Nonlinear and learning control of a one-dimensional magnetic manipulator

J. Damsteeg
Nonlinear and learning control of a one-dimensional magnetic manipulator

MASTER OF SCIENCE THESIS

For the degree of Master of Science in Systems and Control at Delft University of Technology

J. Damsteeg

June 4, 2015

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology
This thesis describes the control design for a magnetic manipulator. The experimental setup has four coils (electromagnets) which can be used to shape the magnetic field above the magnets by controlling the currents through all coils. By doing this, a steel ball can be positioned horizontally in one degree of freedom. The control objective consists of accurate and fast regulation of the ball. The magnetic force created by the coils is highly nonlinear. An empirical model is used to approximate the force exerted by each coil. This leads to a constrained nonlinear control problem.

Multiple nonlinear controllers are designed: a Feedback Linearization (FL) controller as benchmark, a State Dependent Riccati Equation (SDRE) controller, a Constrained-SDRE (MPC approach) and a Nonlinear Model Predictive Controller (NMPC). Furthermore, two learning controllers are designed: a Reinforcement Learning controller and an Imitation Learning controller based on Local Linear Regression (LLR).

All controllers are evaluated in simulation. A satisfactory performance was achieved for both the FL controller and the constrained-SDRE controller. The SDRE controller obtained a slightly slower response. The NMPC is not feasible in real-time, although it is useful as it indicates a nearly optimal behaviour.

The simulation study of the learning controllers showed that the actor-critic RL controller is able to learn the desired behaviour, although the overshoot is not prevented and it takes a large number of trials to converge. The Imitation Learning controller performs better as it is able to match the performance from the C-SDRE controller in a small number of training and learning trials.

Based on the results in the simulation study, four of them are implemented on the experimental setup. From the results of the model-based controllers on the setup it can be concluded that the Constrained-SDRE performs best in terms of settling time, overshoot and control effort. The imitation learning controller is able to match the performance of the C-SDRE controller, and performs better than the C-SDRE in terms of adapting to a different ball size.
# Table of Contents

**Acknowledgements**  vii

---

## 1 Introduction

1-1 Motivation  .................................................. 1
1-2 Problem statement and research goals  .................................. 2
1-3 Outline  .................................................. 2

## 2 Literature review

2-1 Introduction  .................................................. 5
2-2 Comparable setups and applications  .................................. 5
2-2-1 Magnetic manipulator in one direction  ....................... 6
2-2-2 Planar magnetic manipulation  .................................. 7
2-2-3 Medical applications  ....................................... 7
2-2-4 Stepping motors  ............................................ 8
2-2-5 Conclusions  ................................................. 9
2-3 System analysis and control methods  .................................. 9
2-3-1 Lyapunov's linearization method  .................................. 9
2-3-2 Gain scheduling  ............................................. 10
2-3-3 Feedback Linearization  ....................................... 10
2-3-4 Lyapunov's direct method  ..................................... 11
2-3-5 Sliding mode control  .......................................... 12
2-3-6 State Dependent Riccati Equation control  .................. 12
2-3-7 Constrained State Dependent Riccati Equation - MPC approach  .................................. 14
2-3-8 Summary and conclusion  ....................................... 14
2-4 Learning control: Reinforcement learning  .................................. 15
2-4-1 Introduction  ................................................. 15
2-4-2 Reinforcement learning  ....................................... 15
### Table of Contents

#### 3 Modelling

- 3-1 Experimental setup ........................................... 20
- 3-2 Modelling the magnetic field ................................. 21
  - 3-2-1 Models used in literature ............................... 21
  - 3-2-2 Analytical model of magnetic force .................. 22
  - 3-2-3 Finite Element Method and Boundary Integral Methods . 23
  - 3-2-4 Empirical model derivation ............................ 24
- 3-3 State space model ............................................. 25
  - 3-3-1 Friction ...................................................... 25
  - 3-3-2 Magnetic force ............................................. 26
  - 3-3-3 System identification based on empirical model ...... 26
  - 3-3-4 Continuous-time State space model .................. 29

#### 4 Control design

- 4-1 Introduction ..................................................... 31
- 4-2 Discretization of continuous time model .................. 31
- 4-3 Observer design ............................................... 32
- 4-4 Feedback Linearization control ............................. 33
- 4-5 State Dependent Riccati Equation .......................... 37
- 4-6 Constrained State Dependent Riccati Equation control - MPC approach .... 39
  - 4-6-1 Introduction and motivation ............................ 39
  - 4-6-2 MPC problem formulation ............................... 41
- 4-7 Nonlinear Model Predictive Controller ..................... 44
- 4-8 Conclusion ...................................................... 46

#### 5 Evaluation on experimental setup

- 5-1 Selection of the sampling frequency ........................ 47
- 5-2 Feedback Linearization control ............................. 47
- 5-3 State Dependent Riccati Equation control ................ 49
- 5-4 Constrained State Dependent Riccati Equation control - MPC approach .... 51
- 5-5 Conclusions ..................................................... 53

#### 6 Learning control

- 6-1 Introduction ..................................................... 55
- 6-2 Actor-Critic Reinforcement Learning ....................... 55
  - 6-2-1 Introduction ............................................... 55
  - 6-2-2 Reinforcement Learning .................................. 56
  - 6-2-3 Actor-critic method ...................................... 57
  - 6-2-4 Approximating critic and actor by Radial Basis Functions .................. 57
  - 6-2-5 Exploration and exploitation in actor-critic ........ 58
  - 6-2-6 Actor critic basic algorithm ............................ 59
# Table of Contents

6-2-7 Simulation results of actor-critic RL ........................................ 60
6-2-8 Implementation of RL on the experimental setup .......................... 63

6-3 Local Linear Regression Imitation learning control .......................... 63
  6-3-1 Introduction ........................................................................... 63
  6-3-2 Online learning using LLR ................................................... 64
  6-3-3 Simulation results of Imitation learning .................................. 66
  6-3-4 Experimental results of Imitation learning .............................. 69

6-4 Conclusions ................................................................................. 73

7 Conclusions and further research .................................................... 75
  7-1 Conclusions .............................................................................. 75
  7-2 Future work ............................................................................... 76

A Hardware ..................................................................................... 79
  A-1 Position measurement ............................................................. 79
  A-2 Electronics ............................................................................... 79

Bibliography ..................................................................................... 81

Glossary ........................................................................................... 85
  List of Acronyms ............................................................................ 85
  List of Symbols ............................................................................... 86
Acknowledgements

First of all, I would like to thank my supervisor Prof. Robert Babuška for introducing me to this subject and assisting me during the graduation project. His enthusiasm for control engineering in general, motivating comments and useful new ideas helped me a lot throughout the whole graduation project.

Furthermore, I would like to thank my daily supervisor, Subramanya Nageshrao for all of his help during this thesis, by providing me with new ideas, feedback and comments on the thesis report.

I also want to say thanks to the technicians who helped me with the problems and challenges with the experimental setup: Jos van Driel, Kees Slinkman and Will van Geest. I am thankful for their time and patience during the preparations of the experiments.

Finally, I am very grateful to my wife, Selma, for all her patience and support during the progress, and critical comments on this report.

Delft, University of Technology
June 4, 2015

J. Damsteeg
Chapter 1

Introduction

In this thesis a controller is designed for the magnetic manipulation setup. The setup (figure 1-1) has four electromagnets which can be used to shape the magnetic field above the magnets. By doing this the steel ball can be positioned in one direction, along with the direction of the coils. The control task consists of accurate and fast stabilization to a desired set point.

![Figure 1-1: Experimental setup, which has four current-controlled coils to position a steel ball in one-direction. The position of the ball is measured by a laser sensor.](image)

1-1 Motivation

Magnetic manipulation has several advantages compared to traditional manipulation approaches. First of all, magnetic manipulation is contactless. A gripper is not necessary, which opens up new possibilities for actuation on a micro-scale. Furthermore, it enables actuation in new, hazardous environments for which it is not possible to use a mechanical actuator. Finally, a magnetic manipulator enables parallel actuation.

This requires intelligent control methods to achieve fast and accurate actuation. This is not trivial as the magnetic force of the coils is nonlinear with the distance of the ball from the axis of the coil [1]. Furthermore, several effects are hard to model accurately, e.g. the
influence of the steel ball itself on the magnetic field [2]. This increases the complexity of the overall system and makes it harder to obtain an accurate model. For model based control design this limits the achievable control performance.

1-2 Problem statement and research goals

The goal is to improve the results by [2] and [1] on a comparable setup. In those articles a settling time of approximately 1 s (experimental result, 1D setup) to 2 s (simulation, on 2D setup) was achieved. The overshoot was reduced to zero by [2] on the one-dimensional setup. The simulated trajectories on the planar manipulation setup have a significant overshoot, in the order of 10% [1].

The controller should fulfill the following requirements:

- Fast and accurate positioning: in terms of settling time (less than 1 second) and overshoot < 2% and a steady-state error less than 1%.

- Control method must be generalizable w.r.t. degrees-of-freedom (DOF), object size, setup geometries et cetera.

- The controller should use the four electromagnets efficiently. This is measured by the consumed energy by all four coils. This is not provided in previous literature, but it is a useful measure to compare the control methods throughout this thesis.

1-3 Outline

Chapter 2 gives an overview of the relevant literature on magnetic manipulation. This includes literature on comparable setups and problems, modelling and control aspects of the design problem. This chapter considers the magnetic manipulation design problem from a wider perspective than the particular 1D setup only, which is the main focus from Chapter 3 on. The model will be derived and further explained, after which several controllers are designed in Chapter 4. All designed controllers are compared in Chapter 5, which includes both simulations and actual experiments on the setup. The full outline is illustrated in the following figure:
Figure 1-2: Roadmap of this thesis
Chapter 2

Literature review

2-1 Introduction

This chapter covers relevant literature in the field of magnetic manipulation of physical systems. Magnetic manipulation consists of a wider class of problems where the objective is to create and shape a magnetic force field such that an object can be positioned within that field. This is sometimes also referred to as programmable force/vector fields [2]. Several goals can be achieved with this concept, with entirely different geometries and for different amounts of degrees of freedom; varying from only one dimension (e.g. [3]) to a large number of degrees-of-freedom e.g. a 5 DOF magnetic manipulator in a fluid as described in [4]. From Chapter 3 on, this thesis will focus on the particular setup described above. An appropriate model will be selected after which several controllers are designed. First, however, this chapter covers the broader class of problems, by looking at comparable problems, applications and implementations of control concepts for such problems. This will put the problem into perspective and may help to solve comparable problems and challenges.

2-2 Comparable setups and applications

There exist several articles [1] [2] which describe models and controllers designed for magnetic manipulator setups that are comparable to the one described in the introduction of Chapter 4. It is interesting to briefly discuss their objectives and results, as it may be useful for this thesis. The main advantages of magnetic manipulation are threefold:

- Contactless. A mechanical actuator, e.g. a gripper, is not necessary. This gives new possibilities for positioning micro-scale objects [2]. This is not trivial due to scaling effects, which means that the surface tension starts to dominate over the mass. The object may stick to the gripper when it should be released.
• New environments. In the human body or in aggressive chemicals it may be beneficial to use magnetic manipulation, and it may be possible that this contactless positioning is then the only feasible option [5].

• Parallel actuation. By using magnets it is possible to shape the force field such that several objects are positioned at the same time. This gives new possibilities in many different working fields and existing systems. Many conventional mechanical actuators focus on one individual object only [1].

One of the downsides of magnetic actuation is that the control design may not always be trivial [2]. It is hard to model the magnetic field accurately and it quickly leads to a constrained, nonlinear and multi-input control problem. Advanced control methods are necessary to achieve reasonable control performance, in terms of rise time, overshoot and efficient usage of the capacity of the coils.

Currently, several magnetic manipulators have been designed with the goal to manipulate an object in a fluid [5], [4]. The latter research works on a manipulator of a small needle in an eye in three dimensions. This is realised by three large coils surrounding a human eye. This will be further explained in this section.

A significant part of comparable researches is done in the medical field. This environment has many useful applications for magnetic actuation. This will be explained and further elaborated on in subsection 2-2-3. Furthermore, this section will cover future possibilities of magnetic actuation.

### 2-2-1 Magnetic manipulator in one direction

The Magnetic Manipulator (sometimes also abbreviated to 'MagMan') concept consists of a number of electromagnets which are used to position a ferromagnetic object (e.g. a steel ball), without any direct contact between the coils and the object. This is done by shaping the magnetic field by controlling the current through the coils. Note that an electromagnet cannot exert a repelling force on the steel ball. This is a hard constraint for the actuators, which makes the overall control problem more complex. A constrained multi-input control problem needs to be formulated if one wants to position the ball without (or minimal) overshoot.

In this thesis a one-dimensional experimental setup is considered. This makes it possible to focus on more sophisticated methods, before expanding this to multiple dimensions. Preliminary research on a one-dimensional setup is available in [2] and [3]. In both papers several controllers have been designed. The latter research worked on the modelling of the magnetic field, designed and produced the experimental setup, and designed a Feedback Linearization (FL) strategy together with a gain scheduling controller. They were only able to test their controllers in simulation.

In the research in [2], which was performed at the same department as [1], an adaptive learning controller was designed. This research was able to improve the conventional controllers with Reinforcement Learning by supplying an additive input on a FL controller.
2-2-2 Planar magnetic manipulation

The work of [2] and [1], focused on designing a controller for the planar manipulation setup. Feedback linearization and some learning controllers were designed to deal with unmodelled phenomena like the eddy currents and the interaction between the coils [6]. Furthermore, an online identification procedure was implemented to estimate the parameters in the model. A reasonable performance was achieved with a feedback linearization controller on a planar manipulation setup. Although the performance in terms of settling time and overshoot is not specifically mentioned, it can be concluded from the figures that a relatively slow response was achieved. The settling time is approximately 2 seconds, with an overshoot of around 10% [1].

It is interesting to note that the research in [2] was limited by the sampling rate of the vision sensor, which was the only way to measure the smallest balls (20 mm, 32.7 g). Their alternative sensor, a touch foil, was not able to measure a smaller ball with satisfactory accuracy. The touch foil is also used in [1]. It can be questioned whether this is a valid method to measure the position of the ball(s), as the touch foil makes direct contact with the ball. This might influence the dynamics of the ball. There exist alternative methods to measure the position of the ball without exerting an extra (disturbing) force on the ball, e.g. by a vision sensor such as a camera.

Some of the existing applications also use permanent magnets to be able to exert larger forces on the object. Furthermore, those researches mostly focus on linear models or (feedback) linearizing controllers as opposed to nonlinear control strategies which directly incorporate the nonlinearities in the feedback control strategy.

2-2-3 Medical applications

For medical applications of magnetic manipulators the main goal is to design a minimal invasive system. This is beneficial for a patient as the damage to the body can be significantly reduced. Furthermore, new possibilities arise like site-specific drug delivery [7]. The possibility of magnetic manipulation in a medical environment was already noticed in the 80's and 90's [7]. Recent research designed manipulators for micro fabricated magnetic structures (nano particles), which can then be manipulated within the human body for many different purposes (as actuator in a surgery, cargo delivery and heating therapy [8].

Magnetic manipulation also has some disadvantages. In a medical environment the presence of magnets may conflict with operating tools, e.g. in a surgery room. For these environments it is highly beneficial to design a system without using permanent magnets. This makes the control design harder, but is a big advantage for medical applications as the electromagnets can be turned off when they are not used.

In article [4] a small needle is manipulated inside an eye in three dimensions, by several coils surrounding a human eye. Their goals is to control very small objects, 'micro robots', which leads to different physical properties ¹. Their research is relevant though, as they manipulate an object by using electromagnets only. In their more recent work [9] a manipulator is

¹On a micro scale, the surface tension dominates over the mass, compared to a scale in the order of magnitude of centimeters. This can be explained from the fact that the mass scales with a factor \( \frac{1}{10^3} \), while the surface tension scales with \( \frac{1}{10^2} \).
designed for an artificial bacterial flagella (ABF), which is a circulator robot (only 60 micrometers in length) with a magnet on top.

Actuators on a micro-scale can be used for targeted drug delivery, which enables the medical specialists to target the medicine to specific locations in the body [9].

2-2-4 Stepping motors

In a stepping motor the magnetic field is shaped such that the rotor spins with a certain torque and angular speed. Although the geometry is very different from a magnetic manipulator, comparable models, controllers and side effects may arise in such systems. Therefore stepping motors are analysed to see whether the gained knowledge in this field can be used for the current thesis work.

Stepping motors, or the class of brushless permanent magnet (PM) motors, are widely used in many applications. Stepping motors are most used in low-cost applications for position control without position feedback. Most of these motors are cylindrical, with a rotary part (rotor) and a stationary part (stator). All brushless motors are built with electrical windings on the stator and permanent magnets on the rotor [10].

Figure 2-1: Stepping motor with three phases [10]

Both a stepping motor and the magnetic manipulator setup create a magnetic force field such that the position and velocity of an object can be controlled. The manipulated object, however, differs as the rotor of a stepping motor is a permanent magnet. This enables the coils of the stator to exert a repelling force, which is not possible on the magnetic manipulator setup. This fundamental difference makes it hard to directly apply the control methods from stepping motors to the magnetic manipulator.

One of the observed phenomena in stepping motors is the cogging force. This effect, also known as cogging torque, describes the interaction of the rotor magnets acting on the stator teeth or poles independent of any current [10]. Cogging torque occurs when the permanent magnets are close to the ferromagnetic material present on the rotor, which is maximal when the teeth of the rotor are aligned with the stator. Note that this effect is very small compared to the actual torque produced by a motor. However, for specific scenarios this may be a
problem. In article [11], a feedforward cogging torque compensator is designed. An experimental setup was designed to measure the cogging torque, which enabled them to perform system identification on the cogging torque model. This model was used in their feedforward controller. Although the magnetic manipulator in this thesis does not contain any permanent magnets, there may occur comparable effects. As explained earlier, the electromagnets consist of copper windings around a iron core. The latter is used to magnify the magnetic field. However, this core may act as a soft magnet, which means it will influence surrounding ferromagnetic material. This effect might be observed when the steel ball is located close to the axis of the coil. The remaining magnetic field attracts the ball, even if no current flows through the coil. The main difference with the problem in [11] is that the cogging torque in a stepping motor occurs periodically and depends on the rotor angle w.r.t. the stator [11]. In the magnetic manipulator this might also depend on time as the magnetic field from the iron core varies with time. It should be verified whether this effect is significant and how it can be compensated.

2-2-5 Conclusions

In general, magnetic actuation has a clear advantage of (potentially) quick actuation without making direct contact. This gives new possibilities to work in dangerous environments like aggressive chemicals.

Furthermore, the potential of quick switching makes this a very useful actuator for many mechanical applications. It also opens up possibilities for very small micro-robot to achieve several different tasks. This may vary from medical applications (as explained in the previous section) but in general such micro-robots can be used in many micro-scale fabrications. Several research has been done on magnetic manipulators on a comparable scale of the experimental setup of this thesis.

2-3 System analysis and control methods

Several feedback control methods are described in the relevant literature. In this section the most common control methods for non-linear systems are described. This includes linear controllers based on the linearized systems. From there on, nonlinear control methods are introduced, which directly incorporate the nonlinearities in the feedback.

2-3-1 Lyapunov’s linearization method

This is a popular and conventional control approach to output or state feedback of nonlinear models. In almost all real systems several nonlinearities are introduced. With this method the nonlinear model is linearized around the required operating point, leading to a linear state space representation of the system. The original system model is thereby simplified into a linear form. This is clearly beneficial from a control point of view as many control concepts are available for linear systems. These systems and available control concepts are intuitive and have been proven to be successful in many practical applications. Popular examples of
linear control methods are PID control and model-based optimal control design, e.g. Linear-Quadratic Regulator (LQR).

The main limitation of the Lyapunov’s linearization method is that the linear approximation is only valid close to the operating point. Depending on the order of the original model this clearly affects the performance, especially when the system state is not close to the operating point [12].

2-3-2 Gain scheduling

This technique is related to the previous method, as the model is linearized first to be able to design a controller using linear control theory. There are several methods that apply this basic concept. The idea of this general concept is to select several operating points and design a linear controller for each point. The controller is interpolated based on the operating points. This controller thus consists of scheduled control laws, depending on the current state, which results in a global compensator [12].

2-3-3 Feedback Linearization

This control method uses an exact state transformation and feedback, instead of a linear approximation of the original system dynamics. The system itself is not linearized to state space form, but the system model is redefined such that the nonlinearities are cancelled. Within this context two main methods can be distinguished: input-state and input-output linearization. The first method completely linearizes the state equation, the goal of the latter is to obtain a simple and direct input-output relationship [13].

Feedback linearization can be considered as a method in which the original system models are transformed into equivalent models of a simpler form. It is important to mention that the complexity of a model is partly based (e.g. dependent), on the choice of the reference frame or coordinate system [12]. This may lead to different state space representations, the system and input matrices are not unique, which may result in unnecessary complexities in the analysis of the system dynamics and control design.

In general, FL can be applied to a nonlinear system of the form:

\begin{align*}
\dot{x} &= f(x) + g(x)u \\
y &= h(x)
\end{align*}

(2-1)

(2-2)

With \(f(x), g(x)\) and \(h(x)\) are nonlinear functions of \(x\). The control design now focuses on the question whether it is feasible to find a state feedback control input:

\[u = \alpha(x) + \beta(x)v\]

such that a linear (equivalent) system is obtained, which will have the form

\[\dot{z} = Az + Bv\]

(2-3)

Feedback linearization can also be used in combination with robust and adaptive nonlinear control design [12].

J. Damsteeg Master of Science Thesis
This control method is used in comparable researches on the magnetic manipulator (see [1]). This technique will be explained in more detail in section 4-4. From there on, a Feedback Linearization controller will be designed which forms a benchmark for this thesis work. This makes it possible to compare the more sophisticated methods in this thesis to the current work in this field.

2-3-4 Lyapunov’s direct method

This method is based on the observation that if the total energy of a mechanical system keeps dissipating, it can be concluded that the system is stable and will eventually settle to a certain equilibrium point.

Note that a qualitative definition of stability is as follows: a system is stable 'if starting the system somewhere near its desired operating point implies that it will stay around the point ever after’ [12].

The first objective is to formulate a Lyapunov function $V(x)$, belonging to an autonomous system $\dot{x} = f(x)$, which fulfills the following requirements:

1. $V(0) = 0$
2. The function $V(x)$ is positive definite
3. The time-derivative along all state trajectories is negative semi-definite: $\dot{V}(x) \leq 0$

Such a function can be graphically represented as follows:

![Figure 2-2: Example of a Lyapunov function [14]. As the derivative $\frac{dV}{dt}$ is negative for all states, the trajectory converges to the origin]

If this function exists for the system $\dot{x} = f(x)$, the equilibrium point $x = 0$ is stable.
If the derivative of $V(x)$ is strictly negative definite: $\dot{V}(x) < 0$, it can be concluded that the system is asymptotically stable.

This method can thus be used to determine whether an uncontrolled system is stable. However, if one can formulate a $V(x)$ function for a closed-loop system with an internal controller, the stability of a controller can be analysed. Finally, and most relevant in this context, a controller can be designed such that the closed-loop system dynamics can be described using a Lyapunov function. This is the essential principle when Lyapunov’s direct method is applied for control design purposes.
2-3-5 Sliding mode control

For many applications the model of the corresponding system is imprecise. The complexity of obtaining a precise model is often traded for simpler and approximate models. Moreover, the plant may have unknown parameters. These different model inaccuracies can be classified in two (main) categories:

- structured uncertainties (inaccuracies in the model itself)
- unstructured uncertainties (i.e. underestimation of system order)

Sliding mode control is a robust control method which is able to explicitly deal with modelling uncertainties. [12].

The control problem is to make sure that state $x$ tracks a specific trajectory, $x_d$, with the existence of model imprecision on $f(x)$ and $b(x)$.

Consider the following single input nonlinear system:

$$x^{(n)} = f(x) + g(x)u$$  \hfill (2-4)

The tracking error is now equal to:

$$\tilde{x} = x - x_d = [\tilde{x} \, \dot{x} \, ... \, \dot{x}^{n-1}]^T$$  \hfill (2-5)

From there a time-varying sliding surface $S$ is defined:

$$s(x; t) = (\frac{dx}{dt} + \lambda)^{n-1} \tilde{x}$$  \hfill (2-6)

where $\lambda$ is strictly positive, and $\tilde{x}$ is the tracking error as defined above [12].

The control problem is now to remain on the surface $S(t)$ for all $t > 0$. The unique solution of differential equation $s = 0$ is $\tilde{x} = 0$, for the initial condition $x_d(0) = x(0)$. The two main steps of this control method are thus to design a certain stable sliding surface and secondly to design a control law which makes sure the states goes towards that surface (for which $s = 0$). A large benefit of this method is that the controller can be designed such that the states converge in finite time. Furthermore, sliding mode control is a robust method which is a main advantage for uncertain systems.

2-3-6 State Dependent Riccati Equation control

This control technique considers a nonlinear system of the form

$$\dot{x} = f(x) + g(x)u$$  \hfill (2-7)

with state $x$ and control input $u$. Note that $f$ and $g$ are some nonlinear functions. Such system can in general be rewritten to the extended linear form with State-Dependent-Coefficients (SDC):

$$\dot{x} = A(x)x + B(x)u$$  \hfill (2-8)
where \(A(x)x = f(x)\) and \(B(x) = g(x)\). It is known (see [15]) that for a system with multiple variables there are infinite number of ways to retrieve the State-Dependent-Coefficients (SDC) form [15] from the general nonlinear structure 2-7.

For this system structure it is possible to design a State Dependent Riccati Equation (SDRE) controller. This basically is a nonlinear, suboptimal, state feedback control method [15]. It can be seen as equivalent to LQR, except matrices \(A\) and \(B\) may now depend on \(x(t)\) (and thus 'state-dependent'). Such systems cannot be solved (without using linearization techniques) with the normal linear techniques as \(A(x(t))\) and \(B(x(t))\) are now time variant. Note that a LQR controller is only suitable for Linear Time Invariant (LTI) systems. However, in discrete time the matrices \(A\) and \(B\) are treated constant during each sample period \(k\), as \(x(k)\) then has a known value at time step \(k\). This makes it possible to compute the control action by solving an LQ (infinite horizon) optimal control problem each instant, by calculating a new solution of the Riccati equation (in continuous time, for simpler notation. For discrete time systems, the theory is analogue to continuous time)

\[
A^T(x)P(x) + P(x)A(x) - P(x)B(x)R^{-1}(x)B^T(x)P(x) + Q(x) = 0 \tag{2-9}
\]

Note that the above equation shows that SDRE control design also offers the flexibility of using state dependent design matrices \(Q(x)\) and \(R(x)\).

This solution of equation (2-9) leads to an update of matrix \(P(\geq 0)\), which thus depends on state \(x\). Based on the new value of matrix \(P(x)\), the state feedback gain is then also updated and implemented in the new time step: 

\[
u(x) = -K(x)x.
\]

With

\[
K(x) = R^{-1}(x)B^T(x)P(x) \tag{2-10}
\]

The main condition for stabilizability is that the rank of the \(n \times nm\) state dependent controllability matrix

\[
M_c(x) = [B(x) A(x)B(x) ... A^{n-1}(x)B(x)] \tag{2-11}
\]

has rank \((M_c) = n\) [16]

It is important to note that, contrary to LQR, this method is suboptimal with respect to the performance index \(J\):

\[
J = \frac{1}{2} \int_{t_0}^{\infty} (x^TQ(x)x + u^TR(x)u)dt \tag{2-12}
\]

See also [15] for more in depth information and proofs.

The main advantage of this method is its simplicity. The algorithm has proven to be effective in several practical applications. Note that the above derivation is made for a continuous-time system. On the experimental setup a discrete-time variant is needed. This is analogue to the continuous-time framework, and will be further dealt when an actual SDRE controller is designed in section 4-5.

---

\[\text{This approach is also known as extended/apparent linearization or SDC parametrization [16]}\]
2-3-7 Constrained State Dependent Riccati Equation - MPC approach

The concept of SDRE as described in the previous section, has some significant advantages compared to other linearization methods. This controller is not able, however, to directly incorporate input constraints. As mentioned in section 2-2, the magnetic actuators have a (lower bound) input constraint for this setup as they can only attract the steel ball. This constraint is ignored in the computation of the optimal control (LQR) input.
To overcome this a controller is needed that is able to deal with these constraints. The most popular method to design an optimal, multi-input, constrained control method is the concept of Model Predictive Control. This model-based controller predicts the future state evolution and optimizes the control input in real-time based on this prediction. The (input) constraints can be directly taken into account in the optimization.

Constrained State Dependent Riccati Equation control (as referred to by [17]) includes the implementation of state and input constraints. This is a large advantage as this makes it possible to explicitly take saturation of the actuators into account during the optimization procedure. This is done by applying the concept of Model Predictive Control (MPC) based on Quadratic Programming (QP), in combination with the control principle in a SDRE controller. It can also be referred to as 'State Dependent Model Predictive Control'. This concept was also used in [18]

As explained in the previous section 2-3-6 some nonlinear system can be rewritten into SDC form. This makes it possible to use a linear state-space system during each sample time. Instead of computing an optimal control input based on LQR (in which it is not feasible to explicitly incorporate the constraints), an optimal control sequence is computed based on state prediction. The constraints on the state and/or inputs are incorporated in the optimization each sample time. By combining the concepts of MPC together with SDRE, linear MPC theory can be applied. This means that the use of a nonlinear Model Predictive Control can be avoided. This is beneficial as such a controller is more complex and computationally much more intensive (which will be shown in section 4-7). Note that a clear restriction is that the prediction is still based on a constant input-matrix ($B$), which is state dependent. This also holds for the SDRE controller, which means that C-SDRE has a clear preference. A better performance is expected compared to SDRE, as long as it is computationally feasible on the setup. Note that the matrices $H$ and $f$ in the framework of QP must be recalculated each sample time. It should be later verified if this is feasible for the real-time implementation of this controller.

2-3-8 Summary and conclusion

Several 'conventional' controllers have been explained in the previous subsections. It is now interesting to briefly analyze the main advantages and disadvantages of these control concepts, before the focus slightly changes to learning control. In general an advantage of conventional control is that this leads to a systematic approach and theoretical guarantees of stability and robustness. A downside is that the control design may be time consuming as it is conceptually difficult. It certainly requires control engineering expertise. Furthermore, conventional control is limited for (highly) nonlinear systems: sometimes a linear controller cannot stabilize, the
system: either because the modelled nonlinearities are not specifically taken into account in the control design, or because the nonlinearities are not included in the model at all (uncertainties or unknown effects).

2-4 Learning control: Reinforcement learning

2-4-1 Introduction

As a human we learn by interacting with the environment. This gives rich information about the surroundings, from which the consequences of certain actions can be learned. Using sensors the feedback of the environment is used to change future behaviour. This eventually makes it possible to learn what to do to reach certain goals in the future, without any available model of the environment. This data based principle is called Reinforcement Learning (RL). Several learning approaches are available; the main concept focuses on goal-directed learning from interaction [19].

2-4-2 Reinforcement learning

This subsection briefly describes the basic working principle of Reinforcement Learning, including the definitions of the frequently used terms. The control concept is then explained in more detail and applied in Chapter 6.

Basic concept and terms

Within the framework of Reinforcement Learning (RL) it is important to understand the common basic terms and definitions. The essential subelements of RL are:

- a controller, or agent
- a policy: the mapping from state $x$ to input $u$
- a reward function: this function gives an immediate return for each state-action couple
- value function: estimation of the accumulated reward
- a model of the environment (if available, not strictly necessary)

After each action that is taken by the agent, the reward function, returns a scalar value indicating the immediate performance. This indicates the quality of the previous action by the agent. The actions of the agent are based on the policy, which leads to a certain action to take in a certain state. The main goal of RL is to find the policy that maximizes the total accumulated reward/return. This makes the method of efficiently estimating the value function the core component of a reinforcement learning algorithm. This goal can only be achieved if the agent is able to predict the future rewards. This is done by a value function which is continuously updated during the lifetime of the agent [20].
The RL framework is suitable for problems modelled as a Markov Decision Process (MDP). This basically consists of four elements, \( X, U, f \) and \( \rho \), in which \( X \) denotes the state, \( U \) the (continuous) action space, \( f \) the model (state transition probability density function), and \( \rho \) the reward function. It is now assumed those elements do not change over time.

Reinforcement Learning basically comes down to mapping states to actions, by learning how to optimize a reward function. The most important aspects of this concept are trial-and-error searching and a (delayed) reward after each action [19]. The reward function is based on the current action \( u \) and current state \( x \). This is used by the controller to update the value function internally. Based on the value function (which in figure 2-3 depends on the state only) a new action is computed. The basic principle of RL is illustrated by the following scheme:

![Image of RL basic principle]

**Figure 2-3:** RL basic principle [21]

### Reinforcement Learning methods

In the context of RL, three main methods can be distinguished: [20]

- **Actor only** - This method uses a parametrized family of policies over which the optimization procedure is directly used. A big advantage is that continuous actions can be generated. However, the policy gradient method comes with high variance in the estimates of the gradient, which leads to slow learning.

- **Critic only** - This method is based on temporal difference learning, with a lower variance in the estimates of the expected returns. The learning is done by greedy actions. It uses a discretization of the continuous action space.

- **Actor-critic** - The combination of a parametrized actor with a critic. This makes continuous actions possible, without the need for optimization procedures on the value function. It is now also possible to have low-variance knowledge of the performance. The critic makes an estimate of the expected return, which makes quick learning possible.

The overall good properties of actor-critic, caused by the combination of the advantages of actor-only and critic-only methods, makes it a preferred method in the Reinforcement...
Learning (RL) framework. The actor-critic methods use policy gradients to update the value function (critic). In this method both the critic and the actor are separately parametrized. This method is further elaborated on and explained in detail in Chapter 6.
In this chapter the modelling of the particular 1D setup is explained. The models which are used in articles ([1], [2] and [3]) are discussed, after which a model (structure) is chosen. Based on this model several feedback controllers will be designed in Chapter 4.

**Introduction**

This thesis focuses on the control design for the magnetic manipulator setup. Before a controller can be designed, a model of the experimental setup is required which is as accurate as possible. Using this model the controller can be simulated before it is actually implemented on the setup. Furthermore, an accurate mathematical model enables the design of model-based controllers.

Deriving a model of a physical setup can be done in several ways. An analytical model can be devised based on the known physical properties of the magnet field and the dynamics of the ball. It will be shown, however, that this is not trivial for all geometries when electromagnets are involved. The formula of Biot-Savart is a good starting point to derive the magnitude of the magnetic force field close to the coils of the magnetic manipulator.

This section also covers alternative methods to find expressions of the magnetic field. Based on literature results, it is shown how the magnetic field can be modelled using a Finite Element Method (FEM). Furthermore, the resulting model of an empirical analysis from articles [1] [2] is discussed. The advantages and disadvantages of the existing methods are explained after which the model is chosen that will be implemented in this thesis.
3-1 Experimental setup

The experimental setup consists of four electromagnetic coils of 3000 windings each. A track is fixed above the coils, on which a steel ball \( (m = 0.032 \text{ kg}) \) is placed. The steel ball does not make direct contact with the four electromagnets, and can only move in one direction. The schematic of the setup is illustrated in 3-1 with all relevant dimensions.

The coils are controlled by a Pulse Width Modulation (PWM) module using two identical boards which are specifically designed for a similar setup at the CTU in Prague (see [1]). This board consist of the electronics for the current control of the coils, communication and an ARM processor. The communication with the PC is done through serial connection, using the RS232 protocol. Each sample period an ASCII string containing the two currents is sent from Matlab to the boards (as each board activates two coils). See appendix A for more in-depth information about the board, specifications and communication formats.

The position of the ball is measured by a laser sensor (Micro-Epsilon, Opto 1401) with a range of 200 mm and a sampling frequency of 1000 Hz, its dynamic resolution is 200 \( \mu \text{m} \), (static 40 \( \mu \text{m} \)). The position data is acquired using a data acquisition board (Humusoft MF624), which enables fast data sampling and processing to the Windows PC.
3-2 Modelling the magnetic field

The model of the setup consists of the ball dynamics, friction and the magnetic force induced by the four coils. Those aspects are discussed in this section. The magnetic force model is the most essential part of the total model.

In the first subsection the existing models for a magnetic field are briefly discussed. For the experimental setup in this thesis, the main challenge is to find the relation between the magnetic force and the distance of the steel ball to the coil axis (center). The actual derivation and model choice is done from subsection 3-2-2 on.

3-2-1 Models used in literature

For the particular setup in this thesis, the model used in [1] and [2] (in both researches a comparable setup is used) are most relevant. This empirical model is the only relevant model for the literature study which was performed in this thesis.

In [1] a controller is designed for a planar manipulation (horizontal, 2D) setup, for which the geometry of the ball in combination with the coils is comparable with the setup used in this thesis. Their model was experimentally identified as a one dimensional force profile; the force on the ball was measured as function of the distance from the coil center. This means it is directly applicable for this thesis. Based on force gauge measurements, the analytical expression was fitted as follows:

\[ F(x_c) = -\frac{\alpha x_c}{(x_c^2 + \beta)^3} \] (3-1)

\(x_c\) [m] is the distance from the ball to the center (axis) of the coil and F the force [N] exerted by the magnetic field.

Parameters \(\alpha\) and \(\beta\) were identified by step response measurements on the setup. The experiments and parameter fitting by [3] resulted in \(\alpha = 0.19\) and \(\beta = 0.33\).

In practice the parameters may differ for each coils as they may not be completely identical: different number of windings and a slightly different geometry will lead to inconsistent behaviour for each electromagnet. This means that for each coil both parameters has to be estimated.

Furthermore, an estimate of the friction \(b\) is needed. This depends on the track, ball surface and the mass of the ball.

Both parameters, \(\alpha\) and \(\beta\), have to be identified on the setup. The parameters depend on the number of windings, geometry and the distance between the coils and the track. It is most likely that those parameters will probably also differ for each coil in the setup. The parameters can be identified with a gauge meter and data fitting, or by applying system identification theory (which is also done in paper [3]). The latter method uses measured input-output data. This is further explained and applied in section 3-3-3.

Unmodelled effects in literature

In the available literature on a similar setup (see [1] and [2]) the mutual inductance of two inductors was not taken into account. In those articles it is not shown how significant this
Furthermore, the steel ball induces a (counter)current in the coils. This will influence the magnetic field created by the same coils \cite{2}. This disturbance depends on the distance of the ball from each coil, and on the velocity as the electromotive force is proportional to the speed of the object (which follows from Faraday’s law of induction).

The kick-back effect is not modelled in \cite{1} and \cite{2}. This effect is caused by the impedance of an inductor. It resists sudden changes of the current flowing through it. When an inductor is charged, this means that the electric energy is stored in the magnetic field, the inductor will resist any change in current. In the mechanical domain the inductance is equivalent to a mass with a certain inertia. An inductor is described by the relation $V = L \frac{di}{dt}$, with inductance $L$ [H], $V$ the voltage [V] and $i$ the current [A]. This shows that the current cannot be switched off instantly, as this would imply an infinite voltage \cite{22}. The inductor will have some time transient which is not modelled. This may reduce the overall control performance for model-based control approaches. This effect will be smaller, and potentially negligible, on the current setup, as a Pulse Width Modulation (PWM) module is used to control the current in the coils \cite{23}.

Furthermore, the third order model from equation (3-1) may also be inaccurate in practice. The two parameters may change with the temperature change in the coils. When current flows through the wires, some energy will be dissipated as heat. This influences the ‘resistance’ to current changes, which on his turn influences the behaviour of the inductor in time. This has a direct effect on the magnetic force produced by the coils.

### 3-2-2 Analytical model of magnetic force

The magnetic manipulator uses four coils, to create the magnetic field. Each coil consist of copper wire with current flowing through it. When a current flows through a wire, it creates a magnetic field. The electrical energy is then stored in the magnetic field. This is illustrated in figure 3-2. When the wires are placed in the shape of a straight tube (or helix), a coil is commonly referred to as a solenoid.

![Figure 3-2: Solenoid field lines \cite{24}](image)

Using the Biot-Savart law, the magnetic force can be derived as a function of distance to the coil. The Biot-Savart law relates the magnetic field to the electric current flowing through the coil: \cite{24}
3-2 Modelling the magnetic field

\[ B = \frac{\mu_0}{4\pi} \int \frac{Idl \times \vec{r}}{|\vec{r}|^2} \]  

(3-2)

With \( I \) the current, \( \mu_0 \) the magnetic constant, radius \( r \) and \( dl \) a vector with the length of the differential element of the wire.

The radius of the solenoids is constant. This means that the formula can be changed using polar coordinates. It is assumed the current is constant throughout the coil. This would be the case for an ideal conductor only, but this is considered to be a small effect. The magnetic force in the axis of the coil can be determined analytically. Outside the axis of the coil, however, this leads to an integral which is unsolvable algebraically (as shown in paper [25]). These results suggest that an alternative method is required to come up with an accurate model of the force created by the magnetic field.

3-2-3 Finite Element Method and Boundary Integral Methods

The Finite Element Method (FEM) is a good alternative for modelling the magnetic field. In the past, several software packages have been designed in order to do FEM analysis for different geometries. FEM is a numerical technique to find an approximate solution to the partial differential equations. The basic principle is to divide the whole problem into simpler parts, which are called finite elements. [25]

![Figure 3-3: Force model using BIM analysis][3]

Boundary Integral Methods consists of many partial differential equations which can be solved by analytical expressions. This is a much more efficient and faster method compared to FEM analysis. Both methods lead to comparable models as the empirical approach presented in the next section. With the Boundary Integral Methods (BIM) analysis in paper [3] a force model (figure 3-3) was obtained.
3-2-4 Empirical model derivation

In the research of [1] an empirical model of the system was derived. This was done by measuring the force on the ball above an array of coils, to identify the force profile corresponding to a certain distance of the ball from the coil centre (see figure 3-4). The position was measured by a resistive foil underneath the steel ball.

![Figure 3-4: Force gauge measurements on a planar magnetic setup [1]](image)

The measured data was used to identify the relation between force and distance to the centre of the coil, which led to the following one-dimensional analytical expression for one coil:

\[
F(x) = g(x_c)I = \frac{-\alpha x_c}{(x_c^2 + \beta)^3} I
\]

(3-3)

with \(x_c\) [m] equal to the distance from the ball to the center (axis) of the coil, \(\alpha\) and \(\beta\) two empirical parameters, \(I\) the current [A] and \(F\) the force [N] exerted by the magnetic field. This model can be visualized as follows:

![Figure 3-5: Force model visualization. The graph shows the empirical model which leads to the relation between force, current through a coil and the distance of the ball to the centre of the coil.](image)
3-3 State space model

In the section the complete model of the system is derived. The friction and forces are first explained in detail, after which the overall model is shown (in State-Dependent-Coefficients (SDC) form).

3-3-1 Friction

In general, several classical friction models have been developed. As described by [26], in the static models it is assumed the friction force opposes the motion and that the magnitude of this force does not depend on the velocity and/or contact area. Four classical models are:

![Friction Models](image)

**Figure 3-6**: Four different friction models. Figure a shows the Coulomb friction, figure b combines Coulomb friction with viscous friction. The friction model in c is identical to the one in b, although now stiction is also included. Figure d shows a more realistic behaviour for smaller velocities, as the friction force is continuously decreasing from the static friction level [26].

The friction in 3-6a, the force is modelled using Coulomb friction. This assumes the friction is constant for increasing velocity. In viscous friction models, which occurs when an object is moving in a fluid, there is a strong relation between the velocity of the object and the friction force. This is shown by figure 3-6b.

In the model of 3-6c, the stiction effect is also included. This describes the observation that at zero velocity, a larger force need to be applied before the object starts to move. Directly after this the kinetic friction force is lower. The discontinuous description of the stiction model can also be modelled by a continuous transition, which is called the Stribeck function (figure 3-6d) [26].

It is important to know whether it can be assumed if the steel ball rolls or slips. In case the ball rolls, the friction coefficient will be much lower. The model choice itself is not trivial,
and most friction terms introduce highly nonlinear terms in the model. It can be questioned whether a complicated and more accurate model is beneficial when it is not specifically incorporated in the control design. As this thesis focuses more on the input nonlinearities caused by the magnetic force field, it seems favourable to use a simpler (viscous) friction model. Note that viscous friction is a linear friction model. Furthermore, the rolling friction is in general lower than sliding friction, and the force is small compared to the magnetic force [3]. In section 3-3-3 it will be verified whether a viscous friction model sufficiently describes the dynamics (in combination with the magnetic force model). The viscous (or dynamic/kinetic) friction is then modelled as a force opposite to the velocity of the ball, with a magnitude of $b\dot{x}$. Parameter $b$ is the viscous/dynamic friction coefficient, $[\text{Ns/m}]$.

### 3-3-2 Magnetic force

As explained above, the third order magnetic model (equation 3-1) is chosen:

$$F(x) = g(x_c) I = \frac{-\alpha x_c}{(x_c^2 + \beta)^3} I$$

(3-4)

with $x_c$ [m] the distance from the ball to the center (axis) of the coil, $I$ the current [A] and $F$ the force [N] exerted by the magnetic field.

Both $\alpha$ and $\beta$ cannot be derived directly as they do not have any physical meaning. As already mentioned in section 3-2-1 system identification is needed to quantify the three parameters. This will be discussed in the next subsection.

### 3-3-3 System identification based on empirical model

Several methods are suitable to identify the parameters $\alpha$ and $\beta$, and the viscous friction term $b$. In this section a model-based system identification is used as the empirical model (see equation (3-3)) results in literature have shown that this model is sufficient to describe the relevant dynamics of the system.

The parameter estimation is done by optimizing the parameters of the model ($\alpha$, $\beta$ and viscous friction $b$), such that the least-squares error between the response on the setup and the prediction based on the model is minimized. This is a nonlinear optimization problem, solved by $\text{lsqnonlin}$ in Matlab. As explained in section 3-2-1, each coil is slightly different in practice. If each coil is modelled separately, a large optimization problem is obtained with 9 variables. The problem has many local minima, which is solved by a multi-start implementation of the $\text{lsqnonlin}$ algorithm.

It is not trivial to produce a step response on the setup with a sufficiently rich input signal, in terms of frequency content, to be used as input of the parameter estimation. This is partly caused by the fact that the system is unstable. Therefore a closed-loop experiment is done using a model based controller which uses an initial guess of the parameters. A stabilizing controller used which hopefully contains enough relevant frequencies compared to the final behavior. A Kalman filter is used to reduce noise on the position measurement and the velocity. This is done to reduce amplification of the noise in the feedback.

An experiment was done with four step responses with different set points, which are all
located at the axis of a coil. This means that for each coil the relevant dynamics are present, which consists of acceleration and deceleration of the ball. The performance of the resulting estimated parameters is evaluated using the Variance Accounted For (VAF). The definition in the book [27] is used to calculate the VAF value. The VAF value is by definition a percentage, the higher the value the better the prediction of the model. Figure 3-7 shows the measured response and the final predicted model after the multi-start optimization procedure.

![Least-squares non-linear optimization result. VAF = 98.91 %](image)

**Figure 3-7:** Least-squares nonlinear optimization result for all four coils. The dynamics are predicted reasonably well, the VAF value is 98.91%. The parameters found are: $\alpha_1 = 18.56$, $\beta_1 = 4.02$, $\alpha_2 = 5.20$, $\beta_2 = 1.42$, $\alpha_3 = 35.40$, $\beta_3 = 7.36$, $\alpha_4 = 10.89$, $\beta_4 = 5.22$ and the viscous friction coefficient: $b = 0.0776$.

The control input sequences are shown in figure 3-8. It shows that each coil is at least used once to accelerate and once to decelerate the ball.

Although this seems a very good result with a low prediction error, a validation needs to be done. By doing this it can be verified whether the estimated parameters actually describe the system dynamics well. The number of iterations of the *lsqnonlin* algorithm is limited to prevent overfitting. It is feasible to reach VAF-values up to 99.39%, but for those parameters the validation does not give satisfactory performance.

Master of Science Thesis  
J. Damsteeg
Validation

The resulting set of parameter from figure 3-7 are now validated with a new set input-output samples from the experimental setup.

The model parameters from figure 3-7 perform reasonable well on the validation set. It must be noted that the same reference values are used in the validation set, which contains comparable dynamics of the system.
On the setup this set of parameters did not result in a satisfactory performance. By selecting different parameters for each coil independently the risk of overfitting is still very high. It is also possible to assume all coils are the same and only find two parameters for all coils. This limits the total parameters to three, which reduces the optimization significantly. Furthermore, it limits the risk of overfitting the measured input-output data. This procedure gives a lower VAF values, but the prediction accuracy is much more consistent for different validation sets (with validation VAF values around 60 %). Furthermore, this simplified representation gives the best results for the model-based controllers on the experimental setup. The final model parameters (for each coil) are: $\alpha = 5.52$, $\beta = 1.75$ and the viscous friction coefficient: $b = 0.0161$.

In general it can be concluded that it is not trivial to come up with an accurate model of the magnetic field created by the coils. Although the empirical model seems to describe the relevant dynamics well, it is hard to retrieve an accurate overall model which also performs well for the validation data. The reason for this is twofold. First, the empirical model does not include all effects which are described in section 3-2-1. Secondly, it is hard to generate rich input-output (in terms frequency content) data on the experimental setup as the system has four inputs and is highly unstable. This confirms the expectation that a learning controller might be required to deal with the unmodelled effects. This is also mentioned and elaborated on in [2].

3-3-4 Continuous-time State space model

With the above considerations for modelling the magnetic force and friction, this leads to the following model of the complete system. The state vector consists of the position $x_1$ [m], and velocity $x_2$ [m/s]:

\[
x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x \\ \dot{x} \end{bmatrix}
\]

The dynamics of the ball, the magnetic force and the viscous friction together lead to the following two dynamic differential equations:

\[
\dot{x} = \begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} -\frac{b}{m} x_2 + \frac{1}{m} \sum_{i=1}^{4} g(x_1, i) u_i \\ \frac{x_2}{m} - \frac{\alpha}{m} (x_1 - 0.025 i) \end{bmatrix}
\]

With

\[
g(x_1, i) = \frac{-\alpha (x_1 - 0.025 i)}{((x_1 - 0.025 i)^2 + \beta)^3}
\]

And $u_i$ is $I$, which is the current [A] flowing through a coil, $i$ equal to the coil number ($i = 1 : 4$), $g(x_1, i)$ the nonlinear magnetic force equation (3-1), $m$ the mass [kg] of the steel ball and $b$ the viscous friction $[\text{Ns} \cdot \text{m}]$ of the ball.

This (general) nonlinear form can be written in State-Dependent-Coefficients (SDC) form
(where $x_1(t)$ is written as $x_1$ to improve the readability):

$$
\dot{x} = 
\begin{bmatrix}
\dot{x} \\
\dot{\dot{x}}
\end{bmatrix} = 
\begin{bmatrix}
0 & 1 \\
0 & -\frac{b}{m}
\end{bmatrix}
\begin{bmatrix}
x_1 \\
\dot{x}_1
\end{bmatrix} + 
\begin{bmatrix}
0 \\
\frac{g(x_1,1)}{m} \\
\frac{g(x_1,2)}{m} \\
\frac{g(x_1,3)}{m} \\
\frac{g(x_1,4)}{m}
\end{bmatrix} u
$$

(3-7)

With $x = \begin{bmatrix} x \\ \dot{x} \end{bmatrix}$ and input $u = \begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \end{bmatrix}$, the four currents through the coils.

The laser sensor measures the position of the ball, state $x_1$, which leads to the following relation for output $y$:

$$
y = \begin{bmatrix} 1 & 0 \end{bmatrix} x
$$

(3-8)

Note that for implementation on the setup a discretized model is required. This will be dealt with in Chapter 4.
Chapter 4

Control design

4-1 Introduction

This chapter focuses on the control design for the described setup, using the model which is derived in the previous chapter. First a benchmark controller is designed based on the principle of feedback linearization. This enables comparison of this thesis work to preliminary work available in literature [1], [2]. From there on, three nonlinear, non-adaptive controllers are designed. All four controllers are based on conventional control design techniques. A simulation study is performed to compare the performance of the controllers. It will also be decided whether, and which of, those controllers will be implemented on the experimental setup.

4-2 Discretization of continuous time model

For the implementation of model-based controllers a discrete-time model is required. The model (3-7) is discretized as:

\[
\begin{align*}
    x(k+1) &= A_d x(k) + B_d u(k) \\
    y(k) &= C_d x(k) + D_d u(k)
\end{align*}
\]

(4-1)

These state space matrix can be calculated using the exact discretization:

\[
A_d = e^{A_t} \quad \text{(4-2)}
\]

\[
B_d = \int_0^T e^{A_t} dt B \quad \text{(4-3)}
\]

With \( A \) the continuous-time system matrix. Note that matrix \( A_d \) in 4-1 is independent of state \( x \), while matrix \( B_d \) does depend on \( x(k) \). This means that \( A_d \) can be determined offline,
while matrix $B_d$ must be recomputed each sample time based on the updated continuous-time matrix $B(x)$. The matrix $B(x)$ into the discrete-time matrix $B_d(x)$ must be converted real-time. The Taylor expansion is used to enable fast computation of $B_d(x_k)$:

$$e^{At} = I + At + \frac{1}{2!}A^2t^2 + \frac{1}{3!}A^3t^3 + \cdots = \sum_{i=0}^{\infty} \frac{1}{i!}A^it^i$$

(4-4)

4-3 Observer design

As explained in section 3-1 the position is measured by a laser sensor. It is thus required to build an observer to estimate the velocity. An observer estimates the state, $\hat{x}(k+1)$ based on the model, previous estimate $\hat{x}(k)$ and (position) measurement $y(k)$:

$$\hat{x}(k+1) = A\hat{x}(k) + Bu(k) + K(y(k) - C\hat{x}(k) - Du(k))$$

(4-5)

The observer gain $K$ should be determined such that the error, the difference between the real state $x(k)$ and estimate $\hat{x}(k)$ is driven to zero as time $k$ goes to infinity. The gain matrix $K$ can be determined using pole placement or $K$ can be chosen equal to the Kalman gain. The latter method is a widely used for optimal filtering in presence of process and measurement noise. A linear Kalman filter is used in combination with the SDRE approach, which means the Kalman gain is redetermined each time step based on the linearized system at this sample. An alternative solution would be to use the extended Kalman filter [27], which is suitable for nonlinear systems.

The SDRE approach in combination with the Kalman gain is implemented in this thesis. As the system can be written in SDC form, it is slightly more favourable compared to the extended Kalman filter. The filter is tuned for smooth and fast tracking of the state. The Kalman filter is supplied with information about the process noise, $Q_n$, which corresponds to the model quality, and the measurement noise $R_n$, which indicates the quality of the measurement. The Kalman filter tuning parameters are:

$$Q_n = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad R_n = 1e-4$$

(4-6)

This gives the Kalman gain, $L = \begin{bmatrix} L_1 \\ L_2 \end{bmatrix}$.

The observer is tuned on a step response, with an initial offset of 0.01 m relative to the real initial condition, $x_0 = 0.025$. This is done to verify how fast the observer is able to converge to the true state. Noise is also added to the output $y$ to design the observer for a realistic scenario. The result is shown in figure 4-1.
Figure 4-1: Observer design in simulation. The SDRE Kalman approach has some difficulties to track the actual state initially, especially for the estimation of the velocity. The position estimate converges quickly, with some overshoot relative to the true state $x_1$. After 0.16 seconds the observer is able to estimate the true velocity as well. Both states are tracked asymptotically, sufficiently fast. The obtained Kalman gain is as small as possible in order to prevent amplification of noise.

4-4 Feedback Linearization control

The basic concept of Feedback Linearization (FL) is to algebraically transform a certain nonlinear system into linear equations. The linearization is partly done in the feedback signal itself, by using the inverse of the model, and/or by a state transformation. The nonlinearities are then cancelled in the closed-loop dynamics, which enables the use of linear control methods. FL is suitable for a large set of nonlinear models (depending on the ‘relative degree’, see [12]), and has some clear benefits compared to Lyapunov’s linearization method. In section 2-3-1 it is shown that a linear controller based on a linearized model is only valid close to the linearization point. Two main methods are distinguished in this FL context:

- Input-state linearization: a feedback control law and a change of variables is found which transforms an original system of the form: $\dot{x} = f(x) + g(x)u$ into $\dot{z} = Az + Bv$ using a change of variables $z = T(x)$ and a control law of the form $u = \alpha(x) + \beta(x)v$. 

Master of Science Thesis J. Damsteeg
A linear input-state relation is thus obtained. This does not necessarily mean, however, that the input-output relation is linear.

- **Input-output linearization:** This approach is similar to the previous approach, for both a state transformation and an adapted feedback law are used. The objective, however, differs as this method tries to obtain a linear input-output mapping. This may result in a (partly) nonlinear input-state relation.

This method can also be applied to the model which was explained in the previous chapter. The controller will be used as benchmark, to be able to compare the more advanced methods (which will be designed later on in this chapter) to the current standard in this research field.

\[
dx = \begin{bmatrix} dx_1 \\ dx_2 \end{bmatrix} = \begin{bmatrix} \frac{x_2}{-b/m x_2 + \frac{1}{m} \sum_{i=1}^{4} g(x_1, i) u_i} \end{bmatrix} \tag{4-7} \]

If the input is now chosen as:

\[u_i = \frac{m}{4g(x_1, i)} \left( b/m x_2 + v \right) \tag{4-8}\]

Then the second differential equation in equation 4-7 is now equal to:

\[
\ddot{x} = -\frac{b}{m} x_2 + \frac{1}{m} \sum_{i=1}^{4} g(x_1, i) u_i \\
= \frac{b}{m} x_2 + \frac{1}{m} \sum_{i=1}^{4} g(x_1, i) m \frac{b}{4g(x_1, i)} \left( -\frac{b}{m} x_2 + v \right) \\
= \frac{b}{m} x_2 + \left( \frac{b}{m} x_2 + v \right) = v \tag{4-10}\]

This results in the following linear system:

\[
\dot{x} = \begin{bmatrix} \dot{x} \\ \dot{x} \end{bmatrix} = \begin{bmatrix} x_2 \\ v \end{bmatrix} \tag{4-12}\]

To stabilize the system, state feedback can be applied by taking \(v = -Kx\), with \(K = \begin{bmatrix} k_1 \\ k_2 \end{bmatrix}\) leading to:

\[
\dot{x} = \begin{bmatrix} x_2 \\ v \end{bmatrix} = \begin{bmatrix} x_2 \\ -Kx \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -k_1 & -k_2 \end{bmatrix} x \tag{4-13}\]

Which basically means that the closed loop poles can be fully determined by choosing \(k_1\) and \(k_2\). Note that the main limitation here is that this only works as desired for a sufficiently accurate model. All modelling errors will not be cancelled by the feedback (including the inverse of \(g(x_1, i)\), for example), and will decrease the control performance.
Alternative method

Although the above methods are most commonly used, the research in article [1] developed a different feedback linearization technique. Based on the desired force on the ball, the current through the coil is computed based on a numerical optimization. This can be illustrated with the following scheme:

\[ \begin{align*}
\text{Controller} & \quad \rightarrow \quad F_{\text{req}} \\
\text{Linearization} & \quad \rightarrow \quad U_{m \times n} \\
\text{Coils (field)} & \quad \rightarrow \quad F \\
\text{Ball} & \quad \rightarrow \quad x
\end{align*} \]

Figure 4-2: Feedback Linearization approach using numerical optimization from [1]

The linearization block consists of an optimization procedure. This computes the control inputs corresponding to the required force, which was determined by the controller, by solving the following optimization problem:

\[
\begin{align*}
\text{minimize} & \quad \| x_I \| + c \| A_g x - b_f \| \\
\text{subject to} & \quad 0 \leq x \leq 1
\end{align*}
\]

In which \( x_I \) denotes the input vector with four currents, \( A_g \) the row vector with functions \( g(x,i) \) and \( b_f \) the required force. The first term limits the size of the control inputs, the second term is to generate a force \( A_g x_I \), which should become (approximately) equal to the required force, \( b \). The parameter \( c \) is included to make the force error negligible [2].

FL Control design

The Feedback Linearization controller with the control input from formula 4-8 is now evaluated in simulation. All controllers are simulated with an initial condition of \( x_0 = 0.05 \text{ m} \), which corresponds to the location of the fourth coil. The reference is the origin, which is the position of the second coil. The feedback linearized system is controlled by the feedback gain \( K \) (see 4-13) which is designed as a LQR controller. The weights of the quadratic cost function are:

\[
Q = \begin{bmatrix} 90 & 0 \\
0 & 1 \end{bmatrix} \quad R = \begin{bmatrix} 1e^{-4} & 0 & 0 & 0 \\
0 & 1e^{-4} & 0 & 0 \\
0 & 0 & 1e^{-4} & 0 \\
0 & 0 & 0 & 1e^{-4} \end{bmatrix}
\]

This leads to the following step response:
Figure 4-3: FL step response with an initial condition of $x_0 = 0.05$ m. This corresponds to the position of the fourth coil. In the figure above the ball is thus positioned from the fourth coil to the second coil ($x = 0$ m).

The step response is evaluated by analyzing the rise time, settling time and overshoot. The rise time is defined as the time between reaching 10% and 90% of the steady-state value. The settling time is defined as the time when the error becomes smaller than 2% of the peak value (in this case 0.050). The overshoot is the percentage overshoot relative to the final output $y_f$ (compared to the initial condition $x_0$). This definitions hold for all performance evaluations in this thesis. The step response of figure 4-3 has the following performance:

<table>
<thead>
<tr>
<th>FL performance</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Rise time [s]</td>
<td>0.24</td>
</tr>
<tr>
<td>Settling time [s]</td>
<td>0.43</td>
</tr>
<tr>
<td>Overshoot [%]</td>
<td>0.8</td>
</tr>
<tr>
<td>Control Effort [J]</td>
<td>15.7</td>
</tr>
</tbody>
</table>

Table 4-1: Performance of the FL controller

This is a relatively fast response, with very limited overshoot. This is not surprising, as in simulation there is no model mismatch between the model (inverse) used in the FL controller.
and the simulated system. This may be very different on the experimental setup as modelling errors will reduce the performance of the controller.

## 4-5 State Dependent Riccati Equation

In this section a (discrete-time) SDRE controller is designed, and verified in simulation. This control scheme was already introduced in 2-3-6. A discrete time version of the controller is now required. The discrete time model was obtained in section 4-2. The computation of the discrete system matrix $A$ can be done offline. The input matrix $B(x)$, however, must be determined online (at each sample time) as it depends on state $x(k)$. The online computation is done using the Taylor expansion approximation, see equation (4-4).

At each time step, the SDRE control method determines an optimal feedback gain for the following nonlinear system:

$$ x(k + 1) = A_d x(k) + B_d(x(k)) u(k), \quad x(0) = x_0 $$  \hspace{1cm} (4-15)

$$ y(k) = C_d x(k) + D_d u(k) $$  \hspace{1cm} (4-16)

The control objective is to find the control sequence $u_0, u_1, u_2, u_{N-1}$ such that the cost function $J$ is minimized. The function $J$ is defined as follows:

$$ J = \frac{1}{2} \sum_{k=0}^{\infty} (x^T Q(x)x + u^T R(x)u) $$  \hspace{1cm} (4-17)

With $x$ equal to the state at current time step $k$ ($x = x(k)$), $Q \leq 0$ and $R > 0$.

By following a procedure which is analogue to the continuous time derivation, the optimal feedback gain is determined by the solution of the Riccati equation, which is now represented in discrete time as the Discrete Time Algebraic Riccati equation (DARE):

$$ A_k^T P A_k - (A_k^T P B_k)(R_k + B_k^T P B)^{-1}(B_k^T P A) + Q_k = 0 $$  \hspace{1cm} (4-18)

This equation can be solved numerically using Matlab, after which the new control input is determined based on the solution $P_{k+1}$:

$$ u_k = -K x_k = -(R_k + B_k^T P_{k+1} B_k)^{-1} B_k^T P_{k+1} A_k x_k $$  \hspace{1cm} (4-19)

The state feedback gain $K$ is determined in Matlab with the function `dlqr`.

The the controller is tuned such that the settling time is as small as possible, with limited overshoot ($<2\%$). This leads to the following parameters for the quadratic cost function (see (4-17)):

$$ Q_a = \begin{bmatrix} 37 & 0 \\ 0 & 0.33 \end{bmatrix} \quad R_a = \begin{bmatrix} 1e-4 & 0 & 0 & 0 \\ 0 & 1e-4 & 0 & 0 \\ 0 & 0 & 1e-4 & 0 \\ 0 & 0 & 0 & 1e-4 \end{bmatrix} $$

This control method does not take the saturation of the coils into account. Negative inputs are also computed by the controller. This would would imply a repelling force, which is
physically not feasible. Therefore the control input is clipped, both on the lower bound (0 A) and upper bound (0.55 A).

The controller is simulated with an initial condition of $x_0 = 0.050 \text{ m}$, which is the position of the fourth coil, and the reference is the second coil. This leads to the following step response:

![Simulated step response, SDRE. Position:](image)

**Figure 4-4:** Simulated step response of the SDRE controller, which was tuned for minimal settling time

With the following performance:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SDRE performance</strong></td>
<td></td>
</tr>
<tr>
<td>Rise time [s]</td>
<td>0.23</td>
</tr>
<tr>
<td>Settling time [s]</td>
<td>0.44</td>
</tr>
<tr>
<td>Overshoot [%]</td>
<td>1.5</td>
</tr>
<tr>
<td>Control Effort [J]</td>
<td>5.7</td>
</tr>
</tbody>
</table>

**Table 4-2:** Performance of the SDRE controller

The settling time and the rise time are defined as in the previous section 4-4.

In figure 4-5 it can be seen the unsaturated control actions are much larger than 0.55 A, and that negative inputs are computed as well. During the first 0.2 seconds the fourth coil, $u_4$
saturates at 0 A. During this period it will increase the second part of the quadratic cost function, although this control input does not have any influence on the system. This is the main reason that this control method is not optimal w.r.t. to the cost function 4-17.

4-6 Constrained State Dependent Riccati Equation control - MPC approach

4-6-1 Introduction and motivation

In the previous section a SDRE controller was designed. This nonlinear control technique provides some advantages over FL and direct linearization methods. However, SDRE has two main disadvantages:

- incorrect infinite horizon prediction with a constant model
- input-constraints are not taken into account

The first disadvantage is caused by the infinite horizon prediction which is based on the linearized model at the current time step $k$. This will lead to a suboptimal solution with respect to cost function 4-17. This problem can be dealt with when the prediction takes into account that the model changes during the future state evolution. This will be further discussed in section 4-7, in which a nonlinear predictive controller is designed.

The second disadvantage also limits the control performance, mainly regarding the lower bound constraint. As discussed earlier (see 2-2-1), the coils can only attract the steel ball. A negative control input leads to a comparable magnetic field as a positive input. Furthermore, the upper saturation bound is 0.55 A which is required in order to prevent the coils from overheating. This can be expressed as both a lower and upper bound constraint for each input $u_i$: $0 \leq u_i \leq 0.55$. If this is not taken into account, the computed control inputs are
clipped, which is done for the feedback linearization and SDRE approach. This, however, results in suboptimal control as the infinite horizon (LQR) optimization was done with the unsaturated inputs (in the second term of the cost function, $u^T R u$, the input is clearly not always equal to the saturated input).

Two feasible methods to deal with the first disadvantage of the SDRE approach are:

- Assisting Saturated (AS) control based on LQR [28]
- Model Predictive Control approach (also referred to as Constrained-SDRE by [17])

The first method, assisting saturated control, is beneficial when only a part of the control inputs are outside the bounds. The unsaturated inputs are used to compensate ('assist') the saturated ones [28]. This can be applied to the SDRE controller by increasing the attractive force of a certain coil to compensate for a repelling force (negative input, computed by the controller) by the neighboring coil, which is not feasible in practice.

The second method uses the well-known MPC framework to optimize a finite horizon cost function (equation 4-20) subject to input and/or output constraints.

$$J = \frac{1}{2} \sum_{j=0}^{N_p} \left( x_j^T Q_j x_j + u_j^T R_j u_j \right)$$  \hspace{1cm} (4-20)

With $N_p$ equal to the prediction horizon, relative time step $j$ (w.r.t. to current time step). The optimization is performed with a rolling horizon approach. The optimization vector $\bar{u}$ consists of the optimal control sequence which leads to the optimal state trajectory. The first optimal control action is chosen at time step $k$, which means that $u_k = \bar{u}(0)$. The horizon is now shifted one step, and the procedure is repeated at time step $k + 1$.

The second method chosen based on the research by [17]. The MPC framework is widely used, which means that sufficient knowledge and several algorithms are available to implement this algorithm. From a control perspective, the MPC approach is favourable as it takes the saturation into account in the predicted future states (horizon), and makes an optimal decision for combining the several inputs to generate the required force on the ball. The main limitation and challenge for this approach is to make it feasible in real-time on the experimental setup.
4-6-2 MPC problem formulation

Derivation of the problem statement

The goal of this method is to find the optimal sequence \( u^*(0), u^*(1), \ldots, u^*(N_p) \) which minimizes the cost function [29]:

\[
J(x(0), U) = \sum_{j=0}^{N_p} \left( x(j)^T Q x(j) + u(j)^T R u(j) \right)
\] (4-21)

With \( U = \bar{u}(0) = [u(0)^T \ u(1)^T \ \ldots \ u(N_p - 1)^T] \), the cost function can be rewritten to [17]:

\[
J(x(0), U) = x(0)^T Q x(0) + \begin{bmatrix}
\bar{Q}
\end{bmatrix}
\begin{bmatrix}
Q & 0 & \ldots & 0 \\
0 & Q & \ldots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \ldots & Q
\end{bmatrix}
\begin{bmatrix}
x(1) \\
x(2) \\
\vdots \\
x(N_p)
\end{bmatrix}
+ u(0)^T R u(1) + \begin{bmatrix}
\bar{R}
\end{bmatrix}
\begin{bmatrix}
0 & R & \ldots & 0 \\
0 & 0 & \ldots & R \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \ldots & 0
\end{bmatrix}
\begin{bmatrix}
u(0) \\
u(1) \\
\vdots \\
u(N_p - 1)
\end{bmatrix}
\] (4-22)

With the state vector being equal to:

\[
\begin{bmatrix}
x(1) \\
x(2) \\
\vdots \\
x(N_p)
\end{bmatrix}
= \begin{bmatrix}
B & 0 & \ldots & 0 \\
AB & B & \ldots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
A^{N_p-1}B & A^{N_p-2}B & \ldots & B
\end{bmatrix}
\begin{bmatrix}
\bar{S}
\end{bmatrix}
\begin{bmatrix}
u(0) \\
u(1) \\
\vdots \\
u(N_p - 1)
\end{bmatrix}
\] (4-23)

The cost function can be rewritten to:

\[
J(x(0), U) = x(0)^T Q x(0) + \left( \bar{S} U + \bar{T} x(0) \right)^T \bar{Q} \left( \bar{S} U + \bar{T} x(0) \right) + U^T \bar{R} U
= \frac{1}{2} U^T \begin{bmatrix}
R + S^T Q S & x(0)^T 2 \bar{T} T Q S U + \frac{1}{2} x(0)^T \bar{Q} + \bar{T} T Q \bar{T} x(0)
\end{bmatrix} U
\] (4-24)

The third term, \( \frac{1}{2} x(0)^T Y x(0) \), does not depend on the optimization variable \( u(k) \). The remaining terms form the basis for the Quadratic Programming (QP) problem formulation:

\[
\min_{\bar{u}} \frac{1}{2} \bar{u}^T H \bar{u} + c^T \bar{u} \quad \text{subject to } A \bar{u} \leq b
\] (4-25)

With \( H \) and \( c \) the matrices from equation (4-24), \( \bar{u} \) the optimization variable and \( A \) and \( b \) inequality constraints.

This problem can be solved in Matlab using `quadprog`, with the trust-region-reflective algorithm. In the next section it will be shown that it is feasible to compute the QP problem.

Master of Science Thesis

J. Damsteeg
real-time in Matlab, with the right selection of the termination tolerance and maximum number of iterations.

**Combining SDRE and linear MPC**

Note that in the previous section the MPC is derived for a linear system. This is not directly applicable to the nonlinear model. This is dealt with by combining the concept of State Dependent Riccati Equation (SDRE) and the derived linear MPC concept. This method uses the SDC structure of the model (see equation 3-7). This basically means that the two matrices, $H$ and $c$, which determine the Quadratic Programming (QP) problem are state-dependent, as they both depend on input-matrix $B(x(k))$.

The main challenge is to compute matrices $H$ and $c$ real-time. Each time step those matrices must be recalculated, which is computationally extensive. Even for an efficient calculation it is not feasible to compute those matrices online. This is solved by calculating them off-line, and store them into a lookup-table. This leads to a one-dimensional table, as the input matrix $B(x)$ depends on the position $x_1$ only.

This method specifically takes the constraints into account, but is still limited as the predictions are done with the constant model at the position $x_1(k)$. This will be further explained in section 4-7.

**Simulation results**

The algorithm is now evaluated in simulation, with the same initial conditions and reference value as in the previous simulations. The control parameters were tuned for satisfactory performance in terms of rise time, overshoot and settling time.

$$N_p = 15 \quad Q = \begin{bmatrix} 45 & 0 \\ 0 & 0.24 \end{bmatrix} \quad R = \begin{bmatrix} 0.002 & 0 & 0 & 0 \\ 0 & 0.002 & 0 & 0 \\ 0 & 0 & 0.002 & 0 \\ 0 & 0 & 0 & 0.002 \end{bmatrix}$$
The step response is as follows:

![Step response, MPC. Position:](image1)

![Velocity:](image2)

![Control inputs](image3)

**Figure 4-6:** Simulated step response of the Constrained-SDRE control [17] which applies the MPC concept to take the lower and upper bound constraints into account. During each time step the model is linearized, based on which the optimal control sequence is determined for the next \( N_p \) time steps, the control horizon.

With the following performance:

<table>
<thead>
<tr>
<th>C-SDRE performance</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Rise time [s]</td>
<td>0.23</td>
</tr>
<tr>
<td>Settling time [s]</td>
<td>0.41</td>
</tr>
<tr>
<td>Overshoot [%]</td>
<td>1.9</td>
</tr>
<tr>
<td>Control Effort [J]</td>
<td>8.0</td>
</tr>
</tbody>
</table>

**Table 4-3:** Simulation performance of C-SDRE (MPC) controller

This C-SDRE controller outperforms both the FL and SDRE controller, in terms of rise time and settling time. Furthermore, the size of the required inputs are lower then the previous two controllers.
4-7 Nonlinear Model Predictive Controller

The Constrained-SDRE improves the SDRE significantly by taking the constraints into account in the rolling horizon optimization. However, as also mentioned in the previous section, the prediction is still based on a ‘static’ model at the current time step. This limits the prediction accuracy, as the force exerted by each coil may change significantly in only a few time steps (see 3-5). To predict the future state trajectory accurately while also taking the constraints into account, a Nonlinear Model Predictive Controller is designed in this section. This method minimizes the following general cost function:

$$\text{minimize } J_N(x_0, u(\cdot)) = \sum_{k=0}^{N_p-1} l(x(k, x_0), u(k))$$  \hspace{1cm} (4-26)

with optimization vector \([u_0, u_1, \ldots, u(N_p - 1)]\), and prediction model:

$$x(k + 1, x_0) = f(x(k, x_0), u(k))$$ \hspace{1cm} (4-27)

The optimization is done each time step, based on the new observed state, for the length of the prediction horizon. The first input \(u^*(0)\) of the resulting vector is applied to the system, and the horizon is shifted one time step. It is thus equivalent to the linear MPC problem with the use of a receding horizon approach, although now a much harder nonlinear optimization needs to be performed each time step. This is computationally intensive and is the main reason why this method is only applied in relatively slow processes, with large sample rates (such as the process industry, where reactors are controlled with sample periods in the order of magnitude of minutes [30]).

This method is implemented for the model of the magnetic manipulator. Although it might not be feasible for real-time application on the setup, it will give a near-optimal theoretical solution for the regulation problem.

The method is implemented using the NMPC algorithm by Lars Grüne [31]. This Matlab code applies the rolling horizon approach for the NMPC problem with the use of \texttt{fmincon}.

A quadratic cost function is used, with the following tuning parameters:

$$N_p = 10 \quad Q = \begin{bmatrix} 100 & 0 \\ 0 & 0.22 \end{bmatrix} \quad R = \begin{bmatrix} 0.001 & 0 & 0 & 0 \\ 0 & 0.001 & 0 & 0 \\ 0 & 0 & 0.001 & 0 \\ 0 & 0 & 0 & 0.001 \end{bmatrix}$$

This leads to the following simulated step response:

J. Damsteeg

Master of Science Thesis
With the following performance:

<table>
<thead>
<tr>
<th>Controller:</th>
<th>NMPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rise time [s]</td>
<td>0.20</td>
</tr>
<tr>
<td>Settling time [s]</td>
<td>0.41</td>
</tr>
<tr>
<td>Overshoot [%]</td>
<td>1.8</td>
</tr>
<tr>
<td>Control Effort [J]</td>
<td>9.0</td>
</tr>
</tbody>
</table>

Although this performance will not be reachable on the setup, it does give an interesting impression of the near-optimal solution. In the simulation above the ball is positioned from the initial position, coil 4, to the second coil ($x_1 = 0$). The controller uses both coil two and three to accelerate the ball, and then decelerates the ball with a saturated input on the third coil. This means that at almost all time steps the largest possible force is exerted on the ball, both during acceleration and deceleration. The control effort is slightly larger than the SDRE and C-SDRE controllers.

This theoretically (near) optimal behaviour is useful to compare with the actual performance on the experimental setup.
4-8 Conclusion

In this section four controllers were designed. The first three controllers are real-time feasible and will also be implemented on the experimental setup. The Nonlinear Model Predictive Controller is not real time feasible on the setup, but indicates the desired behaviour of the four coils for fast regulation. The performance of the four controllers can be seen in table 4-5.

<table>
<thead>
<tr>
<th>Controller:</th>
<th>FL</th>
<th>SDRE</th>
<th>C-SDRE</th>
<th>NMPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rise time [s]</td>
<td>0.24</td>
<td>0.23</td>
<td>0.23</td>
<td>0.20</td>
</tr>
<tr>
<td>Settling time [s]</td>
<td>0.43</td>
<td>0.44</td>
<td>0.41</td>
<td>0.41</td>
</tr>
<tr>
<td>Overshoot [%]</td>
<td>0.8</td>
<td>1.5</td>
<td>1.9</td>
<td>1.8</td>
</tr>
<tr>
<td>Control Effort [J]</td>
<td>15.7</td>
<td>5.7</td>
<td>8.0</td>
<td>9.0</td>
</tr>
</tbody>
</table>

Table 4-5: Simulation performance of all four controllers

The Feedback Linearization controller performs reasonable well with fast regulation in combination with the lowest overshoot. The control effort is very large though, almost twice as large as the C-SDRE controller. The SDRE controller is slightly slower although it takes the least amount of control effort to fulfil the regulation task. The C-SDRE has the best overall performance, which is comparable to the nonlinear MPC.

The first three controllers will be implemented and evaluated on the experimental setup in the next chapter.
Chapter 5

Evaluation on experimental setup

In this chapter the three model-based controllers, FL, SDRE and C-SDRE, are implemented on the experimental setup.

The nonlinear MPC is not implemented on the setup. The simulation study showed that this method is not real-time feasible.

The three controller are evaluated for a step response with the same initial condition \( (x_0 = 0.05 \text{ m}) \) and reference position (0 m) as the simulation study of the previous chapter.

5-1 Selection of the sampling frequency

The determination of the sample time is a fundamental problem in signal processing and many control applications. For closed-loop control systems it is important to analyze the highest frequency of interest in combination with the bandwidth of the closed loop system. The bandwidth is directly related to the rise time of the controlled system. For a first order system, a commonly used rule of thumb is to retrieve 4 to 10 samples per rise time. For second order closed-loop systems, however, it is more reasonable to have at least 20-25 samples per rise time [26]. A simple experiment shows that the fastest achievable rise time is approximately 0.3 seconds. This is also visible in figure 5-1, in which both coil two and three saturate to accelerate the ball. In order to have 25 samples during this rise time, a sampling period of approximately 0.012 s is required. This is rounded to a sample period of 0.01 s. It will be evaluated in practice whether this is fast enough to capture all dynamics while preventing aliasing as well. It will also be considered whether a higher sampling rate is computationally feasible and if this will improve the control performance.

5-2 Feedback Linearization control

The Feedback Linearization controller is implemented on the experimental setup. The initial control parameters are used for the first trial, and from there the controller is tuned to achieve
Evaluation on experimental setup

satisfactory results. Compared to the control parameters from simulation, the penalty on the position error is decreased to reduce the overshoot.

If this penalty is too low, however, the controller tries to repel the ball with a negative input on the closest coil. This negative input is not sent to the coils as it is physically impossible to repel the steel ball. A larger penalty on $x_1$ is required to obtain a positive input on the other coil which should attract the ball. This is a trade-off between overshoot and speed of the response.

Another observation is that retuning is required for different reference values in order to achieve the desired behavior. In other words, the controller does not have a consistent performance. This may be caused by the use of the model inverse; the performance of the controller thereby depends on the model (in)accuracy. The model inaccuracy partly depends on the current state of the system. This is not unlikely with the highly nonlinear magnetic force field (3-3) in combination with unmodelled effects such as the induced current by the movement of the steel ball. For the experiment of figure 5-1, with a sampling frequency of 100 Hz, the following tuning parameters were selected:

$$Q = \begin{bmatrix} 33 & 0 \\ 0 & 1 \end{bmatrix}, \quad R = \begin{bmatrix} 1e^{-4} & 0 & 0 & 0 \\ 0 & 1e^{-4} & 0 & 0 \\ 0 & 0 & 1e^{-4} & 0 \\ 0 & 0 & 0 & 1e^{-4} \end{bmatrix}$$

The resulting response on the experimental setup is as follows:

![Figure 5-1: Step response of FL on experimental setup](image)

J. Damsteeg  Master of Science Thesis
The controller performance is:

<table>
<thead>
<tr>
<th>Controller:</th>
<th>FL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rise time [s]</td>
<td>0.31</td>
</tr>
<tr>
<td>Settling time [s]</td>
<td>0.64</td>
</tr>
<tr>
<td>Overshoot [%]</td>
<td>0.51</td>
</tr>
<tr>
<td>Control Effort [J]</td>
<td>21.4</td>
</tr>
</tbody>
</table>

Table 5-1: Performance of FL on the experimental setup

This controller has a reasonable performance in terms of settling time and overshoot. One of the disadvantages, which was also mentioned earlier, is that a large control effort is required to achieve this response (which is confirmed in article [16]). The controller does not make a trade-off between the size of each input and the actual influence of this decision on the force exerted on the ball. This is the main argument for the alternative approach in article [1] which was explained in section 4-4. In article [1] an optimization is included in the loop to make an optimal decision regarding the required force and the size of the control inputs.

5-3 State Dependent Riccati Equation control

The SDRE controller is also evaluated on the experimental setup. All computations are done online, as Matlab is able to solve the Riccati equation sufficiently fast. It takes some tuning effort to get a satisfactory performance with this controller. It is hard to prevent overshoot while having a fast response as well. A fast response is obtained with the parameters from the simulation study, although the overshoot is large (> 20%). The penalty on the position $x_1$ is reduced to obtain a more smooth behaviour. A trade-off between speed and overshoot must be made, which eventually results in the following tuning parameters:

$$Q = \begin{bmatrix} 15 & 0 \\ 0 & 0.8 \end{bmatrix} \quad R = \begin{bmatrix} 0.001 & 0 & 0 & 0 \\ 0 & 0.001 & 0 & 0 \\ 0 & 0 & 0.001 & 0 \\ 0 & 0 & 0 & 0.001 \end{bmatrix}$$

The results of the SDRE controller on the experimental setup can be seen in figure 5-2:
Figure 5-2: Step response and control inputs of SDRE controller on experimental setup

With the following performance:

<table>
<thead>
<tr>
<th>Controller:</th>
<th>SDRE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rise time [s]</td>
<td>0.36</td>
</tr>
<tr>
<td>Settling time [s]</td>
<td>0.80</td>
</tr>
<tr>
<td>Overshoot [%]</td>
<td>5.2</td>
</tr>
<tr>
<td>Control Effort [J]</td>
<td>7.2</td>
</tr>
</tbody>
</table>

Table 5-2: Control performance of SDRE on experimental setup

This performance is clearly worse than the FL control results in the previous section. The large difference in performance is somewhat surprising as both controllers ignore the lower and upper bound constraints. Furthermore, it is expected that modeling errors will have a larger influence on the FL controller.

These results clearly differ from the simulation and experimental results in [13]. In that article the SDRE and FL are compared on a magnetic levitation system. It is shown that the SDRE outperforms the FL. In the experiment in [13], however, the constraints have less influence and the experiment consists of only one actuator (a vertical coil, under which the ball is positioned vertically).
One of the reasons that the FL performs better than the SDRE might be that the SDRE is suboptimal w.r.t the quadratic cost function as the negative input constraints are not taken into account. The SDRE controller therefore decides to accelerate the ball by a repelling force, which (in some cases) theoretically requires a smaller input than an attracting force by the other coil, although this is not physically possible. As a result the system is much slower, with a small control effort. It is not feasible to tune the controller into a more aggressive response by increasing the penalty on $x_1$ without a larger overshoot of the system. It can be concluded that a suboptimal SDRE controller is outperformed by the FL controller, except when a more efficient system with a low control effort has the highest priority.

5-4 Constrained State Dependent Riccati Equation control - MPC approach

In this section the Constrained State Dependent Riccati Equation (C-SDRE) method is implemented on the experimental setup. The computation of matrices $H$ and $c$ of equation 4-25 each time step and solving the QP problem are computationally extensive. This makes it challenging for this algorithm to work in real-time. As described in subsection 4-6-2 this can be solved by computing matrices $H$ and $c$ offline and storing them in a lookup-table. The Quadratic Programming problem, however, is solved online as it depends on both states. The QP is supplied with the previous solution vector, $\bar{u}(k)$. From there only a few iterations are required to come up with a satisfactory new optimum. The first QP solution takes the largest computation time. Therefore the initial input vector $\bar{u}(1)$ is determined offline, and is used during the first time step. From the second time step on, the QP solver can find the solution in time (i.e. without exceeding the sample period during each time step).

The parameters from the simulation in section 4-6-2 resulted in a slightly too aggressive behaviour on the setup, with large overshoot. Therefore the penalties on both the velocity and the control inputs were increased to achieve a more desired smooth behaviour. This led to the following tuning parameters:

$$ Q = \begin{bmatrix} 45 & 0 \\ 0 & 0.4 \end{bmatrix} \quad R = \begin{bmatrix} 0.006 & 0 & 0 & 0 \\ 0 & 0.006 & 0 & 0 \\ 0 & 0 & 0.006 & 0 \\ 0 & 0 & 0 & 0.006 \end{bmatrix} $$

With the following response on the setup:
Notice that the C-SDRE uses the third coil to accelerate, with a negligible contribution of the second coil. The controller is already decelerating the ball before the second coil can exert a significant attracting force on the ball.

As this controller takes the constraints into account while also optimizing the quadratic cost function, a balanced trade-off can be made between the control input and the resulting behaviour on the setup. This is visible in the graph above.

The step response has the following performance:

<table>
<thead>
<tr>
<th>Controller:</th>
<th>C-SDRE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rise time [s]</td>
<td>0.30</td>
</tr>
<tr>
<td>Settling time [s]</td>
<td>0.57</td>
</tr>
<tr>
<td>Overshoot [%]</td>
<td>0</td>
</tr>
<tr>
<td>Control Effort [J]</td>
<td>6.9</td>
</tr>
</tbody>
</table>

The table above shows that the C-SDRE controller performs best on all four measures. It can be concluded that the C-SDRE is the fastest, most accurate and efficient controller. Another benefit is that a wider range of tuning parameters is allowed, while still having a satisfactory performance. Note that this was much harder for the FL and SDRE controllers, as the tuning
parameters were very sensitive. For the C-SDRE, for example, the weight on the control inputs can be made much smaller (up to $R = \text{diag}(0.0005)$), resulting in a more aggressive response. The overshoot is then still within the bound of 2%, although it does take larger control inputs.

## 5-5 Conclusions

A final comparison between all three controllers can now be made.

<table>
<thead>
<tr>
<th>Controller:</th>
<th>FL</th>
<th>SDRE</th>
<th>C-SDRE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rise time [s]</td>
<td>0.31</td>
<td>0.36</td>
<td>0.30</td>
</tr>
<tr>
<td>Settling time [s]</td>
<td>0.64</td>
<td>0.80</td>
<td>0.57</td>
</tr>
<tr>
<td>Overshoot [%]</td>
<td>0.5</td>
<td>5.2</td>
<td>0</td>
</tr>
<tr>
<td>Control Effort [J]</td>
<td>21.4</td>
<td>7.2</td>
<td>6.9</td>
</tr>
</tbody>
</table>

**Table 5-4:** Performance of all three controllers on the experimental setup

Compared to the simulation results all controllers are slower on the experimental setup. The SDRE has the largest performance drop, and does not meet the performance requirement on the overshoot (< 2%). The FL performs surprisingly well compared to the other controllers on the setup, although the settling time is 0.21 seconds larger compared to the simulation. Furthermore, it can be concluded that the FL controller is not reliable as it requires retuning of the parameters for different reference/initial values. Although the C-SDRE is also significantly slower than in simulation, the C-SDRE outperforms the other two controllers on each performance measure. The overshoot is now even limited to 0%, which is better than in in simulation (1.9%).

It is interesting to see whether this controller can be improved by a self-learning controller, which is designed in Chapter 6.
Chapter 6

Learning control

6-1 Introduction

As explained in Chapter 2, the Reinforcement Learning framework learns a policy based on interaction with the environment. In this chapter an Actor-Critic RL controller is designed. This learning method should be able to deal with the nonlinearities which are present in the magnetic manipulator setup, without explicit use of the model during the learning process. The actor-critic method with the continuous state and action space.

In subsection 6-3 a second learning method, imitation learning, is designed. This is a supervised learning controller, which estimates the inverse model by interaction with the environment. The generated data is used to determine the required input to steer the system along the desired reference trajectory.

Both learning controllers will be implemented and evaluated on the experimental setup.

6-2 Actor-Critic Reinforcement Learning

6-2-1 Introduction

As explained earlier in section 2-4-2, three RL types can be distinguished: actor-only, critic-only and actor-critic. The actor-only method deals with the discrete actions in the RL framework by parametrizing the policy. The use of policy gradient method however, leads to slow learning as it contains high variance [20].

The critic-only method uses temporal difference learning, which contains a much lower variance leading to faster learning. The policy is derived by greedy actions, which means the agent is exploiting current knowledge of the values of the actions. To do this an optimization in each state-space is required to find the action leading to highest (estimated) value. This is
mostly not feasible for a continuous action space. Although it is possible to use a discretization of the action space, this will not lead to the true optimum. The third presented method, actor-critic, combines the advantages of both methods. This will be shown in this section, after which the knowledge is applied to design an actual actor-critic RL controller for the magnetic manipulator.

6-2-2 Reinforcement Learning

The agent in the RL framework is a state feedback controller. The control law is also called the policy, and is denoted by \( u = \pi(x) \). The goal of the agent is to find the policy that maximizes a certain goal, which is defined as a reward over a longer period of time. The immediate reward at a certain state-action pair, is given by a reward function as function of the state-action pair at the current time step: \( r_{k+1} = \rho(x_k, u_k) \).

The reward function is a key ingredient of a learning algorithm, as this indicates the desired behaviour of the agent. The goal of the agent is to find a policy \( \pi(x) \) to maximize the long term cumulative reward, also called the discounted return. The cumulative reward is estimated by the value function:

\[
V^\pi(x_0) = R^\pi(x_0) = \sum_{k=0}^{\infty} \gamma^k \rho(x_k, \pi(x_k))
\]

In which the discount factor, \( \gamma \in [0,1) \), induces a 'pseudo-horizon' for the optimization, bounds the infinite sum \(^1\) and incorporates the uncertainty in the future.

The equation above can be rewritten to:

\[
V^\pi(x_0) = \rho(x_0, \pi(x_0)) + \gamma \rho(x_1, \pi(x_1)) + \gamma^2 \rho(x_2, \pi(x_2)) + \cdots
\]

This leads to the Bellman equation [19] for \( V^\pi \), at state \( x_k \):

\[
V^\pi(x_k) = \rho(x_k, \pi(x_k)) + \gamma V^\pi(x_{k+1})
\]

By using the state transition function \( x_{k+1} = f(x_k, u_k) \), the Bellman equation can be rewritten to:

\[
V^\pi(x_k) = \rho(x_k, \pi(x_k)) + \gamma V^\pi(f(x_k, \pi(x_k)))
\]

The main goal of the algorithm is now to find the optimal V-function \( V^* = \max_\pi V^\pi \), which is achieved if the Bellman equation holds. The informal way to express this was stated by [19]: ’Intuitively, the Bellman optimality equation expresses the fact that the value of a state under an optimal policy must equal the expected return for the best action from that state’.

One way to find the optimal value function \( V^* \) is by using the iterative temporal difference (TD) method. The temporal difference is defined as the difference between the left hand and right hand side of the Bellman equation (6-3) during the current \( (k\text{th}) \) iteration. The objective is to find the near-optimal policy as the temporal difference converges to (approximately) zero.

[^1]: This and also the previous statement are true as it can be proved that the infinite sum of \( \gamma^n \) is finite for \( \gamma \in [0,1) \)

J. Damsteeg Master of Science Thesis
The temporal difference is denoted by $\Delta$:

$$
\Delta = \rho(x, \pi(x)) + \gamma V^\pi(f(x, \pi(x))) - V^\pi(x)
$$

(6-4)

This value can then be used to update the value function each iteration:

$$
V^\pi(x_k) \leftarrow (1 - \alpha_k) V^\pi(x_k) + \alpha_k [r_{k+1} + \gamma V^\pi(x_{k+1})]
$$

(6-5)

In which $\alpha_k$ is the learning rate [-]. This principle of using the temporal difference to update the value function is used in the actor critic method. This will be explained in more detail in the next subsection.

### 6-2-3 Actor-critic method

The actor critic control method uses both an actor and a critic, which are are parametrized to deal with the continuous action and state space. It can be illustrated with the following control scheme:

![Figure 6-1: Actor-critic working principle](image)

Note that figure 6-1 is consistent with the basic RL scheme from figure 2-3. The controller consists of a critic and an actor.

The actor consists of the policy $\pi(x)$ which enables the agent to choose its actions based on the current state. The policy is continuously updated by the critic. This consists of a state value function, which estimates the future return based on the current state and (scalar) reward.

### 6-2-4 Approximating critic and actor by Radial Basis Functions

In practice, the action and state space are in fact continuous. In the actor-critic framework this is dealt with by parametrizing the actor and the critic in $\varphi$ and $\theta$, respectively.

For the actor this leads to:

$$
\hat{\pi}(x, \varphi) = \varphi^T \hat{\phi}(x)
$$

(6-6)

And the critic:

$$
\hat{V}(x, \theta) = \theta^T \hat{\phi}(x)
$$

(6-7)
where $\hat{\phi}(x)$ represents a normalized basis function. When a Radial Basis Function network is chosen to approximate $\pi$ and $V$, this results in the following expression for $\hat{\phi}(x)$:

$$\hat{\phi}_i(x) = \frac{\phi_i(x)}{\sum_j \phi_j(x)} \quad (6-8)$$

$$\phi_i(x) = e^{-\frac{1}{2}(x-c_i)^T B^{-1}(x-c_i)} \quad (6-9)$$

A network of normalized radial basis function in 1-D (with $c = [-0.06 -0.03 0 0.03 0.06]^T$) looks as follows:

![Five normalized Radial Basis Function in one dimension](image)

**Figure 6-2: Five Radial Basis functions in 1D**

The system of this thesis consists of two states. The RBF network for both the actor and the critic can thus be represented as a surface in a three dimensional space. The vectors $\varphi$ and $\theta$ consist of $n$ values, where $n$ represents the total number of RBFs. The number of Radial Basis Function’s is part of the control design. A larger number of functions improves the accuracy of the approximation, but slows down the process as it takes more computation time to compute the estimated value $\hat{V}(x)$ and control input $u$ for the critic and the actor, respectively. This is a trade-off in the design procedure, which can be determined based on the simulation time and performance.

### 6-2-5 Exploration and exploitation in actor-critic

To find the estimated optimal value function, the agent needs to explore enough states (ideally all states infinitely often) to converge to the optimum $\hat{V}^*$. Note that due to the approximation of $V(x)$ and $\pi(x)$ the true optimal value will not be found. It is assumed that a reasonably accurate optimum is sufficient in this framework.

Reaching sufficient number of states is realized by exploration: at some iterations random actions need to be selected.

On the other hand, exploitation is also essential. This consists of using the current knowledge to choose the best actions (as estimated in the current iteration). Convergence and the time to converge to the optimum is related to the crucial trade-off between exploration and
exploitation. [19].

In this actor-critic algorithm, exploration is done using a certain exploration term added to the input according the current policy:

\[ u_k = \hat{\pi}(x_k, \varphi_k) + \tilde{u}_k \quad (6-10) \]

Where \( \hat{\pi} \) denotes the actor and \( \tilde{u}_k \) the exploration term. This is determined by some random number (with a zero mean, and a standard deviation of 1) multiplied by a constant term \( u_{\text{exp}} \).

### 6-2-6 Actor critic basic algorithm

Using the above explained theory, it is possible to describe all steps that will be implemented in the actor-critic algorithm. Remember that the goal of the iteration is to find the optimal policy \( \pi^*(x) \) which results from the optimal estimate of the value function \( \hat{V}^* \).

At time step \( k \), the four samples: \( x_k, u_k, x_{k+1}, r_{k+1} \) are used. The current state \( x_k \) is known, the input is determined based on the policy and the exploration term:

\[ u_k = \hat{\pi}(x_k, \varphi_k) + \tilde{u}_k. \]

This input is applied on the system and in the next time step state \( x_{k+1} \) can be observed and the immediate reward \( r_{k+1} \) is received. A quadratic reward function is used:

\[ r_{k+1} = \rho(x_k, u_k) = -x_k^T Q x_k - u_k^T R u_k \quad (6-11) \]

With matrices \( Q \) and \( R \) the weights on the states and inputs, respectively.

The next step is to formulate the Bellman equation, at iteration \( k \):

\[ V^\pi(x_k) = \rho(x_k, \pi(x_k)) + \gamma V^\pi(f(x_k, \pi(x_k))) \quad (6-12) \]

From this the temporal difference can be determined, which gives an estimate of the current prediction error:

\[ \Delta = \rho(x_k, \pi(x_k)) + \gamma V^\pi(f(x_k, \pi(x_k))) - V^\pi(x_k) \quad (6-13) \]

This temporal difference (TD) is used to update both the actor and critic. Note that they are approximated by a RBF network which depends on two parameter vectors \( \varphi \) and \( \theta \). Parameter \( \alpha_c \) is the learning rate of the critic. Those are updated as follows, starting with the critic:

\[ \theta_{k+1} = \theta_k + \alpha_c \Delta_k \frac{\partial \hat{V}(x, \theta)}{\partial \theta} \bigg|_{x=x_k, \theta=\theta_k} \quad (6-14) \]

The update rule can be interpreted as making a step closer to the optimum of \( \hat{V} \) using the temporal difference: if the TD is positive, the old estimate of \( \hat{V} \) was too low, so the value function needs to be increased. And vice versa if the TD is negative. This is done using this update rule. The update rule for the actor:

\[ \varphi_{k+1} = \varphi_k + \alpha_a \Delta_k \frac{\partial \hat{\pi}(x, \varphi)}{\partial \varphi} \bigg|_{x=x_k, \varphi=\varphi_k} \quad (6-15) \]

With \( \alpha_a \) the learning rate of the actor.
6-2-7 Simulation results of actor-critic RL

The actor-critic RL is now evaluated in simulation. The first simulations showed that the RL controller has difficulties to learn the desired behaviour. It takes some tuning effort to make it converge in the first place. Therefore the problem is simplified by only positioning the ball from the third \( x_0 = 0.025 \) to the second coil \( x_r = 0 \).

An observer is needed to estimate the velocity of the ball. A model-free observer was designed, by filtering the signal and differentiate to get the velocity. This did not yield satisfactory results compared to the model-based observer from section 4-3. Although Reinforcement Learning is model-free, this observer still requires a model.

From a zero initial control law it is relatively easy to learn to accelerate the ball towards the reference value. The controller is rewarded for any movement in the direction of the reference value (see equation (6-11)). The main challenge for the RL controller is to learn to decelerate the ball. This requires a specific timing and a sufficiently large input from the third coil. Both the actor and the critic are initialized at zero. The exploration rates for the second \((0.1)\) and third coil \((0.05)\) are larger than the other two coils \((0.001)\). In practice this means the learning problem is reduced to two coils only. The exploration rates converge to zero when the last trial is reached. The other parameters are listed in table 6-1.

<table>
<thead>
<tr>
<th>Parameter:</th>
<th>value</th>
<th>unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor learning rate</td>
<td>0.4</td>
<td></td>
</tr>
<tr>
<td>Critic learning rate</td>
<td>0.6</td>
<td></td>
</tr>
<tr>
<td>Discount factor</td>
<td>0.7</td>
<td></td>
</tr>
<tr>
<td>Exploration noise</td>
<td>([0.001 0.1 0.05 0.001])</td>
<td>A</td>
</tr>
<tr>
<td>Time per trial</td>
<td>2</td>
<td>s</td>
</tr>
<tr>
<td>Number of RBF’s</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>RBF intersection</td>
<td>0.5</td>
<td></td>
</tr>
</tbody>
</table>

**Table 6-1:** RL parameters used for simulation

The reward function (from equation (6-11)) has the following parameters:

\[
Q = \begin{bmatrix}
10000 & 0 \\
0 & 10
\end{bmatrix} \\
R = \begin{bmatrix}
1e^{-4} & 0 & 0 & 0 \\
0 & 1e^{-4} & 0 & 0 \\
0 & 0 & 1e^{-4} & 0 \\
0 & 0 & 0 & 1e^{-4}
\end{bmatrix}
\]

The learning process, in terms of the sum of rewards per trial, is as follows:
The sum of rewards has a large variation over the trials. This is partly caused by the instability of the system; a relatively small exploration term on the inputs can cause instability. The step response after 30000 trials is as follows:

Figure 6-3: Sum of rewards per trial in simulation. The sum of rewards keeps increasing during a large number of trails. Around the 30000th trial it converges to a value of approximately -50.

Figure 6-4: Step response of 30000th trial in simulation. The initial position is the third coil, $x_0 = 0.025$ m, and the reference value is 0 m, which is the location of the second coil. Each trial has a duration of 2 seconds.
The step response has the following performance:

<table>
<thead>
<tr>
<th>Controller: RL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rise time [s]</td>
</tr>
<tr>
<td>Settling time [s]</td>
</tr>
<tr>
<td>Overshoot [%]</td>
</tr>
<tr>
<td>Control Effort [J]</td>
</tr>
</tbody>
</table>

**Table 6-2:** Control performance of RL after 30000 learning trials in simulation

The actor for the second and third input and the critic are as follows:

![Actor u2](image1)

![Actor u3](image2)

![Final value function after learning (Critic)](image3)

**Figure 6-5:** Actor (u2 and u3) and critic after 30000 trials in simulation

This is a reasonable performance compared to the four controllers from Chapter 4. The overshoot, however, is larger than required. This may be caused by the quadratic reward function, in which the initial error is responsible for a large part of the total reward in each trial [32]. The behavior close to the reference value has only a small contribution in the total reward.
In the same article [32] an absolute reward function was compared with a quadratic one. An absolute reward function was also evaluated for the same simulation as above, but this did not give better results in terms of the performance measures and convergence time.

The RL controller quickly learns to accelerate the ball in the right direction. After 100 iterations the controller stabilizes the ball within 2 seconds, with a oscillatory behavior. It takes around 3500 trials to learn the right timing for the third coil to decelerate as well. From there it takes a large number of trials to improve this to the satisfactory performance of the 30000th trial. This is a clear downside of this method, even though these iterations can be done in simulation. The learned policy forms the initial policy for the first trial on the setup. From there on, the RL should learn the desired behavior on the experimental setup.

6-2-8 Implementation of RL on the experimental setup

Although the RL performed reasonably well in simulation, it is questionable whether the algorithm can learn the desired behavior sufficiently fast and accurate on the setup. Note that the RL could not match the performance of the C-SDRE in simulation, even after a large number of learning trials.

The learned policy and value function from the previous section are used as initial values in the experiments on the setup. The RL has large difficulties to stabilize the system using the initial policy from simulation (figure 6-5). The ball is accelerated towards the reference position, but the exerted force to decelerate the ball is insufficient to stop the ball in time. The initial policy is not sufficient, and the task seems too complex to learn on the setup.

An alternative for initializing the policy from the simulation result, is using the behaviour from a model-based controller on the setup. The state-action pairs which belong to the stabilizing controllers can be used to generate a policy \( \hat{\pi} \) from equation (6-6). The vector \( \varphi \) which represents the actor can be retrieved using the method described in article [33]. Although this method achieved reasonable performance on the setup (the behaviour from C-SDRE was imitated), the RL was not able to improve from there. Furthermore, this RL controller was not able to adapt to a different geometry. Based on these results it can be concluded that the RL controller does not perform better than the C-SDRE. Therefore a second learning controller is designed in the next section.

6-3 Local Linear Regression Imitation learning control

6-3-1 Introduction

A main advantage of Reinforcement Learning is that this does not require any model knowledge. It should learn the control actions that lead to the desired behaviour by interacting with the environment. However, this did not yield satisfactory results in simulation as a very large number of trials was needed to the desired behaviour (see section 6-2-7). On the experimental setup the learned behaviour from simulation was not successful, and the algorithm was not able to adapt to the experimental setup. An alternative approach is a supervised learning technique. These learning methods use more informative feedback compared to RL. For each input sample the correct output is known, directing the controller to the supplied
desired behaviour. Note that this differs from RL, where only a reward function is supplied, giving a direct reward at each time step. This does not include a reference trajectory which the system should follow. Reinforcement learning requires very precise tuning of parameters ([20], [34]), which is also mentioned in section 6-2-7. Furthermore, in the RL framework the controller generally starts learning the entire control law from scratch [34]. For the magnetic manipulator this is not strictly necessary, as a reasonable control law is already given by the C-SDRE method.

To improve the results and to overcome the drawbacks from RL a supervised learning controller might be a much better alternative approach in terms of performance, tuning effort and convergence time. This is further elaborated in this section.

6-3-2 Online learning using LLR

In this section an imitation learning controller is designed based on Local Linear Regression (LLR). This method is completely data driven. Based on multiple reference trajectories a reference model can be defined. This mapping defines at each state \( x(k) \) the desired next state \( x_{\text{des}}(k) \). The task is to control the system according to the desired behaviour specified by the reference model. The required control actions to achieve this behaviour is learned by interaction with the environment and storing the observations of the process state and control action [34]. Each input-output sample, consisting of the state \( k \), the next state \( k + 1 \) and the belonging control input \( u(k) \), is stored in the memory. The controller uses this memory, learned from online data, as an inverse model [34]. The memory database with all (relevant) input-output samples thus approximates the following inverse model function:

\[
u = P^{-1}(x, x_{\text{des}})
\]  

(6-16)

Using the inverse model to control the system is realized by reading samples using LLR. the control input to steer the system to the desired state is determined by taking the neighbouring samples and linearly interpolate around the query point \( x_q = x(k), x_{\text{d}}(k) \).
This desired next state, \( x_{\text{d}} \) is also determined based on Local Linear Regression, from the reference model \( R(x) \):

\[
x_{\text{des}, k+1} = R(x_k).
\]

Figure 6-6: Model based imitation scheme [34]. The model reference is denoted by \( R(x) \), the process model by \( P(x, u) \). Both \( R(x) \) and \( P(x, x_{\text{des}}) \) consist of input-output samples. The process model is learning online.
Although this method was designed mainly for robots which interact with humans, such as household robots, it may also be applicable for the magnetic manipulator. The desired behaviour is known from model-based controllers, and the memory can be initialized using samples from the stabilizing controller (e.g. the C-SDRE controller from section 4-6).

### Kd-tree search

Reading and storing samples to the memory is time-consuming when the memory size increases. Although storing the new samples can be done offline after each trial is finished, reading and interpolating the input $u_k$ samples from the inverse model must be done online. This can be time-consuming as the neighbouring points of the query point are stored throughout the whole memory. The first option is to search through the full memory to check the distances of all points to the point of interest. This can be regarded as the ‘naive’ approach, and was used for the LLR imitation learning method in the article [34]. In this research the algorithm was also validated on two experimental setups. Both setups, however, were controlled at sample periods between 0.03 and 0.05 seconds. The magnetic manipulator is controlled at a maximum sample period of 0.01 s, which limits the available search time in the data memory significantly.

Searching and storing data can be done much faster with the use of a tree structure. This was implemented for imitation learning in [35]. A kd-tree is a binary search tree structure in which all samples are stored. This makes it much more efficient to search neighbouring samples in the memory. Figure 6-7 shows the illustration of such a structure, including the names of the different elements in the tree.

The working principle can be explained with the following example from figure 6-8. The figure on the left shows 16 data points, divided into subsets. At each node the tree splits into two branches based on the splitting criterion. This determines how the data is stored in the memory. The query point to be evaluated with local linear regression is marked with the green cross.

![Figure 6-7](image)

**Figure 6-7:** Names of tree structure components [35]. The root is the starting point which contains of the total memory with all samples. By using splitting criteria the memory is divided into subsets, the branches. The tree eventually leads to leaves, which represent all individual samples.
A new sample is evaluated by searching the three, starting at the root and then going left or right based on the splitting criterion at each node. This leads to a close sample (point 16 in this example). From there a circle is drawn around the query point with the radius of the distance from the query point to the current best point. The distance from each node that overlaps with this circle can now be evaluated. By doing this the actual closest point to the query point is found. Note that it is only required to evaluate the points 9, 11, 12, 14 and 16 to find the closest point, which in this case is point 12. For the number of required neighboring points ($k_p$) the same procedure is done to find the list of the closest points to $x_q$.

The kd-tree search reduces the size of the search for the closest neighbouring points significantly. Theoretically, the complexity of the search problem reduces from $O(N)$ to $O(log(N))$ (see [36]).

### 6-3-3 Simulation results of Imitation learning

The imitation learning method with LLR is now evaluated in simulation. An initial process model $P(x, x_{des})$ is created by using the C-SDRE controller for several initial conditions varying around the initial position on which the learning controller will be evaluated ($x_0 = 0.050$ m), with a sample period $T_s = 0.01$s.

It is relatively easy for the controller to track the reference model (output samples only) when it is exactly equal to the training set trajectory (input-output samples) provided by the C-SDRE controller, although the control input is now determined based on the stored inverse model $P^{-1}$. The same trajectory is repeated which does not prove any learning ability. Therefore the training runs are done with a much slower controller. This makes it possible to see whether the controller is now able to learn to control the system according to the faster reference model. The first step is to train the controller with the C-SDRE controller, with tuning parameters: $Q = \begin{bmatrix} 20 & 0 \\ 0 & 0.25 \end{bmatrix}$, $R = diag(0.001)$.
Figure 6-9: Training run based on the slower C-SDRE controller. The response is tuned for a smooth and relatively slow behaviour such that the imitation learning algorithm can improve this response by interacting with the system and learning the approximate inverse model $u = P^{-1}(x, x_{des})$.

The fastest response possible in simulation is used as reference model.

Figure 6-10: Reference model based on the fastest C-SDRE controller response in simulation. It is created for four initial conditions close to the initial point $x_0 = [0.05 \ 0]^T$. 
After the 5 training runs from the slower C-SDRE controller, the imitation learning controller is simulated with 25 learning trials. The number of neighbouring samples, $k_p = 15$

![Position](image1)

![Velocity](image2)

![Control inputs](image3)

**Figure 6-11:** Simulated step response after 25 learning trials

The performance of both step responses are stated in table 6-3

<table>
<thead>
<tr>
<th>Controller:</th>
<th>C-SDRE</th>
<th>LLR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rise time [s]</td>
<td>0.34</td>
<td>0.25</td>
</tr>
<tr>
<td>Settling time [s]</td>
<td>0.67</td>
<td>0.49</td>
</tr>
<tr>
<td>Overshoot [%]</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Control Effort [J]</td>
<td>10.7</td>
<td>16.6</td>
</tr>
</tbody>
</table>

**Table 6-3:** Comparison between initial C-SDRE controller which is used to generate the training set, and the LLR imitation learning controller after 25 trials, $k_p = 15$

These results shows that after 25 learning trials the imitation learning controller is able to stabilize the system much faster than the initial training run. The learned behaviour now matches the performance of the fastest C-SDRE controller from simulation in section 4-6-2. This shows the potential of this method applied to the magnetic manipulator. Note that is not relevant to compare the control effort of both controllers, as the C-SDRE training run was deliberately tuned slower, to enable to the imitation learning controller to improve this
result in terms of the other measures. When this method is implemented on the experimental method, it might also learn to deal with the unmodelled phenomena (see section 3-2-1) which are not taken into account in the C-SDRE controller.

6-3-4 Experimental results of Imitation learning

This method is now implemented on the experimental setup. The process memory is initialized using 10 training runs with the C-SDRE controller. Each run started at a different initial condition, varying around the initial point $x_0 = [0.05 \ 0]^T$ for which the controller is evaluated.

The initial training runs were done with the fastest C-SDRE controller. This however, did not yield satisfactory results as the algorithm was not able to stabilize the system during the actual learning trials. To improve this, the training data is generated based on a slower controller. Based on the experiments done in section 6-3-3 the imitation learning should be able to learn to control the system according to the (faster) reference model. The generated initial process model $P^{-1}(x, x_{des})$ and reference model $R(x_k)$ are as follows:

![Figure 6-12: Reference model (top figure) and process model training data (bottom figure). The training data was generated by the C-SDRE on the setup, with a slightly slower response than the controller described in section 5-4. From there the controller learns to follow desired behaviour as supplied in the reference model. This was generated with the fastest C-SDRE available (using comparable tuning parameters as in section 5-4).](image-url)
A simulation is now performed, with 10 training runs based on the C-SDRE controller. After 13 additional learning trials the step response and control inputs are as follows:

![Step response LLR](image)

**Figure 6-13:** Step response of LLR Imitation learning controller on experimental setup. The process memory was trained with 10 runs of the C-SDRE controller for various initial conditions. This step response was achieved after 13 additional learning trials

The performance measures are:

<table>
<thead>
<tr>
<th>Controller:</th>
<th>LLR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rise time [s]</td>
<td>0.30</td>
</tr>
<tr>
<td>Settling time [s]</td>
<td>0.59</td>
</tr>
<tr>
<td>Overshoot [%]</td>
<td>1.4</td>
</tr>
<tr>
<td>Control Effort [J]</td>
<td>7.6</td>
</tr>
</tbody>
</table>

**Table 6-4:** Control performance of LLR Imitation learning on experimental setup

This performance is comparable to the fastest C-SDRE controller from section 5-4, with a slightly larger overshoot and control effort. This shows that this learning controller can improve the slower C-SDRE controller, and is able to come close to the performance of the best C-SDRE controller. It is now interesting to verify whether the LLR imitation learning controller can outperform the C-SDRE for a different ball size.

J. Damsteeg

Master of Science Thesis
Imitation learning for different ball size

The current steel ball (diameter 0.020 m, ball mass 0.032 kg) is replaced by a larger ball with diameter 0.024 m, mass 0.056 kg.

The goal of this experiment is to verify whether the LLR imitation learning controller can cope with a different geometry, by adapting itself towards the desired control behaviour. The results are compared with the C-SDRE, for which the model is supplied with the new mass. This gives the following result:

![Step response, MPC](image)

**Figure 6-14:** Step response of C-SDRE learning controller on experimental setup, with a larger ball ($m = 0.0563$. Although the correct mass is included in the model, the performance decreases.

The performance clearly decreases compared to the result in section 5-4. In some runs the controller has a large steady-state error. The controller needs retuning of control parameters to make sure it settles faster. It is now verified whether the LLR is able to adapt its control actions to the new ball size. The observer is supplied with the new value of the mass, and ten training runs are done based on the above C-SDRE controller. From there the imitation learning controller needs 20 trials to learn the desired behaviour. It must be noted that the controller is somewhat inconsistent, the system is not stabilized in all trials. The response after 20 additional trials is:
Figure 6-15: Step response of LLR Imitation learning controller on experimental setup. The process memory was trained with 10 runs of the C-SDRE controller for various initial conditions. This step response was achieved after 20 additional learning trials.

The performance measures for both controllers are:

<table>
<thead>
<tr>
<th>Controller:</th>
<th>C-SDRE</th>
<th>LLR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rise time [s]</td>
<td>0.32</td>
<td>0.33</td>
</tr>
<tr>
<td>Settling time [s]</td>
<td>0.98</td>
<td>0.65</td>
</tr>
<tr>
<td>Overshoot [%]</td>
<td>0</td>
<td>3.2</td>
</tr>
<tr>
<td>Control Effort [J]</td>
<td>15.9</td>
<td>9.9</td>
</tr>
</tbody>
</table>

Although both controllers are not consistent over different trials, it can be concluded that the imitation learning LLR controller performs better. The response is significantly faster in terms of settling time, although the overshoot is larger than desired. The required effort is significantly lower. The imitation learning controller can learn the desired behaviour by interacting with the setup, and is thereby slightly favourable compared to the C-SDRE controller. The main benefit of this method is that it uses the data, gathered on the experimental setup, directly to control the coils. This makes this method less dependent on the model compared to the C-SDRE, which is entirely model based.
6-4 Conclusions

In this chapter two learning controller were designed: a Reinforcement Learning controller and an imitation learning controller based on LLR. The RL controller learns the control policy by interacting with the environment and optimizing the total accumulated reward which is specified by the reward function. This method in itself is model-free, and is thereby expected to compensate for effects that are not taken into account by the model-based controllers. In the simulations it was shown that it requires a large tuning effort to make sure the algorithm converges to the desired behaviour. This is not trivial for the highly nonlinear and unstable system. The main challenge is to learn to decelerate the ball in time. The problem was simplified to only two coils to limit the size of the actor space. Although it takes a large number of trials, the controller is able to converge to a reasonable performance. The response is fast in terms of settling time, although the overshoot is larger than desired (around 5%). On the experimental setup, however, this performance is not achieved. The controller is unable to converge to a stabilizing policy.

As an alternative to RL, a supervised learning method is introduced. Based on successful experiments in [35] and [34], the imitation learning method, which uses an efficient search-tree Local Linear Regression technique, is implemented on the system. In simulation it is shown that this controller is able to imitate the behaviour supplied by a slower version of the C-SDRE controller. Based on the fast reference model the learning controller can improve from the training data towards a performance which is comparable to the fastest C-SDRE. A small number of learning trials is required to obtain a satisfactory response. On the experimental setup the imitation learning was able to imitate the desired behaviour as well, although the response is less consistent compared to the C-SDRE controller. Finally, an experiment is done to verify which controller performed better for a different ball size. It was shown that the imitation learning controller performs better than the C-SDRE in terms of settling time and control effort. The downside of the imitation learning controller is the inconsistent performance. Furthermore, the overshoot is slightly larger than desired.
Conclusions and further research

This chapter summarizes and concludes the work performed in this thesis. In section 7-2 several recommendations are done for future research.

7-1 Conclusions

In this thesis a controller is designed for a magnetic manipulation setup. The empirical model from article [1] is adopted, based on their experimental results with a feedback linearization controller.

Four model-based controllers were designed. In the simulation study a satisfactory performance is achieved for both the Feedback Linearization (FL) controller and the Constrained State Dependent Riccati Equation (C-SDRE) controller. The SDRE controller obtained a slightly slower response. The constraints are not taken into account, which makes it suboptimal with respect to the quadratic cost function. In the Nonlinear Model Predictive Controller (NMPC) the prediction is based on the nonlinear model, while also taking the constraints into account in the optimization. The NMPC, however, is not real-time feasible on the setup due to the computational complexity.

On the experimental setup the constrained-SDRE (MPC approach) performs best in terms of settling time, overshoot and control effort. The FL controller performed reasonably well, although it requires a large control effort to steer the ball to the reference position. The SDRE does not obtain a satisfactory result on the setup. The response is slow with a larger overshoot than desired.

From there on, two learning controllers were designed, an actor-critic RL controller and an Imitation Learning controller based on a search-tree implementation of Local Linear Regression (LLR). The simulation study showed that the actor-critic RL controller is able to learn the desired behaviour, although the overshoot is not prevented and it takes a large number of trials to converge. The Imitation Learning controller performs better as it is able to match the performance from the C-SDRE controller in a small number of training and learning trials.
On the experimental setup the RL controller does not converge to the desired behaviour. The Imitation Learning controller, however, is able to match the performance of the C-SDRE controller, and performs slightly better than the C-SDRE in terms of adapting to a different ball size. The C-SDRE controller has more difficulties in adapting to the larger ball, the response is slightly overdamped with a large control effort.

7-2 Future work

In this section some recommendations are done for future research on the magnetic manipulator setup, and on magnetic manipulation in general.

Planar manipulation

Although this thesis focused on the control design for a one-dimensional setup, the C-SDRE controller can also be implemented on a planar manipulation setup. This setup consists of 16 coils in a horizontal, two dimensional array. It is interesting to compare the C-SDRE from this thesis with the feedback linearizing controller which was designed in [1]. The C-SDRE controller can also be implemented on a 2-dimensional system (see 2D-model in article [2]). The main challenge will be to enable the C-SDRE to work in real-time, as the computational complexity is significantly larger. The planar manipulation setup has 16 actuators and 4 states (position and velocity in both dimensions). A C-implementation of the C-SDRE on a micro controller should then be considered to decrease the computation time.

Improve velocity estimation

One of the challenges on this magnetic manipulator setup is to make a reliable estimate of the velocity. In this thesis a model-based observer is used, which is limited by the accuracy of the model. This also influences the model-free learning controllers as those depend on an accurate velocity estimate. A better solution should be considered. Alternatives are the use of an Extended Kalman Filter (EKF) based on the nonlinear model, or an additional sensor to measure the velocity directly. The second solution will enable the learning controllers to be fully independent of any model.

Combining RL with LLR (model) learning

The implementation of the actor-critic RL controller did not obtain a satisfactory performance. In this thesis the actor and the critic are parametrized by a Radial Basis Function (RBF) network. In the thesis of [20] an alternative function approximator is introduced in combination with RL. Local Linear Regression (LLR) is used to approximate both the actor and the critic. By using LLR with actor-critic RL a better performance was obtained compared to RL with an RBF network [37]. This combination should be considered as an interesting option to improve the control performance on the magnetic manipulator.
Combining RL with C-SDRE

Another alternative control method can be designed by combining a stabilizing C-SDRE controller with an additive input which is determined by the RL controller. The additive input compensates the unmodelled effects that limit the performance of the C-SDRE controller. Such an approach, in combination with a FL controller, is implemented in [2]. One of the benefits is that the baseline controller stabilizes the system, which means that the task of the RL primarily consists of learning to compensate for the unmodelled effects. As the C-SDRE controller improved the performance of the FL controller on the same experimental setup [2], the combination of C-SDRE and RL might lead to better performance on the setup.

Reference tracking with C-SDRE

The results from this thesis can also be expanded to a reference tracking controller. The C-SDRE controller (MPC approach) can be adapted by parametrizing the MPC problem using incremental inputs [29]. If the future reference signal is known beforehand, this formulation enables the controller to take anticipative control actions. The drawback, however, is that a much larger control problem needs to be solved by the Quadratic Programming solver.
A-1 Position measurement

The position of the ball is measured with a optoNCDT 1401 laser sensor from Micro-Epsilon. The sampling frequency of this sensor is 1000 Hz. The measuring range is 20 mm, with a static resolution of 0.6 µm and a dynamic resolution (at 1 kHz) of 3 µm. The sensor has a current output. The voltage is measured across a resistor ($R = 497 \, \Omega$). The voltage is sampled and acquired by a Humusoft MF624 Data Acquisition board.

A-2 Electronics

The coils are controlled by two boards, of which the circuit is shown in figure A-1. Each board controls the current for two coils. The currents are sent from the PC to the board using the RS232 protocol.

![Circuit of the board to control the currents in two coils](image)

*Figure A-1: Circuit of the board to control the currents in two coils*
The currents are converted to an ASCII string, in the format: \[ q \text{ ABCD}[\text{CR}] \], in which AB and CD are strings that represent the currents for each coil: \[ A = 33 + \frac{x}{80} \text{ and } B = 33 + x \mod 80 \], in which \( x \) is the current shifted by 2048, such that \( x = 2048 \) with zero current, \( x = 0 \) with minimum range current, and \( x = 4096 \) is the maximum current (0.55 A). Symbol \% represents the division remainder (modulo).

The following Matlab-code was used to convert the currents to the ASCII strings and sent them to the board:

```matlab
com = rs232('GetParams','default'); % get default RS232 parameters
com.Port = 'COM6'; % define port number
com.BaudRate = 115200; % define Baud rate
com.ReadTimeout = 1; % define short timeout
com.WriteTimeout = 1; % define short timeout
rs232('open', com); % open the port

com2 = rs232('GetParams','default'); % get default RS232 parameters
com2.Port = 'COM7'; % define port number
com2.BaudRate = 115200; % define Baud rate
com2.ReadTimeout = 1; % define short timeout
com2.WriteTimeout = 1; % define short timeout
rs232('open', com2); % open the port

u(:,k) = min(0.55, max(0, u(:,k))); % Saturate currents
u_setup(:,k) = 2048 + round((u(:,k)/0.55)*2048);
A_out = 33 + fix(u_setup(1,k)/80); % A coil
B_out = 33 + rem(u_setup(1,k),80);
C_out = 33 + fix(u_setup(2,k)/80); % C coil
D_out = 33 + rem(u_setup(2,k),80);
Str12 = sprintf('%s%s%s%s', 'q ', char(A_out), char(B_out), char(C_out), char(D_out)); % Coil 1 and 2

A34_out = 33 + fix(u_setup(3,k)/80); % A coil
B34_out = 33 + rem(u_setup(3,k),80);
C34_out = 33 + fix(u_setup(4,k)/80); % C coil
D34_out = 33 + rem(u_setup(4,k),80);
Str34 = sprintf('%s%s%s%s', 'q ', char(A34_out), char(B34_out), char(C34_out), char(D34_out)); % Coil 3 and 4

rs232('write', com, uint8(abs(Str12) 13)); % Sent currents to Coil 1 and 2
rs232('write', com2, uint8(abs(Str34) 13)); % Send currents to Coil 3 and 4
```


J. Damsteeg

Master of Science Thesis


## Glossary

### List of Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOF</td>
<td>degrees-of-freedom</td>
</tr>
<tr>
<td>SDRE</td>
<td>State Dependent Riccati Equation</td>
</tr>
<tr>
<td>LTI</td>
<td>Linear Time Invariant</td>
</tr>
<tr>
<td>LQR</td>
<td>Linear-Quadratic Regulator</td>
</tr>
<tr>
<td>SDC</td>
<td>State-Dependent-Coefficients</td>
</tr>
<tr>
<td>FEM</td>
<td>Finite Element Method</td>
</tr>
<tr>
<td>RL</td>
<td>Reinforcement Learning</td>
</tr>
<tr>
<td>C-SDRE</td>
<td>Constrained State Dependent Riccati Equation</td>
</tr>
<tr>
<td>MPC</td>
<td>Model Predictive Control</td>
</tr>
<tr>
<td>NMPC</td>
<td>Nonlinear Model Predictive Controller</td>
</tr>
<tr>
<td>FL</td>
<td>Feedback Linearization</td>
</tr>
<tr>
<td>QP</td>
<td>Quadratic Programming</td>
</tr>
<tr>
<td>MDP</td>
<td>Markov Decision Process</td>
</tr>
<tr>
<td>PWM</td>
<td>Pulse Width Modulation</td>
</tr>
<tr>
<td>A-C</td>
<td>Actor-Critic</td>
</tr>
<tr>
<td>RBF</td>
<td>Radial Basis Function</td>
</tr>
<tr>
<td>TD</td>
<td>temporal difference</td>
</tr>
<tr>
<td>BIM</td>
<td>Boundary Integral Methods</td>
</tr>
<tr>
<td>AS</td>
<td>Assisting Saturated</td>
</tr>
</tbody>
</table>

Master of Science Thesis J. Damsteeg
**Glossary**

**VAF**  
Variance Accounted For

**LLR**  
Local Linear Regression

**List of Symbols**

\( \alpha \)  
Empirical constant in magnetic force model

\( \alpha_k \)  
Learning rate [-]

\( \beta \)  
Empirical constant in Magnetic force model

\( \gamma \)  
Discount factor

\( \pi \)  
Policy, control law

\( \rho \)  
Reward function

\( \theta \)  
Parameter vector (or matrix with \( n \) columns for \( n \) inputs \( u \)) to approximate \( \pi(x) \)

\( \varphi \)  
Parameter vector to approximate \( V(x) \)

\( A \)  
Continuous-time system matrix

\( B(x) \)  
Continuous-time input matrix in SDC form

\( B_d \)  
Discrete-time system matrix in SDC form

\( B_d(x) \)  
Discrete-time input matrix in SDC form

\( i \)  
Coil number (\( i = 1 : 4 \))

\( u \)  
Control input vector

\( V^\pi \)  
Value function, estimated cumulative reward

\( x \)  
System state

\( x_1 \)  
Ball position [m]

\( x_2 \)  
Ball velocity [m/s]

\( b \)  
Dynamic/viscous friction \([N \cdot s/m]\)

\( I \)  
Current [A] flowing through a coil

\( m \)  
Steel ball mass [kg]

\( F \)  
Magnetic force