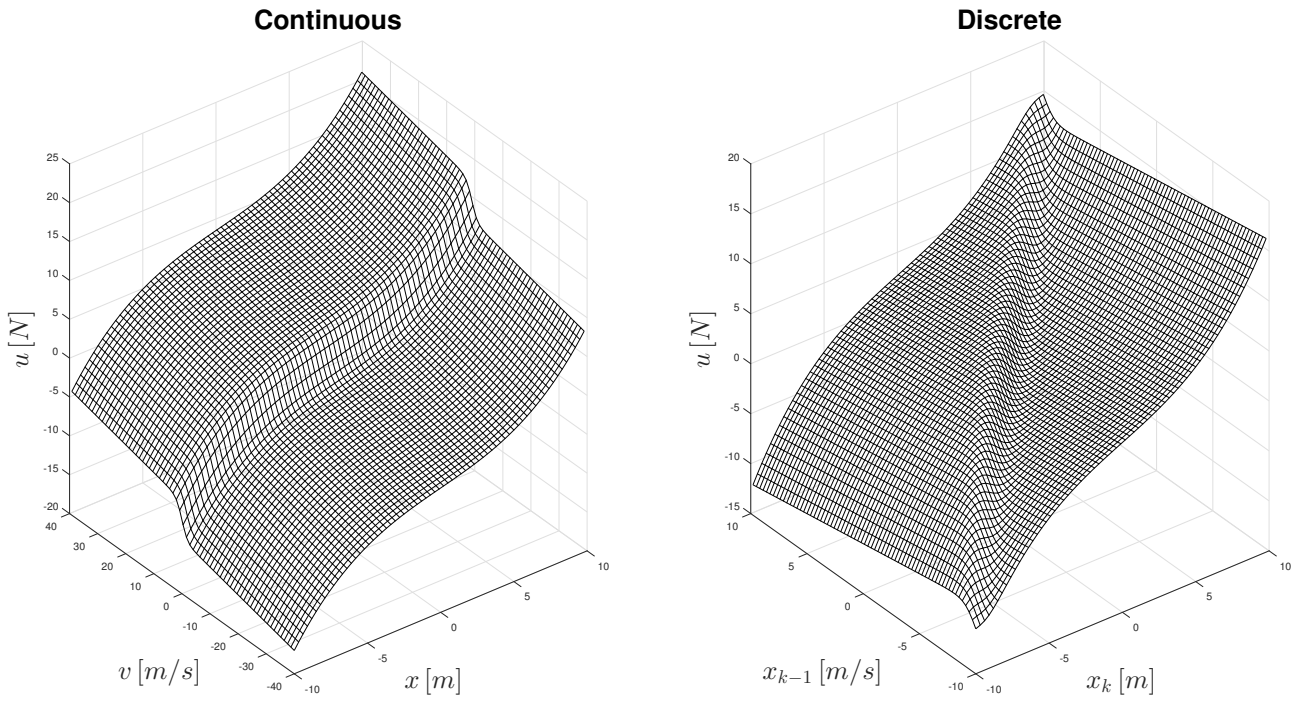

Figure 3.10: Comparison of inverse dynamic models in discrete and continuous form

$$u(t) = b_1 v(t) + b_2 \frac{1}{1 + e^{-v(t)}} + b_3 x(t) + b_4 x^3(t) \tag{3.28}$$

$$u_k = \frac{b_1}{T_s}(x_k - x_{k-1}) + b_2 \frac{1}{1 + e^{-\frac{x_k - x_{k-1}}{T_s}}} + b_3 x_k + b_4 x_k^3(t) \tag{3.29}$$

The parameter $b_1 = 0.1\,N \cdot s \cdot m^{-1}$ represents the viscous friction, the parameter $b_2 = 5\,N \cdot s \cdot m^{-1}$ represents the dry friction using a sigmoid function, and the parameters $b_3 = 0.5\,N \cdot m^{-1}$ and $b_4 = 0.008\,N \cdot m^{-3}$ represent a nonlinear stiffness. We can see, that the discretisation causes a transformation of the state space in such a way that a nonlinear shape, which was parallel with one of the continuous space axes, no longer is.

3.4.2 Noisy Signal Derivative

If a continuous form of the inverse model is used to control the DC motor, a way to effectively determine the measured noisy signal time derivatives has to be found.

It is known, that using a combination of a lowpass filter and the numerical difference to determine a noisy signal derivative is possible only if the measuring sample time is much faster than the measured system dynamics. Since the numerical difference behaves like a high frequency amplifier, it also prevents determining higher order derivatives by completely loosing the signal in the noise.

Also any filtration influences the measured dynamics and thus deviates the parameters estimated from such a signal.

A different approach is used in this thesis. To determine one or more derivatives of a signal, several samples are approximated by a polynomial of a sufficient order using the LS method (equation 2.7) and the polynomial is analytically differentiated to acquire an approximation of

the signal derivatives. An example of the whole situation is shown in the figure 3.11, where thirty measured samples are approximated by a fourth order polynomial to acquire two consecutive derivatives of the signal.

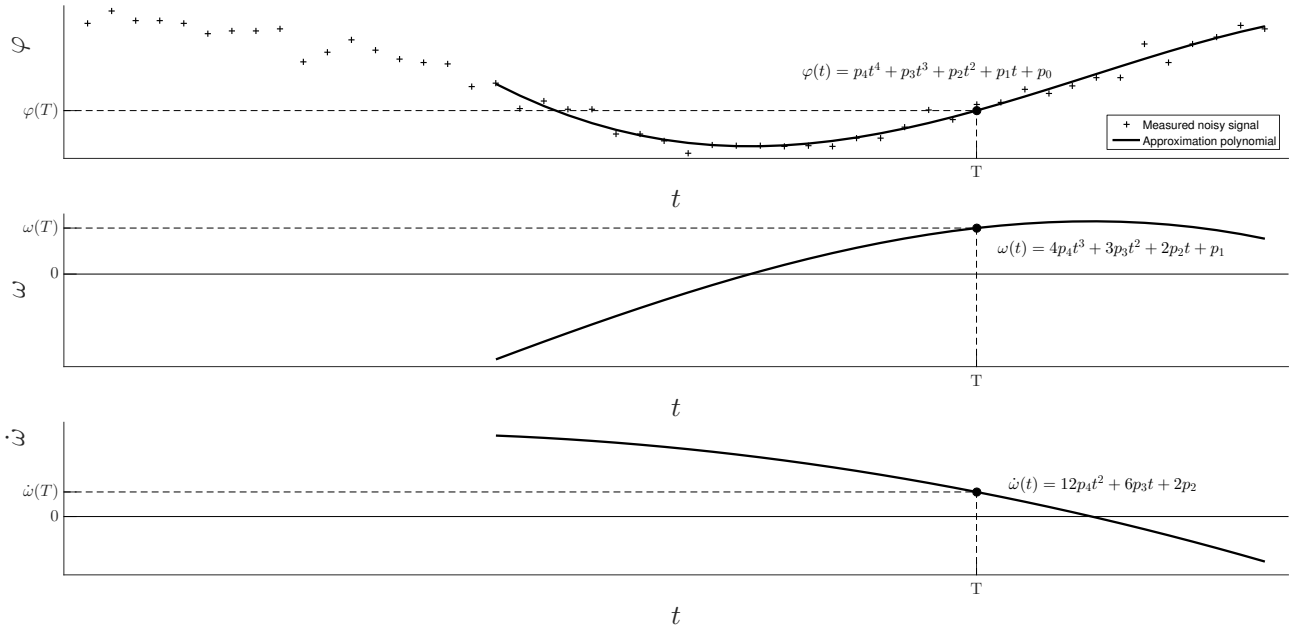

Figure 3.11: Calculating derivative using approximation polynomial

The actual values of the measured signal and its derivatives in a specific moment can be calculated as an evaluation of the corresponding polynomial. Depending on the signals SNR, about three consecutive derivatives can be calculated.

The figure 3.12 shows a comparison of this method with different approaches. The initial signal is the rotation of an electronic throttle valve described in the section 4.1.1, measured by a potentiometer sensor.

We can see, that the polynomial approach gives much better results, than a simple difference and filtration approach.

To implement this polynomial method for derivative calculation, a correct order of the polynomial has to be selected. It depends on the number of samples used in each step and the number of consecutive derivatives that need to be calculated.

Because we need to perform several derivatives of the polynomial, the minimal order of the polynomial has to be higher than the number of the derivatives needed, so that all of them exist and so that the last derivative is still approximated at least by a first order polynomial. Preferably, a polynomial of one or two orders higher than the minimum should be used to get the best results.

The order of the approximation polynomial is also limited by the number of samples used for LS estimation. Obviously, there has to be more samples, than the parameters of the polynomial so that the LS estimation has exact results, but at least two or three times more samples than the estimated parameters should be used.

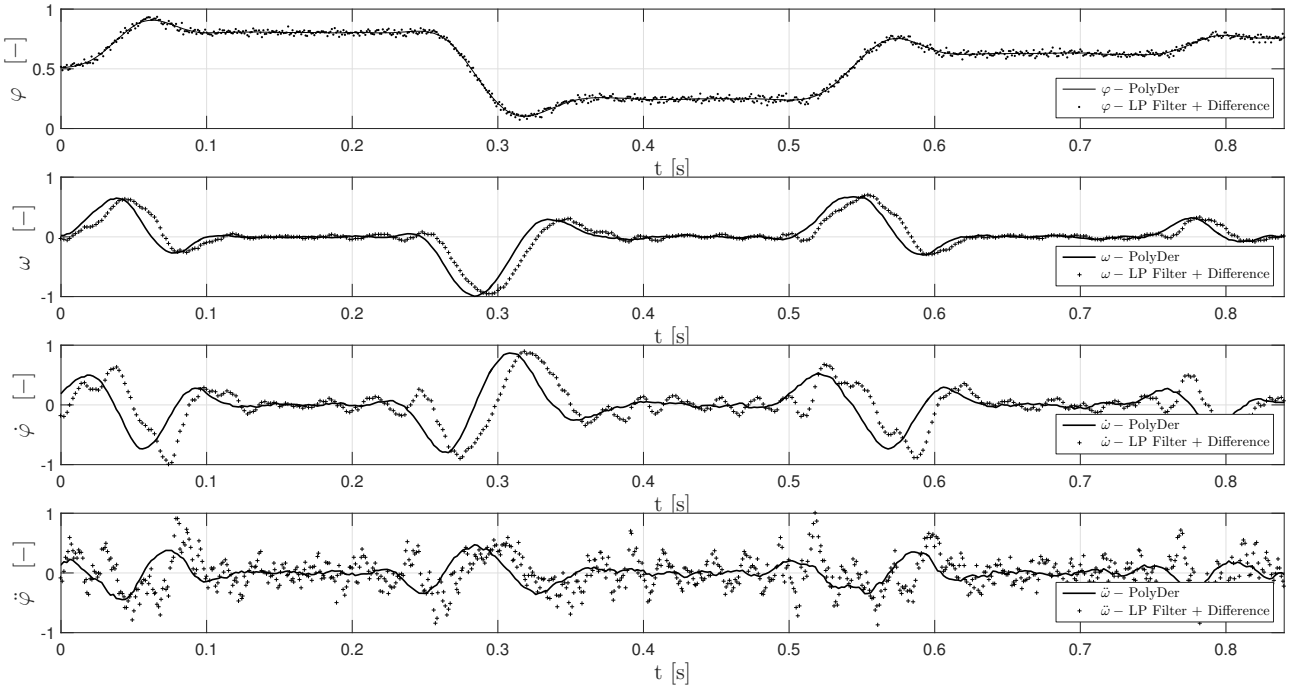

Figure 3.12: Comparicon of noisy signal derivative acquired by various methods

3.4.3 DC Motor Adaptive Control

The previously described methods can form a DC motor adaptive controller with a feedforward compensation. Besides the inverse models with a structure according to the equations 3.26 and 3.27, the model can also cover other effects, for example a torque of an asymmetrical motor load like a pendulum or a nonlinear viscous friction. The only limitation is the requirement, that the function describing the effect must be suitable for the RLS estimation.

The whole structure of the adaptive control algorithm is in the figure 3.13. The principle is based on the composite control using the inverse model as described in the section 2.1, but is also enhanced by an on-line parameter estimation algorithm.

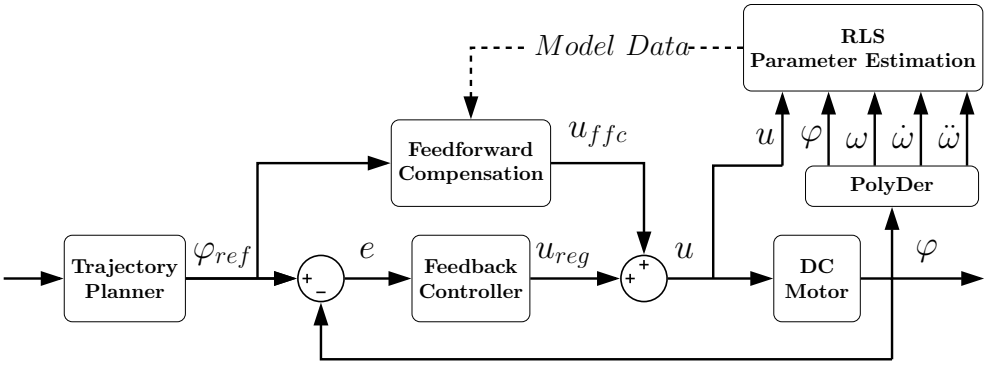

Figure 3.13: Block diagram of DC motor adaptive control

The estimation algorithm (represented by the **RLS Learning** block in the figure 3.13) is using the RLS method to fit a model parameters to the pattern data. The same model with the last estimated parameters is used in the **Feedforward compensation** block to calculate the compensation input.

The model used for a DC motor dynamics compensation can have various structures used in different situations. The next section describes an experiment with several of them.

3.4.4 Experimental Results

The control algorithm described in the previous section was implemented using the Matlab/Simulink environment and a real-time I/O card Humusoft MF624 and through a standard duty driven H-bridge made in our laboratory used to control an educational stand with a Maxon DC motor shown in the figure 3.14.

All of the model structures listed in the table 3.2, were subjected to the same experiment. An angular position control of the motor for $\mathbf{100}\boldsymbol{s}$, with a $\mathbf{10}$ second long learning period. The reference signal was generated with a random number generator with a uniform probability density function and a third order lowpass filter with a time constant $\boldsymbol{\tau} = \mathbf{0.06}\boldsymbol{s}$. The feedback controller was a P-type controller with a gain $\mathbf{0.01}$.

Figure 3.14: Block diagram of DC motor adaptive control

Tale table also contains some statistical results over the $\mathbf{90}\boldsymbol{s}$ long measuring period. The $\boldsymbol{MSE}$ $[\boldsymbol{rad}]$ column represents a mean square error of the difference between the reference and the measured angular position. The column $\overline{\boldsymbol{U}}_{\boldsymbol{FFC}}$ $[-]$ represents the average compensation duty calculated by the feedforward compensation and the column $\overline{\boldsymbol{U}}_{\boldsymbol{P}}$ $[-]$ represents the control duty calculated by the P-type controller.

Also note that the model structure equations are functions of $\boldsymbol{\varphi}$, which represents the angular position of the motor shaft, or $\boldsymbol{\omega}$, which represents the angular velocity, and they output a normalised control signal in the interval $\langle \mathbf{-1}; \mathbf{1} \rangle$, which is then converted to the H-bridge duty and rotation direction.

From the results we can see, that all tested model structures were able to greatly overcome the influence of the feedback controller and were able to maintain a stable control almost by themselves. This leads to a conclusion, that using an inverse model as a feedforward compensator is an effective way to create an adaptive controller.

Furthermore, we can see that it is useful to use a complete DC motor model (N. 3) over the incomplete ones (N. 1,2), because it produces much more accurate control. The reason is that

N.	Type	Equation	Description	MSE	$\overline{U}_{FFC}$	$\overline{U}_P$
1.	continuous	$u = b_1\omega + b_2 sgn(\omega)$	J and L are not compensated	0.3757	0.2062	0.0048
2.	continuous	$u = b_1\omega + b_2\dot{\omega} + b_3 sgn(\omega)$	L is not compensated	0.3241	0.2067	0.0044
3.	continuous	$u = b_1\omega + b_2\dot{\omega} + b_3\ddot{\omega} + b_4 sgn(\omega)$	complete compensation	0.1428	0.2120	0.0029
4.	continuous	$u = b_1\omega + b_2\dot{\omega} + b_3\ddot{\omega} + b_4 sgn(\omega) + b_5 \sin(\varphi)$	unbalanced inertia connected	0.2976	0.2144	0.0043
5.	discreet	$u = b_1\omega_k + b_2\omega_{k-1} + b_3\omega_{k-2} + b_4\omega_{k-3} + b_5\omega_{k-4} + b_6 sgn(\omega_k)$	velocity based compensation	0.1717	0.2072	0.0032
6.	discreet	$u = b_1\varphi_k + b_2\varphi_{k-1} + b_3\varphi_{k-2} + b_4\varphi_{k-3} + b_5\varphi_{k-4} + b_6 sgn(\varphi_k - \varphi_{k-1})$	position based compensation	1.6360	0.2072	0.0093

Table 3.2: List of various DC motor compensation models

the DC motor current, which is usually used to handle most of the torque load dynamics, is not used here.

Model N. 4 was able to control the nonlinear part of the system caused by an unbalanced load connected to the motor shaft. The load is shown on one of the motors in the figure 3.14.

Equation term	Average influence	Prevailing effect
$b_1\omega$	58.74 %	Viscous friction
$b_2\dot{\omega}$	0.39 %	Inertia
$b_3\ddot{\omega}$	0.12 %	Inductance
$b_4 sgn(\omega)$	28.16 %	Dry friction
$b_5 \sin(\varphi)$	12.59 %	Nonlinear load

Table 3.3: The influence percentage of the model N.4 terms

Finally, the discreet variants of the model structure generally performed worse than the continuous ones, even though the used more discreet samples than needed. The reason for that is that the measured signal contained noise and that the model N. 6 used an unprocessed signal which worsened its ability to model the systems dynamic.

The model N. 5 performed better, because it used directly the angular velocity ω, determined by the polynomial approximation and differentiation algorithm.

The figure 3.15 shows the learning process of the DC motor composite control algorithm with the model N. 4. The RLS method can estimate the model parameters very quickly. Sfter that, the P-type controller is necessary only when the there is an unexpected torque load.

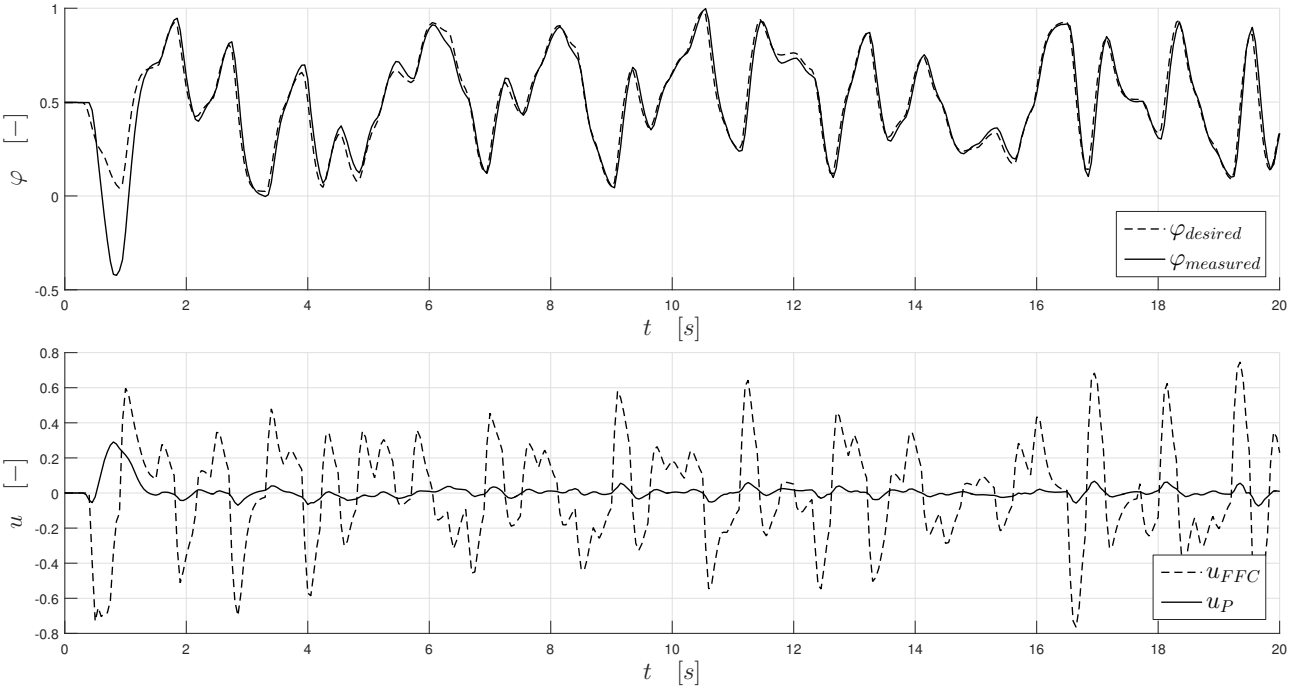

Figure 3.15: DC motor adaptive control learning

It was also possible to determine the effect of each term of the model's structure as their average addition to the control output. The table 3.3 contains the comparison in percentage. The results confirmed that educational stand DS motors have very significant friction effects, probably thanks to its planetary gear.

However, the terms representing inertia and inductance seem to be negligible, the statistical results it the table 3.2 show that they have a significant influence of the positional control precision, especially when such a weak P-type feedback controller is used.

4 Experiments with Real Systems

This chapter contains the final experiments conducted to test the control of various automotive actuators. The selected actuators are described in the section 4.1.

The section 4.2 described the design of the adaptive control algorithm and its implementation on the dSPACE hardware that provides enough computational power to use the local approximation methods in their complete form, described in the chapter 3.

Finally, the section 4.3 describes the implementation of a simplified version of the adaptive control algorithm on a development board with a dsPIC microcontroller, which was developed in the Mechatronics Laboratory.

4.1 Automotive Actuators

All automotive actuators used for actual experiments are DC servo drives with a positional feedback - a linear or nonlinear torsion spring. They also contain a very strong dry friction, which is a significant obstacle when using regular feedback control algorithms.

The actuators provide a measurement of the motor shaft angular rotation and current. However, the current measurement was not used.

The motor current is an inner state and does not contain any new information about the systems dynamics, but it could be used to significantly improve the precision of estimating the derivatives of the measured rotation.

The actuators differ mainly in the torsion spring load characteristics, both dry and viscous friction degree, angular rotation measuring principle and its SNR, gearing backlash and the overall dynamic properties.

Generally, the dynamic properties of these actuators come from the DC motor equations mentioned in the section 3.4.1: the voltage equation 3.21 and the momentum equation 3.22, which can be written in an more general case as the equation 4.1.

$$J\frac{d\omega}{dt} = C_\phi i(t) - f(\varphi) - g(\omega) \tag{4.1}$$

The nonlinear function $\boldsymbol{g(\omega)}$ represents the influence of the friction effects. In this thesis, the friction - both dry and viscous - are modelled as two separate effects according to the equation 4.2, which allows the use of the dimensional reduction approach described in the section 3.2.3. The parameter $\boldsymbol{b}$ represents the coefficient of viscous friction and the parameter $\boldsymbol{T}$ represents the coefficient of the dry friction. [17].

$$g(\omega) = b\omega + T sign\,(\omega) \tag{4.2}$$

The nonlinear function $\boldsymbol{f(\varphi)}$ represents the torsion spring load characteristic as a function of the angular rotation $\boldsymbol{\varphi}$. The characteristics of each actuator were measured during repeated, very slow opening and closing cycles of the valves or servos.

4.1.1 Electronic Throttle Valve

The electronic throttle valve serves in a vehicle to regulate airflow to the engine. The figure 4.1 shows an illustration photo.

The throttle valve is connected through a gearing to a DC motor shaft and the valve's angular rotation is measured with a potentiometer. Also, a nonlinear torsion spring is connected to the valve's shaft, which ensures that the valve stays slightly open in case it is not powered properly.

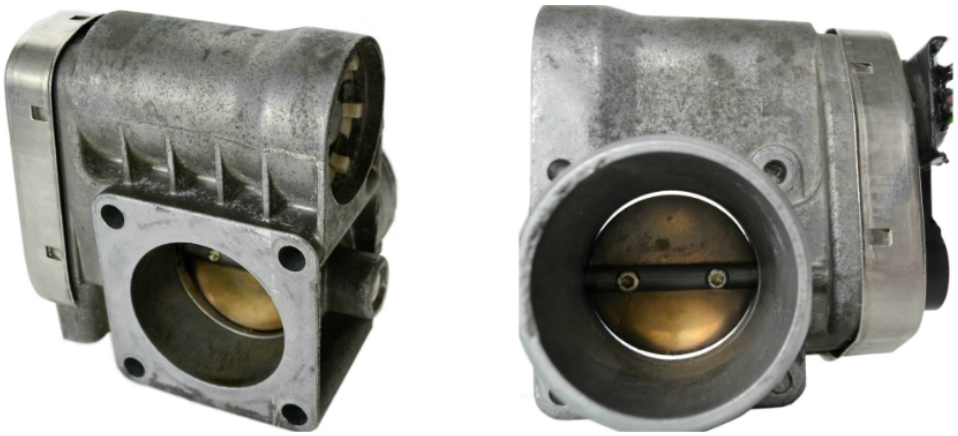

Figure 4.1: Electronic throttle valve

The figure 4.2 shows a measured quasi-static load characteristic of the throttle. The nonlinearity takes place within a very narrow angle just around the stable position.

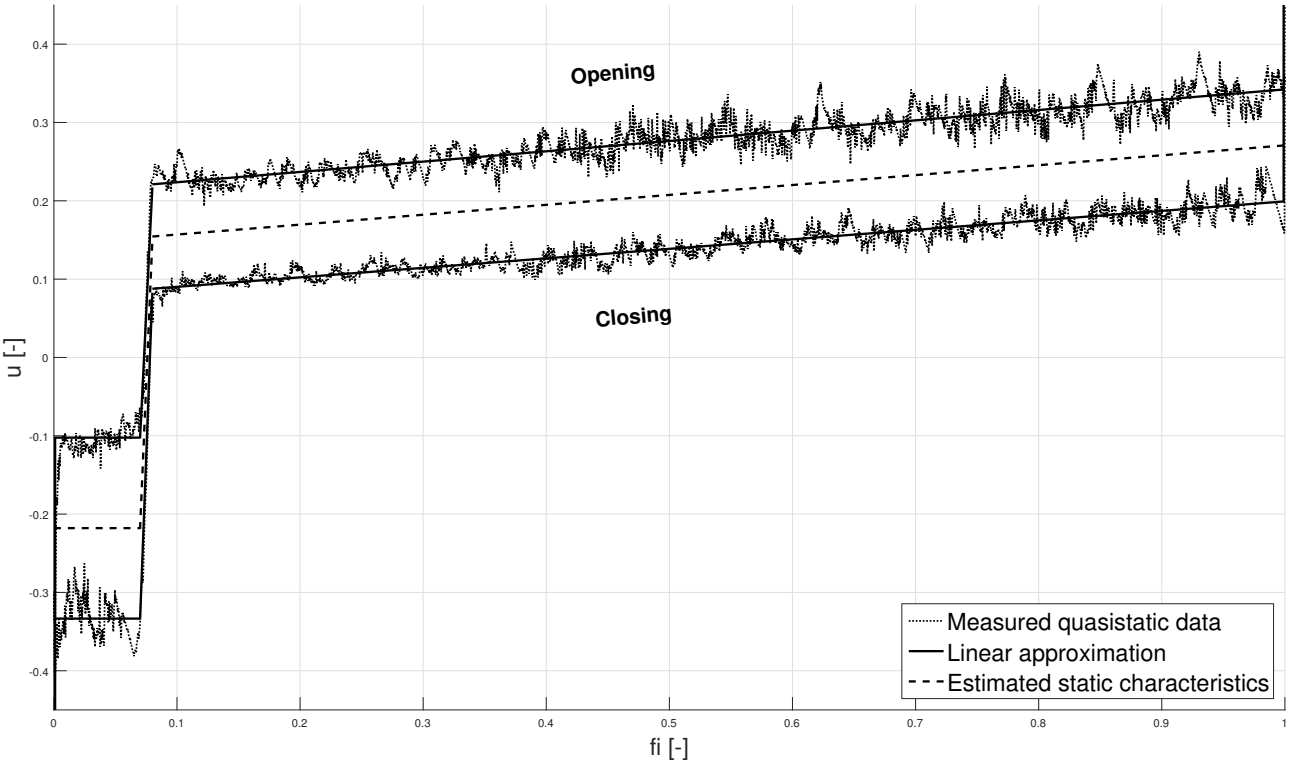

Figure 4.2: Static characteristic of the electronic throttle valve

4.1.2 EGR Valve

The EGR (Exhaust Gas Recirculation) valve is used to regulate a repeated burning of the gases in a vehicle's engine, which helps to control the engine cylinders temperature and to reduce formation of harmful NOx gases. The figure 4.3 shows an illustration photo of an EGR valve.

The EGR valve's construction is similar to the electronic throttle. The differences are a nonlinear gearing and a different rotation sensor. This valve uses Hall sensors, which determine the valve's angular rotation, leading to a less noisy signal.

Also, there is a difference in the overall dynamics of both systems.

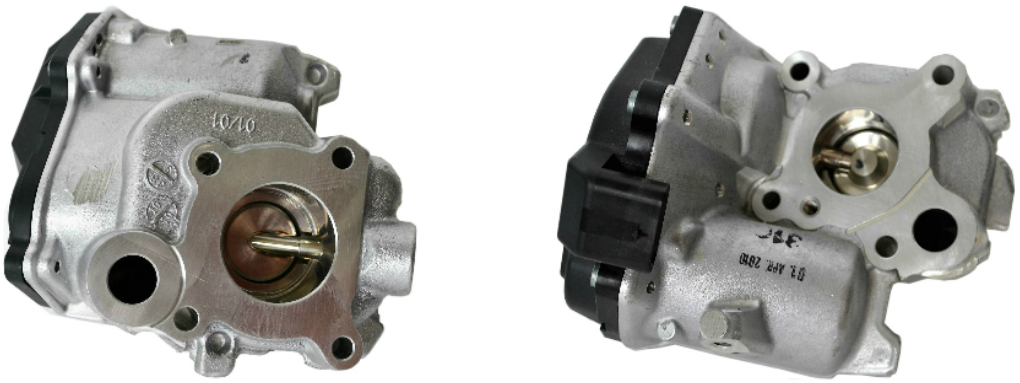

Figure 4.3: EGR valve

The figure 4.4 shows a quasi-static load characteristic of the EGR valve. The torsion spring has slightly different stiffness than the throttle valve and there is also a greater influence of the gearing cogs and its nonlinearity in higher angles.

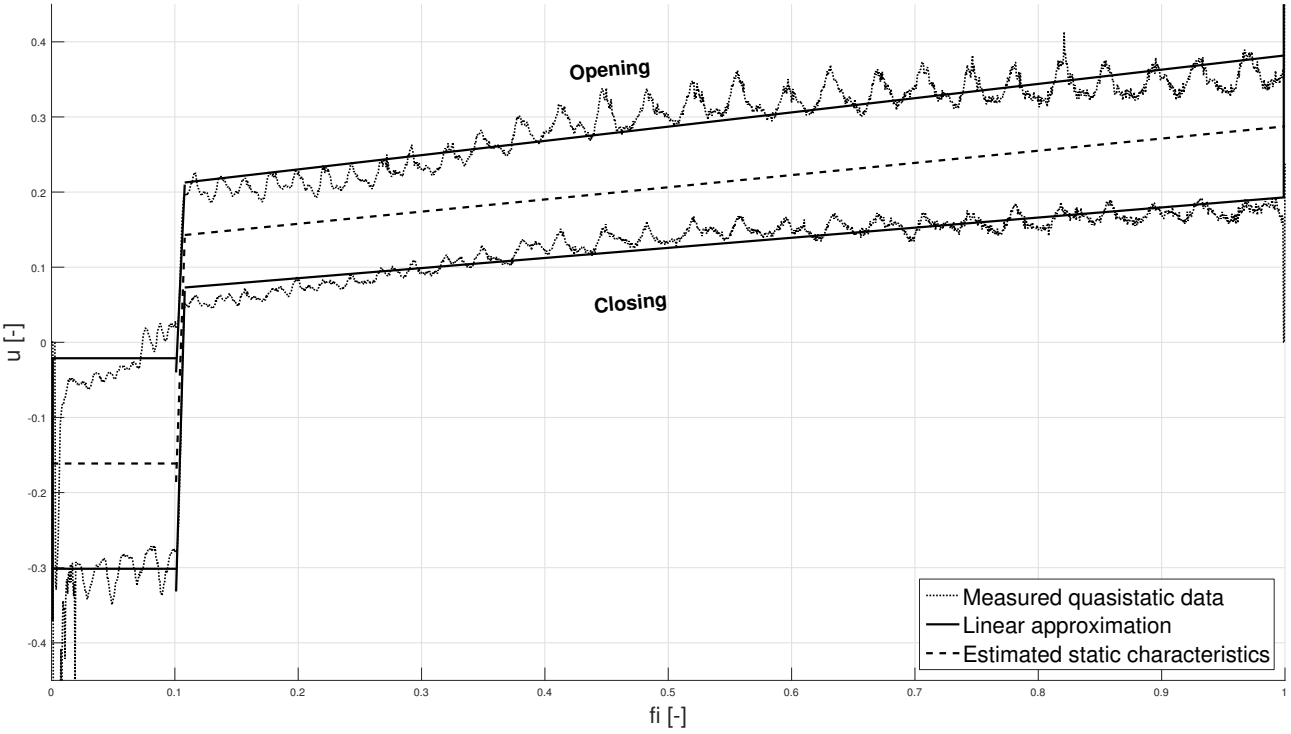

Figure 4.4: Static characteristic of the EGR valve

4.1.3 Servo Drive

The servo drive used for experiments serves as a general purpose automotive actuator, for example to intake manifold adjustments or for a variety of flap applications. The figure 4.5 shows its picture. The servos position is also measured with a potentiometer but with a significantly higher level of noise. Also,, it gearing has much higher backlash that the gearings of the valves.

Figure 4.5: Servo

The figure 4.4 shows a quasi-static load characteristic of the servo drive. A opposed to the other systems, it is almost linear.

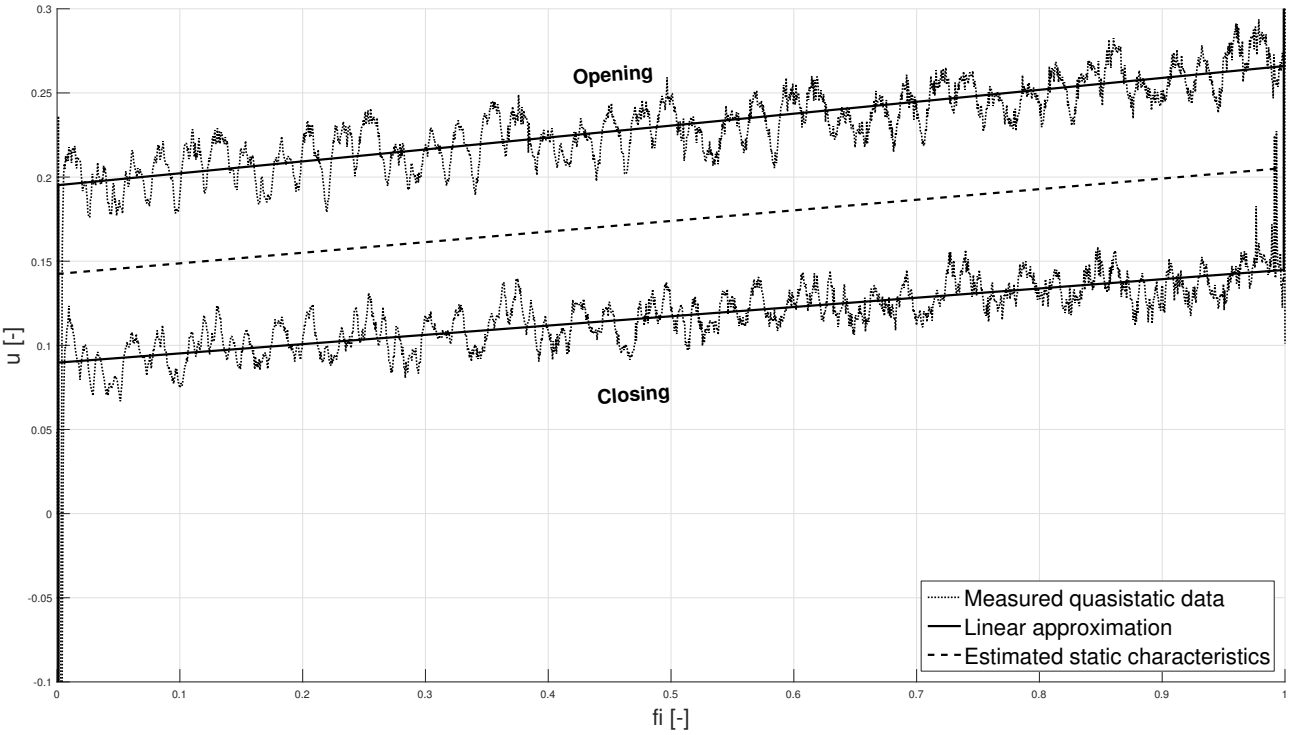

Figure 4.6: Static characteristic of the servo drive

4.2 dSPACE Implementation

To test the methods described in the previous chapters, a real-time control hardware dSPACE with a measuring card DS2201 was used. In contains an Intel core i7 processor and my common peripherals and it is also possible to program it from the Matlab/Simulink environment and inspect a running application on-line using an utility software called Control Desk.

Using these tools, the final adaptive control algorithm was designed based on the local approximation methods described in the chapter 3.

This section contains a description of the control algorithm and several experiments conducted with the automotive actuators described in the previous section 4.1

4.2.1 Adaptive Control Algorithm

The control algorithm is based on the composite control principle described in the section 2.1. It is very similar to the DC motor adaptive control algorithm, described in the section 3.4.3, except it used more than one local model.

The figure 4.7 shows a block diagram of the algorithm which was used for a rotation control of the above-described actuators. Thy angular rotation reference signal was generated as random steps with a period $\boldsymbol{T_{ref} = 0.3\,s}$ and processed by the trajectory planner.

The random steps were generated in such a way that $\mathbf{70\,\%}$ of the time the reference signal was approximately in the nonlinear region of valves. A standard P-type controller with a gain $\boldsymbol{K_P = 50}$ was also a part of the algorithm.

To calculate the necessary derivatives of the angular rotation reference signal for the Feedforward Compensation block, a simple numerical difference could be used because it was an ideal signal with no noise. However, to determine the same derivatives of the measured signal for the Local Linear Learning block, the polynomial approximation approach, described in the section 3.4.2, hat to be used. The processes are represented by the blocks Numerical Derivative and PolyDer.

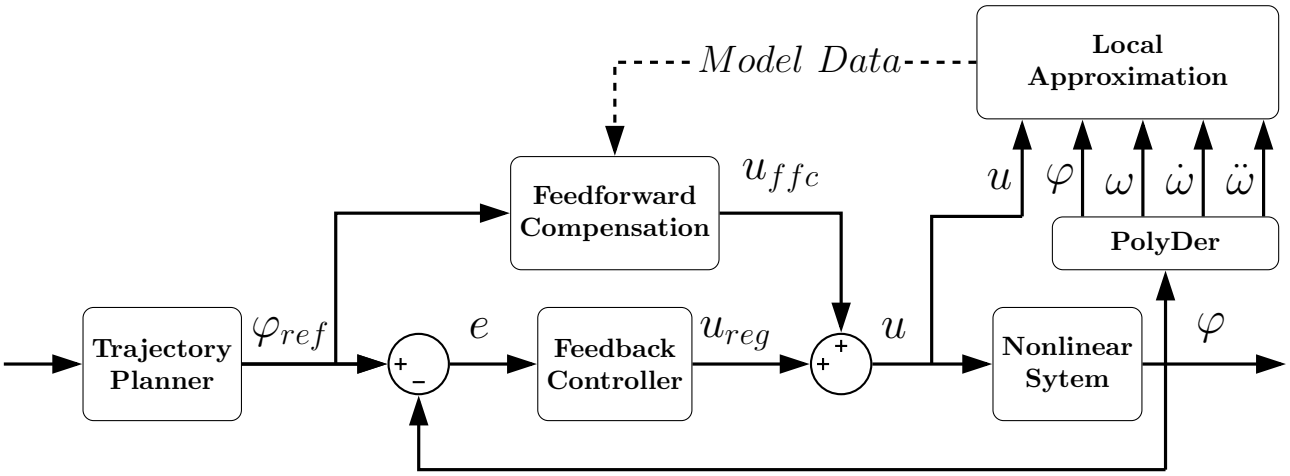

Figure 4.7: Block diagram of the Adaptive Control Algorithm

The Local Approximation Learning block represents the specific approximation method used to learn the inverse model of the controlled system. Two different learning methods were used: the RFWR method with two different model structures and the LOLIMOT method. The

actual model data are then used by the Feedforward compensation block to evaluate the inverse model and determine the compensation input $\boldsymbol{u_{FFC}}$.

Considering the equation 4.1 the inverse dynamic model of the actuators can be written as the equation 4.3. Thanks to the dimensional reduction approach described in the section 3.2.3, it can be modelled using a two-dimension local approximation, along the variables $\boldsymbol{\varphi}$ and $\boldsymbol{\omega}$, with a local model structure according to the equation 4.4.

$$u(t) = f(\varphi) + g(\omega) + b_3\dot{\omega} + b_4\ddot{\omega} \tag{4.3}$$

$$u(t) = b_1\varphi + b_2\omega + b_3\dot{\omega} + b_4\ddot{\omega} + b_5 \tag{4.4}$$

When we also consider the equation 4.2, which replaces the unknown friction function $\boldsymbol{g(\omega)}$ by a known, simpler function that meets the requirements for the RLS parameter estimation, the general inverse dynamic model of the actuators can be written as the equation 4.5. Using the same dimensional reduction approach, this inverse model can modelled by a one-dimensional local approximation, along the variable $\boldsymbol{\varphi}$, with a local model structure according to the equation 4.6.

$$u(t) = f(\varphi) + b_2\omega + b_3\dot{\omega} + b_4\ddot{\omega} + b_5 sgn(\omega) \tag{4.5}$$

$$u(t) = b_1\varphi + b_2\omega + b_3\dot{\omega} + b_4\ddot{\omega} + b_5 sgn(\omega) + b_6 \tag{4.6}$$

Both local models' structures were used in the experiments. The one-dimensional case (equation 4.6) with the RFWR, and the two-dimensional case (equation 4.4) with both LOLIMOT and RFWR, resulting in three different approximation methods (named **RFWR_1D**, **RFWR_2D** and **LOLIMOT_2D**) used to control each of the three automotive actuators.

For a comparison, the experiments were also conducted with the P-type controller which is a part of the algorithm, alone and with a PI-type controller with gains $\boldsymbol{K_P = 100}$ and $\boldsymbol{K_I = 50}$, which were acquired experimentally. All experiments were realised with a calculation period $\boldsymbol{T_s = 0.001\,s}$ for $\boldsymbol{400\,s}$, where the first $\boldsymbol{100\,s}$ were determined to the inverse model learning and the $\boldsymbol{u_{FFC}}$ signal was proportionally diminished.

Two statistical approaches were used to asses the performance of these control methods. The first was the common mean squared error (**MSE**) between the reference and the measured angular rotation to compare adaptive control algorithm using different approximation methods with the regular feedback controllers.

The second approach tried to evaluate the influence of both parts of the control algorithm - the feedback and the feedforward controller. It was assumed that if the feedforward compensation has a significantly higher influence on the control process than the feedback controller, it is the inverse model approximation is working correctly. The feedforward compensation efficiency $\boldsymbol{E_{FFC}}$ is calculated according to the equation 4.7. To express the overall efficiency during the whole experiment, the average feedforward compensation efficiency $\overline{\boldsymbol{E_{FFC}}}$ is used.

$$E_{FFC}(t) = \frac{|u_{FFC}(t)|}{|u_{FFC}(t)| + |u_P(t)|} \tag{4.7}$$

4.2.2 Experimental Results

From the measurement of every experiment, several graphical and statistical outputs can be created. The statistical summary of all experiments is in table 5.1, which compares the precision, the feedforward compensation efficiency and the number of local models used by each method with a regular P or PI type feedback controller.

Since the graphical outcomes from the experiments are extensive, this section describes in detail one of the nine experiments possible for three approximation methods and three actuators. A complete list of all experimental data can be found in the Appendix B.

The described experiment was conducted using the **RFWR_2D** as the local approximation method of the adaptive algorithm to control the Electronic throttle valve.

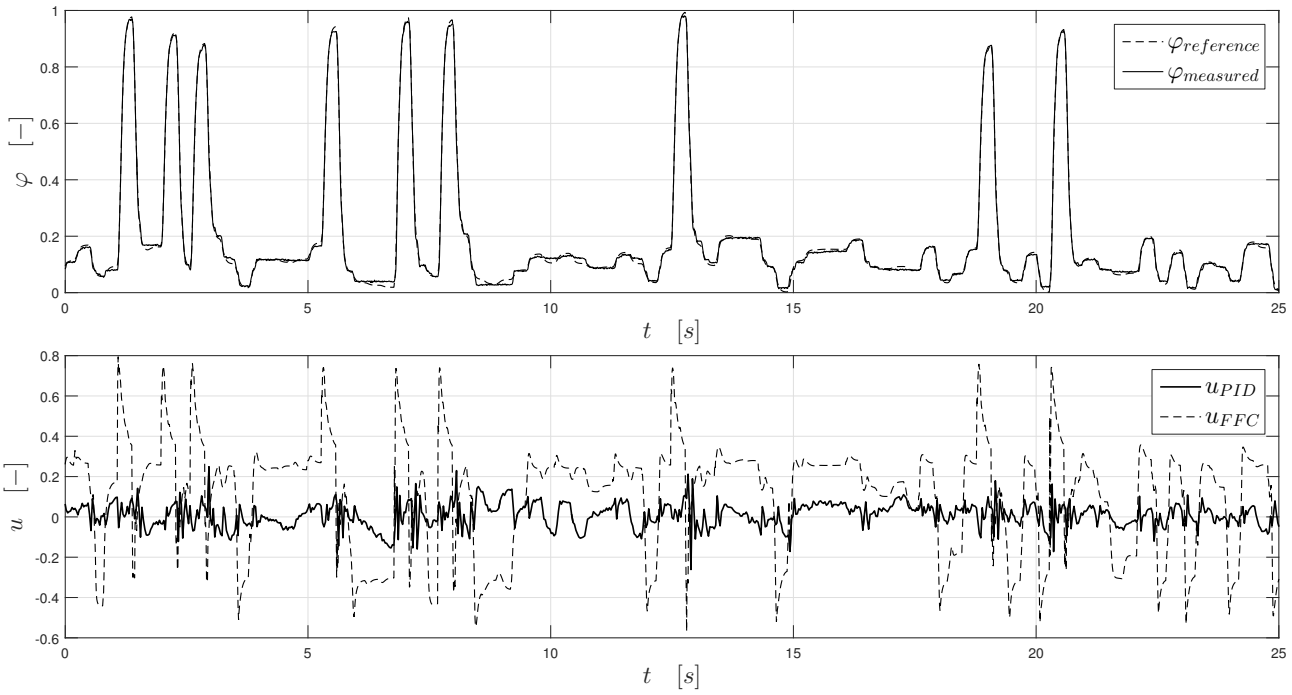

Figure 4.8: Eletronic throttle valve control using the RFWR_2D method

The most important outcome of the experiment is the comparison of the reference and the measured angular position signal. The figure 4.8 shows this comparison together with the outputs of the feedforward compensation and the P-type controller.

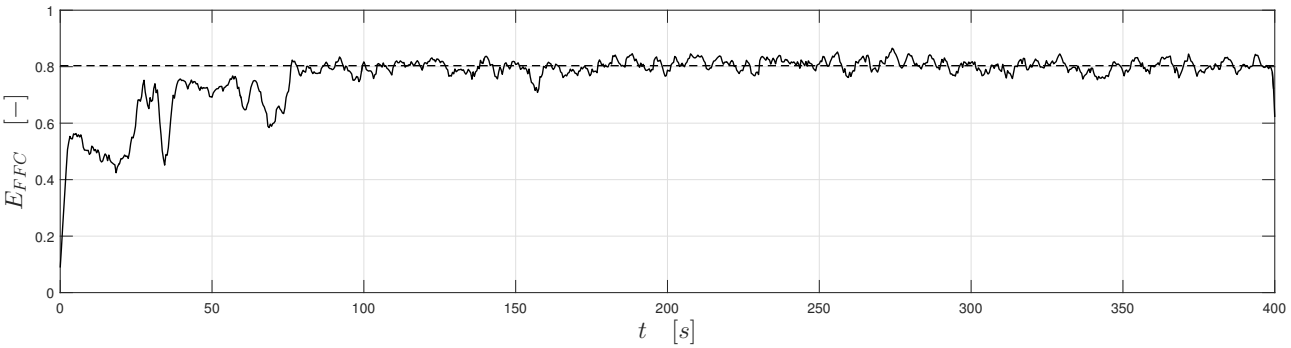

Figure 4.9: Feedforward compensation efficiency

It is clear that the feedforward compensation works very well, because the feedback controller output is much lower. To evaluate this ratio, the figure 4.9 shows the efficiency $\boldsymbol{E_{FFC}}$ of the feedforward compensation calculated according to the equation 4.7. The average efficiency after the 100 second long learning period was $\overline{\boldsymbol{E_{FFC}}} = \mathbf{80.4\,\%}$.

The efficiency signal plotted in the figure 4.9 was smoothed using a Gaussian window over a 10 second long interval.

The RFWR algorithm had two learning dimensions, which means that the inverse model representation can be easily plotted together with the models' kernel functions. The figure 4.10 shows the inverse model approximation as a function of the angular rotation $\boldsymbol{\varphi}$ and the angular velocity $\boldsymbol{\omega}$. The remaining variables $\dot{\boldsymbol{\omega}}$ and $\ddot{\boldsymbol{\omega}}$ are considered zero.

As a comparison, the figures 4.11 and 5.1 show the approximation done by the other methods.

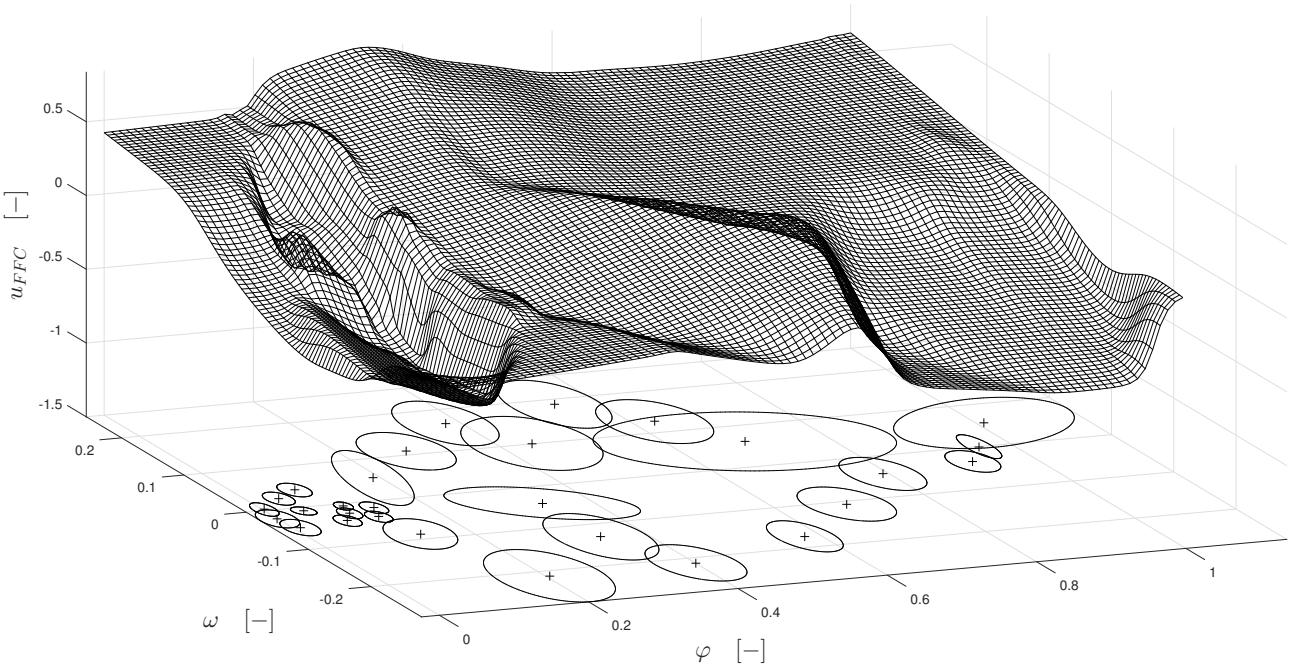

Figure 4.10: Electronic throttle valve's inverse model approximation by the RFWR_2D algorithm.

4.3 MCU Implementation

After successful implementation and testing with the dSPACE, the local approximation method were intended to be used with a dsPIC microcontroller.

After some initial experiments it was found, that it is not possible to implement any of the LS variants on the microcontroller through the automatic code generation from the Simulink environment, because of its computational complexity and mainly numerical instability.

The used microcontroller (dsPIC33FJ128MC804) has a 16-bit word length, which is not enough to calculate the matrix inversion in the equation 2.7 of the LS method precisely enough.

Also, when implementing the RLS method, it showed up that calculations with the inverse covariance matrix from the equation 2.9 are not precise, because its elements are usually by many orders of magnitude different than the system inputs and outputs. This leads to an

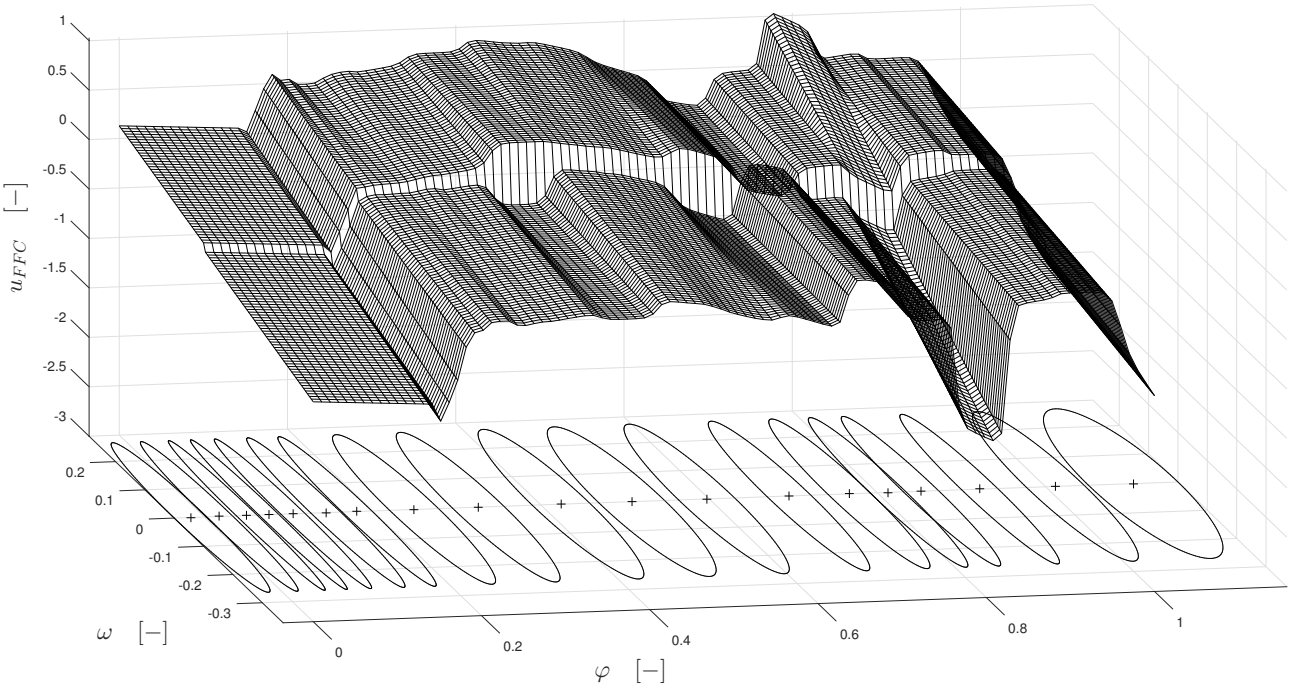

Figure 4.11: Electronic throttle valve's inverse mode approximation by the RFWR_1D algorithm.

instability of the calculation.

Using a 32-bit fixed-point arithmetic was also not possible, because of its computational demands when implementing on a 16-bit chip.

For those reasons, a different local approximation method was developed. It woks like a simple grid based Look-Up table with linear interpolation (see section 2.3.1 or [1] for further description), which was adjusted by a statistical algorithm, which adds new points to the grid similarly to the LOLIMOT.

The new grid point is added between the two existing points with the highest statistical deviation from the measured data, up to reaching 50 grid points. Thanks to the dimensional reduction approach, the modelled systems could be approximated by local models with a structure according to the equation 4.6, which means the table grid is only one-dimensional along the variable $\boldsymbol{\varphi}$.

Parameters $\boldsymbol{b_1}$ and $\boldsymbol{b_6}$ are calculated using the linear interpolation between the table points and the remaining parameters are the same for the whole system.

This local approximation method (named **LUT**) was used to control the three actuators during the same position control experiment as the methods in the previous section. The Figure 4.12 the control process and the comparison of the feedforward and the feedback parts' outputs of the adaptive control algorithm. The statistical results are also listed in the table 5.1.

The figure 4.13 shows the Electronic throttle valve's inverse model approximation along the grid dimension φ with the other variables considered zero. This very simple local approximation method was also able to learn the inverse model precisely enough to maintain stable position control of the throttle valve.

Results from the remaining experiments (EGR valve and Servo drive) are also listed in the table 5.1 and in the Appendix B.

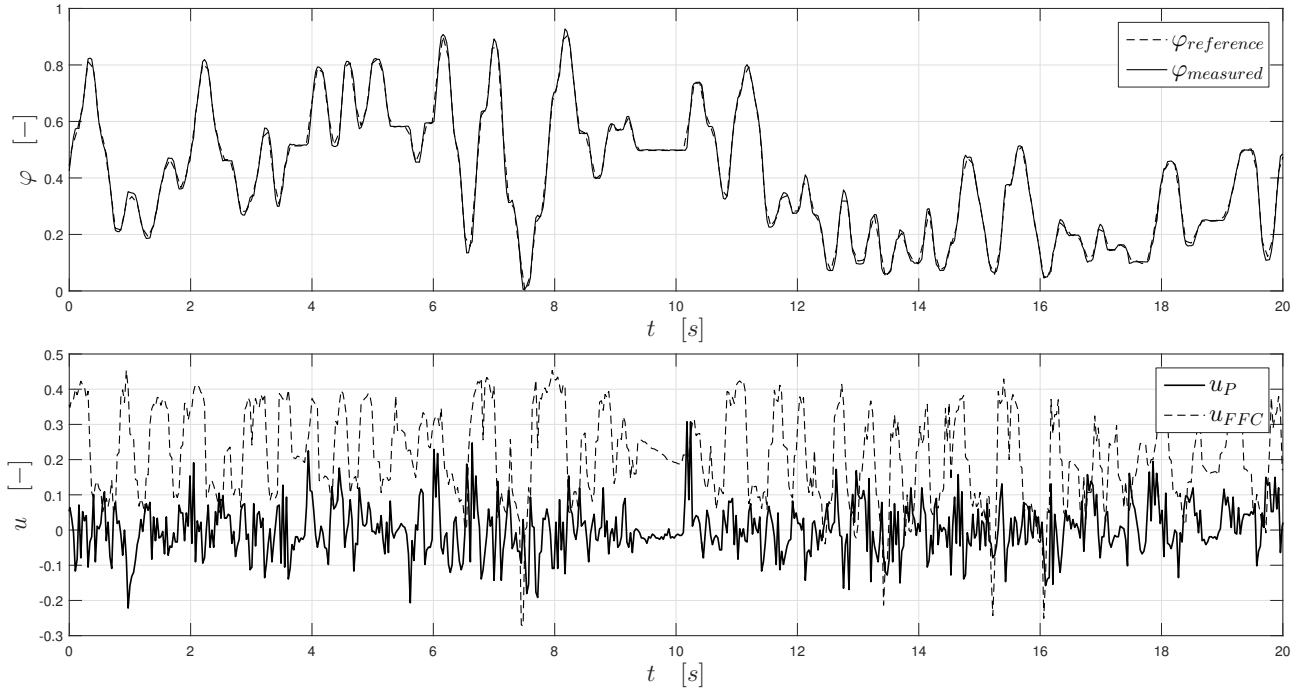

Figure 4.12: Eletronic throttle valve control using the LUT method

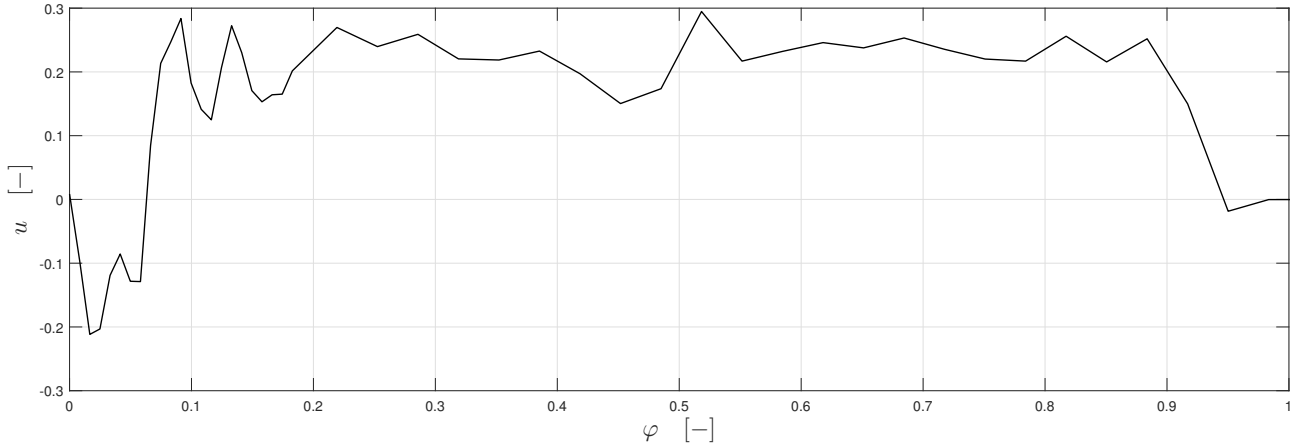

Figure 4.13: Electronic throttle valve's inverse mode approximation by the LUT algorithm.

5 Conclusion

The outcome of this thesis is a design of an adaptive control algorithm, which is based on the composite control principle and uses an inverse dynamic model of the controlled system as a feedforward dynamics compensator.

The inverse dynamic model of the controlled system is acquired using a local approximation method and the algorithms adaptivity is based on the approximation methods ability to learn and improve the dynamic's approximation on-line, using a measured I/O data.

The algorithm was tested using the Simulink environment and then during actual experiments with various automotive actuators, servo drives and valves, with a nonlinear positional feedback, using a dSpace real-time control hardware.

Additionally, a simplified version of the adaptive control algorithm was implemented on an standard I/O board with a dsPIC microcontroller using automatic code generation from the Matlab/Simulink environment.

The adaptive algorithm can use three different local approximation methods, a simple look-up table with linear interpolation for the microcontroller implementation, and adjusted versions of standard method RFWR and LOLIMOT for the dSPACE implementation.

The conducted experiments are described in the chapter 4. The results show that the local approximation methods in general have a potential to create an inverse dynamic model of a system precisely enough to successfully control ty system.

The figure 5.1 shows the inverse model approximation of the Electronic throttle valve using the LOLIMOT algorithm during a real-time learning and control experiment.

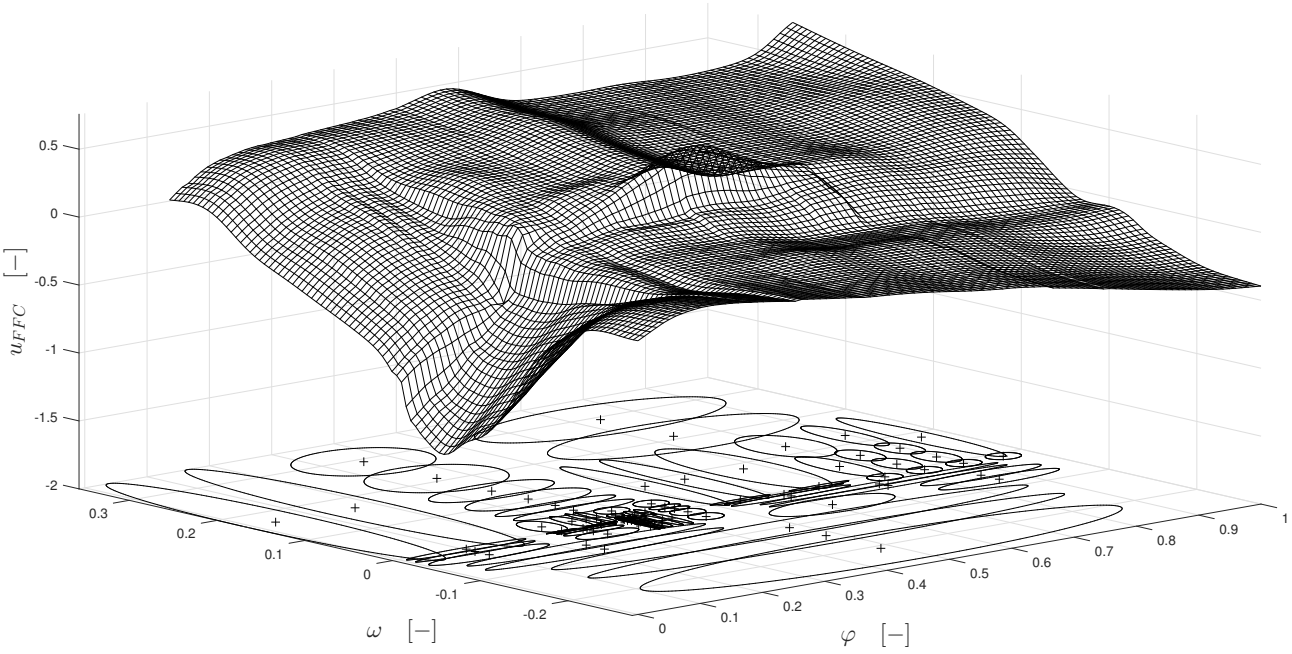

Figure 5.1: Electronic throttle valve's inverse mode approximation by the LOLIMOT_1D algorithm.

Not even a significant dry friction which takes place in the used actuators and represents a severe nonlinearity, prevents the algorithm from maintaining a stable control.

The adaptive algorithm typically takes several tens of seconds to learn the controlled system's dynamics but even after that period, it is able to adjust the model if needed.

The effectiveness of the algorithm was determined in comparison with a feedback controller. It is based on an assumption, that if the inverse model is approximated correctly, the influence of the feedforward compensation during a control process if significantly higher, than that of the feedback controller which is also a part of the composite control algorithm.

A summary of the experimental results is listed in the table 5.1. The experiments are compared according to the average efficiency $\overline{\boldsymbol{E_{FFC}}}$, the control process mean squared error $\boldsymbol{MSE}$, and the number of local models $\boldsymbol{N}$. All tested variants achieved much more precise control than the P-type and the PI-type controllers alone. The P-type controller is otherwise a part of the adaptive composite control algorithm.

	Electronic Throttle Valve			**EGR Valve**			**Servo Drive**		
Algorithm	$\overline{\boldsymbol{E_{FFC}}}$	$\boldsymbol{MSE}$	$\boldsymbol{N}$	$\overline{\boldsymbol{E_{FFC}}}$	$\boldsymbol{MSE}$	$\boldsymbol{N}$	$\overline{\boldsymbol{E_{FFC}}}$	$\boldsymbol{MSE}$	$\boldsymbol{N}$
RFWR_1D	81.3 %	$3.07 \cdot 10^{-4}$	21	76.9 %	$1.41 \cdot 10^{-4}$	18	86.1 %	$0.96 \cdot 10^{-4}$	10
RFWR_2D	80.4 %	$1.23 \cdot 10^{-4}$	38	71.9 %	$6.04 \cdot 10^{-4}$	21	83.4 %	$1.00 \cdot 10^{-4}$	16
LOLIMOT_2D	81.6 %	$0.94 \cdot 10^{-4}$	81	78.2 %	$1.21 \cdot 10^{-4}$	81	85.2 %	$0.55 \cdot 10^{-4}$	82
LUT	73.0 %	$1.77 \cdot 10^{-4}$	50	63.9 %	$15.1 \cdot 10^{-4}$	50	65.9 %	$4.12 \cdot 10^{-4}$	10
P Controller	–	$21.5 \cdot 10^{-4}$	–	–	$24.1 \cdot 10^{-4}$	–	–	$16.6 \cdot 10^{-4}$	–
PI Controller	–	$13.3 \cdot 10^{-4}$	–	–	$22.3 \cdot 10^{-4}$	–	–	$8.74 \cdot 10^{-4}$	–

Table 5.1: Summary of experimental results

The differences between the local approximation methods are mainly in their computational complexity, number of user defined parameters and an overall stability.

During the simulations and the experimental tests of the designed control algorithm or its separate parts, it was also necessary to solve several practical problems, which led to some interesting findings.

Most of all, the solution of determining several time derivatives of a measured noisy signal using an approximation polynomial and its analytical derivatives, described in the section 3.4.2. In comparison to the usual method, i.e. using a lowpass filtration and numerical difference, it was shown to be much more effective, especially in determining higher order derivatives of the same signal.

It is expected, that it could also be used in many other real situations that need to calculate the numerical derivative of a noisy signal.

Another outcome of this thesis, which was originally intended to be a simplified experiment to evaluate the basic principles of local linear approximation and mainly using the Recursive Least Squares method (RLS) for parameter estimation, is an adaptive control algorithm for a DC motor which is described in the section 3.4.

The RFWR algorithm that generally uses many local models to approximate a system's dynamics was simplified in a way that uses only one local model with various structures described in the section 3.4.4, which are used as a feedforward compensator for a DC motor position

control. The various model structures are able to compensate the linear DC motor dynamics and also some nonlinear effects like a dry friction or an unbalanced inertia.

The simplification, using only one model, allows to test the RLS parameter estimation stability and an overall susceptibility to noise and the inverse model imperfections, without having to deal with optimizing the local models' distribution, which is the most complicated part.

Beside the result, that when the above-mentioned solution for signal derivatives is used, the whole system shows very promising results, it also turns out to be a very effective way to control an arbitrary DC motor with state dependent torque load.

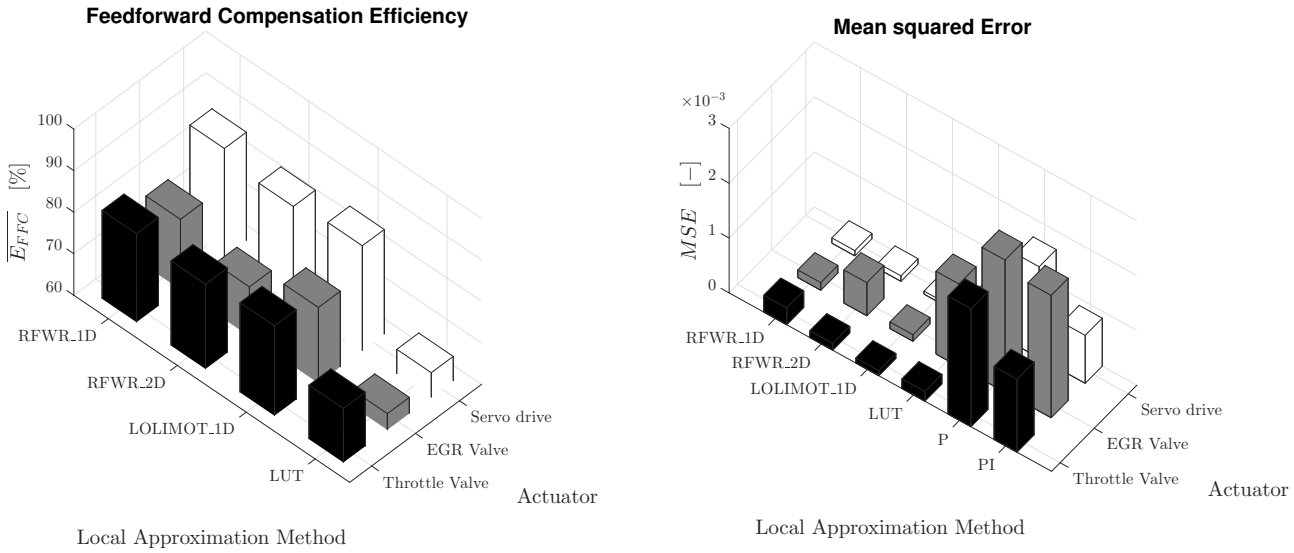

Figure 5.2: Visual comparison of the experimental results

Another significant outcomes resulting from the analysis of various local approximation methods, described mainly in the chapter 3, are several adjustments to the original methods so that they can be used for a real-time approximation, so that they optimise the distribution of local model solely according to such system states which have the potential to contain a nonlinearity.

This approach, described in the section 3.2.3, helped to significantly lower the computational complexity and increase the overall algorithm stability by reducing the number of local models needed to approximate a system's inverse dynamics.

The final outcome of this thesis which is worth mentioning are the other adjustments made to the RFWR algorithm described in the section 3.2. They are about making the algorithm more stable and effective when placing new local models and optimising the existing ones.

In summary, the local approximation methods seem to be a promising tool for control and analysis of nonlinear dynamic systems, which can have an important position in fields like Mechatronics and Robotics in the future, thanks to the recent improvements in the performance and price of industrial microcontrollers.

5.1 Suggestions for Further Development

For further developments in the field of local approximation methods, it would be appropriate to test the adaptive control algorithm proposed in this thesis with different dynamic systems.

Some of the educational models available in the Mechatronics laboratory, for example a rotational inverse pendulum, or a magnetic levitation of an iron ball, are suitable for this task.

Also, the whole algorithm could be implemented on a microcontroller directly through the C language, which could further decrease the computational complexity of the algorithms, or a microcontroller with a FPU unit could be used.

Another possibility for further experiments is the adaptive control algorithm for a DC motor described in the section 3.4. It can be improved in many ways, like using the current measurement, estimating the torque load with a state estimator like the Kalman filter or extend the whole algorithm to be used with a BLDC motor, which is beginning to be used in the industry more often because of its almost maintenance-free operation.

Further development could be also devoted to the algorithm to determine a noisy signal's derivative using an approximation polynomial analytic differentiation described in the section 3.4.2. For example, its implementation on an FPGA would be very interesting.

Bibliography

[1] NELLES, Oliver. *Nonlinear system identification: from classical approaches to neural networks and fuzzy models*. New York: Springer, c2001. ISBN 35-406-7369-5.

[2] ATKESON, Christopher G., Andrew W. MOORE a Stefan SCHAAL. Locally weighted learning. *Artificial Intelligence Review* [online]. 1996, **11**(1/5), 11-73 [cit. 2016-04-28]. DOI: 10.1023/A:1006559212014. ISSN 02692821. url: http://link.springer.com/10.1023/A:1006559212014

[3] ATKESON, Christopher G., Andrew W. MOORE a Stefan SCHAAL. Locally weighted learning for control. *Artificial Intelligence Review* [online]. 1997, **11**(1/5), 75-113 [cit. 2016-04-28]. DOI: 10.1023/A:1006511328852. ISSN 02692821. url: http://link.springer.com/10.1023/A:1006511328852

[4] ENGLERT, Peter. *Locally Weighted Learning* [online]. Darmstadt, Germany, 2012, , 9 [cit. 2016-04-28]. url: http://www.ausy.informatik.tu-darmstadt.de/uploads/Teaching/AutonomousLearningSystems/Englert_ALS_2012.pdf

[5] SCHAAL, S., S. VIJAYAKUMAR a C.G. ATKESON. *Local dimensionality reduction, Advances in neural information processing systems 10: proceedings of the 1997 conference ; [presented at the Eleventh Annual Conference on Neural Information Processing (NIPS), held in Denver, Colorado from December 1 to December 6, 1997]* [online]. Electronic version. Cambridge, Mass. [u.a.]: MIT Press, 1998, s. 633-639 [cit. 2016-04-28]. ISBN 0262100762. url: http://papers.nips.cc/paper/1387-local-dimensionality-reduction.pdf

[6] SCHAAL, Stefan a Christopher G. ATKESON. Constructive Incremental Learning from Only Local Information. *Neural Computation* [online]. 1998, **10**(8), 2047-2084 [cit. 2016-04-28]. DOI: 10.1162/089976698300016963. ISSN 08997667. url:https://pdfs.semanticscholar.org/0d1d/0b6e29e3b7d6b357537b9e1908b852a37a0e.pdf

[7] BIRATTARI, Mauro a Gianluca BONTEMPI. *The Lazy Learning Toolbox* [online]. 1999, , 30 [cit. 2016-04-28]. DOI: 10.1.1.45.3853. url: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.45.3853

[8] GREPL, Robert. Adaptive composite control of electronic throttle using local learning method. In: *2010 IEEE International Symposium on Industrial Electronics* [online]. Brno, the Czech Republic: IEEE, 2010, s. 58-61 [cit. 2016-04-28]. DOI: 10.1109/ISIE.2010.5637899. ISBN 9781424463909. url: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5637899

[9] ZHAO, Y. a J.A. FARRELL. A Locally Weighted Learning Method for Online Approximation Based Control. In: *Proceedings of the 44th IEEE Conference on Decision and Control* [online]. Seville, Spain: IEEE, 2005, s. 2694-2701 [cit. 2016-04-28]. DOI: 10.1109/CDC.2005.1582570. ISBN 0780395670. url: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1582570

[10] SU, J., J. WANG a Y. XI. Incremental Learning With Balanced Update on Receptive Fields for Multi-Sensor Data Fusion. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)* [online]. 2004, **34**(1), 659-665 [cit. 2016-04-28]. DOI: 10.1109/TSMCB.2002.806485. ISSN 10834419. url: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1262536

[11] NAKANISHI, Jun, Jay A. FARRELL a Stefan SCHAAL. Composite adaptive control with locally weighted statistical learning. *Neural Networks* [online]. 2005, **18**(1), 71-90 [cit. 2016-04-28]. DOI: 10.1016/j.neunet.2004.08.009. ISSN 08936080. url: http://linkinghub.elsevier.com/retrieve/pii/S0893608004001728

[12] SCHAAL, Stefan, Christopher G. ATKESON a Sethu VIJAYAKUMAR. Scalable techniques from nonparametric statistics for real time robot learning. *Applied Intelligence* [online]. 2002, **17**(1), 49-60 [cit. 2016-04-28]. DOI: 10.1023/A:1015727715131. ISSN 0924669x. url: http://link.springer.com/10.1023/A:1015727715131

[13] VIJAYAKUMAR, Sethu, Aaron D'SOUZA a Stefan SCHAAL. Incremental Online Learning in High Dimensions. *Neural Computation* [online]. 2005, **17**(12), 2602-2634 [cit. 2016-04-28]. DOI: 10.1162/089976605774320557. ISSN 08997667. url: http://www.mitpressjournals.org/doi/abs/10.1162/089976605774320557

[14] LAMBRECHTS, Paul, Matthijs BOERLAGE a Maarten STEINBUCH. *Trajectory Planning and Feedforward Design for High Performance Motion Systems*. Evanston, Ill: American Automatic Control Council, 2004. ISBN 0780383354.

[15] ORMONEIT, Dirk a Trevor HASTIE. *Optimal Kernel Shapes for Local Linear Regression: Advances in neural information processing systems*. 12. Cambridge, Mass. [u.a.]: MIT Press, 2000. ISBN 0262194503.

[16] BROWN, Peter A. Ellipses and Linear Algebra. In: *Department of Mathematics, University of Washington* [online]. Washington, 2001 [cit. 2016-05-13]. url: https://www.math.washington.edu/ king/coursedir/m308a01/Projects/m308a01-pdf/brown.pdf

[17] ASTROM, K.J., H. OLLSON, C.C. de WIT, M. GAVFERT a P. LISCHINSKY. *Friction Models and Friction Compensation* [online]. 1998, , 37 [cit. 2016-05-16]. url: http://cats-fs.rpi.edu/ wenj/ECSE446S06/astrom_friction.pdf

List of Abbreviations

LWL	Locally Weighted Learning
LS	Least Squares Method
RLS	Recursive Least Squares Method
PLS	Partial Least Squares Method
RFWR	Receptive Field Weighted Regression
LWPR	Locally Weighted Projection Regression
LOLIMOT	Local Linear Model Tree
EGR	Exhaust Gas Recirculation
PID	Proportional-Integrational-Derivative controller
SISO	Single-input Single-output system
SNR	Signal to Noise Ration
FFC	Feedforward Compensation
FPU	Floating Point Unit
MSE	Mean Square Error
DC	Direct Current
MCU	MicroController Unit

List of Figures

List of Tables

Appendix

A Electronic Appendixes

This thesis contains electronic appendixes:

- **01_Local_Learning_Methods**
  Contains the Matlab implementation of the LWL, RFWR and LOLIMOT method, along with its simulations.
- **02_DC_motor_RLS_Learning**
  This folder contains several Simulink files for controlling a DC motor using the MF624 I/O card with different model structures, as described in the section 3.4.1.
- **03_dSPACE_Loc_Learn_Control**
  The dSPACE implementation of the local approximation methods for control of the actuators using Simulink.
- **04_EduKit_Loc_Learn_Control Implementation**
  The microcontroller implementation of the local approximation methods for control of the actuators using Simulink.
- **05_Photos**
  Photos of the actuators, microcontroller board, and power H-bridge.

B Experimental Data

The following list contains a visualization of the data measured during all of the experiments described in the chapter 4.

I. Actuator: **Electronic Throttle Valve** Approximation Method: **RFWR_1D**

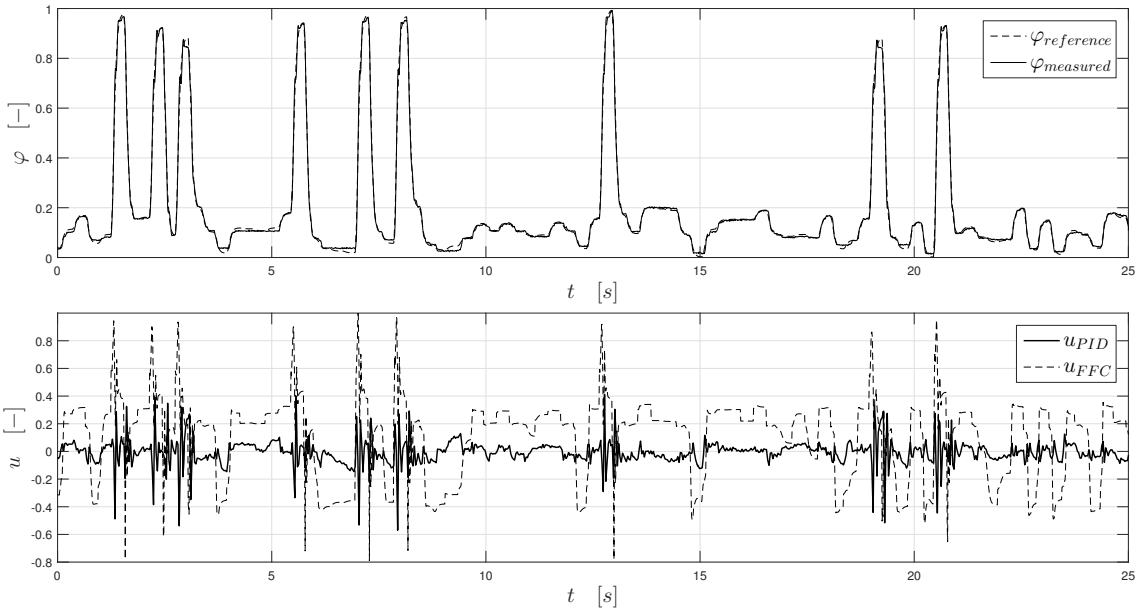

Figure 5.3: Electronic throttle valve control using the RFWR_1D method

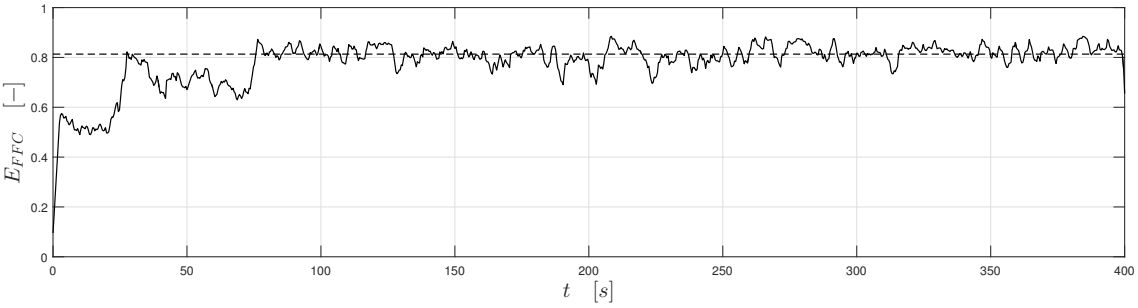

Figure 5.4: Feedforward compensation efficiency

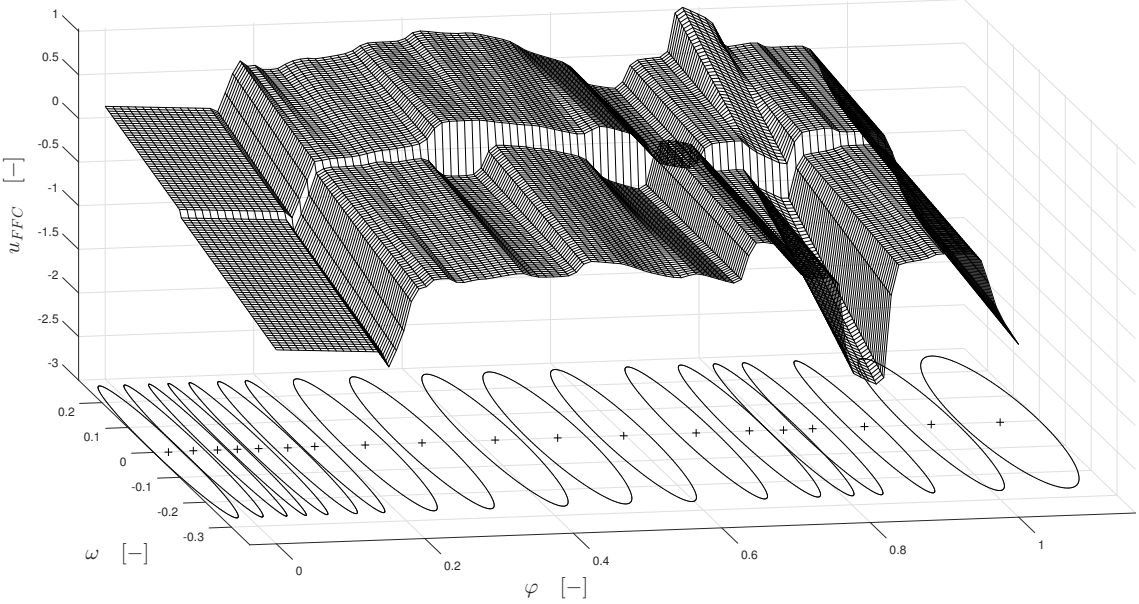

Figure 5.5: Electronic throttle valve's inverse model approximation by the RFWR_1D algorithm.

II. Actuator: **Electronic Throttle Valve** Approximation Method: **RFWR_2D**

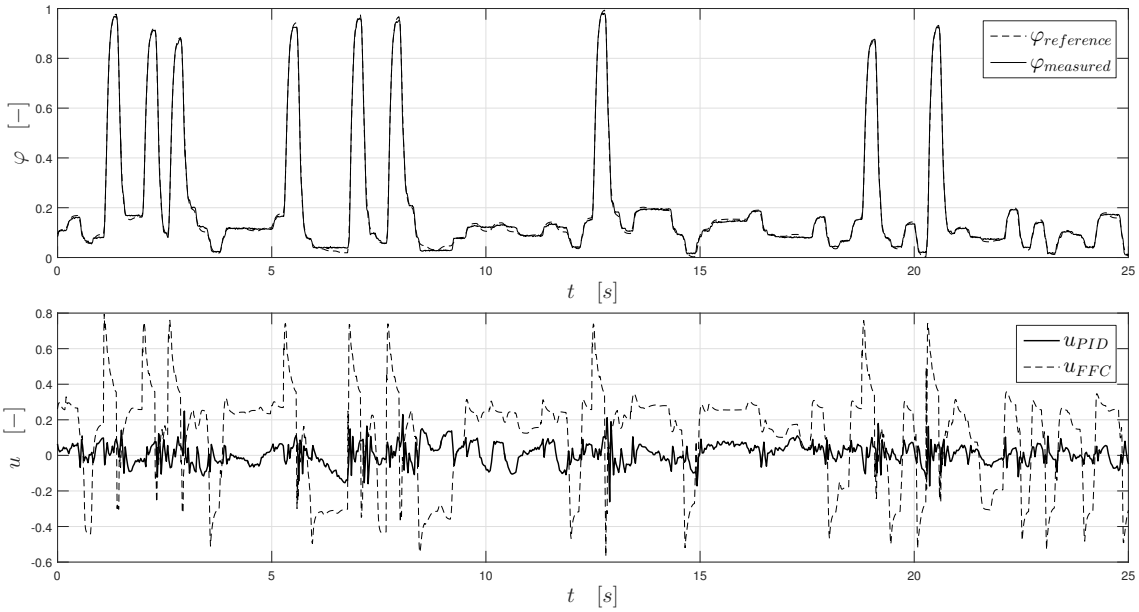

Figure 5.6: Electronic throttle valve control using the RFWR_2D method

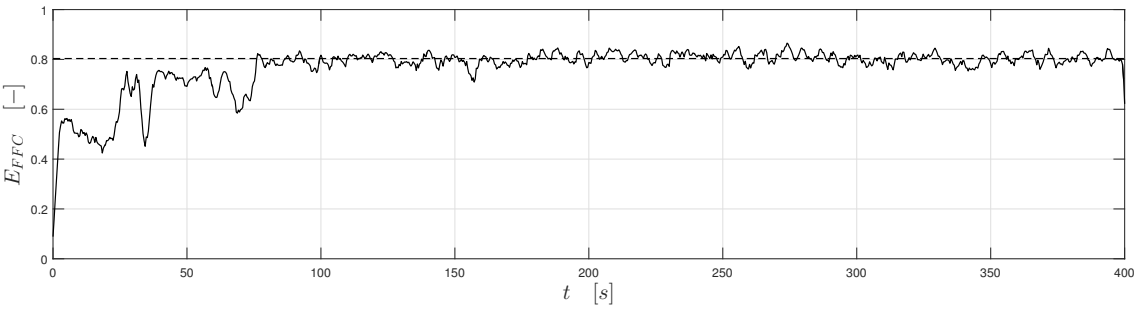

Figure 5.7: Feedforward compensation efficiency

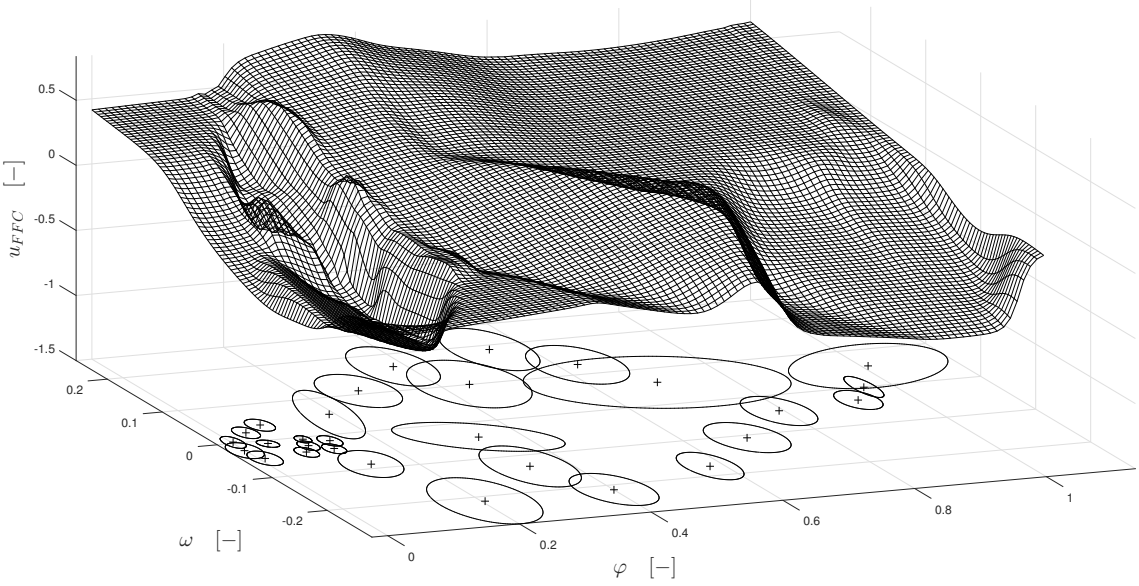

Figure 5.8: Electronic throttle valve's inverse model approximation by the RFWR_2D algorithm.

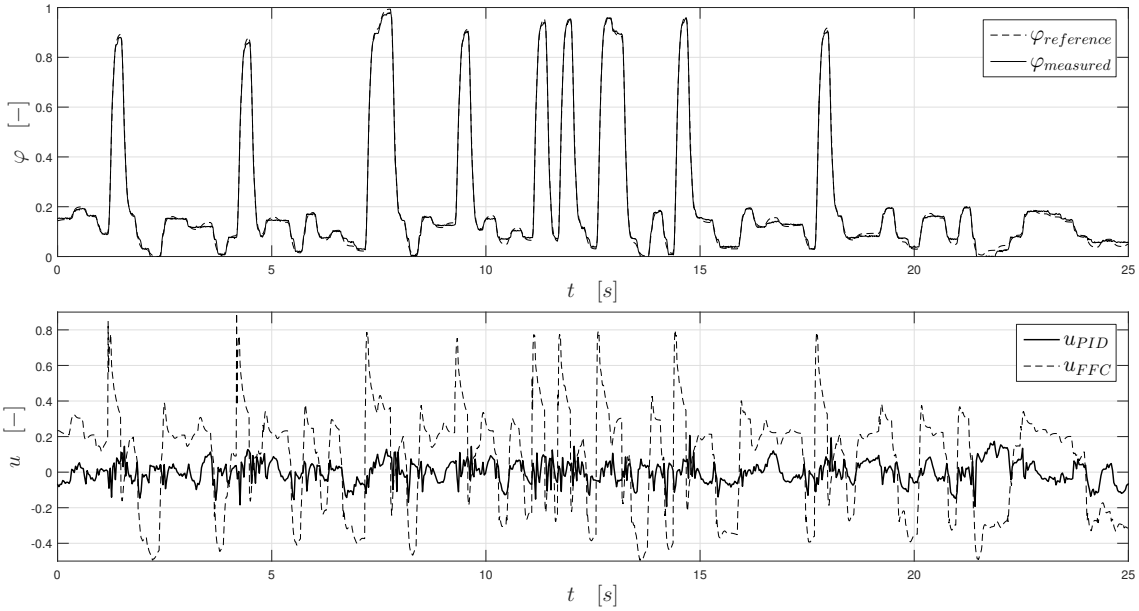

Figure 5.9: Electronic throttle valve control using the LOLIMOT_2D method

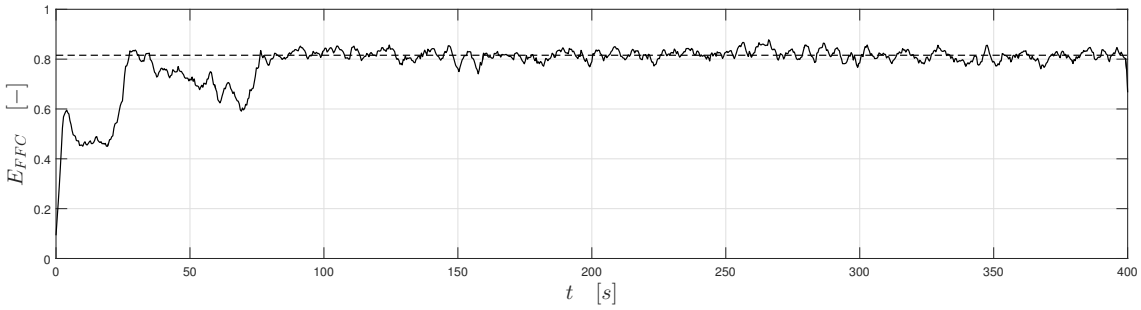

Figure 5.10: Feedforward compensation efficiency

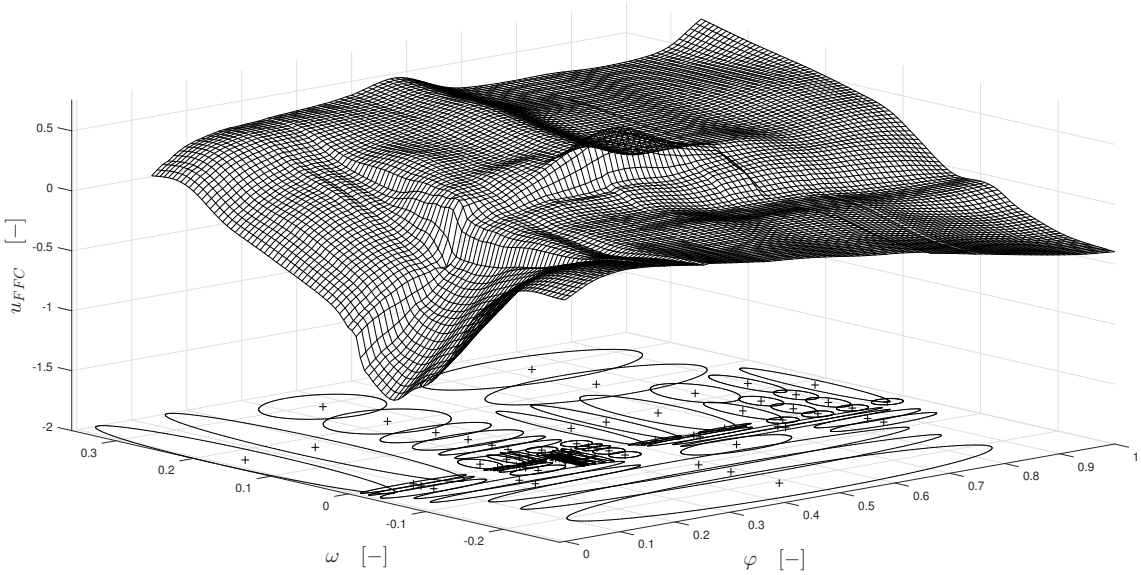

Figure 5.11: Electronic throttle valve's inverse model approximation by the LOLIMOT_2D algorithm.

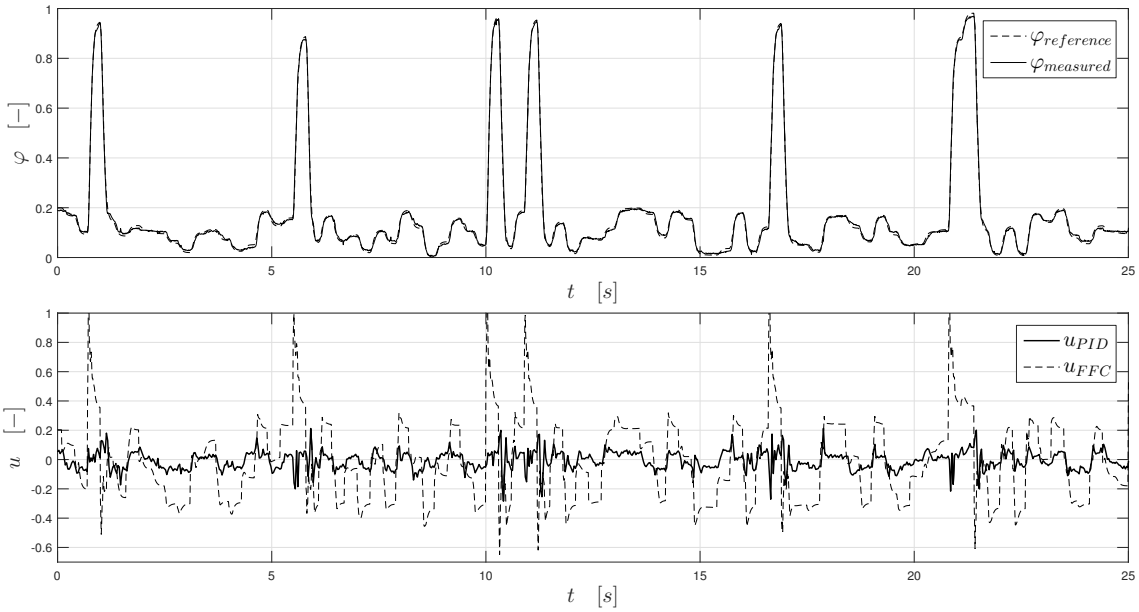

Figure 5.12: EGR valve control using the RFWR_1D method

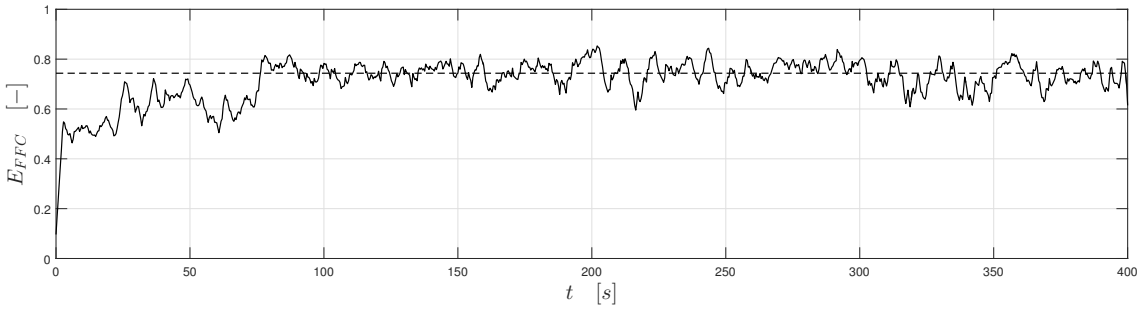

Figure 5.13: Feedforward compensation efficiency

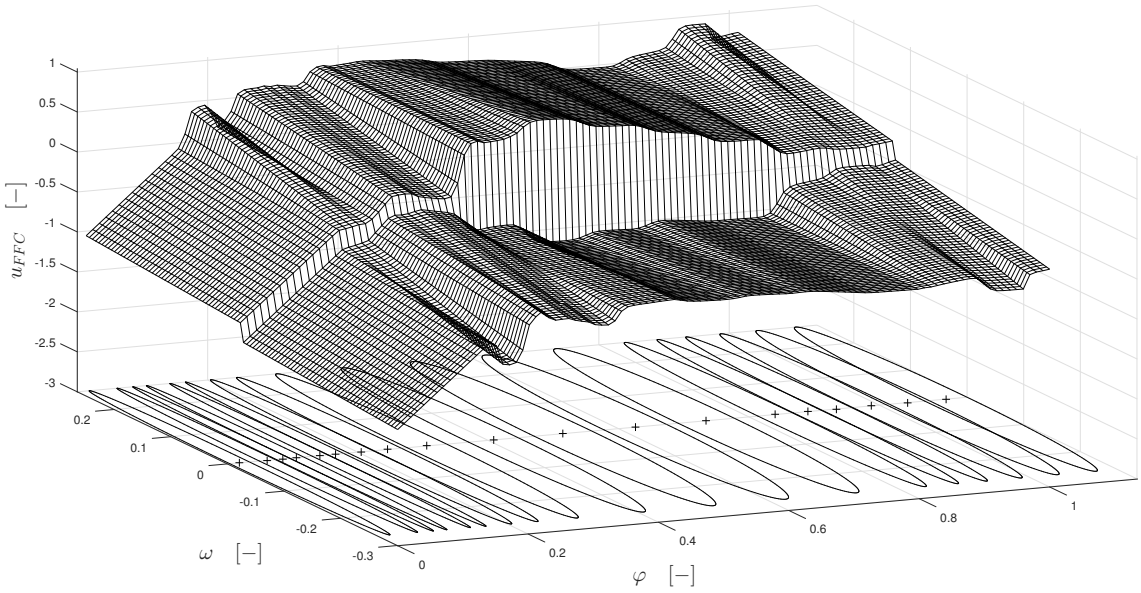

Figure 5.14: EGR valve's inverse model approximation by the RFWR_1D algorithm.

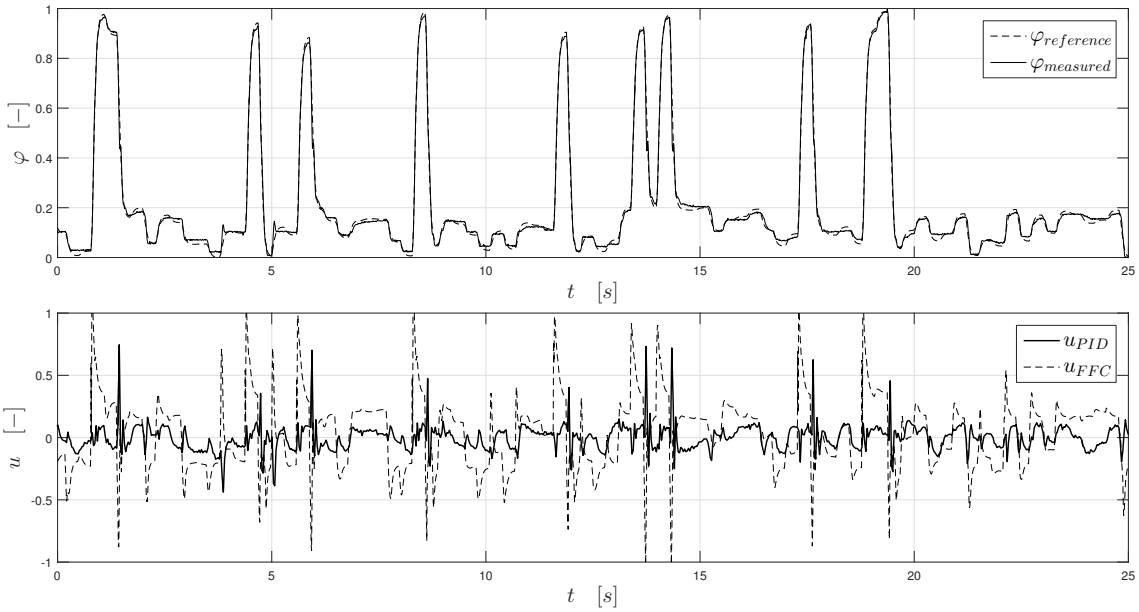

Figure 5.15: EGR valve control using the RFWR_2D method

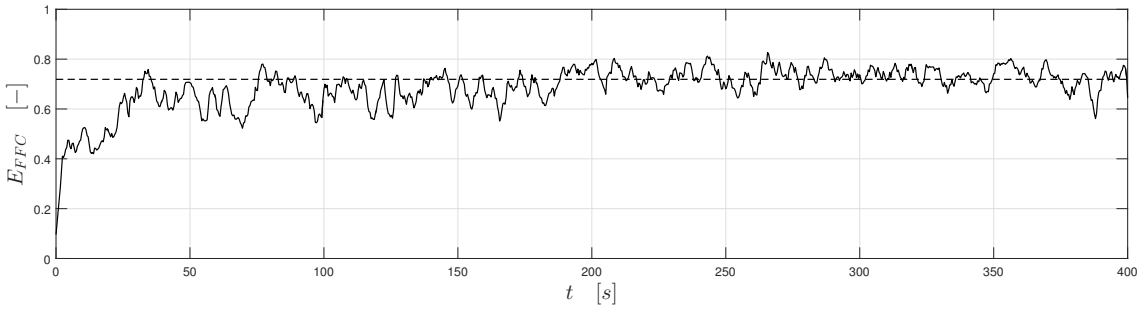

Figure 5.16: Feedforward compensation efficiency

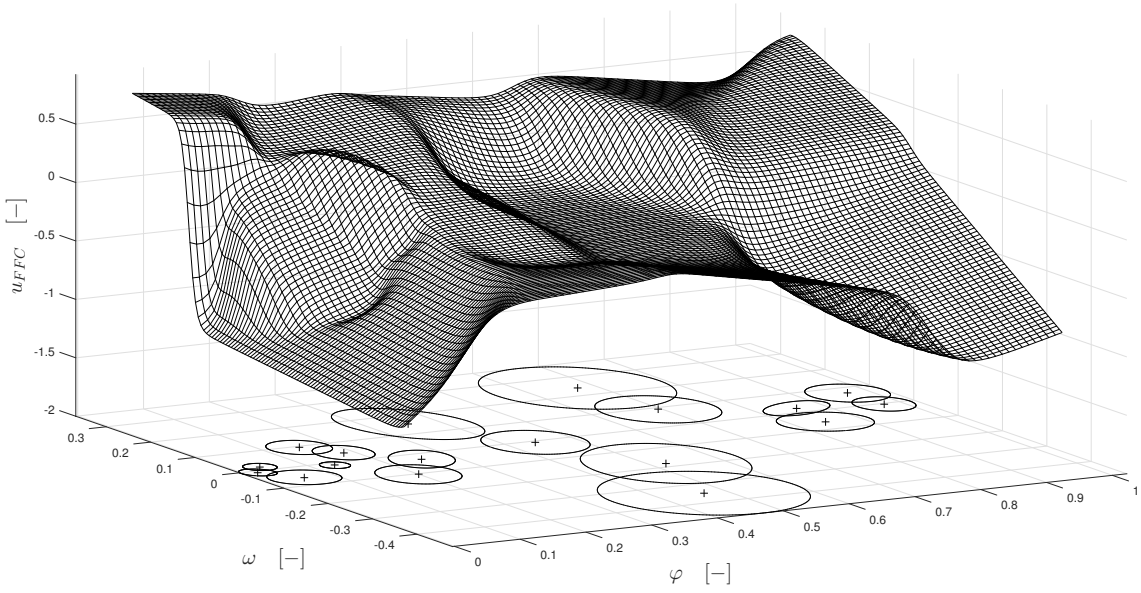

Figure 5.17: EGR valve's inverse model approximation by the RFWR_2D algorithm.

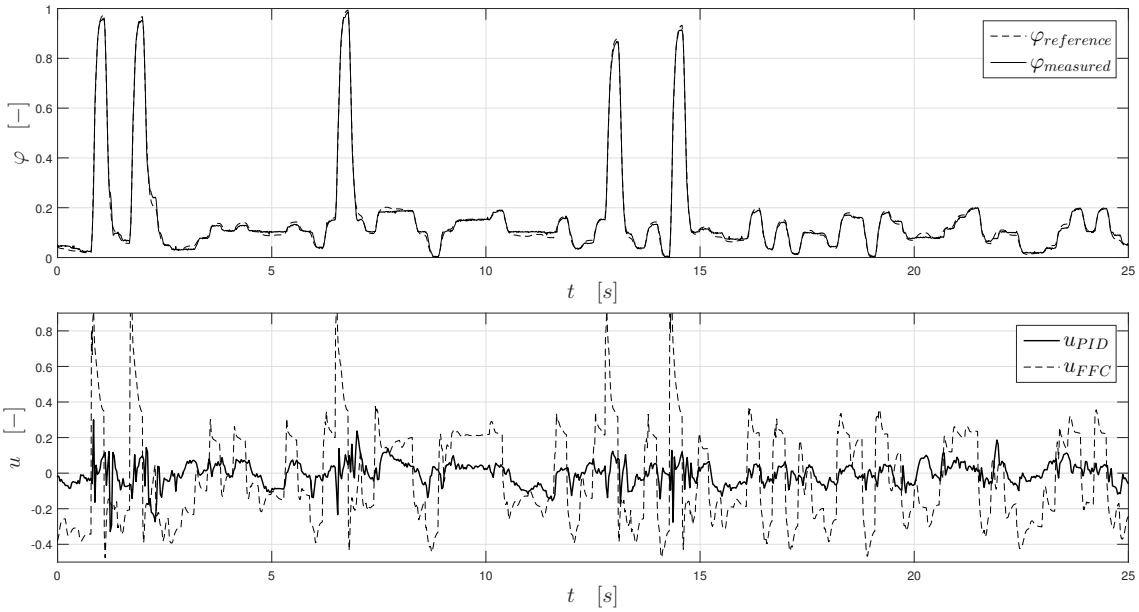

Figure 5.18: EGR valve control using the LOLIMOT_2D method

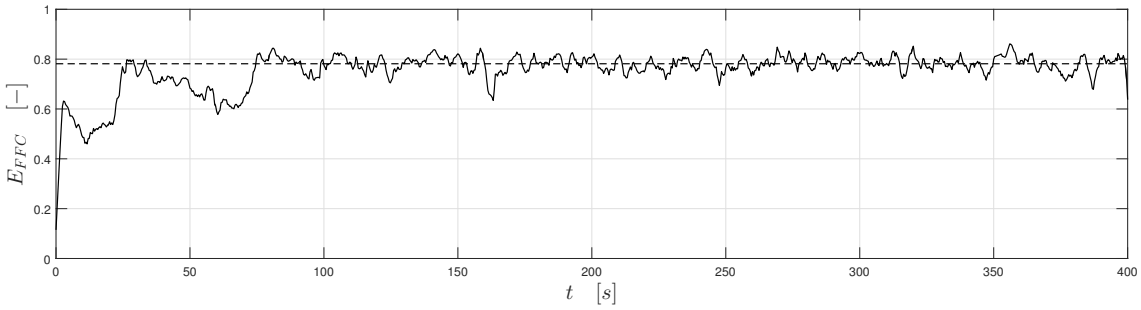

Figure 5.19: Feedforward compensation efficiency

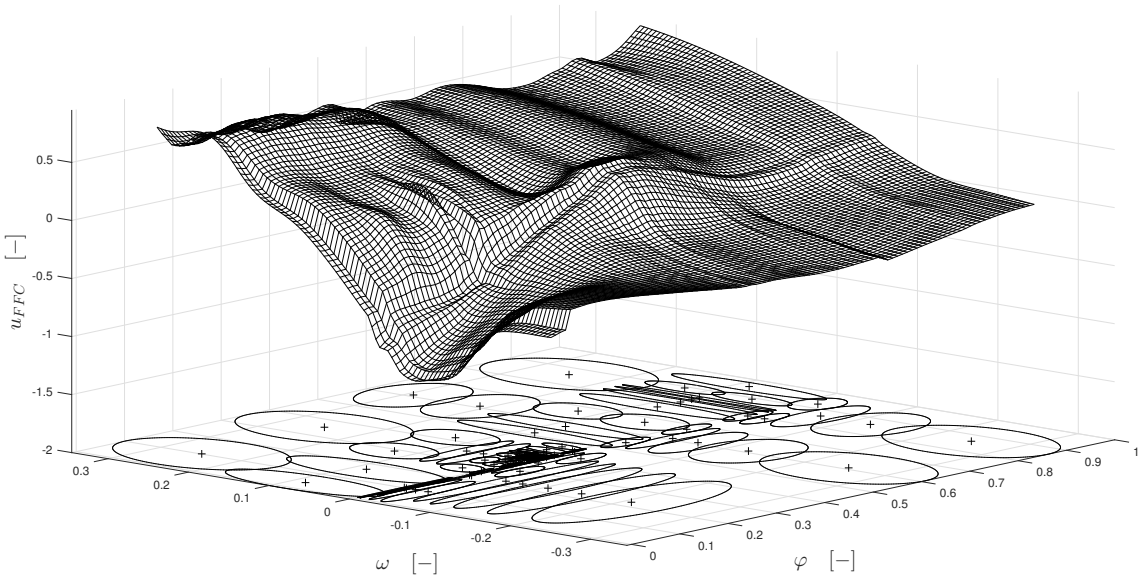

Figure 5.20: EGR valve's inverse model approximation by the LOLIMOT_2D algorithm.

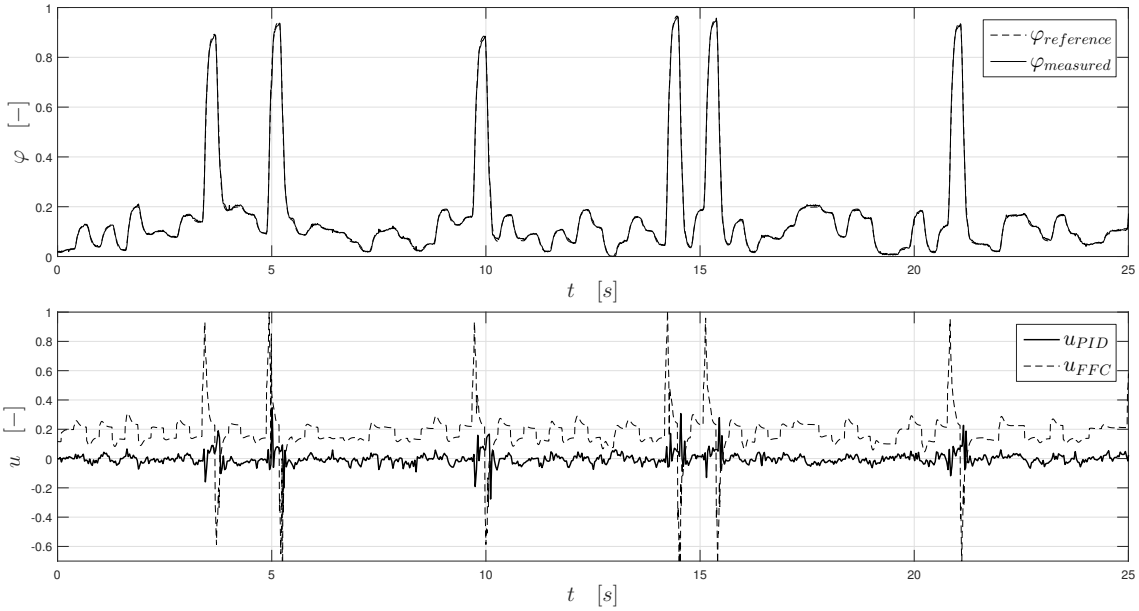

Figure 5.21: Servo drive valve control using the RFWR_1D method

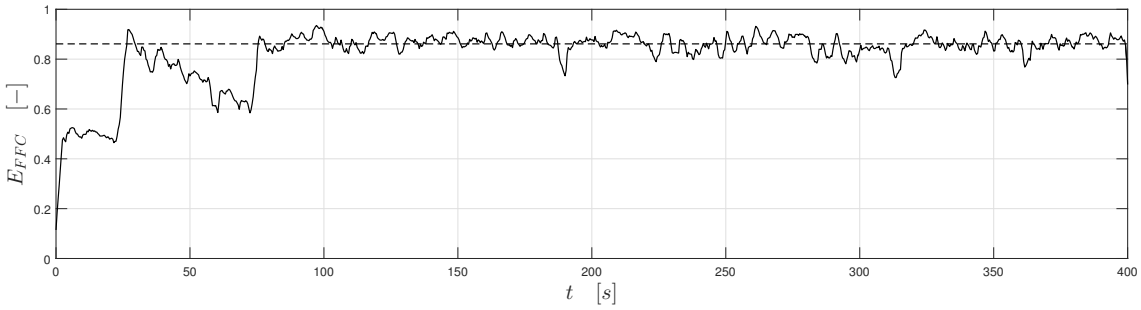

Figure 5.22: Feedforward compensation efficiency

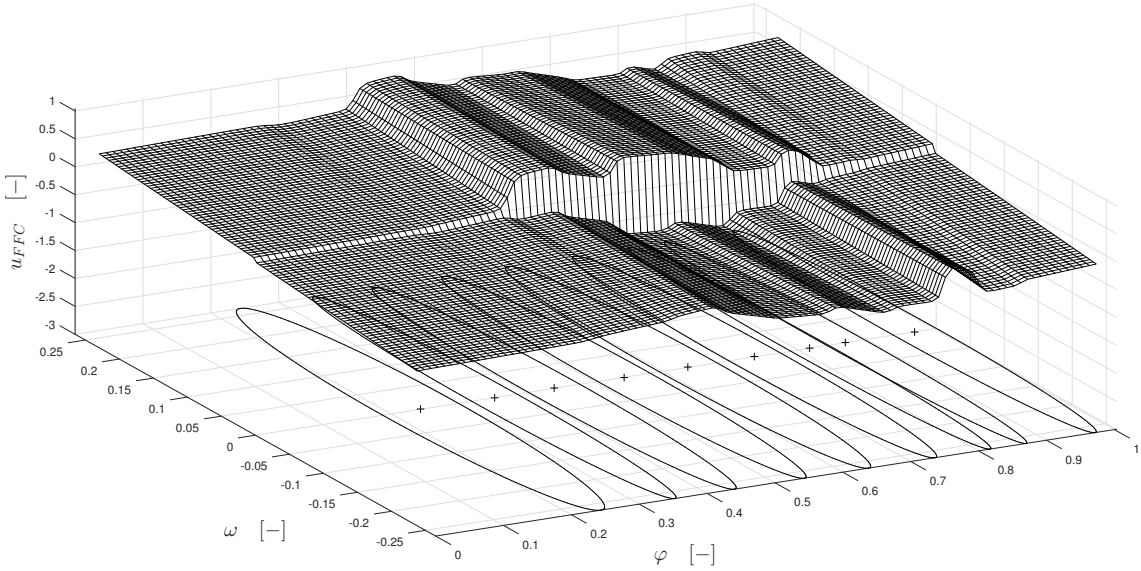

Figure 5.23: Servo drive's inverse model approximation by the RFWR_1D algorithm.

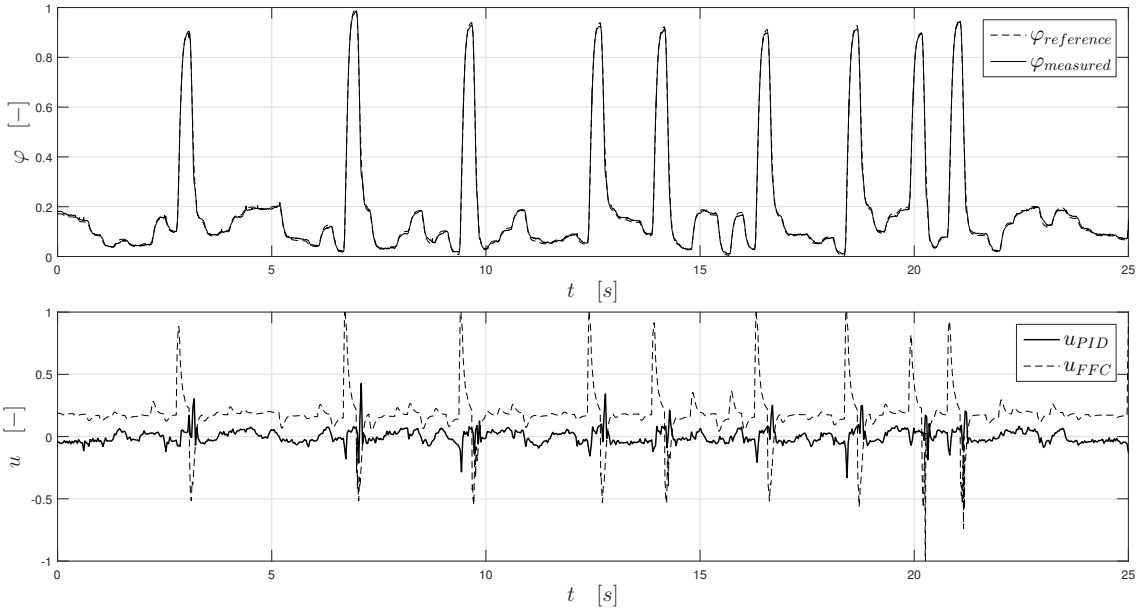

Figure 5.24: Servo drive valve control using the RFWR_2D method

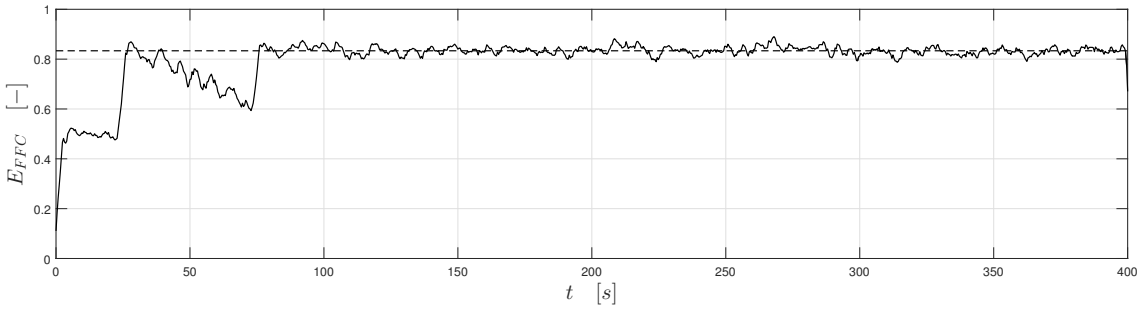

Figure 5.25: Feedforward compensation efficiency

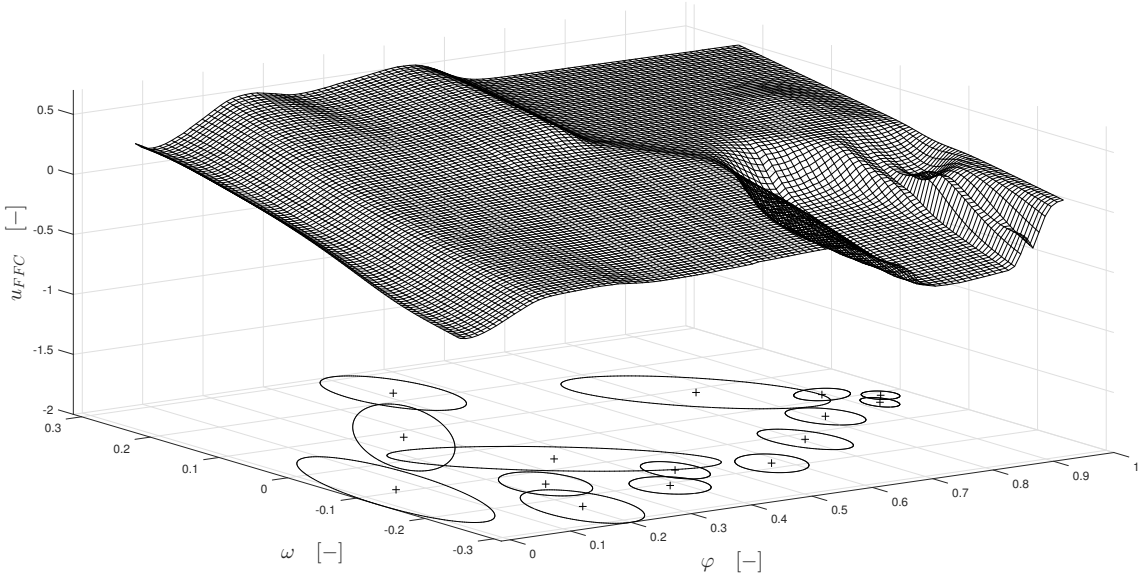

Figure 5.26: Servo drive's inverse model approximation by the RFWR_1D algorithm.

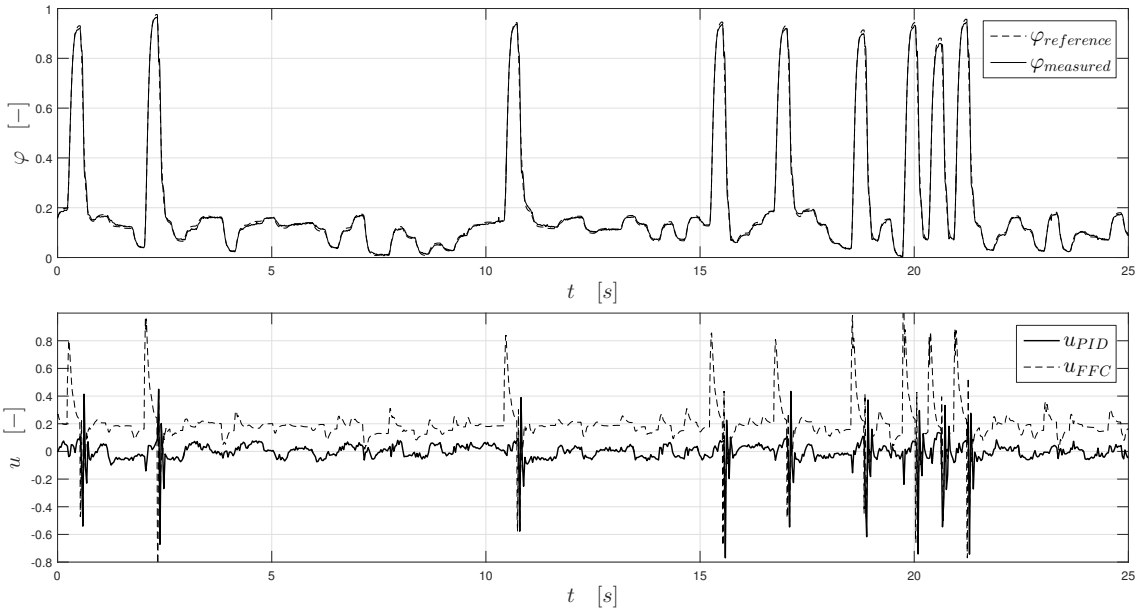

Figure 5.27: Servo drive valve control using the LOLIMOT_2D method

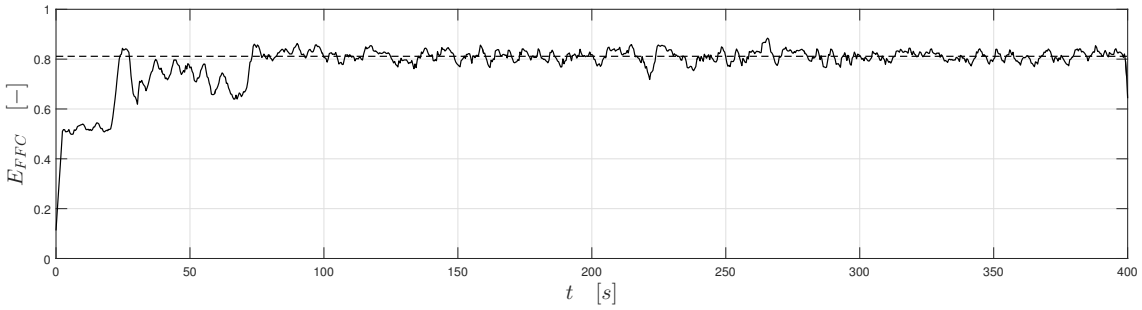

Figure 5.28: Feedforward compensation efficiency

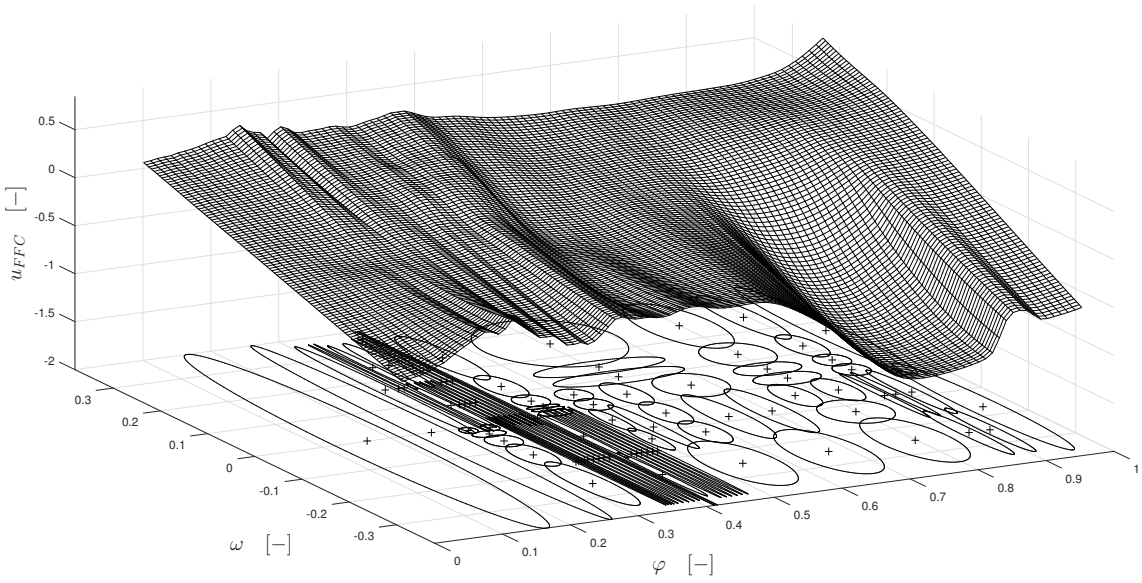

Figure 5.29: Servo drive's inverse model approximation by the LOLIMOT_1D algorithm.

X. Actuator: **Electronic Throttle Valve** Approximation Method: **LUT**

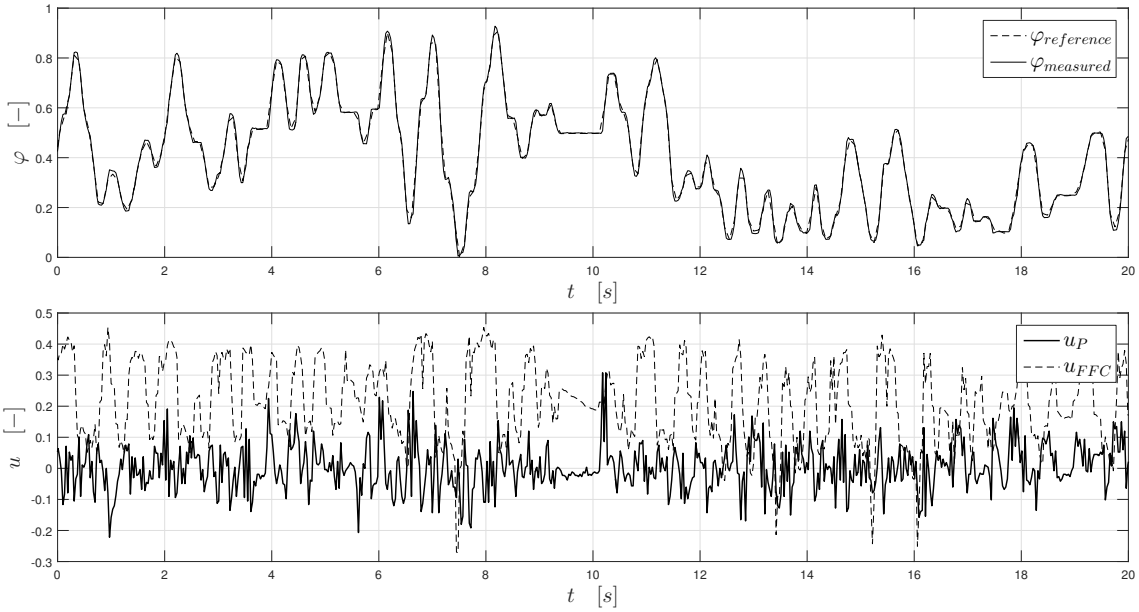

Figure 5.30: Electronic throttle valve control using the LUT method

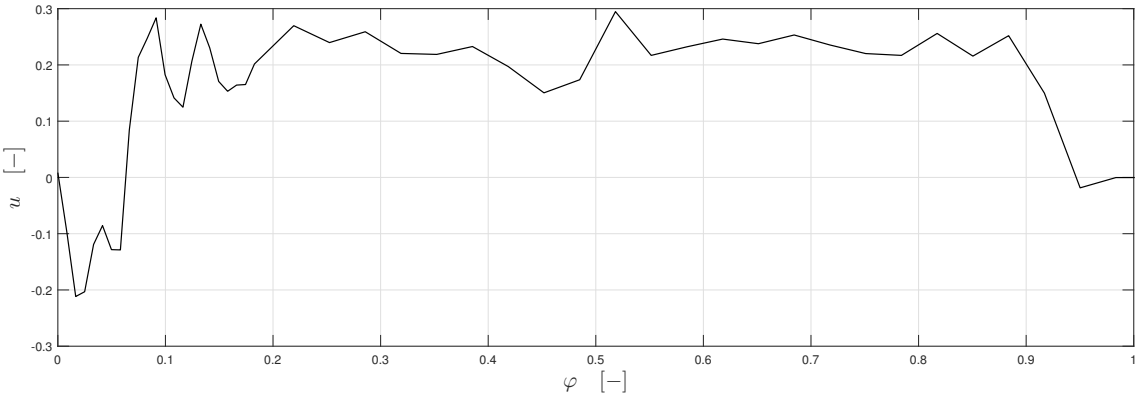

Figure 5.31: Electronic throttle valve's inverse model approximation by the LUT algorithm.

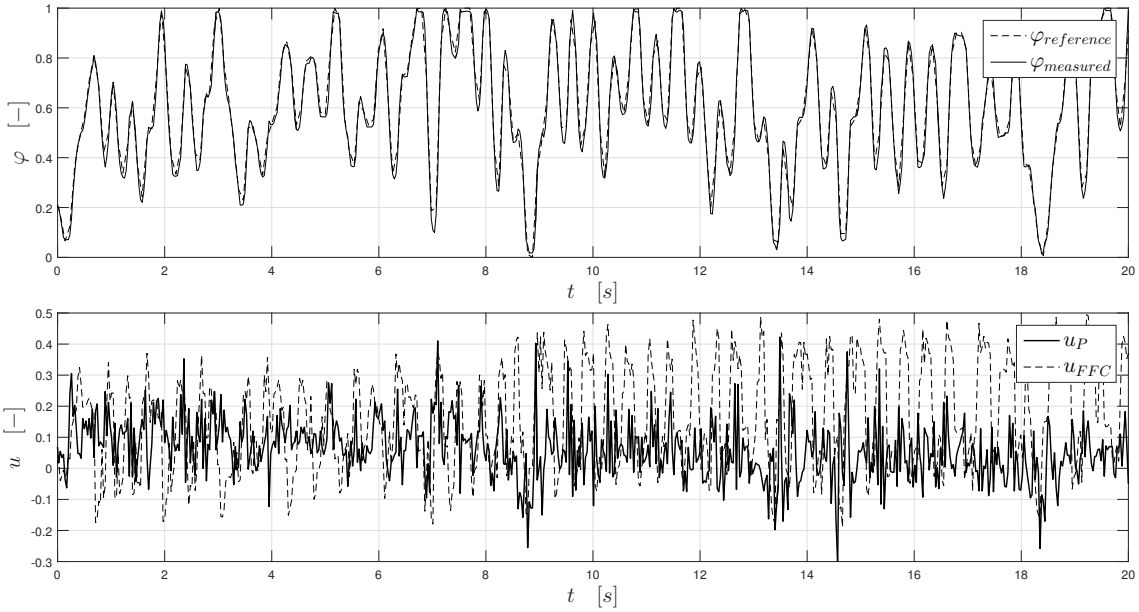

Figure 5.32: EGR valve control using the LUT method

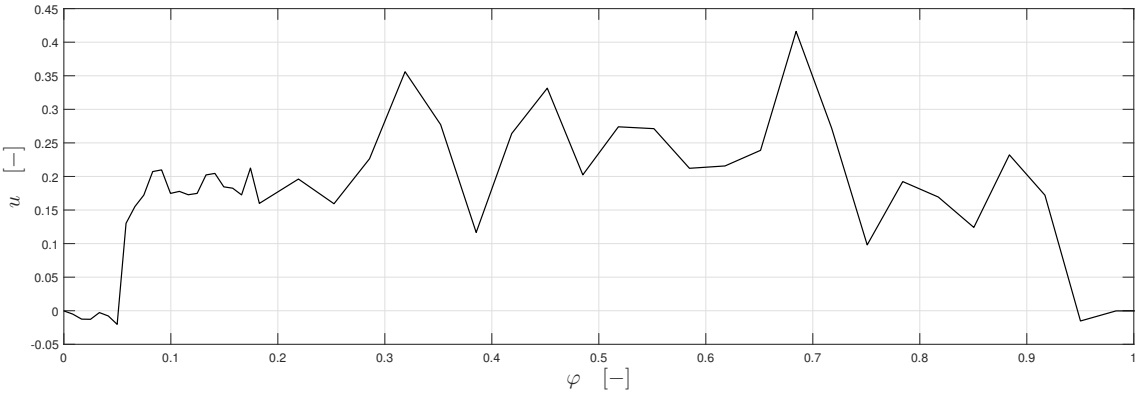

Figure 5.33: EGR valve's inverse model approximation by the LUT algorithm.

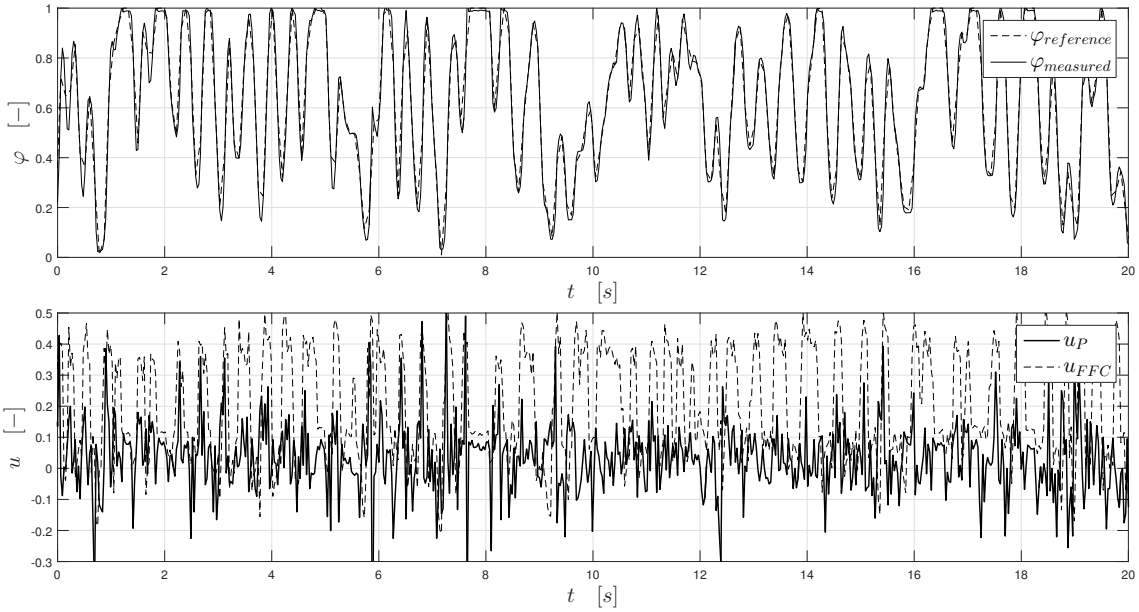

Figure 5.34: Servo Drive control using the LUT method

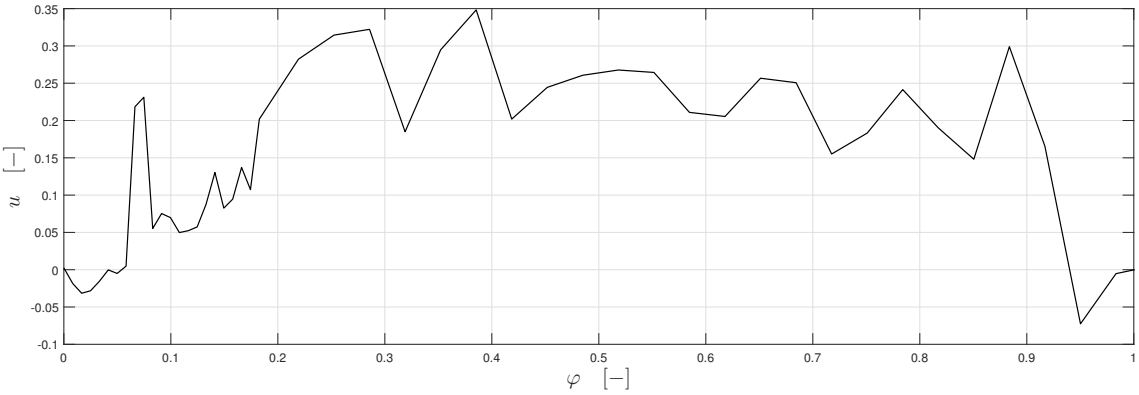

Figure 5.35: Servo drive's inverse model approximation by the LUT algorithm.