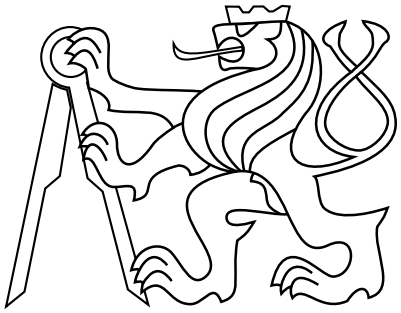
Czech Technical University in Prague

Faculty of Electrical Engineering
Department of Cybernetics



Control of an unmanned aerial vehicle with suspended payload

Master's Thesis

Martin Jiroušek

Prague, August 2023

Study programme: Cybernetics and Robotics

Supervisor: Ing. Jan Chudoba

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, 14. 08. 2023 Martin Jiroušek

Acknowledgments

Firstly, I would like to thank my family for their support during my studies and throughout my life. I would also like to express my gratitude to Ing. Jan Chudoba for guiding me throughout my master's degree. Working under his supervision taught me invaluable lessons and honed my skills as an engineer. Lastly, I would like to thank Ing. Tomáš Báča, PhD., for his willingness to adjust and provide the system upon which this thesis is based.

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Jiroušek Martin** Personal ID number: **474615**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Cybernetics and Robotics**

Branch of study: **Cybernetics and Robotics**

II. Master’s thesis details

Master’s thesis title in English:

Control of an Unmanned Aerial Vehicle with Suspended Payload

Master’s thesis title in Czech:

Řízení bezpilotního letounu se zavěšeným břemenem

Guidelines:

The goal of the thesis is the design of a control system for an unmanned multi-rotor helicopter with a suspended load. The designed regulator should enable stabilization of the system and improve the ability to damp the oscillations of the suspended pendulum. The control system will be provided with information about the position of the helicopter determined by the motion capture system. An essential part of the solution will be an estimator of the pendulum states. The output of the controller will be the input control signals for the existing built-in control system of the helicopter ensuring the required orientation in space. The control system will be implemented on the helicopter's on-board computer in C/C++ language.
1. The student becomes familiar with the theory and issues of feedback control of a multi-rotor helicopter.
2. The student conducts research on the state of the issue in the field and selects an appropriate solution to the given problem.
3. The student proposes his solution to the flight problem with pendulum stabilization or adapts an existing solution and tests this solution in simulations.
4. He further implements the designed state estimator and control system in the C/C++ language into the on-board system of the unmanned vehicle.
5. If the hardware condition allows, the student will perform an experimental verification of the proposed solution on a real platform.

Bibliography / sources:

[1] Baca, Tomas, et al. "The MRS UAV system: Pushing the frontiers of reproducible research, real-world deployment, and education with autonomous unmanned aerial vehicles." Journal of Intelligent & Robotic Systems 102.1 (2021): 1-28.
[2] K. Klausen, T. I. Fossen and T. A. Johansen, "Nonlinear control of a multirotor UAV with suspended load," 2015 International Conference on Unmanned Aircraft Systems (ICUAS), Denver, CO, USA, 2015, pp. 176-184, doi: 10.1109/ICUAS.2015.7152289.
[3] V. Prkačin, I. Palunko and I. Petrović, "State and parameter estimation of suspended load using quadrotor onboard sensors," 2020 International Conference on Unmanned Aircraft Systems (ICUAS), Athens, Greece, 2020, pp. 958-967, doi: 10.1109/ICUAS48674.2020.9213840.

Name and workplace of master's thesis supervisor:

Ing. Jan Chudoba Intelligent and Mobile Robotics CIIRC

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

Date of master's thesis assignment: **30.01.2023** Deadline for master's thesis submission: **14.08.2023**

Assignment valid until: **22.09.2024**

Ing. Jan Chudoba Supervisor's signature	prof. Ing. Tomáš Svoboda, Ph.D. Head of department's signature	prof. Mgr. Petr Páta, Ph.D. Dean's signature

III. Assignment receipt

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

.	
Date of assignment receipt	Student's signature

Abstract

This thesis deals with the design, implementation, and validation of a control system for an Unmanned Aerial Vehicle (UAV) with a suspended payload. The dynamics of the system were thoroughly investigated, and both nonlinear and linear mathematical models were derived. To estimate unmeasured state variables, including the position of the load, a Kalman filter was designed and incorporated into the control loop. The closed-loop control was achieved using a Quadratic Programming Model Predictive Control (QP-MPC). Additionally, a trajectory generator was formulated as a Quadratic Programming Optimal Control Problem (QP-OCP) to plan a sway-free trajectory. The resulting system was rigorously evaluated using the Gazebo robotic simulator, demonstrating its capability in effectively dampening load oscillations, tracking the UAV's target trajectory, and rejecting external disturbances.

Keywords Unmanned Aerial Vehicle, Suspended payload, Model Predictive Control, Kalman Filter, Sway-free trajectory, Optimal Control

Abstrakt

Tato práce se zabývá návrhem, implementací a ověřením řídicího systému pro bezpilotní letecký prostředek (UAV) se zavěšeným břemenem. Dynamika systému byla důkladně analyzována a byl odvozen nelineární a lineární matematický model systému. Za účelem odhadu neměřených stavových proměnných, včetně polohy břemene, byl navržen Kalmanův filtr a následně začleněn do řídicí smyčky. Uzavřená řídicí smyčka je zajištěna kvadratickým prediktivním regulátorem (QP-MPC). Dále byl formulován generátor trajektorie jakožto kvadratický optimální problém řízení (QP-OCP) za účelem plánování trajektorie bez kmitání. Výsledný systém byl pečlivě ověřen v robotickém simulátoru Gazebo a prokázal svou schopnost účinně tlumit kmitání nákladu, sledovat požadovanou trajektorii UAV a odolávat vnějším poruchám.

Klíčová slova Bezpilotní prostředky, Zavěšené břemeno, Prediktivní řízení na základě modelu, Kalmanův filtr, Trajaktorie bez kmitání, Optimální řízení

Abbreviations

GPS Global Positioning System

IMU Inertial Measurement Unit

LKF Linear Kalman Filter

LTI Linear time-invariant

LiDAR Light Detection and Ranging

MAV Micro Aerial Vehicle

MPC Model Predictive Control

MRS Multi-robot Systems Group

RTK Real-time Kinematics

UAV Unmanned Aerial Vehicle

MEMS Microelectromechanical Systems

FFT Fast Fourier Transform

LQR Linear-quadratic Regulator

SFE Safe Flight Envelope

LP Linear Programming

QP Quadratic Programming

QP-MPC Quadratic Programming Model Predictive Control

QP-OCP Quadratic Programming Optimal Control Problem

PDF Probability Density Function

mocap Motion Capture

NMPC Nonlinear Model Predictive Control

ODE Ordinary Differential Equation

ESC Electronic Speed Controller

Contents

Chapter 1

Introduction

Unmanned Aerial Vehicles (UAVs) are a rapidly evolving field within mobile robotics, with a multitude of potential practical applications. Currently, UAVs are predominantly used in industrial contexts for inspections, surveillance, search and rescue operations, transportation, environmental monitoring, and more. Research continues globally to explore and expand the potential uses of these versatile machines.

Two major factors have driven the rapid development and widespread adoption of Micro Aerial Vehicles (MAVs). First, advancements in computational technology have made increased processing power more affordable and accessible. This allows for high-frequency control loops necessary for stabilizing MAV. Second, the mobile phone industry's widespread incorporation of Microelectromechanical Systems (MEMS) accelerometers into Inertial Measurement Units (IMUs) in their devices led to mass production of these sensors. This mass production significantly reduced the price of these sensors and improved their accessibility for the MAV industry.

With the mass adoption of MAVs came a demand for increased autonomy. This spurred further research into navigation, control, communication, and the cooperation of multiple UAVs. Once the control of a single MAV was sufficiently mastered, the control community's focus expanded in several directions, most notably towards interaction with the environment. This is a significant challenge due to the under-actuated nature of common UAVs. Interacting with the environment amplifies the degree of under-actuation, making it a hot and ongoing research topic. Some notable advancements in this subfield include UAV flight while balancing an inverted pendulum [33], constructing tensile structures [28], building a wall [1], and human-UAV interaction [27], among others.

1.1 Motivation

The scenario of transporting suspended loads is prevalent in military operations, typically carried out by helicopters. Historically, only highly trained pilots could perform these tasks, as they require significant skill and experience. Mishandling of the payload could lead to disturbances resulting in catastrophic failure [16]. Consequently, more reliable control systems have been developed to perform these tasks, reducing the need for human intervention. This challenge extends to UAVs. Simulations reveal that when controlled by a general-purpose controller, certain combinations of payload weight and cable length can induce unwanted oscillations or even cause resonance, leading to system failure.

With the rising adoption of UAVs, the challenge of handling suspended payloads has become relevant for the civilian domain. Commercial delivery services, food delivery providers, and even medical suppliers have expressed interest in implementing UAV deliveries. Despite

Attachment	Mechanism	Impact on Control	Benefit
rigid	simple	alters rotational inertia, easy to control	easy mechanical implementation
semi-rigid	complex and heavy	alters rotational inertia, easy to control	tolerates less precise attachment
suspension	simple and light-weight	can destabilize the system, challenging to control	easy mechanical implementation, allows to carry large and heavy payload

Table 1.1: Overview of approaches for attaching payloads to UAV

the current regulatory, technical, and practical issues, some such systems are already in operation at the time of writing this thesis. For example, Zipline delivers medical supplies in rural areas of Ghana and Rwanda.

Different approaches exist for attaching payloads to UAVs, each with its unique characteristics and challenges, as summarized in Table 1.1. Two of these approaches are depicted in Figure 1.1. The suspension of a load represents the most lightweight and efficient method for attaching a load, provided non-aggressive flight is assumed. However, this method introduces a challenging control problem [13]. Numerous practical tasks, such as UAV carrying a sensor into inaccessible places or UAVs designed to capture objects like Eagle.one1[1] [7], could benefit from an effective slung load controller as well. Therefore, studying a system that allows for the control of a UAV with a suspended payload has a clear practical motivation.

(a) Rigid attachment [2] (b) Suspension [10]

Figure 1.1: Methods of attaching payloads to UAV

1.2 Related Works

Described control problem has been extensively studied in the literature. However several open problems still persist and complicate control design and implementation. In the literature there are roughly two main approaches to design of described controller - nonlinear, and optimal control.

[1] https://eagle.one

Nonlinear Control

The work of Klausen et al. [23] utilizes a back-stepping nonlinear method. While the resulting tracking of the target UAV's position was independent of the load's amplitude in simulations, the states of the load were not controlled. This resulted in large amplitudes of the load during the tracking of the target trajectory. This issue was partly addressed by the implementation of input shaping [36], but the proposed solution was only verified using simulations based on the same model as that used for the control design. In [14], a passivity-based controller was designed, requiring knowledge of the payload's state and higher derivatives, thus limiting its practical application. In [25], the flexibility of the rope is modeled and taken into account, allowing for aggressive maneuvers with proven asymptotic stabilization of the load during UAV translation. However, the experimental validation was limited, as the cable was in the final test approximated as a single link.

Optimal Control

The second widely-used approach is rooted in optimal control theory. Two aspects can be optimized: the trajectory and feed-forward control actions, and the feedback control. Trajectory optimization can be precomputed and solved offline, allowing for highly nonlinear, and even hybrid, models and complex objective functions. This approach is evident in [24], where a task of generating a trajectory for passing through a small window is solved using an iterative LQG algorithm. Similarly, in [32], a sway-free trajectory is derived via dynamic programming optimization. Compared to these examples, Rigatos [6] proposes an algorithm that linearly approximates the model in each iteration and calculates gains for an H_∞ controller by solving algebraic Riccati equations. Model Predictive Control (MPC), as employed in [8], [22], and [21], is a versatile tool for this problem, allowing for optimization of both trajectory and feedback control. Although the linearly-constrained version of MPC is limited to non-aggressive flights, this is not a problem when aiming for real-world deployment.

It's worth noting that the solutions proposed in the aforementioned works are often verified only in simulations. Experimental works conducted on real hardware always employed a Motion Capture (mocap) system, implying that all results were achieved using the ground-truth position of the load. In more practical scenarios, the states of the load are not known and must be estimated.

State Estimation

Most methods in literature utilize the fact that the load exerts a disturbance on the UAV, which can be measured by an IMU or deduced from position changes. Some works further utilize additional sensors to support the estimation process, including a down-facing onboard camera [17], a load cell measuring tension [18], and a Light Detection and Ranging (LiDAR) [5]. Extended Kalman filters have been developed and experimentally verified in [4] and [5]. In [11], the task of simultaneous estimation of the load's states and the rope's length was tackled using Fast Fourier Transform (FFT).

1.3 Task Setup

The objective of this thesis is to implement a control system for a UAV with a suspended payload. This control system should be designed to utilize the UAV's position and orientation measurements, provided by a mocap system[2]. Preparing the system for real-world deployment necessitates the estimation of the load's position, as it will not be directly measured. The UAV, for which the control system is designed, is equipped with a low-level controller, known as the 'flight controller', which stabilizes the UAV and manages its attitude (i.e. orientation and thrust). Therefore, the designed control system will output the desired orientation and thrust of the UAV. The resulting regulator is intended to be implemented in C/C++ on an onboard computer, that the UAV is equipped with.

1.4 Contributions

This study was not designed to contest or redefine the control or estimation results of existing state-of-the-art solutions. Instead, the goal was to establish a deployable system that could operate on a platform without the need for additional sensors to support the estimation of the load's position. To the best of my knowledge, no such system has been reported in literature thus far. Most documented control systems relied on mocap systems for localization of the UAV and also the load, and while some works did develop estimators, they did not include a controller and frequently utilized additional instrumentation.

The control system introduced here consists of a sway-free trajectory generator, an Model Predictive Control (MPC) feedback controller, and a Kalman filter. This control system has been implemented as a module of the open-source Multi-robot Systems Group (MRS) system [3]. Although real-world hardware testing of the system has not yet been conducted, the simulation results are promising, and the system is ready for real-world experimental verification.

1.5 Mathematical Notation

Throughout the thesis, vector notation is frequently employed to enhance the readability and conciseness of equations. However, the use of vector notation can also introduce confusion due to the lack of complete uniformity in vector conventions found in the existing literature. Consequently, it is crucial to establish the specific convention adopted within this thesis. The general notation and nomenclature are defined in Table 1.2.

1.5.1 Block Matrices

In the case of matrices that are structured into blocks, partial or complete block notation can be utilized. The most commonly recurring substructures in this thesis are the zero vector or matrix $\mathbf{0}$ and the identity matrix $\mathbf{I}$. To specify their dimensions, a subscript can be used, as described in Table 1.2. However, when the dimensions are clear from the context, the subscript

[2]Throughout the thesis, the dependence on mocap was eliminated and substituted for Real-time Kinematics (RTK) Global Positioning System (GPS) and flight controller's estimate of attitude.

can be omitted. As and example the following matrix can be expressed in several equivalent ways:

$$\begin{bmatrix} 1 & 1 & 0 \\ 2 & 0 & 1 \\ 3 & 0 & 0 \end{bmatrix} = \begin{bmatrix} \mathbf{v} & \mathbf{I} \\ 3 & \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{v} & \mathbf{I}_2 \\ 3 & \mathbf{0}_{1,2} \end{bmatrix} = \left[\begin{array}{c|c} 1 & \mathbf{I} \\ 2 & \\ \hline 3 & \mathbf{0} \end{array}\right],$$

where additional definition of $\mathbf{v} = \begin{bmatrix} 1 & 2 \end{bmatrix}^T$ is necessary.

1.5.2 Derivatives

To prevent confusion in multivariate derivatives, they are introduced in more detail. Consider a scalar function of multiple variables $f : \mathbb{R}^n \to \mathbb{R}$. The total derivative of $f(\mathbf{x})$ is represented as a row vector, while its gradient is a column vector:

$$\frac{\mathrm{d}f(\mathbf{x})}{\mathrm{d}\mathbf{x}} = (\nabla f(\mathbf{x}))^T = \begin{bmatrix} \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}_{(1)}} & \cdots & \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}_{(n)}} \end{bmatrix}.$$

Similarly, the partial derivative of $f(\mathbf{y}, \mathbf{z})$ with respect to $\mathbf{y}$ is also a row vector. In order to avoid transposition of a partial derivative, let us define operator *partial gradient* $\nabla_{\cdot}$:

$$\frac{\partial f(\mathbf{y}, \mathbf{z})}{\partial \mathbf{y}} = (\nabla_{\mathbf{y}} f(\mathbf{y}, \mathbf{z}))^T = \begin{bmatrix} \frac{\partial f(\mathbf{y},\mathbf{z})}{\partial \mathbf{y}_{(1)}} & \cdots & \frac{\partial f(\mathbf{y},\mathbf{z})}{\partial \mathbf{y}_{(l)}} \end{bmatrix}.$$

Now, consider a vector function of a single variable. The derivative of this function is a column vector, while its gradient is a row vector.

Combining these concepts, we can establish a general rule for the derivative of a vector function $\mathbf{h} : \mathbb{R}^n \to \mathbb{R}^m$:

$$\frac{\mathrm{d}\mathbf{h}(\mathbf{x})}{\mathrm{d}\mathbf{x}} = \begin{bmatrix} \frac{\mathrm{d}\mathbf{h}_{(1)}(\mathbf{x})}{\mathrm{d}\mathbf{x}} \\ \vdots \\ \frac{\mathrm{d}\mathbf{h}_{(m)}(\mathbf{x})}{\mathrm{d}\mathbf{x}} \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathbf{h}(\mathbf{x})}{\partial \mathbf{x}_{(1)}} & \cdots & \frac{\partial \mathbf{h}(\mathbf{x})}{\partial \mathbf{x}_{(n)}} \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathbf{h}_{(1)}(\mathbf{x})}{\partial \mathbf{x}_{(1)}} & \cdots & \frac{\partial \mathbf{h}_{(1)}(\mathbf{x})}{\partial \mathbf{x}_{(n)}} \\ \vdots & \ddots & \vdots \\ \frac{\partial \mathbf{h}_{(m)}(\mathbf{x})}{\partial \mathbf{x}_{(1)}} & \cdots & \frac{\partial \mathbf{h}_{(m)}(\mathbf{x})}{\partial \mathbf{x}_{(n)}} \end{bmatrix}.$$

Aforementioned matrix is also known as Jacobian matrix. Similarly partial derivative of the function is defined as:

$$\frac{\partial \mathbf{h}(\mathbf{y}, \mathbf{z})}{\partial \mathbf{y}} = \begin{bmatrix} \frac{\partial \mathbf{h}_{(1)}(\mathbf{y},\mathbf{z})}{\partial \mathbf{y}_{(1)}} & \cdots & \frac{\partial \mathbf{h}_{(1)}(\mathbf{y},\mathbf{z})}{\partial \mathbf{y}_{(l)}} \\ \vdots & \ddots & \vdots \\ \frac{\partial \mathbf{h}_{(m)}(\mathbf{y},\mathbf{z})}{\partial \mathbf{y}_{(1)}} & \cdots & \frac{\partial \mathbf{h}_{(m)}(\mathbf{y},\mathbf{z})}{\partial \mathbf{y}_{(l)}} \end{bmatrix}.$$

$\mathbf{x}$, $\boldsymbol{\alpha}$	vector, pseudo-vector, or tuple
$\underline{\mathbf{x}}$, $\underline{\boldsymbol{\omega}}$	unit vector or unit pseudo-vector
$\mathbf{X}, \boldsymbol{\Omega}$	matrix
$\mathbf{I}_N, \mathbf{I}$	$N \times N$ identity matrix, identity matrix of dimensions which are clear from the context
$\mathbf{0}_{N,M}, \mathbf{0}$	zero vector or matrix with N rows and M columns, zero vector or matrix of dimensions which are clear from the context
$\mathcal{W}, \mathcal{B}, \mathcal{L}$	Cartesian coordinate systems
$\mathbf{R}_{\mathbf{x}}$	matrix representing rotation around $\mathbf{x}$ axis
$\mathbf{R}_{\mathcal{B}}^{\mathcal{W}}$	matrix representing rotation from $\mathcal{B}$ to $\mathcal{W}$ coordinate system
$\mathbf{x}_{(n)} = \mathbf{x}^{\intercal}\underline{\mathbf{e}}_n$	n^{th} vector element (row)
$x[n]$	x at the sample n
$f(\cdot)$	scalar function $f : \mathbb{R}^n \to \mathbb{R}$
$\mathbf{f}(\cdot)$	vector function $f : \mathbb{R}^n \to \mathbb{R}^m$
$\frac{\mathrm{d}x}{\mathrm{d}t}, \dot{x}, \frac{\mathrm{d}^2x}{\mathrm{d}t^2}, \ddot{x}$	1^{st} and 2^{nd} time derivative of x
$\frac{\mathrm{d}\mathbf{f}(\mathbf{x})}{\mathrm{d}\mathbf{x}}$	total derivative of a function
$\frac{\partial \mathbf{f}(\mathbf{y},\mathbf{z})}{\partial \mathbf{y}}$	partial derivative of a function
$\nabla \mathbf{f}(\mathbf{x}) = \frac{\mathrm{d}\mathbf{f}(\mathbf{x})}{\mathrm{d}\mathbf{x}}^T$	gradient of a function
$\nabla_{\mathbf{x}} \mathbf{f}(\mathbf{y},\mathbf{z}) = \frac{\partial \mathbf{f}(\mathbf{y},\mathbf{z})}{\partial \mathbf{y}}^T$	partial gradient of a function
$\mathrm{s}(\cdot), \mathrm{c}(\cdot)$	abbreviations of $\sin(\cdot)$ and $\cos(\cdot)$
$SO(3)$	3D special orthogonal group of rotations
$p(x\|y,u)$	probability densitity function of x conditioned by y and u
$x[n\|n-1]$	x at the sample n conditioned by sample $n-1$
$\hat{x}$	estimate of x

Table 1.2: Mathematical notation, nomenclature and notable symbols.

Chapter 2

Mathematical Model of the System

This chapter is dedicated to the derivation of the mathematical model of the system under study. While various approaches exist for controlling a system without prior knowledge (also known as black-box systems), such as empirical tuning of controllers, these methods typically prove effective only for low-order stable systems. However, the system being investigated in this study falls outside of that category. One method to incorporate prior knowledge is by analyzing the physics of the system and developing its mathematical model. The model based approach offers several advantages, including constraining the set of controller candidates, enabling the analysis of the boundary between "difficult" and "impossible" scenarios, and utilizing frameworks established by the control system community. Nonetheless, there exists a risk of incorporating erroneous prior knowledge, which could introduce bias. Consequently, it is crucial to derive an accurate and appropriate mathematical model of the system.

2.1 Mechanical Parameters of the System

The dynamics of the system are directly influenced by its mechanical properties. All significant parameters that impact the system's dynamics are listed in Table 2.1. While these parameters can vary between different flights, it is assumed that they remain constant during a single flight. As a result, these parameters are treated as constants throughout the derivation of the model.

Physical Property	**Notation**
mass of the UAV	m_{uav}
mass of the load	m_l
length of the cable	l

Table 2.1: Overview of mechanical parameters of the system

2.2 Lagrangian Formalism

The Lagrangian formalism is a widely utilized modeling approach based on the Euler-Lagrange equations:

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial \mathcal{L}(\mathbf{q},\dot{\mathbf{q}})}{\partial \dot{\mathbf{q}}_{(k)}} - \frac{\partial \mathcal{L}(\mathbf{q},\dot{\mathbf{q}})}{\partial \mathbf{q}_{(k)}} = 0, \qquad k = 1,\ldots,N,$$

where $\mathcal{L}(\mathbf{q}, \dot{\mathbf{q}})$ denotes the Lagrange function, also known as the Lagrangian, and $\mathbf{q} \in \mathbb{R}^N$ represents the vector of generalized coordinates. The set of equations (2.1) can also be expressed compactly in vector form using the notation introduced in Section 1.5:

$$\frac{\mathrm{d}}{\mathrm{d}t}\nabla_{\dot{\mathbf{q}}}\mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}) - \nabla_{\mathbf{q}}\mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{0}. \tag{2.1}$$

Lagrangian is generally defined as

$$\mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}) = \mathcal{T}^*(\mathbf{q}, \dot{\mathbf{q}}) - \mathcal{V}(\mathbf{q}), \tag{2.2}$$

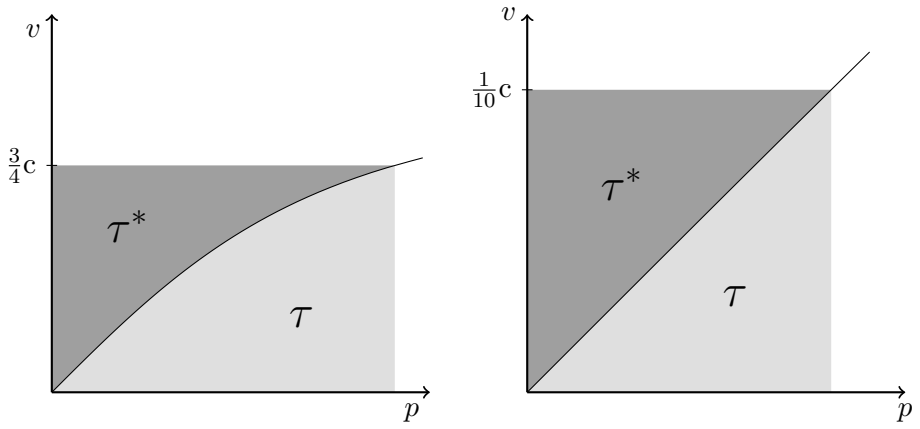
where $\mathcal{T}^*(\mathbf{q}, \dot{\mathbf{q}})$ represents the kinetic co-energy and $\mathcal{V}(\mathbf{q})$ represents the potential energy.

To clarify the notation, the * sign is not an indicator of the co-energy case; rather, it is an operator representing the Legendre transformation. This transformation defines the fundamental relation between energy and co-energy. In the case of mechanical systems moving much slower than the speed of light, the relationship between momentum p and velocity v is approximately linear. Consequently, the kinetic co-energy becomes identical to kinetic energy. This simplification has a straightforward and intuitive interpretation since the kinetic energy is defined as:

$$\mathcal{T} = \int_{t_0}^{t_1} P(t)\mathrm{d}t,$$

and therefore refers to the area under the curve in Fig. 2.1. Hence, throughout the thesis, these terms can be interchanged without loss of correctness, and the Lagrangian can be formulated as:

$$\mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}) = \mathcal{T}(\mathbf{q}, \dot{\mathbf{q}}) - \mathcal{V}(\mathbf{q}). \tag{2.3}$$



(a) General nonlinear relation

(b) Approximately linear part of the relation valid for most mechanical systems

Figure 2.1: Momentum-velocity and kinetic energy-coenergy relation

2.3 Generalized Coordinates

The Lagrangian formalism is characterized by a structured procedure consisting of clearly defined steps. However, when it comes to the selection of generalized coordinates,

there is room for creativity and flexibility. The objective of this step is to identify a set of independent variables that is as small as possible, yet expressive enough to describe the mechanical configuration of the system. These chosen coordinates, known as "generalized" coordinates, often encompass a combination of translational and angular variables.

2.3.1 Coordinate Systems

As previously mentioned, the Lagrangian formalism necessitates the use of generalized coordinates and the Lagrange function as inputs, both expressed within an inertial reference frame[1]. However, it may be advantageous to introduce additional coordinate systems. These coordinate systems can simplify the definition of generalized coordinates, facilitate the formulation of the Lagrangian, and provide a more systematic approach overall.

The role of the inertial frame of reference is fulfilled by the coordinate system $\mathcal{W}$. While a perfect inertial frame of reference does not exist, for the scale of the problem and precision required for its solution, a coordinate system fixed on the Earth's surface approximates an inertial frame of reference well enough. The coordinate system $\mathcal{W}$ is a Cartesian right-handed system, with the z-axis pointing away from the Earth's center [2].

The next coordinate system, denoted as $\mathcal{B}$, is bound to the frame of the UAV, with its origin located at the center of the UAV's mass. Since the UAV is not motionless and experiences acceleration with respect to the inertial frame of reference $\mathcal{W}$, $\mathcal{B}$ is clearly not an inertial frame. The relation between these two coordinate systems determines the displacement and orientation of the UAV. Additionally, the motivation behind introducing the third coordinate system, $\mathcal{L}$, is to describe the configuration of the load. The origin of this coordinate system is situated at the center of mass of the load, and its z-axis points towards the center of mass of the UAV. A visual depiction of the aforementioned coordinate systems can be found in Fig. 2.2.

To ensure clarity regarding the reference frame in which the generalized coordinates are expressed, the following notation is introduced. A vector of generalized coordinates $\mathbf{q}$ expressed in coordinate system $\mathcal{B}$ is denoted as $\mathbf{q}^{\mathcal{B}}$. In cases where the upper-script is omitted, it indicates that the vector is expressed in the inertial frame $\mathcal{W}$. This notation hopefully enhances readability and avoids ambiguity in understanding the coordinate system of the generalized coordinates.

2.3.2 Generalized Coordinates of the UAV

The mechanical configuration of the UAV can be fully described by a displacement vector $\mathbf{s}_{\text{uav}} = \begin{bmatrix} x & y & z \end{bmatrix}^T$ expressed in the coordinate system $\mathcal{W}$, along with the intrinsic Tait-Bryan angles $\Gamma = \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^T$. The introduced angles adhere to the right-hand rule and are in more detail described in Table 2.2. The rotation matrix $\mathbf{R}_{\mathcal{B}}^{\mathcal{W}} \in SO(3)$, representing the transformation from the UAV-frame coordinate system to the inertial coordinate system,

[1] Coordinate has no meaning without a coordinate system.

[2] Defining the orientation of the z-axis with respect to the gravitational acceleration is tempting, but due to the lack of agreement within the physics community regarding its orientation convention, it would lead to an ambiguous definition.

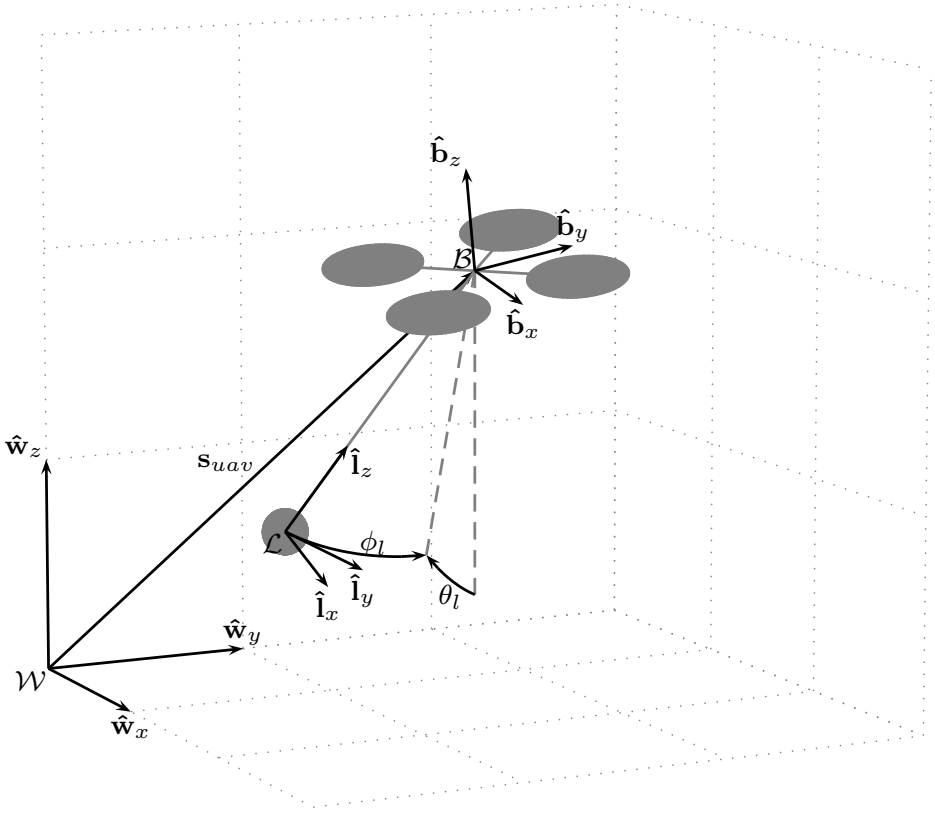


Figure 2.2: Illustration of coordinate systems and generalized coordinates of the load

can be obtained through three consecutive rotations:

$$\begin{aligned}
\mathbf{R}_{\mathcal{B}}^{\mathcal{W}}(\phi,\theta,\psi) &= \mathbf{R}_z(\psi)\mathbf{R}_y(\theta)\mathbf{R}_x(\phi),\\
\mathbf{R}_x(\phi) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix},\\
\mathbf{R}_y(\theta) &= \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix},\\
\mathbf{R}_z(\psi) &= \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}.
\end{aligned} \tag{2.4}$$

The sequence of rotations leading to the final orientation of the UAV in the inertial coordinate system is also illustrated in Fig. 2.3. Consequently, the generalized coordinates describing the UAV are defined as:

$$\mathbf{q}_{uav} = \begin{bmatrix} x & y & z & \phi & \theta & \psi \end{bmatrix}^T.$$

2.3.3 Generalized Coordinates of the Load

Similar to a UAV, the mechanical configuration of the load can also be described using six generalized coordinates, resulting in a total of twelve generalized coordinates for the entire

name	notation	axis of rotation
roll	ϕ	$\underline{\mathbf{w}}_x$
pitch	θ	$\mathbf{R}_x(\phi)\underline{\mathbf{w}}_y$
yaw	ψ	$\mathbf{R}_y(\theta)\mathbf{R}_x(\phi)\underline{\mathbf{w}}_z$

Table 2.2: Description of Tait-Bryan angles

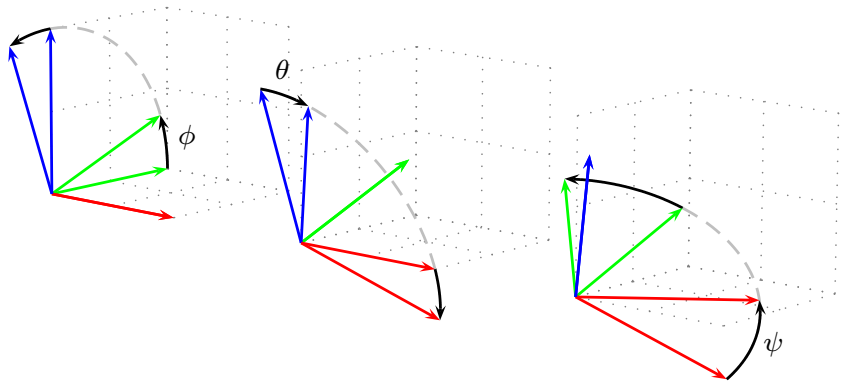


Figure 2.3: Visualization of rotations defined by Tait-Bryan angles. Axes follow RGB convention.

system. It is important to note that the dynamics of the system depend not only on the generalized coordinates but also on their first derivatives. Consequently, the dimensionality of the state vector for the derived model is doubled or potentially increased even further. In such cases involving high-order systems, it is a common practice to reduce the model by disregarding the least relevant elements of the state vector. This reduction is necessary to enable system analysis and control design. Therefore, it is advisable to proactively reduce the dimensionality of the state vector during the modeling process by introducing suitable assumptions.

Although the problem formulation assumes a load connected to the UAV by a flexible cable, modeling such a case would be challenging, and the resulting complex model would be impractical for control design. Flexible structures are classified as spatially distributed systems and are typically described by partial differential equations. To make the model more applicable, it is common practice to approximate the partial differential equations by a set of Ordinary Differential Equations (ODEs). In this case, the rope is divided into several rigid-body segments, with each segment described by a single ODE. The accuracy of the approximation depends on the number of segments, allowing for a trade-off between model precision and complexity based on the specific problem requirements.

For the purposes of this thesis, it is assumed that the UAV performs only non-aggressive maneuvers, ensuring that the cable remains in tension. Under this assumption, the cable behaves similarly to a stiff rod, and it is sufficient to approximate it using a single rigid body segment. The general nonholonomic constraint, expressed as the Euclidean distance between the UAV and the load being less or equal to the cable length, is transformed into a holonomic constraint:

$$||\mathbf{s}_{\text{uav}} - \mathbf{s}_l|| - l = 0,$$

where $\mathbf{s}_l$ is the displacement vector of the load. This constraint reduces the number of gen-

eralized coordinates by one, allowing the displacement of the load to be intuitively expressed using the UAV displacement ($\mathbf{s}_{uav}$) and two angles (θ_l and ϕ_l) that describe the orientation of the rod. Since the task assumes a general load without any other prior knowledge beyond its stiffness, and considering that the load orientation is not relevant to the problem, the load is approximated as a point mass. This simplification further reduces the number of generalized coordinates.

Based on the declared assumptions, the mechanical configuration of the load can be described using two angular coordinates $\mathbf{q}_l = \begin{bmatrix} \phi_l & \theta_l \end{bmatrix}$. These coordinates parametrize the matrix $\mathbf{R}_{\mathcal{L}}^{\mathcal{W}} \in SE(3)$, which represents a sequence of consecutive intrinsic rotations:

$$\mathbf{R}_{\mathcal{L}}^{\mathcal{W}}(\phi_l, \theta_l) = \mathbf{R}_x(\phi_l)\mathbf{R}_y(\theta_l). \tag{2.5}$$

The generalized coordinates for the entire system are defined as:

$$\bar{\mathbf{q}} = \begin{bmatrix} \mathbf{q}_{\text{uav}}^T & \mathbf{q}_l^T \end{bmatrix}^T = \begin{bmatrix} x & y & z & \phi & \theta & \psi & \phi_l & \theta_l \end{bmatrix}^T. \tag{2.6}$$

2.4 Lagrange Function

In the context of Lagrangian mechanics, the system's behavior is described in terms of its energies: kinetic and potential. The system's energy transitions between these two forms, but their differential remains constant in scenarios without energy dissipation. This fundamental concept underlies the definition of the Lagrangian as:

$$\mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}) = \mathcal{T}^*(\mathbf{q}, \dot{\mathbf{q}}) - \mathcal{V}(\mathbf{q}).$$

2.4.1 Potential Energy

Potential energy is a form of stored energy that has the potential to be converted into other forms of energy and perform work. In the context of the problem, the only source of potential energy is the gravitational field, and thus the following simple equation can be utilized:

$$\mathcal{V} = gmh, \tag{2.7}$$

where g represents gravitational acceleration, m denotes the total mass of the system, and h stands for the height of the center of mass above a reference height with zero potential. For the sake of simplicity, the zero-potential level is chosen such that it contains the origin of the coordinate system $\mathcal{W}$. The task of formulating the potential energy boils down to expressing Equation (2.7) using the generalized coordinates defined in Equation (2.6) and the mechanical parameters of the system.

The displacement of the load can be expressed in the coordinate system $\mathcal{B}$ using Equation (2.5) as follows:

$$\mathbf{s}_l^{\mathcal{B}} = \mathbf{R}_{\mathcal{L}}^{\mathcal{W}} \begin{bmatrix} 0 & 0 & -l \end{bmatrix}^T$$

and further, in the coordinate system $\mathcal{W}$, by consecutive translation:

$$\mathbf{s}_l = \mathbf{s}_l^{\mathcal{B}} + \mathbf{s}_{uav} = \mathbf{R}_{\mathcal{L}}^{\mathcal{W}} \begin{bmatrix} 0 & 0 & -l \end{bmatrix}^T + \mathbf{s}_{uav}. \tag{2.8}$$

The equation (2.7) translates for the studied system to:

$$\mathcal{V}(\bar{\mathbf{q}}) = g(m_{uav}z + m_l z_l) = g \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} (m_{uav}\mathbf{s}_{uav} + m_l \mathbf{s}_l). \tag{2.9}$$

Plugging Equation (2.8) into Equation (2.9) yields:

$$\mathcal{V}(\bar{\mathbf{q}}) = gm_{uav}z + m_l(z - l\cos\theta_l \cos\phi_l)). \tag{2.10}$$

2.4.2 Kinetic energy

Since the assigned problem assumes the use of an embedded attitude controller, there is no need to derive the model of the rotational dynamics directly. Instead, the model of the closed-loop rotational dynamics is introduced in a separate section. Therefore, the rotational kinetic energy can be disregarded, and the focus is only on translational kinetic energy. The translational kinetic energy of the system can be obtained by summing the kinetic energy of both the UAV and the load:

$$\mathcal{T}(\dot{\bar{\mathbf{q}}}) = \frac{1}{2}\left(m_{uav}\left|\left|\dot{\mathbf{s}}_{uav}\right|\right|^2 + m_l\left|\left|\dot{\mathbf{s}}_{uav} + \dot{\mathbf{s}}_l^{\mathcal{B}}\right|\right|^2\right). \tag{2.11}$$

The velocity of the load expressed in frame $\mathcal{B}$ has the following form:

$$\dot{\mathbf{s}}_l^{\mathcal{B}} = \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{R}_{\mathcal{L}}^{\mathcal{W}}(\phi_l, \theta_l)\begin{bmatrix} 0 \\ 0 \\ -l \end{bmatrix} = \begin{bmatrix} -l\cos(\theta_l)\dot{\theta}_l \\ -l\sin(\theta_l)\sin(\phi_l)\dot{\theta}_l + l\cos(\theta_l)\cos(\phi_l)\dot{\phi}_l \\ l\cos(\phi_l)\sin(\theta)_l\dot{\theta}_l + l\cos(\theta_l)\sin(\phi_l)\dot{\phi}_l \end{bmatrix}. \tag{2.12}$$

Putting Equation (2.12) into Equation (2.11) yields:

$$\begin{aligned}\mathcal{T}(\dot{\bar{\mathbf{q}}}) = \frac{1}{2}\Bigg(m_{uav}\left(\dot{x}^2 + \dot{y}^2 + \dot{z}^2\right) + m_l\Bigg(&\left(\dot{x} - l\dot{\theta}_l\cos(\theta_l)\right)^2 + \\ &\left(-l\dot{\theta}_l\sin(\theta_l\sin(\phi_l) + l\dot{\phi}_l\cos(\theta_l)\cos(\phi_l) + \dot{y}\right)^2 + \\ &\left(l\dot{\theta}_l\sin(\theta_l)\cos(\phi_l) + l\dot{\phi}_l\cos(\theta_l)\sin(\phi_l) + \dot{z}\right)^2\Bigg)\Bigg).\end{aligned} \tag{2.13}$$

The Lagrange function of the system is obtained by substituting equations (2.10) and (2.13) into equation (2.2).

2.4.3 Reduced Generalized Coordinates

It is worth noting that neither the potential energy nor the kinetic energy depends on the Tait-Bryan angles ϕ, θ, and ψ. This arises because the rotational dynamics of the UAV's body is not included in the model. Although the orientation of the UAV does not directly appear in the Lagrangian, it influences the system through the external forces exerted by the propellers. Since the left-hand sides of the equations in (2.1) corresponding to the Tait-Bryan angles of the UAV are identically zero, these equations can be omitted, resulting in a reduced set of equations of motion and a simplified model. The reduced generalized coordinates are defined as:

$$\mathbf{q} = \begin{bmatrix} x & y & z & \phi_l & \theta_l \end{bmatrix}^T.$$

To maintain clarity and consistency in notation, the following identity will be used throughout the rest of the thesis:

$$\mathcal{L}(\bar{\mathbf{q}}, \dot{\bar{\mathbf{q}}}) = \mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}).$$

2.5 External and Dissipative Forces

The Euler-Lagrange equations given by (2.1) are applicable only when no external forces, except for those associated with potential energy, act on the system. In other words, these

equations hold true only for the scenario of free fall in a vacuum. However, when considering the presence of rotating propellers and the thrust they exert, as well as the dissipation of energy, the right-hand sides of the equations become non-zero:

$$\frac{\mathrm{d}}{\mathrm{d}t}\nabla_{\dot{\mathbf{q}}}\mathcal{L}(\mathbf{q},\dot{\mathbf{q}}) - \nabla_{\mathbf{q}}\mathcal{L}(\mathbf{q},\dot{\mathbf{q}}) = \mathbf{f}_{ext}(\Gamma, F) + \mathbf{f}_{dis}(\dot{\mathbf{q}}, \mathbf{q}), \tag{2.14}$$

where $\mathbf{f}_{ext}(\cdot)$ represent external forces, $\mathbf{f}_{dis}(\cdot)$ accounts for dissipative forces and F denotes magnitude of the collective thrust of the propellers. In this context, the external forces correspond to the collective thrust exerted by the propellers. Considering the mechanical design of a general under-actuated UAV, it is evident that the collective thrust aligns with the $\hat{\mathbf{b}}_z$ axis. It can be expressed in the coordinate system $\mathcal{W}$ using the rotation matrix given by (2.4) as follows:

$$\mathbf{f}_{ext}(\Gamma, F) = \begin{bmatrix} F_x(\Gamma,F) \\ F_y(\Gamma,F) \\ F_z(\Gamma,F) \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{\mathcal{B}}^{\mathcal{W}}(\Gamma) \\ \mathbf{0} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ F \end{bmatrix} = F \begin{bmatrix} \sin(\theta)\cos(\psi)\cos(\phi) + \sin(\psi)\sin(\phi) \\ \sin(\theta)\sin(\psi)\cos(\phi) - \cos(\psi)\sin(\phi) \\ \cos(\theta)\cos(\phi) \\ 0 \\ 0 \end{bmatrix}, \tag{2.15}$$

where F_x, F_y, and F_z are the force components expressed in the $\mathcal{W}$ coordinate frame.

The main source of dissipation in the system is not explicitly specified in the problem formulation. It is likely that air drag plays a significant role when dealing with a large and light load. On the other hand, friction in the joint connecting the cable and the UAV may be more prominent when dealing with a dense object. In the latter case, the dissipation would be negligible, and therefore, only the air drag component of dissipation is modeled. Since no prior knowledge regarding the shape of the load or the aerodynamic parameters is provided, it is not necessary to incorporate a precise and complex model. Hence, a simple linear model is employed. This model assumes a centrally symmetrical load, represented by a single drag coefficient, denoted as $d > 0$.

Assuming no wind, the force caused by air drag is linear with respect to the translational velocity of the load expressed in $\mathcal{W}$ coordinates. To determine its effect on the pendulum, it is necessary to transform the force to the coordinate system $\mathcal{L}$. Subsequently, the tangential part of the transformed force (i.e., the coordinates corresponding to the $\underset{\sim}{\mathbf{l}}_x$ and $\underset{\sim}{\mathbf{l}}_y$ axes) can be mapped to the corresponding angular variables. This can be expressed as:

$$\mathbf{f}_{dis}^{l}(\dot{\mathbf{q}}, \mathbf{q}) = -d \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix} \mathbf{R}_y^{-1}(\theta_l)\mathbf{R}_x^{-1}(\phi_l)\dot{\mathbf{s}}_l^{\mathcal{W}}. \tag{2.16}$$

On the other hand, the radial part of the transformed force does not affect the angular variables of the load. Instead, it affects the UAV itself[3]. However, the way it acts on the UAV must be considered in the $\mathcal{W}$ coordinate system. This line of thought involves several consecutive

[3]Imagine a degenerated load in the form of a parachute

transformations:

$$\mathbf{f}_{dis}^{uav}(\dot{\mathbf{q}},\mathbf{q}) = -d\mathbf{R}_x(\phi_l)\mathbf{R}_y(\theta_l)\begin{bmatrix}0&0&0\\0&0&0\\0&0&1\end{bmatrix}\mathbf{R}_y^{-1}(\theta_l)\mathbf{R}_x^{-1}(\phi_l)\dot{\mathbf{s}}_l^{\mathcal{W}}. \tag{2.17}$$

Combining equations (2.16) and (2.17) yields the total air drag acting on the system:

$$\mathbf{f}_{dis}(\dot{\mathbf{q}},\mathbf{q}) = \begin{bmatrix}\mathbf{f}_{dis}^{uav}\\ \mathbf{f}_{dis}^{l}\end{bmatrix} = -\mathbf{D}(\phi_l,\theta_l)\dot{\mathbf{q}}$$
$$-d\begin{bmatrix}\mathrm{s}^2(\theta_l) & \mathrm{s}(\theta_l)\mathrm{c}(\theta_l)\mathrm{s}(\phi_l) & \mathrm{s}(\theta_l)\mathrm{c}(\theta_l)\mathrm{c}(\phi_l) & 0 & 0\\ \mathrm{s}(\theta_l)\mathrm{c}(\theta_l)\mathrm{s}(\phi_l) & \mathrm{c}^2(\theta_l)\mathrm{s}^2(\phi_l) & \mathrm{c}^2(\theta_l)\mathrm{s}(\phi_l)\mathrm{c}(\phi_l) & 0 & 0\\ \mathrm{s}(\theta_l)\mathrm{c}(\theta_l)\mathrm{c}(\phi_l) & \mathrm{c}^2(\theta_l)\mathrm{s}(\phi_l)\mathrm{c}(\phi_l) & \mathrm{c}^2(\theta_l)\mathrm{c}^2(\phi_l) & 0 & 0\\ 0 & \mathrm{c}(\phi_l) & \mathrm{s}(\phi_l) & l\mathrm{c}(\theta_l) & 0\\ \mathrm{c}(\theta_l) & \mathrm{s}(\theta_l)\mathrm{s}(\phi_l) & \mathrm{s}(\theta_l)\mathrm{c}(\phi_l) & 0 & l\end{bmatrix}\dot{\mathbf{q}}. \tag{2.18}$$

2.6 Equations of Motion

Expressing equation (2.14) directly can be quite overwhelming. However, it can be manipulated into a form that is often used in manipulation robotics and improves readability:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + (\mathbf{C}(\dot{\mathbf{q}},\mathbf{q}) + \mathbf{D}(\mathbf{q}))\,\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \mathbf{f}_{ext}(\Gamma, F). \tag{2.19}$$

The vector functions mentioned in the equation are defined as follows:

$$\mathbf{M}(\mathbf{q}) = \begin{bmatrix}m_{uav}+m_l & 0 & 0 & 0 & -lm_l\mathrm{c}(\theta_l)\\ 0 & m_{uav}+m_l & 0 & lm_l\mathrm{c}(\theta_l)\mathrm{c}(\phi_l) & -lm_l\mathrm{s}(\theta_l)\mathrm{s}(\phi_l)\\ 0 & 0 & m_{uav}+m_l & lm_l\mathrm{c}(\theta_l)\mathrm{s}(\phi_l) & lm_l\mathrm{s}(\theta_l)\mathrm{c}(\phi_l)\\ 0 & lm_l\mathrm{c}(\theta_l)\mathrm{c}(\phi_l) & lm_l\mathrm{c}(\theta_l)\mathrm{s}(\phi_l) & l^2m_l\mathrm{c}^2(\theta_l) & 0\\ -lm_l\mathrm{c}(\theta_l) & -lm_l\mathrm{s}(\theta_l)\mathrm{s}(\phi_l) & lm_l\mathrm{s}(\theta_l)\mathrm{c}(\phi_l) & 0 & l^2m_l\end{bmatrix}, \tag{2.20}$$

$$\mathbf{C}(\dot{\mathbf{q}},\mathbf{q}) = \begin{bmatrix}0&0&0 & 0 & l\dot{\theta}_l m_l\mathrm{s}(\theta_l)\\ 0&0&0 & -lm_l(\dot{\theta}_l\mathrm{s}(\theta_l)\mathrm{c}(\phi_l)+\dot{\phi}_l\mathrm{c}(\theta_l)\mathrm{s}(\phi_l)) & -lm_l(\dot{\theta}_l\mathrm{c}(\theta_l)\mathrm{s}(\phi_l)+\dot{\phi}_l\mathrm{s}(\theta_l)\mathrm{c}(\phi_l))\\ 0&0&0 & lm_l(\dot{\phi}_l\mathrm{c}(\theta_l)\mathrm{c}(\phi_l)-\dot{\theta}_l\mathrm{s}(\theta_l)\mathrm{s}(\phi_l)) & lm_l(\dot{\theta}_l\mathrm{c}(\theta_l)\mathrm{c}(\phi_l)-\dot{\phi}_l\mathrm{s}(\theta_l)\mathrm{s}(\phi_l))\\ 0&0&0 & -\frac{1}{2}l^2\dot{\theta}_l m_l\mathrm{s}(2\theta_l) & -\frac{1}{2}l^2m_l\dot{\phi}_l\mathrm{s}(2\theta_l)\\ 0&0&0 & \frac{1}{2}l^2m_l\dot{\phi}_l\mathrm{s}(2\theta_l) & 0\end{bmatrix},$$

$$\mathbf{g}(\mathbf{q}) = \begin{bmatrix}0 & 0 & g(m_{uav}+m_l) & glm_l\cos(\theta_l)\sin(\phi_l) & glm_l\sin(\theta_l)\cos(\phi_l)\end{bmatrix}^T,$$

and $\mathbf{D}(\mathbf{q})$, $\mathbf{f}_{ext}(\Gamma, f)$ already being defined by (2.18) and (2.15) respectively. The equations of motion presented in this section are consistent with the model described in [23], except for the dissipative vector function $\mathbf{D}(\mathbf{q})$. The author of the paper opted to include only the friction in the joint connecting the cable and the UAV, while neglecting the air-drag of the load. As a result, there is a difference in the models of dissipation between the two approaches.

2.7 Model of the Embedded Control Loop

Every UAV requires some form of control in order to achieve stable flight. At the lowest level, Electronic Speed Controllers (ESCs) are used to control the rotations per minute of

the propellers. Above that, there is typically an embedded controller, such as the popular Pixhawk, which enables control at various levels, including the orientation (attitude) of the UAV and the collective thrust. The controller developed in this thesis utilizes this control loop and outputs the desired target orientation $(\phi_d, \theta_d, \psi_d)$ and target collective thrust (F_d) as control actions. However, to design an effective controller, it is necessary to account for the dynamics of the inner control loop, which requires a model of its closed loop behavior.

The embedded controller is often treated as a black box from the user's perspective. Its step response typically resembles a real exponential function, allowing it to be approximated by a first-order system with a general transfer function:

$$G(s) = \frac{K}{Ts+1},$$

where K represents the steady-state gain and T is the time constant. The previous work [9] and [20] has demonstrated the satisfactory nature of this approximation. By converting these transfer functions into state-space representations and combining them, the following model is obtained:

$$\begin{bmatrix} \dot{\theta} \\ \dot{\phi} \\ \dot{\psi} \\ \dot{F} \end{bmatrix} = \begin{bmatrix} -\frac{1}{T_\theta} & 0 & 0 & 0 \\ 0 & -\frac{1}{T_\phi} & 0 & 0 \\ 0 & 0 & -\frac{1}{T_\psi} & 0 \\ 0 & 0 & 0 & -\frac{1}{T_F} \end{bmatrix} \begin{bmatrix} \theta \\ \phi \\ \psi \\ F \end{bmatrix} + \begin{bmatrix} \frac{K_\theta}{T_\theta} & 0 & 0 & 0 \\ 0 & \frac{K_\phi}{T_\phi} & 0 & 0 \\ 0 & 0 & \frac{K_\psi}{T_\psi} & 0 \\ 0 & 0 & 0 & \frac{K_F}{T_F} \end{bmatrix} \begin{bmatrix} \theta_d \\ \phi_d \\ \psi_d \\ F_d \end{bmatrix} = \tag{2.21}$$

$$\dot{\mathbf{x}}_a = \mathbf{A}_a \mathbf{x}_a + \mathbf{B}_u \mathbf{u}.$$

The model described in (2.21) assumes complete decoupling of the control loops. Practical experiments, as reported in [9], have shown that this assumption holds fairly well for well-designed embedded controllers and non-aggressive maneuvers.

2.8 State Transition Model

To provide a comprehensive understanding of the system dynamics within the context of the task, it is necessary to establish the relationship between the equations of motion defined by (2.19) and the closed-loop dynamics of the embedded attitude controller (2.21). By merging these models together, we obtain the following representation:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \left(\mathbf{C}(\dot{\mathbf{q}}, \mathbf{q}) + \mathbf{D}(\mathbf{q})\right)\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \mathbf{f}_{ext}(\mathbf{x}_a), \tag{2.22}$$

$$\dot{\mathbf{x}}_a = \mathbf{A}_a \mathbf{x}_a + \mathbf{B}_u \mathbf{u}. \tag{2.23}$$

The set of equations (2.22) and (2.23) represents a complete nonlinear model of the system. Equation (2.22) is a second-order ODE. To convert it into a set of two first-order ODEs, we can introduce a substitution:

$$\dot{\mathbf{q}} = \nu, \tag{2.24}$$

$$\mathbf{M}(\mathbf{q})\dot{\nu} + \left(\mathbf{C}(\nu, \mathbf{q}) + \mathbf{D}(\mathbf{q})\right)\nu + \mathbf{g}(\mathbf{q}) = \mathbf{f}_{ext}(\mathbf{x}_a). \tag{2.25}$$

However, equation (2.25) is still in an implicit form, i.e. $\tilde{\mathbf{f}}(\dot{\nu}, \nu, \mathbf{q}, \mathbf{x}_a) = 0$. In certain situations, it can be advantageous to explicitly express the relationship between $\dot{\nu}$ and the

remaining variables as $\dot{\nu} = \bar{\mathbf{f}}(\nu, \mathbf{q}, \mathbf{x}_a)$. This can be achieved by inverting the vector function $\mathbf{M}$ and rearranging equation (2.25), resulting in:

$$\begin{aligned} \dot{\mathbf{q}} &= \nu, \\ \dot{\nu} &= -\mathbf{M}^{-1}(\mathbf{q}) \left((\mathbf{C}(\nu, \mathbf{q}) + \mathbf{D}(\mathbf{q})) \nu + \mathbf{g}(\mathbf{q}) - \mathbf{f}_{ext}(\mathbf{x}_a) \right), \\ \dot{\mathbf{x}}_a &= \mathbf{A}_a \mathbf{x}_a + \mathbf{B}_u \mathbf{u}. \end{aligned} \tag{2.26}$$

Equations (2.26) represent the state transition model of the system, expressed as a set of first-order explicit ODEs. The notion of state of the system, denoted by $\mathbf{x}$, augments already introduced variables:

$$\mathbf{x} = \begin{bmatrix} \mathbf{q}^T & \nu^T & \mathbf{x}_a^T \end{bmatrix}^T = \begin{bmatrix} x & y & z & \phi_l & \theta_l & \dot{x} & \dot{y} & \dot{z} & \dot{\phi}_l & \dot{\theta}_l & \theta & \phi & \psi & F \end{bmatrix}^T.$$

Similarly, the right-hand side of equations (2.26) can be combined into a single vector function, resulting in the standard form of the state transition model:

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

Conversion of the implicit model into its explicit form however comes at some cost. The inversion of the matrix $\mathbf{M}(\mathbf{q})$ introduces a singularity in the model. Looking at (2.20), it is evident that $\mathbf{M}(\mathbf{q})$ becomes singular when $\theta_l = \frac{\pi}{2}$, making it impossible to invert. While this singularity is a result of the chosen representation and does not indicate any physical issue, it may cause numerical instability near the singularity. Nevertheless, considering the assumptions made in Section 2.3.3, which constrain the amplitude of θ_l and ϕ_l, the singularity is far from being reached, and the explicit form of the model can still be used effectively.

2.9 Linearization

Since most control system frameworks for analysis and control desing focus on linear systems, it is a common practise to approximate the nonlinear state space model by a linear one. This approximation, often referred to as linearization, is performed around an operating point. As the name suggests, operating point is typically the state in which the system spends the most time. Therefore it makes sense to have the lowest approximation error around this point. In the case of the studied system, a natural choice for the operating point $\mathbf{x}_0$ is the hover state, where the load is motionless directly beneath the UAV. The operating point can be defined as:

$$\mathbf{x}_0 = \begin{bmatrix} \mathbf{s}_{uav0}^T & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & g(m_{uav} + m_l) \end{bmatrix}^T, \tag{2.27}$$

where $\mathbf{s}_{uav0}^T$ denotes displacement of the UAV at the operating point. It should be noted that the linear approximation does not depend on the UAV displacement, so $\mathbf{s}_{uav0}$ can be chosen arbitrarily. Similarly, the operating yaw angle can also be chosen freely and is therefore arbitrarily set to zero. The operation thrust is set to compensate for the gravitational force $\mathbf{g}(\cdot)$, resulting in zero vertical velocity of the UAV. The choice of the operating point (2.27) is further justified by the symmetry of the system. Since the system is symmetric with respect to its states (except for the vertical velocity, which is affected asymmetrically by gravity), it is not meaningful to choose positive or negative values for elements of the operating point. This would lead to an asymmetric model of a symmetric system.

The linear time-invariant (LTI) state transition model of the system can be described by the equation:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}, \tag{2.28}$$

where matrices $\mathbf{A}$ and $\mathbf{B}$ are defined as:

$$\mathbf{A} = \frac{\partial \mathbf{f}(\mathbf{x},\mathbf{u})}{\partial \mathbf{x}}(\mathbf{x}_0) =$$

$$\left[\begin{array}{c|ccccc|c} & 0 & 0 & 0 & 0 & 0 & \\ & 0 & 0 & 0 & 0 & 0 & \\ \mathbf{0} & 0 & 0 & 0 & 0 & 0 & \mathbf{0} \\ & 0 & \frac{gm_l}{m_{uav}} & 0 & -\frac{g(m_{uav}+m_l)}{lm_{uav}} & 0 & \\ & -\frac{gm_l}{m_{uav}} & 0 & 0 & 0 & -\frac{g(m_{uav}+m_l)}{lm_{uav}} & \\ \hline & \frac{d}{lm_{uav}} & 0 & 0 & 0 & \frac{d(m_{uav}+m_l)}{l^2 m_{uav} m_l} & \\ & 0 & \frac{d}{lm_{uav}} & 0 & -\frac{d(m_{uav}+m_l)}{l^2 m_{uav} m_l} & 0 & \\ \mathbf{I} & 0 & 0 & -\frac{d}{m_{uav}+m_l} & 0 & 0 & \mathbf{0} \\ & 0 & \frac{d}{m_{uav}} & 0 & -\frac{d(m_{uav}+m_l)}{lm_{uav} m_l} & 0 & \\ & -\frac{d}{m_{uav}} & 0 & 0 & 0 & -\frac{d(m_{uav}+m_l)}{lm_{uav} m_l} & \\ \hline & \frac{g(m_{uav}+m_l)}{m_{uav}} & 0 & 0 & 0 & \frac{g(m_{uav}+m_l)}{lm_{uav}} & \\ & 0 & -\frac{g(m_{uav}+m_l)}{m_{uav}} & 0 & \frac{g(m_{uav}+m_l)}{lm_{uav}} & 0 & \\ \mathbf{0} & 0 & 0 & 0 & 0 & 0 & \mathbf{A}_a \\ & 0 & 0 & \frac{1}{m_{uav}+m_l} & 0 & 0 & \end{array}\right]^T,$$

$$\mathbf{B} = \frac{\partial \mathbf{f}(\mathbf{x},\mathbf{u})}{\partial \mathbf{u}} = \begin{bmatrix} \mathbf{0} \\ \mathbf{B}_u \end{bmatrix}. \tag{2.29}$$

2.10 Measurement Model

To provide a comprehensive mathematical description of the control problem, a measurement model is developed to establish the relationship between the physically measured variables, the system states, and the system inputs. Since only the position and Tait-Brian angles of the UAV are directly measured, it is necessary to incorporate this fact into the mathematical model.

A vector of outputs $\mathbf{y} = \begin{bmatrix} x_m & y_m & z_m & \theta_m & \phi_m & \psi_m \end{bmatrix}^T$ is defined to represent the measured variables. This vector is explicitly related to the state vector $\mathbf{x}$ and the input vector $\mathbf{u}$ through a generally nonlinear function:

$$\mathbf{y} = \mathbf{g}(\mathbf{x},\mathbf{u}).$$

In this case, where some states are directly measured, the measurement model simplifies to a linear relationship:

$$\mathbf{y} = \mathbf{C}\mathbf{x} = \begin{bmatrix} \mathbf{I}_3 & \mathbf{0} & \mathbf{0} & \mathbf{0}_{3,1} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_3 & \mathbf{0}_{3,1} \end{bmatrix}, \tag{2.30}$$

where $\mathbf{C}$ is the corresponding measurement matrix.

2.11 Discretization

Most iterative algorithms inherently rely on a discrete-time model of the system. Hence, it is practical to discretize the Linear time-invariant (LTI) state-space model presented by equations 2.28 and 2.29. Various methods are available to achieve this, each optimizing different objectives. Some aim to resemble the original system's step or impulse response, while others focus on frequency domain similarities. The primary requirement for the chosen method is its simple closed-form formula explicitly dependent on the sampling period. As the measurement or control loop may not be perfectly periodic, using a fixed discrete model introduces errors. Hence, an advantageous approach is to use online discretization, performed at each iteration of the loop and based on the time difference between the current and last iterations. Since many discretization algorithms are designed to be run only once, and may be complicated with long run-times, a simpler yet sufficiently accurate method is presented in this work.

The discrete-time LTI state-space is defined as follows:

$$\mathbf{x}[t+1] = \bar{\mathbf{A}}\mathbf{x}[t] + \bar{\mathbf{B}}\mathbf{u}[t], \tag{2.31}$$

$$\mathbf{y}[t] = \bar{\mathbf{C}}\mathbf{x}[t] + \bar{\mathbf{D}}\mathbf{u}[t]. \tag{2.32}$$

The objective is to find matrices $\bar{\mathbf{A}}, \bar{\mathbf{B}}, \bar{\mathbf{C}}, \bar{\mathbf{D}}$ for the discrete-time model. The simplest method of numerical integration, the forward Euler method, is employed for this task. This method is based on the first-order Taylor expansion:

$$\mathbf{x}(t+T_s) = \mathbf{x}(t) + T_s\dot{\mathbf{x}}(t), \tag{2.33}$$

where T_s denotes sampling period. Applying this method to the continuous transition model (2.28) yields:

$$\mathbf{x}(t+T_s) = \mathbf{x}(t) + T_s\left(\mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t)\right) = \left(\mathbf{I} + T_s\mathbf{A}\right)\mathbf{x}(t) + T_s\mathbf{B}\mathbf{u}(t), \tag{2.34}$$

which results in discrete-time state transition matrices as functions of the sampling period:

$$\bar{\mathbf{A}}(T_s) = \mathbf{I} + T_s\mathbf{A}, \qquad \bar{\mathbf{B}}(T_s) = T_s\mathbf{B}. \tag{2.35}$$

The state output matrix $\mathbf{C}$ remains unaffected by discretization, thus the equation $\bar{\mathbf{C}} = \mathbf{C}$ holds. As the system has no direct feed-through ($\mathbf{D} = \mathbf{0}$), the formula for $\bar{\mathbf{D}}$ is not relevant.

2.12 Summary

This chapter has focused on understanding the physics of the system by introducing variables that describe the state of the system and their relation to different coordinate frames. These variables were then used to formulate a Lagrange function and derive a nonlinear state transition model that captures the dynamics of the system. Furthermore, the linear approximation of the nonlinear model was derived, resulting in a LTI model. Subsequently, a straightforward online discretization method was introduced, enabling the utilization of the model in quasi-periodical algorithms. The main contributions of this chapter are the derived LTI model (2.28), (2.29) and the nonlinear state space model (2.15), (2.18), (2.20), (2.21) (2.26), providing a comprehensive understanding of the system's dynamics.

Chapter 3

State Estimator

This chapter focuses on the design of an estimator for estimating the states of the controlled system. Some classes of controllers operate only on measurements of controlled variables. PID controllers, for example, which are well-established in the industry, belong to this category. Their performance, however, is relatively limited, making them ill-suited for controlling systems with complex dynamics. Moreover, since their behaviour is solely based on the output variables, they often lack context of state variables that are not measured. Consequently, they are unable to handle constraints on unmeasured states. To circumvent some of these issues, they are frequently stacked into different structures. In a previous study [9], which tackled a similar setup without a load, a cascade of PID controllers was employed, facilitating constraints on the UAV's velocity. This approach, however, is sensitive to precise parameter tuning. Furthermore, due to the cascade nature of this approach, a delay is introduced, resulting in inferior performance compared to controllers that are more informed. Therefore, this chapter introduces a state estimator that enables the use of state-dependent controllers.

3.1 Stochastic Systems

A state estimator is defined as a function or mapping $\mathbf{h}(\cdot)$, responsible for determining the internal state based on observed data and system dynamics:

$$\hat{\mathbf{x}}[t] = \mathbf{h}(\mathbf{u}[0], \mathbf{y}[0], \ldots, \mathbf{u}[t-1], \mathbf{y}[t-1]) = \mathbf{h}(\mathcal{D}^{t-1}), \tag{3.1}$$

where $\hat{\cdot}$ denotes an estimated value and $\mathcal{D}^n$ is a sequence of observed inputs and outputs up to and including the nth sample. If the system is deterministically modeled, all variables in (3.1) are deterministic, and the function $\mathbf{h}(\cdot)$ is often termed a state observer. However, as most systems contain a certain level of stochasticity, it can be useful to adopt a probabilistic system model and treat the mentioned variables as random variables.

The model of a stochastic system can be formulated in multiple ways depending on the use case. One of the most common representations is the following discrete-time model, which incorporates additive noise:

$$\begin{aligned} \mathbf{x}[t+1] &= \mathbf{f}(\mathbf{x}[t], \mathbf{u}[t]) + \mathbf{v}[t], \\ \mathbf{y}[t] &= \mathbf{C}\mathbf{x}[t] + \mathbf{e}[t], \end{aligned} \tag{3.2}$$

Here, $\mathbf{v}[t]$ stands for the process noise, while $\mathbf{e}[t]$ symbolizes measurement noise, both at sample t. Given that our model is inherently imperfect - as some physical phenomena were deliberately ignored to simplify it - the state transition introduces some level of error. This error is represented by the process noise. Furthermore, measurements often deviate from the ground truth value due to sensor noise. This discrepancy is accounted for by the measurement noise.

Assume both process and measurement noises to be independently and identically distributed, drawn from a joint normal Probability Density Function (PDF):

$$p\left(\begin{bmatrix} \mathbf{v}[t] \\ \mathbf{e}[t] \end{bmatrix}\right) = \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{Q} & \mathbf{S} \\ \mathbf{S}^T & \mathbf{R} \end{bmatrix}\right).$$

Independence of samples implies that the noise isn't time-correlated. This noise model is termed Gaussian white noise, and the class of estimators that assumes it is referred to as Kalman filters. The concept of stochastic systems and filtering is visually represented in Fig. 3.1.

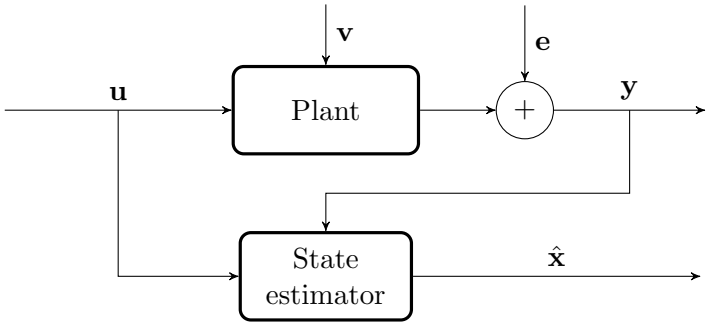


Figure 3.1: General concept of stochastic systems and state estimation

3.2 Bayesian Justification

Since the system's state and measurements are modelled as random variables it is reasonable to express the model in terms of probabilities:

$$\begin{aligned} &\text{state transition model} \quad p(\mathbf{x}[t+1]|\mathbf{x}[t], \mathbf{u}[t], \mathbf{y}[t]), \\ &\text{measurement model} \quad p(\mathbf{y}[t]|\mathbf{x}[t], \mathbf{u}[t]). \end{aligned} \tag{3.3}$$

Consider the typical discrete scheme where a measurement is first obtained, followed by a state transition calculation. If there's a correlation between the process and measurement noise, indicated by $\mathbf{S} \neq \mathbf{0}$, the measurement can bring insights about the process noise. This information can be harnessed to enhance the precision of the predicted state. As a result, the state transition model is in general conditioned by this measurement. Nevertheless, in many instances, the absence of correlation is presumed, leading to a simplified state transition model:

$$p(\mathbf{x}[t+1]|\mathbf{x}[t], \mathbf{u}[t], \mathbf{y}[t]) = p(\mathbf{x}[t+1]|\mathbf{x}[t], \mathbf{u}[t]).$$

This assumptions allows to execute following steps separately.

Assuming that a conditioned PDF $p\left(\mathbf{x}[t]|\mathcal{D}^{t-1}\right)$, alongside the observations $\mathbf{y}[t], \mathbf{u}[t]$, are provided, the PDF can be updated as follows:

$$p\left(\mathbf{x}[t]|\mathcal{D}^{t}\right) = \frac{p\left(\mathbf{y}[t]|\mathbf{x}[t], \mathbf{u}[t], \mathcal{D}^{t-1}\right)}{p\left(\mathbf{y}[t]|\mathbf{u}[t], \mathcal{D}^{t-1}\right)} p\left(\mathbf{x}[t]|\mathbf{u}[t], \mathcal{D}^{t-1}\right) \propto \underbrace{p\left(\mathbf{y}[t]|\mathbf{x}[t], \mathbf{u}[t]\right)}_{\text{measurement model}} \underbrace{\left(\mathbf{x}[t]|\mathcal{D}^{t-1}\right)}_{\text{prior}}. \tag{3.4}$$

When executing the final step of (3.4), two pivotal concepts are invoked:

- By definition, the state provides comprehensive information about the system. Hence, any further observations are redundant.
- A natural control condition is assumed, implying that the control is based on previous observations and doesn't introduce new insights.

Upon employing (3.4) to filter the state estimate, the state's future trajectory can be projected as:

$$p\left(\mathbf{x}[t+1], \mathcal{D}^t\right) = \int \underbrace{p\left(\mathbf{x}[t+1]|\mathbf{x}[t], \mathcal{D}^t\right)}_{\text{state transition model}} \underbrace{p\left(\mathbf{x}[t]|\mathcal{D}^t\right)}_{\text{updated prior}} \mathrm{d}\mathbf{x}[t]. \tag{3.5}$$

As already indicated, this sequential approach is possible thanks to assumption on uncorrelated noise.

Subsequent to these steps, the resulting Probability Density Function (PDF) updates in accordance with the current time-frame, transitioning from $p\left(\mathbf{x}[t]|\mathcal{D}^{t-1}\right)$ to $p\left(\mathbf{x}[t+1]|\mathcal{D}^t\right)$. Within this context, (3.4) is termed the *data-update*, whereas (3.5) is designated as the *time-update*. The iterative procedure, encapsulated by (3.4) and (3.5), underpins the foundational theory for all Kalman filters operating under the assumption $\mathbf{S} = \mathbf{0}$. Distinct Kalman filter variants diverge based on the model, referenced in (3.3), they adopt or implementation of steps (3.4) and (3.5).

3.3 Linear Kalman Filter

This section introduces a widely adopted estimation technique known as the Linear Kalman Filter (LKF). It represents as a pivotal advancement within the control system domain, extending the applicability of automatic control to domains previously unexplored by this field. Arguably, one of the most notable accomplishments of the LKF was its pivotal contribution to the successful moon landing.

3.3.1 Linear Discrete-Time Stochastic Systems

Linear Kalman filter formulates stochastic model (3.3) as a linear discrete-time state space model:

$$\begin{aligned} \mathbf{x}[t+1] &= \mathbf{A}\mathbf{x}[t] + \mathbf{C}\mathbf{u}[t] + \mathbf{v}[t], \\ \mathbf{y}[t] &= \mathbf{C}\mathbf{x}[t] + \mathbf{D}\mathbf{u}[t] + \mathbf{e}[t]. \end{aligned} \tag{3.6}$$

Given that the mean values of noise are zero, the dynamics of these mean values align with the deterministic discrete-time LTI state space:

$$\begin{aligned} \hat{\mathbf{x}}[t+1] &= \mathbf{A}\hat{\mathbf{x}}[t] + \mathbf{C}\mathbf{u}[t], \\ \hat{\mathbf{y}}[t] &= \mathbf{C}\hat{\mathbf{x}}[t] + \mathbf{D}\mathbf{u}[t], \end{aligned} \tag{3.7}$$

where input $\mathbf{u}[t]$ is considered deterministic, since we have full authority over it. Similarly dynamics of error term, $\tilde{\mathbf{x}}[t] = \mathbf{x}[t] - \hat{\mathbf{x}}[t]$, can be expressed as:

$$\begin{aligned} \tilde{\mathbf{x}}[t+1] = \mathbf{x}[t+1] - \hat{\mathbf{x}}[t+1] &= \mathbf{A}\tilde{\mathbf{x}}[t] + \mathbf{v}[t], \\ \tilde{\mathbf{y}}[t] = \mathbf{y}[t] - \hat{\mathbf{y}}[t] &= \mathbf{C}\tilde{\mathbf{x}}[t] + \mathbf{e}[t]. \end{aligned} \tag{3.8}$$

Using model (3.8), one can derive the evolution of the covariance matrix:

$$\operatorname{cov}\left(\begin{bmatrix}\mathbf{x}[t+1]\\ \mathbf{y}[t]\end{bmatrix}\right) = \varepsilon\left(\begin{bmatrix}\tilde{\mathbf{x}}[t+1]\\ \tilde{\mathbf{y}}[t]\end{bmatrix}\begin{bmatrix}\tilde{\mathbf{x}}[t+1]\\ \tilde{\mathbf{y}}[t]\end{bmatrix}^T\right) = \begin{bmatrix}\mathbf{A}\mathbf{P}_x[t]\mathbf{A}^T+\mathbf{Q} & \mathbf{A}\mathbf{P}_x[t]\mathbf{C}^T+\mathbf{S}\\ \mathbf{C}\mathbf{P}_x[t]\mathbf{A}^T+\mathbf{S}^T & \mathbf{C}\mathbf{P}_x[t]\mathbf{C}^T+\mathbf{R}\end{bmatrix}, \quad (3.9)$$

where $\mathbf{P}_x[t]$ represents the covariance matrix of the state at sample t.

3.3.2 Optimality

Given the linearity of the measurement model, the data-update process, as outlined in (3.4), remains a linear function of the current observation. As demonstrated in [31], the Linear Kalman Filter stands as the optimal linear mean square estimator (LMSE), irrespective of the noise distribution. In scenarios with a normal distribution, the mean square estimate (MSE) manifests as a linear function of the observation, thereby aligning with the LMSE. Under the assumption of Gaussian white noise, the Linear Kalman Filter achieves optimal performance in minimizing mean square error when compared with all possible estimators, inclusive of nonlinear variants. Nevertheless, in situations where this Gaussian assumption is not satisfied, the LKF retains its position as the optimal estimator within the realm of linear approaches.

3.3.3 Linear Mean Square Estimate

Estimation is in general defined as a task of finding estimate $\hat{\bar{\mathbf{x}}}(\bar{\mathbf{y}})$, where $\bar{\mathbf{x}}$ represents the random vector subject to estimation, and $\bar{\mathbf{y}}$ denotes the observed data's random vector. Relation between these two vectors is defined by a joint probability density function $p(\bar{\mathbf{x}}, \bar{\mathbf{y}})$. Quality of an estimate is often determined by an objective function formulated as an mean square error:

$$J_{MS} = \varepsilon((\bar{\mathbf{x}} - \hat{\bar{\mathbf{x}}}_{MS}(\bar{\mathbf{y}}))^T(\bar{\mathbf{x}} - \hat{\bar{\mathbf{x}}}_{MS}(\bar{\mathbf{y}}))) = \int\int (\bar{\mathbf{x}} - \hat{\bar{\mathbf{x}}}_{MS}(\bar{\mathbf{y}}))^T(\bar{\mathbf{x}} - \hat{\bar{\mathbf{x}}}_{MS}(\bar{\mathbf{y}}))p(\bar{\mathbf{x}}, \bar{\mathbf{y}})\ \mathrm{d}\bar{\mathbf{x}}\ \mathrm{d}\bar{\mathbf{y}} \quad (3.10)$$

To acquire the mean square estimate, one must minimize (3.10). However, the challenge lies in accessing the joint probability density function $p(\bar{\mathbf{x}}, \bar{\mathbf{y}})$, which is typically intricate to determine in practical scenarios.

As a simplifying measure, the estimate is often postulated to be a linear function of the observation, leading to the formulation of a linear mean square estimate:

$$\hat{\bar{\mathbf{x}}}_{LMS}(\bar{\mathbf{y}}) = \mathbf{A}\bar{\mathbf{y}} + \mathbf{b}. \quad (3.11)$$

Furthermore, it's generally accepted that both vectors, $\bar{\mathbf{x}}$ and $\bar{\mathbf{y}}$, are drawn from a joint normal probability density function:

$$p\left(\begin{bmatrix}\bar{\mathbf{x}}\\ \bar{\mathbf{y}}\end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix}\mu_x\\ \mu_y\end{bmatrix}, \begin{bmatrix}\mathbf{P}_{xx} & \mathbf{P}_{xy}\\ \mathbf{P}_{yx} & \mathbf{P}_{yy}\end{bmatrix}\right).$$

Plugging (3.11) into (3.10) results in the objective function for the linear mean square estimate:

$$J_{LMS} = \mathcal{E}((\bar{\mathbf{x}} - \mathbf{A}\bar{\mathbf{y}} - \mathbf{b})^T(\bar{\mathbf{x}} - \mathbf{A}\bar{\mathbf{y}} - \mathbf{b})) = \operatorname{tr}\left(\mathcal{E}\left((\bar{\mathbf{x}} - \mathbf{A}\bar{\mathbf{y}} - \mathbf{b})(\bar{\mathbf{x}} - \mathbf{A}\bar{\mathbf{y}} - \mathbf{b})^T\right)\right) =$$
$$\operatorname{tr}\left(\mathbf{P}_{xx} + \mathbf{A}(\mathbf{P}_{yy} + \mu_y\mu_y^T)\mathbf{A}^T + (\mathbf{b} - \mu_x)(\mathbf{b} - \mu_x)^T + 2\mathbf{A}\mu_x(\mathbf{b} - \mu_x)^T - 2\mathbf{A}\mathbf{P}_{yx}\right), \quad (3.12)$$

where tr(·) denotes a trace of matrix. Minimum of (3.12) is found by taking a derivative of the objective function with respect to $\mathbf{A}$ and $\mathbf{b}$ and declaring it to be equal to zero:

$$\begin{aligned} \frac{\partial J_{LMS}}{\partial \mathbf{A}} &= 2\mathbf{A}(\mathbf{P_{yy}} + \mu_y \mu_y^T) + 2(\mathbf{b} - \mu_x)\mu_y^T - 2\mathbf{P}_{xy} &= 0, \\ \frac{\partial J_{LMS}}{\partial \mathbf{b}} &= 2(\mathbf{b} - \mu_x) + 2\mathbf{A}\mu_y &= 0. \end{aligned} \tag{3.13}$$

Set of equation (3.13) is solved by:

$$\mathbf{A} = \mathbf{P}_{xy}\mathbf{P}_{yy}^{-1}, \qquad \mathbf{b} = \mu_x - \mathbf{P}_{xy}\mathbf{P}_{yy}^{-1}\mu_y \tag{3.14}$$

Inserting (3.14) into (3.11) provides a closed form for the linear mean square estimate:

$$\hat{\bar{\mathbf{x}}}_{LMS}(\bar{\mathbf{y}}) = \mu_x + \mathbf{P}_{xy}\mathbf{P}_{yy}^{-1}(\bar{\mathbf{y}} - \mu_y). \tag{3.15}$$

The error covariance associated with the mean square estimate is expressed as:

$$\begin{aligned} &\mathbf{P}_{\hat{\bar{\mathbf{x}}}_{LMS}} = \mathcal{E}\left(\left(\bar{\mathbf{x}} - \hat{\bar{\mathbf{x}}}_{LMS}(\bar{\mathbf{y}})\right)\left(\bar{\mathbf{x}} - \hat{\bar{\mathbf{x}}}_{LMS}(\bar{\mathbf{y}})\right)^T\right) = \\ &\mathcal{E}\left(\left((\bar{\mathbf{x}} - \mu_x) - \mathbf{P}_{xy}\mathbf{P}_{yy}^{-1}(\bar{\mathbf{y}} - \mu_y)\right)\left((\bar{\mathbf{x}} - \mu_x) - \mathbf{P}_{xy}\mathbf{P}_{yy}^{-1}(\bar{\mathbf{y}} - \mu_y)\right)^T\right) = \mathbf{P}_{xx} - \mathbf{P}_{xy}\mathbf{P}_{yy}^{-1}\mathbf{P}_{yx} \end{aligned} \tag{3.16}$$

These findings directly apply to the LKF and play a critical role in formulating the LKF algorithm.

3.3.4 Algorithm

Drawing upon the foundational theory outlined in Section 3.2 and synthesizing the results from Sections 3.3.3 and 3.3.1, this section lays out the algorithmic architecture of the linear Kalman filter (LKF).

Given a predetermined conditional PDF of the state as:

$$p\left(\mathbf{x}[t]|\mathcal{D}^{t-1}\right) = \mathcal{N}\left(\hat{\mathbf{x}}[t|t-1], \mathbf{P}[t|t-1]\right).$$

By utilizing the measurement model, the joint conditional PDF of both the state and the output can be articulated as:

$$p\left(\left[\begin{array}{c}\mathbf{x}[t]\\ \mathbf{y}[t]\end{array}\right]\middle|\mathcal{D}^{t-1}\right) = \mathcal{N}\left(\left[\begin{array}{c}\hat{\mathbf{x}}[t|t-1]\\ \hat{\mathbf{y}}[t|t-1]\end{array}\right], \left[\begin{array}{cc}\mathbf{P}[t|t-1] & \mathbf{P}[t|t-1]\mathbf{C}^T\\ \mathbf{C}\mathbf{P}[t|t-1] & \mathbf{C}\mathbf{P}[t|t-1]\mathbf{C}^T + \mathbf{R}\end{array}\right]\right). \tag{3.17}$$

Data Update

The formulations for the data-update step, as defined by (3.4), are derived by integrating (3.17) into both (3.15) and (3.16). Thus, the data-updated representation of the estimate is expressed as:

$$\hat{\mathbf{x}}[t|t] = \hat{\mathbf{x}}[t|t-1] + \mathbf{L}[t](\mathbf{y}[t] - \underbrace{\mathbf{C}\hat{\mathbf{x}}[t|t-1] - \mathbf{D}\mathbf{u}[t]}_{\hat{\mathbf{y}}[t|t-1]}),$$

$$\mathbf{L}[t] = \mathbf{P}[t|t-1]\mathbf{C}^T\left(\mathbf{C}\mathbf{P}[t|t-1]\mathbf{C}^T + \mathbf{R}\right)^{-1}.$$

Here, $\mathbf{L}[t]$ is designated as the Kalman gain. Concurrently, the covariance associated with the estimate is updated as:

$$\mathbf{P}[t|t] = \mathbf{P}[t|t-1] - \mathbf{L}[t]\mathbf{C}\mathbf{P}[t|t-1].$$

Time Update

The time-update step (3.5) applies the transition model for the mean value, as given by (3.7), and can be concisely represented by:

$$\hat{\mathbf{x}}[t+1|t] = \mathbf{A}\hat{\mathbf{x}}[t|t] + \mathbf{B}\mathbf{u}[t].$$

Correspondingly, the expression governing the time-updated covariance of the estimate employs the derived covariance dynamics from (3.9), resulting in:

$$\mathbf{P}[t+1|t] = \mathbf{A}\mathbf{P}[t|t]\mathbf{A}^T + \mathbf{Q}.$$

The procedural steps, coupled with the associated PDFs of the random variables, are illustrated in Fig. 3.2. The LKF algorithm's iterative cycle is triggered by sequentially incoming measurements. The process initiates with a data update, subsequently availing the resulting estimate to the control system. This is followed by the time update phase, post which the algorithm awaits the next measurement input. Crucially, the decoupling of the estimation into two distinct steps is feasible due to the underlying assumption of uncorrelation between the process and measurement noise. In situations where $\mathbf{S} \neq \mathbf{0}$, the estimation would necessitate a unified approach, culminating in a Riccati equation formulation.

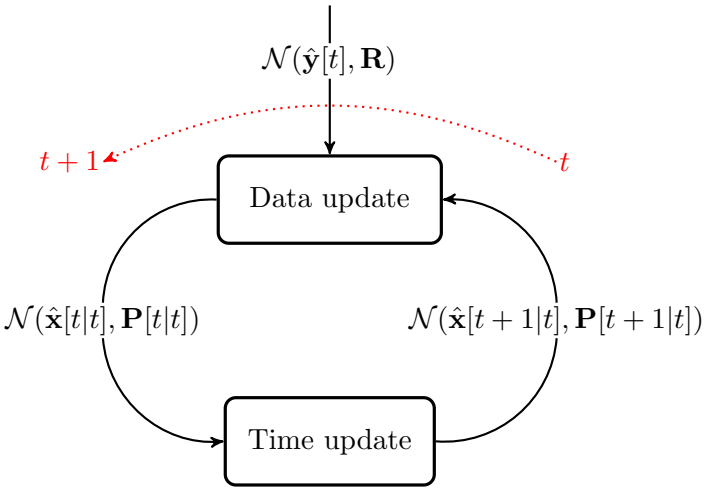


Figure 3.2: Kalman filter scheme

3.3.5 Practical Aspects

Real-world applications often pose the challenge of ascertaining the precise covariance values for both process and measurement noise. These unknowns render their determination nontrivial, leading to a common approach where they serve as tuning parameters. To streamline the tuning process, both covariance matrices are frequently configured to be diagonal.

The process noise covariance, $\mathbf{Q}$, serves as a metric of trust in the model. A smaller diagonal entry in $\mathbf{Q}$ signifies a heightened trust in the associated state variable's transition model. This implies that the estimator would place a higher reliance on the model predictions. Analogously, a smaller entry in the measurement noise covariance matrix, $\mathbf{R}$, underscores an increased confidence in the corresponding sensor's measurement, directing the estimator to lean more towards actual sensor readings.

Crucially, it is the relative proportion between the elements of **Q** and **R** that dictates the behavior of the estimator rather than their absolute magnitudes. The calibration of these parameters revolves around striking a balance between the smoothness of the state estimate (characterized by a small **Q** and large **R**) and its responsiveness (typified by a large **Q** and small **R**).

Chapter 4

Model Predictive Control

In the realm of optimal control problem formulation, there are typically two primary methodologies based on the element subjected to optimization. The *indirect approach* aims to optimize performance through the derivation of an optimal controller. Once this optimal controller is identified, it remains fixed, negating the need for subsequent optimization. While this approach is computationally efficient, its versatility is limited by the fixed nature of the derived controller. Conversely, the *direct approach* directly optimizes control actions. This method necessitates online optimization, implying that an algorithm capable of real-time optimization must be incorporated. Within the direct approach domain, MPC stands out as a particularly prominent methodology.

MPC has witnessed a surge in its application over the past decade. Its origins can be traced back to the process control field, where it begun to be utilized in chemical plant operations starting in the 1980s. Given the inherently slow dynamics of chemical processes, there was no pressing need for rapid control action computation, making the computationally-intensive MPC a viable choice for such scenarios. However, as computational capabilities advanced, coupled with algorithmic refinements, MPC began to permeate other sectors. This expansion included fields characterized by rapid dynamics, such as automotive and UAV industry.

4.1 Discrete-Time Optimal Control Problem on Finite Horizon

To delve deeper into the nuances of MPC, one must first familiarize with the foundational concept of *discrete-time optimal control problem on finite horizon*. In this setup, envision a system regulated by a digital feed-forward controller over a span of N iterations. The primary objective is to determine an optimal sequence of control actions, $\mathbf{u}[t], \ldots, \mathbf{u}[t+N]$, that adheres to the designated objective function $J(\cdot)$. This objective function effectively acts as a metric, or norm, to assess the quality of control, gauging performance based on a combination of the system's states and control actions:

$$J\left(\mathbf{u}[t], \ldots, \mathbf{u}[t+N], \mathbf{x}[t], \ldots, \mathbf{x}[t+N]\right). \tag{4.1}$$

However, directly tasking a generic optimization solver to minimize the objective function (4.1), without acknowledging the intrinsic interdependencies amongst optimization variables, would be useless. These interdependencies stem from the system's inherent transition model:

$$\mathbf{x}[k+1] = \mathbf{f}(\mathbf{x}[k], \mathbf{u}[k]), \qquad k = t, \ldots, t+N-1. \tag{4.2}$$

In accordance with the recursive nature of (4.2), a defined initial state is vital:

$$\mathbf{x}[t] = \mathbf{x}_0. \tag{4.3}$$

Thus, equations (4.2) and (4.3) serve as mandatory constraints, that specify the dynamical context of the task.

For enhanced similarity to real-world scenario, incorporating additional constraints is often deemed advantageous. Recognizing that every actuator possesses inherent limitations, passing these constraints to the solver is beneficial:

$$\mathbf{u}_{min} \leq \mathbf{u}[k] \leq \mathbf{u}_{max}, \qquad k = t, \ldots, t+N-1. \tag{4.4}$$

Furthermore, certain system states may threaten stability, particularly pertinent in the UAV context. The Safe Flight Envelope (SFE), as outlined in [15], defines a set of conditions that prevent loss of control. Implementing SFE safeguards ensures the UAV's operational parameters remain within this safe envelope. A pragmatic approach within the MPC paradigm involves constraining states to remain encapsulated within the SFE:

$$\mathbf{x}_{min} \leq \mathbf{x}[k] \leq \mathbf{x}_{max}, \qquad k = t, \ldots, t+N-1. \tag{4.5}$$

One of MPC's standout attributes is its systematic capability to impose constraints on both the control actions and the system states. While alternative control strategies might offer similar functionalities, their execution lacks the finesse and clarity inherent to MPC. Traditional controllers, when pushed to their operational limits, tend to underperform compared to their output under more moderate conditions [1]. This underscores MPC's performance consistency, irrespective of the proximity of control actions and states to their saturation thresholds.

Combining the objective function (4.1) with constraints (4.2), (4.3), (4.4), and (4.5) culminates in the comprehensive formulation of the discrete-time optimal control problem on a finite horizon:

$$\begin{aligned} \min_{\substack{\mathbf{u}[t],\ldots,\mathbf{u}[t+N-1] \\ \mathbf{x}[t],\ldots,\mathbf{x}[t+N]}} \quad & J\left(\mathbf{u}[t],\ \ldots, \mathbf{u}[t+N-1], \mathbf{x}[t], \ldots, \mathbf{x}[t+N]\right), \\ \text{s.t.} \quad & \mathbf{x}[k+1] = f(\mathbf{x}[k], \mathbf{u}[k]), \\ & \mathbf{x}[t] = \mathbf{x}_0, \\ & \mathbf{u}_{min} \leq \mathbf{u}[k] \leq \mathbf{u}_{max}, \\ & \mathbf{x}_{min} \leq \mathbf{x}[k] \leq \mathbf{x}_{max}, \qquad k = t, \ldots, t+N-1. \end{aligned} \tag{4.6}$$

It's crucial to note that the control action $\mathbf{u}[t+N]$ can be excluded from the optimization problem's formulation since it doesn't influence states within an N sample horizon. As alluded to earlier, the sequence resulting from (4.6) possesses an open-loop characteristic. The resulting control actions function as feed-forward directives, while states might act as a reference trajectory.

4.2 MPC Control Loop

In order to establish a feedback controller, the discrete-time optimal control problem is solved in each control iteration. This involves setting the initial condition $\mathbf{x}_0$ to the current state estimate $\hat{\mathbf{x}}[t]$, followed by the solution of the optimization problem (4.6). Despite obtaining full sequence of control actions and trajectory, only the first control action $\mathbf{u}[t]$ is executed, rendering the remainder redundant. Consequently, from the user's perspective, MPC

[1] A good example is a common PID and the heuristic solution of anti-windup

functions as a black box that takes the current state as input and returns the appropriate control action. The distinction between MPC and state feedback may not be apparent to the user, as in certain cases, the MPC solution can yield an analytical solution equivalent to state feedback. This assertion can be formally concluded through proof in 4.4.4.

The parameter N, indicating length of the open-loop sequence, is referred to as the *prediction horizon* in the context of MPC. Given the typical continuity of the controlled system, it is appropriate to interpret the prediction horizon not only in terms of iterations, but also as a span of continuous time. Thus, the continuous prediction horizon is computed as $N_t = N\,T_s$, where T_s denotes the discretization interval of the transition model.

The computational demand of MPC is notably significant due to the optimization being performed in every iteration. The dimensionality of the optimization problem is inherently determined by the prediction horizon. Thus, reducing the prediction horizon diminishes the computational burden of MPC. Nonetheless, shortening the continuous prediction horizon inevitably sacrifices the consideration of the further-ahead future, resulting in compromised performance. One could argue that this can be tackled by increasing the discretization period T_s. This however again compromises performance, since the precision of the model is reduced and the controller looses the ability to compensate for fast dynamics. Hence, a common technique aimed at reducing computational burden of the MPC while not having significant impact on performance, is presented.

4.2.1 Reduction of Dimensions

In the context of a discrete-time optimal control problem, the reliability of predicted states and associated control actions diminishes as the prediction horizon extends further into the future. This assertion is anchored in two primary considerations:

1. The transition model employed for predictive purposes is inherently imperfect. As such, errors within the model are cumulatively integrated, magnifying with each subsequent predicted state. This progressive accumulation renders the control actions aligned with these states progressively less relevant.
2. Inherently, the future remains an unpredictable entity. Certain environmental factors remain beyond our control and evade straightforward modeling. Within the UAV context, these factors may encompass unpredictable disturbances, such as wind gusts, or unforeseen shifts in reference points.

Supporting this notion, empirical data from simulations and experiments frequently demonstrates a convergence of control actions towards a steady-state value as predictions reach further into the future. Consequently a compelling idea to allocate more emphasis to control actions intended for the imminent future, while de-prioritizing actions intended for more distant time horizons arises.

In response, one can introduce a *control horizon* $N_c \leq N$. So far, the prediction horizon determined the number of future states to be optimized and matched the number of optimized control actions. Analogously, the control horizon governs the number of consecutive control actions available for shaping the desired behavior. This is manifested in the modification of the control actions constraint within the original optimization problem (4.6) as follows:

$$\begin{aligned} \mathbf{u}_{min} \leq \mathbf{u}[j] \leq \mathbf{u}_{max}, \qquad & j = t, \ldots, t + N_c - 1, \\ \mathbf{u}[l] = 0, \qquad & l = t + N_c, \ldots, t + N - 1. \end{aligned}$$

Consequently, this technique reduces the number of optimization variables, while retaining the same time frame for penalizing states. Extensive experimentation demonstrates that this approach substantially reduces computational resource demands without substantially compromising performance. The conserved resources can then be redirected towards extending the prediction horizon, which significantly outweighs the marginal performance decline due to the shortened control horizon.

4.3 Context

While MPC embraces numerous sub-variants, the foundational concept delineated earlier remains a constant. The chief differentiation between these subclasses lies in the nature of the transition model, $f(\mathbf{x}[t], \mathbf{u}[t])$, which inherently shapes the constraints of the associated optimization problem. When the model is linear, it's categorized under *Linear Model Predictive Control* (LMPC). In contrast, a nonlinear state transition model designates the method as *Nonlinear Model Predictive Control* (NMPC).

Parallel to the transition model's influence, the choice of the objective function provides another dimension of classification. The flexibility of this function allows customization to specific tasks; however, its formulation often poses challenges. Unsuitable design could result in a non-convex function, trapping solvers in local optima, or potentially leading to undesirable behaviour. Therefore, common objective functions are often based on the notion of p-norms, which measure the distance between predicted and target states, as well as the control effort. As depicted in Table 4.1, each p-norm possesses a unique statistical justification tailored for varied applications. For instance, scenarios demanding heightened robustness frequently resort to the l_1 norm [38].

p-norm	name	definition	minimizer
l_1	Manhattan norm	$\|\|\mathbf{x}\|\|_1 := \sum_{i=1}^{n} \|x_i\|$	median
l_2	Euclidean norm	$\|\|\mathbf{x}\|\|_2 := \sqrt{\mathbf{x}^T\mathbf{x}}$	mean
l_∞	Infinity-norm	$\|\|\mathbf{x}\|\|_\infty := \max(\|x_1\|, \ldots, \|x_n\|)$	mid-range

Table 4.1: Overview of commonly used p-norms and their minimizers

Objective function based on l_1 or l_∞ norm can be formulated as a Linear Programming (LP), while l_2 norm leads to Quadratic Programming (QP). Consequently, in scenarios involving the previously mentioned objective functions paired with LMPC, the resulting problem is convex. A plethora of QP or LP solvers can be therefore applied, allowing for primarily focus on system modeling, horizon selection, and objective function parameter tuning, rather than on optimization solver implementation.

Contrastingly, Nonlinear Model Predictive Control (NMPC) poses a more intricate optimization problem. Even in the presence of a convex objective function, the introduction of nonlinear constraints most often results in the final problem being in non-convex domain, demanding sophisticated optimization strategies. Predominantly, such strategies employ Newton-type optimization schemes encapsulated within shooting or collocation methods [26].

Ensuring precise UAV control down to orientation and thrust levels necessitates a controller operating in time periods of tens of milliseconds. Achieving this temporal precision

is more seamless with LMPC. Given the assigned task of tracking target UAV's position, it becomes reasonable to minimize the average tracking error, thus favoring a quadratic objective function. Consequently, rest of this sections will predominantly focus on the nuances of linearly constrained quadratic cost MPC - QP-MPC.

4.4 Formulations of QP-MPC

Building on the previously defined criteria of a quadratic objective function and a linear model, these parameters can be explicitly reflected in the foundational formulation (4.6), resulting in:

$$\begin{aligned} \min_{\substack{\mathbf{u}[t],\dots,\mathbf{u}[t+N-1]\\ \mathbf{x}[t],\dots,\mathbf{x}[t+N]}} \quad & \frac{1}{2}\sum_{k=t}^{t+N-1}\left(\,\mathbf{x}^T[k]\mathbf{Q}\mathbf{x}[k]+\mathbf{u}^T[k]\mathbf{R}\mathbf{u}[k]\right)+\frac{1}{2}\mathbf{x}^T[t+N]\mathbf{S}\mathbf{x}[t+N], \\ \text{s.t.} \quad & \mathbf{x}[k+1]=\mathbf{A}\mathbf{x}[k]+\mathbf{B}\mathbf{u}[k], \\ & \mathbf{x}[t]=\mathbf{x}_0, \\ & \mathbf{u}_{min}\le\mathbf{u}[k]\le\mathbf{u}_{max}, \\ & \mathbf{x}_{min}\le\mathbf{x}[k]\le\mathbf{x}_{max}, \qquad k=t,\dots,t+N-1. \end{aligned} \tag{4.7}$$

The optimization problem (4.7) is often called *Quadratic Programming Optimal Control Problem* (QP-OCP). The summative component within the objective function, termed the *running cost*, primarily penalizes control actions and states, excluding the terminal state. The emphasis on penalizing the terminal state, described as the *terminal cost*, tends to be more pronounced. This is attributable to the terminal state's pivotal role in approximating the state's evolution beyond the horizon. To ensure strict convexity of the objective function, matrices $\mathbf{Q}$ and $\mathbf{S}$ must be positively semi-definite ($\mathbf{Q},\mathbf{S}\succeq 0$), while matrix $\mathbf{R}$ should be positively definite ($\mathbf{R}\succ 0$).

For effective deployment of a universal QP solver, the aforementioned formulation demands transformation into a standard QP configuration:

$$\begin{aligned} \min_{\mathbf{z}} \quad & \mathbf{z}^T\mathbf{W}\mathbf{z}+\mathbf{c}^T\mathbf{z}, \\ \text{s.t.} \quad & \mathbf{V}\mathbf{z}\le\mathbf{b}. \end{aligned} \tag{4.8}$$

Certain solvers are designed to accommodate equality constraints. Clearly single equality constraint is functionally equivalent to a pair of inequalities:

$$\mathbf{V}\mathbf{z}=\mathbf{d} \quad\Leftrightarrow\quad (\mathbf{V}\mathbf{z}\le\mathbf{d} \quad\wedge\quad \mathbf{d}\le\mathbf{V}\mathbf{z})\,.$$

Therefore, in the sake of comprehensive analysis, we shall consider the possibility of equality constraints in subsequent discussions, thus not constraining our exploration to inequalities exclusively.

4.4.1 Simultaneous (Sparse) Formulation

Let us stack states and control actions to following vectors:

$$\bar{\mathbf{x}}^T=\left[\;\mathbf{x}^T[t+1]\quad\dots\quad\mathbf{x}^T[t+N]\;\right], \qquad \bar{\mathbf{u}}^T=\left[\;\mathbf{u}^T[t]\quad\dots\quad\mathbf{u}^T[t+N-1]\;\right] \tag{4.9}$$

and similarly the cost-matrices:

$$\bar{\mathbf{Q}} = \begin{bmatrix} \mathbf{Q} & & & \\ & \ddots & & \\ & & \mathbf{Q} & \\ & & & \mathbf{S} \end{bmatrix}, \qquad \bar{\mathbf{R}} = \begin{bmatrix} \mathbf{R} & & \\ & \ddots & \\ & & \mathbf{R} \end{bmatrix}$$

and also the state-space matrices:

$$\bar{\mathbf{A}} = \begin{bmatrix} \mathbf{0} & & & \\ \mathbf{A} & \mathbf{0} & & \\ & \ddots & \ddots & \\ & & \mathbf{A} & \mathbf{0} \end{bmatrix}, \qquad \bar{\mathbf{B}} = \begin{bmatrix} \mathbf{B} & & \\ & \ddots & \\ & & \mathbf{B} \end{bmatrix}, \qquad \bar{\mathbf{A}}_0 = \begin{bmatrix} \mathbf{A} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix}. \tag{4.10}$$

Optimization problem (4.7) can be then rearranged using (4.9) - (4.10) into form (4.8) as following:

$$\begin{aligned} \min_{\bar{\mathbf{x}},\bar{\mathbf{u}}} \quad & \frac{1}{2} \begin{bmatrix} \bar{\mathbf{x}} \\ \bar{\mathbf{u}} \\ 1 \end{bmatrix}^T \begin{bmatrix} \bar{\mathbf{Q}} & \mathbf{0} & \bar{\mathbf{q}} \\ \mathbf{0} & \bar{\mathbf{R}} & \bar{\mathbf{r}} \\ \bar{\mathbf{q}}^T & \bar{\mathbf{r}}^T & 0 \end{bmatrix} \begin{bmatrix} \bar{\mathbf{x}} \\ \bar{\mathbf{u}} \\ 1 \end{bmatrix}, \\ \text{s.t.} \quad & \bar{\mathbf{x}} = \bar{\mathbf{A}}\bar{\mathbf{x}} + \bar{\mathbf{B}}\bar{\mathbf{u}} + \bar{\mathbf{A}}_0\mathbf{x}[t], \\ & \mathbf{x}[t] = \mathbf{x}_0, \\ & \bar{\mathbf{u}}_{min} \leq \bar{\mathbf{u}} \leq \bar{\mathbf{u}}_{max}, \\ & \bar{\mathbf{x}}_{min} \leq \bar{\mathbf{x}} \leq \bar{\mathbf{x}}_{max}. \end{aligned} \tag{4.11}$$

Here $\bar{\mathbf{u}}_{min}, \bar{\mathbf{x}}_{min}, \bar{\mathbf{u}}_{max}$ and $\bar{\mathbf{x}}_{max}$ are vectors of stacked lower and upper limits respectively and $\bar{\mathbf{r}}, \bar{\mathbf{q}}$ are linear coefficients corresponding to $\mathbf{c}$ term in (4.8). Form (4.11) is very flexible. Notably, the bounds placed on both states and control actions, along with the quadratic and linear coefficients, can be adjusted independently for each sample. Additionally, even the underlying model can be modified on a per-sample basis, underscoring the adaptability of this formulation.

4.4.2 Sequential (Dense) Formulation

One could argue that given the initial state, transition model and sequence of control actions, the future states can be determined. This is a key idea of the alternative, *sequential*, formulation. By considering the linear transition model, it becomes evident that a state can be expressed as a function of the preceding state and the applied control action:

$$\mathbf{x}[t+1] = \mathbf{A}\mathbf{x}[t] + \mathbf{B}\mathbf{u}[t].$$

Taking this notion a step further, by substituting for $\mathbf{x}[t+1]$, we derive:

$$\mathbf{x}[t+2] = \mathbf{A}\mathbf{x}[t+1] + \mathbf{B}\mathbf{u}[t+1] = \mathbf{A}^2\mathbf{x}[t] + \mathbf{A}\mathbf{B}\mathbf{u}[t] + \mathbf{B}\mathbf{u}[t+1].$$

This process can be extended, resulting in the ability to express an arbitrary state $\mathbf{x}[t+n]$ as a function of any previous state $\mathbf{x}[t]$ and the sequence of control inputs from that moment until iteration $t+n$:

$$\mathbf{x}[t+n] = \mathbf{A}^n\mathbf{x}[t] + \mathbf{A}^{n-1}\mathbf{B}\mathbf{u}[t] + \mathbf{A}^{n-2}\mathbf{B}\mathbf{u}[t+1] + \cdots + \mathbf{B}\mathbf{u}[t+n-1].$$

This relation can be used to express whole trajectory. Upon being organized in matrix form, a coherent structure emerges:

$$\underbrace{\begin{bmatrix} \mathbf{x}[t+1] \\ \mathbf{x}[t+2] \\ \vdots \\ \mathbf{x}[t+N] \end{bmatrix}}_{\bar{\mathbf{x}}} = \underbrace{\begin{bmatrix} \mathbf{B} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{AB} & \mathbf{B} & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}^{N-1}\mathbf{B} & \mathbf{A}^{N-2}\mathbf{B} & \dots & \mathbf{B} \end{bmatrix}}_{\hat{\mathbf{B}}} \underbrace{\begin{bmatrix} \mathbf{u}[t] \\ \mathbf{u}[t+1] \\ \vdots \\ \mathbf{u}[t+N-1] \end{bmatrix}}_{\bar{\mathbf{u}}} + \underbrace{\begin{bmatrix} \mathbf{A} \\ \mathbf{A}^2 \\ \vdots \\ \mathbf{A}^N \end{bmatrix}}_{\hat{\mathbf{A}}} \mathbf{x}[t] \tag{4.12}$$

This relationship is useful for eliminating states in the objective function of the simultaneous formulation (4.11), resulting in:

$$\begin{aligned} \tilde{J}(\bar{\mathbf{u}}, \mathbf{x}[t]) = \frac{1}{2}\left(\hat{\mathbf{B}}\bar{\mathbf{u}} + \hat{\mathbf{A}}\mathbf{x}[t]\right)^T \bar{\mathbf{Q}}\left(\hat{\mathbf{B}}\bar{\mathbf{u}} + \hat{\mathbf{A}}\mathbf{x}[t]\right) + \frac{1}{2}\bar{\mathbf{u}}^T\bar{\mathbf{R}}\bar{\mathbf{u}} + \frac{1}{2}\mathbf{x}^T[t]\mathbf{Q}\mathbf{x}[t] = \\ \frac{1}{2}\bar{\mathbf{u}}^T \underbrace{\left(\hat{\mathbf{B}}^T\bar{\mathbf{Q}}\hat{\mathbf{B}} + \bar{\mathbf{R}}\right)}_{\mathbf{H}} \bar{\mathbf{u}} + \mathbf{x}^T[t] \underbrace{\hat{\mathbf{A}}^T\bar{\mathbf{Q}}\hat{\mathbf{B}}}_{\mathbf{F}^T} \bar{\mathbf{u}} + \frac{1}{2}\mathbf{x}^T[t]\left(\hat{\mathbf{A}}^T\bar{\mathbf{Q}}\hat{\mathbf{A}} + \mathbf{Q}\right)\mathbf{x}[t] \end{aligned} \tag{4.13}$$

Since the current state $\mathbf{x}[t]$ is a given, the final element of (4.13) is constant. While its omission alters the optimum, the optimizer remains unaffected. The identity (4.12) can be also leveraged in the constraints on states. The formulation of sequential MPC takes the following shape:

$$\begin{aligned} \min_{\bar{\mathbf{u}}} \quad & \frac{1}{2}\bar{\mathbf{u}}^T\mathbf{H}\bar{\mathbf{u}} + \mathbf{x}^T[t]\mathbf{F}^T\bar{\mathbf{u}}, \\ \text{s.t.} \quad & \mathbf{x}[t] = \mathbf{x}_0, \\ & \bar{\mathbf{u}}_{min} \leq \bar{\mathbf{u}} \leq \bar{\mathbf{u}}_{max}, \\ & \bar{\mathbf{x}}_{min} \leq \hat{\mathbf{B}}\bar{\mathbf{u}} + \hat{\mathbf{A}} \leq \bar{\mathbf{x}}_{max}. \end{aligned} \tag{4.14}$$

Presented procedure is often referred to as *condensation*.

4.4.3 Formulation Choice

A comparison between the simultaneous (4.11) and sequential (4.14) formulations primarily underscores differences in the model incorporation methodologies. The simultaneous approach integrates the model by enforcing relations between the optimized variables via constraints. Conversely, the sequential formulation embeds the model directly into the objective function, optimizing solely over control actions.

At first glance, the sequential formulation might appear to be the superior choice due to its dimensionality reduction feature. However, this assumption is misleading. The simultaneous method's advantage lies in its generation of sparse matrices. Solvers can exploit such sparse structures, resulting in enhanced efficiency compared to case of non-sparse problems of equivalent dimensions. In [34] it is concluded that while the dense formulation usually outperforms in standard scenarios, its efficiency degrades with respect to longer horizons. Specifically, using the Interior Point Method [29], the dense formulation's scaling with horizons is cubic. However, for MPC challenges necessitating extended horizons, the sparse approach provides significant speedup due to its linear scaling relative to horizons. Thus, the choice of formulation is tied to the problem's nuances.

Furthermore, as highlighted in [19], the decision between sparse (simultaneous) and dense (sequential) formulations need not be binary. Over recent years, the development of *partial condensing* algorithms and solvers has affected this choice. As Axehill noted:

> ‘control engineers working with MPC should not ask whether to formulate the problem in a sparse or dense way, but instead what is the correct level of sparsity for the problem at hand to obtain maximum performance‘ [19].

The essence of partial condensation is the reformulation of the original problem into multiple smaller dense blocks. When these blocks are diagonally merged, the resulting matrix, formulating the optimization problem, emerges. The matrix's sparsity is intuitively dependent on the dense block's dimensions. A minimal block size yields a formulation identical to the common sparse one. Conversely, maximizing the block size culminates in a formulation mirroring the dense one.

4.4.4 Solution of Unconstrained QP-MPC

While QP-MPC typically necessitates iterative solutions, a closed-form solution is obtainable when neither the control actions nor states are constrained. Given the convex nature of the objective function, it possesses a global extremum. When the optimization set remains unconstrained, the optimization problem becomes a matter of locating the gradient of the objective function where it equates to the zero vector:

$$\nabla_{\bar{\mathbf{u}}}\tilde{J}(\bar{\mathbf{u}}, \mathbf{x}[t]) = \mathbf{H}\bar{\mathbf{u}} + \mathbf{F}\mathbf{x}[t] = \mathbf{0}.$$

This equation naturally yields a sequence of control actions defined by:

$$\bar{\mathbf{u}} = -\mathbf{H}^{-1}\mathbf{F}\mathbf{x}[t].$$

Given that MPC utilizes only the initial control action, the comprehensive formula for the control action to be employed is:

$$\mathbf{u}[t] = -\begin{bmatrix} \mathbf{I} & \mathbf{0} & \dots & \mathbf{0} \end{bmatrix}\mathbf{H}^{-1}\mathbf{F}\mathbf{x}[t].$$

This finding is significant. The unconstrained QP-MPC operates as a unique form of state feedback. Consequently, its deployment is straight-forward. The omission of an optimization solver not only simplifies the implementation but also ensures that computational demands are minimal.

4.4.5 Tracking

So far the objective function was formulated to solve a regulation problem, i.e. the problem of driving the system to the zero state. The task of this thesis is however tracking of target trajectory or state. For this purpose the objective function is modified such that the tracking error $\mathbf{e}[t] = \mathbf{x}_r[t] - \mathbf{x}[t]$ is penalized instead of the state itself. After substitution into the objective function from (4.7), the expression can be further rearanged as described by (4.15). Setting the reference consequently boils down to adjusting linear parameter of state, denoted by $\mathbf{q}$ and augmenting it to the final $\bar{\mathbf{q}}$ from (4.11).

$$\frac{1}{2}\sum_{k=t}^{t+N-1}\left((\mathbf{x}_d[k]-\mathbf{x}[k])^T\mathbf{Q}(\mathbf{x}_d[k]-\mathbf{x}[k])+\mathbf{u}^T[k]\mathbf{R}\mathbf{u}[k]\right)+$$

$$\frac{1}{2}(\mathbf{x}_d[t+N]-\mathbf{x}[t+N])^T\mathbf{S}(\mathbf{x}_d[t+N]-\mathbf{x}[t+N])=$$

$$\overbrace{\frac{1}{2}\mathbf{x}_d^T[t+N]\mathbf{S}\mathbf{x}_d[t+N]}^{\text{const.}}\overbrace{-\mathbf{x}_d^T[t+N]\mathbf{S}}^{\mathbf{q}[N]}\mathbf{x}[t+N]+\frac{1}{2}\mathbf{x}^T[t+N]\mathbf{S}\mathbf{x}[t+N]+$$

$$\sum_{k=t}^{t+N-1}\left(\underbrace{\frac{1}{2}\mathbf{x}_d^T[k]\mathbf{Q}\mathbf{x}_d[k]}_{\text{const.}}\underbrace{-\mathbf{x}_d^T[k]\mathbf{Q}\mathbf{x}[k]}_{\mathbf{q}[k-t]}+\frac{1}{2}\mathbf{x}^T[k]\mathbf{Q}\mathbf{x}[k]+\frac{1}{2}\mathbf{u}^T[k]\mathbf{R}\mathbf{u}[k]\right) \quad (4.15)$$

Chapter 5

Implementation Aspects

Building on the theoretical frameworks of LKF and MPC presented in Chapters 3 and 4 respectively, this chapter delves into their pragmatic implementation. Notably, it demonstrates how the MPC technique can be utilized for trajectory generation. While certain software platforms, like Matlab, offer automated code generation for control loop components, the resulting code is often not very human-readable, making subsequent adjustments and maintenance cumbersome. As an alternative, a manual implementation was carried out in C++. The designated execution environment for the developed code is an onboard computer, such as the Intel NUC, equiped with a Linux operating system.

The practical realization of the system depends on several parameters, the tuning of which largely draws from empirical methods. To facilitate this, the Gazebo-based simulation platform [3] was utilized. Since the actual implementation poses limited challenges, especially when leveraging external libraries, the main objective of this chapter is to offer a pragmatic perspective on methods that were previously elaborated in a theoretical context.

5.1 Model

At the core of every component lies the system's model. It's encapsulated as a class shared across all modules, offering nonlinear, continuous LTI state-space and discrete LTI state-space representations. Furthermore, this centralized model provides interfaces for both reading and modifying system parameters[1].

To simplify both the model and the subsequent components, certain assumptions are made. The dynamics of thrust, for instance, are neglected. It is presumed that the exerted thrust mirrors the desired value precisely, implying a flawless inner control loop. Consequently, this related state is omitted from the model. A keen reader may have noticed in (2.29) that linearizing the model at $\psi = 0$ results in the UAV's yaw angle having no influence over other state variables. Within the linear model, the yaw dynamics operate independently from the other system's states. However, this abstraction remains valid only in proximity to $\psi = 0$. As there is no explicit requirement to manage the UAV's heading, the desired yaw angle defaults to zero. This relies on the flight controller's capability to accurately track this value, allowing the yaw state variable to be entirely removed from the model. Such simplifications not only facilitate the employment of a linear estimator but also enhance the computational efficiency of the MPC.

In the context of system parameters, it is assumed that prior knowledge of these parameters is available. These parameters can be determined by measuring the mechanical attributes of the system. However, obtaining accurate parameters for the flight controller's model can be more challenging. A systematic approach involves conducting a human-supervised flight

[1] This capability offers prospects for adaptive adjustments to the system model in subsequent work.

and then applying linear regression to the acquired data, aiming to minimize the least square error. For a more detailed explanation of the method outlined, refer to [9] and [20].

5.2 Kalman Filter

The LKF was implemented using the *Eigen* linear algebra library, ensuring efficient and numerically stable matrix operations. With the computational elegance offered by *Eigen*, the code becomes rather straightforward, allowing primary focus on tuning of parameters and practical adaptations.

During the LKF implementation, the idea of transitioning away from the near-perfect localization provided by the mocap system emerged. Such a shift could complicate estimation but also paves the way for broader deployment capabilities. The utilized simulation platform, a component of the MRS system [3], can simulate RTK GPS and the position estimate rooted in this localization method. Hence, the LKF was adapted and refined to rely on this estimation rather than the almost exact mocap system.

5.2.1 Tuning

As discussed in Section 3.3.5, empirical tuning is a prevalent approach, particularly when dealing with parameters like noise covariance that, despite their statistical interpretations, are often elusive. Ground truth measurements, essential for noise assessment, are seldom available. This leaves one estimating noise, ironically creating an estimation loop. Similarly, process noise poses challenges due to the inherent unmeasurability of all system states.

Despite these challenges, empirical noise covariance tuning was conducted in the simulator. Striking a balance between rapid response and estimation smoothness is inherently subjective. Ideally, the system's response is swift with minimal steady-state error, but noise should remain low enough to ensure smooth control. This iterative tuning process yielded the following covariance matrices:

parameter	**value**												
		x	y	z	ϕ_l	θ_l	$\dot{x}$	$\dot{y}$	$\dot{z}$	$\dot{\phi}_l$	$\dot{\theta}_l$	θ	ϕ	
$\mathbf{Q}_{KF}$	diag([	1	1	1	1	1	10	10	100000	100	100	1	1	])
$\mathbf{R}_{KF}$	diag([	100	100	100	1	1	])							
		x	y	z	θ	ϕ								

Table 5.1: Overview of Kalman filter's parameters

It should be noted that an inconsistency emerged during the tuning process. The derived model from Chapter 2 predicted that, in a steady state, propeller force should counteract the combined gravitational forces of the UAV and the payload. However, simulation data indicated a marginally larger steady-state control input, prompting the need for an enlarged variance in the vertical velocity process noise, as reflected in $\mathbf{Q}_{KF}$.

5.2.2 Asynchronous Scheme

While the control loop is configured to operate at a consistent frequency, real-world scenarios sometimes induce irregularities, causing control iterations to either delay or preempt. The inherent limitation of the standard LKF, which leans on a discrete state-space model, is its inability to accommodate these inconsistencies, especially during the time-update phase. To tackle this, a dynamic approach was implemented that utilizes the continuous state-space model. By discretizing this model based on the current interval between measurements, it ensures a more adaptive and accurate time-update step. The rest of the algorithm remains unchanged, leading to standard date-update.

5.2.3 Disturbance Estimation

The disturbance estimator stands as a valuable extension to the Kalman filter, providing insights in diverse contexts, whether it's wind disturbances during flight or unexpected environmental interactions. Moreover, it presents arguably the most systematic route to negate steady-state errors when employing an MPC.

However, implementing a disturbance estimator presents its own complexities. Given the UAV's perspective, the load's actions manifest as disturbances. This means that the load's state is inferred from these disturbances, albeit indirectly, by extracting insights from the UAV states and control inputs. A potentially more refined approach might involve introducing a novel state to represent disturbances and basing the load state estimation on this variable. This state variable could be also utilized to resolve steady state error and other issues. While attempts inspired by [20] were initiated, they did not yield efficient outcomes. Future work may revisit and perfect this aspect.

5.3 MPC

Implementing the MPC in a real-world scenario involves addressing a multitude of practical challenges:

1. **Reference Definition**: It's essential to determine the nature of the reference.
   - Is it a single point or an entire trajectory?
   - How feasible are the given references?
   - Are the references known in advance or are they dynamic?
2. **Computational Aspects**:
   - Establishing suitable prediction and control horizons is critical.
   - The choice of MPC formulation and the corresponding solver must align with the specific problem's requirements.
3. **Objective Function Tuning**:
   - Properly tuned objective function is imperative to ensure the system behaves as desired.
4. **Constraints**:
   - Are there specific constraints that the system must adhere to?
   - Implementing these constraints without causing infeasibility is a significant challenge.

This section delves into each of these considerations, providing insights and strategies for effective MPC deployment in real-world scenarios.

5.3.1 Setup

When designing the controller, understanding the operational context and the framework, it's supposed to be part of, is vital. The MRS system, employed in this thesis, already incorporates a trajectory generator, producing reference points and feed-forward control actions. However, this generator does not consider the complexities introduced by a cable-suspended payload, leading to potential misbehaviors. Thus, its utility in this context is limited.

The fundamental UAV control objective is target position tracking. Although this can be achieved without a predefined trajectory, large transitions can amplify tracking errors, leading to actuator saturation or constraint activation. Although MPC technique handles these in general exceptionally well, they can, in some cases, result in ill-conditioned optimization problem and the solver may give up and call the problem unfeasible. Thus, a specialized trajectory generator (TG), conscious of the suspended payload's dynamics, becomes essential. Fig. 5.1 provides an architectural overview of the proposed control framework.

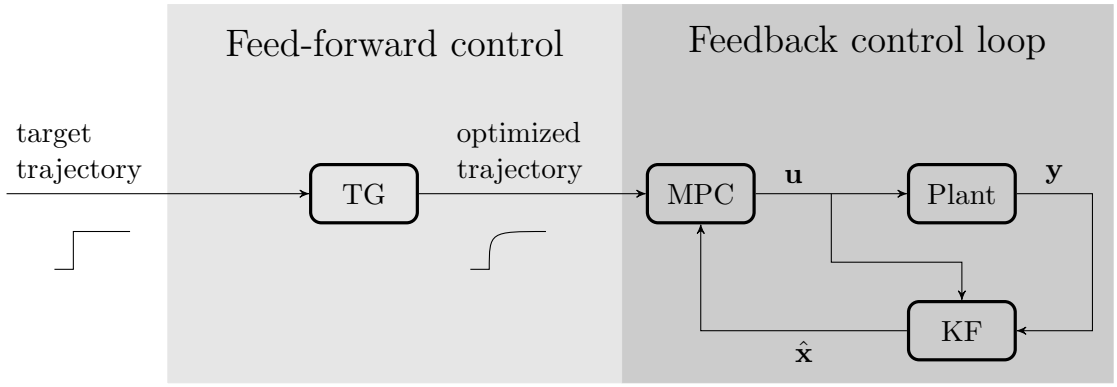


Figure 5.1: Architecture of the control framework

5.3.2 Trajectory Generator

Utilizing MPC as the primary feedback controller simplifies the trajectory generation process. As elaborated in the prior chapter, the sequence of control actions, resulting from the finite discrete-time optimal control problem, inherently adopts an open-loop structure. This sequence not only provides feed-forward commands but also lays the foundation for determining a reference trajectory. This prompts the question: Is an external trajectory generator essential when MPC inherently computes an internal trajectory and feed-forward sequence in every iteration? There are two main relevant arguments:

1. **Single Computation**: The trajectory generation is envisioned as a single, non-periodical computation. Unlike real-time operations, there isn't an immediate and strict deadline for this computation. The UAV is already stabilized by the MPC and therefore delay in trajectory generation doesn't pose significant issue. This enables the optimization of trajectories over extended horizons compared to typical MPC cycles.
2. **Flexible Discretization**: Given that the UAV's dynamics are stabilized by the MPC, there's flexibility to employ a larger model discretization period for trajectory generation. A larger discretization period inherently translates to sparser trajectory points, extending the prediction span. However, this approach has a limitation. Overly large discretization periods can compromise the system's controllability, pushing solvers towards infeasibility.

Consequently, the trajectory generator necessitates its unique parameter tuning process to strike an optimal balance between prediction accuracy and computational efficiency.

5.3.3 Acados and Solvers

Acados stands out as an advanced software package, specifically tailored for solving optimal control and estimation challenges. It boasts a wide variety of algorithms and solvers dedicated to handling an array of optimization problems, with NMPC being a prominent use-case. Most importantly, for the context of this thesis, Acados integrates an array of state-of-the-art QP solvers, encompassing the likes of HPIPM, qpOASES, qpDUNES, and OSQP, under a unified interface.

This unified approach offers a distinct advantage. Traditional methodologies often entail rigorous research to pinpoint an ideal solver, followed by problem formulation tailored to the specific solver interface. Transitioning to another solver, under such circumstances, requires re-implementing the problem to match the new solver's interface, adding to the overhead. Acados efficiently mitigates this challenge. With its unified interface, switching between solvers is seamless, boiled down to a mere argument modification. This facilitates effortless selection of the most performance-optimized solver tailored for the problem at hand.

The Acados input format, termed as OCPQP, is inherently aligned with a sparse formulation. However, flexibility is at the core of Acados, enabling transitions between formulation types. Users can seamlessly transition from a sparse formulation to its dense counterpart through the *condensation* feature. Furthermore, this condensation can be fine-tuned, offering a degree of partial condensation based on user-defined parameters.

Interacting with Acados is intuitive:

1. **Problem Formulation**: Users have the leverage to modify the OCPQP formulation. Common modifications include adjusting objective function parameters or setting references. This is typically performed during initialization and, subsequently, with each iteration to define the initial conditions.
2. **Solver Execution**: Post formulation, the solver can be invoked to determine the solution for the defined problem.

The interface extends the capability to define elements of the OCPQP formulation as time-variant, allowing independent adjustments for every iteration. Moreover, the problem's dimensions can dynamically adapt across the prediction horizon. This particularly fits techniques like control horizon adjustments or input blocking, making their implementation straightforward. In essence, Acados emerges as a versatile tool, characterized by flexibility and practicality, making it indispensable for embedded MPC development endeavors. For a deeper dive, readers can refer to:

[12] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. van Duijkeren, A. Zanelli, *et al.*, *Acados: A modular open-source framework for fast embedded optimal control*, arXiv:1910.13753 [math], Nov. 2020. [Online]. Available: `http://arxiv.org/abs/1910.13753` (visited on 07/29/2023)

5.3.4 Horizons

When calibrating the parameters for MPC, it's essential to begin with setting the prediction and control horizons, in addition to determining the model's discretization period. A widely acknowledged heuristic for the prediction horizon is ensuring its duration encompasses the system's principal dynamics and transient behaviors. Considering that the system's dynamic characteristics are defined in continuous time, it's logical to express the prediction horizon similarly. This leads to a direct relationship for the continuous-time horizon, t_h:

$$t_h = NT_d,$$

where N is the discrete-time horizon, and T_d represents the discretization frequency of the model.

Empirical simulations indicate poor MPC performance when the continuous prediction horizon falls short of the natural period of the pendulum symbolizing the load. An approximate relation for the pendulum's natural frequency, in terms of the cable's length l, is given by:

$$T_l(l) \approx 2\pi\sqrt{\frac{l}{g}}.$$

Derived from the linearized pendulum model, this equation is most accurate for minor amplitude oscillations. Based on these insights, the lower bound for the prediction horizon is $N\,T_d > T_l(l)$. Nevertheless, increasing the continuous time horizon continually enhances performance. The optimal performance, based on empirical observations, is approximately 2.5 times T_l, leading to:

$$NT_d \approx 5\pi\sqrt{\frac{l}{g}}. \tag{5.1}$$

Until now, it has been assumed that both the prediction and control horizons are identical. Typically, while control action sequences converge to a steady state, altering the control horizon or blocking inputs doesn't profoundly affect control performance. However, simulations indicate deviations from this trend. Given the harmonic nature of the load's dynamics and its damping properties, the development of control actions exhibits periodic traits. When the control horizon is extensive, or mirrors the prediction horizon, the MPC confidently executes aggressive control actions, anticipating subsequent compensatory actions to stabilize the load. A truncated control horizon, on the other hand, influences the MPC to adopt a conservative approach, minimizing load swings but at the expense of increased tracking error. These interpretations are consistent with simulated outcomes, revealing peak performance when control and prediction horizons align.

Having established the equivalency of control and prediction horizons without implementing input blocking, the single horizon length, N, can be defined by assessing solver performance and computational constraints. A typical control loop frequency for such control levels is 100 Hz. Allocating computational resources for the Kalman filter and other controller components, based on hardware benchmarks, results in a computational duration of 1 ms. This necessitates control action computations to meet a 9 ms deadline. Optimal configurations are derived through experimentation with various solvers and problem formulations, and are outlined in Table 5.2.

To finalize, the model's discretization period is determined using the empirical optimum (5.1), mirroring synchronous sampling schemes commonly used in non-stationary signals analysis. As a result, the sampling period is tailored to the load dynamics, rendering the MPC

entity	N	N_c	formulation	M	solver	run-time
MPC	100	100	partial condensing	10	HPIPM	8 ms
trajectory generator	250	250	partial condensing	2	HPIPM	150 ms

Table 5.2: Overview of tuned parameters of optimization problems. N denotes prediction horizon, N_c control horizon and M is level of condensation.

adaptive to the cable length. In scenarios with shorter cables and rapid load dynamics, the MPC samples with greater density and a reduced continuous horizon. Conversely, longer cables with slower dynamics lead to sparse sampling but an extended continuous horizon. Accounting for the UAV's dynamics further constrains the discretization period within the bounds of $[T_d^{min}, T_d^{max}]$. Empirical simulation results, determining subjective controller performance, define this interval as $[20\,\text{ms}, 150\,\text{ms}]$. The interrelation between the discretization period and cable length is illustrated in Fig. 5.2.

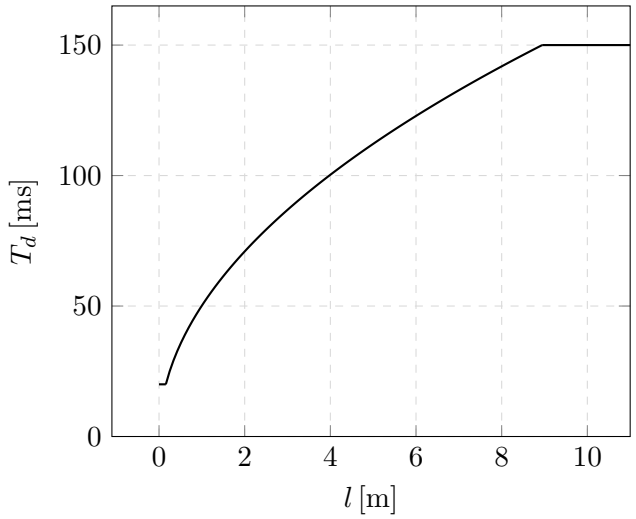


Figure 5.2: Relation between length of the cable l and the discretization period of the model T_d

5.3.5 Tuning of Objective Function

The objective function's parameters are tailored to ensure the resulting behavior aligns with specified requirements. Nevertheless, these requirements can be challenging to articulate explicitly, resulting in the project's inherent vagueness. The subsequent discourse aims to discuss the precise objectives and the associated issues. Specifically, the presented techniques facilitate the resolution of two seemingly contradictory challenges:

1. Tracking the UAV's target position, irrespective of the load's state.
2. Dampening the load's oscillations, without prioritizing the UAV's target position.

These tasks represent the extremities of a continuum. Thus, any relevant objective can be conceptualized as an affine combination of these contrasting goals. Among the listed challenges, the second lacks pragmatic utility, as real-world applications primarily demand accurate target position tracking, underscoring the significance of the first task. However, the

presented methods are dependent on a linear model. This model's validity diminishes as the load's amplitude grows. Consequently, the solution to effectively track the UAV's position requires that the load's amplitude remains low. This implies that the optimal approach, given the constraints of the tools available, requires a synthesis of the aforementioned tasks.

The tuning process of the objective function's parameters, albeit empirical, offers more intuitive engagement compared to other control methods like PID. This stems from the fact that the objective function's parameters are intrinsically linked to the system's states and control actions, ensuring their adjustments are more interpretable. The parameters that yield the desired behavior are included in Table 5.3.

Parameter	**Value**												
		x	y	z	ϕ_l	θ_l	$\dot{x}$	$\dot{y}$	$\dot{z}$	$\dot{\phi}_l$	$\dot{\theta}_l$	$\dot{\theta}$	$\dot{\phi}$	
$\mathbf{Q}_{MPC}$	diag([	100	100	1000	100	100	0	0	0	0	0	0	0	])
$\mathbf{Q}_{TG}$	diag([	1	1	100	100	100	0	0	0	0	0	0	0	])
$\mathbf{R}_{MPC}$	diag([	50	50	0.01	])									
$\mathbf{R}_{TG}$	diag([	0.01	0.01	0.001	])									
		θ_d	ϕ_d	F										

Table 5.3: Overview of parameters of the objective function of the MPC and the trajectory generator (TG)

5.3.6 Steady State Error

The optimization problem's objective function, as described in (4.15), inherently does not ensure a zero steady-state tracking error. One widely recognized strategy to counter this limitation is the incorporation of a *delta input formulation* [37]. However, its effectiveness manifests only when the reference isn't an equilibrium of the model. Under such circumstances, the model indicates that non-zero control actions are required to track the reference without a steady-state error, making the delta input formulation effective. Nonetheless, when the tracking error originates from disturbances, this formulation does not yield requested results. This limitation arises since the disturbance is not integrated into the MPC's predictive modeling of future trajectories. Hence, even with delta formulation, disturbances remain uncompensated. This characteristic is particularly relevant for most practical situations. The desired steady-state is with no exception an equilibrium, considering the under-actuated nature of UAVs. As a result, the steady-state tracking error is predominantly induced by external disturbances (e.g. wind) or model discrepancies (e.g. inaccuracies due to improperly mounted IMUs).

One might propose the integration of a disturbance estimator, expanding the model to encompass it, and subsequently embedding this estimate into the MPC's predictions. However, attempts on implementation of this methodology failed, as discussed in Section 5.2.3. Consequently, a less direct strategy was adopted. By incorporating an additional integrator layer above the MPC, it's possible to adjust the UAV's target position, effectively countering the disturbances and ensuring zero steady-state tracking error. This approach, while partially effective, is not without its challenges and issues. A sluggish integrator may lag in compensating for fast disturbances like wind, while an overly aggressive one could introduce oscillations. Further, it's susceptible to windup, requiring the incorporation of heuristic mechanisms to

mitigate this secondary issue. As implemented, this solution predominantly compensates for offsets in measured attitude and lacks effectiveness in managing fast external disturbances.

5.3.7 Constraints

A fundamental attribute of MPC, as discussed in Section 4.1, is its systematic treatment of constrained control actions and states. This capability safeguards the UAV from venturing beyond the predefined SFE. Additionally, it ensures that the system predominantly stays within a set of states wherein the linear model retains its validity. This is realized through constraints imposed on control actions[2] and velocity.

These constraints are typically enforced in two stages. Primarily, they are integrated into the trajectory generator to prevent any generated trajectory from violating them. Subsequently, these constraints are further embedded within the MPC. In an ideal environment without disturbances and with perfect models, the MPC wouldn't require further constraints, as the trajectory would behave well. However, real-world scenarios often feature disturbances capable of inducing excessive tilting, potentially resulting in mission failure. Consequently, the MPC must possess constraints as well, albeit less strict ones than the trajectory generator. These milder constraints often remain inactive, allowing the UAV to catch up with the target trajectory even if temporary delays occur due to disturbances. It's important to note that the magnitude of these constraints depend on the payload mass, as naturally looser attitude constraints might lead to a descent if the payload is heavy. A comprehensive overview of these constraints is presented in Table 5.4.

entity	constrained variable	value
TG	control action	$0.3\,\mathrm{rad}$
	horizontal velocity	$4\,\mathrm{m\,s^{-1}}$
MPC	control action	$0.5\,\mathrm{rad}$
	horizontal velocity	$5\,\mathrm{m\,s^{-1}}$

Table 5.4: List of constraints for 0.5 kg heavy payload and 2 kg heavy UAV.

It's worth noting that the introduction of these constraints can, in certain situations, introduce infeasibility issues of the optimization problem's solution. Such circumstances could potentially trigger complications during real-world deployment, as an optimal control action might not be determined in that specific iteration of the control loop. Furthermore, the subsequent iteration might not yield a different outcome. This issue can culminate in total system failure, necessitating the activation of fallback controllers. However, this challenge is effectively mitigated through the adoption of *soft constraints*. Instead of enforcing constraints rigidly, as

[2] And consequently attitude.

depicted in (4.11) or (4.14), these constraints are reformulated as:

$$\begin{bmatrix} \mathbf{I} \\ \vdots \\ \mathbf{I} \end{bmatrix} \mathbf{u}_s + \begin{bmatrix} \mathbf{u}_{min} \\ \vdots \\ \mathbf{u}_{min} \end{bmatrix} \leq \bar{\mathbf{u}} \leq \begin{bmatrix} \mathbf{u}_{max} \\ \vdots \\ \mathbf{u}_{max} \end{bmatrix} + \begin{bmatrix} \mathbf{I} \\ \vdots \\ \mathbf{I} \end{bmatrix} \mathbf{u}_s,$$
$$\begin{bmatrix} \mathbf{I} \\ \vdots \\ \mathbf{I} \end{bmatrix} \mathbf{x}_s + \begin{bmatrix} \mathbf{x}_{min} \\ \vdots \\ \mathbf{x}_{min} \end{bmatrix} \leq \bar{\mathbf{x}} \leq \begin{bmatrix} \mathbf{x}_{max} \\ \vdots \\ \mathbf{x}_{max} \end{bmatrix} + \begin{bmatrix} \mathbf{I} \\ \vdots \\ \mathbf{I} \end{bmatrix} \mathbf{x}_s.$$

Here, $\mathbf{x}_s$ and $\mathbf{u}_s$ denote state and control action slack variables respectively. These variables function as decision variables subjected to minimization. However, their associated costs are considerably higher than that of other variables. Consequently, these slack variables generally assume a value of zero, manifesting only in scenarios where no other viable alternatives exist, and where the solution's feasibility is at risk.

5.3.8 Preview

A particularly useful attribute of MPC is its ability to consider the forthcoming evolution of a reference trajectory, provided that such information is available. This characteristic gives rise to an anticipatory behavior within the control system. To harness this distinctive capability, the trajectory generator accepts both a reference point(s) and a corresponding time-stamp(s). This time-stamp represents the intended instant at which the UAV should ideally reach the specified reference point. Consequently, the generated trajectory results in more seamless and less costly transition, with respect to the objective function.

Chapter 6

Experiments

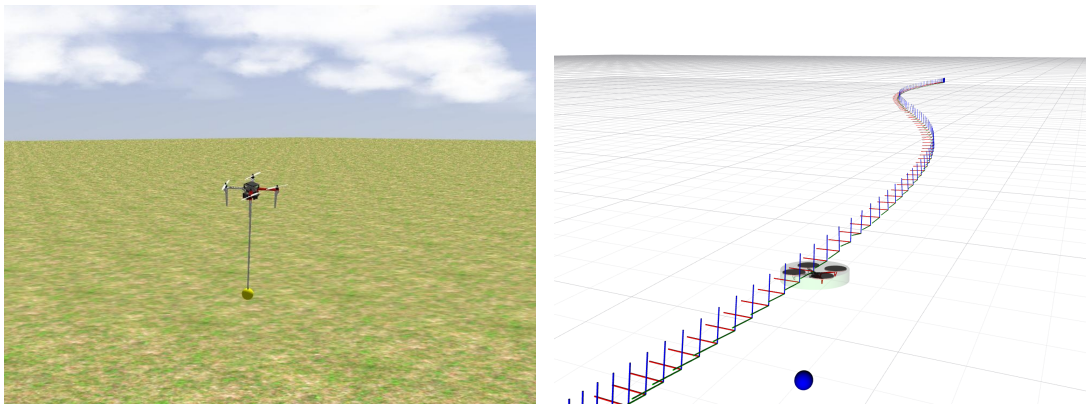
This chapter focuses on the experimental validation of the framework introduced earlier, specifically the parameterization detailed in Section 5.3.5. Firstly, the simulation platform utilized for these tests will be detailed. Subsequent sections will delve into individual tests of each component of the framework. By evaluating each component in isolation, we aim to provide a granular understanding of their respective performances within the broader system. Following this, the integrated framework will be assessed, showcasing its ability to address real-world challenges. Lastly, introduced framework will be compared against a state-of-the-art general-purpose control system.

6.1 Simulator

The subsequent set of experiments were conducted utilizing a simulation platform developed by the MRS at the Czech Technical University in Prague. The MRS system's simulator [3] is built upon the physics simulation engine *Gazebo* [30]. This simulation environment is designed to seamlessly bridge the gap between simulation and real-world application, incorporating real-world complexities such as sensor noise, sensor offset, and imperfections in the flight controller. The findings of [3] affirm the high degree of resemblance between real-world experimental outcomes and simulation results.

The MRS system offers versatile control options across various levels and is capable of implementing lower-level controls. In the context of this thesis, the system provides the tracking of desired UAV attitude and thrust. Developed on the foundation of the *Robot Operating System* (ROS), the architecture of the MRS system is highly modular. Within its modular structure, the controller module has been replaced by the newly introduced implementation.

The subsequent simulations are executed employing a model representative of the DJI F450 platform. This UAV platform weights approximately 2 kg. It has been observed that the intensity of disturbances and the complexity of the control problem tend to increase proportionally with the payload's mass. Nonetheless, the weight of the payload is constrained by the UAV's maximum thrust capacity. When the payload's weight reaches the upper threshold (approximately 1.5 kg for the F450), the UAV is unable to execute maneuvers without experiencing altitude loss. In order to formulate a challenging yet practical control problem, a payload mass of 0.5 kg is chosen. This weight strikes a balance between imposing disturbance on the UAV and allowing the UAV to perform aggressive maneuvers without compromising altitude stability. For the sake of consistency, experiments were consistently conducted with a 1 m long cable, unless explicitly specified otherwise.



(a) Gazebo (b) RViz

Figure 6.1: Graphical user interface of the simulation platform

6.2 Kalman Filter

While assessing the estimator's performance can present challenges in practical scenarios, the simulator offers the advantage of providing ground truth values, enabling direct comparisons. An properly performing Kalman filter's estimation should smoothly track the ground truth state without compromising control smoothness, while simultaneously maintaining a responsiveness to ensure robust flight characteristics. To evaluate these attributes, two types of experiments were conducted.

6.2.1 Common Flight

This experiment verifies the Kalman filter's performance under normal flight conditions. The primary focus of this experiment is on the Kalman filter's behavior, rather than the specifics of control performance. Notably, a poorly performing controller might lead to substantial load oscillations, consequently making the estimation more challenging. Since the Kalman filter utilizes a linear model of the system, it may struggle in the presence of large oscillations. Successful performance in case of such disturbances would suggest an even more effective performance under conditions with smaller oscillations. Thus, the experiment was conducted using a moderately tuned SE(3) controller [35]. The simulated scenario was designed to make the UAV perform position step response in forward direction. The magnitude of the step was empirically selected to generate significant load oscillations. The outcomes of this simulation are illustrated in Figure 6.3.

6.2.2 Effect of Disturbance

While the system typically operates in alignment with its model, it's critical to evaluate the Kalman Filter's robustness when exposed to phenomena not encapsulated within the model. Real-world instances of such scenarios might include strong wind gusts, the payload encountering obstacles, and other unexpected disturbances. An estimator heavily relaying on its model can produce delayed responses, potentially leading to misguided control actions. Consequently, the control system's overall robustness is linked to the estimator's robustness.

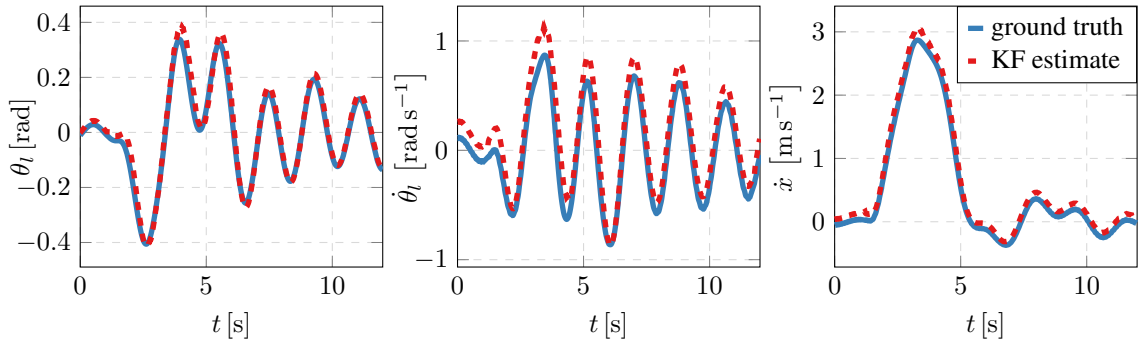


Figure 6.2: Performance of the Kalman filter in the normal flight scenario

For this experiment, the SE(3) controller from the prior test was employed. The introduced disturbance is simulated as an impulsive external force exerted on the payload in the direction of axis $\hat{\mathbf{l}}_x$. Prior to this disturbance, the system remained stationary, having stabilized to its steady-state condition. The outcomes from this simulation are depicted by Fig. 6.3.

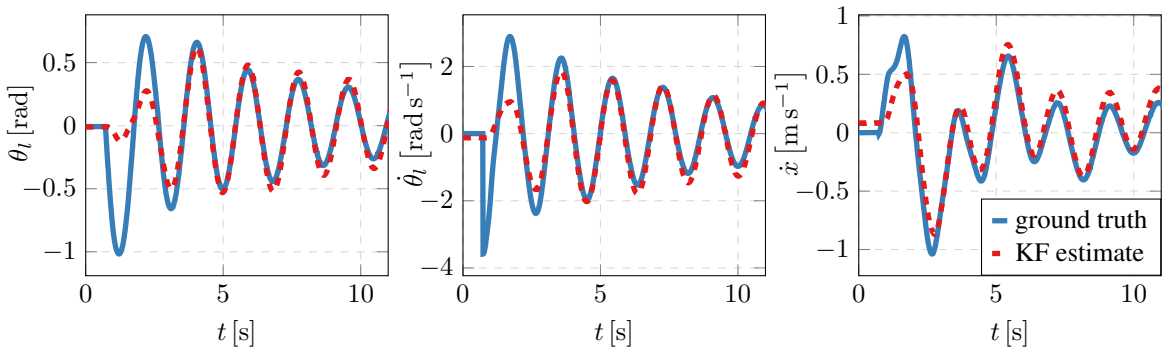


Figure 6.3: Performance of the Kalman filter when an impulse disturbance is applied on the load

The outcomes from the initial experiment, as illustrated in Fig. 6.2, confirm that the Kalman filter's estimate properly tracks the ground truth state, even during significant load amplitudes (up to 0.4 rad $\approx 23°$). The subsequent experiment demonstrates ability of the estimator to track the ground truth even in case of external disturbance acting on the payload. As depicted by Fig. 6.3, after the application of the impulse disturbance, the estimation error dissipates within approximately $1.5\,T_l$. Following experiments will reveal, if the response is fast enough to allow for robust control. In summation, the Kalman filter exhibits satisfactory performance, validating its integration into the control system.

6.3 Trajectory Generator

The trajectory generator has significant impact on tracking performance of the control system since it dictates the reference trajectory for the MPC. Thus, it is necessary to thoroughly validate its performance. To assess its behavior, an experiment was structured in which the trajectory generator encounters a challenging target reference- a short step in

position that doesn't provide the load sufficient time to damp naturally. This experiment was repeated under two distinct scenarios:

- The trajectory generator immediately encounters a step in the target reference when tasked.
- The generator is provided with the forthcoming evolution of the reference, allowing it to exploit the preview capability. Specifically, it is informed that the step will occur after a span of 7.5 s.

The results of these simulations are illustrated in Fig. 6.4 and Fig. 6.5. Notably, both scenarios result in smooth position tracking with a minor overshoot. The benefit of the previewing mechanism is evident, It allows for gentler transition, thus reducing the peak amplitude of the load and the UAV's tilt angles. It can be noticed that in case of absence of preview, the UAV after accelerating counteracts the induced swing by tilting in the opposite direction.

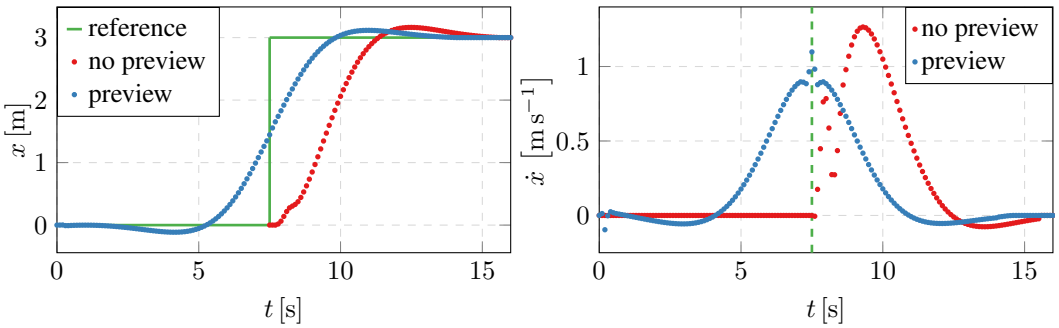


Figure 6.4: Translational variables of generated trajectories

As illustrated in 6.5, the trajectory generator adopts a coherent strategy. The UAV's tilt mirrors the load's angle, acting as if the cable's attachment were rigid, without any joint flexibility. This mirroring ensures minimal tangential forces on the load. Consequently, the angle of the load relative to the UAV's body is nonzero only in two cases:

1. The UAV has to reposition itself.
2. The UAV tries to counteract the swing.

This observed behavior can be consequently segmented into the induction phase and the compensation phase, as portrayed in Fig. 6.5. These phases are determined based on signs of angular rate of the load and the relative angle of the load (in this experiment $\theta - \theta_l$). If they align, the UAV accelerates the angular rate; otherwise, it aims to diminish it. As illustrated in 6.5, the generated trajectory alternates between these phases, striking a balance between position tracking and load damping. The intuitiveness of this behavior provides confidence in the trajectory generator's effectiveness.

6.4 MPC

The validation of the MPC is centered around its capability to accurately follow the generated trajectory without inducing additional oscillations and its robustness with respect to external disturbances. In order to isolate sources of error and test solely performance of the MPC, the MPC utilized ground truth values of unmeasured states in following experiments.

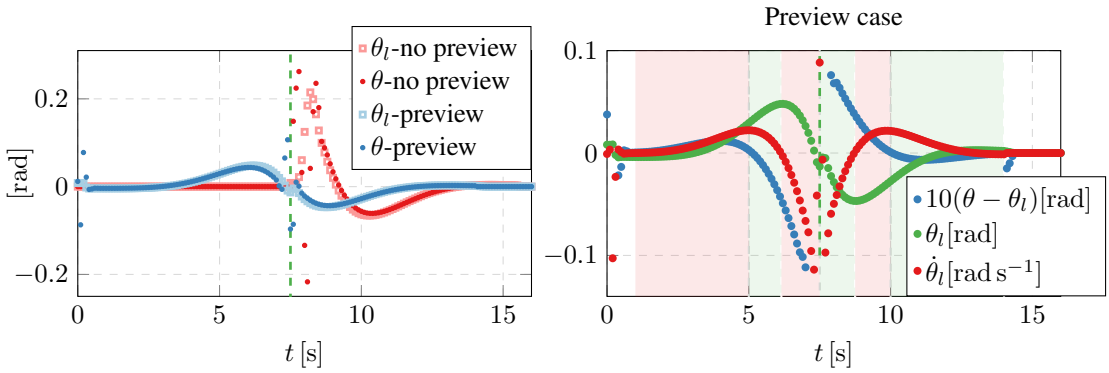


Figure 6.5: Generated angles of the UAV and the load and their interpretation

6.4.1 Trajectory Tracking

For this test, the trajectory generator was supplied a step reference of magnitude 30 m. This generated trajectory subsequently served as the MPC's reference. The target step size was selected to ensure that the generator's velocity constraint ($|\dot{x}| \leq 4\,\mathrm{m\,s^{-1}}$) was activated. Beyond the primary objective of evaluating the MPC's tracking capabilities, this setup also verified the integrity of the generated trajectory, especially concerning its adherence to specified constraints.

The experimental results, showcased in Figure 6.6, proves the MPC's exceptional trajectory tracking abilities. As an outcome, the velocities executed mirror the generated values, eliminating the need for additional adjustments based on the MPC's velocity tracking error. Although the trajectory generator accounted for the velocity constraints, the nature of soft constraints makes the actual maximum velocity slightly exceeded the prescribed limit.

Referring back to Section 5.3, it's evident that the MPC is specifically penalized for deviations in position tracking and the amplitude of the load's attitude. This design choice gives the MPC freedom to determine its own attitude and associated control actions. As such, it finetunes the generated trajectory within a more granular and higher resolution timeframe. Effect of this tow-level optimization becomes obvious when observing the load's attitude in Figure 6.6, which surpasses the generator's predictions. Notably, the UAV's motion strategy ensures that upon reaching its terminal velocity, the load remains stationary relative to the UAV. Additionally, an interesting behavior emerges before the UAV's forward movement, where it induces a counter-swing. This maneuver intuitively provides the UAV with a rapid acceleration capacity and at the same time compensating the swing.

6.4.2 Disturbance Rejection

Though the outcomes of the prior experiment might be considered impressive, a standard feedback controller, given an ideal trajectory and feed-forward control actions, might produce similar results. However, it would likely fail when confronted with unexpected disturbances, like a wind gust. The subsequent experiment examines the MPC's robustness and its capability to counteract such perturbations. The experimental design closely mirrors the one discussed in Section 6.2.2, especially in terms of the introduced disturbance and initial conditions.

The data, as delineated in Fig. 6.7, indicates that the payload's oscillations were almost entirely suppressed within its initial cycle. This was accomplished while maintaining

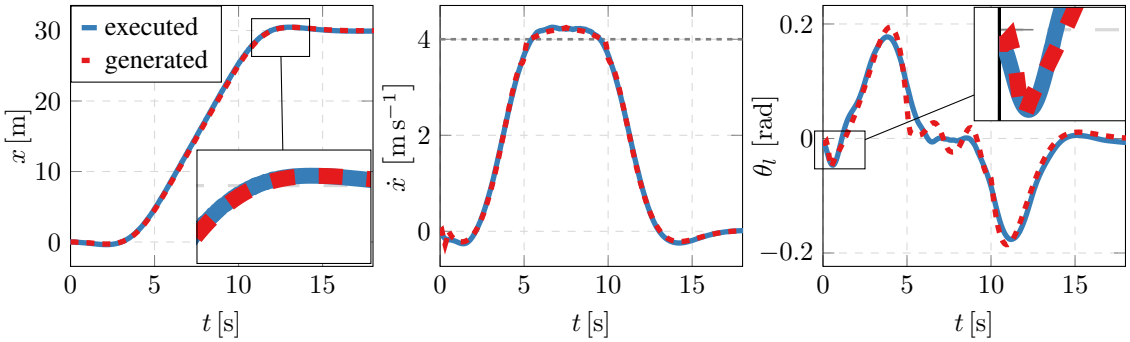


Figure 6.6: Tracking capabilities of the MPC when utilizing ground-truth state

a respectable positional tracking error. The observed control strategy is again coherent. As illustrated in Figure 6.7, the control actions are set to propel the UAV in alignment with the load's trajectory, effectively damping its motion.

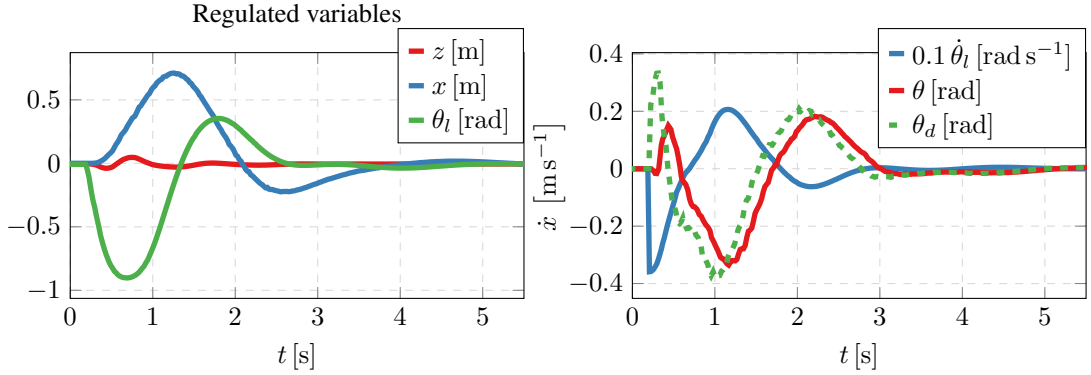


Figure 6.7: Demonstration of MPC's robustness when utilizing ground-truth state

In retrospection, the conducted tests reveal that the implemented MPC accurately follows the desired trajectory when supplied with accurate state information. Furthermore, the MPC consistently outperforms the trajectory generator's predictions concerning the resulting load attitude angles. The controller's ability to effectively counter impulse disturbances applied on the load was also proved. This capability to manage disturbances efficiently within a limited space and to dampen the load's motion in a single cycle underscores the proficiency of the formulated feedback controller. Also an interplay between trajectory generator and MPC in tracking task proves to be effective.

6.5 System Integration

Each component of the control system has been individually tested and validated up to this point. Although each component demonstrated commendable performance on its own, there's no implicit assurance that their collective operation in the integrated control system will yield analogous results. Past observations indicate that integrating a state estimator with a controller often leads to loss of the margins guaranteed by the control methodology. A relevant illustration is the augmentation of the Linear-quadratic Regulator (LQR) with a Kalman filter. As aptly summarized in John Doyle's paper titled "Guaranteed margins for

LQG regulators", the conclusion was simply: "There are none" [39]. Thus, it's crucial to verify that the dynamics and error associated with estimation do not adversely impact the MPC's effectiveness.

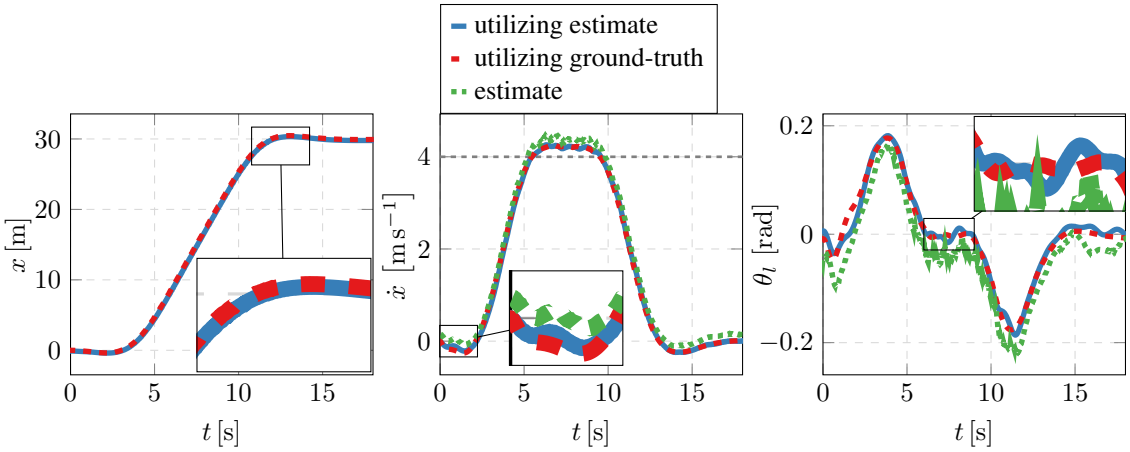


Figure 6.8: MPC's tracking performance when utilizing state estimate

6.5.1 Verification

To ensure the robustness of the integrated system, the same tests as discussed in Section 6.4 were replicated. Simulations, as showcased in Fig. 6.8, confirm that the MPC's tracking performance, when provided with estimates from the Kalman filter, aligns closely with its performance when using ground truth states. This outcome was anticipated given that the trajectory is primarily generated to guide the system's behavior as requested. Notably, even when the MPC operates based on the Kalman filter's estimates, the actual performance concerning the load's attitude surpasses the originally predicted trajectory, much like in the ground truth scenario.

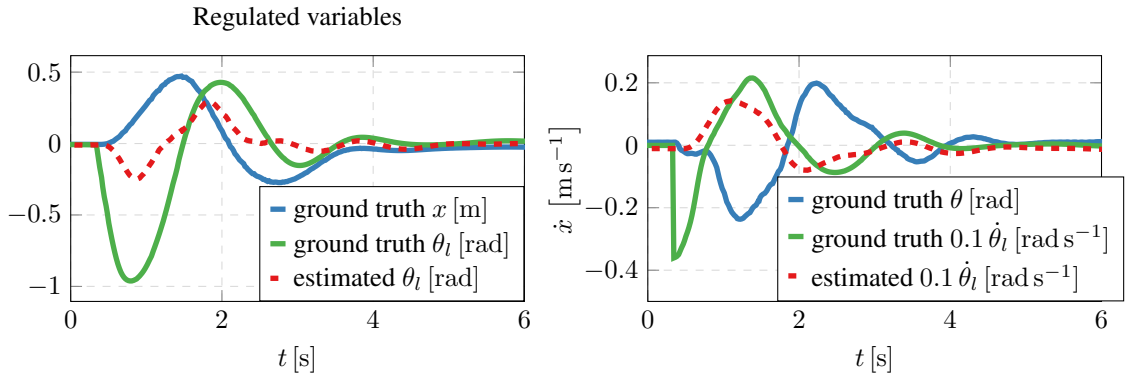


Figure 6.9: Demonstration of MPC's robustness when utilizing state estimate

In this context, one of the most demanding tests for the system is its capability to reject disturbances. As illustrated in the previous experiment (refer to Fig. 6.3), there's a noticeable delay before the Kalman filter's estimate aligns with the ground-truth value. However, the results showcased in Fig. 6.9 confirm the MPC's robustness in managing this estimation error without compromising its operational effectiveness significantly. The pendulum's swing

is effectively damped within two of its natural periods- a commendable outcome given the constraints and available information provided to the control system.

So far, the control system was tasked to track only a step position reference. In order to demonstrate its capabilities to track more complex trajectories one more tracking experiment was conducted. This time the UAV was assigned with following a figure-8 trajectory. Moreover in order to track the requested trajectory more precisely, the elements of $\mathbf{Q}_{TG}$ penalizing the load's amplitude were reduced from 100 to 1. Requested, generated and executed trajectories are depicted by Fig. 6.10. Relevant statistics are included in Table 6.1, however it should be again noted that they reflect performance of this subjectively tuned parametrization.

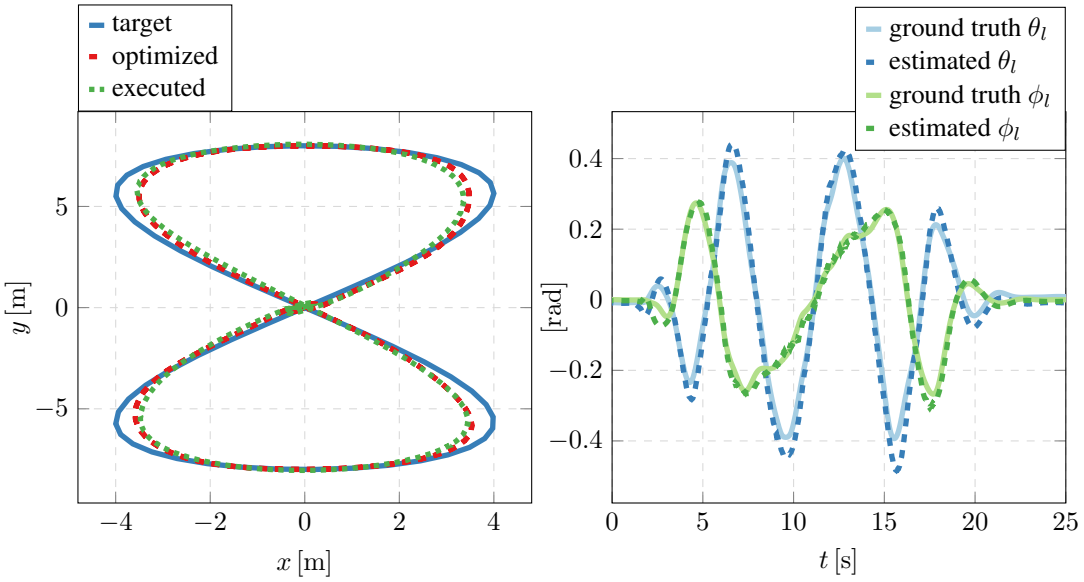


Figure 6.10: Testing capabilities to track complex trajectory

Variable	Mean	Standard Deviation
trajectory generator's position tracking error	0.2957 m	0.2904 m
MPC's position tracking error with respect to target trajectory	0.4267 m	0.3873 m
MPC's position tracking error with respect to optimized trajectory	0.1944 m	0.1527 m
Regulation error of load's amplitudes	0.1827 rad	0.1520 rad
KF's tracking error of load's amplitudes	0.0471 rad	0.0505 rad

Table 6.1: Statistics of the figure-8 tracking experiment. All errors were calculated using Euclidean norm

6.5.2 Comparison to General-Purpose Controller

An additional experiment was carried out to showcase the advantages of the developed controller over a general-purpose regulator. For benchmarking purposes, we employed an extended SE(3) controller in conjunction with an MPC tracker. These components, integral

to the MRS system, have been optimized for agile flight scenarios, as underscored in [3]. While the adoption of MPC controllers is on the rise, this combination remains exemplary in its performance, warranting its designation as state-of-the-art. The chosen benchmark was particularly apt as it mirrors the agility and speed characteristics of our implemented MPC. This ensured a pronounced contrast between the two methods, allowing for a clear evaluation of their respective strengths and weaknesses.

In this experiment, an unfeasible step reference position is set. To further stress-test the versatility of our method in managing varied load configurations, the cable's length was extended to 2.5 m. It's worth noting that neither the MRS tracker nor our trajectory generator was informed about future development of the reference trajectory and therefore couldn't use preview feature.

Simulated results are visually presented in Fig. 6.11. Upon analysis, while the reference solution achieves the target position in a shorter time and with a diminished overshoot compared to the new framework, this efficiency is offset by ensuing oscillations. Conversely, our proposed method effectively damps the payload swing, stabilizing the system at a steady state much faster. An important consideration is that while a marginal overshoot in the UAV's position might seem beneficial initially, in real-world scenarios where collisions with obstacles are strictly forbidden, the more relevant metric becomes the combined overshoot of both the UAV and the payload. Therefore, although the UAV may maintain its position accurately, introducing a considerable amplitude in the load's oscillation can amplify the overall position overshoot.

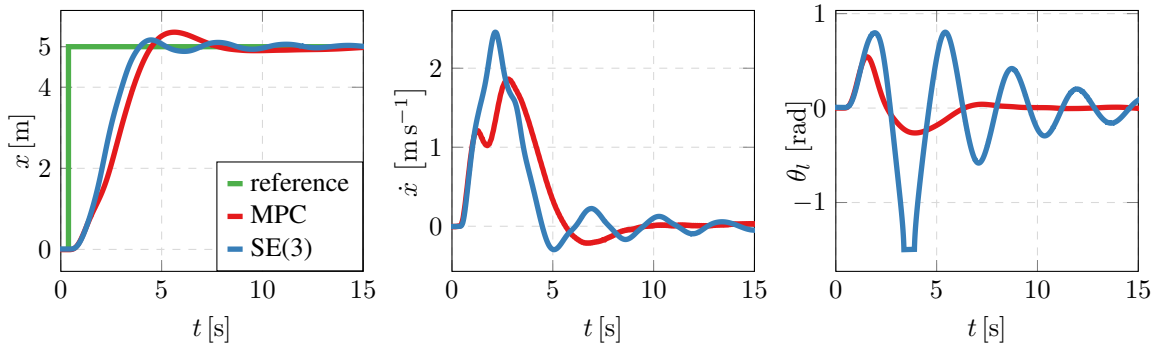


Figure 6.11: Comparison of presented system's and general-purpose state-of-the-art controller's tracking capabilities

The experimental data underscores the efficient synergy between the implemented Kalman filter, MPC, and trajectory generator. The trio effectively addresses real-world challenges, relying solely on UAV position and attitude measurements. The robustness of the closed-loop system is evident, demonstrating its resilience to disturbances. When pitted against a state-of-the-art, generic control framework, our proposed system exhibits superior performance, particularly in scenarios that mirror practical applications.

6.6 Versatility and Performance of the Framework

The experiment discussed in Section 6.5.2 admittedly favors scenarios where the reference control framework might underperform. Nevertheless, the underlying motivation for this experiment is based on practical considerations. The findings highlight a significant challenge

faced by controllers designed without accounting for a UAV transporting a payload suspended by a cable: their reliability and performance predictability can be substantially diminished.

Contrastingly, the performance of our proposed framework isn't easily affected by the mechanical variations of the suspended load. Instead, the control system is parameterized according to these mechanical characteristics. Consequently, if accurate parameters are provided, the system demonstrates consistent behavior. Simulations further confirm that minor deviations from these parameters still permit stable UAV flight. However, a comprehensive analysis concerning the robustness with respect to these parameters remains a subject for future investigations.

This framework's versatility extends beyond just the type of systems it can govern; it also encompasses a broad spectrum of tasks it can efficiently tackle. As long as task-specific criteria can be mapped to the parameters of a constrained QP problem, our framework is likely adept at handling it. The conversion to QP objectives, owing to their structure, remains intuitive. Task requirements are typically addressed by adjusting the elements of the $\mathbf{Q}_{TG}$ matrix, which impacts position tracking errors and load amplitudes. Similarly, if a task demands unique robustness specifications, they can be modulated using elements from the $\mathbf{Q}_{MPC}$ matrix. Achieving this level of adaptability with alternative methodologies can be challenging. Take, for example, the LQR. Although it shares a similar intuitive parameter design based on states, its implementation requires the resolution of a Riccati equation, which can limit its flexibility. In contrast, our framework offers online adjustments to QP parameters. Such dynamic tunability can be harnessed by higher-level software or directly by operators, facilitating real-time behavioral adjustments if deemed necessary.

In essence, this framework showcases versatility, effectively managing a spectrum of systems and tasks. While the specific parameterization evaluated in this chapter offers promising outcomes, it isn't the sole focus of this research. The principal contribution is the establishment of a universal control framework distinguished by its system and task-agnostic characteristics. At a macroscopic level, this controller can be aptly described as a tool finely tuned by both the system it governs and the behavior it's expected to exhibit. This conceptual approach is visually represented in Fig. 6.12.

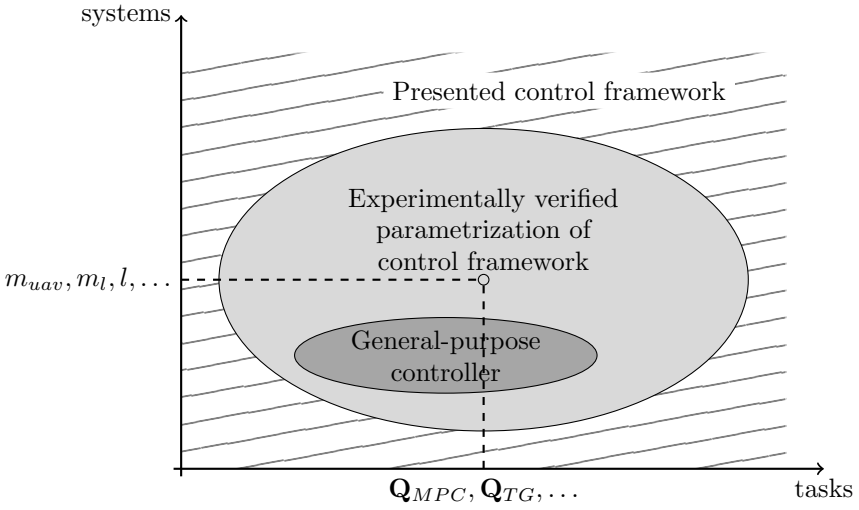


Figure 6.12: High level view on controllers and their abilities to solve different tasks handling different hardware.

6.7 Summary

This chapter provides a detailed examination of the experimental results obtained from a specific configuration of the introduced framework. Each individual component of this framework was found to meet its performance specifications, suggesting that they could be seamlessly integrated or utilized as distinct modules. One key takeaway is the versatility of the trajectory generator. It doesn't only produce a feasible trajectory and associated feed-forward control actions in context of this framework, but these outputs can enhance the performance of any given controller with regard to the dynamics of the suspended load. Furthermore, the feedback MPC enhances this performance by optimizing on a denser and shorter time frame. This two-level optimization means that the final performance exceeds the predictions of the trajectory generator.

The experiments underscored the effectiveness of the LKF in providing state estimates that are both smooth and responsive. The integration of the LKF and MPC manifested as a robust and efficient combination. In scenarios with impulse disturbances impacting the load, the MPC successfully stabilized the UAV, granting the LKF sufficient time to refine its state estimates. Concurrently, the LKF's estimates maintained the accuracy required by the MPC for stabilization. In summary, the evaluated configuration of the control framework demonstrated considerable effectiveness and holds promise for practical applications.

Chapter 7

Conclusion

In this thesis a software, that allows for stable and agile control of an UAV carrying a cable suspended payload, was developed. The dynamical model of the system has been derived and it served as a foundation for other components. The introduced system contains Kalman filter that estimates unmeasured state variables including angle and angular rate of the payload. QP-MPC was then employed to close the control loop. This closed loop was extended by a trajectory generator in form of linearly constrained quadratic program. All these components were successfully implemented in C/C++ as a module of MRS system.

Experiments conducted in simulator proved that the implemented framework allows for smooth and accurate tracking of target position and is robust with respect to external disturbances. Presented and tested parametrization of the system focused on dampening the load and proved effective in that regard, moreover it outperformed general-purpose state of the art control system. The introduced framework was designed such that it allows for online and intuitive adjustment of behaviour thus offering a tool that can be easily tailored to different mechanical setups and different assigned tasks. The assignment of this thesis has therefore been fulfilled successfully. To summarize, following tasks have been accomplished:

- Dynamics of the system were studied and their mathematical model has been derived.
- A Kalman filter has been derived and implemented to estimate unmeasured states variables.
- A model predictive controller has been implemented to ensure short time horizon optimal control.
- A trajectory generator has been implemented to turn infeasible trajectories into desirable ones.
- Experiments have been conducted to verify capabilities of the solution to track target position or trajectory while dampening the load.

Few additional facts however should be noted. Although assigned task assumed utilization of motion capture system, a platform used for simulations motivated the author to assume less reliable and less accurate measurements of position provided by Real Time Kinematics GPS and attitude provided by the low-level flight controller. This made the problem more challenging but at the same time more applicable. The results should be still applicable for the original motion capture system assignment. Although the resulting implementation of the control framework was ready to be tested on real hardware, the hardware platform did not allow it. In order to test these results, MRS system would have to be interfaced to the non-Pixhawk low-level flight controller and motion capture system. Neither MRS system nor hardware platform were completely ready for that and additional obstacles would have to be tackled thus resulting in further not easily estimated amount of work.

7.1 Future Work

The main focus of future work is on experimental verification of the system on real hardware. Apart from that during solving the problem, few ideas emerged. The obvious one relates to constraints on zero heading. Although common load carrying task does not typically require independent control of heading, it makes the system incomplete, less versatile and therefore will be addressed. Next practical improvement would be in extension of Kalman filter, that would allow for estimation of disturbance. Current implementation suffers from persistent wind disturances and offset sensors. Disturbance estimator would solve these practical issues. Another significant improvement could be achieved by employing an offline nonlinear model predictive control for trajectory generation. Not only it would result in accurately generated trajectory even for aggressive maneuvers, it would also allow for nonlinear constraints such as limiting size of the UAV's velocity vector.

Chapter 8

References

[1] T. Baca, R. Penicka, P. Stepan, *et al.*, "Autonomous cooperative wall building by a team of Unmanned Aerial Vehicles in the MBZIRC 2020 competition," en, *Robotics and Autonomous Systems*, vol. 167, p. 104 482, Sep. 2023, ISSN: 0921-8890. DOI: 10.1016/j.robot.2023.104482. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889023001215 (visited on 08/06/2023).

[2] Matternet. "Rigid attachment implemented by matternet." (2022), [Online]. Available: https://mttr.net (visited on 05/26/2023).

[3] T. Baca, M. Petrlik, M. Vrba, *et al.*, "The MRS UAV System: Pushing the Frontiers of Reproducible Research, Real-world Deployment, and Education with Autonomous Unmanned Aerial Vehicles," *Journal of Intelligent & Robotic Systems*, vol. 102, no. 26, pp. 1–28, 1 May 2021.

[4] S. Lee and H. Son, "Antisway Control of a Multirotor With Cable-Suspended Payload," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 6, pp. 2630–2638, Nov. 2021, Conference Name: IEEE Transactions on Control Systems Technology, ISSN: 1558-0865. DOI: 10.1109/TCST.2020.3035004.

[5] M. Patel and P. Ferguson, "Tracking and Estimation of a Swaying Payload Using a LiDAR and an Extended Kalman Filter," in *2021 IEEE International Symposium on Robotic and Sensors Environments (ROSE)*, Oct. 2021, pp. 1–7. DOI: 10.1109/ROSE52750.2021.9611771.

[6] G. Rigatos, "A nonlinear optimal control approach for the UAV and suspended payload system," *Cybernetics and Physics*, vol. 10, pp. 27–39, Jun. 2021. DOI: 10.35470/2226-4116-2021-10-1-27-39.

[7] Y. Stasinchuk, M. Vrba, M. Petrlík, *et al.*, "A Multi-UAV System for Detection and Elimination of Multiple Targets," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, ISSN: 2577-087X, May 2021, pp. 555–561. DOI: 10.1109/ICRA48506.2021.9562057.

[8] N. Urbina-Brito, M.-E. Guerrero-Sánchez, G. Valencia-Palomo, O. Hernández-González, F.-R. López-Estrada, and J. Hoyo, "A Predictive Control Strategy for Aerial Payload Transportation with an Unmanned Aerial Vehicle," *Mathematics*, vol. 9, p. 1822, Aug. 2021. DOI: 10.3390/math9151822.

[9] M. Jiroušek, "Position control of robotic hexacopter," BA thesis, Czech technical university in Prague, 2020.

[10] MRS. "Uav carying a suspended payload." (2020), [Online]. Available: http://mrs.felk.cvut.cz/mbzirc2020 (visited on 05/26/2023).

[11] V. Prkačin, I. Palunko, and I. Petrović, "State and parameter estimation of suspended load using quadrotor onboard sensors," in *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*, ISSN: 2575-7296, Sep. 2020, pp. 958–967. DOI: 10.1109/ICUAS48674.2020.9213840.

[12] R. Verschueren, G. Frison, D. Kouzoupis, *et al.*, *Acados: A modular open-source framework for fast embedded optimal control*, arXiv:1910.13753 [math], Nov. 2020. [Online]. Available: http://arxiv.org/abs/1910.13753 (visited on 07/29/2023).

[13] D. K. D. Villa, A. S. Brandão, and M. Sarcinelli-Filho, "A Survey on Load Transportation Using Multirotor UAVs," en, *Journal of Intelligent & Robotic Systems*, vol. 98, no. 2, pp. 267–296, May 2020, ISSN: 1573-0409. DOI: 10.1007/s10846-019-01088-w. [Online]. Available: https://doi.org/10.1007/s10846-019-01088-w (visited on 08/07/2023).

[14] A. R. Godbole and K. Subbarao, "Nonlinear control of unmanned aerial vehicles with cable suspended payloads," en, *Aerospace Science and Technology*, vol. 93, p. 105 299, Oct. 2019, ISSN: 1270-9638. DOI: 10.1016/j.ast.2019.07.032. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1270963818325185 (visited on 11/14/2022).

[15] S. Sun and C. De Visser, "Quadrotor Safe Flight Envelope Prediction in the High-Speed Regime: A Monte-Carlo Approach," Jan. 2019. DOI: 10.2514/6.2019-0948.

[16] H. Wang, Y. Liu, C. Lin, and Y. Zhou, "Modeling and Simulation of a Slung-load System for the Helicopter," in *2019 Chinese Control And Decision Conference (CCDC)*, ISSN: 1948-9447, Jun. 2019, pp. 1148–1153. DOI: 10.1109/CCDC.2019.8833478.

[17] S. Tang, V. Wüest, and V. Kumar, "Aggressive Flight With Suspended Payloads Using Vision-Based Control," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1152–1159, Apr. 2018, Conference Name: IEEE Robotics and Automation Letters, ISSN: 2377-3766. DOI: 10.1109/LRA.2018.2793305.

[18] S. J. Lee and H. J. Kim, "Autonomous swing-angle estimation for stable slung-load flight of multi-rotor UAVs," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 4576–4581. DOI: 10.1109/ICRA.2017.7989532.

[19] D. Axehill, "Controlling the level of sparsity in MPC," en, *Systems & Control Letters*, vol. 76, pp. 1–7, Feb. 2015, ISSN: 01676911. DOI: 10.1016/j.sysconle.2014.12.002. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0167691114002680 (visited on 07/29/2023).

[20] T. Baca, "Model predictive control of micro aerial vehicle using onboard microcontrolle," M.S. thesis, Czech technical university in Prague, 2015.

[21] F. Gonzalez, A. Heckmann, S. Notter, M. Zurn, J. Trachte, and A. Mcfadyen, "Non-linear model predictive control for UAVs with slung/swung load," en, in *IEEE International Conference on Robotics and Automation*, Seattle,United States, 2015, pp. 1–1. [Online]. Available: https://eprints.qut.edu.au/72665/ (visited on 08/07/2023).

[22] R. P. K. Jain, "Transportation of Cable Suspended Load using Unmanned Aerial Vehicles: A Real-time Model Predictive Control approach," en, 2015. [Online]. Available: https://repository.tudelft.nl/islandora/object/uuid%3A4c6b4a94-4f15-4e67-8c30-eb8156aab406 (visited on 08/07/2023).

[23] K. Klausen, T. I. Fossen, and T. A. Johansen, "Nonlinear Control of a Multirotor UAV with Suspended Load," English, in *2015 International Conference on Unmanned Aircraft Systems (icuas'15)*, ISSN: 2373-6720 WOS:000388438500022, New York: Ieee, 2015, pp. 176–184, ISBN: 978-1-4799-6010-1.

[24] C. d. Crousaz, F. Farshidian, and J. Buchli, "Aggressive Optimal Control for Agile Flight with a Slung Load," 2014. [Online]. Available: https://www.semanticscholar.org/paper/Aggressive-Optimal-Control-for-Agile-Flight-with-a-Crousaz-Farshidian/c4c7210bd9a31e6ee92fbd46252393b428575d1b (visited on 08/07/2023).

[25] F. A. Goodarzi, D. Lee, and T. Lee, *Geometric Stabilization of Quadrotor UAV with a Payload Connected by Flexible Cable*, arXiv:1309.6717 [math], May 2014. [Online]. Available: http://arxiv.org/abs/1309.6717 (visited on 08/07/2023).

[26] J. Hedengren, R. A. Shishavan, K. Powell, and T. Edgar, "Nonlinear Modeling, Estimation and Predictive Control in APMonitor," *Faculty Publications*, Nov. 2014. [Online]. Available: https://scholarsarchive.byu.edu/facpub/1667.

[27] F. Augugliaro and R. D'Andrea, "Admittance control for physical human-quadrocopter interaction," in *2013 European Control Conference (ECC)*, Jul. 2013, pp. 1805–1810. DOI: 10.23919/ECC.2013.6669643.

[28] F. Augugliaro, A. Mirjan, F. Gramazio, M. Kohler, and R. D'Andrea, "Building tensile structures with flying machines," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, ISSN: 2153-0866, Nov. 2013, pp. 3487–3492. DOI: 10.1109/IROS.2013.6696853.

[29] O. Mikuláš, "Quadratic programming algorithms for fast model-based predictive control," BA thesis, Czech technical university in Prague, 2013.

[30] O. S. R. Foundation. "Gazebo simulator." (2012), [Online]. Available: https://gazebosim.org (visited on 05/24/2023).

[31] J. Humpherys, P. Redd, and J. West, "A Fresh Look at the Kalman Filter," *SIAM Review*, vol. 54, no. 4, pp. 801–823, 2012, Publisher: Society for Industrial and Applied Mathematics, ISSN: 0036-1445. [Online]. Available: https://www.jstor.org/stable/24248361 (visited on 07/18/2023).

[32] I. Palunko, R. Fierro, and P. Cruz, "Trajectory generation for swing-free maneuvers of a quadrotor with suspended payload: A dynamic programming approach," in *2012 IEEE International Conference on Robotics and Automation*, ISSN: 1050-4729, May 2012, pp. 2691–2697. DOI: 10.1109/ICRA.2012.6225213.

[33] M. Hehn and R. D'Andrea, "A flying inverted pendulum," in *2011 IEEE International Conference on Robotics and Automation*, ISSN: 1050-4729, May 2011, pp. 763–770. DOI: 10.1109/ICRA.2011.5980244.

[34] J. L. Jerez, E. C. Kerrigan, and G. A. Constantinides, "A condensed and sparse QP formulation for predictive control," in *2011 50th IEEE Conference on Decision and Control and European Control Conference*, ISSN: 0743-1546, Dec. 2011, pp. 5217–5222. DOI: 10.1109/CDC.2011.6160293.

[35] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor UAV on SE(3)," in *49th IEEE Conference on Decision and Control (CDC)*, ISSN: 0191-2216, Dec. 2010, pp. 5420–5425. DOI: 10.1109/CDC.2010.5717652.

[36] M. Bisgaard, A. la Cour-Harbo, and J. Bendtsen, "Input Shaping for Helicopter Slung Load Swing Reduction," Aug. 2008, ISBN: 978-1-60086-999-0. DOI: 10.2514/6.2008-6964.

[37] F. Borrelli and M. Morari, "Offset free model predictive control," in *2007 46th IEEE Conference on Decision and Control*, ISSN: 0191-2216, Dec. 2007, pp. 1245–1250. DOI: 10.1109/CDC.2007.4434770.

[38] A. Bemporad and M. Morari, "Robust model predictive control: A survey," en, in *Robustness in identification and control*, A. Garulli and A. Tesi, Eds., ser. Lecture Notes in Control and Information Sciences, London: Springer, 1999, pp. 207–226, ISBN: 978-1-84628-538-7. DOI: 10.1007/BFb0109870.

[39] J. Doyle, "Guaranteed margins for LQG regulators," *IEEE Transactions on Automatic Control*, vol. 23, no. 4, pp. 756–757, Aug. 1978, Conference Name: IEEE Transactions on Automatic Control, ISSN: 1558-2523. DOI: 10.1109/TAC.1978.1101812.

Chapter A

Appendix A - CD Content