CZECH TECHNICAL UNIVERSITY IN PRAGUE
Faculty of Electrical Engineering
Department of Control Engineering

AcouMan — Acoustophoretic Manipulation Platform

Master's thesis

Josef Matouš

Supervisor
Ing. Martin Gurtner

2020

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Matouš Josef**	Personal ID number:	**457195**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Control Engineering**		
Study program:	**Cybernetics and Robotics**		
Branch of study:	**Cybernetics and Robotics**		

II. Master’s thesis details

Master’s thesis title in English:

AcouMan — Acoustophoretic Manipulation Platform

Master’s thesis title in Czech:

AcouMan — akustoforetická manipulační platforma

Guidelines:

Design and assemble a device capable of manipulating several objects floating on a liquid surface through shaping the surrounding acoustic field. The acoustic field will be shaped by a 16-by-16 array of ultrasonic transducers. The shape of the acoustic field will be optimized in feedback so that the manipulated objects move towards specified positions.
1. Design, manufacture, and assemble the mechanical and electrical parts of the device.
2. Formulate the optimization problem of forming such an acoustic field that the manipulated objects move towards the desired positions.
3. Multi-object manipulation - manipulate independently several spherical objects.
4. Multi-object manipulation - manipulate independently several planar objects.
5. Assemble the manipulated objects to some predefined formations.

Bibliography / sources:

[1] J. Matouš, A. Kollarčík, M. Gurtner, T. Michálek, and Z. Hurák, “Optimization-based Feedback Manipulation Through an Array of Ultrasonic Transducers,” IFAC-PapersOnLine, vol. 52, no. 15, pp. 483–488, 2019.
[2] A. Marzo, R. McGeehan, J. McIntosh, S. A. Seah, and S. Subramanian, “Ghost touch: Turning surfaces into interactive tangible canvases with focused ultrasound,” in Proceedings of the 2015 International Conference on Interactive Tabletops & Surfaces, 2015, pp. 137–140.
[3] H. Ebel and P. Eberhard, “Optimization-Driven Control and Organization of a Robot Swarm for Cooperative Transportation,” IFAC-PapersOnLine, vol. 52, no. 15, pp. 115–120, 2019.

Name and workplace of master’s thesis supervisor:

Ing. Martin Gurtner, Department of Control Engineering, FEE

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

doc. Ing. Zdeněk Hurák, Ph.D., Department of Control Engineering, FEE

Date of master’s thesis assignment: **31.01.2020** Deadline for master's thesis submission: **22.05.2020**

Assignment valid until: **30.09.2021**

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

Declaration

I hereby declare that this thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

Josef Matouš
May 2020

Acknowledgments

First and foremost, I would like to thank my supervisor, *Martin Gurtner*, for his advice on algorithms and proofreading. Without him, this thesis would not be possible (or, at least, it would have been of much worse quality). I would also like to thank *Zdeněk Hurák*, head of the AA4CC research group, for inviting me to the group and for making the creation of the experimental platform possible.

I also cannot forget to mention and thank my friends and colleagues from the lab. Namely, I would like to thank *Loi Do* for making this beautiful template, and *Adam Kollarčík* for building the first version of the experimental platform with me. I would also like to thank this duo and some other friends for fixing the *table football* (or *foosball*) game, which has proven to be an excellent source of procrastination.

Last but not least, I would like to thank my family for their support. They have been really great, especially for the past two months, during which I have been working on this thesis at home due to the lockdown.

Abstract

This thesis describes the design and development of an acoustophoretic manipulation platform *AcouMan*. The platform uses a matrix of ultrasonic transducers to steer millimeter-scale objects floating on a water surface. The transducers driven by phase-shifted square-wave signals allow shaping the acoustic field. The phase-shifts are periodically determined through real-time numerical optimization so that a desired shape of the acoustic field is achieved. An implementation of a tailored solver was necessary to meet the real-time requirements. The thesis explores the manipulation of spherical and planar floating objects. For both, a mathematical model is designed that describes the motion of the object due to the force effect of the acoustic field. The control system for spheres is based on a linear-quadratic regulator (LQR); the control system for planar objects is based on model predictive control (MPC) and particle swarm optimization (PSO). Besides positioning single and multiple objects, the performance of the control methods is also demonstrated through assembling several planar objects into predefined tight formations.

Keywords: ultrasound, non-contact manipulation, LQR, MPC, real-time optimization

Abstrakt

Tato práce popisuje návrh a vývoj akustoforetické manipulační platformy *AcouMan*. Tato platforma využívá pole ultrazvukových měničů pro řízení milimetrových objektů plovoucích na vodní hladině. Ultrazvukové měniče jsou řízeny obdélníkovými signály s různými fázovými posuvy, což umožňuje tvarovat akustické pole. Tyto fázové posuvy se nastavují periodicky podle výsledků numerické optimalizace tak, aby vzniklo akustické pole s požadovaným tvarem. Pro funkčnost v reálném čase bylo nutné implementovat solver šitý na míru. Tato práce zkoumá bezkontaktní manipulaci s kulovými a planárními objekty. Pro oba případy je navržen matematický model objektu, který popisuje jeho pohyb v důsledku silového působení akustického pole. Systém pro řízení kulových objektů je založen na lineárně-kvadratickém regulátoru (LQR), zatímco systém pro planární objekty je založen na prediktivním řízení (MPC) a *tzv. particle swarm* optimalizaci (PSO). Výkon těchto řídicích metod je kromě pozicování jednoho či více objektů demonstrován také skládáním planárních objektů do předepsaných těsných formací.

Klíčová slova: ultrazvuk, bezkontaktní manipulace, LQR, MPC, optimalizace v reálném čase

Contents

1 | Introduction

In this thesis, I present *AcouMan*, a novel platform for contactless acoustophoretic manipulation of multiple objects. The platform can manipulate spherical objects (balls) and planar objects (thin, 3D-printed shapes) that float in a container filled with water. The objects are affected by forces that arise from the shape of the surrounding acoustic field. The field is shaped by a 16-by-16 array of ultrasonic transducers. The position of the objects is measured, allowing for precise feedback manipulation.

1.1 Motivation

AcouMan is the second iteration of an acoustophoretic platform. In 2018, I wrote and successfully defended a Bachelor's thesis about the first version of the platform [1]. A year later, I have co-authored and presented a paper about this version at the *8th IFAC Symposium on Mechatronic Systems* [2]. Albeit functional, the first version had some drawbacks. The biggest issue was its small manipulation area and the fact that it was able to control only one object at a time. Also, the developed control system could only manipulate spherical objects. Thus, the first version served as a proof of concept.

This thesis aims to fully explore the possibilities of this method of ultrasonic manipulation. The newly built platform should have a larger manipulation area, should be able to steer multiple objects at once, and it should be able to manipulate floating planar objects, not just spheres.

1.2 State of the art

There are numerous applications that use an array of ultrasonic transducers. The most prominent example is the acoustic levitation. A comprehensive overview of progress in the development of acoustic levitation is provided in a paper by *Andrade et al.* [3]. In short, there are several methods that utilize both standing [4, 5, 6], and traveling [7] waves and can manipulate objects of various sizes, ranging from microscopic [5, 6] to millimeter-scale [4, 7]. The platforms mentioned above, as well as the majority of others, use open-loop control — they do not need to measure the position of manipulated objects, because they are naturally stable. An example of closed-loop control is the so-called *Ultra-Tangibles* platform developed by *Marshall et al.* [8]. This platform allows for interactive manipulation of multiple objects. However, compared to *AcouMan*, it is far more complicated — each ultrasonic transducer has its own microcontroller.

Another example of a platform that uses an array of ultrasonic transducers is the so-called *Ghost Touch* developed by *Marzo et al.* [9]. This platform is very similar to *AcouMan* in the way it shapes the acoustic field. The transducers are driven by square-wave voltage signals with appropriate phase shifts to generate acoustic pressure focal points in desired positions. Controlled positioning of these focal points enables drawing in the sand, inducing flow in a fluid, and to create, move with, or pop bubbles in a soap solution. It is precisely this effect of focal points on various substances that the *AcouMan* platform attempts to harness for the purpose of contactless manipulation.

1.3 Thesis outline

The thesis is structured in accordance with the assignment. Each bullet point from the assignment corresponds to one chapter. In Chapter 2, I present the mechanical and electronic components of the platform. In Chapter 3, I define the optimization problem for shaping the acoustic field and describe an algorithm for solving it. In Chapter 4, I design a method for independent manipulation of multiple spherical objects. This procedure involves modeling the spherical object, designing a control system, identifying the parameters of the model, and tuning the control system parameters. In Chapter 5, I repeat this design procedure for planar objects. In Chapter 6, I present an algorithm for assembling these planar objects into predefined formations. Finally, I conclude this thesis in Chapter 7.

2 | Hardware

In this chapter, I briefly describe the mechanical and electronic components of the platform. As mentioned in the introduction, *AcouMan* is the second iteration of an acoustophoretic platform; the original version was built as a part of my Bachelor's thesis. A photograph of both platforms is shown in Figure 2.1a.

2.1 Mechanical design

Similarly to its predecessor, the new platform is composed of three laser-engraved plexiglass panels. The bottom panel serves as a mount for the Raspberry Pi camera module and the LED lights. The bottom panel of the new version is spray-painted with matte black paint to reduce reflections, and the LED strips illuminating the manipulation area are covered with translucent plastic, which acts as a diffuser.

The middle panel is the manipulation area. In the previous version, I used a Petri dish as a container for the floating object, as shown in Figure 2.1c. Because the effective manipulation area of the new version is larger, I can no longer use the Petri dish. To solve this problem, I have built a pool with 3D-printed walls. Detail of the pool is shown in Figure 2.1b.

The top panel holds the array of ultrasonic transducers. The previous version utilized a laser-cut box as housing, while the new version has the printed circuit board (PCB) with transducers attached directly to the plexiglass panel. The array of transducers is also larger, increasing from 64 to 256 actuators.

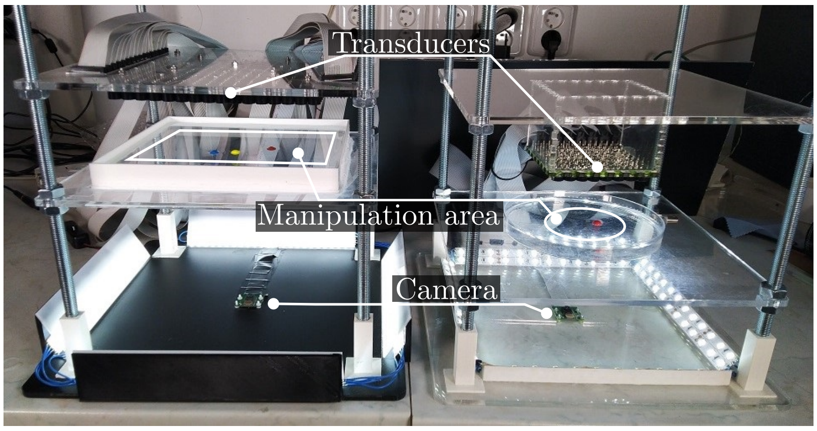
2.2 Electronics

The electronic components of the platform consist of 256 *Murata MA40S4S* ultrasonic transducers, four signal generators to drive them, and a *Raspberry Pi* computer. A diagram with connections between these components is shown in Figure 2.2.

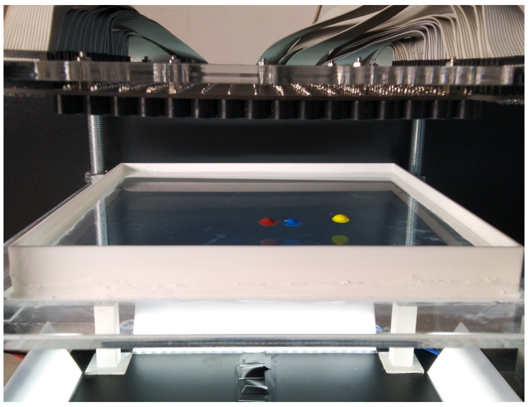
The *Raspberry Pi* is the central 'brain' of the platform. It processes the images from the camera module[1], runs the control algorithm, and communicates with the signal generators. The control algorithm is implemented in Simulink, allowing for rapid prototyping. The Simulink models are available on the AcouMan GitLab repository (see Appendix A.1).

The signal generator is based on the *Terasic DE0-Nano* field-programmable gate array

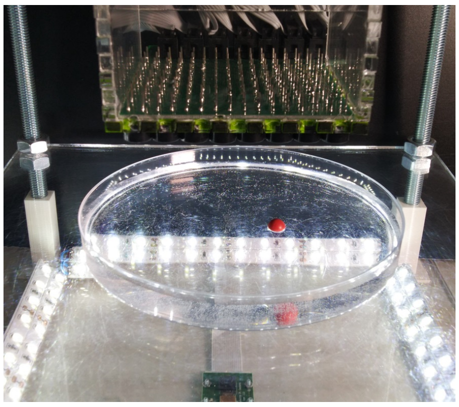
[1]The computer vision system is available at `https://github.com/aa4cc/raspi-ballpos`.



(a) A side-by-side comparison between the old (right) and new (left) platform



(b) Manipulation area of the new platform



(c) Manipulation area of the old platform

Figure 2.1: Photographs of the old and the new acoustophoretic platform

(FPGA) board, and a custom-made 'shield' add-on. The FPGA board generates 64 phase-shifted square-wave signals having 3.3 volts peak-to-peak (V_{pp}), which are then amplified to 16 V_{pp} by the shield. The phase-shifts and duty cycles of individual signals are set via USB from the *Raspberry Pi*. The duty cycles are only used to turn individual channels on (50 % duty) or off (0 % duty), not to regulate the power of the transducers. The resolution of the phase-shifts is one degree ($\pi/180$ radians).

Since the signals that drive the transducers are generated on four separate FPGA boards, the boards need to be synchronized. The synchronization method as well as the FPGA program was designed by an undergraduate student Petr Brož[2]. His solution employs a master-slave architecture. One FPGA board, configured as the master, transmits high-frequency pulses, which mark the rising edge of a zero phase-shift signal. Other boards receive these pulses

[2]The code is available at `https://github.com/aa4cc/fpga-generator`.

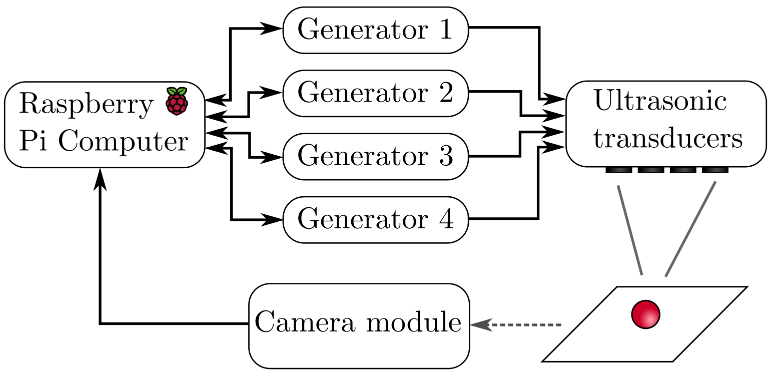


Figure 2.2: Principal signal diagram of the platform

and adjust accordingly. When the FPGA boards receive the new set of phase-shifts and duty cycles, they do not reconfigure their outputs immediatelly. Instead, they wait for a trigger pulse transmitted from one of the general purpose input-output (GPIO) ports of the *Raspberry Pi*. Therefore, the driving signals change simultaneously. In addition, the FPGA generators also transmit an acknowledgement whether they received the new set of phase-shifts or duty cycles correctly.

My contribution consists mainly of the communication script that runs on the *Raspberry Pi*. The script has a defined number of communication attempts. It transmits the given set of phase-shifts and duty cycles to every generator until it receives an acknowledgement or runs out of attempts. Upon transmitting the last message, the script sends the trigger pulse to the generators.

3 | Shaping the acoustic field

In this chapter, I present the mathematical model of the acoustic field, explain why it is necessary to use numerical optimization, formulate the optimization problems and algorithms for solving them.

3.1 Model of the acoustic field

In the previous chapter, I stated that the ultrasonic transducers are driven by phase-shifted signals. In this section, I explain how the different phase-shifts affect the shape of the acoustic field.

The mathematical model, which describes the field, uses acoustic pressure in complex phasor notation. When calculating the total acoustic pressure, P, at a given point, (x, y, z), with a given vector of phase-shifts, $\mathbf{\Phi} = [\varphi_1, \ldots, \varphi_N]$, I assume the principle of superposition:

$$P(x, y, z, \mathbf{\Phi}) = \sum_{i=1}^{N} P^{(i)}(x, y, z, \varphi_i)\,, \tag{3.1}$$

where $P^{(i)}(x, y, z, \varphi_i)$ is the contribution from the i^{th} transducer. This contribution is expressed as a product of position- and phase-shift-dependent parts:

$$P^{(i)}(x, y, z, \varphi_i) = M^{(i)}(x, y, z)\, \mathrm{e}^{\mathrm{j}\varphi_i}\,. \tag{3.2}$$

Model of the position-dependent part used in this thesis was proposed in [9], and is defined as:

$$M^{(i)}(x, y, z) = P_0\, f_{dir}(x, y, z)\, \frac{1}{d}\, \mathrm{e}^{\mathrm{j}kd}\,, \tag{3.3}$$

where P_0 is the nominal pressure at the distance of 1 m, f_{dir} is the so-called directivity function, d is the distance between the point and the transducer, and k is the wavenumber.

For the directivity function, I employ a far-field model of a vibrating piston source [10, p. 179-182] [1]:

$$f_{dir} = \frac{2\,\mathrm{J}_1(k\,r\,\sin\theta)}{k\,r\,\sin\theta}\,, \tag{3.4}$$

where r is the radius of the transducer, J_1 is the first-order Bessel function of the first kind, and θ is the angle between the vertical axis of the transducer and the transducer-point

[1] Due to the high computational cost of evaluating the Bessel function, the directivity in the actual algorithm is pre-computed at several given points and then approximated by linear interpolation.

connector. Assuming that the coordinate system is centered at the transducer, the sine of θ is equal to

$$\sin\theta = \frac{\sqrt{x^2+y^2}}{d} = \sqrt{\frac{x^2+y^2}{x^2+y^2+z^2}}\,. \tag{3.5}$$

3.2 Formulating the optimization problem

While it is relatively straightforward to calculate the shape of the acoustic field generated by a given set of phase-shifts, the inverse problem (*i.e.*, finding phase-shifts that generate a prescribed acoustic field) is far from being trivial, and to my best knowledge, it does not have a closed-form solution.

Objects exposed to the acoustic field tend to move from high-pressure to low-pressure areas. For the purpose of manipulation, it is desirable to create areas with different pressure, which push the objects. To shape the acoustic field, it is necessary to use numerical optimization with a carefully selected criterion. I have explored three possible criteria for this optimization problem.

3.2.1 Amplitude fitting

The phase of the acoustic pressure has no observable effect on objects placed in the field. It is, therefore, sufficient to consider only the amplitude.

The simplest criterion, which I already used in my Bachelor's thesis, is defined as follows: Let there be n points, $(x_1, y_1, z_1), \dots, (x_n, y_n, z_n)$, with desired amplitudes, $A_1, \dots, A_n$. The optimal phase-shifts are obtained by solving a nonlinear least squares problem:

$$\mathbf{\Phi}^* = \operatorname*{arg\,min}_{\mathbf{\Phi}\in\mathbb{R}^N} \sum_{i=1}^{n} \left(|P(x_i, y_i, z_i, \mathbf{\Phi})|^2 - A_i^2 \right)^2 . \tag{3.6}$$

The amplitude of the total acoustic pressure is squared, because it is actually easier to calculate the square than the amplitude itself. An efficient formula for calculating the amplitude is presented in Appendix B.1.

3.2.2 Local maximum constraint

Having a prescribed amplitude at a given point is often not sufficient. To have the desired effect on the manipulated object, the specified point must also be a local maximum.

The correct way of ensuring that the specified point is a local maximum would be to minimize eigenvalues of the spatial Hessian matrix (which is equivalent to minimizing the sum of the second spatial derivatives of the acoustic pressure) and simultaneously have the spatial gradient equal to zero. Unfortunately, calculating the second spatial derivatives is computationally prohibitive.

A computationally manageable criterion uses only the first-order condition for a local extreme:

$$\begin{aligned}\mathbf{\Phi}^* = \underset{\mathbf{\Phi}\in\mathbb{R}^N}{\arg\min} \sum_{i=1}^{n} \Bigg[& w_p \left(|P(x_i, y_i, z_i, \mathbf{\Phi})|^2 - A_i^2\right)^2 + \\ & \left(\frac{\partial\,|P(x_i, y_i, z_i, \mathbf{\Phi})|^2}{\partial x}\right)^2 + \left(\frac{\partial\,|P(x_i, y_i, z_i, \mathbf{\Phi})|^2}{\partial y}\right)^2 \Bigg],\end{aligned} \tag{3.7}$$

where w_p is a weight that allows to adjust the trade-off between having the required amplitude and having zero gradient. Since the movement of objects is restricted to the xy plane, minimizing the spatial derivative with respect to z is not necessary. An efficient formula for calculating the spatial derivatives is presented in Appendix B.2.

Compared to criterion in (3.6), this criterion attempts to achieve a local maximum by minimizing the square of the spatial derivatives. This approach does not guarantee a local maximum (in fact, it can also result in a local minimum or a saddle point). However, thanks to the nature of the underlying mathematical model, minimizing this criterion results in a local maximum if the specified amplitude is high enough.

3.2.3 Gradient fitting

A different approach to creating high- and low-pressure areas is based on the assumption that the pressure difference between these zones is proportional to the steepness of the acoustic pressure gradient. The criterion is defined as follows: Let there be n points, $(x_1, y_1, z_1), \ldots, (x_n, y_n, z_n)$, with desired value of spatial derivatives, $(P_{x_1}, P_{y_1}), \ldots, (P_{x_n}, P_{y_n})$. The optimal phase-shifts are obtained by solving a nonlinear least squares problem:

$$\mathbf{\Phi}^* = \underset{\mathbf{\Phi}\in\mathbb{R}^N}{\arg\min} \sum_{i=1}^{n} \left[\left(\frac{\partial\,|P(x_i, y_i, z_i, \mathbf{\Phi})|^2}{\partial x} - P_{x_i}\right)^2 + \left(\frac{\partial\,|P(x_i, y_i, z_i, \mathbf{\Phi})|^2}{\partial y} - P_{y_i}\right)^2 \right]. \tag{3.8}$$

Unlike the two previous criteria, the amplitude of the acoustic pressure is not considered here. This can lead to potential issues if the resulting amplitude is too high, as high-amplitude acoustic pressure can break the surface tension of water and create bubbles.

3.3 Optimization algorithm

All three criteria listed above lead to nonlinear least squares problems. There are various methods for solving this class of problems[11]; one of them is the Levenberg-Marquardt (LM) algorithm.

The LM algorithm is used to solve a nonlinear least squares problem, which is generally defined as:

$$\mathbf{p}^* = \arg\min_{\mathbf{p}} \|f(\mathbf{p}) - \mathbf{y}\|_2^2 \,, \tag{3.9}$$

where $\mathbf{p}$ is the vector of free variables (in the acoustic field shaping problem, $\mathbf{p}$ would be the vector of phase-shifts), f is the nonlinear function, $\mathbf{y}$ is the vector of fitted values, and $\|.\|_2$ represents the Euclidian norm of a vector.

The LM algorithm is iterative; in every iteration, the estimate $\mathbf{p}$ is updated by adding a vector $\boldsymbol{\delta}_{\mathbf{p}}$. The value of $\boldsymbol{\delta}_{\mathbf{p}}$ is determined by solving a set of linear equations:

$$\left(\mathbf{J}^{\mathrm{T}}\,\mathbf{J} + \mu\,\mathbf{I}_N\right)\,\boldsymbol{\delta}_{\mathbf{p}} = -\mathbf{J}^{\mathrm{T}}\,\left(f(\mathbf{p}) - \mathbf{y}\right) \,, \tag{3.10}$$

where $\mathbf{J}$ is the Jacobian matrix of f, $\mathbf{I}_N$ is an $N \times N$ identity matrix, and μ is a damping coefficient. The value of μ changes depending on the performance of the algorithm — it increases when the algorithm fails to converge and decreases otherwise.

The set of equations in (3.10) is in a form that can be efficiently solved using a 'kernel trick,' described in [12, p. 332]. The kernel trick lowers the complexity of solving (3.10) by transforming the original system of N equations and N unknowns to a smaller system of n equations with n unknowns. Formally, the solution can be expressed as:

$$\boldsymbol{\delta}_{\mathbf{p}} = -\left(\mathbf{J}^{\mathrm{T}}\,\mathbf{J} + \mu\,\mathbf{I}_N\right)^{-1}\,\mathbf{J}^{\mathrm{T}}\,\left(f(\mathbf{p}) - \mathbf{y}\right) \,. \tag{3.11}$$

The following identity

$$\left(\mathbf{J}^{\mathrm{T}}\,\mathbf{J} + \mu\,\mathbf{I}_N\right)^{-1}\,\mathbf{J}^{\mathrm{T}} = \mathbf{J}^{\mathrm{T}}\,\left(\mathbf{J}\,\mathbf{J}^{\mathrm{T}} + \mu\,\mathbf{I}_n\right)^{-1} \,, \tag{3.12}$$

holds true for any matrix $\mathbf{J} \in \mathbb{R}^{n \times N}$ and scalar $\mu > 0$ (proof in [12, p. 333]). This identity can be substituted to (3.11) to obtain

$$\boldsymbol{\delta}_{\mathbf{p}} = -\mathbf{J}^{\mathrm{T}}\,\left(\mathbf{J}\,\mathbf{J}^{\mathrm{T}} + \mu\,\mathbf{I}_n\right)^{-1}\,\left(f(\mathbf{p}) - \mathbf{y}\right) \,, \tag{3.13}$$

where we need to invert a smaller n-by-n matrix instead of the original N-by-N matrix in (3.11). Furthmore, we can make use of the special structure of the underlying linear system of equations in (3.13) and solve the system efficiently by QR factorizaion. First, let us define a stacked matrix

$$\bar{\mathbf{J}} = \begin{bmatrix} \mathbf{J}^{\mathrm{T}} \\ \sqrt{\mu}\,\mathbf{I}_n \end{bmatrix} \,. \tag{3.14}$$

The following identity holds true for the stacked matrix

$$\bar{\mathbf{J}}^{\mathrm{T}}\,\bar{\mathbf{J}} = \mathbf{J}\,\mathbf{J}^{\mathrm{T}} + \mu\,\mathbf{I}_n \,. \tag{3.15}$$

Now, let us decompose the matrix $\bar{\mathbf{J}}$ into QR factors. The matrix $\bar{\mathbf{J}}$ has dimensions $(N+n) \times n$.

Therefore, its $\mathbf{R}$ factor is composed of an $n \times n$ upper triangular matrix $\mathbf{R}_1$, and an $N \times n$ matrix of zeros. Thus, the term $\bar{\mathbf{J}}^{\mathrm{T}}\,\bar{\mathbf{J}}$ can be expressed as

$$\bar{\mathbf{J}}^{\mathrm{T}}\,\bar{\mathbf{J}} = \left(\mathbf{Q}\begin{bmatrix}\mathbf{R}_1\\ 0\end{bmatrix}\right)^{\mathrm{T}}\mathbf{Q}\begin{bmatrix}\mathbf{R}_1\\ 0\end{bmatrix} = \mathbf{R}_1^{\mathrm{T}}\,\mathbf{R}_1\,. \tag{3.16}$$

Combining (3.15) and (3.16) yields the following identity

$$\mathbf{J}\,\mathbf{J}^{\mathrm{T}} + \mu\,\mathbf{I}_n = \mathbf{R}_1^{\mathrm{T}}\,\mathbf{R}_1\,. \tag{3.17}$$

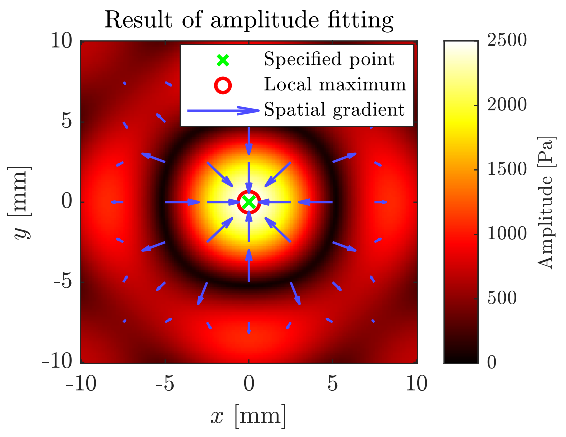
Substituting this identity into (3.13) yields

$$\boldsymbol{\delta}_{\mathbf{p}} = -\mathbf{J}^{\mathrm{T}}\,\mathbf{R}_1^{-1}\,\mathbf{R}_1^{-\mathrm{T}}\,\left(f(\mathbf{p}) - \mathbf{y}\right)\,. \tag{3.18}$$

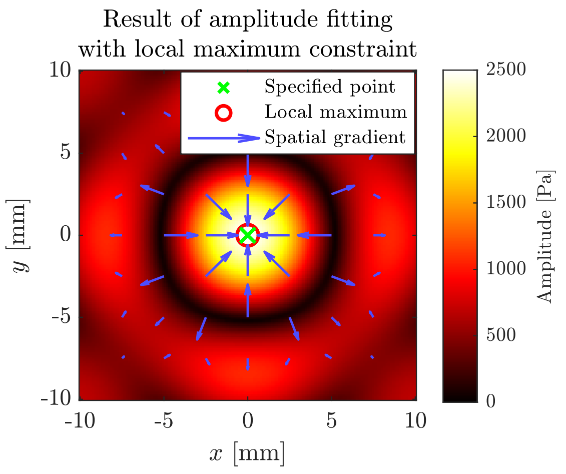
Solving for $\boldsymbol{\delta}_{\mathbf{p}}$ using the 'kernel trick' has an algorhitmic complexity of Nn^2 [12, p. 333], while solving the set of N equations defined in (3.10) directly has a complexity of N^3. Since N (the number of transducers) is 256 and n (the number of fitted values) is at most 4, the kernel trick brings a major improvement over direct calculation.

3.4 The resulting shape of the acoustic field

A comparison between acoustic fields generated by amplitude fitting (*i.e.*, by minimizing criterion (3.6)), and by amplitude fitting with local maximum constraints (*i.e.*, by minimizing criterion (3.7) with $w_p = 1$) is shown in Figures 3.1 and 3.2. The blue arrows show the direction and magnitude of the spatial derivatives of the acoustic pressure, the green cross marks the desired position of the high-pressure point, and the red circle marks the actual position of the local maximum of acoustic pressure.

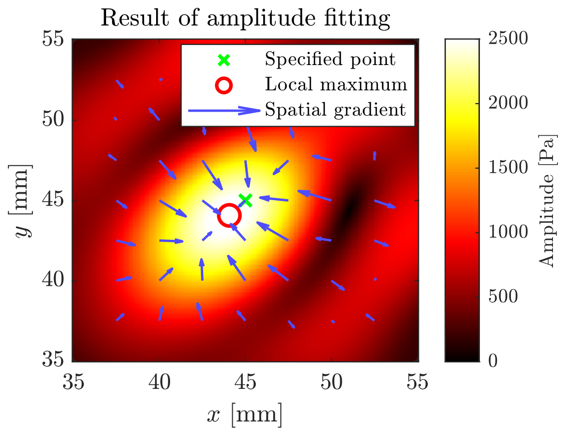


(a) The acoustic field shaped by minimizing criterion (3.6)

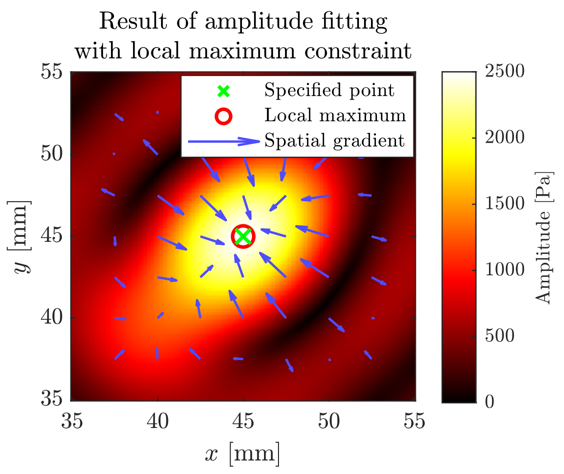


(b) The acoustic field shaped by minimizing criterion (3.7)

Figure 3.1: Slices of the acoustic pressure field at $z = 75\,\mathrm{mm}$. The required amplitude is $2500\,\mathrm{Pa}$ at $(x, y, z) = (0, 0, 75)$.



(a) The acoustic field shaped by minimizing criterion (3.6)



(b) The acoustic field shaped by minimizing criterion (3.7)

Figure 3.2: Slices of the acoustic pressure field at $z = 75\,\text{mm}$. The required amplitude is 2500 Pa at $(x, y, z) = (45, 45, 75)$.

If the specified high-pressure point has its x- and y-coordinates equal to zero, the resulting field has an identical shape, and the specified point is the local maximum in both cases, as shown in Figure 3.1. However, if the specified point is not at the center, the amplitude fitting method does not generate a field, where the specified point is the local maximum. This can be seen in Figure 3.2a, where the distance between the specified point and the local maximum is approximately 2 millimeters. Figure 3.2b shows that this issue is solved by adding the spatial derivatives of the amplitude of the acoustic field to the cost function (see (3.7)).

The acoustic field generated by gradient fitting is shown in Figure 3.3. While this method generates a low- and high-pressure area near the specified point, it can also create a second local maximum. This second maximum may have an unpredictable effect on the manipulated object.

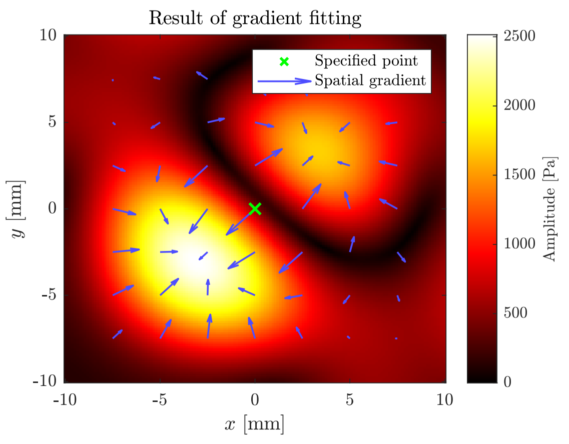


Figure 3.3: The acoustic field shaped by minimizing criterion (3.8). The required spatial derivatives with respect to x and y are both $-7 \times 10^8\,\text{Pa}^2\,\text{m}^{-1}$ at $(x, y, z) = (0, 0, 75)$.

Another important aspect that needs to be addressed is the physical limitations of the acoustic field. Minimizing the ‘amplitude fitting’ criteria (3.6) and (3.7) can generate amplitudes of acoustic pressure as high as 3500 pascals. As mentioned above, such high amplitudes tend to break the surface tension of water and create bubbles, which cause random disturbances to the manipulated objects. To prevent this, the generated amplitudes must be limited to 2500 pascals. We can enforce this limit by limiting the required amplitude since the amplitude of the resulting local maximum is close to the required value, as shown in Figures 3.1 and 3.2.

The spatial gradients of the acoustic pressure modulus are limited as well. Through trial and error, I have concluded that the resulting spatial gradient is limited to approximately $1 \times 10^9\,\mathrm{Pa}^2\,\mathrm{m}^{-1}$ in all directions, *i.e.,* the following inequality holds for the spatial derivatives:

$$\left(\frac{\partial\,|P(x_i, y_i, z_i, \mathbf{\Phi})|^2}{\partial x}\right)^2 + \left(\frac{\partial\,|P(x_i, y_i, z_i, \mathbf{\Phi})|^2}{\partial y}\right)^2 \leq \left(10^9\right)^2 . \tag{3.19}$$

3.5 Algorithm benchmarking

To assess the performance of the proposed algorithms, I have created a benchmark program. The program measures the execution time of individual algorithms on problems with one up to four fitted points, 256 transducers, and various values for the required amplitudes or spatial gradients.

I have tested the LM algorithm with the three proposed criteria, as well as the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm used in my Bachelor’s thesis [1]. The results are shown in Figure 3.4. As you can see, the LM algorithm brings a significant improvement in execution time over BFGS. In the following chapters, I will, therefore, use the LM algorithm with both amplitude and gradient fitting.

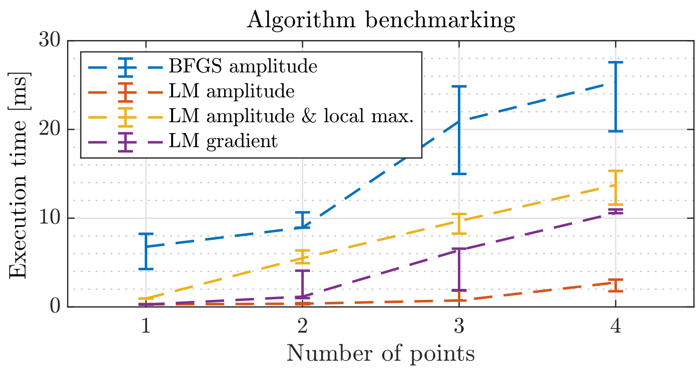


Figure 3.4: Results of algorithm benchmarking. The dashed lines go through medians, the error bars span from minima to maxima.

4 | Manipulation of floating spherical objects

In this chapter, I describe the mathematical models and control systems involved in manipulating multiple floating spherical objects. All of the described experiments were done with polypropylene balls having 10 mm in diameter.

4.1 Modeling a floating object

Due to its symmetry, a floating sphere can be modeled as a point mass with coordinates x and y. According to Newton's second law, the dynamics of the object are:

$$m\,\ddot{x} = -c_d\,\dot{x} + F_x\,, \qquad\qquad m\,\ddot{y} = -c_d\,\dot{y} + F_y\,, \tag{4.1}$$

where m is the mass of the object, c_d is the coefficient of drag, and F_x and F_y are the x- and y-components of the acoustophoretic force, respectively.

When modeling the acoustophoretic force, I used a control-oriented empirical approach. This means that the created model is simple enough to be computed in real-time and made to reflect the behavior of the system rather than the underlying physics. Depending on the method of acoustic field shaping, I propose two mathematical models: an amplitude fitting model and a gradient fitting model. These will be discussed next.

4.1.1 'Amplitude fitting' model

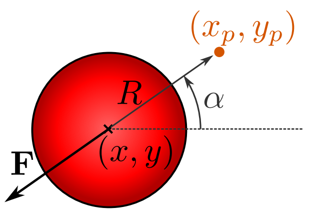
In the case of amplitude fitting, the control strategy is to create a high-pressure point at a fixed distance from the center of the sphere. The coordinates of this point, (x_p, y_p), can be expressed as:

$$x_p = x + R\,\cos\alpha\,, \qquad\qquad y_p = y + R\,\sin\alpha\,, \tag{4.2}$$

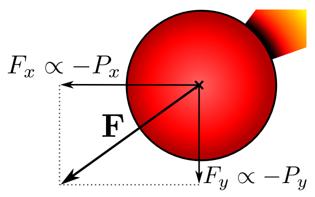
where R is a fixed distance, and α is a given angle.

Let P be the amplitude of the high-pressure point. Then, the components of the acoustophoretic force are equal to:

$$F_x = -c_p\,\max\left(P - 800, 0\right)\cos\alpha\,, \qquad F_y = -c_p\,\max\left(P - 800, 0\right)\sin\alpha\,, \tag{4.3}$$



(a) Acoustophoretic force created by a high-pressure point (represented by orange color).



(b) Acoustophoretic force created by a gradient of acoustic pressure (represented by color gradient in the upper right corner).

Figure 4.1: Acoustophoretic forces depending on the shape of the acoustic field

where c_p is a conversion coefficient. The max function in the formulas implements a dead zone — the acoustophoretic force is zero until the high-pressure point exceeds 800 pascals. This dead zone is caused by the fact that the force depends on the difference of pressures rather than the absolute value, and the amplitude outside the high-pressure point is usually 800 Pa, as can be seen in Figures 3.1 and 3.2. The relation between acoustic pressure and force is illustrated in Figure 4.1a.

Let v_x and v_y be velocities of the sphere in x- and y-coordinate, respectively. Let us further define inputs u_x and u_y as

$$u_x = -\max\left(P - 800, 0\right)\cos\alpha\,, \qquad u_y = -\max\left(P - 800, 0\right)\sin\alpha\,. \tag{4.4}$$

Let us define two sets of states, $\mathbf{x}_x = [v_x,\, x]^{\mathrm{T}}$ and $\mathbf{x}_y = [v_y,\, y]^{\mathrm{T}}$. Combining the equations above yields the following state-space description of the system:

$$\dot{\mathbf{x}}_x = \underbrace{\begin{bmatrix} -\frac{c_d}{m} & 0 \\ 1 & 0 \end{bmatrix}}_{\mathbf{A}_p} \mathbf{x}_x + \underbrace{\begin{bmatrix} \frac{c_p}{m} \\ 0 \end{bmatrix}}_{\mathbf{B}_p} u_x\,, \qquad \dot{\mathbf{x}}_y = \mathbf{A}_p\,\mathbf{x}_y + \mathbf{B}_p\,u_y\,, \tag{4.5a}$$

$$x = \underbrace{\begin{bmatrix} 0 & 1 \end{bmatrix}}_{\mathbf{C}_p} \mathbf{x}_x\,, \qquad y = \mathbf{C}_p\,\mathbf{x}_y\,. \tag{4.5b}$$

4.1.2 'Gradient fitting' model

If the acoustic field is shaped by gradient fitting, the point with the specified gradient is equivalent to the center of the sphere. Let P_x and P_y be the spatial derivatives of acoustic pressure modulus with respect to x and y. Then, the components of the acoustophoretic force are equal to

$$F_x = -c_g\,P_x\,, \qquad F_y = -c_g\,P_y\,, \tag{4.6}$$

where c_g is a conversion coefficient. The relation between acoustic pressure gradient and force is illustrated in Figure 4.1b.

Let us use the same sets of states as in the previous section. Combining (4.1) and (4.6) yields the following state space description:

$$\dot{\mathbf{x}}_x = \underbrace{\begin{bmatrix} -\frac{c_d}{m} & 0 \\ 1 & 0 \end{bmatrix}}_{\mathbf{A}_g} \mathbf{x}_x + \underbrace{\begin{bmatrix} -\frac{c_g}{m} \\ 0 \end{bmatrix}}_{\mathbf{B}_g} P_x \,, \qquad \dot{\mathbf{x}}_y = \mathbf{A}_g\, \mathbf{x}_y + \mathbf{B}_g\, P_y \,, \tag{4.7a}$$

$$x = \underbrace{\begin{bmatrix} 0 & 1 \end{bmatrix}}_{\mathbf{C}_g} \mathbf{x}_x \,, \qquad y = \mathbf{C}_g\, \mathbf{x}_y \,. \tag{4.7b}$$

4.2 Control system

The control system is created in Simulink. It runs on the Raspberry Pi with a sampling frequency of 25 Hz. The system comprises five distinct function blocks: a computer vision block, a bank of predictors (one for each manipulated object), a controller, an optimization algorithm, and a block for communication with the signal generators. A diagram with signals exchanged between these blocks is shown in Figure 4.2.

The computer vision block measures the position of the spheres based on images from the camera. I use the *raspi-ballpos* script, available at AA4CC research group's github[1], for this task. The optimization algorithm is described in Chapter 3, and the communication script is described in Chapter 2. In the following subsections, I will describe the predictor and the controller.

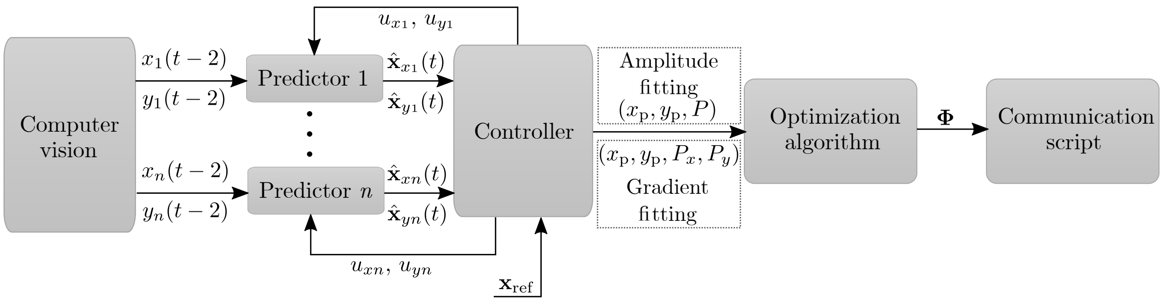

Computer vision
$x_1(t-2)$
$y_1(t-2)$
$x_n(t-2)$
$y_n(t-2)$
Predictor 1
Predictor n
u_{x1}, u_{y1}
u_{xn}, u_{yn}
$\hat{\mathbf{x}}_{x1}(t)$
$\hat{\mathbf{x}}_{y1}(t)$
$\hat{\mathbf{x}}_{xn}(t)$
$\hat{\mathbf{x}}_{yn}(t)$
Controller
$\mathbf{x}_{\text{ref}}$
Amplitude fitting $(x_\text{p}, y_\text{p}, P)$
$(x_\text{p}, y_\text{p}, P_x, P_y)$ Gradient fitting
Optimization algorithm
$\mathbf{\Phi}$
Communication script

Figure 4.2: Signal diagram of the control system.

[1] https://github.com/aa4cc/raspi-ballpos

4.2.1 Predictor

Due to the time it takes to process the camera image and the lag in the camera interface, the measured position is delayed by approximately 80 milliseconds, which is equivalent to two control periods. To compensate for this delay, I have designed a Kalman predictor. First, the measurement is brought into a Kalman filter, together with the delayed inputs. The filter outputs delayed estimates of states, $\hat{\mathbf{x}}_x(t-2)$ and $\hat{\mathbf{x}}_y(t-2)$. Undelayed estimates are obtained by applying the discretized state equation:

$$\hat{\mathbf{x}}_x(t) = \bar{\mathbf{A}}^2\,\hat{\mathbf{x}}_x(t-2) + \bar{\mathbf{B}}\,u_x(t-1) + \bar{\mathbf{A}}\,\bar{\mathbf{B}}\,u_x(t-2) \tag{4.8a}$$

$$\hat{\mathbf{x}}_y(t) = \bar{\mathbf{A}}^2\,\hat{\mathbf{x}}_y(t-2) + \bar{\mathbf{B}}\,u_y(t-1) + \bar{\mathbf{A}}\,\bar{\mathbf{B}}\,u_y(t-2)\,, \tag{4.8b}$$

where $\bar{\mathbf{A}}$ and $\bar{\mathbf{B}}$ are matrices obtained by ZOH discretization of the system described in (4.5a), or (4.7a), depending on the control mode.

4.2.2 Controller

The controller is a relatively complex block and can be further decomposed into function blocks. The schematic of the controller is shown in Figure 4.3

First, the estimated states are subtracted from reference setpoints and then multiplied by a state feedback gain $\mathbf{K}$. The controlled system has a pole in origin. Therefore, it is possible to achieve zero steady-state error when tracking a constant position reference without augmenting the system with integral control.

Then, the output of the collision avoidance system is added to the result. The collision avoidance system uses a variant of the commonly used virtual repulsive potential field (first introduced in [13]). The system outputs control action increments for the i^{th} object, Δu_{xi} and Δu_{yi}, based on the following formulas:

$$\Delta u_{xi} = c_r \sum_{j\in\{1,\dots,n\}\setminus\{i\}} \frac{\hat{x}_i - \hat{x}_j}{d_{ij}^3}\,, \qquad \Delta u_{yi} = c_r \sum_{j\in\{1,\dots,n\}\setminus\{i\}} \frac{\hat{y}_i - \hat{y}_j}{d_{ij}^3}\,, \tag{4.9}$$

where c_r is the repulsive force coefficient and d_{ij} is the mutual distance between i^{th} and j^{th} object.

Let us denote the resulting control actions $\tilde{u}_x$ and $\tilde{u}_y$. These control actions need to be limited, so they do not exceed the physical capabilities of the acoustic field. In the case of amplitude fitting, the acoustic pressure must be lower than 2500 Pa. From (4.4), the relation between the control actions and the required amplitude, P, is:

$$u_x^2 + u_y^2 = (P - 800)^2\,, \quad \text{if } P \geq 800\,. \tag{4.10}$$

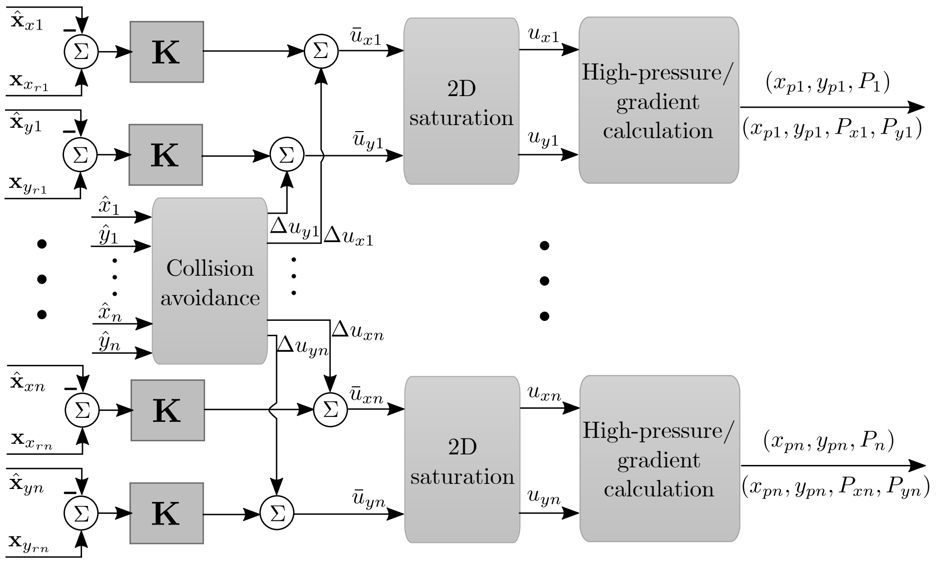


Figure 4.3: A signal diagram of the controller block.

Therefore, the control actions need to be limited, so that

$$u_x^2 + u_y^2 \leq 1700^2 \,. \tag{4.11}$$

In the case of gradient fitting, the optimization problem cannot yield a gradient of amplitude modulus greater than $1 \times 10^9\,\mathrm{Pa^2\,m^{-1}}$ in any direction. Therefore, the required gradients need to be limited, so that

$$P_x^2 + P_y^2 \leq \left(10^9\right)^2 \,. \tag{4.12}$$

In both cases, the constraint has the identical form, which can be generally expressed as $u_x^2 + u_y^2 \leq u_{max}^2$. To satisfy this constraint, I propose a 'two-dimensional saturation' function. Let us define an auxiliary variable $\bar{u} = \sqrt{\bar{u}_x^2 + \bar{u}_y^2}$. If $\bar{u}$ is less or equal than u_{max}, the saturated outputs are equal to the inputs. However, if $\bar{u}$ is greater than u_{max}, the outputs are:

$$u_x = \frac{u_{max}}{\bar{u}}\,\bar{u}_x \,, \qquad\qquad u_y = \frac{u_{max}}{\bar{u}}\,\bar{u}_y \,. \tag{4.13}$$

In the case of gradient fitting control, the saturated control actions are equal to the required spatial derivatives and the coordinates of the fitted point are equal to the estimated coordinates of the object. However, when using the amplitude fitting control mode, the saturated control actions need to be converted into position and amplitude of a high-pressure point. Recalling (4.2), the relations between the control actions and the coordinates of the point are:

$$x_p = \hat{x} + R\,\cos\alpha \,, \qquad\qquad y_p = \hat{y} + R\,\sin\alpha \,, \tag{4.14}$$

where the cosine and sine of the angle α are:

$$\cos\alpha = -\frac{u_x}{\sqrt{u_x^2 + u_y^2}}\,, \qquad \sin\alpha = -\frac{u_y}{\sqrt{u_x^2 + u_y^2}}\,. \tag{4.15}$$

Recalling (4.10), the relation between the control actions and the amplitude is:

$$P = \sqrt{u_x^2 + u_y^2} + 800\,. \tag{4.16}$$

4.3 Identification and parameter tuning

In this section, I explain how are the tasks of model identification and parameter tuning related, what issues does this relation create, and how did I solve them. To prevent working with too large or too small numbers, I have decided to use SI-prefixed units for some of the quantities and parameters. Namely, I am using mm for coordinates, $\mathrm{mm\,s^{-1}}$ for velocities, kPa for pressure, and $\mathrm{kPa^2\,m^{-1}}$ (equivalent to 10^6 $\mathrm{Pa^2\,m^{-1}}$) for the gradient of amplitude modulus.

4.3.1 Model identification

Recalling (4.5a) and (4.7a), the matrices $\mathbf{A}$ and $\mathbf{B}$ of the state-space description are in a form

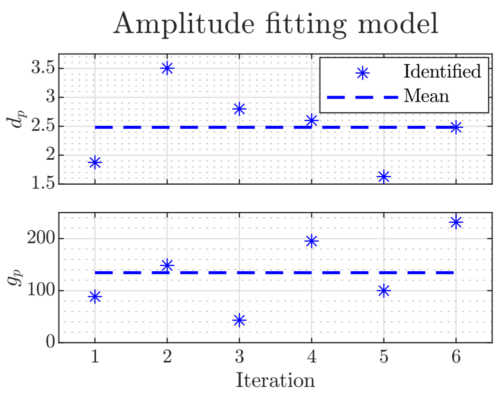
$$\mathbf{A} = \begin{bmatrix} -d & 0 \\ 1 & 0 \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} g \\ 0 \end{bmatrix}, \tag{4.17}$$

where d and g are unknown parameters, which need to be identified.

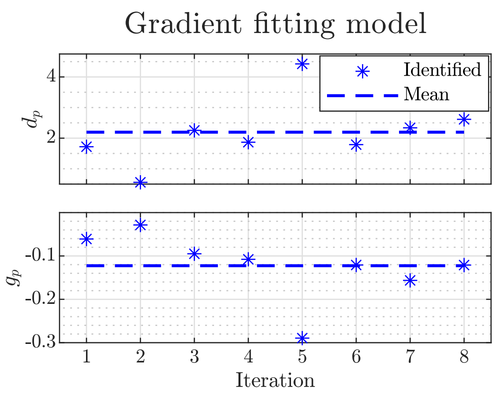
Due to the unstable dynamics of the system, I have decided to use closed-loop identification. However, this is a kind of chicken-and-egg problem. We want to identify the model of the system in a closed loop, but to close the loop, we plan to use a model-based controller (LQR+Kalman filter), which we cannot design without the identified model. While a simple P controller can replace the LQR for the purpose of identification, the Kalman predictor is essential in estimating the object's current position, which is then used to calculate the desired position of the high-pressure point or the gradient.

To solve this problem, I employ an iterative approach. I start with an initial guess of parameters d and g. Then, I perform an experiment with the closed-loop system and measure the inputs and outputs. Then, I refine the estimated parameters using the measured data, and the greybox identification tool provided in the Matlab System Identification Toolbox. This process of measurement and parameter update is repeated until the estimates converge.

Results of the iterative identification are shown in Figure 4.4. I am using the mean of all experiments as the final value of the parameters. The values of d slightly differ in amplitude



(a) Identified parameters of amplitude fitting model.



(b) Identified parameters of gradient fitting model.

Figure 4.4: Results of iterative parameter identification.

fitting and gradient fitting, with $d_p = 2.48$ and $d_g = 2.20$. The value of g_p is 134 and the value of g_g is -0.122.

4.3.2 Parameter tuning

The value of the LQ-optimal state feedback gain $\mathbf{K}$ is determined by the weight matrices $\mathbf{Q}$ and $\mathbf{R}$. I am using a diagonal matrix for $\mathbf{Q}$, and since the model of the object is composed of two identical SISO systems, $\mathbf{R}$ is a scalar. Therefore, there are three parameters that need to be tuned.

I begin by setting both $\mathbf{R}$ and the first diagonal element of $\mathbf{Q}$ to one. Then, I set the second diagonal element of $\mathbf{Q}$ so that the simulated response of the system to a step in position reference settles as fast as possible without overshooting. The resulting matrix $\mathbf{Q}$ is identical for amplitude fitting and gradient fitting model:

$$\mathbf{Q} = \begin{bmatrix} 1 & 0 \\ 0 & 10 \end{bmatrix} . \tag{4.18}$$

The value of $\mathbf{R}$ is then set so that the control action is not saturated if the position error is less than 10 mm. The resulting value of $\mathbf{R}$ is 1×10^{-8} for amplitude fitting, and 1×10^{-5} for gradient fitting.

Another set of iteratively tuned parameters are the noise characteristics for the Kalman filter. To avoid confusion with the LQR weights, let us denote the process noise covariance matrix as $\mathbf{W}$ and the measurement noise covariance as $\mathbf{V}$. Let us assume that in the continuos-time system, the disturbances only affect the derivative of velocity. Therefore, the continuos-time stochastic state-space equations of the system are:

$$\dot{\mathbf{x}} = \mathbf{A}\,\mathbf{x} + \mathbf{B}\,u + \mathbf{\Gamma}\,w\,, \tag{4.19a}$$

$$y = \mathbf{C}\,\mathbf{x} + v\,, \tag{4.19b}$$

where $\mathbf{\Gamma} = \begin{bmatrix}1 & 0\end{bmatrix}^{\mathrm{T}}$, and w and v are white, zero-mean uncorrelated Gaussian stochastic processes with covariances $E\left[w^2\right] = q$ and $E\left[v^2\right] = \mathbf{V}$. While the covariance of the measurement noise, $\mathbf{V}$, is identical in the continuous- and discrete-time systems, the covariance of the discrete-time process noise needs to be calculated using the following formula[14, p. 397]:

$$\mathbf{W} = \int_0^{T_s} \mathrm{e}^{\mathbf{A}\tau}\,\mathbf{\Gamma}\,q\,\mathbf{\Gamma}^{\mathrm{T}}\,\mathrm{e}^{\mathbf{A}^{\mathrm{T}}\tau}\,\mathrm{d}\tau\,, \tag{4.20}$$

where T_s is the sampling period.

By default, the Matlab implementation of the Kalman filter uses the steady-state Kalman gain, obtained by solving the dual LQR problem [15]. Therefore, the performance of the filter is determined by the ratio of the process and measurement noise covariances rather than by their actual values. If the Kalman filter is optimal, its innovation sequence (*i.e.*, the difference between measured and predicted outputs) is uncorrelated [16]. Therefore, the matrices $\mathbf{W}$ and $\mathbf{V}$ can be iteratively tuned until the normalized autocovariance of the resulting innovation sequence is minimal. The value of $\mathbf{V}$ can be obtained by calculating the autocovariance of the measured position when the object remains at rest. The resulting value of $\mathbf{V}$ is 1.17. The matrices $\mathbf{W}_p$ and $\mathbf{W}_g$ for amplitude fitting and gradient fitting model, respectively, are:

$$\mathbf{W}_p = \begin{bmatrix} 12.0 & 0.12 \\ 0.12 & 0.00162 \end{bmatrix}, \qquad \mathbf{W}_g = \begin{bmatrix} 24.7 & 0.247 \\ 0.247 & 0.00333 \end{bmatrix}. \tag{4.21}$$

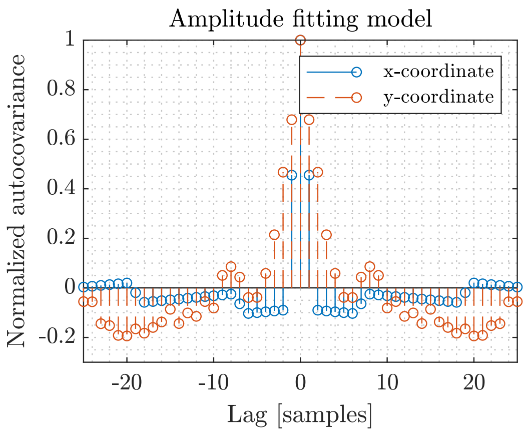
The resulting autocovariance sequences are shown in Figure 4.5.

4.4 Results

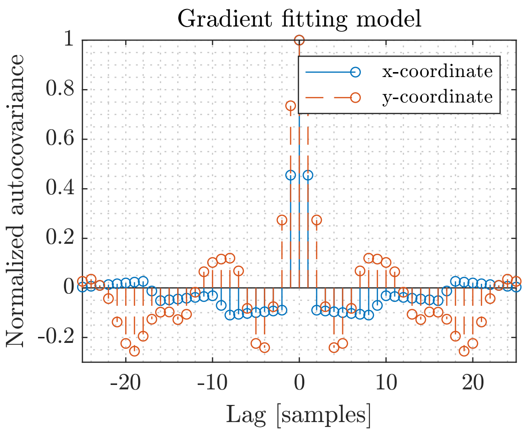
To test the ability of the control system to track a moving reference, I have designed two types of reference trajectories. Let x_{ri} and y_{ri} be the position reference of the i^{th} object. Then, the first trajectory type can be described by the following equations:

$$x_{ri}(t) = r_x \cos\left(\vartheta_i(t)\right)\,, \qquad y_{ri}(t) = r_y \sin\left(2\,\vartheta_i(t)\right)\,, \qquad \vartheta_i(t) = \frac{2\pi}{T}\left(t + \frac{i}{n}T\right)\,, \tag{4.22}$$

where r_x and r_y are the radii in x- and y-coordinates, T is the period of the trajectory, and n is the total number of objects. The shape of this trajectory for $r_x = r_y = 40$ and $n = 3$ is shown in Figure 4.6a. Note that the objects follow identically shaped trajectories and only



(a) Amplitude fitting model



(b) Gradient fitting model

Figure 4.5: Normalized autocovariance of innovation sequences

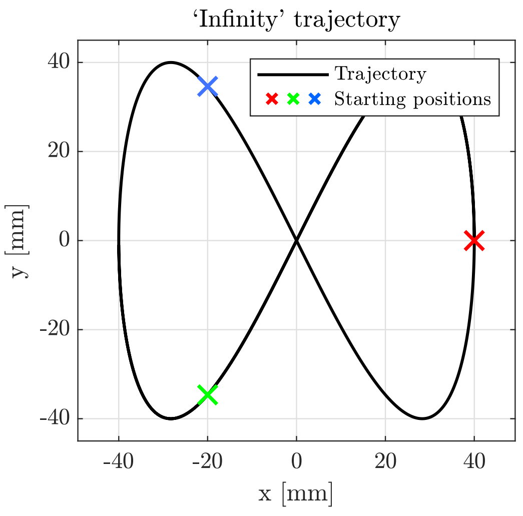
differ in starting positions (*i.e.*, $x_{ri}(0)$ and $y_{ri}(0)$). Due to its shape, I am referring to this type as the 'Infinity' trajectory.

The second type is inspired by the movement of the end effector of a robotic arm with two links and two free joints. It is defined by the following equations:

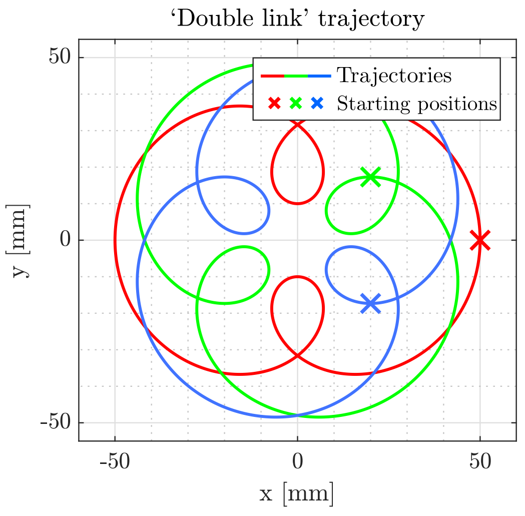
$$x_{ri}(t) = r_1 \cos(\vartheta_{1i}(t)) + r_2 \cos(\vartheta_{2i}(t)) \,, \qquad \vartheta_{1i}(t) = \frac{2\pi}{T_1}\, t \,, \tag{4.23a}$$

$$y_{ri}(t) = r_1 \sin(\vartheta_{1i}(t)) + r_2 \sin(\vartheta_{2i}(t)) \,, \qquad \vartheta_{2i}(t) = \frac{2\pi}{T_2}\left(t + \frac{i}{n} T_2\right) \,, \tag{4.23b}$$

where r_1 and r_2 are the radii of the links, and T_1 and T_2 are the periods of the links. The



(a) 'Infinity' trajectory for $r_x = r_y = 40$



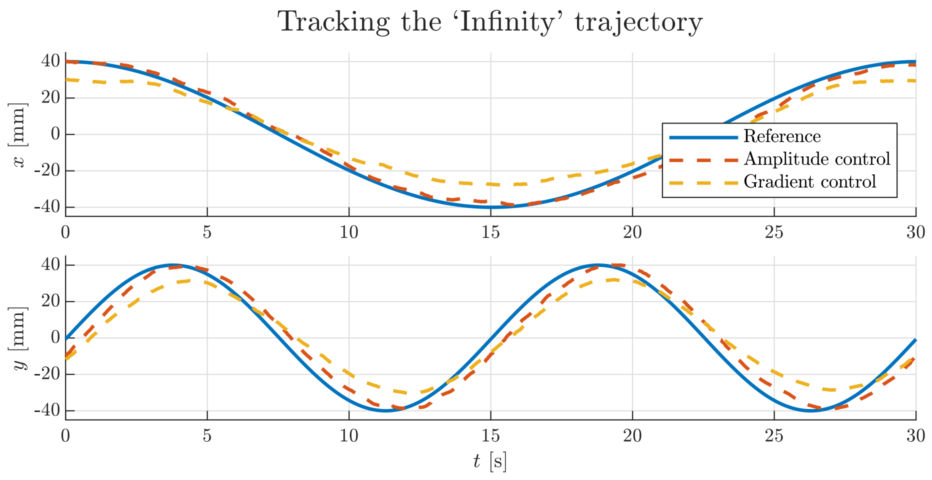
(b) 'Double link' trajectory for $r_1 = 30$, $r_2 = 20$, and $T_1 = 3\,T_2$

Figure 4.6: The tested trajectories

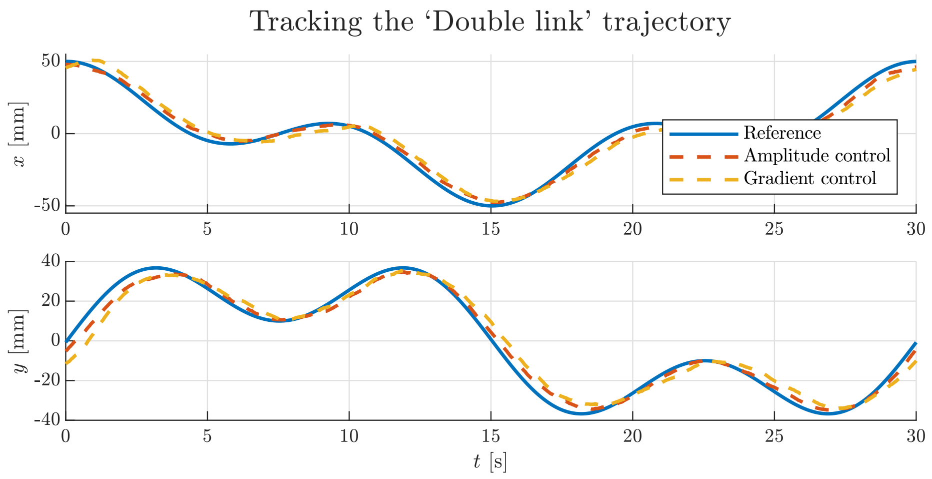
shape of the trajectories for $r_1 = 30$, $r_2 = 20$, $T_1 = 3\,T_2$, and $n = 3$ is shown in Figure 4.6b.

Note that the objects follow identically shaped but differently rotated trajectories. Also, note that at all times, the reference positions form a regular polygon (in this case, an equilateral triangle). For brevity, I am referring to this type of trajectory as the ‘Double link’ trajectory.

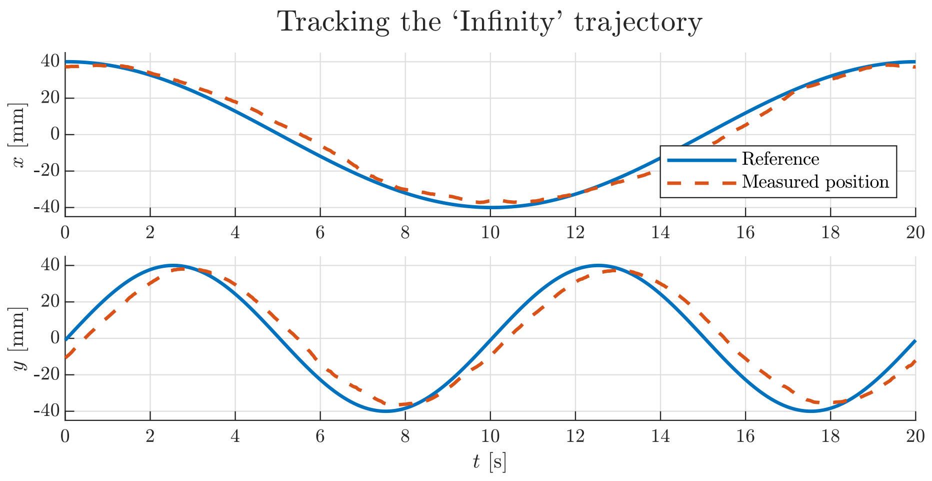
Results of trajectory tracking are shown in Figure 4.7. Figures 4.7a and 4.7b show the comparison between amplitude fitting and gradient fitting control when tracking the trajectories. In both cases, amplitude fitting control is better than gradient fitting control in terms of distance from the reference trajectory. As shown in Figure 4.7c, amplitude fitting control can track the ‘Infinity’ trajectory with a period of 20 seconds, reaching speeds of up to 20 mm s^{-1} and accelerations of up to 14 mm s^{-2}, whereas gradient fitting control fails when tracking such a fast-moving reference. Figure 4.7d shows how the control system responds to changing the tracked trajectory. Supplementary video (see Appendix A.2) then shows how the collision avoidance system prevents the objects from getting too close to each other during the change.



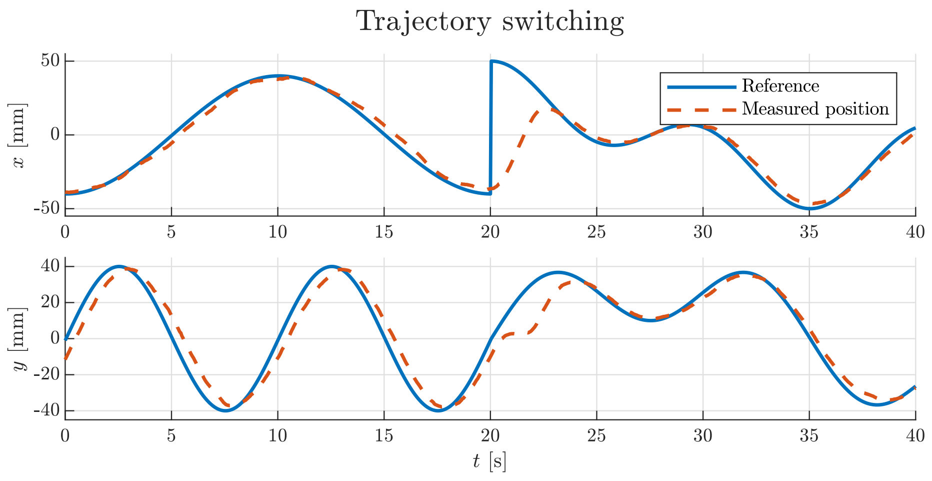
(a) Performance of amplitude fitting and gradient fitting control when tracking the ‘Infinity’ trajectory



(b) Performance of amplitude fitting and gradient fitting control when trracking the ‘Double link’ trajectory



(c) Performance of amplitude fitting control when tracking the 'Infinity' trajectory with a 20 second period



(d) Performance of amplitude fitting control when switching between the tracked trajectories

Figure 4.7: Results of trajectory tracking with floating spherical objects

5 | Manipulation of planar objects

In this chapter, I describe the modifications to the model and the control system needed to manipulate planar objects. As a test subject, I am using an isosceles triangle, 3D-printed from thermoplastic polyurethane (TPU 95A) with a 2 mm thickness. For modeling and control design, I am using only the 'amplitude fitting' method, since it has proven to be more precise than the 'gradient fitting' method.

5.1 Mathematical model

First, let us establish some conventions regarding the triangular object. Let us denote its vertices as E, F, and G, with the side $\overline{EF}$ being the base and the vertex G being the apex. The size of the triangle is given by the length of its base, l, and its height, h. Let H be the centroid of the triangle. The position of the triangle is defined by the coordinates of the centroid, x and y, and its orientation, θ, defined as the angle of the vector $\overrightarrow{HG}$. The conventions are illustrated in Figure 5.1a.

Let us, once more, use Newton's second law to obtain the dynamical equations of the object. Furthermore, let us assume that the drag force and torque are proportional to the velocity and the angular speed, respectively. Therefore, the resulting dynamical equations are:

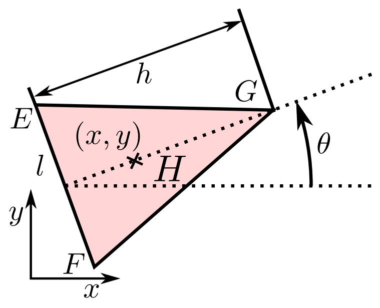
$$m\,\ddot{x} = -c_{dt}\,\dot{x} + F_x\,, \qquad m\,\ddot{y} = -c_{dt}\,\dot{y} + F_y\,, \qquad J\,\ddot{\theta} = -c_{dr}\,\dot{\theta} + T\,, \tag{5.1}$$

where m is the mass of the object, J is the moment of inertia, c_{dt} and c_{dr} are the translational and rotational coefficients of drag, respectively, F_x and F_y are the components of the acoustophoretic force, and T is the torque induced by the acoustophoretic force.

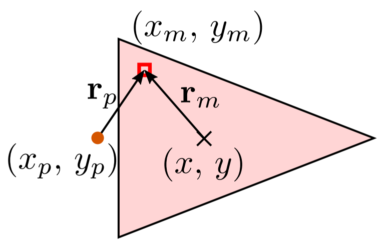
Having the dynamical equations (5.1), it remains to figure out how the pressure field relates to the net forces F_x, F_y, and net torque T acting on the manipulated object. For this purpose, we can view the acoustophoretic force as analogous to the Coulomb force. Let (x_p, y_p) be the coordinates of a high-pressure point with an amplitude P. Let ΔP be the amplitude of pressure after applying the dead zone introduced in (4.3), *i.e.*,

$$\Delta P = \max\left(P - 800, 0\right)\,. \tag{5.2}$$

Let us consider an infinitesimally small element of the mass of the triangular object, $\mathrm{d}m$,



(a) Definition of position and orientation



(b) Vectors used in the calculation of forces

Figure 5.1: Geometry of the triangle

with coordinates (x_m, y_m). Then, the acoustophoretic force $\mathrm{d}\mathbf{F} = [\mathrm{d}F_x,\, \mathrm{d}F_y]^\mathrm{T}$ and torque $\mathrm{d}T$ acting on this element are proportional to:

$$\mathrm{d}\mathbf{F} \propto \frac{\Delta P}{\|\mathbf{r}_p\|^3}\mathbf{r}_p\,\mathrm{d}m\,, \qquad \mathrm{d}T \propto \det\left[\mathbf{r}_m \quad \tfrac{\Delta P}{\|\mathbf{r}_p\|^3}\mathbf{r}_p\right]\,\mathrm{d}m\,, \tag{5.3}$$

where $\mathbf{r}_m = [x_m - x,\, y_m - y]^\mathrm{T}$ and $\mathbf{r}_p = [x_p - x,\, y_p - y]^\mathrm{T}$. An illustration of this model of forces is shown in Figure 5.1b.

According to (5.3), the acoustophoretic force and torque are proportional to ΔP. Thus, let us define a normalized 'pseudo-force', $\mathbf{f} = [f_x,\, f_y]^\mathrm{T}$, and 'pseudo-torque', τ, that are independent on ΔP. The net acoustophoretic force and torque are then given by:

$$F_x \propto f_x\,\Delta P\,, \qquad F_y \propto f_y\,\Delta P\,, \qquad T \propto \tau\,\Delta P\,, \tag{5.4}$$

To determine $\mathbf{f}$ and τ, we need to integrate (5.3) with $\Delta P = 1$. For that, I propose three models.

The first model assumes that the acoustophoretic force affects the full volume of the triangular object. Therefore, $\mathbf{f}$ and τ can be expressed using the following integrals:

$$\mathbf{f} = \iint_{\triangle EFG} \frac{1}{\|\mathbf{r}_p\|^3}\mathbf{r}_p\,\mathrm{d}y_m\,\mathrm{d}x_m\,, \qquad \tau = \iint_{\triangle EFG} \det\left[\mathbf{r}_m \quad \tfrac{1}{\|\mathbf{r}_p\|^3}\mathbf{r}_p\right]\,\mathrm{d}y_m\,\mathrm{d}x_m\,, \tag{5.5}$$

where the double integral denotes integration over the surface of the triangle EFG.

The second model assumes that the acoustophoretic force affects only edges, which are exposed to the high-pressure point. For instance, in the situation shown in Figure 5.1b, the only exposed edge is $\overline{EF}$. In that situation, $\mathbf{f}$ and τ can be expressed using the following

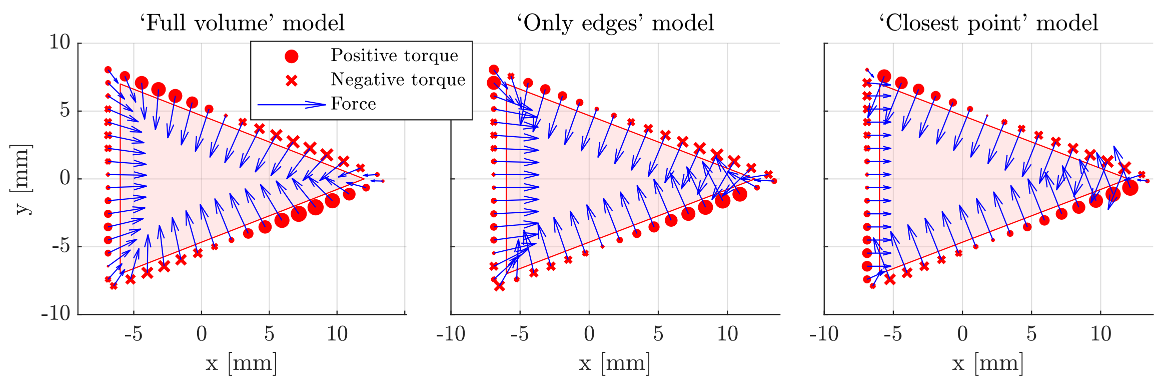


Figure 5.2: Models of pseudo-forces and pseudo-torques

integrals:

$$\mathbf{f} = \int_0^1 \frac{1}{\|\mathbf{r}_p\|^3}\mathbf{r}_p \, \mathrm{d}\rho\,, \qquad x_m = (1-\rho)\,x_E + \rho\, x_F\,, \tag{5.6a}$$

$$\tau = \int_0^1 \det\left[\mathbf{r}_m \quad \tfrac{1}{\|\mathbf{r}_p\|^3}\mathbf{r}_p\right]\, \mathrm{d}\rho\,, \qquad y_m = (1-\rho)\,y_E + \rho\, y_F\,, \tag{5.6b}$$

where $(x_E,\, y_E)$ are the coordinates of the vertex E, and $(x_F,\, y_F)$ are the coordinates of the vertex F.

The third model assumes that the acoustophoretic force affects only the nearest point on the perimeter of the triangle. Therefore, $\mathbf{f}$ and τ can be expressed as:

$$\mathbf{f} = \frac{1}{\|\mathbf{r}_p\|^3}\mathbf{r}_p\,, \quad \tau = \det\left[\mathbf{r}_m \quad \tfrac{1}{\|\mathbf{r}_p\|^3}\mathbf{r}_p\right]\,, \quad \begin{bmatrix} x_m \\ y_m \end{bmatrix} = \underset{[x_m,\, y_m]^\mathrm{T} \in \triangle EFG}{\arg\min} \left\| \begin{bmatrix} x_p - x_m \\ y_p - y_m \end{bmatrix} \right\|_2 . \tag{5.7}$$

These three models are shown in Figure 5.2. The blue arrows represent the direction and magnitude of the 'pseudo-force' for different high-pressure points around the triangle. The red markers represent the coordinates of the high-pressure points, as well as the direction and magnitude of the 'pseudo-torque' — a circle means positive torque, a cross means negative torque, and the size of the marker increases with the magnitude.

Let us define three sets of states, $\mathbf{x}_x = [v_x,\, x]^\mathrm{T}$, $\mathbf{x}_y = [v_y,\, y]^\mathrm{T}$, and $\mathbf{x}_\theta = [\omega,\, \theta]^\mathrm{T}$, where $v_x = \dot{x}$, $v_y = \dot{y}$, and $\omega = \dot{\theta}$. Combining the models of the 'pseudo-force' and 'pseudo-torque' with the dynamical equations from (5.1) yields the following state-space description:

$$\dot{\mathbf{x}}_x = \mathbf{A}_t\,\mathbf{x}_x + \mathbf{B}_t\, f_x\, \Delta P\,, \qquad x = \mathbf{C}\,\mathbf{x}_x\,, \tag{5.8a}$$

$$\dot{\mathbf{x}}_y = \mathbf{A}_t\,\mathbf{x}_y + \mathbf{B}_t\, f_y\, \Delta P\,, \qquad y = \mathbf{C}\,\mathbf{x}_y\,, \tag{5.8b}$$

$$\dot{\mathbf{x}}_\theta = \mathbf{A}_r\,\mathbf{x}_\theta + \mathbf{B}_r\, \tau\, \Delta P\,, \qquad \theta = \mathbf{C}\,\mathbf{x}_x\,, \tag{5.8c}$$

where $\mathbf{C} = [0\ 1]$, and the matrices $\mathbf{A}_t$, $\mathbf{A}_r$, $\mathbf{B}_t$, and $\mathbf{B}_r$ are in the same form as in (4.17), *i.e.*,

$$\mathbf{A} = \begin{bmatrix} -d & 0 \\ 1 & 0 \end{bmatrix} , \qquad \mathbf{B} = \begin{bmatrix} g \\ 0 \end{bmatrix} . \tag{5.9}$$

5.2 Control system

The control system for triangular objects has the same structure as the control system for spherical objects presented in Section 4.2. The major difference between the systems is in the controller. I propose two controllers, an LQR-based, and an MPC-based controller.

5.2.1 LQR

Let $\mathbf{x}_{xr}$, $\mathbf{x}_{yr}$ and $\mathbf{x}_{\theta r}$ be the reference setpoints. Furthermore, let us define errors $\tilde{\mathbf{x}}_x$, $\tilde{\mathbf{x}}_y$ and $\tilde{\mathbf{x}}_\theta$ as:

$$\tilde{\mathbf{x}}_x = \mathbf{x}_x - \mathbf{x}_{xr} , \qquad \tilde{\mathbf{x}}_y = \mathbf{x}_y - \mathbf{x}_{yr} , \qquad \tilde{\mathbf{x}}_\theta = \mathbf{x}_\theta - \mathbf{x}_{\theta r} . \tag{5.10}$$

Now, let us define the LQR control actions as:

$$u_x = -\mathbf{K}_t \, \tilde{\mathbf{x}}_x , \qquad u_y = -\mathbf{K}_t \, \tilde{\mathbf{x}}_y , \qquad u_\theta = -\mathbf{K}_r \, \tilde{\mathbf{x}}_\theta , \tag{5.11}$$

where $\mathbf{K}_t$ and $\mathbf{K}_r$ are state feedback gain for the translational and rotational systems, respectively. From (5.8), it follows that the control inputs should be equal to:

$$u_x = f_x \, \Delta P , \qquad u_y = f_y \, \Delta P , \qquad u_\theta = \tau \, \Delta P . \tag{5.12}$$

The ‘pseudo-force’ and ‘pseudo-torque’ depend on the position of the high-pressure point. Therefore, we need to find x_p, y_p and ΔP that satisfy (5.12). Since the formulas for the ‘pseudo-force’ and ‘pseudo-torque’ in (5.5), (5.6) and (5.7) are too complex to be optimized in real-time, I use a lookup table. I take m evenly spaced points around the triangle (similarly as in Figure 5.2) and calculate the values of $\mathbf{f}$ and τ for each point. Therefore, each entry of the lookup table is a five-tuple

$$(x_{p,i}, \, y_{p,i} \, , f_{x,i} \, , f_{y,i} \, , \tau_i) \; , \qquad i = 1, \, 2, \, \ldots, \, m . \tag{5.13}$$

The values of the ‘pseudo-forces’ and ‘pseudo-torques’ are pre-computed for a triangle situated at the origin with its orientation equal to zero. Therefore, the LQR control inputs need to be transformed into $\bar{u}_x$ and $\bar{u}_y$ so that

$$\begin{bmatrix} \bar{u}_x \\ \bar{u}_y \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} u_x \\ u_y \end{bmatrix} . \tag{5.14}$$

Finding the optimal position and amplitude of the high-pressure point involves solving the following mixed-integer program:

$$i^*,\, \Delta P^* = \underset{i\in\mathbb{N},\, \Delta P\in\mathbb{R}}{\arg\min} \left[(\bar{u}_x - f_{xi}\,\Delta P)^2 + (\bar{u}_y - f_{yi}\,\Delta P)^2 + w_\tau\,(u_\theta - \tau_i\,\Delta P)^2 \right], \tag{5.15a}$$

$$\text{subject to } \; i \in \{1, 2, \ldots, m\}\,, \tag{5.15b}$$

$$0 \le \Delta P \le 1700\,, \tag{5.15c}$$

where w_τ is a torque weight. The 'pseudo-forces' and 'pseudo-torques' are sampled at $m = 300$ points. The relatively small size of the problem allows the use of a brute force approach: we enumerate all i. For each i, we compute the optimal amplitude ΔP_i^*, which is given by the saturated weighted average

$$\Delta P_i^* = \max\left(\min\left(\frac{\frac{\bar{u}_x}{f_{x,i}} + \frac{\bar{u}_y}{f_{y,i}} + \sqrt{w_\tau}\,\frac{\bar{u}_\theta}{\tau_i}}{2+\sqrt{w_\tau}}, 1700 \right), 0 \right), \tag{5.16}$$

and store the corresponding value of the cost function. The solution of (5.15) is then readily given by i and ΔP_i^* resulting in the smallest value of the cost function.

The final coordinates of the high-pressure point are obtained form the following transformation:

$$\begin{bmatrix} x_p \\ y_p \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_{p,i^*} \\ y_{p,i^*} \end{bmatrix}, \tag{5.17}$$

and the amplitude of the high-pressure point is:

$$P = \Delta P^* + 800\,. \tag{5.18}$$

5.2.2 MPC

Let us define a complete state of the triangular object $\mathbf{x} = \left[\mathbf{x}_x^{\mathrm{T}}\ \mathbf{x}_y^{\mathrm{T}}\ \mathbf{x}_\theta^{\mathrm{T}}\right]^{\mathrm{T}}$. The discrete-time state-space description of the system is then given by

$$\mathbf{x}(t+1) = \underbrace{\begin{bmatrix} \bar{\mathbf{A}}_t & 0 & 0 \\ 0 & \bar{\mathbf{A}}_t & 0 \\ 0 & 0 & \bar{\mathbf{A}}_r \end{bmatrix}}_{\bar{\mathbf{A}}} \mathbf{x}(t) + \underbrace{\begin{bmatrix} \bar{\mathbf{B}}_t & 0 & 0 \\ 0 & \bar{\mathbf{B}}_t & 0 \\ 0 & 0 & \bar{\mathbf{B}}_r \end{bmatrix}}_{\bar{\mathbf{B}}} \underbrace{\begin{bmatrix} f_x(t) \\ f_y(t) \\ \tau(t) \end{bmatrix}}_{\boldsymbol{\nu}(t)} \Delta P(t)\,, \tag{5.19}$$

where $\bar{\mathbf{A}}_t$, $\bar{\mathbf{A}}_r$, $\bar{\mathbf{B}}_t$, and $\bar{\mathbf{B}}_r$ are ZOH-discretized state matrices from (5.8).

Let h_p be the length of the prediction horizon and let h_c be the length of the control horizon of the MPC ($h_p \ge h_c$). Furthermore, let us define a complete error of the triangular object

$\tilde{\mathbf{x}} = \begin{bmatrix}\tilde{\mathbf{x}}_x^{\mathrm{T}} & \tilde{\mathbf{x}}_y^{\mathrm{T}} & \tilde{\mathbf{x}}_\theta^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}}$. The goal of model predictive control is to minimize the following criterion:

$$J = \frac{1}{2}\sum_{i=1}^{h_p} \tilde{\mathbf{x}}^{\mathrm{T}}(t+i)\,\mathbf{Q}\,\tilde{\mathbf{x}}(t+i) + \frac{1}{2}\sum_{i=0}^{h_c-1} \mathbf{R}\,\left(\Delta P(t+i)\right)^2 \,, \tag{5.20}$$

where $\mathbf{Q}$ is a 6-by-6 positive definite matrix and $\mathbf{R}$ is a positive scalar.

Now, let us express the criterion in a compact matrix form. To do so, let us define a stacked vector of states, $\mathbf{X}$, a stacked vector of references, $\mathbf{X}_r$, and a stacked vector of amplitudes, $\mathbf{P}$:

$$\mathbf{X} = \begin{bmatrix}\mathbf{x}(t+1)\\ \vdots \\ \mathbf{x}(t+h_p)\end{bmatrix}, \qquad \mathbf{X}_r = \begin{bmatrix}\mathbf{x}_r(t+1)\\ \vdots \\ \mathbf{x}_r(t+h_p)\end{bmatrix}, \qquad \mathbf{P} = \begin{bmatrix}\Delta P(t)\\ \vdots \\ \Delta P(t+h_c-1)\end{bmatrix}, \tag{5.21}$$

where $\mathbf{x}_r(t+i) = \mathbf{x}_r(t)$ if the future references are not known. Let us also define two matrices

$$\bar{\mathbf{Q}} = \underbrace{\begin{bmatrix}\mathbf{Q} & & \\ & \ddots & \\ & & \mathbf{Q}\end{bmatrix}}_{h_p \text{ times}}, \qquad \bar{\mathbf{R}} = \underbrace{\begin{bmatrix}\mathbf{R} & & \\ & \ddots & \\ & & \mathbf{R}\end{bmatrix}}_{h_c \text{ times}}. \tag{5.22}$$

Now, the criterion J can be expressed in a compact matrix form:

$$J = (\mathbf{X} - \mathbf{X}_r)^{\mathrm{T}}\,\bar{\mathbf{Q}}\,(\mathbf{X} - \mathbf{X}_r) + \mathbf{P}^{\mathrm{T}}\,\bar{\mathbf{R}}\,\mathbf{P}\,. \tag{5.23}$$

The system presented in (5.19) is nonlinear due to the multiplication of inputs $\boldsymbol{\nu}$ and ΔP. However, if the sequence of inputs $\boldsymbol{\nu}(t)$ is fixed, the system becomes linear time-varying. For the moment, let us postpone the discssuion on how to choose the sequence $\boldsymbol{\nu}(t)$ and assume that we have already fixed it. Then, the goal of the MPC is to find an optimal sequence of amplitudes, $\mathbf{P}^*$, so that:

$$\mathbf{P}^* = \arg\min_{\mathbf{P}\in\mathbb{R}^{h_c}} J\,, \tag{5.24a}$$

$$\text{subject to } \mathbf{x}(k+1) = \begin{cases}\bar{\mathbf{A}}\,\mathbf{x}(k) + \bar{\mathbf{B}}\,\boldsymbol{\nu}(k)\,\Delta P(k)\,, & k = t,\,\ldots,\,t+h_c-1\,,\\ \bar{\mathbf{A}}\,\mathbf{x}(k)\,, & k = t+h_c,\,\ldots,\,t+h_p-1\,,\end{cases} \tag{5.24b}$$

$$\mathbf{x}(t) = \mathbf{x}_0\,, \qquad \begin{bmatrix}0\\ \vdots \\ 0\end{bmatrix} \le \mathbf{P} \le \begin{bmatrix}1700\\ \vdots \\ 1700\end{bmatrix}. \tag{5.24c}$$

The optimization problem presented in (5.24) is the so-called sparse MPC formulation, which contains equality constraints. We can eliminate these constraints by deriving the condensed

MPC formulation [17]. First, we need to express the stacked states. From the state-space equations, we obtain

$$\mathbf{X} = \hat{\mathbf{A}}\,\mathbf{x}_0 + \mathcal{C}\,\mathbf{P}\,, \tag{5.25}$$

where

$$\hat{\mathbf{A}} = \begin{bmatrix} \bar{\mathbf{A}} \\ \bar{\mathbf{A}}^2 \\ \vdots \\ \bar{\mathbf{A}}^{h_p} \end{bmatrix}, \quad \mathcal{C} = \begin{bmatrix} \bar{\mathbf{B}}\,\boldsymbol{\nu}(t) & & & \\ \bar{\mathbf{A}}\,\bar{\mathbf{B}}\,\boldsymbol{\nu}(t) & \bar{\mathbf{B}}\,\boldsymbol{\nu}(t+1) & & \\ \vdots & \vdots & \ddots & \\ \bar{\mathbf{A}}^{h_c-1}\,\bar{\mathbf{B}}\,\boldsymbol{\nu}(t) & \bar{\mathbf{A}}^{h_c-2}\,\bar{\mathbf{B}}\,\boldsymbol{\nu}(t+1) & \cdots & \bar{\mathbf{B}}\,\boldsymbol{\nu}(t+h_c-1) \\ \bar{\mathbf{A}}^{h_c}\,\bar{\mathbf{B}}\,\boldsymbol{\nu}(t) & \bar{\mathbf{A}}^{h_c-1}\,\bar{\mathbf{B}}\,\boldsymbol{\nu}(t+1) & \cdots & \bar{\mathbf{A}}\,\bar{\mathbf{B}}\,\boldsymbol{\nu}(t+h_c-1) \\ \vdots & \vdots & & \vdots \\ \bar{\mathbf{A}}^{h_p-1}\,\bar{\mathbf{B}}\,\boldsymbol{\nu}(t) & \bar{\mathbf{A}}^{h_p-2}\,\bar{\mathbf{B}}\,\boldsymbol{\nu}(t+1) & \cdots & \bar{\mathbf{A}}^{h_p-h_c}\,\bar{\mathbf{B}}\,\boldsymbol{\nu}(t+h_c-1) \end{bmatrix}. \tag{5.26}$$

Substituting (5.25) to (5.23) yields:

$$J = \frac{1}{2}\mathbf{P}^{\mathrm{T}}\mathcal{C}^{\mathrm{T}}\,\bar{\mathbf{Q}}\,\mathcal{C}\,\mathbf{P} + \left(\hat{\mathbf{A}}\,\mathbf{x}_0 - \mathbf{X}_r\right)^{\mathrm{T}}\bar{\mathbf{Q}}\,\mathcal{C}\,\mathbf{P} + \underline{\left(\hat{\mathbf{A}}\mathbf{x}_0 - \mathbf{X}_r\right)^{\mathrm{T}}\bar{\mathbf{Q}}\left(\hat{\mathbf{A}}\mathbf{x}_0 - \mathbf{X}_r\right)} + \frac{1}{2}\mathbf{P}^{\mathrm{T}}\,\bar{\mathbf{R}}\,\mathbf{P}\,. \tag{5.27}$$

The underlined term is constant with respect to the optimized variable and can be therefore omitted in the criterion. Thus, the optimal sequence of amplitudes is obtained by solving the following quadratic program:

$$\mathbf{P}^* = \operatorname*{arg\,min}_{\mathbf{P}\in\mathbb{R}^{h_c}} \frac{1}{2}\mathbf{P}^{\mathrm{T}}\,\mathbf{H}\,\mathbf{P} + \mathbf{F}^{\mathrm{T}}\,\mathbf{P}\,, \tag{5.28a}$$

$$\text{subject to} \quad \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \le \mathbf{P} \le \begin{bmatrix} 1700 \\ \vdots \\ 1700 \end{bmatrix}, \tag{5.28b}$$

where

$$\mathbf{H} = \mathcal{C}^{\mathrm{T}}\,\bar{\mathbf{Q}}\,\mathcal{C} + \bar{\mathbf{R}}\,, \qquad \mathbf{F}^{\mathrm{T}} = \left(\hat{\mathbf{A}}\,\mathbf{x}_0 - \mathbf{X}_r\right)^{\mathrm{T}}\bar{\mathbf{Q}}\,\mathcal{C}\,. \tag{5.29}$$

Note that the matrices of the state-space representation in (5.19) are block diagonal. This property means that $\hat{\mathbf{A}}$ and $\mathcal{C}$ are sparse — they are, effectively, blocks of block diagonal matrices. Consequently, calculating $\mathbf{H}$ and $\mathbf{F}$ is too lengthy due to the multiplication of many zero elements.

However, if the matrix $\mathbf{Q}$ is in a block diagonal form

$$\mathbf{Q} = \begin{bmatrix} \mathbf{Q}_t & 0 & 0 \\ 0 & \mathbf{Q}_t & 0 \\ 0 & 0 & \mathbf{Q}_r \end{bmatrix} , \tag{5.30}$$

we can express $\mathbf{H}$ and $\mathbf{F}$ using smaller, dense matrices:

$$\mathbf{H} = \mathcal{C}_x^{\mathrm{T}} \bar{\mathbf{Q}}_t \, \mathcal{C}_x + \mathcal{C}_y^{\mathrm{T}} \bar{\mathbf{Q}}_t \, \mathcal{C}_y + \mathcal{C}_\tau^{\mathrm{T}} \bar{\mathbf{Q}}_r \, \mathcal{C}_\tau \, , \tag{5.31a}$$

$$\mathbf{F}^{\mathrm{T}} = \left(\hat{\mathbf{A}}_t \, \mathbf{x}_{x0} - \mathbf{X}_{xr}\right)^{\mathrm{T}} \bar{\mathbf{Q}}_t \, \mathcal{C}_x + \left(\hat{\mathbf{A}}_t \, \mathbf{x}_{y0} - \mathbf{X}_{yr}\right)^{\mathrm{T}} \bar{\mathbf{Q}}_t \, \mathcal{C}_y + \left(\hat{\mathbf{A}}_r \, \mathbf{x}_{\theta 0} - \mathbf{X}_{\theta r}\right)^{\mathrm{T}} \bar{\mathbf{Q}}_r \, \mathcal{C}_\tau \, , \tag{5.31b}$$

where $\hat{\mathbf{A}}_t$, $\hat{\mathbf{A}}_r$, $\mathcal{C}_x$, $\mathcal{C}_y$, $\mathcal{C}_\tau$, $\bar{\mathbf{Q}}_t$, and $\bar{\mathbf{Q}}_r$ have the same structure as $\hat{\mathbf{A}}$, $\mathcal{C}$, and $\bar{\mathbf{Q}}$, but use the smaller matrices of the translational and rotational subsystems (proof in Appendix B.3).

Having defined a quadratic program as well as a method for efficient calculation of its matrices, we now need a tool for solving it. To solve the quadratic program, I have implemented the accelerated gradient projection method, described in [18]. This method has proven to converge to the optimal solution fast, and is easily implemented for quadratic programs with box constraints. The method iteratively searches for the optimal value of $\mathbf{P}$. Let $\mathbf{P}_k$ be the guess at k^{th} iteration. The method then defines a 'momentum' for the next iteration as:

$$\mathbf{s}_{k+1} = \mathbf{P}_k + \beta_k \left(\mathbf{P}_k - \mathbf{P}_{k-1}\right) \, , \tag{5.32}$$

where β_k is a coefficient that should be equal to zero for $k = 0$ and gradually increase to one, *e.g.*,

$$\beta_k = \begin{cases} 0 \, , & k = 0 \, , \\ \frac{k-1}{k+2} \, , & k > 0 \, . \end{cases} \tag{5.33}$$

Then, the next value of $\mathbf{P}$ is given by:

$$\mathbf{P}_{k+1} = \max\left(\min\left(\mathbf{s}_{k+1} - \lambda\left(\mathbf{H}\, \mathbf{s}_{k+1} + \mathbf{F}\right), 1700\right), 0\right) \, , \tag{5.34}$$

where λ is the step size.

So far, we tried to find an optimal sequence of amplitudes for a given sequence of $\boldsymbol{\nu}(t)$, *i.e.*, the 'pseudo-forces' and 'pseudo-torques.' Now, we aim to find the optimal sequence of $\boldsymbol{\nu}(t)$. Similarly to the LQR, we use a lookup table, which reduces the problem to finding an optimal array of lookup table indices. Unlike the LQR, a brute force approach is not viable since the number of all possible combinations grows exponentially with the length of the control horizon. To solve this problem, I employ a particle swarm optimization (PSO) algorithm, described in [19]. The algorithm is iterative, and works as follows: we have a population of particles $\mathbf{z}_1$, $\mathbf{z}_2$, $\ldots$, $\mathbf{z}_n$. Each particle represents a solution (in our case, a sequence of indices, which will define our $\boldsymbol{\nu}(t)$). In addition, the particles have velocities

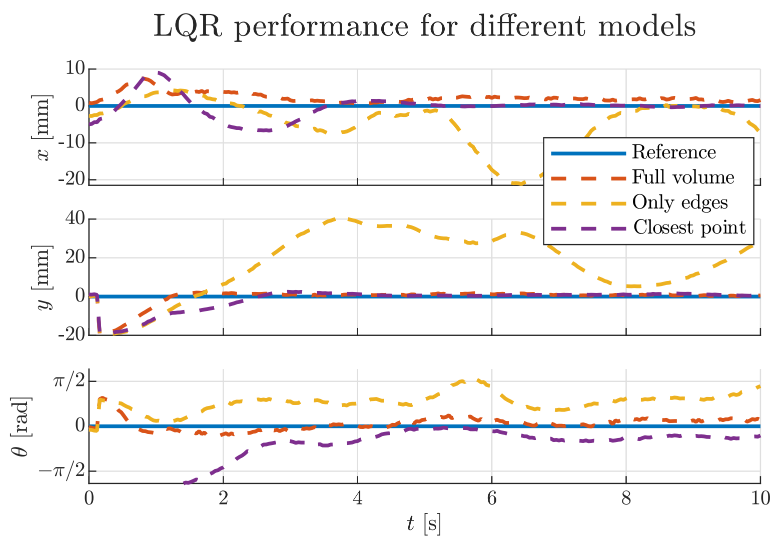


Figure 5.3: Steady-state behavior of the LQR controller after four iterations

$\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ — vectors of the same dimension as the particles. Both the solutions and the velocities are initialized randomly. In every iteration, the particles are updated using the following formula:

$$\mathbf{v}_i^{(k+1)} = w_k\,\mathbf{v}_i^{(k)} + \mathcal{U}(0,2)\left(\mathbf{b}_i - \mathbf{z}_i^{(k)}\right) + \mathcal{U}(0,2)\left(\mathbf{b}_g - \mathbf{z}_i^{(k)}\right), \tag{5.35a}$$

$$\mathbf{z}_i^{(k+1)} = \mathbf{z}_i^{(k)} + \mathbf{v}_i^{(k)}, \tag{5.35b}$$

where w_k is the 'inertia weight,' $\mathcal{U}(0,2)$ is a random uniformly distributed number between 0 and 2, $\mathbf{b}_i$ is the best recorded solution (solution that results in the smallest value of the criterion J) for the i^{th} particle, and $\mathbf{b}_g$ is the globally best solution. The value of w_k should initially be relatively high (*e.g.*, 0.9) and gradually decrease (*e.g.*, to 0.4). Note that since the solution should be positive integers, the values of $\mathbf{z}_i$ are rounded. Also, we need to use modulo arithmetics so that the values of $\mathbf{z}_i$ remain between 1 and m (the number of entries in the lookup table).

After a fixed number of iterations, we take $\mathbf{b}_g$ as the optimal sequence of high-pressure points, together with the corresponding sequence of amplitudes. However, we only use the first point of this sequence, since the whole MPC algorithm is repeated every control period. Consequently, solving for $\boldsymbol{\nu}(t)$ using PSO requires solving n quadratic programs (one for each particle) every iteration. In the current setting, PSO performs seven iterations with five particles, which results in solving 35 quadratic programs every control period.

5.3 Identification and parameter tuning

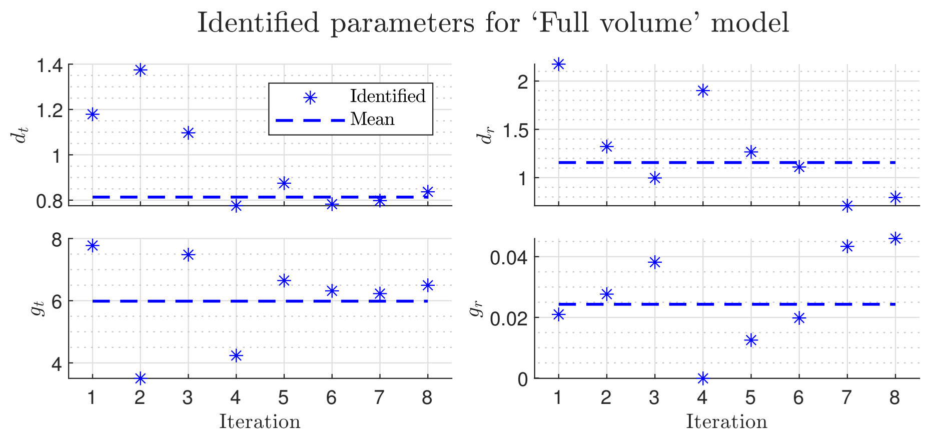


Figure 5.4: Results of iterative model identification

5.3.1 Model identification

This process is similar to the one described in Section 4.3. We begin with an initial guess of model parameters and iteratively refine them.

In addition, we also need to determine, which model of forces and torques is most suitable. To do so, we compare the performance of the control systems when stabilizing the triangular object. This comparison is shown in Figure 5.3. The control system based on the 'Only edges' model defined in (5.6) fails to stabilize the object. The other two control systems manage to stabilize the object, but the control system based on the 'Closest point' model defined in (5.7) has a slower settling time as well as greater oscillations, especially in the orientation θ. Therefore, it is reasonable to further consider only the 'Full volume' model defined in (5.5).

The results of iterative model identification are shown in Figure 5.4. The final values of the model parameters are:

$$d_t = 0.814\,, \qquad d_r = 1.16\,, \qquad g_t = 5.99\,, \qquad g_r = 0.0243\,. \tag{5.36}$$

5.3.2 Parameter tuning

The weight matrices $\mathbf{Q}$ and $\mathbf{R}$ are tuned similarly as in the previous chapter. The resulting closed-loop system should be as fast as possible without overshooting, and the required amplitude of pressure should not be saturated (*i.e.*, should be less than 1700 pascals) if the error in position is less than 10 mm. For orientation, the threshold is 5 degrees.

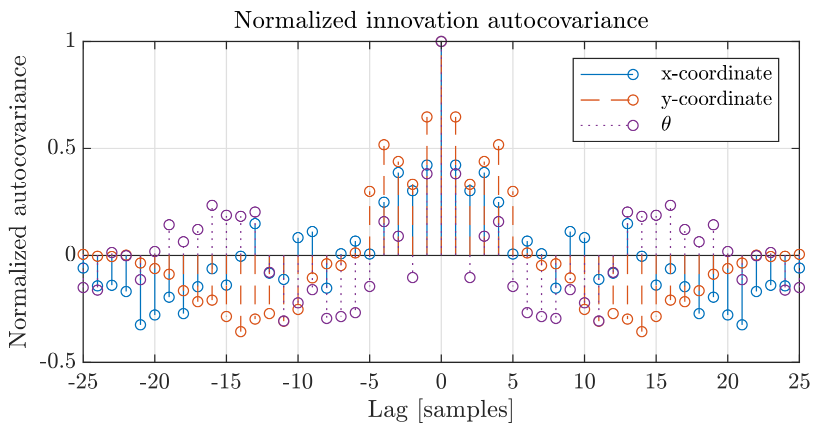


Figure 5.5: Normalized autocovariance of innovation sequences

The resulting LQR weights are:

$$\mathbf{Q}_t = \begin{bmatrix} 1 & 0 \\ 0 & 10 \end{bmatrix}, \qquad \mathbf{R}_t = 40\,, \qquad \mathbf{Q}_r = \begin{bmatrix} 1 & 0 \\ 0 & 10 \end{bmatrix}, \qquad \mathbf{R}_r = 8 \times 10^{-5}\,, \tag{5.37}$$

and the resulting MPC weights are:

$$\mathbf{Q}_t = \begin{bmatrix} 1 & 0 \\ 0 & 10 \end{bmatrix}, \qquad \mathbf{Q}_r = 100 \begin{bmatrix} 1 & 0 \\ 0 & 10 \end{bmatrix}, \qquad \mathbf{R} = 1\,. \tag{5.38}$$

The control horizon length is five samples, and the prediction horizon length is ten samples.

The noise characteristics for the Kalman filter are also obtained similarly as in the previous chapter. The covariance of the measured position, $\mathbf{V}_t$, and the measured orientation, $\mathbf{V}_r$, is:

$$\mathbf{V}_t = 0.873\,, \qquad \mathbf{V}_r = 0.001\,, \tag{5.39}$$

and the covariance matrices of the translational and rotational process noise are:

$$\mathbf{Q}_t = \begin{bmatrix} 0.933 & 0.0187 \\ 0.0187 & 0.965 \end{bmatrix}, \qquad \mathbf{Q}_r = \begin{bmatrix} 5.09 & 0.102 \\ 0.102 & 5.25 \end{bmatrix} \times 10^{-3}\,. \tag{5.40}$$

The normalized innovation autocovariance sequences are shown in Figure 5.5. These sequences are considerably higher than the ones of spherical objects shown in Figure 4.5, which is probably caused by some unmodeled coupling between the translational and rotational dynamics.

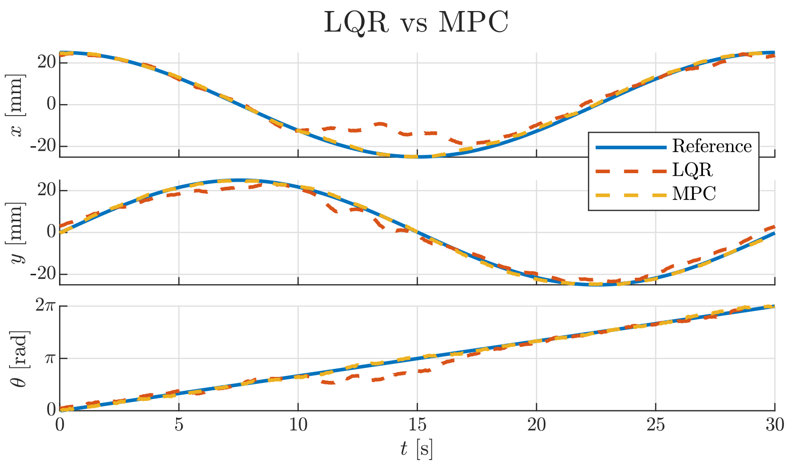


Figure 5.6: Comparison between LQR and MPC controllers

5.4 Results

To assess the performance of the control systems, I have designed a testing trajectory, where the object rotates and orbits around a given center. The reference position, x_{ri} and y_{ri}, and the reference orientation, θ_{ri}, of the i^{th} object are given by:

$$x_{ri}(t) = x_{ci} + r_i \cos(\vartheta_i(t)) \ , \quad y_{ri}(t) = y_{ci} + r_i \sin(\vartheta_i(t)) \ , \quad \vartheta_i(t) = \frac{2\pi}{T_o}\left(t + \frac{i}{n}T_o\right) \ , \tag{5.41a}$$

$$\theta_{ri}(t) = \omega\left(t + \frac{i}{n}T_r\right) \ , \qquad \omega = \frac{2\pi}{T_r} \ , \tag{5.41b}$$

where (x_{ci}, y_{ci}) are the coordinates of the center of orbit, r_i is the orbital radius, T_o is the orbital period, and T_r is the period of rotation.

The comparison between the LQR and MPC controllers when tracking this ‘orbit’ trajectory with $x_c = y_c = 0$, $r = 25$, and $T_o = T_r = 30$ is shown in Figure 5.6. Clearly, MPC is more robust, allowing it to track the reference more precisely. Footage of the MPC with two triangular objects is featured in the supplementary video (see Appendix A.2).

6 | Assembling planar objects

In the previous chapter, I have presented a method for manipulation of planar objects. In this chapter, I describe an algorithm that can assemble these objects into a prescribed formation and present the results.

6.1 Assembly algorithm

To assemble the objects into a formation, we need a higher-level algorithm that outputs reference positions for individual objects based on their current and goal positions. The proposed algorithm is sequential — it brings the objects into goal positions one by one based on a predetermined priority.

The priority of the objects is determined by their goal position in the formation. If the object is on the 'outside' of the formation, *i.e.*, if it has at least one edge that does not touch any other object, it has a low priority. If the object is on the 'inside' of the formation, *i.e.*, if all of its edges touch other objects, it has a high priority. The actual order among the low- and high-priority objects is arbitrary, as long as the high-priority objects precede the low-priority objects when being assembled. In the example formation shown in Figure 6.1, the red triangle with goal position G_1 is on the inside, and so it has the highest priority over the green triangles with goal positions G_2, G_3, and G_4.

Objects with lower priority are initially guided to a 'waiting' position. The waiting position is defined as the goal position offset by a given distance. The direction of this offset is given by a vector, which is perpendicular to the edge that touches another object, or a sum of these vectors if multiple edges touch other objects. In the example formation shown in Figure 6.1, the waiting positions are denoted W_2, W_3, and W_4, and the corresponding perpendicular vectors are denoted $\mathbf{r}_2$, $\mathbf{r}_3$, and $\mathbf{r}_4$.

During the operation, the objects are guided to their waiting, or goal positions based on two simple rules:

1. In the beginning, only the object with the highest priority is guided to its goal position. All other objects are guided to their waiting positions.
2. Guidance of an object is switched from waiting to goal position only if the object has reached its waiting position, and all higher-priority objects have reached their goal positions.

These two rules ensure that the objects are assembled sequentially. In the example formation

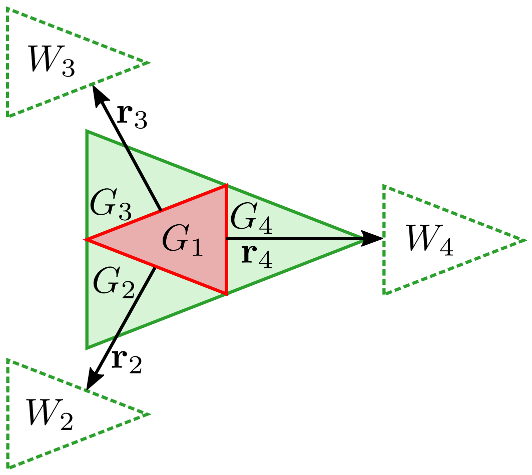


Figure 6.1: An example of a formation of triangular objects. The filled triangles represent the goal positions, the dotted-lined triangles represent the waiting positions.

shown in Figure 6.1, these rules mean that the object with goal position G_1 is brought to its goal first, followed by G_2, then G_3, and finally G_4.

Having defined the goal and waiting positions, we now need a planning algorithm that guides the objects to these positions. In fact, we need two distinct algorithms for guidance to the waiting and goal position, because these two actions have contradictory requirements. When the object is being guided to its waiting position, it needs to avoid contact with other objects, as the planar objects tend to adhere to each other due to surface tension. On the contrary, objects guided to goal position need to come in contact with other objects in the formation.

To guide the objects to their waiting positions, I employ a potential field planning algorithm, described in [20]. The guided object behaves as if it was exposed to an attractive force towards its waiting position, and to repulsive forces from other objects. The reference velocities of the i^{th} object, $v_{x_{ri}}$ and $v_{y_{ri}}$, are:

$$v_{x_{ri}}(t) = c_v\, v_{x_{ri}}(t-1) + c_a\, (x_{wi} - x_i(t)) + c_r \sum_{j\in\{1,\dots,n\}\setminus\{i\}} \frac{x_i(t) - x_j(t)}{d_{ij}^3(t)}\,, \tag{6.1a}$$

$$v_{y_{ri}}(t) = c_v\, v_{y_{ri}}(t-1) + c_a\, (y_{wi} - y_i(t)) + c_r \sum_{j\in\{1,\dots,n\}\setminus\{i\}} \frac{y_i(t) - y_j(t)}{d_{ij}^3(t)}\,, \tag{6.1b}$$

where $(x_{wi},\, y_{wi})$ are the coordinates of the waiting position, c_v is the coefficient of retained velocity — it should be a positive number less than one, c_a is the coefficient of attractive force, c_r is the coefficient of repulsive force, and d_{ij} is the mutual distance between the i^{th} and j^{th} object. The reference position of the i^{th} object, $(x_{ri},\, y_{ri})$, is then given by:

$$x_{ri}(t) = x_i(t) + v_{x_{ri}}(t)\,, \qquad y_{ri}(t) = y_i(t) + v_{y_{ri}}(t)\,. \tag{6.2}$$

To guide the objects to their goal positions, I employ a simpler potential field planner without the repulsive forces. The reference velocities are then given by:

$$v_{x_{ri}}(t) = c_v\, v_{x_{ri}}(t-1) + c_a\, (x_{gi} - x_i(t)) \,, \tag{6.3a}$$

$$v_{y_{ri}}(t) = c_v\, v_{y_{ri}}(t-1) + c_a\, (y_{gi} - y_i(t)) \,, \tag{6.3b}$$

where (x_{wi}, y_{wi}) are the coordinates of the goal position. The reference position is then given by the same formula as in (6.2).

6.2 Results

The results of the assembly algorithm can be best seen in the supplementary video (see Appendix A.2). Selected frames from the video demonstating the assembly of four objects are shown in Figure 6.3. The measured positions are plotted in Figure 6.2. As you can see, all four objects are brought to their goal positions within sixteen seconds.

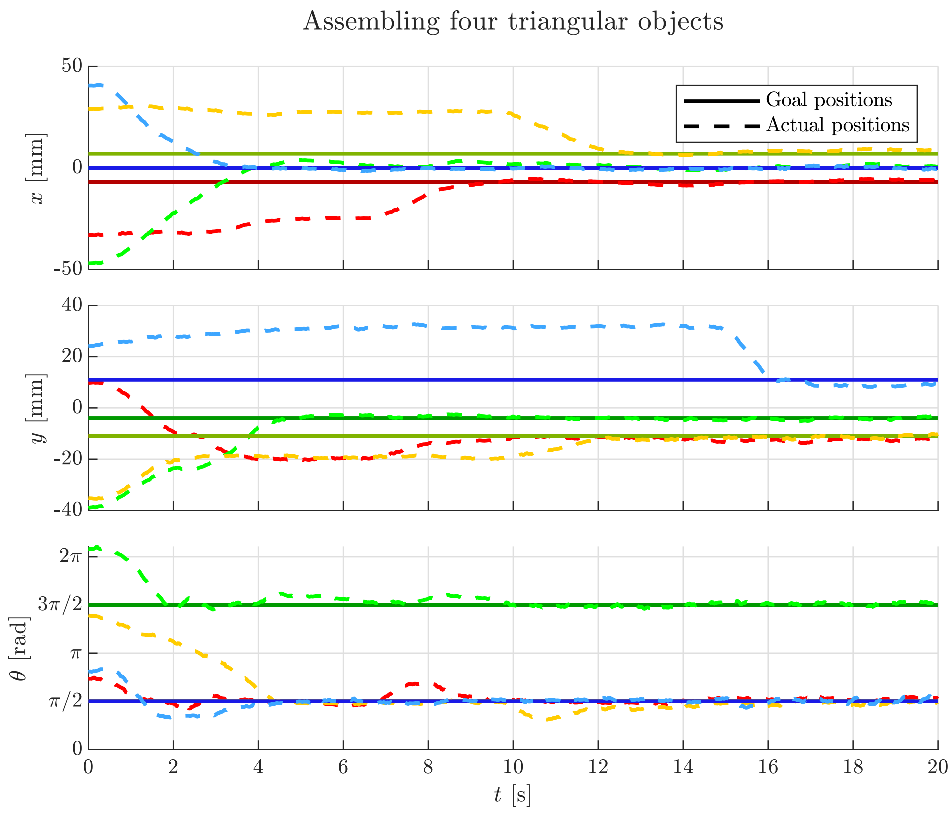
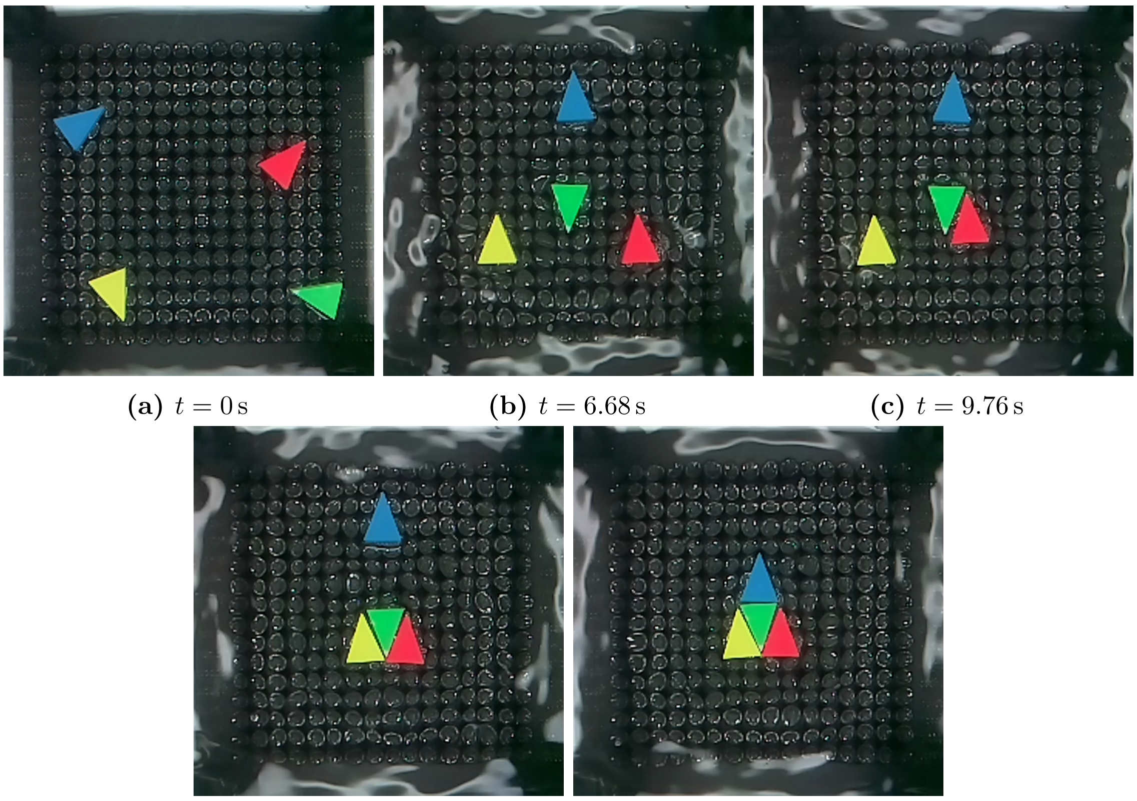


Figure 6.2: Results of assembling four triangular objects. The full lines represent the goal positions, the dashed lines represent the measured positions. The assigned priorities (from highest to lowest) are: green, red, yellow, blue.



(a) $t = 0\,\mathrm{s}$ (b) $t = 6.68\,\mathrm{s}$ (c) $t = 9.76\,\mathrm{s}$

(d) $t = 14.44\,\mathrm{s}$ (e) $t = 16.84\,\mathrm{s}$

Figure 6.3: Selected frames from the supplementary video showing the process of sequential assembly

7 | Conclusion

The goal of this thesis was to develop a platform for multi-object manipulation by focusing the acoustic pressure field. The design of this platform is based on the previous version developed as a part of my Bachelor's thesis. The previous version could manipulate only one spherical object at a time on a 40×40 mm manipulation area, reaching speeds of up to $10\,\mathrm{mm\,s^{-1}}$ [1]. The newly developed platform uses four times more transducers than the previous version (a 16-by-16 array compared to an 8-by-8 array), allowing it to manipulate multiple objects at once on a 100×100 mm manipulation area (as can be seen when tracking the 'Double link' trajectory), reaching speeds of up to $20\,\mathrm{mm\,s^{-1}}$ (see Section 4.4). Thus, by increasing the number of transducers four times, I have increased the manipulation area more than six times; and by implementing a better-tuned controller, I have increased the maximum speed twice.

Apart from increasing the active manipulation area, the increased number of transducers also makes solving the optimization problem of focusing the acoustic filed computationally more demanding. As a result, the original BFGS solver could not solve the problem within the control period. To deal with this problem, I have implemented the LM algorithm with the kernel trick, which is five to nine times faster than the previously used BFGS (see Figure 3.4).

The larger manipulation area also allows for the manipulation of more generally shaped, planar objects. I have developed a mathematical model of acoustophoretic forces and torques acting on these objects. It turns out that manipulating the planar objects is more complicated than spherical objects, and simple state feedback is insufficient. Therefore, I have decided to use model predictive control with my own tailored implementation of an efficient solver. I have also designed an algorithm for assembling the planar objects into predefined formations. I have experimentally verified that the platform can assemble up to four objects at once (see Section 6.2 or the supplementary video linked in Appendix A.2).

7.1 Future improvements

There are still some unexplored possibilities regarding the planar objects. Although I have discarded the gradient control because it performed worse on the spherical objects, it may be worthwhile to explore the gradient control of planar objects. Also, it would be interesting to try if the developed methods for controlling triangular objects work for other shapes as well. Unfortunately, I was unable to manufacture other shapes in the later stages of my master thesis due to the COVID-19 situation.

Appendices

A | Online attachments

A.1 GitLab repository

The AcouMan project is located at `https://gitlab.fel.cvut.cz/aa4cc/acouman`. The project contains the following relevant repositories:

- **AcouMan main:** Main repository. Contains a documentation and submodules with other repositories.
- **simulink-controller:** Contains the control systems for spherical and triangular objects implemented in Simulink.
- **experiments:** Contains the data as well as the code to replicate the results shown in this thesis.
- **Master Thesis:** Contains source files for this thesis.
- **matlab:** Contains Matlab scripts for calibrating the platform, optimization, and visualization of the acoustic field.
- **manufacturing:** Three repositories containing the drawings, printed circuit board (PCB) designs, and 3D models needed for building the platform.

A.2 Supplementary video

The supplementary video is available at `https://youtu.be/D5RClzG8gOU`. The video is split into parts showing manipulation of spherical and triangular objects (see the video description on YouTube).

B | Formulas and derivations

B.1 Deriving the formula for total acoustic pressure

This section and the following one present derivations of efficient formulas for calculating the modulus of acoustic pressure and its spatial derivatives, mentioned in Chapter 3.

Let us begin with the modulus of acoustic pressure, which is denoted as $|P(x, y, z, \mathbf{\Phi})|^2$. Combining (3.1) and (3.2), the squared amplitude of the total pressure can be expressed as:

$$|P(x, y, z, \mathbf{\Phi})|^2 = \left(\sum_{i=1}^{N} M^{(i)}(x, y, z)\, \mathrm{e}^{\mathrm{j}\varphi_i}\right) \left(\sum_{i=1}^{N} \bar{M}^{(i)}(x, y, z)\, \mathrm{e}^{-\mathrm{j}\varphi_i}\right) , \tag{B.1}$$

where $\bar{M}$ denotes the complex conjugate.

Let us define vectors $\mathbf{u} = \left[\mathrm{e}^{\mathrm{j}\varphi_1}, \ldots, \mathrm{e}^{\mathrm{j}\varphi_N}\right]^{\mathrm{T}}$ and $\mathbf{m} = \left[M^{(1)}(x, y, z), \ldots, M^{(N)}(x, y, z)\right]^{\mathrm{T}}$. Now, the bracketed terms in (B.1) can be expressed and rearranged into:

$$|P(x, y, z, \mathbf{\Phi})|^2 = \left(\mathbf{m}^{\mathrm{T}}\, \mathbf{u}\right) \left(\mathbf{u}^{\dagger}\, \bar{\mathbf{m}}\right) = \mathbf{u}^{\dagger}\, \bar{\mathbf{m}}\, \mathbf{m}^{\mathrm{T}}\, \mathbf{u}\, , \tag{B.2}$$

where $\mathbf{u}^{\dagger}$ is the conjugate transpose of $\mathbf{u}$, and $\bar{\mathbf{m}}$ is the vector of complex conjugated elements of $\mathbf{m}$.

While the formula in (B.2) is mathematically correct, its numerical implementation involves multiplication of complex vectors, where the imaginary parts cancel out. Therefore, this formula is not computationally efficient.

Let us therefore further define a matrix $\mathbf{p} = [\mathbf{m}_{\mathrm{r}}, \mathbf{m}_{\mathrm{i}}]$ composed of vectors of real ($\mathbf{m}_{\mathrm{r}}$) and imaginary ($\mathbf{m}_{\mathrm{i}}$) parts of $\mathbf{m}$. Now, the product $\bar{\mathbf{m}}\, \mathbf{m}^{\mathrm{T}}$ can be expressed as:

$$\bar{\mathbf{m}}\, \mathbf{m}^{\mathrm{T}} = \mathbf{p}\, \mathbf{p}^{\mathrm{T}} + \mathrm{j}\, \mathbf{p} \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \mathbf{p}^{\mathrm{T}} , \tag{B.3}$$

while the vector $\mathbf{u}$ can be expressed as:

$$\mathbf{u} = \mathbf{c}_{\mathbf{\Phi}} + \mathrm{j}\, \mathbf{s}_{\mathbf{\Phi}}\, , \tag{B.4}$$

where $\mathbf{c}_{\mathbf{\Phi}}$ and $\mathbf{s}_{\mathbf{\Phi}}$ are vectors of cosines and sines of elements from $\mathbf{\Phi}$, respecitvely. Substituting

(B.3) and (B.4) into (B.2), we obtain:

$$\begin{aligned}|P(x,y,z,\mathbf{\Phi})|^2 &= \left(\mathbf{c}_{\mathbf{\Phi}}^{\mathrm{T}} - \mathrm{j}\,\mathbf{s}_{\mathbf{\Phi}}^{\mathrm{T}}\right)\left(\mathbf{p}\,\mathbf{p}^{\mathrm{T}} + \mathrm{j}\,\mathbf{p}\begin{bmatrix}0 & -1\\ 1 & 0\end{bmatrix}\mathbf{p}^{\mathrm{T}}\right)(\mathbf{c}_{\mathbf{\Phi}} + \mathrm{j}\,\mathbf{s}_{\mathbf{\Phi}}) =\\ &= \mathbf{c}_{\mathbf{\Phi}}^{\mathrm{T}}\,\mathbf{p}\,\mathbf{p}^{\mathrm{T}}\,\mathbf{c}_{\mathbf{\Phi}} + \mathbf{s}_{\mathbf{\Phi}}^{\mathrm{T}}\,\mathbf{p}\,\mathbf{p}^{\mathrm{T}}\,\mathbf{s}_{\mathbf{\Phi}} + \mathbf{c}_{\mathbf{\Phi}}^{\mathrm{T}}\,\mathbf{p}\begin{bmatrix}0 & -2\\ 2 & 0\end{bmatrix}\mathbf{p}^{\mathrm{T}}\,\mathbf{s}_{\mathbf{\Phi}}\,.\end{aligned} \tag{B.5}$$

The solver presented in Chapter 3 also requires the gradient of the absolute pressure. Differentiating (B.5) with respect to $\mathbf{\Phi}$ yields:

$$\begin{aligned}\nabla_{\mathbf{\Phi}}|P(x,y,z,\mathbf{\Phi})|^2 &= -2\,\mathrm{diag}\left(\mathbf{s}_{\mathbf{\Phi}}\right)\,\mathbf{p}\,\mathbf{p}^{\mathrm{T}}\,\mathbf{c}_{\mathbf{\Phi}} + 2\,\mathrm{diag}\left(\mathbf{c}_{\mathbf{\Phi}}\right)\,\mathbf{p}\,\mathbf{p}^{\mathrm{T}}\,\mathbf{s}_{\mathbf{\Phi}}\\ &- \mathrm{diag}\left(\mathbf{s}_{\mathbf{\Phi}}\right)\,\mathbf{p}\begin{bmatrix}0 & -2\\ 2 & 0\end{bmatrix}\mathbf{p}^{\mathrm{T}}\,\mathbf{s}_{\mathbf{\Phi}} + \mathrm{diag}\left(\mathbf{c}_{\mathbf{\Phi}}\right)\,\mathbf{p}\begin{bmatrix}0 & -2\\ 2 & 0\end{bmatrix}\mathbf{p}^{\mathrm{T}}\,\mathbf{c}_{\mathbf{\Phi}}\,,\end{aligned} \tag{B.6}$$

where diag(.) is a diagonal matrix with elements from the given vector. Some terms can be grouped, resulting in the following formula:

$$\begin{aligned}\nabla_{\mathbf{\Phi}}|P(x,y,z,\mathbf{\Phi})|^2 =& 2\left(\mathrm{diag}\left(\mathbf{c}_{\mathbf{\Phi}}\right)\,\mathbf{p}\begin{bmatrix}0 & -1\\ 1 & 0\end{bmatrix} - \mathrm{diag}\left(\mathbf{s}_{\mathbf{\Phi}}\right)\,\mathbf{p}\right)\mathbf{p}^{\mathrm{T}}\,\mathbf{c}_{\mathbf{\Phi}}\\ &+ 2\left(\mathrm{diag}\left(\mathbf{c}_{\mathbf{\Phi}}\right)\,\mathbf{p} - \mathrm{diag}\left(\mathbf{s}_{\mathbf{\Phi}}\right)\,\mathbf{p}\begin{bmatrix}0 & -1\\ 1 & 0\end{bmatrix}\right)\mathbf{p}^{\mathrm{T}}\,\mathbf{s}_{\mathbf{\Phi}}\,.\end{aligned} \tag{B.7}$$

B.2 Calculating spatial derivatives of acoustic pressure

Let us define a vector $\mathbf{m}_x = \left[\frac{\partial\,M^{(1)}(x,y,z)}{\partial\,x}, \dots, \frac{\partial\,M^{(N)}(x,y,z)}{\partial\,x}\right]^{\mathrm{T}}$. By differentiating (B.2) with respect to x, we obtain:

$$\frac{\partial\,|P|^2}{\partial\,x} = \mathbf{u}^{\dagger}\,\bar{\mathbf{m}}_x\,\mathbf{m}^{\mathrm{T}}\,\mathbf{u} + \mathbf{u}^{\dagger}\,\bar{\mathbf{m}}\,\mathbf{m}_x^{\mathrm{T}}\,\mathbf{u}\,. \tag{B.8}$$

Notice that the two terms on the right-hand side of the equation above are complex conjugates of each other. Therefore, the spatial derivative is equal to:

$$\frac{\partial\,|P|^2}{\partial\,x} = 2\,\mathfrak{Re}\left\{\mathbf{u}^{\dagger}\,\bar{\mathbf{m}}_x\,\mathbf{m}^{\mathrm{T}}\,\mathbf{u}\right\}\,, \tag{B.9}$$

where $\mathfrak{Re}\,\{.\}$ is the real part of a complex number.

Similarly to the previous section, I will first derive all necessary formulas for calculating the spatial derivative, and then provide a formula for efficient calculation. Let us start with $\mathbf{m}_x$. By differentiating (3.3) with respect to x, we obtain:

$$\frac{\partial\,M^{(i)}(x,y,z)}{\partial\,x} = P_0\left[\frac{1}{d}\,\frac{\partial\,f_{dir}(x,y,z)}{\partial\,x} + f_{dir}(x,y,z)\left(-\frac{1}{d^2} + \frac{\mathrm{j}\,k}{d}\right)\frac{\partial\,d}{\partial\,x}\right]\mathrm{e}^{\mathrm{j}kd}\,. \tag{B.10}$$

Since the distance d is defined as $d = \sqrt{x^2+y^2+z^2}$, its spatial derivative is:

$$\frac{\partial\, d}{\partial\, x} = \frac{x}{\sqrt{x^2+y^2+z^2}} = \frac{x}{d}\,. \tag{B.11}$$

By applying chain rule to (3.4), the spatial derivative of directivity function can be expressed as:

$$\begin{aligned}\frac{\partial\, f_{dir}(x,y,z)}{\partial\, x} &= \frac{\partial\, f_{dir}(\sin\theta)}{\partial\,(\sin\theta)}\,\frac{\partial\,\sin\theta}{\partial\, x}\\ &= \left[\frac{\mathrm{J}_0(k\,r\,\sin\theta)}{\sin\theta} - \frac{2\mathrm{J}_1(k\,r\,\sin\theta)}{k\,r\,\sin^2\theta} - \frac{\mathrm{J}_2(k\,r\,\sin\theta)}{\sin\theta}\right]\frac{x\,z^2}{d^3\,\sqrt{x^2+y^2}}\,.\end{aligned} \tag{B.12}$$

Now, I will do a similar "trick" as in the previous section and define a matrix $\mathbf{p}_x$ composed of real and imaginary parts of $\mathbf{m}_x$. The term $\bar{\mathbf{m}}_x\,\mathbf{m}^{\mathrm{T}}$ can be now expressed as:

$$\bar{\mathbf{m}}_x\,\mathbf{m}^{\mathrm{T}} = \mathbf{p}_x\,\mathbf{p}^{\mathrm{T}} + \mathrm{j}\,\mathbf{p}_x \begin{bmatrix}0 & -1\\ 1 & 0\end{bmatrix}\mathbf{p}^{\mathrm{T}}\,, \tag{B.13}$$

and the spatial derivative from (B.9) can be expressed as:

$$\begin{aligned}\frac{\partial\,|P|^2}{\partial\, x} =& 2\,\Re\mathfrak{e}\left\{\left(\mathbf{c}_{\mathbf{\Phi}}^{\mathrm{T}} - \mathrm{j}\,\mathbf{s}_{\mathbf{\Phi}}^{\mathrm{T}}\right)\left(\mathbf{p}_x\,\mathbf{p}^{\mathrm{T}} + \mathrm{j}\,\mathbf{p}_x\begin{bmatrix}0 & -1\\ 1 & 0\end{bmatrix}\mathbf{p}^{\mathrm{T}}\right)\left(\mathbf{c}_{\mathbf{\Phi}} + \mathrm{j}\,\mathbf{s}_{\mathbf{\Phi}}\right)\right\} =\\ =& 2\left(\mathbf{c}_{\mathbf{\Phi}}^{\mathrm{T}}\,\mathbf{p}_x\,\mathbf{p}^{\mathrm{T}}\,\mathbf{c}_{\mathbf{\Phi}} + \mathbf{s}_{\mathbf{\Phi}}^{\mathrm{T}}\,\mathbf{p}_x\,\mathbf{p}^{\mathrm{T}}\,\mathbf{s}_{\mathbf{\Phi}} + \mathbf{c}_{\mathbf{\Phi}}^{\mathrm{T}}\,\mathbf{p}_x\begin{bmatrix}0 & -1\\ 1 & 0\end{bmatrix}\mathbf{p}^{\mathrm{T}}\,\mathbf{s}_{\mathbf{\Phi}}\right.\\ &\left. -\mathbf{s}_{\mathbf{\Phi}}^{\mathrm{T}}\,\mathbf{p}_x\begin{bmatrix}0 & -1\\ 1 & 0\end{bmatrix}\mathbf{p}^{\mathrm{T}}\,\mathbf{c}_{\mathbf{\Phi}}\right)\,.\end{aligned} \tag{B.14}$$

The gradient of the spatial derivative is:

$$\begin{aligned}\nabla_{\mathbf{\Phi}}\left(\frac{\partial\,|P|^2}{\partial\, x}\right) = 2\Bigg(&-\operatorname{diag}(\mathbf{s}_{\mathbf{\Phi}})\left(\mathbf{p}_x\,\mathbf{p}^{\mathrm{T}} + \mathbf{p}\,\mathbf{p}_x^{\mathrm{T}}\right)\mathbf{c}_{\mathbf{\Phi}}\\ &+\operatorname{diag}(\mathbf{c}_{\mathbf{\Phi}})\left(\mathbf{p}_x\,\mathbf{p}^{\mathrm{T}} + \mathbf{p}\,\mathbf{p}_x^{\mathrm{T}}\right)\mathbf{s}_{\mathbf{\Phi}} - \operatorname{diag}(\mathbf{s}_{\mathbf{\Phi}})\,\mathbf{p}_x\begin{bmatrix}0 & -1\\ 1 & 0\end{bmatrix}\mathbf{p}^{\mathrm{T}}\,\mathbf{s}_{\mathbf{\Phi}}\\ &+\operatorname{diag}(\mathbf{c}_{\mathbf{\Phi}})\,\mathbf{p}\begin{bmatrix}0 & 1\\ -1 & 0\end{bmatrix}\mathbf{p}_x^{\mathrm{T}}\,\mathbf{c}_{\mathbf{\Phi}} - \operatorname{diag}(\mathbf{c}_{\mathbf{\Phi}})\,\mathbf{p}_x\begin{bmatrix}0 & -1\\ 1 & 0\end{bmatrix}\mathbf{p}^{\mathrm{T}}\,\mathbf{c}_{\mathbf{\Phi}}\\ &+\operatorname{diag}(\mathbf{s}_{\mathbf{\Phi}})\,\mathbf{p}\begin{bmatrix}0 & 1\\ -1 & 0\end{bmatrix}\mathbf{p}_x^{\mathrm{T}}\,\mathbf{s}_{\mathbf{\Phi}}\Bigg)\end{aligned} \tag{B.15}$$

Due to axial symmetry along the z-axis, the spatial derivative with respect to y is almost identical to the spatial derivative with respect to x.

B.3 Proving the MPC matrix identity

Recalling (5.29), we aim to simplify the calculation of matrices **H** and **F**, defined as

$$\mathbf{H} = \mathcal{C}^{\mathrm{T}}\,\bar{\mathbf{Q}}\,\mathcal{C} + \bar{\mathbf{R}}\,, \qquad \mathbf{F}^{\mathrm{T}} = \left(\hat{\mathbf{A}}\,\mathbf{x}_0 - \mathbf{X}_r\right)^{\mathrm{T}}\bar{\mathbf{Q}}\,\mathcal{C}\,. \tag{B.16}$$

Due to the structure of $\bar{\mathbf{A}}$, $\bar{\mathbf{B}}$, and $\boldsymbol{\nu}$ defined in (5.19), the following identities hold true:

$$\bar{\mathbf{A}}^n = \begin{bmatrix} \bar{\mathbf{A}}_t^n & & \\ & \bar{\mathbf{A}}_t^n & \\ & & \bar{\mathbf{A}}_r^n \end{bmatrix}, \qquad \bar{\mathbf{A}}^n\,\bar{\mathbf{B}}\,\boldsymbol{\nu} = \begin{bmatrix} \bar{\mathbf{A}}_t^n\,\bar{\mathbf{B}}_t\,f_x \\ \bar{\mathbf{A}}_t^n\,\bar{\mathbf{B}}_t\,f_y \\ \bar{\mathbf{A}}_r^n\,\bar{\mathbf{B}}_r\,\tau \end{bmatrix}. \tag{B.17}$$

By direct calculation, the element of **H** on i^{th} row and j^{th} column, $\mathbf{H}_{ij}$ is:

$$\begin{aligned} \mathbf{H}_{ij} &= \left[\sum_{k=max(i,j)}^{h_p} \boldsymbol{\nu}^{\mathrm{T}}(t+i-1)\,\bar{\mathbf{B}}^{\mathrm{T}}\left(\bar{\mathbf{A}}^{k-i}\right)^{\mathrm{T}}\mathbf{Q}\,\bar{\mathbf{A}}^{k-j}\,\bar{\mathbf{B}}\,\boldsymbol{\nu}(t+j-1)\right] + \delta_{ij}\,\mathbf{R}\,, \\ &i,\, j = 1,\, 2,\, \ldots,\, h_c\,, \end{aligned} \tag{B.18}$$

where δ_{ij} is the Kronecker delta operator. Substituting the identities from (B.17), $\mathbf{H}_{ij}$ can be expressed using three sums:

$$\begin{aligned} \mathbf{H}_{ij} = &\left[\sum_{k=max(i,j)}^{h_p} f_x(t+i-1)\,\bar{\mathbf{B}}_t^{\mathrm{T}}\left(\bar{\mathbf{A}}_t^{k-i}\right)^{\mathrm{T}}\mathbf{Q}_t\,\bar{\mathbf{A}}_t^{k-j}\,\bar{\mathbf{B}}_t\,f_x(t+j-1)\right] \\ &+ \left[\sum_{k=max(i,j)}^{h_p} f_y(t+i-1)\,\bar{\mathbf{B}}_t^{\mathrm{T}}\left(\bar{\mathbf{A}}_t^{k-i}\right)^{\mathrm{T}}\mathbf{Q}_t\,\bar{\mathbf{A}}_t^{k-j}\,\bar{\mathbf{B}}_t\,f_y(t+j-1)\right] \\ &+ \left[\sum_{k=max(i,j)}^{h_p} \tau(t+i-1)\,\bar{\mathbf{B}}_r^{\mathrm{T}}\left(\bar{\mathbf{A}}_r^{k-i}\right)^{\mathrm{T}}\mathbf{Q}_r\,\bar{\mathbf{A}}_r^{k-j}\,\bar{\mathbf{B}}_r\,\tau(t+j-1)\right] + \delta_{ij}\,\mathbf{R}\,. \end{aligned} \tag{B.19}$$

Therefore, the matrix **H** can be expressed as:

$$\mathbf{H} = \mathcal{C}_x^{\mathrm{T}}\,\bar{\mathbf{Q}}_t\,\mathcal{C}_x + \mathcal{C}_y^{\mathrm{T}}\,\bar{\mathbf{Q}}_t\,\mathcal{C}_y + \mathcal{C}_\tau^{\mathrm{T}}\,\bar{\mathbf{Q}}_r\,\mathcal{C}_\tau\,, \tag{B.20}$$

where

$$\mathcal{C}_x = \begin{bmatrix} \bar{\mathbf{B}}_t\,f_x(t) & & & \\ \bar{\mathbf{A}}_t\,\bar{\mathbf{B}}_t\,f_x(t) & \bar{\mathbf{B}}_t\,f_x(t+1) & & \\ \vdots & \vdots & \ddots & \\ \bar{\mathbf{A}}_t^{h_c-1}\,\bar{\mathbf{B}}_t\,f_x(t) & \bar{\mathbf{A}}_t^{h_c-2}\,\bar{\mathbf{B}}_t\,f_x(t+1) & \cdots & \bar{\mathbf{B}}_t\,f_x(t+h_c-1) \\ \bar{\mathbf{A}}_t^{h_c}\,\bar{\mathbf{B}}_t\,f_x(t) & \bar{\mathbf{A}}_t^{h_c-1}\,\bar{\mathbf{B}}_t\,f_x(t+1) & \cdots & \bar{\mathbf{A}}_t\,\bar{\mathbf{B}}_t\,f_x(t+h_c-1) \\ \vdots & \vdots & & \vdots \\ \bar{\mathbf{A}}_t^{h_p-1}\,\bar{\mathbf{B}}_t\,f_x(t) & \bar{\mathbf{A}}_t^{h_p-2}\,\bar{\mathbf{B}}_t\,f_x(t+1) & \cdots & \bar{\mathbf{A}}_t^{h_p-h_c}\,\bar{\mathbf{B}}_t\,f_x(t+h_c-1) \end{bmatrix}, \tag{B.21}$$

$$\bar{\mathbf{Q}}_t = \begin{bmatrix} \mathbf{Q}_t & & \\ & \ddots & \\ & & \mathbf{Q}_t \end{bmatrix}, \qquad \bar{\mathbf{Q}}_r = \begin{bmatrix} \mathbf{Q}_r & & \\ & \ddots & \\ & & \mathbf{Q}_r \end{bmatrix}, \tag{B.22}$$

and the matrices $\mathcal{C}_y$ and $\mathcal{C}_\tau$ are defined analogously using the corresponding state-space matrices and inputs.

Similarly, the i^{th} element of the vector $\mathbf{F}$ is defined as:

$$\mathbf{F}_i = \sum_{k=i}^{h_p} \left(\bar{\mathbf{A}}^k \, \mathbf{x}_0 - \mathbf{x}_r(t+k)\right)^{\mathrm{T}} \mathbf{Q} \, \bar{\mathbf{A}}^{k-j} \, \bar{\mathbf{B}} \, \boldsymbol{\nu}(t+i-1) \,, \tag{B.23}$$

which can be expressed using three sums:

$$\begin{aligned} \mathbf{F}_i = & \left[\sum_{k=i}^{h_p} \left(\bar{\mathbf{A}}_t^k \, \mathbf{x}_{x0} - \mathbf{x}_{xr}(t+k)\right)^{\mathrm{T}} \mathbf{Q}_t \, \bar{\mathbf{A}}_t^{k-j} \, \bar{\mathbf{B}}_t \, f_x(t+i-1)\right] \\ & + \left[\sum_{k=i}^{h_p} \left(\bar{\mathbf{A}}_t^k \, \mathbf{x}_{y0} - \mathbf{x}_{yr}(t+k)\right)^{\mathrm{T}} \mathbf{Q}_t \, \bar{\mathbf{A}}_t^{k-j} \, \bar{\mathbf{B}}_t \, f_y(t+i-1)\right] \\ & + \left[\sum_{k=i}^{h_p} \left(\bar{\mathbf{A}}_r^k \, \mathbf{x}_{\theta 0} - \mathbf{x}_{\theta r}(t+k)\right)^{\mathrm{T}} \mathbf{Q}_r \, \bar{\mathbf{A}}_r^{k-j} \, \bar{\mathbf{B}}_r \, \tau(t+i-1)\right] . \end{aligned} \tag{B.24}$$

Therefore, the vector $\mathbf{F}$ can be expressed as:

$$\mathbf{F}^{\mathrm{T}} = \left(\hat{\mathbf{A}}_t \, \mathbf{x}_{x0} - \mathbf{X}_{xr}\right)^{\mathrm{T}} \bar{\mathbf{Q}}_t \, \mathcal{C}_x + \left(\hat{\mathbf{A}}_t \, \mathbf{x}_{y0} - \mathbf{X}_{yr}\right)^{\mathrm{T}} \bar{\mathbf{Q}}_t \, \mathcal{C}_y + \left(\hat{\mathbf{A}}_r \, \mathbf{x}_{\theta 0} - \mathbf{X}_{\theta r}\right)^{\mathrm{T}} \bar{\mathbf{Q}}_r \, \mathcal{C}_\tau \,, \tag{B.25}$$

where

$$\hat{\mathbf{A}}_t = \begin{bmatrix} \bar{\mathbf{A}}_t \\ \bar{\mathbf{A}}_t^2 \\ \vdots \\ \bar{\mathbf{A}}_t^{h_p} \end{bmatrix}, \qquad \mathbf{x}_{x0} = \mathbf{x}_x(t) \,, \qquad \mathbf{X}_{xr} = \begin{bmatrix} \mathbf{x}_{xr}(t+1) \\ \vdots \\ \mathbf{x}_{xr}(t+h_p) \end{bmatrix}, \tag{B.26}$$

and $\hat{\mathbf{A}}_t$, $\mathbf{x}_{y0}$, $\mathbf{x}_{\theta 0}$, $\mathbf{X}_{yr}$, and $\mathbf{X}_{\theta r}$ are defined analogously.

C | References

[1] Matouš, J. Manipulation of objects on a surface of a liquid using an array of ultrasonic transducers. *Bachelor thesis*, 2018. Available from: http://hdl.handle.net/10467/76751

[2] Matouš, J.; Kollarčík, A.; et al. Optimization-based Feedback Manipulation Through an Array of Ultrasonic Transducers. *IFAC-PapersOnLine*, volume 52, no. 15, 2019: pp. 483 – 488, ISSN 2405-8963, doi:10.1016/j.ifacol.2019.11.722, 8th IFAC Symposium on Mechatronic Systems MECHATRONICS 2019.

[3] Andrade, M. A. B.; Pérez, N.; et al. Review of Progress in Acoustic Levitation. *Brazilian Journal of Physics*, Dec. 2017: pp. 1–24, ISSN 0103-9733, 1678-4448, doi:10.1007/s13538-017-0552-6.

[4] Ochiai, Y.; Hoshi, T.; et al. Pixie dust: graphics generated by levitated and animated objects in computational acoustic-potential field. *ACM Transactions on Graphics (TOG)*, volume 33, no. 4, 2014: p. 85.

[5] Courtney, C. R.; Demore, C. E.; et al. Independent trapping and manipulation of microparticles using dexterous acoustic tweezers. *Applied Physics Letters*, volume 104, no. 15, 2014: p. 154103.

[6] Courtney, C. R. P.; Drinkwater, B. W.; et al. Dexterous manipulation of microparticles using Bessel-function acoustic pressure fields. *Applied Physics Letters*, volume 102, no. 12, 2013: p. 123508.

[7] Marzo, A.; Seah, S. A.; et al. Holographic acoustic elements for manipulation of levitated objects. *Nature Communications*, volume 6, 2015: p. 8661.

[8] Marshall, M.; Carter, T.; et al. Ultra-tangibles: creating movable tangible objects on interactive tables. ACM Press, 2012, p. 2185.

[9] Marzo, A.; McGeehan, R.; et al. Ghost Touch: Turning Surfaces into Interactive Tangible Canvases with Focused Ultrasound. In *Proceedings of the 2015 International Conference on Interactive Tabletops & Surfaces*, ITS '15, ACM, 2015, pp. 137–140.

[10] Kinsler, L. E.; Frey, A. R.; et al. *Fundamentals of acoustics*. Wiley, fourth edition, 2000.

[11] Madsen, K.; Nielsen, H.; et al. Methods for Non-Linear Least Squares Problems (2nd ed.). 2004. Available from: http://www2.imm.dtu.dk/pubdb/views/edoc_download.php/3215/pdf/imm3215.pdf

[12] Boyd, S.; Vandenberghe, L. *Introduction to Applied Linear Algebra: Vectors, Matrices, and Least Squares*. Cambridge, UK ; New York, NY: Cambridge University Press, first edition, Aug. 2018, ISBN 978-1-316-51896-0.

[13] Dimarogonas, D. V.; Kyriakopoulos, K. J. Distributed cooperative control and collision avoidance for multiple kinematic agents. In *Proceedings of the 45th IEEE Conference on Decision and Control*, 2006, pp. 721–726.

[14] Franklin, G. F.; Powell, J. D.; et al. *Digital Control of Dynamic Systems*. 1200 Pilarcitos Ave., Half Moon Bay, CA: Ellis-Kagle Press, third edition, 1998, ISBN 978-0-9791226-0-6.

[15] MathWorks. *Control System Toolbox Reference*. 2020. Available from: https://www.mathworks.com/help/pdf_doc/control/control_ref.pdf

[16] Kailath, T. An innovations approach to least-squares estimation–Part I: Linear filtering in additive white noise. *IEEE Transactions on Automatic Control*, volume 13, no. 6, 1968: pp. 646–655.

[17] Wright, S. J. Efficient Convex Optimization for Linear MPC. In *Handbook of Model Predictive Control*, edited by S. V. Raković; W. S. Levine, Birkhäuser, 2019, ISBN 978-3-319-77489-3.

[18] Giselsson, P.; Doan, M. D.; et al. Accelerated gradient methods and dual decomposition in distributed model predictive control. *Automatica*, volume 49, no. 3, 2013: pp. 829 – 833, ISSN 0005-1098, doi:10.1016/j.automatica.2013.01.009.

[19] Poli, R.; Kennedy, J.; et al. Particle swarm optimization. *Swarm Intelligence*, volume 1, no. 1, June 2007: pp. 33–57, ISSN 1935-3820, doi:10.1007/s11721-007-0002-0.

[20] Hwang, Y. K.; Ahuja, N. A potential field approach to path planning. *IEEE Transactions on Robotics and Automation*, volume 8, no. 1, 1992: pp. 23–32.