CHALMERS
UNIVERSITY OF TECHNOLOGY

Learning Local Models of Multi-axis Robot for Improved Feed-Forward Control

A study of performance using data-driven modeling methods applied in feed-forward control.

Master's thesis in Master Program: System, Controls and Mechatronics

VIKTOR JOHANSSON

Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2017

Master's thesis 2017:EX029

Learning Local Models of Multi-axis Robot for Improved Feed-Forward Control

A study of performance using data-driven modeling methods applied in feed-forward control.

VIKTOR JOHANSSON

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Signals and Systems
Division of Systems and Control
Chalmers University of Technology
Gothenburg, Sweden 2017

Learning Local Models of Multi-axis Robot for Improved Feed-Forward Control
A study of performance using data-driven modeling methods applied in feed-forward control.
VIKTOR JOHANSSON

Supervisor: Stig Moberg, ABB Robotics
Supervisor: Mikael Norrlöf, ABB Robotics
Examiner: Jonas Sjöberg, Signals and Systems

Master's Thesis 2017:EX029
Department of Signals and Systems
Division of Systems and Control
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2017

Abstract

Achieving high path accuracy in multi-axis industrial robot manipulators is an important feature in applications such as water-jet and laser cutting. Due to the complexity of robot manipulators, accurate kinematic and dynamic models of the system are needed for high-accuracy model-based control. Data driven identification and learning approaches have previously been shown to capture the behaviour of undesired effects in a local model, such when they are applied in feed-forward control performance has been increased to a certain point.

This thesis examines and evaluates the performance of applying two different approaches of creating feed-forward velocity control aimed for friction compensation. The objective is to evaluate the approaches in terms of generality, where the method involved testing on two simulation platforms and on a real 6 degree of freedom robot manipulator. Performance was evaluated in terms of the path error of the tool as well as the motor position error.

The first approach shows that applying a previously identified LuGre friction model to each joint on the robot manipulator resulted in a reduction of the path error of the tool as well to the motor position error. Both a static and dynamic variant of the model was tested, where the dynamic show better performance. Testing showed that friction compensation through this model was more difficult for higher velocities, however despite this it was deemed a general solution to friction compensation.

The second approach showed that applying Learning Feed-Forward Control (LFFC) using B-spline networks (BSN) to each joint enabled friction compensation. Learning took place in an offline manner where weights in the network were updated in between motions, and applied in feed-forward control for a number of iterations. Both a static BSN and an extended BSN showed improved performance for velocities which it was trained on. Results were comparable to the first approach, and showed worse performance when used on other velocities than those used in training.

Further work indicated that increased performance could be gained through re-identification of the LuGre model. Nonlinear identification methods such a NARX was also investigated and showed promising results of modeling friction.

It was concluded that there is a clear trade-off between the amount of work needed and the performance gained. A learning approach could save time compared to an identification approach, but also limits the performance.

Keywords: Robot, Manipulator, Friction Compensation, LFFC, BSN, LuGre, NARX

Acknowledgements

First I would like to thank my supervisors at ABB Robotics, Dr. Stig Moberg and Dr. Mikael Norrlöf for their continuous guidance and for supplying me with their expertise within the area of robotics. I am grateful for the time you have taken out of your busy schedule in order to help me with anything from proofreading to helpful discussions.

I would also like to thank my supervisor Prof. Jonas Sjöberg at Chalmers University of Technology for providing me with insightful comments and discussions during the project, as well as guidance during the writing of this thesis.

Finally, a thank you is extended to PhD Student Erik Hedberg at Linköping University for taking an interest in my project and providing me with a fresh perspective. Your help during the start-up phase and proofreading of this thesis is greatly appreciated.

Viktor Johansson, Västerås, June 2017

Contents

1 Introduction

This thesis deals with the application of data-driven identification and learning approaches when applying feed-forward velocity control to a six degree of freedom ABB industrial robot system. The work is aimed at compensation for undesired effects within the system when performing high precision tasks, such as water jet cutting. The work conducted within this thesis is mainly focused on friction compensation, since it is considered to be the largest source of error in the given applications.

This chapter presents a brief overview of the project. Section 1.1 presents the background of the work for this thesis, followed by the contributions in Section 1.2. Furthermore, a method of evaluating the performance of the different approaches is presented in Section 1.3. The chapter is concluded with a presentation of the thesis outline in Section 1.4.

1.1 Background

In an industrial setting, robots performing tasks such as assembly and painting is a common phenomenon. Among the many areas of application one important requirement is high path-following accuracy when for example performing water jet or laser cutting. These types of tasks often require the robot to perform the task quickly with minimum tracking error on a sub-millimeter precision level. In the case of multi-axis industrial robots manipulators, this can be a demanding requirement considering that the robots are in general large and heavy while at the same time struggling with flexibilities in the links, hysteresis, backlash, friction in gearboxes, torque ripples in motors and so on.

To improve the path accuracy of multi-axis industrial robots, a range of different methods can be applied. One such method is iterative learning control (ILC), which has proven feasible in practice. Iterative learning control is a method of learning a control scheme for a particular motion by iterating the motion and improving control based on the tracking error. The control is most often applied as an extension to an already existing feedback control. In [1] the method is described to be mainly considered when an identical task is repeated for the same path or trajectory. However, when the system changes path or trajectory, the learned controller for the previous path does not extend to other paths, meaning that it cannot be generalized.

An alternative method is to apply feed-forward control to the robot. This type of control aims to compensate for internal disturbances in the system by compensating for its effect prior to it happening. In feed-forward control the inverse of the plant model is often used to determine what the control output should be, such that the desired reference signal is achieved. In this way the disturbances in the plant can be accounted for, which in turn would result in higher path-following accuracy for different trajectories and speeds.

Utilizing feed-forward control however requires accurate models of the plant. Deriving a model of the plant can be done through different methods, such as physical modeling and linear/non-linear system identification methods, which are mentioned in [2]. Often an input-output model is necessary such that the control signal can be calculated and added to already existing control. This model has the ability to include a range of different kinematic and dynamic behaviour which accurately describes the plant. In the case of multi-axis industrial robots, actuator models and flexibility models as mentioned in [3, p. 113-138] can serve as examples of models that would provide additional precision to the accuracy of the system model.

However, deriving a complete and accurate model of the system is in general complicated, due to the many components of the system and that many of them are complex and often nonlinear. Therefore it is often preferable to model these effects locally such that particular nonlinearities, internal disturbances and model errors can be compensated for. In the context of industrial robots, these could be the effects of errors in the model of inertia, friction in the motor, flexibility in the links etc.

In order to model and identify local parts of multi-axis industrial robots there exists a range of different data-driven approaches. One such example is presented in [4], where the authors describe a methodology to learn the inverse dynamics of the robot structure through a semi-parametric approach. They utilize a parametric model of the robot constructed through rigid body dynamics (RBD) and a non-parametric approach using incremental kernel methods. They show that the combination of both models provides a trait to enable robotic systems to adapt to mutable conditions of the environment and the mechanical properties of the robot. This could for example be used to mitigate certain internal disturbances and increase the performance.

Furthermore, in [2] multiple data-driven nonlinear identification methods are presented. Among the methods are various types of function approximators for systems as well as neural network methods. These are mentioned as Feed-Forward Neural Network (FFNN) and Recurrent Neural Network (RNN) which can be used for identifying nonlinear systems without extensive modeling. Depending on the choice of general functions for the neural networks, they are associated with different names.

Other methods such a Gaussian Process Regression (GPR) have also been utilized to learn system dynamics, as can be seen in [5]. The authors presents an online learning approach based on drifting GPR which are shown to be on par with other state of the art methods for learning inverse dynamics. Related to this, in [6] the authors even use GPR for constructing a feedforward control of a robot performing laser cutting and showing that the path tracking accuracy is improved.

In addition to previously described approaches, learning feed-forward control (LFFC), which is presented in [7], is another method of applying feed-forward control as an addition to a feedback controlled system. The method is similar to performing system identification, however instead of learning based on the output of the system, LFFC learns by minimizing the error of control signals. In the mentioned thesis two different types of neural networks were presented as function approximators, namely Multi Layer Perceptron (MLP) network and B-spline neural networks (BSN), which have been applied in control applications. In [8] the authors apply feed-forward control using BSN to a two-link rigid robot arm, which increases accuracy of random motions performed by the robot.

1.2 Contributions

This thesis examines the performance with respect to tradjectory tracking, when applying different data-driven learning and identification methods in feed-forward control to a six degree of freedom (DOF) ABB robot. The objective is to compensate for undesired effects such that an improvement can be seen to the path-following accuracy of the tool center position (TCP) for different trajectories and trajectory speeds.

The contributions consist of two approaches for performing compensation of the nonlinear friction appearing in the motors and the gearboxes of the robot through feed-forward control. The solutions involve an identification approach and a learning approach for capturing the effects of friction which shows an ability to increase the path-following accuracy for small and circular low-velocity TCP motions in multiple robot configurations. The proposed control approaches are evaluated within a surface of $40 \times 40cm^2$ in front of the robot, parallel to the ground.

1.3 Method

To evaluate any considered approach of applying different learning or identification techniques aimed for feed-forward control, these are tested on the intended system of application. In order to show that an approach correctly models and compensates for an undesired effect, the feed-forward components shall show improved performance in a number of different configurations, such that the solution can be seen as general.

The intended system of application is in this case an ABB industrial robot manipulator with 6 DOF set up to perform water jet cutting during low-velocity motions. To enable investigation of a considered approach, the control strategy including the feed-forward components are tested and evaluated in a simulation scenario, as well on the real robot system. A successful approach should in this case be considered general for all of the mentioned scenarios. In this thesis, these correspond to the following:

- **One-axis simulation model**

  The control strategy is tested on a simplified simulation model of the robot, where only one axis of the robot is considered. This is the simplest case.

- **Robot simulation model**

  Showing that the indented control strategy works for the simulation of the complete robot system is a key result for investigating its properties on the real system.

- **Real robot system**

  The real application for which the control strategy is mainly considered. This relates to the main goal, namely to show that the intended approach works in the real application.

The performance of a studied approach is measured in each of the mentioned testing scenarios. In order to show generality, the result of the considered approaches should improve the performance of TCP path-following accuracy as well as reducing the motor position error for a number of robot configurations. Evaluation of results will therefore be done with regards to these quantities of measure, when motions are performed at different locations as well as with different velocities.

Key aspects of the evaluation includes parameters such as maximum deviation from the intended path, as well as the standard deviation from the intended path. Successful implemented approaches show improvement in all of these areas, compared to the scenario of not using the approach.

1.4 Thesis Outline

Chapter 2: Robotic System and Test Platform

This chapter aims to present the robotic system considered in this thesis. The first part of this chapter presents an introduction to the theory of industrial manipulators, where among other things friction phenomenons are discussed. The last part covers the experimental platform on which considered approaches are tested. The chapter ends with a description of the test motions used for evaluation.

Chapter 3: System Identification and Learning

This chapter aims to introduce the reader to some concepts and theory in system identification and learning. The first part of this chapter is focused on linear and nonlinear system identification, where the second part presents theory concerning the learning of a feed-forward control structure during operation of a system. For learning purposes, B-spline networks are considered as a function approximator.

Chapter 4: Feed-Forward Control using LuGre Friction Model

A first approach of compensating for friction is presented in this chapter, where feed-forward control using a previously identified LuGre friction model is applied to a robot manipulator. A static and dynamic variant of the model is evaluated, where both show improved performance and can within the scope of this thesis be considered a general solution. It is concluded that the dynamic model performs better in general.

Chapter 5: Learned Feed-Forward Control with BSN

An approach of learning feed-forward control for friction compensation is here presented and evaluated on a robot manipulator. The approach utilizes a B-spline network in order to construct two different control structures, where one can be considered as a static approach, where as the other aims to approximate dynamic behaviours. Both models shows improved performance for patterns where the velocity motion remains the same and are therefore considered a semi-general solution. The conclusion is that the dynamic approximation in general generates better performance.

Chapter 6: Extended Friction Identification

Based on the conclusions made in the previous chapters, this chapter presents methods of performing system identification for finding dynamic models of friction. The work presented in this chapter covers the re-identification of the dynamic LuGre friction model for one joint, as well as a nonlinear identification approach using NARX models. The results indicate that the re-identification might generate better performance if applied to the robot manipulator, and that a nonlinear identification approach might be able to capture more of the friction phenomenons.

Chapter 7: Conclusion and Future Works

The last chapter summarizes and concludes the work performed in this thesis. Potential future work is also presented in this part. The chapter is concluded with a discussion of ethical and sustainability aspects.

2

Robotic System and Test Platform

This chapter aims to introduce some theoretical aspects in modeling and in the representation of multi-axis industrial robots aimed for control, which are relevant for the fields of study considered in this thesis. The chapter also includes a description of the experimental platform and how tests are performed in order to evaluate additional control applied to a robot system. The outline is such that methods of modeling the robotic motion of the considered robot structure is given in Section 2.1. Following this, a description of the experimental platform considered in this thesis is given in Section 2.2.

2.1 Robot Modelling

In the field of robotics, a robotic system can be anything from an automated vehicle to a humanoid and involves many different technical fields. In this thesis, multi-axis industrial robots are in focus, which can be referred to as controlled robot manipulators aimed for manufacturing purposes, where its arm-like structure enables flexible motions within its workspace. The considered robot is illustrated in Figure 2.1, which shows a multi-axis industrial robot consisting of a total of six revolving joints, which also marks its degrees of freedom (DOF). Each joint is interconnected through links in a chain like structure, which resembles that of an arm. The illustrated robot is within the scope of this thesis used for an application of water-jet cutting, where the tool consists of a nozzle for high-pressure water. The TCP can be seen as any tool equipped to joint q_6 in the Figure 2.1.

To enable the motion of the TCP on these types of robots, considering position and orientation, the joints of the robot must be actuated. This is performed through an electrical motor connected to a gearbox in every joint, which has to be controlled in such a manner that the motion in every joint yields the desired TCP motion. To perform this, mapping of joints and TCP are required, as well as physical models of the robotic system such that control structures can be formulated. The most important methods of modeling and representing robot manipulators aimed for control can be found in [3], [9], where as the following sections aim to present the most important aspects of the robot model.

2.1.1 Kinematics

Kinematic models of the robot describes the motion of the joints and the TCP in the robot without regard for the forces applied to the system. As can be seen in Figure 2.1, the position of the robot can be seen as a chain of links where the length and the angular position of each joint determines the TCP. TCP in this case correspond to a position in Cartesian space denoted $\boldsymbol{p}$ and an orientation $\boldsymbol{\phi}$ which can be formulated as

$$X = \begin{bmatrix} \boldsymbol{p} \\ \boldsymbol{\phi} \end{bmatrix} \tag{2.1}$$

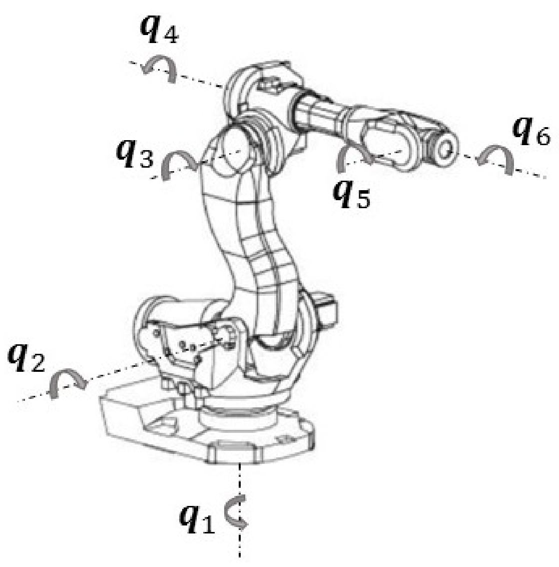


Figure 2.1: Model of a multi-axis industrial robot manipulator. The model shows revoluting joints and their corresponding marked angles.

The robot kinematics can thus be seen as a geometric representation of the robot where the vector of joint angles $\boldsymbol{q}$ helps to represent the TCP of the robot. This can be viewed as

$$X = \Gamma(\boldsymbol{q}) \tag{2.2}$$

where $\Gamma(.)$ is a nonlinear function representing the mapping of angles to TCP of the robot and is commonly referred to as the forward kinematics model. Similarly, the angular positions can be represented as a function of the TCP namely

$$\boldsymbol{q} = \Gamma^{-1}(X) \tag{2.3}$$

which is called the inverse kinematics model.

2.1.2 Robot dynamics

Robot dynamics refer to the relationship between the motions of the robot and the forces acting on the system. A common way to model the dynamics of a robot manipulator is through its angular motion properties and its joint torque. This is called the inverse dynamics and is given in the joint space formulation as

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau} \tag{2.4}$$

and is here given under the assumption that the manipulator is rigid. $\boldsymbol{q} \in \mathbb{R}^N$ is the column vector of angular positions where N is the number of joints. Furthermore $\boldsymbol{M}(\boldsymbol{q}) \in \mathbb{R}^{NxN}$ is the matrix of inertia while $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} \in \mathbb{R}^N$ represent the vector of Coriolis and centrifugal terms and $\boldsymbol{g}(\boldsymbol{q}) \in \mathbb{R}^N$ is a vector of gravity torques, and and $\boldsymbol{\tau} \in \mathbb{R}^N$ is the vector of joint torques.

The components in the inverse dynamics model (2.4) can be derived through a number of methods where the two most commonly used in robotics are the Newton-Euler formulation and the Lagrange formulation. The first method makes use of Newton's and Euler's equations for rigid bodies where as the second makes use of energy properties of the mechanical system to derive the equations of motion. The methods are further explained in [9].

Furthermore, in (2.4) some physical properties are omitted, such as the effects of friction which could have siginificant contributions in some robot manipulators. Also, the assumption of the robot manipulator being rigid is in general not true, since the joints might be experiencing effects such as flexibilities in links, gearboxes and bearings.

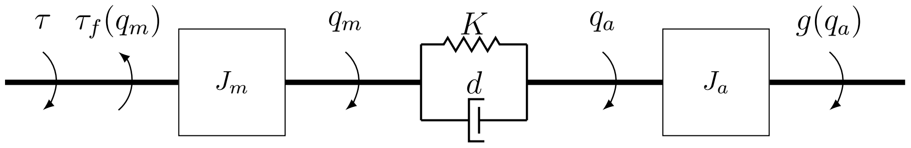


Figure 2.2: Simplified model of one axis of the robot showing flexibilities and physical properties. J_a and J_m are the arm and motor inertia, where as q_m denotes the motor angle and q_a the arm angle. Friction is here represented as the moment τ_f and is modeled on the motor side. The gravity $g(q_a)$ is modeled on the arm side. The moment applied to the axis is dentoed τ.

Modeling flexibilities in robot manipulators are in general a difficult problem, however approaches of extensively modeling this have been discussed in [10]. A common simple representation is shown in Figure 2.2, which is referred to as a flexible joint model. The figure illustrates the model of a single joint on the robot manipulator, where the flexibilities are modeled in the gearbox through a spring and a damper. The model can be expressed as

$$J_m\ddot{q}_m = \tau + K(q_a - q_m) + d(\dot{q_a} - \dot{q_m}) - \tau_f(\dot{q}_m) \tag{2.5}$$

$$J_a\ddot{q}_a = g(q_a) + K(q_m - q_a) + d(\dot{q_m} - \dot{q_a}) \tag{2.6}$$

where q_m represents the angle seen from the motor side and q_a the angle on the arm side, i.e. on the other side of the gearbox. J_m and J_a are the moments of inertia on either side of the gearbox, while K and d are the spring and dampening constants. τ can be seen as the torque which is applied to the motor. In comparison with the complete inverse model (2.4) the friction is here included as $\tau_f(\dot{q}_m)$, which here is modeled on the motor side of the robot.

2.1.3 Robot Control

The inverse dynamics model described in (2.4) is a model which is suitable for construction of a control structure, since motors are generally actuated through torque, or electrical current which has a linear relationship to torque.

Figure 2.3 illustrates a possible control structure in a control circuit that could be applied to a robot manipulator. The robot manipulator is here denoted $\boldsymbol{G}$, where as the components $\boldsymbol{F}$ and $\boldsymbol{FF}$ denotes feedback and feed-forward control respectively. This type of control structure enables the use of feed-forward components containing robot model components such that performance is increased, where as the feedback control can be constructed to be robust enough to compensate for unmodeled behaviours. Because the inverse dynamics model is given in joint space form, the reference signal $\boldsymbol{r}$ is often given in terms of the desired angular positions and derivatives thereof. The measured outputs of the system are denoted $\boldsymbol{y}$.

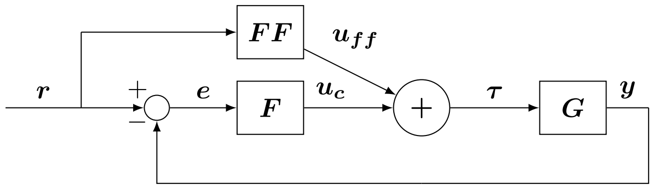


Figure 2.3: Common structure of a control circuit for a robot system where $\boldsymbol{G}$ represents the robot manipulator, $\boldsymbol{F}$ the feedback control and $\boldsymbol{FF}$ the feed-forward control. $\boldsymbol{r}$ is the reference signal which often is given in terms of angular positions and derivatives thereof. The same can be said for the measurements $\boldsymbol{y}$.

2.1.4 Friction

As mentioned in Section 2.1.2, friction can have a large impact on the performance of a robot manipulator. In general, friction corresponds to the complex behaviour of surfaces interacting on a microscopic level. Depending on properties of the surfaces, such as the material and the level of surface finishing in the machinery method, friction effects will vary. In general the effects of friction cause a tangential force to the motion, which acts a type of resistance to the system in question. The behaviour of friction is dependant on several physical properties, such as temperature, velocity and lubrication, which will affect the resistance in different ways.

The friction between two surfaces can be viewed in terms of two main regimes, namely pre-sliding and gross-sliding, as described in [11]. The pre-sliding regime refers to the point in time where there is relative motion between the two surfaces, but there are still points of unbroken contact as well as micro-slips across the surfaces. Assuming that a constant force causes this behaviour, if the same force is removed, the surfaces will not contain the same state as prior to the application of the first force. In this case the surfaces experience something called displacement, where not all micro-slips are recovered, resulting in a remaining constant force acting on the surfaces. This is often what is referred to as a hysteresis behaviour and is an important part in friction modeling. Gross-sliding on the other hand refers to the case a force is applied above a certain threshold, where true sliding occurs and all points of contact are broken.

In modeling of friction, a common classification is to assume that the models are either *static* or *dynamic*. A static friction model in general aims to capture the forces of friction during constant velocity motion. Such a model only depends on current velocity values, which makes it unable to capture the dynamic behaviour of the friction, such as the ones experienced during the pre-sliding regime. Dynamic models however has the ability of representing these phenomena, such as the hysteresis behaviour which are commonly experienced at low velocities. The trade-off is of course that a static model is in general easier to devise, compared to a dynamic model.

In robot manipulator applications where high precision is required, friction is known to cause problems, especially for low-velocity motions. In these cases the friction often appears in relation to the motor and gearboxes. One model which has been extensively used in robot applications is the LuGre model, as described in [12]. The LuGre model is a result of the works between control groups at Lund and Grenoble and can be seen as an extension of the Dahl model [13]. It aims to capture the most relevant aspects of the friction behaviour and contains only a few parameters.

The LuGre model is formulated as dynamic function and has the following form

$$\dot{z} = v - \sigma_0 \frac{|v|}{g(v)} z \tag{2.7}$$

$$g(v) = f_c + (f_s - f_c)e^{-(v/v_s)^2} \tag{2.8}$$

$$F_d(v) = \sigma_0 z + \sigma_1 \dot{z} + f(v) \tag{2.9}$$

where v is the relative velocity of the contact surfaces and z is the internal state which contributes to the dynamic behaviour of the friction. The function $f(v)$ is the viscous friction which is most commonly given as a linear function $f(v) = f_v v$. $g(v)$ captures the Coulomb friction and Stribeck effect where the parameters f_c is the Coloumb friction force, f_s the stiction force and v_s the Stribeck velocity. Furthermore, as stated in [12] for small displacements, the LuGre model produces a spring-like behaviour through the parameters σ_0 and σ_1, which correspond to the spring and dampening constant respectively. F_d is then the friction force output of the function.

For constant velocities the LuGre model also has a static representation, namely

$$F_s(v) = g(v)\,\mathrm{sign}(v) + f(v) \tag{2.10}$$

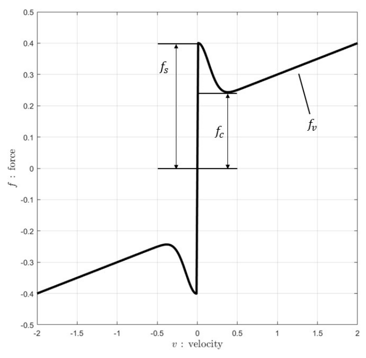


Figure 2.4: Illustration of the static LuGre function and how its parameters correspond to the different parts of the function.

where $g(v)$ is given from (2.8). An illustration of the static friction model can be seen in Figure 2.4. The illustration shows each of the mentioned friction phenomena, such as the Stribeck effect, the stiction force, Coulomb friction force and the viscous friction force. Note however that none of the LuGre models includes the temperature.

To incorporate any friction model in the useful inverse dynamics model (2.4), the friction model can be seen as an additive component which is only dependant on the velocity of the system. Since friction is affecting every joint idependently, the input to the friction model is therefore the angular velocities, each with its correpsonding friction function. The modified inverse dynamics model of the robot manipulator can therefore be seen as

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{\tau}_f(\dot{\boldsymbol{q}}) = \boldsymbol{\tau} \tag{2.11}$$

where $\boldsymbol{\tau}_f(\dot{\boldsymbol{q}})$ is the added torque friction vector, where every element is a function of the respective angular velocity. The function can thus take the form of any of the LuGre functions (2.9) and (2.10), as well as any other friction model.

2.2 Experimental Platform

To evaluate the performance of the considered feed-forward component, an experimental platform was used. The platform includes a control system with feedback and feed-forward components and can be used on either simulation models or a real robot system model. This platform is especially designed for ABB industrial robots, where the outputs from the system can be evaluated.

As stated in Section 1.3, three different system models are considered, namely a one-axis simulation model, a robot simulation model and a real robot system. The control structure together with the target systems can be used to simulate or test any desired robotic motion, such that the behaviour can be examined. An important feature with the given platform is that the control structure can be extended to include further control, such as additional feed-forward control.

In the following sections, the described experimental platform will be described in more detail in order to create an understanding of how tests and simulations are performed.

2.2.1 Control System Overview

The experimental platform can be viewed as an extended version of the control structure shown in Figure 2.3. In the extended system two components have been added, namely $\boldsymbol{LFF}$ and the Learning Algorithm blocks, which can be seen in Figure 2.5. These represent the objective of applying

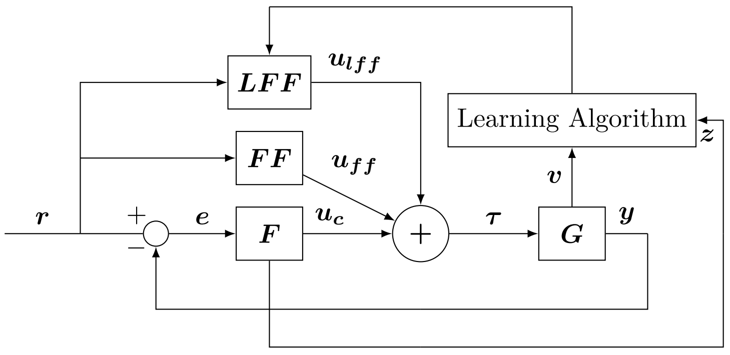


Figure 2.5: Schematic overview of the experimental platform which has been expanded from Figure 2.3. $\boldsymbol{F}$ is here the feedback controller, $\boldsymbol{FF}$ is the feed-forward control and $\boldsymbol{LFF}$ is the additional feed-forward component.

different identification and learning methods in feed-forward control. The platform supports the sampling of different signals within the system, which here are represented as the signals $\boldsymbol{v}$ and $\boldsymbol{z}$. This can be used in the Learning Algorithm or in an identification experiment, such that the additional feed-forward component $\boldsymbol{LFF}$ can be designed. The signal generated from this component is then added to the pre-existing torque motor control.

The robot system $\boldsymbol{G}$ displayed in Figure 2.5 can be viewed as either of the three considered systems, where the control circuit can be applied regardless of which of the three is chosen. Normal control of the motors in the system is on the experimental platform enabled through the components $\boldsymbol{F}$ and $\boldsymbol{FF}$ in Figure 2.5. The signals generated from these components are based on the angular joint references and measurements. In this particular system, the general reference signal can be seen containing

$$\boldsymbol{r} = \boldsymbol{q} \tag{2.12}$$

where $\boldsymbol{q}$ is the row vector of angular position in each joint on the 6 DOF robot system. The feed-forward component $\boldsymbol{FF}$ in the system contains some of the models in the inverse dynamic equation (2.4). Together with the reference signals, these components already contribute to performance being increased within on the robot system.

Any additional model errors or effects are compensated for through the feedback controller $\boldsymbol{F}$. In relation to the extended inverse dynamics model of the robot as shown in (2.11), the remaining effects which the controller compensates for can be seen as

$$\boldsymbol{u_c} = \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{\tau_f}(\dot{\boldsymbol{q}}) \tag{2.13}$$

where $\boldsymbol{g}(\boldsymbol{q})$ is the gravity term of the robot and $\boldsymbol{\tau_f}(\dot{\boldsymbol{q}})$ is the friction term. The feedback controller utilizes the error in both angular position and angular velocity for each individual joint. On the experimental platform the angular position is measured, where as the velocity is numerically differentiated from position.

2.2.2 One-mass model

The first simulation model is the one-axis model. As indicated before, this model aims to simulate the robot system where one joint is controlled at any given time. The model assumes that the robot arm is completely rigid and unaffected by gravity. This model could be seen as model of the first joint in a robot system, where the joint is perpendicular to the ground. The joint is described through a linear motor model with additive nonlinear friction model, namely

$$J\ddot{q} = u - \tau_f(\dot{q}) \tag{2.14}$$

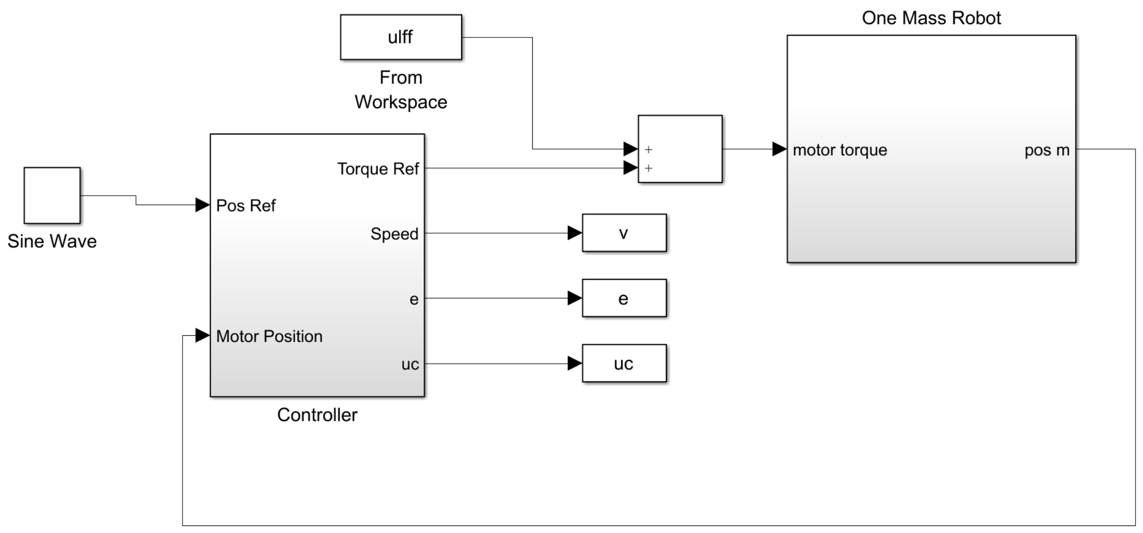


Figure 2.6: Simulink model of the one axis model.

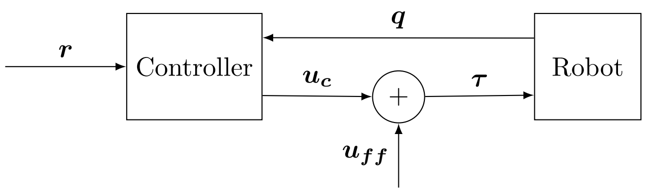


Figure 2.7: An overview of how motions are executed on the experimental platform. A time series of reference signals and feed-feedforward signals is fed to the system such that motions are enabled.

where q is the angular position of the joint, $\tau_f(\dot{q})$ the nonlinear friction component and u the input torque signal.

A Simulink model is given for this motor model together with the control structure previously presented. A high level picture of this can be seen in Figure 2.6. Here the controller takes as input a measured position and a reference position, which it later differentiates in order to extract the velocity. The feed-forward components, such as the inertia J is combined with the controller. Any additional feed-forward control can be applied through the use of the **ulff** component, which can be applied from Matlab.

2.2.3 Robot simulation model and real system

To simulate the robot system and the behaviour of the applied control, a full scale simulation model of the robot is included in the test platform on which simulation is possible. The test platform is set up in the same way as a real robot system.

The robot simulation model is a virtual representation of the real robot and have been derived from CAD-data and experiments on the robot. This enables the simulation of as accurate to reality testing as possible, such that any intended control can be evaluated.

To enable simple execution of robotic motions, programs can be constructed in a scripting language called RAPID, which is commonly used to program ABB industrial robots. These programs are upon execution interpreted by a so called path generator, which generates reference time series signals, as well as the feed-forward signals presented in Section 2.2.1. Normally these signals are directly fed to the control system presented in Figure 2.7, which then executes the desired motion. In this thesis however these signals are recorded, such that they can be altered and manually fed to the system. Thus it is possible to execute desired motions while still applying additional feed-forward control.

2.2.4 Measurements

In the case of the presented simulation models, more or less any data can be sampled from the system. However for the real robot application, this is limited to the measurements of the system.

The 6 DOF industrial robot is equipped with a total of 6 resolvers, one for each joint in the system. These measure the angular position of the joints.

In addition to this, the TCP is measured with an external measurement unit, such that the result of applying feed-forward control can be evaluated. This is performed through the use of a Leica laser tracking system where the TCP is tracked with a precision of 20 μm, with sample time of 1 ms. The measurement system utilizes a reflector attached to the TCP, and a laser is used to continuously track the reflector. Based on measured distance from the measurement unit and the reflector, together with angular measurement in two dimensions, the system can give an accurate estimate of the position.

2.2.5 Robot Motion Tests

In order to evaluate the performance of implemented feed-forward control, each of the presented systems are put through a series of tests. The tests must be varied in terms of both velocity and position in order to investigate generality of the solutions. This means that each system should be tested in terms of different robot configurations and different velocities relevant to the intended area of application. Since the considered application is water jet cutting, the simulated and real robot are tested through performing different low-velocity circular TCP motion patters at different locations.

In this thesis, increased TCP accuracy is considered within the workspace of a 40x40cm area in front of the robot parallel to the ground. Inside of this area, circular TCP motions is performed at several location with different TCP velocities. In this particular case, three circular motions with different radius is considered for one test motion. The reference of this motion for the TCP can be seen in the Figure 2.8, which shows a total of three circles where each circle has its origin in the same place. The different radius considered are 1 mm, 3 mm and 5 mm.

This motion is considered for four different locations within the workspace at three different TCP velocities. The placement of these can be seen in Figure 2.9 where each motion is placed roughly in each corner of the workspace. The positions are denoted as

$$p_i \quad : \quad i \in \{1, 2, 3, 4\} \tag{2.15}$$

where i represents one of the four different positions. The TCP velocities considered for tests at or around each location are 10 mm/s, 40 mm/s and 100 mm/s. Denote these as

$$v_j \quad : \quad j \in \{10, 40, 100\} \tag{2.16}$$

Performance of these tests are evaluated in terms of the following quantities:

1. Reduced TCP path error
2. Maximum deviation from path
3. Root mean square of path error
4. Reduced motor position error

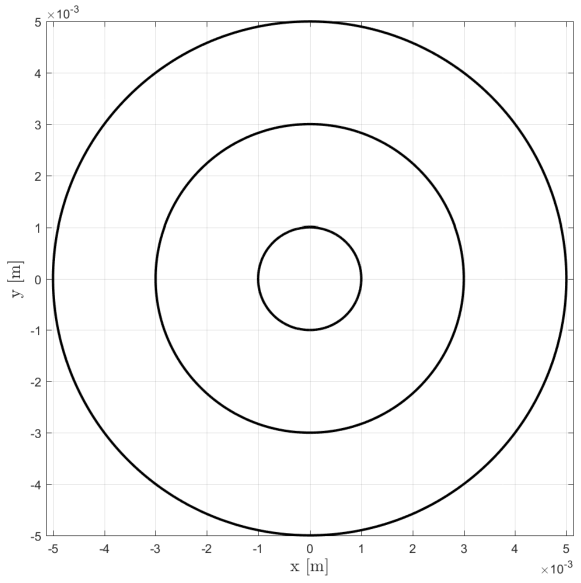


Figure 2.8: The reference circular motion which the TCP is supposed to track. A total of three circles can be seen, each with a separate radius of $1mm$, $3mm$ and $5mm$.

The quantities 1-2 are evaluated through observing the recorded motion of the robot. Furthermore, 3 is calculated through the root mean square (RMS) of the error between the recorded path and the reference for each given sampled point. This can be seen calculated as

$$RMS = \sqrt{\frac{1}{N}\sum_{i=1}^{N} e_i^2} \tag{2.17}$$

where e is the error and N is the number of sampled points. Quantity number 4 is also calculated through the same approach, however here the motor position error is used instead.

Note that for the case during simulation of the one-axis model, only the measured quantity 4 is considered.

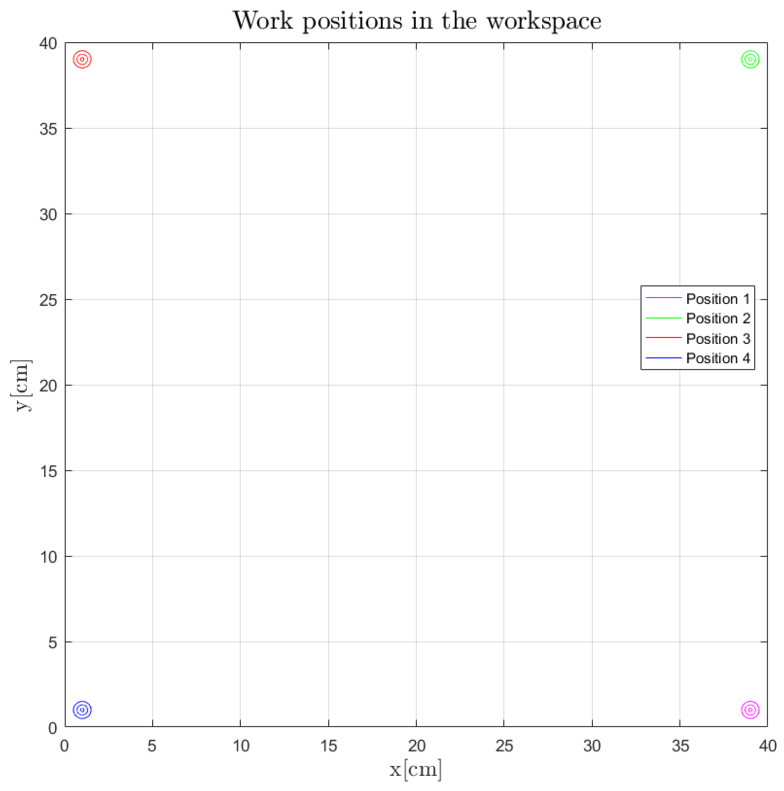


Figure 2.9: Visualisation of the layout of the circular motions in the workspace. A total of 4 positions are considered in this thesis, where three circles of radius 1 *mm*, 3 *mm* and 5 *mm* are used.

3 System Identification and Learning

Identification and learning can be seen as data-driven methods or approaches for creating or complementing models which aims to capture complete or parts of system input-output behaviour. In terms of physical systems, data often refers to measurements which are sampled from the system through a series of experiments. These are often designed such that the desired effects to model are more prominent than other effects within the system. The data is then used in order to fit a model to the data, such that the output of the model can reproduce the same results as the effects would do on a real system. In reality a model can never accurately represent the real system, so the best one can hope for is a good representation of the real effect.

Depending on prior information and physical insight of the real system, it is possible to set up a model which already represents the behaviour of the system, however the parameters might be unknown. This is commonly referred to as a *grey-box* model and can be used to more accurately model a system or effect. The problem becomes harder however when no prior information is available. In this case the model is usually referred to as a *black-box* model.

In this thesis, two approaches of learning or capturing system effects in the form of grey-box and black-box models are examined, namely System Identification and Learning Feed-Forward Control. System identification is a common method of utilizing measurements from experiments in order to construct and verify a suitable model. This is normally done in an offline manner, where data have been sampled and a number of models are tested and evaluated. Learning Feed-Forward Control however can be seen as a method of applying a control structure in a controlled system and is aimed for direct compensation of the system effect such that once the effect is compensated for, the control structure contains a model of the effect. This can be seen as a type of machine learning or adaptive control where the control structure is updated during operation of the system.

3.1 System identification

The general problem of identification can be seen as describing the relationship between past inputs and outputs to future outputs, such that it accurately models the desired system or effect. This can be represented as

$$y(t) = g(\varphi(t), \theta) \tag{3.1}$$

where y is the model output and g is the function which describes the relationship between the inputs and outputs of the model. θ is a vector containing the function parameters and φ contains the regressors, namely previous inputs $u(t)$ and outputs $y(t)$ such that

$$\varphi(t) = [y(t-1)\ y(t-2)\ ...\ y(t-n_a)\ u(t)\ u(t-1)\ ...\ u(t-n_b)] \tag{3.2}$$

where n_a and n_b is the number of considered signals as inputs to the model. Note that normally $y(t)$ is considered as a continuous function, however since data is collected through sampling it is represented as a discrete model.

The model presented in (3.1) can be seen as either a dynamic function or a static function. Systems are normally dynamic, meaning that the current output not only depends on the current input, but also on the system state or inputs prior to the current output. This could be seen as the model containing prior outputs and/or inputs of the model, i.e $y(t-k)$ and/or $u(t-k)$ as input. Depending on prior information about the system, a static or dynamic model can be used to capture a system or effect. Choosing appropriate regressors is therefor a large part of the system identification procedure.

Not only is it important that the model structure is well designed, but data must also be accurately representing the system or effect. This means that data must be exhaustive enough such that it captures accurate system properties as well as not being influenced by other effects or disturbances. If such data exists, the model can be fit to the data and thus replicating the behaviour such that given another input, the model accurately describes the output of the system effect.

Generally for a model to fit to data, the error between measured output and the model output for given input should be small. Since the choice of regressors is often predetermined through some grey-box or black-box structure, the problem of performing model fit can be seen as finding the parameters which minimizes a cost function $V(\theta)$, namely

$$\hat{\theta} = \arg\min_{\theta} V(\theta) \tag{3.3}$$

where the cost function $V(\theta)$ often is based on error between the model and the measurement. Let $\hat{y}(t|\theta) = g(\varphi, \theta)$ denote that the model is a predictor of the true model y and let N the number of samples generated from y. A common cost function in this scenario is the error squared cost function, which is quadratic given the error for finite sample series

$$V(\theta) = \sum_{k=1}^{N} (y(k) - g(\varphi(k), \theta))^2 \tag{3.4}$$

where y is the measured data. This is the most common cost function, but naturally any other cost function can be used. The model which minimizes the cost function can therefore be seen as a solution to the optimization function.

Depending on the function g, different methods of solving the optimization problems can be used. g can be either seen as linear or nonlinear, where the complexity of finding the solution and the ability to accurately represent the system is directly related to the choice. In the next two sections, models and solutions to the individual problems will be discussed.

3.1.1 Linear identification

The model mentioned in (3.1) can considered to correspond to a linear identification problem if the function is linear in terms of the estimation parameters θ as well as the regressors $\varphi(t)$. A typical representation of such a model is given as

$$y(t) = g(\varphi(t), \theta) = \varphi^T(t)\theta \tag{3.5}$$

where the output of the model is a linear combination of past inputs and outputs of the system. In order to fit the model to data, the same optimization problem as described in (3.3) can be used to fit the model through data. Assuming that the same cost function is used, namely (3.4) the problem becomes a least square problem which can be easily solved.

If the system to identify is also given as a linear time-invariant model, different types of ready made models can be used in the black-box case, where the choice of regressors determines the type of model. Examples of these are for example ARX, ARMAX, BJ and OE. Theory of how to chose such a model and the representation of the models is presented in [2, p.79-139].

3.1.2 Nonlinear identification

When a linear model is not rich enough to represent a system or system effect, nonlinear models can be used. In such a model, the representation is often given in the form represented in (3.1) and states that the relationship between the parameters to estimate θ and the regressors φ can be nonlinear.

A normal approach to model a nonlinear system is to assume the structure to be

$$g(\varphi(t)\theta) = \sum_k \theta(k) g_k(\varphi(t)) \tag{3.6}$$

where $g_k(\varphi(t))$ is referred to as a basis function. This model can be seen given in [14], where different types of basis functions are presented. In the model above the output can be regarded as linear in terms of its parameters θ. A common basis function is for example the sigmoid function, which is given as

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{3.7}$$

which could be viewed as a smooth version of a step function.

Other models such as neural networks do not have the same structure as presented in (3.6). The solution to the optimization problem then becomes more difficult. Here typical optimization tools are gradient descent methods, such as Back-Propagation, which are common in the neural network case.

3.2 Learning Feed-Forward Control

Learning Feed-Forward Controll (LFFC) is a method of learning a control structure on an already existing controlled system. The purpose is to compensate for the effects of reproducible disturbances within the system and thus bypassing the need of first deriving a model for feed-forward control. The method, as presented in [7] is based on a concept called Learning Control, which is defined as a control system that contains a function approximator where its input-output mapping is adapted during control.

The objective with the adaption is to achieve the desired behaviour of the system, such as reduced position error. The function approximator is defined such that a function $F((.), \boldsymbol{w})$ is an input-output mapping where the parameter vector $\boldsymbol{w}$ is selected such that the best approximation of a function $f(.)$ is chosen. The difference compared to LFFC is that this concept does not limit the scope to feed-forward control.

The application of an LFFC can be seen in Figure 3.1. The feed-forward component decides its output based on the reference signals sent to the component, where function approximator $F(\boldsymbol{r}, \boldsymbol{w})$ is used to generate the output. Prior to this, the parameters $\boldsymbol{w}$ has been updated based on some cost function being minimized, where the input ranges from the control error $\boldsymbol{e}$ or the control signal $\boldsymbol{u_c}$.

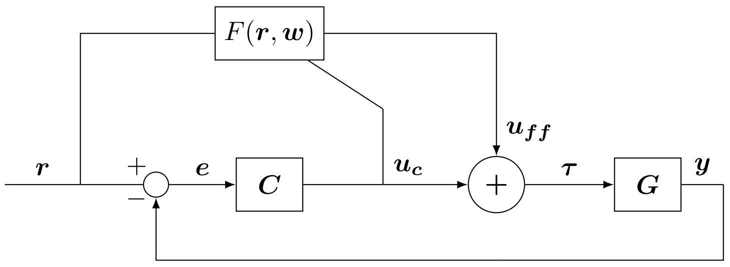


Figure 3.1: Control circuit of applying LFFC where a feed-forward component in the form of a function approximator $F(\boldsymbol{r},\boldsymbol{w})$ generates a control output to the system based on the reference signal $\boldsymbol{r}$. The feedback control signal $\boldsymbol{u_c}$ is used for updating the parameters $\boldsymbol{w}$ in the function approximator based on a cost function being minimized.

The method of LFFC has previously been called Feedback Error Learning (FEL) when the function approximator Multi Layer Perceptron (MLP) neural network was used. This was applied by Kawato [15], where it was also shown that when using the output of the feedback controller as a learning signal on robot manipulators, the feed-forward component converges to the inverse of the plant and compensated for reproducible disturbances.

In [7] some issues regarding the choice of MLP as a function approximator was presented. The adaption of weights in the MLP involves a great deal of complex computations which could present problems in real time control applications. Furthermore, the MLP showed problematic generalizability during learning on highly correlated data, since the network tends to fit the last presented data. This is due to the weights being adapted globaly, meaning that all parameters are updated when presented with a new output. The thesis [7] instead proposes to use a function approximator called B-spline networks, which do not suffer from the same issues as the MLP network.

3.2.1 B-spline Networks

A B-spline network (BSN) is a network based function approximator, much like an MLP network where inputs are mapped to basis functions called B-splines, which in turn are associated with a corresponding weight. When presented with training data, the weights are adapted through the use of back-propagation where a cost function is being minimized. The B-splines are defined over a determined interval, where the output of the B-spline is only non-zero when presented with an input within the given interval. This leads to a local adaption of weights, since only a few B-splines are active during inputs, resulting in the update of only a few weights. Learning on highly correlated data is therefor possible with this structure.

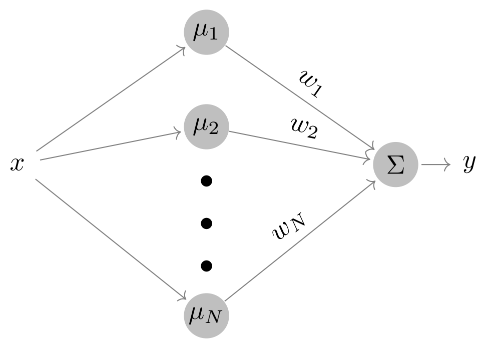


Figure 3.2: B-spline network structure where x is the input to the N number of B-splines in the hidden layer. Each B-spline μ_i is assigned a weight w_i were the output is a linear combination of B-splines and weights.

The structure of a B-spline network can be seen in Figure 3.2. The network contains one hidden layer where the so called B-splines are defined. The input x can be seen mapped to each individual B-spline within the network, where N represents the number of B-splines. Together with each B-spline there is also a weight, which are then all summed in order to generate the output y. The output function can therefor be represented as

$$y(x) = \sum_{i=1}^{N} \mu_i(x) \cdot w_i \tag{3.8}$$

where the B-splines are given in the form of the function $\mu_i(x)$ given that x is the input.

B-splines themselves are basis functions which are defined over a set interval. Each B-spline gives an output value within the interval of $[0, 1]$ and is often classified in terms of order. A B-spline of order j is often represented as $\mu_i^{(j)}(x)$ and consists of piecewise polynomial function of order $j-1$. In the base case, a B-spline of order $j = 1$ is a constant function which is non-zero within the an interval. This is represented as

$$\mu_i^1(x) = \begin{cases} 1 & if\ \lambda_i \leq x < \lambda_{i+1} \\ 0 & otherwise \end{cases} \tag{3.9}$$

where λ_i and λ_{i+1} are the interval which the function is defined within. Here λ is called a knot, and is a member in a determined set called a knot vector $\boldsymbol{\lambda}$. The knot vector is a design parameter in the construction of BSN and is given as ordered set of ascending values

$$\boldsymbol{\lambda} = [\lambda_1, \lambda_2, ..., \lambda_m] \in \mathcal{R}^m \tag{3.10}$$

which defines the input space for which a function can be approximated. The placement of the knots decides the width of the B-splines, which is an important design aspect. The tighter the placements of knots get, the more flexible the function becomes in approximating specific features. However it comes with higher computational cost, since more knots need to be defined in order to maintain the input space.

From this set of knots and the first order B-spline, higher order B-splines are defined recursively through the following function, namely

$$\mu_i^{(j)}(x) = \frac{x - \lambda_i}{\lambda_{i+j} - \lambda_i} \mu_i^{(j-1)}(x) + \frac{\lambda_{i+j+1} - x}{\lambda_{i+j+1}\lambda_{i+1}} \mu_{i+1}^{(j-1)}(x) \tag{3.11}$$

where the output of each individual B-spline for a few low-order B-spline cases are shown in Figure 3.3. In this thesis B-spline up to a order of $j = 3$ are considered, which are shown in the figures. The knot vector used in this case is given as $\boldsymbol{\lambda} = [1, 2, 3, 4, 5]$.

In the case that x is a vector of inputs, the output of a B-spline is the tensor product for each of 1-dimensional B-splines such that

$$\mu_{i_1,i_2}^{(j)}(x_1, x_2) = \mu_{i_1}^{(j)}(x_1)\mu_{i_2}^{(j)}(x_2) \tag{3.12}$$

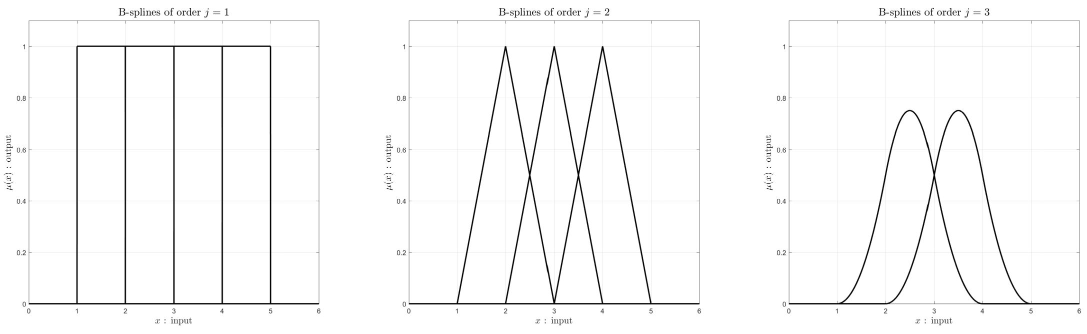
where knots have been defined separately for each of the input spaces.

For each of the B-splines of the chosen order, there is as mentioned an associated weight. The weights are updated through the minimization of a cost function. B-spline network weights can be updated both online and offline, meaning that it can be performed during or after a training motion is completed. This results in two cost functions, namely the online cost function

$$J = \frac{1}{2}(y_m - y)^2 \tag{3.13}$$

where y_m is data from the function to learn and y is in this case the output of the function given in (3.8). Similarly for the offline case, the cost function becomes

$$J = \frac{1}{2}\sum_k (y_m(k) - y(k))^2 \tag{3.14}$$



(a) 1st order B-spline **(b)** 2d order B-spline **(c)** 3d order B-spline

Figure 3.3: The placement of B-splines for different orders when the given knot vector is $\boldsymbol{\lambda} = [1, 2, 3, 4, 5]$.

where all the measurements are included. The weights in the network, as presented in Figure 3.2 are updated through back-propagation. This is performed by taking the gradient with respect to the weights and adapting the weights with the corresponding value. This is performed as

$$\begin{aligned}\Delta w_i = -\gamma \frac{\partial J}{\partial w_i} = -\gamma \frac{\partial J}{\partial y}\frac{\partial y}{\partial w_i} &= \gamma (y_m - y) \frac{\partial \sum\limits_{i=1}^{N} \mu_i^{(j)}(x) \cdot w_i}{\partial w_i} \\ &= \gamma (y_m - y) \mu_i^{(j)}(x)\end{aligned} \tag{3.15}$$

where Δw_i is the added weight update to weight w_i and γ is the learning rate, which is chosen between the interval of $[0, 1]$. x is here the input to the system. In the offline case, this becomes

$$\Delta w_i = \gamma \sum_k (y_m(k) - y(k)) \mu_i^{(j)}(x(k)) \tag{3.16}$$

For the offline case [7] also proposes to normalize the weight update in order to prevent large weight adaptations. This is done in the following manner

$$\Delta w_i = \gamma \frac{\sum\limits_k (y_m(k) - y(k)) \mu_i^{(j)}(x(k))}{\sum\limits_k \mu_i^{(j)}(x(k))} \tag{3.17}$$

where the offline adaptions are divided by the sum of all outputs of the given B-spline. By performing a normalization the effect of infrequent large errors become less influential, which helps the learning of the general behaviour.

3.2.2 BSN in Control Applications

B-splines networks have been seen used in many control applications where one desirable feature is the width and placements of knots, which essentially represents the defined input space and the precision of the approximation within certain intervals. An example of this is shown [16] where the authors use B-spline networks to approximate functions like friction and cogging in motors by applying it in a feed-forward component. The utilize prior knowledge about the system when they chose knot placement and width in order to effectively give more precision in the learning scenario.

In control applications, when trying to approximate different physical functions within a system, more than one input is often considered for the feed-forward component. In the case of BSN this can be problematic in terms of the curse of dimensionality. When the number of inputs increases, the number of weights within the system increase exponentially, resulting in high computational complexity. Further problems are also that copious amounts of data are necessary in order to properly train a network.

To bypass this problem, it is often common to apply a principle of parsimony. The principle states that the best models are often those with the simplest acceptable structures with minimum number of parameters, which is stated in [17]. Based on this, it is often best to split a high dimensional BSN into several smaller BSN which contains a small number of inputs. In order to apply this method, some insight into the system is useful in order to motivate certain divisions of the BSN. However there exists methods of automatically applying empirical modelling techniques such as ASMOD [18].

In learning feed-forward control applications, as stated previously about Feedback Error Learning, the learning signal during control was the control signal u_c. Similarly the update of weights for the BSN case should also contain the same learning signal. This is performed in the offline case as

$$\Delta w_i = \gamma \frac{\sum_k u_c(k)\mu_i^{(j)}(x(k))}{\sum_k \mu_i^{(j)}(x(k))} \tag{3.18}$$

where the error between the model and the and the measured value is replaced with the learning signal $u_c(k)$.

4
Feed-Forward Control using LuGre Friction Model

The friction within robot manipulator system is as described in Section 2.1.4 is a large source of error in terms of high accuracy performance. A natural approach to compensate for the friction is to use a known model of the friction as a feed-forward component in the system. Such a model usually has theoretical and experimental support of accurately describing the behaviour of the friction, which is a key aspect in applying feed-forward control. The LuGre model is a model which has strong support in literature, which is a motivation to why it is interesting to examine if including the friction model in a feed-forward component can improve the control performance.

Prior to the work of this thesis, parametrized models for static and dynamic LuGre friction have been identified on the exact same robot described in this thesis. These models are applied in the simulation model of the robot manipulator, in order to simulate the behaviour of the actual friction within the system. Identification of the LuGre model will therefore not be covered in this section. Instead the contribution of this chapter is to examine the result achieved when applying an already existing parametrized model as a feed-forward component. The result of this work will be considered as a frame of reference for other methods as well as to show that the implementation of the LuGre friction model indeed works.

The chapter is distributed such that a short description of the LuGre models used will be presented in Section 4.1, followed by the implementation of the models as feed-forward components in Section 4.2. The chapter then proceeds to present results from using these components in simulation and on a real robot in Sections 4.3 and 4.4, with the aim of showing that this type of friction compensation satisfies the objective of increasing the performance. The chapter is concluded with an evaluation of the performance and correctness of the used LuGre models in Section 4.5.

4.1 LuGre Model Description

The static and dynamic LuGre models presented in Section 2.1.4 make up the two feed-forward components considered in this chapter. Each joint in the robot manipulator system suffers from the negative effects of friction where the majority of friction comes from the gearboxes in each joint. Through information supplied from ABB Robotics, it is assumed that a total of 90% of the friction resides within the gearbox and 10% from the motor in each joint. Therefore there is a strong motivation to apply friction compensation to the control of each individual joint.

The representation of the LuGre models differ slightly from the general models mentioned in Section 4. Based on the representations presented in equations (2.8)-(2.10), the static component for one joint can be expressed as follows

$$F_s(v) = [f_c + (f_s - f_c)e^{-(v/v_s)^2}]\tanh(\beta v) + f_v(v) \tag{4.1}$$

where f_v is the viscous friction parameter, f_s is the stiction parameter, f_c the coulomb friction parameter and v_s the Stribeck velocity. In the equation above, the sign function have been replaced with an approximation in order to remove its discontinuous behaviour. The approximation is a hyperbolic function, where the approximation is given according to

$$\text{sign}(v) \approx \tanh(\beta v) \tag{4.2}$$

where β represents the steepness of the slope. In the used model, this constant will always be $\beta = 5$.

The dynamic LuGre friction model is based on the equations (2.7)-(2.8) and remain unchanged. The model used in a feed-forward component for a single joint can be seen in the more compact form as

$$\begin{aligned} \dot{z} =& v - \sigma_0 \frac{|v|}{g(v)} z \\ g(v) =& f_c + (f_s - f_c) e^{-(v/v_s)^2} \\ F_d(v) =& \sigma_0 z + \sigma_1 \dot{z} + f_v v \end{aligned} \tag{4.3}$$

where the state z must be estimated in order for the model to be used.

The parameters of the presented models have been estimated prior to this project, meaning that for each individual joint a set of parameters are given. The models here can therefore be directly implemented in a feed-forward component and applied to the system.

4.2 Implementation

Given the static and the dynamic LuGre equation in (4.1) and (4.3), together with its corresponding parameters, the feed-forward component can be implemented in the system according to Figure 4.1. The Figure shows the control circuit extracted from the system schematic in Figure 2.3 together with the intended LuGre friction compensation $\boldsymbol{F(\dot{q})}$ as an added term to the already exsisting control. The LuGre model is only dependent on the velocity state of the robot manipulator, which is why only the velocity reference is input to the feed-forward component.

Due to the fact that friction in each individual joint is independent of each other, the friction compensation is an added scalar term to each individual joint. This can be written as

$$\boldsymbol{\tau} = \boldsymbol{u} + \boldsymbol{u_{lff}} = \boldsymbol{u} + \boldsymbol{F(v)} = \boldsymbol{u} + \begin{pmatrix} F_1(\dot{q}_1^{ref}) \\ F_2(\dot{q}_2^{ref}) \\ F_3(\dot{q}_3^{ref}) \\ F_4(\dot{q}_4^{ref}) \\ F_5(\dot{q}_5^{ref}) \\ F_6(\dot{q}_6^{ref}) \end{pmatrix} \tag{4.4}$$

where τ is the torque control signal sent to the robot manipulator. This in turn enables a separate implementation of each friction function in the individual joints. An implementation of the static LuGre model is straightforward due to its static property. For the dynamic model however, since it is a differential equation, it must be solved with regards to the state z, before it can be applied.

In the case of controlling the robot, control signals are sent with a sample time of $h = 4$ ms. Solving of the dynamic LuGre model is done through the use of the Euler forward method with the given sample time. Taking the dynamic equation of state z in the LuGre equation (2.7), the expression can be rewritten as

$$\dot{z} = v - \sigma_0 \frac{|v|}{g(v)} z \approx \frac{z(k+1) - z(k)}{h} \tag{4.5}$$

where v is the velocity. The dynamic LuGre equation incorporates the equation $g(v)$ which is given in (2.10). The expression for the update of the state z is then given as

$$z(k+1) \approx z(k) + hv(k) - h\sigma_0 \frac{|v(k)|}{g(v(k))} z(k) \tag{4.6}$$

which can be incorporated in the output equation (4.3).

A common problem with applying the Euler forward method to a dynamic system is that it is numerically unstable in relation to the step size. It was found during simulation that having a step size of $h = 4$ms resulted in an unstable behaviour for the given LuGre models. Due to the inability to change the sample time in the control signal, linear interpolation was used in order to effectively increase the number of velocity data points and thus enabling a lower sampling time in the update of the state z (4.6). Experiments show that an interpolation of 40 data points was sufficient to ensure stable output from the dynamic LuGre friction model for the evaluation tests.

The resulting implementation of the LuGre model in Matlab code with interpolated speed can be viewed in appendix A.1. This code is effectively used in order to create the output torque which is fed to the system, both in simulation and in practice.

4.3 Model Simulation

As a part in evaluating the feed-forward friction compensation using static and dynamic LuGre, both of the models should be tested in a simulated environment. The available simulation models described in this thesis are the one-axis model and the robot simulation, which have been presented in Section 2.2. These can be seen as the optimal scenarios of testing in that the contents of the feed-forward components, partly or completly represent the system friction model. Successful implementation of the dynamic LuGre model should therefore compensate for the friction effects and thus improving the performance of the system. The evaluation of the static LuGre function aims to show that performance is gained through the assumption of a more simple friction model.

Improved performance is considered when the path accuracy of the TCP and joint position tracking is increased. A successful implementation of the LuGre models should therefore show improvement to the joint position error and final tool center position error. Furthermore, improved performance should be seen on multiple locations and speeds, such that the compensation through the feed-forward component can be seen as general. This is relevant for both simulation models.

Since the one-axis model only contains one motor and therefor no flexibilities, of the two mentioned evaluation criteria, only the motor position error is considered. In order to compensate for this, another measure of performance is analysed, namely the output of the feed-forward components. This is compared with the friction of the system.

In the following sections results from the one-axis model simulation and the robot simulation will be presented. The results from each simulation scenario will be compared with the corresponding simulations without any feed-forward compensation.

4.3.1 One-axis Model Simulation

Simulation of the one-axis model aims to show that in a very simple case, where only one axis is considered, feed-forward compensation is working. Each LuGre model is therefore applied as a feed-forward component in the system with the goal of removing the effects of the model friction, which here is generated from the exact same dynamic LuGre model presented in (4.3). Prior to applying the feed-forward control, the system is run without it in order to observe the effects. The reference signal in this simulation is simply a sinusoidal signal applied as a position reference to the system, where the velocity reference is given as the derivative of the position signal. The reference signal, as applied according to Figure 4.1 is in this case generated through

$$\boldsymbol{q_{ref}} = A\sin(2\pi ft) \tag{4.7}$$

where $A = 1$ and $f = 1$. Since the simulation model only contains one axis, $\boldsymbol{q_{ref}}$ is scalar. The velocity signal, which is used by the LuGre friction models is applied as

$$\boldsymbol{\dot{q}_{ref}} = 2\pi A\cos(2\pi ft) \tag{4.8}$$

The reasoning behind the choice of using a sinusoidal reference signal is simply to simulate the motion of one motor in a robot manipulator system performing circular TCP motions.

The friction function plots for the static and dynamic LuGre model as feed-forward components for the given reference signal can be seen in Figure 4.2a and 4.2b. In the Figures, the torque output of each feed-forward component can be seen as a function of the velocity, given (4.8) as input. These are compared to the friction output of the system. Recall that the model of the friction used in the simulated motor is in fact the same dynamic LuGre model applied as a feed-forward component. It is therefore natural that the friction curve for the one mass model and the dynamic LuGre feed-forward component are the same in Figure 4.2a, seeing as state z in (4.6) is initialized to the same value. The static LuGre feed-forward component however can be seen making a static approximation of the dynamic model friction.

The simulation of these models are performed individually during a 10 second duration. The velocity reference signal is fed into the system as shown in the control circuit in Figure 4.1, which enables the feed-forward control. The resulting output of running the static friction can be seen in Figure 4.3a and the dynamic in Figure 4.3b. Here the velocity output of the system is plotted in the top, showing the signal before and after the LuGre model is applied. Below the velocity plot is the motor position error, also with before and after plots, while finally in the bottom the control signal from the feedback component $\boldsymbol{u_c}$ is shown for the runs before and during the application of the feed-forward component.

By observing the reduced motor position error of the system as the red curve in the middle graph shown in Figure 4.3a when the static LuGre is used, the feed-forward components compensates for the friction. However at the points in time that the velocity changes direction, a small but noticable error can be seen in the same plot. This indicates that the model did not compensate for the friction entirely. Compare this to 4.3b, where the friction component is completely captured by the feed-forward component, which can be seen from the zero control signal sent from the feed-back controller and the motor position error being zero.

Friction compensation through the use of these models can also be seen working for different reference signals, such as when the frequency of the reference signal given in (4.7) is increased to 3 Hz. This change of reference signal can be seen as case when the TCP velocity on the robot manipulator is increased, but still performing the same circular pattern.

The friction behaviour in the system can be seen to change for this reference signal as presented in Figure 4.4, where the peak of the system friction is larger. The dynamic function follows the system friction, where as the static function only follows parts of it. The output for these models show the same thing as before, that there are still errors for the static case, shown in Figure 4.5. However both models still show significant increase in the motor position tracking. This improvement is important for in order to motivate simulation on the simulated robot.

4.3.2 Robot Simulation

During simulation of the robot manipulator, reference motions at different locations and speeds are used such that performance can be evaluated. The motions considered includes a subset of

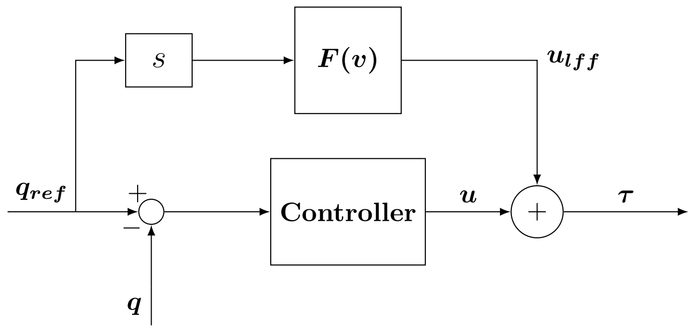
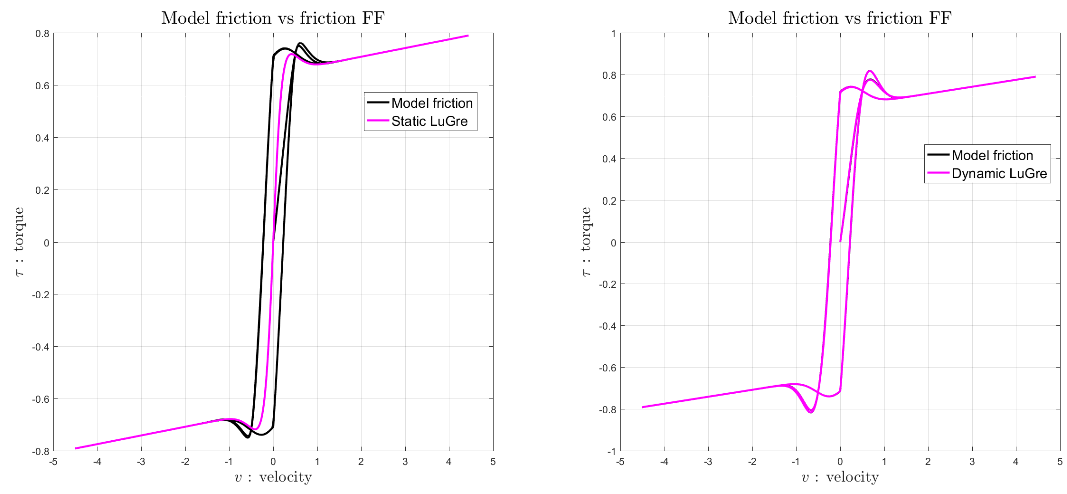


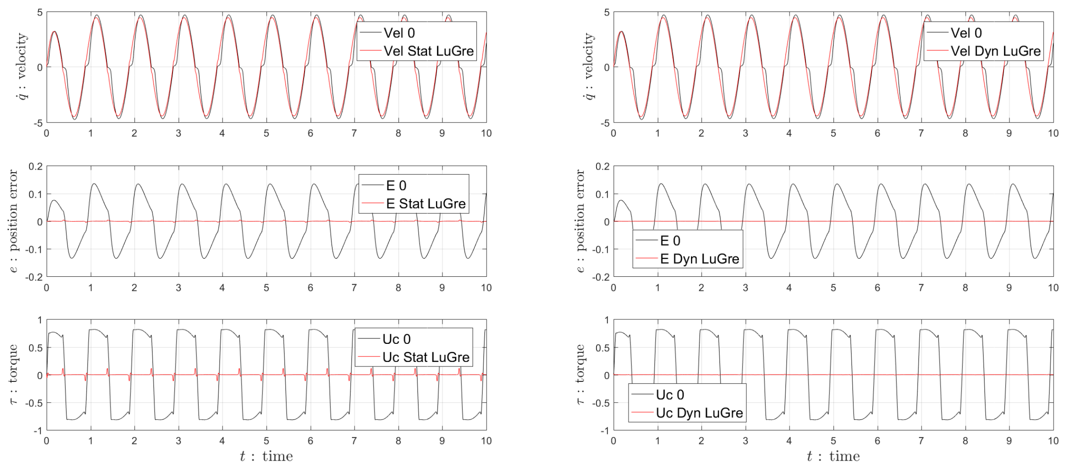
Figure 4.1: Feed-forward control $\boldsymbol{F(v)}$ for the LuGre friction models is applied to the existing control structure. The controller shown in the circuit consists of both feedback and feed-forward control.



(a) Static LuGre

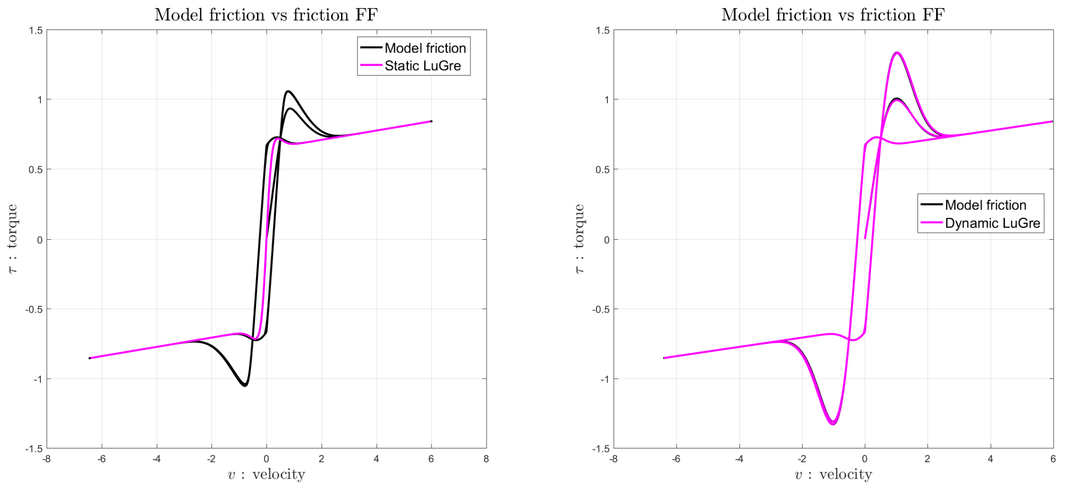
(b) Dynamic LuGre

Figure 4.2: The feed-forward output torque of the LuGre feed-forward components as the purple lines for the velocity reference signal given in (4.8), compared to the measured system friction as the black lines. The static feed-forward control makes an averaging approximation of the dynamic friction where are the dynamic follows the friction curve.



(a) Static LuGre **(b)** Dynamic LuGre

Figure 4.3: One-mass system outpus after being subject to the reference signal (4.8), before (red line) and after (black line) the LuGre components have been applied. The top graph shows the output velocity of the system, the middle graph shows the motor position error and the bottom graph shows the output of the feedback controller. There is a significant performance increase seen from the decrease in control signal and motor position error after the application of the feed-forward components. A small oscillating behaviour however can be seen for the static LuGre model when velocity changes direction.



(a) Static LuGre **(b)** Dynamic LuGre

Figure 4.4: The feed-forward output torque of the LuGre feed-forward components as the purple lines for the velocity reference signal given in (4.8) where the frequency is increased to 3Hz, compared to the measured the system friction as the black lines.

the positions presented in Figure 2.9, namely around position 1 and 3. For these positions and motions, different velocities are considered such that TCP velocity 10 mm/s, 40 mm/s and 100 mm/s are tested on each of the mentioned locations. The first case, which is considered the base case is position 1 with TCP velocity 10 mm/s.

The results of applying these models on TCP velocity 10 mm/s on position 1 can be seen from the TCP graph and the RMS graph shown in Figure 4.6. In the TCP graph in Figure 4.6a, the recorded output from the robot manipulator applying each of the two LuGre controllers, here marked as red and green are shown together with the case where no friction compensation is applied, i.e. the blue tradjectory. All of this is shown in relation to the reference circular pattern shown in black, which the robot is meant to follow.

In the same Figure, the smaller circles with their top point in the origin represent the lead-in and lead-out that is performed right before and after performing the actual circle. Lead-in is a smaller circular TCP motion that is performed prior to performing a circle, where as the Lead-out is the same motion but after the circle is complete, which brings TCP back to its initial position. This is done in order to make a smoother start and finish of the TCP and is not considered as a part of the result.

It can be seen in Figure 4.6a that the usage each individual LuGre component contributes to the roundness of the circle trajectory. It can also be observed that for the larger circles, TCP errors are generally smaller. If the smallest circle with its center in the origin is observed, it can be seen that the dynamic LuGre model helps the robot manipulator perform a more smooth motion, in comparison to the static LuGre model.

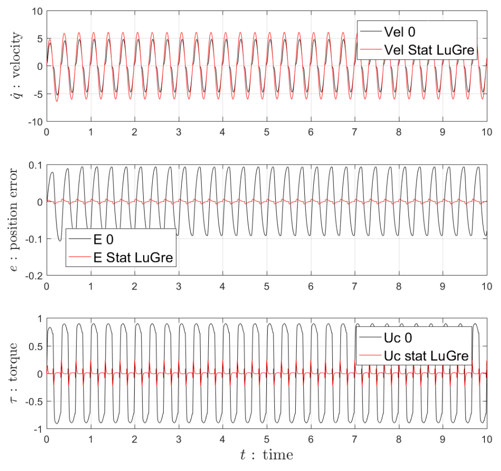
In Figure 4.6b the RMS of the motor position error is shown for same motion. The equation for calculating the RMS is given in (2.17). In the calculation of the motor position error RMS measurements where the robot is standing still have been removed. In the Figure the static and dynamic LuGre component as well as the case without any friction compensation are shown and compared to each other. It can also be seen from this graph that the overall motor position error is smaller for the used models. Between the static and dynamic model, the static LuGre can be seen better for joint 3, where as in the rest of joints, the dynamic has lesser RMS error, however on joint 1 and 2, the difference is small.

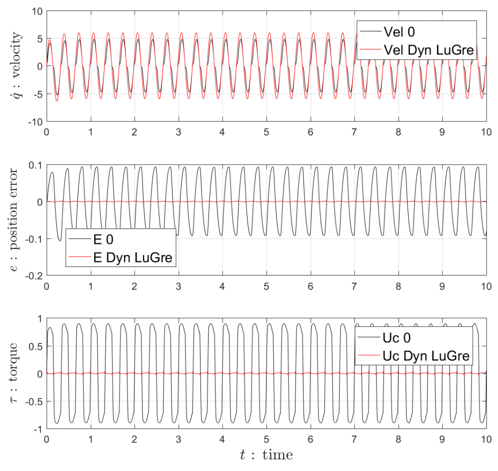
In simulation around position 3 with the same TCP velocity, namely 10 mm/s, the same phenomenon can be observed as for the first position, namely that the circle roundness in TCP increases while the RMS for the motor position error decreases when the feed-forward components are applied. This can be seen from the Figures 4.7a and 4.7b. The dynamic LuGre can also be seen to perform worse in regards to the RMS for joint 3.

For tests conducted at higher TCP velocity, both using the feed-forward components and without, it can be seen that for smaller circles the TCP of the robot has trouble following the circular path. However, the results gained from the feed-forward components still show a decrease in the TCP path error. This can be seen in the case when the TCP velocity is 40 mm/s and 100 mm/s while motion is run around the first position, namely in Figure 4.8 and 4.9 respectively. Note also that in the RMS graph for motor position error, the feed-forward components provide less over all error in comparison to without the feed-forward component. The friction compensation therefore shows improvement in regards to the motor position tracking and the TCP error, despite the general TCP performance being bad.

4.4 Robot Experiments

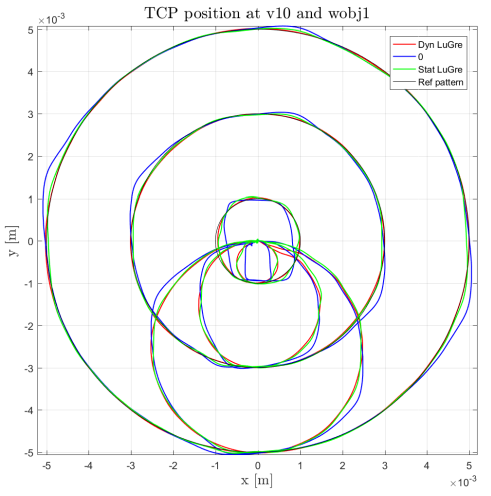
Implementation of the LuGre feed-forward components on real robot is the last step in examining the performance of the models. Prior tests showed improved results on simulation, which motivates the test on real robot. The tests conducted here are the same as presented in Section 2.2.5. In contrast to the robot simulation, in this section TCP path error will be examined based on

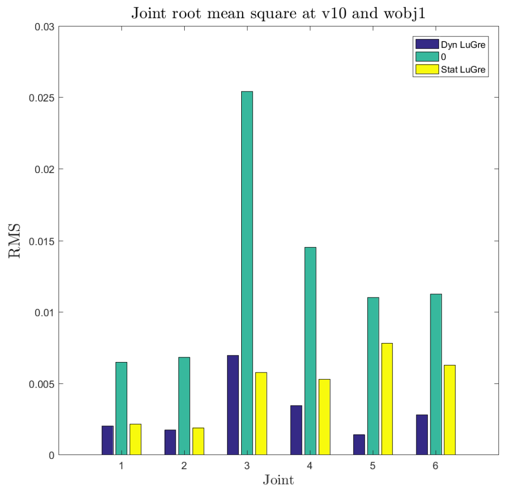




(a) Static LuGre. **(b)** Dynamic LuGre.

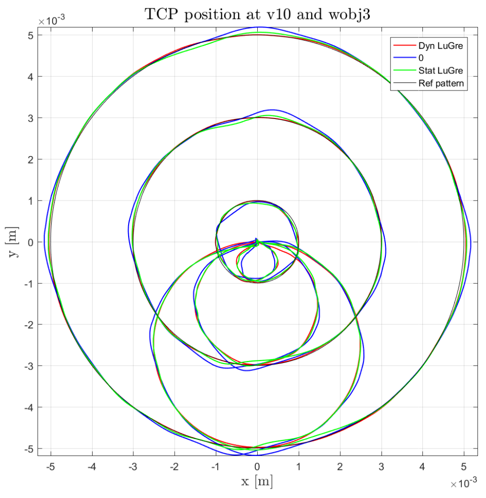
Figure 4.5: The output of the One-mass system after being subject to the reference signal (4.8) with frequency of 3Hz, before and after the LuGre components have been applied. The top graph shows the output velocity of the system, the middle graph shows the motor position error and the bottom graph shows the output of the feedback controller. Motor position error is reduced, but not as much as could be seen in Figure 4.3.

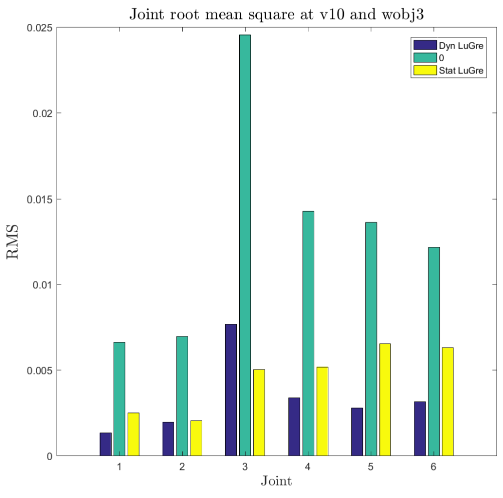




(a) TCP path during motion. **(b)** Root mean square of the motor position error.

Figure 4.6: Simulation results at TCP velocity 10 mm/s around position 1, where the TCP and the RMS of the motor position error is compared for cases with and without the applied LuGre components. The legend 0 marks the case where no friction compensation is applied. The smaller circles with their top in the origin represents the lead-in and lead-out. Both components show increased accuracy of following the black circles, where the deviation from paths are smaller. The RMS can be seen to be lower for the dynamic LuGre model, with exception of joint 3.

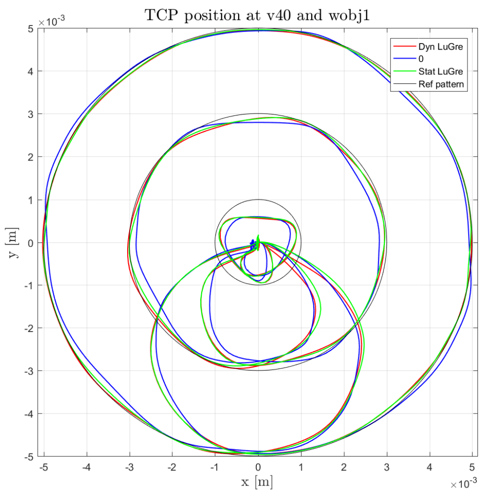


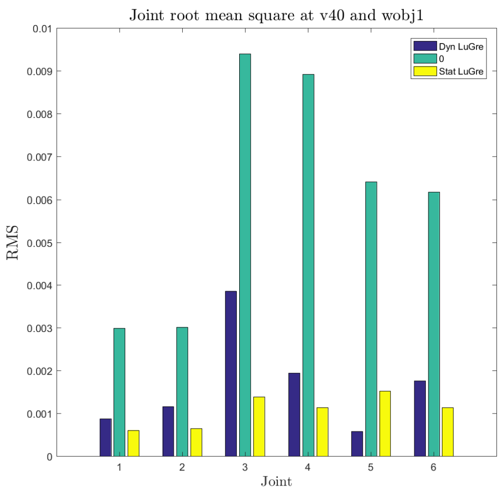


(a) TCP path during motion.

(b) Root mean square of the motor position error.

Figure 4.7: Simulation results at TCP velocity 10 mm/s around position 3, where the TCP and the RMS of the motor position error is compared for cases with and without the applied LuGre components. The result here is the same as in the first tested position, which indicates that the methods perform similarly on different locations at the same velocities.





(a) TCP path during motion.

(b) Root mean square of the motor position error.

Figure 4.8: Simulation results at TCP velocity 40 mm/s around position 1, where the TCP and the RMS of the motor position error is compared for cases with and without the applied LuGre components. Both models show problems with following the inner circle and however TCP path error is still improved for the other circles. Despite this the RMS of motor position error is significantly decreased.

maximum deviation from path and standard deviation from path i.e. RMS for TCP error, such that the improvement can be examined in detail.

The result from performing motions around position 1 at 10 mm/s TCP velocity shows that the LuGre friction components in general improve the TCP error, however does not achieve full path following accuracy of the circles. In the Figure 4.10a the smallest circle can be seen to be followed more accurately and has more roundness, especially in the case when the dynamic LuGre is applied. It can be seen however that in the top of the circles, the friction compensation reduces the deviation from the path, but not fully. In the case of the RMS of the motor position error for the same test in Figure 4.10b, the error can be seen significantly reduced. Here however it is not clear that the dynamic LuGre model is superior to the static, since it only has lesser RMS error for joint 2. Observation of the same TCP velocity, motion i.e 10 mm/s, but performed on the other positions, showed the same results. The plots are not shown here due to space.

Friction compensation at higher velocities on the real robot manipulator shows an inability to track the TCP reference, which also was observed in simulation. The application of feed-forward components however contributes to less deviation and more roundness. This can be seen for TCP velocity 40 mm/s in Figure 4.11a and TCP velocity 100 mm/s in Figure 4.12a when performed on position 1. The latter shows great difficulty following any of the TCP reference circles. As seen in simulation the RMS for motor position error is still reduced for the higher velocities, despite the performance of in the TCP. This is displayed in the Figures parallel to the previous TCP graphs, namely Figures 4.11b and 4.12b.

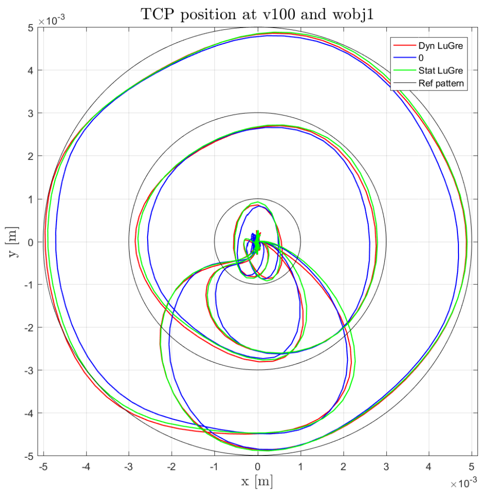
To further investigate the performance of these models in the application on the robot, the maximum deviation from the TCP path and the RMS of the path error for all four positions can be observed in Table 4.1. The tables display the results for each individual circle at each location. From all of these tables it can be seen that the dynamic model outperforms the static in every location except during the motion of 5 mm radius circle on position 3. It can also be observed that both models perform significantly better regards in regards to these quantities, compare to not having friction compensation. This indicates that performance does increase.

At higher velocities, such as 40 mm/s the change in maximum path deviation and RMS of the path error from no compensation to the LuGre models is not as large as before. This can be viewed in Table 4.2. It can also be seen that the difference between the LuGre models has also decreased. The results from these test are in line with the observed TCP, that the improvement at larger velocities is not as large.

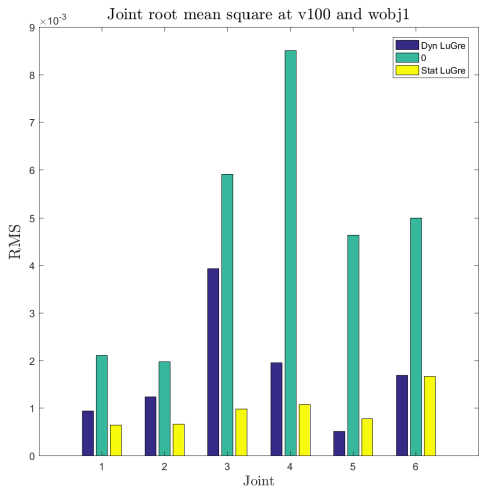
If the results displayed in this section is compared to the friction compensation observed during simulation in Section 4.3, it can be seen that almost the same TCP behaviour appears before and during feed-forward compensation. The deviations from the TCP paths which can be seen at the points in time where the TCP changes traveling direction in the workspace, can also be seen both in simulation and in real application. An example of this can be viewed during the motion of 10 mm/s TCP velocity around position 1, which is shown for the simulation in Figure 4.6a and in practice in Figure 4.10a. The TCP deviations can be seen to be slightly different in practice and that the friction compensation does not manage to remove the deviation as notably as done in simulation. This indicates that there might be a difference in model accuracy of the friction models in reality compared to simulation.

4.5 Evaluation of Performance

The result of these tests have shown that the LuGre models manage to reduce the motor position error and the TCP error for both different TCP velocities and for different positions. In the case of the dynamic LuGre feed-forward component applied in simulation, where the friction applied is a matching model of the system friction, it is implied and shown that performance increases. Due to the fact that effects seen in simulation and on the real robot are similar, the simulated model can

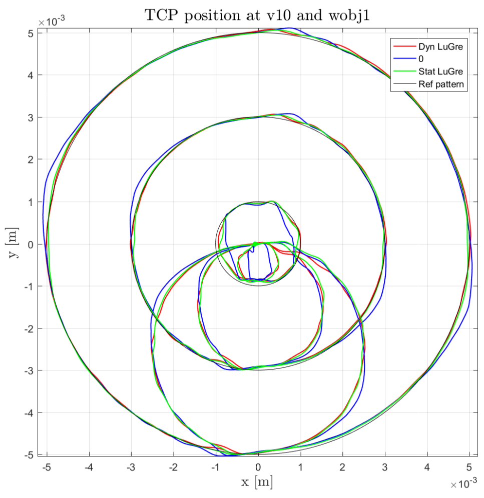


(a) TCP path during motion.

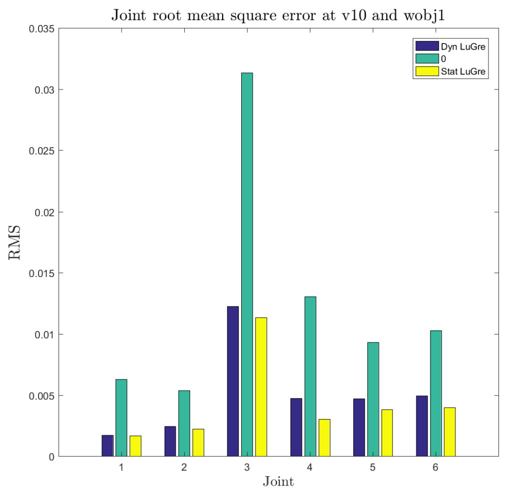


(b) Root mean square of the motor position error.

Figure 4.9: Simulation results at TCP velocity 100 mm/s around position 1, where the TCP and the RMS of the motor position error is compared for cases with and without the applied LuGre components. All models have trouble with following the reference circles, however some improvement can be seen, both in TCP and in the RMS.

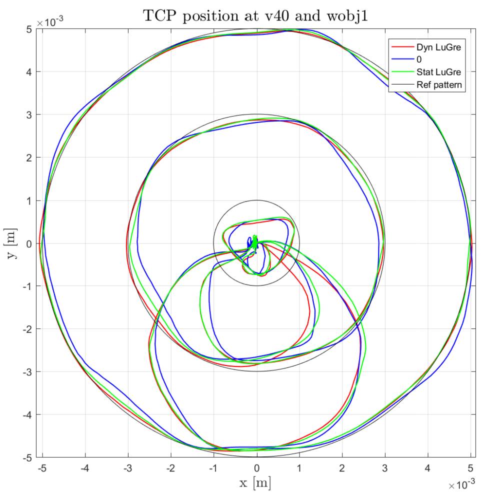


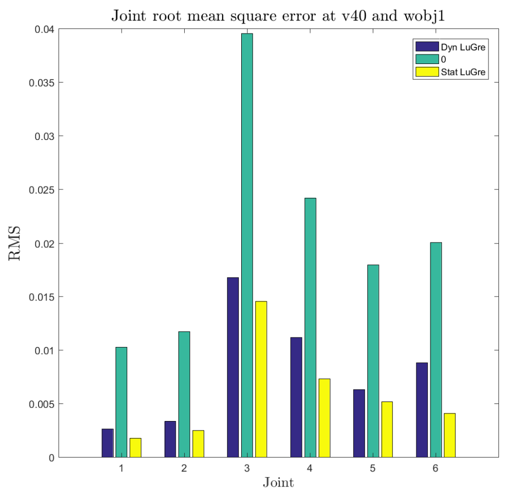
(a) TCP path during motion.



(b) Root mean square of the motor position error.

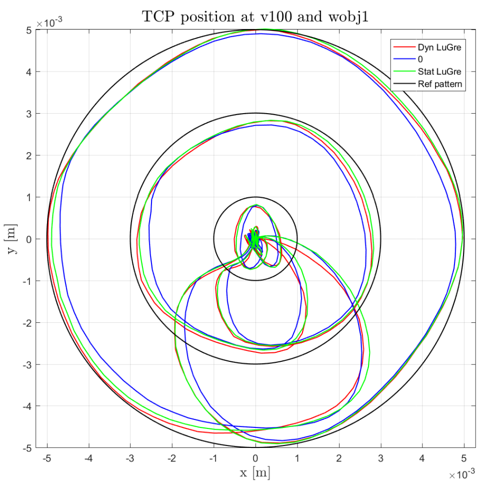
Figure 4.10: Application results on real robot at TCP velocity 10 mm/s around position 1, where the TCP and the RMS of the motor position error is compared for cases with and without the applied LuGre components. The real application shows similar results to the simulation case, which is that the TCP path error is decreased. The motor position error RMS can also be seen to be reduced.

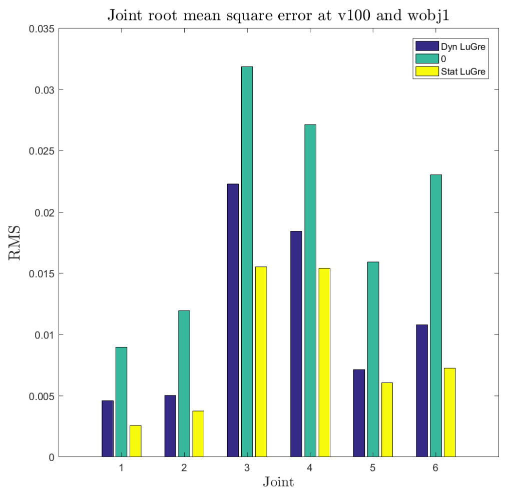




(a) TCP path during motion. **(b)** Root mean square of the motor position error.

Figure 4.11: Real application results at TCP velocity 40 mm/s around position 1, where the TCP and the RMS of the motor position error is compared for cases with and without the applied LuGre components. Results are similar to the simulation case, which is that the models provide limited performance increase to the smallest circle. For the rest however the performance is more noteworthy. RMS can also be seen reduced, here however the dynamic model provides less performance increase than the static model in terms of RMS.





(a) TCP path during motion. **(b)** Root mean square of the motor position error.

Figure 4.12: Real application results at TCP velocity 100 mm/s around position 1, where the TCP and the RMS of the motor position error is compared for cases with and without the applied LuGre components. Very limited performance increase can be seen to the TCP error and RMS position error.

Table 4.1: Maximum deviation from the reference TCP path and root mean square of the path error in *mm* for each individual circle for the cases where the LuGre feed-forward components are used and not used. The reference motion used corresponds to 10 *mm/s* TCP velocity. The tables show improvement when friction compensation is applied, as well as that the dynamic model performs better than the static in almost all cases.

(a) Max deviation without friction compensation.

	Pos. 1	Pos. 2	Pos. 3	Pos. 4
Circ. r1	0.31	0.24	0.23	0.31
Circ. r3	0.18	0.26	0.3	0.22
Circ. r5	0.23	0.26	0.27	0.27

(b) RMS without feed-forward.

	Pos. 1	Pos. 2	Pos. 3	Pos. 4
Circ. r1	0.14	0.109	0.122	0.138
Circ. r3	0.074	0.095	0.109	0.098
Circ. r5	0.078	0.091	0.106	0.105

(c) Max deviation with static LuGre.

	Pos. 1	Pos. 2	Pos. 3	Pos. 4
Circ. r1	0.15	0.19	0.15	0.13
Circ. r3	0.11	0.18	0.17	0.11
Circ. r5	0.11	0.19	0.12	0.15

(d) RMS with static LuGre.

	Pos. 1	Pos. 2	Pos. 3	Pos. 4
Circ. r1	0.068	0.081	0.075	0.06
Circ. r3	0.045	0.077	0.066	0.047
Circ. r5	0.041	0.075	0.058	0.062

(e) Max deviation with dynamic LuGre.

	Pos. 1	Pos. 2	Pos. 3	Pos. 4
Circ. r1	0.13	0.14	0.13	0.12
Circ. r3	0.07	0.13	0.1	0.09
Circ. r5	0.07	0.11	0.13	0.09

(f) RMS with dynamic LuGre.

	Pos. 1	Pos. 2	Pos. 3	Pos. 4
Circ. r1	0.067	0.068	0.067	0.056
Circ. r3	0.03	0.049	0.037	0.04
Circ. r5	0.032	0.051	0.042	0.041

Table 4.2: Maximum deviation from the reference TCP path and root mean square of the path error in *mm* for each individual circle for the cases where the LuGre feed-forward components are used and not used. The reference motion used corresponds to 40 *mm/s* TCP velocity. Improvement when friction compensation is applied can here be seen not a large as for lower TCP velocities. However both models show increased performance, where the dynamical is in general better.

(a) Max deviation of TCP without feed-forward.

	Pos. 1	Pos. 2	Pos. 3	Pos. 4
Circ. r1	0.5	0.63	0.91	0.58
Circ. r3	0.41	0.35	0.38	0.46
Circ. r5	0.31	0.28	0.35	0.36

(b) RMS of TCP without feed-forward.

	Pos. 1	Pos. 2	Pos. 3	Pos. 4
Circ. r1	0.362	0.324	0.41	0.343
Circ. r3	0.187	0.188	0.206	0.209
Circ. r5	0.12	0.141	0.14	0.147

(c) Max deviation of TCP with static LuGre.

	Pos. 1	Pos. 2	Pos. 3	Pos. 4
Circ. r1	0.48	0.52	0.93	0.59
Circ. r3	0.2	0.3	0.32	0.23
Circ. r5	0.14	0.25	0.18	0.2

(d) RMS of TCP with static LuGre.

	Pos. 1	Pos. 2	Pos. 3	Pos. 4
Circ. r1	0.307	0.263	0.378	0.28
Circ. r3	0.109	0.133	0.133	0.117
Circ. r5	0.066	0.115	0.084	0.083

(e) Max deviation of TCP with dynamic LuGre.

	Pos. 1	Pos. 2	Pos. 3	Pos. 4
Circ. r1	0.55	0.58	0.89	0.65
Circ. r3	0.17	0.24	0.31	0.24
Circ. r5	0.15	0.2	0.17	0.21

(f) RMS of TCP with static LuGre.

	Pos. 1	Pos. 2	Pos. 3	Pos. 4
Circ. r1	0.309	0.281	0.374	0.289
Circ. r3	0.09	0.131	0.131	0.111
Circ. r5	0.063	0.103	0.075	0.077

be seen as a good approximation of the real robot manipulator. Knowing this, the compensation through the use of the dynamic LuGre model on the real robot can therefore be seen as capturing some part of the friction within the system.

The static component is observed to be performing similarly to the dynamic component, in that motor position error and TCP error is decreased in comparison to not using friction compensation. This can be seen from both simulation and the component applied in practice. Already in the one-axis model the static friction shows large gain in motor position tracking, compared to the dynamic which only performs slightly better. Further testing follows the same pattern, where both TCP tracking and motor position error is improved significantly. The static component can thus be seen to also be a valid friction compensation, since performance is similar to the dynamic case.

As the result also show, neither of the models manage to reduce the TCP path error significantly at higher velocities, despite showing improvement to the motor position tracking. As indicated from simulation, even when applying an ideal friction function, the friction compensation fails to remove all of the TCP path error. This could be the result of two things, namely that the friction model inaccurately describes the friction at higher velocities, or that other effects in the system become more dominant at higher velocities. It is known that the flexibilities within the system are more likely to effect the performance of the TCP path error when the robot manipulator performs fast motions. It could therefore be a potential explanation for the result seen during application of the feed-forward components.

In regards to the objective for friction compensation using feed-forward control stated in given in Section 1.2, the implemented LuGre model shows improved results for the multiple given scenarios within the given workspace. The models presented in this chapter fulfill this objective through the performance increase presented. Within this scope, the LuGre model used in feed-forward control can be considered an acceptable friction compensation method.

It should be stated however that the result documented in this chapter show the behaviour and performance of a parametrized model which are given prior to this thesis. Information about the methodology of the parameters estimated is therefor not known, which limits the analysis on the validity of the models used on the real robot system. Effects such as deterioration or simply small changes over time might cause the model tested here to not be as accurate anymore. Therefore performing parameter estimation of the LuGre on the real robot again is an interesting aspect in terms of performance. The validation and application of such a model could show a potential improvement to the friction compensation.

5 Learned Feed-Forward Control with BSN

Learned feed-forward control (LFFC) can be viewed as a way of iteratively approximating a model structure aimed for direct compensation of unwanted system properties. In contrast to the previously described approach of applying an identified model as a feed-forward component, this methodology can be described as a way of iteratively learning a black-box structure during operation of the robot manipulator system. The goal here is instead to capture the effects inside the robot manipulator system by learning from the observable contents of the controller, which in theory should contain information about the system properties not yet captured by a feed-forward model.

This chapter therefore aims to present the approach of using Learned Feed-Forward Control using B-spline networks in order reduce the effects of friction in motors, such that the performance of the robot manipulator system in regards to the accuracy of motor position and tool center position is increased. The chapter is distributed such that first the design of two different B-spline networks is presented in Section 5.1, followed by a description of how the B-spline networks are trained as feed-forward components in Section 5.2. These feed-forward components are then evaluated in simulation and in practice on a real robot in order to show that performance is increased, which is presented in Sections 5.3 and 5.4. The chapter is concluded with a Section where the results are evaluated, namely Section 5.5.

5.1 LFFC BSN Design

The design of the LFFC requires that the user makes a certain amount of choices regarding the model structure and function approximation. In regard to the robot manipulator system described in this thesis, friction in motors are likely to be the largest source of error, and is therefore a reasonable system effect to compensate for. As for friction compensation using LFFC, B-spline networks (BSN) have been seen used as a function approximator on more than one occasion, partly due to the ability of incorporating prior knowledge about the system in order to achieve more accurate learning. In [16], the authors apply parsimonious LFFC using BSN as a function approximator in order to learn friction compensation on a motor system. The aim with parsimonious LFFC was to avoid complex issues such as the curse of dimensionality and learning from correlated data, which has been mentioned in Section 3.2.2. The achievement of applying friction compensation in this manner is a strong motivation to investigating its properties on a full scale robot manipulator system, containing more than one motor.

Much like the feed-forward control using the LuGre friction model, the purpose of LFFC in this scenario is to compensate for the effects of friction. A general assumption in modeling friction is that the effect is mostly dependent on the velocity of the motor, independent of the other motors in a robot manipulator system. This emphasizes the parsimonious structure of LFFC in that a reduced dimensions lead to better learning. The same assumption can therefore be applied to the design of a LFFC using BSN, such that a feed-forward component should take as input only the velocity of the motor and map said velocity to a torque signal.

Using a LFFC with BSN function approximator with only the velocity as input requires consideration due to the dynamic nature of friction and the fact that a BSN is a static function approximator. Learning friction is therefore problematic in the sense that the friction will never truly be approximated by using a single-input single-output (SISO) model in this manner. Therefore two different models have been designed, much like the approach of using both static and dynamic LuGre model as a feed-forward component, namely a static BSN and a extended BSN.

By simply performing learning using a SISO LFFC with BSN as is, a static friction model can be made to compensate for parts of the friction behaviour, marking the first designed model. The second model focuses on capturing parts of the dynamic behaviour of friction, which is done through the use of two different BSN, each using a learning signal which has been filtered based on the angular acceleration reference signal of the motors. In the following subsection, a more detailed description of the design of these models is presented, where the design of the static BSN is covered first, followed by the extended BSN.

5.1.1 Static BSN

The static BSN LFFC is designed as SISO model where the velocity is mapped to output friction torque in order to compensate for friction as a feed-forward component. This can be written as

$$\boldsymbol{u_{lff}} = \boldsymbol{BSN(\dot{q})} = \begin{pmatrix} BSN(\dot{q}_1^{ref}) \\ BSN(\dot{q}_2^{ref}) \\ BSN(\dot{q}_3^{ref}) \\ BSN(\dot{q}_4^{ref}) \\ BSN(\dot{q}_5^{ref}) \\ BSN(\dot{q}_6^{ref}) \end{pmatrix} \tag{5.1}$$

where $\boldsymbol{u_{lff}}$ is the vector of control signals sent as an additive component to the control of the motors. The feed-forward component $\boldsymbol{BSN(\dot{q})}$ can therefore be seen as a component containing 6 B-spline networks, where each individual motor is sent a scalar signal from a BSN that is only dependent on the same motor velocity $\dot{q}_i^{ref}$.

The application of this component can be seen in Figure 5.1. The static BSN, namely $\boldsymbol{BSN(\dot{q})}$ is implemented in the same manner as the feed-forward component used in LuGre friction, presented in Figure 4.1. The difference between the control circuits however is the added signal $\boldsymbol{z}$ in the Figure. In previous works [7], [8], [16] the learning signal in the control circuit of a LFFC is not shown in this way, but instead directly taking the feedback control signal $\boldsymbol{u_c}$ as a learning signal. The representation displayed here simply states that a number of different signals can be extracted from the feedback control in order to be used in learning.

An important design parameter in the construction of BSN is the number knots in the network as well as the placement and width of these knots. As presented in 3.2.1, there is a trade-off between performance, modeling noise and time complexity. In the design of the static BSN, the recorded control signal in terms of velocity of the robot manipulator is considered, which is shown in Figure 5.2. Here the control signal from the feedback control $\boldsymbol{u_c}$ for each joint is displayed for the motion of performing circular TCP motions at the largest considered velocity, 100 mm/s. It can be seen that the largest velocity interval for which the control output is given is within the interval of $[-20, 20]rad/s$. The entire knot interval, which also corresponds to the input space is therefore set to be this interval. This is also the case for the extended BSN, described in Section 5.1.2.

The density of the knots, i.e. the width and the number of knots is set to be higher within the interval $[-1, 1]rad/s$ due to the effect of the nonlinearity being larger within the interval, especially for lower velocity motions. A total of 41 knots have been evenly defined within the smaller interval,

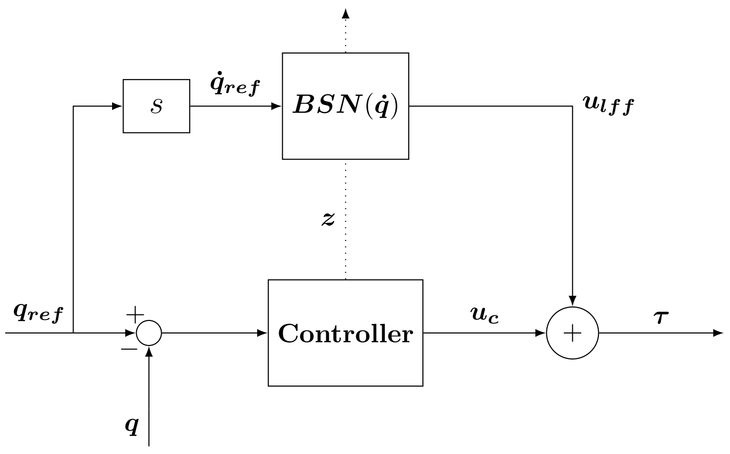


Figure 5.1: Feed-forward control using LFFC with static BSN where the angular reference $\dot{\boldsymbol{q}}_{\boldsymbol{ref}}$ is mapped by the $\boldsymbol{BSN}$ component to a torque output. The signal $\boldsymbol{z}$ represents possible contents of the controller which could be used for training signal.

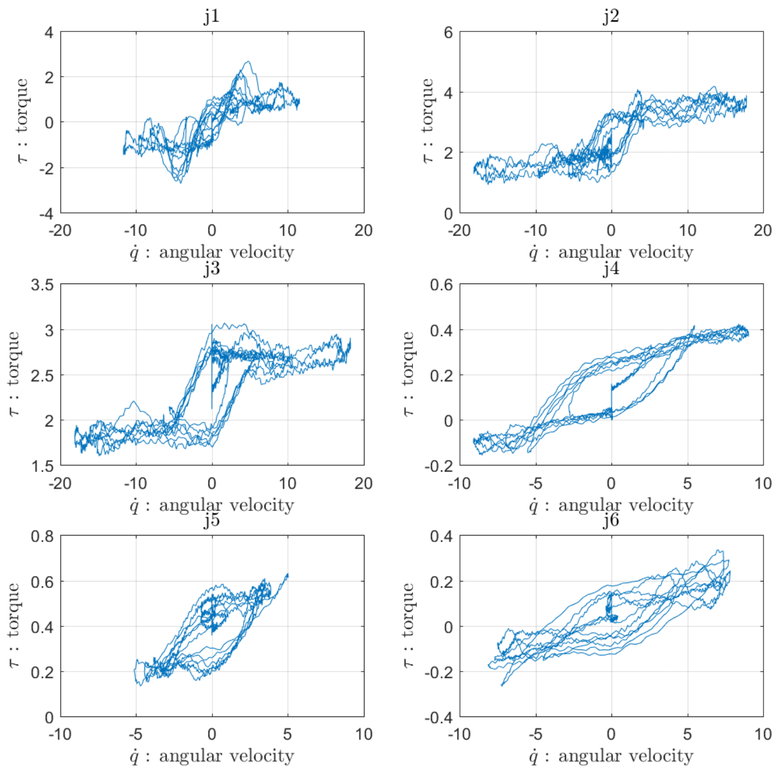


Figure 5.2: Feedback control signal output given in terms of velocity for all joints when performing circular TCP motions at 100 mm/s. The largest velocity interval for which the control signal is given is $[-20, 20]rad/s$ for joint 3, thus the input space in the knot vector is given as this interval.

where as outside of this interval 38 knots reside with equal spacing in between. To further enforce the nonlinear learning, second order B-splines were used. The resulting distribution of B-splines in the BSN can be seen in Figure 5.3. The density of B-splines can be seen to be higher within the described interval, due to the width being smaller compared to the rest of the B-splines.

5.1.2 Extended BSN

The extended BSN as a feed-forward that aims to approximate the dynamic behaviour of the friction by applying two BSN in the same feed-forward component, utilizing the input reference velocity and its corresponding acceleration. Depending on the direction of the input acceleration, the training signal and the input signal is fed to either of the BSN, one BSN handling accelerations larger or equal to 0, while the other takes care of the rest. This can be visualized through the following function

$$\boldsymbol{H}(\dot{q}_i) = \begin{cases} \boldsymbol{H_1}(\dot{q}_i) & , \ddot{q}_i \geq 0 \\ \boldsymbol{H_2}(\dot{q}_i) & , \ddot{q}_i < 0 \end{cases} \tag{5.2}$$

where $\boldsymbol{H}(\dot{q}_i)$ represents the output of the LFFC on axis i through the use of the two BSN's. The functions $\boldsymbol{H_1}(\dot{q}_i)$ and $\boldsymbol{H_2}(\dot{q}_i)$ are essentially the same, with the difference being that they receive different training data during the learning procedure. Note that both of these functions still correspond to SISO models, despite using the acceleration direction to choose inputs and learning signals.

This function is based on the works of [16], where the authors make the case that the stiction may be dependent on the case where a motor starts or stops. This can be translated to that the friction model is partly dependent on the direction of the acceleration and therefore promoting the use of two BSN where the usage of each is determined by the acceleration. By simply observing the reference acceleration, both the input data and learning data can be filtered and used in each BSN and thus approximating a dynamic behaviour, much like the one of the LuGre model.

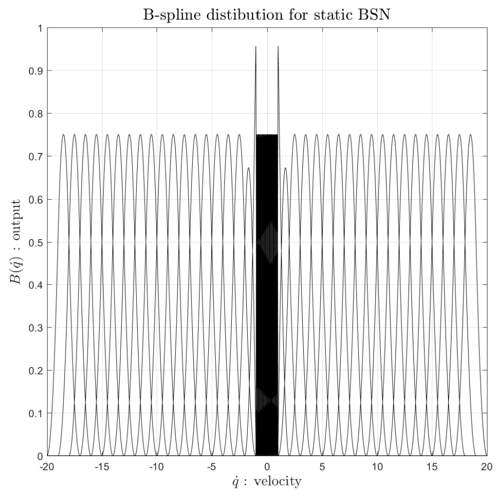
The application of the extended BSN can therefore be seen as a modified version of the static BSN previously presented in Section 5.1.1. This is illustrated in Figure 5.4 where the acceleration is included in the feed-forward component, but not as an input in the conventional sense. $\boldsymbol{H_1}(\dot{q}_i)$ is then trained with only inputs and system data $\boldsymbol{z}$ when the acceleration is larger or equal to zero, where as $\boldsymbol{H_2}(\dot{q}_i)$ is only trained with data when the acceleration is less than zero. The actual training signals used in $\boldsymbol{z}$ is the same as in the static BSN function.

Regarding the distribution of knots in $\boldsymbol{H_1}(\dot{q}_i)$ and $\boldsymbol{H_2}(\dot{q}_i)$, it can be seen following the same pattern as described in Section 5.1.1 and shown in Figure 5.3. The range of the velocity reference values and the nonlinearity remains the same for both the static and extended BSN's, which motivates the use of the same number of knots, knots placement and knot density.

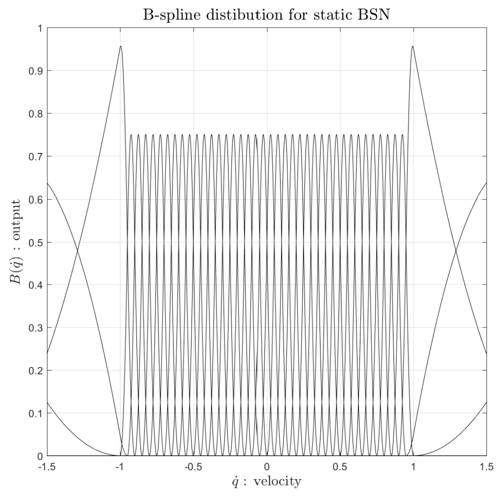
5.2 Training the B-spline networks

Training a B-spline network involves updating the weights during a procedure where the BSN aim is to reduce the effects of friction within the system, such that performance is increased. To enable this, two important aspects must be considered, namely the choice of training signal which promotes learning of the friction and the choice of training procedure of which the B-spline networks updated.

In the following sections, these two aspects will be discussed. The choice of training signal can be seen presented in Section 5.2.1, while the training procedure can be seen presented in Section 5.2.2.



(a) All of the defined B-splines.



(b) Closeup on the B-splines defined within the interval $[-1, 1]$ rad/s.

Figure 5.3: The graphs show the distribution of the B-splines in the designed BSN feed-forward controllers. The same distribution is used for all of the motors in the robot manipulator system.

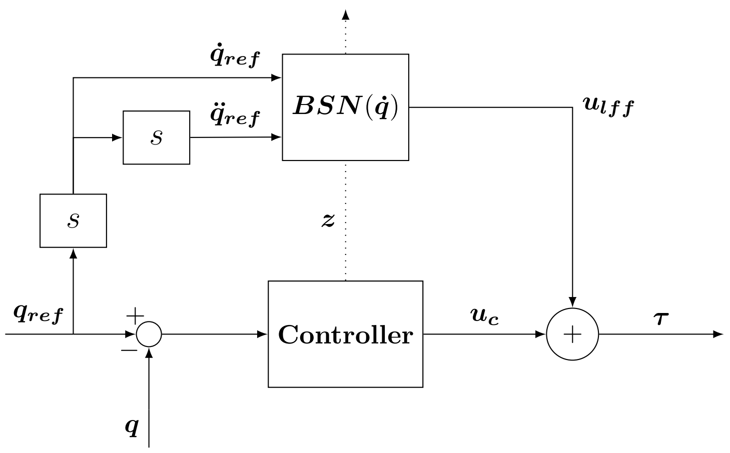


Figure 5.4: Feed-forward control using LFFC with static BSN where the angular reference $\dot{q}_{ref}$ is mapped by the BSN component to a torque output. The acceleration $\ddot{q}_{ref}$ is here used in order to determine which of the two B-spline networks to use, both in training and in application. The signal z represents possible contents of the controller which could be used for training signal.

5.2.1 Choice of training signals

In order to enable learning of the system friction within the B-spline networks, a learning signal must be chosen. The weights in the BSN are adapted according to the learning rule (3.18), which is based on the gradient of a cost function being minimized. The cost function in itself is a measure of the performance of the BSN applied to the system. A well performing BSN in this case should compensate for the friction in such a manner that the motor position tracking and preferably the tool center position tracking is increased. It is therefore an important aspect of the BSN design that the appropriate learning signal is chosen.

Considering the control application at hand, measurable data on the arm side of the robot, such as the tool center position is not considered in the learning scenario. Therefore the choice of training signals are limited to the motor side, where the available training signals are restricted to the motor reference signals, controller input and contents of the controller. This can be interpreted as the signals named in the Figures 5.1 and 5.4.

A natural choice of learning signal $\boldsymbol{z}$ would be the position error, namely

$$\boldsymbol{z} = \boldsymbol{e_p} = \boldsymbol{q_{ref}} - \boldsymbol{q} \tag{5.3}$$

since the primary objective with LFFC is to increase the position tracking, which corresponds to a reduction of the position error in the motors. However reduced position error also means a reduction in the control signal in the feedback control, which inherently contains more information about the system dynamics and therefore also friction. In Section 3.2, the use of the control signal in learning was implied leading to the learning of the inverse plant dynamics, which could embody the repeatable system disturbances. Furthermore, in implementations of friction compensating LFFC using BSN, such as [16],[19] the feedback control signal was used as the learning signal.

In the case of the robot manipulator system examined in this thesis, shown in Figure 2.5 the control signals $\boldsymbol{u_c}$ and $\boldsymbol{u_{ff}}$ contains the feedback control signal as well as the feedforward signal. This is further explained in Section 2.2.1 where the feed-forward control compensates for the Coriolis and centrifugal components $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}$ as well as the Inertia component $\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}}$ in the modified inverse dynamics model (2.11). The remaining effects, such as the disturbances, friction and the gravity components can be seen compensated for by the feedback controller in its signal $\boldsymbol{u_c}$. The feedback control signal can thus be considered undesired as the gravity component is not part of the learning scheme.

Therefore an additional signal is chosen as learning signal, namely the signal $\boldsymbol{u_p}$. This signal is essentially the difference between the feedback control signal $\boldsymbol{u_c}$ and the signal generated by an integral part within the feedback controller, here named $\boldsymbol{u_I}$. The integral part is meant to be a compensation for the unmodeled gravity in the feed-forward control of the system. Thus extracting this signal from the controller, the new learning signal can be calculated as

$$\boldsymbol{z} = \boldsymbol{u_p} = \boldsymbol{u_c} - \boldsymbol{u_I} \tag{5.4}$$

Further motivation of this selection of signals can be seen in the Figure 5.5 where the feedback control signal $\boldsymbol{u_c}$ for motor joint 2 in the robot manipulator system is plotted against the control signal $\boldsymbol{u_p}$ during the process of performing circular TCP motions at 10 mm/s. As can be seen from the figure, the mean of the signal is shifted in the $\boldsymbol{u_p}$ component, indicating that the signal $\boldsymbol{u_p}$ is without gravitation.

5.2.2 Training procedure

When the designed B-spline networks are deployed, they must first be trained using the defined learning signal. In this thesis, due to system limitations, only offline learning is considered. Therefore a specific training procedure has been defined in order to properly train the B-spline network.

The training procedure is initiated when a training reference signal is sent to the system during simulation or task execution. During the runtime of the system, the training signal, namely $\boldsymbol{u_p}$ is continuously recorded. Upon termination of the training reference signal, the recorded signal is then used together with the reference training signal in order to update the BSN weights according to the weight adaption rule presented in (3.18). The adaption of weights for the static BSN and the dynamic approximation of BSN are indifferent, such that for each BSN in the each of the cases, the weights are updated according to

$$\Delta w_{i,j} = \gamma \frac{\sum_{k}^{N} u_{i,p}(k)\mu(\dot{q}_i^{ref}(k))}{\sum_{k}^{N} \mu(\dot{q}_i^{ref}(k))} \tag{5.5}$$

where i denotes the joint and j denotes the specific B-spline in the function. N is the number of samples that were recorded during the execution of the system, where in extended BSN case it will a subset of all recorded samples based on acceleration of the recorded samples. $\dot{q}_i^{ref}$ is the velocity reference signal and $u_{i,p}$ the recorded training data $\boldsymbol{u_p}$ on axis i. Since the update is performed offline, the adaption is normalized, as can be seen by the denominator. After the weights have been updated, the same training procedure is done for a number of iterations, where the number is decided beforehand.

In order to validate the performance of the training such that termination of the training can be decided, the mean squared error from each iteration is used to calculate the gradient of both the training signal and the motor position error in between iterations. The mean squared error can be seen as a adaption of the cost function presented in (3.14) and is calculated as

$$MSE(k) = \frac{1}{N_k} \sum_{i=1}^{N_k} \boldsymbol{e}_i^2(k) \tag{5.6}$$

where $\boldsymbol{e}$ is the motor position or the training signal. k represents the training iteration where N_k is the number of sampled data points during that iteration. This can be further used in the calculation of the gradient, namely

$$\Delta MSE(k) = MSE(k) - MSE(k-1) = \frac{1}{N_k} \sum_{i=1}^{N_k} \boldsymbol{e}_i^2(k) - \frac{1}{N_{k-1}} \sum_{i=1}^{N_{k-1}} \boldsymbol{e}_i^2(k-1) \tag{5.7}$$

In the case that the gradient of both is decreasing within a set level of tolerance, the training is continued. The training may also be continued in the case of one or more axis showing improvement. In this case the weights are locked for the joints that have not shown improvement. Should the cost instead be seen increasing, the lowest earlier iteration of weights are chosen as the desired weights. In both simulation and application of the B-spline networks on robot, mean squared error for these signals are shown after each iteration as an illustration of the learning.

5.3 Model Simulation

Similar to the simulation of the LuGre friction model, the BSN feed-forward components discussed in this chapter are both tested in the one-axis model simulation and in robot simulation. The simulation of the B-spline networks are done in two phases, namely the training phase and the evaluation phase. The training phase is when a reference signal is sent to the system and weights are updated after the reference signal has been completed. This is further explained in Section 5.2.2. Evaluation of the models is done after the training procedure has been completed. During the evaluation step several reference signals are run, but without updating the weights.

In the following sections the resulting output data will be shown for both the training phase and the evaluation phase. Simulation results are mainly going to be shown for the evaluation phase where the actual learned model can be seen acting on the system. The training of the B-spline networks is performed only using the same reference motion over several iterations, but evaluated using different motions as well as the trained motion. In Section 5.3.1 simulation on the one-axis model is presented, followed by simulation on the virtual robot manipulator in the proceeding Section 5.3.2.

5.3.1 One-axis Model Simulation

Simulation of the static BSN and the extended BSN on the one-axis model are performed using a sinusoid velocity reference signal. The same reference signal that was used in the dynamic LuGre simulation on the One-mass system, namely (4.8) are used in the following simulations.

The training phase for both models can be seen in Figure 5.6, where the normalized mean squared error for the position $\boldsymbol{e_p}$ and for the training signal $\boldsymbol{u_p}$ are presented after each iterations. The weights are all initially set to 0. A total number of 15 iterations are then performed for both models with a learning rate of $\gamma = 0.8$, where the static model can be seen in Figure 5.6a and the dynamic model in Figure 5.6b. Here the learning can be seen to quickly converge and effectively reduce the error in the single simulated motor.

Post training, the friction functions for both models can be seen in Figure 5.7, where the friction function is compared with the actual friction applied to the system, which is obtainable from the simulation model. In Figure 5.7a the static function shown as blue can be seen to accurately approximate the friction function outside of velocities $[-1, 1]$ rad/s, where as inside the interval an average static approximation of the dynamic friction can be seen. In Figure 5.7b the dynamic approximation of the friction can be seen following the friction function. The blue line shows the output of the feed-forward component when velocities with positive acceleration are fed through the component, where as the red curve shows output for negative acceleration.

In the evaluation phase the system output can be seen for the static BSN in Figure 5.8a and for the dynamic in Figure 5.8b. In both of these figures, the same velocity reference signal have been fed through the system and the output of the system is observed. In the top most graph the measured velocity of the motor is shown for the first iteration when no friction compensation is applied and the last iteration when the learning is finished. The middle graph shows the motor position error, also for the first and last iteration. In both of these two graphs the first iteration is marked as black and the last as red. In the bottom graph the output of the feedback controller $\boldsymbol{u_c}$ is compared during no friction compensation to the very last iteration in the learning.

As can be seen from the Figures 5.8a and 5.8b, in both of the cases the feed-forward component manages to make the output motor velocity more smooth, while as the same time reducing the motor position error. The bottom graph in these Figures also gives the indication of that the feed-forward component compensates for friction, due to the feedback control signal at the last iteration is almost zero. In the case of the extended BSN shown in Figure 5.8b, the small oscillating behaviour of $\boldsymbol{u_c}$ in the transition of going from negative to positive velocity can be seen slightly reduced compared to the static BSN shown in Figure 5.8a.

In order to further validate the model, the trained BSN are instead run with a reference signal where the frequency has been increased, namely 3 Hz. This is done in order to simulate the feed-forward component being applied the robot manipulator performs the same movement but with higher TCP velocity. The results can be seen in Figure 5.9, where the layout is the same as before. Here the red signal in the graphs shows output information from the previously trained BSN components acting in the system using the changed reference signal. As can be seen, the components manage to reduce the error, but not to the same degree as when applied during the training motion. This can be seen from the remaining position error in both Figure 5.9a and 5.9b.

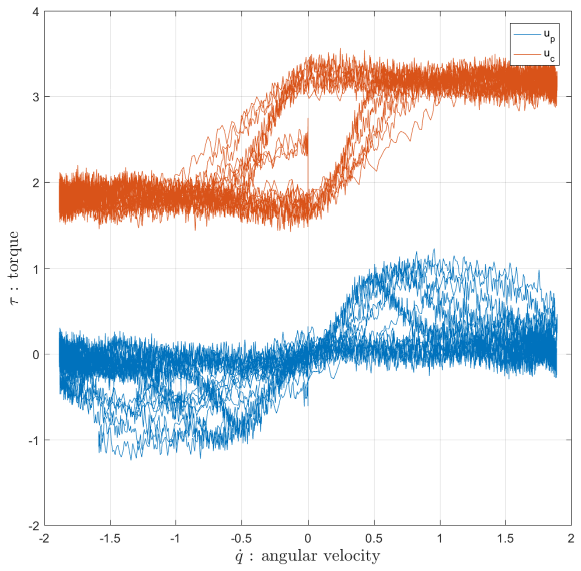
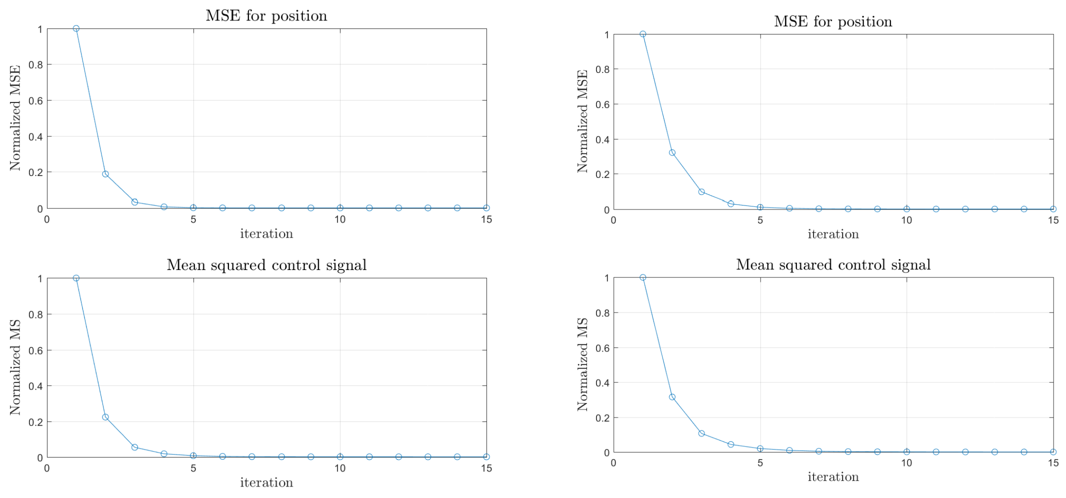


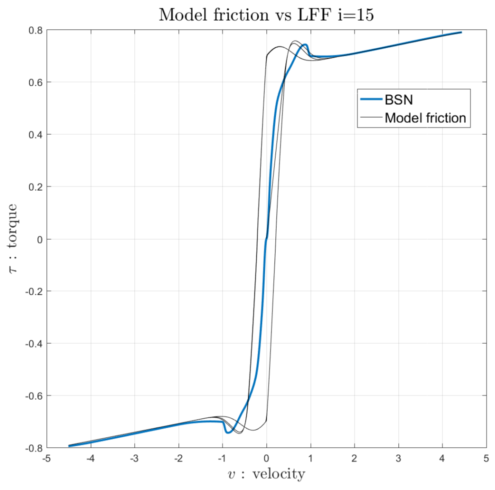
Figure 5.5: The total control feedback output $\boldsymbol{u_c}$ in red for joint 2 compared to a part of the same signal $\boldsymbol{u_p}$ in blue, where the integral part of the $\boldsymbol{u_c}$ signals is not present in $\boldsymbol{u_p}$. The signals are given for TCP motion 10 mm/s around position 1, where the difference in offset between the signals is likely a result of gravity in the system, which gives motivation for $\boldsymbol{z} = \boldsymbol{u_p}$ as the learning signal.

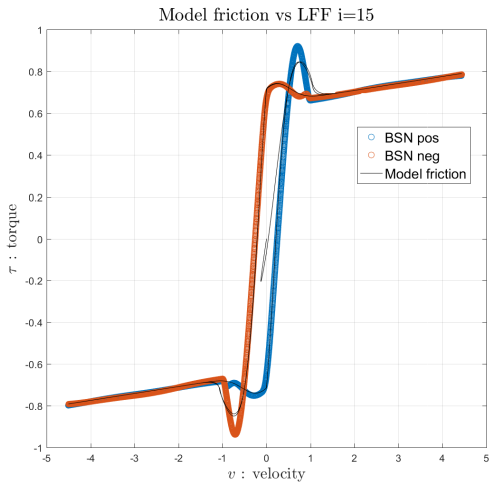


(a) Static BSN

(b) Dynamic BSN

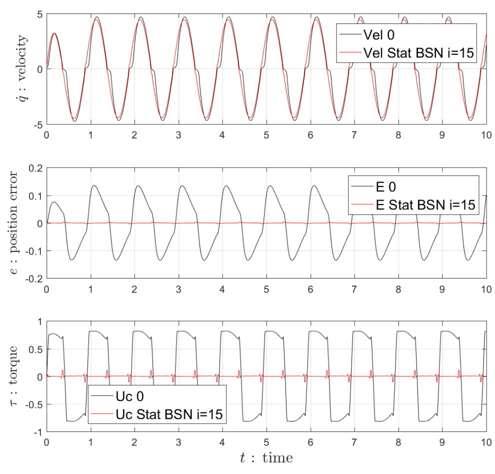
Figure 5.6: Normalized mean square of position error $\boldsymbol{e_p}$ and normalized mean square of the control signal $\boldsymbol{u_p}$ after the application of the learnt friction function from the previous iteration. From iteration 1 where no friction compensation is applied to the last iteration of learning a friction function, the values can be seen to converge to almost zero, indicating that the model reduces the effects of friction.

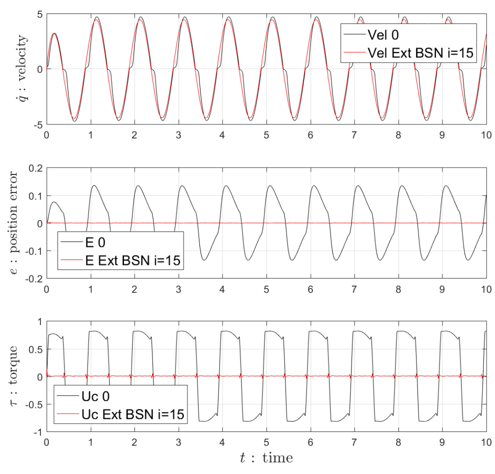




(a) Static BSN. **(b)** Extended BSN.

Figure 5.7: The feed-forward output torque of the trained BSNs as the red and blue lines for the velocity reference signal given in (4.8), compared to the measured system friction as the black lines. The static BSN can be seen making an approximation of the dynamic behaviour, where as the extended BSN are fit to the measured friction. The blue curve in the extended BSN case represents the BSN for positive accelerations and the red represents the BSN for negative accelerations.





(a) Static BSN. **(b)** Extended BSN.

Figure 5.8: One-mass system outpus after being subject to the reference signal (4.8), before and after the BSN have been applied. The top graph shows the output velocity of the system, the middle graph shows the motor position error and the bottom graph shows the output of the feedback controller. There is a significant performance increase seen from the decrease in control signal and motor position error after the application of the feed-forward components. A small oscillating behaviour in the control signal can however be seen for both of these signals when velocity changes direction. This result is similar to the simulation LuGre models shown in Figure 4.3.

The feed-back control signal $\boldsymbol{u_c}$ can also be seen non-zero at several occasions, indicating that the feed-forward components do not completely compensate for the friction effects.

5.3.2 Robot Simulation

For the robot manipulator, the B-spline networks are trained using reference motions with TCP around position 1, as shown in Figure 2.9 with a velocity of 10 mm/s. At the start all weights in the B-spline networks are set to be zero and the learning rate is chosen as $\gamma = 0.95$. Post training, the BSNs are tested on the same motion, as well as on position 3 in the workspace. Results from running the trained BSNs at a higher TCP velocity will also be shown. In the validation phase, TCP position and the root mean square (RMS) of the motor position error will be used to evaluate the performance. In a well performing case the RMS should be small.

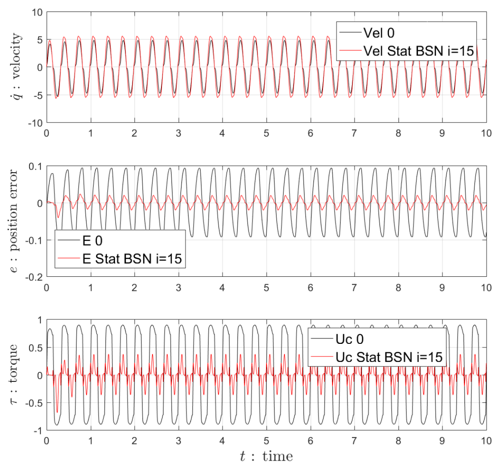
The result of the training phase can be seen in 5.10 where the mean square error for the position ($\boldsymbol{e_p}$) is shown for each joint on the robot manipulator, both for the static BSN and the extended BSN. In the same Figures, the mean square of the training signal $\boldsymbol{u_p}$ is shown such that changes in the control signal of the feedback component can be seen. The training phase can be seen continuing for a total of 10 iterations.

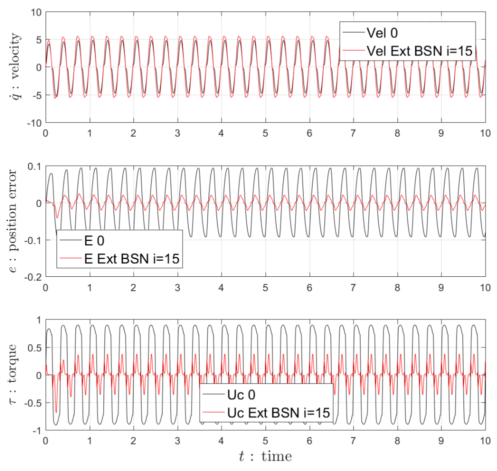
In the learning of the static BSN presented in Figure 5.10a, both the error and the control signal can be seen decreasing and tending towards convergence. The same can be seen for the dynamic approximation shown in Figure 5.10b, however here there is a tendency of the mean square error continuing its decrease. If this is compared with the one-axis model simulation case the decrease is not as significant as can be seen by comparing this to the Figure 5.6.

The resulting friction functions can be seen in Figure 5.11, where the output torque of the feed-forward components can be seen for the reference velocity motion trained on. In the figures the feed-forward components are compared to the dynamic LuGre friction as a frame of reference. In Figure 5.11a the static BSN feed-forward component can be seen almost learning a function which approximates the dynamic LuGre for each joint. In the the extended BSN case, shown in Figure 5.11b, the BSN component can be seen making a dynamic approximation, which is similar to an averaging of the dynamic LuGre model. However for the joints 4,5 and 6, the learned friction from the extended BSN deviates from the dynamic LuGre model, where as the static BSN still makes a reasonable approximation.

The learned friction models are then tested on the same reference signal and around the same position as the training such that the TCP error and motor position error can be evaluated. The result of this can be seen in the Figures presented in 5.12, where the TCP position is show in Figure 5.12a and the RMS of the motor position error can be seen in Figure 5.12b. The TCP plot can be seen to have the same Lead-in and Lead-out behaviour as in the LuGre case in Chapter 4. In the Figures, both the static BSN and the extended BSN are shown in relation to each other, as well as the case when there is no friction compensation is applied. It is clear from the RMS error plot that both feed-forward components manage to improve in regards to the motor position error, where the dynamic BSN performs better on all joints except on joint 4. In the case of the TCP plot, it can be seen that the path deviation that happens during no friction is smaller during the friction compensation.

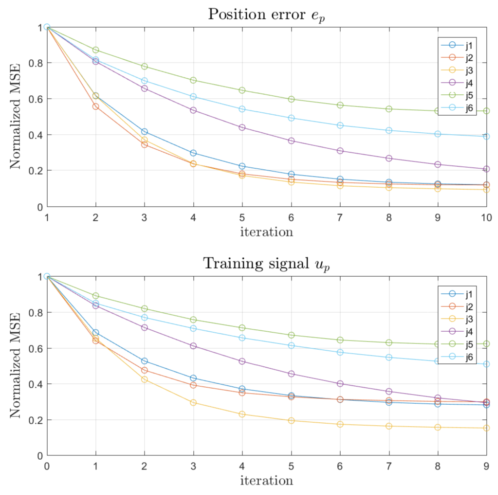
When the model is instead run with a higher velocity, namely 40 mm/s, the result can be seen in Figure 5.13. It is visible from the rms plot in Figure 5.13b that when applying friction compensation the reduction of the RMS is not as significant as before, and the dynamic and static BSN are very similar in performance. From the TCP plot it can be seen that neither model increases the accuracy significantly on the smaller circles. It should be noted however, since the weights are initialized to zero and trained on a low velocity motion, the weights for the BSN at higher velocities will not have been excited. It is therefore reasonable that the performance of running higher TCP velocity motions will not be as good.

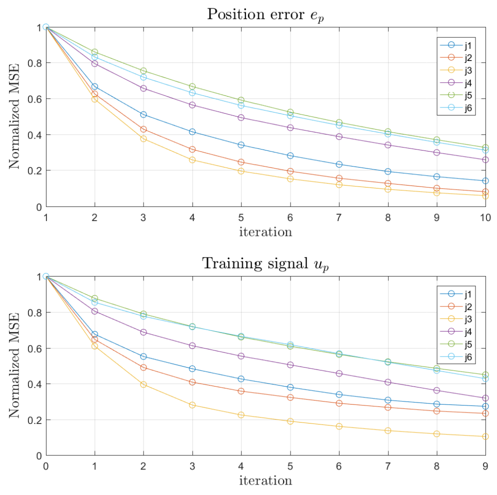




(a) Static BSN. **(b)** Extended BSN.

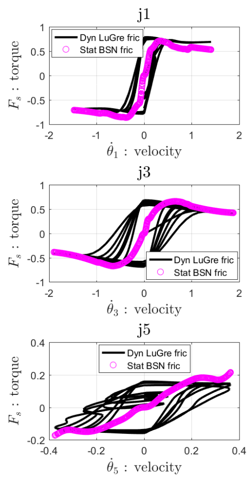
Figure 5.9: One-mass system outputs after being subject to the evaluation reference signal (4.8) with a frequency of 3 Hz, before and after the trained BSN components have been applied. The top graph shows the output velocity of the system, the middle graph shows the motor position error and the bottom graph shows the output of the feedback controller. Here the performance increase is seen through the reduced control signal and motor position error after application of the components, however these are not as close to zero as shown in Figure 5.8.



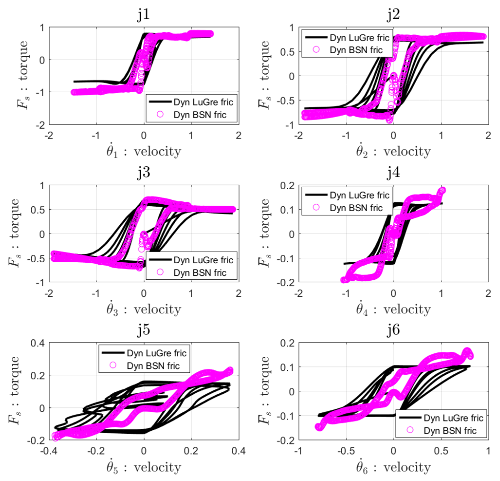


(a) Static BSN. **(b)** Extended BSN.

Figure 5.10: Normalized mean squared position error $\boldsymbol{e_p}$ and training signal $\boldsymbol{u_p}$ for all 6 joints during the training procedure in the simulated environment. Training is performed around TCP position 1 with the TCP velocity 10 mm/s. There is a trend of the static BSN to converge, where as the extended has a tendency of decreasing further.

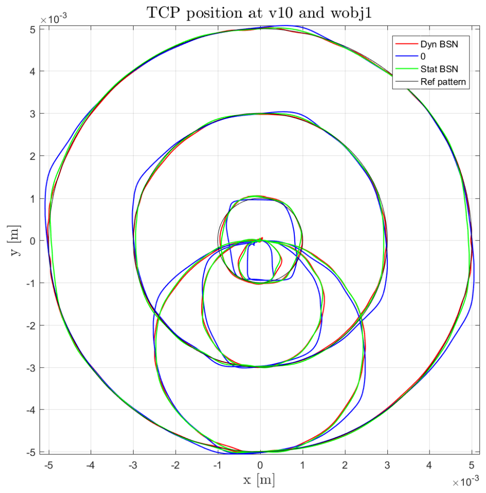


(a) Static BSN.

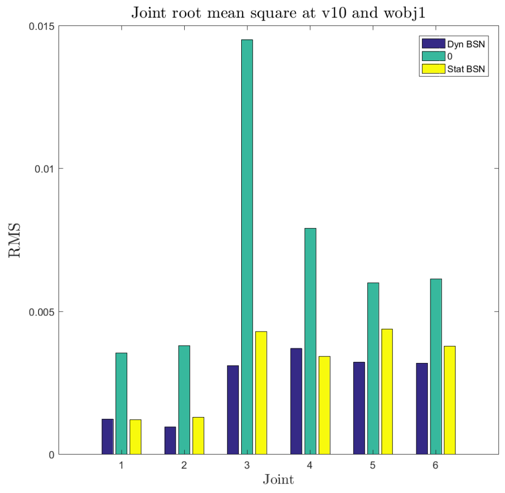


(b) Extended BSN.

Figure 5.11: Feed-forward output torque from BSN components for each joint trained in simulated environment around TCP position 1 with TCP velocity of 10 mm/s. The output is compared to the output of each joints corresponding dynamic LuGre model. The components can be seen learning a friction function similar to the LuGre model, where the extended approximates the hysteresis behaviour of the LuGre friction model.



(a) TCP position



(b) Root mean square of the error.

Figure 5.12: Simulation comparison of TCP motion and RMS of the motor position error when applying static BSN and extended BSN at 10 mm/s TCP velocity on position 1. The legend 0 marks the case where no friction compensation is applied. Both components show increased accuracy of following the black circles, where the deviation from paths are smaller. The RMS can be seen to be lower for the dynamic LuGre model, with exception of joint 4.

When simulation at another location is performed the result can be seen in Figure 5.14. Here the TCP velocity is again 10 mm/s, but the location has changed to the upper left hand corner of the workspace, namely to position 3. The friction compensation is similar to the performance received when evaluation on position 1. The RMS reduction is here more significant, and the path deviations are smaller in comparison to evalutation on the trained motions.

5.4 Robot Experiments

The last part of examining the B-spline networks as feed-forward components includes test on a real robot system. Here learning will be performed in the same way as mentioned in the robot simulation. Furthermore, in order to get a better grip on the learning, test involving learning at a higher velocity will also be included.

The presentation of the test can be found in the following two subsections. In Section 5.4.1 results will be presented where learning have been done with the 10 mm/s TCP velocity reference signal. Following this, Section 5.4.2 will contain results obtained from learning at velocity 40 mm/s. Validation at different locations and velocities will be shown in both Sections. Also in both parts, TCP position criteria such as maximum deviation and root mean square of TCP error are presented.

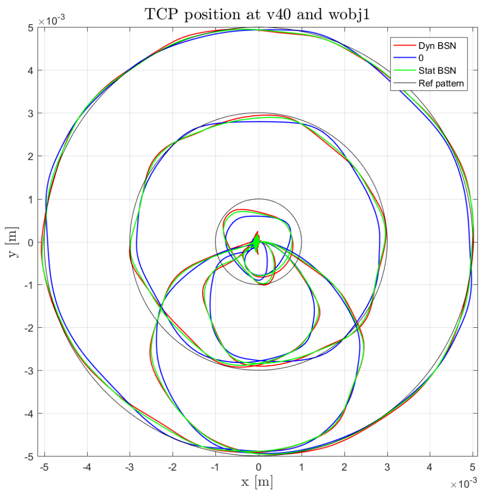
5.4.1 BSN learning at 10 mm/s

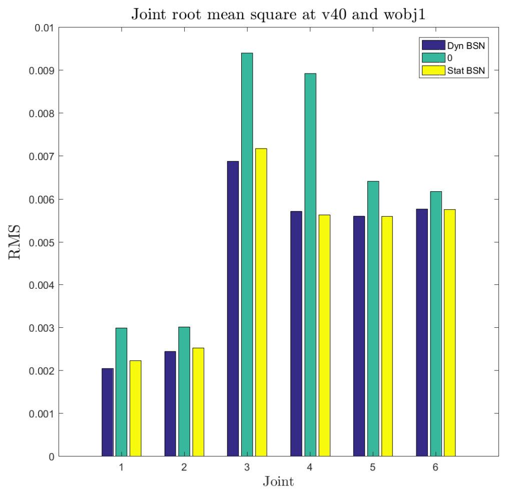
Similar to the robot simulation conducted in Section 5.3.2, learning is first performed using the lower velocity reference signal. The learning can be viewed in Figure 5.15 where both of the B-spline networks are trained for a total of 10 iterations where the learning rate was $\gamma = 0.95$. In the Figures normalized MSE for both the motor position error and the training signal can be seen for each joint on the robot manipulator. The MSE can be seen decreasing with each iteration and indicating that the model is learning. In the case of the static BSN the learning shown in Figure 5.15a the gradient can be seen to decrease as the number of iteration increases, where as in the dynamic case the the decrease is more constant throughout the learning. Furthermore, it can be seen from the graphs that for joints 4, 5 and 6 learning the reduction is not as significant as in joints 1, 2 and 3.

Upon learning termination, the friction functions for both static and extended BSN are shown in Figures 5.16a and 5.16b. In the figures the reference velocity is shown in relation to output torque of the feed-forward components, and are compared to the dynamic LuGre function for the same velocity reference motion. The LuGre models are exactly the same as the ones presented in Chapter 4 and in the robot simulation. In the static BSN friction plot in Figure 5.16a friction curves can be seen approximating the LuGre friction model, where the result are similar to the learned friction functions on the simulated robot, namely in Figure 5.11a. For the learned friction functions for the extended BSN, the friction functions can be seen to have the a hysteresis behaviour as shown in simulation, however the friction functions deviate from the LuGre models. Similar to simulation, joints 4, 5 and 6 show very different behaviours compared to the LuGre curves.

After running the models on the same reference signal as used in training, the resulting TCP position and RMS of the motor position error can be seen in Figure 5.17 for both BSN models. As in simulation both the TCP position and the RMS of motor error is improved when BSN friction compensation is applied to the system. Is not clear however which of the models perform better in this case, especially from the TCP graph. On the RMS graph however, the extended BSN seem to perform better on joint 1, 2 and 3, but on the rest of the joints, the static BSN can be seen to perform better. The same observations can be made from the evaluation at the other three locations but not shown here due to space.

More specific data on the performance of the BSN components can be seen in table 5.1. In tables the RMS for TCP error during circular motion is shown together with the largest deviation from

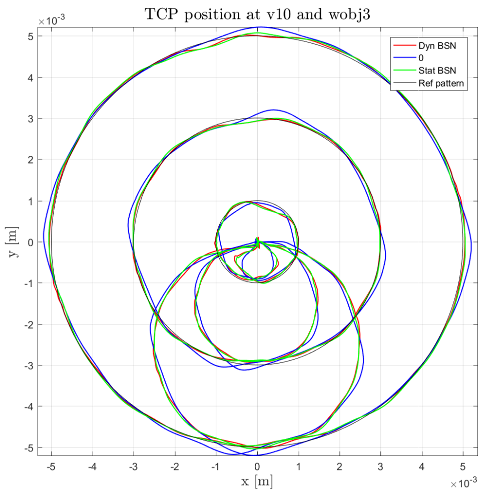


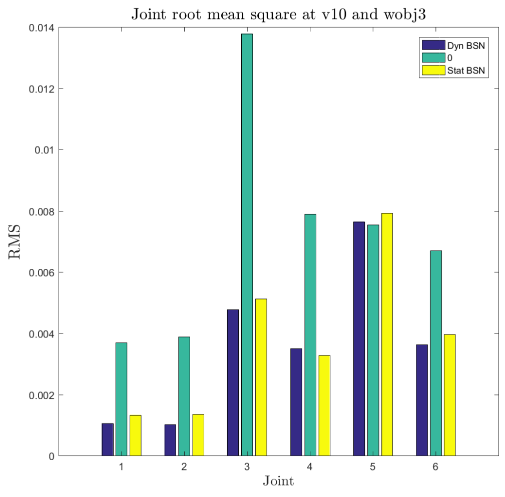


(a) TCP position

(b) Root mean square of the error.

Figure 5.13: Simulation comparison of TCP motion and RMS of the motor position error when applying static BSN and extended BSN at 40 *mm/s* TCP velocity on position 1. The results are compared to the same simulation without applied friction compensation. The performance gain is not as significant as during the velocity it was trained upon. The RMS plot show this through the small reduction of motor position error through the use of the applied components.

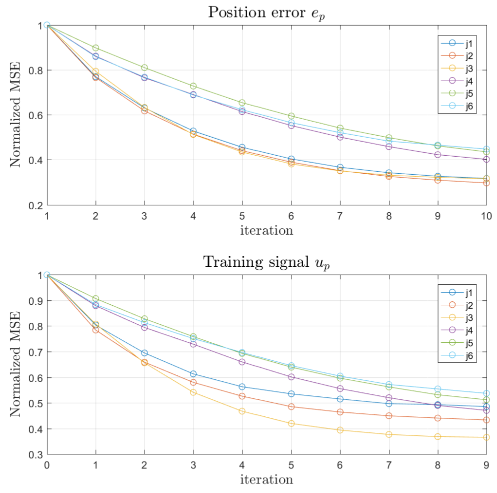




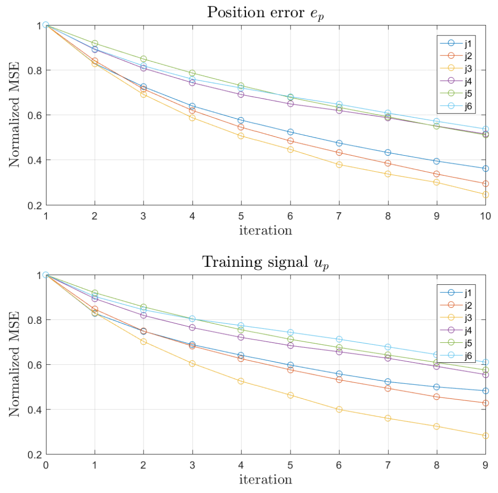
(a) TCP position

(b) Root mean square of the error.

Figure 5.14: Simulation comparison of TCP motion and RMS of the motor position error when applying static BSN and extended BSN at 10 *mm/s* TCP velocity on position 3. The results are similar to the simulation on position 1, which indicates that the components perform the same similarly on different locations.

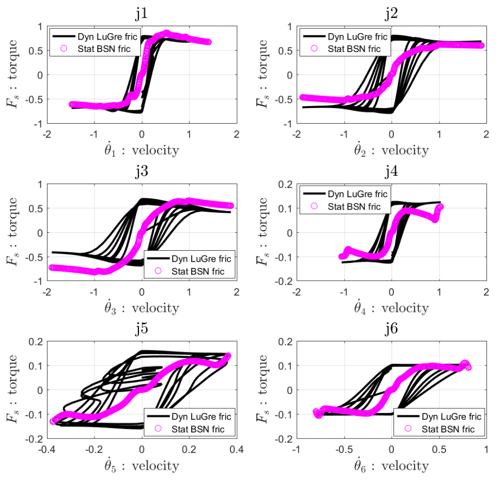


(a) Static BSN.

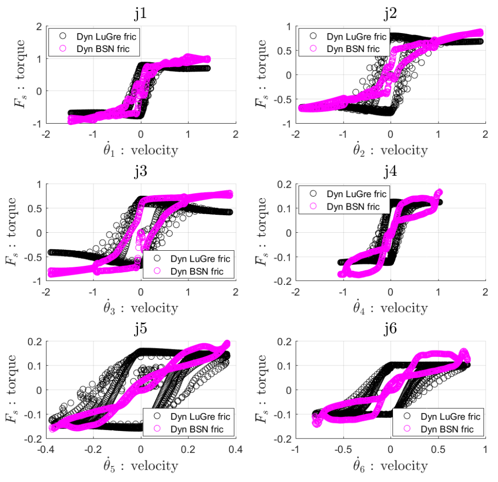


(b) Extended BSN.

Figure 5.15: Normalized mean squared position error and training signal $\boldsymbol{u_p}$ for all 6 joints during the training procedure on the real robot. Training is performed around TCP position 1 with the TCP velocity 10 mm/s. For the static BSN the learning tends towards converging, where as the extended has a trend performing further learning.



(a) Static BSN.



(b) Extended BSN.

Figure 5.16: Feed-forward output torque from BSN components trained on the real robot around TCP position 1 with TCP velocity of 10 mm/s. The result is shown for each joint shown for the same reference motion as trained upon. The output is compared to the output of each joints corresponding dynamic LuGre model. The learnt friction models display slightly different behaviours compared the simulation in that for joint 3, for example has different behaviour for larger velocities.

the circles. This is shown for each individual circle when the TCP velocity is 10 mm/s, as well as for all the positions in the workspace. The motions are also compared to the case without any friction compensation. It can be seen from the tables that the improvement from including friction compensation using BSN is large, given both cases. An observation from the tables is also that the extended model performs better in almost every single location, compared to the static model.

When the learned B-spline networks are used on a higher velocity the same phenomenon viewed in simulation can be seen here, that the RMS reduction of the motor position error in the case of friction compensation is not as significant as performed on the trained velocity reference. As described in the simulation scenario, it is unlikely that large performance will be seen, since the networks have not been trained for the given velocities.

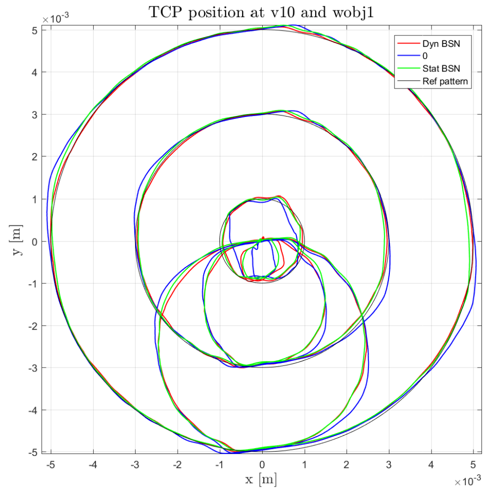
5.4.2 BSN learning at 40 mm/s

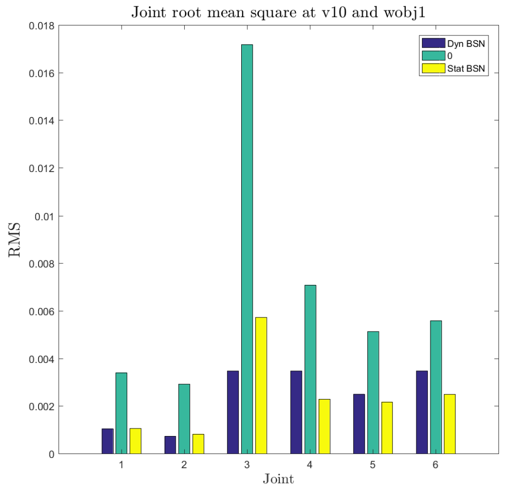
For learning performed at higher velocity, namely when the TCP velocity is 40 mm/s, the resulting learning can be seen for both BSN feed-forward components in Figure 5.18. The learning rate is $\gamma = 0.95$ and the number of iterations are 10, which is the same as used on learning at TCP velocity 10 mm/s. The learning for all joints can be seen to be more similar to one another, which was not the case for the training at lower TCP velocity. For both components, the gradient of the learning can be seen to be reduced for the higher iterations and almost converging to a set value.

The corresponding friction functions can be seen in Figure 5.19. The friction functions can be seen covering a larger interval of velocities in comparison with learning at lower velocitys. In the dynamic case, namely in Figure 5.19b the friction function can be seen to match the LuGre dynamic model on almost all joints, where as on learning at TCP velocity 10 mm/s, this was not the case for joints 4, 5 and 6.

The first validation is made on the position 1, where the TCP and RMS graphs are shown in Figure 5.20. From the TCP Figure 5.20a it is clear that the friction compensation does not manage to improve the TCP position for the smallest circle. On the other circles, the TCP position can be seen to be following the reference circle slightly more compared to before. In the RMS plots however, a large reduction in the RMS can be seen for both BSN components on all joints. The extended BSN can be seen slightly better on a majority of the joints. This was also seen on the three other positions but is not shown here due to space.

By comparing the maximum TCP deviation and the RMS of the TCP error for the presented motion at the different positions in Table 5.2, it can be seen that friction compensation is in general better than without. Only at a few scenarios for the smallest circles where the TCP path accuracy has been visually observed to be bad, the friction compensation is worse. However the performance gain in terms of these quantities are not as large as in the lower velocity cases, presented in Table 5.1. It can also be seen that the extended BSN performs better in most scenarios, if only slightly.

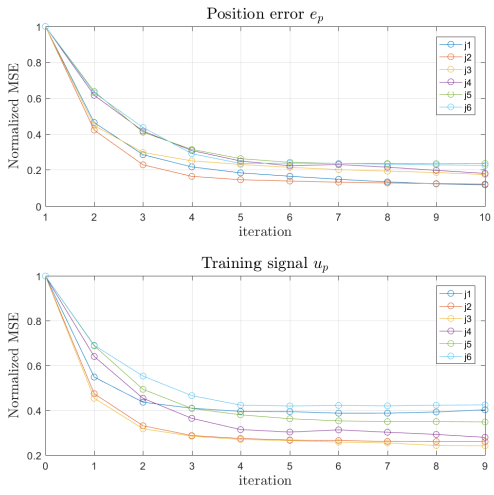


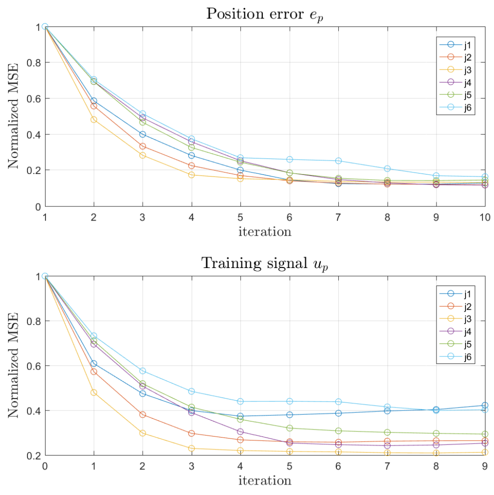


(a) TCP position

(b) Root mean square of the error.

Figure 5.17: Real robot comparison of TCP motion and RMS of the motor position error when applying static BSN and extended BSN at 10 mm/s TCP velocity on position 1. The results are compared to the same simulation without applied friction compensation. Increased performance can be seen for both models for the trained velocity motion. The extended BSN performs better on the three first joints, however not on the last ones.

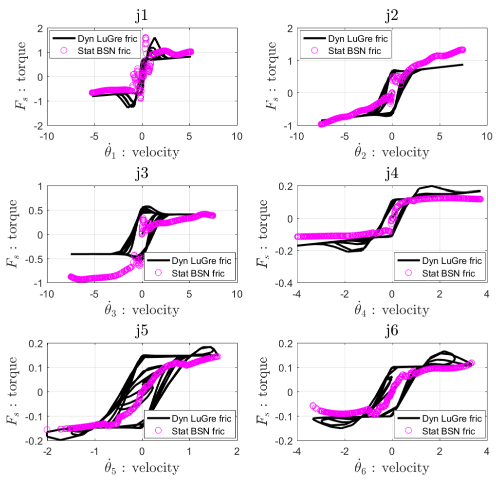




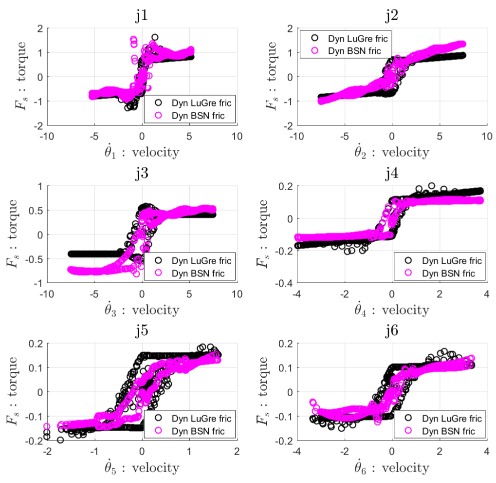
(a) Static BSN.

(b) Extended BSN.

Figure 5.18: Normalized mean squared position error and training signal $\boldsymbol{u_p}$ for all 6 joints during the training procedure on the real robot. Training is performed around TCP position 1 with the TCP velocity 40 mm/s. For both models on all joints the learning seem to converge.

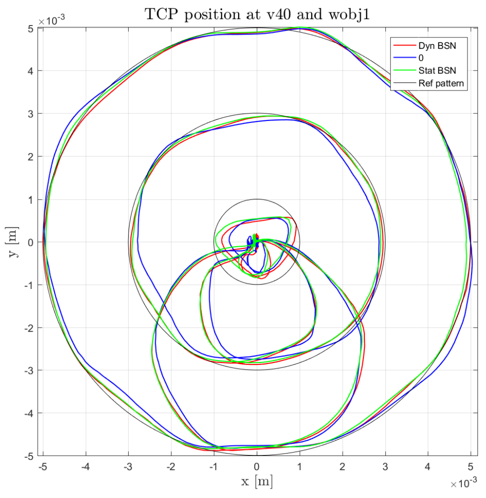


(a) Static BSN.

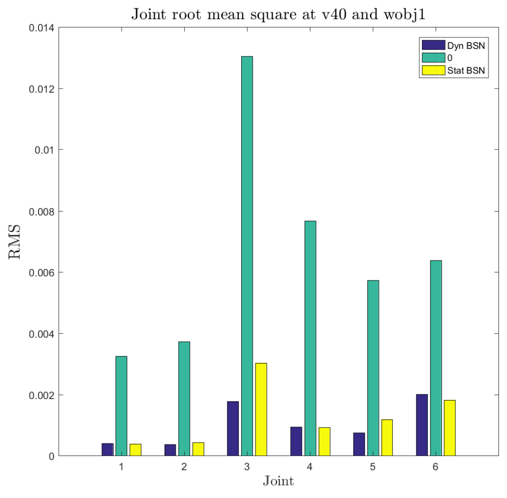


(b) Extended BSN.

Figure 5.19: Feed-forward output torque from BSN components trained on the real robot around TCP position 1 with TCP velocity of 40 mm/s. The result is shown for each joint shown for the same reference motion as trained upon. The output is compared to the output of each joints corresponding dynamic LuGre model.



(a) TCP position



(b) Root mean square of the error.

Figure 5.20: Real robot comparison of TCP motion and RMS of the motor position error when applying static BSN and extended BSN at 40 mm/s TCP velocity on position 1. Learning has been done at 40 mm/s. It is clear from the graphs that both models increase to performance, since both the roundness is increased as well as the RMS begin decreased significantly.

Table 5.1: Maximum deviation from the reference TCP path and root mean square of the path error in *mm* for each individual circle when applying static and extended BSN in feed-forward control. Results are shown in comparison with no friction compensation, when the reference motion corresponds to 10 *mm/s* TCP velocity. The BSNs are trained and evaluated for the same reference motion. Both components reduce the maximum deviation and RMS of path error significantly, where the extended has in general the best performance.

(a) Max deviation without friction compensation.

	Pos. 1	Pos. 2	Pos. 3	Pos. 4
Circ. r1	0.31	0.24	0.23	0.31
Circ. r3	0.18	0.26	0.3	0.22
Circ. r5	0.23	0.26	0.27	0.27

(b) RMS without friction compensation.

	Pos. 1	Pos. 2	Pos. 3	Pos. 4
Circ. r1	0.14	0.109	0.122	0.138
Circ. r3	0.074	0.095	0.109	0.098
Circ. r5	0.078	0.091	0.106	0.105

(c) Maximum deviation with static BSN

	Pos. 1	Pos. 2	Pos. 3	Pos. 4
Circ. r1	0.13	0.16	0.17	0.15
Circ. r3	0.12	0.22	0.17	0.16
Circ. r5	0.12	0.21	0.13	0.15

(d) RMS with static BSN

	Pos. 1	Pos. 2	Pos. 3	Pos. 4
Circ. r1	0.066	0.075	0.09	0.078
Circ. r3	0.054	0.081	0.075	0.068
Circ. r5	0.049	0.082	0.061	0.075

(e) Maximum deviation with extended BSN

	Pos. 1	Pos. 2	Pos. 3	Pos. 4
Circ. r1	0.13	0.14	0.17	0.15
Circ. r3	0.09	0.18	0.14	0.14
Circ. r5	0.1	0.19	0.13	0.15

(f) RMS with extended BSN

	Pos. 1	Pos. 2	Pos. 3	Pos. 4
Circ. r1	0.057	0.064	0.076	0.084
Circ. r3	0.045	0.076	0.056	0.072
Circ. r5	0.042	0.072	0.054	0.07

Table 5.2: Maximum deviation from the reference TCP path and root mean square of the path error in *mm* for each individual circle when applying static and extended BSN in feed-forward control. Results are shown in comparison with no friction compensation, when the reference motion corresponds to 40 *mm/s* TCP velocity. The BSNs are trained and evaluated for the same reference motion. Both models show less maximum deviation and RMS for path error in most circles, compared to no friction compensation. It can be seen that the change for the smaller circle is not as significant as for the larger circles.

(a) Max deviation of TCP without feed-forward.

	Pos. 1	Pos. 2	Pos. 3	Pos. 4
Circ. r1	0.5	0.63	0.91	0.58
Circ. r3	0.41	0.35	0.38	0.46
Circ. r5	0.31	0.28	0.35	0.36

(b) RMS of TCP without feed-forward.

	Pos. 1	Pos. 2	Pos. 3	Pos. 4
Circ. r1	0.362	0.324	0.41	0.343
Circ. r3	0.187	0.188	0.206	0.209
Circ. r5	0.12	0.141	0.14	0.147

(c) Max deviation of TCP with static BSN.

	Pos. 1	Pos. 2	Pos. 3	Pos. 4
Circ. r1	0.47	0.61	0.86	0.53
Circ. r3	0.17	0.3	0.4	0.2
Circ. r5	0.17	0.21	0.24	0.19

(d) RMS of TCP with static BSN.

	Pos. 1	Pos. 2	Pos. 3	Pos. 4
Circ. r1	0.263	0.264	0.377	0.25
Circ. r3	0.099	0.131	0.145	0.112
Circ. r5	0.085	0.106	0.088	0.093

(e) Max deviation of TCP with extended BSN.

	Pos. 1	Pos. 2	Pos. 3	Pos. 4
Circ. r1	0.52	0.6	0.86	0.57
Circ. r3	0.19	0.28	0.37	0.21
Circ. r5	0.16	0.25	0.21	0.18

(f) RMS of TCP with extended BSN.

	Pos. 1	Pos. 2	Pos. 3	Pos. 4
Circ. r1	0.273	0.269	0.383	0.257
Circ. r3	0.092	0.138	0.148	0.119
Circ. r5	0.075	0.113	0.082	0.087

5.5 Evaluation of Performance

Taking the simulation and the application of the learned feed-forward components in to consideration, it can be observed that the learning feed-forward control using B-spline networks for friction compensation improves the TCP position and the motor position error. Improvement on the motor position as can already be seen on the One-mass system, and is later observed on the simulation and the application of the components on robot manipulator. From the graphs showing RMS of motor position error in both simulation (figure 5.12) and in application on real robot (figure 5.17) the B-spline networks both show improvement. Observing the TCP graphs in the same Figures, the motion seen when the BSNs are applied has a smaller path error. The same phenomenon was observed when the trained BSN models are run on different locations in the workspace but with the same TCP velocity.

From the learned friction function in simulation, namely in Figure 5.11 it can be seen that for many joints, the resulting function is similar to the LuGre model. Seeing as the robot in simulation is using the dynamic LuGre model as the actual model for friction, a function similar to this should theoretically compensate for the friction in the system. Since the friction functions are similar to the LuGre model and that performance is increased, the BSN components can be seen compensating for the friction. The transition to the real domain when applied to robot, improvement is still visible despite not knowing exactly how friction behaves. It can therefore be stated that the BSN manages to capture some undesired parts of the system related to the velocity of the joints.

Considering the difference in performance between the statical BSN and the extended BSN however, it is not clear from simulation and application on real robot which model performs better in regard to the TCP and RMS of motor error. In the One mass model, the dynamic BSN can be seen performing slightly better in regard to the position error. On simulation and application on the real system, this is not obvious as the TCP position is similar in both cases, while RMS for motor position error seems to vary depending on the location. Through observing the maximum TCP deviation and RMS of the TCP error in the Table 5.1, the extended BSN performs better in almost all instances, in comparison with the statical BSN. The same was observed for higher velocities in 5.2, however not as frequently. Since the extended BSN more often shows larger improvement than the static, the dynamic model performs better in general.

A drawback with the tested B-spline networks is that they not perform as well on TCP velocities which the BSNs have not been trained upon, which was seen in simulation and also stated in the robot experiments. This is believed to be the result of two effects, namely that the B-spline network provide zero friction compensation within a certain velocity interval, and that the shape of the hysteresis behaviour in the control signal changes when TCP velocity increases.

When the BSNs are trained on low velocity and then tested on a higher velocity, results indicate that the performance becomes worse. This can be seen as partly caused through the fact that the weights in BSN are all initialized to zero before learning starts. This causes B-spline which are defined on velocities outside of the training motion to never become activated in training, which in turn will yield zero output for those velocities. Since the maximum velocity in each motor increases as TCP velocity increases, it is natural that the BSN to do not perform well in these situations. This problem however could be solved by making sure that the weights are non-zero and correspond to some initial friction function. This could for example be a static and constant friction function.

Considering the case when the BSN are trained on higher velocities and tested on lower. From simulation it is known that the hysteresis behaviour of the LuGre model changes depending on the TCP velocity. A learned B-spline network, even in the extended case will be less likely to approximate the behaviour on the new motion due to it still having two static components. As mentioned in the design of the BSN components in Section 5.1, capturing a complete dynamic behaviour is impossible, which means that in these cases the results will always be inaccurate.

However, in regards to the stated objective with friction compensation given in Section 1.2, the approach using LFFC with BSN can be seen as successful in that for a trained motion at a certain velocity, the feed-forward components show improvement on multiple location, given that the TCP velocity is the same. Therefore this solution can be seen as a semi-general solution, since it works well for a given velocity motion at different positions.

6

Extended Friction Identification

The LuGre friction model which was evaluated in terms of performance in Chapter 4 was given prior to the works of this thesis. As such the accuracy of the model in terms of validation is therefore unknown and remains a potential source of error. From the experiments performed the LuGre models in a feed-forward component showed promising result, both from simulation and from test on the real robot. A key result was that for the dynamical case, the performance was improved beyond the points of the static model. The dynamic properties somehow manages to capture parts of the friction behaviour during the pre-sliding regime as explained in 2.1.4, which the static model is not designed to capture. The same was observed for the dynamic approximation of friction used by the extended BSN presented in Chapter 5. This gives motivation to the fact that an accurate dynamical model could improve the performance further.

Furthermore, the LuGre model is in general not a perfect model of the friction. As also described in Section 2.1.4 the LuGre model aims to model the most relevant aspects of friction, inherently could produce potential errors. Therefore an interesting approach to investigate is to use nonlinear black-box identification methods to capture more of the friction behaviour as compared to the LuGre model.

This chapter therefore aims to present the re-identification of the LuGre model as well as a nonlinear identification method through the use of nonlinear ARX model (NARX). In Section 6.1 the identification of LuGre is presented for one axis on the robot manipulator, and in Section 6.2 the same is performed for the NARX model, however here instead of utilizing actual data of the robot output from a LuGre model is used instead.

6.1 LuGre Friction Identification

Accurately describing the joint friction within the system is the main motivation to perform identification with a non-linear dynamic model. Since the dynamic LuGre model in feed-forward control has shown improving results on the robot manipulator system, it is interesting to redo and validate the parameter estimation of the model, such that further improvement can be seen.

The method of performing identification of the LuGre model presented in this thesis is based on the method presented in [20]. In the article the authors present a way of performing identification of the dynamic LuGre model by first estimating the parameters of the static LuGre model, which is then used in the estimation of the dynamic model. Special experiments are conducted in each of the scenarios in order to capture important physical effects of the system. The identification procedure presented in this thesis differs from [20] in that a slightly different static model is used as well as utilizing prior knowledge of the robot system models for compensation of certain effects, instead of performing some of the suggested experimental tests.

In the following sections the procedure of identifying the LuGre model will be presented for one axis on the robot manipulator, namely joint 1. First the estimation of the parameters of the static LuGre model will presented in Section 6.1.1, which is then used in order to perform the identification of the last two parameters in the dynamic model, which is presented in Section 6.1.2. The last section presents the process of validating the LuGre model and includes a comparison with the given parameters, used in Chapter 4.

6.1.1 Static Friction Identification

To enable the identification of any friction model, data of the friction must be gathered. Considering the system limitations, data is acquired on the motor side of the robot, through position sensors and the calculated torque signal sent to the drive system. The feedback controller generates a torque which compensates for model uncertainties including the friction effects. The system input can thus be seen as containing various effects of different physical properties and among these the friction.

Recall the system description through the inverse rigid body dynamics model presented in (2.11), the friction $\boldsymbol{\tau_f}(\dot{\boldsymbol{q}})$ is seen as an additive vector component dependent on the angular velocity of the motors. The torque input to the system can be seen as the variable τ in the model. A desired scenario in order to generate friction data is therefore

$$\boldsymbol{\tau_f}(\dot{\boldsymbol{q}}) = \boldsymbol{\tau} \tag{6.1}$$

which means that the components $\boldsymbol{M}(\boldsymbol{q})$, $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ or $\boldsymbol{g}(\boldsymbol{q})$ must either be known or compensated for through a specific experiment. As mentioned in [20], if data for example is gathered during the motion of constant velocities and that only one joint is moved at any given time, the inertia term $\boldsymbol{M}(\boldsymbol{q})$ and the Coriolis term $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ can be viewed as negligible, and thus only leaving the gravity component as the only remaining effect.

Since the first joint is perpendicular to the ground, it is not subject to gravity which removes the gravity component from the equation. However this is not the case for the remaining joints in the system. To solve this issue, [21] proposes a method where the constant joint motion is performed in one direction and is then repeated for the the opposite direction, creating two signals for a single joint subject to gravity, namely

$$\begin{aligned} \tau_f(\dot{q}) + g(q) &= \tau^+ \\ \tau_f(-\dot{q}) + g(q) &= \tau^- \end{aligned} \tag{6.2}$$

Under the assumption that $\tau_f(\dot{q}) = \tau_f(-\dot{q})$ the subtraction of the signals can give the value of friction according to

$$\tau_f(\dot{q}) = \frac{\tau^+ - \tau^-}{2} \tag{6.3}$$

which enables the generation of friction data. The gravity component can of course be calculated through information about mass distribution in the links and simply removed from the input torque signal.

The generation of friction data for the static LuGre model is therefore performed by applying constant velocity reference signals to the joint during a period of time long enough to make sure that the acceleration have converged to zero. After this, the same motion is performed in the opposite direction, back to the starting position. Position is measured during this time together with the control input for the joint. The position is numerically differentiated in order to get the measured velocities. The constant reference velocities are chosen in the interval $[0.01, 14] rad/s$. To model the Stribeck phenomenon a distribution of velocities is chosen such that more data points are achieved close to 0 rad/s. Since the motions of the robot considered in this thesis are performed in relatively low angular velocities, data is not sampled for above $14 rad/s$. Denote the sampled values as $X = [x_1, x_2, ..., x_N]$ for the velocities derived from position and $Y = [y_1, y_2, ..., y_N]$ for the sampled friction data from the input torque where N is the number of sampled points.

The function to fit to the generated data to is given in equation (2.10) and represents the static LuGre friction. To get rid of the discontinuity it is possible to use a trick. The data generated from the experiments can be seen as values only generated at positive velocities, for which the static LuGre equation can be written as

$$F_s(v) = g(v) + f_v v = f_c + (f_s - f_c)e^{-(v/v_s)^2} + f_v v \tag{6.4}$$

under the assumption that the static LuGre model is symmetric. The parameters to estimate are f_c, f_s and f_v. The parameter v_s in the model is the Stribeck velocity and represents how quickly the function $g(v)$ approaches the Coulomb friction f_c. This parameter is chosen through observation of the data points, where as the rest of the components are considered parameters to estimate, namely

$$\theta = [f_c \; f_s \; f_v]^T \tag{6.5}$$

The function is given in terms of its parameters θ and regressors φ as

$$F_s(\varphi, \theta) = \varphi^T \theta = \begin{bmatrix} 1 - e^{-(v(k)/v_s)^2} & e^{-(v(k)/v_s)^2} & v(k) \end{bmatrix} \begin{bmatrix} f_c \\ f_s \\ f_v \end{bmatrix} \tag{6.6}$$

The estimation problem can then be formulated as an optimization according to

$$\begin{aligned} \hat{\theta} &= \arg\min_{\theta} V(\theta) \\ &s.\ t. \\ &\theta \geq \mathbf{0} \end{aligned} \tag{6.7}$$

where $\hat{\theta}$ is the value that minimizes the cost function $V(\theta)$. For parameter estimation of the static LuGre model the least square estimation method is considered, where the optimization goal is to minimize the sum of squared residuals. As such the cost function is given as

$$V(\theta) = \sum_{i=1}^{N} (y_i - F_s(\varphi, \theta))^2 \tag{6.8}$$

where y_i is the measured torque control signal.

The optimization problem stated in (6.7) given the cost function in (6.8) is then solved through the use of the least square method.

The result can be seen in Figure 6.1 where the red line illustrates the estimated function and the blue circles mark the sampled data points. As can be seen from the Figure, the approximating function matches the data points well. The parameters are

$$\theta_s = \begin{bmatrix} f_c \\ f_s \\ f_v \\ v_s \end{bmatrix} = \begin{bmatrix} 0.5367 \\ 0.5634 \\ 0.0437 \\ 0.15 \end{bmatrix} \tag{6.9}$$

The complete static LuGre model can now be expressed, including the sign(v) factor which was previously removed in (6.4).

6.1.2 Dynamic Friction Identification

Based on the parameters derived from the identification of the static LuGre model, the stiffness parameter and the dampening parameter σ_0 and σ_1 must be estimated in the extended, dynamic model. Given that the function to estimate is no longer static, other means of experiments must be conducted in order to capture the dynamic behaviour of friction within the system. In the mentioned article [20], the authors propose to use a low frequency torque signal and measure the resulting velocity.

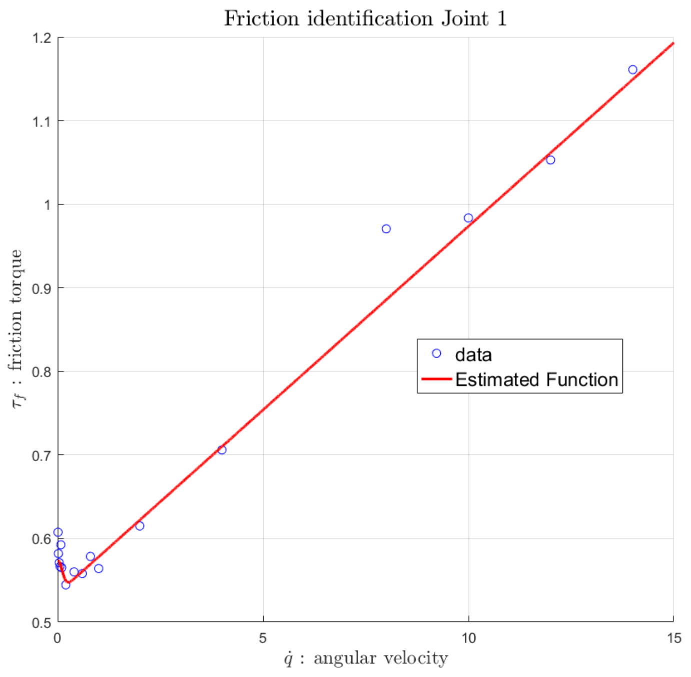


Figure 6.1: Identified static LuGre friction curve based on estimated friction data sampled on the first joint on the robot manipulator. The function is considered a good fit for the data points.

The approach from [20] is adopted here and a sinusoidal reference trajectory is generated for a single joint on the robot manipulator and signals such as position and input torque are recorded. The velocity can then be derived by numerical differentiation of the position. This enables the possibility of creating a velocity/torque mapping which has reduced influence by other physical properties and can thus be used in order to estimate the two remaining parameters. This choice of reference signal relates to the dynamics of the robot. Consider the simplified flexible dynamic model of one joint in motion, given in Figure 2.2. The equations for this model together with a friction term can be described as

$$\begin{aligned} J_m \ddot{q}_m &= u + K(q_a - q_m) + d(\dot{q}_a - \dot{q}_m) - f(\dot{q}_m) \\ J_a \ddot{q}_a &= \tau_g + K(q_m - q_a) + d(\dot{q}_m - \dot{q}_a) \end{aligned} \tag{6.10}$$

where u is the input torque, J_m represents the motor inertia and J_a the arm inertia. τ_g can here be seen as the gravity torque. The system here can also be seen containing a stiffness constant K and a dampening constant d. Due to the rigid structure of the robot arm, if a low enough velocity reference signal is applied, effects from the flexibilities will not be excited during the robot motion. This can be seen as difference between the arm angle and angular velocities accurately following the motor angle. Thus the assumption of $\ddot{q}_a = \ddot{q}_m = \ddot{q}$ can be applied, such that the system can be seen to behave according to

$$J\ddot{q} = u + \tau_g - f(\dot{q}) \tag{6.11}$$

where $J = J_a + J_m$. The input signal can therefore be seen containing the gravity τ_g, the moment of inertia $J\ddot{q}$ and the friction.

Due to the reference motion, the angular velocity and angular acceleration is always changing, which means that the torque signal will contain contribution from the moment of inertia. The gravity is also an issue, however for the first joint of the robot manipulator which is perpendicular to the ground, no gravity is acting on the joint. In this case, system models of the inertia in terms an exact mass matrix $\boldsymbol{M}(q)$ and gravity vector $\boldsymbol{\tau}(\boldsymbol{q})$ for the robot manipulator used. The inertia

Table 6.1: Representation of the gathered signals used for parameter estimation and validation in the case of identifying the dynamic LuGre model. The column named Data record is simply an identity for the reference signal, where the other columns contain the parameters to the reference generation function shown in (6.13)

Data record	Amplitude: A	Frequency: f (Hz)
1	1.3	0.25
2	1.3	0.5
3	1.3	1.0
4	1.3	2.0
5	1.3	5.0
6	5	0.25
7	5	0.5
8	5	1.0
9	5	2.0
10	5	5.0
11	5	0.25
12	10	0.5
13	10	1.0
14	10	2.0
15	10	5.0

value is assumed given, with an estimated error of up to 5%. The friction value can be estimated according to

$$\hat{f}_1(\dot{q}_1) = u_1 - J_1\ddot{q}_1 + \tau_{g1}(\boldsymbol{q}) = u_1 - J_1\ddot{q}_1 + 0 \tag{6.12}$$

where the gravity is removed for the first joint, while J_1 is given and $\ddot{q}_1$ is numerically differentiated twice from the measured angular position.

The reference signal designed for performing the tests is generated through the following function

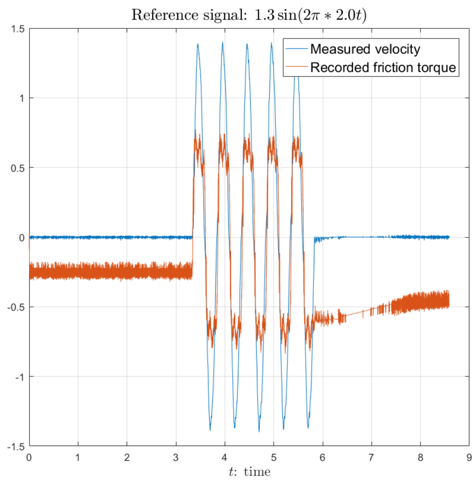
$$\dot{q}_{ref} = A\sin(2\pi f t) \tag{6.13}$$

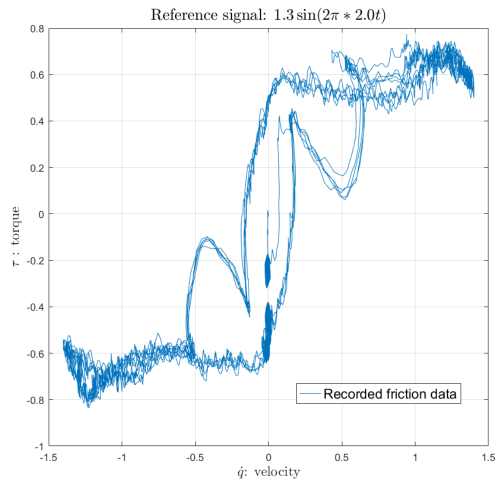
where the parameters A defines the maximum velocity and the frequency f defines the acceleration. Since the output of the LuGre model changes in accordance with acceleration and velocity, reference signals for a number of combinations of these are generated. The combination of values together with a data measurement series number, here denoted Data Record, which is given in Table 6.1. The data record is defined in order to simply refer to a measurement series where a specific reference signal is used.

An example of the data generated by applying the reference signal to the system can be seen in Figure 6.2, where the differentiated velocity from position is shown together with the estimated friction according to (6.12). The Figure 6.2a shows the two signals in combination with time, where as Figure 6.2b shows the two signals against each other. The signals shown in the Figure has been filtered through a second order butterworth filter with a cut-off frequency of 40 Hz. The reference signal used here corresponds to $\dot{q}_{ref} = 1.3\sin(2\pi * 1.0 * t)$.

In order to perform parameter estimation, a number of recorded signal from Table 6.1 are chosen to be used as estimation data. In the case of the performed identification, the set of data records $D = \{2, 4, 7, 9, 12, 14\}$ are used, where as the rest of the data records, namely $V = \{1, 3, 5, 6, 8, 10, 11, 13, 15\}$ are used for validation data. The mentioned records in set D are concatenated in terms of samples and used together in the estimation procedure. For future reference, denote the estimation velocity data $\boldsymbol{v} = [v_1, v_2, ..., v_N]$ and the friction data $\hat{\boldsymbol{f}} = [\hat{f}_1, \hat{f}_2, ..., \hat{f}_N]$ where N is the number of total samples in the set D.

The dynamic LuGre model to be identified is given in its continuous form given in (4.3). Given that the data is sampled, a discretized version of the model is required. The discretized model applied in the implementation of the dynamic LuGre on the robot in Section 4.2 is also used here.





(a) Recorded signals as functions of time. **(b)** Torque on y-axis, velocity on x-axis.

Figure 6.2: Graph shows measured velocity and input torque for the reference signal $1.3\sin(2\pi * 2.0t$. Here the estimated dynamic behaviour of friction can be seen.

The LuGre model for the identification is given as

$$\begin{aligned} z(k+1) =& z(k) + hv(k) - h\sigma_0 \frac{|v(k)|}{g(v(k))} z(k) \\ \dot{z}(k) =& v(k) - \sigma_0 \frac{|v(k)|}{g(v(k))} z(k) \\ g(v) =& f_c + (f_s - f_c) e^{-(v(k)/v_s)^2} \\ F_d(v) =& \sigma_0 z(k) + \sigma_1 \dot{z}(k) + f_v v(k) \end{aligned} \tag{6.14}$$

where h is the same step time as the sample time, which is 0.5 ms.The parameters f_c, f_s, f_v and v_s are the parameters given in (6.9). Denote the dynamic parameters to be estimated as

$$\theta_d = [\sigma_0 \ \sigma_1]^T \tag{6.15}$$

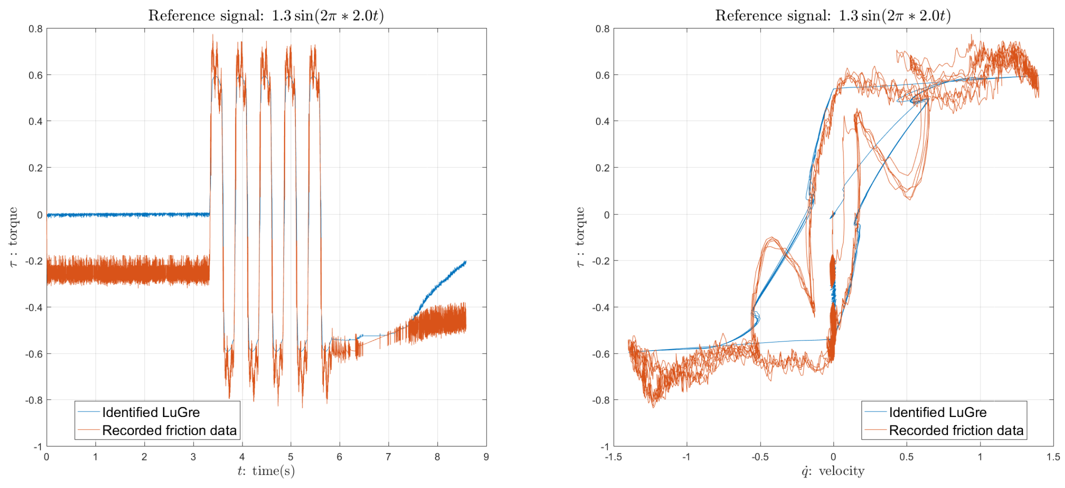
In order to be able to estimate the parameters above, the LuGre model must be able to generate an output for a given time k. To be able to do this the internal state $z(k)$ must be given. Since it cannot be measured, the state is instead simulated from a starting state $z(0)$, given all $k-1$ velocity measurements. The function to estimate is therefore given as

$$F_d\Big(v(k)\Big|\theta_d, \theta_s, z(k)\Big) \tag{6.16}$$

where the function's current output can be calculated through the equations in (6.14) and the current measurement, given the static and dynamic parameters θ_s and θ_d as well as the previous simulated internal state $z(k)$.

The optimization problem is then the same as in (6.7), where the cost function is still the least square cost function, but here changed and given as

$$V(\theta_d) = \sum_{k=1}^{N} \Big(\hat{f}_k - F_d\Big(v(k)\Big|\theta_d, \theta_s, z(k)\Big)\Big)^2 \tag{6.17}$$



(a) Data as a function of time.

(b) Torque as a function of velocity.

Figure 6.3: Identified LuGre friction model for part of the estimation data, namely the reference signal $1.3\sin(2\pi * 2.0t)$ compared to the gathered friction data for the same movement. The model can be seen performing a good fit towards the data.

The optimization problem is solved then by applying Matlab's nonlinear optimization solver `fmincon`, given the constrains that all of the given parameters to estimate are larger or equal to zero. Since the optimization problem belongs to the class of non-convex optimization problems, there is no guarantee that a global minimum will be found. Instead randomized initial values are generated such that the optimization is run iteratively a number of times, where the best solution is chosen according to the given data. The resulting estimated parameters are then given as

$$\begin{aligned} \sigma_0 &= 69.9787 \\ \sigma_1 &= 0.6753 \end{aligned} \tag{6.18}$$

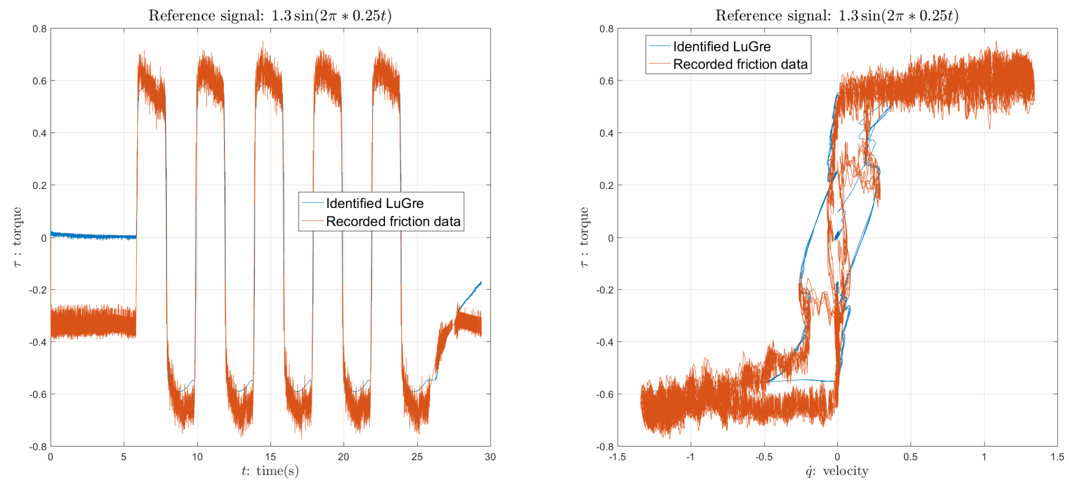
The resulting function output of the estimated dynamic LuGre model from measured velocity input can be seen in Figure 6.3, where it is compared with the input control torque signal generated from the signal $1.3\sin(2\pi * 2.0t)$, which were used in the estimation of the parameters. The function can be seen approximating the behaviour of the friction in the system for the given measurement data.

6.1.3 Validation

The performance of the identified LuGre model is evaluated through a number of validation experiments. As mentioned in the previous section, data records from Table 6.1 that are not used in the identification procedure are used for evaluation of the LuGre model. Performance is here measured by how well the model fits with the recorded data. A well performing identified model should match the hysteresis of the signal and contain a a good fit, in average, to the higher velocities.

The estimated LuGre model can be seen approximating the validation data from the signal $1.3\sin(2\pi * 0.25t)$ in Figure 6.4. The LuGre model and the recorded data for the signal are shown in two cases, namely as functions of time and as functions of the measured velocity. This is shown in Figure 6.4a and 6.4b respectively. The LuGre model can be seen following the outline of the recorded data and capturing parts of the hysteresis and making an average approximation of the noisy higher velocity components.

Further validation of the model when testing different signals can be seen in Figures 6.5. Here reference signals of different amplitudes and frequencies are used in the LuGre model. As it can be



(a) Signals in time. **(b)** Signals in measured velocity.

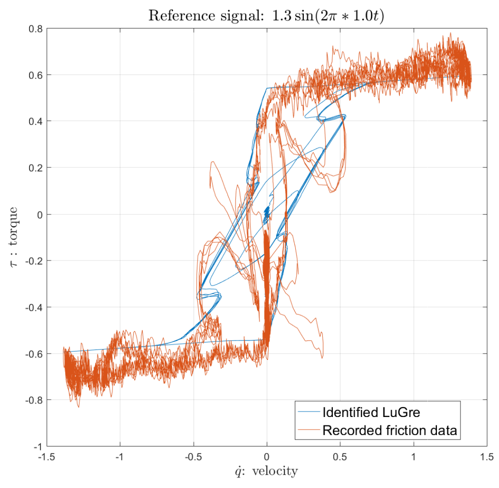
Figure 6.4: Identified LuGre friction model for the validation data of the reference signal $1.3\sin(2\pi * \frac{1}{4}t)$ compared to the gathered friction data for the same movement. The model also performas a good fit on the validation data.

seen the identified model have a tendency to follow the data, even if certain sinusoidal behaviour appears at higher velocities. The LuGre model does not manage to capture these behaviours, which results in a linear approximation of these signals at the higher velocities.

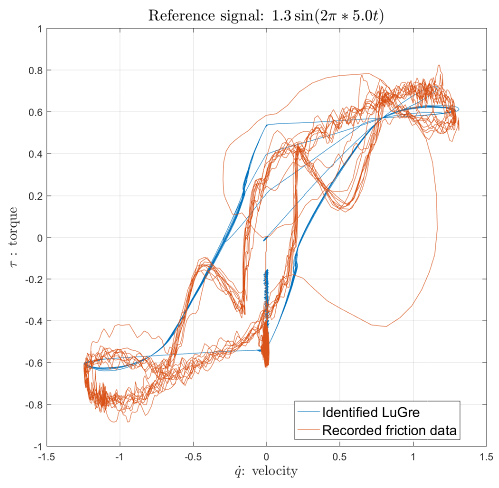
If the validation data is considered in comparison with the original dynamic LuGre model, the results are shown in Figure 6.6. Here the diagrams show the output of the original LuGre model for the given measured input velocities and compared to the estimated friction output. From these diagrams it is obvious that the model fails to represent the behaviour of the estimated friction signal, where as from the previous validation, better approximation could be seen.

Due to the increased performance of the identification presented here, there is the possibility that parts of the previously presented result of the LuGre model presented in Chapter 4 could be improved using the identified model here. Since the project was limited in terms of time, identification of all joints were not possible. This also resulted in that the models where not implemented on the robot manipulator and tested.

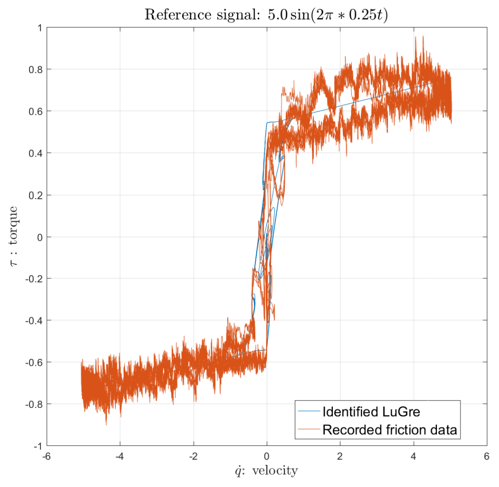
It should be noted however that for larger velocities, such as when the reference signal considered is $10.0\sin(2\pi * 5.0t)$, neither the original LuGre model nor the re-identified model manages to capture the hysteresis behaviour. This can be observed from the diagrams in Figure 6.7 where the blue curve represents the model outputs and the red curve is the estimated friction output. Therefore the LuGre model can be seen working capturing effects up to a certain extent.



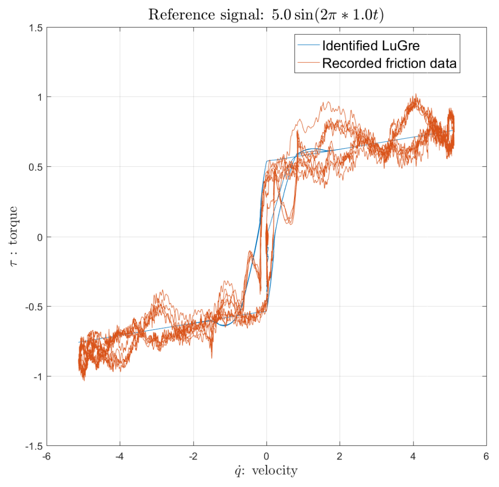
(a) Reference velocity: $1.3\sin(2\pi * 1.0t)$



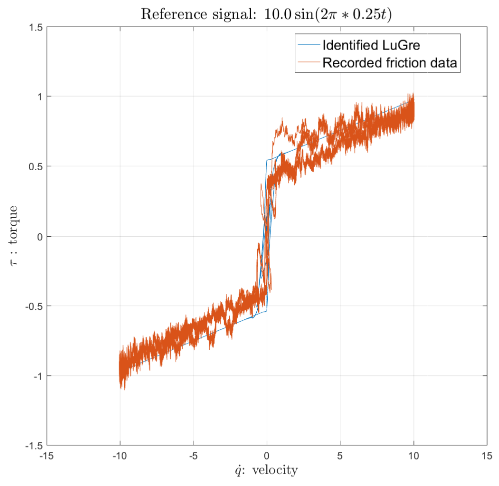
(b) Reference velocity: $1.3\sin(2\pi * 5.0t)$



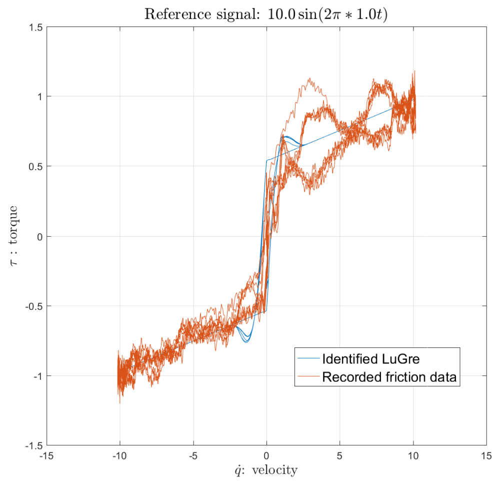
(c) Reference velocity: $5.0\sin(2\pi * 0.25t)$



(d) Reference velocity: $5.0\sin(2\pi * 1.0t)$

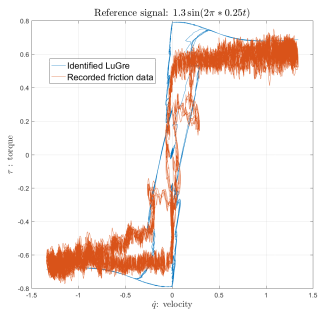


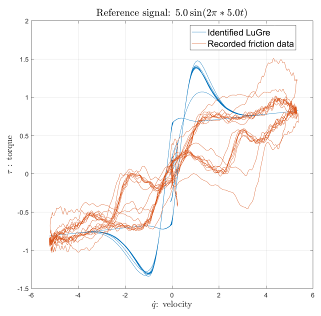
(e) Reference velocity: $10.0\sin(2\pi * 0.25t)$

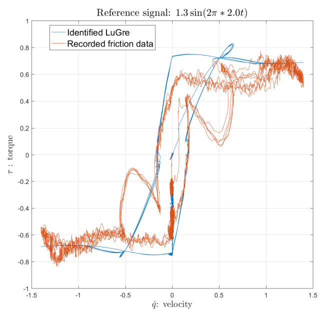


(f) Reference velocity: $10.0\sin(2\pi * 1.0t)$

Figure 6.5: Validation result where the LuGre model is tested on motion not previously trained upon, where the red curve shows the estimated friction output of the robot system. The blue curves represent the identified LuGre model. As can be seen, a good fit is performed for all cases.

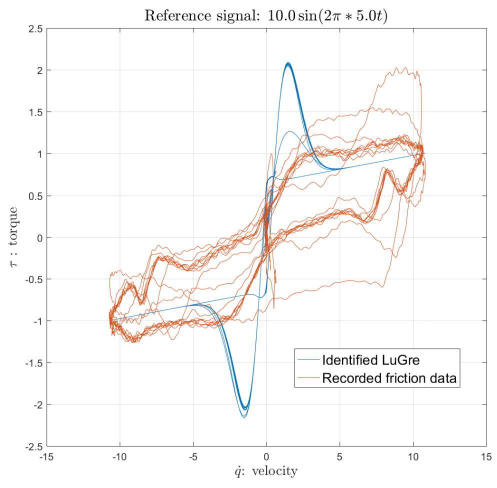


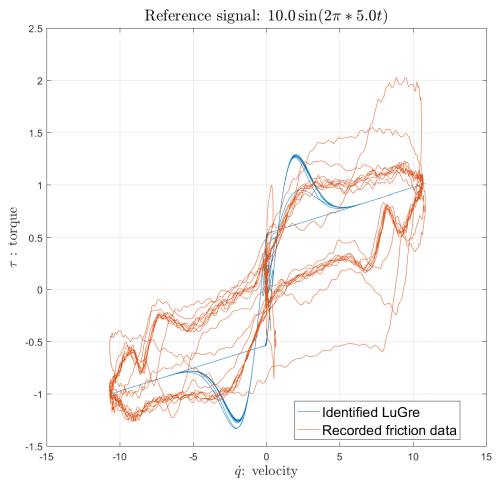




(a) Ref. vel. $1.3\sin(2\pi * \frac{1}{4}t)$. **(b)** Ref. vel. $5.0\sin(2\pi * 5.0t)$. **(c)** Ref. vel. $1.3\sin(2\pi * 2.0t)$.

Figure 6.6: The diagrams show the friction output as a function of velocity of the original dynamic LuGre model for joint 1 used in testing in Section 4, compared to the estimated friction output of the robot manipulator system for joint 1. Three different excitations are shown, where the frequency of the sinusoidal input are modified. The original LuGre model can here be seen performing a bad fit.





(a) Original LuGre model. **(b)** New LuGre model.

Figure 6.7: The diagrams show the model output of the original and the re-identified LuGre model when validated on the reference signal $10.0\sin(2\pi * 5.0t)$ compared to the estimated friction output. The hysteresis behaviour is not captured by either of the models.

6.2 Friction Identification using NARX

Friction identification through the use of a black-box model is here considered in terms of a non-linear auto-regressive (NARX) model. The assumption on the structure of the friction model is here somehow neglected in order to potentially capture effects which where not modeled by the LuGre friction model. As was demonstrated in Section 6.1.3 and in Figure 6.7, the LuGre model fails to capture the complete dynamics of the estimated friction, which remains a motivation to investigate other identification options.

In the following sections, mainly two NARX models are examined where one model is proposed from literature and the other one can be seen as an attempt to introduce more generality. The models have been identified through the use of data generated from the LuGre model such that the models can be examined in an ideal scenario.

6.2.1 Formulation of Models

The first model is based on the works of [22], where the authors use a black-box model in the form of a small neural network containing sigmoid functions to approximate the friction behaviour in a robot system. In the article the neural network used current and prior velocity as input, namely $\varphi(t) = [v(t)\ v(t-1)]^T$ which managed to capture parts of the friction and increase the performance of the system.

The same approach is adopted here, were the regressors are the same, and the nonlinear function g consists of a single layer sigmoid network consisting of three neurons. An illustration of the network can be seen in Figure 6.8 where the elements in the regression vector containing $v(t)$ and $v(t-1)$ are mapped to each neuron, where every edge in the figure has a corresponding weight. The input of the sigmoid function is a scalar value containing a linear combination of input signals and corresponding weights. The output function can then be seen as

$$y = g(\varphi(t), \theta) \tag{6.19}$$

where θ is the vector of weights. The implementation here is slightly different compared to [22] in that the delayed output is mapped to all neurons in the hidden layer, where as in the article, this was only mapped to one neuron.

The second model follows the same structure as the previously described model, however here the regressor is changed. This model can be seen as an attempt of making the model more general, i.e. to better fit to the validation data. The regressor for this model is instead given as

$$\varphi(t) = \begin{bmatrix} v(t) \\ y(t-1) \end{bmatrix} \tag{6.20}$$

where the change is to remove the delayed input $v(t-1)$ and instead introduce a delayed output $y(t-1)$.

6.2.2 Data Generation

The estimation data on which identification is generated from the discretized identified dynamic LuGre model as stated in equation (6.14) for joint 1, where the parameters are given from (6.9) and (6.18). The input to the model was the reference signal

$$v(t) = A\sin(2\pi f t) \tag{6.21}$$

where $f = 1$ and $A = 1$. The generation of data was performed with a sample time of $T = 0.5\ ms$ and simulated for a total of 6 s. For the validation data, the generation of data followed the same pattern, however the frequency was changed to $f = 0.5$.

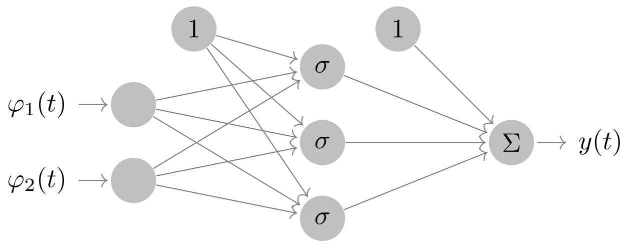


Figure 6.8: Neural network structure for the NARX model with the elements of the regressor as input. The hidden layer has three neurons each containing a sigmoid function σ as presented in (3.7) where the input is a linear combination of input signals and corresponding weights, such that its a scalar operation.

$\varphi_1(t)$
$\varphi_2(t)$
NARX
$y(t)$

Figure 6.9: NARX model representation of the sigmoid network presented in Figure 6.8.

6.2.3 Identification and Validation

Identification with the given models is done by applying the System Identification Toolbox in Matlab. The identification of the model was run with prediction focus for a total of 1000 iterations. The chosen numerical search method for both models are the Levenberg-Marquardt least squares search.

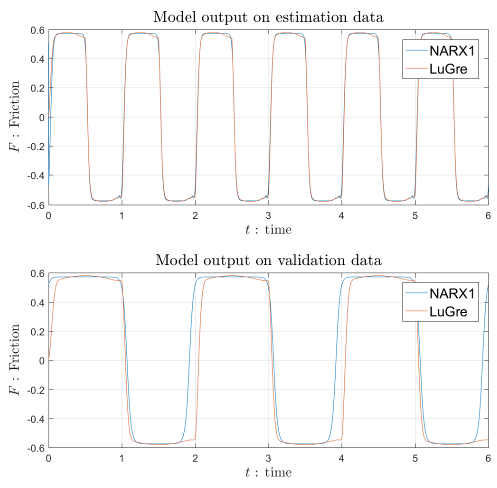
In the System Identification Toolbox the previously described models were formed as NARX models where the `sigmoidnet` was the chosen nonlinear function. A total of 3 hidden neurons was chosen for each model and the linear term was removed. For the first model, the regressor was chosen to be $\varphi(t) = [v(t)\ v(t-1)]^T$ and for the second model $\varphi(t) = [v(t)\ y(t-1)]^T$.

The result of the identification and validation for the first model can be found in the diagrams given in Figure 6.10. In the left hand diagram the model output for the dynamic LuGre model and the identified model are compared, where the output is given as a function of time. In the top graph the models are compared based on the estimation data, where as in the bottom the graph the the model outputs for the validation data is shown. The right hand diagrams however shows the model outputs as a function of velocity instead.

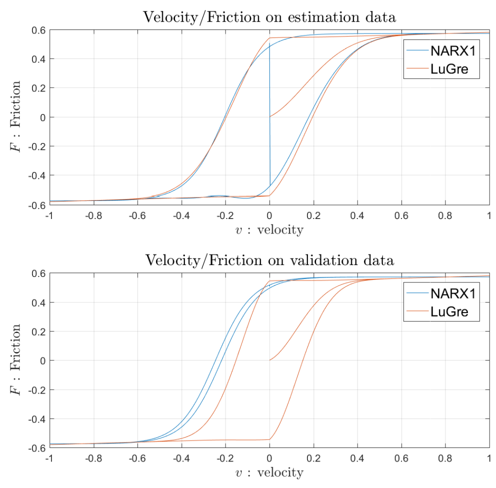
For the estimation data, the first model approximates both the time-friction and the velocity-friction behaviour well, as is seen in the top diagrams in Figures 6.10a and 6.10b respectively. For the validation data, the velocity friction model fails to capture the behaviour for velocities with positive accelerations, which can be seen in the bottom diagram of Figure 6.10a. The velocity-friction function is here not performing well for the identified model.

For the second model, the results are presented in the same manner in Figure 6.11. In the case of the velocity-friction approximation, the model output in the velocity friction case for estimation data, the model performs worse fit to the identification data compared to the first model in Figure 6.10b. However for the validation data, the model output has better fit compared to the first model, even if the velocity friction function does not fit with the dynamic LuGre model.

The identification and validation of these two models indicate that there might be an extension of the regressors, such that the model could capture the dynamics of the LuGre model. This model could in turn be identified using estimated friction data from the real robot and potentially capture more of the friction behaviour which the LuGre model neglects.

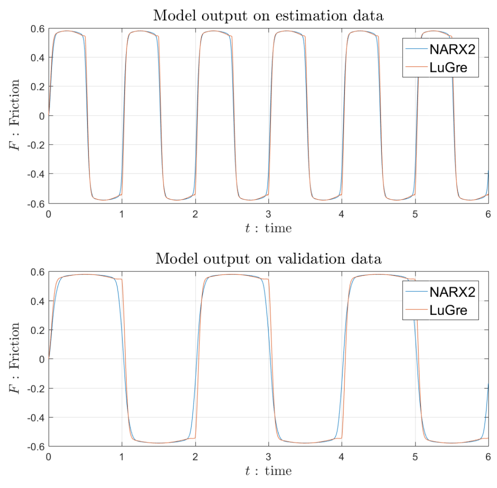


(a) Model output in time.

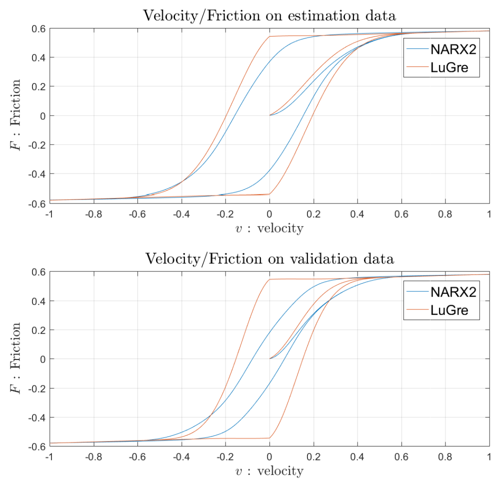


(b) Model output in velocity.

Figure 6.10: The diagrams show the output of the NARX model where the regressor was $\varphi(t) = [v(t)\ v(t-1)]^T$, compared to the LuGre model from which data was generated. The upper graph shows the output for the estimated data, where as the lower graphs show the output from validation data. The model can be seen performing a good fit for the estimation data, however the validation data, it can be seen that the NARX model for velocities with positive acceleration does not perform well.



(a) Model output in time.



(b) Model output in velocity.

Figure 6.11: The diagrams show the output of the NARX model where the regressor was $\varphi(t) = [v(t)\ y(t-1)]^T$, compared to the LuGre model from which data was generated. The upper graph shows the output for the estimated data, where as the lower graphs show the output from validation data. The model can be seen performing a slightly worse fit to the estimation data compared to the model in Figure 6.10, where as for the validation data, it performs a better fit. The model is still not the desired one, since the hysteresis behaviour in validation case is not similar to the LuGre model.

7

Conclusion and Future Works

This chapter summarizes and concludes the work of performing friction modeling and compensation through feed-forward control. In Section 7.1 the conclusion of the work of this thesis is presented. Possible extensions and future work is further presented in Section 7.2. The chapter is concluded in Section 7.3 with some ethical and sustainability aspects that could be considered from the results presented in this thesis.

7.1 Conclusion

Capturing the complete and complex dynamics of friction is hard, especially in a robot manipulator. There is an obvious relationship between the amount of work needed to derive or identify a model and the performance of such a model. In this thesis it has been shown that through a relatively easy learning approach with limited amount of work, friction compensation can be achieved to a level which is similar to feed-forward control using an identified model. The trade-off is the generality of the solution, where an identified dynamic model in general performs better.

The trade-off could be seen in the Chapters 4 and 5, where in the former chapter, an identified static and dynamic LuGre friction model given prior to this work was used in feed-forward velocity control, where it showed increased performance for several motions and speeds. In the latter chapter the approach of learning feed-forward control using B-spline networks during robot operation was investigated, where a static and an approximation of the dynamic friction showed to also increased performance for the same motion in different locations. The learnt controllers fails to show significant improvement on other velocities not used in training, where a proposed solution to part of the problem was to initialize the weights prior to learning. The identified model was here deemed to be general by the applied method of evaluation, where as for the learning approach it was deemed semi-general due to its limitations.

From the work of Chapter 6, where re-identification of the LuGre models were performed for one joint of the robot manipulator, show that further improvement could be gained from a more correct model. In the same chapter the potential of using a nonlinear ARX was explored, which showed an ability to capture a friction model, based on ideal data. Performing system identification in this manner however required extensive amounts of work in order to identify a friction model for one joint. It was argued that perhaps with further modeling more friction effects could be captured.

In this regard, friction is not the only undesired effect which affects the high-accuracy applications for robot manipulators. Other gearbox nonlinearities such as backlash and other hysteresis behaviours are for example known to affect the system in a negative manner, which a friction model is unable to capture. Because of this, achieving even better performance for the tool center position in robot manipulators also requires extensive modeling of other effects than is examined in this thesis.

7.2 Future Works

In water-jet cutting small low velocity circular TCP motions are only a sub-category of problematic scenarios which friction is bound to affect. In this regard, the evaluation of friction compensation in this thesis only considers motions within a limited workspace of 40x40 cm^2. Thus the evaluation of generality of a friction compensation approach is also limited. Its therefore considered to be a future work to extend the motions and workspace of the robot, such that a more general evaluation can be made.

Furthermore, the identification work conducted in Chapter 6 was not tested in neither simulations environments, nor on a real robot. The re-identification showed improvement on one joint due to its fit to estimated friction data, and could therefore be seen as a potential future work to perform and test on all joints of the robot manipulator.

In the case of the NARX identification in the same chapter, further validation and examination of model structure would be necessary in order to capture proper friction dynamics. The data applied in this identification was not generated from the real robot system, which is also extended as future work.

An important point which is not discussed in the thesis is the stability of B-spline networks when applied as a learning feed-forward control. As discussed in [7], there are stability methods when BSN is considered for the ILC case, however when non-repetitive motions are considered, a clear analysis has yet to be derived. This can also be considered as future work in order to make sure that the system does not become unstable during learning.

Furthermore, in this thesis when performing weight adaption of the BSN, the prediction error $(y_m(k) - y(k))$ in (3.17) is replaced with the control signal $\boldsymbol{u_p}$. This replacement is not trivial and motivation has not been found in literature. This is not discussed in [7], however in [15] there is a given proof for a similar case, but under different assumptions. It is therefore considered as future work to show the theory behind the replacement.

7.3 Ethical and Sustainability Aspects

With regards to the ethical aspects of the work conducted in this thesis, it can be stated that robots in industry have a history of taking over tasks in which a human was previously involved. In many cases these tasks are tedious and require a high level of attention of the humans in question, especially when it comes to high accuracy tasks. The potential of increasing the accuracy in robots which are normally not solely designed on achieving high accuracy could replace the need for humans in this position, as the development and production cost decreases. This could potentially result in humans focusing on more creative tasks in which robots are known to be lacking.

Since the result presented in this thesis increases the accuracy in robots which are not solely designed on achieving high accuracy, this work could be extended to be applied to other pre-existing robots. This in turn could create more value for the customer while at the same time save resources being spent building new robots.

Bibliography

[1] S. Yin, X. Li, H. Gao, and O. Kaynak, "Data-based techniques focused on modern industry: An overview", *IEEE Transactions on Industrial Electronics*, vol. 62, no. 1, pp. 657–667, Jan. 2015, ISSN: 0278-0046. DOI: 10.1109/TIE.2014.2308133.

[2] L. Ljung, *System identification: Theory for the user*, English, 2. Upper Saddle River, N.J: Prentice Hall, 1999, ISBN: 9780136566953;0136566952;

[3] B. Siciliano and O. Khatib, *Springer handbook of robotics*, English, 2nd. Cham: Springer International Publishing, 2016.

[4] R. Camoriano, S. Traversaro, L. Rosasco, G. Metta, and F. Nori, "Incremental semiparametric inverse dynamics learning", in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 544–550. DOI: 10.1109/ICRA.2016.7487177.

[5] F. Meier and S. Schaal, "Drifting gaussian processes with varying neighborhood sizes for online model learning", in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 264–269. DOI: 10.1109/ICRA.2016.7487143.

[6] C. Wang, Y. Zhao, Y. Chen, and M. Tomizuka, "Nonparametric statistical learning control of robot manipulators for trajectory or contour tracking", *Robotics and Computer-Integrated Manufacturing*, vol. 35, pp. 96–103, 2015, ISSN: 0736-5845. DOI: http://dx.doi.org/10.1016/j.rcim.2015.03.002. [Online]. Available: //www.sciencedirect.com/science/article/pii/S0736584515000368.

[7] W. Velthuis, "Learning feed-forward control - theory, design and applications", University of Twente, Enschede, The Netherlands, PhD thesis, 2000, pp. xii–200, ISBN: ISBN 90-36514126.

[8] N. D. Cuong and T. X. Minh, "Learning feed-forward control for a two-link rigid robot arm", *International Journal of Electronics and Electrical Engineering*, vol. 3 no. 4, pp. 279–284, 2015. DOI: http://dx.doi.org/10.12720/ijeee.3.4.279-284. [Online]. Available: http://www.ijeee.net/index.php?m=content&c=index&a=show&catid=43&id=193.

[9] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot modeling and control.* wiley New York, 2006, vol. 3.

[10] S. Moberg, "Modeling and control of flexible manipulators", PhD thesis, Linköping University Electronic Press, 2010.

[11] U. Parlitz, A. Hornstein, D. Engster, F. Al-Bender, V. Lampaert, T. Tjahjowidodo, S. Fassois, D. Rizos, C. Wong, K. Worden, *et al.*, "Identification of pre-sliding friction dynamics", *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 14, no. 2, pp. 420–430, 2004.

[12] K. J. Åstrom and C. Canudas-de-Wit, "Revisiting the lugre friction model", *IEEE Control Systems*, vol. 28, no. 6, pp. 101–114, Dec. 2008, ISSN: 1066-033X. DOI: 10.1109/MCS.2008.929425.

[13] P. R. Dahl, “A solid friction model”, DTIC Document, Tech. Rep., 1968.

[14] J. Sjöberg, H. Hjalmarsson, and L. Ljung, *Neural networks in system identification.* Linköping University, 1994.

[15] M. Kawato, “Feedback-error-learning neural network for supervised motor learning”, *Advanced neural computers*, vol. 6, no. 3, pp. 365–372, 1990.

[16] T. J. A. de Vries, W. J. R. Velthuis, and L. J. Idema, “Application of parsimonious learning feedforward control to mechatronic systems”, *IEE Proceedings - Control Theory and Applications*, vol. 148, no. 4, pp. 318–322, Jul. 2001, ISSN: 1350-2379. DOI: 10.1049/ip-cta:20010556.

[17] K. Bossley and C. Harris, “Neurofuzzy modelling approaches in system identification”, Address: Faculty of Engineering and Applied Science, PhD thesis, University of Southampton, 1997. [Online]. Available: https://eprints.soton.ac.uk/250027/.

[18] T. KAVLI, “Asmod - an algorithm for adaptive spline modelling of observation data”, *International Journal of Control*, vol. 58, no. 4, pp. 947–967, 1993. DOI: 10.1080/00207179308923037. eprint: http://dx.doi.org/10.1080/00207179308923037. [Online]. Available: http://dx.doi.org/10.1080/00207179308923037.

[19] J. Matuško, F. Kolonić, Š. ILeš, and A. Slutej, “Friction compensation of gantry crane model based on the b-spline neural compensator”, in *Proceedings of 14th International Power Electronics and Motion Control Conference EPE-PEMC 2010*, Sep. 2010, T5-180-T5-186. DOI: 10.1109/EPEPEMC.2010.5606601.

[20] M. Indri, I. Lazzero, A. Antoniazza, and A. M. Bottero, “Friction modeling and identification for industrial manipulators”, in *2013 IEEE 18th Conference on Emerging Technologies Factory Automation (ETFA)*, Sep. 2013, pp. 1–8. DOI: 10.1109/ETFA.2013.6647958.

[21] A. Carvalho Bittencourt and S. Gunnarsson, “Static Friction in a Robot Joint: Modeling and Identification of Load and Temperature Effects”, *Journal of Dynamic Systems Measurement, and Control*, vol. 134, no. 5, 2012.

[22] Z. Korendo and T. Uhl, “Dedicated neural network design for friction compensation in robot drives”, *Journal of Theoretical and Applied Mechanics*, vol. 40, no. 3, pp. 595–610, 2002.

A

Appendix 1

A.1 Implementation of Dynamic LuGre function

Matlab implementation of the discretized dynamic LuGre model presented in Section 4.2.

```
function [ F ] = dynLuGre_upsample(vin,h,s0,s1,Fc,Fs,vs,SigV,sample)

% Interpolate samples
M=length(vin);
v=interp1(1:M,vin,1:1/sample:M);
gv=Fc+(Fs-Fc)*exp(-(v/vs).^2);
N=size(v,1);


z=zeros(M+1,1);
F=zeros(size(vin));

for i=1:M
    % Calculate the current friction sample
    zdot = vin(i)-s0*abs(vin(i))/gv(1+(i-1)*sample)*z(i);
    F(i)=s0*z(i) + s1*zdot + SigV*vin(i);


    % Since no more data is available, stop the looping
    if(M==i)
        break;
    end

    % Initiate the upsampling, calculate z(i+1)
    Z_it = z(i)+h/sample*zdot;
    % For each interpolated sample, perform update on z
    for j=1:sample-1
    idx = 1+(i-1)*sample+j; % index of the interpolated samples
    zdot = v(idx)-s0*abs(v(idx))/gv(idx)*Z_it;
    Z_it = Z_it+h/sample*zdot;
    end
    % Update z
    z(i+1)=Z_it;

end
end
```