1-31-2013

Agile load transportation systems using aerial robots

Ivana Palunko

Ivana Palunko

Candidate

Electrical and Computer Engineering

Department

This dissertation is approved, and it is acceptable in quality and form for publication:

Approved by the Dissertation Committee:

prof. Rafael Fierro , Chairperson

prof. Chaouki Abdallah

prof. Meeko Oishi

prof. Svetlana Poroseva

Agile Load Transportation Systems Using Aerial Robots

by

Ivana Palunko

B.S., in Electrical Engineering, University of Zagreb, Croatia, 2005
M.S., in Electrical Engineering, University of Zagreb, Croatia, 2007

DISSERTATION

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Doctorate of Philosophy
Engineering

The University of New Mexico

Albuquerque, New Mexico

December, 2012

Dedication

To my parents Ljubica and Nikola.

Acknowledgments

I am pleased that here I would be able to thank the people without whom this dissertation would not have come to the end, in one way or another.

First and foremost, I would like to express my deepest gratitude to my advisor, Prof. Rafael Fierro, for his friendly advice, patience, enthusiasm and encouragement through the past four years. In addition, I would like to thank the members of my Ph.D committee, Prof. Meeko Mitsuko K. Oishi, Prof. Svetlana Poroseva and Prof. Chaouki T. Abdallah, for their time and helpful comments. I am also indebted to Sandra Faust and Prof. Lydia Tapia for their expertise and fruitful discussions that helped me improve Chapter 6.

This list would not be complete without mentioning Prof. Peter Dorato whose love and passion for science was a great inspiration and motivation to continue through the murky plains of research for all these years.

A special thanks goes to fellow students from the MARHES Lab: Domagoj Tolić, Nicola Bezzo, Patricio Cruz and Dean Galarowicz. Thank you for begin there through all the crashes and long hours of coding and tuning in the first steps of autonomous flight. Also, thank you for your help in making things fly smoothly and for having patience to stick with me through numerous repetitions.

Finally, I would like to thank all my friends and family for not giving up on me, although I didn't always have enough time for them in the last four years, and for their love and support through this adventure.

Agile Load Transportation Systems Using Aerial Robots

by

Ivana Palunko

B.S., in Electrical Engineering, University of Zagreb, Croatia, 2005
M.S., in Electrical Engineering, University of Zagreb, Croatia, 2007

Ph.D., Engineering, University of New Mexico, 2012

Abstract

In this dissertation, we address problems that can occur during load transport using aerial robots, i.e., small scale quadrotors. First, detailed models of such transportation system are derived. These models include nonlinear models of a quadrotor, a model of a quadrotor carrying a fixed load and a model of a quadrotor carrying a suspended load. Second, the problem of quadrotor stabilization and trajectory tracking with changes of the center of gravity of the transportation system is addressed. This problem is solved using model reference adaptive control based on output feedback linearization that compensates for dynamical changes in the center of gravity of the quadrotor. The third problem we address is a problem of a swing-free transport of suspended load using quadrotors. Flying with a suspended load can be a very challenging and sometimes hazardous task as the suspended load significantly alters the flight characteristics of the quadrotor. In order to deal with suspended load flight, we present a method based on dynamic programming which is a model based offline method. The second investigated method we use is based on the Nelder-Mead algorithm which is an optimization technique used for nonlinear unconstrained optimization problems. This method is model free and it can be used for offline or online generation of the swing-free trajectories for the suspended load. Besides the swing-free maneuvers with suspended load, load trajectory tracking is another problem we solve in this dissertation. In order to solve this problem we use a Nelder-Mead based algorithm. In addition, we use an online least square policy iteration algorithm. At the end, we propose a high level algorithm for navigation in cluttered environments considering a quadrotor with suspended load. Furthermore, distributed control of multiple quadrotors with suspended load is addressed too. The proposed hierarchical architecture presented in this doctoral dissertation is an important step towards developing the next generation of agile autonomous aerial vehicles. These control algorithms enable quadrotors to display agile maneuvers while reconfiguring in real time whenever a change in the center of gravity occurs. This enables a swing-free load transport or trajectory tracking of the load in urban environments in a decentralized fashion.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

Unmanned aerial vehicles are increasingly being considered as means of performing complex functions or assisting humans in carrying out dangerous missions within dynamic environments. Other possible applications include search and rescue, disaster relief operations, environmental monitoring, wireless surveillance networks, and cooperative manipulation. Creating these types of autonomous vehicles places severe demands on the design of control schemes that can adapt to different scenarios and possible changes of vehicle dynamics.

One specific type of aerial vehicle, the *quadrotor*, has the capability not only of taking off and landing in a very limited area, but also of carrying more weight than other aerial platforms due to its four propellers. Several research groups [1],[2],[3],[4] have developed notable applications and experiments using multiple quadrotor UAVs as part of their robotic platforms. Existing results range from basic hovering [5] and trajectory tracking [6], to formation control, [7] surveillance [8], aggressive maneuvering [9] and aerobatic flips [10].

Once the functionality of the UAVs advances from simple environmental sensing to modification or manipulation of their external environment, a wide and novel set of practical applications and challenges appear in the aerial robotics research field. In fact, for small-size UAVs, a variety of examples of interacting with external objects have been introduced recently. For instance, individual or cooperative transport of suspended load [11, 12, 13], grasping and manipulating [14, 15], applying force to a wall [16] and building structures [17] are examples where aerial robots interact with their surroundings. An interesting approach that emulates mobile manipulation using a UAV equipped with dexterous arms is introduced in [18]. Furthermore, an innovate idea considers the use of a network of quadrotors to deliver medicines in remote villages [19]. In this doctoral dissertation we are concerned with the challenging problem of using quadrotors to transport and manipulate loads safely and efficiently. Aerial manipulation is extremely important in emergency rescue missions, as well as for military and industrial purposes. For example, safe aerial transport of a victim from a dangerous area is vital in an emergency response. In addition, delivering equipment and supplies to inaccessible places is commonly achieved by employing aerial transportation. Another application is landmine detection with a sensor suspended from a cable as depicted on Figure 1.1.

1.2 Problem statement and motivation

In this section we present five problems that we address in this dissertation. These problems are described and defined in next subsections and are all part of a load transportation system using aerial robots.

1.2.1 Unbalanced aerial vehicle

In the first part of this doctoral dissertation we focus on the problems that can alter the center of gravity of the quadrotor. The coordinates of the center of gravity of the quadrotor are given by a vector $\mathbf{r_G} = [x_G \quad y_G \quad z_G]^T$ measured in $\{\mathcal{A}\}$ which represents the distance from the origin of $\{\mathcal{A}\}$ to the quadrotor's center of gravity as shown in Figure 2.1. Usually, small scale quadrotor are modeled as a rigid body and the vector $\mathbf{r_G}$ is set to zero. This means that the vehicle is balanced, i.e., the center of gravity coincides with the origin of the moving aircraft-fixed coordinate system $\{\mathcal{A}\}$. However, there are at least three cases in which this assumption fails and we address them here:

- Change in CoG of the quadrotor (Figure 2.1). The change in the center of gravity can occur if, for example, the battery slips from the quadrotor body frame or one of the sensors, such as cameras, laser or GPS, detaches which unbalances the quadrotor.

- Change in CoG of the load attached to the quadrotor by a rigid link (Figure 2.2). By a rigid link we mean any type of gripper firmly attached to the body of a quadrotor. The change in the CoG of the load can occur if the load is not gripped at the point of CoG but somewhere on the side. This can happen if the gripper missed the exact gripping point of the load slipped due to flight conditions.

- Change in the position of suspension point of the suspended load (Figure 2.3). The change in the position of the suspension point of the load $\boldsymbol{\rho}_H$ can occur when, for example, one or more of the suspension cables snaps in the case of multiple-point suspension load.

We state the change in the center of gravity formally in the following definition.

Definition 1 (Change in the center of gravity)**.** *Given a system in the form*

$$\dot{\mathbf{q}} = \mathbf{f}(\mathbf{q}, \mathbf{r_G}) + \mathbf{g}(\mathbf{q}, \mathbf{u}, \mathbf{r_G}), \tag{1.1}$$

where $q \in \mathbb{R}^n$ is the system state, $\boldsymbol{f}(.) \in \mathbb{R}^n$ is the vector of system dynamics, $\boldsymbol{g}(.) \in \mathbb{R}^n$ is the input vector, $u \in \mathbb{R}^m$, n is the number of the states, m is the number of inputs and $\mathbf{r_G} \in \mathbb{R}^3$ is the vector representing the distance from the origin of the reference frame to the center of gravity of the rigid body. We say the system given by (1.1) is unbalanced if

$$\mathbf{r_G} = 0 \rightarrow Re(eig(\frac{\partial \mathbf{f}(\mathbf{q}, \mathbf{r_G})}{\partial \mathbf{q}}|_{\mathbf{q^0}})) \leq 0, \forall t \in \mathbb{R} \tag{1.2}$$

and

$$\mathbf{r_G} \neq 0 \rightarrow Re(eig(\frac{\partial \mathbf{f}(\mathbf{q}, \mathbf{r_G})}{\partial \mathbf{q}}|_{\mathbf{q^0}})) > 0, \forall t \in \mathbb{R} \tag{1.3}$$

where $Re(.)$ denotes the real part of the complex number, $eig(.)$ denotes the set of eigenvalues of the Jacobian matrix $(\frac{\partial \mathbf{f}(\mathbf{q}, \mathbf{r_G})}{\partial \mathbf{q}}|_{\mathbf{q^0}})$ of (1.1) in equilibrium point $\mathbf{q^0}$.

Since in all three cases the change in $\mathbf{r_G}$, $\boldsymbol{\rho}_L$ and $\boldsymbol{\rho}_H$ produces additional forces and torques acting on the quadrotor we can observe these three cases and treat them as a change in a single vector $\mathbf{r_G}$. In Chapter 4 we analyze the stability of the system with respect to the change in $\mathbf{r_G}$ and we propose a method based on adaptive control to solve this problem.

1.2.2 Swing-free manipulation with the suspended load

The suspended load exerts forces and torques on the quadrotor given by (2.15). Since $\mathbf{F}_H$ and $\mathbf{T}_H$ are the function of the load displacement angles $\boldsymbol{\eta}_L = [\phi_L \quad \theta_L]$ shown in Figure 2.3, in Chapter 5 of this dissertation we present methods based on dynamic programming and Nelder-Mead algorithm which will ensure swing-free maneuvers of the suspended load. Next definition states the problem.

Definition 2. *Given a small positive constant $\epsilon > 0$ and the time required to accomplish the desired trajectory $t_F > 0$, we say the quadrotor carrying a suspended load performs a swing-free maneuver when $\|\phi_L(t)\| \leq \epsilon$ and $\|\theta_L(t)\| \leq \epsilon$ for any $t \geq t_F$.*

Minimizing the angles $\boldsymbol{\eta}_L$ we are minimizing the force $\mathbf{F_H}$ and the torque $\mathbf{T_H}$ which represent an external disturbance force and torque for the quadrotor baseline attitude controller. In order to solve this problem we are comparing three techniques in order to determine the input trajectory for the quadrotor that will ensure a swing-free motion of the load.

1.2.3 Trajectory tracking with the suspended load

Except the swing-free maneuvers with suspended load, tracking a trajectory is another problem we solve in Chapter 6. which is defined as

Definition 3. *Given a reference trajectory $\mathcal{P}_{ref}(t) = [x_{Lref}(t) \quad y_{Lref}(t) \quad z_{Lref}(t)]^T$, the position of the suspended load $P_L(t) = [x_L(t) \quad y_L(t) \quad z_L(t)]^T$ and a small positive constant $\epsilon_L > 0$, we say that the suspended load is performing trajectory tracking if $|e_L(t)| \leq \epsilon_L$ where $e_L(t) = [x_{Lref}(t) - x_L(t) \quad y_{Lref}(t) - y_L(t) \quad z_{Lref}(t) - z_L(t)]^T$.*

We are interested in solving the problem in which we define a trajectory which we want to accomplish with the suspended load. In order to do this we have to move the quadrotor in a certain way because the quadrotor is acting as the actuator. The problem is that we do not know how a trajectory for the quadrotor should look like in order for the load to achieve tracking. In order to control the position of the load we need to generate a trajectory for the quadrotor that will ensure $e_L(t) \to 0$. We are able to accomplish this because the position vector of the suspended load is a differentially flat output of this system system. Considering this we propose two different techniques in order to solve this problem. First we propose a method based on Nelder-Mead algorithm [20] which we employ in order to learn the parameters of the unknown input trajectory for the quadrotor. The unknown trajectory is approximated with finite Taylor series. The second method is based on reinforcement learning algorithm, least-square policy iteration (LSPI) in which the additional control input for the quadrotor is drawn from LSPI algorithm in order to drive the quadrotor in the right direction. Both of these techniques are model-free.

1.2.4 Load transportation in cluttered environments

In Section 7.1 we are address the problem of transportation of suspended load in cluttered environments.

Let $\mathcal{Q}$ represent the origin of the moving coordinate frame $\{\mathcal{A}\}$ attached to a quadrotor body with suspended load in a Euclidean space $\mathbb{R}^3$, and let convex sets $\mathcal{B}_1, ..., \mathcal{B}_n$ be fixed rigid objects distributed in $\mathcal{W}$ as depicted in Figure 1.2.4. The $\mathcal{B}_i$'s are defined as obstacles. We assume that both the geometry of $\mathcal{Q}, \mathcal{B}_1, ..., \mathcal{B}_n$ and the locations of the $\mathcal{B}_i$'s are known a priori and accurately in a 3-dimensional area-of-interest $\mathcal{W}$. We assume that given an initial position and a goal position of $\mathcal{Q}$ in $\mathcal{W}$, a collision-free path $\mathcal{P}$ is generated using a high-level planner based on any of the well known path planning techniques such as potential field, cell decomposition, etc. [21].

The problem of interest is stated as follows:

A path $\mathcal{P}$ is approximated with a finite number of waypoints w_i such that the straight-line path $\mathcal{P}_{segment}$ between two waypoints belongs to $\mathcal{W}_{free}$. Based on the given waypoints w_i a mid-level planner calculates the minimal distance between the path $\mathcal{P}_{segment}$ and the surrounding obstacles $\boldsymbol{\delta}_{min_{obst-dist}}$. Subsequently, it determines the maximum allowable load displacement angle $\boldsymbol{\eta}_{max_{allow-angle}}$ such that the trajectory of the load $\mathcal{T}_{load}$ belongs to $\mathcal{W}_{free}$. By determining $\boldsymbol{\eta}_{max_{allow-angle}} \neq 0$ we can incorporate as a part of the objective function used for generating a swing-free trajectory $\mathcal{T}_{opt}$ for $\mathcal{Q}$ using dynamic programming algorithm presented in this article. The optimal trajectory is generated in such a way that both, the performed trajectory of the quadrotor $\mathcal{T}_{quadrotor}$ and the performed trajectory of the load $\mathcal{T}_{load}$ belong to $\mathcal{W}_{free}$. Using $\boldsymbol{\eta}_{max_{allow-angle}}$ we can incorporate a safety region for the maximal displacement of the load by narrowing the $\boldsymbol{\delta}_{min_{obst-dist}}$ to $\boldsymbol{\delta}_{safe}$. This approach makes the overall system more robust and reliable.

1.2.5 Load transportation with multiple quadrotors

Because we are dealing with small scale aerial vehicles with limited payload, we need to employ r quadrotors in order to transport loads of larger weight as depicted in Figure 1.2.5. If we define a load with k suspension points $p_1 \dots p_k$ and if each of the suspension points p_j has a linear and angular velocity V_j, then the total velocity V of the center of gravity of the load is defined as $V = \sum_{j=0}^{k} V_j$. Since each of the suspension points p_j is connected to a quadrotor q_i, $i = 1 \dots r$ we must ensure that each quadrotor performs the same motion in order to have . In this doctoral dissertation we consider an application of using binary consensus proposed in [22] for synchronization multiple quadrotors carrying suspended loads and performing swing-free trajectory tracking or load trajectory tracking. By utilizing the binary consensus protocol we are able to achieve distributed tracking for a network of quadrotors.

1.3 Overview of the methods used

1.3.1 Feedback linearization

Through the years feedback linearization is an approach common in nonlinear control design and it has attracted lots of research. The main idea is to algebraically transform nonlinear systems into (fully or partly) linear ones, so that linear control techniques can be applied. This differs from conventional (Jacobian) linearization, because feedback linearization does not approximate the system but the linearization is achieved by the exact state transformation and feedback. The main results about feedback linearization can be found in [23], [24], [25]. The method also has a number of important limitations. It cannot be used for all nonlinear systems, no robustness is guaranteed in the presence of parameter uncertainty or unmodeled dynamics. The applicability of input-state linearization is not always guaranteed, while input-output feedback linearization cannot be applied when the relative degree is not defined. The second problem is due to the difficulty of finding convergent observers for nonlinear systems and, when an observer can be found, the lack of a general separation principle is a problem. The third problem is due to the fact that the exact model of the nonlinear system is not available in performing feedback linearization especially when the linearizing transformation is poorly conditioned.

1.3.2 Model reference adaptive control

Model-reference adaptive control (MRAC) system is composed of four parts: a plant containing unknown parameters, a reference model for compactly specifying the desired output of the control system, a feedback control law containing adjustable parameters, and an adaptation mechanism for updating the adjustable parameters [26]. The plant is assumed to have a known structure, although the parameters are unknown. A reference model is used to specify the ideal response of the adaptive control system to the external command. The main prerequisite for the controller is to have perfect tracking capacity in order to allow the possibility of tracking convergence. That is, when the plant parameters are exactly known, the corresponding controller parameters should make the plant output identical to that of the reference model. When the plant parameters are not known, the adaptation mechanism will adjust the controller parameters so that perfect tracking is asymptotically achieved. If the control law is linear in terms of the adjustable parameters, it is said to be linearly parameterized. Existing adaptive control designs normally require linear parametrization of the parameter in order to guarantee stability and tracking convergence. The adaptation mechanism is used to adjust the parameters in the control law. In MRAC systems, the adaptation law searches for parameters such that the response of the plant under adaptive control becomes the same as that of the reference model, i.e., the objective of the adaptation is to make the tracking error converge to zero and to keep the parameter error bounded. The main issue in adaptation design is to synthesize an adaptation mechanism which will guarantee that the control system remains stable and the tracking error converges to zero as the parameters vary [25].

1.3.3 Dynamic programming

One of the main objectives in optimal control theory is to minimize a certain cost of what is considered an undesirable outcome of a certain situation. The main characteristic of that type of situations is that decisions have to be balanced between the desire for low current cost, with the undesirable high cost in the future. The dynamic programming (DP) approach is one of the techniques that captures this trade off, as it is described in more details in [27]. Decisions are ranked at each step, based on the sum of the present cost and the expected future cost, assuming optimal decision making for subsequent steps. There is a very broad variety of practical problems that can be solved using dynamic programming. In this dissertation, we focus on a broadly applicable problem of optimal control of a dynamic system over a finite horizon (a finite number of steps). This basic problem has two main features

- a dynamic system represented in a discrete form,

  $$x_{k+1} = f_k(x_k, u_k), \quad k = 0, 1, \dots, N-1,$$

  where k represents the discrete time step, x_k is the systems's state which captures the past information that is relevant for future optimization, u_k is the control or decision variable to be selected at time k, N is the horizon or a finite number of steps the control is applied, and f_k is a function that describes the transition of the system from irs current state to the next one.

- a cost function that is additive over time

  $$J = g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k),$$

  where J is the total cost, $g_N(x_N)$ is a terminal cost incurred at the end of the process and $g_k(x_k, u_k)$ is the cost incurred at time-step k.

The problem is formulated as optimization of the total cost J over controls $u_0 \dots u_{N-1}$. The dynamic programming algorithm is based on the principle of optimality. The name is due to Bellman [28], who contributed a great deal to the popularization of DP and to its transformation into a systematic tool.

Definition 4 (Principle of Optimality)**.** *Let $\pi^* = \left\{\mu_0^*, \mu_1^* \dots, \mu_{N-1}^*\right\}$ be an optimal policy for the basic problem, and assume that when using π^*, a given state x_i occurs at time i with positive probability. If we consider the subproblem whereby we are at x_i at time i and wish to minimize* cost-to-go *from time i to time N.*

$$g_N(x_N) + \sum_{k=i}^{N-1} g_k(x_k, u_k),$$

Then the truncated policy $\left\{\mu_i^*, \mu_{i+1}^* \dots, \mu_{N-1}^*\right\}$ is optimal for this subproblem.

The principle of optimality states that an optimal policy can be constructed in a piecewise fashion, first constructing an optimal policy for the terminal subproblem involving the last step, then extending the optimal policy to the terminal subproblem involving the last two steps, and continuing in this manner until an optimal policy for the entire problem is constructed.

1.3.4 Nelder-Mead algorithm

The Nelder-Mead algorithm or simplex search algorithm [20], is one of the best known algorithms for multidimensional unconstrained optimization without derivatives. The basic algorithm is quite simple to understand and very easy to use. For these reasons, it is very popular in many fields of science and technology, especially in chemistry and medicine.

This method does not require any derivative information, which makes it suitable for problems with non-smooth functions. It is widely used to solve parameter estimation and similar statistical problems, where the function values are uncertain or subject to noise. It can also be used for problems with discontinuous functions, which occur frequently in statistics and experimental mathematics.

The Nelder-Mead algorithm is designed to solve the classical unconstrained optimization problem of minimizing a given nonlinear function $f : \mathbb{R}^n \to \mathbb{R}$. The method uses only function values at some points in $\mathbb{R}^n$, and does not try to form an approximate gradient at any of these points. Hence it belongs to the general class of direct search methods [29], [30]. A large subclass of direct search methods, including the Nelder-Mead method, maintain at each step a non degenerate simplex, a geometric figure in n dimensions of nonzero volume that is the convex hull of $n + 1$ vertices. Each iteration of a simplex-based direct search method begins with a simplex, specified by its $n+1$ vertices and the associated function values. One or more test points are computed, along with their function values, and the iteration terminates with bounded level sets.

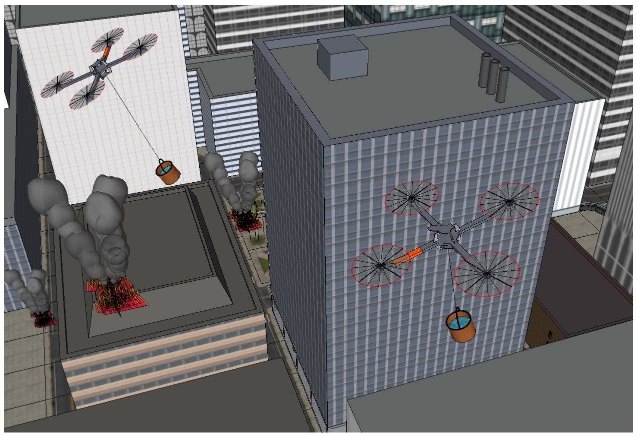
1.3.5 Least square policy iteration

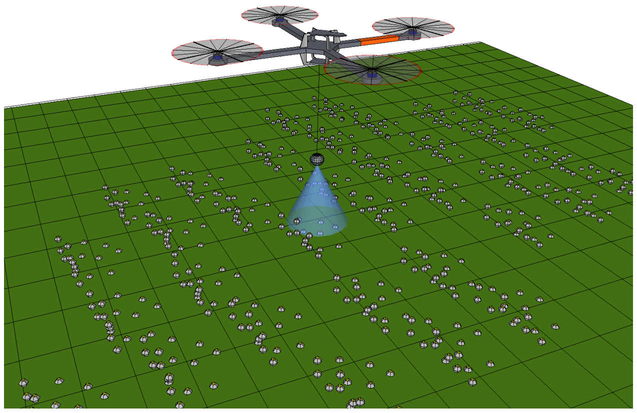
Least-Squares Policy Iteration (LSPI) is a reinforcement learning algorithm designed to solve control problems. It uses value function approximation to cope with large state spaces and batch processing for efficient use of training data. LSPI has been used successfully to solve several large scale problems using relatively few training data. Policy iteration evaluates policies by estimating their value functions, and then uses these value functions to find new, improved policies. The classical policy iteration employs tabular, exact representations of the value functions and policies. However, most problems of practical interest have state and action spaces with a very large or even infinite number of elements, which precludes tabular representations and exact policy iteration. Instead, approximate policy iteration is used [31]. Constructing approximate value functions for the policies considered is the central, most challenging component of approximate policy iteration. While representing the policy can also be challenging, an explicit representation is often avoided, by computing policy actions on-demand from the approximate value function.

1.3.6 Binary consensus

A consensus protocol is a group decision making process that seeks the consent, not necessarily the agreement, of participants and the resolution of objections. One important application of consensus protocols is to provide synchronization of the processes of interest. Synchronization is timekeeping which requires the coordination of events to operate a system in unison. The familiar conductor of an orchestra enables consensus by keeping the orchestra in time. Systems operating with all their parts in synchrony are said to be synchronous or *in sync*.

In this doctoral dissertation we consider an application of using binary consensus proposed in [22] for synchronization multiple quadrotors carrying suspended loads and performing swing-free trajectory tracking or load trajectory tracking. By utilizing the binary consensus protocol we are able to achieve distributed tracking for a network of quadrotors. Among other things the advantages of distributed control protocols are that it avoids a single point of failure, it is more fault tolerant, scalable and robust, and allows for cheaper and simpler agents when comparing with centralized control protocols [32].


(a) Urban Firefighting


(b) Demining

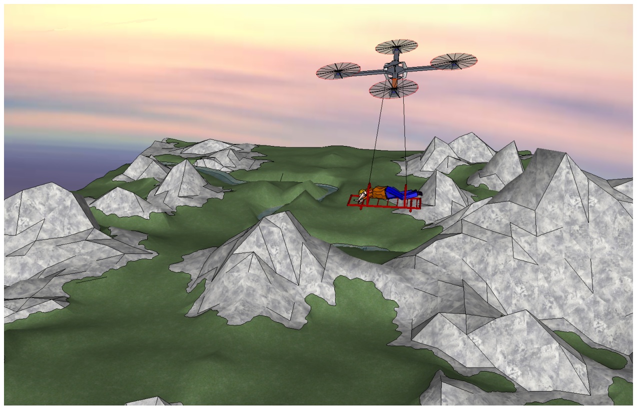

(c) Search and Rescue

Figure 1.1: Motivation for load transportation using aerial robots.

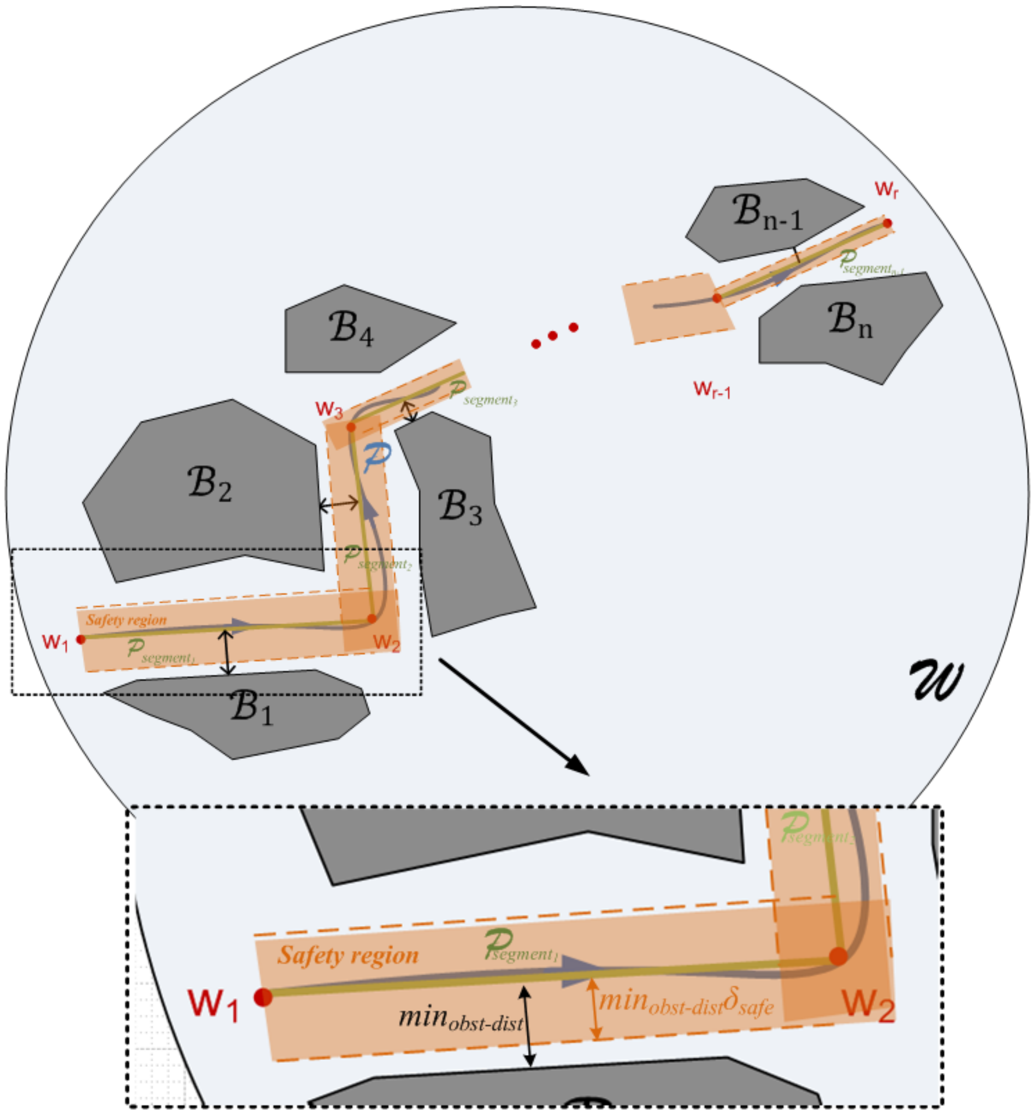


Figure 1.2: Representation of Euclidean space with obstacles and predefined way-points.

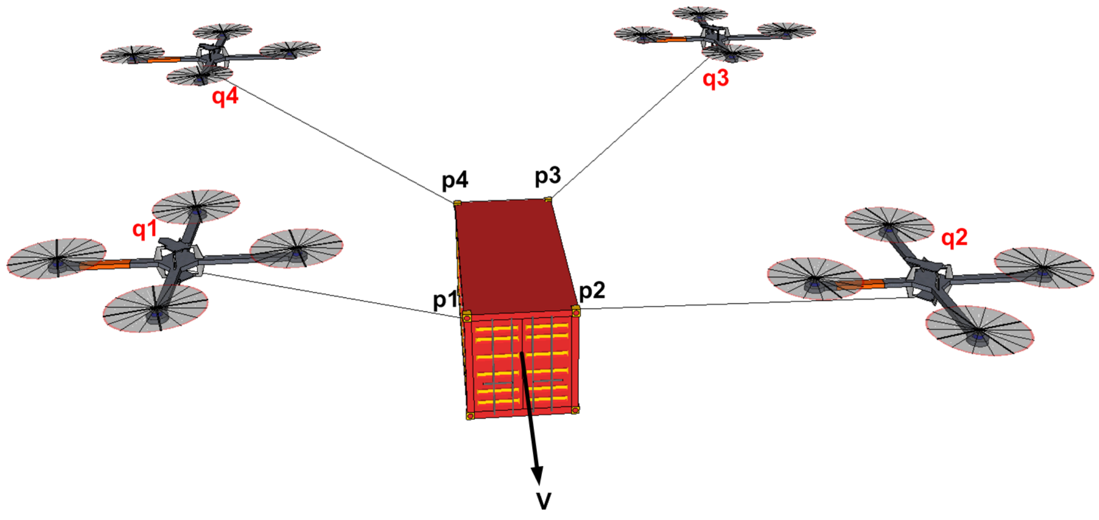


Figure 1.3: Load transport using multiple quadrotors.

Chapter 2

Modeling of the Quadrotor - Load System

Quadrotors belong to a class of rotorcraft unmanned aerial vehicles. Control design for such systems is in general based on their mathematical model, which features high nonlinearities and strong couplings between its different subsystems. The effects of coupling in the case of a small-scale helicopter were investigated in [33]. Modeling is thus, a crucial stage in designing flight controllers. Furthermore, it is important to have high-fidelity models for simulation purposes. The development of an accurate model that includes the flexibility and flapping of rotors, fuselage aerodynamics, downwash effect is very complex [34].

For the purpose of the controller design, in general the quadrotor is modeled as a rigid-body evolving in Cartesian space with a force and torque generating mechanism. The rigid body dynamics can be described with the Newton-Euler equations of motion, or with the energy-oriented approaches such as the Lagrange formulation. Most published papers on modeling and control, especially for small platforms such as quadrotors, use this simple model for control design [35], [36], [37], [38]. For more accurate models, rigid-body model is essentially extended or augmented with simplified rotor dynamics and aerodynamics, using a combination of momentum and blade element theory [39]. Indeed, detailed modeling of rotor dynamics and aerodynamics can be extremely complex because the rotor itself is a multi-body system. Furthermore, the produced aerodynamic forces and torques depend on operating conditions and vehicle motion. Examples of rotor modeling can be found in [34] for full-scale helicopters. In [40] we have extended the model of a small scale helicopter. Most popular techniques used in modeling are first-principles, system-identification, or combination of both approaches.

In this chapter we present mathematical models of the load transportation systems using quadrotors. We start by deriving a nonlinear model of a quadrotor from first principles. The model is presented in both Newton-Euler and Lagrange form. Then we present two distinct quadrotor load systems, a model of a quadrotor carrying a fixed load and a model of a quadrotor carrying a suspended load.

2.1 Nonlinear model of a quadrotor

We start by modeling the quadrotor as a rigid body. For the analysis of the motion of rigid body through 6 DOF we define two coordinate frames as indicated in Figure 2.1. The moving coordinate frame $\{\mathcal{A}\}$ is fixed to the quadrotor and is called the aircraft-fixed reference frame. The origin of the aircraft-fixed frame is chosen to coincide with the Center of Gravity (CoG) when CoG is in the principal plane of symmetry and the vehicle is considered to be balanced. Ground-fixed reference frame $\{\mathcal{G}\}$ is considered to be the inertial frame. The position and orientation of the vehicle are described relative to the inertial reference frame $\{\mathcal{G}\}$ while the linear and angular velocities of the vehicle are expressed in the aircraft-fixed coordinate frame $\{\mathcal{A}\}$. The Z_A-axis in aircraft-fixed coordinate frame $\{\mathcal{A}\}$ is pointing downwards which follows the con-

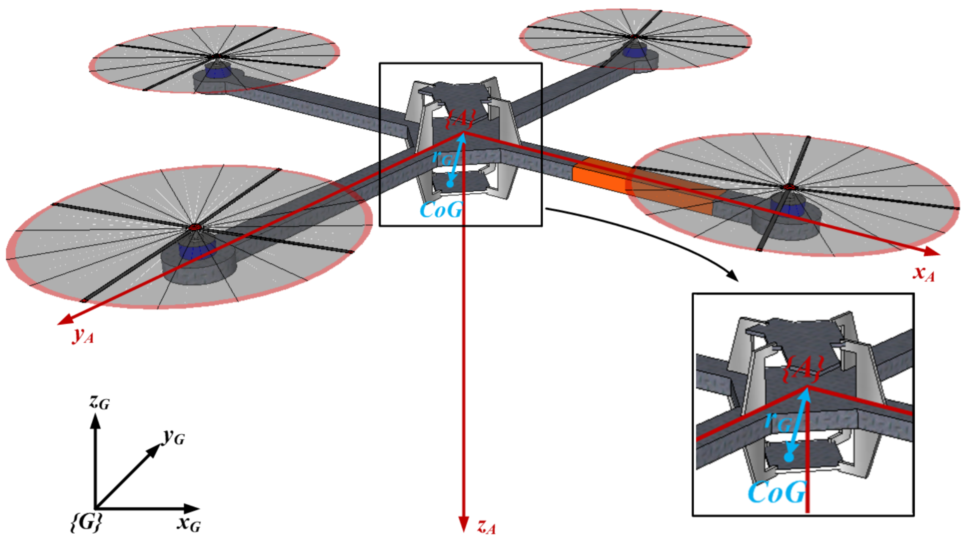


Figure 2.1: The coordinate frames used for deriving the model of the quadrotor.

vention in aerospace design. The following variables are used to describe quadrotor kinematics and dynamics,

$\boldsymbol{\eta_1} = [x \quad y \quad z]^T$ - position of the origin of $\{\mathcal{A}\}$ measured in $\{\mathcal{G}\}$,

$\boldsymbol{\eta}_2 = [\phi \quad \theta \quad \psi]^T$ - angles of roll (ϕ), pitch (θ) and yaw (ψ) that parametrize locally the orientation of $\{\mathcal{A}\}$ with respect to $\{\mathcal{G}\}$,

$\boldsymbol{\nu}_1 = [u \quad v \quad w]^T$ - linear velocity of the origin of $\{\mathcal{A}\}$ relative to $\{\mathcal{G}\}$ expressed in $\{\mathcal{A}\}$ (i.e., body-fixed linear velocity),

$\boldsymbol{\nu}_2 = [p \quad q \quad r]^T$ - angular velocity of $\{\mathcal{A}\}$ relative to $\{\mathcal{G}\}$ expressed in $\{\mathcal{A}\}$ (i.e., body-fixed angular velocity),

$\mathbf{r}_G = [x_G \quad y_G \quad z_G]^T$ - distance from the origin of $\{\mathcal{A}\}$ to the quadrotor's center of gravity.

The transformation matrix between two reference frames is obtained by matrix multiplication of the three basic orthogonal rotation matrices that belong to the special orthogonal group $SO(3, \mathbb{R})$, [41]. The aircraft-fixed linear velocity vector $\boldsymbol{\nu}_1$ and the position rate vector $\dot{\boldsymbol{\eta}}_1$ are related through a transformation matrix ${}^G_A\mathbf{R}\left(\boldsymbol{\eta}_2\right)$ according to

$$\dot{\boldsymbol{\eta}}_1 = \frac{d\boldsymbol{\eta}_1}{dt} = {}^G_A\mathbf{R}\left(\boldsymbol{\eta}_2\right)\boldsymbol{\nu}_1. \tag{2.1}$$

The aircraft-fixed angular velocity vector $\boldsymbol{\nu}_2$ and the Euler rate vector $\dot{\boldsymbol{\eta}}_2$ are related through a transformation matrix $\mathbf{Q}\left(\boldsymbol{\eta}_2\right)$ according to:

$$\dot{\boldsymbol{\eta}}_2 = \mathbf{Q}\left(\boldsymbol{\eta}_2\right)\boldsymbol{\nu}_2, \quad \mathbf{Q}\left(\boldsymbol{\eta}_2\right) = \begin{bmatrix} 1 & s\phi t\theta & c\phi t\theta \\ 0 & c\phi & -s\phi \\ 0 & \frac{s\phi}{c\theta} & \frac{c\phi}{c\theta} \end{bmatrix}, \tag{2.2}$$

with $\mathbf{Q}\left(\boldsymbol{\eta}_2\right)$ being singular for $\theta = \pm\frac{\pi}{2}$. This singularity does not represent a problem in our design because our aircraft will not execute aggressive maneuvers as to achieve the pitch of 90°. Otherwise, this problem can be circumvented by many different methods, (e.g., quaternion representation).

The condensed representation of systems kinematics is

$$\begin{bmatrix} \dot{\boldsymbol{\eta}}_1 \\ \dot{\boldsymbol{\eta}}_2 \end{bmatrix} = \begin{bmatrix} {}^G_A\mathbf{R}\left(\boldsymbol{\eta}_2\right) & 0 \\ 0 & \mathbf{Q}\left(\boldsymbol{\eta}_2\right) \end{bmatrix} \begin{bmatrix} \boldsymbol{\nu}_1 \\ \boldsymbol{\nu}_2 \end{bmatrix}, \quad \dot{\boldsymbol{\eta}} = \mathbf{J}_R\left(\boldsymbol{\eta}\right)\boldsymbol{\nu}. \tag{2.3}$$

2.1.1 Quadrotor dynamics

In this subsection Euler's first and second axioms are used to derive the rigid body equations of motion. Consider the aircraft-fixed coordinate system frame $\{\mathcal{A}\}$ rotating with angular velocity $\boldsymbol{\omega} = [\omega_1\ \omega_2\ \omega_3]$ about the ground-fixed coordinate system frame $\{\mathcal{G}\}$. The quadrotor's inertia tensor $\mathbf{I}_A$ is defined as:

$$\mathbf{I}_A = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix}; \mathbf{I}_A = \mathbf{I}_A^T > 0;$$

where I_x, I_y and I_z are the moments of inertia about the X_A, Y_A and Z_A axes. Since the principal axes of $\{\mathcal{A}\}$ are aligned with quadrotor axes we can write $I_{xy} = I_{yx} =$

$I_{xz} = I_{zx} = I_{zy} = I_{yz} = 0$. Now, we are ready to write down the Newton Euler's equations. For the sake of simplicity from now on we will refer to the matrix ${}^{G}_{A}\mathbf{R}(\boldsymbol{\eta}_2)$ as $\mathbf{R}(\boldsymbol{\eta}_2)$. The equation for translation in the $\{\mathcal{A}\}$ frame is given by

$$\sum {}^{G}\mathbf{F}_E = m\frac{d}{dt}\left({}^{G}\boldsymbol{\nu}_1\right) = m\frac{d}{dt}\left(\mathbf{R}(\boldsymbol{\eta}_2)\boldsymbol{\nu}_1\right)$$

where m is the mass of the quadrotor and $\mathbf{F}_E$ is the sum of external forces. If we continue

$$\begin{aligned}\sum {}^{G}\mathbf{F}_E &= m\frac{d\mathbf{R}(\boldsymbol{\eta}_2)}{dt}\boldsymbol{\nu}_1 + m\mathbf{R}(\boldsymbol{\eta}_2)\frac{d\boldsymbol{\nu}_1}{dt} \\ &= m\mathbf{R}(\boldsymbol{\eta}_2)\mathbf{S}(\boldsymbol{\nu}_2)\boldsymbol{\nu}_1 + m\mathbf{R}(\boldsymbol{\eta}_2)\frac{d\boldsymbol{\nu}_1}{dt}\end{aligned} \tag{2.4}$$

Linear acceleration vector can be expressed as

$$\frac{d\boldsymbol{\nu}_1}{dt} = \boldsymbol{\nu}_2 \times \boldsymbol{\nu}_1 + \dot{\boldsymbol{\nu}}_1 + \dot{\boldsymbol{\nu}}_2 \times \mathbf{r}_G + \boldsymbol{\nu}_2 \times (\boldsymbol{\nu}_2 \times \mathbf{r}_G).$$

If we express (2.4) in the $\{\mathcal{A}\}$ frame we get

$$\begin{aligned}\sum \mathbf{R}^{-1}(\boldsymbol{\eta}_2)\,{}^{G}\mathbf{F}_E &= m\mathbf{R}^{-1}(\boldsymbol{\eta}_2)\mathbf{R}(\boldsymbol{\eta}_2)\mathbf{S}(\boldsymbol{\nu}_2)\boldsymbol{\nu}_1 + m\mathbf{R}^{-1}(\boldsymbol{\eta}_2)\mathbf{R}(\boldsymbol{\eta}_2)\frac{d\boldsymbol{\nu}_1}{dt} \\ \sum \mathbf{F}_E &= m\left[\boldsymbol{\nu}_2 \times \boldsymbol{\nu}_1 + \dot{\boldsymbol{\nu}}_1 + \dot{\boldsymbol{\nu}}_2 \times \mathbf{r}_G + \boldsymbol{\nu}_2 \times (\boldsymbol{\nu}_2 \times \mathbf{r}_G)\right],\end{aligned} \tag{2.5}$$

where $\mathbf{F}_E = \begin{bmatrix} F_x & F_y & F_z \end{bmatrix}^T$ is a sum of external forces measured in $\{\mathcal{A}\}$. The equation for rotation in the $\{\mathcal{G}\}$ frame is given by

$$\sum {}^{G}\mathbf{T}_E = \frac{d}{dt}\left({}^{G}\mathbf{L}\right) = \frac{d}{dt}\left(\mathbf{R}(\boldsymbol{\eta}_2)\mathbf{I}_A, \boldsymbol{\nu}_2\right) \tag{2.6}$$

where $\mathbf{I}_A$ is the aircrafts inertia tensor and ${}^{G}\mathbf{L}$ is an angular momentum measured in $\{\mathcal{G}\}$. By expanding (2.6) we get

$$\sum {}^{G}\mathbf{T}_E = \mathbf{R}(\boldsymbol{\eta}_2)\mathbf{S}(\boldsymbol{\nu}_2)\mathbf{I}_A\boldsymbol{\nu}_2 + \mathbf{R}(\boldsymbol{\eta}_2)\mathbf{I}_A\frac{d\boldsymbol{\nu}_2}{dt}. \tag{2.7}$$

Expressing (2.7) in body axis we obtain

$$\begin{aligned}\sum \mathbf{R}^{-1}(\boldsymbol{\eta}_2)\,{}^{G}\mathbf{T}_E &= \mathbf{R}^{-1}(\boldsymbol{\eta}_2)\mathbf{R}(\boldsymbol{\eta}_2)\mathbf{S}(\boldsymbol{\nu}_2)\mathbf{I}_A\boldsymbol{\nu}_2 + \mathbf{R}^{-1}(\boldsymbol{\eta}_2)\mathbf{R}(\boldsymbol{\eta}_2)\mathbf{I}_A\frac{d}{dt}\boldsymbol{\nu}_2 \\ \sum \mathbf{T}_E &= \mathbf{I}_A\dot{\boldsymbol{\nu}}_2 + \boldsymbol{\nu}_2 \times \mathbf{I}_A\boldsymbol{\nu}_2 + m\mathbf{r}_G \times (\dot{\boldsymbol{\nu}}_1 + \boldsymbol{\nu}_2 \times \boldsymbol{\nu}_1),\end{aligned} \tag{2.8}$$

where $\mathbf{T}_E = \begin{bmatrix} T_\phi & T_\theta & T_\psi \end{bmatrix}^T$ is the sum of external torques measured in $\{\mathcal{A}\}$. At the end equations (2.5) and (2.8) represent the dynamics of the system and (2.3) represents the kinematics of the system. In expanded form we can write them as

$$\text{Dynamics:} \begin{cases} F_x = m\left[\dot{u} - vr + wq - x_G\left(q^2 + r^2\right) + y_G\left(pq - \dot{r}\right) + z_G\left(pr + \dot{q}\right)\right] \\ F_y = m\left[\dot{v} - wp + ur + x_G\left(qp + \dot{r}\right) - y_G\left(p^2 + r^2\right) + z_G\left(qr - \dot{p}\right)\right] \\ F_z = m\left[\dot{w} - uq + vp + x_G\left(rp - \dot{q}\right) + y_G\left(rq - \dot{p}\right) - z_G\left(q^2 + p^2\right)\right] \\ \\ T_\phi = I_{xx}\dot{p} + \left(I_z - I_y\right)qr + m\left[y_G\left(\dot{w} - uq + vp\right) - z_G\left(\dot{v} - wp + ur\right)\right] \\ T_\theta = I_y\dot{q} + \left(I_{xx} - I_z\right)rp + m\left[z_G\left(\dot{u} - vr + wq\right) - x_G\left(\dot{w} - uq + vp\right)\right] \\ T_\psi = I_z\dot{r} + \left(I_y - I_{xx}\right)pq + m\left[x_G\left(\dot{v} - wp + ur\right) - y_G\left(\dot{u} - vr + wq\right)\right] \end{cases} \tag{2.9}$$

$$\text{Kinematics:} \begin{cases} \frac{d}{dt}\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathbf{R}\left(\phi, \theta, \psi\right)\begin{bmatrix} u \\ v \\ w \end{bmatrix} \\ \\ \frac{d}{dt}\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \mathbf{Q}\left(\phi, \theta, \psi\right)\begin{bmatrix} p \\ q \\ r \end{bmatrix} \end{cases} \tag{2.10}$$

2.1.2 Lagrange representation of the quadrotor model

Quadrotor's 6 DOF nonlinear dynamic equations of motion can be expressed in a compact form as:

$$\mathbf{M}\dot{\boldsymbol{\nu}} + \mathbf{C}\left(\boldsymbol{\nu}\right)\boldsymbol{\nu} + \mathbf{D}\boldsymbol{\nu} + \mathbf{G}\left(\boldsymbol{\eta}\right) = \boldsymbol{\tau} + \boldsymbol{\tau}_{\mathbf{L}}, \tag{2.11}$$

where $\boldsymbol{\eta} = [\boldsymbol{\eta}_1 \quad \boldsymbol{\eta}_2]^T$ is the vector of position and orientation, $\boldsymbol{\nu} = [\boldsymbol{\nu}_1 \quad \boldsymbol{\nu}_2]^T$ is vector of linear and angular velocities and $\mathbf{M}$ is the mass and inertia matrix of the quadrotor given by

$$\mathbf{M} = \begin{bmatrix} m & 0 & 0 & 0 & mz_G & -my_G \\ 0 & m & 0 & -mz_G & 0 & mx_G \\ 0 & 0 & m & my_G & -mx_G & 0 \\ 0 & -mz_G & my_G & I_x & 0 & 0 \\ mz_G & 0 & -mx_G & 0 & I_y & 0 \\ -my_G & mx_G & 0 & 0 & 0 & I_z \end{bmatrix}, \quad \mathbf{M} = \mathbf{M}^T > 0.$$

Matrix $\mathbf{C}(\boldsymbol{\nu})$ consist of Coriolis and centripetal terms. Using results from [42], we achieve a parametrization such that $\mathbf{C}(\boldsymbol{\nu})$ is skew-symmetric

$$\mathbf{C}(\boldsymbol{\nu}) = \begin{bmatrix} 0 & 0 & 0 & m(y_Gq+z_Gr) & -m(x_Gq-w) & -m(x_Gr+v) \\ 0 & 0 & 0 & -m(y_Gp+w) & m(z_Gr+x_Gp) & -m(y_Gr-u) \\ 0 & 0 & 0 & -m(z_Gp-v) & -m(z_Gq+u) & m(x_Gp+y_Gq) \\ -m(y_Gq+z_Gr) & m(y_Gp+w) & m(z_Gp-v) & 0 & I_zr & -I_{yy}q \\ m(x_Gq-w) & -m(z_Gr+x_Gp) & m(z_Gq+u) & -I_zr & 0 & I_{xx}p \\ m(x_Gr+v) & m(y_Gr-u) & -m(x_Gp+y_Gq) & I_yq & -I_xp & 0 \end{bmatrix}$$

Decomposing the vectors of external forces acting on a rigid body we define four distinct vectors $\mathbf{D}\boldsymbol{\nu}$, $\mathbf{G}(\boldsymbol{\eta})$, $\boldsymbol{\tau}$ and $\boldsymbol{\tau}_L$. Dissipative force and torque vector is given by $\mathbf{D}\boldsymbol{\nu}$, where $\mathbf{D}$ is the damping matrix

$$\mathbf{D} = diag\Big(\; c_{\mu x}, \;\; c_{\mu y}, \;\; c_{\mu z}, \;\; c_{\mu\phi}, \;\; c_{\mu\theta}, \;\; c_{\mu\psi} \; \Big), \quad \mathbf{D} = \mathbf{D}^T > 0, \quad \dot{\mathbf{D}} = 0,$$

and c_{μ_x}, c_{μ_y}, c_{μ_z}, $c_{\mu\phi}$, $c_{\mu\theta}$, $c_{\mu\psi}$ are damping coefficients. With $\mathbf{G}(\boldsymbol{\eta})$ we denote the vector of gravitational forces and torques

$$\mathbf{f}_G(\boldsymbol{\eta}_2) = {}^G_A\mathbf{R}^{-1}(\phi, \theta) \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix}, \quad \mathbf{G}(\boldsymbol{\eta}_2) = -\begin{bmatrix} \mathbf{f}_G(\boldsymbol{\eta}_2) \\ \mathbf{r}_G \times \mathbf{f}_G(\boldsymbol{\eta}_2) \end{bmatrix},$$

where g represents the gravitational constant. Control inputs are given as vector $\boldsymbol{\tau}$

$$\mathbf{f}_\tau(\boldsymbol{\eta}_2) = {}^{\mathbf{G}}_{\mathbf{A}}\mathbf{R}^{-1}(\boldsymbol{\eta}_2) \begin{bmatrix} 0 \\ 0 \\ U_1 \end{bmatrix}, \quad \boldsymbol{\tau}(\boldsymbol{\eta}_2, \mathbf{U}) = \begin{bmatrix} \mathbf{f}_\tau(\boldsymbol{\eta}_2) \\ U_2 \\ U_3 \\ U_4 \end{bmatrix},$$

where U_1, U_2, U_3, U_4 represent control forces generated by four rotors [5]. $\boldsymbol{\tau}_L = [\mathbf{F_H} \quad \mathbf{T_H}]$ represents the vector of forces (2.15) and torques (2.16) that the load exerts on the quadrotor.

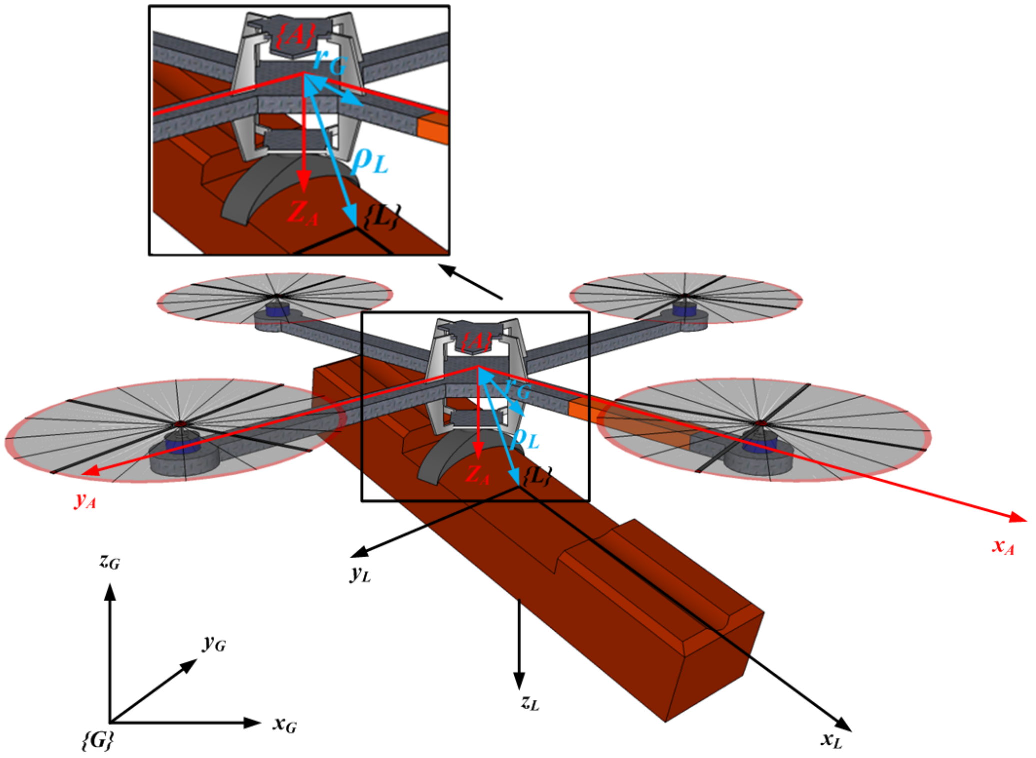


Figure 2.2: Quadrotor carrying a fixed load.

2.2 Quadrotor with the fixed load

In this section we introduce the model of a quadrotor carrying a load fixed to the frame shown in Figure 2.2. Writing the model in Lagrangian form we get

$$\mathbf{M_{FL}}\dot{\boldsymbol{\nu}} + \mathbf{C_{FL}}\left(\boldsymbol{\nu}\right)\boldsymbol{\nu} + \mathbf{D_{FL}}\boldsymbol{\nu} + \mathbf{G_{FL}}\left(\boldsymbol{\eta}\right) = \boldsymbol{\tau}_{\mathbf{FL}}.$$

The mass and inertia matrix of the quadrotor with the fixed load is given by

$$\mathbf{M}_{FL} = \begin{bmatrix} m{+}m_L & 0 & 0 & 0 & (m{+}m_L)z_G & -(m{+}m_L)y_G \\ 0 & m{+}m_L & 0 & -(m{+}m_L)z_G & 0 & (m{+}m_L)x_G \\ 0 & 0 & m{+}m_L & (m{+}m_L)y_G & -(m{+}m_L)x_G & 0 \\ 0 & -(m{+}m_L)z_G & (m{+}m_L)y_G & I_x{+}I_{xL} & -I_{xyL} & -I_{xzL} \\ (m{+}m_L)z_G & 0 & -(m{+}m_L)x_G & -I_{yxL} & I_y{+}I_{yL} & -I_{yzL} \\ -(m{+}m_L)y_G & (m{+}m_L)x_G & 0 & -I_{zxL} & -I_{zyL} & I_z{+}I_{zL} \end{bmatrix},$$

where m_L is the added mass of the load, and I_{xL}, I_{yL} and I_{zL} are the added moments of inertia from the load about the X_L, Y_L and Z_L axes. Since the principal axes of

$\{\mathcal{L}\}$ do not have to be necessarily aligned with the load axes, $I_{xyL} \neq 0$, $I_{yxL} \neq 0$, $I_{xzL} \neq 0$, $I_{zxL} \neq 0$, $I_{zyL} \neq 0$ and $I_{yzL} \neq 0$. The Coriolis and Centripetal matrix is defined as

$$\mathbf{C}_{FL}(\boldsymbol{\nu}) = \left[\begin{matrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ -(m+m_L)(y_Gq+z_Gr) & (m+m_L)(y_Gp+w) & (m+m_L)(z_Gp-v) \\ (m+m_L)(x_Gq-w) & -(m+m_L)(z_Gr+x_Gp) & (m+m_L)(z_Gq+u) \\ (m+m_L)(x_Gr+v) & (m+m_L)(y_Gr-u) & -(m+m_L)(x_Gp+y_Gq) \end{matrix}\right.$$

$$\left.\begin{matrix} (m+m_L)(y_Gq+z_Gr) & -(m+m_L)(x_Gq-w) & -(m+m_L)(x_Gr+v) \\ -(m+m_L)(y_Gp+w) & (m+m_L)(z_Gr+x_Gp) & -(m+m_L)(y_Gr-u) \\ -(m+m_L)(z_Gp-v) & -(m+m_L)(z_Gq+u) & (m+m_L)(x_Gp+y_Gq) \\ 0 & -I_{yxL}q-I_{xzL}p+(I_z+I_{zL})r & I_{yzL}r+I_{xyL}p-(I_y+I_{yL})q \\ -I_{yzL}q+I_{xzL}p-(I_z+I_{zL})r & 0 & -I_{xzL}r-I_{xyL}q+(I_x+I_{xL})p \\ -I_{yzL}r-I_{xyL}p+(I_y+I_{yL})q & I_{xzL}r+I_{xyL}q-(I_x+I_{xL})p & 0 \end{matrix}\right].$$

The damping matrix in this case is non-diagonal and satisfies properties $\mathbf{D_{FL}} = \mathbf{D_{FL}}^T > 0, \quad \dot{\mathbf{D}}_{\mathbf{FL}} = 0$. The vector of gravitational forces and torques

$$\mathbf{f}G(\boldsymbol{\eta}_2) = {}^G_A\mathbf{R}^{-1}(\phi,\theta) \begin{bmatrix} 0 \\ 0 \\ -(m+m_L)g \end{bmatrix}, \quad \mathbf{G}_{FL}(\boldsymbol{\eta}_2) = -\begin{bmatrix} \mathbf{f}_G(\boldsymbol{\eta}_2) \\ \mathbf{r}_G \times \mathbf{f}_G(\boldsymbol{\eta}_2) \end{bmatrix},$$

where g represents the gravitational constant. Control inputs $\boldsymbol{\tau}_{FL}$ are equivalent to the control input vector $\boldsymbol{\tau}$ from (2.11). Vector $\boldsymbol{\tau}_L$ from (2.11) is zero. This is the case because the fixed load - quadrotor system we model as a single rigid body. Therefore, the vector which consist of external forces and torques that load exerts on the quadrotor are contained inside of system matrices, i.e. $\mathbf{M_{FL}}$, $\mathbf{C_{FL}}(\boldsymbol{\nu})$, $\mathbf{D_{FL}}$ and $\mathbf{G_{FL}}(\boldsymbol{\eta})$. While modeling the quadrotor carrying the suspended load, we model it as two separate rigid bodies connected through a cable. These two rigid bodies interact with each other through linear acceleration vector $\dot{\boldsymbol{\nu}}_1$, forces $\mathbf{F_H}$ and torques $\mathbf{T_H}$ which is presented in the next section.

2.3 Slung load model

Very thorough models of external suspended load systems can be found in [43] where specific simulation models for different types of single- and multiple-point suspen-

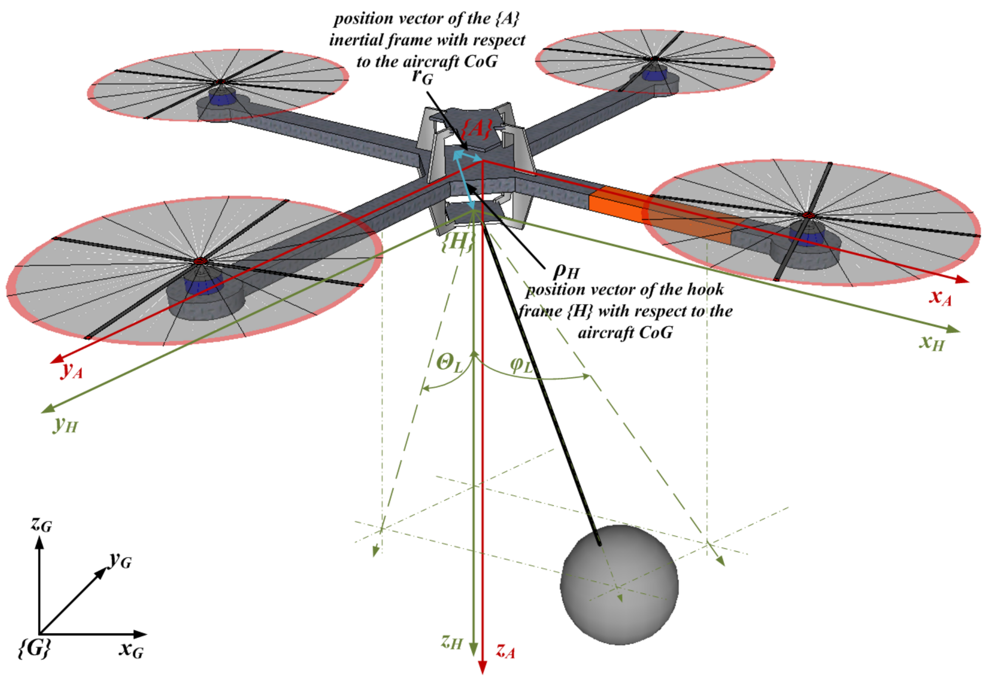

position vector of the {A} inertial frame with respect to the aircraft CoG
r_G
{A}
{H}
ρ_H
position vector of the hook frame {H} with respect to the aircraft CoG
x_A
x_H
y_A
y_H
Θ_L
φ_L
z_G
y_G
{G}
x_G
z_H
z_A

Figure 2.3: Quadrotor carrying a suspended load.

sion, as well as multi-lift variations of the system, are developed. In more recent publications [44], [45] and [46] the single point suspension type models found in [43] have been implemented and simulated. Considering the models presented in the literature, we present the model of the single point suspended load, in this subsection. The external slung load is modeled as a point mass spherical pendulum suspended from a single point. The coordinate systems we use are shown in Figure 2.3. The unit vectors of the $\{\mathcal{H}\}$ coordinate system always remain parallel to those of the aircraft-fixed coordinate system $\{A\}$. The motion of the load is described in polar coordinates using two angles ϕ_L and θ_L, where ϕ_L and θ_L are measured from the z_H axis in direction of x_H and y_H respectively. Therefore, the position vector $\boldsymbol{\rho}_L$ of the load with respect to the suspension point is given by

$$\boldsymbol{\rho}_L = \mathbf{R_{y_H}}(\theta_L)\,\mathbf{R_{x_H}}(\phi_L)\begin{bmatrix}0\\0\\l_L\end{bmatrix} = l_L\begin{bmatrix}\cos(\phi_L)\sin(\theta_L)\\-\sin(\phi_L)\\\cos(\theta_L)\cos(\phi_L)\end{bmatrix}$$

where $\mathbf{R_{y_H}}(\theta_L)$ and $\mathbf{R_{x_H}}(\phi_L)$ are the rotational matrices, and l_L is the length of the cable. The position vector $\boldsymbol{\rho}_H$ of $\{\mathcal{H}\}$ with respect to the quadrotor CoG is given by $\boldsymbol{\rho}_H = [x_H \quad y_H \quad z_H]^T$. The absolute velocity $\boldsymbol{\nu}_L$ of the load is given by

$$\boldsymbol{\nu}_L = \boldsymbol{\nu}_1 + \dot{\boldsymbol{\rho}} + \boldsymbol{\nu}_2 \times \boldsymbol{\rho} \tag{2.12}$$

where $\boldsymbol{\nu}_1$ is the linear velocity of the quadrotor, $\boldsymbol{\rho} = \boldsymbol{\rho}_L + \boldsymbol{\rho}_H$ is the position vector of the load with respect to the CoG of the quadrotor, and $\boldsymbol{\nu}_2$ is the angular velocity of the quadrotor. The absolute acceleration $\dot{\boldsymbol{\nu}}_L$ of the load is

$$\dot{\boldsymbol{\nu}}_L = \dot{\boldsymbol{\nu}}_1 + \ddot{\boldsymbol{\rho}} + \dot{\boldsymbol{\nu}}_2 \times \boldsymbol{\rho} + 2\boldsymbol{\nu}_2 \times \dot{\boldsymbol{\rho}} + \boldsymbol{\nu}_2 \times (\boldsymbol{\nu}_2 \times \boldsymbol{\rho}) \tag{2.13}$$

where $\dot{\boldsymbol{\nu}}_1$ is the linear acceleration of the quadrotor. The vector given by $\mathbf{g_{GL}}(\boldsymbol{\eta})$ represents the vector of gravitational forces and moments

$$\mathbf{G}_L = \mathbf{R_x}(\phi)^{-1}\,\mathbf{R_y}(\theta)^{-1}\begin{bmatrix}0\\0\\m_L g\end{bmatrix} = m_L g\begin{bmatrix}\sin(\theta)\\\cos(\theta)\sin(\phi)\\\cos(\theta)\cos(\phi)\end{bmatrix}$$

where ϕ and θ are respectively the roll and pitch angles of the quadrotor, m_L is the mass of the load and g is the gravitational constant. By enforcing the torque equilibrium about the suspension point, we get

$$\mathbf{f}_{\tau L}(\phi_L, \theta_L, \boldsymbol{\nu}, \boldsymbol{\eta}) = -\boldsymbol{\rho}_L \times (-m_L\dot{\boldsymbol{\nu}}_L + \mathbf{G}_L), \quad \mathbf{f}_{\tau L} = 0, \tag{2.14}$$

where $\boldsymbol{\nu}$ and $\boldsymbol{\eta}$ are vectors of quadrotor states. In expanded form, (2.14) is a system of three second order equations in Cartesian coordinates in $\{\mathcal{H}\}$ frame. By solving these three equations for $\ddot{\phi}_L$ and $\ddot{\theta}_L$, we obtain the equations of motion for the given system in polar coordinates. The symbolic computation for obtaining these equations is performed using *Mathematica*, and because of the length of the equations, is omitted. The suspended load introduces additional terms denoted by $\boldsymbol{\tau}_L$ in the equations of motion of the quadrotor. The force $\mathbf{F_H}$ that the load exerts on the quadrotor is given by

$$\mathbf{F_H} = -m_L \mathbf{f_{GL}} \tag{2.15}$$

and the torque $\mathbf{T_H}$ is given by

$$\mathbf{T_H} = \boldsymbol{\rho}_H \times \mathbf{F_H}. \tag{2.16}$$

Both $\mathbf{F_H}$ and $\mathbf{T_H}$ are functions of ϕ_L and θ_L, as well as of the quadrotor states $\boldsymbol{\nu}$ and $\boldsymbol{\eta}$.

2.4 Conclusion

In this chapter we present three distinct mathematical models which describe, model of a single quadrotor derived in Section 2.1, model of the quadrotor with a fixed load derived in Section 2.2 and the model of a quadrotor with a suspended load in Section 2.3. With these three models we covered modeling of load transportation systems for fixed and suspended payloads. Since this is a novel area of research there is still room form improvement. One can continue to extend these models by investigating the influence of the downwash from the rotors to the load. Furthermore, how the suspension system influences the stability and maneuverability of the whole system.

Chapter 3

Baseline attitude controller for quadrotor

In this chapter we present different baseline attitude controllers. Baseline attitude controller represents a low level controller in charge of quadrotor's orientation and position.

The problem of controller design for quadrotors attracts the attention of many researchers from control and robotics community, because it presents an interesting control challenge and an excellent testbed for developing and testing new control algorithms. Existing controllers can be classified into three main categories, linear, nonlinear, and learning-based control methods.

Conventional approaches to flight control and most initial attempts to achieve autonomous helicopter and quadrotor flight have been based on linear controllers such as PID, LQR [3] and H_{inf}. Design and implementation of these types of controllers is straightforward. Furthermore, there are many available tools for tuning and performance and robustness analysis. Despite the limitations, such as performance degradation when the system leaves the nominal working point and difficulty in proving the asymptotic stability of the complete closed-loop system, these controllers are the most widely accepted methods for control design.

In order to overcome some of the limitations of linear control design, a variety of nonlinear flight controllers have been developed and applied to rotorcraft control. Nonlinear controllers are in general based on the nonlinear model of the system. Among these, feedback linearization [40], dynamic inversion [47], backstepping [48], adaptive control [38] and model predictive control [49] have received much of the attention and have been successfully applied to both quadrotor control.

The main characteristic of the learning-based control methods is that the system model is not used, but several trials and flight data are required in order to train the system. Among the used methods, fuzzy logic [50], human-based learning [51], [52], and neural networks [53] are the most popular.

3.1 Lead-Lag cascade baseline controller

In this section we present the design of the output feedback controller using cascade lead-lag controllers. For the design, we are using the model presented in Section 2.1, which is linearized around an equilibrium point and can be represented in state space form as,

$$\dot{\mathbf{q}} = \mathbf{A}\,\mathbf{q} + \mathbf{B}\,\mathbf{u}, \quad \mathbf{y} = \mathbf{F}\,\mathbf{q}, \tag{3.1}$$

where $\mathbf{u} = [U_1\ U_2\ U_3\ U_4]^T$ is the input vector, $\mathbf{y} = [x \quad y \quad z \quad \phi \quad \theta \quad \psi]^T$ is the output vector, and $\mathbf{q} = [\boldsymbol{\eta} \quad \boldsymbol{\nu}]^T, \mathbf{q} \in \mathbb{R}^{12}$ is the state space vector. As the design technique we use the classical pole placement method. Controller scheme is given in Figure 3.1. The scheme of the controller consists of six cascaded lead-lag controllers where cascade is placed over control of the (pitch $\leftrightarrow$ x directional movement) and (roll $\leftrightarrow$ y directional movement) due to the properties of the quadrotor dynamics.

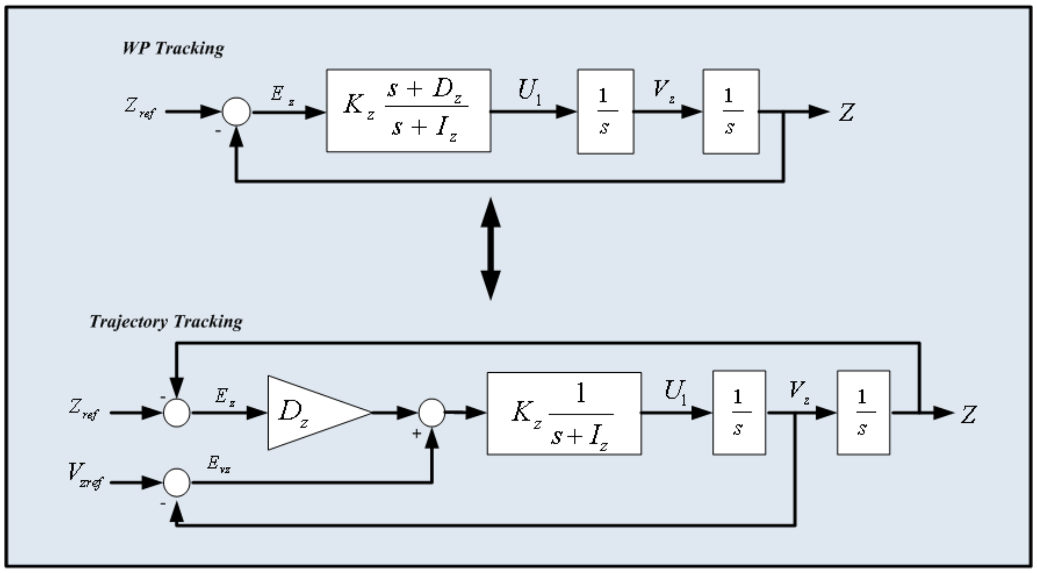


Figure 3.1: Baseline controller for one axis.

The described controller is then used to control the full nonlinear quadrotor model (2.11). Stability of the output feedback controller using cascade lead-lag controllers is verified through MIMO Bode analysis using Control Systems Toolbox and the results are given in Figure 3.1. Controller performance is verified through simulation results provided in Figure 3.3.

3.2 Controller design using feedback linearization

The basic idea of feedback linearization (FL) is to transform nonlinear system dynamics into linear system dynamics ([23], [25], [54]). Conventional control techniques like pole placement and linear quadratic optimal control theory can then be applied to the linear system. In robotics, this technique is commonly referred to as computed torque control ([25]). The control objective is to transform the vehicle dynamics (2.11) into a linear system $\dot{\boldsymbol{\nu}} = \boldsymbol{\vartheta}$, where $\boldsymbol{\vartheta}$ can be interpreted as a commanded

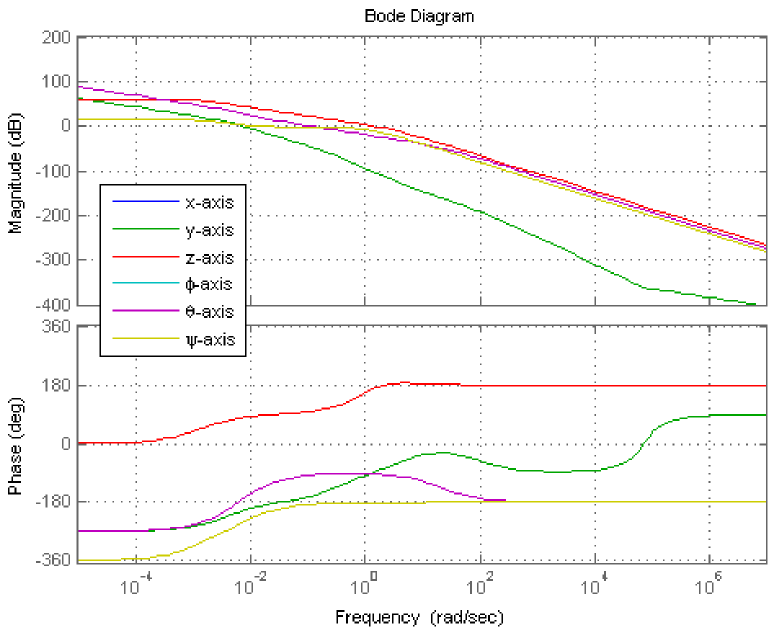


Figure 3.2: MIMO Bode plot for quadrotor output feedback control using cascade of PD controllers

acceleration vector. The nonlinearities can be canceled out by simply selecting the control algorithm as follows

$$\boldsymbol{\tau} = \mathbf{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{D}\boldsymbol{\nu} + \mathbf{G}(\boldsymbol{\eta}) + \mathbf{M}\boldsymbol{\vartheta}, \tag{3.2}$$
$$\dot{\boldsymbol{\nu}} = \boldsymbol{\vartheta},$$

where the commanded acceleration vector $\boldsymbol{\vartheta}$ is chosen using pole placement technique. Considering the nature of the dynamical system we are dealing with, we can write

$$\mathbf{y} = \mathbf{F}\boldsymbol{\eta} = [z \quad \phi \quad \theta \quad \psi]^T,$$
$$\dot{\mathbf{y}} = \mathbf{F}\dot{\boldsymbol{\eta}} = \mathbf{F}\boldsymbol{\nu},$$
$$\ddot{\mathbf{y}} = \mathbf{F}\ddot{\boldsymbol{\eta}} = \mathbf{F}\dot{\boldsymbol{\nu}} = \mathbf{F}\mathbf{M}^{-1}[-\mathbf{C}(\boldsymbol{\nu})\boldsymbol{\nu} - \mathbf{D}\boldsymbol{\nu} - \mathbf{G}(\boldsymbol{\eta}) + \boldsymbol{\tau}],$$
$$\ddot{\mathbf{y}} = \mathbf{F}\dot{\boldsymbol{\nu}} = \mathbf{F}\boldsymbol{\vartheta},$$

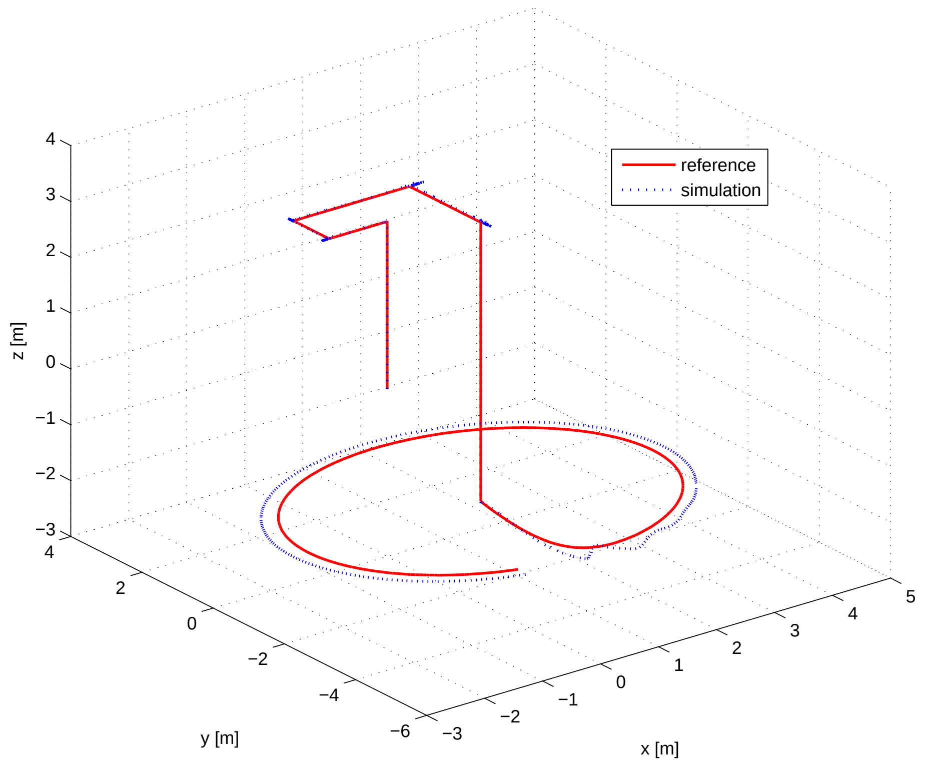


Figure 3.3: Trajectory and waypoint tracking using the cascade lead-lag controller.

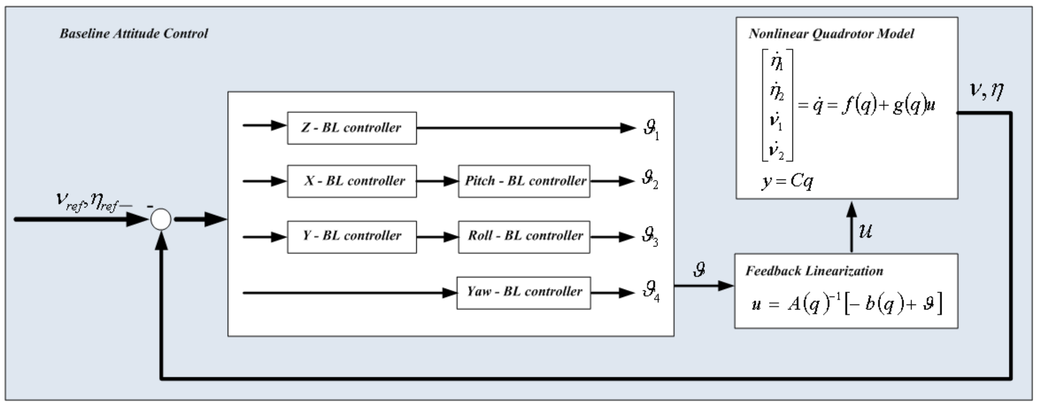


Figure 3.4: Feedback linearization

where $\mathbf{y} \in \mathbb{R}^4$ represents system outputs. Due to this property, we can proceed with the controller design using the input-output feedback linearization algorithm presented in ([23]). The model derived in (Section 2.1) can be represented in state space form

$$\dot{\mathbf{q}} = \mathbf{f}(\mathbf{q}) + \mathbf{g}(\mathbf{q})\,\mathbf{u}, \quad \mathbf{y} = \mathbf{h}(\mathbf{q}), \tag{3.3}$$

where $\mathbf{u} = [u_1\, u_2 \dots u_m]^T$, $\mathbf{y} = [y_1\, y_2 \dots y_l]^T$, and

$$\mathbf{g}(\mathbf{q}) = [g_1(\mathbf{q}) \dots g_m(\mathbf{q})], \quad \mathbf{h}(\mathbf{q}) = [h_1(\mathbf{q}) \dots h_l(\mathbf{q})]^T,$$

are $n \times m$ matrix, an l-dimensional vector, and n is the system state space dimension, respectively. Since the quadrotor model is a nonlinear underactuated system (i.e., the number of inputs m is less than the number of outputs l), in order to deal with this restriction, we exploit the differential flatness property. Loosely speaking, differentially flat systems are systems in which all states and inputs can be expressed as functions of the outputs and a finite number of their derivatives. The formal definition can be found in [55]. Moreover, these systems have the useful property that there is a one-to-one mapping between trajectories in output space and trajectories in state and input space. The nonlinear system given in (3.3) has a relative degree r_i at a point $\mathbf{q}^0$ if

$$L_{gj} L_{gf}^k h_i(\mathbf{q}) = 0,$$

for all $1 \le j \le 4$, $k < r_i - 1$, $1 \le i \le 4$, and for all $\mathbf{q}$ in a neighborhood of $\mathbf{q}^o$ the 4×4 matrix

$$\alpha(q) = \begin{bmatrix} L_{g1} L_f^{r_1-1} h_1(\mathbf{q}) \dots L_{g4} L_f^{r_1-1} h_1(\mathbf{q}) \\ \vdots \\ L_{g1} L_f^{r_4-1} h_4(\mathbf{q}) \dots L_{g4} L_f^{r_4-1} h_4(\mathbf{q}) \end{bmatrix},$$

is nonsingular at $\mathbf{q} = \mathbf{q}^o$ ([23]). If these two conditions are satisfied, we can write,

$$\begin{bmatrix} y_1^{r_1} \\ y_2^{r_2} \\ y_3^{r_3} \\ y_4^{r_4} \end{bmatrix} = \begin{bmatrix} L_f^{r_1} \\ L_f^{r_2} \\ L_f^{r_3} \\ L_f^{r_4} \end{bmatrix} \boldsymbol{\alpha}(\mathbf{q}).$$

If $\boldsymbol{\alpha}(q)$ is invertible at $\mathbf{q}^o$, then the state feedback is given by $\mathbf{u} = \boldsymbol{\alpha}(\mathbf{q})^{-1}[-\mathbf{b}(\mathbf{q}) + \boldsymbol{\vartheta}]$ and it will result in a closed-loop system that is linear from input $\mathbf{u}$ to output $\mathbf{y}$. Due to the fact that the quadrotor model considered in this paper is a complex MIMO nonlinear system, before starting the design process, we must first check the existence conditions for feedback linearization ([23]). According to Lemma 5.2.1. and Remark 5.2.1. in ([23]), the condition that the matrix $\mathbf{g}(\mathbf{q}^o)$ has rank m ($m = 4$ in our case) is necessary for the existence of any set of m output functions such that the system has some relative degree at $\mathbf{q}^o$. It is not difficult to verify that rank($\mathbf{g}(\mathbf{q}^o) = 4$) for our model. If the system has a relative degree equal to n at $\mathbf{q}^o$, it can be transformed into a fully linear and controllable system according to Lemma 4.2.1. in ([23]). In this case, the system has trivial zero dynamics. If the system in (3.3) with $\mathbf{f}(0) = 0$, $\mathbf{h}(0) = 0$ and if $\mathbf{y}(t) = 0$ for all t, then necessarily $\mathbf{q}(0) = 0$ and $\mathbf{u}(t) = 0$ for all t, and the system has trivial zero dynamics. In the case of our control system that condition is not fulfilled because $\mathbf{f}(0) \neq 0$, thus, the quadrotor system has nontrivial zero dynamics.

Since the matrix $\boldsymbol{\alpha}(\mathbf{q}^o)$ is nonsingular, we can use exact state feedback to linearize the system and obtain its relative degree. The relative degree is $r = 2 + 2 + 2 + 2 = 8$, and the dimension of the system is $n = 12$. Hence, four states belong to zero dynamics. Therefore the system is not fully feedback linearizable but only partially, and can be decomposed into a linear and controllable part, and a part which represents zero dynamics. In order to be able to design a controller which will stabilize and control this type of a system, we have to identify which states belong to the zero dynamics and to prove that they are stable. We approach the notion of zero dynamics through the idea of *zeroing the output* ([23]). In order to yield $\mathbf{y}(t) = 0$ for all times, the system must evolve on the subset,

$$Z^* = \left\{\mathbf{q} \in \mathbb{R}^n : L_f^k h_i(\mathbf{q}) = 0, 0 \leq k \leq r_i - 1, 1 \leq i \leq m\right\}.$$

We can construct the vector field

$$\mathbf{f}^*(\mathbf{q}) = \mathbf{f}(\mathbf{q}) + \mathbf{g}(\mathbf{q})(-\mathbf{A}^{-1}\mathbf{b}(\mathbf{q}))$$

tangent to Z^*. Any trajectory of the closed loop system $\dot{\mathbf{q}} = \mathbf{f}^*(\mathbf{q})$ starting at the point of Z^* remains in Z^*. The vector field $\mathbf{f}^*(\mathbf{q})|_{Z^*}$, that is a constraint of $\mathbf{f}^*(\mathbf{q})$ to Z^* describes the zero dynamics of the system. We define four states of the zero dynamics as follows

$$\begin{aligned} \dot{q}_1 &= q_7, & \dot{q}_2 &= q_8, \\ \dot{q}_7 &= -\tfrac{c_{\mu x}}{m} q_7, & \dot{q}_8 &= -\tfrac{c_{\mu y}}{m} q_8, \end{aligned} \tag{3.4}$$

where $\mathbf{q} \in \mathbb{R}^{12}$ is a state vector and $\frac{c_{\mu x}}{m}$ and $\frac{c_{\mu y}}{m}$ are positive constants.

Theorem 3.2.1. *The zero dynamics given by (3.4) is stable.*

Proof. Since the zero dynamics in this case is a linear system, the stability analysis is trivial. Considering the structure of the system (3.4), we can analyze it as two separate systems of second order $\dot{\boldsymbol{\rho}} = \mathbf{A}_i\boldsymbol{\rho},\ i = 1, 2,\ \boldsymbol{\rho} \in \mathbb{R}^2$. Therefore, we can use the well-known theory of second-order linear systems ([54]). Since each of these systems has one eigenvalue of matrix $\mathbf{A}_i$ zero and the other eigenvalue has a negative real part (i.e., it is stable), the matrices $\mathbf{A}_i$ have a nontrivial null space. Any vector in the null space of $\mathbf{A}_i$ is an equilibrium point for the system, i.e., the system has an equilibrium subspace rather than an equilibrium point. From the phase portrait (Figure 3.2) we can see that all trajectories converge to the equilibrium subspace (q_7 axis i.e., q_8 axis) since the nonzero eigenvalues are stable. Therefore, the zero dynamics are stable, and we are able to design the controller to stabilize and control the nonlinear quadrotor model. □

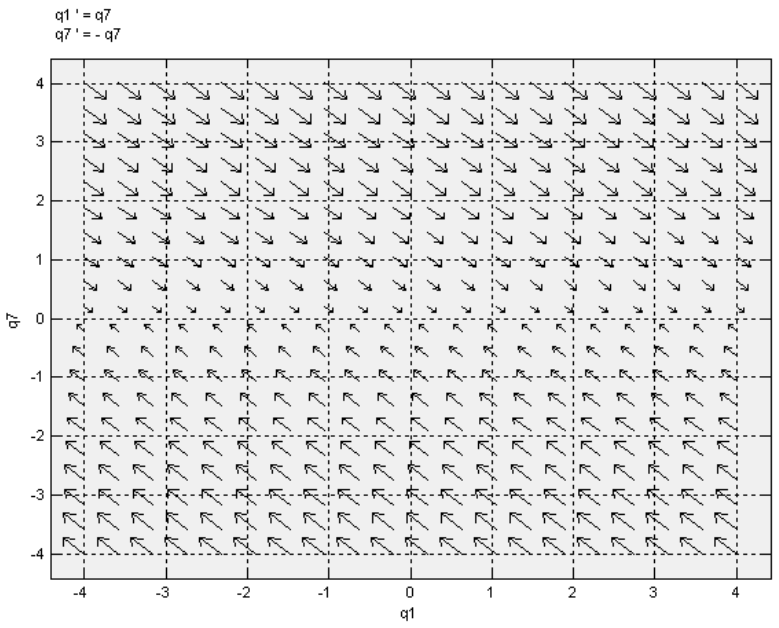


Figure 3.5: Phase portrait for a second order subsystem of the zero dynamics.

The proposed controller consists of two parts: the first part is nonlinear which linearizes the system through feedback linearization; the second part of the controller is linear and its purpose is to stabilize and control all six DoF of the quadrotor (i.e., orientation $\boldsymbol{\eta}_2$ and position $\boldsymbol{\eta}_1$) through a cascade structure. The proposed control algorithm is implemented in *Matlab* and simulation results show its efficiency (Figure 3.2). One of the main drawbacks of feedback linearization is that the controller is not able to deal with model uncertainties. In this proposal we use the change in CoG as the uncertain parameter. The controller based on feedback linearization derived above fails to stabilize the unbalanced quadrotor (Figure 4.5). This fact has motivated us to design an adaptive controller presented in the Chapter 4.

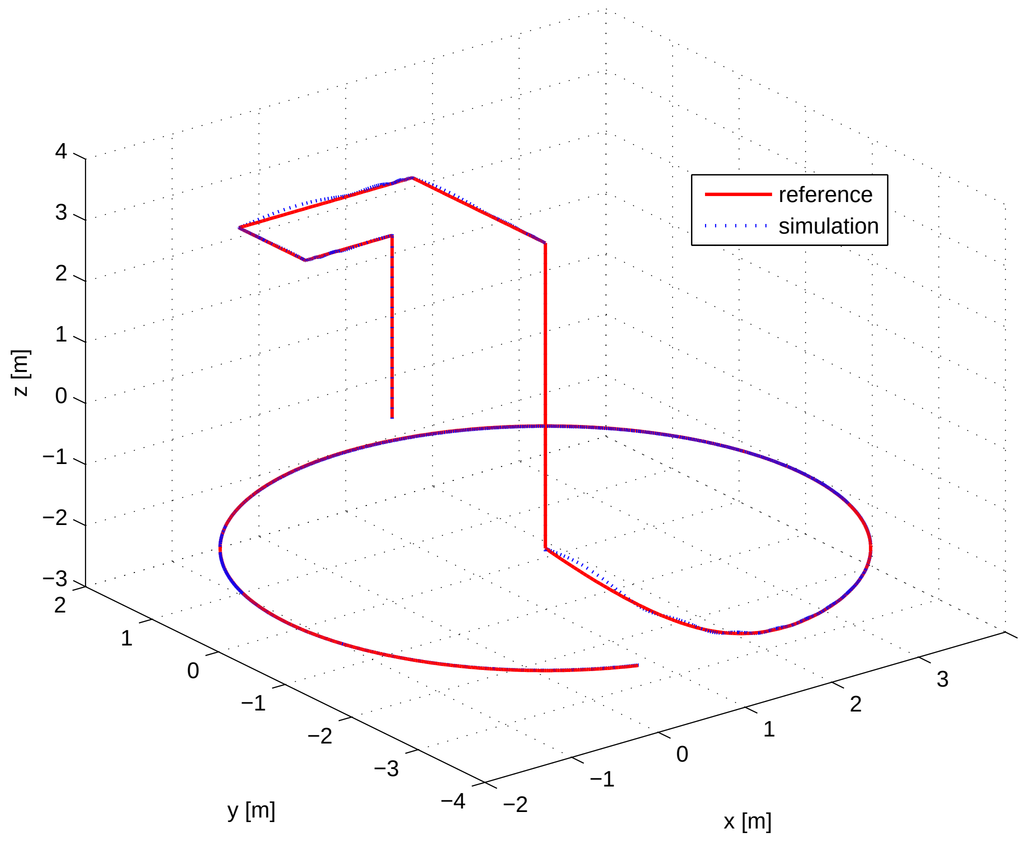


Figure 3.6: Trajectory and waypoint tracking using the controller based on feedback linearization.

3.3 Baseline self-tuning controller

In order to tune the parameters of controllers presented in previous sections we developed a self-tuning technique based on optimization theory. The procedure for tuning is depicted on Figure 3.7. First, we obtain transfer functions for each of the six axis around an equilibrium point for the system presented in Section 2.1. Then using *Matlab SISO tool* we find an initial stable controller for each transfer function. Having this, we start the self-tuning procedure as shown on Figure 3.7. Our goal is to minimize the objective function

$$J = \int |et|\, dt \tag{3.5}$$

where $e = [x_{ref} - x \; y_{ref} - y \; z_{ref} - z \; \phi_{ref} - \phi \; \theta_{ref} - \theta \; \psi_{ref} - \psi]^T$, $e \in \mathbb{R}^6$ is the tracking error, $t \in \mathbb{R}$ represents time, and $|\cdot|$ is a vector norm. This function is minimized, by changing the parameters of the controllers, gains, poles and zeros of each controller for each of the six axis. We use the solver from *Matlab Optimization Toolbox* for solving this optimal control problem. This procedure is repeated six times, as we are tuning each of the axis separately ensuring stability, by imposing the bounds on the parameters in each optimization step. At the end this problem of self-tuning is defined as a non-convex constrained nonlinear optimization problem with hybrid behavior. By hybrid behavior we are referring to jumps in the tuning procedure in moments when we start tuning the next axis. Since each of the axis is stable before entering the tuning process

3.4 Conclusion

In this section we present the baseline controller for a quadrotor. First we design a lead-lag controller based on a model linearized around a working point. Secondly we designed a controller based on output feedback linearization. Both of the controllers were tunes using the self-tuning technique described in the last section and were implemented in simulation. The lead-lag controller was implemented experimentally which was described in more details in [56]. Currently there is a considerable amount of research in the design of the baseline control of quadrotors with both simulation and experimental results. In this chapter we improved upon the tuning process with the hybrid self-tuning technique based on optimization. The room for future research would be to extend the hybrid self-tuning controller for experimental setup which would ensure modularity and speed up the process of tuning baseline controllers.

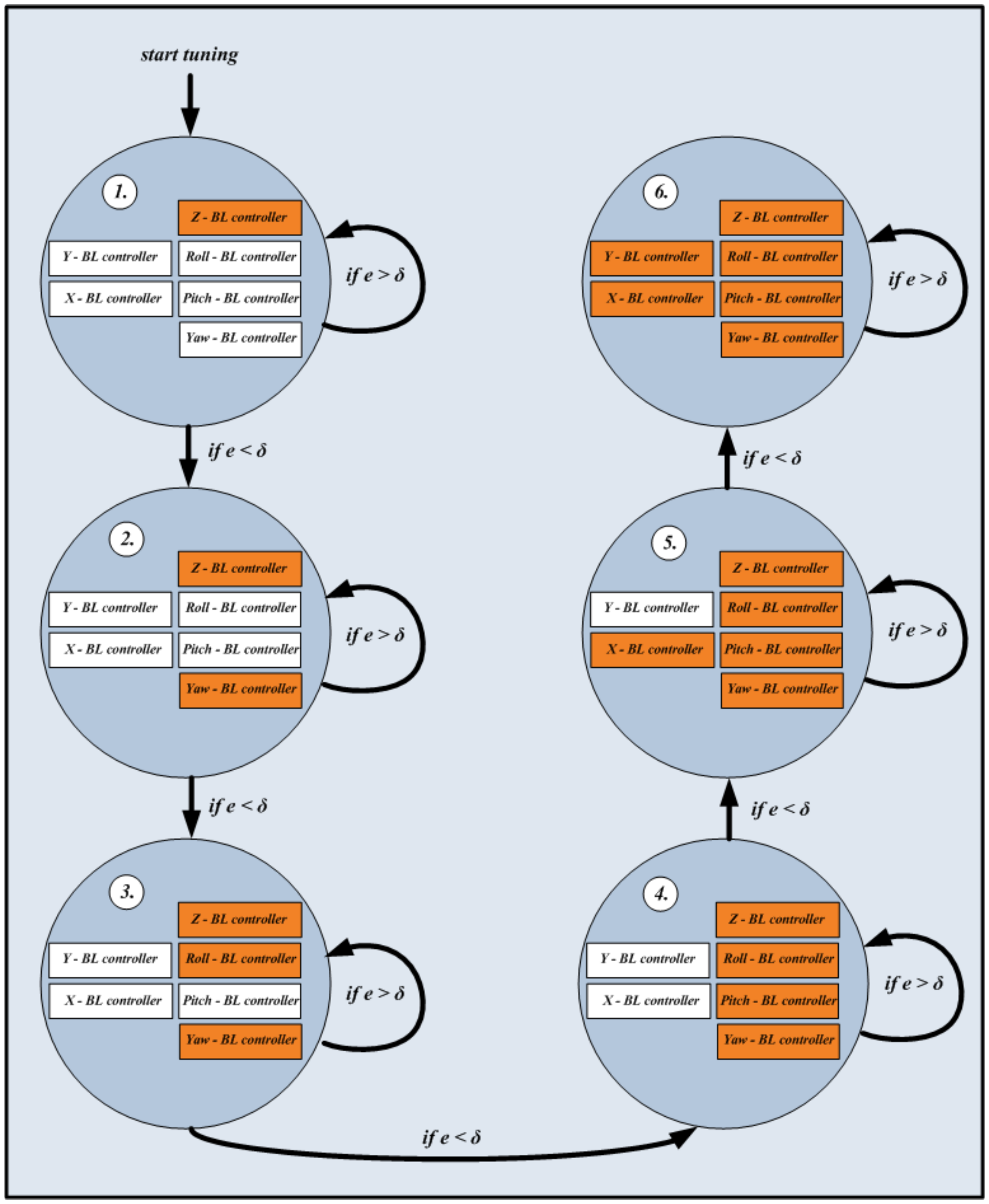


Figure 3.7: Hybrid Self Tuning Controller block diagram

Chapter 4

The change in the center of gravity of the quadrotor

4.1 Analysis of the nonlinear model with respect to the center of gravity

One of the main drawbacks of feedback linearization is that the controller is not able to deal with model uncertainties. In this proposal we use the change in CoG as the uncertain parameter. The controller based on feedback linearization derived above fails to stabilize the unbalanced quadrotor (Figure 4.5).

Considering the problem of linearization in the presence of uncertainty we assume that uncertainties are modeled as a perturbation of 3.3

$$\begin{aligned}
\dot{\mathbf{q}} &= \mathbf{f}(\mathbf{q}) + \mathbf{\Delta f}(\mathbf{q}) + \mathbf{g}_1(\mathbf{q})\,\mathbf{u}_1 + \ldots + \mathbf{g}_p(\mathbf{q})\,\mathbf{u}_p + \mathbf{\Delta g}_1(\mathbf{q})\,\mathbf{u}_1 + \ldots + \mathbf{\Delta g}_p(\mathbf{q})\,\mathbf{u}_p,\\
\mathbf{y}_1 &= \mathbf{h}_1(\mathbf{q}),\\
&\vdots \\
\mathbf{y}_p &= \mathbf{h}_p(\mathbf{q}),
\end{aligned} \tag{4.1}$$

where in the discussion of the *CoG* problem we may treat it as the perturbation $\mathbf{\Delta f}(\mathbf{q})$. Using the following proposition we are able to determine robustness of feedback linearization.

Proposition 1 (MIMO Robust Linearization [24])**.** *Consider the uncertain nonlinear system 4.2. Assume that the nominal system 3.3 has vector relative degree $\gamma_1, \ldots, \gamma_p$. Then, if the perturbation $\boldsymbol{\Delta f}$, $\boldsymbol{\Delta g}_j$ satisfy for $j = 1, \ldots, p$*

$$\begin{aligned}
&L_{\Delta f} L_f^i h_j \equiv 0 \quad for \quad 0 \le i \le \gamma_j - 1,\\
&L_{\Delta g_k} L_f^i h_j \equiv 0 \quad for \quad 0 \le i \le \gamma_j - 1, 1 \le k \le p
\end{aligned} \tag{4.2}$$

the linearizing control law for the nominal system also linearizes the perturbed system.

Using 4.2 we compute

$$\begin{bmatrix} L_{\Delta f} L_f^{r_1-1} h_1(\mathbf{q}) \ldots L_{\Delta f} L_f^{r_1-1} h_1(\mathbf{q}) \\ \vdots \\ L_{\Delta f} L_f^{r_4-1} h_4(\mathbf{q}) \ldots L_{\Delta f} L_f^{r_4-1} h_4(\mathbf{q}) \end{bmatrix} = \begin{bmatrix} 0 & y_G & -x_G & 0 \\ \frac{m y_G}{I_x} & 0 & 0 & 0 \\ \frac{-m x_G}{I_y} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \neq 0 \tag{4.3}$$

This shows us that feedback linearization for the nominal system (3.3) is not able to linearize the perturbed system (4.2). Figure 4.1 shows how the eigenvalues of the closed loop system change with respect to coordinates of CoG. We can see that the system becomes unstable for a certain movement of the center of gravity of the

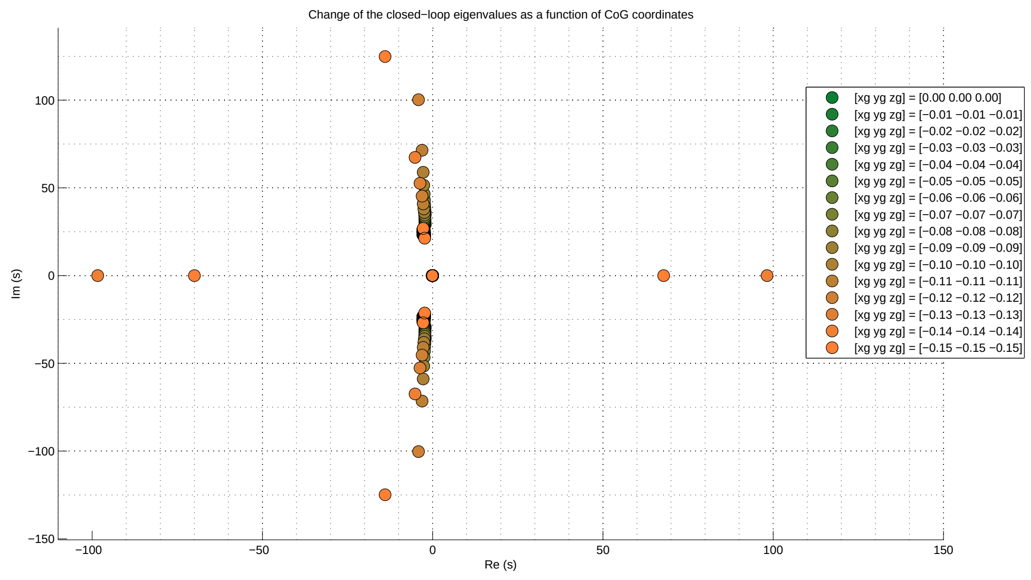


Figure 4.1: Eigenvalues of the Jacobians of the closed-loop system change with respect to coordinates of CoG.

system. Similarly on Figure 4.2 we can see that the eigenvalues of the Jacobian matrix of the closed loop system becomes stable as we increase the moments of inertia of the system.

4.2 Adaptive control for the change in the center of gravity

The adaptive controller is derived using the algorithm proposed in [25]. Considering the control algorithm (3.2) we can write

$$\boldsymbol{\tau} = \hat{\mathbf{C}}\left(\boldsymbol{\nu}\right)\boldsymbol{\nu} + \mathbf{D}\boldsymbol{\nu} + \hat{\mathbf{G}}\left(\boldsymbol{\eta}\right) + \hat{\mathbf{M}}\boldsymbol{\vartheta}, \tag{4.4}$$

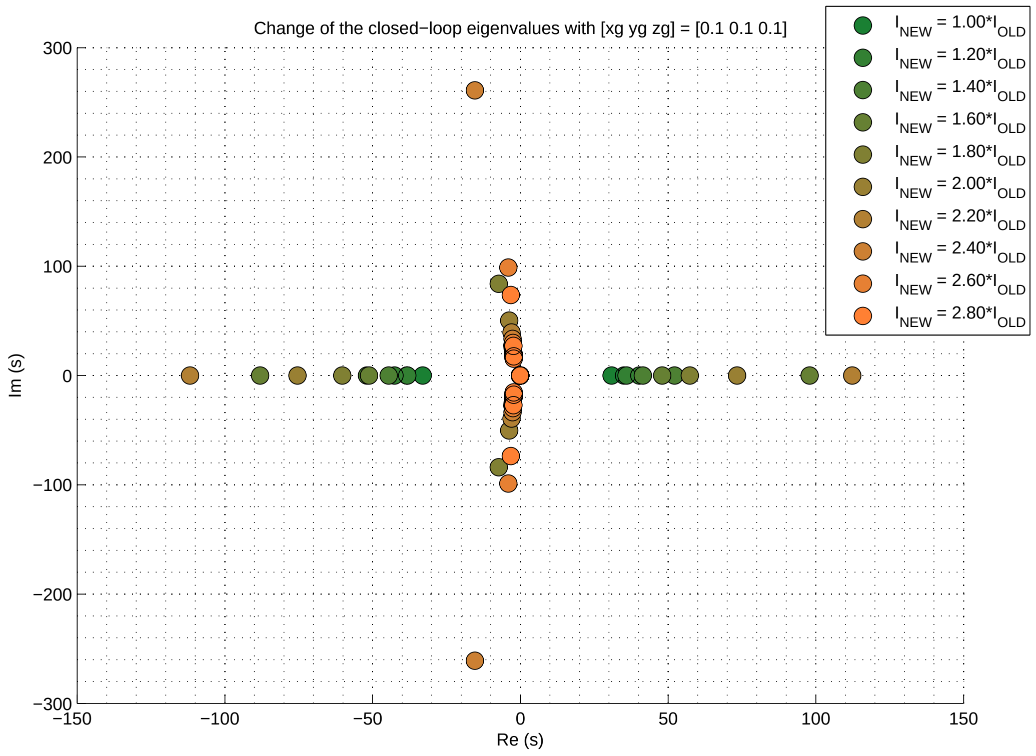


Figure 4.2: Eigenvalues of the Jacobians of the closed-loop system become stable with different inertia.

where the *hat* denotes estimates of the adaptive parameter. Now, the error dynamics can be denoted as

$$\mathbf{M}\left[\dot{\boldsymbol{\nu}} - r\right] = \left[\hat{\mathbf{M}} - \mathbf{M}\right] \boldsymbol{\vartheta} + \left[\hat{\mathbf{C}}\left(\boldsymbol{\nu}\right) - \mathbf{C}\left(\boldsymbol{\nu}\right)\right] \boldsymbol{\nu} + \left[\hat{\mathbf{G}}\left(\boldsymbol{\eta}\right) - \mathbf{G}\left(\boldsymbol{\eta}\right)\right].$$

Because quadrotor equations of motion are linear in the parameter vector $\boldsymbol{\gamma} = \mathbf{r}_G$, we can apply the following parameterization

$$\boldsymbol{\Phi}\left(\boldsymbol{\nu}, \boldsymbol{\eta}\right) \tilde{\boldsymbol{\gamma}} = \left[\hat{\mathbf{M}} - \mathbf{M}\right] \boldsymbol{\vartheta} + \left[\hat{\mathbf{C}}\left(\boldsymbol{\nu}\right) - \mathbf{C}\left(\boldsymbol{\nu}\right)\right] \boldsymbol{\nu} + \left[\hat{\mathbf{G}}\left(\boldsymbol{\eta}\right) - \mathbf{G}\left(\boldsymbol{\eta}\right)\right].$$

In the above expression, $\tilde{\boldsymbol{\gamma}} = \hat{\boldsymbol{\gamma}} - \boldsymbol{\gamma} \in \mathbb{R}^{3\times1}$ is the unknown parameter error vector and $\boldsymbol{\Phi}\left(\boldsymbol{\nu}, \boldsymbol{\eta}\right) \in \mathbb{R}^{6\times3}$ is a known matrix function of measured signals usually referred

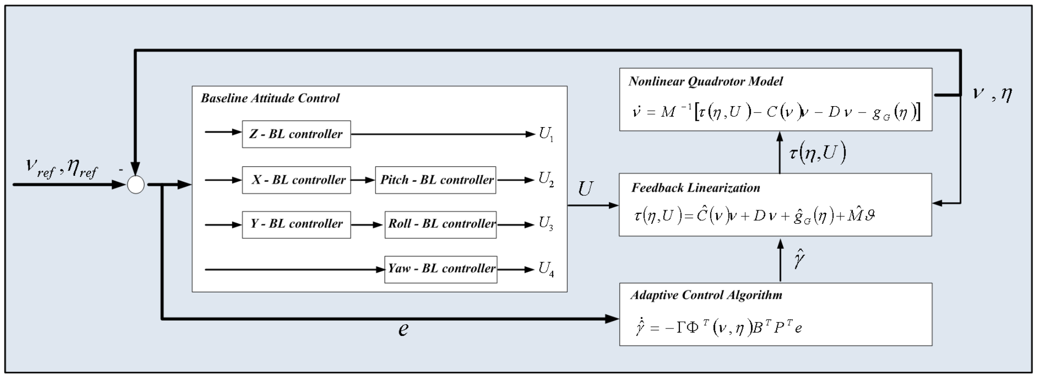

ν_{ref}, η_{ref}
-
Baseline Attitude Control
Z - BL controller
X - BL controller
Pitch - BL controller
Y - BL controller
Roll - BL controller
Yaw - BL controller
U_1
U_2
U_3
U_4
U
e
Nonlinear Quadrotor Model
$\dot{\nu} = M^{-1}[\tau(\eta,U) - C(\nu)\nu - D\nu - g_G(\eta)]$
ν, η
$\tau(\eta,U)$
Feedback Linearization
$\tau(\eta,U) = \hat{C}(\nu)\nu + D\nu + \hat{g}_G(\eta) + \hat{M}\vartheta$
$\hat{\gamma}$
Adaptive Control Algorithm
$\dot{\hat{\gamma}} = -\Gamma\Phi^T(\nu,\eta)B^T P^T e$

Figure 4.3: Adaptive feedback linearization

to as the regressor matrix.

$$\boldsymbol{\Phi}(\boldsymbol{\nu},\boldsymbol{\eta}) = m\begin{bmatrix} -r^2-q^2 & pq-\vartheta_6 & pr+\dot{q} \\ pq+\vartheta_6 & -p^2-r^2 & qr-\vartheta_4 \\ pr-\vartheta_5 & qr+\vartheta_4 & -p^2-q^2 \\ 0 & -qu+pv+\vartheta_3+g\cos\theta\cos\phi & -ru+pw+g\cos\theta\sin\phi \\ qu-pv-\vartheta_3-g\cos\theta\cos\phi & 0 & -rv+qw+g\sin\theta \\ ru-pw-g\cos\theta\cos\phi & rv-qw-g\sin\theta & 0 \end{bmatrix}. \tag{4.5}$$

We obtain (4.5) by rewriting (2.5) (2.8)

$$\begin{aligned} \sum \mathbf{F}_E &= m\left[\boldsymbol{\nu}_2 \times \boldsymbol{\nu}_1 + \dot{\boldsymbol{\nu}}_1 + \dot{\boldsymbol{\nu}}_2 \times \mathbf{r}_G + \boldsymbol{\nu}_2 \times (\boldsymbol{\nu}_2 \times \mathbf{r}_G)\right] \\ &= m\dot{\boldsymbol{\nu}}_1 + m\begin{bmatrix} qw-rv \\ ru-pw \\ pv-qu \end{bmatrix} + m\begin{bmatrix} -r^2-q^2 & pq-\dot{r} & pr+\dot{q} \\ pq+\dot{r} & -p^2-r^2 & qr-\dot{p} \\ pr-\dot{q} & qr+\dot{p} & -p^2-q^2 \end{bmatrix} r_G \\ \sum \mathbf{T}_E &= \mathbf{I}_A\dot{\boldsymbol{\nu}}_2 + \boldsymbol{\nu}_2 \times \mathbf{I}_A\boldsymbol{\nu}_2 + m\mathbf{r}_G \times (\dot{\boldsymbol{\nu}}_1 + \boldsymbol{\nu}_2 \times \boldsymbol{\nu}_1) \\ &= \mathbf{I}_A\dot{\boldsymbol{\nu}}_2 + \begin{bmatrix} (I_z-I_y)qr \\ (I_{xx}-I_z)pr \\ (I_y-I_{xx})pq \end{bmatrix} - m\begin{bmatrix} 0 & qu-pv-\dot{w} & ru-pw+\dot{v} \\ -qu+pv+\dot{w} & 0 & rv-qw-\dot{u} \\ -ru+pw-\dot{v} & -rv+qw+\dot{u} & 0 \end{bmatrix} \mathbf{r}_G. \end{aligned} \tag{4.6}$$

In a compact form we can write

$$\begin{bmatrix} \sum \mathbf{F}_E \\ \sum \mathbf{T}_E \end{bmatrix} = \begin{bmatrix} m\mathbf{I}_{3\times3} & 0 \\ 0 & \mathbf{I}_A \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{\nu}}_1 \\ \dot{\boldsymbol{\nu}}_2 \end{bmatrix} + \begin{bmatrix} m(qw-rv) \\ m(ru-pw) \\ m(pv-qu) \\ (I_z-I_y)qr \\ (I_x-I_z)pr \\ (I_y-I_x)pq \end{bmatrix} + m\begin{bmatrix} -r^2-q^2 & pq-\dot{r} & pr+\dot{q} \\ pq+\dot{r} & -p^2-r^2 & qr-\dot{p} \\ pr-\dot{q} & qr+\dot{p} & -p^2-q^2 \\ 0 & -qu+pv+\dot{w} & -ru+pw-\dot{v} \\ qu-pv-\dot{w} & 0 & -rv+qw+\dot{u} \\ ru-pw+\dot{v} & rv-qw-\dot{u} & 0 \end{bmatrix} \mathbf{r}_G$$

where

$$\begin{bmatrix} \sum \mathbf{F}_E \\ \sum \mathbf{T}_E \end{bmatrix} = \mathbf{D}\boldsymbol{\nu} + \mathbf{G}\left(\boldsymbol{\eta}\right) + \boldsymbol{\tau} \tag{4.7}$$

Writing the expression for the tracking error dynamics in state-space form yields,

$$\dot{\mathbf{e}} = \mathbf{A}\mathbf{e} + \mathbf{B}\boldsymbol{\Phi}\left(\boldsymbol{\nu}, \boldsymbol{\eta}\right)\tilde{\boldsymbol{\gamma}}, \tag{4.8}$$

where $\mathbf{e} = \begin{bmatrix}\tilde{\boldsymbol{\eta}} & \dot{\tilde{\boldsymbol{\eta}}}\end{bmatrix}^T = [\boldsymbol{\eta}_{ref} - \boldsymbol{\eta} \quad \dot{\boldsymbol{\eta}}_{ref} - \dot{\boldsymbol{\eta}}]^T \in \mathbb{R}^{12\times 1}$ is a vector of the tracking error, $\mathbf{A} \in \mathbb{R}^{12\times 12}$ is a matrix containing the parameters of the linear controller and $\mathbf{B} = [0 \quad M^{-1}]^T \in \mathbb{R}^{12\times 6}$.

Theorem 4.2.1. *The tracking error given by (4.8) is asymptotically stable and parameter error $\tilde{\gamma} = \hat{\gamma} - \gamma$ is bounded.*

Proof. We start by choosing a Lyapunov function candidate as

$$V(e, \tilde{\gamma}, t) = \mathbf{e}^T\mathbf{P}\mathbf{e} + \tilde{\boldsymbol{\gamma}}^T\boldsymbol{\Gamma}^{-1}\tilde{\boldsymbol{\gamma}},$$

where $\mathbf{P} = \mathbf{P}^T > 0$ satisfies Lyapunov stability equation for linear systems and $\boldsymbol{\Gamma} = \boldsymbol{\Gamma}^T > 0$. By differentiating $V(\mathbf{e}, \tilde{\boldsymbol{\gamma}}, t)$ with respect to time we get,

$$\dot{V}(\mathbf{e}, \tilde{\boldsymbol{\gamma}}, t) = \mathbf{e}^T(\mathbf{A}^T\mathbf{P} + \mathbf{P}\mathbf{A})\mathbf{e} + 2\mathbf{e}^T\mathbf{P}\mathbf{B}\boldsymbol{\Phi}\left(\boldsymbol{\nu}, \boldsymbol{\eta}\right)\tilde{\gamma} - 2\tilde{\boldsymbol{\gamma}}^T\boldsymbol{\Gamma}^{-1}\dot{\tilde{\boldsymbol{\gamma}}},$$

By choosing the parameter update rule (assuming $\dot{\boldsymbol{\gamma}} = 0$) as

$$\dot{\tilde{\gamma}} = -\boldsymbol{\Gamma}\ \boldsymbol{\Phi}^T\left(\boldsymbol{\nu}, \boldsymbol{\eta}\right)\ \mathbf{B}^T\ \mathbf{P}^T e,$$

we get

$$\dot{V}(\mathbf{e}, \tilde{\boldsymbol{\gamma}}, t) = \mathbf{e}^T(\mathbf{A}^T\mathbf{P} + \mathbf{P}\mathbf{A})\mathbf{e} \leq -\mathbf{e}^T\mathbf{Q}\mathbf{e} \leq 0,$$

where $\mathbf{Q} = \mathbf{Q}^T > 0$ is the matrix that satisfies Lyapunov stability equation for linear systems. However, $\dot{V}(\mathbf{e}, \tilde{\boldsymbol{\gamma}}, t)$ is only negative semidefinite because $\dot{V}(\mathbf{e}, \tilde{\boldsymbol{\gamma}}, t) = 0$ for $\mathbf{e} = 0$ irrespective of the value of $\tilde{\boldsymbol{\gamma}}$; that is, $\dot{V}(\mathbf{e}, \tilde{\boldsymbol{\gamma}}, t) = 0$ along $\tilde{\boldsymbol{\gamma}}$ - axis.

$$\ddot{V}(\mathbf{e}, \tilde{\boldsymbol{\gamma}}, t) = -\dot{\mathbf{e}}^T \mathbf{Q}\mathbf{e} - \mathbf{e}^T \mathbf{Q}\dot{\mathbf{e}} = \mathbf{e}^T \left(\mathbf{A}^T \mathbf{Q} + \mathbf{Q}\mathbf{A}\right) \mathbf{e} + \tilde{\boldsymbol{\gamma}} \boldsymbol{\Phi} \left(\boldsymbol{\nu}, \boldsymbol{\eta}\right)^T \mathbf{B}^T \mathbf{Q}\mathbf{e} + \mathbf{e}^T \mathbf{Q}\mathbf{B}\boldsymbol{\Phi}\left(\boldsymbol{\nu}, \boldsymbol{\eta}\right) \tilde{\boldsymbol{\gamma}}$$

since

$$\|\mathbf{e}\| < \infty, \ \|\tilde{\boldsymbol{\gamma}}\| < \infty, \ A, B, Q = const, \ \|\boldsymbol{\Phi}\left(\boldsymbol{\nu}, \boldsymbol{\eta}\right)\| < \infty \tag{4.9}$$

then

$$\left\|\ddot{V}(\mathbf{e}, \tilde{\boldsymbol{\gamma}}, t)\right\| < \infty. \tag{4.10}$$

By showing that $\ddot{V}(\mathbf{e}, \tilde{\boldsymbol{\gamma}}, t)$ is bounded we show that $\dot{V}(\mathbf{e}, \tilde{\boldsymbol{\gamma}}, t)$ is uniformly continuous in time. Now, by applying Barbalat's lemma we prove that $\mathbf{e}$ asymptotically converges to zero and $\tilde{\boldsymbol{\gamma}}$ is bounded. $\square$

The proposed adaptive controller showed in (Figure 4.3) is implemented in Matlab/Simulink and its performance is shown in (Figure 4.6).

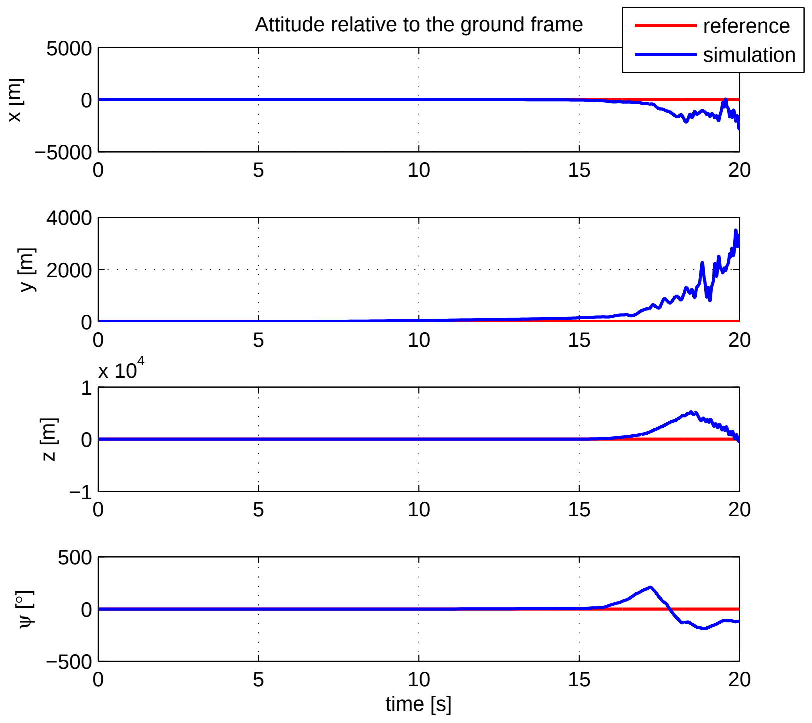
4.2.1 Simulation results

The controllers derived and presented in Chapter 3 are simulated using Matlab/Simulink. The linear output feedback controller is used to control the full nonlinear quadrotor model given in Section 2.1. The simulations show that the system tracks a given trajectory with accuracy when dealing with a balanced nonlinear quadrotor model (Figure 3.3). In the case of an unbalanced quadrotor this controller fails to stabilize the system (Figure 4.4).

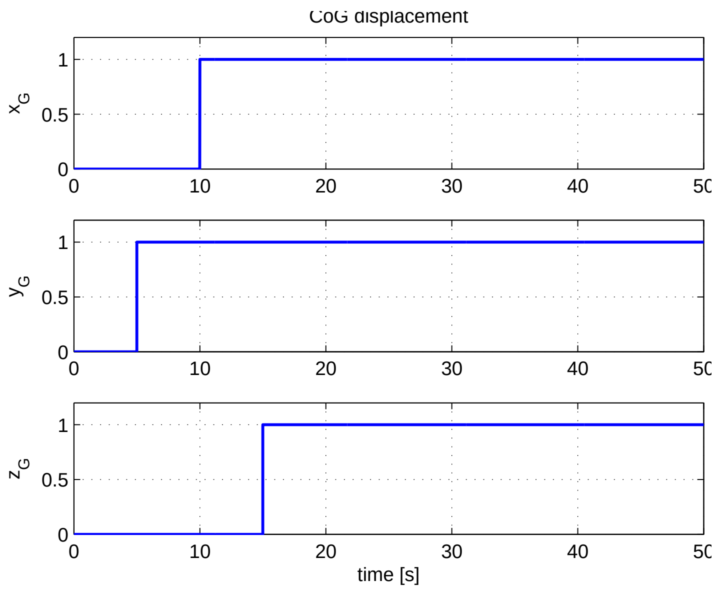
The nonlinear controller based on the input-output feedback linearization (Section 3.2) yields an accurate tracking performance when dealing with a balanced nonlinear quadrotor model (Figure 3.2). In the case of an unbalanced quadrotor this controller fails to stabilize the system as well (Figure 4.5). By adding the adaptive part to the nonlinear controller based on output feedback linearization (Section 4.2), we solve the stabilization and tracking problem. The proposed algorithm succeeds in stabilizing an unbalanced quadrotor (Figure 4.6). The adaptive feedback linearization shows good tracking performance when dealing with dynamical changes in quadrotor CoG (Figure 4.7). Moreover, the control algorithm shows robustness with respect to the external disturbance forces and moments exerted by a suspended load while dynamic change in the quadrotor CoG occurs (Figure 5.16).

4.3 Conclusion

In this chapter we analyzed the influence of the center of gravity of the flying robot to the overall system stability. To the best of our knowledge the present literature reports analysis in the frequency domain presented in [57]. In this chapter we move a step further and besides the analysis we design an adaptive controller which is able to stabilize the system and ensure trajectory tracking despite the changes in the center of gravity. For future work one can investigate the region of attraction of the proposed adaptive controller and propose an algorithm that would improve the performance of the current one.

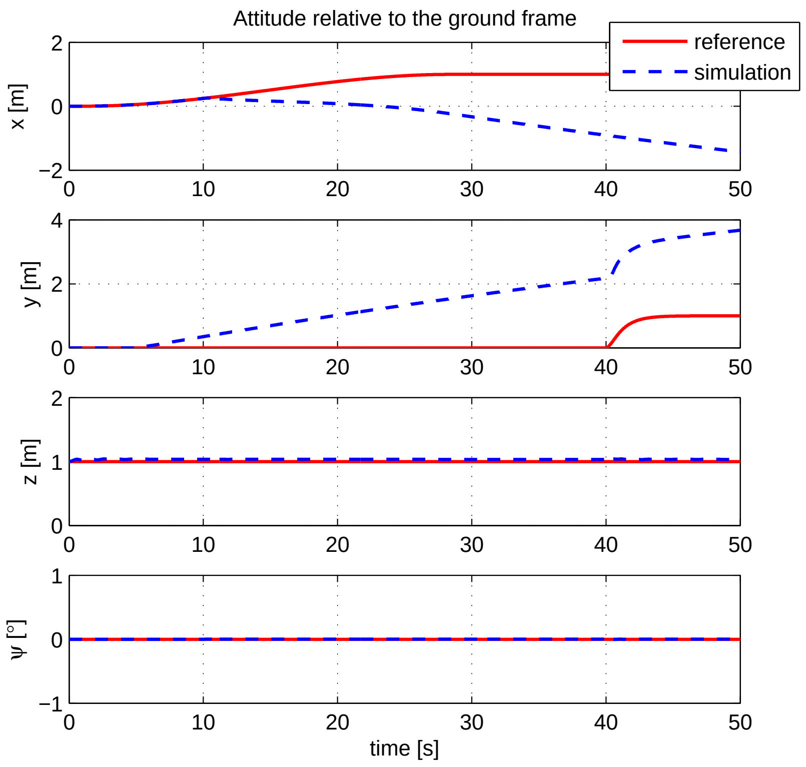


(a) Attitude

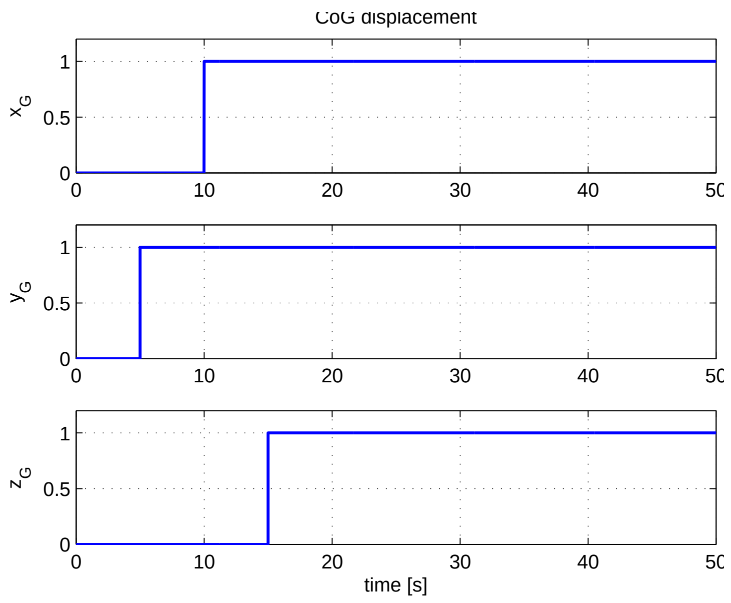


(b) Coordinates of changed CoG

Figure 4.4: Failure of the linear output feedback control algorithm to stabilize the quadrotor due to changes in CoG.

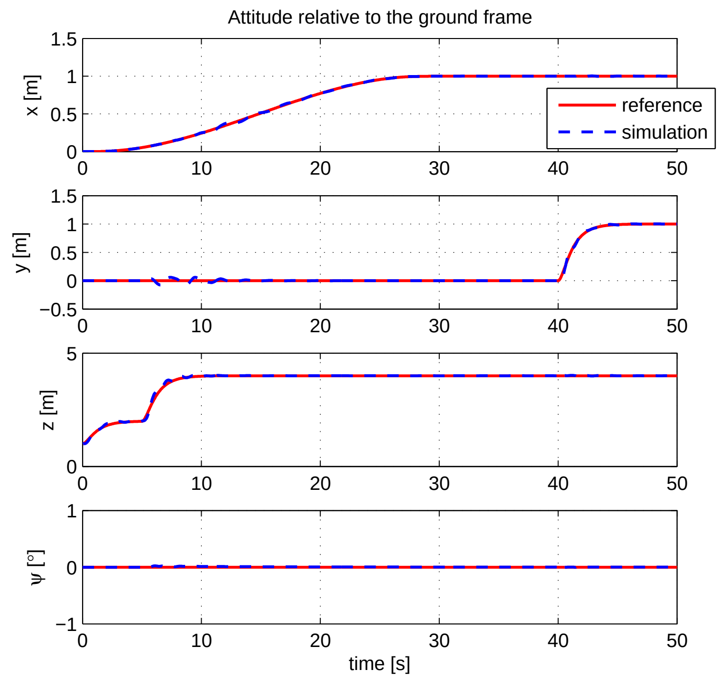


(a) Attitude

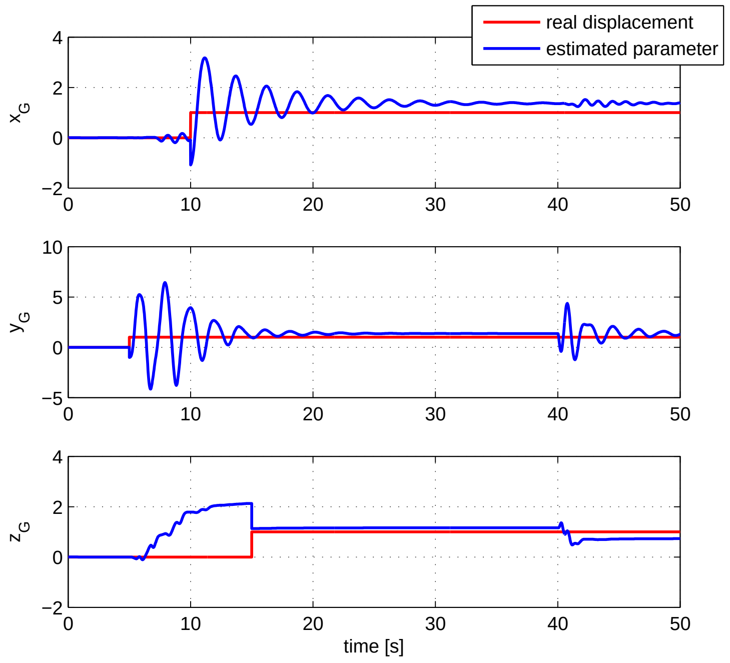


(b) Changed coordinates of CoG

Figure 4.5: Failure of the feedback linearization algorithm to stabilize the quadrotor due to changes in CoG.

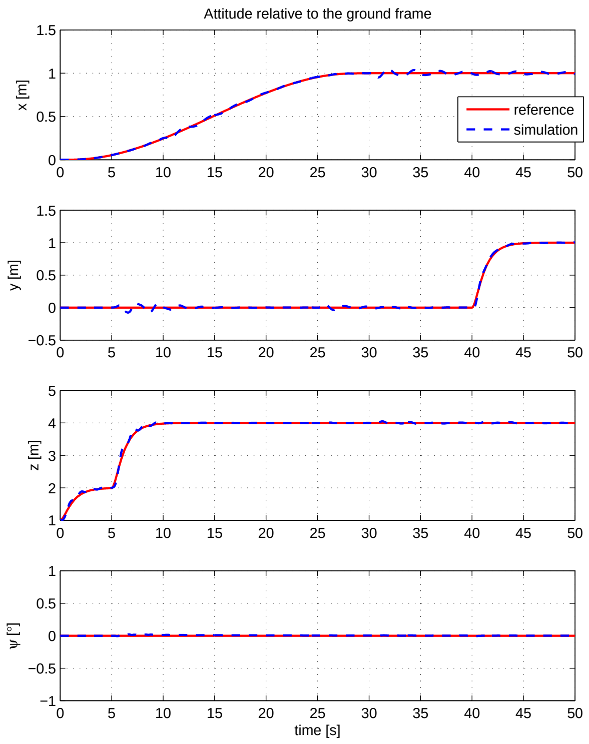


(a) Attitude

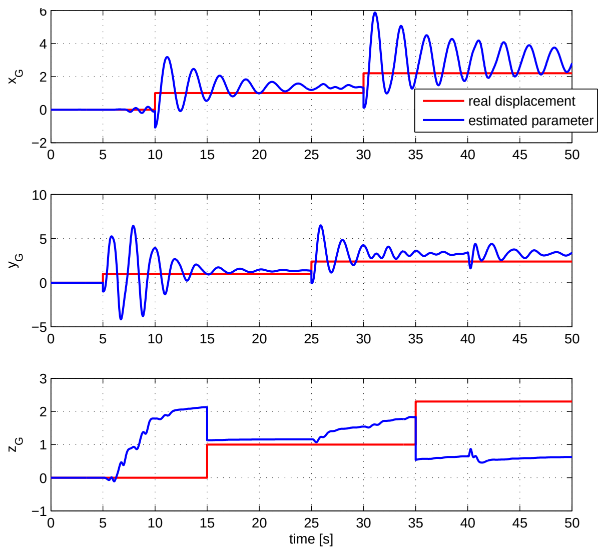


(b) Real and estimated coordinates of CoG

Figure 4.6: Performance of the adaptive controller used for stabilization of change in CoG.

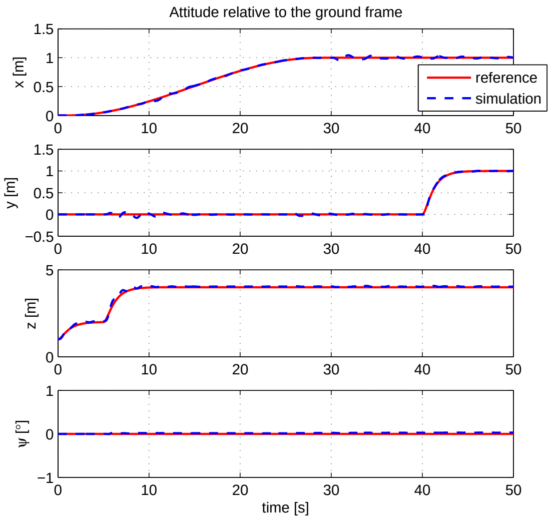


(a) Attitude

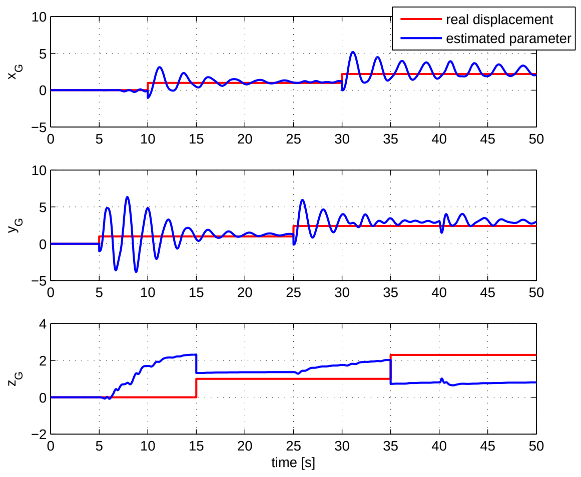


(b) Changed coordinates of CoG and estimated parameters

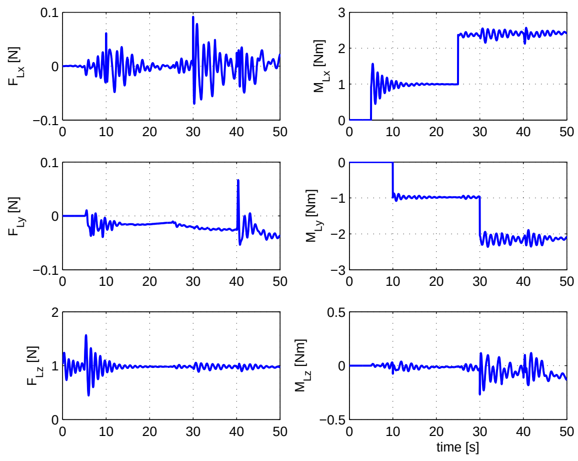
Figure 4.7: Adaptive algorithm used for tracking while compensating for dynamic change in CoG.



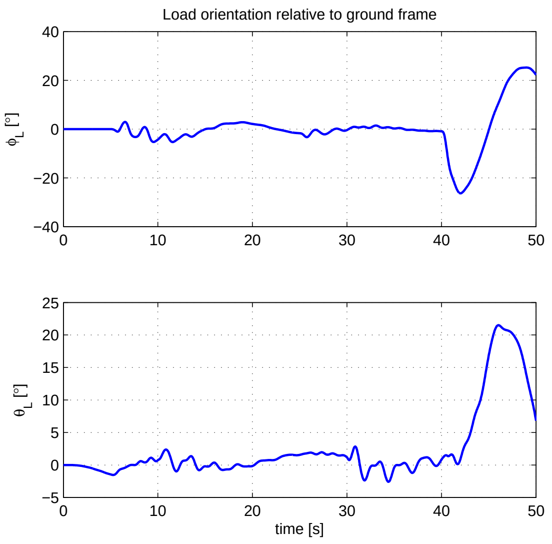
(a) Attitude



(b) Estimated and real CoG parameters



(c) Forces and moments from suspended load



(d) Load Orientation

Figure 4.8: Robustness of adaptive algorithm used for tracking while compensating for dynamic change in CoG.

Chapter 5

Trajectory generation for swing-free load transportation

Transport of suspended objects using a robot or crane is a common application. At the end of a transport motion, the suspended object naturally continues to swing. Suppression of this residual oscillation has been a topic of research for many years. Both open- and closed-loop strategies have been explored, in this chapter we focus on an open-loop technique presented in [58] and [59] applied for quadrotor carrying a suspended load Figure 5.3. As a closed-loop technique we present a learning algorithm based on Nelder-Mead algorithm used for solving nonlinear unconstrained optimization problem.

5.1 Trajectory generation using input shaping

Input shaping is a control technique for reducing vibrations. The method works by creating an input signal that cancels its own vibration. That is, vibration caused by the first part of the command signal is canceled by vibration caused by the second part of the command. Input shaping is implemented by convolving a sequence of impulses, an input shaper, with any desired input, in our case we are convolving the initial cubic trajectory as shown on Figure 5.1. The shaped trajectory obtained by the convolution is used as an input trajectory for the quadrotor with suspended load. The amplitudes and time locations of the impulses are obtained from the system's natural frequencies and damping ratios. One of the main disadvantages of the impulse-convolution method is that it can produce trajectories that are non-smooth to the point of being unusable for implementation on the real system. Furthermore we can see from Figure 5.2 that input shaping is not robust to the change in the length of the suspension cable.

5.2 Swing-free trajectory generation using DP

Dynamic programming (DP) is founded on the principle of optimality [60]. An optimal sequence of decisions has the property that whatever the initial state and decision, the remaining decisions must constitute an optimal sequence of decisions for the remaining problem [28]. The subsequent procedure follows the approach of [58] and [61] and outlines the method of applying DP to a discrete time piecewise linear system. We begin with the general form of the discrete time system:

$$q_{k+1} = A_k q_k + B_k u_k \tag{5.1}$$

Note that system and input matrices A_k and B_k can be time varying. Given initial state q_0, we would like to find the optimal sequence of inputs that will minimize the scalar objective function:

$$\Gamma(q, u) = \sum_{k=1}^{N} \Gamma_k(q_k, u_k) \tag{5.2}$$

This objective function can be structured as quadratic in the input u_k and state q_k as shown in equation 5.3, where γ_k, y_k, z_k, Q_k, R_k, and S_k are time varying coefficients.

$$\Gamma_k \;=\; \gamma_k + q_k^T y_k + q_k^T y_k + +\frac{1}{2}[q_k^T Q_k q_k + 2q_k^T R_k u_k + u_k^T S_k u_k]$$

The foundation of DP is the optimal value function. Beginning from any point i, the optimal sequence can be computed recursively backward to $i = 1$. The form of the optimal value function is thereby

$$\Lambda_i = \min_{(u_i,\ldots,u_N)} \sum_{k=i}^{N} \Gamma_k(q_k, u_k) \tag{5.3}$$

Since the objective function is quadratic in form, the optimal value function will also be defined as a quadratic, where ζ_i, ν_i, and W_i are the coefficients

$$\Lambda_i(q_i) = \zeta_i + q_i^T \nu_i + \frac{1}{2} q_i^T W_i q_i \tag{5.4}$$

Using Bellmans principle of optimality, the backward recursive relation can now be formed.

$$\Lambda_i = \min_{u_i}(\Gamma_i + \Lambda_{i+1}) \tag{5.5}$$

Substituting (5.3) and (5.3) into (5.5) yields the recursive relation

$$\begin{aligned}\zeta_i + q_i^T \nu_i + \frac{1}{2} q_i^T W_i q_i = \min_{u_i}\{&\zeta_{i+1} + q_{i+1}^T \nu_{i+1} + \frac{1}{2} q_{i+1}^T W_{i+1} q_{i+1} + \gamma_i + q_i^T y_i \\ &+u_i^T z_i + \frac{1}{2}[q_i^T Q_i q_i + 2q_i^T R_i u_i + u_i^T S_i u_i]\}\end{aligned} \tag{5.6}$$

Substituting (5.1) into (5.7) and simplifying it we get

$$\begin{aligned}\zeta_i + q_i^T \nu_i + \frac{1}{2} q_i^T W_i q_i = \min_{u_i}\{&\zeta_{i+1} + \gamma_i + q_i^T h_{4i} + \\ u_i^T h_{5i} + \frac{1}{2}&[q_i^T H_{1i} q_i + 2q_i^T H_{2i} u_i + u_i^T H_{3i} u_i]\}\end{aligned}$$

where

$$\begin{aligned}
H_{1i} &= Q_i + A_i^T W_{i+1} A_i,\\
H_{2i} &= R_i + A_i^T W_{i+1} B_i,\\
H_{3i} &= S_i + B_i^T W_{i+1} B_i,\\
h_{4i} &= y_i + A_i^T v_{i+1},\\
h_{5i} &= z_i + B_i^T v_{i+1}.
\end{aligned} \tag{5.7}$$

Differentiating the right hand side of (5.7) with respect to u_i, then equating to zero results in

$$u_i = -H_{3i}^{-1}[H_{3i}q_i + h_{5i}]. \tag{5.8}$$

Substituting this solution back into (5.7) and equating terms of like degree in q_i results in the following recursive equations:

$$\begin{aligned}
\zeta_i &= \zeta_{i+1} + \gamma_i - \frac{1}{2}h_{5i}^T H_{3i}^{-1} h_{5i}\\
\nu_i &= h_{4i} - H_{2i}H_{3i}^{-1}h_{5i}\\
W_i &= H_{1i} - H_{2i}H_{3i}^{-1}H_{2i}^T
\end{aligned} \tag{5.9}$$

The initial values for (5.10) are given by

$$\begin{aligned}
\zeta_N &= \gamma_N,\\
\nu_N &= y_N,\\
W_N &= Q_N.
\end{aligned} \tag{5.10}$$

The procedure for applying this algorithm is as follows:

1. Determine v_N and W_N using (5.11). As will be shown, y_N and Q_N can be extracted directly from the objective function.

2. Calculate v_i and W_i recursively for $i = N-1$ to $i = 1$ using (5.10), while storing matrices H_{3i}^{-1}, H_{2i}^{T} , and H_{3i}^{-1}, h_{5i} in the process.

3. Calculate u_i and q_i recursively for $i = 1$ to $i = N - 1$ using (5.8) and (5.1), respectively.

The simplest candidate for the objective function follows a minimum energy principle. Using trajectory accelerations as the input, rather than jerk, we improve computational efficiency while still allowing a zero end constraint for the trajectory velocities. The state and input of the system are consequently given by

$$\begin{aligned} q &= \begin{bmatrix} \eta_L & \nu_L & \eta & \nu \end{bmatrix}^T, \\ u &= \dot{\nu} \end{aligned} \tag{5.11}$$

where η_L and ν_L are load displacement angles and angular velocities, and η, ν and $\dot{\nu}$ are quadrotor attitude, velocity and acceleration vectors. Since the quadrotor exhibits high performance trajectory tracking (Figure 5.4(a), Figure 5.4(b) and Figure 5.4(c)), for computational efficiency of the DP algorithm, dynamic model of the quadrotor is approximated by the corresponding kinematic model. To suppress the residual oscillations, a penalty weight must be introduced into the objective function. In [59] and [58], the value of penalty weight p is found empirically. The weighted terms are chosen to represent the sum of squares of the final state error. In this paper we determined the weighted terms using optimization solver by minimizing the rate of convergence of the DP algorithm. This results in the objective function

$$\Gamma_k = \frac{1}{2}[q_k^T Q_k q_k + 2q_k^T R_k u_k + u_k^T S_k u_k] + \frac{1}{2}p\,[x_F - x_N]^T\,[x_F - x_N]\,. \tag{5.12}$$

Matching like terms with those of the general form of the objective function leads to the following observations

$$\begin{aligned} \zeta_N &= \frac{1}{2} p x_F^T x_F, \\ y_N &= -p x_F, \\ z_N &= 0, \\ Q_N = Q_k &= q_p I_n, \\ R_N &= R_k, \\ S_N = S_k &= I_n, \end{aligned} \tag{5.13}$$

in which I is an identity matrix and n is the number of state variables. The optimal weighted vectors are found to be

$$\begin{aligned} p &= \begin{bmatrix} 99.69 & 99.69 & 101.68 & 101.68 & \mathbf{1}_{1\times12} \end{bmatrix}, \\ qq &= \begin{bmatrix} -9e^{-4} & 2e^{-3} & 9e^{-4} & 2e^{-3} & \mathbf{1}_{1\times12} \end{bmatrix}, \\ R_k &= \begin{bmatrix} R_{k42} & \mathbf{O}_{4\times4} \\ \mathbf{O}_{12\times2} & \mathbf{O}_{12\times4} \end{bmatrix}, \\ R_{k42} &= \begin{bmatrix} 0 & 0.3 \\ 0 & 0.3 \\ 0.3 & 0 \\ 0.3 & 0 \end{bmatrix}, \end{aligned} \tag{5.14}$$

where $p \in \mathbb{R}^{1\times16}$, $qq \in \mathbb{R}^{1\times16}$ and $R_k \in \mathbb{R}^{16\times6}$.

5.2.1 Simulation Results

The algorithm described in the previous section requires the beginning and ending states to be known prior to the optimization. An initial trajectory estimate is also required for the first optimization pass in order to compute required A_k and B_k matrices. In this paper, we used cubic polynomial position trajectories (Fig. 5.4) for the initial simulation of the system used in the first optimization pass. The cubic trajectories result in residual oscillations of approximately 6.1° for swing (Figure 5.5(b)), and 6.15° for rock Figure 5.5(a). Obtained optimal trajectories suppress the residual oscillations to less than 5% of the initial oscillation magnitudes. The algorithm requires three passes before reaching the convergence with computation time of 3.4s. Figure 5.6 and Figure 5.7 show the case of generation of optimal trajectories for multiple waypoints. Given the desired waypoints the optimal trajectory is generated for every waypoint separately suppressing the residual oscillations for each waypoint (Figure 5.7(b)).

The optimal trajectories obtained using dynamic programming are sufficiently smooth, see Figure 5.9, and are suitable for implementation on the real system. By sufficiently smooth we mean that at least their fourth derivative, which is jerk in our case, is a continuous function. One of the drawback of the dynamic programming is that this method is model based and therefore is sensitive to model uncertainties. We can see in Figure 5.10 that this method is not robust with respect to changes in the length of the suspension cable.

5.2.2 Experimental results

The effectiveness of the methodology proposed in this paper is verified experimentally.

First, we would like to show show robustness of the proposed method with respect to unmodeled actuator dynamics, noise and system delays. In our case the actuator is a quadrotor whose model and attitude control design are presented in Sec. 2.1. Since the quadrotor shows almost perfect trajectory tracking (Figure 5.4(a), Figure 5.4(b) and Figure 5.4(c)) in the DP algorithm, we assume perfect tracking, i.e., the full nonlinear quadrotor model is replaced by a double integrator. The performance of the swing-free trajectory tracking in an ideal case is shown in Figure 5.11(a). However, on Figure 5.4(c) we can see that $\ddot{e} \neq 0$ where $\ddot{e}$ represents the acceleration tracking error of a quadrotor. Therefore, in the case when we use the full nonlinear model with an attitude controller for swing-free trajectory tracking, we can see that the performance is different than in the ideal case (Figure 5.11(b)) but it is still satisfying. By implementing the proposed method on an experimental system (Figure 5.11(c)), we see that the performance deteriorates due to imperfect tracking. On the other hand, the attenuation of the load displacement angles ϕ_L and θ_L is achieved as shown in Figure 5.12(a) and Figure 5.12(a). Therefore, the proposed method is robust enough and shows good performance even with the lack of perfect trajectory tracking. Videos of simulations and experiments can be found at [62]. Furthermore in the second set of experiments we tried to emulate swing-free trajectory tracking in urban environments motivated in the first section by simulations presented in Figure 5.8. We build a maze of obstacles shown in Figure 5.13. The quadrotor is flying above the obstacles while carrying the suspended load through narrow corridors. First, quadrotor was tracking an initial 3D trajectory with cubic profile with respect to time. Then using the dynamic programming based algorithm we have computed a 3D trajectory with an optimal swing-free profile with respect to time. The experimental data including quadrotor position and load position are shown in Figure 5.13.

Since it is hard to analyze raw experimental data just by visual inspection, we utilize the tools from signal processing, i.e., we computed power spectral density for each signal. Before processing, from each raw signal we extracted the part which corresponds to the swing-free behavior. Power spectral density, describes how the power of a signal or a time series is distributed with frequency, see Figure 5.13(c). By taking the integral of the power spectral density we compute the total power for each signal. We repeated the experiment in four trials and analyzed them using power spectral density. The data are presented in Table 5.1. In all four cases the data

Table 5.1: Total power P_{tot} of the load displacement signals - experimental results

	Signal [°]	$\mathbf{P_{tot}}$ **for Init. Traj. [W]**	$\mathbf{P_{tot}}$ **for Opt. Traj. [W]**
trial #1	ϕ_L	31.0634	26.7398
	θ_L	36.0171	26.3692
trial #2	ϕ_L	28.3503	26.2140
	θ_L	39.4729	30.3817
trial #3	ϕ_L	34.3385	21.5736
	θ_L	34.0470	28.9020
trial #4	ϕ_L	38.8807	28.5435
	θ_L	35.4629	33.8539

show less energy for the load displacement while tracking the optimal trajectory.

Videos of simulations and experiments can be found at [62].

5.3 Swing-free trajectory generation using Taylor series

5.3.1 Function approximation using Taylor series

Function approximation using using a finite number of terms of its Taylor series is a common practice. Taylor's theorem gives quantitative estimates on the error in this approximation.

Theorem 5.3.1 (Taylor's Theorem)**.** *Let $k \geq 1$ be an integer and let the function $f : \mathbb{R} \to \mathbb{R}$ be k times differentiable at the point $a \in \mathbb{R}$. Then there exists a function $h_k : \mathbb{R} \to \mathbb{R}$ such that*

$$f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \cdots + \frac{f^k(a)}{k!}(x-a)^k$$

and $\lim_{x \to a} h_k(x) = 0$

Any finite number of initial terms of the Taylor series of a function is called a Taylor polynomial. The Taylor series of a function is the limit of that function's Taylor polynomials, provided that the limit exists. A function may not be equal to its Taylor series, even if its Taylor series converges at every point. A function that is equal to its Taylor series in an open interval (or a disc in the complex plane) is known as an analytic function.

Taylor's theorem gives an approximation of a k times differentiable function around a given point by a k-th order Taylor-polynomial. For analytic functions the Taylor polynomials at a given point are finite order truncations of its Taylor's series, which completely determines the function in some neighborhood of the point.

5.3.2 Nelder - Mead algorithm

The Nelder-Mead algorithm or simplex search algorithm [20], is one of the best known algorithms for multidimensional unconstrained optimization without derivatives. The basic algorithm is quite simple to understand and very easy to use. For these reasons, it is very popular in many fields of science and technology, especially in chemistry and medicine.

The method does not require any derivative information, which makes it suitable for problems with non-smooth functions. It is widely used to solve parameter estimation and similar statistical problems, where the function values are uncertain or subject to noise. It can also be used for problems with discontinuous functions, which occur frequently in statistics and experimental mathematics.

5.3.3 Learning the swing-free trajectory parameters

In this work we are using Taylor polynomial of 10th order to determine the trajectory given a certain task. By a certain task we mean two distinct actions that we want to accomplish using quadrotor carrying the suspended load. First, is a swing-free maneuver defined in Section 1.2.2 for which we are trying to find the trajectory previously generated by dynamic programming, model based method presented in Section 5.2. Since we don't know the optimal trajectory that will ensure a swing-free maneuver of the load, we define Taylor polynomial with unknown coefficients that will approximate the optimal swing-free trajectory. The unknown coefficients we learn using Nelder-Mead algorithm which is one of the bet method for solving nonlinear multidimensional optimization problems. Since Nelder-Mead algorithm is unconstrained optimization solver and swing-free optimization does not have one global minimum we are defining the set of constraint for this algorithm in order to to ensure the performance we want. By not having one global minimum we mean that for example since the solver is unconstrained it will give us as the best solution trajectory that does not move form the initial point, where the velocity and acceleration of the quadrotor are zero at all times.

Since we want to perform a swing-free maneuver while quadrotor moves from point $\eta_{10} = [X0 \quad Y0 \quad Z0]^T$ to the point $\eta_{1F} = [XF \quad YF \quad ZF]^T$ in T seconds we have to define a set of constraints that will transform our unconstrained Nelder-Mead algorithm into a constrained version of it. In Figure 5.14 we depict the constraints we impose to the unknown swing-free trajectory. If we write down the constraints we get the following. Function that represents the position in x-axis is defined as

$$f_x(t) = f_{x0} + f_{x1}t + \frac{f_{x2}}{2!}t^2 + \cdots + \frac{f_{x10}}{10!}t^{10} \tag{5.15}$$

where $t \in \mathbb{R}$ represents time. Now we impose the constraints

$$\begin{aligned} f_x(0) &= X0 \quad \Rightarrow \quad f_{x0} = X0 \\ f_x(T_X) &= XF \quad \Rightarrow \quad f_{x0} + f_{x1} + \cdots + f_{x10} = XF \end{aligned} \tag{5.16}$$

where $X0$ and XF represent the initial and the final point in the x-axis respectively, and T_x represents the time to reach the final point.

Differentiating (5.15) with respect to time we get the function that represents the velocity

$$\dot{f}_x(t) = f_{x1} + \frac{2 * f_{x2}}{2!} t + \cdots + \frac{10 * f_{x10}}{10!} t^9. \tag{5.17}$$

The constraints on the velocity are

$$\begin{aligned} \dot{f}_x(0) &= 0 \quad \Rightarrow \quad f_{x1} = 0 \\ \dot{f}_x(T_X) &= 0 \quad \Rightarrow \quad 2 * f_{x2} + 3 * f_{x3} + \cdots + 10 * f_{x10} = 0 \\ \dot{f}_x(t) &> 0, \quad \forall t \in [0 \quad T]. \end{aligned} \tag{5.18}$$

Differentiating (5.17) with respect to time we get the function that represents the acceleration

$$\ddot{f}_x(t) = \frac{2 * f_{x2}}{2!} + \cdots + \frac{10 * 9 * f_{x10}}{10!} t^8. \tag{5.19}$$

The constraints on the acceleration are

$$\begin{aligned} \ddot{f}_x(0) &= 0 \quad \Rightarrow \quad f_{x2} = 0 \\ \ddot{f}_x(T_X) &= 0 \quad \Rightarrow \quad 3 * 2 * f_{x3} + \cdots + 10 * 9 * f_{x10} = 0. \end{aligned} \tag{5.20}$$

Solving (5.17), (5.19) and (5.21) we get the initial coefficients with which we start the learning process. With these constraints we are ensuring that the approximated trajectory will be feasible for the quadrotor to accomplish. The value function we are minimizing is given by

$$V = \eta_{LF} Q \eta_{LF}^T, \tag{5.21}$$

where η_{LF} is the vector of load displacement angles and angular velocities at the end of the maneuver, at time T, and $Q > 0, Q \in \mathbb{R}^4$ is a weighting matrix. Constraints defined above leave a very small "window" for the solver to find the parameters f_{xi}. Therefore, we define the penalty function as

$$F = P * \left| \left[\begin{array}{cccccc} f_x(0) & f_x(T) - XF & \dot{f}_x(0) & \dot{f}_x(T) & \ddot{f}_x(0) & \ddot{f}_x(T) \end{array} \right] \right|_1 \tag{5.22}$$

where $P > 0, P \in \mathbb{R}$ is a large positive number and $|\cdot|_1$ is a vector 1-norm. The reason why we chose this type of penalty function is that this way solver knows if the result he is obtaining is converging or not. Otherwise by putting only a constant as a penalty function the solver has no way of knowing if he is searching in the right direction or not.

The method presented here is depicted by a scheme on Fig 5.15. This is a iterative, model-free method. What the algorithm presented here is obtaining from the system is the measurements of the load displacement angles and angular rates. Then the algorithm gives back the parameters for the trajectory defined by (5.15) by minimizing (5.21). Forces and torques exerted by the suspended load decrease when the load displacement angles are decreased as shown in Figure 5.16(c) and Figure 5.16(d).

5.4 Conclusion

In this chapter we present three different methods to obtain swing-free motion of the quadrotor carrying the suspended load. All three methods show promising results.

Input shaping and dynamic programming are model based method so they require a prior knowledge of the system. The third method is model free and requires only measurements of the displacement angles.

By combining adaptive control and swing-free trajectory generation we develop a hierarchical controller that ensures robustness of the quadrotor baseline attitude controller with respect to model uncertainties (change in the CoG of the quadrotor) and to external forces $\mathbf{F_H}$ and torques $\mathbf{T_H}$ exerted by the suspended load. Besides that, the system is able to perform agile swing-free maneuvers which can be useful in a wide range of applications.

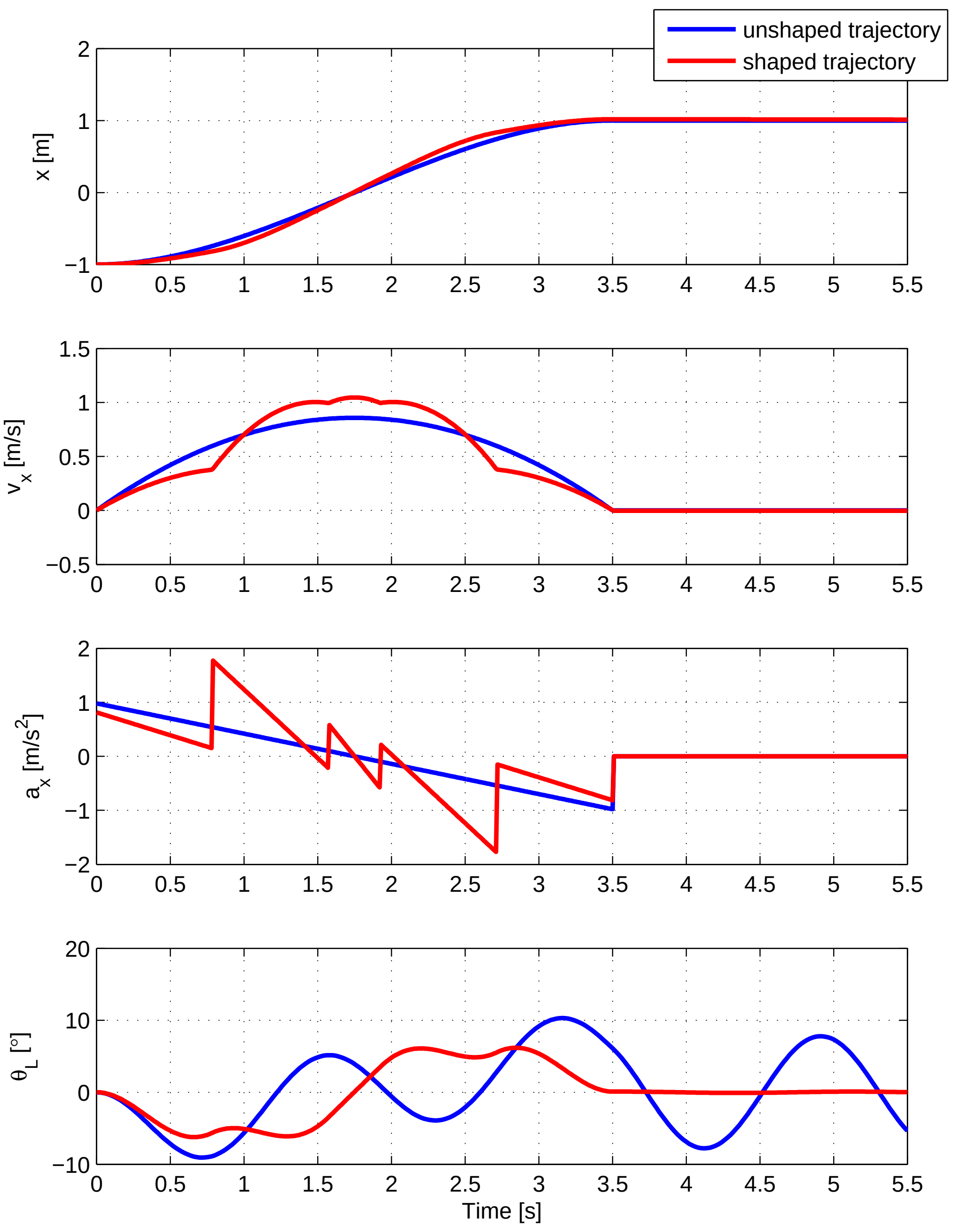


Figure 5.1: Swing free using input shaping.

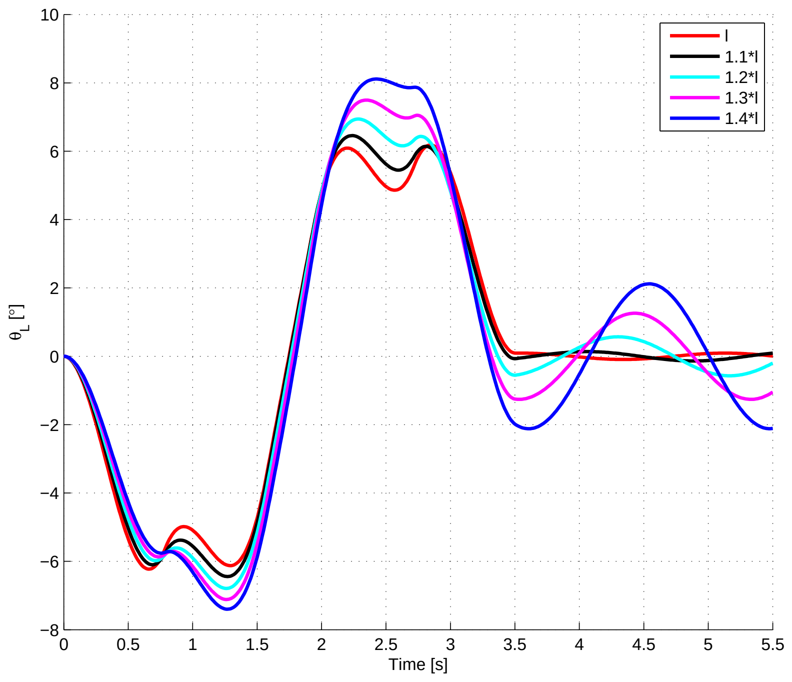
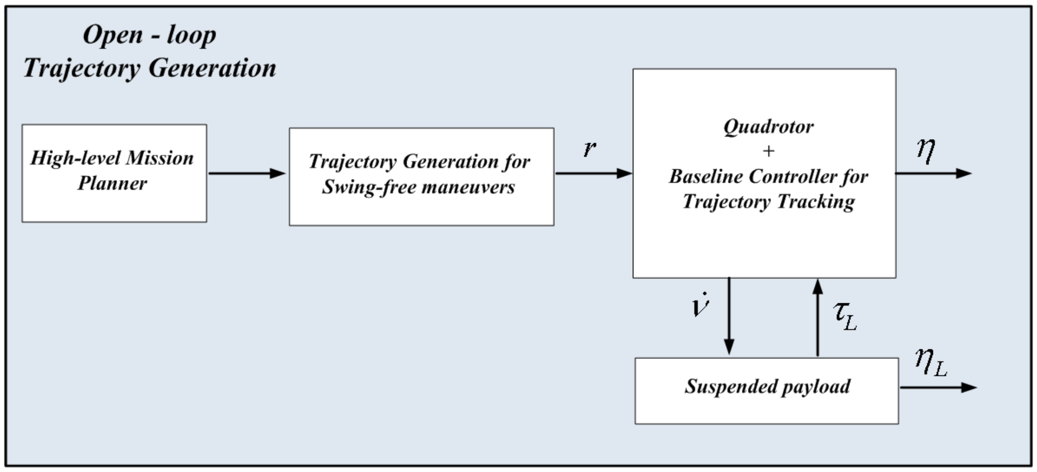
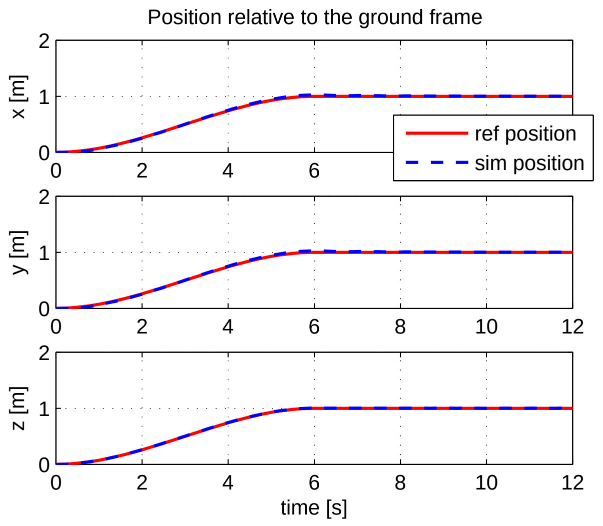


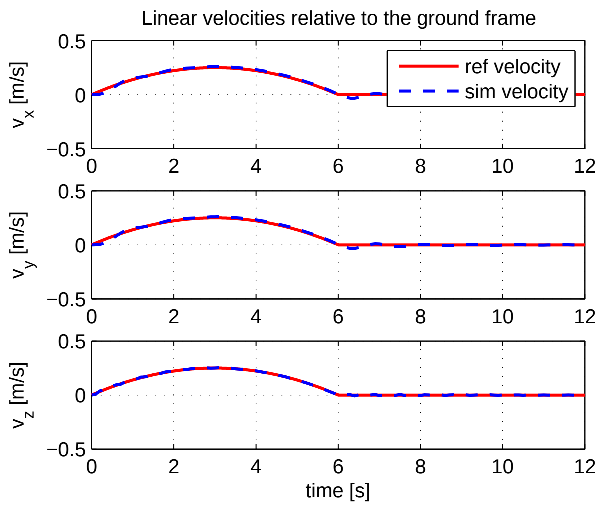
Figure 5.2: Robustness of input shaping with respect to unknown length of the load.



Open - loop
Trajectory Generation
High-level Mission Planner
Trajectory Generation for Swing-free maneuvers
r
Quadrotor + Baseline Controller for Trajectory Tracking
η
ν̇
τ_L
η_L
Suspended payload

Figure 5.3: Controller scheme with DP based trajectory builder.





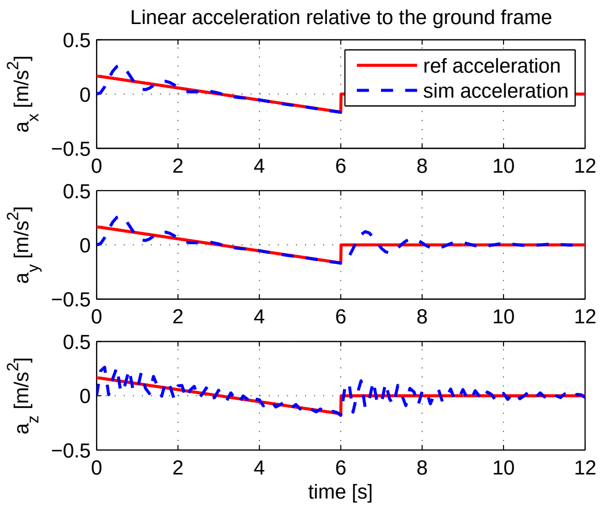
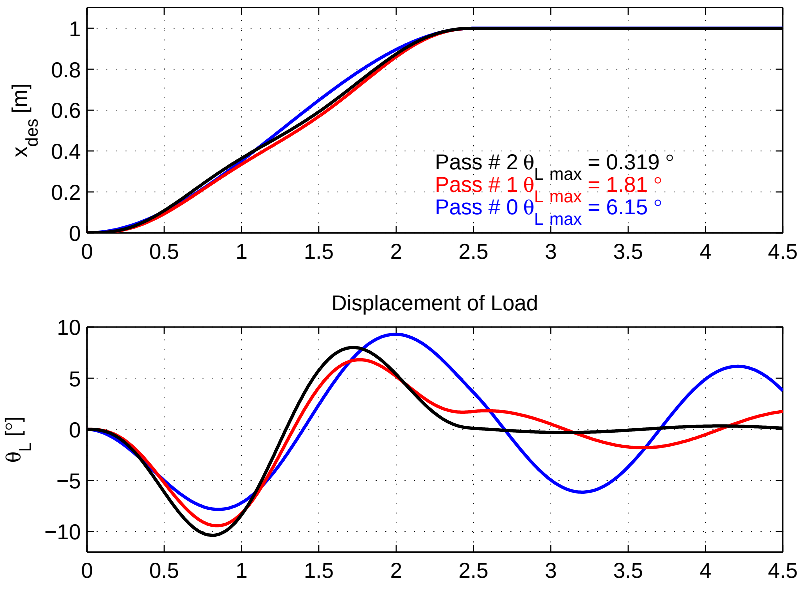
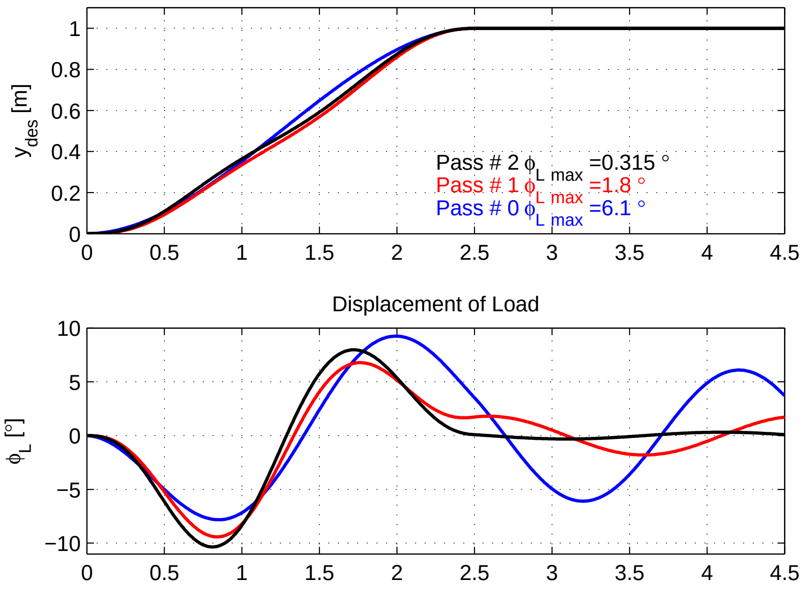


Figure 5.4: Quadrotor trajectory tracking - simulation results.

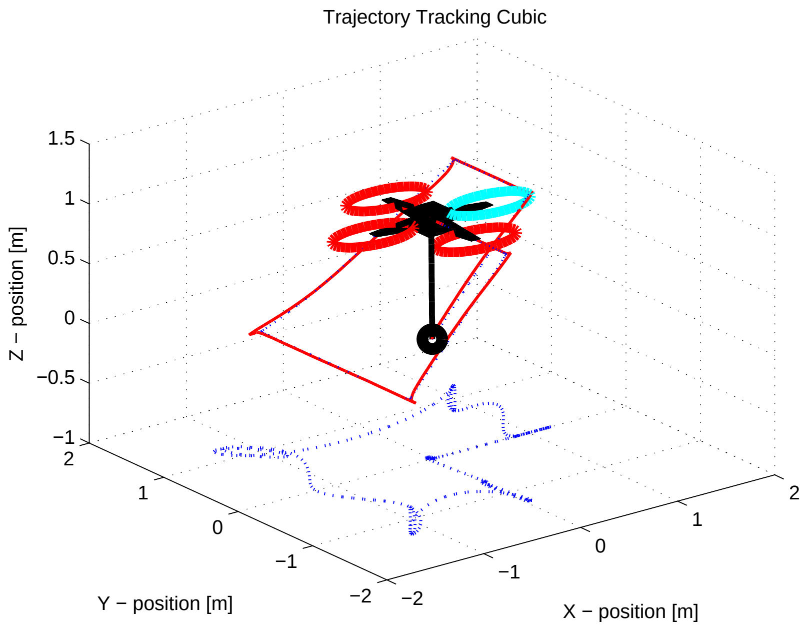


(a) X trajectory and swing

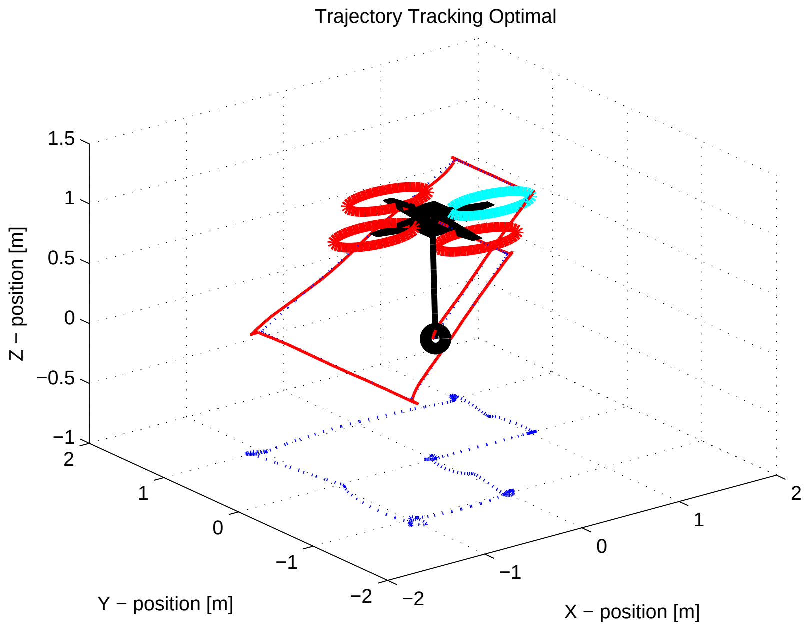


(b) Y trajectory and rock

Figure 5.5: Optimal and cubic trajectories with load displacement considering one waypoint

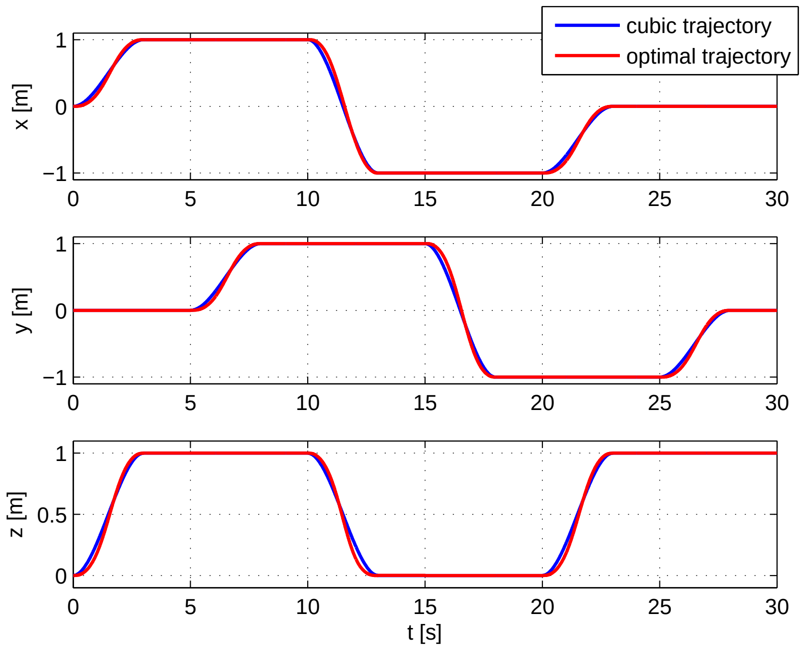


(a) Cubic trajectory

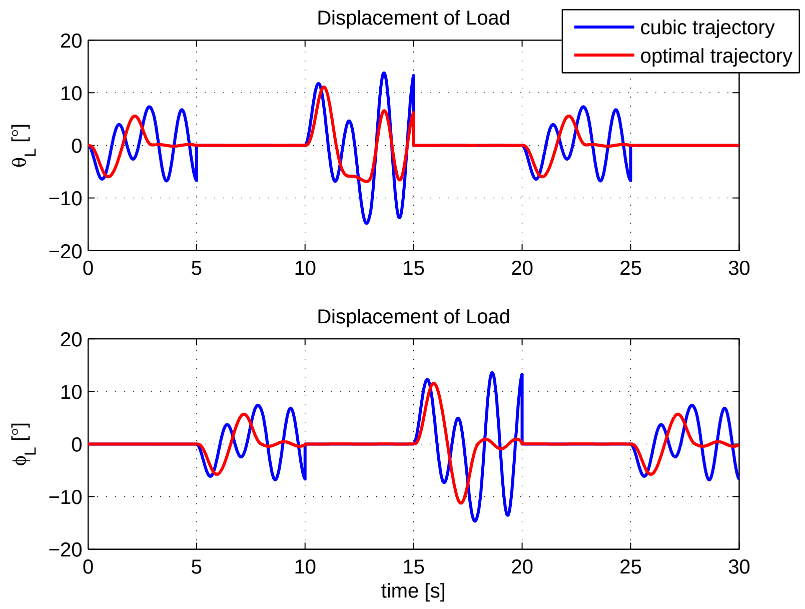


(b) Optimal trajectory

Figure 5.6: 3D representation of trajectories considering multiple waypoints.



(a) Trajectories



(b) Angular displacement of the Load

Figure 5.7: Optimal and cubic trajectories with load displacement considering multiple waypoints.

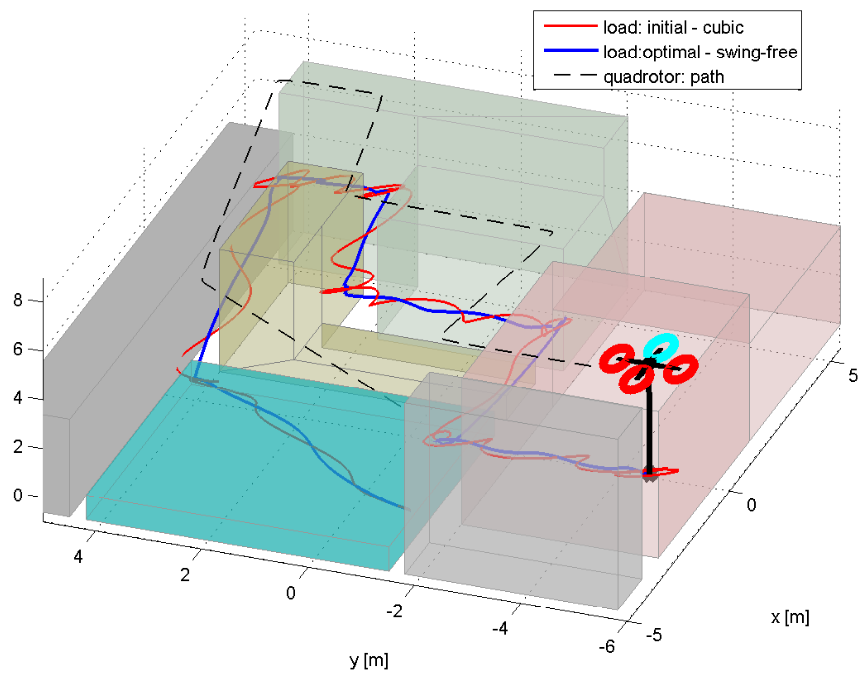


Figure 5.8: 3D representation of trajectories considering multiple waypoints with obstacles.

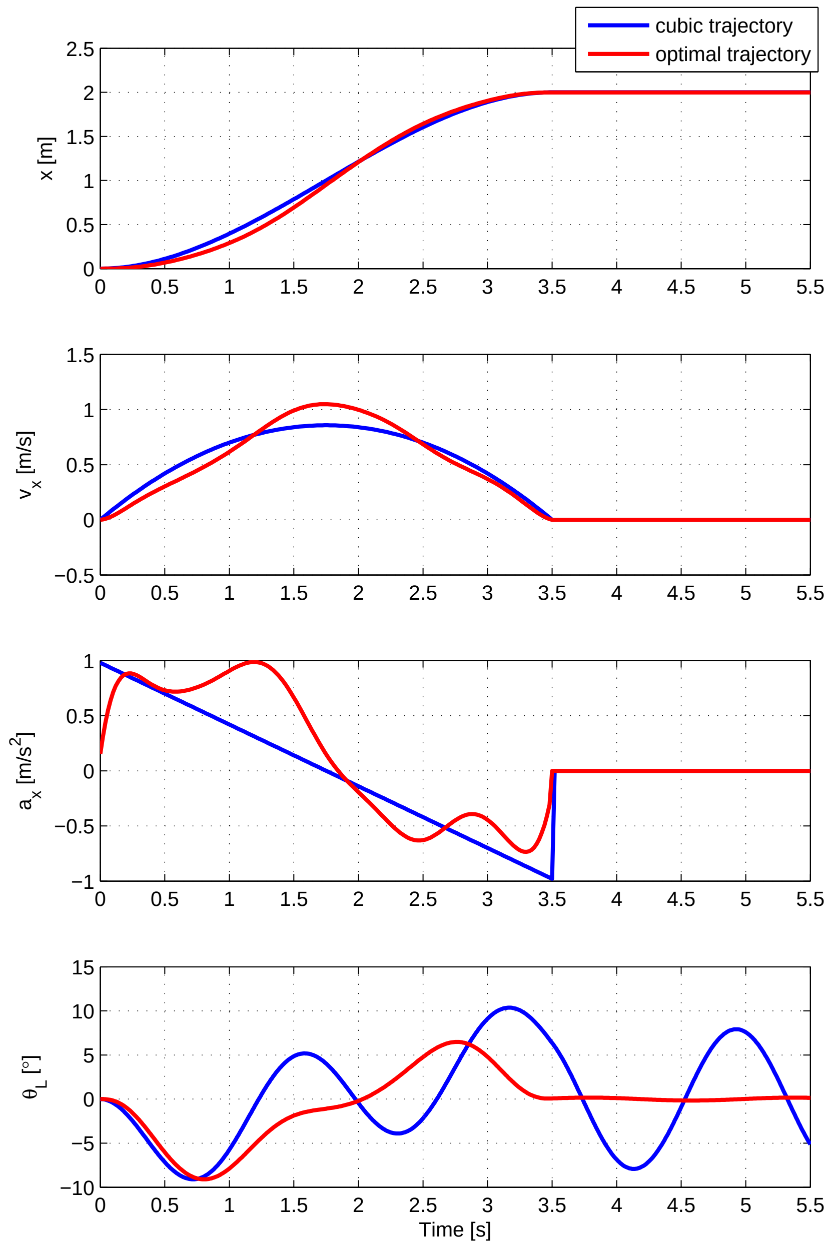


Figure 5.9: Swing free using dynamic programming.

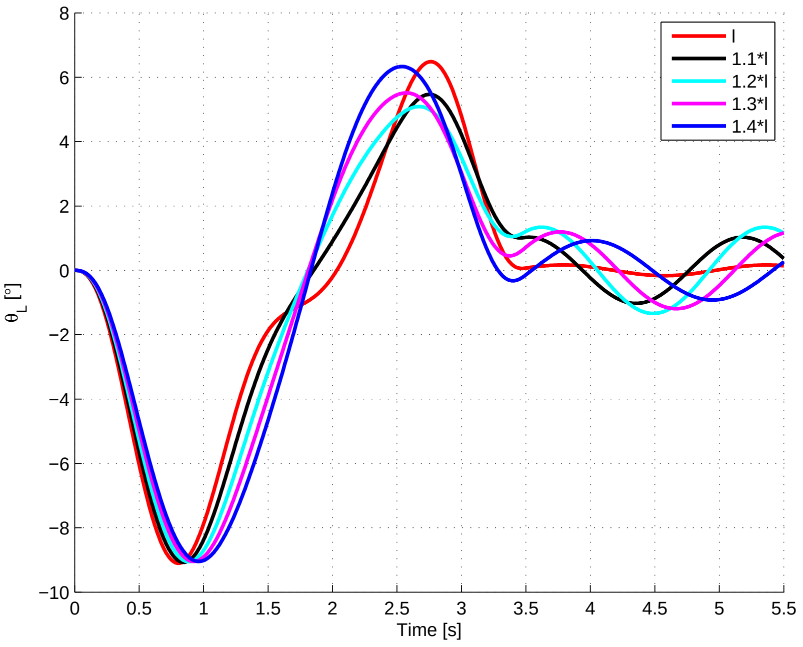
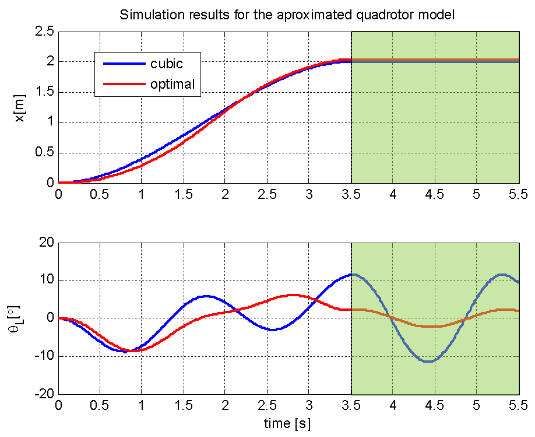
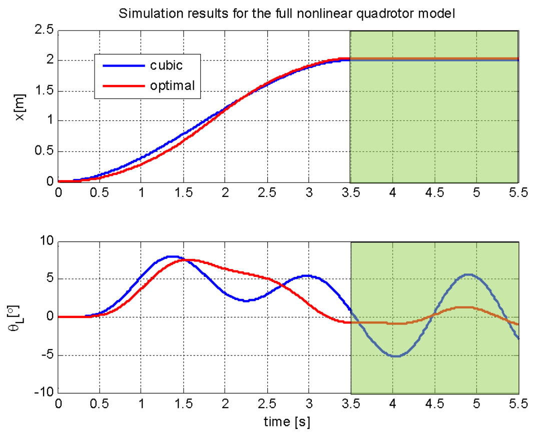


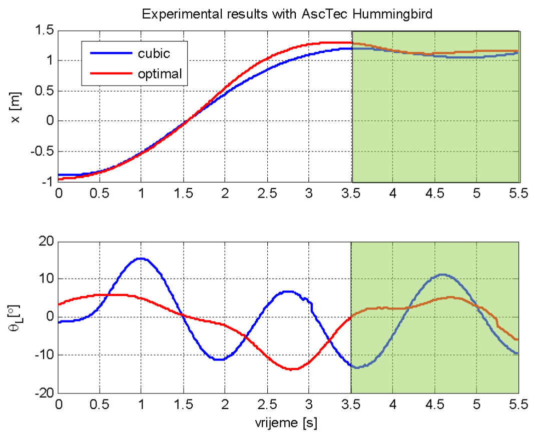
Figure 5.10: Robustness of dynamic programming with respect to unknown length of the load.



(a) Kinematic model of the quadrotor

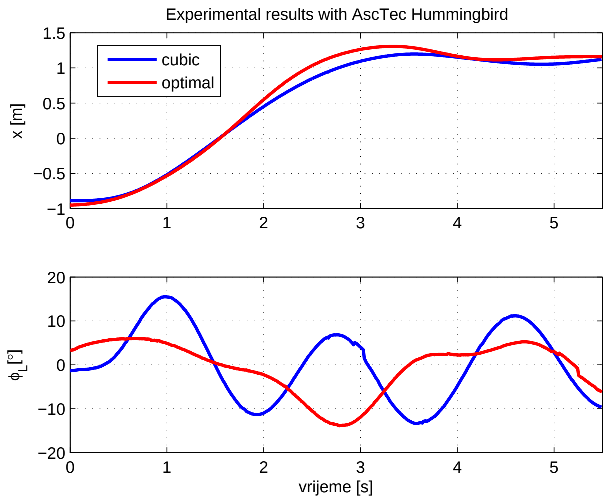


(b) Nonlinear model of the quadrotor with attitude controller

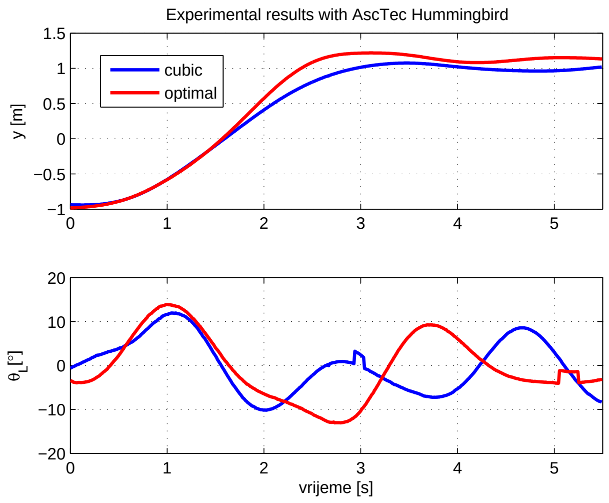


(c) Experimental quadrotor

Figure 5.11: Robustness of the proposed method considering the unmodeled dynamics of the quadrotor

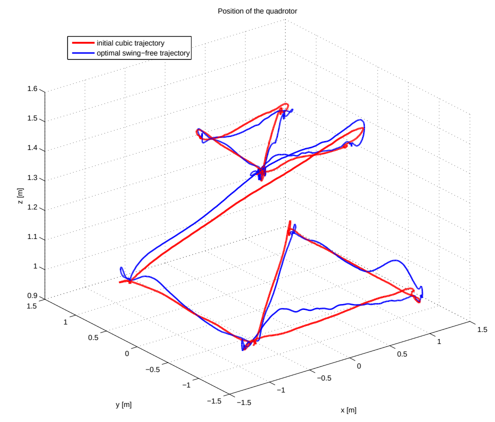


(a) X trajectory and swing

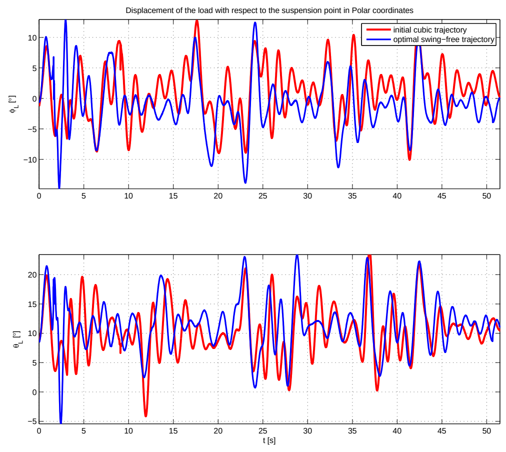


(b) Y trajectory and rock

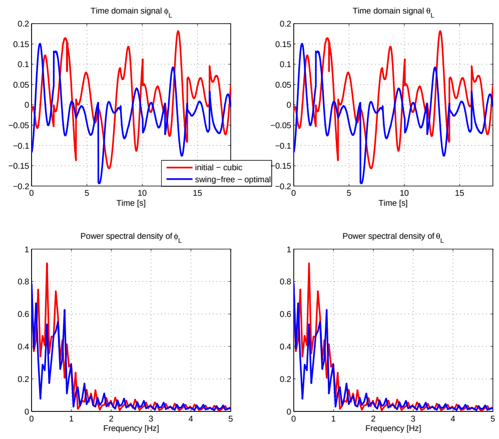
Figure 5.12: Experimental result for swing-free trajectory tracking



(a) Quadrotor position.



(b) Load displacement in Polar coordinates.



(c) Power Spectral Density.

Figure 5.13: Experimental results for swing-free trajectory tracking.

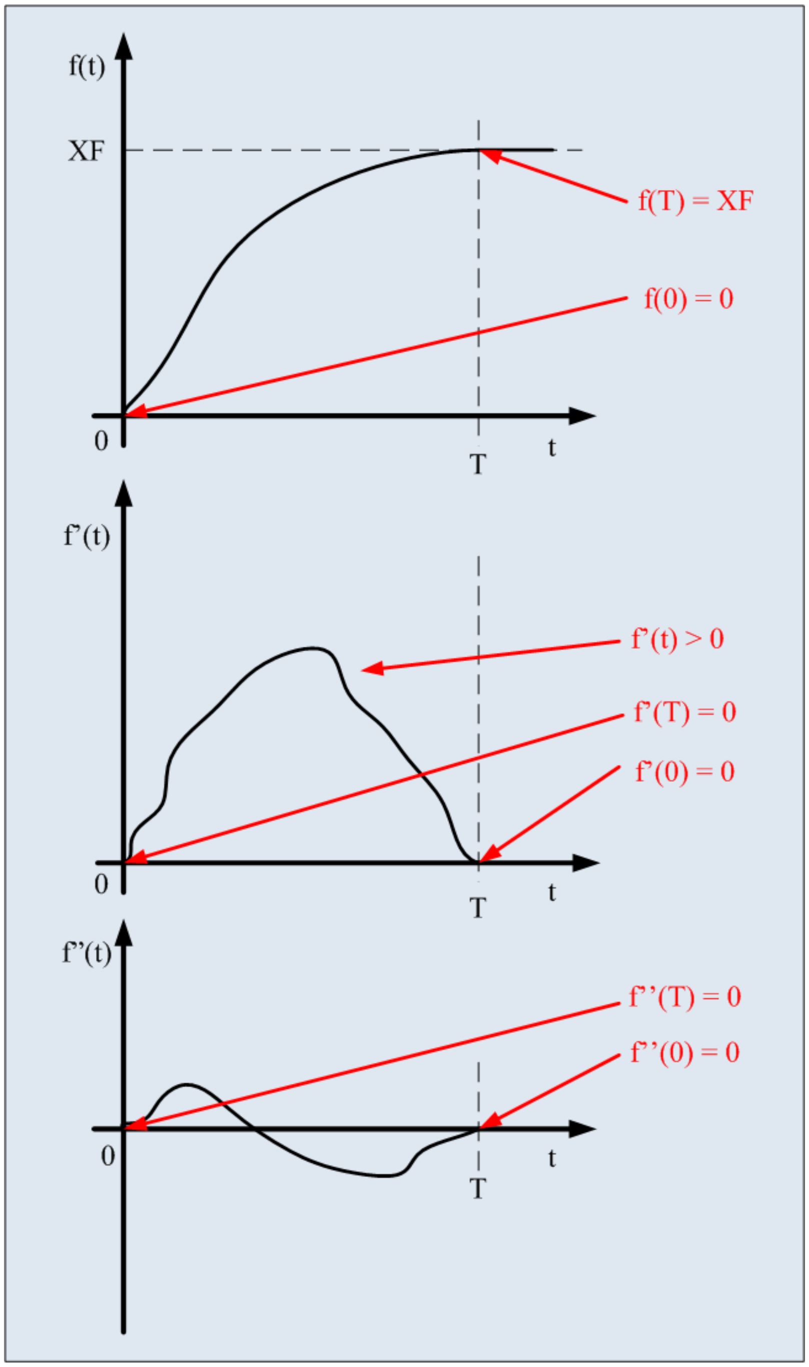


Figure 5.14: Constraints on the function represented by Taylor polynomial.

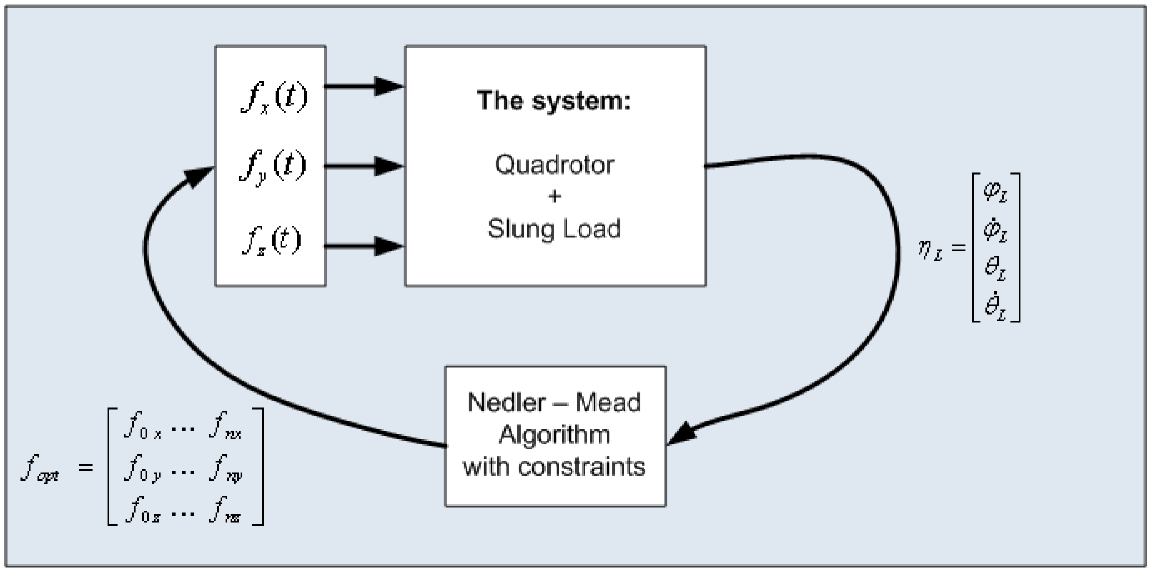
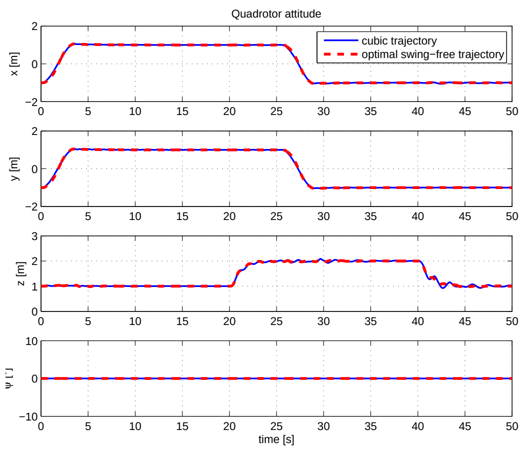
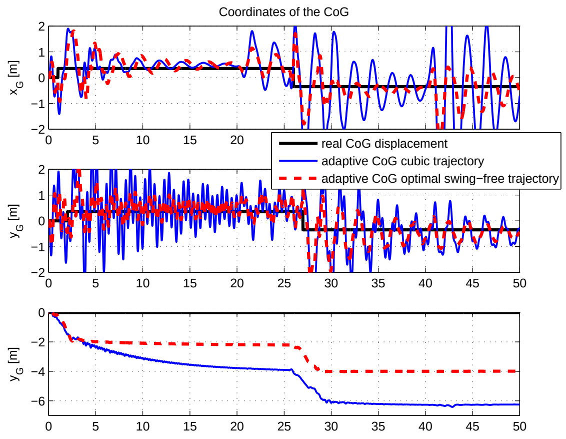

$f_x(t)$
$f_y(t)$
$f_z(t)$
The system:
Quadrotor
+
Slung Load
$\eta_L = \begin{bmatrix} \varphi_L \\ \dot{\varphi}_L \\ \theta_L \\ \dot{\theta}_L \end{bmatrix}$
Nedler – Mead Algorithm with constraints
$f_{opt} = \begin{bmatrix} f_{0x} \cdots f_{nx} \\ f_{0y} \cdots f_{ny} \\ f_{0z} \cdots f_{nz} \end{bmatrix}$

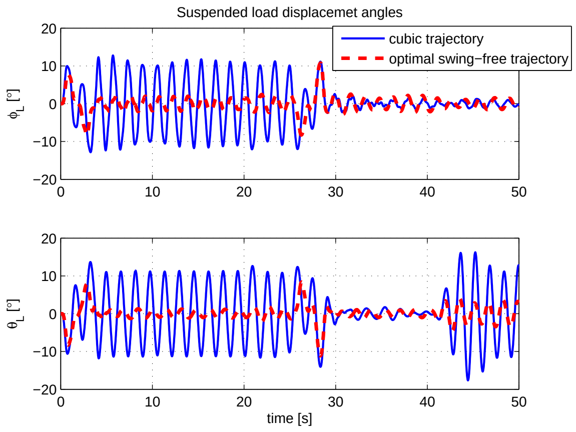
Figure 5.15: Block schemes for learning the coefficients of the Taylor approximating function.



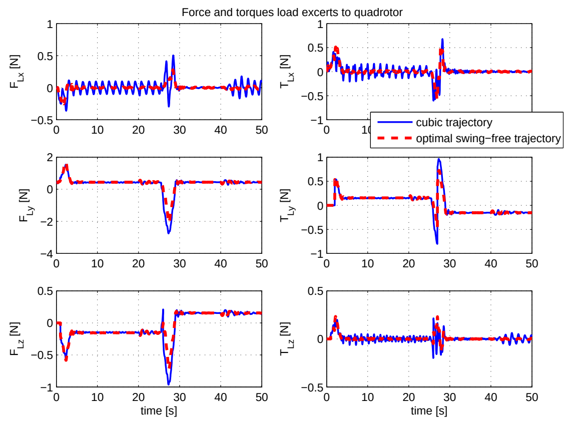
(a) Position and Heading.



(b) Estimated and real CoG coordinates.



(c) Suspended load displacement angles.



(d) Forces and torques from suspended load.

Figure 5.16: Adaptive feedback linearization algorithm with swing-free trajectory tracking.

Chapter 6

Trajectory tracking with suspended load

In most of the research about suspended load and small scale aerial robots the main focus is to maintain stability of the aerial robot - load system or to reduce the oscillations of the suspended load. In this chapter we are moving a step further where we develop algorithms for trajectory tracking with suspended load.

Reinforcement learning, a field of machine learning, is a family of algorithms that solve Markov Decision Processes (MDP) [63]. MDP is defined by a set of states, actions that perform process transitions from one state to another, and a reward value at every state. A solution to MDP is a sequence of actions that maximize the accumulated reward beginning at the start state. MDP assumes the Markovian property, meaning that the next state of the system depends only on the previous state and the action applied. Likewise, solving an MDP in a state s does not depend on how the state s was reached.

In the terms of reinforcement learning, a policy is a function that for each state specifies an action to be taken. Our goal is to find a policy that yields the highest return. Value function with respect to a policy at a state is an expected value of the accumulated reward starting in the current state and following the given policy.

Formally, let (S, A, P, R) be a MDP, where $S \subset R^n$ is state space, A is a finite action space, $P_a(s, s') = Pr(s_{t+1} = s'|s_t = s, a_t = a)$ is transition probability that the transitions the MDP from state s to state s' when action a is applied. We assume that our process is deterministic, thus $\exists s' \in S$ s.t. $P_a(s, s') = 1$ and 0 for all other states. Lastly $R : S \rightarrow R$ is an immediate reward at a state s. The goal is to find a policy $\pi : S \rightarrow A$ that maximizes $V_\pi : S \rightarrow R$ expected accumulated reward when policy π is followed. We assume R and therefore V are not known in advance. We further assume that we do not know the state transition function P.

In control theory, state space S is typically denoted X, action space A is U the action space of the controller, probability transition is referred to f transition function, and reward function R corresponds to ρ reward signal. Policy π in control theory terms is h [64]. In this dissertation, we use the control theory notation.

Trajectory tracking for helicopters based on reinforcement learning resulted in autonomous aggressive helicopter flights. This work was summarized by [51]. This line of research relies on apprenticeship learning to achieve aggressive autonomous maneuver such as as flips, rolls, loops, chaos, tic-tocs, and auto-rotation landings. An expert pilot demonstrates the target trajectory and the cost function is learned from the target trajectory. Parameters for initial linear model are learned from observing an expert fly non-aggressive maneuvers. Then the trajectory is broken down into small segments so that each can be separately learned using a linear dynamics model.

Online least square policy iteration has been successfully implemented in order to solve a benchmark control problems of balancing an inverted pendulum and balancing and riding a bicycle [64], [65]. The method shows promising convergence properties [66] and stability of the algorithm [65]. These and the fact that the method is model free are main reasons why we chose this method to solve our problem.

6.1 Unconstrained Nelder-Mead algorithm

Second task we want to accomplish using a quadrotor with suspended load is load tracking of a trajectory by generating the appropriate input trajectory for the quadrotor by learning its parameters.

For solving this problem we are using similar methodology we used in Section 5.3. The major difference is that for this problem we are using unconstrained Nelder-Mead algorithm. The objective function we are minimizing is given by

$$V = t \cdot e_L^T R e_L \cdot t, \tag{6.1}$$

where $e_L = [x_{Lref} - x_L \quad y_{Lref} - y_L \quad z_{Lref} - z_L]^T$ is the tracking error of the load and $R > 0, R \in \mathbb{R}$ is the weighting matrix. The method presented here is depicted by a scheme on Figure 6.1. We obtained Taylor series parameters after 197 iterations of Nelder-Mead algorithm. The coefficients are given as a vector $[f_0 \dots f_{10}] = [0.0001 \quad 0.2918 \quad 0.0011 \quad 0.0003 \quad -0.0003 \quad 0.0003 \quad -0.0001 \quad 0.0001 \quad 0.0006 \quad -0.0002 \quad 0.0007]$ and simulation results are presented in Figure 6.2.

6.2 Least square policy iteration

Reinforcement learning is a promising paradigm for learning optimal control algorithms. Policy iteration (PI) algorithms for reinforcement learning iteratively evaluate and improve control policies. State-of-the-art, least-squares techniques for policy evaluation are sample-efficient and have relaxed convergence requirements.

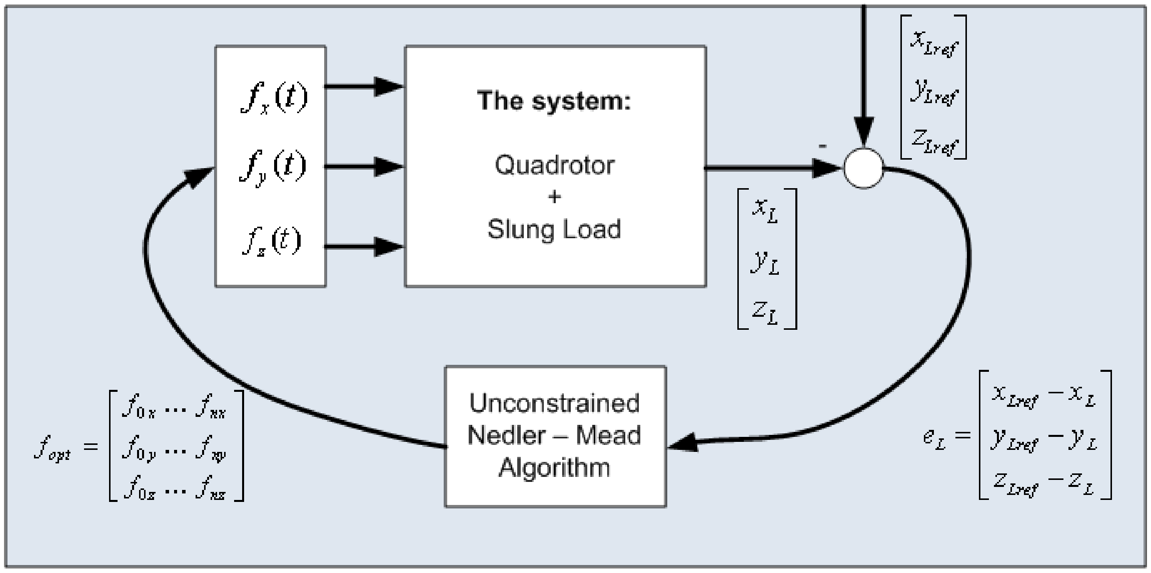

$f_x(t)$
$f_y(t)$
$f_z(t)$
The system:
Quadrotor + Slung Load
$\begin{bmatrix} x_{Lref} \\ y_{Lref} \\ z_{Lref} \end{bmatrix}$
-
$\begin{bmatrix} x_L \\ y_L \\ z_L \end{bmatrix}$
Unconstrained Nedler – Mead Algorithm
$f_{opt} = \begin{bmatrix} f_{0x} \cdots f_{nx} \\ f_{0y} \cdots f_{ny} \\ f_{0z} \cdots f_{nz} \end{bmatrix}$
$e_L = \begin{bmatrix} x_{Lref} - x_L \\ y_{Lref} - y_L \\ z_{Lref} - z_L \end{bmatrix}$

Figure 6.1: Block schemes for learning the coefficients of the Taylor approximating function.

Least square policy iteration (LSPI) is a reinforcement learning algorithm. Reinforcement learning (RL) algorithms are model-free and can be divided into three categories value iteration, policy iteration and policy search algorithms. Value iteration algorithms which search for the optimal value function in order to obtain the optimal policy. Policy iteration algorithms construct value functions which are then used in order to construct new, improved policies. Policy search algorithms use optimization techniques to directly search for an optimal policy. Offline RL algorithms use data collected in advance while online RL algorithms learn a solution by interacting with the process. Online RL algorithms must balance the need to collect the informative data (exploration) with the need to control the process well (exploitation). A quadrotor with a suspended load can be represented in a form of a deterministic Markov Decision Process (MDP)

$$x_{k+1} = f(x_k, u_k)$$

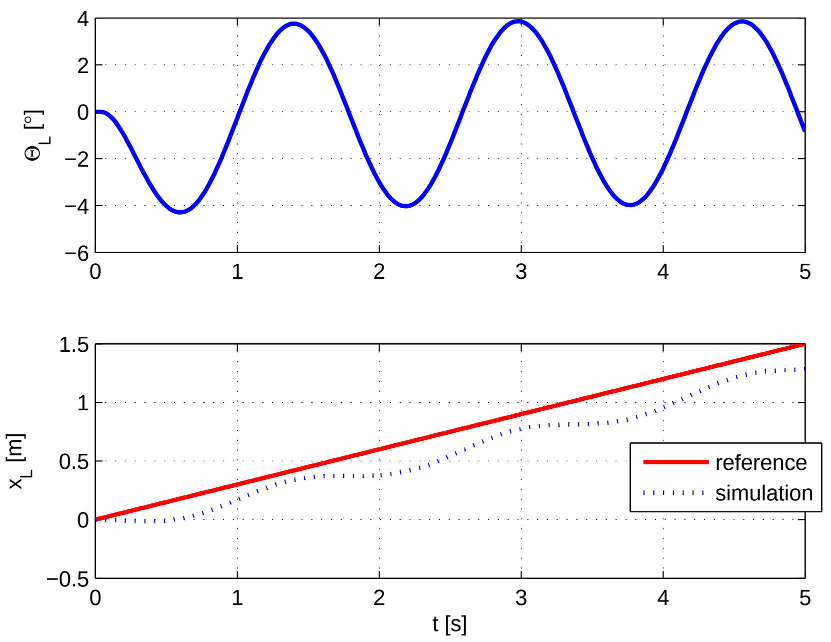


Figure 6.2: Load trajectory tracking using Nelder-Mead algorithm.

where X is the state space of the process, U is the action space of the controller, $f : X \times U \to X$ is the transition function of the process and $\rho : X \times U \to \mathbb{R}$ is the reward function

$$r_{k+1} = \rho(x_k, u_k).$$

The controller chooses actions according to its policy $h : X \to U$

$$u_k = h(x_k).$$

The goal is to find an optimal policy that maximizes the return from any initial state x_0. The return R is a cumulative aggregation of rewards

$$R^h(x_0) = \sum_{k=0}^{\infty} \gamma^k \rho(x_k, h(x_k))$$

where $\gamma \in [0,\ 1]$ is the discount factor. State-action value function (Q-function) $Q^h : X \times U \to \mathbb{R}$ of a policy h is given by

$$Q^h(x,u) = \rho(x,u) + \gamma R^h(f(x,u)).$$

The optimal Q-function is

$$Q^*(x,u) = \max_h Q^h(x,u).$$

The optimal policy (greedy policy in Q^*)

$$h(x) \in arg\max_u Q(x,u)$$

The Bellman equation for Q^h

$$Q^h(x,u) = \rho(x,u) + \gamma Q^h(f(x,u), h(f(x,u)))$$

and the Bellman equation for Q^*

$$Q^*(x,u) = \rho(x,u) + \gamma \max_{u'} Q^*(f(x,u), h(f(x,u'))).$$

Policy iteration evaluates policies by constructing their value functions which are then used to construct new, improved policies. Policy evaluation is evaluated at every iteration l solving the Bellman equation for Q^{h_l} of the current policy h_l. Policy improvement is then calculated using

$$h_{l+1}(x) \in arg\max_u Q^{h_l}(x,u).$$

The sequence of Q-functions produced by policy iteration asymptotically converges to Q^* as $l \to \infty$. In continuous spaces, policy evaluation cannot be solved exactly, and the value function has to be approximated. Linearly parametrized Q-function approximator $\hat{Q}$ consists of n basis function (BFs) $\phi_1, \ldots, \phi_n : X \times U \to \mathbb{R}$ and n dimensional parameter vector θ

$$\hat{Q} = \sum_{l=1}^{n} \phi_l(x,u)\theta_l = \phi^T(x,u)\theta$$

where $\phi(x,u) = [\phi_1(x,u),\ldots,\phi_n(x,u)]^T$. Control action u is a scalar which is bounded to an interval $U = [u_L \quad u_H]$.

Approximation of the value function is a difficult problem, because it involves finding an approximate solution to a Bellman equation. In this dissertation we use state-dependent basis functions and orthogonal polynomials of the action variable which separates approximation over the state space from approximation over the action space. Orthogonal polynomials are chosen because it is simple to solve the maximization problem over action variables and orthogonality ensure better conditioned regression problem at the policy improvement steps. As approximators over the state-action space we use Chebyshev polynomials of the first kind which are defined as

$$\begin{aligned}\psi_0(\bar{u}) &= 1,\\ \psi_1(\bar{u}) &= \bar{u},\\ \psi_{j+1}(\bar{u}) &= 2\bar{u}\psi_j(\bar{u}) - \psi_{j-1}(\bar{u}),\end{aligned} \tag{6.2}$$

where

$$\bar{u} = -1 + 2\frac{u - u_L}{u_H - u_L}.$$

These polynomials are defined on the interval $[-1 \quad 1]$. The online least square policy iteration algorithm is described by algorithm (1). The difference between an offline and an online variation of LSPI is that the online algorithm interacts with the environment at every iteration (line 10). Since we do not want for the algorithm to get stuck in the local minima, exploration and exploitation actions have to be balanced (line 9). To be able to converge quickly the algorithm needs to exploit in the best way the interaction with the environment, on the other hand in order to find the global minima it has to explore the action space with probability ε_k. As the time passes, the algorithm is relaying more on the current policy than on exploration, so

ε_k decays exponentially as the number of steps increases. Instead of waiting until many samples are passed, the online LSPI improves the policy by solving for the Q-function parameters every K_θ iterations using the current values of the Γ, Λ and z. When $K_\theta = 1$ then the policy is improved at every iteration step and is called fully optimistic.

Convergence of the LSPI algorithm using Chebyshev polynomials

Chebyshev polynomials are good approximators because they show uniform convergence as approximators. The following lemma states the uniform convergence of Chebyshev approximation as previously stated in [66] and [67]

Lemma 1 (Uniform convergence of Chebyshev approximation). *For all k, for all $f \in C^k[a;b]$,*

$$\|f - C_n\|_\infty \leq K$$

$$K = \left(4 + \frac{4}{\pi^2}\ln n\right)\frac{(n-k)!}{n!}\left(\frac{\pi}{2}\right)^k\left(\frac{b-a}{2}\right)^k\|f^k\|_\infty$$

and hence $C_n \longrightarrow f$ uniformly.

The next two theorems are taken from [66] and [67] and they ensure mean convergence of the approximate policy iteration algorithm with Chebyshev approximators.

Theorem 6.2.1 (Mean convergence of approximate PI). *Let $\hat{h}_0 \dots \hat{h}_n$ be the sequence of policies generated by an approximate policy iteration algorithm and let $\hat{V}_{\hat{h}_0} \dots \hat{V}_{\hat{h}_n}$ be the corresponding approximate value functions. Further assume that, for each fixed policy $\hat{h}_n$, the MDP is reduced to a Markov chain that admits an invariant probability measure $\mu_{\hat{h}_n}$. Let $\{\epsilon_n\}$ and $\{\delta_n\}$ be positive scalars that bound the mean errors in approximations to value functions and policies (over all iterations) respectively, that* *is* $\forall n \in \mathbb{N}$,

$$\mathbb{E}_{\mu_{\hat{h}_n}} \left\| \hat{V}_{\hat{h}_n} - V_{\hat{h}_n} \right\|_\infty \leq \epsilon_n, \tag{6.3}$$

and

$$\mathbb{E}_{\mu_{\hat{h}_n}} \left\| M_{\hat{h}_{n+1}} \hat{V}_{\hat{h}_n} - M\hat{V}_{\hat{h}_n} \right\|_\infty \leq \delta_n, \tag{6.4}$$

Suppose the sequences $\{\epsilon_n\}$ *and* $\{\delta_n\}$ *converge to* 0 *and* $\lim_{n\to\infty} \sum_{i=0}^{n-1} \gamma^{n-1-i}\epsilon_i = \lim_{n\to\infty} \delta^{n-1-i}\epsilon_i = 0$*, e.g.,* $\epsilon_i = \delta_i = \gamma^i$ *Then, this sequence eventually produces policies whose performance converges to the optimal performance in the mean:*

$$\lim_{n\to\infty} \mathbb{E}_{\mu_{\hat{h}_n}} \left\| M_{\hat{h}_{n+1}} \hat{V}_{\hat{h}_n} - M\hat{V}_{\hat{h}_n} \right\|_\infty = 0, \tag{6.5}$$

Proof. see [66] □

Theorem 6.2.2 (Convergence with Chebyshev polynomials)**.** *Suppose we have the same assumptions as in theorem 6.2.1 and further assume that, for any policy* $h \in \Pi$*, the value function* V^h *is in* $C^k[-1,1]$ *and the invariant density function* f^h *is known. Theorem 6.2.1 holds for the least square approximate policy iteration algorithm with Chebyshev polynomial approximation.*

Proof. see [66] □

Implementation of the LSPI algorithm for trajectory tracking with the suspended load using a quadrotor

Usually K_θ should be a number greater than zero but not too large. In our case $K_\theta = 7$. The reward function is defined as

$$r = -c_1 |u_k| - c_2 e_{Lk}^2 - c_3 \theta_{Lk}^2, \tag{6.6}$$

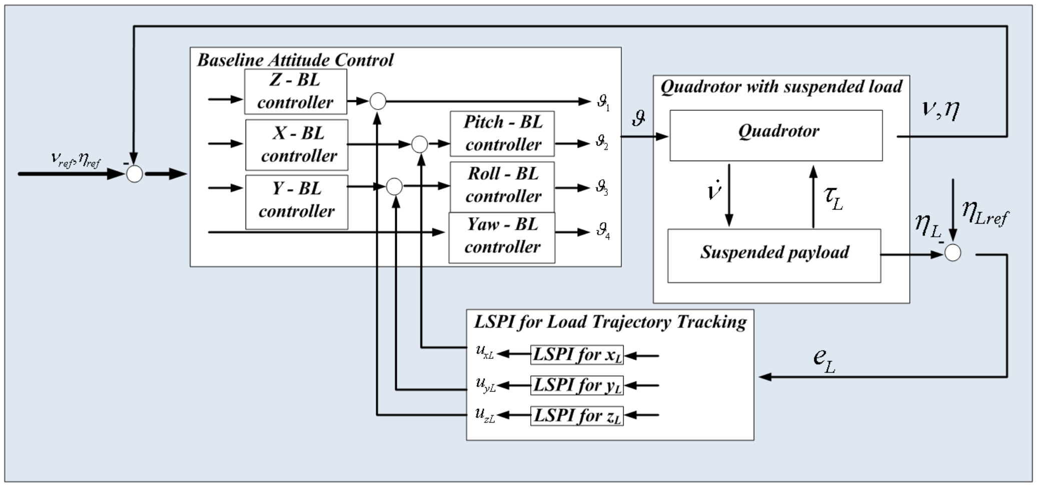


Figure 6.3: Block scheme for load trajectory tracking using LSPI.

where c_1, c_2, c_3 are positive constants, u_k is the action at step k, $e_{Lk} = x_{Lref} - x_L$ is the load tracking error and θ_{Lk} is the load displacement angle. Online LSPI is implemented in simulation and the results are shown in Figure 6.4 and Figure 6.6. The implementation is depicted by the scheme on Figure 6.3. Since in LSPI action is scalar, in order to implement 2D or 3D trajectory tracking we had to develop three different functions with three LSPI algorithms providing us with 3 control actions for the quadrotor, as depicted in Figure 6.3.

6.2.1 Simulation and experimental results

The algorithm described in the previous section is an online LSPI, meaning that during every simulation step, one iteration of the algorithm is performed while interacting with the system. Therefore, the algorithm provides an action at every simulation step, based on the previous outcome of the action or based on exploration.

We use the nonlinear model of the quadrotor and the suspended load described in

[68] for simulation purposes. The model provides environment feedback to the LSPI algorithm in form of the load tracking error e_{Lk} and the load displacement angle θ_{Lk} for the LSPI algorithm. Figure 6.3 illustrates vectors e_{Lk}, and θ_{Lk} as they relate to the position and the displacement angle of the load.

The algorithm and simulations are written and executed in Matlab and Simulink 2011, on a Windows machine in Marhes laboratory.

Straight line load trajectory simulation

For our first simulation, we have load tracking a straight line in xy plane going back and forth. Red lines in Figure 6.4(b) and 6.4(d) show the targeted trajectory.

We tracked the performance of the algorithm for 800 steps and these results are presented in Figure 6.4 in blue lines. Figure 6.4(a) shows the quadrotor trajectory resulting from the learning. Figures 6.4(b) and 6.4(d) present the load trajectory and displacement from the targeted trajectory. We can see that the load closely follows the reference trajectory, not diverging more than few centimeters from it.

Figure 6.4(c) depicts load tracking error that the agent is trying to minimize. The error is bounded which shows stability of the algorithm over 800 iteration steps.

To additionally confirm the convergence and stability of the LSPI algorithm we refer to the Figure 6.4 which shows simulation results for 3 simulation trials. We can see that the load starts tracking the set trajectory very fast in each of the simulation trials.

Lissajeou curve load trajectory simulation

To show that with LSPI algorithm the system is able to perform more complex trajectories, we chose a Lissajeou curve as a second trajectory to track. Figure 6.6(d) in red line shows Lissajeou curve. Figure 6.6(a) shows the quadrotor's trajectory after the learning. Figures 6.6(a) and 6.6(b) depict load's resulting trajectory (in blue) compared with desired trajectory. We can see that the load is following the reference closely. Figure 6.6(c) shows the load's error from the desired trajectory. Also in this case, the load tracking error is bounded. We can see the periodicity of the error, and also the trend for the peaks to reduce as the learning continues. The peak errors reduce from over 0.5 meters in the first 4,000 iterations to under 0.5 meters in the iterations over 8,000. Note, that this is error of the load from target trajectory, not from quadrotor. This means that, since this method does not directly control the load's swing, the displacement is due to the load's swing, and the quadrotor is leading the load closely to the desired trajectory. In both cases of the reference trajectories for the suspended load, the quadrotor reference is set to hover. LSPI algorithm drives the quadrotor to perform trajectories given in Figures 6.4(a) and 6.6(a).

Lissajeou curve load trajectory experiment

To verify that the simulation results are sound and that the produced LSPI trajectories are valid and to access true load displacement from the targeted load trajectory we performed experiments on a Hummingbird quadrotor (see Figure 6.7). The experimental results were performed in MARHES lab at University of New Mexico. More detailed description of the experimental testbed can be found in [56]. We generated trajectories for the quadrotor by learning in simulation, and tested them on a real quadrotor with suspended load. The learning was performed on 50Hz. We measured the quadrotor and load's trajectories and the load's displacement from the desired trajectory. We compare the experimental results with the simulation predictions.

Figure 6.8(a) compares the actual quadrotor trajectory as flown (in blue) with the trajectory predicted by simulation (in red). The differences between them are negligible, never exceeding more than 15 cm. This is a very promising result.

Figures 6.8(b) and 6.8(c) compare the load trajectory recorded in the experiment (in blue) with the simulation predicted trajectory (in red) and target trajectory (in black). We have a very interesting result here that shows that the experimental trajectory was much closer to the target trajectory then the simulation prediction. The reason for this discrepancy is that the model does not account for the load dumping effect, thus we obtain better results in reality than what simulations predict.

Again, to additionally confirm the convergence and stability of the LSPI algorithm we show 3 sets of simulations where the load tracks a y coordinate of Lissajeou curve as depicted in Figure 6.5(b). We can see that the load starts tracking the set trajectory very fast in each of the simulation trials.

Tracking and controlling the position of a suspended load can be critical to mission success. In this section we present a model-free approach to solve this problem involving a reinforcement learning algorithm. This method converges quickly to learn the policy function that minimizes the error in the load from the expected trajectory. We show results in simulation of this policy on a variety of trajectories from a simple straight line to a more complex Lissajeou curve. In both cases, the error is bounded to centimeters. In order to further validate the simulation, experiments were run on the experimental testbed. In these experiments, we see that the load is able to follow the Lissajeou curve trajectory closely.

The reinforcement learning methods presented are flexible and can be used both offline and online for trajectory tracking. This flexibility makes it highly appropriate for quadrotor policy learning.

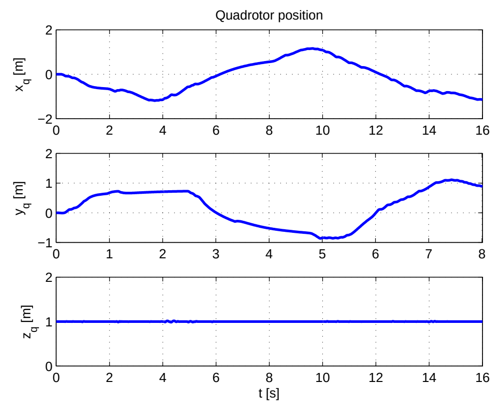
Videos of simulations and experiments can be found on [62].

6.3 Conclusion

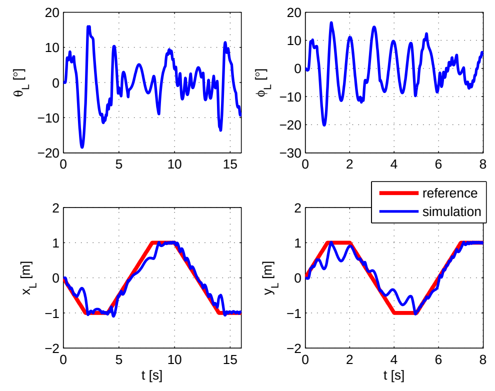
Tracking and controlling the position of a suspended load can be critical to mission success. In this chapter we present a model-free approaches to solve this problem involving algorithms based on Nelder-Mead algorithm and on reinforcement learning. Trajectory tracking with suspended load in combination with swing-free trajectories gives us more control over the system making it easier to manipulate safely in cluttered environments.

Algorithm 1 Online LSPI with ε - greedy exploration

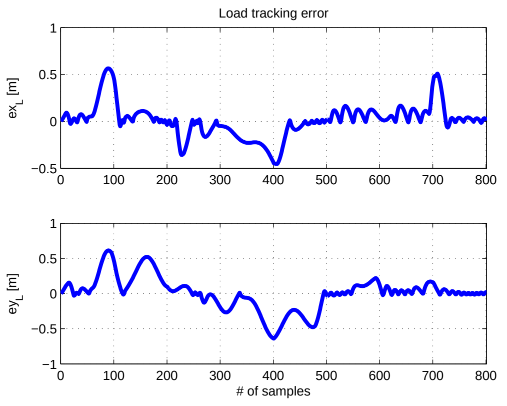
```
 1: Input: discount factor $\gamma$,
 2: BFs $\phi_1, \dots \phi_n : X \times U \to \mathbb{R}$,
 3: policy improvement interval $K_\theta$, exploration $\{\varepsilon_k\}_{k=0}^{\infty}$,
 4: a small constant $\beta_\Gamma > 0$
 5: $l \leftarrow 0$, initialize policy $h_0$,
 6: $\Gamma_0 \leftarrow \beta_\Gamma I_{n \times n}$, $\Lambda_0 \leftarrow 0$, $z_0 \leftarrow 0$
 7: measure initial state $x_0$
 8: for every time step $k = 0, 1, 2, \dots$ do
 9:     $u_k \leftarrow \begin{cases} h_l(x_k) : \text{ with prob.} 1 - \varepsilon_k \text{ (exploit)} \\ \text{uni. rand. ac. in U : with prob.} \varepsilon_k \text{ (explore)} \end{cases}$
10:     apply $u_k$, measure next state $x_{k+1}$ and reward $r_{k+1}$
11:     $\Gamma_{k+1} \leftarrow \Gamma_k + \phi(x_k, u_k)\phi^T(x_k, u_k)$
12:     $\Lambda_{k+1} \leftarrow \Lambda_k + \phi(x_k, u_k)\phi^T(x_{k+1}, h_l(x_{k+1}))$
13:     $z_{k+1} \leftarrow z_k + \phi(x_k, u_k)r_{k+1}$
14:     if $k = (l + 1)K_\theta$ then
15:         solve $\frac{1}{k+1}\Gamma_{k+1}\theta_l = \frac{1}{k+1}\Lambda_{k+1}\theta_l + \frac{1}{k+1}z_{k+1}$
16:         $h_{l+1}(x) \leftarrow \arg\max_u \phi^T(x, u)\theta_l, \forall x$
17:         $l \leftarrow l + 1$
18:     end if
19: end for
```



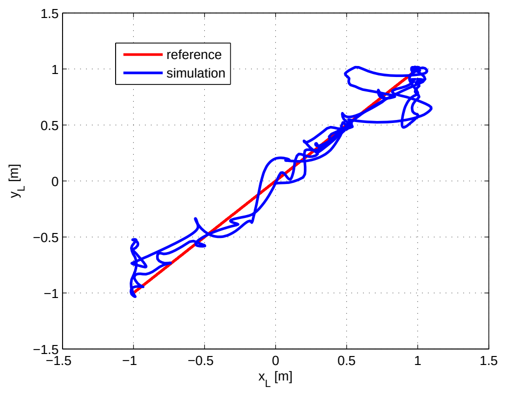
(a) Quadrotor position.



(b) Load displacement in Polar and Cartesian coordinates.

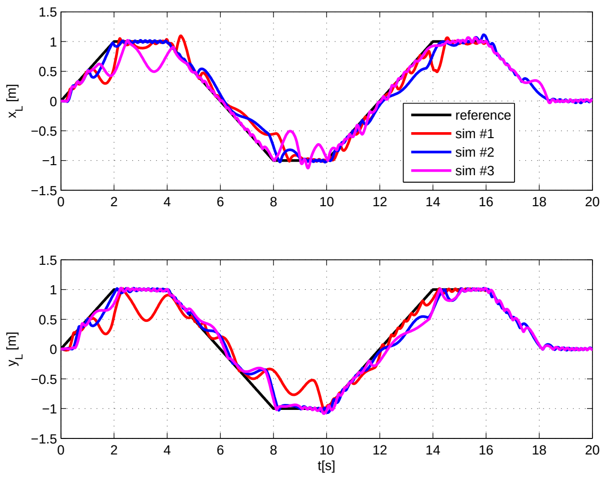


(c) Load tracking error.

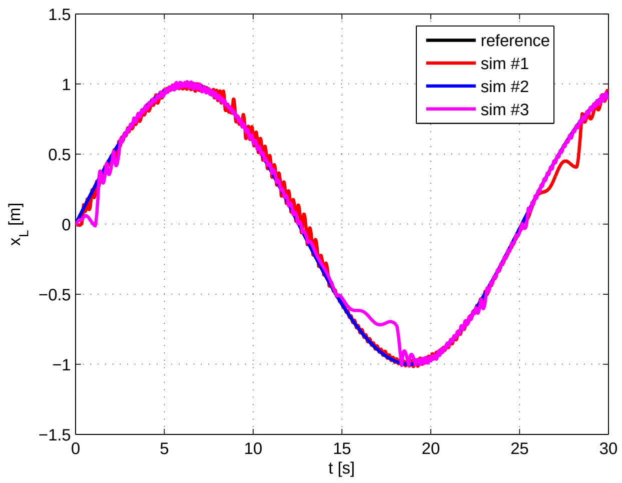


(d) Load displacement in 2D.

Figure 6.4: Simulation results for online LSPI used for suspended load tracking the straight line.

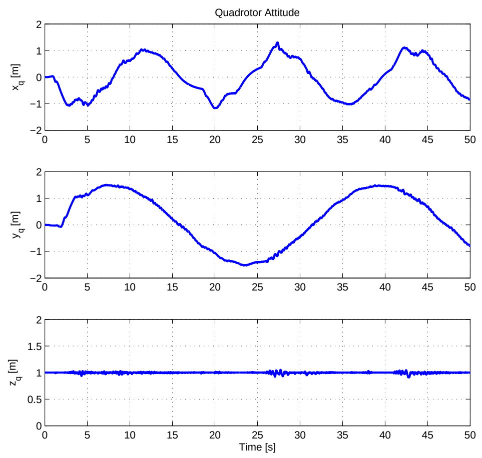


(a) Suspended load tracking a straight line trajectory

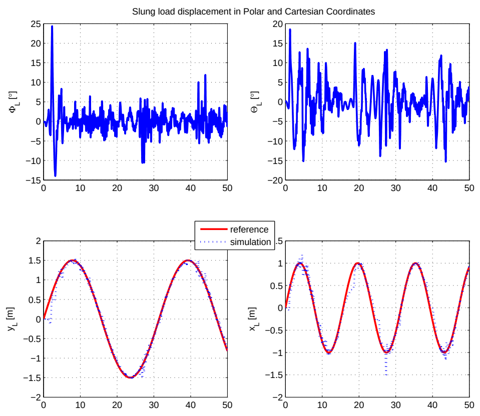


(b) Suspended load tracking a sinusoidal trajectory

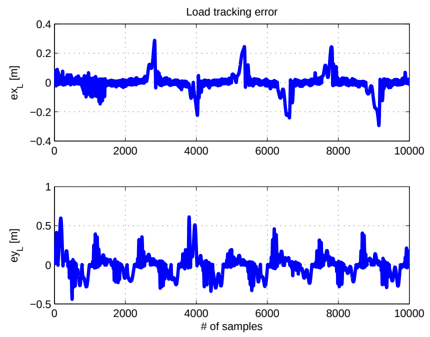
Figure 6.5: Simulation results for online LSPI used in multiple simulation trials.



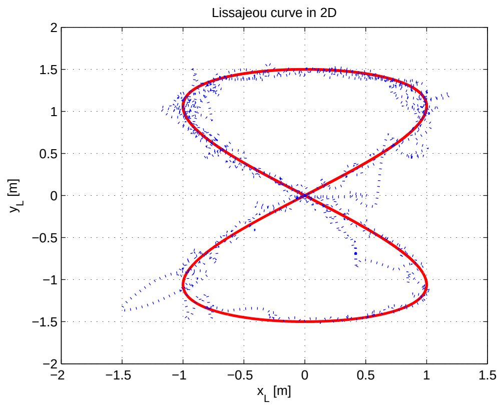
(a) Quadrotor position.



(b) Load displacement in Polar and Cartesian coordinates.



(c) Load tracking error.



(d) Load displacement in 2D.

Figure 6.6: Simulation results for load tracking of Lissajeou curve using online LSPI.

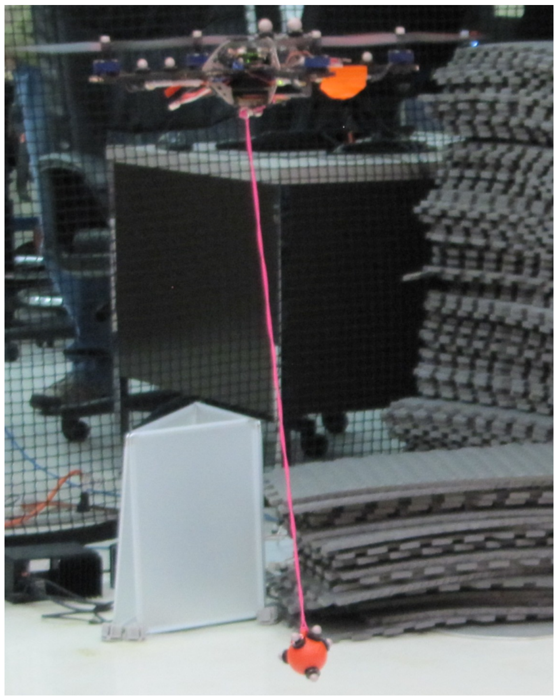
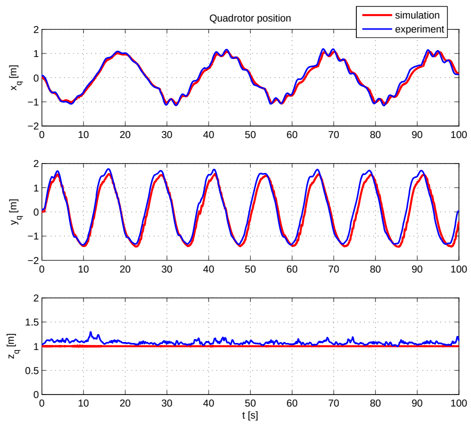
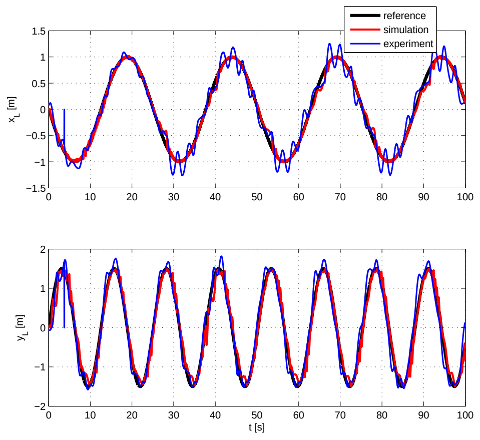


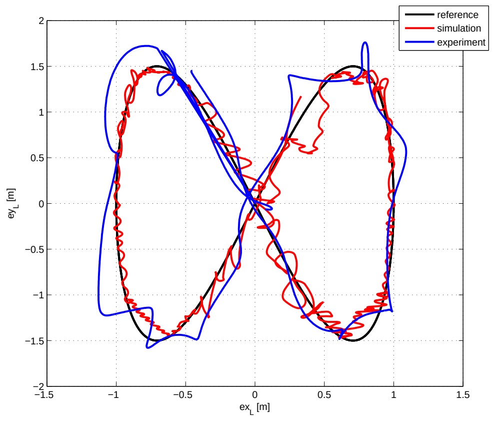
Figure 6.7: Hummingbird quadrotor with a suspended load at MARHES Lab University of New Mexico.



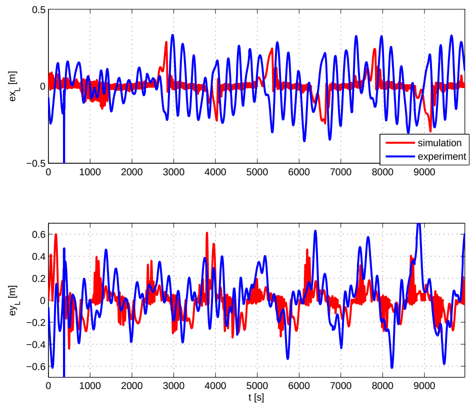
(a) Quadrotor position.



(b) Suspended load position.



(c) Load displacement in 2D.



(d) Load tracking error.

Figure 6.8: Experimental results using the trajectory generated in simulation using LSPI.

Chapter 7

Navigation in cluttered environments and distributed trajectory tracking

7.1 Navigation in cluttered environments

In this section we present a set of simulations as a motivation for use of swing-free trajectory tracking in urban environments. Let $\mathcal{Q}$ represent the origin of the moving coordinate frame $\{\mathcal{A}\}$ attached to a quadrotor body with suspended load in a Euclidean space $\mathbb{R}^3$, and let convex sets $\mathcal{B}_1, ..., \mathcal{B}_n$ be fixed rigid objects distributed in $\mathcal{W}$ (Figure 1.2.4). The $\mathcal{B}_i$'s are defined as obstacles. We assume that both the geometry of $\mathcal{Q}, \mathcal{B}_1, ..., \mathcal{B}_n$ and the locations of the $\mathcal{B}_i$'s are known apriori and accurately in a 3-dimensional area-of-interest $\mathcal{W}$. We assume that given an initial position and a goal position of $\mathcal{Q}$ in $\mathcal{W}$, a collision-free path $\mathcal{P}$ is generated using a high level planner based on any of the well known path planning techniques such as potential field, cell decomposition, etc. [21].

The problem of interest is stated as follows: A path $\mathcal{P}$ is approximated with a finite number of waypoints w_i such that the straight-line path $\mathcal{P}_{segment}$ between two waypoints belongs to $\mathcal{W}_{free}$. Based on given waypoints w_i the mid-level planner calculates the minimal distance between the path $\mathcal{P}_{segment}$ and the surrounding obstacles $min_{obst-dist}$. Subsequently, it determines the maximum allowable load displacement angle $max_{allow-angle}$ such that the trajectory of the load $\mathcal{T}_{load}$ belongs to $\mathcal{W}_{free}$. By determining $max_{allow-angle} \neq 0$ we the value of the vector x_F in the objective function (5.12) with respect to a swing-free policy considered in Sec. 5.2. Using the principle of $max_{allow-angle}$ we can incorporate a safety region for the maximal displacement of the load by narrowing the $min_{obst-dist}$ by δ_{safe}. This approach makes the overall system more robust and reliable. Subsequently, given $max_{allow-angle}$, the mid-level planner generates an optimal trajectory $\mathcal{T}_{opt}$ for $\mathcal{Q}$ using the DP algorithm presented in Sec. 5.2. The optimal trajectory is generated in such a way that both, the performed trajectory of the quadrotor $\mathcal{T}_{quadrotor}$ and the performed trajectory of the load $\mathcal{T}_{load}$ belong to $\mathcal{W}_{free}$.

```
**for** $i = 1 \to n-1$ **do**
    $w_{start} \leftarrow w_i$
    $w_{end} \leftarrow w_{i+1}$
    **find** $min_{obst-dist_{i+1}}$
    **find** $max_{allow-angle_{i+1}}$ **considering** $\delta_{safe}$
    **generate** $\mathcal{T}_{opt_i}$
**end for**
```

Simulation results for mid-level planner: A motivational case study

In this subsection we present a set of simulations for a case study for the use of the mid-level planner explained in Sec. 7.1. In a Euclidean space $\mathbb{R}^3$, we define six fixed rigid objects defined as convex sets $\mathcal{B}_1, ..., \mathcal{B}_6$ distributed in $\mathcal{W}$ (Figure 7.1). The

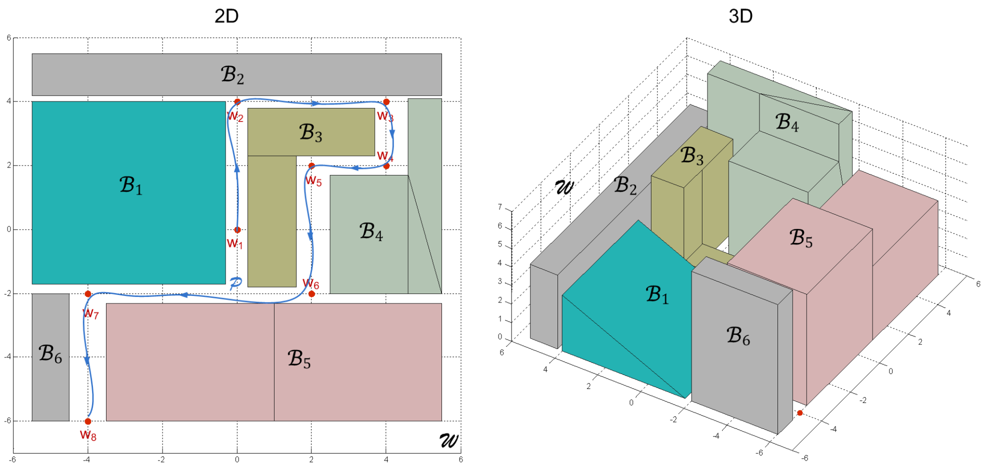


Figure 7.1: A Motivational case study: Representation of Eucledian space with obstacles and predefined waypoints

path $\mathcal{P}$ is approximated with eight waypoints $w_1, ..., w_8$ where the straight-line paths $\mathcal{P}_{segment_1}, ..., \mathcal{P}_{segment_7}$ belong to $\mathcal{W}_{free}$.

The main goal of the mid-level planner is to generate optimal trajectories between given waypoints in such a way that the overall time for completing the mission t_{total} is minimal, while both, $\mathcal{T}_{quadrotor}$ and $\mathcal{T}_{load}$ still belong to $\mathcal{W}_{free}$, i.e. the presented system performs collision-free maneuvers.

By using standard swing-free policy, where the displacement angle of the load is required to be zero, we obtain the optimal trajectory presented in Figure 7.2. We can see that a quadrotor performs optimal trajectory tracking without collision, (Figure 7.2(b)). Using the initial cubic trajectories the displacement angles are to large to avoid collision (Figure 7.3).

Although the swing-free policy gives us satisfactory results considering collision-free maneuvers, the overall time for completing the mission t_total is not minimal. Figures 7.4, 7.5 show four different ways of how to build required trajectories. The obstacles are positioned in such a way that the vertical corridors are wider than the horizontal ones. This is done just for easier interpretation of the simulation results and sets no constraints to the overall algorithm. First case is a swing-free policy, where the load displacement angle is minimized but the overall time is not taken into consideration. The overall time for completing the mission is $t_{total_1} = 28s$. In the second case we generate the fastest possible trajectory where the overall system is still stable. The overall time for completing the mission is $t_{total_2} = 17s$. In the forth and the third case we are using the mid-level planner. The overall time for completing the mission in both cases are similar $t_{total_3} = 21.4s$, $t_{total_4} = 24.2s$. The third case shows that the proposed method is sensitive to initial conditions of the load displacement angles and they had to be taken into consideration while using the mid-level planner. This means that the mid-level planner has to set the $max_{allow-angle_i}$ considering the next path segment $\mathcal{P}_{segment_{i+1}}$ not the current one. The fourth case represents the optimal collision-free trajectory as shown in Figure 7.4, with $t_{total_4} = 24.2s$.

7.2 Load transport using multiple quadrotors

Recently load manipulation with multiple unmanned aerial vehicles came in the scope of research community. Some of the relevant papers in this field are [11], [13], [15]. In all of these cases the control algorithm was centralized. By centralized we mean that the control algorithm for multiple aerial vehicles is implemented in one main computer. In this section we are proposing the control algorithm for load manipulation in a decentralized fashion using consensus in order to synchronize the group of quadrotors. The goal is to obtain a system we are able to control a single system, from a control and computational perspective.

The synchronization phenomenon in nature is common. The design of distributed communication and control protocols is an important issue in the construction of networked systems. With the advances of embedded systems and communication networks, the coordination and consensus of distributed groups of agents attracted a great deal of attention from the control community [69], [70] [71] [72]. In this dissertation we use binary consensus in order to achieve swing-free trajectory tracking in a distributed fashion.

The communication topology set for application described in this section is a directed tree, or a directed graph (Figure 7.6(b)). We consider a network of quadrotors which can be represented, with slight abuse of notation, in Lagrangian form

$$\mathbf{M}_i\dot{\boldsymbol{\nu}}_{\boldsymbol{i}} + \mathbf{C}_i\left(\boldsymbol{\nu}_{\boldsymbol{i}}\right)\boldsymbol{\nu}_{\boldsymbol{i}} + \mathbf{D}_i\boldsymbol{\nu}_{\boldsymbol{i}} + \mathbf{g}_{Gi}\left(\boldsymbol{\eta}_{\boldsymbol{i}}\right) = \boldsymbol{\tau}_{\boldsymbol{i}}, \quad i = 1\ldots n, \tag{7.1}$$

where $\boldsymbol{\eta}_i \in \mathbb{R}^{6\times1}$ is the vector of position and orientation, $\boldsymbol{\nu}_i \in \mathbb{R}^{6\times1}$ is vector of linear and angular velocities, $\mathbf{M}_i \in \mathbb{R}^{6\times6}$ is the mass and inertia matrix, matrix $\mathbf{C}_i\left(\boldsymbol{\nu}_i\right) \in \mathbb{R}^{6\times6}$ consists of Coriolis and centripetal terms, $\mathbf{D}_i \in \mathbb{R}^{6\times6}$ is the damping matrix, $\mathbf{g}_{Gi}\left(\boldsymbol{\eta}_i\right) \in \mathbb{R}^{6\times1}$ is the vector of gravitational forces and moments and $\boldsymbol{\tau}_i \in \mathbb{R}^{6\times1}$ represents control inputs. The leader is indexed by 0 and it can be either a physical or a virtual system. By physical we mean that the in the network of quadrotors carrying suspended load, one is denoted as a leader. And by a virtual leader, the nonlinear model of the quadrotor with suspended load would be run in simulation while the rest of the network would be physical quadrotors performing distributed tracking using binary control protocol. In this application we are interested in investigating how binary control protocol can be used for distributed swing-free trajectory tracking with multiple quadrotors. We assume that the acceleration of the leader is bounded $\|\dot{\boldsymbol{\nu}}\| \leq \dot{\boldsymbol{\nu}}_{\max}$. The control input vector is given by

$$\boldsymbol{\tau}_{\boldsymbol{i}} = \boldsymbol{\chi}_{\boldsymbol{i}} + \boldsymbol{\kappa}_{\boldsymbol{i}}, \tag{7.2}$$

where $\boldsymbol{\chi}_{\boldsymbol{i}}$ denotes feedback linearization term given as

$$\boldsymbol{\chi}_i = \mathbf{C}_i\left(\boldsymbol{\nu}_i\right)\boldsymbol{\nu}_i + \mathbf{D}_i\boldsymbol{\nu}_i + \mathbf{g}_{Gi}\left(\boldsymbol{\eta}_i\right) + \mathbf{M}_i\boldsymbol{\vartheta}_i, \tag{7.3}$$

and $\boldsymbol{\kappa_i}$ denotes cooperative control given as

$$\boldsymbol{\kappa_i} = \frac{1}{\sum_{i\in N_i} a_{ij} + b_i}\left(\sum_{i\in N_i} a_{ij}\left(s_1 \operatorname{sgn}\left(\boldsymbol{\eta}_j - \boldsymbol{\eta}_i\right)\right) + s_2\left(\operatorname{sgn}\left(\boldsymbol{\nu}_j - \boldsymbol{\nu}_i\right)\right)\right.$$
$$\left. + b_i\left(s_1 \operatorname{sgn}\left(\boldsymbol{\eta}_0 - \boldsymbol{\eta}_i\right)\right) + s_2\left(\operatorname{sgn}\left(\boldsymbol{\nu}_0 - \boldsymbol{\nu}_i\right)\right)\right) \tag{7.4}$$

where $s_1 > s_2 + \dot{\boldsymbol{\nu}}_{\max}$,$s_2 > \dot{\boldsymbol{\nu}}_{\max}$, and $b_i \geq 0$, $b_i > 0$ if and only if there is an edge from the leader node 0 to node i.

At the end we take the adaptive control and swing-free trajectory generation developed for a single quadrotor and expand it to a network of four quadrotors (Figure 7.6(b)). The leader, denoted by 0 node is tracking the optimal swing-free trajectory. The leader information is available only to follower 1. Simulation results show a promising method for distributed swing-free trajectory tracking (Figure 7.7(b)). As the tracking error propagates from leader to followers the damping of the residual oscillations deteriorate, but are still significantly dampened.

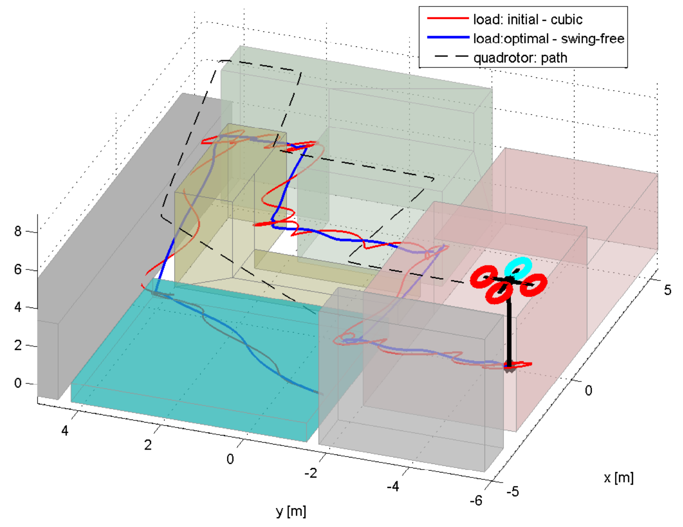
One of the reasons we are able to achieve good trajectory tracking is binary consensus applied to Lagrangian systems. Binary consensus for Lagrangian systems is proven to achieve synchronization in finite time in [22]. The theorem is stated in the following.

Theorem 7.2.1. *Consider the networked Lagrangian systems described by (7.1) with the local feedback control (7.3) and the cooperative binary control input (7.5). Let the communication graph be a directed tree. Then the networked Lagrangian systems synchronize in finite time.*

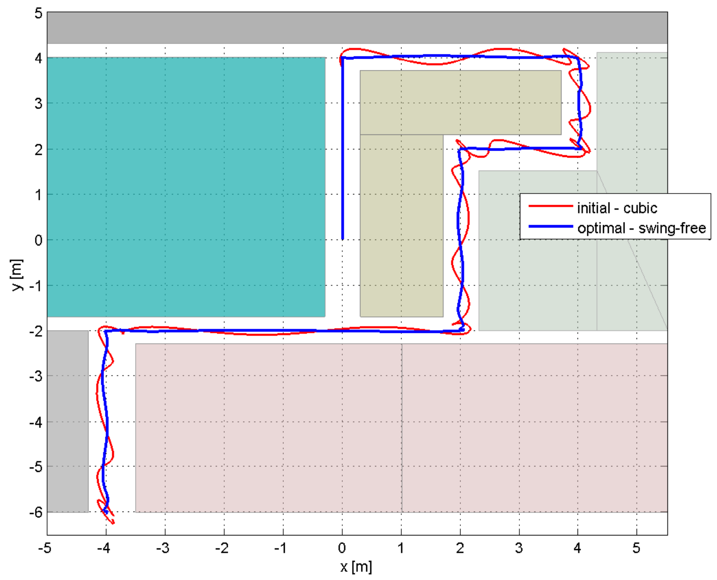
Proof. see [22] □

7.3 Conclusion

in this chapter we propose two algorithms, one for navigation in cluttered environments and the other for distributed swing-free maneuvers with suspended load. There is a lot of room for further development of these algorithms. For example, one can incorporate an online algorithm for navigation with moving obstacles together with a multi quadrotor - payload system.

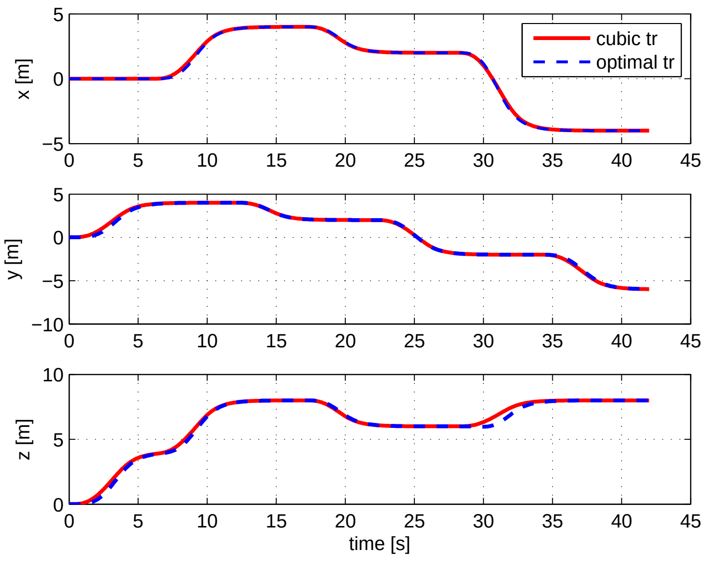


(a) 3D view.

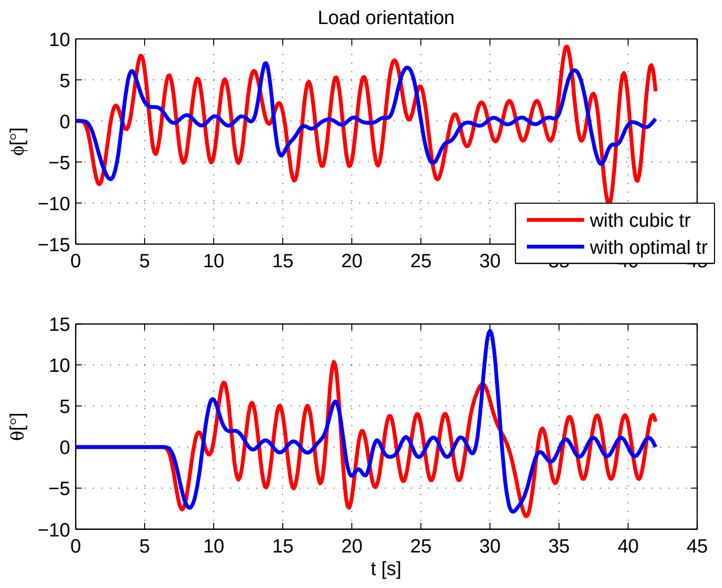


(b) 2D view.

Figure 7.2: 3D and 2D representation of trajectories considering multiple waypoints with obstacles.

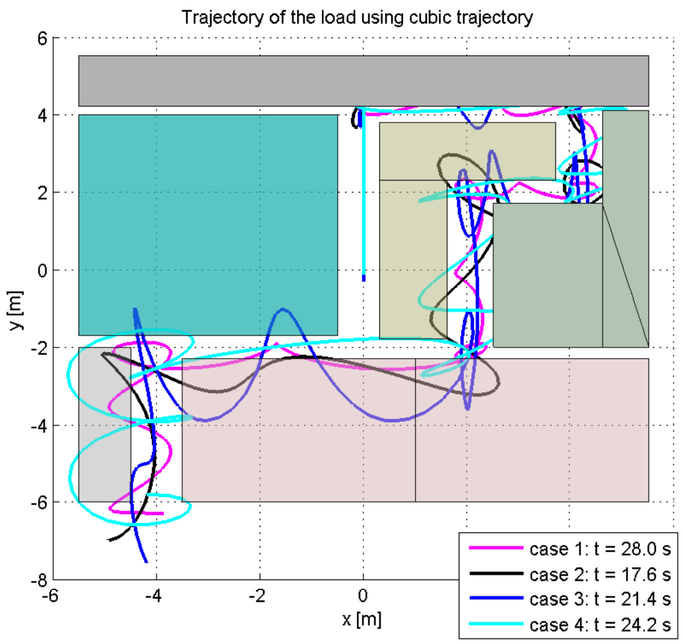


(a) Cubic and Optimal trajectories

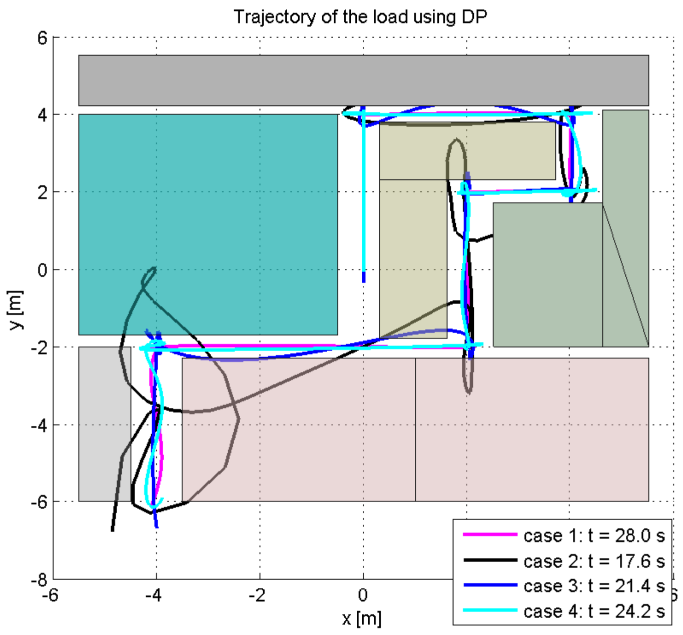


(b) Load displacement angles

Figure 7.3: Trajectories considering multiple waypoints with obstacles - the swing - free policy.

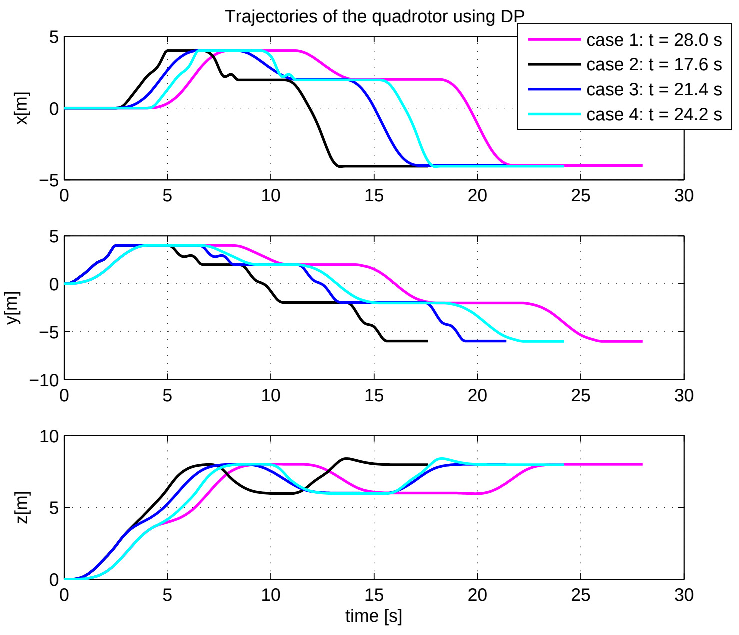


(a) Cubic trajectories with obstacles

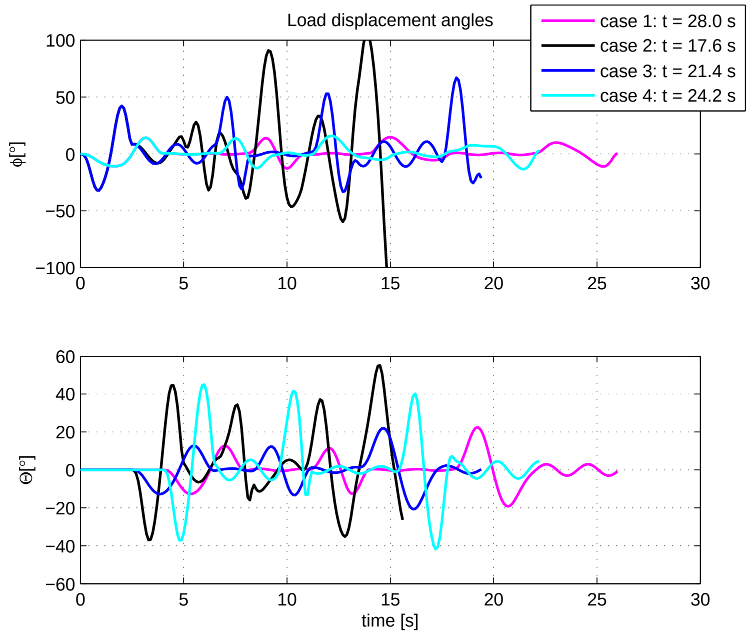


(b) Optimal trajectories with obstacles

Figure 7.4: 2D representation of trajectories considering multiple waypoints with obstacles - a four case study.

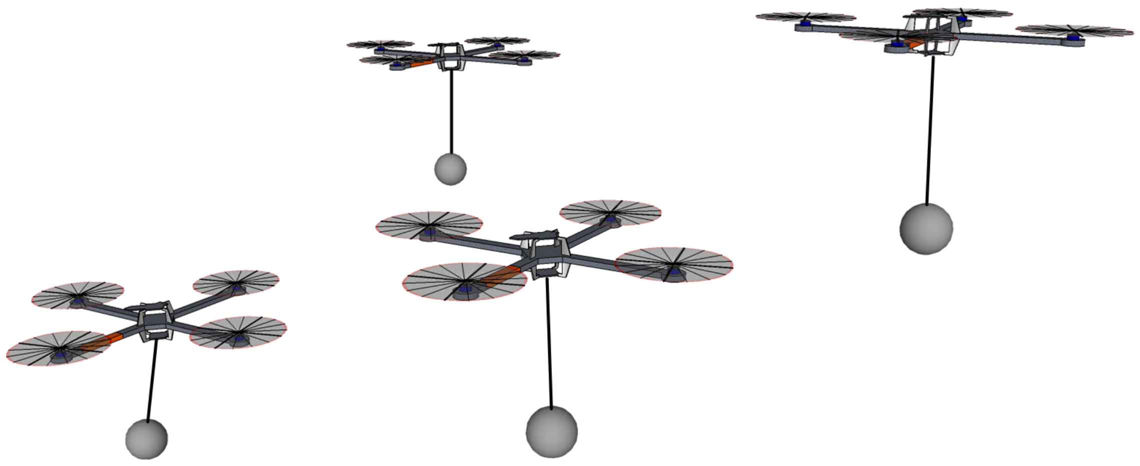


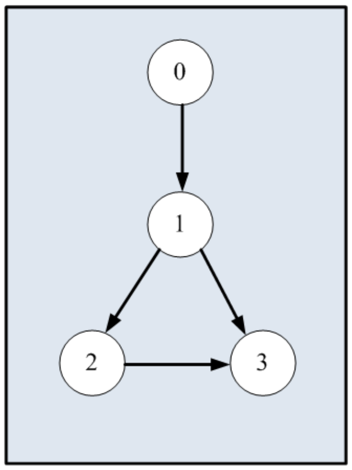
(a) Optimal trajectories



(b) Load displacement angles

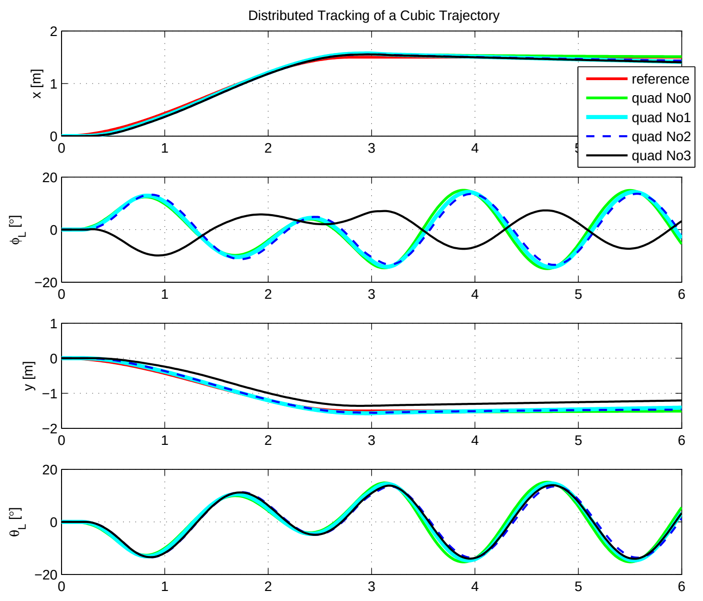
Figure 7.5: Trajectories considering multiple waypoints with obstacles - a four case study.



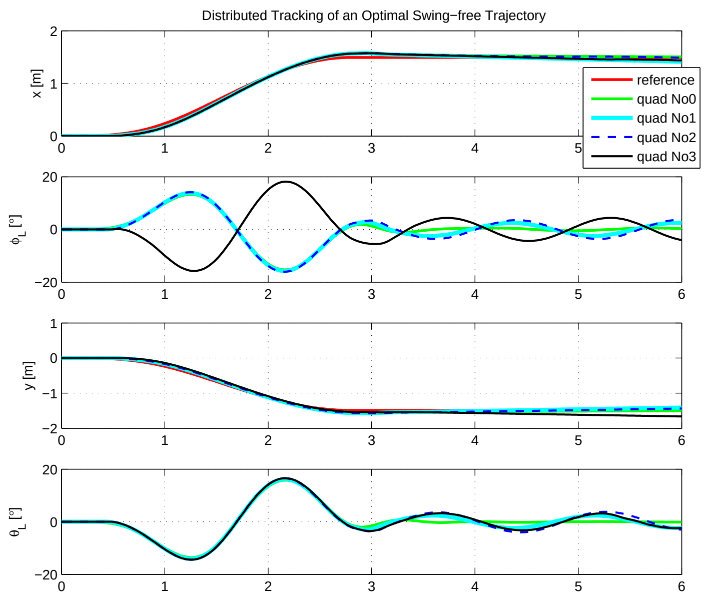


(b) Diagram of the communication graph with four quadrotors.

Figure 7.6: Suspended load transport using four quadrotors.



(a) Distributed tracking of a cubic trajectory - 4 quadrotors.



(b) Distributed tracking of an optimal trajectory - 4 quadrotors.

Figure 7.7: Binary consensus protocol for distributed swing-free trajectory tracking.

References

[1] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, “The GRASP multiple micro-uav test bed,” *IEEE Robotics and Automation Magazine*, vol. 17, no. 3, pp. 56–65, September 2010.

[2] G. Hoffman, D. Rajnarayan, S. Waslander, D. Dostal, J. Jang, and C. Tomlin, “The stanford testbed of autonomous rotorcraft for multi agent control (STARMAC),” in *The 23rd Digital Avionics Systems Conference*, October 2004, pp. 12.E.4–121–10.

[3] J. How, B. Bethke, A. Frank, D. Dale, and J. Vian, “Real-time indoor autonomous vehicle test environment,” *IEEE Control Systems Magazine*, vol. 28, no. 2, pp. 51–64, April 2008.

[4] S. Lupashin, A. Schöllig, M. Hehn, and R. D’Andrea, “The flying machine arena as of 2010,” in *IEEE International Conference on Robotics and Automation*, Shanghai, China, May 2011, pp. 2970–2971.

[5] S. Bouabdallah, P. Murrieri, and R. Siegwart, “Design and control of an indoor micro quadrotor,” in *IEEE International Conference on Robotics and Automation*, vol. 5, Barcelona, Spain, April 2004, pp. 4393 – 4398.

[6] G. Hoffman, S. Waslander, and C. Tomlin, “Quadrotor helicopter trajectory tracking control,” in *AIAA Guidance, Navigation and Control Conference and Exhibit*, April 2008, pp. 1–14.

[7] A. Schöllig, F. Augugliaro, and R. D’Andrea, “A platform for dance performances with multiple quadrocopters,” in *IEEE International Conference on Intelligent Robots and Systems.*

[8] M. Valenti, D. Dale, J. How, and D. Pucci de Farias, “Mission health management for 24/7 persistent surveillance operations,” in *AIAA Conference on Guidance, Navigation and Control*, Hilton Head, SC, August 2007, pp. 1–18.

[9] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," in *International Symposium on Experimental Robotics*, New Delhi, India, December 2010.

[10] S. Lupashin, A. Schöllig, M. Sherback, and R. D'Andrea, "A simple learning strategy for high-speed quadrocopter multi-flips," in *IEEE International Conference on Robotics and Automation*, May 2010, pp. 1642–1648.

[11] N. Michael, J. Fink, and V. Kumar, "Cooperative manipulation and transportation with aerial robots," in *Robotics: Science and Systems*, Seattle, WA, June 2009.

[12] M. Bernard and K. Kondak, "Generic slung load transportation system using small size helicopters," in *IEEE International Conference on Robotics and Automation*, Kobe, Japan, May 2009, pp. 3258–3264.

[13] I. Maza, K. Kondak, M. Bernard, and A. Ollero, "Multi-UAV cooperation and control for load transportation and deployment," *Journal of Intelligent and Robotic Systems*, vol. 57, no. 1, pp. 417–449, January 2010.

[14] P. Pounds, D. Bersak, and A. Dollar, "Grasping from the air: Hovering capture and load stability," in *IEEE International Conference on Robotics and Automation*, Shangai, China, May 2011, pp. 2491–2498.

[15] D. Mellinger, Q. Lindsey, M. Shomin, and V. Kumar, "Design, modelling, estimation and control for aerial grasping and manipulation," in *IEEE International Conference on Intelligent Robots and Systems*, San Francisco, USA, September 2011, pp. 2668–2673.

[16] A. Albert, S. Trautmann, T. Howard, T. Nguyen, M. Frietsch, and C. Sauter, "Semi-autonomous flying robot for physical interaction with environment," in *IEEE Conference on Robotics Automation and Mechatronics*, Singapore, June 2010, pp. 441–446.

[17] Q. Lindsey, D. Mellinger, and V. Kumar, "Construction of cubic structures with quadrotor teams," June 2011.

[18] C. Korpela, T. Danko, and P. Oh, "Designing a system for mobile manipulation from an unmaned aerial vehicle," in *IEEE Conference on Technologies for Practical Robot Applications*, April 2011, pp. 109–114.

[19] (2011, August) Matternet wants to deliver meds with a network of quadrotors. [Online]. Available: http://spectrum.ieee.org/automaton/robotics/medical-robots/mini-uavs-could-be-the-cheapest-way-to-deliver-medicine

[20] J. A. Nelder and R. Mead, “A simplex method for function minimization,” *The Computer Journal*, vol. 7, no. 1, pp. 318 – 313, January 1965.

[21] J.-C. Latombe, *Robot Motion Planning.* Kluwer Academic Publishers, 1991.

[22] G. Chen and F. Lewis, “Synchronizing networked lagrangian systems via binary control protocols,” in *IFAC World Congress*, August-September 2011.

[23] A. Isidori, *Nonlinear Control Systems*, 3rd ed. Springer, 2001.

[24] S. Sastry, *Nonlinear Systems-Analysis, Stability and Control.* Springer, 1999.

[25] J. Slotine and W. Li, *Applied Nonlinear Control.* Prentice-Hall, 1991.

[26] N. Hovakimyan and C. Cao, $\mathcal{L}_1$ *Adaptive Control Theory - Guaranteed Robustness with Fast Adaptation*, ser. Advances in Design and Control. SIAM, 2010.

[27] D. P. Bertsekas, *Dynamic Programming and optimal control, Vol. I*, 3rd ed. Belmont, Massachusetts: Athena Scientific, 2005.

[28] R. E. Bellman, *Dynamic Programming.* Princeton University Press, New Jersey, 1957.

[29] R. M. L. T. G. Kolda and V. Torczon, “Optimization by direct search: New perspectives on some classical and modern methods,” *SIAM Review*, vol. 45, no. 3, pp. 385 – 482, 2003.

[30] M. J. D. Powell, “Direct search algorithms for optimization calculations,” *Acta Numerica, Cambridge University Press*, pp. 287 – 336, 1998.

[31] L. Busoniu, R. Babuska, B. de Schutter, and D. Ernst, *Reinforcement Learning and Dynamic Programming Using Function Approximators.* CRC Press, Taylor & Francis Group, 2009.

[32] F. Bullo, J. Cortés, and S. Martínez, *Distributed Control of Robotic Networks*, ser. Applied Mathematics Series. Princeton University Press, 2009, electronically available at http://coordinationbook.info.

[33] I. Palunko and B. S., “Small helicopter control design based on model reduction and decoupling,” *Journal of Intelligent and Robotic Systems*, vol. 54, no. 1-3, pp. 201–228, 2009.

[34] A. R. S. Bramwell, G. Done, and D. Balmford, *Bramwell helicopter dynamics.* Oxford: Butterworth-Heinemann, 2001.

[35] H. Huang, G. Hoffmann, S. Waslander, and C. Tomlin, “Aerodynamics and control of autonomous quadrotor helicopters in aggressive maneuvering,” in *IEEE International Conference on Robotics and Automation*, may 2009, pp. 3277 – 3282.

[36] P. Pounds, R. Mahony, and P. Corke, “Modelling and control of a large quadrotor robot,” *IFAC Control Engineering Practice*, vol. 18, no. 7, pp. 691 – 699, 2010.

[37] S. Bouabdallah, P. Murrieri, and R. Siegwart, “Towards autonomous indoor micro VTOL,” *Autonomous Robots*, vol. 18, pp. 171–183, March 2005.

[38] I. Palunko and R. Fierro, “Adaptive feedback controller design and quadrotor modeling with dynamic changes of center of gravity,” in *IFAC World Congress*, August-September 2011.

[39] E. S. N. P. Pierre-Jean Bristeau, Philippe Martin, “The role of propeller aerodynamics in the model of a quadrotor uav,” in *IEEE European Control Conference*, Budapest, Hungary, 2009, pp. 683–688.

[40] I. Palunko, R. Fierro, and C. Sultan, “Nonlinear modeling and output feedback control design for a small-scale helicopter,” in *IEEE Mediterranean Conference on Control and Automation*, June 2009, pp. 1251 –1256.

[41] G. Arfken, *Mathematical Methods for Physicists*, 3rd ed. Academic Press, 1985.

[42] S. Sagatun and T. Fossen, “Lagrangian formulation of underwater vehicles’ dynamics,” in *Conference Proceedings of Systems, Man, and Cybernetics*, vol. 2, Charlottesville, VA, October 1991, pp. 1029 – 1034.

[43] L. S. Cicolani and G. Kanning, “Equations of motion of slung-load systems, including multilift systems,” NASA Technical Paper, 1992.

[44] R. Stuckey, “Mathematical modelling of helicopter slung-load systems,” Techical report, December 2001.

[45] D. Fusato, G. Guglieri, and R. Celi, “Flight dynamics of an articulated rotor helicopter with an external slung load,” *Journal of the American Helicopter Society*, vol. 46, no. 1, pp. 3–14, January 2001.

[46] M. Bisgaard, J. Bendtsen, and A. la Cour-Harbo, “Modelling of a generic slung load system,” in *AIAA Modeling and Simulation Technologies Conference and Exhibit*, 2006.

[47] A. Das, K. Subbarao, and F. Lewis, "Dynamic inversion with zero-dynamics stabilization for quadrotor control," *IET Control Theory and Applications*, vol. 3, no. 3, pp. 303–314, March 2009.

[48] S. Bouabdallah and R. Siegwart, "Backstepping and sliding-mode techniques applied to an indoor micro quadrotor," in *IEEE International Conference on Robotics and Automation*, April 2005, pp. 2247 – 2252.

[49] A. Bemporad, C. A. Pascucci, and C. Rocchi, "Hierarchical and hybrid model predictive control of quadcopter air vehicles," in *IFAC Conference on Analysis and Design of Hybrid Systems*, vol. 3, no. 1, April 2009, pp. 14 – 19.

[50] R. Garcia and K. Valavanis, "The implementation of an autonomous helicopter testbed," *Journal of Intelligent and Robotic Systems*, vol. 54, pp. 423–454, 2009.

[51] P. Abbeel, A. Coates, and A. Ng, "Autonomous helicopter aerobatics through apprenticeship learning," *International Journal of Robotics Research*, vol. 29, 2010.

[52] V. Gavrilets, E. Frazzoli, B. Mettler, M. Piedmonte, and E. Feron, "Aggressive maneuvering of small autonomous helicopters: A human-centered approach," *International Journal of Robotics Research*, vol. 20, no. 10, pp. 795–807, 2001.

[53] T. Dierks and S. Jagannathan, "Output feedback control of a quadrotor uav using neural networks," *IEEE Transactions on Neural Networks*, vol. 21, no. 1, pp. 50–66, January 2010.

[54] H. K. Khalil, *Nonlinear Systems*, 3rd ed. Prentice Hall, 2002.

[55] M. Fliess, J. Levine, P. Martin, and P.Rouchon, "Flatness and defect of nonlinear systems:Introductory theory and examples," CAS internal report A-284, January 1994.

[56] I. Palunko, P. Cruz, and R. Fierro, "Agile load transportation: Safe and efficient load manipulation with aerial robots," *IEEE Robotics and Automation Magazine*, vol. 19, no. 9, pp. 69–80, September 2012.

[57] P. Pounds, D. Bersak, and A. Dollar, "Stability of small-scale uav helicopters and quadrotors with added payload mass under pid control," *Autonomous Robots*, vol. 33, pp. 129–142, 2012.

[58] G. Starr, J. Wood, and R. Lumia, "Rapid transport of suspended payloads," in *IEEE International Conference on Robotics and Automation*, April 2005, pp. 1394 – 1399.

[59] D. Zameroski, G. Starr, J. Wood, and R. Lumia, "Rapid swing-free transport of nonlinear payloads using dynamic programming," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 130, no. 4, July 2008.

[60] L. Cooper and M. W. Cooper, *Introduction to Dynamic Programming.* Pergamon Press, 1982.

[61] R. e. a. Robinett, *Flexible Robot Dynamics and Control*, 3rd ed. Kluwer Academic, Dordrecht, 2002.

[62] (2012, August) MARHES - simulation and experimetal videos. [Online]. Available: http://marhes.ece.unm.edu/index.php/Ipalunko:Home

[63] R. Sutton and A. Barto, *A Reinforcement Learning: an Introduction.* MIT: MIT Press, 1998.

[64] L. Buşoniu, R. Babuška, B. De Schutter, and D. Ernst, *Reinforcement Learning and Dynamic Programming Using Function Approximators.* Boca Raton, Florida: CRC Press, 2010.

[65] M. G. Lagoudakis and R. Parr, "Least-squares policy iteration," *Journal of Machine Learning Research*, vol. 4, pp. 1107–1149, December 2003.

[66] J. Ma and W. B. Powell, "A convergent recursive least squares policy iteration algorithm for multi-dimensional markov decision process with continuous state and action spaces," in *IEEE Conference on Approximate Dynamic Programming and Reinforcement Learning*, March 2009.

[67] W. B. Powell and J. Ma, "A review of stochastic algorithms with continuous value function approximation and some new approximate policy iteration algorithms for multi-dimensional continuous applications," *Journal of Control Theory and Applications*, vol. 9, no. 3, pp. 336 – 352, 2011.

[68] I. Palunko, R. Fierro, and P. Cruz, "Trajectory generation for swing-free maneuvers of a quadrotor with supended payload: A dynamic programming approach," in *IEEE International Conference on Robotics and Automation*, St. Paul, MN, USA, May 2012.

[69] J. Cortés and F. Bullo, "Coordination and geometric optimization via distributed dynamical systems," *SIAM Journal on Control and Optimization*, vol. 44, no. 5, pp. 1543–1574, October 2005.

[70] A. Jadbabaie, J. Lin, and A. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 988 – 1001, June 2003.

[71] S. Martinez, J. Cortes, and F. Bullo, "Motion coordination with distributed information," *IEEE Control Systems Magazine*, vol. 27, no. 4, pp. 75–88, August 2007.

[72] R. Olfati-Saber and R. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1520 – 1533, September 2004.