Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Control Engineering

Trajectory planning for distributed magnetic manipulation

Bachelor's thesis

Adam Uchytil

Supervisor
Ing. Jiří Zemánek, Ph.D.

Academic year 2021/2022

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Uchytil Adam**	Personal ID number:	**492389**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Control Engineering**		
Study program:	**Cybernetics and Robotics**		

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Trajectory planning for distributed magnetic manipulation

Bachelor’s thesis title in Czech:

Plánování trajektorií pro distribuovanou manipulaci pomocí magnetického pole

Guidelines:

The aim of this work is to develop a system for motion planning in distributed manipulation using magnetic field. The control system currently uses an algorithm that commands a coil array according to the desired forces and the current positions. This work should focus on a higher-level control that will plan the motion of objects with respect to actuator constraints, collision avoidance, etc.
Individual tasks:
1. Complete, document, and test the NVIDIA Jetson-based position measuring system and compare it to the RaspberryPi-based system.
2. Migrate the control system to the NVIDIA Jetson platform. (For example, using an automatic compilation of the control algorithm from Simulink)
3. Design and validate an algorithm for object trajectories planning that will meet actuator constraints and collision avoidance. The state space can be discretized for planning purposes. Consider using RRT* algorithm.
4. Prepare a demonstration of the proposed algorithms on the laboratory model of magnetic manipulation (Magman), e.g., steel balls playing a song on a xylophone.

Bibliography / sources:

[1] Karaman, Sertac, and Emilio Frazzoli. "Sampling-based algorithms for optimal motion planning." The international journal of robotics research 30.7 (2011): 846-894.
[2] Hodan, Dominik. Reinforcement learning for manipulation of collections of objects using physical force fields. Bachelor thesis. CTU FEE. 2020
[3] Simonian, Aram. Feedback control for planar parallel magnetic manipulation. Master thesis. CTU FEE. 2014
[4] Richter, Filip. Extension of the platform for magnetic manipulation (in Czech). Master thesis. CTU FEE. 2017

Name and workplace of bachelor’s thesis supervisor:

Ing. Jiří Zemánek, Ph.D. Department of Control Engineering FEE

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **28.01.2022** Deadline for bachelor thesis submission: **20.05.2022**

Assignment valid until: **30.09.2023**

Ing. Jiří Zemánek, Ph.D.	prof. Ing. Michael Šebek, DrSc.	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student’s signature

Declaration of authorship

Acknowledgments

First and foremost, I would like to express gratitude to my parents, whose care, attention and support allowed me to study and focus on this thesis. I would also like to thank my supervisor, Ing. Jiří Zemánek, PhD., for his expertise and support. Moreover, I thank all the members of AA4CC. I especially thank Krištof for proofreading this thesis and the general companionship in the lab. I also must thank Martin for his many brilliant insights. I can not forget to thank my friends and girlfriend, whose support got me going till the end.

Abstract

This bachelor's thesis deals with planning trajectories for the distributed magnetic manipulation platform MagMan with the possibility of generalization to other distributed manipulation platforms. Initially, moving MagMan to a new position measuring system is described, and the new system is benchmarked. Later, two algorithms for planning trajectories that are collision-free and obstacle-free are proposed. The algorithms are based on nonlinear mathematical programs initialized by a path generated by RRT* algorithm. Furthermore, an LQR trajectory tracking controller and a state estimation system are designed. In the end, the algorithms are experimentally evaluated on the actual platform.

Keywords: distributed manipulation, MagMan, mathematical programming, RRT*, trajectory planning

Abstrakt

Tato bakalářská práce se zabývá plánováním trajektorií pro platformu distribuované magnetické manipulace MagMan s případnou možností rozšíření i na další platformy distribuované manipulace. Nejprve je popsán přesun platformy MagMan na nový systém měření pozic objektů. Přínos tohoto nového systému je také experimentálně vyhodnocen. Následně jsou navrženy dva algoritmy pro plánování trajektorií bez kolizí s překážkami i bez kolizí mezi jednotlivými objekty. Tyto algoritmy jsou založeny na nelineárních matematických programech inicializovaných cestou z RRT* algoritmu. Pro sledování trajektorií je navržen LQR regulátor spolu se systémem pro odhadování stavu. Na závěr jsou navržené plánovací algoritmy experimentálně vyhodnoceny pomocí experimentů na skutečné platformě.

Klíčová slova: distribuovaná manipulace, MagMan, matematické programování, RRT*, plánování trajektorií

Contents

Nomenclature

Acronyms

AA4CC	Advanced Algorithms for Control and Communications.
LQR	Linear–Quadratic Regulator.
NLP	Nonlinear Program.
QP	Quadratic Program.
TP-BVP	Two Point Boundary Value Problem.

Mathematical symbols

$\|\|\mathbf{A}\|\|_\mathrm{F}$	Frobenius norm of matrix $\mathbf{A}$.
$\mathbb{R}$	The set of real numbers.
$\mathbf{A} \succ \mathbf{0}$	The matrix $\mathbf{A}$ is positive definite.
$\mathbf{A} \succeq \mathbf{0}$	The matrix $\mathbf{A}$ is positive semidefinite.
$\mathbf{A}^\mathrm{T}$	Transposition of matrix $\mathbf{A}$.
$\mathcal{I}_N$	The index set $\{1, 2 \ldots, N\}$.

1 Introduction

Planning trajectories for distributed manipulation is a challenging task. Distributed manipulation uses many actuators, shaping a force field to manipulate an object. Therefore when planning, both the motion of the individual objects and the underlying force field must be considered. I attempt to tackle this task for one specific instance of distributed manipulation. That is MagMan (see Section 1.2), the platform for distributed manipulation using a magnetic field.

Motivation & Goals

The distributed magnetic manipulation platform currently uses simple predefined trajectories without considering actuator constraints and trajectory feasibility. Moreover, no collision and obstacle avoidance is considered. The trajectories are tracked purely using a feedback controller. A system for planning trajectories based on desired positions and velocities would allow the distributed magnetic manipulation to achieve more complex tasks. For example, distributed manipulation can serve as an intelligent conveyor belt, where getting the manipulated objects to some desired positions without colliding is essential. Moreover, for entertainment purposes, the distributed manipulation could play the xylophone. In that case, the velocity of the manipulated objects is also important, as it determines the strength of the tone.

The thesis aims to develop such a planning system. The system could also be generalized to other distributed manipulation platforms and thus lay the groundwork for further related research.

Structure

In Chapter 2, I briefly go over moving the MagMan platform to a new position measuring system I developed for my semestral project. In chapter Chapter 3, I present the general trajectory planning problem as well as specifics of planning for the distributed magnetic manipulation. I then develop the planning problem into a mathematical program and solve it in Chapter 4. I present the design of a feedback controller for tracking purposes in Chapter 5. Lastly, I describe and evaluate experiments on the real platform in Chapter 6.

1.1 Distributed manipulation

In *distributed manipulation*, the actuators are not part of the manipulated object but are actually distributed in space. The actuators are typically arranged in a grid formation. Each actuator only exerts a force on manipulated objects in its vicinity and can be controlled independently. Together the actuators create a force field. The goal of the control system is to shape the field in a way which actuates the manipulated object.

The distributed manipulation and the design of the underlying control systems have been a research interest of the Advanced Algorithms for Control and Communications group (AA4CC) at the Department of Control Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague for over ten years. The members of AA4CC have developed multiple platforms for distributed manipulation. These include the following platforms: *DEPMan*, a platform for micromanipulation using a non-uniform electric field or so-called dielectrophoresis, described in [ZMH18]; *AcouMan*, a platform for manipulation using acoustophoresis, that is, the shaping of the surrounding acoustic pressure field (see [Mat+19]) and *MagMan*, a platform for manipulation using a magnetic field, described in Section 1.2.

1.2 The MagMan platform

MagMan is an experimental platform for distributed magnetic manipulation developed by AA4CC. Its main use is for experimental evaluation of distributed control algorithms.

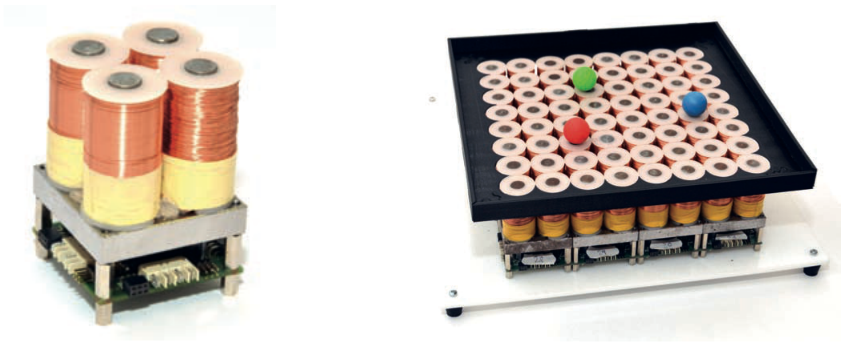

(a) Single module **(b)** Complete platform

Figure 1.1: The MagMan platform (taken from [Zem18], pp. 82, 83)

1.2.1 Decription

The platform is composed of modules, each module containing four coils in a square array and electronics that allow for command-based control of a current flowing through the coils. One such module is shown in Figure 1.1 (a). The modules communicate with each other and the outside world using IAE 485 bus. The current platform comprises 16 such modules, which translates to 8×8 coils as is shown in Figure 1.1 (b). However, theoretically, an arbitrary grid-based configuration could be accomplished. The platform also has a built-in IAE 485 to USB converter. Thus the platform can be controlled conveniently over USB.

MagMan currently manipulates steel balls trough feedback control. Positions of the individual balls are obtained from camera captured image using an image processing algorithm. Given the desired and current position of the balls, currents through individual coils are calculated by the control algorithm. These desired currents are then passed to the platform over USB. The platform then sets desired currents through the individual coils, which generate a magnetic field that exerts an attractive force on the balls, effectively manipulating them.

Both image processing and control calculations were originally done on Raspberry Pi. During my semestral project, I developed a new ball position measuring system on NVIDIA Jetson AGX Xavier with Basler acA1300-200uc high frame rate camera. Benchmarks of the new system as the description of migrating the control algorithms to the NVIDIA platform are given later in Chapter 2. Diagram of the current platform setup is given in the Figure 1.2.

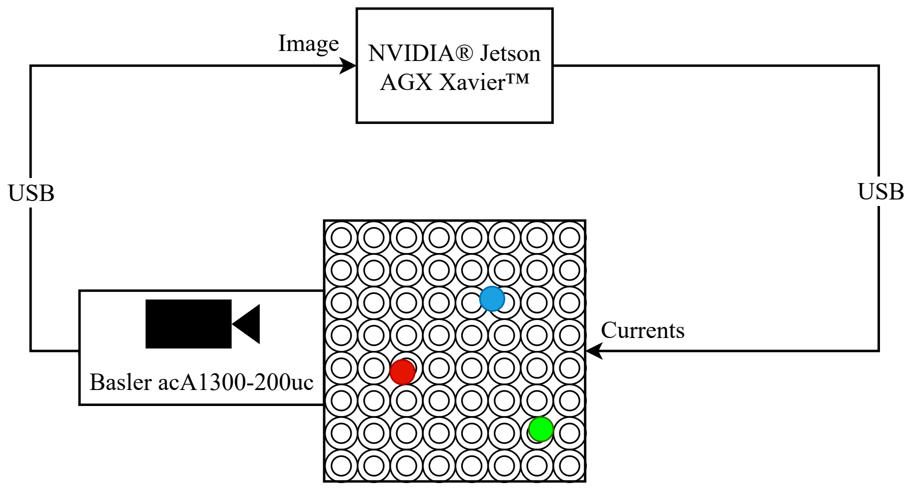


Figure 1.2: Current MagMan setup

1.2.2 Mathematical model

For clarity, I will now give a short derivation of the underlying mathematical model in use of the MagMan platform. Similar derivation can be found in the introduction of another bachelor's thesis [Hod20], pp. 4-5. For complete and rigorous derivation, please refer to [Zem18], pp. 91-111.

Magnetic field of a single coil

The magnetic field generated by a single coil is modelled as a magnetic monopole. The field can be separated as

$$\mathbf{B}_{\text{coil}}(i, \mathbf{r}) = f(i)\mathbf{B}_{\text{MP}}(\text{r}), \tag{1.1}$$

where $f(i)$ describes how the field scales with current i flowing through the coil and $\mathbf{B}_{\text{MP}}(\mathbf{r})$ how the field changes with position $\mathbf{r}$. The current dependent part of Equation (1.1) was found by fitting measured data to be

$$f(i) = 0.998 \arctan\left(2.51|i| + 5.38i^2\right) \operatorname{sgn}(i), \ |i| \leq 440\,\text{mA}. \tag{1.2}$$

The magnetic field of the monopole is then given as

$$\mathbf{B}_{\text{MP}}(\mathbf{r}) = -\frac{q_m}{4\pi}\nabla\frac{1}{\mathbf{r}} = \underbrace{-\frac{q_m}{4\pi}}_{a}\frac{\mathbf{r}}{|\mathbf{r}|^3}, \tag{1.3}$$

where q_m is the strenght of the monopole at the origin and it was estimated that $a = 3.565 \times 10^{-5}$.

Force generated by a single coil

The force exerted on a steel ball by a magnetic field generated from a single coil can be described as

$$\mathbf{F}(i, x, y, z) = k\nabla\mathbf{B}^2_{\text{coil}} = ka^2 f^2(i)\nabla\frac{1}{(x^2 + y^2 + z^2)^2}, \tag{1.4}$$

where k is a constant dependent on the radius of the ball, its permeability and permeability of the surrounding. Assuming planar motion with fixed $z = d$, the x-component of the force then takes the form of

$$F_x(i, x, y) = \underbrace{-4ka^2}_{c} f(i)^2 \frac{x}{(x^2 + y^2 + d^2)^3}, \tag{1.5}$$

identically can be derived the y-component. For ball of radius 10 mm it was found that $c = -2.041 \times 10^{-10}$ and $d = 13.3$ mm.

Force generated by multiple coils

More generally, following from expression in Equation (1.5), a new function can be introduced

$$G_{n,x}(x, y) = \frac{cx}{[(x - x_n)^2 + (y - y_n)^2 + d^2]^3}, \tag{1.6}$$

where (x_n, y_n) are coordinates of the n-th coil. The x-component of the force exerted on the ball by the superposed magnetic field of multiple coils is then modelled as

$$F_x(x, y, i_1, \dots, i_N) = \sum_{n=1}^{N} f^2(i_n) G_{n,x}(x, y), \tag{1.7}$$

where i_n is the current flowing through the n-th coil. Now expressing both components in vector form yields following

$$\mathbf{F}(\mathbf{r}, i_1, \dots, i_N) = \underbrace{\begin{bmatrix} G_{1,x}(\mathbf{r}) & G_{2,x}(\mathbf{r}) & \cdots & G_{N,x}(\mathbf{r}) \\ G_{1,y}(\mathbf{r}) & G_{2,y}(\mathbf{r}) & \cdots & G_{N,y}(\mathbf{r}) \end{bmatrix}}_{\mathbf{G}(\mathbf{r})} \underbrace{\begin{bmatrix} f^2(i_1) \\ f^2(i_2) \\ \vdots \\ f^2(i_N) \end{bmatrix}}_{\mathbf{v}}. \tag{1.8}$$

It must be stated that this formulation assumes that the interaction between the coils is minimal. That is, their magnetic fields do not significantly overlap. If the interaction were not negligible, then the sum in Equation (1.7) would also need to contain off-diagonal members, meaning Equation (1.8) would a quadratic form in $f(i_n)$.

Ball dynamics

The following set of ODEs describes the planar dynamics of the steel ball

$$\ddot{x} = \frac{F_x}{m_{\text{eff}}}, \qquad \ddot{y} = \frac{F_y}{m_{\text{eff}}}, \tag{1.9}$$

where

$$m_{\text{eff}} = \frac{7}{5} m, \tag{1.10}$$

is the effective mass putting together the actual mass of the ball m and its moment of inertia.

1.2.3 Control

I will now describe how the control is handled on the MagMan platform, emphasizing feedback linearization, which allows for control based on the desired force.

Feedback linearization

As the ball dynamics is linear, the only modelled nonlinearity remains in the force-current model. Letting

$$\mathbf{v} = \begin{bmatrix} f^2(i_1) & f^2(i_2) & \cdots & f^2(i_N) \end{bmatrix}^{\mathrm{T}}, \tag{1.11}$$

can the current flowing through the k-th coil be obtained as

$$i_k = f^{-1}\left(\sqrt{v_k}\right), \tag{1.12}$$

where f^{-1} is to be interpreted as the inverse of the function f. Given a desired force $\mathbf{F}^{\text{des}}$ acting on the ball, finding a suitable $\mathbf{v}$ reduces to the problem solving following system of equations

$$\mathbf{F}^{\text{des}} = \mathbf{G}(\mathbf{r})\mathbf{v}, \tag{1.13}$$

which also must satisfy

$$\mathbf{0} \leq \mathbf{v} \leq \mathbf{1}. \tag{1.14}$$

Constraint on values of $\mathbf{v}$ arises from the current limits introduced in Equation (1.2). The presented problem can be solved in terms of constrained least-squares, which in turn leads to the ensuing *quadratic program* (QP)

$$\begin{aligned} \min_{\mathbf{v}} \quad & \frac{1}{2}||\mathbf{F}^{\text{des}} - \mathbf{G}(\mathbf{r})\mathbf{v}|| \\ \text{s.t.} \quad & \mathbf{0} \leq \mathbf{v} \leq \mathbf{1}. \end{aligned} \tag{1.15}$$

In the case of wanting to exert different forces on different balls, one natural extension of the problem from Equation (1.15) is to try to minimize the sum of squared errors of desired and real forces. Formally, given M balls balls with their respective position vectors $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_M$ and desired forces $\mathbf{F}_1^{\text{des}}, \mathbf{F}_2^{\text{des}}, \ldots, \mathbf{F}_M^{\text{des}}$ one possible quadratic program formulation is

$$\begin{aligned} \min_{\mathbf{v}} \quad & \frac{1}{2}\sum_{k=1}^{M}||\mathbf{F}_k^{\text{des}} - \mathbf{G}(\mathbf{r}_k)\mathbf{v}||^2 \\ \text{s.t.} \quad & \mathbf{0} \leq \mathbf{v} \leq \mathbf{1}. \end{aligned} \tag{1.16}$$

Notice that for $M = 1$ this problem reduces to the formulation from Equation (1.15). There is one problem with such a formulation. The optimal $\mathbf{v}$ minimizes a purely algebraic criterion. This may, in turn, lead to the excitation of a coil that negligibly contributes to the resulting force. Such behaviour is undesirable as it leads to unnecessary heating up of the platform. One way to fight this behaviour is to penalize the values of $\mathbf{v}$. Such penalization is accomplished with the following and final quadratic program formulation

$$\begin{aligned} \min_{\mathbf{v}} \quad & \frac{1}{2}\sum_{k=1}^{M}||\mathbf{F}_k^{\text{des}} - \mathbf{G}(\mathbf{r}_k)\mathbf{v}||^2 + \lambda\mathbf{v}^{\mathrm{T}}\mathbf{v} \\ \text{s.t.} \quad & \mathbf{0} \leq \mathbf{v} \leq \mathbf{1}, \end{aligned} \tag{1.17}$$

where $\lambda > 0$ is a tunable parameter.

The presented QPs are solved only for neighbouring coils of the manipulated balls. If the balls share no coils in their neighbourhood, the large QP is completely decomposed into smaller QPs for each ball. If some balls share neighbouring coils, the QPs need to be solved collaboratively. Details of how these QPs are solved are presented in [GZH22].

As the current scaling factors $\mathbf{v}$ for some desired force ball force $\mathbf{F}^{\text{des}}$ are obtained by solving a QP and converted to desired currents as per Equation (1.12), a so-called *feedback linearization* scheme arises. Refer to Figure 1.3, as the whole force to currents allocation process can be viewed as an approximate inversion of the mathematical model, represented by the gain $\mathbf{g}^{-1}$. This allows for the desired-force-based control, without worrying about the current allocation process.

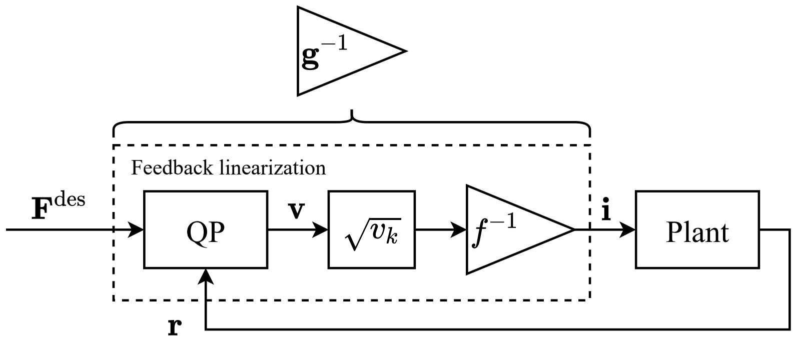


Figure 1.3: Feedback linearization of the MagMan platform. The symbol $\mathbf{i}$ represents the currents outputted by the control loop.

Higher level control

The higher-level control of the platform currently consists of a PD regulator, which turns the deviation from reference trajectory into correcting force. The feedback loop is depicted in Figure 1.4.

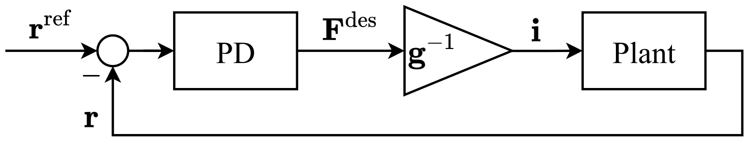


Figure 1.4: Feedback linearization of the MagMan platform. The symbol $\mathbf{i}$ represents the currents outputted by the control loop.

2 Position measuring system

An essential part of the MagMan platform is feedback, which is provided using a camera and image processing algorithm. As my semestral project, I developed a new position measuring system for the platform. There were two primary motivations for doing so. The first one is an increase of the resolution of the captured image and thus increasing the accuracy of the position measurement itself. The second one is an increase of the rate at which the frames are processed as the image processing speed is the bottleneck of the whole control loop.

Not too long ago, the platform relied on a Raspberry Pi 4B with a V1 camera. I upgraded the camera to a Basler acA1300-200uc high-speed camera. As the image processing algorithm, I implemented a RANSAC type detector which I massively parallelized on CUDA cores of NVIDIA Jetson AGX Xavier. The whole algorithm is written in C++. I will not go into more implementation details as that is not the subject of this thesis. There remain two things left to do. Move the control algorithms from the Raspberry Pi to the new NVIDIA platform and compare the two systems. Both of which I seek to do in this chapter.

2.1 Moving to NVIDIA Jetson AGX Xavier

2.1.1 The Real Time Linux Kernel

Real-time control algorithms have a strict deadline to meet. This deadline is given by the sample rate at which the control loop runs. Linux kernel was not developed with control applications in mind. It maximizes throughput at the expense of latency and determinism [Mad19]. Fortunately, there exists the so-called PREEMPT_RT patch[1], which changes the Linux kernel, making it more suitable for control applications. This project aims to decrease latencies and increase the predictability of the kernel just by modifying the existing code [RMF19]. NVIDIA already provides the patched Linux kernel[2] for the Jetson platform. So I only really needed to install an appropriate package from the repository.

2.1.2 Simulink automated code generation

The control algorithms of the MagMan platform are implemented in Matlab Simulink. NVIDIA provides a target[3] for the Simulink Coder, allowing for automatic code generation and deploying the algorithms on the Jetson platform. The problem with this approach is

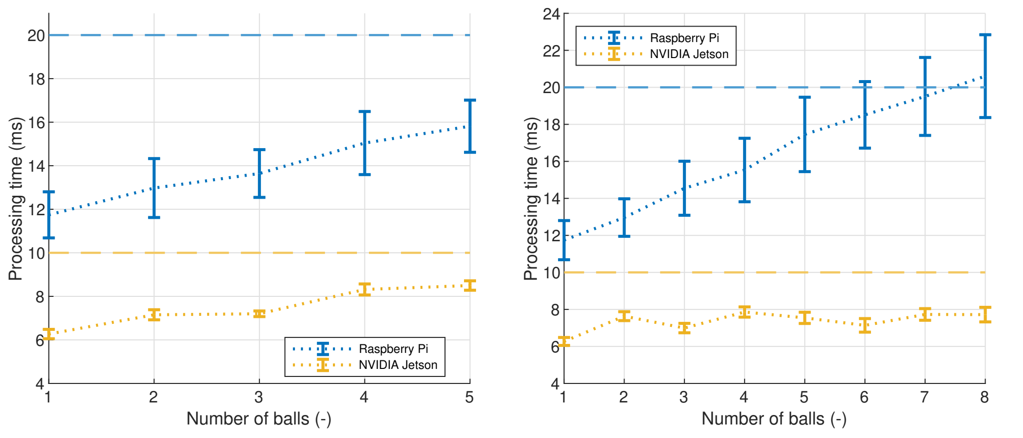
1 https://wiki.linuxfoundation.org/realtime/start
2 https://docs.nvidia.com/jetson/archives/r34.1/DeveloperGuide/text/SD/Kernel/KernelCustomization.html
3 https://www.mathworks.com/hardware-support/nvidia-jetson.html

that it does not generate real-time code. Thankfully, we have our own in house developed Simulink Coder Target for Linux[4], which can generate real-time compliant code. As I was already using this target on the Raspberry Pi platform, the moving of the control algorithms was just a matter of providing the correct compiler suite, which I obtained from NVIDIA's website.

2.2 Benchmarks

2.2.1 Processing time

I compared the two different systems on their image processing time, more precisely, the time it takes from the moment the camera provides the frame to the moment when the image coordinates of the balls are available. The benchmarks consisted of 30 seconds of closed-loop control of the platform, which is a real use-case test. Moreover, I tested how the processing time scales with the number of same and different colored balls. The results are provided in Figure 2.1 (b) and Figure 2.1 (a).



(a) Scaling with number of different colored balls detected.

(b) Scaling with number of same colored balls detected.

Figure 2.1: Comparison of the image processing time scalability. Vertical bars represent standard deviation. Dashed lines represent targeted sample time of each platform.

Following the results, I can state that the Jetson based solution satisfied the demand of being faster than the Raspberry Pi one. Even more important is that the Jetson solution does not overrun the targeted control loop sample time. Violating the sample time can have detrimental effects on the control, as no feedback is provided for one or more samples. On the other hand, the Raspberry Pi based system does overrun when six or more balls are present. One interesting observation is the Jetson based system does not appear to scale

4 https://github.com/aa4cc/ert_linux/

with the number of balls, or at least not as much as the Raspberry Pi one. The scaling is more observable with different colored balls on the Jetson Based system.

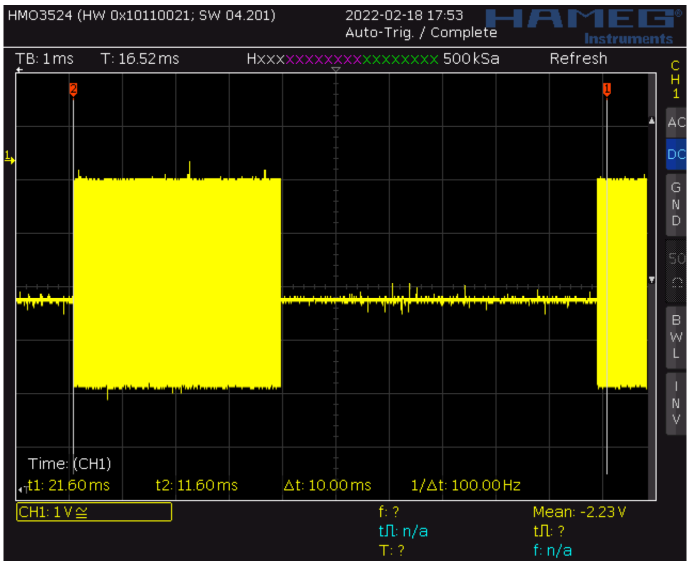


Figure 2.2: Coil packet timing without the RT_PREEMPT patch and NVIDIA provided Simulink target. Cursors show the 10 ms control loop sample time.

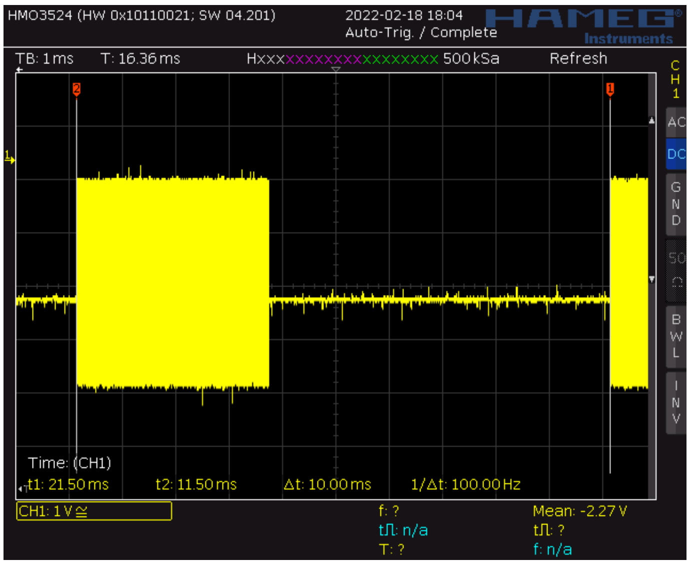


Figure 2.3: Coil packet timing with the RT_PREEMPT patch and our Simulink target. Cursors show the 10 ms control loop sample time.

2.2.2 Coil packet jitter

Additionally, I tried to find the effect of using the RT_PREEMPT kernel patch and our Simulink Target. With the control algorithm running, I connected an oscilloscope to the IAE 485 bus and recorded the coil commands being sent by Jetson to MagMan. I did this for two cases. In the first case, I used Jetson with stock kernel version 4.9.253 and Simulink target provided by NVIDIA. I utilized the RT_PREEMPT patched kernel and our Simulink target in the second case. Captured timing of the packets is depicted in Figure 2.2 and Figure 2.3. Ideally, the packets should be timed at the control loop sample time, which is 10 ms, which appears to be the case for the real-time setup. The non-real-time setup displays an average packet timing jitter of approximately 200 µs.

2.3 Conclusion

My goal was to move the MagMan from an old Raspberry Pi based platform to the new NVIDIA Jetson based platform, for which I developed a new position measuring system during the winter term. I moved the control algorithms using an automatic code generation from Simulink, based on our in-house real-time target. I also supplied the NVIDIA Jetson platform with a real-time patched kernel. This setup appears to be better than using the target provided by NVIDIA and the standard kernel, as it reduces the jitter of the coil current packets sent by the platform.

In addition, I benchmarked the new position measuring system against the old one. I did so in a real test case. The new system runs faster. Moreover, it does not overrun the targeted control loop sample time of 10 ms, unlike the old system, which does so for a control loop sample time of 20 ms.

3 Trajectory planning

Trajectory planning, loosely speaking, solves the task of getting some system, typically a mechanical one, from one state to some goal state. No more information is needed for the system to reach the goal state. That is where it differs from the task of so-called path planning. *Path planning* produces an obstacle-free geometric path, whereas a trajectory is a path in the state space, parametrized in time and satisfying dynamical constraints together with control inputs, which drive the system through the planned states.

In this chapter, I go over state of the art of trajectory planning. I then formulate the general trajectory planning task mostly from the system and control theory perspective. I apply this formulation to distributed magnetic manipulation, and, I discuss the way I solve the proposed planning problem.

3.1 State of the art

Trajectory planning problems often arise in many fields of science and technology. One such field is robotics, where trajectory planning is used to plan the motion of robots, from manipulators to fully autonomous robotic platforms. An increasingly popular way to solve these problems is using so-called sampling-based planners like *Rapidly-exploring random trees* (RRTs), first introduced in [LaV98]. These planners are asymptotically complete, which means they are able to find a collision-free path with the probability of one, as the number of steps goes to infinity. There even exists asymptotically optimal versions such as RRT*, presented in [KF11]. Paths planned by such a planner can be then smoothed out to fit dynamical constraints, parametrized in time to create a trajectory, and tracked using a feedback controller, as is proposed in [La 11]. For example, in [Ma+15], this is achieved using model prediction.

Formulating these problems as mathematical programs is often called *trajectory optimization*. An excellent introductory resource for trajectory optimization is chapter 10 in [Ted22]. Methods of trajectory optimization differ in how the planning problem is transcribed into the mathematical program. The paper [Kel17] offers great and practical overview of a few of these methods. Trouble arises when the formulated programs are non-convex, as they are prone to converging to local minima or, worse yet, local infeasibility, thus providing no solution at all.

This behaviour motivated the attempts to embed optimal control into sampling-based planners. For example, [WB13] presented an algorithm in which state space is sampled using RRT* and in which pairs of states are exactly and optimally connected. That is but for systems with linear controllable dynamics and unconstrained inputs. Moreover, [SL14] and

[Xie+15] implanted general nonlinear solvers into sampling-based algorithms for systems with nonlinear dynamics and constraints.

3.2 General formulation

I will consider the trajectory planning problem for a dynamical system represented by the *state equation*

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), \tag{3.1}$$

where $\mathbf{x}(t) \in \mathcal{X}$ is the *state* and $\mathcal{X} = \mathbb{R}^n$ the *state space*. Analogously $\mathbf{u}(t) \in \mathcal{U}$ is the *control input* and $\mathcal{U} = \mathbb{R}^m$ the *control input space*. If Equation (3.1) represents a mechanical system, such as a robot, then the state space is usually composed of positions and velocities, while the control inputs consist of individual forces, torques, voltages, and currents applied by and to actuators. Such a state space is often called the phase space by the robotics community.

One important property of this representation is that, to obtain the system's response to control input on some time interval $[t_\mathrm{I}, t_\mathrm{F}]$, only the state $\mathbf{x}_\mathrm{I}$ at the initial time is needed. The response is obtained via integrating

$$\mathbf{x}(t) = \mathbf{x}(t_\mathrm{I}) + \int_{t_\mathrm{I}}^{t} \mathbf{f}(\mathbf{x}(\tau), \mathbf{u}(\tau))\, \mathrm{d}\tau, \tag{3.2}$$

for every $t \in [t_\mathrm{I}, t_\mathrm{F}]$.

I will now introduce the notion of trajectory. A *trajectory* is a tuple $(\mathbf{x}(\cdot), \mathbf{u}(\cdot), T)$ with a given initial state $\mathbf{x}(0)$, such that $\mathbf{x} : [0, T] \to \mathcal{X}$, $\mathbf{u} : [0, T] \to \mathcal{U}$ and satisfying Equation (3.1) for every $t \in [0, T]$. Furthermore, I will call $\mathbf{x}(\cdot)$ *state trajectory*, $\mathbf{u}(\cdot)$ *control input trajectory* and T *control horizon*.

In a sense, a trajectory is just one of the solutions to the Equation (3.1) on a given time interval. The definition does not consider if a real system can execute the trajectory or not. There may be obstacles in the state space and actuator constraints, which make realizations of some trajectories impossible. To remedy that, I introduce $\mathcal{X}_\mathrm{free} \subseteq \mathcal{X}$, that is the part of the state space free of obstacles and satisfying imposed constraints, thus I call it *free state space*. Similarly, $\mathcal{U}_\mathrm{free} \subseteq \mathcal{U}$ is the *free input space*, that contains control inputs within some defined bounds. Now I have everything I need to formulate the task of trajectory planning.

▶ **Formulation 3.1 (Task of trajectory planning).** Given an initial state $\mathbf{x}_0$ and the goal region $\mathcal{X}_\mathrm{goal} \subseteq \mathcal{X}_\mathrm{free}$ the task of trajectory planning is to find a suitable trajectory $(\mathbf{x}(\cdot), \mathbf{u}(\cdot), T)$, satisfying following:

1. $\mathbf{x}(0) = \mathbf{x}_0$,
2. $\mathbf{x}(T) \in \mathcal{X}_\mathrm{goal}$,
3. $\mathbf{x}(t) \in \mathcal{X}_\mathrm{free}$ for every $t \in [0, T]$,

4. $\mathbf{u}(t) \in \mathcal{U}_{\text{free}}$ for every $t \in [0, T]$.

If the control horizon T is provided beforehand, the problem is called *fixed final-time*. Otherwise, it is called *free final-time*. ◀

The literature, for example [LaV06], usually calls this class of problems *motion planning under differential constraints*. Problems from this class are considered to be extremely difficult. Nevertheless, it may sometimes be possible to find the best trajectory in some sense. I will use optimal control theory to give concrete meaning to "best". Thus I formulate the task of optimal trajectory planning.

▶ **Formulation 3.2 (Task of optimal trajectory planning).** Extending the previously formulated task so that the resulting trajectory also minimizes some *cost functional* J is called the task of *optimal trajectory planning*. I will borrow the typical form of the cost functional from optimal control ([LVS12], p. 131), that is

$$J_{\mathbf{x}(\cdot),\mathbf{u}(\cdot),T} = \phi(\mathbf{x}(T)) + \int_0^T L(\mathbf{x}(t), \mathbf{u}(t))\,\mathrm{d}t. \tag{3.3}$$

Function ϕ weighs the state at the control horizon's end, while L is the additive cost which weighs both states and control inputs along the entire trajectory. I will call the planned trajectory optimal and denote it as $(\mathbf{x}^*(\cdot), \mathbf{u}^*(\cdot), T)$. ◀

Two different cost functionals often produce entirely different trajectories. The choice of the cost functional should be motivated by the demands on the real system's performance. Some popular cost choices can be found in [Kir04], pp. 29-34.

3.3 Trajectory planning for MagMan

The MagMan platform manipulates steel balls. Therefore, trajectory planning problems arise naturally. For example, given a steel ball at an initial position and some desired position, what route should the ball take and what forces need to be exerted by the magnetic field to move it along the way? What if obstacles are present? There may be even situations when it is desirable to reach not only some position but also velocity. For instance, zero velocity at the desired position to stop the ball. Things complicate further when multiple balls are taken into account, as they must not collide during the motion.

In this section, I seek to formalize these problems in the language of the previous section. Thus also preparing the ground for their solving. Typically I will search for trajectories in the state space comprised of the ball's position and velocities. As for control input, I will use the force acting on the balls. That is because the force-to-currents allocation for the platform is already solved. Still, while planning the force, I need to consider the constraints imposed by the force's origin.

3.3.1 Single ball

I start with formulating state space, control input space and constraints on them for a single ball case to later generalize it to cases with multiple balls.

State equation

The ball dynamics is modelled as linear. Thus the state equation can be written out using standard matrix form

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t). \tag{3.4}$$

I introduce state and control input as

$$\mathbf{x}(t) = \begin{bmatrix} x(t) \\ \dot{x}(t) \\ y(t) \\ \dot{y}(t) \end{bmatrix} \quad \text{and} \quad \mathbf{u}(t) = \begin{bmatrix} F_x(t) \\ F_y(t) \end{bmatrix}. \tag{3.5}$$

Following from the mathematical model introduced in Section 1.2.2 the matrices take form of

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{and} \quad \mathbf{B} = \begin{bmatrix} 0 & 0 \\ 1/m_{\text{eff}} & 0 \\ 0 & 0 \\ 0 & 1/m_{\text{eff}} \end{bmatrix}. \tag{3.6}$$

It is evident that $\mathcal{X} = \mathbb{R}^4$ and $\mathcal{U} = \mathbb{R}^2$.

Platform bounds

As for the free state space, I need to take physical bounds of the platform into account, that is,

$$x_{\text{lb}} \leq x \leq x_{\text{ub}} \quad \text{and} \quad y_{\text{lb}} \leq y \leq y_{\text{ub}}, \tag{3.7}$$

where $x_{\text{lb}} = y_{\text{lb}} = -25\,\text{mm}$ and $x_{\text{ub}} = y_{\text{ub}} = 200\,\text{mm}$.

Obstacle avoidance

Next, I want to discuss the question of obstacle avoidance. On the platform, there may be present some obstacle regions. The balls must not collide with these regions. Such a feat can be achieved by imposing another constraint on the state space in the form of

$$\text{dist}_O(x, y) \geq r_{\text{ball}} + r_\varepsilon, \quad \forall O \in \overline{O}. \tag{3.8}$$

Function dist_O provides the distance from the centre of the ball to the boundary of the obstacle region O and $\overline{O}$ is the set of all obstacle regions. The situation is depicted in

Figure 3.1. To avoid the obstacle the distance must be greater than the ball's radius. I add safety margin r_ε to increase robustness.

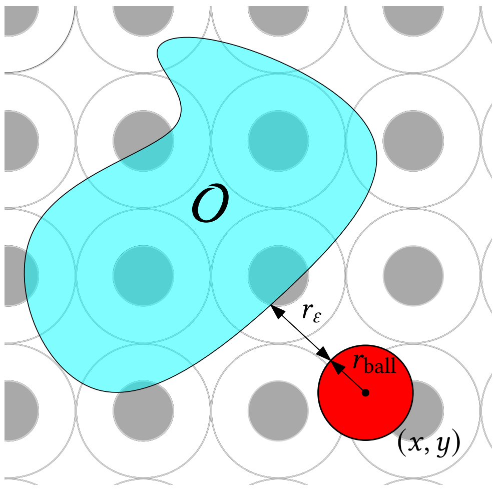


Figure 3.1: Obstacle avoidance

Control inputs

The formulation gets more complicated when I try to express the free control input space. From the nature of the magnetic field, the maximal and minimal forces that can act on the steel ball are highly position-dependent. Moreover, the force bounds are coupled. For example, exerting maximal force in the x-direction may severely limit the range of force in the y-direction. Therefore, the most straightforward way to impose constraints on the control inputs is to have them satisfy the force-current model from Equation (1.13). That is

$$\mathbf{u} = \mathbf{G}(\mathbf{x})\mathbf{v}, \tag{3.9}$$

for some $\mathbf{v}$, such that

$$\mathbf{0} \leq \mathbf{v} \leq \mathbf{1}. \tag{3.10}$$

3.3.2 Multiple balls

The previous single ball formulation easily extends to multiple balls for the most part.

State equation

First, I introduce the extended state equation where state and control input are created by conjugating M single-ball systems from the previous section (see Equation (3.4))

$$\dot{\overline{\mathbf{x}}}(t) = \overline{\mathbf{A}}\overline{\mathbf{x}}(t) + \overline{\mathbf{B}}\overline{\mathbf{u}}(t), \tag{3.11}$$

where

$$\overline{\mathbf{x}}(t) = \begin{bmatrix} \mathbf{x}_1(t) \\ \mathbf{x}_2(t) \\ \vdots \\ \mathbf{x}_M(t) \end{bmatrix} \quad \text{and} \quad \overline{\mathbf{u}}(t) = \begin{bmatrix} \mathbf{u}_1(t) \\ \mathbf{u}_2(t) \\ \vdots \\ \mathbf{u}_M(t) \end{bmatrix}. \tag{3.12}$$

The matrices then take the form of

$$\overline{\mathrm{A}} = \begin{bmatrix} \mathrm{A} & 0 & \cdots & 0 \\ 0 & \mathrm{A} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathrm{A} \end{bmatrix} \quad \text{and} \quad \overline{\mathrm{B}} = \begin{bmatrix} \mathrm{B} & 0 & \cdots & 0 \\ 0 & \mathrm{B} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathrm{B} \end{bmatrix}. \tag{3.13}$$

Then $\mathcal{X} = \mathbb{R}^{4M}$ and $\mathcal{U} = \mathbb{R}^{2M}$. I impose the platform bounds and obstacle avoidance constraints on each ball separately.

Collision avoidance

Unlike with a single ball, the state space is constrained not only by the physical bounds of the platform and obstacle avoidance but also by collision avoidance between individual balls. I try to prevent collisions by imposing

$$\underbrace{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}_{d(x_i, y_i, x_j, y_j)} \geq 2r_{\text{ball}} + r_\varepsilon \quad \forall i, j \in \mathcal{Q}_M, \tag{3.14}$$

where

$$\mathcal{Q}_M = \left\{(i, j) \in \mathcal{I}_M^2 \mid i > j\right\}. \tag{3.15}$$

Figure 3.2 sheds some light on this situation. The distance of the centers of the balls has to be greater then the sum of their radii to prevent a collision. In addition to that, I also include a safety margin $r_\varepsilon > 0$, as the balls influence the magnetic field in their vicinity and other uncertainties are present.

Control inputs

To be complete, I deal with the control inputs constraints in the same fashion as before. I introduce the force model for the conjugate states

$$\overline{\mathrm{G}}(\overline{\mathbf{x}}) = \begin{bmatrix} \mathrm{G}(\mathbf{x}_1) \\ \mathrm{G}(\mathbf{x}_2) \\ \vdots \\ \mathrm{G}(\mathbf{x}_N) \end{bmatrix}. \tag{3.16}$$

Then the control inputs must satisfy

$$\overline{\mathbf{u}} = \overline{\mathbf{G}}(\overline{\mathbf{x}})\mathbf{v} \tag{3.17}$$

for some $\mathbf{v}$, such that

$$\mathbf{0} \leq \mathbf{v} \leq \mathbf{1}. \tag{3.18}$$

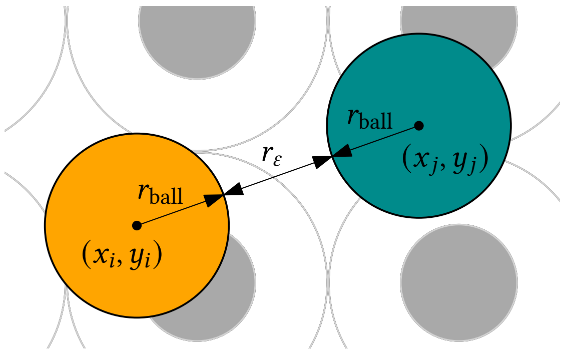


Figure 3.2: Collision avoidance illustration.

3.3.3 Proposed solution

I am planning a trajectory. Therefore I need to satisfy dynamical constraints and also find appropriate control inputs. Formally speaking, I am looking to solve a type of task called the *Two-Point Boundary Value Problem* (TP-BVP). Generally, TP-BVPs are hard to solve. Mathematical optimization methods are employed for their solving. Hence, I propose solving the task at hand using optimization, often called trajectory optimization. Such an approach formulates the trajectory planning task as a mathematical program, which is then solved using a solver.

Mathematical programs include a criterion, which is supposed to be minimized. By suitable choice of such criterion, I am effectively solving the task of optimal trajectory planning as introduced in Formulation 3.2. Furthermore, mathematical programs often benefit from an initial guess of the solution. In the case of trajectory, a path can serve as an initial guess. Thus, I can incorporate a path planning algorithm into the trajectory planning process. Presumably, a path that a sampling-based planner generates.

4 Trajectory optimization

In this section, I discuss the formulation of trajectory planning for distributed magnetic manipulation as a mathematical program. Then I present a way of solving it.

4.1 Designing the cost

Choosing the correct cost is essential to formulating a mathematical program. As I already stated, the different costs can produce very different solutions. I will now present a few cost options which are suitable for planning trajectories for the MagMan platform. Also, I will follow the cost functional notation from Equation (3.3).

4.1.1 Minimum-time

If no control horizon is provided beforehand, it may be desirable to get the ball or balls to their desired state as fast as possible. I can achieve this by letting the cost equal to the control horizon, that is

$$J = T. \tag{4.1}$$

I want to point out that the control horizon is now the optimization variable.

4.1.2 Minimal-control-effort

In optimal control theory, typical cost formulation-usually called *minimal control effort* ([Kir04], p. 33)-has the additive cost

$$L = \mathbf{u}^{\mathrm{T}}(t)\mathbf{u}(t), \tag{4.2}$$

while the final weight is

$$\phi = 0. \tag{4.3}$$

Therefore the cost functional is

$$J = \int_0^T \mathbf{u}^{\mathrm{T}}(t)\mathbf{u}(t)\ \mathrm{d}t. \tag{4.4}$$

As I previously formulated, the sought control inputs have the purpose of the desired force to be exerted by the magnetic field on the steel balls. Therefore, I call this functional *Force cost.*

Another candidate cost is

$$L = \mathbf{v}^{\mathrm{T}}(t)\mathbf{v}(t), \tag{4.5}$$

and the resulting cost functional is

$$J = \int_0^T \mathbf{v}^{\mathrm{T}}(t)\mathbf{v}(t)\ \mathrm{d}t. \tag{4.6}$$

As a reminder to the reader, $\mathbf{v}(t)$ corresponds to a scaling factor proportional to the coil current as introduced in Equation (1.11). Even though I am searching for the desired force, I will still need to optimize over the "currents" to satisfy the force-current model. Therefore, it is reasonable to search for control inputs which could minimize the currents, thus reducing current consumption. Furthermore, I call the resulting cost functional *Current cost*.

4.1.3 Penalizing the force-position dependence

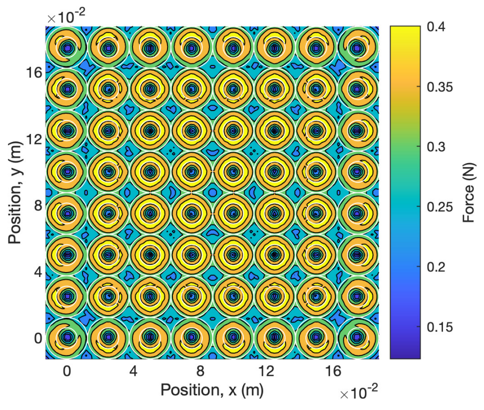
The force the magnetic field exerts on the steel ball is highly spatially dependent. Figure 4.1 (a) illustrates this fact. Exerting force in regions where the force bounds change a lot with position is unwanted. That is mostly related to the problem of feedback tracking the planned trajectories. Suppose the real and planned trajectories diverge so that the maximal force achievable in the real case is much lower than in the planned matter. In that case, it may lead to the trajectories diverging even more as the feedback controller may not be able to exert the necessary correcting force. I can penalize this behaviour by the following additive cost

$$L_{\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\mathbf{x}}} = \left\| \frac{\mathrm{d}\mathbf{u}(t)}{\mathrm{d}\mathbf{x}} \right\|_{\mathrm{F}}^2 = \left\| \frac{\mathrm{d}}{\mathrm{d}\mathbf{x}} [\mathbf{G}(\mathbf{x}(t))\mathbf{v}(t)] \right\|_{\mathrm{F}}^2 \tag{4.7}$$

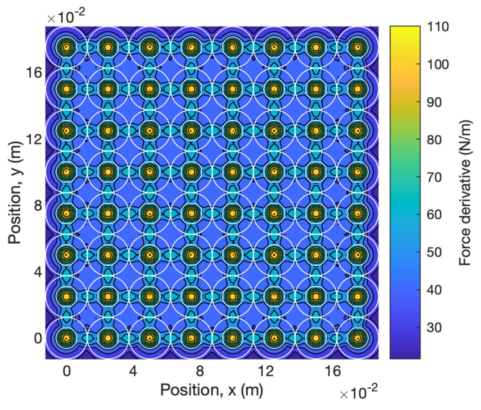
the magnitude of the force spatial derivative squared. As a magnitude, I denote the Frobenius norm of the resulting Jacobian matrix. The maximal values of the force space derivative attainable are plotted in figure Figure 4.1 (b). In a real scenario, to reduce the force-position of the planned trajectory, I add the newly introduced term to some previously introduced cost functional as

$$J = \int_0^T L + \eta L_{\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\mathbf{x}}}\ \mathrm{d}t, \tag{4.8}$$

where $\eta > 0$.



(a) Force



(b) Spatial derivative

Figure 4.1: The simulated maximal force possible exerted and its spatial derivative. Simulated for steell ball with a radius of 10 mm. I performed the simulation by discretizing the platform into segments of 0.175×0.175 mm and solving mathematical program for each segment.

4.2 Direct transcription

The way I formulate the mathematical program is a so-called *direct transcription* as presented, for example, in [Bet09] p. 132. In this approach, I first discretize the continuous-time dynamics of the system and then develop a mathematical program in a finite amount of variables.

4.2.1 Discretization in time

I will denote *discrete-time trajectory* as $(\mathbf{x}[\cdot], \mathbf{u}[\cdot], N)$ and the discretized system as

$$\mathbf{x}[n+1] = \mathbf{f}_\mathrm{d}(\mathbf{x}[n], \mathbf{u}[n]), \tag{4.9}$$

where $\mathbf{x}[n], \mathbf{u}[n]$ are the values of state and control input at the time hn given some *timestep* h and N is the *discrete control horizon.* There are various ways to discretize dynamical systems in time, and I choose the very popular *Runge-Kutta scheme of the 4th order.* Which has the following form

$$\mathbf{f}_\mathrm{d}(\mathbf{x}[n], \mathbf{u}[n]) = \mathbf{x}[n] + \mathbf{k}_1/6 + \mathbf{k}_2/3 + \mathbf{k}_3/4 + \mathbf{k}_4/6, \tag{4.10}$$

where

$$
\begin{aligned}
\mathbf{k}_1 &= h\mathbf{f}(\mathbf{x}[n], \mathbf{u}[n]),\\
\mathbf{k}_2 &= h\mathbf{f}(\mathbf{x}[n] + \mathbf{k}_1/2, , \mathbf{u}[n]),\\
\mathbf{k}_3 &= h\mathbf{f}(\mathbf{x}[n] + \mathbf{k}_2/2, \mathbf{u}[n]),\\
\mathbf{k}_4 &= h\mathbf{f}(\mathbf{x}[n] + \mathbf{k}_3, \mathbf{u}[n]).
\end{aligned}
\tag{4.11}
$$

4.2.2 Mathematical program

The trajectory planning problem for the distributed magnetic manipulation formulated using the direct transcription is

$$
\begin{aligned}
&\min_{\overline{\mathbf{x}}[\cdot],\overline{\mathbf{u}}[\cdot],\overline{\mathbf{v}}[\cdot]} \sum_{k=1}^{N-1} L(\overline{\mathbf{x}}[n], \overline{\mathbf{u}}[n], \overline{\mathbf{v}}[n]) \quad \text{or} \quad \min_{\substack{T,\\ \overline{\mathbf{x}}[\cdot],\overline{\mathbf{u}}[\cdot],\overline{\mathbf{v}}[\cdot]}} T && \text{(4.12a)}\\
&\text{s.t.} \quad \overline{\mathbf{x}}[n+1] = \overline{\mathbf{f}}_\mathrm{d}(\overline{\mathbf{x}}[n], \overline{\mathbf{u}}[n]), \qquad \forall n \in \mathcal{I}_{N-1}, && \text{(4.12b)}\\
&\overline{\mathbf{u}}[n] = \overline{\mathbf{G}}(\overline{\mathbf{x}}[n])\mathbf{v}[n], \qquad \forall n \in \mathcal{I}_{N-1}, && \text{(4.12c)}\\
&\mathbf{0} \le \mathbf{v}[n] \le \mathbf{1}, \qquad \forall n \in \mathcal{I}_{N-1}, && \text{(4.12d)}\\
&x_\mathrm{lb} \le x_l[n] \le x_\mathrm{ub}, \qquad \forall l \in \mathcal{I}_M, \quad \forall n \in \mathcal{I}_{N-1}, && \text{(4.12e)}\\
&y_\mathrm{lb} \le y_l[n] \le y_\mathrm{ub}, \qquad \forall l \in \mathcal{I}_M, \quad \forall n \in \mathcal{I}_{N-1}, && \text{(4.12f)}\\
&d(x_i[n], y_i[n], x_j[n], y_j[n]) \ge (2r_\mathrm{ball} + r_\varepsilon), \quad \forall i,j \in \mathcal{Q}_M, \quad \forall n \in \mathcal{I}_{N-1} && \text{(4.12g)}\\
&\overline{\mathbf{x}}[1] = \overline{\mathbf{x}}_\mathrm{I}, && \text{(4.12h)}\\
&\overline{\mathbf{x}}[N] = \overline{\mathbf{x}}_\mathrm{F}, && \text{(4.12i)}\\
&+ \text{obstacle avoidance constraints}, && \text{(4.12j)}
\end{aligned}
$$

where the obstacle avoidance constraints are

$$
\mathrm{dist}_O(x_l[n], y_l[n]) \ge (r_\mathrm{ball} + r_\varepsilon), \quad \forall O \in \overline{O}, \quad \forall l \in \mathcal{I}_M, \quad \forall n \in \mathcal{I}_{N-1}. \tag{4.13}
$$

The cost functions in Equation (4.12a) can either be the cost functional, where the integral is approximated by the sum of the additive cost values at discrete points in time or the control horizon as in Equation (4.1). I need to point out one crucial fact. When using the minimal-time cost, the discretization time step h stretches or shrinks based on the value of the control horizon, as is given as

$$
h = \frac{T}{N-1}. \tag{4.14}
$$

The constraints are to be interpreted as follows. The Equation (4.12b) represents satisfying the discretized dynamical constraints of the multi-ball system introduced in Equation (3.11). Equation (4.12c) and Equation (4.12d) then represent satisfying the current-force model from Equation (3.17) and Equation (3.18). Constrains in Equation (4.12e) and Equation (4.12f) enforce the platforms bounds from Equation (3.7). Equation (4.12g) is the non-collision constraint introduced in Equation (3.14). Equation (4.12h) and Equation (4.12i) represent the initial and final state conditions, that is the positions and velocities of the balls. Last but not least, Equation (4.13) describes the obstacle avoidance constraint introduced in Equation (3.8).

4.3 Solving the mathematical programs

To solve the given mathematical programs, I formulated them in MATLAB using CasADi [And+18]. CasADi is an open-source framework for nonlinear optimization and algorithmic differentiation. Also, this formulation is the so-called *nonlinear program* (NLP), which requires a general solver to find the solution. Fortunately, CasADi comes with IPOPT, a solver for large-scale nonlinear optimization problems based on the interior point method. According to [Par+16], IPOPT appears to be the better choice for motion planning problems than, for example, SNOPT, which is commercial and based on a sequential quadratic programming method.

4.3.1 Differentiable obstacles

The IPOPT solvers require the constraints to be twice differentiable. For example, distance to a rectangular region is not twice differentiable. I decided to approximate obstacle regions using circles, as the signed distance from circles is given as

$$d(x, y) = \sqrt{(x - c_x)^2 + (y - c_y)^2} - r, \tag{4.15}$$

and therefore is twice differentiable (except (c_x, c_y)). One such obstacle approximation is shown in Figure 4.2.

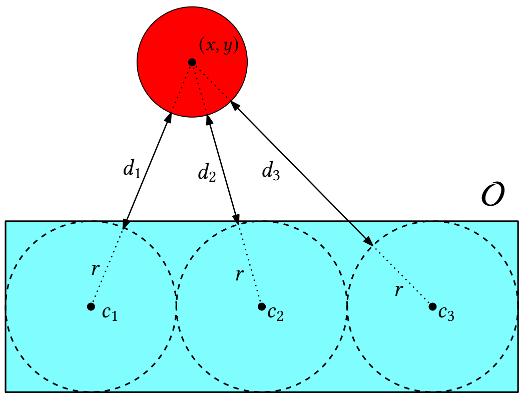
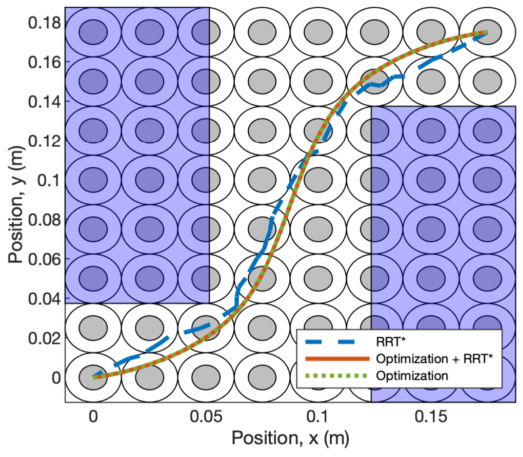


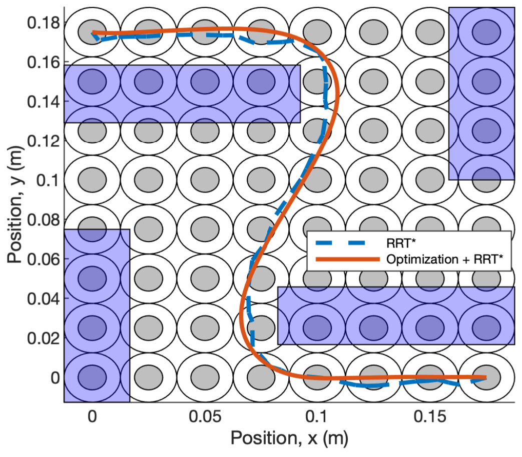
Figure 4.2: Aproximation of the obstacle region using circles.

4.3.2 Initialization using RRT*

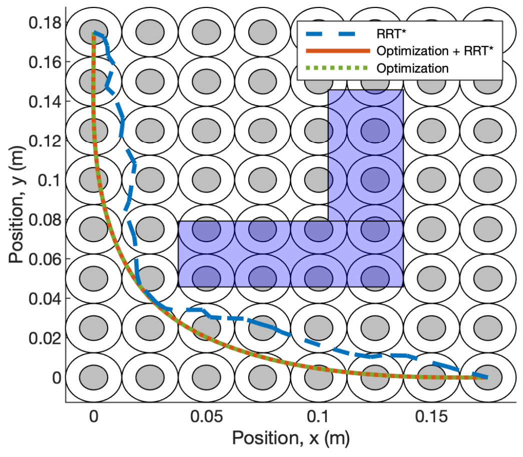
The formulated programs are non-convex, meaning the solutions may not be global minima but just local ones. To which minimum the optimization converges depends highly on the initial guess I provide to the solver. This behaviour is easily graspable by looking at the map of maximal possible forces 4.1 (a). Given some initial path, there is a local minimum created by the coils in the immediate vicinity of such path.



(a) Enviroment 1



(b) Enviroment 2



(c) Enviroment 3

Figure 4.3: Comparison of paths generated by RRT*, paths obtained from optimization initialized by the RRT* and paths obtained from optimization without initialization.

It then makes sense to provide the solver with the most promising initial guess I can.

To that end, I employ the RRT* algorithm [KF11]. Using the sampling-based algorithm, I can plan close to the shortest obstacle-free path and, in the case of multiple balls, even a collision-free path. Practically, I obtain a set of waypoints from the algorithm, which I then fit with splines creating a path. Initial positions of the state variables then correspond to the sampled version of these splines. I tested the effects of RRT* initialization of the mathematical programs on three different obstacles environments. In Figure 4.3, I present the comparison of RRT* planned paths and the paths obtained by solving the mathematical programs, both with and without RRT*. In Table 4.1, I then present the effects of the RRT* initialization on the solving time of the mathematical programs. Notably, for environment 2 without the RRT* initialization, the solver detected the problem as infeasible and failed to provide a solution.

Table 4.1: Conparison of solving time of NLPs presented in Figure 4.3. The control horizon for all problems was 1.5 s and the force cost forumulation was chosen.

Environment	Solution time when initialized with RRT* (s)	Solution time without initialization (s)
1	8.30	51.02
2	24.60	-
3	17.08	73.46

4.4 Two-phase optimization

The control loop of the MagMan platform runs at a fixed rate of 100 Hz. Therefore, for the purpose of feedback trajectory tracking, the planned trajectory need to be sampled at the same rate. This is not the problem for the fixed control horizon task, as the number of samples can be adapted to fit this requirement. In that case, the number of samples is given as

$$N = \left\lfloor \frac{T}{T_s} \right\rfloor + 1, \tag{4.16}$$

where $T_s = 0.01$ s is control loop sample rate.

When solving the minimum time problem, things are not as straightforward. The time step depends on the control horizon, which is a variable I optimize over. Hence I propose a way of solving that, a *two-phase optimization.* In the first phase, I solved the minimal time problem with a fixed amount of samples. In the second phase, I then solve the fixed final time problem with the number of samples given by 4.16 and initialized by the solution of the first phase. The result of the second phase is then a solution to the minimum time problem with the desired time step.

4.5 Algorithms

To be complete, I provide the trajectory planning algorithms that arise from the previous discussion.

4.5.1 Fixed control horizon

In the case of the fixed control horizon T, initial and final ball states $\overline{\mathbf{x}}_\mathrm{I}, \overline{\mathbf{x}}_\mathrm{F}$, I plan the trajectory using Algorithm 1. The employed mathematical program formulation is the additive cost variant of Equation (4.12a).

Algorithm 1: Trajectory planning algorithm

```
  Input: $\overline{\mathbf{x}}_\mathrm{I}, \overline{\mathbf{x}}_\mathrm{F}, T$
  Output: $\overline{\mathbf{x}}[\cdot], \overline{\mathbf{u}}[\cdot]$
1 Set $\overline{\mathbf{q}}_\mathrm{I}$ and $\overline{\mathbf{q}}_\mathrm{F}$ to initial and final ball positions from $\overline{\mathbf{x}}_\mathrm{I}$ and $\overline{\mathbf{x}}_\mathrm{F}$;
2 Find collision and obstacle free path $\overline{\mathbf{q}}(\cdot)$ between $\overline{\mathbf{q}}_\mathrm{I}$ and $\overline{\mathbf{q}}_\mathrm{F}$ using RRT*;
3 Set $N = \lfloor T/T_s \rfloor + 1$;
4 Initialize $\overline{\mathbf{x}}[\cdot]$ of length $N$ using $\overline{\mathbf{q}}(\cdot)$;
5 Solve mathematical program to obtain $\overline{\mathbf{x}}[\cdot]$ and $\overline{\mathbf{u}}[\cdot]$;
```

4.5.2 Minimum-time

In the case of the minimum-time problem, I plan the trajectory using Algorithm 2. The mathematical program employed in the first phase is the minium-time variant of Equation (4.12a). I need to provide the algorithm with initial discrete-time control horizon N' , on which the first phase mathematical program is solved. This is tricky as having too short of a horizon can lead to the infeasibility of the NLP. The mathematical program formulation used in the second phase is the additive cost variant of Equation (4.12a).

Algorithm 2: Minimum-time, two-phase optimization trajectory planning algorithm

```
  Input: $\overline{\mathbf{x}}_\mathrm{I}, \overline{\mathbf{x}}_\mathrm{F}, N'$
  Output: $\overline{\mathbf{x}}[\cdot], \overline{\mathbf{u}}[\cdot]$
1 Set $\overline{\mathbf{q}}_\mathrm{I}$ and $\overline{\mathbf{q}}_\mathrm{F}$ to initial and final ball positions from $\overline{\mathbf{x}}_\mathrm{I}$ and $\overline{\mathbf{x}}_\mathrm{F}$;
2 Find collision and obstacle free path $\overline{\mathbf{q}}(\cdot)$ between $\overline{\mathbf{q}}_\mathrm{I}$ and $\overline{\mathbf{q}}_\mathrm{F}$ using RRT*;
3 Initialize $\overline{\mathbf{x}}'[\cdot]$ of length $N'$ using $\overline{\mathbf{q}}(\cdot)$;
4 Solve mathematical program to obtain $\overline{\mathbf{x}}'[\cdot], \overline{\mathbf{u}}'[\cdot]$ and $T'$;
5 Set $N = \lfloor T'/T_s \rfloor + 1$;
6 Initialize $\overline{\mathbf{x}}[\cdot], \overline{\mathbf{u}}[\cdot]$ of length $N$ using $\overline{\mathbf{x}}'[\cdot], \overline{\mathbf{u}}'[\cdot]$;
7 Solve mathematical program to obtain $\overline{\mathbf{x}}[\cdot], \overline{\mathbf{u}}[\cdot]$;
```

The MATLAB + CasADi implementation of these algorithms can be found in the attachment. The structure of the implementation is presented in Appendix B.

4.6 Generalization

MagMan is not the only platform for distributed manipulation developed at AA4CC, as there are also DEPMan and AcouMan (see Section 1.1). Even though these platforms use different physical force fields to achieve the actuation of the manipulated objects, they have a lot in common. The dynamics of the manipulated objects are described by some state equation

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}), \tag{4.17}$$

where the actuating force $\mathbf{u}$ is obtained from the force model

$$\mathbf{u} = \mathbf{g}(\mathbf{x}, \mathbf{v}) \tag{4.18}$$

for some control input $\mathbf{v} \in \mathcal{V}$, where $\mathcal{V}$ is the set of admissible control inputs. The platforms also encompass the same feedback linearization layer as MagMan (refer to Figure 1.3). Therefore allow for control based on desired forces.

Theoretically, the trajectories for these platforms could be planned in the same manner as I do for MagMan. The dynamics of the manipulated objects can be discretized in time, and the following mathematical program can be developed

$$\begin{aligned}
\min_{\mathbf{x}[\cdot],\mathbf{u}[\cdot],\mathbf{v}[\cdot]} \quad & \sum_{n=1}^{N-1} L(\mathbf{x}[n], \mathbf{u}[n], \mathbf{v}[n]) && \\
\text{s.t.} \quad & \mathbf{x}[n+1] = \mathbf{f}_\mathrm{d}(\mathbf{x}[n], \mathbf{u}[n]), && \forall n \in \mathcal{I}_{N-1}, \\
& \mathbf{u}[n] = \mathbf{g}(\mathbf{x}[n], \mathbf{v}[n]), && \forall n \in \mathcal{I}_{N-1}, \\
& \mathbf{v}[n] \in \mathcal{V}, && \forall n \in \mathcal{I}_{N-1}, \\
& \mathbf{x}[1] = \mathbf{x}_\mathrm{I}, && \\
& \mathbf{x}[N] = \mathbf{x}_\mathrm{F}, && \\
& + \text{platform bounds}, && \\
& + \text{collision and obstacle constraints}. &&
\end{aligned} \tag{4.19}$$

Such a program can be solved in the same way I do for MagMan. That is, initialized by the path obtained from RRT* algorithm.

5 Trajectory tracking

As I solved the problem of finding a feasible trajectory in the previous section, now comes the time to track it. More precisely, to design a controller which will ensure that the balls will perform the planned motion even when disturbances and model inaccuracies are present. I present the design for a single ball only as tracking multiple balls is analogous.

5.1 Controller design

Given the planned optimal discrete-time trajectory $(\mathbf{x}^*[\cdot], \mathbf{u}^*[\cdot], N)$ my goal is to control the variation

$$\delta\mathbf{x}[n] = \mathbf{x}[n] - \mathbf{x}^*[n], \quad \delta\mathbf{u}[n] = \mathbf{u}[n] - \mathbf{u}^*[n] \tag{5.1}$$

for the real system trajectory $\mathbf{x}[n], \mathbf{u}[n]$. It is also resonable to require

$$\delta\mathbf{x}[n] \approx \mathbf{0}, \quad n > N, \tag{5.2}$$

in other words, to keep the balls in their final states after the trajectory ends. Therefore, I first extend the trajectory to the infinite horizon as

$$\mathbf{x}^*[n] = \mathbf{x}^*[N] \quad \text{and} \quad \mathbf{u}^*[n] = \mathbf{0}, \quad k > N. \tag{5.3}$$

I can then achieve trajectory stabilization using the so-called *Linear-Quadratic-Regulator* (LQR), which minimizes the cost

$$J = \frac{1}{2}\sum_{k=1}^{\infty} \delta\mathbf{x}^{\mathrm{T}}[n]\mathbf{Q}\delta\mathbf{x}[n] + \delta\mathbf{u}^{\mathrm{T}}[n]\mathbf{R}\delta\mathbf{u}[n] \tag{5.4}$$

where $\mathbf{Q} \succeq 0$ and $\mathbf{R} \succ 0$. This is done by utilizing a the full-state feedback gain $\mathbf{K}$, which gives the following control law

$$\mathbf{u}[n] = \underbrace{\mathbf{u}^*[n]}_{\text{Feedforward}} \underbrace{-\mathbf{K}(\mathbf{x}[n] - \mathbf{x}^*[n])}_{\text{Feedback}}. \tag{5.5}$$

Notice that two components form the actual control input. One is the feedforward part constituting the optimal control input trajectory; the second one is the feedback part compensating for the deviations from the planned state trajectory. I designed the LQR in MATLAB for the time discretized ball system using the `lqr` function using experimentaly chosen weight matrices

$$\mathbf{Q} = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix} \quad \text{and} \quad \mathbf{R} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}. \tag{5.6}$$

5.2 State estimation

As I previously stated, the states I am considering are composed of positions and velocities of the individual balls. One challenge arises here. I cannot directly measure the velocity component of the state, as the feedback is provided only for the position. Therefore I need to approximate the velocities of the individual balls. I do so using the *filtered derivative system*, given by the transfer function

$$D(z) = \frac{z-1}{Tz + T_s - T}, \tag{5.7}$$

where T is time filter time constant and $T_s = 0.01$ s is the control loop sample time. From the ball position measurements $x(z), y(z)$, I then obtain the single ball state estimate as

$$\hat{\mathbf{x}}(z) = \underbrace{\begin{bmatrix} 1 & 0 \\ D(z) & 0 \\ 0 & 1 \\ 0 & D(z) \end{bmatrix}}_{\mathbf{E}(z)} \begin{bmatrix} x(z) \\ y(z) \end{bmatrix} = \begin{bmatrix} x(z) \\ D(z)x(z) \\ y(z) \\ D(z)y(z) \end{bmatrix}. \tag{5.8}$$

The resulting control loop is shown in 5.2.

The derivative is filtered to reduce noise. Filtering reduces the noise but, unfortunately, introduces a phase delay, there is a tradeoff. I received the best tracking results for filtering time constant value $T = 0.02$ s. Higher values led to greater smoothing, but the phase delay resulted in overshoot and steady-state oscillations. Two actual responses are shown in Figure 5.1. I compare the live estimate using the filtered derivative system and differentiating position response fitted using cubic splines.

Kalman filter

As the velocity estimation is far from optimal, I also attempted to design a Kalman filter, which, in theory, should provide a smoother response with lesser phase delay than a model-free method. I estimated the observation noise covariance matrix from measured data and tried to find the process noise covariance matrix experimentally. I received worse results than with mere filtred derivative. I blame that on the disparity between the model and reality. It is known that there is a significant difference between the force, which should be based on the mathematical model exerted on the ball and the actual force being exerted. Therefore it makes sense to receive better results from purely model-free estimation.

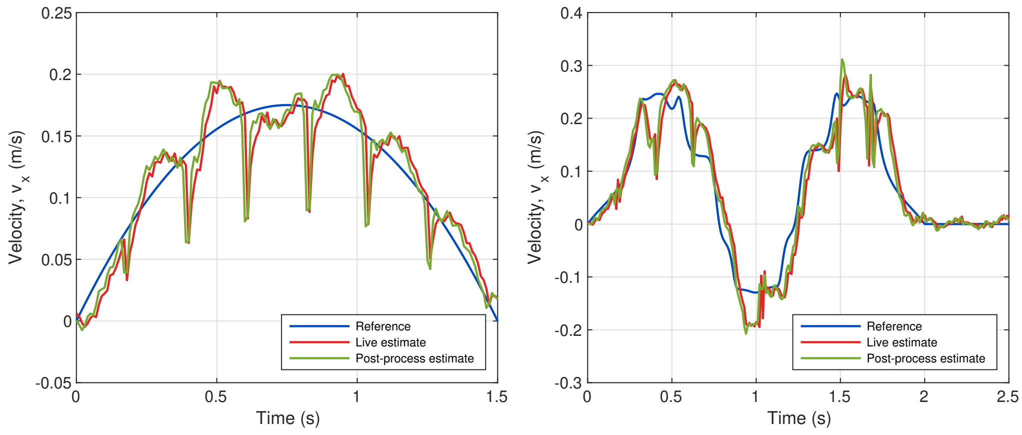


Figure 5.1: Tracking the velocity trajectory. Comparison of state velocity using filtered derivative system with $T = 0.02$ s and velocity obtained from fitting the position response with cubic splines and differentiating.

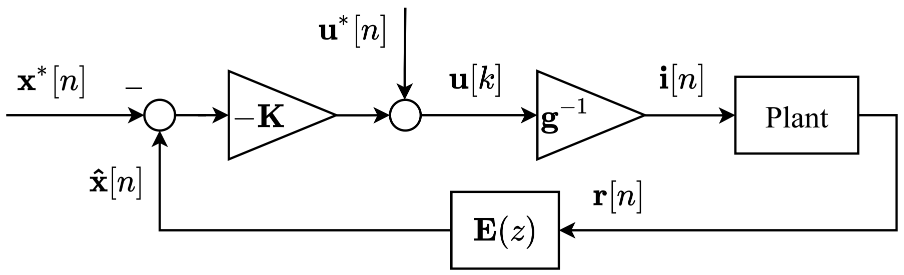


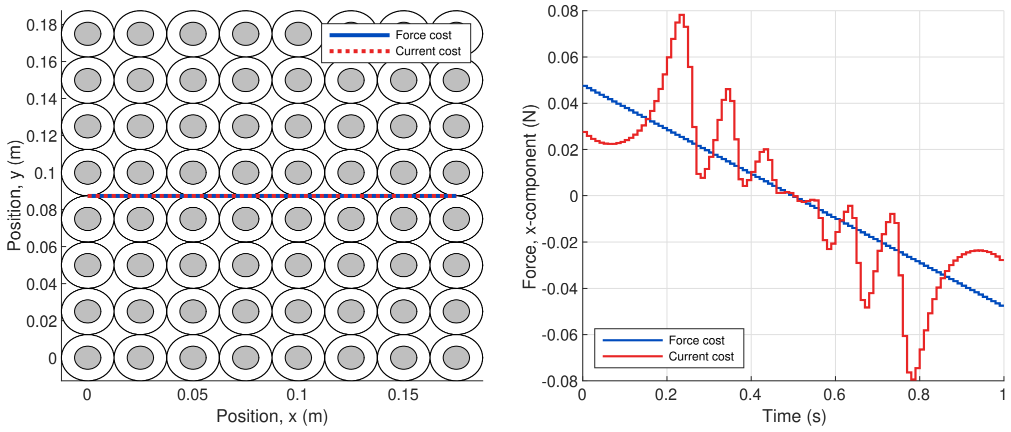
Figure 5.2: Trajectory tracking control loop. The g^{-1} gain represents feedback linearization from Figure 1.3, $\mathbf{K}$ is the full-state feedback gain and $\mathbf{E}(z)$ is the state estimation system from Equation (5.8).

6 Experiments & results

In this chapter, I describe and evaluate planned trajectories and experiments I conducted on the real platform. I examine the distinctions originating from using different cost formulations and penalizing the position dependence of the force. For each experiment, I typically present the planned trajectory first and then describe and evaluate the tracking of the trajectory on the actual MagMan.

6.1 Simple trajectory

I explored the differences in the planned trajectories arising from using the current or the force cost formulations as introduced in Equation (4.4) and Equation (4.6), respectively. At first, I chose a simple goal for ease of demonstration: moving a ball 87.5 mm in the x-direction while also stopping it at the end.

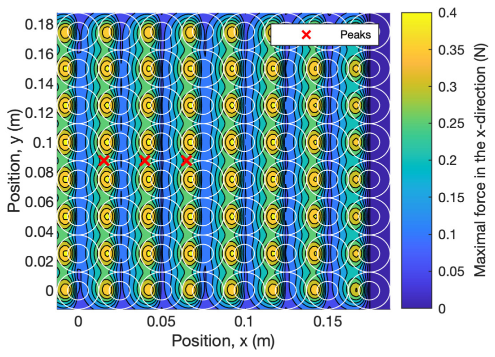


(a) Path **(b)** Force x-component

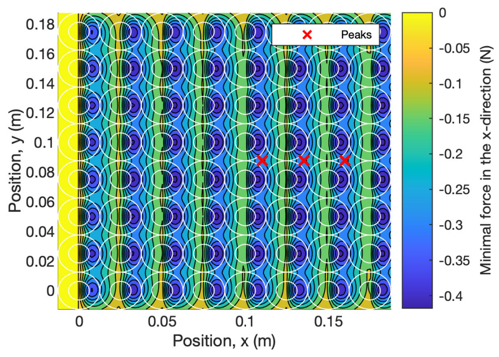
Figure 6.1: Purely horizontal 87.5 mm trajectory. Results of both cost formulations from Equation (4.4) and Equation (4.6) are shown. The control horizon was chosen to be 1 s.

6.1.1 Planned trajectory

The ball takes a straight path to the goal for both cost formulations, as is shown in Figure 6.1 (a). This also means the force is on the ball exerted only in the x-direction, as I present in Figure 6.1 (b). I chose 1 s as the control horizon, which resulted in solving the mathematical program taking 7.16 s for the force cost formulation and 5.25 s for the current cost formulation. In the case of the current cost formulation, the planned force looks something akin to bang-bang control. That is, switching between two extremal values. However, instead of two extremal values, the ball is set into motion and later stopped by a set of short impulses of force, seen as peaks in Figure 6.1 (b). I plotted the positions of the ball, where those peaks of force are planned to be exerted into maps of maximal and minimal forces obtainable in the x-direction. The positions of the peaks which move the ball are included in the maximal forces map Figure 6.2 (a). Likewise, the peaks which stop the ball are included in the minimal forces map Figure 6.2 (b). Notice the placement of the peaks. They are in close proximity to the maximal, respectively, minimal values of force attainable. This behaviour makes sense, as in those positions, the most force for the least amount of current can be achieved.



(a) Maximal



(b) Minimal

Figure 6.2: Maps of maximal and minimal forces obtainable in the x-direction. Including the positions of the force peaks from Figure 6.1 (b). The map was calculated by discretizing the platform and solving a mathematical program for each segment.

6.1.2 Tracking experiment

I conducted experiments to evaluate the trackability of the planned trajectory on the actual platform. The results of these experiments are shown in Figure 6.3. I include the responses of the ball's x-position for both cost formulations, just as before. As for the actual results, the trajectory is tracked satisfyingly in both cases. The force cost formulation seems to result in a more significant overshoot. I observed this consistently over more experiments.

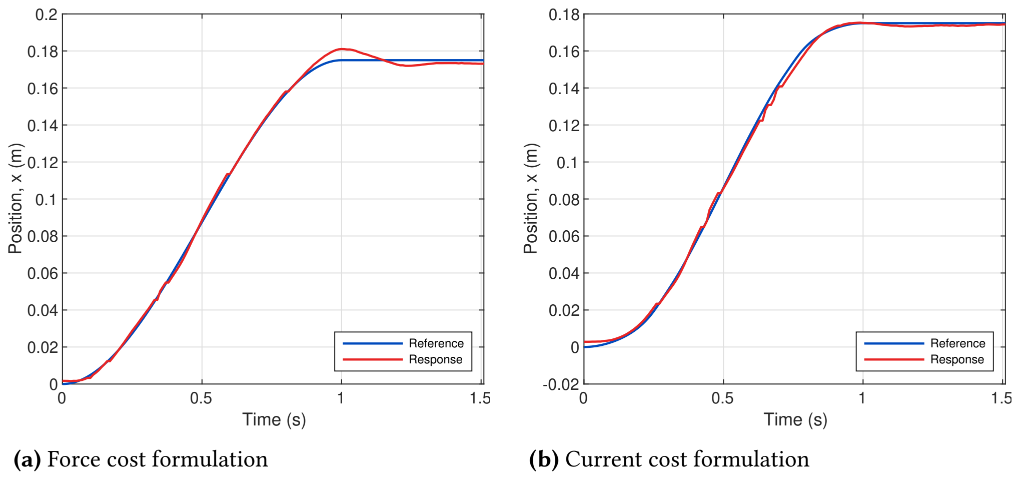


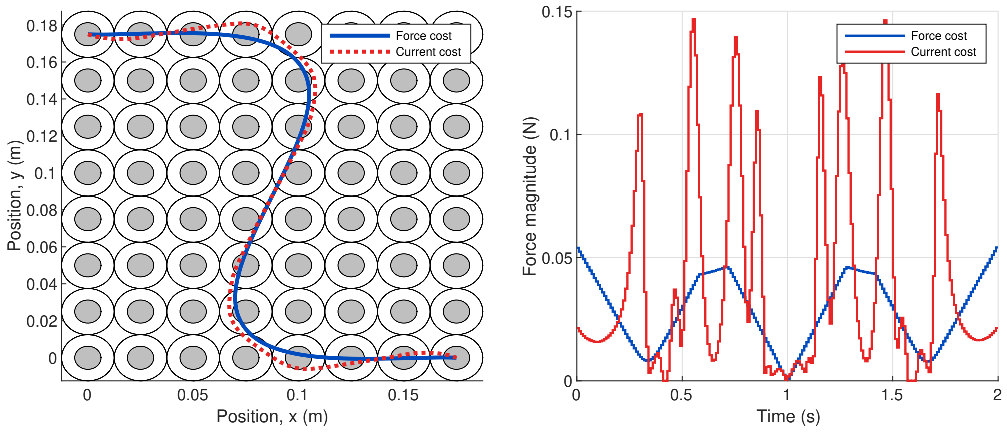
Figure 6.3: Tracking of the planned trajectory presented in Figure 6.1. Results of both cost formulations from Equation (4.4) and Equation (4.6) are shown.

6.2 Complex trajectory

Similarly to the previous section, I evaluated the planned trajectories for both cost formulations. However, the planning goal was not as simple as before. Though I wanted to find a trajectory between some initial and final position while stopping the ball at the end, I did not want it to be a straight path. Thus, I planned in the obstacle environment introduced in Figure 4.3 (b) as it led to a more complex curve, ideal for benchmarking, for example, tracking capabilities. Same as in the RRT* initialization experiment, I chose the starting position to be one of the corner coils, while the coil in the opposing corner is the final position. The time horizon I picked was 1 s. The solving of the mathematical program took 116.82 s for the force cost formulation and 52.36 s for the current cost formulation.

6.2.1 Planned trajectory

The planned path is presented in the Figure 6.4 (a). Notice that the planned paths for the different cost formulations differ this time. Also, this time, I present the magnitude of the planned force in Figure 6.4 (b) as the force generally does not act on the ball only in one direction. Again, in the case of the force cost formulation, the ball is driven mainly by short force impulses. As opposed to the force cost formulation, where the force trajectory is much smoother.

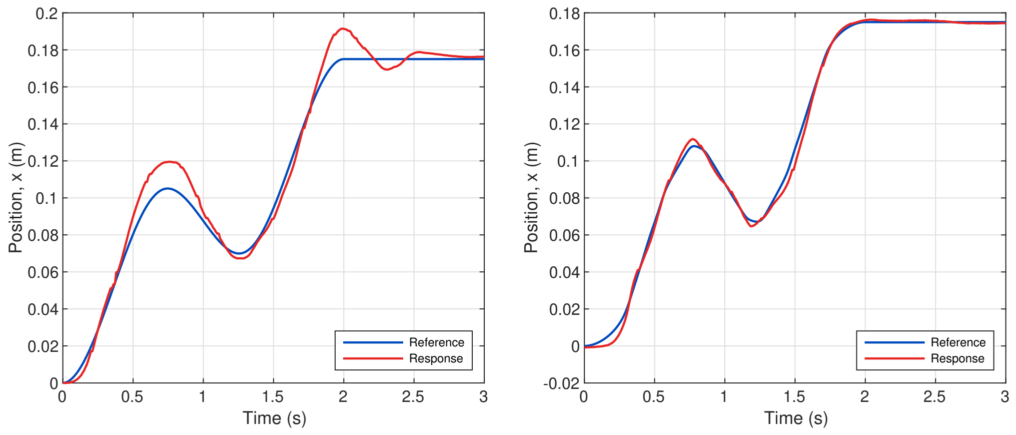


(a) Path **(b)** Force magnitude

Figure 6.4: Complex trajectory generated from obstacle environment in Figure 4.3 (b). Results of both cost formulations from Equation (4.4) and Equation (4.6) are shown. The control horizon was chosen to be 2 s.

6.2.2 Tracking experiment

I again tested the tracking of the planned trajectory on the actual platform. For simplicity, I provide here just the x-position responses in Figure 6.5. The trajectory is satisfyingly tracked by the actual system for the current cost formulation. Similarly, as in the simple trajectory case, the force cost formulation leads to significant overshooting of the trajectory.



(a) Force cost formulation **(b)** Current cost formulation

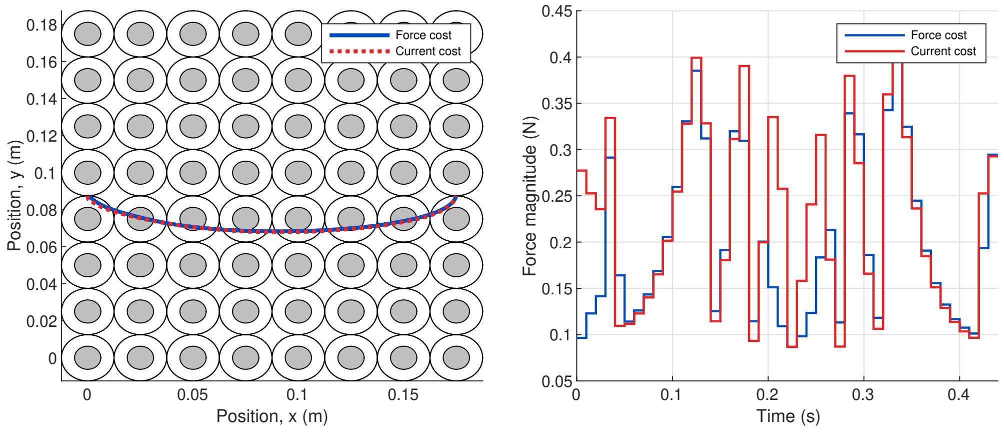
Figure 6.5: Tracking of the planned trajectory presented in Figure 6.4. Results of both cost formulations from Equation (4.4) and Equation (4.6) are shown.

6.3 Minimum-time problem

Next, I experimented with the minimum-time problem. My objective was the same as in Section 6.1. However, this time I desired to bring the ball to its final position in minimal time possible. For this, I utilized the two-phase optimization as presented in Section 4.4. The resulting trajectory horizon was found to be 440 ms, while the first phase with 40 samples took 25.24 s to compute. I computed the second phase for both force and current cost formulations, which took 1.78 s and 1.19 s, respectively.

6.3.1 Planned trajectory

The planned paths are not straight lines (see Figure 6.6 (a)), as opposed to Section 6.1. Also, notice that magnitudes of planned force trajectories presented in Figure 6.6 (b) are similar for both second phase cost formulations. That is a somewhat expected result, as the solution to the linearly constrained problem of the getting system to some final state in minimal time yields bang-bang control.



(a) Path **(b)** Force magnitude

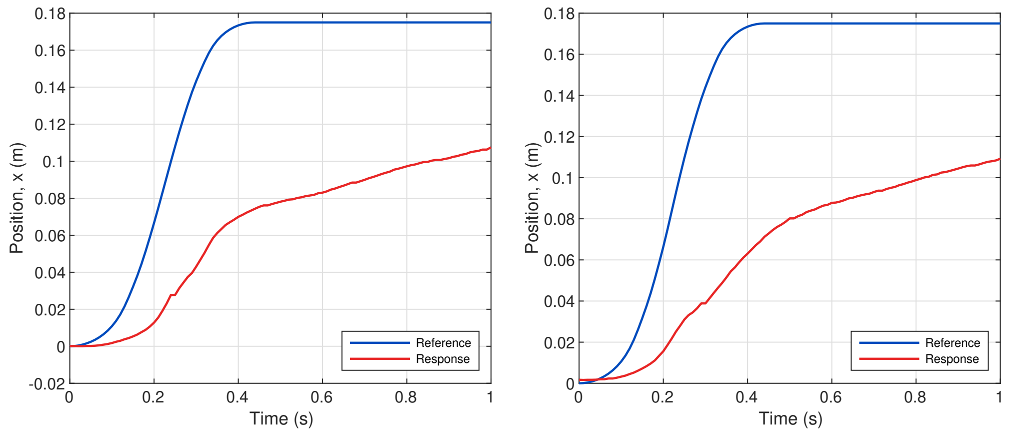
Figure 6.6: Minimum-time trajectory for moving the ball horizontally 87.5 mm. Result of the two-phase optimization scheme from Section 4.4. Both cost forumulation Equation (4.4) and Equation (4.6) are shown for the second phase.

6.3.2 Tracking experiment

I tested the trackability of the minimum-time trajectory on the real platform, for both second phase cost formulations. For the results, see Figure 6.7. It is clear that the trajectories are not tracked in any of the two cases.

There are a few possible explanations for this behaviour. First is the model-reality discrepancy. There are unmodelled phenomena, such as rolling resistance and eddy currents resistance, which act on the ball during its motion. Therefore, the planned trajectory can be faster than what is the platform capable of. Another factor is the force-current allocation. The actual force exerted on the ball is not the same as the desired force. Refer to the force-current allocation Equation (1.17). Other than minimizing the difference between desired and actual force, there is also a term penalizing the currents. Moreover, the problem is solved only for coils in some vicinity of the manipulated ball.

Unfortunately, I could not develop a reliable solution, albeit even a heuristical one. I have tried scaling the time horizon for the second phase and tightening the current limits. These methods seem to help in certain situations but are generally unreliable. Therefore, I leave this as an open problem.



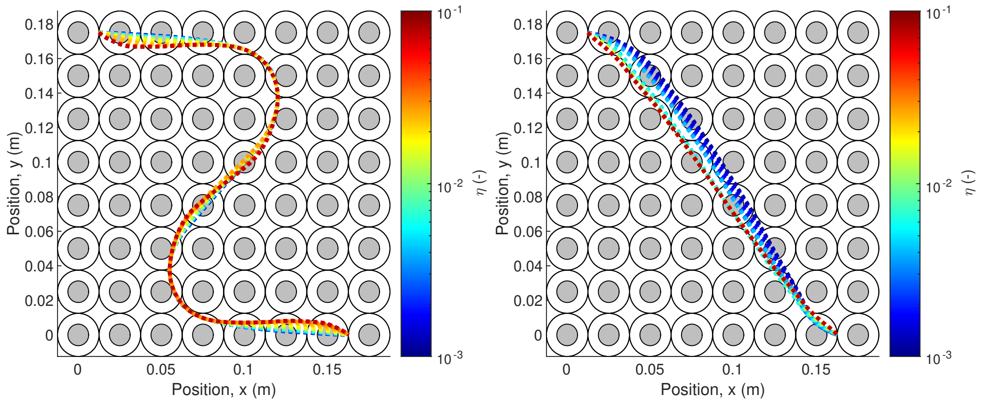
(a) Force cost formulation **(b)** Current cost formulation

Figure 6.7: Tracking of the planned minimum-time trajectory presented in Figure 6.6. Results of both cost formulations from Equation (4.4) and Equation (4.6) are shown.

6.4 Force-position dependency penalization

Another topic that I explored were the effects of penalizing the position dependency of the force. Here I present two paths, one S-shaped (see Figure 6.8 (a)) planned in an obstacle environment, the other one close to straight (see Figure 6.8 (b)) planned without obstacles. I obtained these paths from trajectories planned for a different value of the force dependence penalization factor η (refer to Equation (4.8)). In both cases I used the current cost formulation.

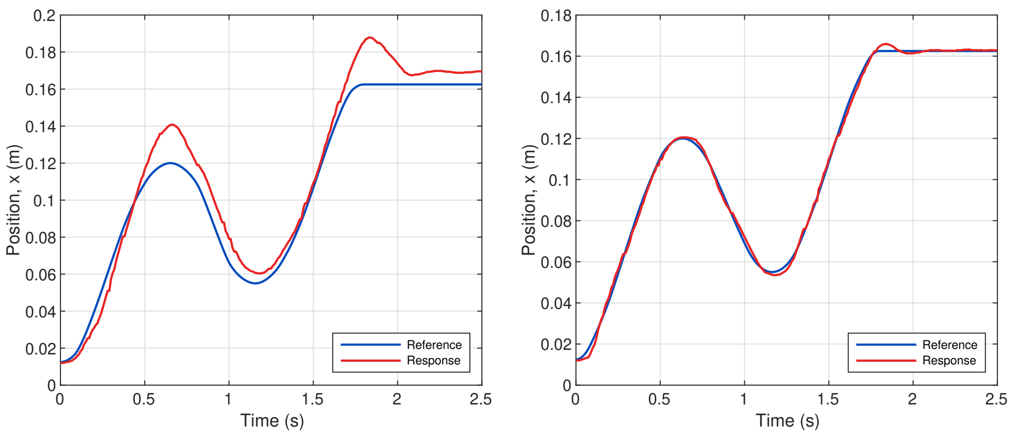
The forms that the resulting paths take based on the penalization factor are not surprising. The larger the value of the factor, the more the ball avoids places with high spatial derivative (see Figure 4.1 (b)).



(a) S-shaped path **(b)** Straight path

Figure 6.8: The planned paths for different values of the force-position dependency penalization factor. The trajectories were calculated for the current cost formulation and control horizon of 1.8 s.

6.4.1 Tracking experiment



(a) $\eta = 0.1$ **(b)** $\eta = 0.005$

Figure 6.9: Tracking the force-position dependency penalized trajectory for different values of the penalization factor.

In theory, the introduced penalization should help with the feedback tracking of the planned trajectories. In reality, that does not appear to be the case. Look at Figure 6.9 (a), which shows the tracking of the S-shaped trajectory for a relatively large value η. The response shows a significant deviation from the reference. Compare it with the response in Figure 6.9 (b) of trajectory planned for a relatively small value of η. This trajectory does not differ much from a trajectory, which is not penalized at all, but it is tracked far better than the heavily penalized one. Such demeanour raises the question of the penalization even making sense. This is further substantiated by the fact that the penalization appears to be computationally taxing. The straight trajectory took 3 s to compute without penalization, while the penalized variant took on average 18 s.

6.5 Collision avoidance

To test the collision avoidance possibilities of my proposed trajectory planning algorithm, I conducted the following experiment. I desired eight balls located on the boundary of the platform to simultaneously switch places with the balls located on the opposite side of the platform. The situation is illustrated in the Figure 6.10.

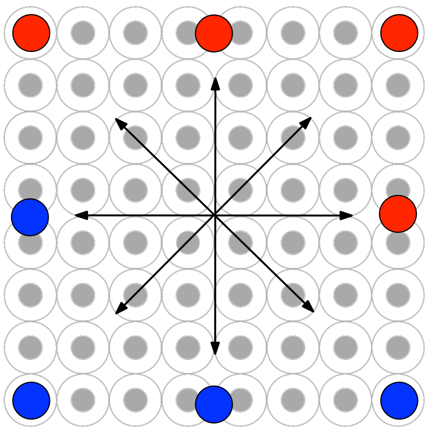
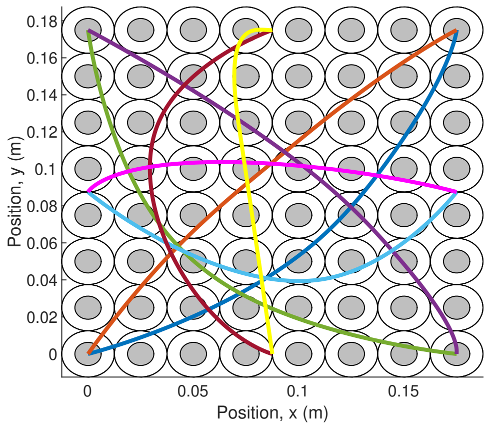


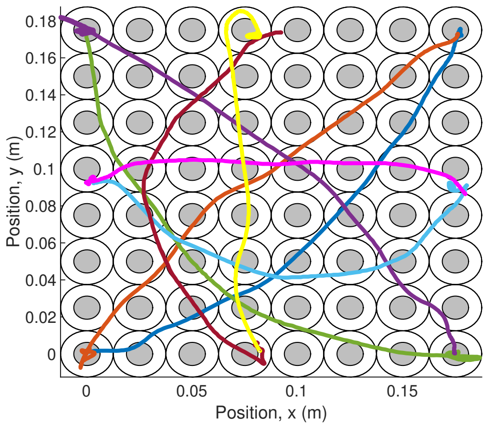
Figure 6.10: Premise of the collision avoidance experiment.

In this experiment, I choose the control horizon to be 2 seconds. Moreover, I used the minimal force cost formulation. I found that the working safety margin r_ϵ for this experiment is $3r_{\text{ball}} = 30\,\text{mm}$.

I repeated the experiment multiple times, and in some runs, one or more of the corner balls did not move or moved late, resulting in a collision. I blame this on the unfortunate placement of the balls directly above the coils. As I showed in Figure 4.1 (b), the spot directly above the coil has the highest spatial derivative of the force. A slight uncertainty in the position measurement can lead to a different force being exerted on the ball than planned. The fact that the distortion given by the camera optics in the corners is the largest does not help either. Nevertheless, in the runs where the balls initially moved, they did not collide. I provide the planned and taken paths from one such run in Figure 6.11. Furthermore, I provide the timelapse photo of the experiment in Figure 6.12 and the video of the experiment in the attachment.



(a) Planned paths



(b) Real paths

Figure 6.11: The planned and real paths taken in the eight ball experiment. Each color represents the path taken by a distinct ball.

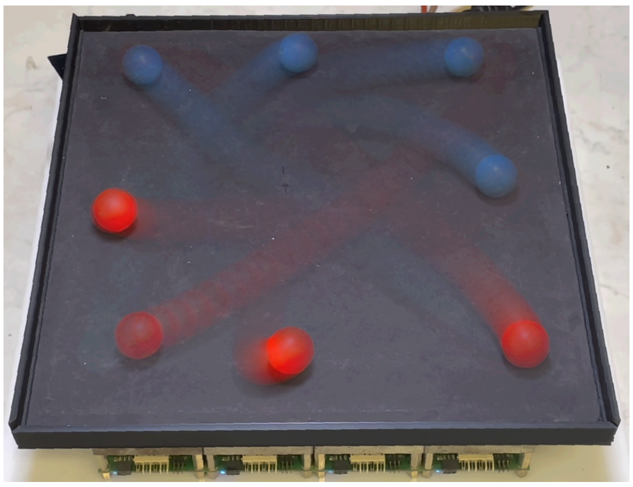


Figure 6.12: Collision avoidance experiment.

6.6 Closing remarks

In this chapter, I showed that if provided with a feasible control horizon, I can plan trajectories for the magnetic manipulation platform MagMan, which are collision-free and obstacle-free. Therefore, there remains the question of finding such a feasible control horizon. One answer could be solving the minimum-time problem since the minimum-time could serve as a lower bound on the feasible control horizon.

I also compared the force and current cost formulations and experimentally evaluated them. Trajectories planned using the force cost formulation, when tracked, show significant overshoot over those planned using the current cost formulation. Hence, I recommend using the current cost formulation when possible. Although, I did not manage to find a case where I could plan the trajectory using one formulation while using the other not.

There is correspondingly the matter of force position-dependency penalization. The experiments showed that high levels of penalization lead to degradation in the trackability of the planned trajectories. I was not able to come up with a reason why that is. Furthermore, even at low levels, the penalization induces significant computational overhead. Thus, I cannot recommend using the penalization.

I am fully aware that the presented algorithms deserve a more thorough computational time and scalability benchmarking. However, as the output of this work should be primarily practical, I prioritized the real platform experiments. If more time were available, I would conduct the aforementioned benchmarks.

7 Conclusions & outlook

In this thesis, I aimed to develop a trajectory planning system for the distributed magnetic manipulation platform MagMan, that could be generalized to other manipulations platforms developed by AA4CC.

In Chapter 2, I described the process of moving MagMan to a new position measuring system I developed during the winter term. I benchmarked the system with the results showing a significant improvement in processing time. The new system runs at the sample rate of 100 Hz, while the old one runs at 50 Hz. Moreover, the old system overruns the target sample time when simultaneously detecting six or more balls. The new one keeps the targeted sample time even when detecting eight balls.

In Chapter 3, I formulated the general trajectory planning problem from the perspective of control theory and also discussed the specifics of planning for MagMan. In Chapter 4, I developed two algorithms to achieve the set thesis goal. The former plans the trajectories for a given control horizon (the arrival time for the manipulated objects). The latter attempts to solve the planning problem in minimum-time. An integral part of these algorithms is a nonlinear mathematical program initialized by an obstacle-free and collision-free path planned by an RRT* algorithm. The minimum-time algorithm employs a two-phase optimization scheme. In the first phase, the minimum-time is found. The correct parametrized trajectory for the minimum-time is then planned in the second phase. I further presented a possible generalization for other distributed manipulation platforms developed by AA4CC.

In chapter Chapter 5, I presented the design of an LQR controller and state estimations for the feedback tracking of the planned trajectories. A simple filtered derivative system provides the state estimation, which is not optimal, as the velocity estimate includes noise and a phase delay.

Both planning algorithms produce the desired trajectories. However, only the first one, when provided with a feasible control horizon, actually produces feedback trackable trajectories. The second one generates trajectories, which are too fast to be tracked by the real platform. This fact arose from experiments conducted on the real platform in Chapter 6. A possible explanation for this behaviour is a model-reality discrepancy, as there are known unmodelled phenomena.

7.1 Future work

I want to prepare a more complex demonstration of the planning capabilities of my developed system. Presumably, the balls could play a song on a xylophone. This demonstration could serve for propagation purposes of the faculty, for example, at the Maker Faire Prague[5] in the summer.

Moreover, I want to solve the minimum-time problem. I would start by developing a more accurate mathematical model of MagMan by identifying and introducing the unmodelled phenomena. A better model goes hand in hand with developing a better state estimation system for the platform. I believe the tracking of the planned trajectories would benefit from better state estimation.

Lastly, further research could be conducted on other distributed manipulation platforms developed by AA4CC. The algorithms I proposed could serve as a basis for exploring trajectory planning possibilities for AcouMan and DEPMan.

5 https://makerfaire.cz/prague/

A Contents of the attachment

`code.zip`	Implementation of the algorithms in MATLAB + CasADi.
`collision_avoidance.mp4`	Video of the collision avoidance experiment.

B Implementation structure

- `code`
  - `algorithms`
    - `planTrajectory.m` Fixed control horizon planning algorithm.
    - `planTrajectoryMinimumTime.m` Minimum-time planning algorithm.
  - `RRTstar`
    - `RRTstar.m` RRT* implementation.
    - `genpath.m` Spline fitting function.
    - `genwaypoints.m` Function returning the waypoints of the path.
    - `nodecost.m` Tree node cost function.
    - `steer.m` Steering function.
    - `isfree.m` Free configuration space indicator function.
  - `classes`
    - `CoilArray.m` MagMan class.
    - `ConfigNode.m` RRT* tree node class.
    - `Obstacle.m` Rectangular obstacle class.
  - `misc`
    - `clrs.mat` Plot colors for different balls.
    - `obstaclesExample.m` Example obstacle environments.

Bibliography

[And+18] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. **CasADi – A software framework for nonlinear optimization and optimal control**. *Mathematical Programming Computation* (In Press, 2018) (see page 25).

[Bet09] John T. Betts. **Practical Methods for Optimal Control and Estimation Using Nonlinear Programming**. 2nd edition. Philadelphia: Society for Industrial and Applied Mathematics, Dec. 17, 2009. 458 pp. ISBN: 978-0-89871-688-7 (see page 23).

[GZH22] Martin Gutner, Jiří Zemánek, and Zdeněk Hurák. **ADMM-based distributed control for distributed manipulation by shaping physical force fields**. *The International Journal of Robotics Research* (2022). In review. ISSN: 0278-3649 (see page 7).

[Hod20] Dominik Hodan. **Reinforcement learning for manipulation of collections of objects using physical force fields**. Bachelor's thesis. Prague: Faculty of Electrical Engineering, Czech Technical University in Prague, June 10, 2020. 62 pp. URL: http://hdl.handle.net/10467/87760 (see page 3).

[Kel17] Matthew Kelly. **An Introduction to Trajectory Optimization: How to Do Your Own Direct Collocation**. *SIAM Review* 59:4 (Jan. 2017). Publisher: Society for Industrial and Applied Mathematics, 849–904. ISSN: 0036-1445. DOI: 10.1137/16M1062569. URL: https://epubs.siam.org/doi/10.1137/16M1062569 (see page 13).

[KF11] Sertac Karaman and Emilio Frazzoli. **Sampling-based algorithms for optimal motion planning**. *The International Journal of Robotics Research* 30:7 (June 2011), 846–894. ISSN: 0278-3649, 1741-3176. DOI: 10.1177/0278364911406761. URL: http://journals.sagepub.com/doi/10.1177/0278364911406761 (see pages 13, 27).

[Kir04] Donald E. Kirk. **Optimal Control Theory: An Introduction**. Courier Corporation, Jan. 1, 2004. 484 pp. ISBN: 978-0-486-43484-1 (see pages 15, 21).

[La 11] Steven M. La Valle. **Motion Planning**. *IEEE Robotics Automation Magazine* 18:2 (June 2011). Conference Name: IEEE Robotics Automation Magazine, 108–118. ISSN: 1558-223X. DOI: 10.1109/MRA.2011.941635 (see page 13).

[LaV06] S. M. LaValle. **Planning Algorithms**. Available at http://planning.cs.uiuc.edu/. Cambridge, U.K.: Cambridge University Press, 2006 (see page 15).

[LaV98] S. LaValle. **Rapidly-exploring random trees : a new tool for path planning**. *The annual research report* (1998). URL: http://msl.cs.illinois.edu/~lavalle/papers/Lav98c.pdf (see page 13).

[LVS12] Frank L. Lewis, Draguna Vrabie, and Vassilis L. Syrmos. **Optimal Control**. John Wiley & Sons, Mar. 20, 2012. 552 pp. ISBN: 978-1-118-12272-3 (see page 15).

[Ma+15] Liang Ma, Jianru Xue, Kuniaki Kawabata, Jihua Zhu, Chao Ma, and Nanning Zheng. **Efficient Sampling-Based Motion Planning for On-Road Autonomous Driving**. *IEEE Transactions on Intelligent Transportation Systems* 16:4 (Aug. 2015). Conference Name: IEEE Transactions on Intelligent Transportation Systems, 1961–1976. ISSN: 1558-0016. DOI: 10.1109/TITS.2015.2389215 (see page 13).

[Mad19] Michael M. Madden. "Challenges Using Linux as a Real-Time Operating System." In: *AIAA Scitech 2019 Forum*. AIAA SciTech Forum. American Institute of Aeronautics and Astronautics, Jan. 6, 2019. DOI: 10.2514/6.2019-0502. URL: https://arc.aiaa.org/doi/10.2514/6.2019-0502 (see page 9).

[Mat+19] Josef Matouš, Adam Kollarčík, Martin Gurtner, Tomáš Michálek, and Zdeněk Hurák. **Optimization-based Feedback Manipulation Through an Array of Ultrasonic Transducers**. *IFAC-PapersOnLine*. 8th IFAC Symposium on Mechatronic Systems MECHATRONICS 2019 52:15 (Jan. 1, 2019), 483–488. ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2019.11.722. URL: https://www.sciencedirect.com/science/article/pii/S2405896319317148 (see page 2).

[Par+16] Diego Pardo, Lukas Möller, Michael Neunert, Alexander W. Winkler, and Jonas Buchli. **Evaluating Direct Transcription and Nonlinear Optimization Methods for Robot Motion Planning**. *IEEE Robotics and Automation Letters* 1:2 (July 2016). Conference Name: IEEE Robotics and Automation Letters, 946–953. ISSN: 2377-3766. DOI: 10.1109/LRA.2016.2527062 (see page 25).

[RMF19] Federico Reghenzani, Giuseppe Massari, and William Fornaciari. **The Real-Time Linux Kernel: A Survey on PREEMPT_RT**. *ACM Computing Surveys* 52:1 (Feb. 21, 2019), 18:1–18:36. ISSN: 0360-0300. DOI: 10.1145/3297714. URL: https://doi.org/10.1145/3297714 (see page 9).

[SL14] Samantha Stoneman and Roberto Lampariello. **Embedding nonlinear optimization in RRT* for optimal kinodynamic planning**. In: *53rd IEEE Conference on Decision and Control*. 53rd IEEE Conference on Decision and Control. ISSN: 0191-2216. Dec. 2014, 3737–3744. DOI: 10.1109/CDC.2014.7039971 (see page 13).

[Ted22] Russ Tedrake. **Underactuated Robotics. Algorithms for Walking, Running, Swimming, Flying, and Manipulation**. 2022. URL: http://underactuated.mit.edu (see page 13).

[WB13] Dustin J. Webb and Jur van den Berg. **Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics**. In: *2013 IEEE International Conference on Robotics and Automation*. 2013 IEEE International Conference on Robotics and Automation. ISSN: 1050-4729. May 2013, 5054–5061. DOI: 10.1109/ICRA.2013.6631299 (see page 13).

[Xie+15] Christopher Xie, Jur van den Berg, Sachin Patil, and Pieter Abbeel. **Toward asymptotically optimal motion planning for kinodynamic systems using a two-point boundary value problem solver**. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015 IEEE International Conference on Robotics and Automation (ICRA). ISSN: 1050-4729. May 2015, 4187–4194. DOI: 10.1109/ICRA.2015.7139776 (see page 14).

[Zem18] Jiří Zemánek. **Distributed manipulation by controlling force fields through arrays of actuators**. PhD thesis. Faculty of Electrical Engineering, Czech Technical University in Prague, 2018. URL: https://support.dce.felk.cvut.cz/mediawiki/images/9/9f/Diz_2018_zemanek_jiri.pdf (see pages 2, 3).

[ZMH18] Jiří Zemánek, Tomáš Michálek, and Zdeněk Hurák. **Phase-shift feedback control for dielectrophoretic micromanipulation**. *Lab on a Chip* 18:12 (June 12, 2018). Publisher: The Royal Society of Chemistry, 1793–1801. ISSN: 1473-0189. DOI: 10.1039/C8LC00113H. URL: https://pubs.rsc.org/en/content/articlelanding/2018/lc/c8lc00113h (see page 2).