

Czech Technical University in Prague  
Faculty of Mechanical Engineering  
Department of Instrumentation and Control Engineering



# Master thesis

Optimal control design of a planar UAV model with  
a load

# Acknowledgment

I wish to thank my family for their support throughout my studies and my supervisor Ing. Jaroslav Buškov, Ph.D. for his patience and helpfulness in supervising this thesis. I thank him for his valuable advice and consultation.

I declare that I have prepared my thesis independently using the literature sources and information that I provide in the Bibliography.

Date: .....

Signature: .....



# ZADÁNÍ DIPLOMOVÉ PRÁCE

## I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Ira** Jméno: **Tomáš** Osobní číslo: **473562**  
Fakulta/ústav: **Fakulta strojní**  
Zadávající katedra/ústav: **Ústav přístrojové a řídicí techniky**  
Studijní program: **Automatizační a přístrojová technika**  
Specializace: **Automatizace a průmyslová informatika**

## II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

**Návrh optimálního řízení planárního modelu UAV s břemenem**

Název diplomové práce anglicky:

**Optimal control design of a planar UAV model with a load**

Pokyny pro vypracování:

- 1) Proveďte rešerši na téma optimální řízení UAV s břemenem.
- 2) Vytvořte zjednodušený (planární) simulační model UAV pro účely návrhu řízení.
- 3) Navrhněte optimální řízení. Porovnejte s vybranou konvenční metodou. Aplikujte na různé scénáře pohybu UAV.
- 4) Experimentálně ověřte funkčnost navrženého řízení.
- 5) Zhodnoťte dosažené výsledky.

Seznam doporučené literatury:

Rubio, Alicia Arce, et al. "Optimal control strategies for load carrying drones." *Delays and Networked Control Systems*. Springer, Cham, 2016. 183-197.  
Hashemi, Danial, and Hamidreza Heidari. "Trajectory planning of quadrotor UAV with maximum payload and minimum oscillation of suspended load using optimal control." *Journal of Intelligent & Robotic Systems* 100.3 (2020): 1369-1381.  
Andrade, Richard, Guilherme V. Raffo, and Julio E. Normey-Rico. "Model predictive control of a tilt-rotor UAV for load transportation." 2016 European Control Conference (ECC). IEEE, 2016.

Jméno a pracoviště vedoucí(ho) diplomové práce:


**Ing. Jaroslav Bušek, Ph.D. U12110.3**


Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

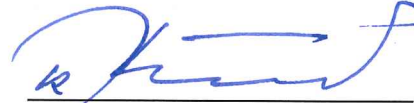
Datum zadání diplomové práce: **27.10.2022**

Termín odevzdání diplomové práce: **26.01.2023**

Platnost zadání diplomové práce: \_\_\_\_\_

  
Ing. Jaroslav Bušek, Ph.D.  
podpis vedoucí(ho) práce

  
podpis vedoucí(ho) ústavu/katedry

  
doc. Ing. Miroslav Španiel, CSc.  
podpis děkana(ky)

## III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

27.10.2022  
Datum převzetí zadání

2

T. Štraš  
Podpis studenta

## **Abstrakt**

The master thesis aims to introduce optimal control algorithms for a UAV with a suspended load and implement a simulation model of the UAV with a suspended load for the control design. Further, the design and implementation of the optimal control strategy and its comparison with a conventional control method. The goal is also to design a laboratory setup of a planar UAV with a suspended load and experimentally verify the algorithms.

## **Keywords**

UAV with a suspended load, LQR, NMPC, Beaglebone Blue

# Contents

Introduction . . . . .	1
1 Literature Review . . . . .	2
1.1 UAVs in General . . . . .	2
1.2 UAVs with a Load . . . . .	2
1.3 Control of a UAV with a Suspended Load . . . . .	2
1.3.1 Optimal Control of a UAV with a Suspended Load . . . . .	3
2 Methods . . . . .	5
2.1 Experimental Setup . . . . .	5
2.1.1 Design of Experimental Setup . . . . .	6
Sensors . . . . .	6
Actuators . . . . .	7
BeagleBone Blue . . . . .	8
BeagleBone Blue & Simulink setup . . . . .	9
Simulink for Controlling BBB . . . . .	9
Troubleshooting with Simulink & Beaglebone Blue . . . . .	10
2.2 Mathematical Model . . . . .	11
2.3 Equations of Motion . . . . .	12
2.4 Equations of Motion in Matrix Form . . . . .	14
2.5 Nonlinear state space model . . . . .	14
2.6 System Identification . . . . .	15
2.6.1 Gray-box Identification . . . . .	15
Identification of Moments of Inertia . . . . .	15
Identification of Friction Coefficients . . . . .	15
Motor Thrust Identification . . . . .	16
2.6.2 Simulation of Nonlinear Model . . . . .	18
2.7 Linearization . . . . .	19
2.7.1 Discrete-time Description of Continuous-time System . . . . .	20
2.7.2 Discretization of Continuous-time system . . . . .	20
2.8 Analysis of Discrete-time State-space . . . . .	22
2.8.1 Stability . . . . .	22
2.8.2 Reachability . . . . .	23
2.8.3 Observability . . . . .	24
2.9 Control Design . . . . .	25
2.9.1 State Feedback Control Design . . . . .	25
State feedback . . . . .	25
Luenberger Observer . . . . .	26
Output feedback . . . . .	26
Integral Control . . . . .	27
2.9.2 Tuning of State Feedback Control . . . . .	28
2.9.3 Simulation validation of State Feedback Control Design . . . . .	31

2.9.4	Linear-quadratic Control . . . . .	39
	Linear-quadratic Regulator . . . . .	39
	Kalman-Bucy filter . . . . .	39
	Tuning of Linear-quadratic Controller . . . . .	40
2.10	Simulation Validation of Linear-quadratic Control . . . . .	44
2.10.1	Nonlinear Model Predictive Control . . . . .	51
	Discretization and Implementation of Optimal Control Problem . . . . .	52
	Tuning the NMPC . . . . .	52
	Simulation Validation of NMPC . . . . .	52
2.11	Comparison of the Control Strategies . . . . .	57
2.12	Validation on the laboratory setup . . . . .	59
	Conclusion and Future Work . . . . .	60
	Appendix A . . . . .	61
	Appendix B . . . . .	62
	Bibliography . . . . .	63
	List of Figures . . . . .	65
	List of Attachments . . . . .	67

# Nomenclature

$\dot{x}, \ddot{x}$	First and second derivative of variable, $x$ , respectively
$\hat{\mathbf{x}}_k$	Estimated value of $\mathbf{x}_k$
$\lambda$	Eigenvalue of the matrix $\mathbf{A}_d$
$\mathbf{A}_c$	State matrix of the continuous-time system
$\mathbf{A}_d$	State matrix of the discrete-time system
$\mathbf{A}_{\text{aug}}$	State matrix of the augmented system
$\mathbf{A}$	State matrix of the nonlinear state-space model
$\mathbf{B}_c$	Input matrix of the continuous-time system
$\mathbf{B}_d$	Input matrix of the discrete-time system
$\mathbf{B}_{\text{aug}}$	Input matrix of the augmented system
$\mathbf{B}$	Input matrix of the nonlinear state-space model
$\mathbf{C}_c$	Output matrix of continuous-time system
$\mathbf{C}_d$	Output matrix of the discrete-time system
$\mathbf{C}$	Viscous damping matrix
$\mathbf{D}_c$	Feedthrough matrix of the continuous-time system
$\mathbf{D}_d$	Feedthrough matrix of the discrete-time system
$\mathbf{f}(\cdot, \cdot)$	State equation
$\mathbf{g}(\cdot, \cdot)$	Output equation
$\mathbf{G}(\mathbf{x}(t))$	Gyroscopic matrix
$\mathbf{K}_0$	Feedback gain
$\mathbf{K}_i$	Gain of integral action
$\mathbf{K}_{\text{LQR}}$	LQR gain
$\mathbf{K}$	Gain of the feedback with integral action
$\mathbf{L}_{\text{LQR}}$	Kalman-Bucy filter gain

$\mathbf{L}(\mathbf{x}(t))$	Input matrix
$\mathbf{L}$	Luenberger observer gain
$\mathbf{M}(\mathbf{x}(t))$	Mass matrix
$\mathbf{P}$	Positive semidefinite matrix
$\mathbf{P}_T$	Terminal weight matrix of the NMPC
$\mathbf{Q}_{\text{mpc}}$	Weight matrix of the NMPC
$\mathbf{Q}_e$	Weight matrix of the Kalman-Bucy filter
$\mathbf{Q}(\mathbf{x}(t))$	Gravitation matrix
$\mathbf{q}(t)$	Generalized coordinates vector
$\mathbf{Q}$	Weight matrix of the LQR
$\mathbf{R}_{\text{mpc}}$	Weight matrix of the NMPC
$\mathbf{R}_e$	Weight matrix of the Kalman-Bucy filter
$\mathbf{R}$	Weight matrix of the LQR
$\mathbf{u}_{\text{max}}$	Upper boundary of $\mathbf{u}_k$
$\mathbf{u}_{\text{min}}$	Lower boundary of $\mathbf{u}_k$
$\mathbf{u}(t)$	System input vector
$\mathbf{v}_k$	Sensing noise
$\mathbf{w}_k$	Process noise
$\mathbf{x}_{\text{Ik}}$	Integral state vector
$\mathbf{x}_{\text{max}}$	Upper boundary of $\mathbf{x}_k$
$\mathbf{x}_{\text{min}}$	Lower boundary of $\mathbf{x}_k$
$\mathbf{x}_{\text{ref}}$	Vector of desired states
$\mathbf{x}(t)$	State vector
$\mathbf{x}_k$	State vector at time $k$
$\mathbf{y}(t)$	Output vector
$\mathcal{C}_n$	Reachability matrix
$\mathcal{L}$	Lagrangian
$\mathcal{O}_n$	Observability matrix
$\mathcal{R}$	Rayleigh's dissipation function



$\mathcal{U}$	Polyhedral set of $\mathbf{u}_k$
$\mathcal{X}$	Polyhedral set of $\mathbf{x}_k$
$\omega$	Angular velocity
$\theta$	Absolute angular displacement of the UAV
$\varphi$	Absolute angular displacement of the pendulum
$a_1$	Distance from the quadrotor's center of gravity to the propeller
$a_2$	Distance from the quadrotor's center of gravity to the cart's center of gravity
$a_3$	Distance from the cart's center of gravity to the pendulum's joint
$B$	Prediction horizon
$b$	Damping coefficient
$c_j$	jth coefficient of linear friction
$E_k$	Kinetic energy
$E_p$	Potential energy
$g$	Acceleration due to gravity
$I_{S_1}$	Pendulum's moment of inertia
$I_{S_2}$	Quadrotor's moment of inertia
$l$	Length of the pendulum
$M$	Torque generated by propellers
$m_1$	Load mass
$m_2$	Linear guide mass
$m_2$	Quadrotor mass
$m_3$	Cart mass
$p_{CL_i}$	ith pole of the closed-loop system
$p_{e_i}$	ith pole of the observer
$q_j$	jth generalized coordinate
$Q_j^{\text{nc}}$	jth generalized force
$T$	Thrust generated by propellers

$T_s$	Sampling time
$t_r$	Rise time
$u_{max}^{(i)}$	Maximum magnitude in control signal in $i$
$x$	Horizontal position of the cart
$y$	Vertical position of the cart
$z_{max}^{(i)}$	Maximum transient error admissible in $z^{(i)}$

# Introduction

In the past decade, Unmanned Aerial Vehicles (UAVs) have witnessed increasing usage due to progress in technology contributing to making them cheaper. UAVs possess many unique abilities. Therefore, UAVs have become popular not only for dangerous rescue missions but also in the hobby community and transportation tasks. Specifically, the utilization of UAVs for emergency assistance and customer package delivery. In those tasks, the UAVs carry a suspended load which dramatically changes the dynamics. In addition, the acceleration and deceleration cause the swing of the payload. Therefore, the control of a UAV with a suspended payload is challenging.

The goal of the master thesis is to introduce optimal control algorithms for a UAV with a suspended load and implement a simulation model. Design optimal control algorithms and verify them in simulation. Furthermore, compare the optimal control algorithms with a conventional control method. In addition, the goal is to build a prototype of an experimental setup for testing the control algorithms for a UAV with a suspended load.

In the Literature review, UAVs with a load are introduced. Furthermore, the optimal control methods of a UAV with a suspended payload are discussed. Methods contain an experimental setup design for testing control algorithms. The mathematical nonlinear model of the experimental setup is derived and implemented in MATLAB/Simulink. In addition, the model is linearized and discretized. Moreover, an analysis of the discrete-time state-space model is performed. Lastly, three control algorithms are implemented. Specifically, output feedback control with integral action, linear-quadratic control, and nonlinear model predictive control. A comparison of their performance is discussed.

# 1 Literature Review

In this section, the UAVs with a load are introduced as well as their control in general. Furthermore, the optimal control methods for a UAV with a suspended payload are discussed.

## 1.1 UAVs in General

An Unmanned Aerial Vehicle (UAV), also called a drone or more specifically quadrotor, is an aircraft without a human pilot on board with the ability to fly semi- or fully autonomously owing to an onboard computer and sensors. In recent years, UAVs have become popular not only for dangerous rescue missions or military missions but also in the hobby community or intelligent transportation. UAVs possess many unique abilities, such as great mobility, hover, vertical take-off, or flying and landing in limited space. Among the numerous applications of drones, aerial load transportation has attracted the attention of several research groups worldwide. The load can be represented as a camera that senses the area of a forest fire or lidar that creates a 3D point cloud of a building or a suspended load.

## 1.2 UAVs with a Load

Two main approaches have been used for load transportation. The first method is based on equipping a quadrotor with grippers [4], [19]. However, it faces the problem of slow response for attitude change because of the additional inertia of the load. The second approach is to connect the payload to the drone by a rope or cable [10], [17]. This approach preserves its agility. On the other hand, the cargo swing dramatically affects the dynamics of the quadrotor. The quadrotor with a suspended load represents an underactuated system, i.e. a system with fewer control inputs than its DOFs. The movement of the quadrotor causes swings of the payload which cannot be directly controlled. Furthermore, a quadrotor itself is also underactuated. Therefore, due to the underactuated property, the control of a quadrotor with a suspended load becomes challenging.

## 1.3 Control of a UAV with a Suspended Load

A list of results, including open-loop control strategies as well as closed-loop methods, have been studied for different purposes. The main task of a quadrotor with a suspended payload can be divided into three flight schemes: takeoff, hover or forward flight, and landing.

During take-off, when the UAV is moving vertically and the load is directly beneath it, the load will not oscillate. In contrast, when the load is situated next to the UAV and it starts taking off, the load will oscillate. This problem has been addressed in [6]. In this work, a nonlinear hybrid controller to track the trajectory of suspended payload has been designed.

The control algorithms for forward motion can be divided into two main groups. The first approach is based on the generation of the flight trajectory and the second one uses an anti-swing controller in the feedback loop.

The damping of oscillations of the suspended payload has been proposed in [18]. In this paper, input shaping is introduced to reduce residual vibration. The strategy modifies a reference command by convolving it with a series of impulses. The authors in [12] solve the problem of damping oscillations by moving in the vertical direction using a nonlinear time-delay control law.

### 1.3.1 Optimal Control of a UAV with a Suspended Load

The following section focuses on different optimal control strategies applied on a UAV with a slung load. Most research in this area has been done on quadrotor position and attitude control using a linear-quadratic controller and model predictive control techniques.

Trachte et al. [20] have presented a controller for the safe and precise operation of a multi-rotor with a slung load in three-dimensional space. The work discusses System Dynamics and Control Simulation Toolbox in MATLAB/SIMULINK. The control design is divided into two major components. The first, top-level controller, i.e. Nonlinear model predictive control (NMPC) or Linear Quadratic Regulator (LQR). The second, low-level proportional attitude control. For top-level control, the NMPC and LQR are evaluated for their ability in managing stabilization and trajectory tracking problems. Specifically, the authors use ACADO toolkit which provides a comprehensive C++ code library suitable for the creation of an algorithm to solve nonlinear optimal control problems. The simulation results show that the LQR control may violate the physical constraints on the system. The NMPC strictly complies with the constraints. Furthermore, the NMPC decreases the overall control effort which is a great advantage when power is limited. However, both optimal control schemes show poor performance for non-predictable disturbance rejection.

In the paper [1], an iterative linear-quadratic regulator (iLQR) for controlling a quadrotor with a cable-suspended heavy rigid body has been presented. For comparison, the classical LQR controller has also been proposed. The controllers have been designed for two control objectives. The first, precisely tracking a given desired trajectory, and the second, an anti-swing load through a transporting task has been studied. The nonlinear dynamics model of the quadrotor with a consideration of the cable-suspended payload is represented. Subsequently, this model is linearised in two modes, a vehicle mode without a load effect during a taking-off task and a switching-to-quad-load system mode considering the load effect. The results have shown that the performance of the iLQR was faster with a small steady-state error than that of the LQR controller. In both control objectives, the iLQR controller has performed better than the LQR controller in terms of cost function, time consumption, and steady-state error.

A hybrid control approach for the swing-free transportation of a double pendulum with a quadrotor has been declared in [7]. The objective of the control strategy is to achieve dampening of the double pendulum oscillations while following a precise trajectory. For this task, a proportional derivative (PD) controller has been combined with a linear model predictive control (MPC). The four PD controllers have been used to compute the quadrotor torque and thrust. Each is in charge of one of the four degrees of freedom. Specifically, the Euler angles and the vertical posi-

tion of the quadrotor. The linear MPC block computes the desired attitude angles minimizing the payload swing. The MPC controller is designed for a discrete-time state-space LTI system, where not all states are measured. Therefore, a Kalman filter for the prediction has been introduced. The experiment has proved that the quadrotor under the combination of PD and MPC controller is capable of following the assigned path as well as robust enough to stick close to the desired parabolic trajectory.

The previously mentioned papers have considered the optimal control strategy in the time domain. The control method which has been proposed in [7] tries to maintain a good balance between transient behavior and frequency-domain performance. This goal has been fulfilled by a mixed  $H_2/H_\infty$  tracking controller based on linear matrix inequality. For the purpose of controller design, the continuous-time linear state space describing the dynamics of a single quadcopter carrying a cable-suspended payload has been used. The controller design has been validated by simulation which has shown that the controller efficiently eliminates the position error and still keeps a smooth change for the quadcopter's attitude. Additionally, extra constraints are added to limit the aggressiveness of the swing angle.

## 2 Methods

The section consists of a description of an experimental setup for testing control algorithms for a planar quadrotor with a suspended payload. The mathematical nonlinear model of the experimental setup is derived together with its identification. Then the nonlinear model is implemented in MATLAB/Simulink. For the control design, the nonlinear model is linearized and discretized. An analysis of the discrete-time state-space model is performed. Finally, three control algorithms are implemented and compared, namely, output feedback control with integral action, linear-quadratic control, and nonlinear model predictive control.

### 2.1 Experimental Setup

The following chapter describes the experimental setup. The goal was to build a low-budget experimental setup for testing control algorithms for a planar quadrotor with a suspended payload. The devices for measuring a quadrotor's position (in centimeters) in 3D are costly. Therefore, the prototype of the experimental setup was designed from scratch.

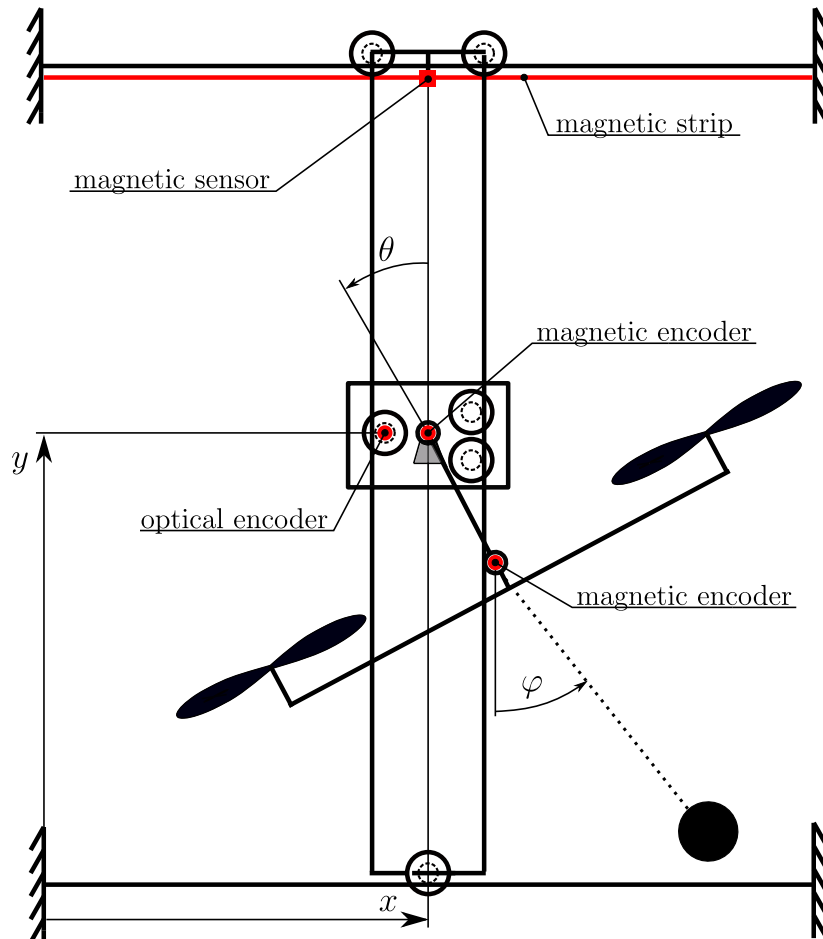


Figure 1: Experimental setup

The proposed experimental setup is displayed in the Figure 1 which consists of a planar quadrotor carrying a suspended load. The quadrotor is attached to a trolley with a rotary joint. The load is suspended with a rope that is attached to the quadrotor also with a rotary joint. Therefore, the quadrotor and suspended load can be represented as a double pendulum. The trolley can translate in  $x$  and  $y$  direction using the track linear guides. The physical realisation of the experimental setup is displayed in the Figure 4.

### 2.1.1 Design of Experimental Setup

The section contains a description of the mechanical components for the experimental setup as well as the electrical parts.

#### Mechanical Components

Most of the mechanical parts, such as the track rollers, quadrotor's body, etc. were designed in Autodesk Inventor and then printed from plastic (PLA, PETG) on a 3D printer to make the construction as light as possible. The track rollers' shafts as well as the pendulums' shafts were machined from steel to ensure stiffness. The trolley was manufactured from plywood.

#### Sensors

The feedback of the system is ensured by several sensors which provide information about the current state of the system.

The knowledge about  $x$  position is provided by an incremental position magnetic sensor AS5304. The sensor composes of two separate parts - a magnetic linear sensor and a magnetic strip. There are 160 pulses per 4.0 mm pole pair length on the standardized quadrature output interface with an index pulse (=ABI-interface). To ensure an accurate measurement, the distance between the strip and the sensor has to be kept constant, approx. 1 mm. The sensor operates on 5 V logic. The sensor's location within the experimental setup is displayed in the Figure 3.

Unlike the  $x$  position measurement, the vertical position could not be measured directly by the magnetic linear sensor because the cart moves at the same time in both directions. Therefore, an indirect measurement using a rotary incremental encoder was used. The output of the sensor is a quadrature 5V signal. An optical incremental encoder by Megatron was attached to the shaft of the track roller. The distance was calculated from the revolutions of the track roller and its diameter.

The angle of the planar quadrotor  $\theta$  and pendulum's angle  $\varphi$  was measured by a magnetic angular encoder AS5048A. The magnetic encoder also consists of two parts - a sensor and a magnet. The magnet was glued to the shaft of the pendulum and quadcopter respectively. The sensor uses a 3.3 V logic level. More details on the sensors can be found in the datasheets. The sensors' location within the experimental setup is displayed in the Figure 2.

The magnetic linear sensor AS5304 and optical encoder from Megatron operate on 5 V logic, whereas the microcontroller Beaglebone Blue operates on 3.3 V logic. For this reason, level shifters were used to transform the voltage level.



## Actuators

Drones have become exceptionally popular over the last few years. The breakthrough not just in the hobby community was motivated by powerful, lightweight, and above all inexpensive brushless DC motor (BLDC). They are not only powerful enough to carry a payload but also relatively rapid to move.

The planar quadrotor was actuated by two BLDC outrunner motors. The choice of motors was a tradeoff between the weight and the thrust that the motor can provide. Consequently, KAVAN Brushless motor C3542-1250 was chosen. As thrust generators, the SPORT 10x5 inches propellers were used. The combination of this propeller and the KAVAN motor can achieve a thrust of up to 1.8 kg. An essential part of a BLDC motor is an electronic speed controller (ESC). The main function of the ESC is to regulate the motor's speed. The ESC was selected with respect to the motor - KAVAN R-60SB was chosen. The ESCs were powered by a 12 V lead-acid battery.

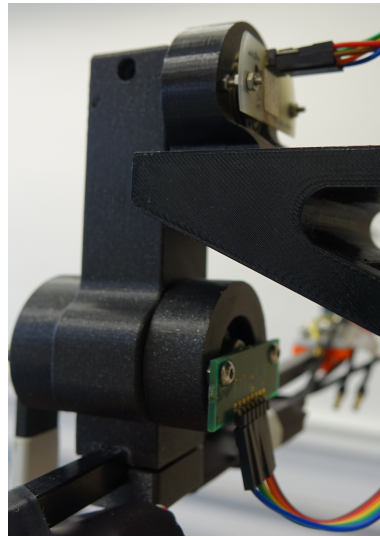


Figure 2: Magnetic encoders

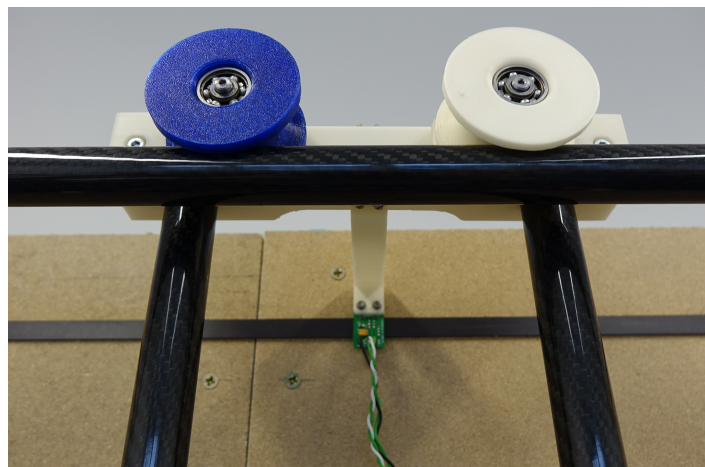


Figure 3: Magnetic linear sensor

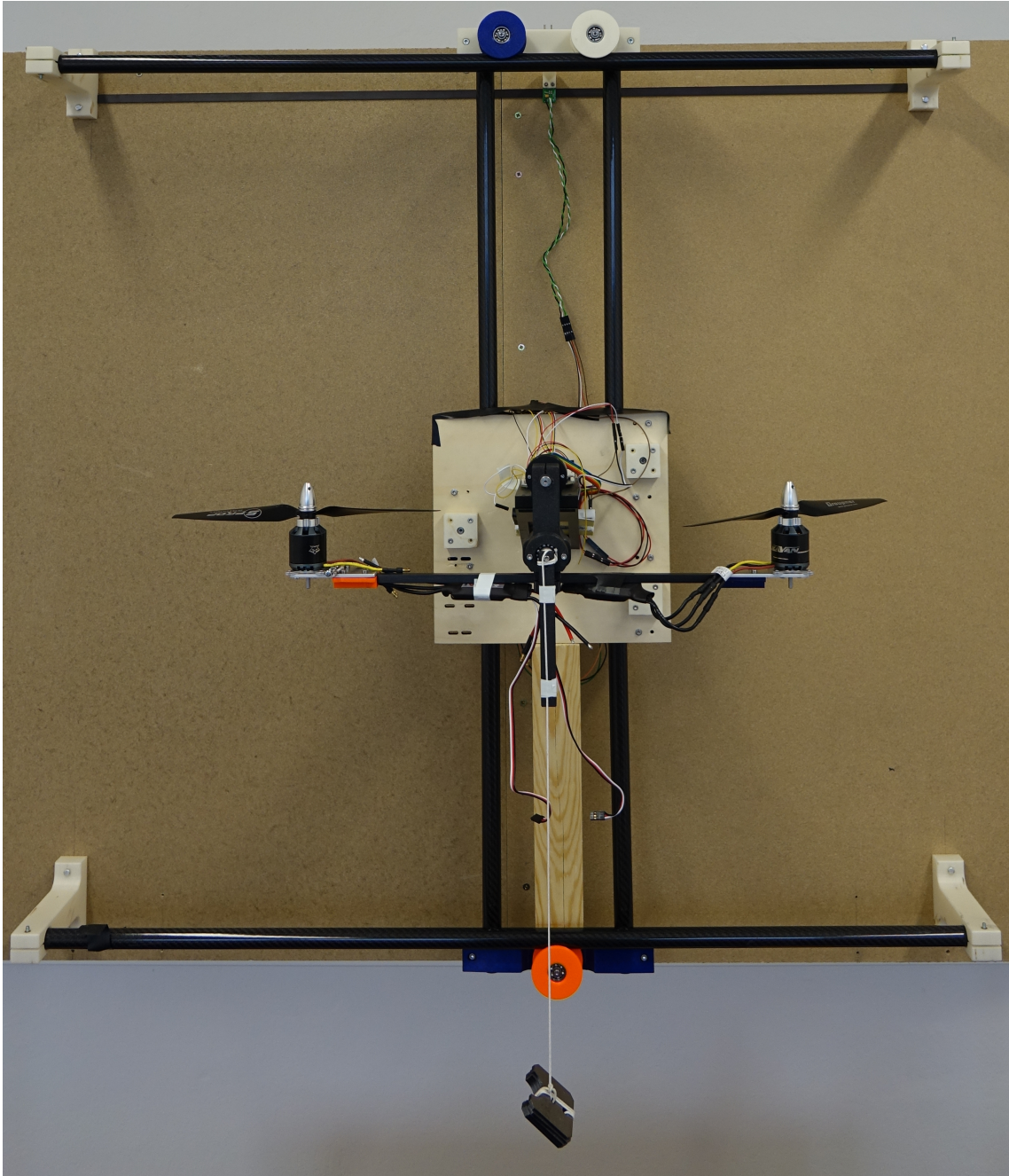


Figure 4: Experimental setup

### **BeagleBone Blue**

The main computing unit was BeagleBone Blue (BBB). BeagleBone Blue is a Linux-based computer for robotics, integrating onto the board the Octavo OSD3358 microprocessor together with all necessary peripherals, such as wifi/Bluetooth, IMU, and much more. The board also has several built-in connectors: 4 DC motors and encoders, 8 servos, SPI, I2C, and UART. The computer is compatible with MATLAB/Simulink. Therefore, it makes a good choice for the development and testing of the control algorithms. Because of the issues with the BeagleBone Blue support package in Simulink, which will be discussed later, a second board had to be used - NodeMCU-32S. This board was used to read data from the magnetic encoders via SPI and then sent by UART to BBB.

The BBB, NodeMCU-32S, sensors, and motors were connected according to the following diagram.

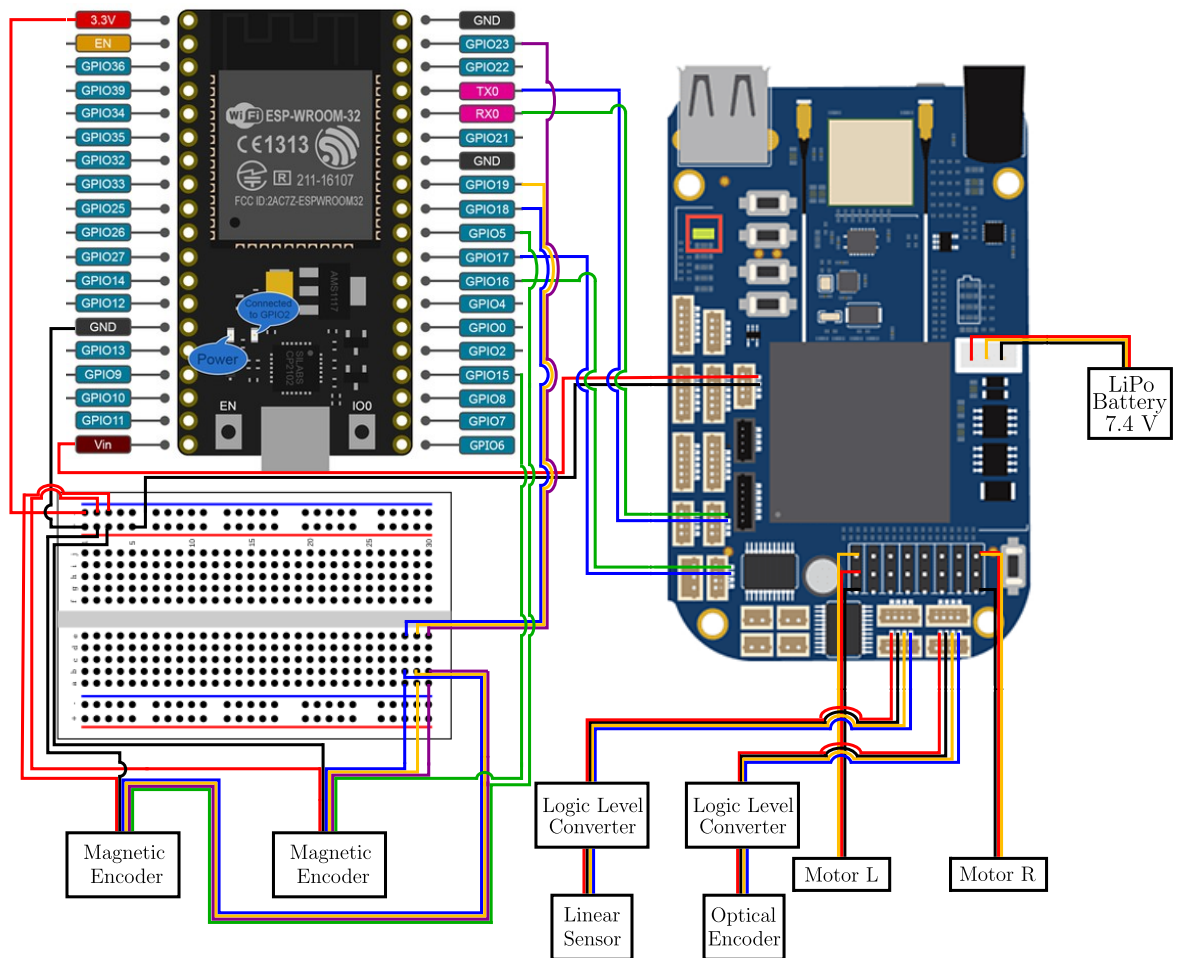


Figure 5: Diagram of control wiring

### BeagleBone Blue & Simulink setup

Before an initial Simulink code could be deployed, several setup steps on BBB had to be done. The approach followed the official BBB website. [13]

First, the Debian image had to be flashed to a microSD card and transferred to the BBB. Next, the Simulink Coder package for BeagleBone Blue hardware had to be installed and configured.

After the successful installation and configuration of the BBB board, a first Simulink model could be created. The following steps were adopted from Mathworks documentation. [15] The BBB has two means of connection - via USB or wireless WiFi connection. The second method was used. The connection between a PC and the BBB blue was established using the command `beagleboneblue('IP address')`

### Simulink for Controlling BBB

The Simulink coder support package for BeagleBone Blue hardware contains blocks for supporting hardware protocols. Those blocks were used to interact between BBB and sensors and motors.

The BLDC motors were controlled using a Servo Motor block. The input is the

shaft position of the servo motor which in the case of BLDC corresponds to the speed of the motor because the block's output signal is a PWM signal in both cases.

The optical encoder and linear magnetic sensor send quadrature output. Thus, the Encoder block was used. The output of the block had to be gained with a constant that links the tick of the quadrature signal with the angle and distance respectively. Those constants were read from datasheets.

The magnetic encoder AS5048A uses the SPI interface. The Simulink Package includes an SPI Register Read block. However, several unsuccessful attempts were made to establish communication. Consequently, another board - NodeMCU-32S was used to read the data from the magnetic encoder via SPI. Then, the data from the NodeMCU-32S board were sent via UART to BBB where an SCI Read block was implemented. The raw data received from the sensor are 12-bit angle values. Accordingly, the raw angle values were converted to radians. In addition to this, the angle in radians is an absolute value. Naturally, another step was to obtain the relative angle regarding the start of the simulation. This was implemented using the MATLAB function block. The input to this block was an unwrapped angle in radians. This was implemented with Unwrap block which ensures that if the difference between two consecutive samples is bigger than a chosen threshold, the block adds to the input value  $2\pi$  radians.

The measured data are stored in BBB as mat files. To get the data to a PC a `getFile()` function was used. This function was added inside StopFcn Callback. Those callbacks are executed at the end of a simulation and the files are transferred to the PC.

### **Troubleshooting with Simulink & Beaglebone Blue**

During the deployment of the Simulink codes, main two problems occurred. The first error, a Checksum mismatch error, aroused when Simulink was restarted, and the files from the previous session were still stored in BBB. This issue can be solved by deleting the files with `deleteFiles()` from BBB or just resaving the Simulink code with a different name. The second issue was with a process running on BBB which highly burdened the CPU. If the process with a specific PID was running, the Simulink code could not be run. This was solved by connecting to the BBB via SSH in the command prompt and killing the process manually.

## 2.2 Mathematical Model

The laboratory setup consists of a trolley with mass  $m_3$ . The mass of the quadrotor and the mass of the suspended load are denoted  $m_2$  and  $m_1$ , respectively. The vertical linear guide has a mass of  $m_4$ . The motion of the planar quadrotor is provided by two motors with propellers that generate thrust  $T$  and torque  $M$ . The quadrotor's angle of rotation is labeled  $\theta$ , whereas the angle of the pendulum is labeled  $\varphi$ .

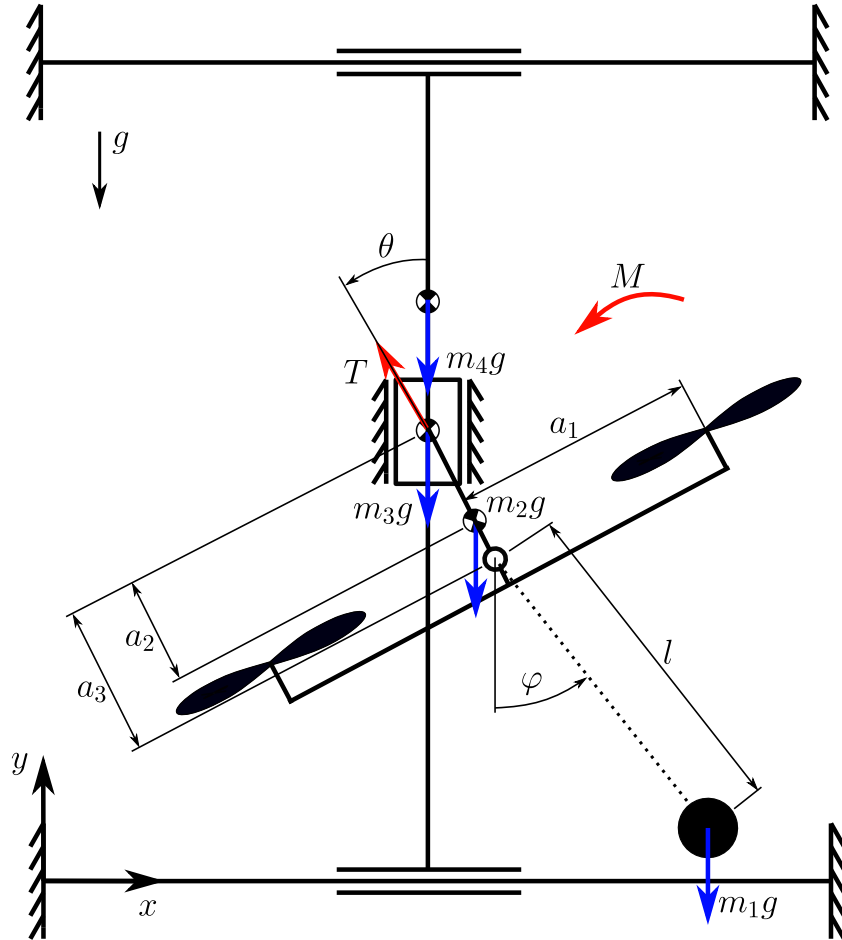


Figure 6: Scheme of the laboratory setup

For the derivation of a simplified mathematical model of the planar quadrotor with a suspended load which describes the laboratory setup, the following assumptions were made:

- friction in joints is linearly dependent on velocity
- the pendulum is considered a mathematical pendulum
- the system undertakes a flat motion in the plane
- the parameters of the system are time-invariant

## 2.3 Equations of Motion

For the derivation of equations of motion, Euler–Lagrange equation of the second kind extended with Rayleigh’s dissipation function in the following form was used:

$$\frac{d}{dt} \left( \frac{\partial \mathcal{L}}{\partial \dot{q}_j} \right) - \frac{\partial \mathcal{L}}{\partial q_j} + \frac{\partial \mathcal{R}}{\partial \dot{q}_j} = Q_j^{\text{nc}}, \quad j = 1, \dots, J, \quad (1)$$

where  $j$  is the number of generalized coordinate,  $q_j$  is the  $j$ th generalized coordinate and  $Q_j^{\text{nc}}$  the  $j$ th generalized nonconservative force. In the case of the considered laboratory setup, the generalized coordinates were chosen as  $q_1(t) = x(t)$ ,  $q_2(t) = y(t)$ ,  $q_3(t) = \theta(t)$  and  $q_4(t) = \varphi(t)$ . As well as the generalized non-conservative forces were selected  $Q_1^{\text{nc}} = 0$ ,  $Q_2^{\text{nc}} = T$ ,  $Q_3^{\text{nc}} = M$  and finally  $Q_4^{\text{nc}} = 0$ .

Lagrangian  $\mathcal{L}$  is given as the difference between the kinetic and potential energy of the system

$$\mathcal{L} \equiv E_k - E_p. \quad (2)$$

The system’s kinetic energy is equal to the sum of the kinetic energies of each of the bodies. Likewise, the kinetic energy of a rigid body can be written according to the König’s theorem as the sum of translation energy and rotational energy relative to the center of mass. Thus, the kinetic energy of the system displayed in the Figure 6 is given as follows

$$E_k = \frac{1}{2} \sum_{n=1}^4 m_n (\dot{x}_{\text{cm}_n}^2(t) + \dot{y}_{\text{cm}_n}^2(t)) + \frac{1}{2} \sum_{n=1}^4 I_{\text{cm}_n} \omega_n^2(t). \quad (3)$$

The potential energy of gravity forces  $V$  is equal to:

$$E_p = \sum_{i=1}^4 V_i = m_1 g y_{\text{cm}_1}(t) + m_2 g y_{\text{cm}_2}(t) + m_3 g y_{\text{cm}_3}(t) + m_4 g y_{\text{cm}_4}(t) \quad (4)$$

Linear velocity-dependent energy dissipation can be handled by use of Rayleigh’s dissipation function  $\mathcal{R}$

$$\mathcal{R} = \frac{1}{2} \sum_{i=1}^4 c_i \dot{q}_i^2(t), \quad (5)$$

where  $c_j$  is the coefficient of linear friction. In later text, the coefficients  $c_j$  are denoted as  $c_1 = c_x$ ,  $c_2 = c_y$ ,  $c_3 = c_\theta$  and  $c_4 = c_\varphi$ .

The coordinates of the mass center appearing in the equations (3) and (4) are equal to

$$\begin{aligned}
x_{\text{cm}_3}(t) &= x(t), \\
y_{\text{cm}_3}(t) &= y(t), \\
x_{\text{cm}_2}(t) &= x(t) + a_2 \sin(\theta(t)), \\
y_{\text{cm}_2}(t) &= y(t) - a_2 \cos(\theta(t)), \\
x_{\text{cm}_1}(t) &= x(t) + a_3 \sin(\theta(t)) + l \sin(\varphi(t)), \\
y_{\text{cm}_1}(t) &= y(t) - a_3 \cos(\theta(t)) - l \cos(\varphi(t)), \\
x_{\text{cm}_4}(t) &= x(t), \\
y_{\text{cm}_4}(t) &= d \\
\omega_1(t) &= \dot{\varphi}(t) \\
\omega_2(t) &= \dot{\theta}(t) \\
\omega_3(t) &= 0 \\
\omega_4(t) &= 0
\end{aligned} \tag{6}$$

The potential energy was obtained by inserting expressions (6) into (4) and the kinetic energy was obtained by inserting derivative of (6) into relationships (3) which after substitution into the (2) gave Lagrangian in form

$$\begin{aligned}
\mathcal{L} &= (m_2(\dot{x} + a_2 \cos(\theta(t))\dot{\theta}(t))^2 + (\dot{y}(t) + a_2 \sin(\theta(t))\dot{\theta}(t))^2) + m_1(\dot{x} + a_3 \cos(\theta(t))\dot{\theta}(t) + \\
&+ l \cos(\varphi(t))\dot{\varphi}(t))^2 + (\dot{y}(t) + a_3 \sin(\theta(t))\dot{\theta}(t) + l \sin(\varphi(t))\dot{\varphi}(t))^2 + m_3(\dot{x}^2 + \dot{y}^2) + \\
&+ I_{s_2}\dot{\theta}^2 + I_{s_1}\dot{\varphi}^2 + gm_1(a_3 \cos(\theta(t)) - y(t) + l \cos(\varphi(t))) - gm_3y(t) - gm_2(y(t) - \\
&- a_2 \cos(\theta(t))) - dgm_4.
\end{aligned} \tag{7}$$

Lagrangian (7) and Rayleigh's dissipation function (5) were derived according to equation (1) to gain the nonlinear equations of motion in form

$$\left\{ \begin{aligned}
&(m_1 + m_2 + m_3 + m_4)\ddot{x} + c_x \dot{x} + (a_2 m_2 + a_3 m_1)\ddot{\theta} \cos(\theta(t)) + l m_1 \ddot{\varphi} \cos(\varphi(t)) - \\
&- (a_2 m_2 - a_3 m_1)\dot{\theta}^2 \sin(\theta(t)) - l m_1 \dot{\varphi}^2 \sin(\varphi(t)) = -T(t) \sin(\theta(t)) \\
&(m_1 + m_2 + m_3)g + c_y \dot{y} + (m_1 + m_2 + m_3)\ddot{y} + l m_1 \ddot{\varphi} \sin(\varphi(t)) + l m_1 \dot{\varphi}^2 \cos(\varphi(t)) + \\
&+ (a_2 m_2 a_3 m_1)\dot{\theta}^2 \cos(\theta(t)) + (a_2 m_2 + a_3 m_1)\ddot{\theta} \sin(\theta(t)) = T(t) \cos(\theta(t)) \\
&(I_{s_2} + a_2^2 m_2 + a_3^2 m_1)\ddot{\theta} + (a_2 m_2 + a_3 m_1)\ddot{y} \sin(\theta(t)) + a_3 l m_1 \ddot{\varphi} \cos(\theta(t) - \varphi(t)) + c_\theta \dot{\theta} + \\
&+ (a_2 m_2 + a_3 m_1)\ddot{x} \cos(\theta(t)) + (a_2 m_2 + a_3 m_1) \sin(\theta(t)) g + a_3 l m_1 \dot{\varphi}^2 \sin(\theta(t) - \varphi(t)) = \\
&= M(t) \\
&(I_{s_1} + l^2 m_1)\ddot{\varphi} + c_\varphi \dot{\varphi} + l m_1 \ddot{y} \sin(\varphi(t)) + g l m_1 \sin(\varphi(t)) + l m_1 \ddot{x} \cos(\varphi(t)) + \\
&+ a_3 l m_1 \ddot{\theta} \cos(\theta(t) - \varphi(t)) - a_3 l m_1 \dot{\theta}^2 \sin(\theta(t) - \varphi(t)) = 0
\end{aligned} \right. \tag{8}$$

## 2.4 Equations of Motion in Matrix Form

The set of second-order ordinary differential equations (8) can be also represented in a more concise and clear matrix form

$$\mathbf{M}(\mathbf{q}(t))\ddot{\mathbf{q}}(t) + (\mathbf{C} + \mathbf{G}(\mathbf{q}(t)))\dot{\mathbf{q}}(t) + \mathbf{Q}(\mathbf{q}(t)) = \mathbf{L}(\mathbf{q}(t))\mathbf{u}(t), \quad (9)$$

where the vector  $\mathbf{q}(t)$  concatenates  $\mathbf{q}(t) = [x(t), y(t), \theta(t), \varphi(t)]^T$ , while the system input  $\mathbf{u}(t)$  concatenates  $\mathbf{u}(t) = [T(t), M(t)]^T$ . The matrices can be found in Appendix.

## 2.5 Nonlinear state space model

The set of  $n$  second-order differential equations can be also represented as a set of  $2n$  first-order equations. A set of first-order nonlinear differential equations representing a time-invariant system has the following form

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \quad (10a)$$

$$\mathbf{y}(t) = \mathbf{h}(\mathbf{x}(t), \mathbf{u}(t)). \quad (10b)$$

The equations above are the so-called state-space model. Furthermore, the function  $\mathbf{f}(\cdot, \cdot)$  is called a state equation and the function  $\mathbf{g}(\cdot, \cdot)$  is called an output equation. The vector  $\mathbf{x}(t)$  is a vector of state variable and vector  $\mathbf{u}(t)$  is a vector of inputs acting on the system. Finally, the vector  $\mathbf{y}(t)$  is the output vector - a vector of variables that are measured. To create a nonlinear state-space model of the experimental setup the state vector was determined as  $\mathbf{x}(t) = [\mathbf{q}(t), \dot{\mathbf{q}}(t)]$  and input vector remained the same as defined in equation (9), i.e  $\mathbf{u}(t) = [T(t), M(t)]^T$ .

The nonlinear state-space system (9) can also be represented by introducing matrices ( $\mathbf{A}$ ,  $\mathbf{B}$ ) as follows

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \quad (11)$$

The relationship between matrices in equation (9) and nonlinear state-space form (11) is following

$$\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{0} & -\mathbf{M}^{-1}(\mathbf{C} + \mathbf{G}) \end{bmatrix} \quad (12)$$

$$\mathbf{B} = \begin{bmatrix} \mathbf{0} \\ \mathbf{M}^{-1}\mathbf{L} \end{bmatrix}. \quad (13)$$



## 2.6 System Identification

The control design is based on the knowledge of the controlled system. In the previous chapter, the nonlinear model has been derived. However, the parameters appearing in the model are unknown. Therefore, the following section is focused on the identification of the parameters.

### 2.6.1 Gray-box Identification

Gray-box identification estimates the structure of the model by means of analytical tools and the parameters of the model are estimated using experimental data. In this case, the parameters were estimated by several experiments. The first and simplest experiment was conducted to estimate the masses  $m_1$ ,  $m_2$ ,  $m_3$ , and  $m_4$  which were measured on a digital scale. In the same fashion, the lengths  $a_1$ ,  $a_2$ ,  $a_3$ , and  $l$  were measured using a ruler and a tape measure.

#### Identification of Moments of Inertia

The identification of moment of inertia  $I_{S_1}$  was based on the assumption that the pendulum is considered a mathematical pendulum where the moment of inertia can be calculated as  $I_{S_1} = m_1 l^2$ . The second moment of inertia  $I_{S_2}$  was analyzed in Autodesk Inventor. In this software, the 3D model of the planar UAV was created and  $I_{S_2}$  was found.

#### Identification of Friction Coefficients

For the identification of friction coefficients, the whole system was separated into subsystems where the specific coordinates were fixed. First, the coordinates  $x$ ,  $y$ , and  $\theta$  were fixed for measuring the coef. of friction  $c_\varphi$ . In this case, the subsystem was assumed a damped harmonic oscillator. The equation of the damped harmonic oscillator is

$$\ddot{\varphi}(t) + b\dot{\varphi}(t) + mgl\varphi(t) = 0 \quad (14)$$

The solution to this equation is given

$$\varphi(t) = \varphi_0 e^{\frac{-bt}{2m}} \cos(\omega t + \Phi), \quad (15)$$

where  $\omega$  is

$$\omega = \sqrt{\frac{g}{l}} \quad (16)$$

The measured data of the subsystem were fitted in MATLAB using the `fminsearch()` function to the equation (15). The fitted function is given as

$$\varphi(t) = 22.82e^{-0.01t} \cos(3.23t - 0.11), \quad (17)$$

Comparing the equation (17) and (15) with known parameters  $m$  and  $l$ , the desired coefficient of friction is calculated  $c_\varphi = 0.013 \text{ kgs}^{-1}$ .

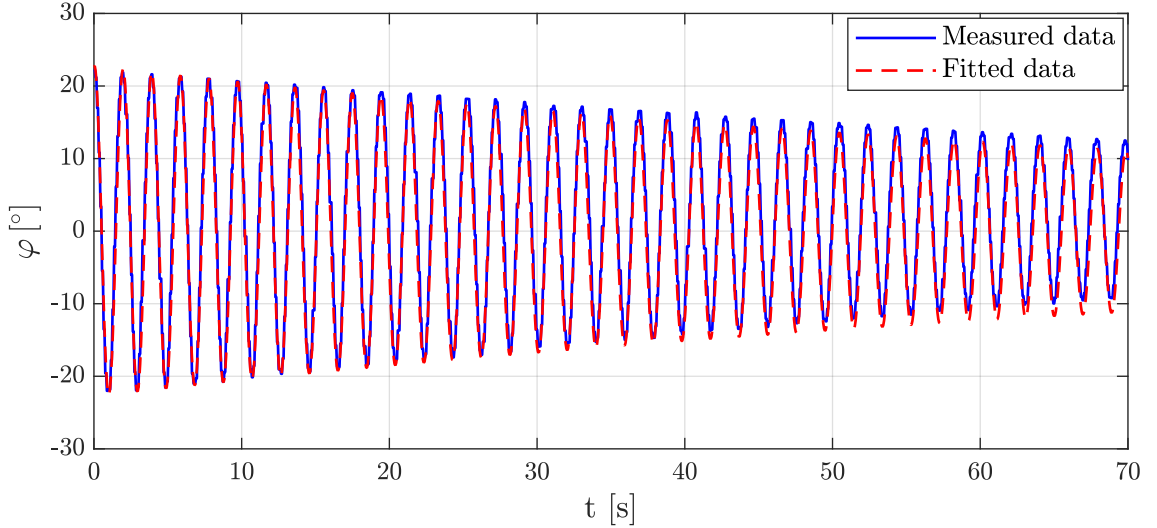


Figure 7: Identification of the coef. of friction  $c_\varphi$ , nmrs = 82 %

The same bearings as for the pendulum joint were used in the joint which restricts the motion of the quadrotor. Therefore, the coefficient of friction  $c_\theta$  was assumed the same as for  $c_\varphi$ .

The coefficients of friction  $c_x, c_y$  were estimated using the force required to move the trolley with a constant velocity. In the horizontal direction, the quadrotor was tilted at an angle of 18.5 degrees, and from equality of forces, the coefficient was estimated  $c_x = 0.04 \text{ kgs}^{-1}$ . Similarly, the  $c_y$  coef. was estimated. The thrust which caused the vertical movement with constant velocity was measured and  $c_y$  was determined as  $c_y = 0.3 \text{ kgs}^{-1}$ .

### Motor Thrust Identification

The motor with ESC was tested experimentally to find the dependency between thrust and MATLAB value. For this purpose, the motor test stand was built as it is displayed in the following figure. The idea is that the thrust generated by the propeller is transferred to the digital scale. The frame is rotationally mounted, directly beneath the BLDC motor, enabling the frame to freely rotate and transfer the thrust to the scale. The horizontal and vertical distances of the frame are the same. The measured data were fitted with the polynomial function of second order

$$T = -9.876^{-5}x^2 + 0.02879x - 0.8069. \quad (18)$$

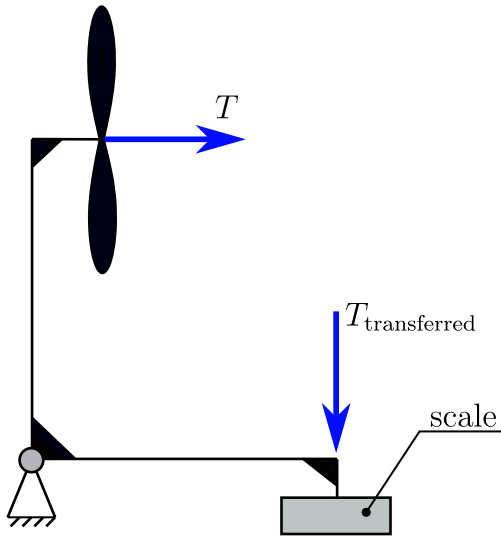


Figure 8: Thrust test stand

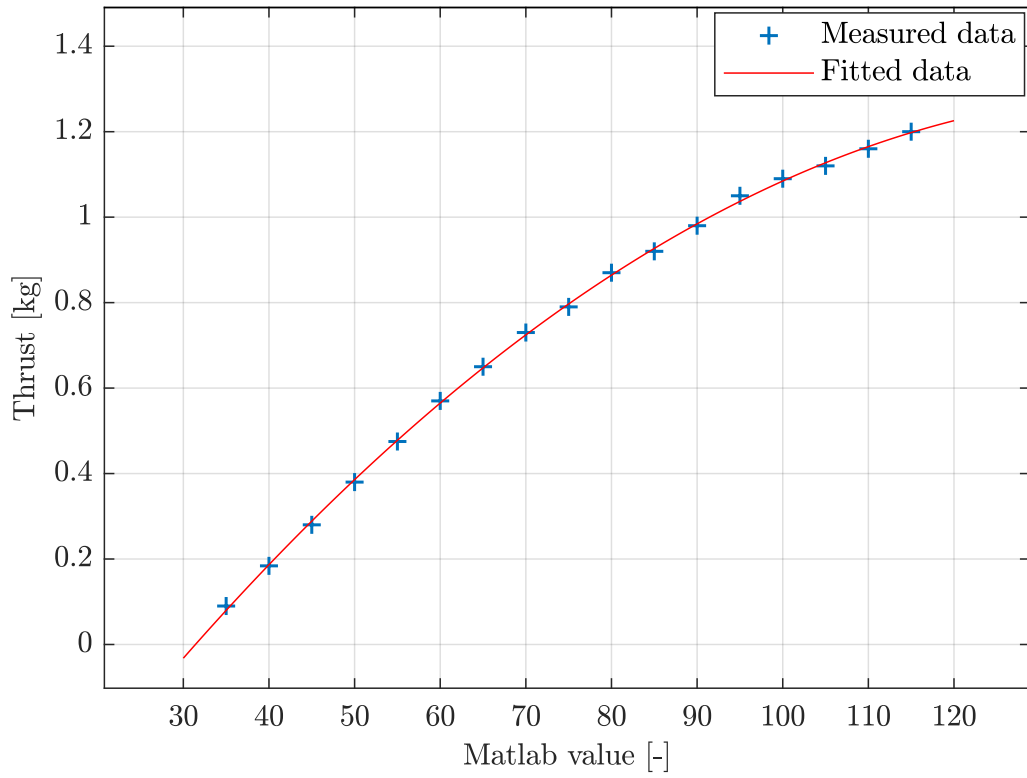


Figure 9: Thrust measurement

## 2.6.2 Simulation of Nonlinear Model

Simulation-based validation of nonlinear dynamics (11) was carried out in MATLAB/Simulink. The matrix differential equation was implemented by MATLAB function block in Simulink together with an integrator. The parameters identified in the previous section were used in the simulation. The parameters are listed in Table 1. The response of this system with a non-zero initial value on  $\varphi = 0.2$  rad is displayed in Figure 10.

parameter	value
$m_1$	0.3 kg
$m_2$	0.72 kg
$m_3$	1.0 kg
$m_4$	0.4 kg
$a_1$	0.225 m
$a_2$	0.07 m
$a_3$	0.066 m
$l$	0.7 m
$I_{S_1}$	0.061 kgm <sup>2</sup>
$I_{S_2}$	0.02 kgm <sup>2</sup>
$c_x$	0.04 kgs <sup>-1</sup>
$c_y$	0.3 kgs <sup>-1</sup>
$c_\theta$	0.013 kg <sup>-1</sup>
$c_\varphi$	0.013 kgs <sup>-1</sup>
$g$	9.81 ms <sup>-2</sup>

Table 1: The system parameters

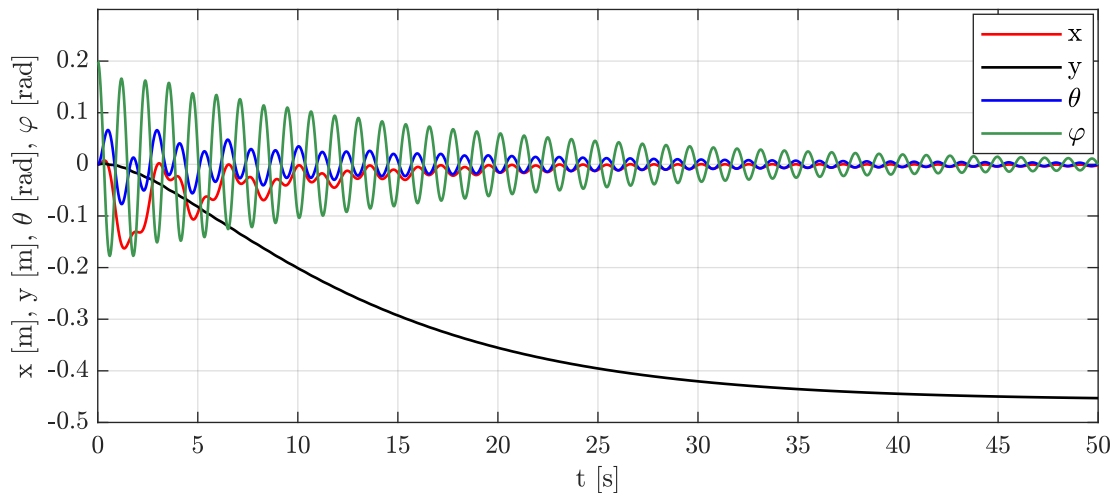


Figure 10: Simulation results of the model (9) with initial conditions  $\mathbf{x}_0 = [0, 0, 0, 0.2, 0, 0, 0, 0]^T$ , (MATLAB/Simulink, ode45, rel. tol. 1e-6, max. step size 0.01)

The response of the system in Figure 10 confirms the behavior of the planar quadrotor with a suspended payload. The pendulum's angle is attenuating over

time because of friction. The  $x$  position as well as the quadrotor's angle  $\theta$  oscillating which is caused by the swaying of the pendulum. As a result of pendulum sway, the vertical position  $y$  is decreasing till the pendulum oscillations are damped. This is caused by the fact that the quadrotor oscillates as well. Therefore, the motor thrust is not always pointing in the vertical direction.

## 2.7 Linearization

The equations (8) or equivalent (11) describing the dynamics of the system are highly nonlinear. However, the most common control design methods are much simpler and straightforward for linear rather than for nonlinear models.

Linearization is a technique for finding a linear approximation of a nonlinear model. As Lyapunov proved, if a linear model is valid near an equilibrium point and is stable, then there exists a region containing the equilibrium within the nonlinear model is stable. Moreover, a crucial role of feedback control is to maintain the controlled system near equilibrium, so this linearized system is a good starting point for control design. [9]

The nonlinear dynamics were approximated by a linear state-space model about the equilibrium point.

Linearized continuous-time state-space model approximating the nonlinear dynamics of the system has the following form

$$\begin{aligned}\delta\dot{\mathbf{x}}(t) &= \mathbf{A}_c\delta\mathbf{x}(t) + \mathbf{B}_c\delta\mathbf{u}(t) \\ \delta\mathbf{y}(t) &= \mathbf{C}_c\delta\mathbf{x}(t) + \mathbf{D}_c\delta\mathbf{u}(t),\end{aligned}\tag{19}$$

where  $\delta\mathbf{x}(t)$  and  $\delta\mathbf{u}(t)$  refer to the deviation from the equilibrium to point out that the model is linearized. In the following text, the  $\delta$  symbols are disregarded. The Jacobian matrices ( $\mathbf{A}_c$ ,  $\mathbf{B}_c$ ,  $\mathbf{C}_c$  and  $\mathbf{D}_c$ ) are given by

$$\mathbf{A}_c = \left. \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right|_{(\mathbf{x}_e, \mathbf{u}_e)}\tag{20}$$

$$\mathbf{B}_c = \left. \frac{\partial \mathbf{f}}{\partial \mathbf{u}} \right|_{(\mathbf{x}_e, \mathbf{u}_e)}\tag{21}$$

$$\mathbf{C}_c = \left. \frac{\partial \mathbf{h}}{\partial \mathbf{x}} \right|_{(\mathbf{x}_e, \mathbf{u}_e)}\tag{22}$$

$$\mathbf{D}_c = \left. \frac{\partial \mathbf{h}}{\partial \mathbf{u}} \right|_{(\mathbf{x}_e, \mathbf{u}_e)}\tag{23}$$

The equilibrium state and the input were determined  $\mathbf{x}_e = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]^T$ ,  $\mathbf{u}_e = [(m_1 + m_2 + m_3)g, 0]^T$ . That corresponds to a state when the quadrotor is hovering. As a result of derivation (10a) according to (20) and (21) the state space matrices were obtained. The output matrix  $\mathbf{C}_c$  was selected considering that only the first four state variables in the state vector were measured, i.e.  $x(t)$ ,  $y(t)$ ,  $\theta(t)$ , and  $\varphi(t)$ . The last matrix appearing in the state-space, the matrix  $\mathbf{D}_c$  was determined as a zero matrix. The matrices are listed in Appendix B.

The state-space model (19) was implemented in Simulink using the State-space model block. The simulation results are showing that the vertical position  $y$  is fixed because of the linearization.

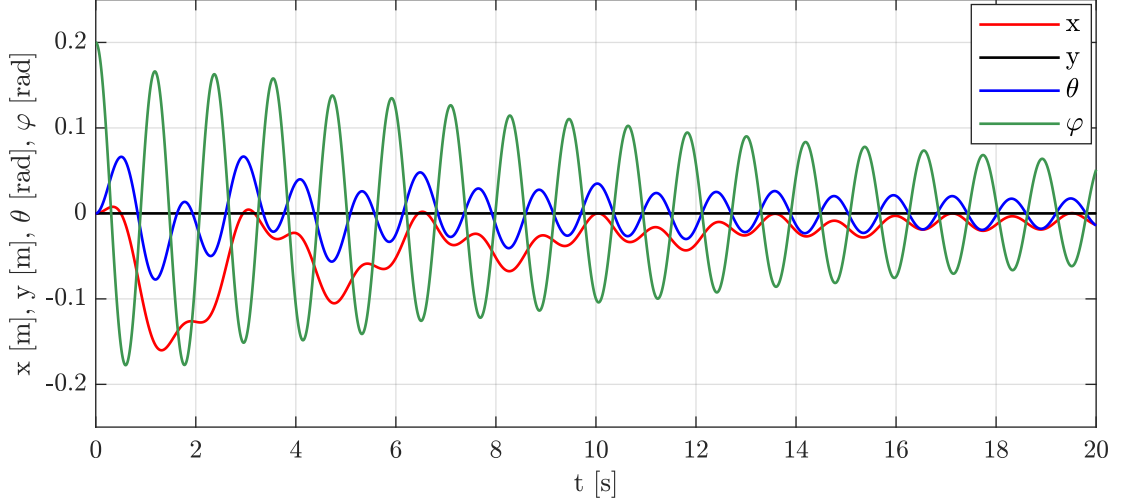


Figure 11: Simulation results of the model (19) with initial conditions  $\mathbf{x}_0 = [0, 0, 0, 0.2, 0, 0, 0, 0]$  and input vector  $\mathbf{u} = [0, 0]^T$ , (MATLAB-Simulink, ode45, rel. tol. 1e-6, max. step size 0.01)

### 2.7.1 Discrete-time Description of Continuous-time System

The equations developed in the previous chapters describe the system in the continuous-time domain, which is assumed to be implemented on analog electronics. However, these days, most control systems use digital computing controllers. Thus, in the following chapter, the discretization of the continuous-time system is represented.

### 2.7.2 Discretization of Continuous-time system

There are many approaches to Discretization - sampling and reconstruction of a continuous-time signal. In this thesis, the main focus was on a zero-order hold converter. Zero-order hold accepts a sample at time  $t$  and maintains its output constant until the next sample is sent. More details on this method can be found for instance in [8].

For the linearized continuous-time state-space model (19) preceded by a zero-order hold, the equivalent discrete-time state-space difference equations are

$$\begin{aligned}\mathbf{x}_{k+1} &= \mathbf{A}_d \mathbf{x}_k + \mathbf{B}_d \mathbf{u}_k \\ \mathbf{y}_k &= \mathbf{C}_d \mathbf{x}_k + \mathbf{D}_d \mathbf{u}_k,\end{aligned}\tag{24}$$

where

$$\mathbf{A}_d = e^{\mathbf{A}_c T_s}, \quad \mathbf{B}_d = \int_0^{T_s} e^{\mathbf{A}_c \tau} \mathbf{B}_c d\tau,\tag{25}$$

where  $T_s$  donates the sampling time.

The lower bound on selecting the sampling frequency brings the Nyquist frequency. However, the sampling frequency given by Nyquist frequency is not satisfactory to get acceptable quality of closed-loop control. Therefore higher frequencies are used. Sample rate draft is a trade-off between computational load on the controller and effectiveness in control. The choice of sampling frequency was done using the rule of thumb which recommends 4 - 10 samples per rise time. To capture the

dynamics of the system, the state with the fastest response has to be considered. In figure 11, it can be observed, that the shortest raising time has a curve that corresponds to the pendulum's angle  $\varphi$ . The raising time  $t_r$  was read as  $t_r \approx 0.3$  s. Therefore, the sampling time was dedicated as  $T_s = 0.05$  s.

The matrices ( $\mathbf{A}_d$ ,  $\mathbf{B}_d$ ) were obtained in MATLAB using `c2d()` function. Those matrices can be found in Appendix B.

The simulation of (24) are showing the same results as for the continuous-time case, but with visible sampling.

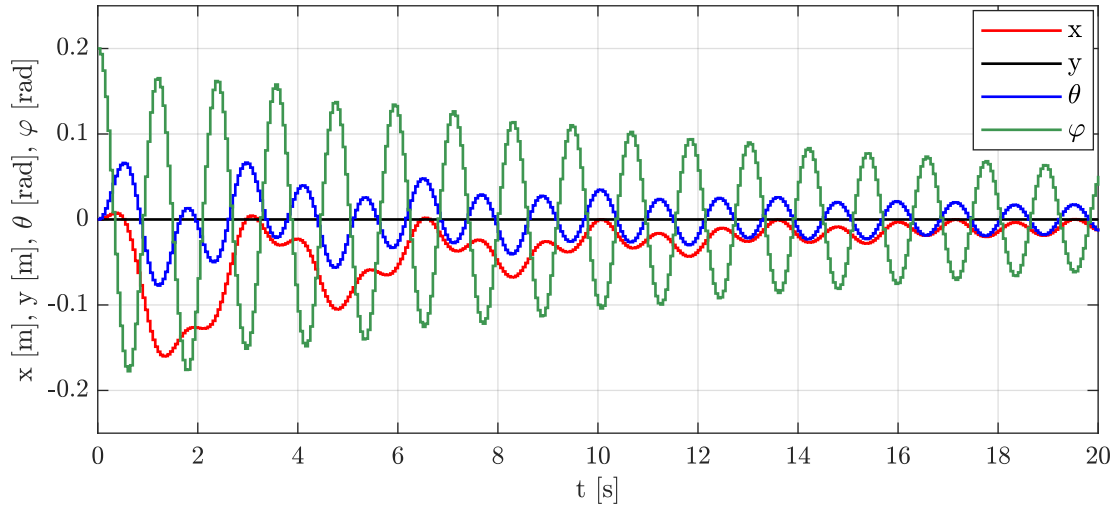


Figure 12: Simulation results of the model (24) with initial conditions  $\mathbf{x}_0 = [0, 0, 0, 0.2, 0, 0, 0, 0]$  and input vector  $\mathbf{u}_0 = [0, 0]^T$ , (MATLAB/Simulink, ode4, Fixed-step size 0.05)

## 2.8 Analysis of Discrete-time State-space

The previous chapter has shown how the nonlinear system was linearized and transformed when sampled. In this section, the key tools for analyzing the discrete-time system represented by state-space, are developed, namely stability, reachability, and observability. Furthermore, those concepts are also important for later control design. The control algorithms were designed for the discrete-time model because of its implementation on an embedded system so the analysis was done also on the discrete-time model.

### 2.8.1 Stability

The notion of stability is the main tool for the analysis of dynamics. There are many types of stability concepts. In this thesis, the focus is on the asymptotic stability of discrete-time linear systems.

Discrete-time linear system (24) with  $\mathbf{A}_d \in \mathbb{R}^{n \times n}$  is asymptotically stable if and only if the matrix  $\mathbf{A}_d$  is Schur, i.e.

$$|\lambda_i(\mathbf{A}_d)| < 1, \quad \forall i = 1, \dots, n. \quad (26)$$

Geometrically, it requires that all eigenvalues lie inside the unit circle in the complex plane.

The eigenvalues of the matrix  $\mathbf{A}_d$  were computed and displayed in the following figures.

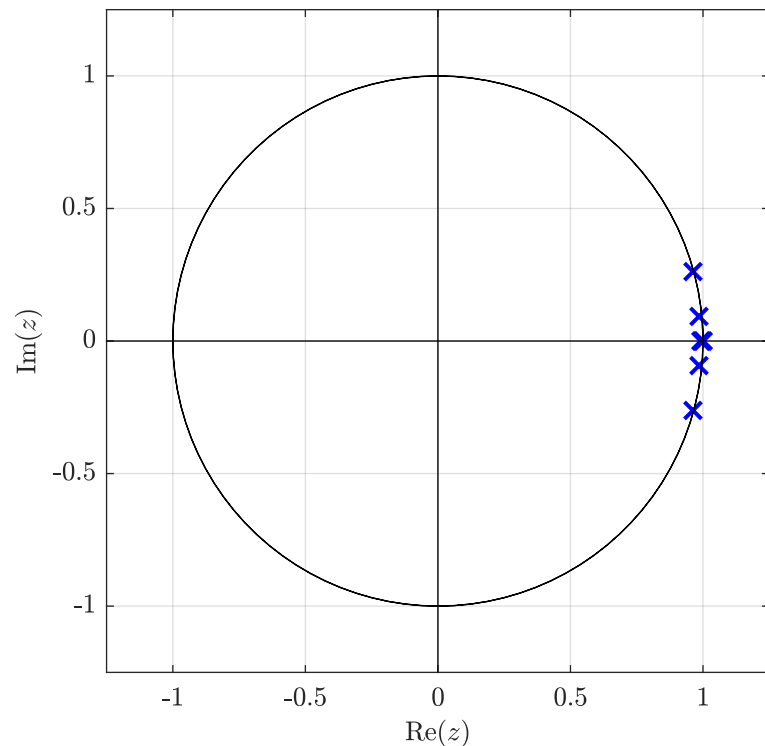


Figure 13: Eigenvalue locations of discrete-time model (24)



In the Figure 13 it can be seen that all eigenvalues lie inside the unit circle. Thus, the system is stable.

Furthermore, in the Figure 14 one can notice there are two pairs of complex-valued eigenvalues. Those eigenvalues correspond to the double pendulum which the quadrotor with a suspended payload form. The remaining four poles lie on the real axis of the complex plane with two poles at the stability boundary.

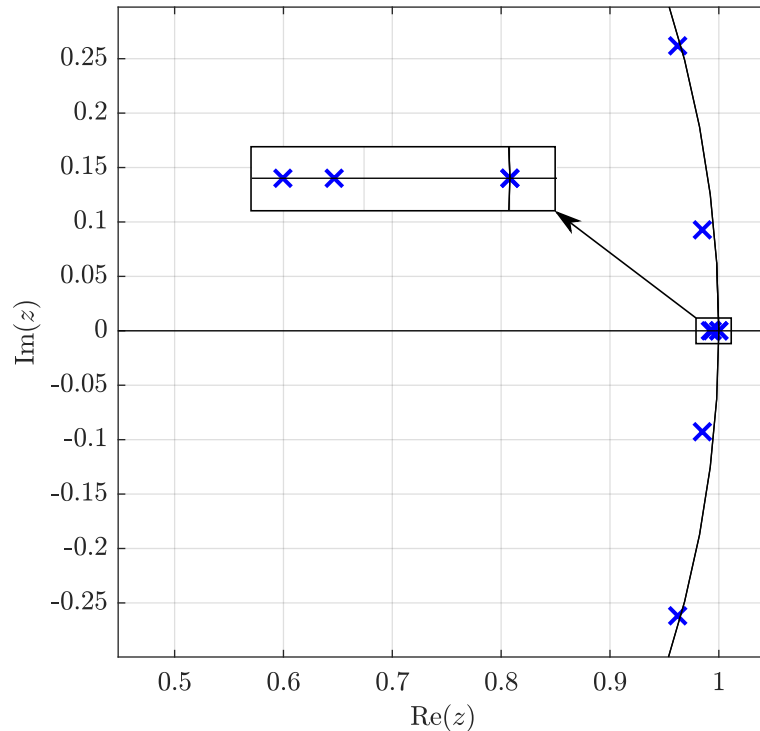


Figure 14: Zoomed eigenvalue locations of discrete-time model (24)

### 2.8.2 Reachability

In short, reachability refers to the question if it is possible to steer a system from an initial state to any other state. This property is crucial for state-space control design methods. The reachability can be found throughout the reachability matrix which is defined as

$$\mathcal{C}_n = [\mathbf{B}_d \quad \mathbf{A}_d \mathbf{B}_d \quad \dots \quad \mathbf{A}_d^{n-1} \mathbf{B}_d] \quad (27)$$

The linear system (24) is reachable if and only if

$$\text{rank}(\mathcal{C}_n) = n \quad (28)$$

The reachability matrix was constructed in MATLAB using `ctrb()` and the rank of this matrix was computed. As the result, the reachability matrix for (24) is full rank, i.e.  $\text{rank}(\mathcal{C}_8) = 8$ .

### 2.8.3 Observability

Observability is related to determining the state just from observations of inputs and outputs. It is especially essential when we are not able to observe the complete state vector, which is common in real applications. In contrast to reachability, observability is important for an observer design. The linear system (24) is observable if and only if

$$\text{rank}(\mathcal{O}_n) = n, \quad (29)$$

where the observability matrix is given

$$\mathcal{O}_n = \begin{bmatrix} \mathbf{C}_d \\ \mathbf{C}_d \mathbf{A}_d \\ \vdots \\ \mathbf{C}_d \mathbf{A}_d^{n-1} \end{bmatrix} \quad (30)$$

In the same fashion as the reachability matrix, the observability matrix was designed in MATLAB using the `obsv()` function. Also, the observability matrix is a full rank, i.e.  $\text{rank}(\mathcal{O}_8) = 8$ .

## 2.9 Control Design

### 2.9.1 State Feedback Control Design

State feedback control designs are mostly based on state-space models. The control input is shaped as a linear combination of the system states. The goal of the control law is to allow the assignment of pole locations for the desired closed-loop system behavior. The advantages of state-space control design are evident when the system to be controlled is the so-called MIMO system. Therefore, this control strategy was used as the conventional method for the control design of the quadrotor with a suspended payload.

#### State feedback

The linear state feedback is given

$$\mathbf{u}_k = -\mathbf{K}_0 \mathbf{x}_k. \quad (31)$$

If the expression above is inserted into the model (24) with assumption of zero  $\mathbf{D}_d$  matrix the closed-loop dynamics is

$$\begin{aligned} \mathbf{x}_{k+1} &= (\mathbf{A}_d - \mathbf{B}_d \mathbf{K}_0) \mathbf{x}_k \\ \mathbf{y}_k &= \mathbf{C}_d \mathbf{x}_k \end{aligned} \quad (32)$$

The dynamics of the closed-loop system is therefore defined by the matrix  $(\mathbf{A}_d - \mathbf{B}_d \mathbf{K}_0)$ . The closed-loop eigenvalues can be assigned arbitrary locations only if the open-loop system is reachable. As has been shown in the previous chapter, the discrete system is reachable. Thus, the state feedback could be designed. The corresponding block diagram to (32) is displayed in the following figure.

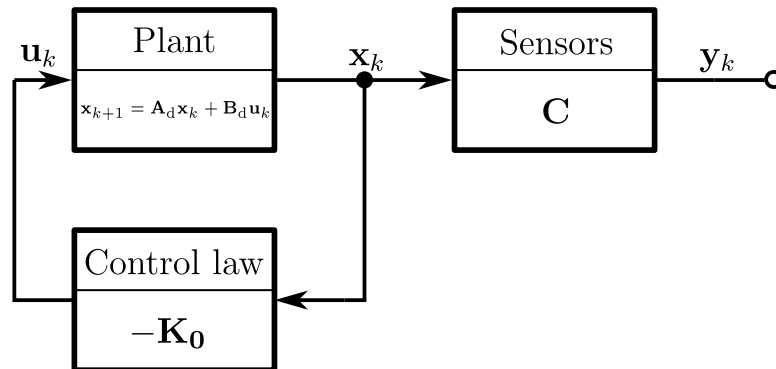


Figure 15: State-space control scheme

## Luenberger Observer

The state feedback control law presented in the previous section assumed all the state variables are accessible for feedback. Typically, not all states are measured. Moreover, in some cases, it is physically impossible to measure all the states. In the case of the quadrotor with a suspended payload, only the first four variables in the state vector are measured. The rest of the state variables have to be estimated using an observer (estimator). The fundamental observer called Luenberger observer was proposed. The Luenberger observer for (24) has form

$$\begin{aligned}\hat{\mathbf{x}}_{k+1} &= \mathbf{A}_d \hat{\mathbf{x}}_k + \mathbf{B}_d \mathbf{u}_k + \mathbf{L}(\mathbf{y}_k - \hat{\mathbf{y}}_k) \\ \hat{\mathbf{y}}_k &= \mathbf{C}_d \hat{\mathbf{x}}_k + \mathbf{D}_d \mathbf{u}_k\end{aligned}\quad (33)$$

for a constant observer gain matrix  $\mathbf{L}$ . The estimator uses the input and output sequence, together with the system model to reconstruct an estimated vector  $\hat{\mathbf{x}}_k$ . Similarly, as for state feedback, the observer dynamics are characterized by eigenvalues of the matrix  $(\mathbf{A}_d - \mathbf{L}\mathbf{C}_d)$ , which can be transformed using a suitable selection of observer gain. The eigenvalues  $(\mathbf{A}_d - \mathbf{L}\mathbf{C}_d)$  can be assigned to arbitrary locations if and only if the system is observable. In the section Analysis of discrete-time state-space, the observability matrix was determined as full rank; hence the observer could be applied.

For computing matrix  $\mathbf{L}$ , the principle of duality was used, because there is a duality relationship between the estimation and control problem.

## Output feedback

As a result of the combination of an observer, and static linear feedback the following output feedback controller is received

$$\begin{aligned}\hat{\mathbf{x}}_{k+1} &= \mathbf{A}_d \hat{\mathbf{x}}_k + \mathbf{B}_d \mathbf{u}_k + \mathbf{L}(\mathbf{y}_k - \mathbf{C}_d \hat{\mathbf{x}}_k) \\ \mathbf{u}_k &= -\mathbf{K}_0 \hat{\mathbf{x}}_k\end{aligned}\quad (34)$$

If the matrix  $\mathbf{K}_0$  is designed that  $(\mathbf{A}_d - \mathbf{B}_d \mathbf{K}_0)$  is Schur, and  $\mathbf{L}$  so that  $(\mathbf{A}_d - \mathbf{L}\mathbf{C}_d)$  is also Schur, then the system is asymptotically stable. These conclusions are done under the assumption that the model and the estimator perfectly matches the true system.

To illustrate the principle of output feedback, the block diagram is displayed in the following figure.

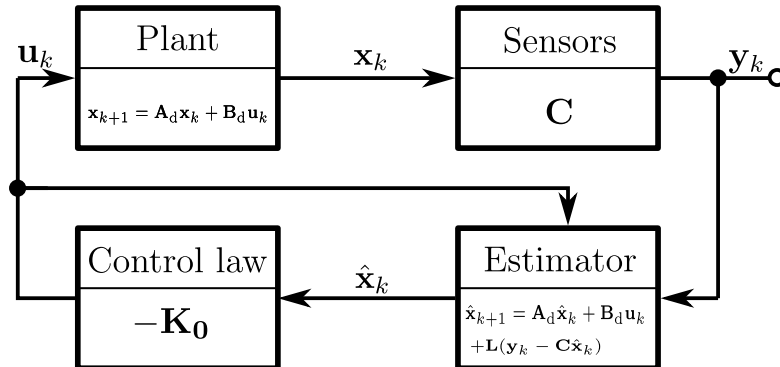


Figure 16: Estimator and controller scheme

## Integral Control

The control law and observer obtained in the previous sections are useful for stabilization. This design does not consider a reference input or reference tracking, which is crucial for the control design of a UAV. In general, there are two common techniques. The first, introduce a compensation term in feedback or feed-forward path. The compensation matrix is computed via state matrices. Therefore, this method is not robust because any changes in the plant parameters cause a nonzero error. The second method is based on introducing an integral action to obtain robust control. Augmenting the state vector with the extra integral state  $\mathbf{x}_I$  leads to the augmented state equations

$$\begin{bmatrix} \mathbf{x}_{k+1} \\ \mathbf{x}_{I_{k+1}} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{A}_d & \mathbf{0} \\ -\mathbf{C}_d & \mathbf{I} \end{bmatrix}}_{\mathbf{A}_{aug}} \begin{bmatrix} \mathbf{x}_k \\ \mathbf{x}_{I_k} \end{bmatrix} + \underbrace{\begin{bmatrix} \mathbf{B}_d \\ \mathbf{0} \end{bmatrix}}_{\mathbf{B}_{aug}} \mathbf{u}_k + \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix} r_I, \quad (35)$$

and the feedback law is

$$\mathbf{u}_k = -[\mathbf{K}_0 \quad \mathbf{K}_I] \begin{bmatrix} \mathbf{x}_k \\ \mathbf{x}_{I_k} \end{bmatrix} \quad (36)$$

or simply

$$\mathbf{u}_k = -\mathbf{K} \begin{bmatrix} \mathbf{x}_k \\ \mathbf{x}_{I_k} \end{bmatrix} \quad (37)$$

Combining the output feedback with integral action, the following block diagram is formed.

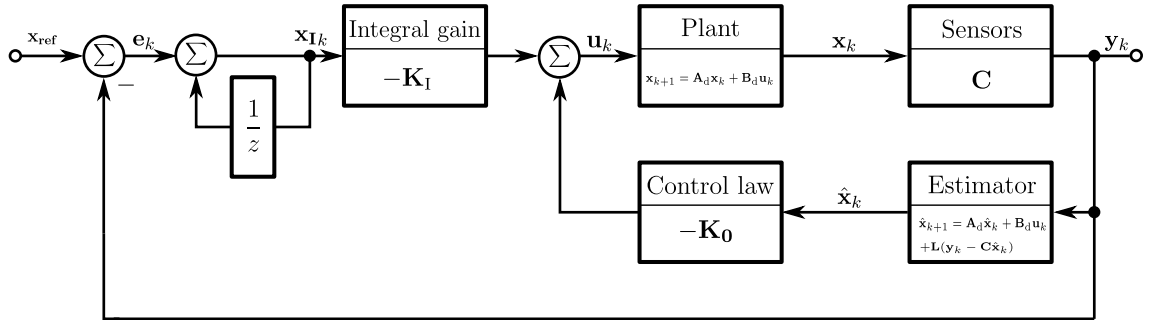


Figure 17: Output feedback control with integral action

The reachability matrix of the augmented system (35) is full rank i.e.  $\text{rank}(\mathcal{C}_{10}) = 10$ , therefore, the augmented system could be used for the control design.

## 2.9.2 Tuning of State Feedback Control

The gain matrix  $\mathbf{K}$  can be determined by several methods. In this thesis, the pole placement technique was used. In MATLAB, the pole placement is implemented via `place()` function which is based on algorithm [11]. The drawback is that none of the desired closed-loop poles may be repeated, i.e the poles must be distinct. This can be overcome by slightly moving the location of repeated poles. On the other hand, it can handle MIMO systems. The output feedback control design with integral action was tuned for three different desired poles location of the closed-loop system. The poles were selected with respect to the desired behavior of the closed-loop system.

In the first closed-loop poles selection for gain matrix  $\mathbf{K}$ , the desired poles were chosen as

$$p_{CL_1} = [0.6, 0.61, 0.9 + 0.15i, 0.9 - 0.15i, 0.9, 0.91, 0.7 + 0.2i, 0.7 - 0.2i, 0.0, 0.0] \quad (38)$$

The following poles map shows the location of closed-loop poles with respect to the location of the open-loop poles. The poles of the closed-loop system are located closer to the imaginary axis which implies a faster response. Furthermore, the two desired poles are located in the origin. Those poles have an integral character.

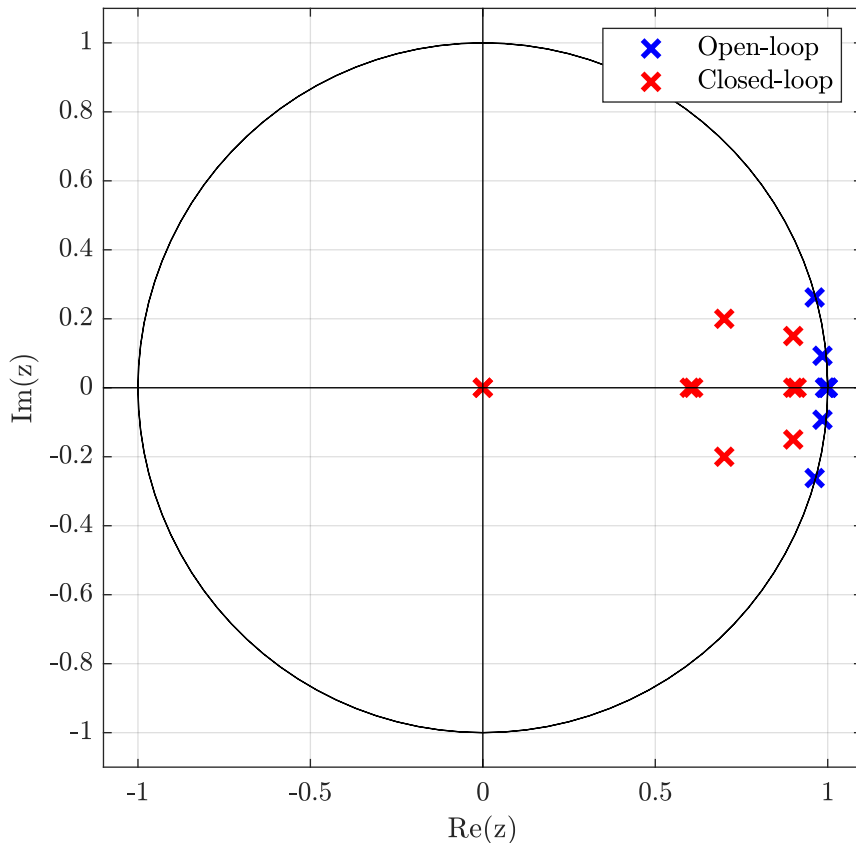


Figure 18: Poles map of Poles location 1

The desired poles location of the estimator matrix  $\mathbf{L}$  was selected five times faster than the dynamics of the closed-loop system, i.e.

$$p_{E_1} = 0.2p_{CL_1} \quad (39)$$

This ensures a faster decay of the estimator errors compared with the desired system dynamics.

In contrast to the first desired poles selection, the second poles' choice of was selected more conservatively

$$p_{CL_2} = [0.9, 0.91, 0.95+0.25i, 0.95-0.25i, 0.9, 0.91, 0.8+0.2i, 0.8-0.2i, 0.0, 0.0]. \quad (40)$$

The poles are situated further from the imaginary which implies a slower response of the closed-loop system coupled with not-so-aggressive input actions.

The desired poles location of the estimator matrix  $\mathbf{L}$  was also selected five times faster than the dynamics of the closed-loop system, i.e.

$$p_{E_2} = 0.2p_{CL_2} \quad (41)$$

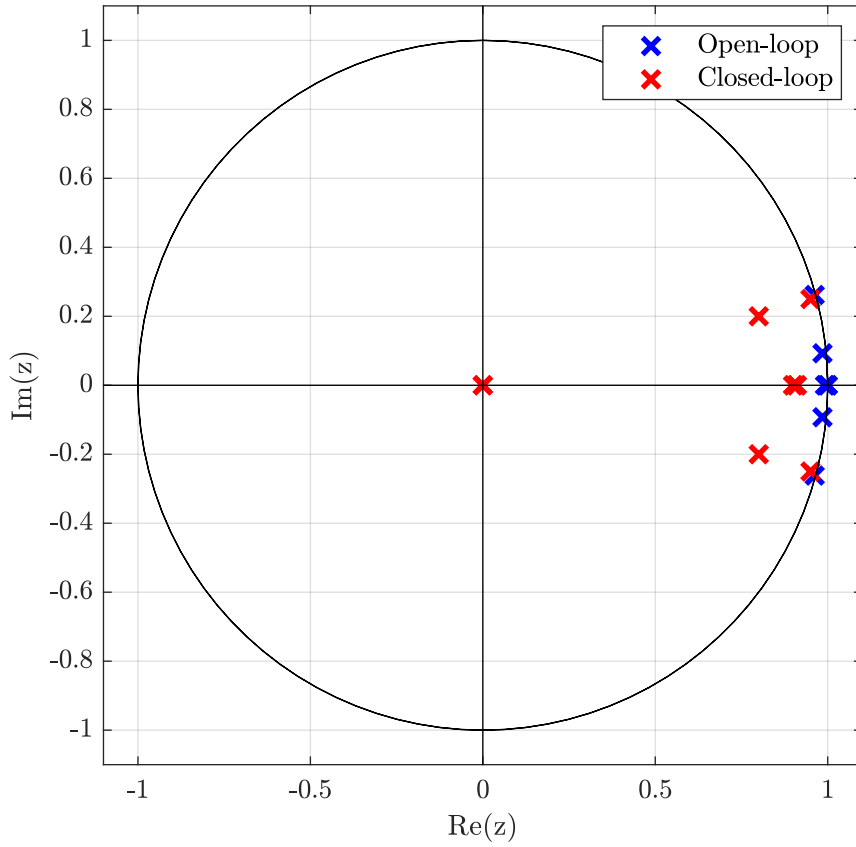


Figure 19: Poles map of Poles location 2

The last poles selection was as follows

$$p_{CL_3} = [0.7 + 0.3i, 0.7 - 0.3i, 0.9 + 0.25i, 0.9 - 0.25i, 0.9 + 0.1i, 0.9 - 0.1i, 0.8 + 0.2i, 0.8 - 0.2i, 0.0, 0.0]. \quad (42)$$

All poles were chosen as complex-conjugate except the ones which correspond to the integrator. The response of the closed-loop system with this poles location should be faster than the second pole's location. In addition, the response should be poorly damped.

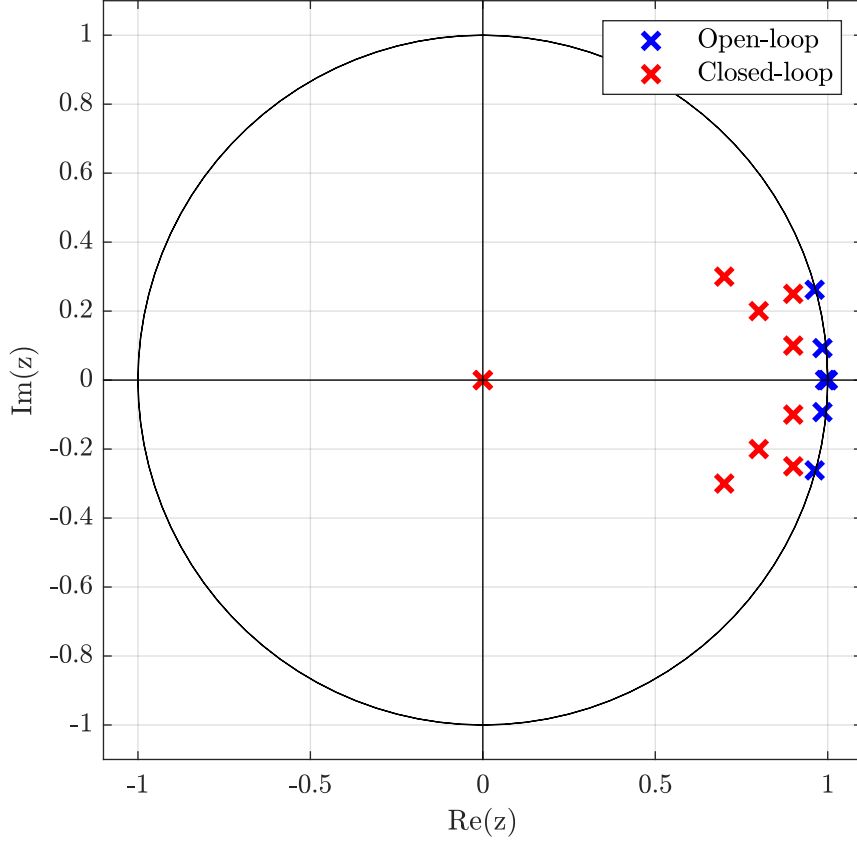


Figure 20: Poles map of Poles location 3

The desired poles location of the estimator matrix  $\mathbf{L}$  was, as in the previous cases, selected five times faster than the closed-loop system.

$$p_{E_3} = 0.2p_{CL_3} \quad (43)$$



### 2.9.3 Simulation validation of State Feedback Control Design

The block diagram of the output feedback control with integral action was implemented in MATLAB/SIMULINK. The designed control was validated on the nonlinear continuous-time model (9) for three different flight scenarios and three different poles location selections. The control's actions as well as the states' responses are displayed in the following figures.

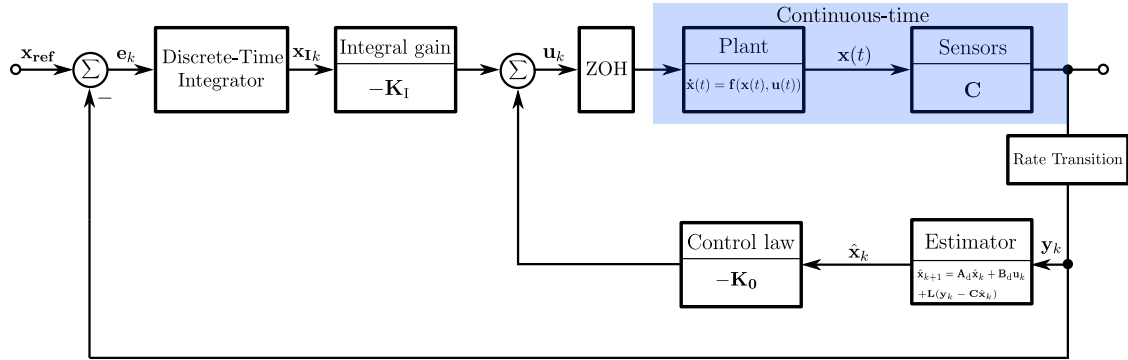


Figure 21: Scheme of the Simulink implementation of output feedback control with integral action

Scenario 1 corresponded to the situation when the trolley moves only in the horizontal direction, i.e  $\mathbf{x}_{\text{ref}} = [0.5, 0]^T$ .

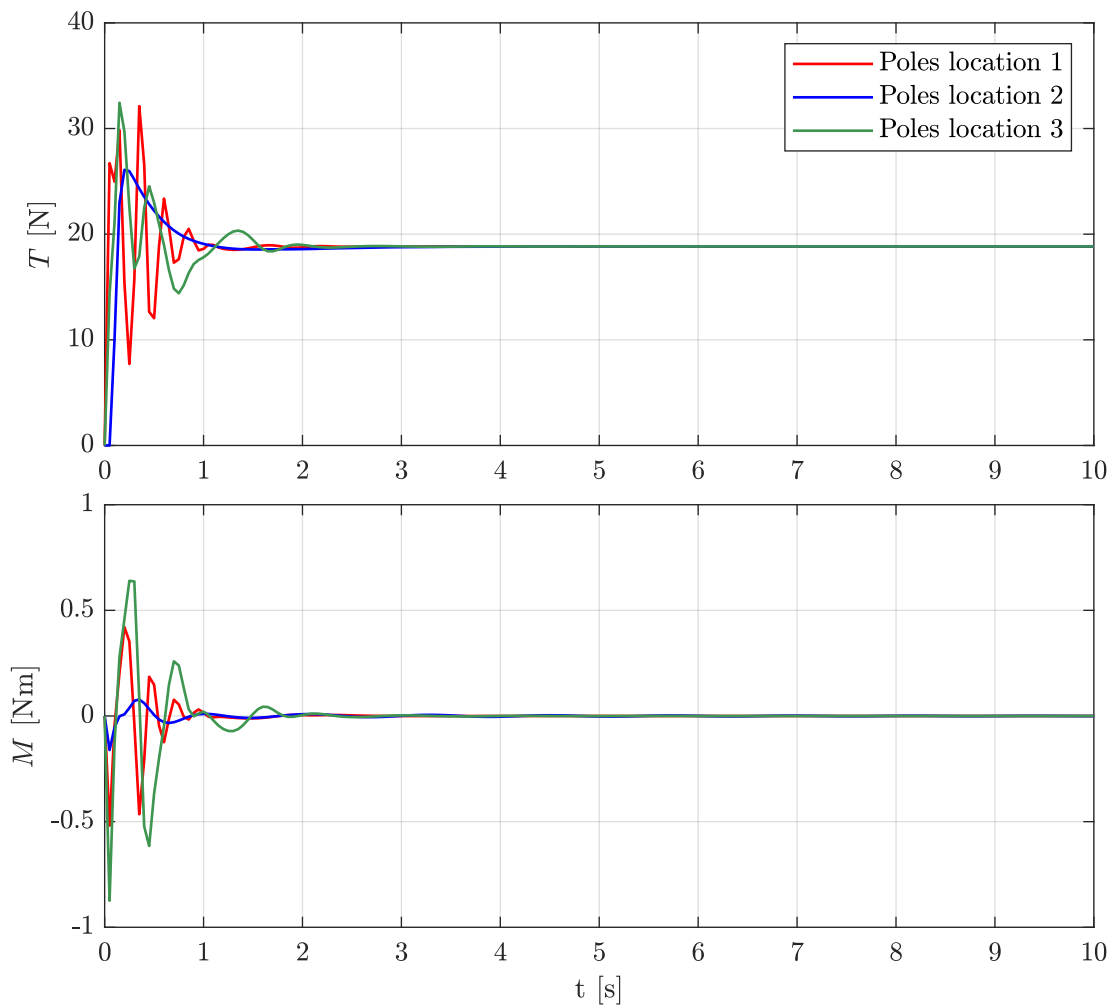


Figure 22: Control actions for scenario 1  
(MATLAB/Simulink, Ode5, Fixed-step size 0.05)

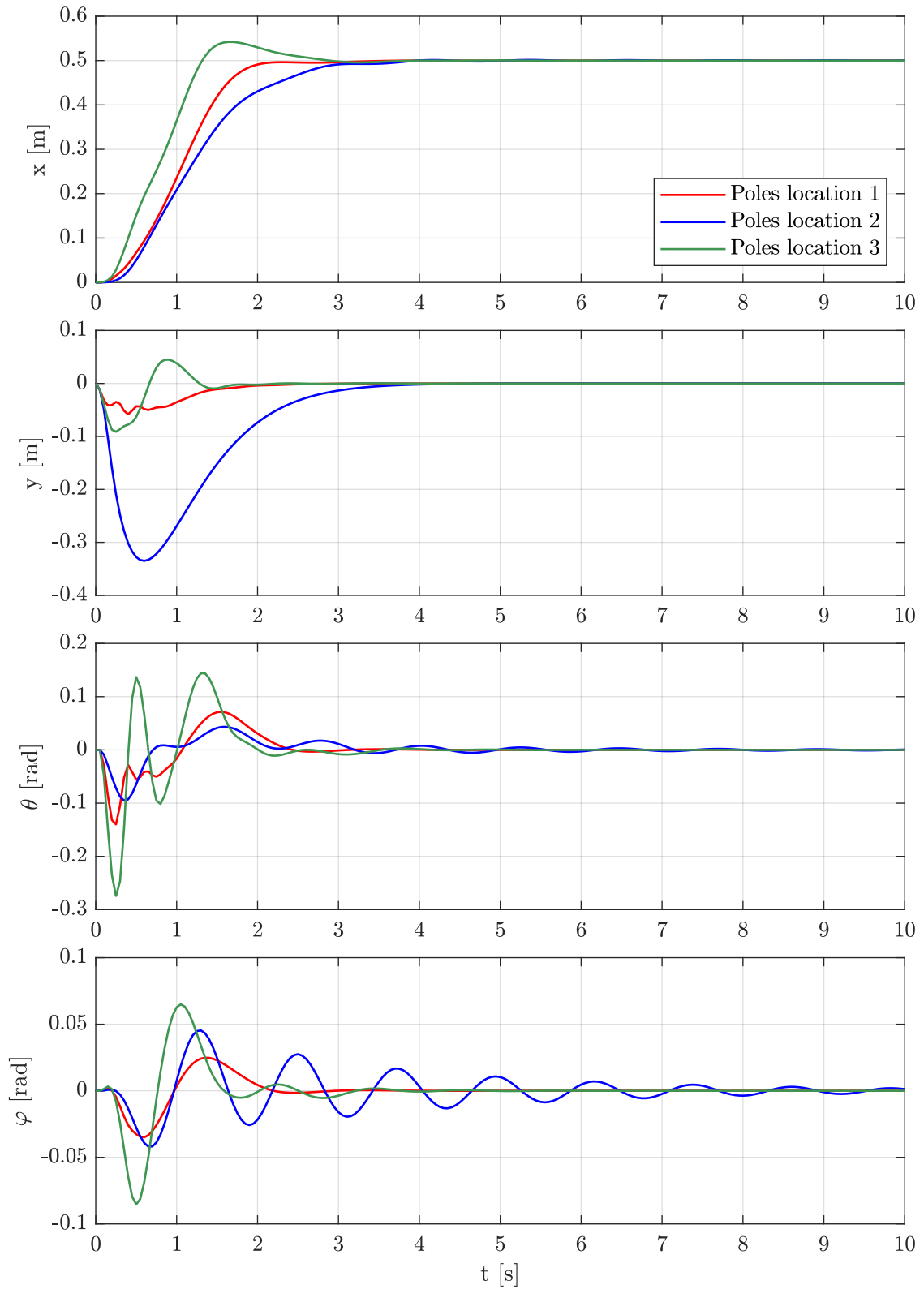


Figure 23: States response for scenario 1  
(MATLAB/Simulink, Ode5, Fixed-step size 0.05)

Scenario 2 referred to the vertical takeoff of the UAV, i.e  $\mathbf{x}_{\text{ref}} = [0.0, 0.5]^T$ .

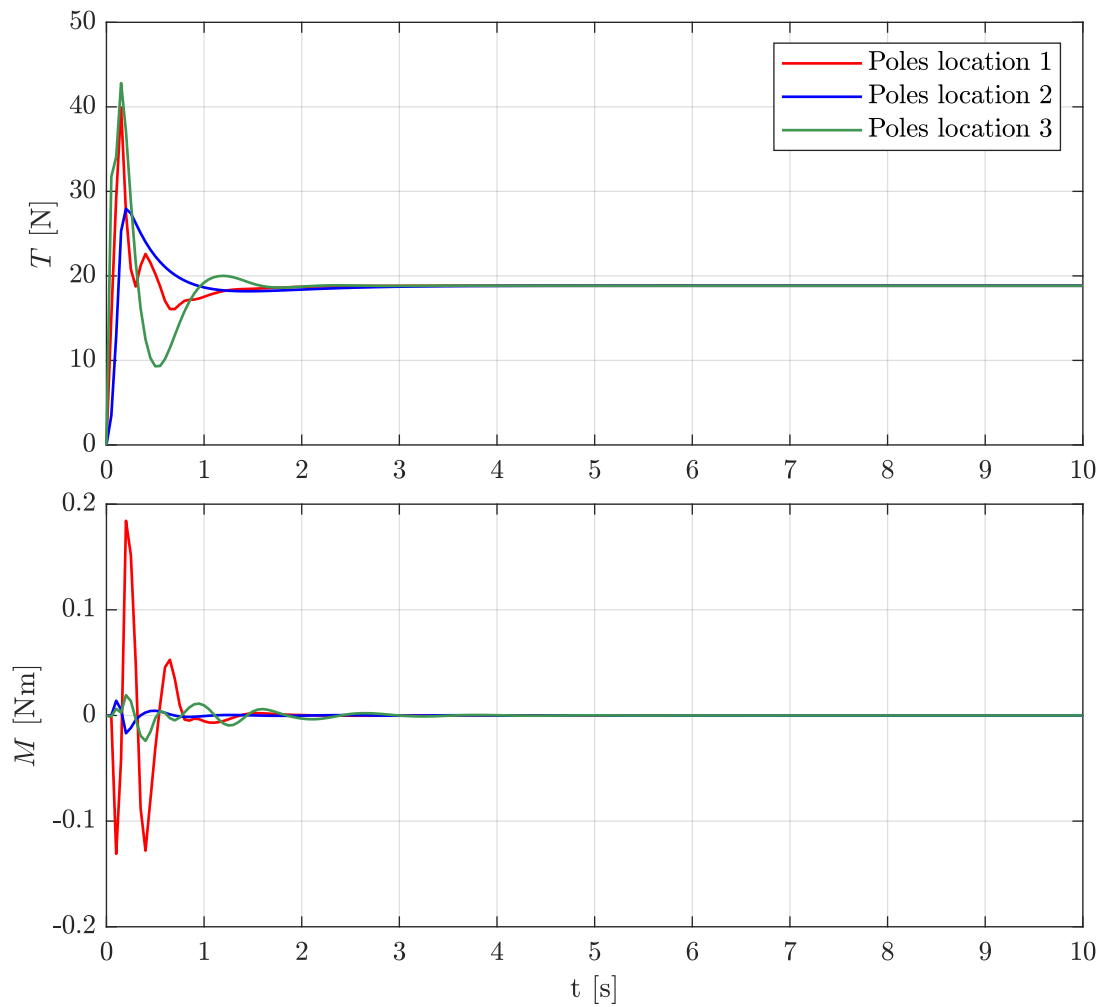


Figure 24: Control actions for scenario 2  
(MATLAB/Simulink, Ode5, Fixed-step size 0.05)

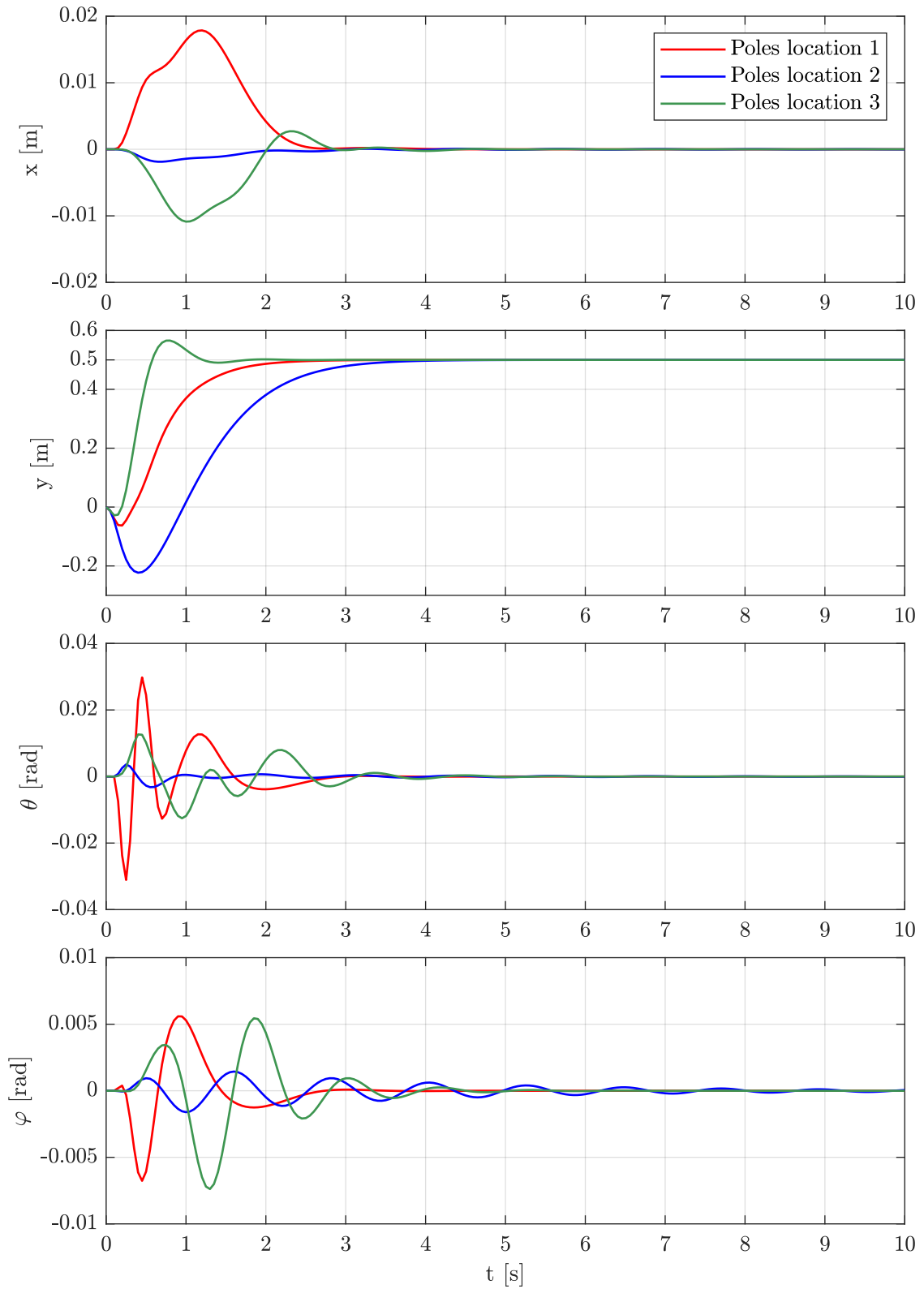


Figure 25: States response for scenario 2  
(MATLAB/Simulink, Ode5, Fixed-step size 0.05)

The last scenario combined the two previous ones. In this case, the most complicated, the UAV flew in the vertical and horizontal position at the same time, i.e.  $\mathbf{x}_{\text{ref}} = [0.5, 0.5]^T$ .

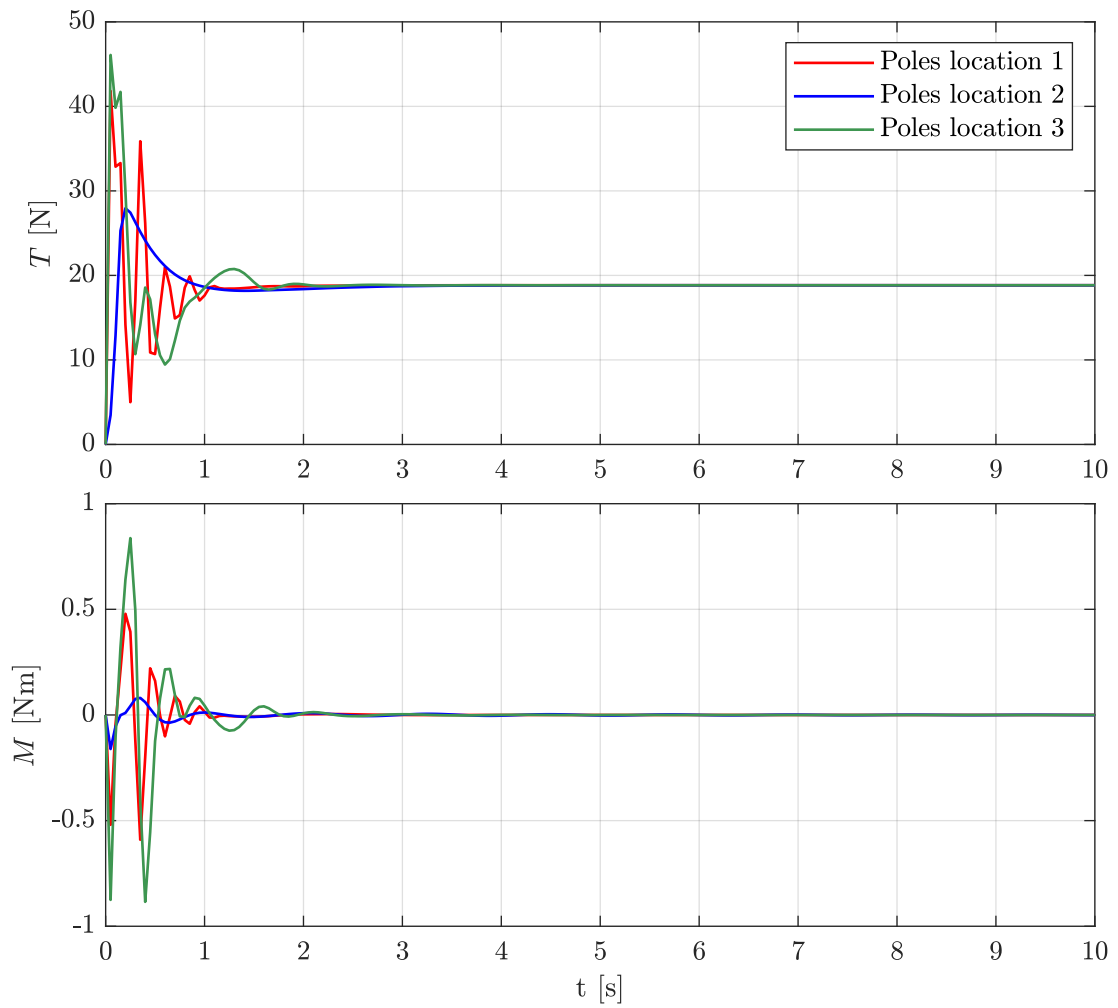


Figure 26: Control actions for scenario 3  
(MATLAB/Simulink, Ode5, Fixed-step size 0.05)

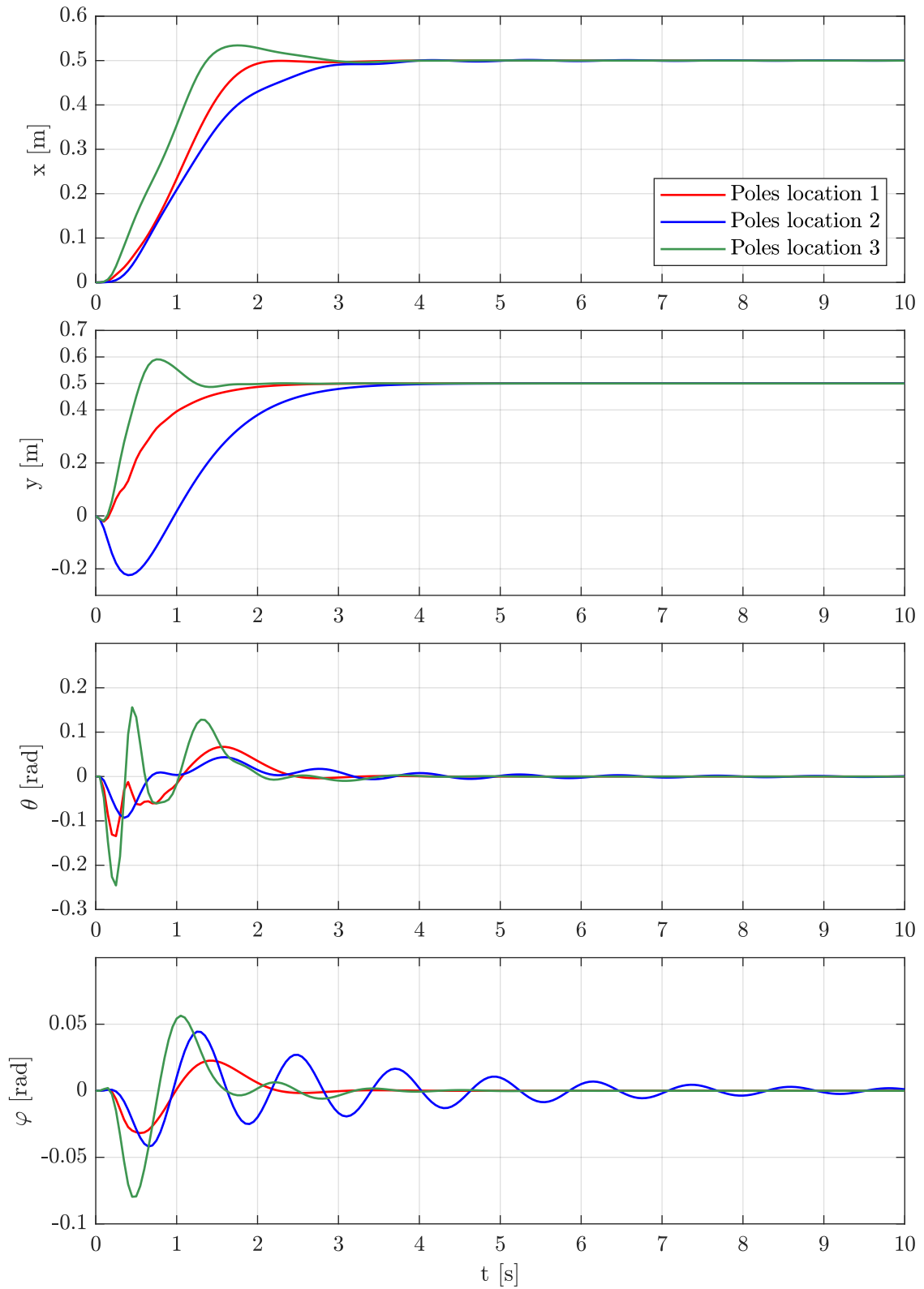


Figure 27: States response for scenario 3  
(MATLAB/Simulink, Ode5, Fixed-step size 0.05)

The predicted behavior of the closed-loop system based on the location of the poles was confirmed in the simulation.

It can be seen in previous figures for all three scenarios, that Poles location 3 has the fastest time response together with the most aggressive control actions. Furthermore, this setting has a significant overshoot in positions  $x$  and  $y$ . Besides that, for the third scenario, the pendulum's angle  $\varphi$  attenuates the fastest. On the other hand, the amplitude of the first oscillation is the biggest.

In contrast, the second poles location has the slowest time response supported by mild control actions. It is most noticeable in the second state when vertical position  $y$  drops after the start of the simulation and after approx. 0.5 seconds, it starts slowly raising towards the desired position. The first setting lies between those two previous ones. This setting behaves reasonably for the first and the third scenario. However, for the vertical takeoff, the peak of torque  $M = 0.18$  Nm produces a great peak in the quadrotor's angle as well as in the horizontal position.



## 2.9.4 Linear-quadratic Control

In the previous chapter, the feedback gain  $\mathbf{K}$  and observer gain  $\mathbf{L}$  were chosen to assign eigenvalues using pole placement. However, modern control design techniques allow finding optimal gains with respect to a specific performance criterion, robustness to model uncertainty, suppression of disturbances, and energy efficiency. This chapter presents the optimal control design for linear systems with quadratic cost functions in the state and controls.

### Linear-quadratic Regulator

In linear-quadratic control, the state feedback gains are entirely determined by the quadratic cost function. The cost function parameters become the tuning parameters in the control design.

Considering the linear discrete-time state-space model and infinite-horizon cost

$$\sum_{k=0}^{\infty} \mathbf{x}_k^T \mathbf{Q} \mathbf{x}_k + \mathbf{u}_k^T \mathbf{R} \mathbf{u}_k, \quad (44)$$

where  $\mathbf{Q} \geq \mathbf{0}$  and  $\mathbf{R} > \mathbf{0}$ . The optimal control feedback law

$$\mathbf{u}_k = -\mathbf{K}_{\text{LQR}} \mathbf{x}_k, \quad (45)$$

where static gain  $\mathbf{K}_{\text{LQR}}$

$$\mathbf{K}_{\text{LQR}} = (\mathbf{R} + \mathbf{B}_d^T \mathbf{P} \mathbf{B}_d)^{-1} \mathbf{B}_d^T \mathbf{P} \mathbf{A}_d \quad (46)$$

and  $\mathbf{P}$  is the positive semidefinite solution to the algebraic Riccati equation

$$\mathbf{P} = \mathbf{Q} + \mathbf{A}_d^T \mathbf{P} \mathbf{A}_d - \mathbf{A}_d^T \mathbf{P} \mathbf{B}_d (\mathbf{R} + \mathbf{B}_d^T \mathbf{P} \mathbf{B}_d)^{-1} \mathbf{B}_d^T \mathbf{P} \mathbf{A}_d. \quad (47)$$

In addition, if  $(\mathbf{A}_d, \mathbf{B}_d)$  is reachable and  $(\mathbf{A}_d, \mathbf{Q}^{1/2})$  is detectable, then the algebraic Riccati equation admits a unique positive semi-definite solution and the close loop is asymptotically stable.

The solution of the algebraic Riccati equation was found in MATLAB by `dlqr()` function. In addition, this function also returns the optimal feedback gain matrix  $\mathbf{K}_{\text{LQR}}$ .

### Kalman-Bucy filter

In the previous chapter on the estimator, the duality between the regulator and the estimator was mentioned. In linear quadratic control, there also exists an optimal estimator using dual quantities, a so-called linear-quadratic estimator or Kalman-Bucy filter.

Kalman-Bucy filter is an estimator for stochastic system

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{A}_d \mathbf{x}_k + \mathbf{B}_d \mathbf{u}_k + \mathbf{w}_k \\ \mathbf{y}_k &= \mathbf{C} \mathbf{x}_k + \mathbf{v}_k \end{aligned} \quad (48)$$

where the process noise  $\mathbf{w}_k$  and sensing noise  $\mathbf{v}_k$  are white Gaussian random sequences with zero mean.

The gain of the Kalman-Bucy filter  $\mathbf{L}_{\text{LQR}}$  can be computed using the dual quantity, i.e.  $\mathbf{L}_{\text{LQR}} = \mathbf{K}_{\text{LQR}}^T$ . Therefore, the gain of the Kalman-Bucy filter was computed in MATLAB using the same function as for the controller. The tuning parameters of the filter are the matrices  $\mathbf{Q}_e$  and  $\mathbf{R}_e$  which are given

$$\mathbf{Q}_e = \text{Cov}(\mathbf{w}_k), \quad \mathbf{R}_e = \text{Cov}(\mathbf{v}_k). \quad (49)$$

The higher the value of  $\mathbf{Q}_e$ , the higher the model uncertainty. Similarly, the higher the value of  $\mathbf{R}_e$ , the higher the sensing noise. Those parameters were chosen with the fact, that the filter was applied to the nonlinear system, so the process noise was high due to nonlinearities. However, the value of  $\mathbf{R}_e$  was small, because there was a high degree of certainty in measurement. Those matrices were evaluated by simulation as

$$\mathbf{Q}_e = 10\mathbf{I}, \quad \mathbf{R}_e = 0.001\mathbf{I}. \quad (50)$$

### Tuning of Linear-quadratic Controller

In linear-quadratic control, the state feedback gains are entirely determined by the quadratic cost function. The cost function parameters become the tuning parameters in the control design. The matrices  $\mathbf{Q}$  and  $\mathbf{R}$  in quadratic cost (44) become the "tuning knobs" in controller design. There are many approaches to tuning the parameters to obtain the desired controller response. In this thesis, the starting point for a design procedure was the fundamental Bryson's tuning rule combined with the tuning by simulation. Bryson's rule suggests using

$$[\bar{\mathbf{Q}}]_{ii} = \frac{1}{z_{max}^{(i)}}, \quad [\bar{\mathbf{R}}]_{ii} = \frac{1}{u_{max}^{(i)}}, \quad (51)$$

where  $z_{max}^{(i)}$  is the maximum transient error that can be admissible in performance  $z^{(i)}$ , and  $u_{max}^{(i)}$  is the maximum magnitude in control signal  $i$ .

There is also an intuitive view of the tuning of the LQR controller. It can be shown that making  $\mathbf{R}$  very small, i.e. when one does not penalize energy used by the control signal, the response becomes arbitrarily fast. This is referred to as cheap control. On the other hand, when  $\mathbf{R}$  tends to infinity, the energy used is heavily penalized, and this case is called expensive control.

Similar to the tuning of state feedback control, the linear-quadratic controller was tuned for three different settings of the tuning parameters. As it was mentioned, the "tuning knobs" for this optimal control strategy are the weight matrix  $\mathbf{Q}$  and  $\mathbf{R}$ . Three following combinations of tuning weight matrices were examined. For each setting, the pole map is displayed to see where the LQ-optimal controller placed the poles.

As the first trial, the control effort was chosen as expensive together with a low penalization on states.

$$\mathbf{Q}_1 = \text{diag}(1, 1, 1, 1, 1, 1, 1, 1, 1, 1) \quad (52)$$

$$\mathbf{R}_1 = \text{diag}(0.01, 0.01) \quad (53)$$

The corresponding poles were

$$p_{\text{CL}_1} = [0.937 + 0.245i, 0.937 - 0.245i, 0.918 + 0.149i, 0.918 - 0.149i, \\ 0.867 + 0.144i, 0.867 - 0.144i, 0.848 + 0.066i, 0.848 - 0.066i, \\ 0.743, 0.002]. \quad (54)$$

In the pole map, it can be seen that the LQR placed all the poles as complex-conjugate pairs besides the last two in (54). As the result, an oscillatory time response is expected. One can also spot that all poles are relatively far away from the origin which implies slow time response and small control actions.

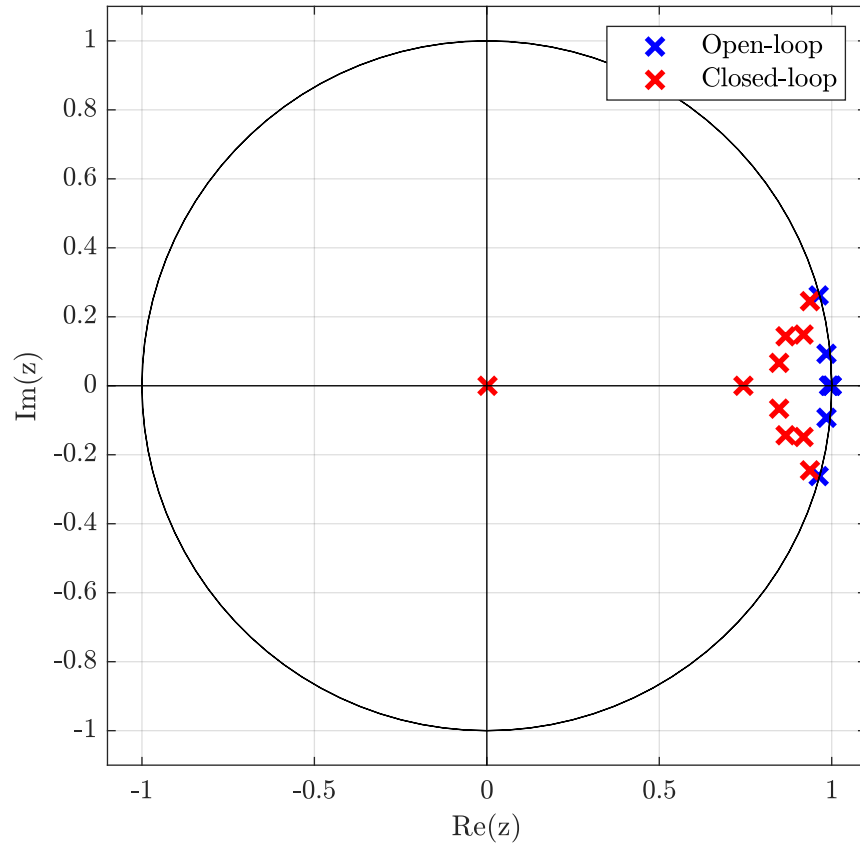


Figure 28: Poles map of Design 1

In the second tuning, the states as well as the control inputs were penalized more harshly, especially, the  $x$  and  $y$  positions were more heavily penalized together with pendulum's and quadrotor's angle.

$$\mathbf{Q}_2 = \text{diag}(20, 20, 10, 10, 1, 1, 1, 1, 50, 50) \quad (55)$$

$$\mathbf{R}_2 = \text{diag}(0.001, 0.001) \quad (56)$$

Corresponding poles were

$$p_{CL_2} = [0.95 + 0.241i, 0.95 - 0.241i, 0.866 + 0.235i, 0.866 - 0.235i, \\ 0.749 + 0.088i, 0.749 - 0.088i, 0.625 + 0.291i, 0.625 - 0.291i, \\ 0.423, 0.00]. \quad (57)$$

The consequence of the penalization can be seen in the following poles map. The poles moved closer to the origin as well as furtherer from the imaginary axis compared to the first design.

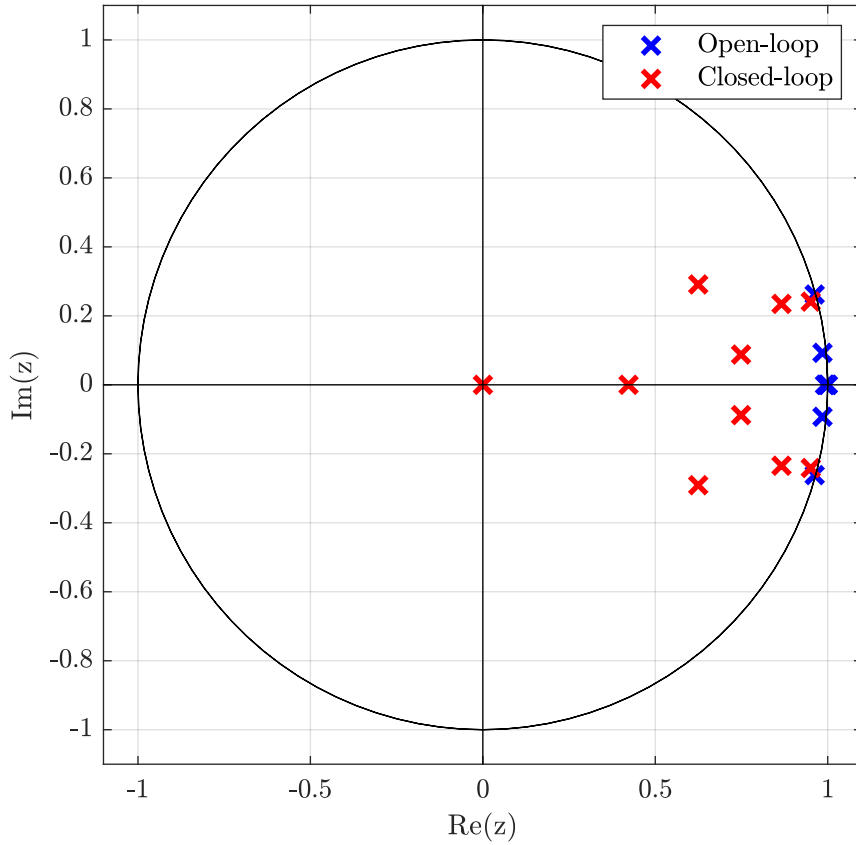


Figure 29: Poles map of Design 2

Finally, the last combination of weight matrices was suggested. In this design, the control became cheaper and states were penalized the most heavily.

$$\mathbf{Q}_3 = \text{diag}(50, 50, 20, 20, 10, 10, 10, 10, 100, 100) \quad (58)$$

$$\mathbf{R}_3 = \text{diag}(0.005, 0.005) \quad (59)$$

Corresponding poles were

$$p_{\text{CL}_3} = [0.954 + 0.242i, 0.954 - 0.242i, 0.852 + 0.246i, 0.852 - 0.246i, \\ 0.722 + 0.089i, 0.722 - 0.089i, 0.541 + 0.304i, 0.541 - 0.304i, \\ 0.323, 0.0]. \quad (60)$$

Poles map shows that the poles were moved even closer to the origin. This indicates faster time response at the expense of more control effort.

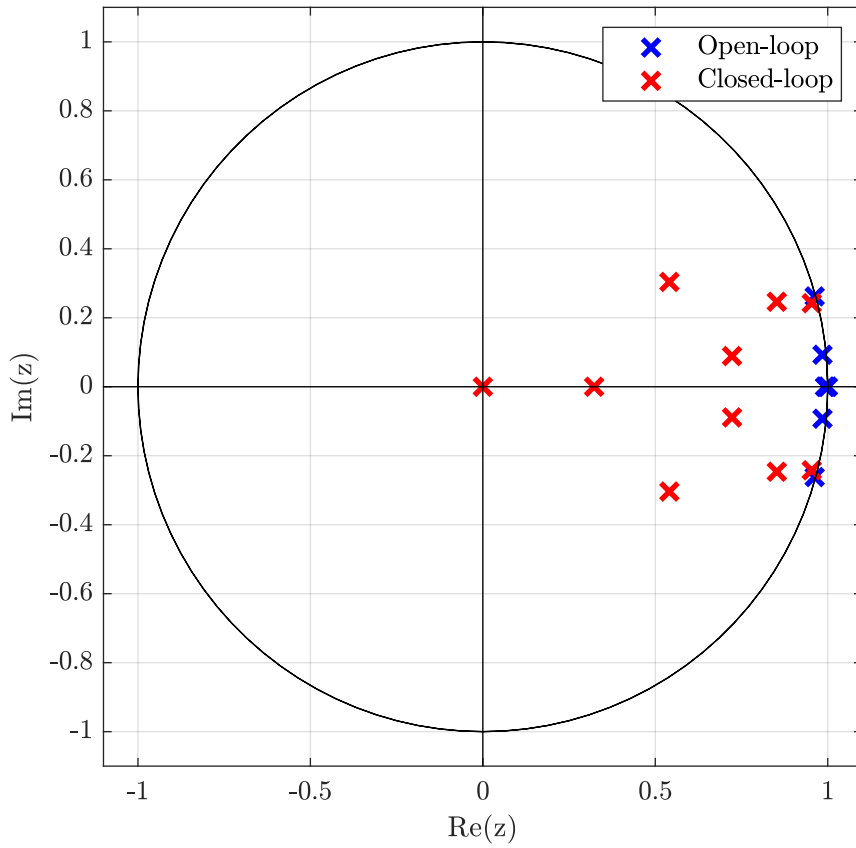


Figure 30: Poles map of Design 3

## 2.10 Simulation Validation of Linear-quadratic Control

The linear-quadratic control was validated using the same block diagram in MATLAB/Simulink as for the state feedback control. However, during the simulation, the problem with numerical instability was encountered when the values of  $\mathbf{R}$  were smaller than in  $\mathbf{R}_3$ . Apart from this, the problem with integral action appeared. The method of integration in the Discrete-Time Integrator block had to be selected as Forward Euler accumulation otherwise the simulation was numerically unstable. Several solvers were tested and the best one seemed to be Ode1be with a fixed step size.

Scenario 1

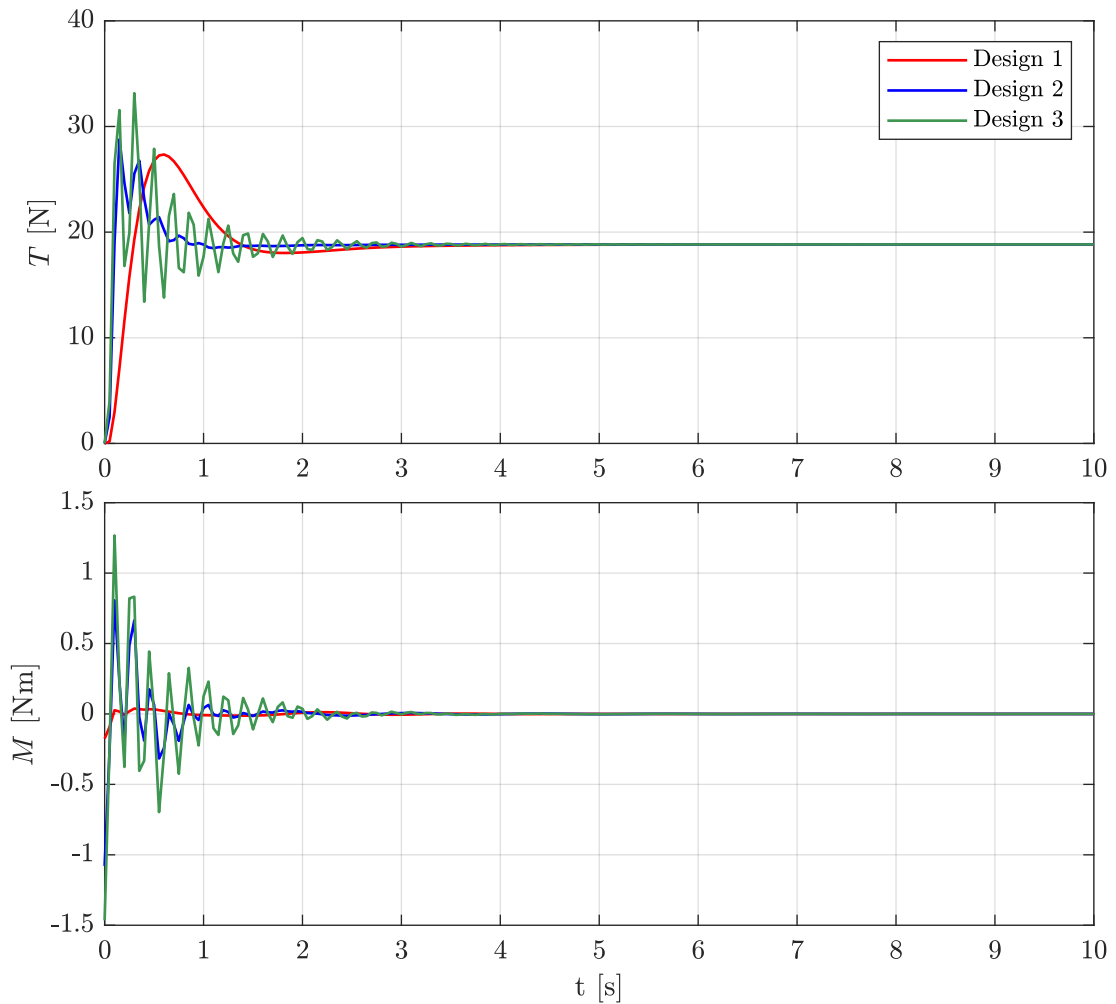


Figure 31: Control actions of Scenario 1,  
(MATLAB/Simulink, Ode1be, Fixed-step size 0.05)

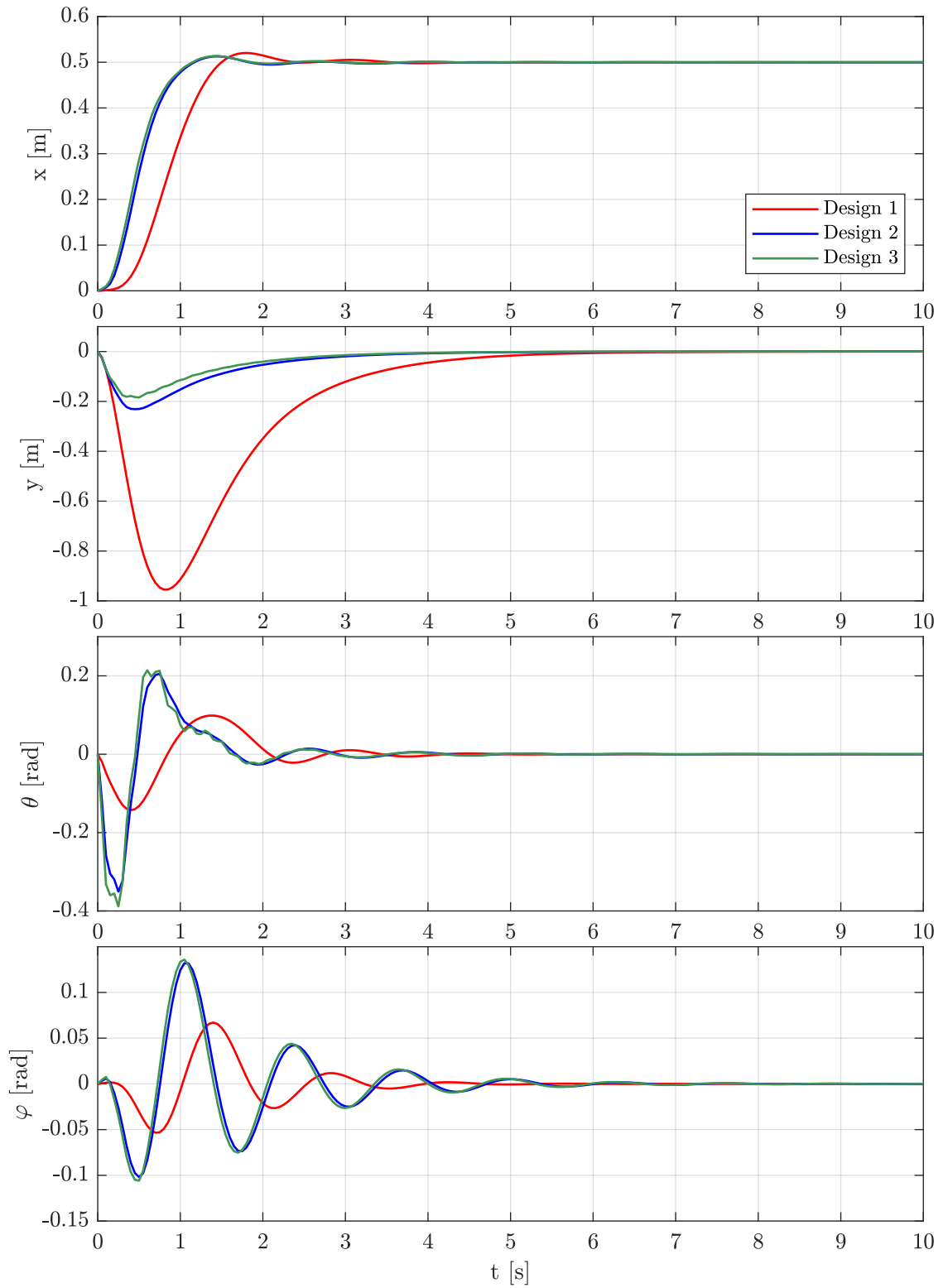


Figure 32: States response of Scenario 1,  
(MATLAB/Simulink, Ode1be, Fixed-step size 0.05)

Scenario 2

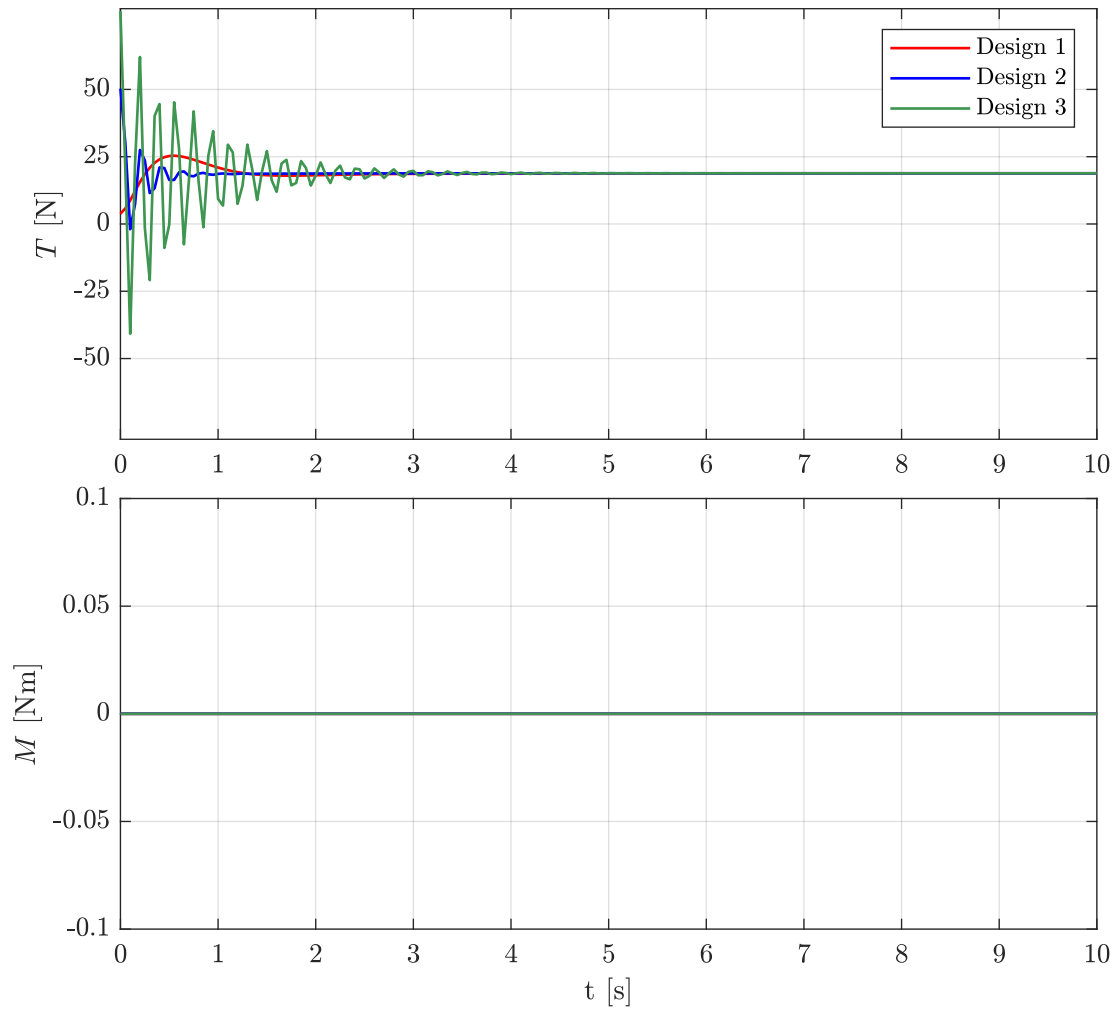


Figure 33: Control actions of Scenario 2,  
(MATLAB/Simulink, Ode1be, Fixed-step size 0.05)



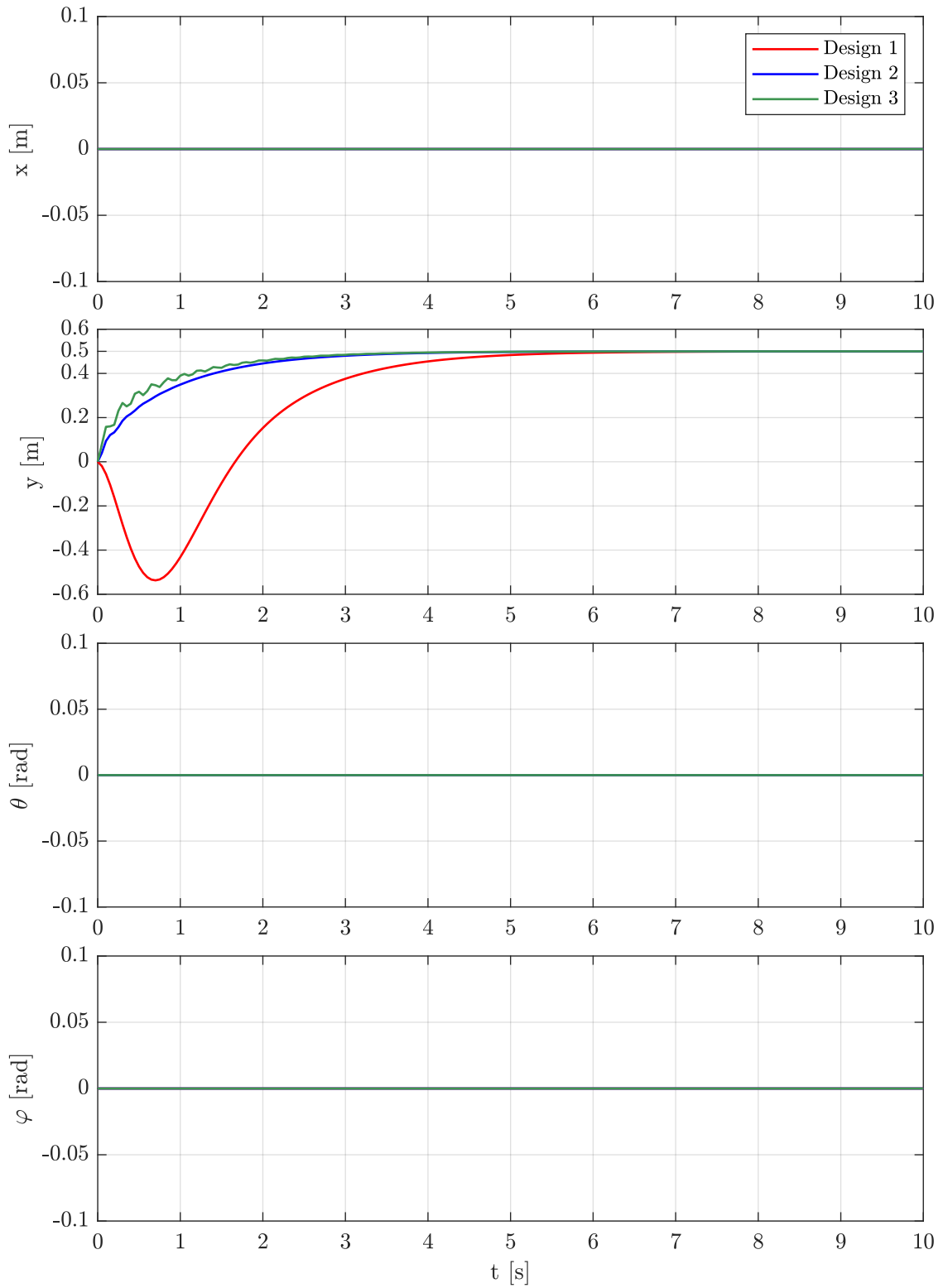


Figure 34: States response of Scenario 2,  
(MATLAB/Simulink, Ode1be, Fixed-step size 0.05)

Scenario 3

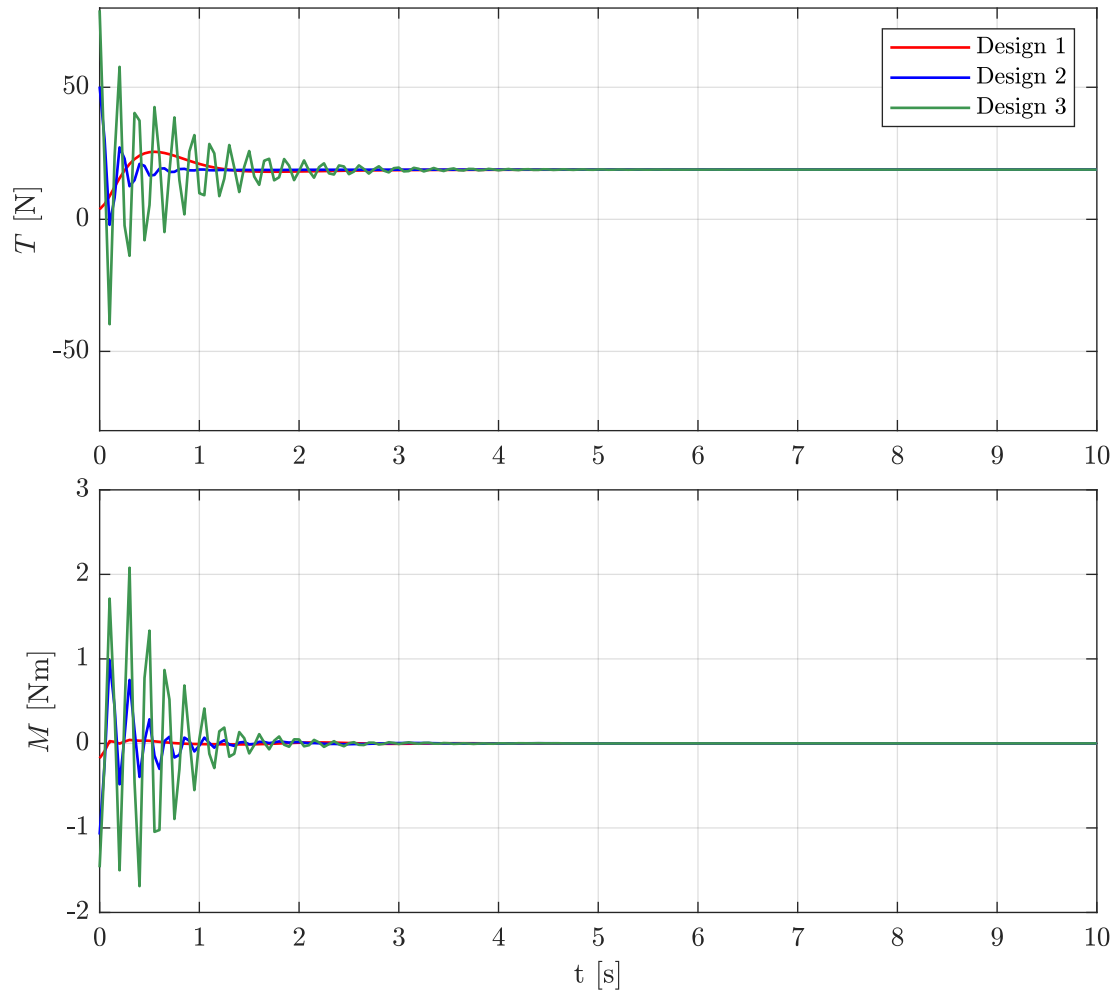


Figure 35: Control actions of Scenario 3,  
(MATLAB/Simulink, Ode1be, Fixed-step size 0.05)

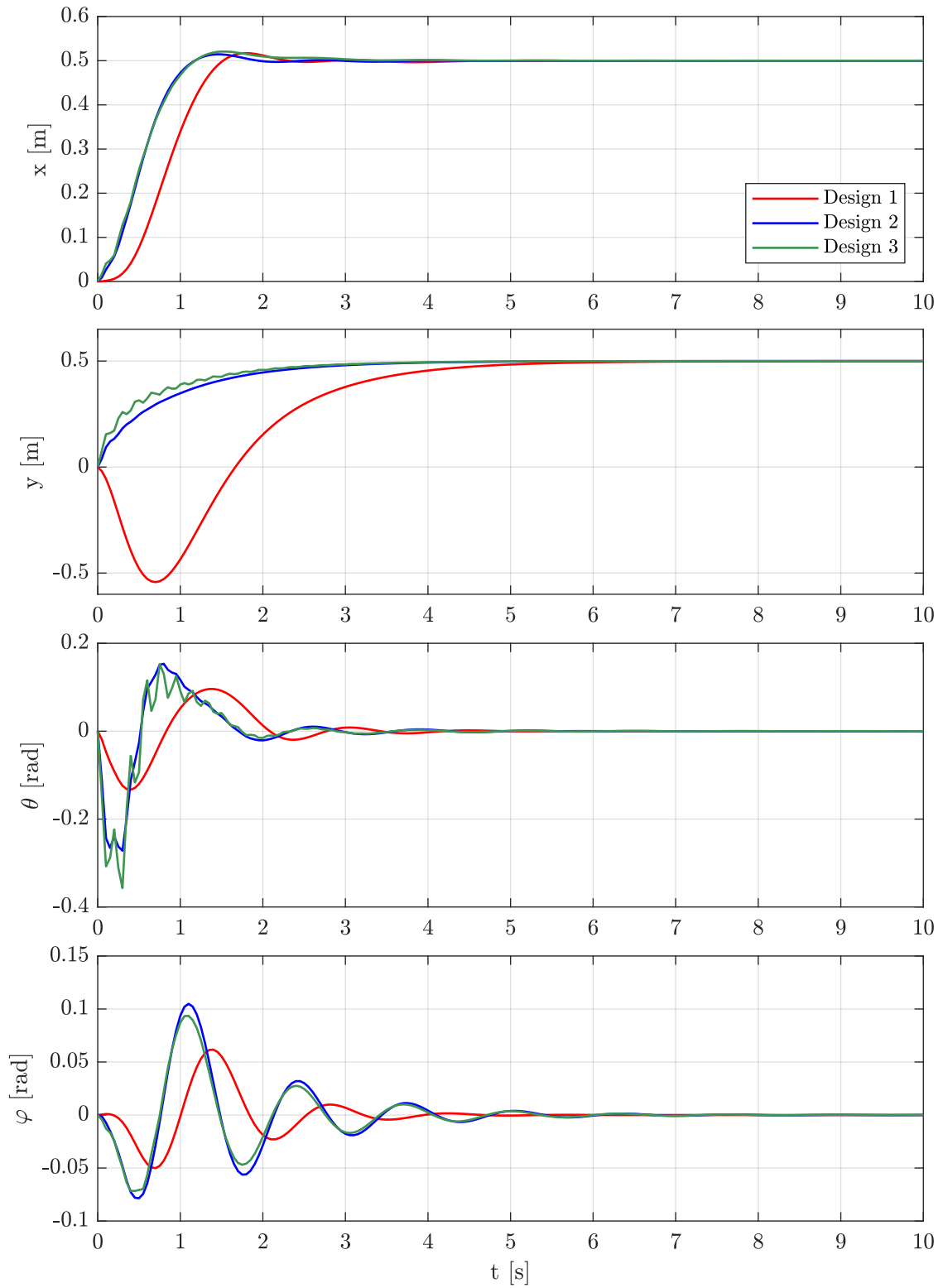


Figure 36: States response of Scenario 3, (MATLAB/Simulink, Ode1be, Fixed-step size 0.05)

The simulation results for the first design when the penalization of states is low and the control actions are expensive showed that the control inputs are the mildest together with the slowest time response. For all scenarios, it can be seen that the  $y$  position goes during the first 0.8 s significantly in the opposite direction. As a result of mild control actions, there are not many pendulum oscillations compared to the other two settings.

The other two settings are similar in terms of the response's speed and reaching the desired positions. However, the control actions for the third design are larger with a peak around 70 N which is far away from what the motors are capable of. In contrast to the pole placement design for the vertical takeoff scenario, none of the LQR designs produced pendulum oscillations.

### 2.10.1 Nonlinear Model Predictive Control

In the previous chapters, the control design was based on the linearized system represented by state space. In the following chapter, a model predictive control (MPC) strategy for the nonlinear system with constraints and a terminal cost will be developed.

Model predictive control is an advanced optimization-based control strategy. In this work, a receding-horizon control strategy was used - at each sampling time, the current state is measured, an optimal control sequence over a constant horizon  $N$  is planned, but then only the first control in the sequence is implemented. Therefore, the receding horizon control is a feedback policy since the control action at each time depends on the actual state at the time. The key idea of model predictive control is to exploit a model of the process to predict and optimize the future behavior of the system. In NMPC, a nonlinear model describing the system is used. Moreover, NMPC can make use of a nonquadratic cost function and nonlinear constraints. Nevertheless, in this thesis, the nonlinear model, quadratic cost function, and linear constraints were implemented

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k). \quad (61)$$

$$\mathbf{x}_k \in \mathcal{X}, \quad \mathbf{u}_k \in \mathcal{U}, \quad k \geq 0. \quad (62)$$

The sets  $\mathcal{U}$ ,  $\mathcal{X}$  are polyhedra described by the following linear inequalities

$$\mathcal{U} = \{\mathbf{u} : H_u \mathbf{u} \leq h_u\}, \quad \mathcal{X} = \{\mathbf{x} : H_x \mathbf{x} \leq h_x\}. \quad (63)$$

The following state and control constraints were considered

$$\mathbf{x}_{\min} \leq \mathbf{x}_k \leq \mathbf{x}_{\max}, \quad \mathbf{u}_{\min} \leq \mathbf{u}_k \leq \mathbf{u}_{\max} \quad (64)$$

with  $\mathbf{x}_{\max} = -\mathbf{x}_{\min} = [1, 1, \pi/4, \pi/4, 2, 2, \pi/2, \pi/2]^T$  and  $\mathbf{u}_{\max} = -\mathbf{u}_{\min} = [35, 1]^T$ . The state and control constraints were selected with respect to the physical system. The Nonlinear MPC problem is formally defined as an optimal control problem

$$\begin{aligned} \min_{\mathbf{u}} \quad & \sum_{k=0}^{N-1} \mathbf{e}_k^T \mathbf{Q}_{\text{mpc}} \mathbf{e}_k + \mathbf{u}_k^T \mathbf{R}_{\text{mpc}} \mathbf{u}_k + \mathbf{e}_T^T \mathbf{P}_T \mathbf{e}_T \\ \text{s.t.} \quad & \mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \\ & \mathbf{e}_k = \mathbf{x}_k - \mathbf{x}_{\text{ref}} \\ & \mathbf{x}_k \in \mathcal{X} \\ & \mathbf{u}_k \in \mathcal{U} \\ & \mathbf{x}_0 = \mathbf{x}_k \end{aligned} \quad (65)$$

In this thesis, the simplified version of nonlinear MPC was used, which does not contain terminal constraints, but does contain a terminal cost - more details can be seen in [5]. In short, the nonlinear MPC problem is stable considering only a terminal cost  $\mathbf{P}_T = \gamma \mathbf{Q}_{\text{mpc}}$ ,  $\gamma > 1$ , i.e. the terminal cost is a scaled-up version of the state cost. The related terminal set will be dependent on how large  $\gamma$  is. A huge  $\gamma$  also affects the performance of the NMPC.

## Discretization and Implementation of Optimal Control Problem

The nonlinear MPC problem (65) was implemented in MATLAB with Casadi. Casadi is an open-source tool for nonlinear optimization. [2] The optimal control problem (65) was first transcribed into a nonlinear program using the direct multiple shooting method. The main idea of this method is to decompose the system integration into short time intervals, i.e. use the system model as a state constraint at each optimization step. In contrast to the single shooting method, state variables also become decision variables in the optimization problem. Also, the multiple shooting method is more suitable for nonlinear systems optimizing over a long horizon. More details on this method can be found in [14] or [3].

The basic structure for the implementation of the direct multiple shooting method in Casadi was code [16]. The code was modified for the planar quadrotor with a suspended load. Besides that, the terminal cost was added together with the reference-input tracking scheme in the cost function. The code can be found in the attachment.

## Tuning the NMPC

A model predictive controller has a wealth of parameters to tune, such as sample time, weight matrices, horizon length, etc.

The weight matrices appearing in the cost function of (65) were tuned mainly by Bryson's rule and by simulation where the starting point was the weight matrices from the LQR controller.

In theory, if the prediction horizon is increased, the performance gets closer to that of the optimal infinite-horizon controller. However, the bigger the prediction horizon  $N$ , the larger the computational burden of the calculation of the optimal solution.

The sample time derived in the previous chapter was used.

## Simulation Validation of NMPC

The NMPC controller was simulated for three different selections of the prediction horizon and  $\mathbf{Q}_{\text{mpc}}, \mathbf{R}_{\text{mpc}}, \mathbf{P}_{\text{T}}$ , namely

$$\begin{aligned} N &= \{4, 8, 30\}, \\ \mathbf{Q}_{\text{mpc}} &= \text{diag}(100, 100, 100, 100, 10, 10, 10, 10), \\ \mathbf{R}_{\text{mpc}} &= \text{diag}(0.001, 0.001), \\ \mathbf{P}_{\text{T}} &= 20\mathbf{Q}_{\text{mpc}}. \end{aligned} \tag{66}$$

And also for three different matrices  $\mathbf{Q}_{\text{mpc}}$

$$\begin{aligned} \mathbf{Q}_{\text{mpc}1} &= \text{diag}(15, 15, 10, 10, 10, 10, 10, 10), \\ \mathbf{Q}_{\text{mpc}2} &= \text{diag}(40, 40, 40, 40, 10, 10, 10, 10), \\ \mathbf{Q}_{\text{mpc}3} &= \text{diag}(80, 80, 200, 200, 20, 20, 20, 20) \\ N &= 12, \\ \mathbf{R}_{\text{mpc}} &= \text{diag}(0.001, 0.001), \\ \mathbf{P}_{\text{T}} &= 20\mathbf{Q}_{\text{mpc}_i}. \end{aligned} \tag{67}$$

Those combinations of tuning parameters were simulated for a scenario when the trolley moves at the same time in both directions.

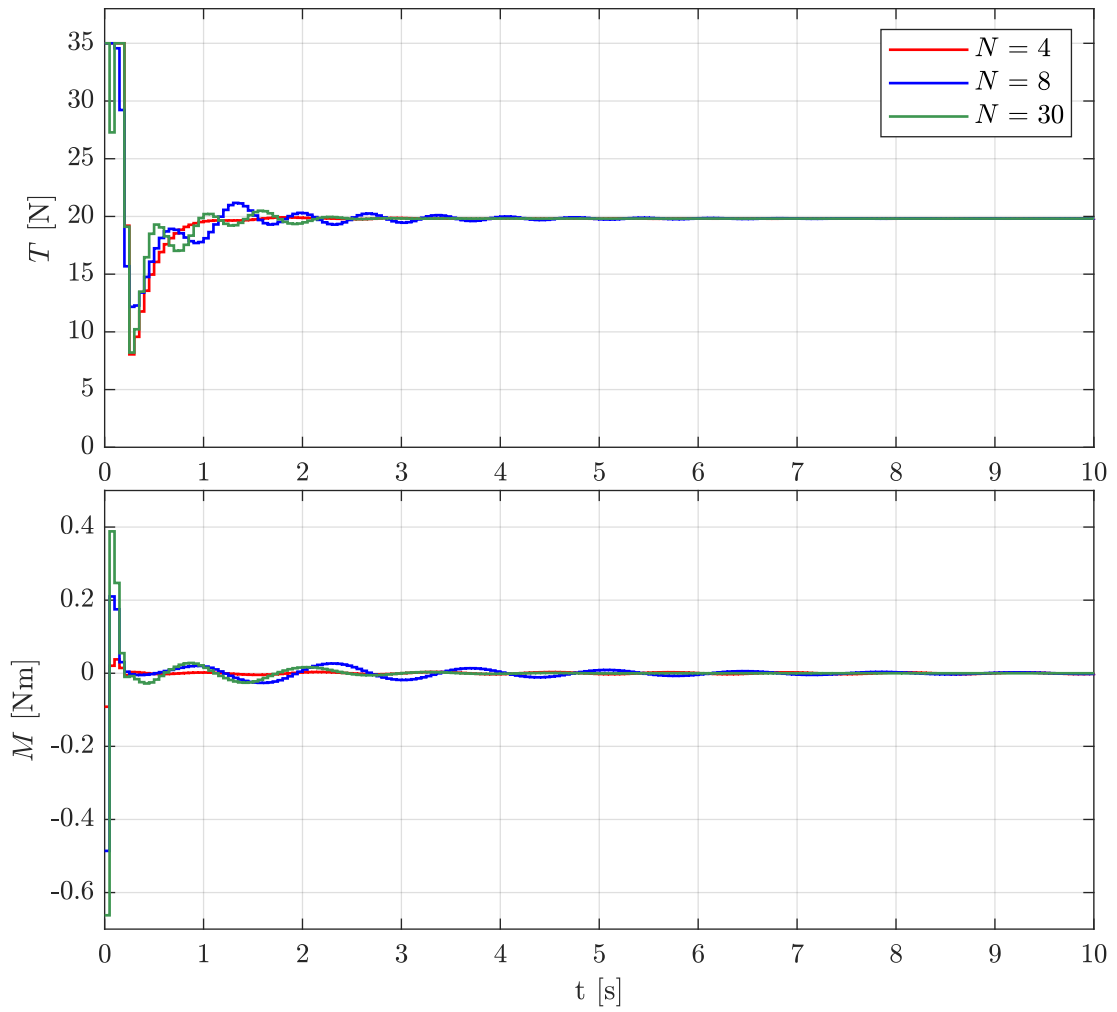


Figure 37: Control actions of NMPC,  $N = \{4, 8, 30\}$

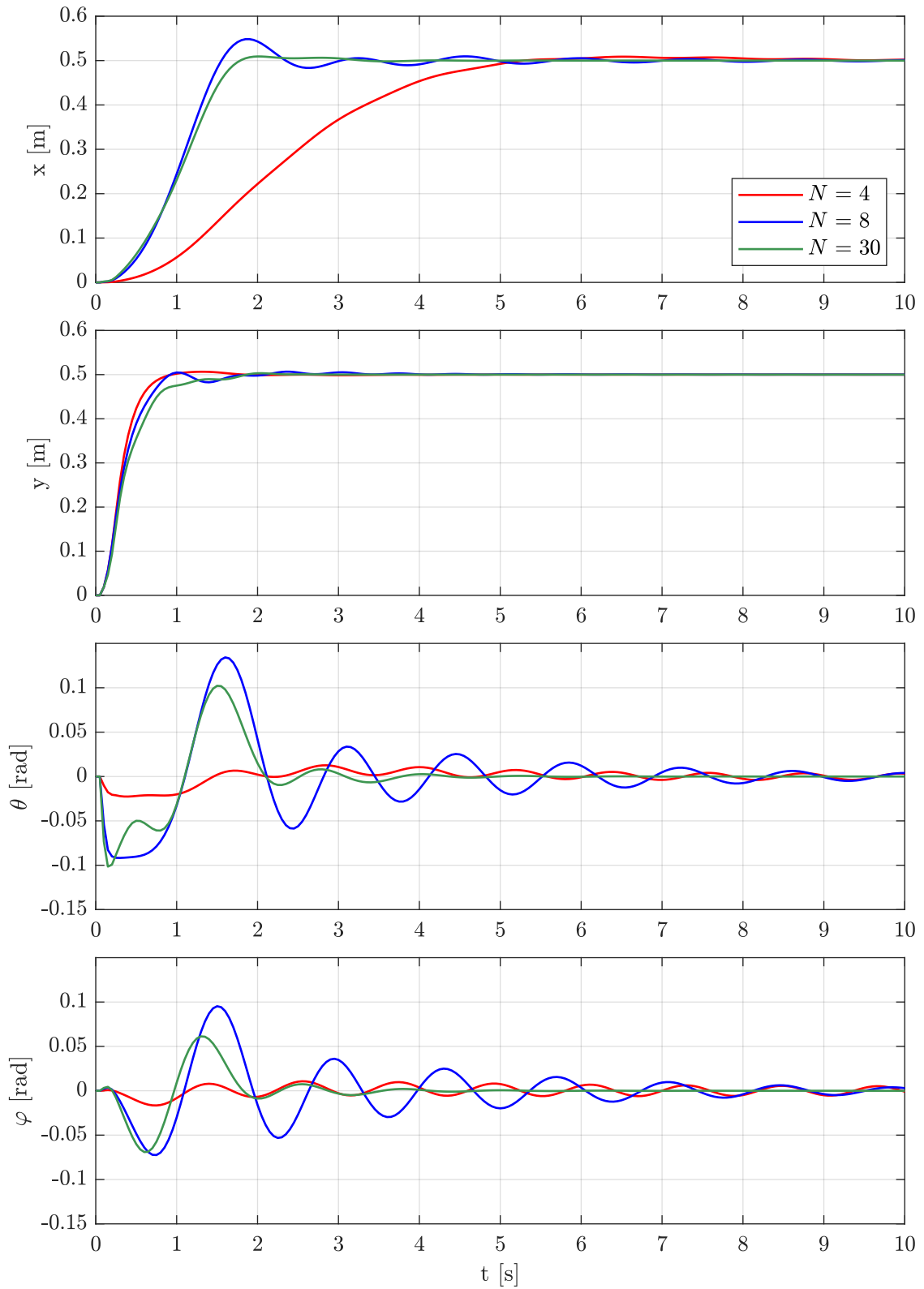


Figure 38: States response of NMPC,  $N = \{4, 8, 30\}$



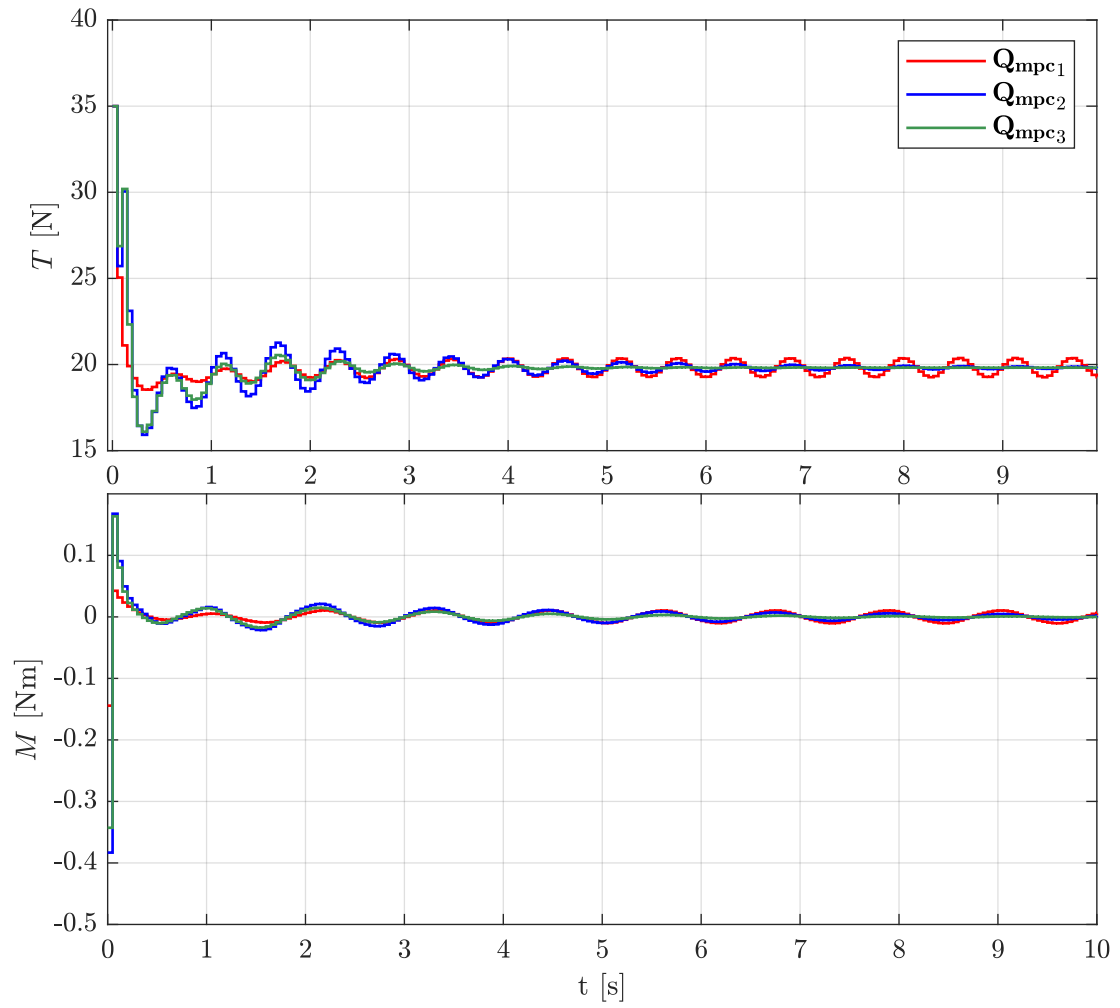


Figure 39: Control actions of NMPC for  $Q_{mpc1}$ ,  $Q_{mpc2}$ ,  $Q_{mpc3}$

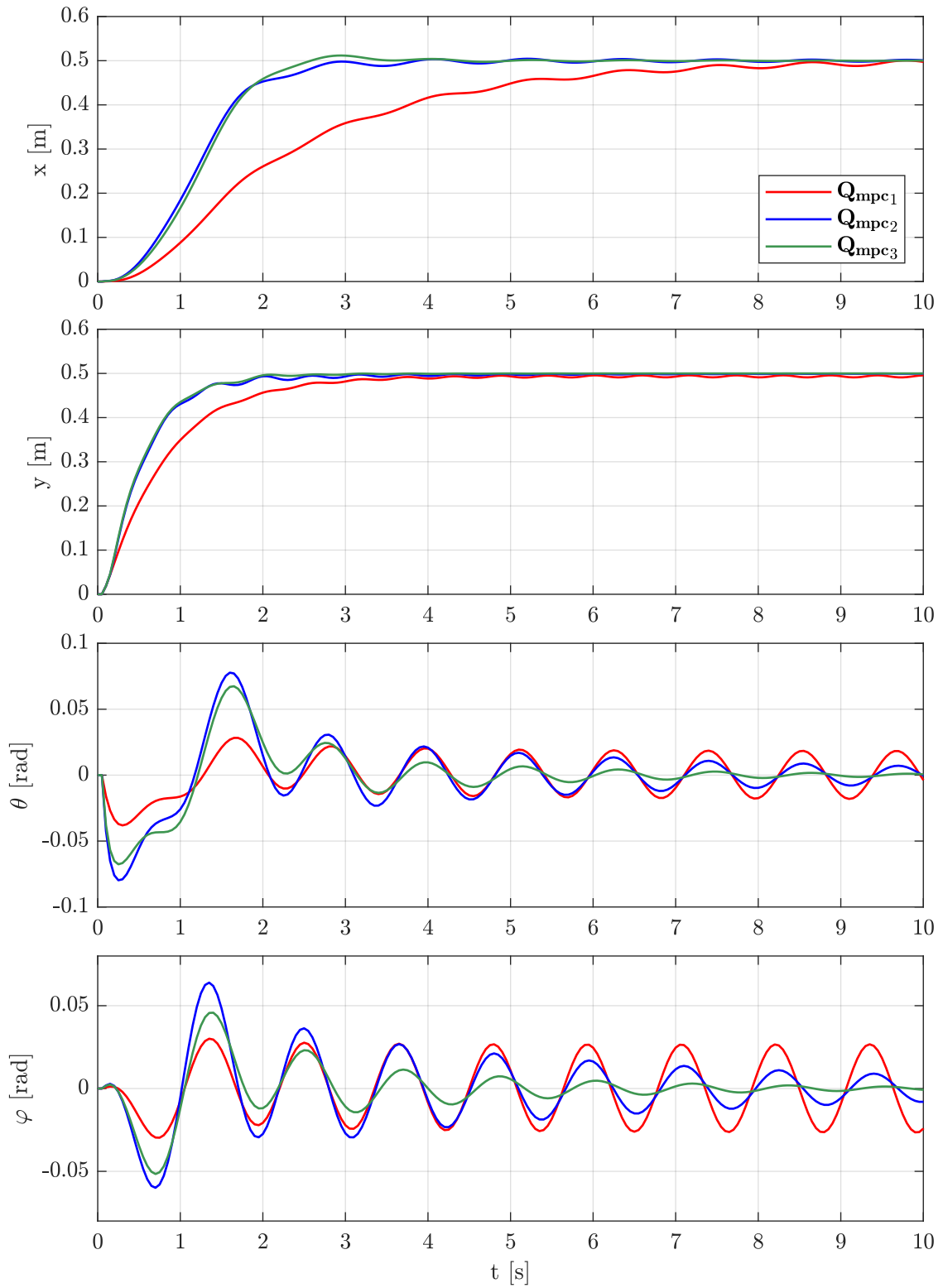


Figure 40: States response of NMPC for  $Q_{mpc1}$ ,  $Q_{mpc2}$ ,  $Q_{mpc3}$

From the simulation results, one can say that the longer prediction horizon resulted in a better time response in terms of time to reach the desired position. On the other hand, the shortest prediction horizon caused a slow response in the horizontal position with the fastest response in the vertical position together with small pendulum oscillations. Consequently, one can say that the faster response in the  $x$  direction resulted in higher pendulum oscillations.

## 2.11 Comparison of the Control Strategies

Lastly and most importantly, the control strategies were compared. Output feedback, linear-quadratic control, and NMPC were compared in simulation for the third flight scenario. Namely, the first poles' location for output feedback, the second tuning design for linear-quadratic control, and for the NMPC the tuning parameters were given by (66) for  $N = 30$ .

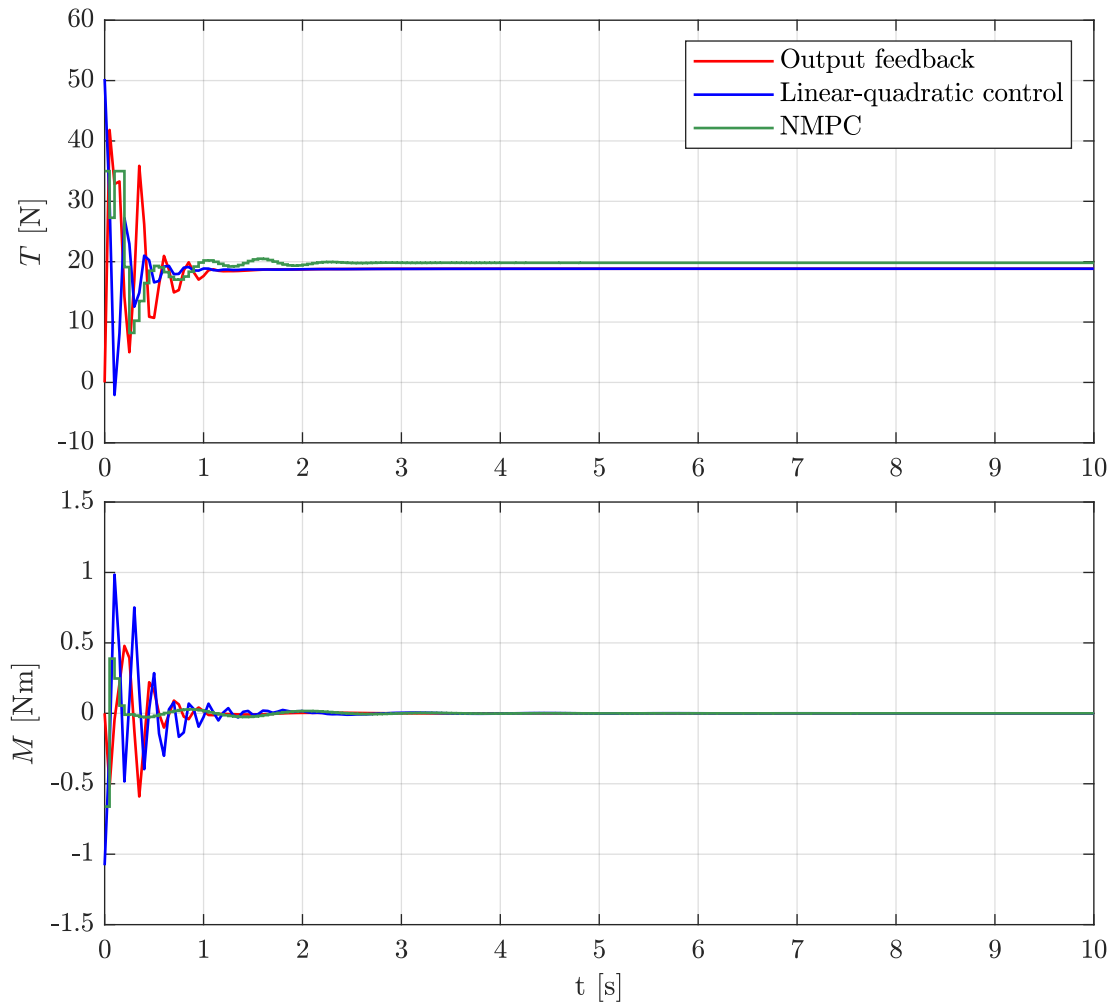


Figure 41: Control inputs comparison for the control strategies

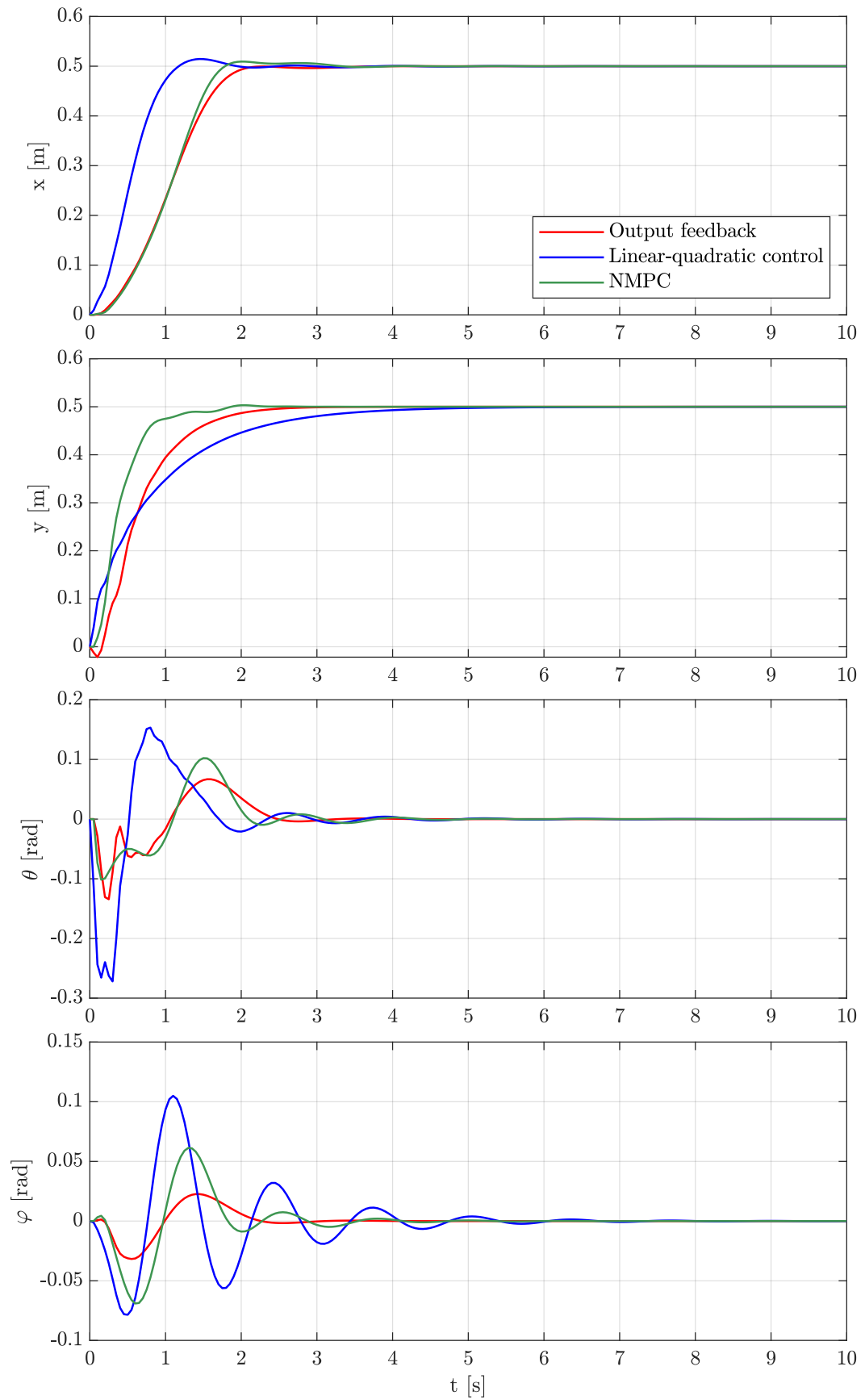


Figure 42: States response comparison for the control strategies

The linear-quadratic control resulted in the fastest time response in the  $x$  direction and the slowest response in  $y$ . For this reason, the pendulum oscillations were greatest. The output feedback performed the slowest response in the horizontal direction coupled with the smallest oscillations in the  $\varphi$ . The NMPC achieved the fastest response in the vertical motion with a reasonable damping effect on the pendulum. Nevertheless, the control actions are incomparable. Whereas the NMPC complies with the constraints with a maximum in thrust  $T = 35$  N and torque  $M = 0.4$  Nm, the linear-quadratic control reached the thrust peak of  $T = 50$  N and torque peak  $M = 1$  Nm. Between those two was output feedback with a maximum in thrust  $T = 42$  N and torque  $M = 0.5$  Nm. It has to be mentioned that it is difficult to compare those three control methods. Especially, the NMPC and output feedback control, because they are based on a different approaches.

## 2.12 Validation on the laboratory setup

The designed algorithms, specifically, output feedback control with integral action and linear-quadratic control were tested on the experimental setup. However, the trolley was too heavy together with high friction coefficient  $c_y$ , and the control actions produced by the motors were large which resulted in a big movement beyond the physical restrictions of the experimental setup.

# Conclusion and Future Work

## Conclusion

The aim of this master's thesis was an introduction to optimal control of a planar UAV model with a suspended load. Further, the creation of a simulation model of the planar UAV for the purpose of the control design as well as the optimal control design and its comparison with a conventional control method. The goal was also the design of an experimental setup and experimental validation of the algorithms.

In the thesis, several optimal control methods of a UAV with a suspended load were listed. Then, the experimental setup for testing control algorithms with a suspended load was designed together with the sensor, actuator, and control unit implementation. After, the setup was identified. The nonlinear mathematical model of the experimental setup was developed. Moreover, this model was linearized and discretized for the purpose of control design. Those models were implemented in MATLAB/Simulink and verified. Then three control methods were designed. First, the conventional output feedback control with integral action and Luenberger observer. Second, the linear-quadratic controller together with the Kalman-Bucy filter. Lastly, nonlinear model predictive control. Those methods were implemented and simulated for three different flight scenarios. Finally, the control algorithms were compared.

The linear-quadratic control resulted in the fastest time response in the horizontal direction and the slowest response in the vertical position. The output feedback performed the slowest response in the horizontal direction coupled with the smallest pendulum oscillations. Nonlinear model predictive control achieved the fastest response in the vertical motion with a reasonable damping effect on the pendulum. Nonetheless, the control actions are incomparable. Whereas the NMPC complies with the constraints with a maximum in thrust  $T = 35$  N and torque  $M = 0.4$  Nm, the linear-quadratic control actions reached the thrust peak of  $T = 50$  N and torque peak  $M = 1$  Nm. Between those two was output feedback with a maximum thrust  $T = 42$  N and torque  $M = 0.5$  Nm.

## Future Work

Improvement in experimental setup design and experimental validation has been left for the future. Future work concerns decreasing the weight of the trolley with the quadrotor as well as the deeper identification.

## Appendix A

$$\mathbf{M}(\mathbf{x}(t)) = \begin{bmatrix} m_1 + m_2 + m_3 + m_4 & 0 & a_2 m_2 \cos(\theta) + a_3 m_1 \cos(\theta) & l m_1 \cos(\varphi) \\ 0 & m_1 + m_2 + m_3 & \sin(\theta) (a_2 m_2 + a_3 m_1) & l m_1 \sin(\varphi) \\ \cos(\theta) (a_2 m_2 + a_3 m_1) & \sin(\theta) (a_2 m_2 + a_3 m_1) & m_2 a_2^2 + m_1 a_3^2 + \mathbf{I} s_2 & a_3 l m_1 \cos(\theta - \varphi) \\ l m_1 \cos(\varphi) & l m_1 \sin(\varphi) & a_3 l m_1 \cos(\theta - \varphi) & m_1 l^2 + \mathbf{I} s_3 \end{bmatrix}$$

$$\mathbf{C} = \begin{bmatrix} c_x & 0 & 0 & 0 \\ 0 & c_y & 0 & 0 \\ 0 & 0 & c_\theta & 0 \\ 0 & 0 & 0 & c_\varphi \end{bmatrix}$$

$$\mathbf{G}(\mathbf{x}(t)) = \begin{bmatrix} 0 & 0 & -\dot{\theta} \sin(\theta) (a_2 m_2 + a_3 m_1) & l m_1 \dot{\varphi} \sin(\varphi) \\ 0 & 0 & \dot{\theta} \cos(\theta) (a_2 m_2 + a_3 m_1) & l m_1 \dot{\varphi} \cos(\varphi) \\ 0 & 0 & 0 & a_3 l m_1 \dot{\varphi} \sin(\theta - \varphi) \\ 0 & 0 & -a_3 l m_1 \dot{\theta} \sin(\theta - \varphi) & 0 \end{bmatrix}$$

$$\mathbf{Q}(\mathbf{x}(t)) = \begin{bmatrix} 0 \\ g (m_1 + m_2 + m_3) \\ g \sin(\theta) (a_2 m_1 + a_3 m_2) \\ g l m_2 \sin(\varphi) \end{bmatrix}$$

$$\mathbf{L}(\mathbf{x}(t)) = \begin{bmatrix} -\sin(\theta) & 0 \\ \cos(\theta) & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$

## Appendix B

$$\mathbf{A}_c = \begin{bmatrix} 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 \\ 0.00 & 0.00 & -8.35 & 1.43 & -0.02 & 0.00 & 0.01 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & -0.16 & 0.00 & 0.00 \\ 0.00 & 0.00 & -5.33 & 6.33 & 0.05 & 0.00 & -0.58 & 0.02 \\ 0.00 & 0.00 & 6.21 & -26.54 & 0.01 & 0.00 & 0.02 & -0.07 \end{bmatrix}$$

$$\mathbf{B}_c = \begin{bmatrix} 0.00 & 0.00 \\ 0.00 & 0.00 \\ 0.00 & 0.00 \\ 0.00 & 0.00 \\ 0.00 & -1.14 \\ 0.52 & 0.00 \\ 0.00 & 44.44 \\ 0.00 & -1.28 \end{bmatrix}$$

$$\mathbf{A}_d = \begin{bmatrix} 1.00 & 0.00 & -0.01 & 0.00 & 0.05 & 0.00 & -0.00 & 0.00 \\ 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.05 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.99 & 0.01 & 0.00 & 0.00 & 0.05 & 0.00 \\ 0.00 & 0.00 & 0.01 & 0.97 & 0.00 & 0.00 & 0.00 & 0.05 \\ 0.00 & 0.00 & -0.42 & 0.07 & 1.00 & 0.00 & -0.01 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.99 & 0.00 & 0.00 \\ 0.00 & 0.00 & -0.26 & 0.31 & 0.00 & 0.00 & 0.97 & 0.01 \\ 0.00 & 0.00 & 0.31 & -1.31 & 0.00 & 0.00 & 0.01 & 0.96 \end{bmatrix}$$

$$\mathbf{B}_d = \begin{bmatrix} 0.00 & -0.00 \\ 0.00 & 0.00 \\ 0.00 & 0.05 \\ 0.00 & -0.00 \\ 0.00 & -0.06 \\ 0.03 & 0.00 \\ 0.00 & 2.19 \\ 0.00 & -0.06 \end{bmatrix}$$



# Bibliography

- [1] Yaser Alothman and Dongbing Gu. ‘Quadrotor transporting cable-suspended load using iterative linear quadratic regulator (ilqr) optimal control’. In: *2016 8th Computer Science and Electronic Engineering (CEECE)*. IEEE. 2016, pp. 168–173.
- [2] Joel A E Andersson et al. ‘CasADi – A software framework for nonlinear optimization and optimal control’. In: *Mathematical Programming Computation* (In Press, 2018).
- [3] Joel AE Andersson et al. ‘CasADi: a software framework for nonlinear optimization and optimal control’. In: *Mathematical Programming Computation* 11.1 (2019), pp. 1–36.
- [4] Frederico Augugliaro et al. ‘The flight assembled architecture installation: Cooperative construction with flying machines’. In: *IEEE Control Systems Magazine* 34.4 (2014), pp. 46–64.
- [5] Andrea Boccia, Lars Grüne and Karl Worthmann. ‘Stability and feasibility of state constrained MPC without stabilizing terminal constraints’. In: *Systems & control letters* 72 (2014), pp. 14–21.
- [6] Patricio J Cruz and Rafael Fierro. ‘Cable-suspended load lifting by a quadro-rotor UAV: hybrid model, trajectory generation, and control’. In: *Autonomous Robots* 41.8 (2017), pp. 1629–1643.
- [7] Julian Estevez et al. ‘A hybrid control approach for the swing free transportation of a double pendulum with a quadro-rotor’. In: *Applied Sciences* 11.12 (2021), p. 5487.
- [8] Gene F Franklin, J David Powell, Michael L Workman et al. *Digital control of dynamic systems*. Vol. 3. Addison-wesley Reading, MA, 1998.
- [9] Gene F Franklin et al. *Feedback control of dynamic systems*. Vol. 8. Prentice hall Upper Saddle River, 2020.
- [10] Farhad A Goodarzi and Taeyoung Lee. ‘Dynamics and control of quadro-rotor UAVs transporting a rigid body connected via flexible cables’. In: *2015 american control conference (ACC)*. IEEE. 2015, pp. 4677–4682.
- [11] Jaroslav Kautsky, Nancy K Nichols and Paul Van Dooren. ‘Robust pole assignment in linear state feedback’. In: *International Journal of control* 41.5 (1985), pp. 1129–1155.
- [12] Matěj Kuře et al. ‘Algorithms for cable-suspended payload sway damping by vertical motion of the pivot base’. In: *Mechanical Systems and Signal Processing* 149 (2021), p. 107131.

- [13] Christine Long. *Getting started*. 2019. URL: <https://beagleboard.org/getting-started>.
- [14] Lalo Magni, Davide Martino Raimondo and Frank Allgöwer. ‘Nonlinear model predictive control’. In: *Lecture Notes in Control and Information Sciences* 384 (2009).
- [15] Mathworks. *Simulink Coder Support Package for BeagleBone Blue Hardware*. 2020. URL: <https://www.mathworks.com/help/supportpkg/beagleboneblue/index.html>.
- [16] Mohamed W. Mehrez. *Sim\_2\_MPC\_Robot\_PS\_mul\_shooting.m*. 2020. URL: [https://github.com/MMehrez/MPC-and-MHE-implementation-in-MATLAB-using-Casadi/blob/master/workshop\\_github/Codes\\_casadi\\_v3\\_5\\_5/MPC\\_code/Sim\\_2\\_MPC\\_Robot\\_PS\\_mul\\_shooting.m](https://github.com/MMehrez/MPC-and-MHE-implementation-in-MATLAB-using-Casadi/blob/master/workshop_github/Codes_casadi_v3_5_5/MPC_code/Sim_2_MPC_Robot_PS_mul_shooting.m).
- [17] Igor Henrique Beloti Pizetta, Alexandre Santos Brandao and Mário Sarcinelli-Filho. ‘Modelling and control of a pvtol quadrotor carrying a suspended load’. In: *2015 International conference on unmanned aircraft systems (ICUAS)*. IEEE. 2015, pp. 444–450.
- [18] James Potter, William Singhose and Mark Costelloy. ‘Reducing swing of model helicopter sling load using input shaping’. In: *2011 9th IEEE International Conference on Control and Automation (ICCA)*. IEEE. 2011, pp. 348–353.
- [19] Justin Thomas et al. ‘Toward autonomous avian-inspired grasping for micro aerial vehicles’. In: *Bioinspiration & biomimetics* 9.2 (2014), p. 025010.
- [20] Jan Erik Trachte, Luis Felipe Gonzalez Toro and Aaron McFadyen. ‘Multi-rotor with suspended load: System dynamics and control toolbox’. In: *2015 IEEE Aerospace Conference*. IEEE. 2015, pp. 1–9.

# List of Figures

1	Experimental setup . . . . .	5
2	Magnetic encoders . . . . .	7
3	Magnetic linear sensor . . . . .	7
4	Experimental setup . . . . .	8
5	Diagram of control wiring . . . . .	9
6	Scheme of the laboratory setup . . . . .	11
7	Identification of the coef. of friction $c_\varphi$ , nmrs = 82 % . . . . .	16
8	Thrust test stand . . . . .	17
9	Thrust measurement . . . . .	17
10	Simulation results of the model (9) with initial conditions $\mathbf{x}_0 = [0, 0, 0, 0.2, 0, 0, 0, 0]$ and input vector $\mathbf{u} = [(m_1 + m_2 + m_3)g, 0]^T$ , (MATLAB/Simulink, ode45, rel. tol. 1e-6, max. step size 0.01) . . . . .	18
11	Simulation results of the model (19) with initial conditions $\mathbf{x}_0 = [0, 0, 0, 0.2, 0, 0, 0, 0]$ and input vector $\mathbf{u} = [0, 0]^T$ , (MATLAB-Simulink, ode45, rel. tol. 1e-6, max. step size 0.01) . . . . .	20
12	Simulation results of the model (24) with initial conditions $\mathbf{x}_0 = [0, 0, 0, 0.2, 0, 0, 0, 0]$ and input vector $\mathbf{u}_0 = [0, 0]^T$ , (MATLAB/Simulink, ode4, Fixed-step size 0.05) . . . . .	21
13	Eigenvalue locations of discrete-time model (24) . . . . .	22
14	Zoomed eigenvalue locations of discrete-time model (24) . . . . .	23
15	State-space control scheme . . . . .	25
16	Estimator and controller scheme . . . . .	26
17	Output feedback control with integral action . . . . .	27
18	Poles map of Poles location 1 . . . . .	28
19	Poles map of Poles location 2 . . . . .	29
20	Poles map of Poles location 3 . . . . .	30
21	Scheme of the Simulink implementation of output feedback control with integral action . . . . .	31
22	Control actions for scenario 1 (MATLAB/Simulink, Ode5, Fixed-step size 0.05) . . . . .	32
23	States response for scenario 1 (MATLAB/Simulink, Ode5, Fixed-step size 0.05) . . . . .	33
24	Control actions for scenario 2 (MATLAB/Simulink, Ode5, Fixed-step size 0.05) . . . . .	34
25	States response for scenario 2 (MATLAB/Simulink, Ode5, Fixed-step size 0.05) . . . . .	35
26	Control actions for scenario 3 (MATLAB/Simulink, Ode5, Fixed-step size 0.05) . . . . .	36
27	States repsonse for scenario 3 (MATLAB/Simulink, Ode5, Fixed-step size 0.05) . . . . .	37

28	Poles map of Design 1 . . . . .	41
29	Poles map of Design 2 . . . . .	42
30	Poles map of Design 3 . . . . .	43
31	Control actions of Scenario 1, (MATLAB/Simulink, Ode1be, Fixed-step size 0.05) . . . . .	44
32	States response of Scenario 1, (MATLAB/Simulink, Ode1be, Fixed-step size 0.05) . . . . .	45
33	Control actions of Scenario 2, (MATLAB/Simulink, Ode1be, Fixed-step size 0.05) . . . . .	46
34	States response of Scenario 2, (MATLAB/Simulink, Ode1be, Fixed-step size 0.05) . . . . .	47
35	Control actions of Scenario 3, (MATLAB/Simulink, Ode1be, Fixed-step size 0.05) . . . . .	48
36	States response of Scenario 3, (MATLAB/Simulink, Ode1be, Fixed-step size 0.05) . . . . .	49
37	Control actions of NMPC, $N = \{4, 8, 30\}$ . . . . .	53
38	States response of NMPC, $N = \{4, 8, 30\}$ . . . . .	54
39	Control actions of NMPC for $\mathbf{Q}_{mpc1}, \mathbf{Q}_{mpc2}, \mathbf{Q}_{mpc3}$ . . . . .	55
40	States response of NMPC for $\mathbf{Q}_{mpc1}, \mathbf{Q}_{mpc2}, \mathbf{Q}_{mpc3}$ . . . . .	56
41	Control inputs comparison for the control strategies . . . . .	57
42	States response comparison for the control strategies . . . . .	58

## List of Attachments

Attachment 1	Euler-Lagrange equations
Attachment 2	Nonlinear model
Attachment 3	System identification
Attachment 4	Discrete-time state feedback
Attachment 5	Output feedback control with integral action
Attachment 6	Linear-quadratic control
Attachment 7	NMPC
Attachment 8	BBB implementation