BRNO UNIVERSITY OF TECHNOLOGY

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF MECHANICAL ENGINEERING

FAKULTA STROJNÍHO INŽENÝRSTVÍ

INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY

DESIGN OF ELECTROMAGNETIC POSITIONING PLATFORM FOR TESTING OF NONLINEAR CONTROL AND IDENTIFICATION ALGORITHMS

REALIZACE ELEKTROMAGNETICKÉ POLOHOVACÍ PLATFORMY PRO TESTOVÁNÍ NELINEÁRNÍCH ŘÍDÍCÍCH ALGORITMŮ A IDENTIFIKAČNÍCH METOD

MASTER'S THESIS

DIPLOMOVÁ PRÁCE

AUTHOR AUTOR PRÁCE	Bc. Matej Rajchl
SUPERVISOR VEDOUCÍ PRÁCE	Ing. Martin Brablc

BRNO 2020

BRNO UNIVERSITY OF TECHNOLOGY FACULTY OF MECHANICAL ENGINEERING

Specification Master's Thesis

Department:	Institute of Solid Mechanics, Mechatronics and Biomechanics
Student:	**Bc. Matej Rajchl**
Study programme:	Applied Sciences in Engineering
Study branch:	Mechatronics
Supervisor:	**Ing. Martin Brablc**
Academic year:	2019/20

Pursuant to Act no. 111/1998 concerning universities and the BUT study and examination rules, you have been assigned the following topic by the institute director Master's Thesis:

Design of Electromagnetic Positioning Platform for Testing of Nonlinear Control and Identification Algorithms

Concise characteristic of the task:

The goal of this thesis is to design and construct a platform, which can be used for testing of nonlinear control, identification, parameter estimation, and signal processing techniques in research and education projects in the Mechatronics laboratory, e.g. adaptive nonlinear control or online parameter estimation. The platform will perform a task similar to a “Ball on a Plate” device, e.i. positioning the steel ball, with the actuating force being provided by several electromagnets instead of tilting of the plate. The design will incorporate current sensing for precise control and a sensor to sense the ball position.

Goals Master's Thesis:

1) Perform a research study of similar devices, positioning techniques, and nonlinear control algorithms. Based on the research study, design and construct the device.
2) Design power and control electronics which can be interfaced with standard control hardware.
3) Create and validate a mathematical model of the device and study its behaviour in simulation.
4) Design and test (in simulation) several suitable nonlinear control algorithms.
5) Implement the most promising control algorithms to the ECU and compare the results.

Recommended bibliography:

NELLES, Oliver. Nonlinear system identification: from classical approaches to neural networks and fuzzy models. Berlin: Springer, 2011. ISBN 978-364-2086-748.

LJUNG, Lennart. System identification: theory for the user. 2nd ed. Upper Saddle River, NJ: Prentice Hall PTR, 1999. ISBN 978-0136566953.

VALÁŠEK, Michael. Mechatronika. Dot. 1. vyd. Praha: České vysoké učení technické, 1996. ISBN 80-010-1276-X.

NOSKIEVIČ, Petr. Modelování a identifikace systémů. Ostrava: Montanex, 1999. ISBN 80-722-50-0-2.

Deadline for submission Master's Thesis is given by the Schedule of the Academic year 2019/20

In Brno,

L. S.

prof. Ing. Jindřich Petruška, CSc.	doc. Ing. Jaroslav Katolický, Ph.D.
Director of the Institute	FME dean

Abstract

This thesis deals with design and construction of electromagnetic positioning platform for testing of nonlinear control and identification algorithms. The plaform is based on shaping of the magnetic field using three electromagnets and positioning of a steel ball using this magnetic field on a resistive touch panel which measures the ball position at each point. The platform is meant mainly for demonstration of different nonlinear control algorithms. Three of which are shown and tested in this thesis.

Abstrakt

Táto diplomová práca sa zaoberá návrhom a konštrukciou elektromagnetickej, polohovacej platformy, pre testovanie nelineárnych riadiacich a identifikačných algoritmov. Platforma je založená na tvarovaní magnetického poľa v každom bode pomocou troch elektromagnetov a polohuje oceľovú guličku po dotykovom paneli ktorý sníma polohu tejto guličky. Platforma má slúžiť hlavne pre demonštráciu rôznych nelineárnych riadiacich algoritmov vo výukovom prostredí. Tri príklady takýchto algoritmov sú ukázané a overené v rámci tejto diplomovej práce.

Keywords

nonlinear control, electromagnetism, positioning, SDRE, feedback linearization, PID, parameter estimation

Klíčová slova

nelineárne riadenie, elektromagnetizmus, polohovanie, SDRE, spätnovazobná linearizácia, PID, odhad parametrov

Reference

RAJCHL, Matej. *Design of Electromagnetic Positioning Platform for Testing of Nonlinear Control and Identification Algorithms.* Brno, 2020. Master's thesis. Brno University of Technology, Faculty of Mechanical Engineering. Supervisor Ing. Martin Brablc, 56 pages

Rozšířený abstrakt

Nelineárne riadenie je jednou z hlavných odvetví modernej teórie riadenia. Hlavný rozvoj tohoto odvetvia nastal práve vďaka rozvoju výpočtovej techniky začiatkom 21. storočia. Nové technológie tiež priniesli modernizáciu klasických riadiacich algoritmov pre nelineárne systémy.

Pre demonštráciu toho, ako tieto algoritmy fungujú musí nelinearita v riadenom systéme spĺňať určité kritériá. Musí byť dostatočne významná na to aby ovplyvnila správanie systému, musí byť spojitá a ideálne aj hladká. Nevhodné nelinearity sú aj také, ktoré nemajú determinsitické správanie, alebo sa nedajú modelovať pomocou deterministického modelu (typu obecný model trenia).

Taktiež systém na ktorom chceme dané riadenie demonštrovať by mal spĺňať určité kritériá. Musí mať taký aktuačný zásah aby nebolo jednoduché ho uriadiť iba s použitím PID regulátoru (ak je akčný zásah dosť silný, tak aj lineárny regulátor dokáže uriadiť veľké množstvo nelineárnych systémov). Mal by mať viacero stupňov volnosti a aj viacero vstupov aby sa dalo na ňom demonštrovať nelineárne stavové riadenie.

Tieto požiadavky viedli k motivácii za touto diplomovou prácou. Cieľom bolo postaviť zariadenie, ktoré by spĺňalo požiadavky uvedené vyššie a slúžilo by pre demonštračné a edukatívne účely. Zároveň bolo žiaduce ukázať, že zariadenie funguje na niektorých moderných algoritmoch nelineárneho riadenia.

Finálna podoba zariadenia bola navrhnutá tak, že pozostáva z troch elektromagnetov smerujúcich nahor, na nich pripevneného odporového dotykového panelu a ocelovej guličky, ktorej polohu snímame dotykovým panelom. Účelom je pomocou elektromagnetov polohovať guličku v rovine.

Riadenou veličinou sú prúdy v troch cievkach, pričom pre každú cievku bol navrhnutý vlastný napätím ovládaný prúdový zdroj. Výstupom je poloha v dvoch osiach dotykového panelu, pre ktorý bol taktiež navrhnutý vlastný obvod, ktorý prevádza digitálne dáta na analógovú hodnotu napätia. Celé zariadenie bolo navvrhnuté tak aby sa dalo pripojiť k bežným typom riadiaceho hardwaru. V prípade tejto práce sme použili kartu MF624 od spoločnosti *Humusoft*.

Systém má teda viacero vstupov (3 požadované prúdy) a viacero stupňov voľnosti a tak spĺňa predpoklady na zariadenie uvedené vyššie.

Pre analýzu správania zariadenia a následné riadenie bol najskôr zostavený numerický model. Tento model bol zostavený na základe nameraných dát pri ovládaní zariadenia z PC. Najskôr bola vybraná správna štruktúra modelu a následne odhadnuté jeho parametre. Po zostavení tohoto modelu bolo analyzované jeho správanie, riaditeľnosť, stabilita a boli otestované tri algoritmy jeho riadenia - PID, Input-state feedback linearization, State-dependent riccati equation. Bola porovnaná ich stabilita a výpočtová náročnosť.

Následne boli uvedené algoritmy použité pre riadenie výsledného zariadenia a kvalita ich riadenia bola porovnaná. Výsledné zariadenie splnilo svoj účel a môže sa dalej rozvíjať v rámci budúcich bakalárskych a diplomových prác.

Design of Electromagnetic Positioning Platform for Testing of Nonlinear Control and Identification Algorithms

Declaration

I hereby declare that this Master's thesis was prepared as an original work by the author under the supervision of Ing. Martin Brablc. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

....
Matej Rajchl
June 21, 2020

Acknowledgements

I would like to give my sincere gratitude to Ing. Martin Brablc for supervising and consulting the progress of this thesis. Also I would like to thank my fiancé Ivica Folučková for emotional, mental and moral support during the creation of this thesis.

Contents

List of used acronyms

BIBO bounded-input bounded-output.

DAC Digital to Analog Converter.

EKF Extended Kalman Filter.

FBL Feedback Linearization.

I2C Inter-Integrated Circuit.

IC Integrated Circuit.

IIR Infinite Impulse Response.

IR Infrared.

KF Kalman Filter.

LTI Linear Time Invariant.

MIMO Multiple-input and multiple-output.

MSE Mean Squared Error.

PBH Popov-Belevitch-Hautus.

PID Proportional Integral Derivative.

SDRE State-Dependent Riccati Equation.

SINDy Sparse Identification of Nonlinear Dynamics.

SISO Single-input single-output.

SPI Serial Peripheral Interface.

SVD Singular Value Decomposition.

Chapter 1

Introduction

Nonlinear control is one of the main branches in control theory and until the beginning of this century did not offer too much development. The biggest advancements came with the increase of computational power of modern computers and most of the mathematical theories could become the algorithms used in various applications and industries today.

The nonlinear control can be found in many different applications such as MagLev trains, aerospace engineering, nonlinear hydraulic valves, crane control, variable reluctance motors, electromagnetic manipulation, petrochemical process control and many others.

The common approach to nonlinear control is the use of some form of coordinate transform, which usually leads to input-output Feedback Linearization (FBL) algorithms or „bang-bang“ type of controller. These controllers were used in many industrial application the latter one being the early MagLev trains and the former being used in spacecraft control [4], [32]. The industry standard for nonlinear control are usually gain scheduling algorithms, because of lower computational power of embedded electronics.

These algorithms evolved over the years and also new algorithms came to existence derived from linear systems and applied to linearized systems via the Jacobian matrices or systems of the state-dependent form.

In educational cases it is not simple to find many illustrative examples on which the modern nonlinear control algorithms are necessary for precise control. Most of the educational models which can be bought from companies like *Quanser* offer just a limited range of nonlinear ones and most of them can be controlled just using regular Proportional Integral Derivative (PID) controller, because their actuation is much stronger than needed for the model which compensates for the nonlinearity using more controllability or stability.

This leads us towards the motivation behind this thesis. The requirement to have an educational model which can demonstrate the modern nonlinear algorithms, in which the nonlinearity is „well-behaved“. This means that the nonlinearity is continuous, deterministic and can be well modelled. Also the model would have to have multiple degrees of freedom and multiple-inputs to demonstrate nonlinear state-space controllers.

Main inspiration behind using the magnetic force as a nonlinear input to the model was the magnetic levitation model with which we have lot of experience. We know it cannot be controlled just by using the PID controller as proven in [27] and the magnetic force is nonlinear and continuous and a model exists which describes the force very accurately. Another advantage the magnetic force brings to the model is that it creates unstable node in the state space of the system, which can be interesting in the field of parameter estimation or adaptive control.

Chapter 2

State of the art

2.1 Similar devices

In this section an overview of similar devices to the one which will be described within this thesis will be provided. By similar device we understand a laboratory or educational device which uses magnetic field to position objects and requires a form of numerical control to achieve this task. Into this category we could include many devices, which use electromagnetic forces to exert motion such as motors, linear actuators or solenoid actuators, but those do not fall into the educational category and therefore will not be a focus of this thesis.

2.1.1 Magnetic levitation

Very simple 1D example of magnetic manipulation is magnetic levitation model. One of the basic laboratory models available is the one which is depicted on figure 2.1. The model consists of a steel ball which levitates under a single electromagnet. The electromagnet provides upwards force while the gravity pulls the ball down. The coil has a fast analog current regulator on input and the ball position is measured with inductive position sensor. The extended description of the device is available at [2]. The advantage of a single coil design is that there is just one nonlinear term however this allows for only one degree of freedom and limiting the other two can be problematic. Usually if the ball oscillates side to side it is impossible to stop (or prevent) this oscillation.

2.1.2 mBot

One of the simpler 2D positioning devices available, is the so called mBot [1]. It is small positioning platform for 2D horizontal manipulation of a small magnet. It is based on single PCB design, which encompasses also the coils right into the PCB. This can be seen on figure 2.2.

This model provides only control in open-loop without any form of feedback or position sensing. The grid is designed to work the way that only one of the „pixels“ is active at a time. So to make the magnet move the pixels need to switch in the correct order. The position of the robot can be controlled with the user input via buttons. The logic and the coil switching is controlled with Atmega328 microcontroller.

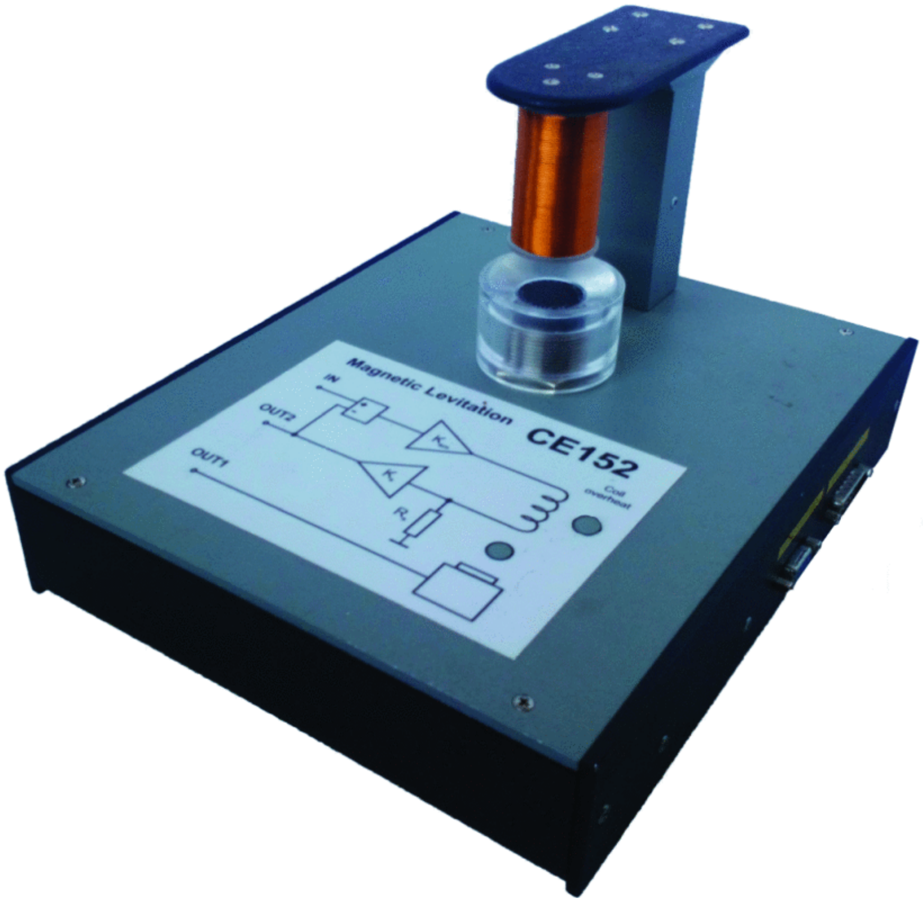


Figure 2.1: **Magnetic levitation** A model CE152 of magnetic levitation from company Humusoft [27]

2.1.3 MagMan

A very similar device developed through several theses ([9][29][38]) at the Czech Technical University in Prague. It consists of array of coils which control the position of a steel ball on a plate. The device is depicted on figure 2.3. The total number of coils can be adjusted. They come in blocks of 2x2 coils, have built in current controller for each coil and ARM processor which communicates on communication bus with higher level control system. The array itself can be viewed as array of intelligent actuators. The position of the ball is sensed with resistive touch panel and also by RGB camera. The platform can control the position of the ball in two directions but can also sense the downward force on resistive touch display.

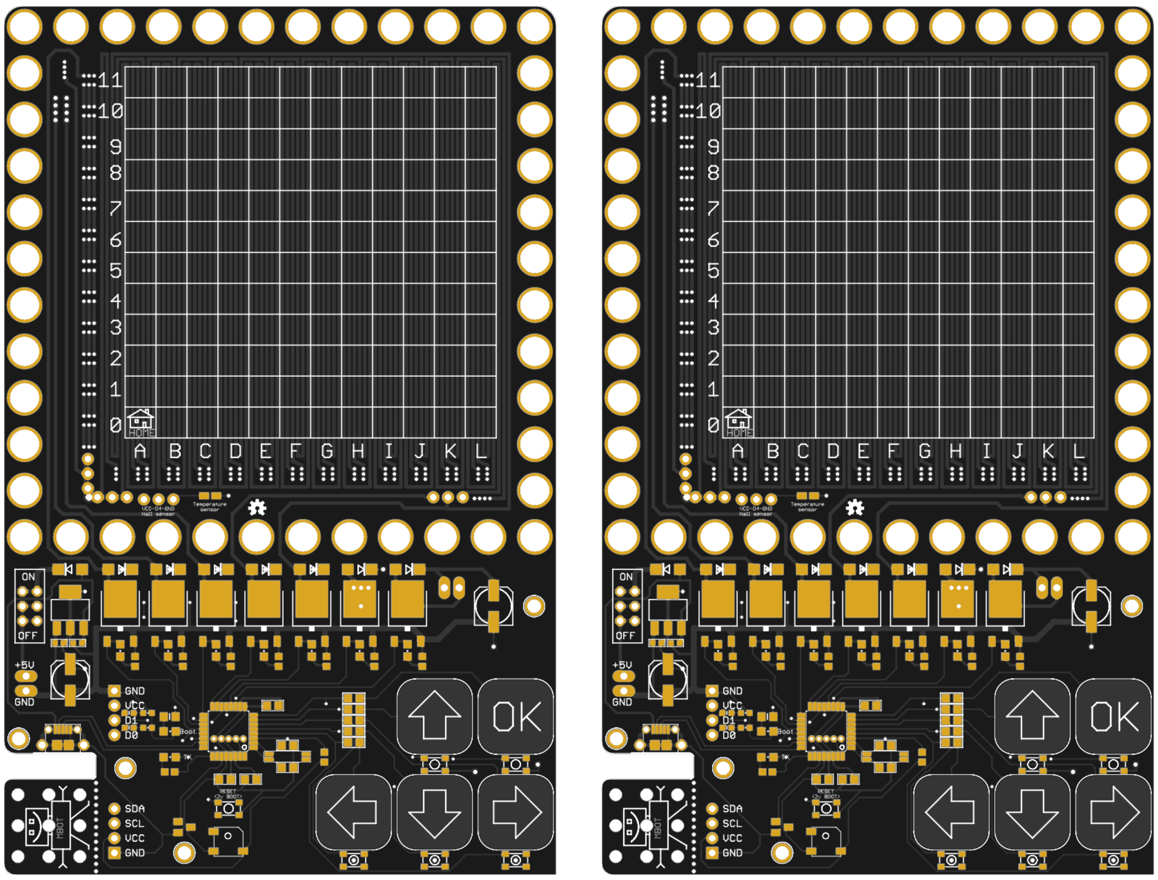


Figure 2.2: **mBot** TopView(left) BottomView(right) [1]

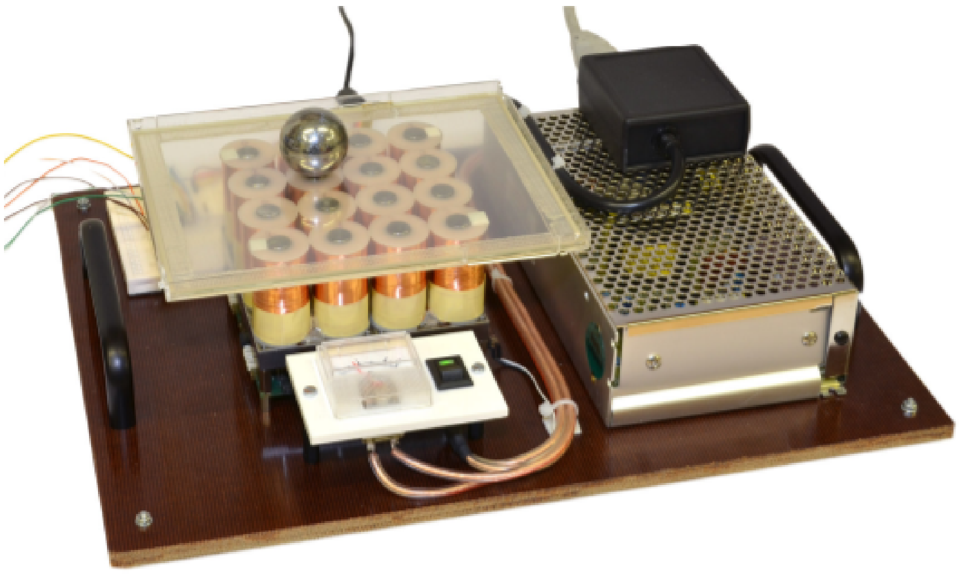


Figure 2.3: **MagMan** [9]

2.2 Electromagnet magnetic field models

Modeling the magnetic field around the electromagnet is no simple task. Usually perfect analytical solution does not exists because it is not possible to define or measure all of geometrical parameters or imperfections of the electromagnet. Also the models usually simplify or approximate some part of the dynamics and this creates approximate models. The most precise models are based on measurement or FEM analysis, but these models are usually not viable for real-time control. In literature there are various approaches described and this section will provide the overview of available models and the difference between them.

2.2.1 Induced dipole moment model

The force on magnetizable object inserted into the magnetic field can be expressed as force on magnetic dipole [38]

$$\mathbf{F} = (\mathbf{m}\nabla)\mathbf{B} \tag{2.1}$$

where $\mathbf{m}$ represents effective magnetic dipole moment and $\mathbf{B}$ represents magnetic flux density. If the object is made from magnetically linear material and has spherical shape and the magnetic field is created by a magnetic monopole located at the origin of the coordinate system this equation expands to

$$\mathbf{F} = k\frac{q_m^2}{16\pi^2}\nabla\frac{1}{(x^2+y^2+z^2)^2} \tag{2.2}$$

where k represents a constant encompassing geometrical and magnetic properties of the manipulated object and q_m corresponds to the magnetic strength of a monopole based on the current through the coil. Calculating the gradient the corresponding forces are derived as (where c represents the coil constant encompassing all the constant terms into a single parameter)

$$F_x = -\frac{i^2cx}{(x^2+y^2+z^2)^3} \tag{2.3}$$

$$F_y = -\frac{i^2cy}{(x^2+y^2+z^2)^3} \tag{2.4}$$

$$F_z = -\frac{i^2cz}{(x^2+y^2+z^2)^3} \tag{2.5}$$

2.2.2 Magnetic coenergy model

Another approach to modelling a magnetic force is using a method which is based on the virtual work [10]. This model is based on calculating the gradient of the coenergy of the magnetic circuit. Coenergy of the coil is defined as

$$W_{co} = \int_\Omega \int_0^H \mathbf{B}dHd\Omega \tag{2.6}$$

where B represents magnetic flux density, H represents magnetic field intensity and Ω represents the integration volume. After substituting for B and H in terms of geometry and current [25] we obtain the following equation

$$W_{co} = \frac{L(x,y,z)i^2}{2} \tag{2.7}$$

where $L(x,y,z)$ is the inductance of magnetic circuit and i represents current through the coil in time t. From standard definition of inductance [25] $L(x,y,z) = \frac{N^2}{R_m}$ (where N is number of turns of the electromagnet and R_m is the magnetic resistance of the magnetic circuit) we can derive a specific magnetic force definition for our case

$$\mathbf{F} = ki^2\nabla\left(\frac{1}{(x^2+y^2+z^2)^{1/2}}\right) \tag{2.8}$$

$$F_x = \frac{-kxi^2}{(x^2+y^2+z^2)^{3/2}} \tag{2.9}$$

$$F_y = \frac{-kyi^2}{(x^2+y^2+z^2)^{3/2}} \tag{2.10}$$

$$F_z = \frac{-kzi^2}{(x^2+y^2+z^2)^{3/2}} \tag{2.11}$$

where k represents the constant terms defining geometry and magnetic properties, i the current through the coil and x, y, z represent distance from the coil in the Cartesian coordinates.

2.3 Nonlinear control algorithms

The subject of nonlinear control deals with the analysis and the design of nonlinear control systems, i.e. of control systems containing at least one nonlinear element. There are various differences between classical linear control and more advanced nonlinear control, however these are very well described by literature [33] [36], and will not be the focus of this section. In this section a small vertical slice of algorithms used for nonlinear control will be presented, with focus on describing the basic principles of algorithms used in later chapters.

2.3.1 Composite PID control

Composite PID control is the simplest step from classical linear control towards algorithms used for nonlinear system control. This approach has been used by various sources for wide range of applications [12] [7] [27] [37]. Main concept which is used is designing a controller and adding a feed-forward compensator which can compensate for nonlinear dynamics.

The feed-forward compensator is usually some form of inverse model of the system. It can be based on the physical equations, some form of black box or general model, fuzzy models, neural networks etc.

For the purpose of this section let us assume that our control problem is a tracking problem, where vector $\mathbf{x_d}$ is a desired state-space trajectory we want to track and lets assume equation 2.12 describes forward model of the system. Then we can specify the inverse model as 2.13. Note that this model does not always have to be obtainable, but for many systems it is possible to derive.

$$\dot{\mathbf{x}} = f(\mathbf{x}) + g(\mathbf{x})\mathbf{u} \tag{2.12}$$

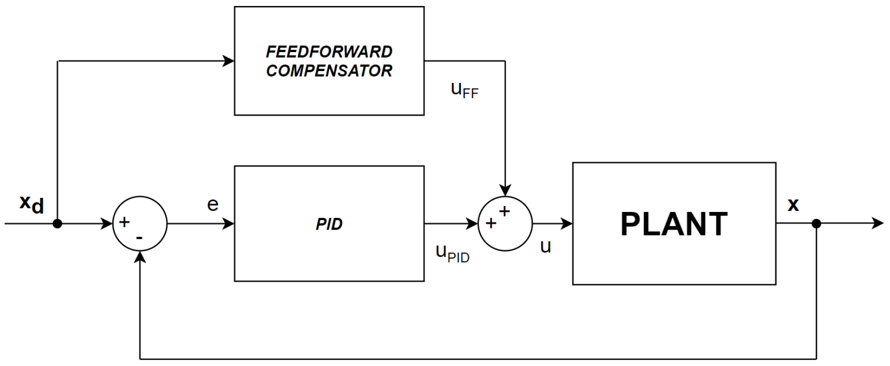


Figure 2.4: **Typical composite controller layout.** A PID with feedforward compensator

$$\mathbf{u} = g(\mathbf{x})^{-1}(\dot{\mathbf{x}} - f(\mathbf{x})) \tag{2.13}$$

Now if we derive equations for the system shown on figure 2.4 we obtain the following

$$\dot{\mathbf{x}} = f(\mathbf{x}) + g(\mathbf{x})(\mathbf{u}_{PID} + \mathbf{u}_{FF}) \tag{2.14}$$

$$\dot{\mathbf{x}} = f(\mathbf{x}) + g(\mathbf{x})(\mathbf{u}_{PID} + g(\mathbf{x_d})^{-1}(\dot{\mathbf{x_d}} - f(\mathbf{x_d}))) \tag{2.15}$$

Expanding and simplifying.

$$\dot{\mathbf{x}} = f(\mathbf{x}) - g(\mathbf{x})g(\mathbf{x_d})^{-1}(f(\mathbf{x_d}) - \dot{\mathbf{x}}) + g(\mathbf{x})\mathbf{u}_{PID} \tag{2.16}$$

So under the assumptions that the trajectory is reachable, the PID controller is tuned and can stabilize the system along the said trajectory. We can approximate that $\mathbf{x} \approx \mathbf{x_d}$ and system dynamics reduce to

$$0 = g(\mathbf{x})\mathbf{u}_{PID} \tag{2.17}$$

We see that this approach does not fully compensate the nonlinear dynamics, but can be further improved by modifying the PID output signal to account for $g(x)$. Although this is not usually done because other methods can be used to solve this problem (which are described in later sections of this chapter). Also because of the imperfections of the model or the inaccuracy of the model parameters the compensation is not perfect and the PID controller does not have zero action.

This approach is widely used in industry and is one of the most simple approaches to nonlinear control. However, it is not ideal and has a lot of limitations. One is that it cannot compensate general nonlinearities and is difficult to apply to MIMO systems, also the inverse model of the system does not have to be always obtainable.

2.3.2 Sliding mode control

A very simple approach to robust nonlinear control is so called sliding mode control (or sliding control). This approach is based on the remark that it is much easier to control first order systems (be they linear, nonlinear or uncertain), than it is to control complex nth order system.

As described in [33], the method is based on defining so called sliding surfaces which reduce the problem from tracking trajectory in n dimensional space to keeping a scalar quantity at zero.

Consider single-input dynamic system of the form

$$x^{(n)} = f(\mathbf{x}, \mathbf{u}) \tag{2.18}$$

and then we consider $\mathbf{x}_\mathrm{d}$ a desired trajectory we want to track, with initial condition

$$\mathbf{x}_d(0) = x(0) \tag{2.19}$$

then a sliding surface is defined as

$$s(\mathbf{x}, t) = (\frac{d}{dt} + \lambda)^{n-1}(\mathbf{x} - \mathbf{x}_d) \tag{2.20}$$

Using a simple variable transformation $\tilde{\mathbf{x}} = \mathbf{x} - \mathbf{x}_d$, we see that λ is by definition positive definite damping constant to keep the sliding surface stable. (e.g. if n=3, then sliding surface will take a form 2.21)

$$s = \ddot{\tilde{x}} + 2\lambda\dot{\tilde{x}} + \lambda^2\tilde{x} \tag{2.21}$$

Now we can define control law such that it keeps our surface dynamics zero. There are different approaches to this task. For example [33] suggests defining u_{eq} as equilibirum state input and u_+ and u_- as inputs which stabilize $\frac{ds}{dt}$ (Figure 2.5 demonstates this).

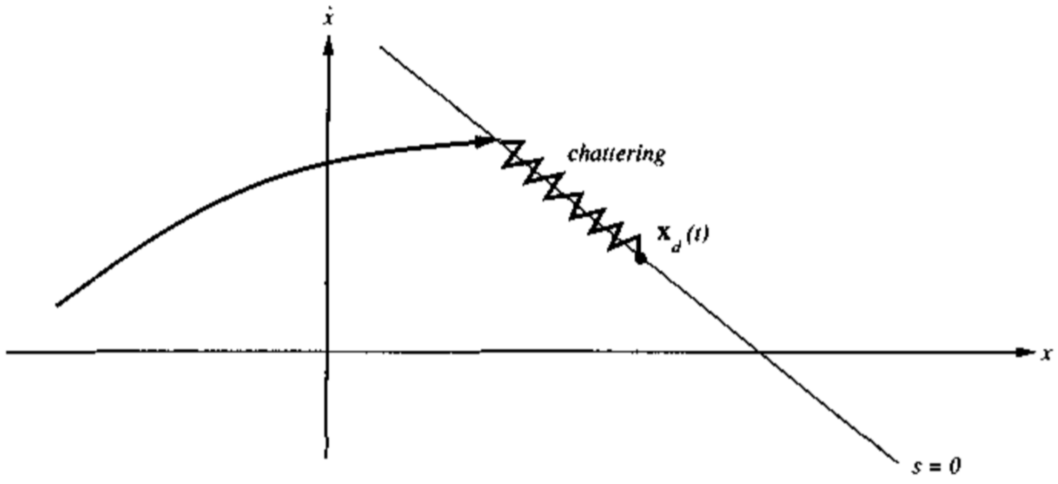


Figure 2.5: **Stabilization around s.** Demonstration of chattering around sliding surface s [33]

Sliding mode approach offers very good performance and is relatively easy to implement, on the other hand the price is very high actuation activity, which can inject a lot of noise to the state measurements (or state observer through input) and can cause problems down the road. Also if we have a system with non-zero phase dynamics this approach can become problematic because of stability concerns. Another downside is that we need full-state feedback and also full-state trajectory planner and keep the initial condition 2.19 satisfied at all times. However we do require the full-state trajectory planner while using almost any other nonlinear control algorithm, so this is may not be an issue.

2.3.3 Gain scheduling

There are many different notions which are referred to as gain scheduling (GS) by literature [23]. From blending controller gains to switching whole controllers or model dynamics based on the state or input of the system. In general, GS encompasses attenuation of nonlinear dynamics in various ranges of operation. For the sake of this thesis an overview of classical gain scheduling techniques will be provided, the reason being GS algorithm shares many aspects with other nonlinear control algorithms and drawing a parallel between them can be beneficial for understanding the other algorithms. Classical GS approach (as described by [23]) is based on decomposing a design of a nonlinear controller into designing a number of linear controllers instead. A typical GS approach follows these steps ([31]):

- Step 1: A family of approximate LTI models of nonlinear plant is derived. these models can be either linearized around equilibrium points, dependent on LTI parameter or dependent on exogenous signals (e.g. states, inputs, arbitrary parameters).
- Step 2: LTI controllers are designed to achieve a specific stability and performance at each operating point. In some cases an analytical solution, which incorporates changing parameter is possible, however in general this is not possible and the controllers are designed empirically.
- Step 3: Controllers are implemented in such a way that their coefficients are scheduled based on current scheduling parameter.
- Step 4: A local performance is observed and the controllers are fine-tuned to match the desired performance or stability criteria.

Using a gain scheduling algorithm has many advantages. Most of the calculations can be performed offline so the resulting controller can be easily implemented on less powerful hardware (e.g. microcontroller). Robustness and stability can be verified locally at each operating point of the GS controller and performance/stability can be fine-tuned depending on the requirements of the controlled process. Downsides of this approach is that it requires a good nonlinear model of the plant (or enough time tuning on real plant), does not guarantee stability or even controllability at any of the operating points during the design and has to be tuned to achieve good performance.

Most of state-dependent control algorithms are usually referred to as GS algorithms, because of their state-dependent format and controller changing behavior, so understanding the basic GS is important.

2.3.4 Feedback linearization

Feedback Linearization (FBL) is a technique well known and successfully used in high performance applications (helicopter and aircraft control, industrial robots, some medical applications etc. [33][28][16]). However until recent years not a lot of publications described using the FBL algorithm for MIMO applications. Main concept behind feedback linearization is the transformation of the coordinate system to simplify dynamics of the system from nonlinear to linear. Afterwards a linear controller is designed, the inverse transform of the coordinates is applied and the nonlinear actuation is calculated to control the system. Intuitively we try to find a way to cancel out the nonlinear terms of our system and design the controller without them.

First let us assume a first-order SISO system

$$\dot{x} = f(x) + g(x)u \tag{2.22}$$

$$\mathbf{y} = h(x) \tag{2.23}$$

in which for simplicity we assume $y = x$. So the input-output feedback linearization is equivalent to input-state feedback linearization. Based on [33] we can derive that our linearizing feedback (in this case) has the form

$$u = g(x)^{-1}(-f(x) + v) \tag{2.24}$$

where v is the new linearized system input. If we substitute 2.24 into 2.22 we can derive that our system has reduced to simple linear first order system

$$\dot{x} = v \tag{2.25}$$

This can be simply extended to SIMO systems or any n$^{\text{th}}$ order system however that is well documented in literature ([33]) and will not be described in this section.

Now assume a MIMO system in the general form

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{2.26}$$

If the function ϕ can be written as a state dependent transformation of the form 2.27, where $F(\mathbf{x})$ and $G(\mathbf{x})u$ are vector fields describing nonlinear dynamics of the system, then we can describe the linearizing control law as 2.28. Where vector (v) represents new input to the system.

$$\dot{\mathbf{x}} = F(\mathbf{x}) + G(\mathbf{x})\mathbf{u} \tag{2.27}$$

$$u = G(\mathbf{x})^{-1}(-F(\mathbf{x}) + \mathbf{v}) \tag{2.28}$$

Note that this approach assumes state-dependent matrix is square and invertible $(det(G(x)) \neq 0)$. However we can get around these strict constraints by using Moore-Penrose pseudoinverse instead of matrix inverse, to get best approximate solution for systems where the number of inputs does not equal to the number of states. [26] The obtained solution is best least-squares or 2-norm (depending on whether the system is under- or over- determined)

So the linearizing feedback takes the following form

$$\mathbf{u} = G(\mathbf{x})^{\dagger}(-F(\mathbf{x}) + \mathbf{v}) \tag{2.29}$$

Substituting 2.29 into 2.27 we get a linear system

$$\dot{\mathbf{x}} = \mathbf{v} \tag{2.30}$$

Now we can design a linear controller (usually a state-space one) for this MIMO system and get a resulting control law as 2.31 where (K) is a controller gain matrix.

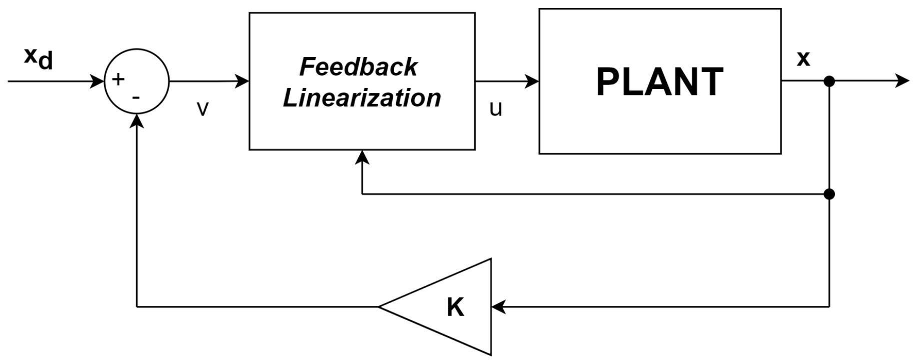


Figure 2.6: **FBL Controller.** A typical configuration of FBL MIMO controller

$$\mathbf{u} = G(\mathbf{x})^{\dagger}(-F(\mathbf{x}) - \mathbf{Kx}) \tag{2.31}$$

A big advantage this approach brings to the table compared to gain scheduling is that we need to tune only a single state space controller which can be used to control the whole MIMO system. If we have a good model of the system the implementation is very easy, especially if we have numerical computational tools such as MATLAB and Simulink. It is computationally more expensive than approaches which use GS algorithm or sliding mode, but the number of steps required to implement this algorithm is much smaller and can be done very quickly. FBL has a few downsides as well. First thing to note is that vector $\mathbf{v}$ can be in the null space of the $G(x)$. Which means that some inputs of $\mathbf{v}$ would not translate to $\mathbf{u}$ (controllability of the system cannot be increased just by coordinate change). Second problem is that the system may become uncontrollable because of the varying rank of $G(x)$.

2.3.5 State-dependent Riccati equation based control

The State-Dependent Riccati Equation (SDRE) is the basis for sub-optimal feedback control of nonlinear quadratic regulator (NQR) problem. The basic thought is to decompose nonlinear function as a sum of standard state-dependent linear transformations and use regular LQR control based on Riccati equation. This approach is intuitive and is very similar to what classical gain-scheduling algorithm does, however the LQR is not calculated beforehand and is recalculated at each time step. This leads to sub-optimal control, which however can be tuned precisely to achieve desired performance and stability. Typical LQR implementation (as described by [17]) for linear continuous state-space system described by equation 2.32

$$\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu} \tag{2.32}$$

is such that we find optimal matrix gain $\mathbf{K}$ which minimizes the following cost function.

$$J = \int_0^{\infty} (\mathbf{xQx^T} + \mathbf{uRu^T} + \mathbf{2x^TNu})dt \tag{2.33}$$

where $\mathbf{u}$ is defined as (2.33) and matrices $\mathbf{Q}$,$\mathbf{R}$,$\mathbf{N}$ provide user-defined weighing between state and actuation.

$$\mathbf{u} = -\mathbf{Kx} \tag{2.34}$$

Optimal $\mathbf{K}$ is then found by solving the algebraic Riccatti equation of the form 2.35 and finding the feedback gain as 2.36

$$\mathbf{A^T P} + \mathbf{PA} - (\mathbf{PB} + \mathbf{N})\mathbf{R}^{-1}(\mathbf{B^T P} + \mathbf{N^T}) + \mathbf{Q} = 0 \tag{2.35}$$

$$\mathbf{K} = \mathbf{R}^{-1}(\mathbf{B^T P} + \mathbf{N^T}) \tag{2.36}$$

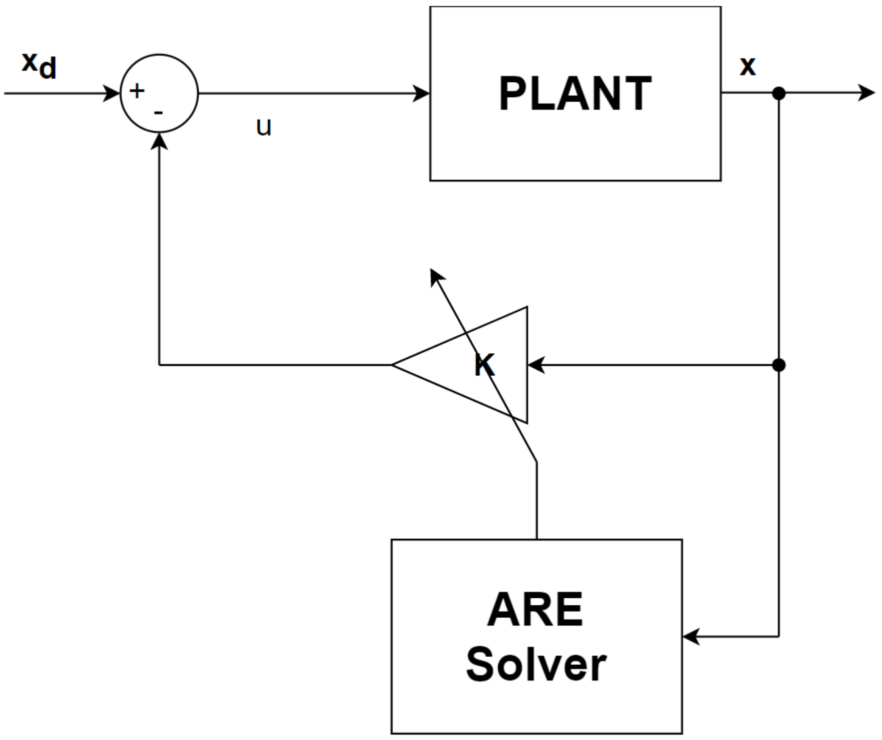


Figure 2.7: **SDRE SS Controller.**

Considering state-depenent case (2.37) it is clear why the optimality does not hold. Solving the Riccatti equation algebraically for non-constant matrices $\mathbf{A}$,$\mathbf{B}$ (or possible for non-constant $\mathbf{Q}$ and $\mathbf{R}$) is very difficult. Some sources approximate solution using Taylor series ([6]), some linearize model at each timestep ([13]), and others ([31]) use the LTI solution and recalculate at each time step. All of these approaches use some kind of approximation of the solution and therefore the solution is sub-optimal.

$$\dot{\mathbf{x}} = \mathbf{A}(\mathbf{x})\mathbf{x} + \mathbf{B}(\mathbf{x})\mathbf{u} \tag{2.37}$$

SDRE control is very easy to implement, is robust and can be tuned by varying the matrices $\mathbf{Q}$ and $\mathbf{R}$. However it comes with a cost of being very computationally demanding on the hardware.

2.3.6 Implementation note

All of the above mentioned algorithms require knowledge of all the true states of the system. To achieve this we use a full-state observer and in this section a short overview of the techniques used in this thesis is provided.

Obtaining the full-state estimation is done by using a full-state observer. In theory there are various types of state observers, most popular being (deterministic) Luenberger observer and (stochastic) Kalman filter. Most of practical applications use the Kalman Filter (KF). In general KF algorithm is very simple and can be used efficiently for many applications such as estimating states and parameters of the system, sensor bias etc [17]. Basic KF works only on LTI systems, however through the years many variants have been developed, which can be easily applied to nonlinear systems - in this thesis we use the Extended Kalman Filter (EKF). This algorithm is exceptionally well described in other sources [17][8] and therefore it will not be described here.

Another issue that all of the above mentioned algorithms face is the issue of trajectory planning. All tracking problems require a trajectory to follow in the state-space. Planning of this trajectory is no easy task and the final trajectory has to comply with several constraints. The system must be able to perform such trajectory (i.e. the dynamics of the trajectory must be slower or equal to the dynamics of the system), the trajectory has to be smooth to avoid step changes in input (which is not usually possible) and it has to be differentiatable as many times as the model requires (n^{th} order model requires at least n derivatives). To solve this problem (within this thesis) n^{th} order IIR filters with numerically calculated derivatives were used to obtain the desired state-space trajectory.

2.4 Parameter estimation

Parameter estimation is a process within the field of system identification, which deals in finding the best parameters of a given system based on the model-data similarities and correlations.

If the given system has the general form 2.38, where β is vector of model parameters, we can define a parameter estimation as finding „the best“ choice of β which will maximize the likeness of mathematical model to the real-world construct we want to model.

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, \beta) \tag{2.38}$$

In most of technical application we can generalize that the task of parameter estimation is to approximate real function $f(\mathbf{x}, \mathbf{u}, \beta)$ by using our measurements z, where z is usually a discreetly measured signal in time containing our state measurements.

Note: Area of parameter estimation and system identification in general is much wider than this simple definition, however it is out of the scope of this thesis to describe the process in detail so this section focuses mainly on procedures used practically in later chapters.

In general the parameters can have various relations to the states of the system, however the simplest case is if the model is linear in parameters. We call a model linear in parameters if we can exactly decompose the model into the form (2.39) where ϕ represents normalized version of the vector function $\mathbf{f}$ which is scaled by parameter vector β.

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \beta) = \beta^{\mathbf{T}}\phi(\mathbf{x}, \mathbf{u})) \tag{2.39}$$

Otherwise we call the model nonlinear in parameters. There are other things to consider as dimensionality reduction and parameter correlations (described in [22]), which can further increase the simplicity of our model.

- **Static estimation**

If our model is linear in parameters the simplest way of solving the minimization problem is to use ordinary least squares ([34]). However this approach requires to measure all of the states at each time step (or we can measure one of the states and calculate higher derivatives).

We define

$$\mathbf{Y} = \mathbf{X}\beta \tag{2.40}$$

where $\mathbf{Y}$ is matrix containing all of the $\dot{\mathbf{z_k}}$ measurements at time k as

$$\mathbf{Y} = \begin{bmatrix} \dot{\mathbf{z}}_\mathbf{1}^\mathbf{T} \\ \dot{\mathbf{z}}_\mathbf{2}^\mathbf{T} \\ \vdots \\ \dot{\mathbf{z}}_\mathbf{n}^\mathbf{T} \end{bmatrix} \tag{2.41}$$

and matrix $\mathbf{X}$ containing all of the results of the matrix function f^* as

$$\mathbf{X} = \begin{bmatrix} \phi(\mathbf{z_1}, \mathbf{u_1})^\mathbf{T} \\ \phi(\mathbf{z_2}, \mathbf{u_2})^\mathbf{T} \\ \vdots \\ \phi(\mathbf{z_n}, \mathbf{u_n})^\mathbf{T} \end{bmatrix} \tag{2.42}$$

Now we can compute β as

$$\beta = (\mathbf{X^T X})^{-\mathbf{1}}\mathbf{X^T Y} \tag{2.43}$$

Advantage of this approach is that it is really easy to implement in MATLAB (single-line command), but on the other hand requires good measurement of states of the system, which does not contain too much noise or bias.

- **Dynamic estimation**

  If our model is nonlinear in parameters or if our state measurement is very noisy we can use dynamic estimation ([24]). This approach is based on using the actual inputs to our system and numerically simulating the model equation (2.38). This gives us response of our system with the current parameters to our input. We then try to find the combination of parameters which minimizes the cost function (e.g. MSE between the model response and measured data), by repeating the simulation over and over with different sets of parameters.

  The advantages of this approach is that we can modify our cost function to only compare one of the states with the measurement (e.g. only position), so we do not need to measure all the states and their derivatives. Also we can find the nonlinear parameters as well as linear ones with this approach. The disadvantage is that it is no longer a single-line calculation but it is an iterative process. Moreover we need to simulate the system at each time step which can take longer time. MATLAB offers a lot of optimization tools in the Optimization toolbox [5] and it is also what we used for this thesis.

2.5 Nonlinear system stability

Several methods have been developed throughout the years to analyze stability of linear and nonlinear systems. For the purpose of this thesis only a basic one will be described, however there exist more advanced and complete methods (e.g. Input-State Stability [3]).

To analyze the stability of nonlinear system we cannot use traditional linear stability theorem. This theorem states that a linear system of the form 2.32 is stable if it satisfies exponential stability (matrix A has negative-real-part eigenvalues) and BIBO stability (bounded input causes a bounded output response) [17].

A simplest step from linear to nonlinear systems is to analyze how nonlinear systems behave around equilibrium points. One way to do this is to analyze the Jacobian matrix around these points [33], other is to pertubate the input and observe the behavior of the system [27]. To find the equilibrium points of the system we need to solve the equation

$$0 = \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{2.44}$$

From equation 2.44 we obtain solutions as groups of vectors $\mathbf{e} = [\mathbf{x_e}, \mathbf{u_e}]$. Now we calculate jacobians at these equilibrium points.

$$\mathbf{J} = \left[\frac{\partial \mathbf{f}(\mathbf{x_e}, \mathbf{u_e})}{\partial \mathbf{x_1}}, \frac{\partial \mathbf{f}(\mathbf{x_e}, \mathbf{u_e})}{\partial \mathbf{x_2}}, \dots, \frac{\partial \mathbf{f}(\mathbf{x_e}, \mathbf{u_e})}{\partial \mathbf{x_n}}\right] = \begin{bmatrix} \frac{\partial f_1(x_e,u_e)}{\partial x_1}, \dots, \frac{\partial f_1(x_e,u_e)}{\partial x_n} \\ \vdots, \ddots, \vdots \\ \frac{\partial f_m(x_e,u_e)}{\partial x_1}, \dots, \frac{\partial f_m(x_e,u_e)}{\partial x_n} \end{bmatrix} \tag{2.45}$$

Now we can analyze how the system behaves around these equilibrium points similarly to linear system. There are different options which are depicted on figure 2.8.

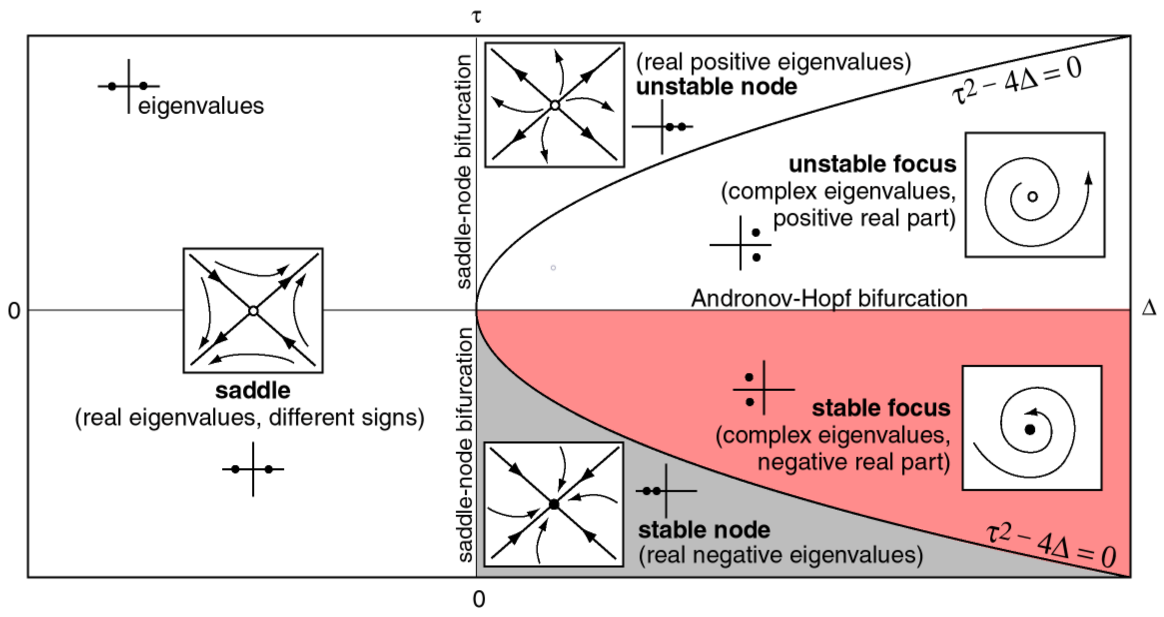

τ
eigenvalues
saddle-node bifurcation
(real positive eigenvalues)
unstable node
τ2 – 4Δ = 0
unstable focus
(complex eigenvalues, positive real part)
0
saddle
(real eigenvalues, different signs)
Andronov-Hopf bifurcation
Δ
saddle-node bifurcation
stable focus
(complex eigenvalues, negative real part)
stable node
(real negative eigenvalues)
τ2 – 4Δ = 0
0

Figure 2.8: **Equillibrium behavior** A different behaviors of the system around equilibrium based on the Jacobian eigenvalues in $\tau - \Delta$ plot (Δ is the determinant of the Jacobian, τ is the trace of the Jacobian) [15]

Note that for some systems there may be no solution to the equation 2.44 and so system like that would have no equilibrium and therefore cannot be asymptotically stable. If such solution exists, we can analyze how the system behaves around said points and for systems with small number of states also visualize it by plotting the vector field $\mathbf{f}(\mathbf{x}, \mathbf{u})$.

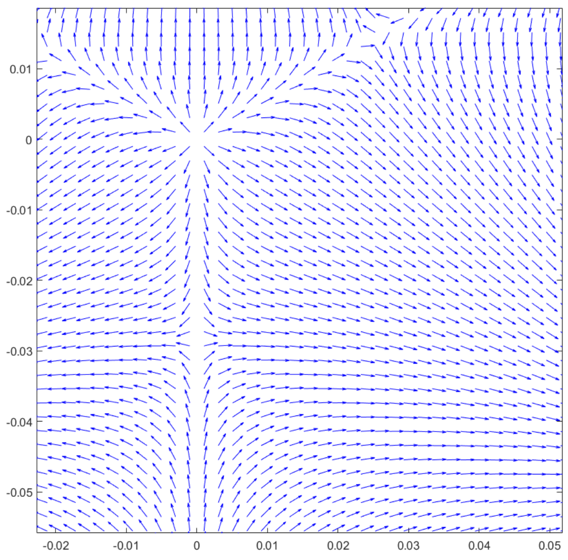


Figure 2.9: **Vector field** A vector field view of the system stability.

2.6 Controllability

2.6.1 Typical approach

Controllability analysis, similar to stability analysis, is a concept which is very well developed for linear systems but much harder to achieve in nonlinear systems. In this subsection a typical approach for linear systems will be presented, which will introduce concepts used for nonlinear cases (in state-dependent form) in later chapters. Typical approach to controllability analysis in linear systems of the form 2.32 is to analyze the controllability matrix. The controllability matrix for a linear nth order system is defined as [8]

$$\mathfrak{C} = [\mathbf{B}, \mathbf{AB}, \mathbf{A^2B}, \mathbf{A^{n-1}B}] \tag{2.46}$$

The system is said to be controllable if the rank of matrix $\mathfrak{C}$ is n. This definition is however not too intuitive and sometimes the results can be misleading since it only tells us the system is controllable but not how controllable.

First the intuitive explanation to how this method works. The matrix $\mathfrak{C}$ is constructed in such a way which allows us to see how the system responds to impulse response. It defines the propagation of input through the system up to the rank n. If the matrix $\mathfrak{C}$ does not have at least n linearly independent columns (does not have rank n) then some of the states are unreachable from the input. However this is not always what is required of our system. In some cases we do not need to reach any state of the system, we only require the unstable dynamics to be controllable/stabilizable, this however is not possible just by analyzing the $\mathfrak{C}$ matrix.

2.6.2 Popov-Belevitch-Hautus controllability test

A different approach to controllability is so called Popov-Belevitch-Hautus test [8]. This test tells us that linear system of n^th order (2.32) is controllable if and only if

$$rank([(\mathbf{A} - \lambda \mathbf{I}), \mathbf{B}]) = n, \forall \lambda \in \mathcal{C} \tag{2.47}$$

Note that the condition 2.47 is true for all λ values which are not eigenvalues. In other words PBH test allows us to test if the system can stabilize a specific eigenvalues of A from which we can determine if the system is stabilizable. It also allows us to determine in which direction the B matrix is deficient to make the system stabilizable or controllable (the B matrix needs to have a component in every eigenvector direction to satisfy the PBH test).

2.6.3 Controllability Gramian

One of the tools to specify the degrees of controllability (whether the system is well controllable or not-so-well controllable) is controllability Gramian [8]. To analyze which states are more or less controllable one must analyze the eigendecomposition of controllability Gramian which is defined for linear system 2.32 as

$$\mathbf{W_c(t)} = \int_0^t e^{\mathbf{A}\tau} \mathbf{B} \mathbf{B^T} e^{\mathbf{A^T}\tau} d\tau \tag{2.48}$$

However as [8] states it is often impractical to compute the Gramian as 2.48 but instead is computed as solution to the Lyapunov equation

$$\mathbf{A W_c} + \mathbf{W_c A^T} + \mathbf{B B^T} = \mathbf{0} \tag{2.49}$$

This is often used but can become a bit computationally expensive. For many applications an approximate solution to the Gramian can be obtained very easily as

$$\mathbf{W_c} \approx \mathbf{W_c^e} = \mathfrak{C}\mathfrak{C}^{\mathbf{T}} \tag{2.50}$$

And the eigendecomposition of $\mathbf{W_c^e}$ reduces to Singular Value Decomposition (SVD) of $\mathfrak{C}$. From this we can obtain Gramian singular vectors. In combination with corresponding singular values these specify an ellipsoid defined in the state-space. This ellipsoid specifies how far can we get per a unit of energy, which tells us which states are more or less controllable.

$$\mathfrak{C} = \mathbf{U \Sigma V^T} \tag{2.51}$$

The columns of unitary matrix $\mathbf{U}$ give us the most controllable directions and matrix of singular values $\mathbf{\Sigma}$ gives us a numerical value for how far can we get in the specified direction per unit energy (as mentioned in the paragraph above).

Chapter 3

Design of mechanical construction and electronics

Based on the research done in chapter 2, a first specification of the device was drafted. The device was to have three coils which would control position of steel ball. The ball could move in two direction, whilst being heavy enough to prevent any slipping or loss of contact with the platform. This means the system would have three actuation inputs for two degrees of freedom making it over-determined, but in theory controllable for any position in between the coils.

The position sensor would have to be chosen with regards to the fact that the ball should be able to move freely above the coils. Also the sensor should not interfere with the magnetic field of the coils or the coils should not interfere with the sensor measurement.

First component around which the whole design was built were the electromagnets. At first we tested the E1AS-0211-24-100 which were after several tests replaced with larger version E1AS-0511-24-100 ([30]). Parameters of the electromagnet can be seen in a table below (3.1).

Parameter	**Value**
Nominal voltage	24 V
Power	6.5 W
Nominal resistance	88 Ω
Max. holding force	750 N
Diameter	50 mm
Height	27 mm

Table 3.1: **Electromagnet parameters**

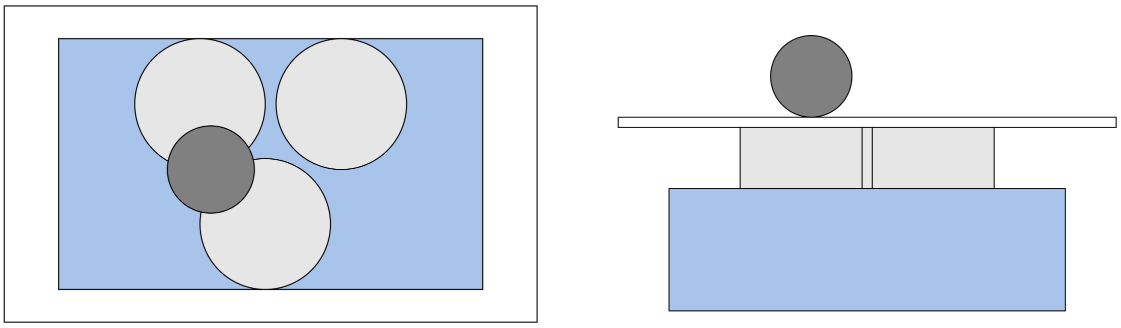


Figure 3.1: **Construction design** Top view(left) Side view(right)

3.1 Position sensor

To perform any kind of positional control a precise measurement of the ball position is required. For one dimensional position measurement there are many different options (Ultrasonic,IR, resistive, laser,...), however in two dimensions (in a plane) the options are more limited. Following sensors were considered:

- **Resistive touch** The resistive touch sensor consists of two separated resistive foils which are deformed by touch and this causes a short between the two. By the change of the resistance, we can measure where this connection occurred. It is very robust and simple solution for a reasonable price, which can also be found in various sizes of the touch panel (from 1 inch to 30 inch in diagonal), making it well suited for our application. However the downside is that it requires some amount of contact force, which means the ball would have to be heavy.

- **Capacitive touch** One of the most common touchscreen sensors is capacitve touch sensor. The operating principle of such sensor is that an insulator (usually glass) is coated with a thin layer of conductive coating (usually Iodium Tin Oxide). After a grounded conductor (such as human body) touches the screen it distorts the electrostatic field of the screen causing a change in capacitance which can be measured and located. The upside of using capacitive touch sensor compared to resistive is that it does not require any contact force to work properly and as well as the resistive touch sensor can be made in any size. But it is unsuitable for our application because the detected object has to be grounded or at least tied to a different potential, which is not a case of a free moving steel ball.

- **Infrared touch** A predecessor to the typical (resistive or capacitive) touchscreen technology was infrared touch screen. The sensor consists of grid of infrared diodes and photo-transistors. After interrupting the grid by and object we can locate which photo-transistors are covered by the object. This allows us to localize where the object is. Very robust solution, but offers very little resolution (usually 5 mm) and rather slow sample time (usually 15 ms). This makes it unsuitable for our application.

- **Camera** One of the more advanced sensing options would be the implementation of motion tracking with a camera. The steel ball would be coated in distinctive paint and image processing algorithm could be implemented to track the ball position. This is a relatively inexpensive solution which can provide good results, however it is also insufficient for our application because of the slow sampling rate of the camera (60 Hz), which could make the control very difficult.

After considering price and performance of the sensors, as well as suitability for our application a resistive touch sensor came out as a best option. The contact force problem would be solved by using a larger steel ball. Resistive touch panel from company Fujitsu NC01152-T10 ([11]) of the „Feather Touch“ line was chosen. This touch panel offers a lot of operating freedom and requires only around 30g of contact force making it well suited for our application.

3.2 Power electronics

3.2.1 Coil drivers

Main concern of the whole design was the uncertainty of the behavior of the system when mutual interaction of magnetic fields occurs. To solve this problem a hardware current controller on each of the coils was required and so a simple circuit was designed containing LM1875 audio power amplifier ([14]) and precise feedback amplifier MCP601 ([19]) to control the current through each of the coils (figure 3.2).

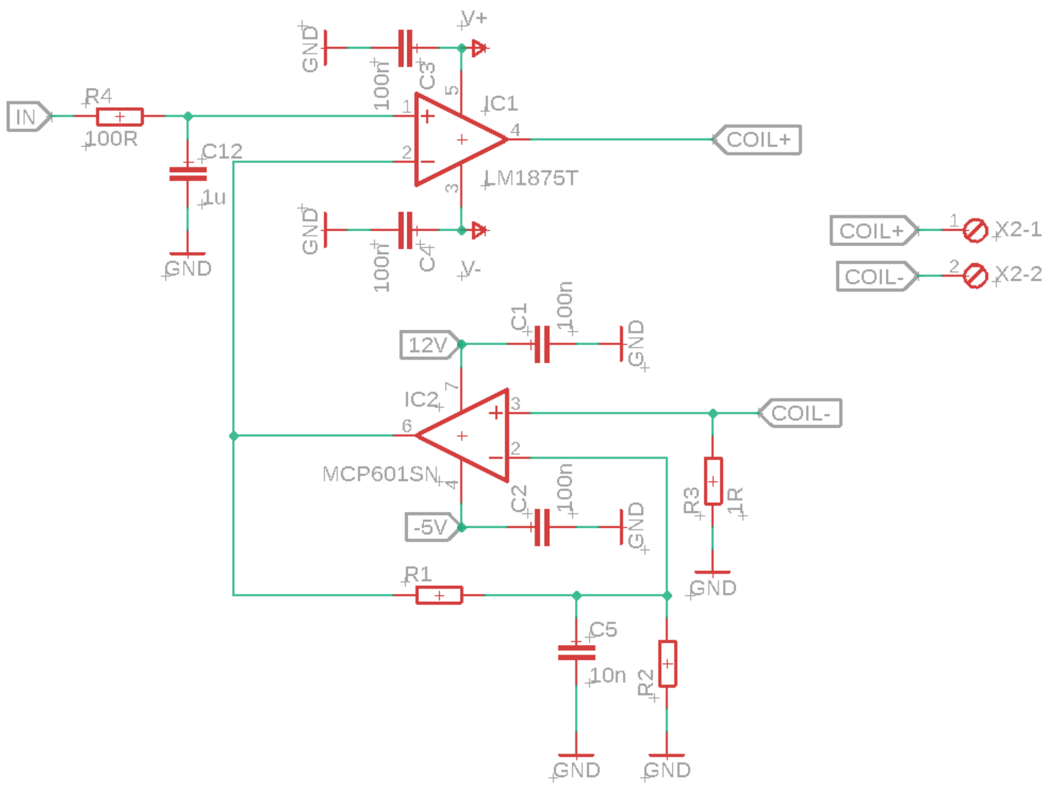


Figure 3.2: **Coil driver**

The transfer loop gain can be tuned by resistors R1 and R2. Resistors change the gain of the MCP601 (the output voltage can go up to 10V). The final transfer loop gain from input voltage to current is as follows

$$I_{out} = \frac{U_{in}}{(1 + \frac{R2}{R1})} \tag{3.1}$$

Time constant of the control loop varies depending on parameters of the coil as well as the power supply. The exact relationship is non trivial however the smaller the time constant of the coil and the higher the voltage the power supply can deliver the faster the response of the control loop.

The step response was measured on the finished device using the oscilloscope and a current probe (the measurement can be seen on figure 3.3). From the measurement we can approximate that the time constant of the current controller is $\tau_c = 0.6ms$, which means that to any discrete controller, which is running with sample time 1 ms or higher, built on top of this one, the change in the current would seem instantaneous.

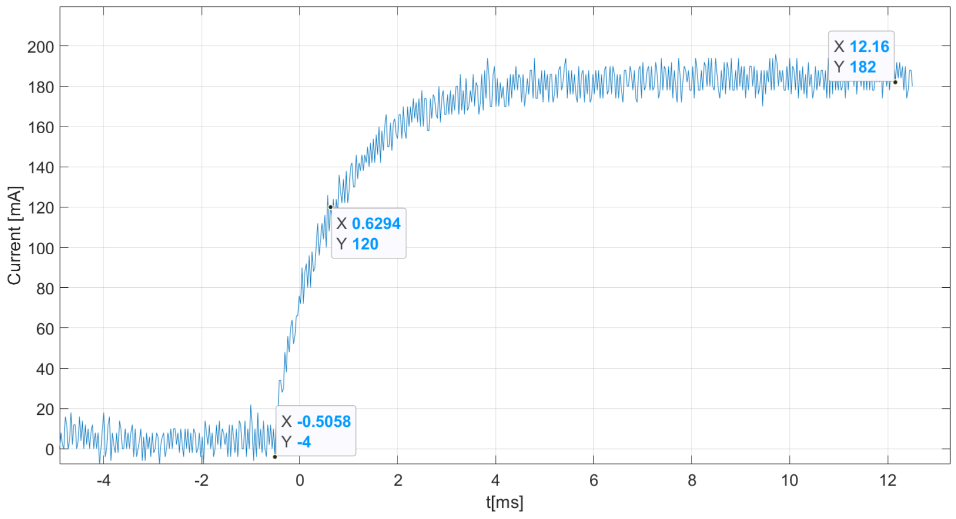


Figure 3.3: **Step current response**

3.3 Sensor signal processing

The selected NC01152-T101 ([11]) touch panel has a standard 4-wire interface for touch panels. This interface provides analog measurements of the x and y positions on the panel, based on the resistance along the x and y directions. To process this signal an IC from Texas Instruments TSC2007 ([35]) was chosen to convert the resistance values into the digital numerical form. The mentioned IC provides communication via the I2C interface and offers up to 12 bits of resolution. This means that the precision in one direction is slightly higher than in the other one (because of the rectangular shape of the touch panel).

To make sure the position signal could be interfaced with any standard hardware the I2C bus could not be the output stage. Therefore a microcontroller was added (dsPIC33FJ128MC804 [20]), which reads the position value on the I2C bus and transmits it forward using the SPI to the MCP4822 dual channel DAC ([21]) which converts it into two analog voltage values. The whole signal chain is shown on figure 3.4. **NOTE: Full schematics are located in appendix A**

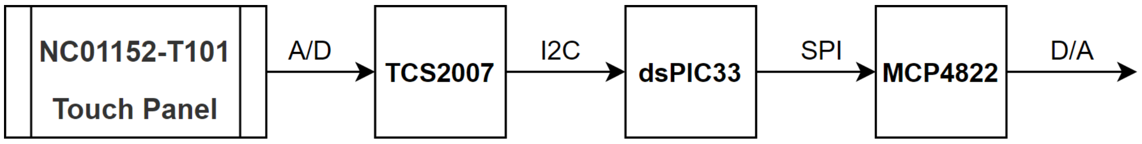


Figure 3.4: **Block diagram - signal processing of touch panel signal**

3.3.1 Power supply

To supply the coil drivers with power, both positive and negative voltage power rails were required. For this purpose 2 standard switched-mode DC power supplies were chosen. MEAN WELL LRS-75-48 and MEAN WELL RS-15-5 were used to create 48V and -5V lines respectively.

Negative voltage power line was needed, because the LM1875 power op-amp is not a rail-to-rail type. This means that it cannot reach 0V on the output with single ended supply. To overcome this limitation a -5V line helps the op-amp to reach zero and also helps demagnetize a magnetic circuit a little bit faster if needed. In the „forward bias“ direction this line also powers a sensor board and a cooling fan which helps to cool the device. The wiring diagram is shown on figure 3.5.

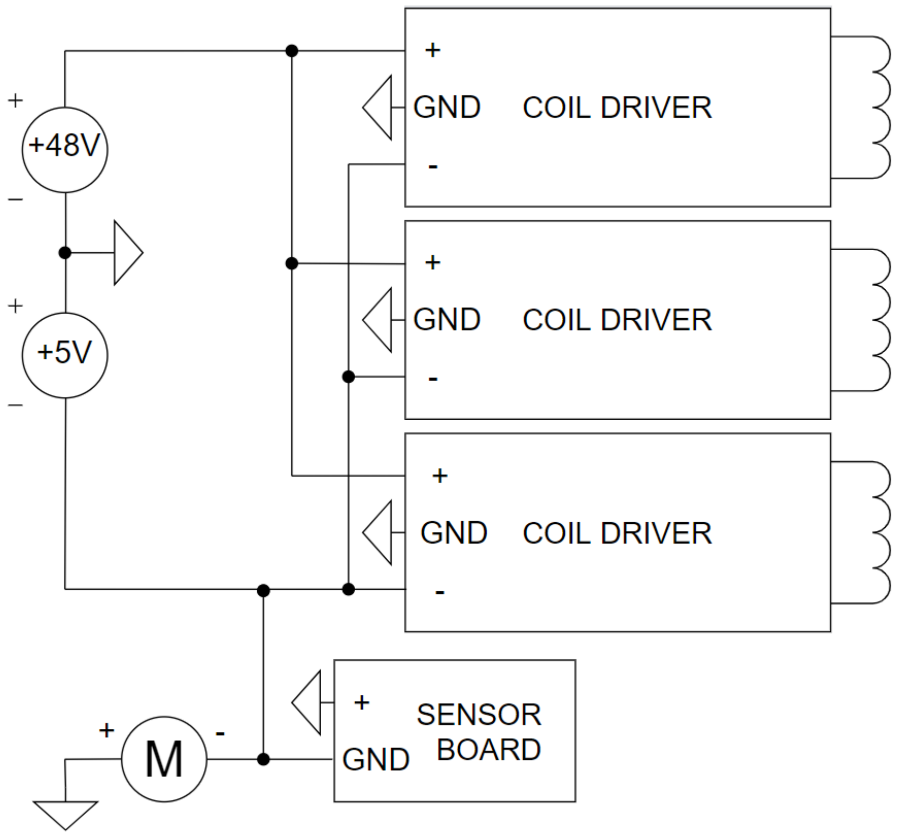


Figure 3.5: **Wiring diagram**

3.4 Final construction

The final construction was built incorporating all the designs from previous sections, however the quality of the final product was significantly affected by the current COVID-19 crisis. Therefore not all of the materials and procedures used would be used under regular circumstances.

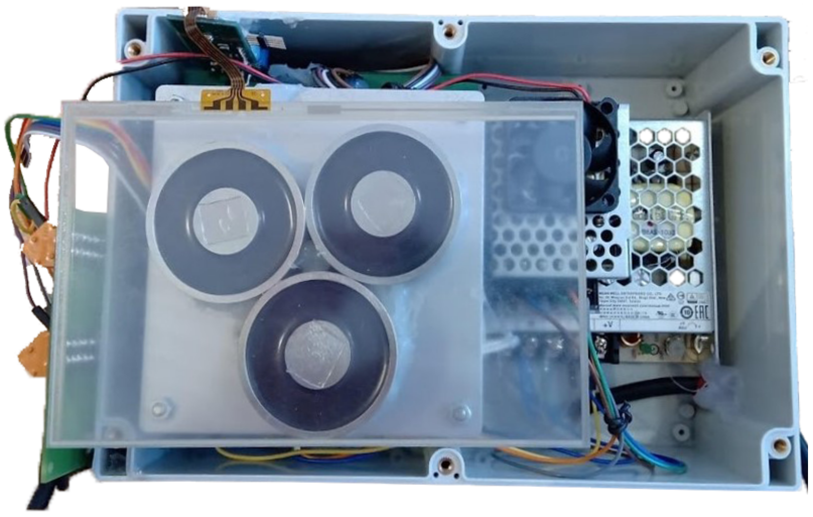


Figure 3.6: **Top view of the device**

The mechanical part of the construction consists of a box in which all of the components were placed and secured with glue. The power electronics were stacked together using metal standoffs with screws. The coils were glued to aluminum plate, which was placed on top of all the power electronics. The touch panel was placed on top of the coils and secured with double sided tape so it could be removed easily if need be. Most of the internal wiring was done using screw terminals and breadboard jumper wires. All of the signal wires (inputs and outputs), were connected to terminal blocks which can be interfaced with any control hardware (in our case MF624 I/O card by company Humusoft). A cooling fan was added to help circulate the air from the bottom of the box to the top for better cooling. The final construction can be seen on figures 3.6 and 3.7.

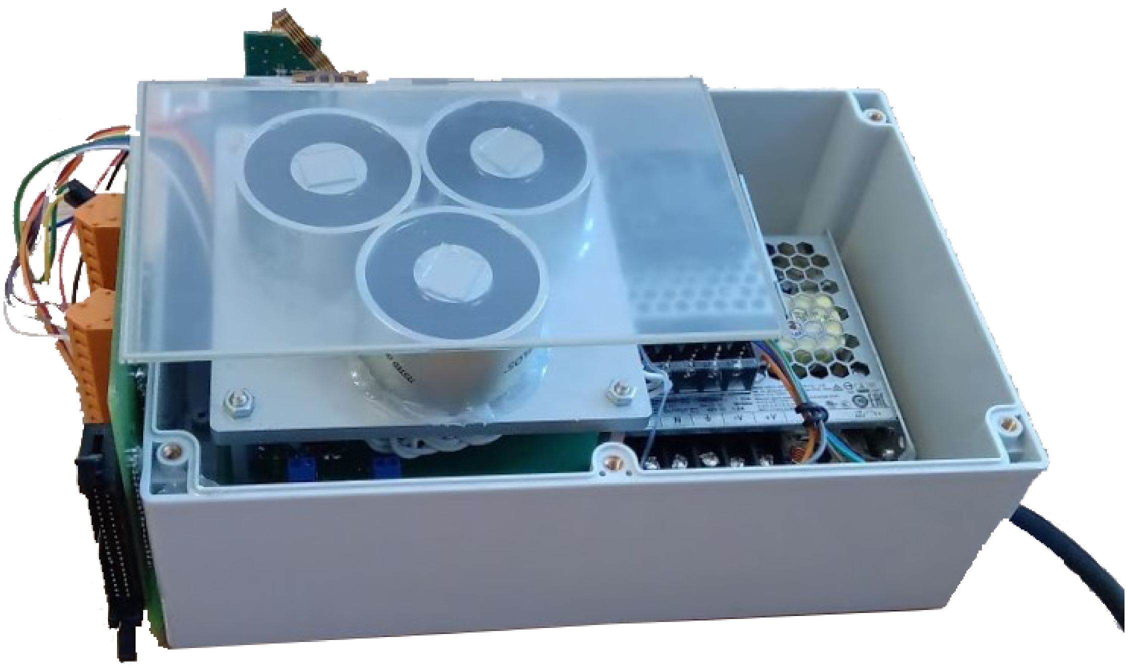


Figure 3.7: **Perspective view of the device**

Chapter 4

Modelling and analysis in simulations

After constructing the device a mathematical model was needed to perform control, state estimation, analysis and test nonlinear control algorithms. This chapter is split into sections which represent the steps of modelling and model validation for the device as well as analysis of the control algorithms on numerical models. All of the following calculations and models were done using MATLAB and Simulink software.

The coordinate system is defined as shown on figure 4.1. The x and y represent position of the ball from the origin.The x_{01}, y_{01} represents position of the first coil, x_{02}, y_{02} position of the second coil, x_{03}, y_{03} position of the third coil. We can now define the state vector as $\mathbf{x} = [x_1, x_2, x_3, x_4]^T$ where $x_1 = x$, $x_2 = \dot{x}$,$x_3 = y$, $x_4 = \dot{y}$.

First step was made to compare how well the models described in section 2.2 fit the data to a numerical model for a single coil. Then the model which fit the data better was chosen and extended for all of the three coils. Then an analysis was performed for goodness of fit, stability and controllability of the model and afterwards several control algorithms were tested.

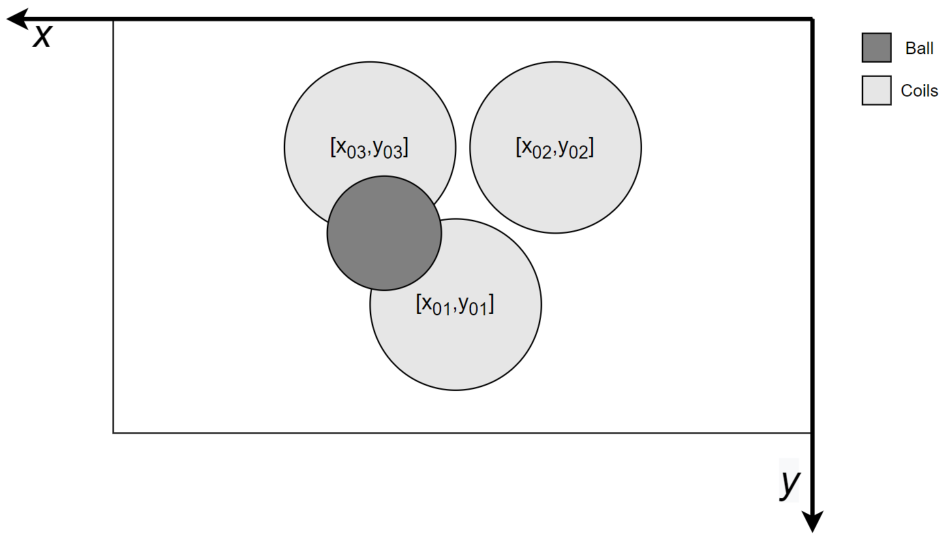


Figure 4.1: **Coordinate system schematic**

4.1 Single solenoid model estimation

In the chapter 2.2 two different models for modelling of the solenoid force on a steel ball were described. To compare them we will construct models of a single coil and compare the two models for the force to see which fits our data better.

To construct the numerical models a data-based modelling approach was chosen mainly because of many uncertainties in mechanical construction, e.g. the construction was not precisely levelled, uncertainty of damping caused by the touch panel (the top foil deforms under the ball as it moves), damping caused by interfering magnetic fields inside the steel ball etc... A Sparse Identification of Nonlinear Dynamics (SINDy) algorithm was chosen to identify the system (the algorithm is more deeply described in [8], in this section only the implementation to our case will be described).

4.1.1 Optimization of linear parameters

Typical SINDy algorithm uses very simple form of static estimation of linear model parameters. We try to approximate the system of the form 4.1 as 4.2,

$$\ddot{\mathbf{x}} = \begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{4.1}$$

$$\ddot{\mathbf{x}} = \theta\xi \tag{4.2}$$

where we specify $\theta(\mathbf{x}, \mathbf{u})$ as 4.3 and ξ represents matrix of linear parameters.

$$\theta(\mathbf{x}, \mathbf{u}) = [\theta_{\mathbf{1}}(\mathbf{x}, \mathbf{u}), \theta_{\mathbf{2}}(\mathbf{x}, \mathbf{u}), \dots, \theta_{\mathbf{m}}(\mathbf{x}, \mathbf{u})] \tag{4.3}$$

where m represents number of nonlinear vector functions by which we want to represent our system. This means we can define many partial functions (in theory infinitely many) and then use only those which have nonzero parameter values associated with them.

More specifically for our case the following set of functions was chosen as seen in eq. 4.4. Assume all as element wise functions on all elements of specified vector element e.g. $X_2 = [x_2(0), x_2(1), \dots, x_2(k)]^T$.

These functions were selected after a longer process of trial and error. Much wider range of functions was tested and afterwards the total set of functions was reduced to only those which were significant to the data. So from a large set of functions we obtained this relatively small subset of functions needed to represent the model.

One of the problems with this approach was that these functions became locally correlated (mainly because of small range of motion of the ball) and this produced multiple „results“ that could represent the measured data. However after many simulations some results became more frequent than others even if we varied the filtering of the data. So this frequent subset was chosen.

$$\theta(\mathbf{x}, \mathbf{u}) = [\mathbf{1}, \mathbf{X_2}, \mathbf{X_4}, \mathbf{F_x}, \mathbf{F_y}] \tag{4.4}$$

We specify the $\mathbf{F_x}$ and $\mathbf{F_y}$ as magnetic forces in the x and y directions. These forces will be calculated based on equations 2.3, 2.4 and 2.9, 2.10, depending on which model we would like to use. Note that the models defined in these equations expect the coil to be placed in the origin so to connect the magnetic force model with our coordinate system we need to shift the coil position in the model by the position vector $[x_{01}, y_{01}]$ in this case.

To obtain the model, first a series of measurements in the open loop of a behaviour of the ball around the first coil with various input currents was performed. The ball was left to oscillate around the coil from various initial conditions. The measured data can be seen on figure 4.2 (in the XY plane is the data plotted in figure 4.3).

NOTE: All of the data measured in this thesis was sampled at 1 kHz.

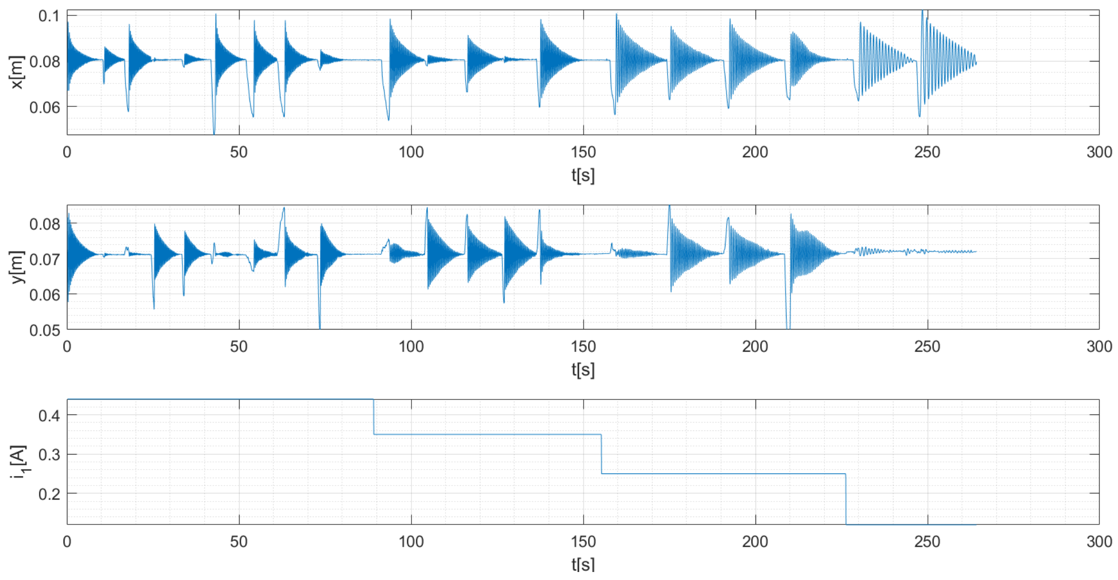


Figure 4.2: **Measured data for single-coil estimation**

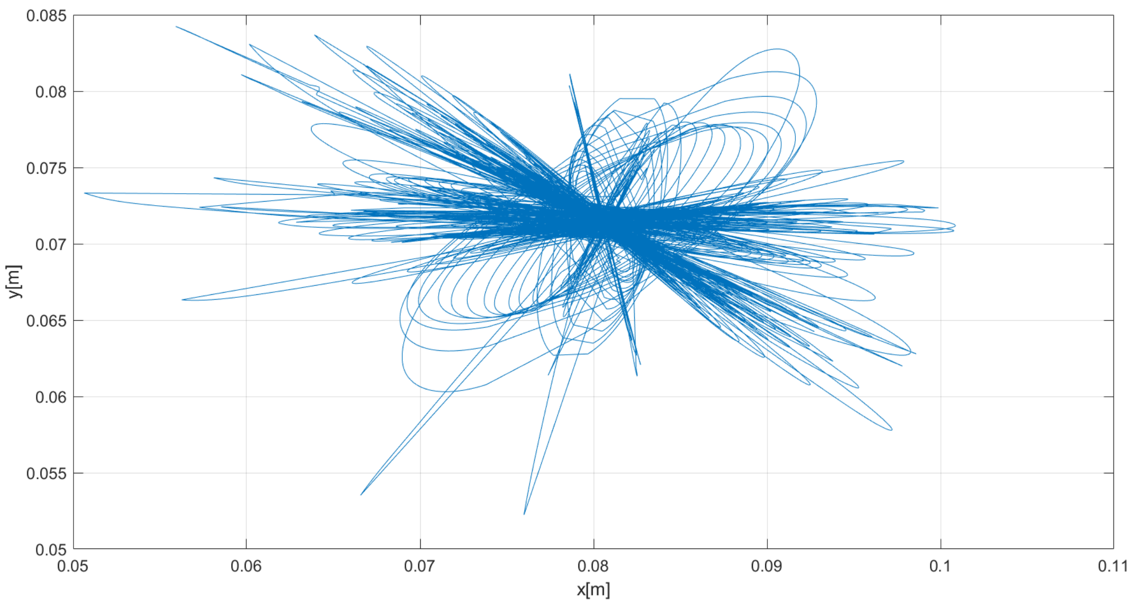


Figure 4.3: **Measured data for single-coil estimation in XY plane**

To compute the derivatives, the data was filtered with a simple two-way zero-phase moving average finite impulse response filter with the window length of 15 elements. This distorts the original data, but considering how slow is the change in position compared to the size of the window, the effects of the filter should be negligible. The derivatives were computed numerically and vectors $\dot{x}$,$\dot{y}$,$\ddot{x}$,$\ddot{y}$ were obtained.

Now the matrix θ can be computed. To obtain the matrix of linear parameters ξ we would normally invert the matrix θ in eq. 4.2 and multiply by it from the left, but since the matrix is not square we need to use the Moore-Penrose pseudo inverse and so we obtain the result as eq. 4.5.

$$\xi = \theta^{\dagger}\ddot{\mathbf{x}}^{\mathbf{T}} \tag{4.5}$$

To obtain the final model, we need to consider the optimization of nonlinear parameters which is described in the following section.

4.1.2 Optimization of nonlinear parameters

The problem with our definition of the magnetic force is that we expect to know the z distance from the center of magnetic attractor. This distance cannot be precisely measured and therefore must be a nonlinear parameter to be optimized. To increase tun-ability of the model we added more parameters to each of the models to obtain the following expressions for the magnetic force. In case of induced dipole model (equations 2.3 and 2.4):

$$F_x = -\frac{i^{(p_1)}cx}{(p_2x^2 + p_3y^2 + p_4)^3} \tag{4.6}$$

$$F_y = -\frac{i^{(p_1)}cy}{(p_2x^2 + p_3y^2 + p_4)^3} \tag{4.7}$$

and in case of magnetic coenergy model(equations 2.9 and 2.10):

$$F_x = -\frac{i^{(p_1)}cx}{(p_2x^2 + p_3y^2 + p_4)^(3/2)} \tag{4.8}$$

$$F_y = -\frac{i^{(p_1)}cy}{(p_2x^2 + p_3y^2 + p_4)^(3/2)} \tag{4.9}$$

where $p1...p4$ represent set of nonlinear parameters different for each of the models. Finally we need to offset the coil centers in these models obtaining final equations for our magnetic forces in the x and y directions.

Induced dipole model:

$$F_x = -\frac{i^{(p_1)}c(x - x_{01})}{(p_2(x - x_{01})^2 + p_3(y - y_{01})^2 + p_4)^3} \tag{4.10}$$

$$F_y = -\frac{i^{(p_1)}c(y - y_{01})}{(p_2(x - x_{01})^2 + p_3(y - y_{01})^2 + p_4)^3} \tag{4.11}$$

Magnetic coenergy model:

$$F_x = -\frac{i^{(p_1)}c(x - x_{01})}{(p_2(x - x_{01})^2 + p_3(y - y_{01})^2 + p_4)^(3/2)} \tag{4.12}$$

$$F_y = -\frac{i^{(p_1)}c(y - y_{01})}{(p_2(x - x_{01})^2 + p_3(y - y_{01})^2 + p_4)^(3/2)} \tag{4.13}$$

Now we can combine these equations with our SINDy algorithm. And apply nonlinear optimization to it. We then define the optimization problem of finding the parameters which best fit our data as

$$\arg\min_{\mathbf{p}\in\mathbb{R}}(\ddot{\mathbf{x}} - \theta\theta^{\dagger}\ddot{\mathbf{x}}^{\mathbf{T}})^2 \tag{4.14}$$

where $\ddot{\mathbf{x}}$ represents computed second derivatives of measured data, $\mathbf{p}$ is vector of non-linear parameters and $\theta(\mathbf{x}, \mathbf{u}, \mathbf{p})$ is defined as eq. 4.4.

This approach combines the static estimation with nonlinear optimization methods. At each iteration of the nonlinear estimation algorithm a new set of linear parameters is calculated. This means that at each iteration we have linear parameters which minimize the cost function and we try to find the nonlinear paramters which minimize it further.

The implementation of a nonlinear optimization algorithm was done using the *MATLAB Optimization Toolbox* from which an implementation of simplex algorithm was used - the function *fminsearch* was used to find the best values of $\mathbf{p}$.

4.1.3 Comparison between the models

A comparison between the models can be seen in figure 4.4. From this we can conclude that with estimated parameters the models behave very similarly and it makes almost no difference which one will be used. Also we can conclude that the models get less accurate further away from the coil.

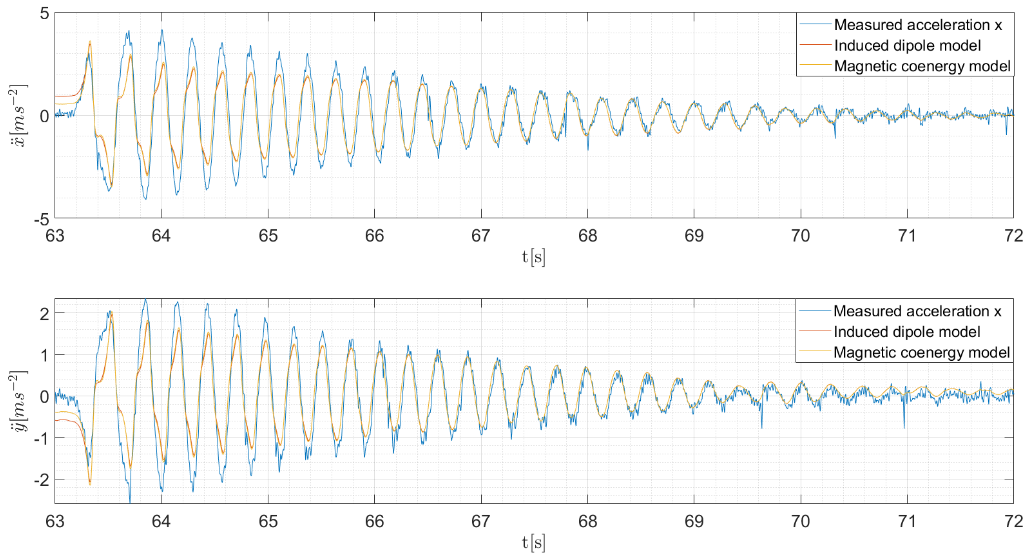


Figure 4.4: **Comparison of the single-coil models**

However to quantify which model is better we need to look at the RMSE of the models with the calculated derivatives from measurements (Table 4.1).

Even though the results are inconclusive, we will use the Induced dipole moment. One reason being the smaller RMSE, but the main reason is that it consists of polynomial non-linearity in denominator of the magnetic force equations (4.10,4.11). This is much simpler to model and calculate, than using a non-integer denominator exponent.

Model	RMSE value $[ms^{-2}]$
Induced dipole	0.5965
Magnetic coenergy	0.6095

Table 4.1: **RMSE of single-coil models**

4.1.4 Properties of the chosen model

The parameters of the chosen model are listed in equations (4.15, 4.16) below.

$$\xi = \begin{bmatrix} -0.1965 & -0.0071 \\ 5.8860 & 0.0221 \\ -0.3454 & 7.7397 \\ -1.63 \cdot 10^{-06} & 3.32 \cdot 10^{-8} \\ 1.47 \cdot 10^{-07} & -1.64 \cdot 10^{-6} \end{bmatrix} \tag{4.15}$$

$$p = [1.1492, 1.9370, 2.440, 0.0011]^T \tag{4.16}$$

To visualize how the model behaves around the coil we plot the magnitude of the vector $\ddot{\mathbf{x}}$ as a scalar value for every point in the xy plane at zero velocity of the ball $(\dot{x}, \dot{y} = 0)$.

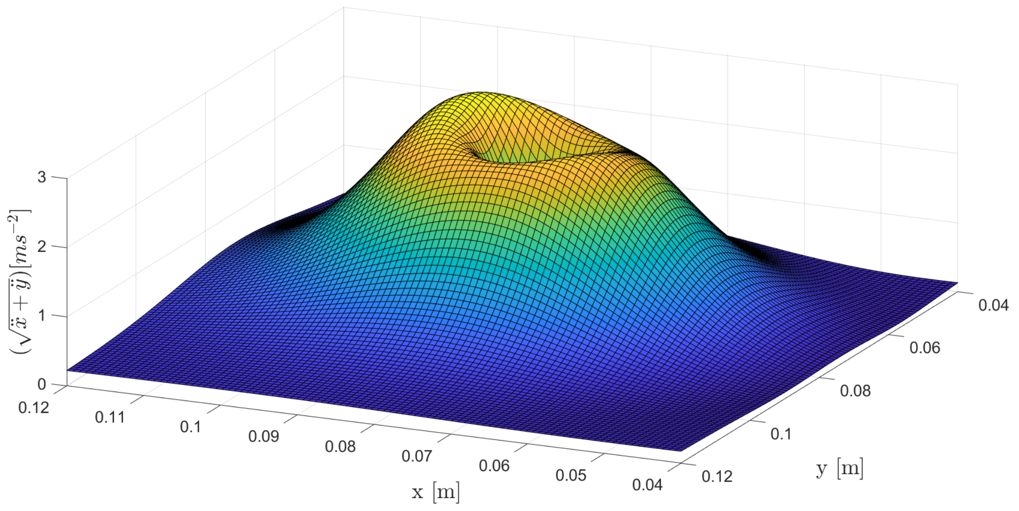


Figure 4.5: **Acceleration magnitude around a single coil**

At the center point of the coil the acceleration is non zero even though the magnetic force is. The acceleration at this point according to model is approx. $0.1996 ms^{-2}$. This is caused because the platform is not precisely leveled and gravitational force causes the ball to roll „downhill“.

An interesting phenomenon appears, when we plot controllability of our velocities as a magnitude vector at each point. We rewrite our system to state dependent form (eq. 2.37) and approximate controllability Gramian at each point using the controllability matrix as described by equation 2.50. Now if we perform an economy SVD we obtain the most controllable directions in the state space which is four dimensional. But since we control only the velocities and not positions in this system we separate these vector elements from each of the four vectors. Now we have a 2 by 4 matrix which contains the numerical approximation of how controllable are velocities in the most controllable directions. We calculate the Frobenius norm of this matrix to numerically approximate how controllable is the system at each point. This is shown on figure 4.6.

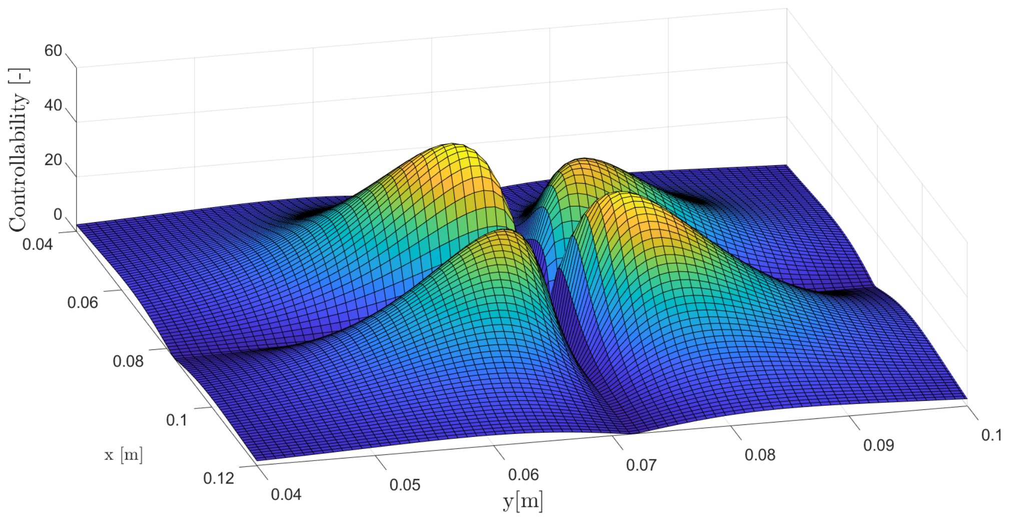


Figure 4.6: **Controllability magnitude around a single coil**

We see that in the x and y directions from the coil center the controllability is lower than in the quadrants between them. This makes logical sense since the coil can only pull the ball towards the center so it is not possible to control the normal velocity to the direction of the pull. In theory the controllability should be lost at these points, but the imperfections of the platform design such as the tilt of the platform cause these points to remain controllable.

4.2 Numerical modelling of the device

The modelling of the complete device with all three coils was similar to the process described in previous section (sec.4.3).

First a new data was measured, with all three active coils, then a new θ matrix was constructed and based this a new nonlinear optimization algorithm was performed.

The measured data in the xy plane can be seen in figure 4.7. The positions of the coils are clearly visible. This demonstrates the behaviour of the system, the stable attractors are at the center points of the coils whilst all the other points are unstable. This will be demonstrated numerically later in this chapter. The unstable behaviour of the system prevents from measuring more data in between the coils (which would be beneficial for our estimation).

The new theta matrix is defined as:

$$\theta(\mathbf{x}, \mathbf{u}) = [\mathbf{1}, \mathbf{X_2}, \mathbf{X_4}, \mathbf{F_{x1}}, \mathbf{F_{y1}}, \mathbf{F_{x2}}, \mathbf{F_{y2}}, \mathbf{F_{x3}}, \mathbf{F_{y3}}] \tag{4.17}$$

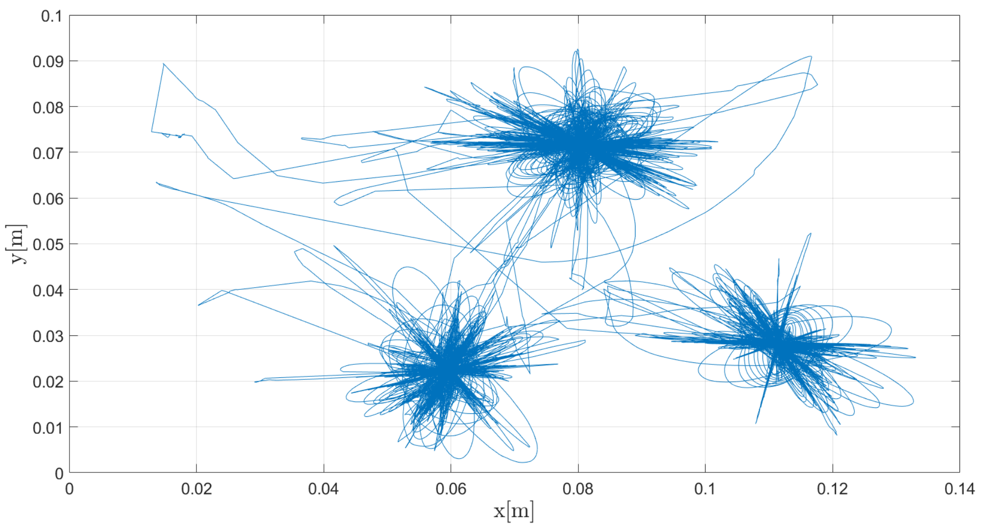


Figure 4.7: **Measured data for parameter estimation**

where $\mathbf{F_{x1}}, ..., \mathbf{F_{y3}}$ perform element wise functions of the following forces (same as in the previous section) $F_{x1}, ..., F_{y3}$. These forces represent forces from three of the coils and are defined as (also see fig. 4.1):

$$F_{x1} = -\frac{i^{(p_1)}c(x - x_{01})}{(p_2(x - x_{01})^2 + p_3(y - y_{01})^2 + p_4)^3} \tag{4.18}$$

$$F_{y1} = -\frac{i^{(p_1)}c(y - y_{01})}{(p_2(x - x_{01})^2 + p_3(y - y_{01})^2 + p_4)^3} \tag{4.19}$$

$$F_{x2} = -\frac{i^{(p_1)}c(x - x_{02})}{(p_2(x - x_{02})^2 + p_3(y - y_{02})^2 + p_4)^3} \tag{4.20}$$

$$F_{y2} = -\frac{i^{(p_1)}c(y - y_{02})}{(p_2(x - x_{02})^2 + p_3(y - y_{02})^2 + p_4)^3} \tag{4.21}$$

$$F_{x3} = -\frac{i^{(p_1)}c(x - x_{03})}{(p_2(x - x_{03})^2 + p_3(y - y_{03})^2 + p_4)^3} \tag{4.22}$$

$$F_{y3} = -\frac{i^{(p_1)}c(y - y_{03})}{(p_2(x - x_{03})^2 + p_3(y - y_{01})^2 + p_4)^3} \tag{4.23}$$

The parameter estimation algorithm was applied in the same manner as in the previous section and the following sets of parameters were obtained.

$$\xi = \begin{bmatrix} -0.1059 & -0.0283 \\ 7.4123 & 0.1703 \\ 0.1628 & 7.4990 \\ -4.05 \cdot 10^{-5} & 2.8 \cdot 10^{-7} \\ 4.16 \cdot 10^{-7} & -3.94 \cdot 10^{-5} \\ -3.99 \cdot 10^{-5} & -3.1815 \cdot 10^{-7} \\ -3.03 \cdot 10^{-7} & -4.10 \cdot 10^{-5} \\ -4.05 \cdot 10^{-5} & -1.0759 \cdot 10^{-7} \\ -4.54 \cdot 10^{-7} & -4.06 \cdot 10^{-5} \end{bmatrix} \tag{4.24}$$

$$p = [1.2454, 5.4084, 5.0827, 0.0036]^T \tag{4.25}$$

This resulted in the fitted data shown on figure 4.8. This figure represents just a section not the whole signal, which is much longer (260 seconds).

Note that we assume that the current can be positive or negative in this context. However both of them cause a pulling force on the steel ball and to keep things simple we assume that the current is semidefinetly positive and don't account for the cases where this is not true.

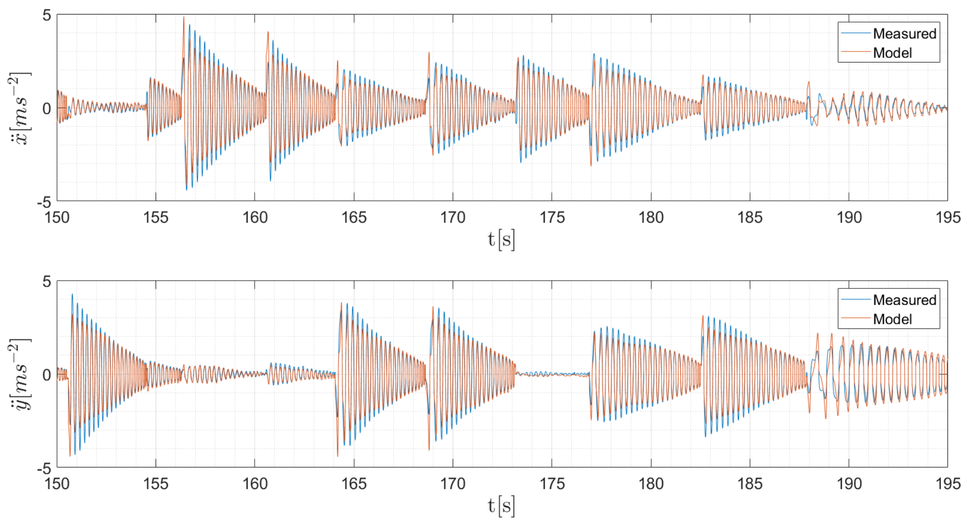


Figure 4.8: **Model data fit**

Now we can plot a similar plot to the figure 4.5, but this time for the complete device (all three of the coils).

4.2.1 Stability

To analyze stability we will use the approach of visualizing the vector field, because solving for the equilibrium points with such a complicated system could prove to be difficult and with the number of states it is still possible to visualize this with a vector field. We assume zero velocities and for each point in the xy plane we plot the resulting acceleration vector.

We need to consider different cases. One with single coil active, with two coils active, with three coils active, with no coils active and with different currents running through each of the coils.

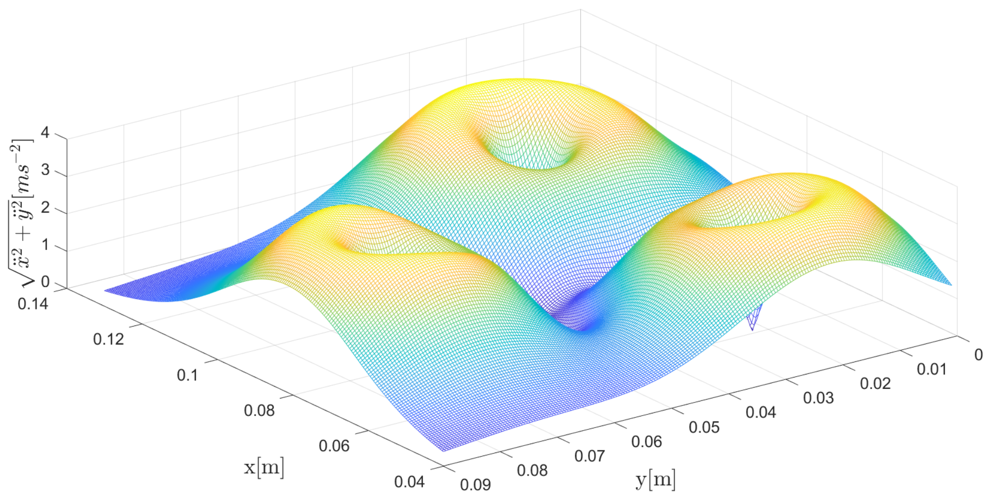


Figure 4.9: **Acceleration magnitude - 3 coils**

First let us assume the case with no coils active. The resulting vector field is shown on figure 4.10. We see that there are no equilibrium points present and in the whole plane the ball accelerates in one direction due to gravitational pull on a slightly tilted platform (note that the vectors in the figure are normalized and their length does not represent the true length of the vectors).

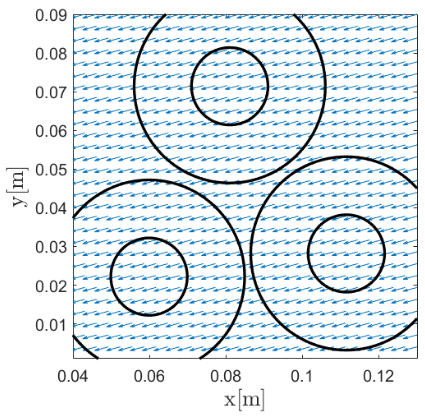


Figure 4.10: **Vector field - 0 active coils**

If we activate one of the coils a single attraction point occurs. The equilibrium point is at the center of the active coil. If we then add another active coil two new equilibrium points are formed. Two are stable at the centers of the coils and one is unstable in between the coils (4.11).

Finally if we add the third active coil another equilibrium point appears. The three attractive points are at the center of each coil and the fourth (unstable) is in between the coils (4.12). By varying the current between the coils we can vary the strength of the attractors as well as the position of the unstable equilibrium point. Thanks to this

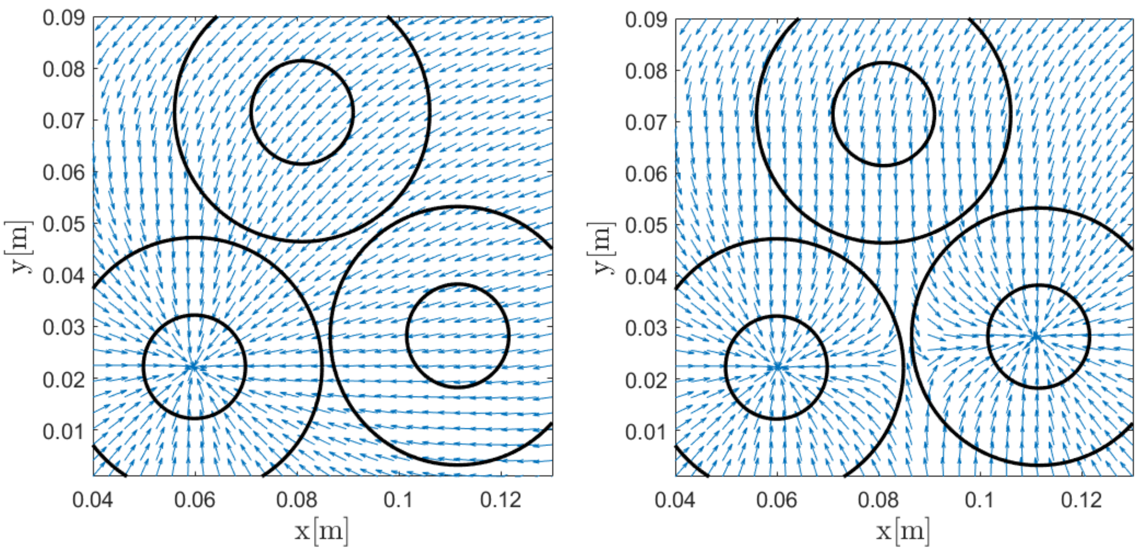


Figure 4.11: **Vector field - 1 and 2 active coils** Left - 1 active coil, Right - 2 active coils

phenomenon we can actually control the position of the ball by keeping in on a plateau formed by the unstable equilibrium point.

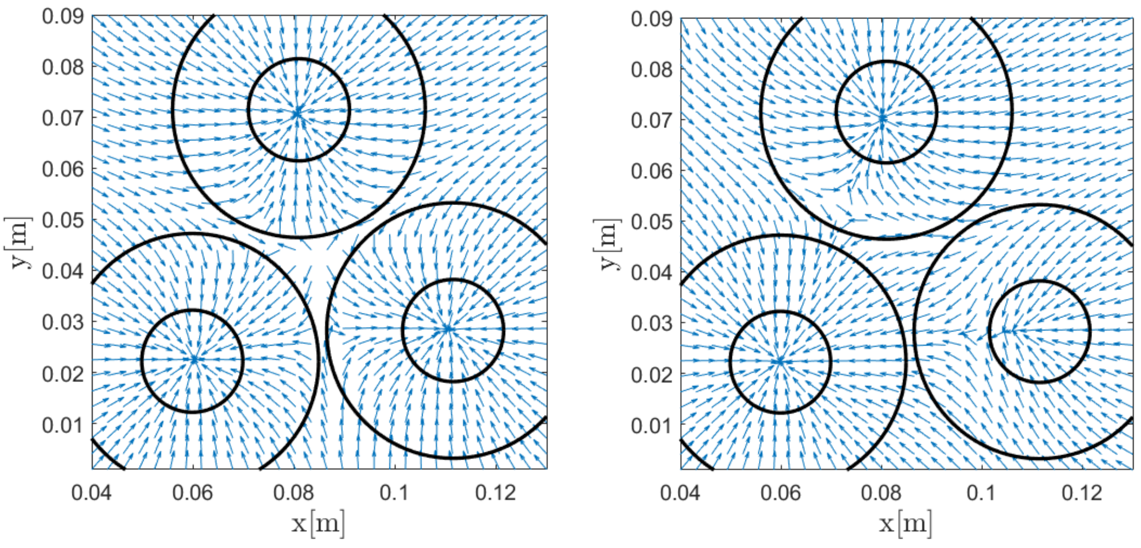


Figure 4.12: **Vector field - 3 active coils** Left - same current, Right - different currents

Please note that in this analysis we purposely omitted the effect of the velocities on stability because this would make the problem four dimensional and hard to analyze. Intuitively we can predict that if the ball is moving at certain velocity, it can gain enough momentum that the coil force would be too weak to stop it (assuming we have limits on the current through the coil). So this would create somewhat of a concept for terminal velocity at each point of the xy plane.

4.2.2 Controllability

To analyze controllability we will use similar technique we used with the single coil controllability analysis. We will obtain controllability Gramian at each point in space with zero velocities using the controllability matrix calculated from state-dependent state space model.

The state-dependent state-space equations were derived as

$$\dot{\mathbf{x}} = \mathbf{A}(\mathbf{x}) \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \mathbf{B}(\mathbf{x}, \mathbf{u}) \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \tag{4.26}$$

where we assume $\mathbf{y} = \mathbf{x}$, $u_1 = i_1^{(p_1)}$,$u_2 = i_2^{(p_1)}$,$u_3 = i_3^{(p_1)}$ and matrices $\mathbf{A}(\mathbf{x})$ and $\mathbf{B}(\mathbf{x}, \mathbf{u})$ are defined as (see also eq. 4.18...4.23)

$$\mathbf{A}(\mathbf{x}) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{\xi_{11}}{x_1} & \xi_{21} & 0 & \xi_{31} \\ 0 & 0 & 0 & 1 \\ 0 & \xi_{22} & \frac{\xi_{12}}{x_3} & \xi_{32} \end{bmatrix} \tag{4.27}$$

$$\mathbf{B}(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} 0 & 0 & 0 \\ \frac{(\xi_{41}F_{x1}+\xi 51F_{y1})}{u_1} & \frac{(\xi_{61}F_{x2}+\xi 71F_{y2})}{u_2} & \frac{(\xi_{81}F_{x3}+\xi 91F_{y3})}{u_3} \\ 0 & 0 & 0 \\ \frac{(\xi_{42}F_{x1}+\xi 52F_{y1})}{u_1} & \frac{(\xi_{62}F_{x2}+\xi 72F_{y2})}{u_2} & \frac{(\xi_{82}F_{x3}+\xi 92F_{y3})}{u_3} \end{bmatrix} \tag{4.28}$$

Note: These equations are practically the same as for the single coil case (for single coil case u_2 and u_3 are zero). This is why these were purposely omitted in the previous section.

Now using these equations we can derive the controllability matrix and controllability Gramian as defined by eq. 2.50. Now we compute the SVD of the controllability matrix at each point and so we obtain the most controllable directions at each point. Similarly to what we have done in previous section with the single coil we now separate the controllability of velocities and norm the value to obtain the following figure 4.13.

We see that the most controllable spot is where the ball is in the center between the coils and the least controllable spots are at the center points of the coils. The centers of the coils form a triangle of low controllability areas and so the expected behavior is that the control algorithms may become unstable in these areas.

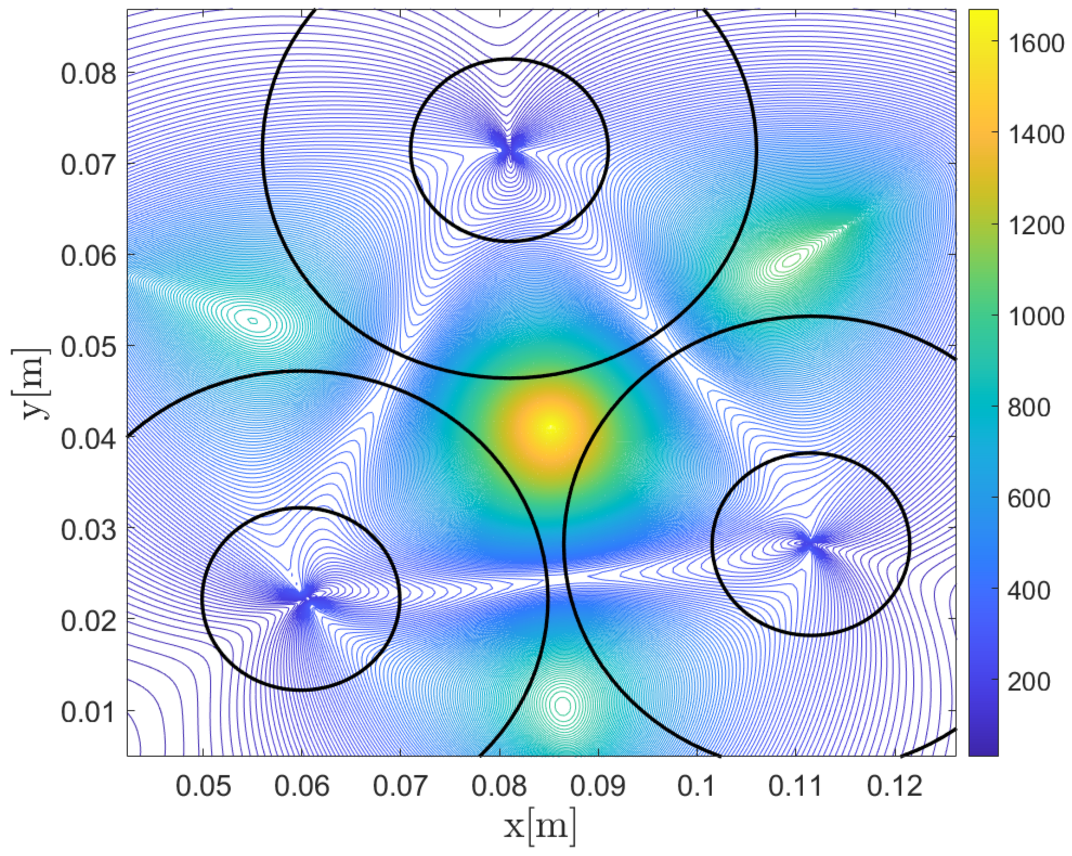


Figure 4.13: **Controllability of the the completed device**

4.3 Analysis of control algorithms using numerical model

In the following subsections, three nonlinear control algorithms will be tested on a numerical model of the system and their final performance compared at the end. First the implementation of each algorithm is described and afterwards a series of tests will be performed. To evaluate the performance of the algorithms two separate criteria will be compared.

First we will compare how much computational time they require to finish. To make this comparison fair we will not be taking the absolute time of the computation because this strongly depends on used hardware, ODE solver and other factors which are not deterministic. Instead we will perform 10 simulations of the system following prescribed trajectory (all of them will be tuned so they can follow the trajectory closely) using each of the algorithms and then normalize the times using one of the algorithms and comparing proportionally how much faster/slower were others.

Second we will compare their stability with respect to disturbances. A zero-mean Gaussian noise will be inserted into the state vector of the system, to simulate real-world signal noise. The variance of the noise will be increasing in the magnitude and the MSE between the prescribed trajectory and the real one will be calculated. Lower the MSE with injected noise the more stable the algorithm is. Note that the other form of disturbance can occur and this is due to parameter uncertainty (our model used for control does not fully represent the system), however since not all of the algorithms are model based, comparison between them using this type of disturbance would not be fair.

The trajectory on which we will be testing the algorithms is depicted in figure 4.14. The trajectory was chosen to be Lipschitz continuous function so it can be differentiated many times and also spans the space in both directions x and y continuously.

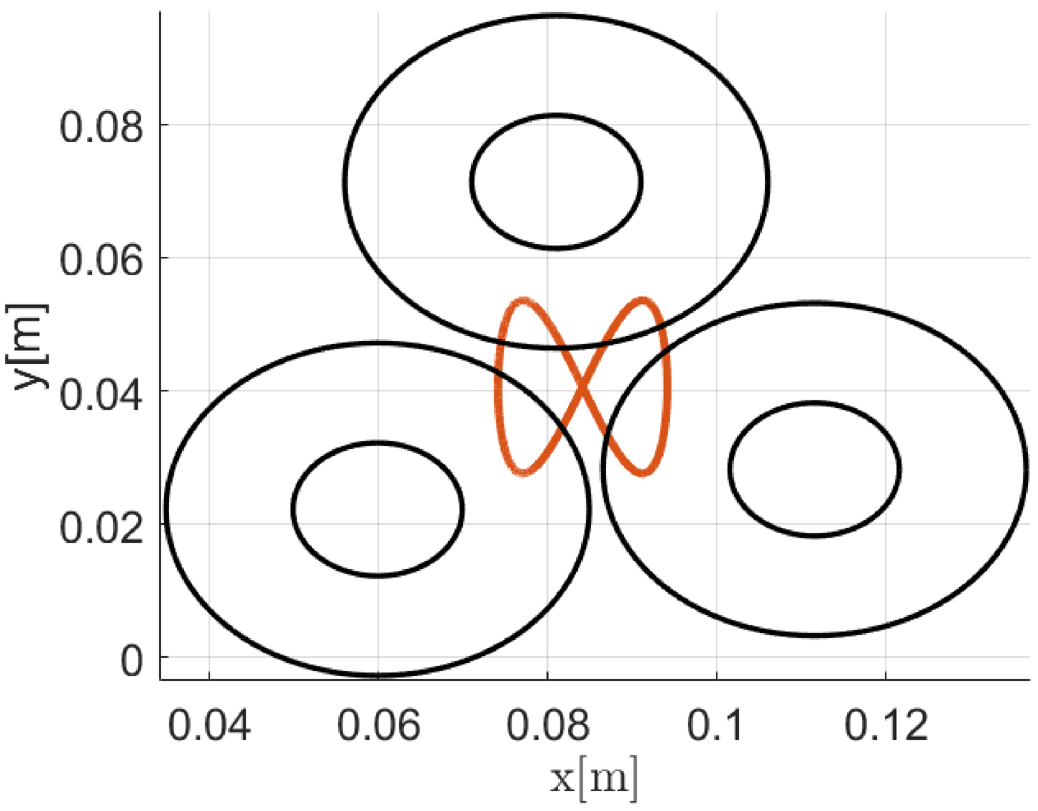


Figure 4.14: **Trajectory for testing of the nonlinear control algorithms**

4.3.1 PID

First algorithm tested is the classical PID controller with coordinate transformation to accommodate for the overdeterminity of the system. This is caused by controlling only two states (positions x and y - the same two PID controllers will be used for the x and y directions.) using three actuation inputs. The PID controllers create two virtual currents u_x and u_y which are then transformed into the real currents u_1,u_2 and u_3. So given a transformation matrix

$$\mathbf{T}(\mathbf{x}) = \begin{bmatrix} \frac{(x-x_{01})}{\sqrt{(x-x_{01})^2+(y-y_{01})^2}} & \frac{(x-x_{02})}{\sqrt{(x-x_{02})^2+(y-y_{02})^2}} & \frac{(x-x_{03})}{\sqrt{(x-x_{03})^2+(y-y_{03})^2}} \\ \frac{(y-y_{01})}{\sqrt{(x-x_{01})^2+(y-y_{01})^2}} & \frac{(y-y_{02})}{\sqrt{(x-x_{02})^2+(y-y_{02})^2}} & \frac{(y-y_{03})}{\sqrt{(x-x_{03})^2+(y-y_{03})^2}} \end{bmatrix} \tag{4.29}$$

we can compute the **u** vector using the Moore-Penrose pseudo-inverse as

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = T^{\dagger} \begin{bmatrix} u_x \\ u_y \end{bmatrix} \tag{4.30}$$

The final PID block diagram can be seen in the diagram 4.15.

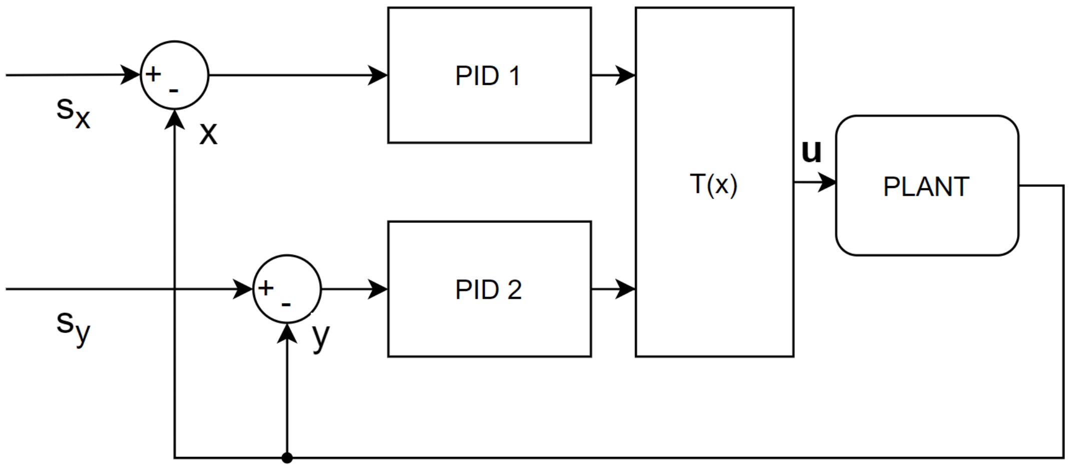


Figure 4.15: **PID block diagram**

4.3.2 Input-State feedback linearization

The feedback linearization algorithm was implemented using the state dependent description of the system (eq. 4.26). At each time step a linearizing input vector is computed using the following equation

$$\mathbf{u} = \mathbf{B}^{\dagger}(-\mathbf{A}(\mathbf{x})\mathbf{x} + \mathbf{v}) \tag{4.31}$$

$\mathbf{v}$ is calculated using the standard LQR controller. To design this controller we need to specify the remaining linearized system which is of the form eq. 4.32

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{u} \tag{4.32}$$

where we label the new $\mathbf{A}$ matrix as $\mathbf{A_{FBL}}$ and the new $\mathbf{B}$ matrix as $\mathbf{B_{FBL}}$.

We see that one of the biggest downsides of this algorithm will be the uncertainty of the model. If the linearization is not precise this system will not be able to compensate for the nonlinearity because the equation 4.31 would not cancel out the nonlinear dynamics and the control may become unstable.

For the system described by 4.27 we design a typical LQR controller with constant K matrix and the linear input $\mathbf{v}$ is then calculated as

$$\mathbf{v} = -\mathbf{K}\mathbf{x} + \mathbf{N}\mathbf{s} \tag{4.33}$$

where $\mathbf{s}$ is a 4 by 1 vector containing the prescribed trajcetory at each point and N is input scaling matrix calculated as

$$\mathbf{N} = \mathbf{B}^{\dagger}(\mathbf{A_{FBL}} - \mathbf{B_{FBL}K}) \tag{4.34}$$

This is the typical inverse of the dc gain of the closed-loop system (assuming the $\mathbf{C}$ to by identity matrix).

4.3.3 State-dependent LQR controller

State-dependent LQR controller is a SDRE based approach and was implemented precisely as described in subsection 2.3.5. This algorithm was the simplest to implement.

At each time step we need to update our state dependent matrices $A(x)$ and $B(x)$ defined by equations 4.27 and 4.28. Afterwards we can use the MATLAB function *icare* (a bit faster than using the typical *lqr*) to solve for the gain matrix K. Afterwards a new input scaling matrix N needs to be calculated which is done by

$$\mathbf{N} = \mathbf{B}^{\dagger}(\mathbf{A} - \mathbf{B} \tag{4.35}$$

The system input $\mathbf{u}$ is then computed using the equation

$$\mathbf{u} = \mathbf{Ns} - \mathbf{Kx} \tag{4.36}$$

It is very simple to implement, however at each time step the ARE needs to be solved and this is computationally complex task, so it is a trade-off between simplicity and computational complexity.

4.3.4 Algorithm comparison

First all of the controllers were tuned to follow the trajectory as best as they can. The trajectory goes over area with lower controllability. This causes some deviation from the prescribed trajectory at this point. The resulting trajectories with their corresponding actuation can be seen in figure 4.16. We see that every algorithm performs different actuation inputs to the system and all of them perform differently while following this trajectory.

Computational time was measured for each algorithm and afterwards noise was added to the x vector (3 different values to each of the algorithms) in the ode solver (*ode1* was used with a step of 0.001 s). The results are shown in table 4.2. The computational complexity is normalized with respect to the PID algorithm which is the fastest.

Algorithm	**Complexity**	**MSE [m^2]**		
		$\sigma^2 = 0.0001$	$\sigma^2 = 0.001$	$\sigma^2 = 0.005$
PID	1	1.54e-6	1.71e-5	1.25e-4
FBL	3.4	7.09e-7	4.82e10	6.62e13
SDRE	15.6	5.50e-8	e.64e-6	8.35e-5

Table 4.2: Table of results

From these results we can conclude that the algorithms loose stability as the noise level increases. The FBL algorithm becomes unstable with very low noise levels. The best performing with respect to the MSE appears to be the SDRE algorithm which is however

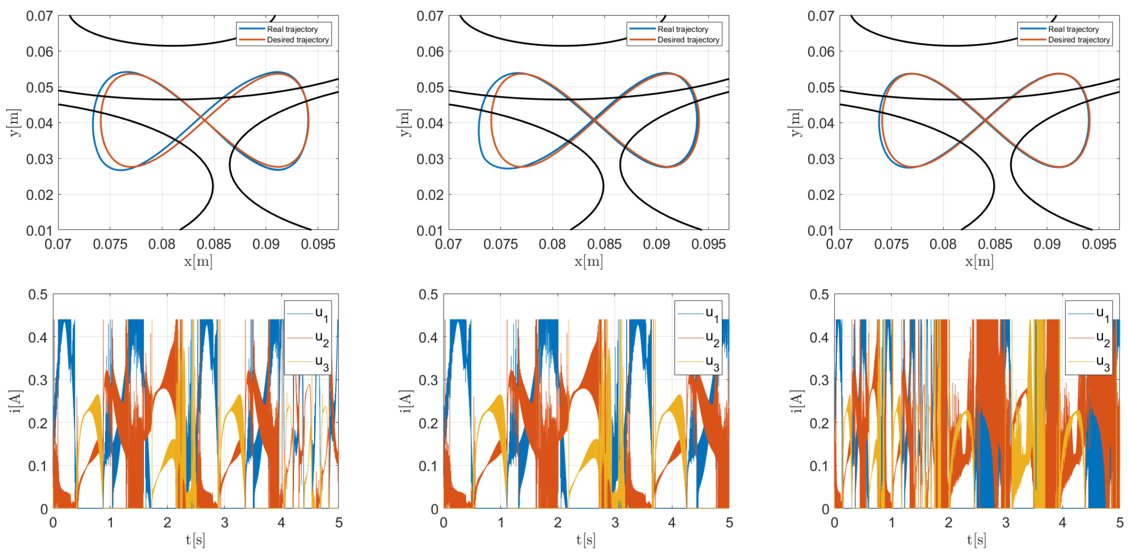


Figure 4.16: **Trajectories and corresponding actuation in time** Left column - PID, Middle column - FBL, Right column - SDRE

very slow compared to the PID which still holds with decent results and is faster. The choice of the best depends on the situation and on the available hardware. The PID controller will have difficulties when the trajectory gets further away from the center (which is the point around which is the PID tuned), whilst the other may not. All of the algorithms will be further tested in the following chapter in practical experiments.

4.4 SDRE implementation to Simulink RealTime

The practical implementation of control algorithms will be performed on IO card MF624 from Humusoft company. This card has full Simulink support to run real-time applications however the Simulink model must contain only blocks which can be compiled to C, which means the Simulink coder must be able to build the used functions to C. However the state-dependent LQR controller requires ARE solver to work, which is not supported by Simulink coder, which means it cannot be used. To overcome this issue a simple custom ARE solver was required. Implementation was done using a method very similar to Schur method described in [18].

The method is used to solve the continuous ARE of the form

$$\mathbf{A^T P + PA - PBR^{-1}B^T P + Q = 0} \tag{4.37}$$

We compute the $\mathbf{Z}$ matrix as

$$\mathbf{Z} = \begin{bmatrix} \mathbf{A} & \mathbf{-BR^{-1}B^T} \\ \mathbf{-Q} & \mathbf{-A^T} \end{bmatrix} \tag{4.38}$$

Now we would usually perform the Schur decomposition of the $\mathbf{Z}$ matrix, however this is not possible because the Simulink Coder does not support the *schur* function which performs it. However we only need to find the non-zero eigenvalues of the $\mathbf{Z}$ matrix and order them to form quasi upper triangular matrix. This can be done with simple search and order routine while using the *eig* function, which returns the eigenvalues and eigenvectors of the $\mathbf{Z}$ matrix.

Now if we define the ordered (highest to the lowest eigenvalue) eigenvectors with non-negative eigenvalues as $\mathbf{U}$, then we can decompose this matrix as

$$\mathbf{U} = \begin{bmatrix} \mathbf{U_1} \\ \mathbf{U_2} \end{bmatrix} \tag{4.39}$$

The solution to the CARE 4.37 is obtained as

$$\mathbf{P} = \mathbf{U_2 U_1^{-1}} \tag{4.40}$$

From this we can compute the feedback gain matrix as defined by eq. 2.36 and implement the state-dependent LQR algorithm to the Simulink real-time target.

Chapter 5

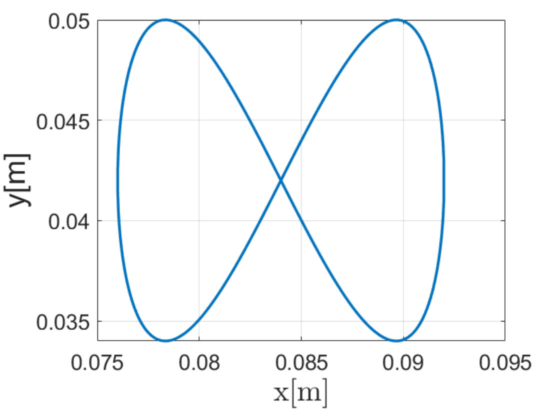
Experiments

In this chapter a series of experiments will be described as well as performed to compare how the nonlinear algorithms described in previous chapter work on a real hardware. In the first section we will define two trajectories and define evaluation criteria for performance. In the second section the experiments will be performed and algorithms compared to each other using the actual device.

NOTE: All of the algorithms use the Extedned Kalman Filter to obtain the best estimate of all of the states of the system. The implementation of the EKF is done using the standard simulink block *Extended Kalman Filter* and so the precise implementation is not described here. The used covariance matrices **Q** and **R** can be seen in appendix B.1.

5.1 Definition of trajectories

Two types of trajectories will be used. One will be continuous very similar to the one used in the the previous chapter with the numerical models. Other one will consist of random stair signal in the x and y direction. The random stairs need to be filtered with at least a first order filter so the signal is differentiatable and can be used in trajectory planner (Trajectories have to be physically executable).The trajectories can be seen in figure 5.1.



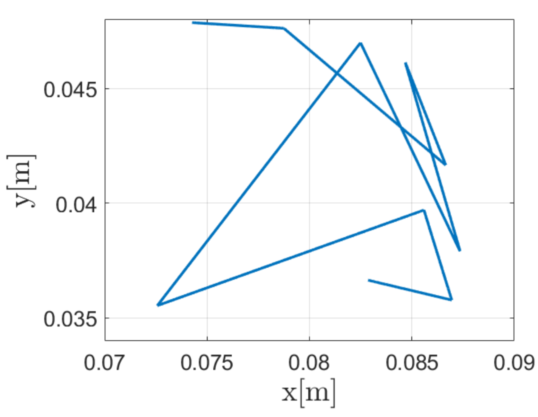


Figure 5.1: **Final trajectories** Left - continuous, Right - Random Stairs

The final tests will be performed with longer time intervals betweem the tests to let the device cool down. The temperature affects the behaviour of the system and so to make sure all of the tests have same conditions the device will have cool down period between the tests. Each algorithm will be run for the duration of 30 seconds and afterwards a MSE value will be calculated for each algorithm. Since all of the algorithms have a trajectory planner present, this value represents also how stable the control loop is with specific algorithm.

5.2 Comparison

First all of the controllers were tuned to be able to follow the prescribed trajectories. The final tuning constants and matrices can be seen in the appendix B. The resulting control can be seen in figures 5.2 and 5.3.

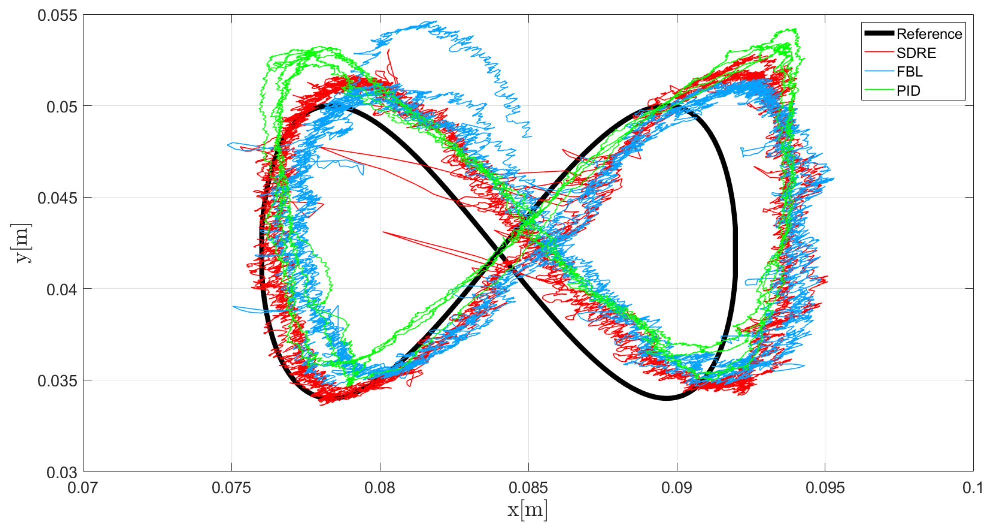


Figure 5.2: **Final algorithm comparison - (Continuous trajectory)**

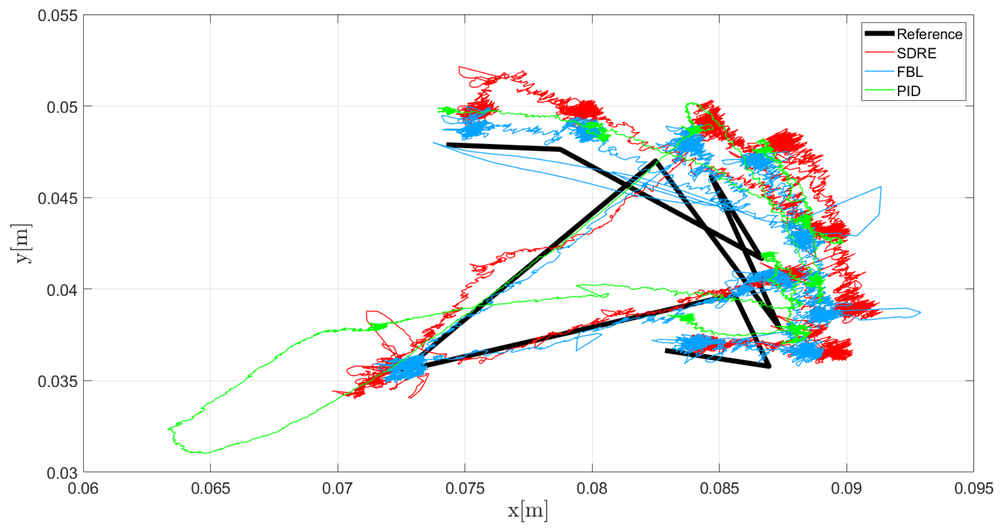


Figure 5.3: **Final algorithm comparison - (Stair trajectory)**

We can see the behavior of the algorithms on the detail of the stair signal for the x position on figure 5.4. We can notice a steady state error which is not constant for all trajectories, this is probably caused by various levels of controllability at different points of the trajectory as well as different accuracies of the model at each point.

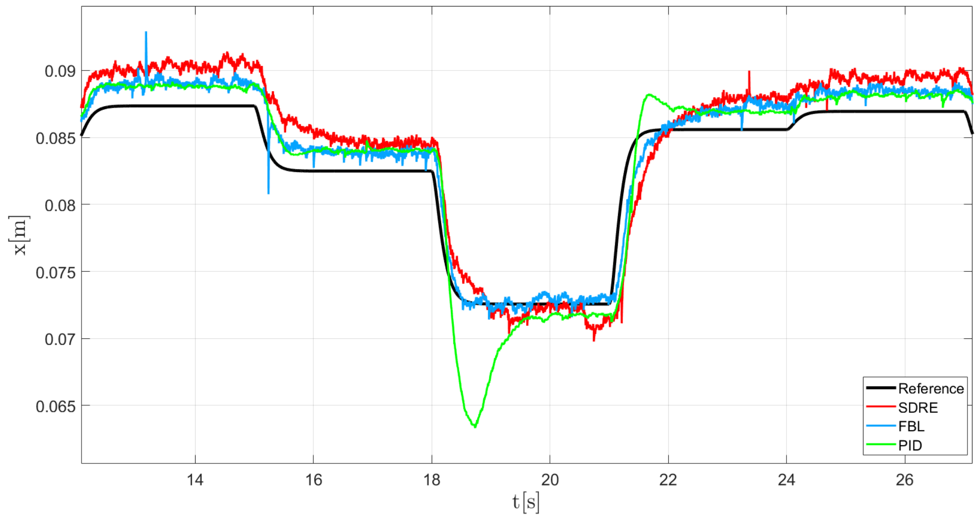


Figure 5.4: **Final algorithm comparison - (Stair trajectory in time)**

The final results can be seen in the table 5.1. The MSE between the reference trajectory and the real trajectory signal was calculated as well as the mean value of the current through the coil for the specific experiment (calculated as $\bar{i} = \bar{i_1} + \bar{i_2} + \bar{i_3}$). This value shows us how aggressive the controller was while trying to achieve the goal. We see that these results do not fully agree with the ones obtained from our numerical models. From these results we see that the FBL and SDRE algorithms have similar performance, whilst the SDRE is little bit more aggressive, so it is possible that if the FBL controller was tuned differently the resulting control might be superior.

Algorithm	**MSE** $[mm^2]$		$\bar{i}[A]$	
	Continuous	Stair	Continuous	Stair
SDRE	69.12	53.07	0.311	0.317
FBL	62.29	52.78	0.267	0.241
PID	114.68	99.63	0.279	2487

Table 5.1: Table of experimental results

Chapter 6

Conclusion

As a main outcome of this thesis an electromagnetic positioning platform was constructed and several nonlinear control algorithms were tested using this platform with positive results.

One of the main goals of this thesis was testing of several nonlinear control algorithms. Three of them were tested, mainly the ones which are not commonly described by literature (FBL, SDRE) with exception of classical PID for reference.

The algorithms were tested using the numerical model of the device (with estimated parameters based on a real data) and also practically on a real device.

In simulations both the performance, computational complexity and stability were considered and the results can be seen in table 4.2. From these results we concluded that SDRE is computationally most expensive but also provides best results with valid alternative the PID controller, which provided very good results but is also computationally very cheap. The FBL algorithm has shown poor performance in simulations compared to others.

However these results were not confirmed using the real device. On a real device the FBL and SDRE algorithms performed similarly, with very promising results that the FBL could work better. The results of these experiments are shown in table 5.1.

The performance of all of the algorithms was impaired by noisy measurements of the positions, which although an Extended Kalman Filter was used was still significant. If a better model was used the KF could probably have been tuned differently to reduce the noise even more. The source of this noise is probably the long wiring leading from and through the devices as well as the rapidly changing magnetic fields in proximity to these wires. The overall resolution of the measurement could also be improved by using smaller touch panel, which would limit the operating range but the current operating range is rather small so this may not be an issue.

Another source of error was the inaccuracy of the model, which is the cause of the steady-state error in the state-space algorithms. The state-space controllers are designed to drive the states towards zero. The reference value shifts this zero to a different point in the state-space. This is done by scaling the reference value by the DC gain of the system which is calculated from the model. So if our model is inaccurate this causes a steady-state error of the resulting control.

Several important points were made while analyzing the behaviour of the system model, such as the controllability is not constant in the whole plane and the levels of controllability change depending on where the ball is positioned. This fact lead to most control algorithms losing stability or precision in regions with lower controllability, and this was confirmed by both experiments and the model (figures 4.16 and 5.2). In further research this could lead to relation between the controller gains **Q** and **R** and the level of controllability in that region. Which could be used while using the SDRE algorithm which recalculates the K matrix at each point.

Some aspects of the work were affected by the current COVID-19 crisis which prevented from usage of better construction techniques, so one of the points which could be improved in the future is the final construction, which was not done using professional tools, however serves well as a proof of concept and can be easily modified in the future thanks to the modular design of all of the elements.

The platform opens the door for many interesting tasks in the future, such as model predictive control, nonlinear boundary value problems, state-space path finding and many other tasks which can be performed.

In conclusion an electromagnetic positioning platform was designed, analyzed and tested. This lead to interesting results which will lead to further research in the future. Some of the results were not conclusive, because are affected by human factor such as how well the controllers are tuned, which depends strongly on personal preference and/or is hard to judge. The final implementation was performed on a real-time IO card and all of the algorithms were proven to work properly on this hardware and further algorithms can be tested in the future.

Bibliography

[1] *2D actuator move micro robot in X/Y 2D space | Hackaday.io*. Available at: https://hackaday.io/project/154496-2d-actuator-move-micro-robot-in-xy-2d-space.

[2] *CE 152 magnetická levitace | Humusoft*. Available at: https://www.humusoft.cz/models/ce152/.

[3] Agrachev, A. A., Morse, A. S., Sontag, E. D., Sussmann, H. J. and Utkin, V. I. Nonlinear and Optimal Control Theory. 2008, p. 347. DOI: 10.1007/978-3-540-77653-6.

[4] Al Muthairi, N. F. and Zribi, M. Sliding mode control of a magnetic levitation system. *Mathematical Problems in Engineering*. jun 2004, vol. 2004, no. 2, p. 93–107. DOI: 10.1155/S1024123X04310033. ISSN 1024123X.

[5] Angermann, A., Rau, M. B. M. and Wohlfarth, U. *MATLAB Optimization Toolbox*. 2008. Available at: https://www.mathworks.com/products/optimization.htmlhttp://www.mathworks.com/products/optimization/.

[6] Beeler, S. and Cox, D. State-Dependent Riccati Equation Regulation of Systems with State and Control Nonlinearities. august 2004.

[7] Brablc, M., Sova, V. and Grepl, R. Adaptive feedforward controller for a DC motor drive based on inverse dynamic model with recursive least squares parameter estimation. In: *Proceedings of the 2016 17th International Conference on Mechatronics - Mechatronika, ME 2016*. 2017. ISBN 9788001058831. Available at: https://ieeexplore.ieee.org/document/7827809.

[8] Brunton, S. L. and Kutz, J. N. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2019.

[9] Filip, J. *Extension of the Control System for the Magnetic Manipulator with a Non-Flat Surface*. 2015. Dissertation. ČVUT Praha. Available at: http://hdl.handle.net/10467/61947.

[10] Fu, W. N., Zhou, P., Lin, D., Stanton, S. and Cendes, Z. J. Magnetic force computation in permanent magnets using a local energy coordinate derivative method. In: *IEEE Transactions on Magnetics*. Mar 2004, vol. 40, 2 II, p. 683–686. DOI: 10.1109/TMAG.2004.824774. ISSN 00189464.

[11] Fujitsu. *Feather Touch 4-wire resistive touch panel designed for multi-touch gesturing applications*. Available at: http://us.fujitsu.com/components/.

[12] Grepl, R. Composite Controller for Electronic Automotive Throttle with Self-tuning Friction Compensator. In: Jabłoński, R. and Březina, T., ed. *Mechatronics*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, p. 73–78. ISBN 978-3-642-23244-2.

[13] Gurtner, M. and Zemánek, J. *Ball in double hoop: demonstration model for numerical optimal control*. Available at: http://github.com/aa4cc/flying-ball-in-hoop.

[14] Instruments, T. LM1875 20W Audio Power Amplifier. 2004. Available at: https://www.ti.com/lit/ds/symlink/lm1875.pdf?ts=1590415581543.

[15] Izhikevich, E. Equilibrium. *Scholarpedia*. 2007, vol. 2, no. 10, p. 2014. DOI: 10.4249/scholarpedia.2014. ISSN 1941-6016. Available at: http://www.scholarpedia.org/article/Equilibrium.

[16] Jribi, R., Maalej, B. and Derbel, N. Robust Adaptive Feedback Linearization Control of Human Exoskeletons. In: *16th International Multi-Conference on Systems, Signals and Devices, SSD 2019*. Institute of Electrical and Electronics Engineers Inc., Mar 2019, p. 747–751. DOI: 10.1109/SSD.2019.8893163. ISBN 9781728118208.

[17] Kwakernaak, H. and Sivan, R. *Linear optimal control systems*. Wiley Interscience, 1972. Wiley-Interscience publication. ISBN 9780471511106. Available at: https://books.google.sk/books?id=mf0pAQAAMAAJ.

[18] Laub, A. A Schur method for solving algebraic Riccati equations. *IEEE Transactions on Automatic Control*. 1979, vol. 24, no. 6, p. 913–921.

[19] Microchip. *MCP601/1R/2/3/4*. 2007. Available at: http://ww1.microchip.com/downloads/en/DeviceDoc/21314g.pdf.

[20] Microchip Technology. *dsPIC33FJ32MC302/304, dsPIC33FJ64MCX02/X04 AND dsPIC33FJ128MCX02/X04*. 2007. Available at: http://ww1.microchip.com/downloads/en/DeviceDoc/70291G.pdf.

[21] Microchip Technology. *MCP4802/4812/4822*. Microchip Technology, 2010. Available at: http://ww1.microchip.com/downloads/en/devicedoc/20002249b.pdf.

[22] Najman, J., Brablc, M., Rajchl, M., Bastl, M., Spáčil, T. et al. *Monte carlo based detection of parameter correlation in simulation models*. 2020. ISBN 9783030299927.

[23] Naus, I. G. J. L. *Gain scheduling Robust Design and Automated Tuning of Automotive Controllers*. 2009.

[24] Nelles, O. *Nonlinear system identification : from classical approaches to neural networks and fuzzy models*. Springer, 2001. 785 p. ISBN 3540673695.

[25] Ong, C. *Dynamic Simulation of Electric Machinery: Using MATLAB/SIMULINK*. Prentice Hall PTR, 1998. ISBN 9780137237852. Available at: https://books.google.sk/books?id=_0weAQAAIAAJ.

[26] Penrose, R. A generalized inverse for matrices. *Mathematical Proceedings of the Cambridge Philosophical Society.* Cambridge University Press. 1955, vol. 51, no. 3, p. 406–413. DOI: 10.1017/S0305004100030401.

[27] Rajchl, M. and Brablc, M. Inverse Model Approximation Using Iterative Method and Neural Networks with Practical Application for Unstable Nonlinear System Control. In: *Proceedings of the 2018 18th International Conference on Mechatronics - Mechatronika, ME 2018.* 2019. ISBN 9788021455443. Available at: https://ieeexplore.ieee.org/document/8624827.

[28] Ramos, O. E. A Comparison of Feedback Linearization and Sliding Mode Control for a Nonlinear System. In: *SHIRCON 2019 - 2019 IEEE Sciences and Humanities International Research Conference.* Institute of Electrical and Electronics Engineers Inc., Nov 2019. DOI: 10.1109/SHIRCON48091.2019.9024871. ISBN 9781728138183.

[29] Richter, F. *Extension of the platform for magnetic manipulation.* 2017. Dissertation. ČVUT Praha.

[30] Selos. *Holding solenoids E1AS.* 2015. Available at: https://www.magnety.sk/sub/magnety.sk/images/E1AS%20EN.pdf.

[31] Shamma, J. S. and Cloutier, J. R. Existence of SDRE stabilizing feedback. *IEEE Transactions on Automatic Control.* mar 2003, vol. 48, no. 3, p. 513–517. DOI: 10.1109/TAC.2002.808473. ISSN 00189286.

[32] Sheen, J.-J. and Bishop, R. H. Spacecraft nonlinear control. jan 1992.

[33] Slotine, J. and Li, W. *Applied Nonlinear Control.* Prentice Hall, 1991. ISBN 9780130408907. Available at: https://books.google.sk/books?id=cwpRAAAAMAAJ.

[34] Tangirala, A. *Principles of System Identification: Theory and Practice.* CRC Press, 2018. ISBN 9781439896020. Available at: https://books.google.sk/books?id=aUHOBQAAQBAJ.

[35] Texas Instruments. *TSC2007.* 2007. Available at: https://www.ti.com/lit/ds/symlink/tsc2007.pdf?ts=1590666017723.

[36] Utkin, V. I. Sliding mode control design principles and applications to electric drives. *IEEE Trans. Ind. Electron.* 1993, p. 23–36.

[37] Yang, J., Sun, R., Cui, J. and Ding, X. Application of composite fuzzy-PID algorithm to suspension system of maglev train. In: *IECON Proceedings (Industrial Electronics Conference).* 2004, vol. 3, p. 2502–2505. DOI: 10.1109/iecon.2004.1432194.

[38] Zemánek, J. *Distributed manipulation by controlling force fields through arrays of actuators.* Dissertation.

Appendix A

Electrical schematics

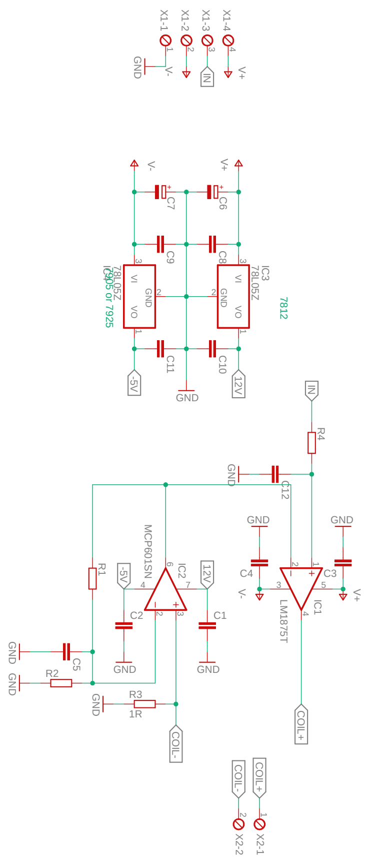


Figure A.1: Electrical schematic of the coil driver

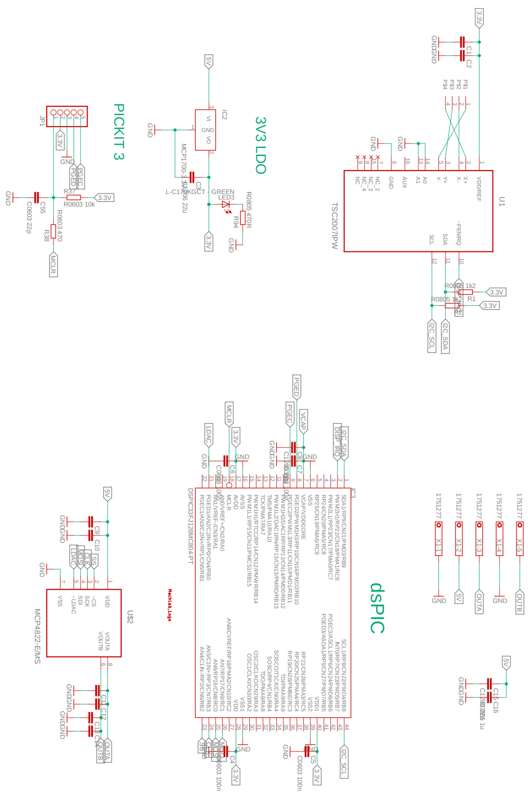


Figure A.2: Electrical schematic of the signal processing board

Appendix B

List of controller and Kalman gains

Kalman filter

$$\mathbf{Q} = \begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.01 \end{bmatrix}, \mathbf{R} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.1}$$

SDRE

$$\mathbf{Q} = \begin{bmatrix} 2410 & 0 & 0 & 0 \\ 0 & 82 & 0 & 0 \\ 0 & 0 & 2410 & 0 \\ 0 & 0 & 0 & 82 \end{bmatrix}, \mathbf{R} = \begin{bmatrix} 0.01 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 0.01 \end{bmatrix} \tag{B.2}$$

FBL

$$\mathbf{Q} = \begin{bmatrix} 3500 & 0 & 0 & 0 \\ 0 & 40 & 0 & 0 \\ 0 & 0 & 3500 & 0 \\ 0 & 0 & 0 & 40 \end{bmatrix}, \mathbf{R} = \begin{bmatrix} 0.01 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 0.01 \end{bmatrix} \tag{B.3}$$

PID

$$P = 92.4, D = 8.32 \tag{B.4}$$