Intelligent Control of a Quadrotor with Suspended Load

by

Arash Mohammadhasani

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Software Engineering and Intelligent Systems

Department of Electrical and Computer Engineering

University of Alberta

Abstract

This thesis targets output tracking problem for payload position and quadrotor yaw in an slung load system (SLS). In spite of its relatively extensive literature, full SLS control is still a challenging problem since its dimension, nonlinearity, and multiple sources of disturbances are not easy to handle using available nonlinear control tools. Based on the inherent property of SLS having a flat output, in this thesis, we tried to solve the robust output tracking using tools from differential geometry and reinforcement learning (RL).

We first construct a control-affine model for SLS which is novel in that it is written in pendulum coordinates and considers disturbances in pendulum dynamics. Our payload is modelled as a single pendulum attached to the center of mass (CoM) of the unmanned aerial vehicle (UAV) with a spherical joint. Based on this model, we use dynamic extension algorithm (DEA) to reformulate SLS dynamics so that the derived system is feedback linearizable and yet complies with our original output tracking problem. Static state feedback linearization of the augmented system then provides linear time-invariant (LTI) tracking error dynamics in the linearizing state coordinates. These dynamics are exponentially stable on a well-defined and practical region of state space for constant disturbances. Moreover, we provide a Maple symbolic script which can be used to apply DEA to general nonlinear control-affine systems.

Finally, we employ ideas from RL to further robustify our proposed DEA-based control scheme against disturbances and to fill its weaknesses in model dependency. We cover all required definitions and algorithms and highlight what distinguishes the proposed approach from conventional RL-based control. Closed-loop performance

is validated in simulation and compared with a state-of-the-art method from the literature.

To my wife and daughter

Acknowledgements

This research was made possible by the support of people in the Applied Nonlinear Controls Lab (ANCL). Of special mention would be my supervisor Dr. Alan F. Lynch for guiding me through the steps necessary to make this work a success. As well, this research was enabled in part by support provided by Canada's national advanced research computing services coordinated by the Digital Research Alliance of Canada.

Table of Contents

1 Introduction **1**
- 1.1 Literature Review . . . 2
- 1.2 Thesis Overview . . . 4
- 1.3 Contribution . . . 4

2 Modelling **6**
- 2.1 Quadrotor Dynamics . . . 6
- 2.2 Pendulum Dynamics . . . 9
- 2.3 SLS Dynamics . . . 13

3 Control Using Dynamic Extension Algorithm **15**
- 3.1 Feedback linearizability of the Slung Load System . . . 16
- 3.2 Dynamic Extension Algorithm . . . 17
- 3.3 Dynamic Full State Feedback Linearization of the Slung Load System 19
  - 3.3.1 Static State Feedback Linearization for Robust Output Tracking 20
  - 3.3.2 Convergence Proof for Constant Disturbances . . . 22
  - 3.3.3 Domain of the Output Tracking Controller . . . 23
- 3.4 Dynamic Extension Algorithm in Symbolic Math . . . 23
- 3.5 Simulations . . . 26

4 Reinforcement Learning Control **33**
- 4.1 Reinforcement Learning . . . 33
  - 4.1.1 Reinforcement Learning Preliminaries . . . 34
  - 4.1.2 Neural Actor-Critic . . . 36
  - 4.1.3 Policy Gradient Theorem . . . 39
  - 4.1.4 Proximal Policy Optimization . . . 40
- 4.2 SLS Output Tracking as an RL Problem . . . 43
- 4.3 Simulations . . . 46
  - 4.3.1 Hyperparameter Tuning . . . 47
  - 4.3.2 Training . . . 48
  - 4.3.3 Validation . . . 48

5 Conclusion and Future Work **56**
- 5.1 Conclusion . . . 56
- 5.2 Future Work . . . 57

Bibliography **58**

A Pendulum Disturbance **62**

List of Tables

3.1 System parameters. 26

4.1 Hyperparameters used in the proposed proximal policy optimization (PPO) algorithm 47

List of Figures

2.1 Model of Quadrotor 7
2.2 Slung load system (SLS) system. The suspended load is modelled as a spherical pendulum attached to the unmanned aerial vehicle (UAV) center of mass (CoM). 10

3.1 Controller Structure 16
3.2 Dynamic extension algorithm (DEA) Control structure: Nonlinear control affine system is dynamically controlled using state feedback x, extended dynamics (ξ), and reference signal y_d. 18
3.3 output tracking controller designed for the SLS using DEA algorithm 21
3.4 First page of the developed Maple script [1]. Although four sample systems are shown here, users can insert any control-affine model. . . 24
3.5 Function selection in designed maple script [1]. 25
3.6 Intermediary dynamics resulted from applying the developed script [1] to the SLS. 25
3.7 Third step of applying the developed script [1] to the SLS. 26
3.8 Position trajectories for the proposed design (FL), and the design in [2] (GC). The same gains are used for both simulations. 28
3.9 Error trajectories for the tracking task. 29
3.10 Inputs for the tracking task. 30
3.11 Configuration variables for the tracking task. 30
3.12 Position trajectories for the tracking task. 31
3.13 Hover stabilization error in presence of constant disturbance. 31
3.14 Inputs $\bar{u}, \tau$ for the hover stabilization task in presence of constant disturbance. 32
3.15 Configuration variables for the hover stabilization task in presence of constant disturbance. 32

4.1 Different machine learning tasks and their associated categories [3]. . 34
4.2 The agent–environment interaction in a Markov decision process. . . 35
4.4 Reinforcement learning (RL)-based linearizing controller for the SLS that consists of a learning part running by proximal policy optimization (PPO) and a nominal part which is built using partial knowledge of the true model. 46
4.5 Heyeprparameter tuning for the proposed RL algorithm (Set1), when pendulum mass is 0.8 of its true value. 50

4.6 Heyeprparameter tuning for the proposed RL algorithm (Set2), when pendulum mass is 0.8 of its true value. 51
4.7 Heyeprparameter tuning for the proposed RL algorithm (Set3), when pendulum mass is 0.8 of its true value. 52
4.8 Training results (average reward) with hyperparameter set (4.31) when quadrotor mass is 0.8 of its true value. 52
4.9 Training results (average steps) with hyperparameter set (4.31) when quadrotor mass is 0.8 of its true value. 53
4.10 Validation result (episode rewards) with hyperparameter set (4.31) when quadrotor mass is 0.8 of its true value. 53
4.11 Validation result (3D positions) with hyperparameter set (4.31) when quadrotor mass is 0.8 of its true value. 54
4.12 Validation result (positions errors) with hyperparameter set (4.31) when quadrotor mass is 0.8 of its true value. 54
4.13 Validation result (positions errors) for constant disturbance vector $d_q = [1, -1.5, -1]^\top$ N, when quadrotor mass is 0.8 of its true value. . 55
4.14 Validation result (sample efficiency) for the case of conventional RL when quadrotor mass is 0.8 of its true value. 55

List of Acronyms

ANCL	Applied Nonlinear Controls Lab
ANN	Artificial Neural Network
CoM	Center of Mass
DDPG	Deep Deterministic Policy Gradient
DEA	Dynamic Extension Algorithm
ES	Exponentially stable
ESC	Electronic Speed Controller
FL	Feedback Linearization
GC	Geometric Controller
LHS	Left Hand Side
LTI	Linear Time-Invariant
MDP	Markov Decision Process
MIMO	Multiple-Input Multiple-Output
ODE	Ordinary Differential Equation
PDE	Partial Differential Equation
PPO	Proximal Policy Optimization
PWM	Pulse Width Modulation
RL	Reinforcement Learning
ROA	Region of Attraction
SITL	Software in the Loop
SLS	Slung Load System
TD3	Twin Delayed DDPG
TRPO	Trust Region Policy Optimization
UAV	Unmanned Aerial Vehicle

Chapter 1

Introduction

Unmanned aerial vehicle (UAV) is a space vehicle that flies without a crew, can be remotely controlled, or can fly autonomously. Over the past three decades, the popularity of UAVs has kept growing at an unprecedented rate. There are currently over 1000 UAV models under development in over 50 countries, serving as indispensable assistants for human operators in a wide variety of military and civil applications [4].

There are three primary types of UAVs: fixed-wing (e.g., airplane), rotary-wing (e.g., helicopter), and multi-rotor (e.g., quadcopter). The multirotor class has drawn particular attention because of its mechanical simplicity and multiple flight envelopes (hover, vertical take-off, and landing, as well as multi-directional flight). Multirotor applications range from their extensive use in photography and surveying to applications like forest fire monitoring. One application that has received interest and attention in both industry and academia and is the focus of this thesis is the use of quadrotors for payload delivery.

Quadrotors can transport loads by attaching them to their rigid bodies or using mechanical or electromagnetic grippers [5, 6]. Approaches like these simplify load attachments and detachments at the cost of reduced maneuverability and load size. Also, regardless of whether the attached load unbalances the UAV or not, when grasping a load, more thrust is required, resulting in higher energy consumption. An alternative approach is an slung load system (SLS), in which the load is suspended from the bottom of the drone by cable(s). A CH-47 Chinook helicopter is a typical example of this type of transportation. What distinguishes SLS from two other approaches is that the separation of the UAV from the payload improves safety and loading capabilities. Moreover, multi-UAV SLSs can also be used to increase lifting capacity and full control of the load pose [7].

However, motion control, even for a single quadrotor carrying a suspended payload, poses multiple challenges. Firstly, dynamics are highly nonlinear and under-

actuated (8-DOF, 4 inputs). Secondly, in cases where the tension in the cable is extremely small or in which the mass of the cable is comparable to that of the suspended load, the mass distribution of the cable must be considered in the dynamics. Nonetheless, if the mass of the cable is continuously distributed, configuration space of infinite dimension would result with partial differential equation (PDE) representing dynamics [8], which in turn complicates the control design. Lastly, there is the coupling between the payload and the vehicle, which can cause dangerous oscillations as described in [9] for full-sized helicopters. A review of the SLS output tracking literature is covered here, along with a comparison to what we will present in this thesis.

1.1 Literature Review

Several criteria could be used to partition the SLS literature.

In terms of modelling, the majority of scholars are focused on using ordinary differential equation (ODE)s to describe SLS dynamics; on the contrary, we can mention the work of [10] where PDEs are used to model the rope flexibility, and [11–13] where authors utilized a chain of serially-connected links with spherical joints to describe the system dynamics. The model adopted in this work is commonly used in the literature and consists of a single pendulum attached to the center of mass (CoM) of the UAV with a spherical joint. The payload is taken as a point mass. This model is accurate when tension in the cable is high and flexibility is negligible. Euler-Lagrange formulas are used to construct the dynamic laws for the pendulum subsystem as well as symbolic mathematics tools in order to derive the equations that resulted.

SLS literature can also be portioned based on controlled output: UAV position [14–17] or payload position [2, 11]. Most literature focuses on UAV position control, but direct control of payload position is more challenging and improves tracking accuracy and transient response. Our proposed method controls payload position and UAV yaw.

Control methods can also be classified on the basis of their control structure. A *full* design is based on the entire SLS dynamics consisting of the coupled UAV and pendulum subsystems [15]. An *inner-outer loop* design is performed on individual subsystems [7, 14, 16, 18]. The analysis of inner-outer loop designs often ignores coupling between the loops. The proposed method in this thesis is a full design and benefits from a complete closed-loop stability analysis.

In terms of robust design, in contrast to a large number of published papers, the literature is remarkably sparse. SLS uncertainties include parametric uncertainties [15, 19] (e.g., unknown payload mass, cable length), external disturbances [16, 20, 21]

(wind and drag forces), and unmodeled dynamics [22, 23] (e.g., pendulum pivot offset from UAV CoM).

Here we describe in more detail how the proposed method compares with some existing SLS control approaches in terms of modelling, control structure, stability results, and considered uncertainties.

A design that considers UAV position as the controlled output is in [15]. Transporting the payload from point to point while reducing swing along the trajectory is the objective of the paper. To control the position, the design method uses an Interconnection and Damping Assignment-Passivity Based Control (IDA-PBC) to increase robustness against parametric uncertainty and unmodeled dynamics. Also, a proportional controller is applied to control the heading angle. To manage the complexity of solving resulted equations, designs are performed using dynamics restricted to longitudinal and lateral planes. The downside, however, is that only stability (as opposed to asymptotic stability) is proved, also the results are restricted to individual channels and no overall proof is provided. In terms of uncertainty, only parametric uncertainties are considered in simulations.

In [14] an inner-outer loop design is employed to control the UAV position. The vehicle translational movements and payload swing are stabilized by nonlinear controllers while a state-dependent differential Riccati equation-based controller is utilized to stabilize rotational dynamics. Two outer loop designs are proven to be asymptotically stable. However, the effect of loop coupling is ignored and, as in [15], to prove stability, a decoupled outer loop is assumed with lateral and longitudinal subsystems. Neither disturbance nor uncertainty is considered.

Work in [16] applies backstepping with integral action and obtains uniform global asymptotic stability for the UAV position tracking error dynamics. As the pendulum position is not directly tracked, an input-shaping technique is employed to minimize pendulum oscillations. The authors did a thorough job in the sense that the proposed design not only considers external disturbances and the effect of air drag (at low speed) but also excludes singularities that are available in Euler-Lagrange-based modelling. It has the disadvantage, however, that the inner-outer loop structure is adopted without investigating the effect of loop coupling on stability.

The article [2] proposes a three-loop control structure whose inner loop tracks the attitude of the UAV. Heading angle and load attitudes are controlled by the middle loop. Lastly, the outermost loop tracks load position. It differs from the previously mentioned papers in that attitude dynamics are modelled using a rotation matrix, and load swing is represented using a 3-dimensional vector. Despite not being minimal, this representation is singularity-free. Regarding control design, it is shown that the entire tracking error dynamics are almost globally exponentially attractive. Exponential stability is proved only locally and the corresponding region

is difficult to determine a priori as it is a function of the system's response. Also, there is no consideration of disturbance or uncertainty.

The work in [20] controls the load position and takes advantage of assumptions that the load is point mass and the cable is massless. Authors have utilized the uncertainty and disturbance estimator (UDE)-based design to withstand the effect of external wind and cruising drag (caused by the inertial velocity of the object). Using the reduction theorem and an added attitude tracking controller in an inner-outer loop design, they were able to ensure the stability of the overall design without requiring a Lyapunov function extension. However, it has the disadvantage of being almost globally asymptotically stable only near the hovering state.

A reinforcement learning (RL)-based controller is proposed in [24]. Similar to [2], the paper uses a singularity-free representation of SLS dynamics and assumes that the cable is taut. However, the design is more like a classical Lyoponouv-based inner-outer loop control in which the gain tuning is left to an RL agent. A deterministic policy gradient agent in an actor-critic structure is rewarded based on how fast the tracking is achieved. The simulation section shows that the proposed design is competitive with energy-based and backstepping designs in terms of minimum cable swing, fast convergence, and disturbance rejection, but the downside is that it is only intended for stabilization, and not tracking.

The focus of this work is to use RL algorithms along with inherent properties of SLS to propose a control structure which is both robust and sample efficient.

1.2 Thesis Overview

Next chapter presents our model of SLS dynamics. In that chapter, we utilize the quadrotor's equations of motion along with pendulum dynamics to provide our overall SLS model in presence of external disturbances.

Our control design consists of two chapters. In the first chapter, we assume we have access to perfect knowledge of the SLS model (no uncertainty), and then design a dynamic extension algorithm (DEA)-based controller to achieve output tracking. Next, we assume uncertainties in the model and use algorithms from RL to robustify our DEA-based controller. Simulations at the end of each sub-chapters are provided to showcase the applicability of proposed controllers.

1.3 Contribution

There are three main contributions to this thesis:

1. Applying dynamic state feedback linearization on SLS[1].

[1]We remark that [2] proves differential flatness of the SLS model. Hence, the SLS is necessarily

2. Providing a maple script to apply DEA on any control-affine system.

3. Implementation of an RL-based linearzing controller on SLS.

We believe that the above contributions are the main ones, although others could be mentioned (e.g., modelling force disturbances on pendulum dynamics, extensive hyperparameter analysis, and Matlab implementations of the proposed RL algorithm using ComputeCanada servers).

dynamically state feedback linearizable. However, this feedback control is not considered in [2]. The proposed method is the only known work which derives this linearizing control.

Chapter 2

Modelling

This chapter focuses on the modelling used for the simulations and theory. In mathematical terms, the objective of this chapter is to provide a model for the quadrotor with suspended load in a control-affine form

$$\begin{aligned} \dot{x} &= f(x) + \sum_{i=1}^{m} g_i(x)u_i + \sum_{i=1}^{p} q_i(x)d_i \\ y_i &= h_i(x), \quad 1 \leq i \leq m \end{aligned} \tag{2.1}$$

with vector fields $f, g_i : \mathcal{M} \to \mathbb{R}^n$ and output functions $h_i : \mathcal{M} \to \mathbb{R}$ defined on an open subset $\mathcal{M} \subseteq \mathbb{R}^n$. In (2.1), $x \in \mathcal{M}$ is the state, and $u = [u_1, \ldots, u_m]^\top \in \mathcal{U} \subset \mathbb{R}^m$ is the control input, and $d = [d_1, \ldots, d_p]^\top \in \mathbb{R}^p$ is a bounded disturbance input.

To achieve this, we first present the quadrotor model, then derive a dynamic model for the pendulum subsystem using Euler-Lagrange principles and the relationship between the quadrotor and load positions. The modelling chapter concludes by combining unmanned aerial vehicle (UAV) and pendulum models to obtain an affine form (2.1) for the slung load system (SLS) system that will be used in the following chapters for control design.

2.1 Quadrotor Dynamics

In this subsection, we consider a quadrotor with a cross configuration and choose a commonly used nonlinear rigid body model for the UAV and follow the presentation in [25]. Nonetheless, a plus configuration gives the same rotational and translational kinematics with only the torque definitions differing.

Quadrotor models are typically described using two frames: the body frame $\mathcal{B}$ and the navigation frame $\mathcal{N}$. Frame $\mathcal{N}$ is assumed inertial, centred at a point on Earth, such as the takeoff point and has orthonormal basis vectors n_1, n_2, n_3. The origin of frame $\mathcal{B}$ is the UAV's center of mass (CoM) and its basis vectors b_1, b_2, b_3.

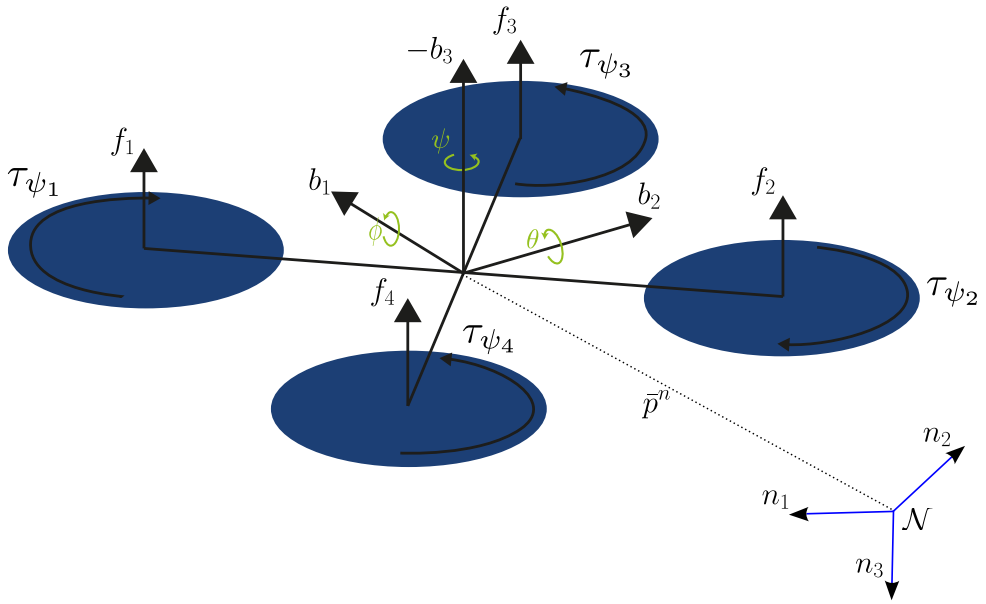


Figure 2.1: Model of quadrotor based on [25]

As shown in Figure 2.1, the frame basis vectors are chosen based on the right-hand rule, with the third axis of each frame pointing downward.

For defining the positive direction of the b_1 axis, there are two possible definitions: a cross configuration where the b_1 axis is between motor 1 and 3, and the plus configuration where b_1 points toward motor 1. The right-hand rule defines roll, pitch, and yaw directions. Accordingly, when viewing the quad from above, the roll will be defined as a positive rotation towards motor 1, the pitch will be positive with the nose pointing up, and the yaw is rotation in the clockwise direction. To transform vectors between frames (2.2) describes the orientation of $\mathcal{B}$ with respect to $\mathcal{N}$. We use roll-pitch-yaw (ϕ-θ-ψ) or ZYX Euler angles (see Figure 2.1) to parametrize R:

$$R_b^n(\eta) = R_3(\psi)R_2(\theta)R_1(\phi) = \begin{bmatrix} \mathrm{c}_\theta\mathrm{c}_\psi & \mathrm{s}_\phi\mathrm{s}_\theta\mathrm{c}_\psi - \mathrm{c}_\phi\mathrm{s}_\psi & \mathrm{c}_\phi\mathrm{s}_\theta\mathrm{c}_\psi + \mathrm{s}_\phi\mathrm{s}_\psi \\ \mathrm{c}_\theta\mathrm{s}_\psi & \mathrm{s}_\phi\mathrm{s}_\theta\mathrm{s}_\psi + \mathrm{c}_\psi\mathrm{c}_\phi & \mathrm{c}_\phi\mathrm{s}_\theta\mathrm{s}_\psi - \mathrm{s}_\phi\mathrm{c}_\psi \\ -\mathrm{s}_\theta & \mathrm{s}_\phi\mathrm{c}_\theta & \mathrm{c}_\phi\mathrm{c}_\theta \end{bmatrix} \tag{2.2}$$

where the rotation matrix $R \in \mathrm{SO}(3) = \{R \in \mathbb{R}^{3\times3} : RR^T = I, \det(R) = 1\}$, $\eta = [\phi, \theta, \psi]^T$, $\mathrm{c}_x = \cos(x)$, and $\mathrm{s}_x = \sin(x)$. The inverse of the rotation matrix is used to convert vectors from the navigation frame to the body frame. The Newton-Euler

formulation is used to get the equations that describe the quadrotor's translational and rotational dynamics.

$$\dot{\bar{p}}^n = \bar{v}^n \tag{2.3a}$$
$$m_q \dot{\bar{v}}^n = -R_b^n \bar{u} n_3 + n_3 g m_q + d_q \tag{2.3b}$$
$$\dot{R}_b^n = R_b^n S(\omega^b) \tag{2.3c}$$
$$J\dot{\omega}^b = -S(\omega^b)J\omega^b + \tau^b \tag{2.3d}$$

where $\bar{p} \in \mathbb{R}^3$ is the position of the UAV, $\bar{v} \in \mathbb{R}^3$ is the linear velocity, $g \in \mathbb{R}$ is the gravitational acceleration, m_q is the mass of the UAV, R is defined in (2.2), $n_3 \in \mathbb{R}^3$ is the third basis vector of $\mathcal{N}$, $\bar{u} \in \mathbb{R}^+$ is the total thrust from four motors, $\omega^b \in \mathbb{R}^3$ is the angular velocity, $J = \text{diag}([J_1, J_2, J_3])$ is the inertia matrix, $\tau^b = [\tau_1, \tau_2, \tau_3]^T \in \mathbb{R}^3$ is the torque due to all propeller thrusts expressed in $\mathcal{B}$, $d_q \in \mathbb{R}^3$ is the disturbance vector due to external wind and/or drag forces, and finally the skew operator $S(\cdot) : \mathbb{R}^3 \to so(3)$ is

$$S(x) = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix}, \quad \text{where } x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

and $so(3) = \{M \in \mathbb{R}^{3\times 3} : M^T = -M\}$.

As shown in Figure 2.1, we assume each propeller generates thrust, $f_i \in \mathbb{R}^3, i = 1, 2, 3, 4$ in the $-b_3$ direction, and we denote the total thrust due to all propellers by the scalar input $\bar{u} \geq 0$, i.e., the total thrust vector is $F = -\bar{u}b_3 = -R_b^n \bar{u} n_3$ and $\|F\| = \bar{u}$. Although not required for the control design, inputs $\bar{u}, \tau$ can be mapped to the physical input to the UAV which are pulse width modulation (PWM) signals to the electronic speed controller (ESC) and denoted by $W_i, i = 1, 2, 3, 4$. Controlling individual propeller speeds creates an input torque τ. Following the usual thrust modelling assumptions [25, 26], there is a one-to-one relation between $(\bar{u}, \tau)$ and the physical inputs $W_i, i = 1, 2, 3, 4$. For a UAV in "cross" configuration we have

$$\begin{bmatrix} \bar{u} \\ \tau \end{bmatrix} = \begin{bmatrix} K_u & K_u & K_u & K_u \\ -\text{c}_\Theta K_u \ell & \text{c}_\Theta K_u \ell & \text{c}_\Theta K_u \ell & -\text{c}_\Theta K_u \ell \\ \text{s}_\Theta K_u \ell & -\text{s}_\Theta K_u \ell & \text{s}_\Theta K_u \ell & -\text{s}_\Theta K_u \ell \\ -K_\tau & -K_\tau & K_\tau & K_\tau \end{bmatrix} \begin{bmatrix} \tilde{W}_1^2 \\ \tilde{W}_2^2 \\ \tilde{W}_3^2 \\ \tilde{W}_4^2 \end{bmatrix} \tag{2.4}$$

where K_u is a thrust constant so that the ith rotor has a thrust vector $f_i = -K_u \tilde{W}_i^2 b_3$, ℓ is arm length, K_τ is a counter torque constant, $\tilde{W}_i$ is a normalized PWM signal defined by $\tilde{W}_i = (W_i - W_{\min})/(W_{\max} - W_{\min})$ where $W_{\min}$ and $W_{\max}$

are the minimum and maximum values of W_i, respectively, and Θ is the angle the arm makes with the b_1 axis. Therefore, the input $\tilde{W}_i \in [0, 1]$ is normalized. Thrust model (2.4) clearly shows that $\bar{u}$ and τ are bounded. Relation (2.4) can be thought of as a "mixer" and can be found in standard autopilot firmware such as PX4 [27]. As is customary, design is performed for u

$$u = [\bar{u}, \tau^\top]^\top \tag{2.5}$$

Looking at quadrotor dynamical equations (2.3) the translational dynamics are described by (2.3a) and (2.3b), while the rotational dynamics by (2.3c) and (2.3d). For the rotational dynamics, the Euler rates can be found in terms of the angular velocity ω by expanding and simplifying (2.3c)

$$\dot{\eta} = W(\eta)\omega^b \tag{2.6}$$

with

$$W(\eta) = \begin{bmatrix} 1 & s_\phi t_\theta & c_\phi t_\theta \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi / c_\theta & c_\phi / c_\theta \end{bmatrix}$$

where $t_\theta = \tan\theta$. It can be seen in (2.6) that there is a singularity at $\theta = \pm 90°$; therefore, this state will need to be specially handled when using this parameterization.

We use the model (2.6) (and not (2.3c)) since it results in a minimal model (with a minimum number of states), which is required for a feedback linearization design (see Chapter 3).

2.2 Pendulum Dynamics

The pendulum dynamics form a coupled subsystem whose configuration space is $\mathbb{R}^3 \times \mathbb{S}^2$ for the pendulum position p and pendulum orientation parameterized by angles α, β. The angle of the pendulum relative to n_1 is α, and β is the angle relative to n_2 (see Figure 2.2). Hence, we take the configuration variable $q = [p^T, \alpha, \beta]^\top$.

We define the position of the pendulum relative to the navigation frame as

$$p = \bar{p} + R_x(\alpha)R_y(\beta)[0, 0, L]^T = \bar{p} + L[\mathrm{s}_\beta, -\mathrm{s}_\alpha \mathrm{c}_\beta, \mathrm{c}_\alpha \mathrm{c}_\beta]^T \tag{2.7}$$

where L denotes the cable length, and the position of the pendulum relative to the origin of $\mathcal{N}$ is denoted by $p \in \mathbb{R}^3$. For simplicity, the mass of the pendulum m_p is assumed concentrated at its endpoint and the rod is assumed massless. In (2.7) we can think of rotating a frame attached to the pendulum $\mathcal{P}$ which is initially aligned

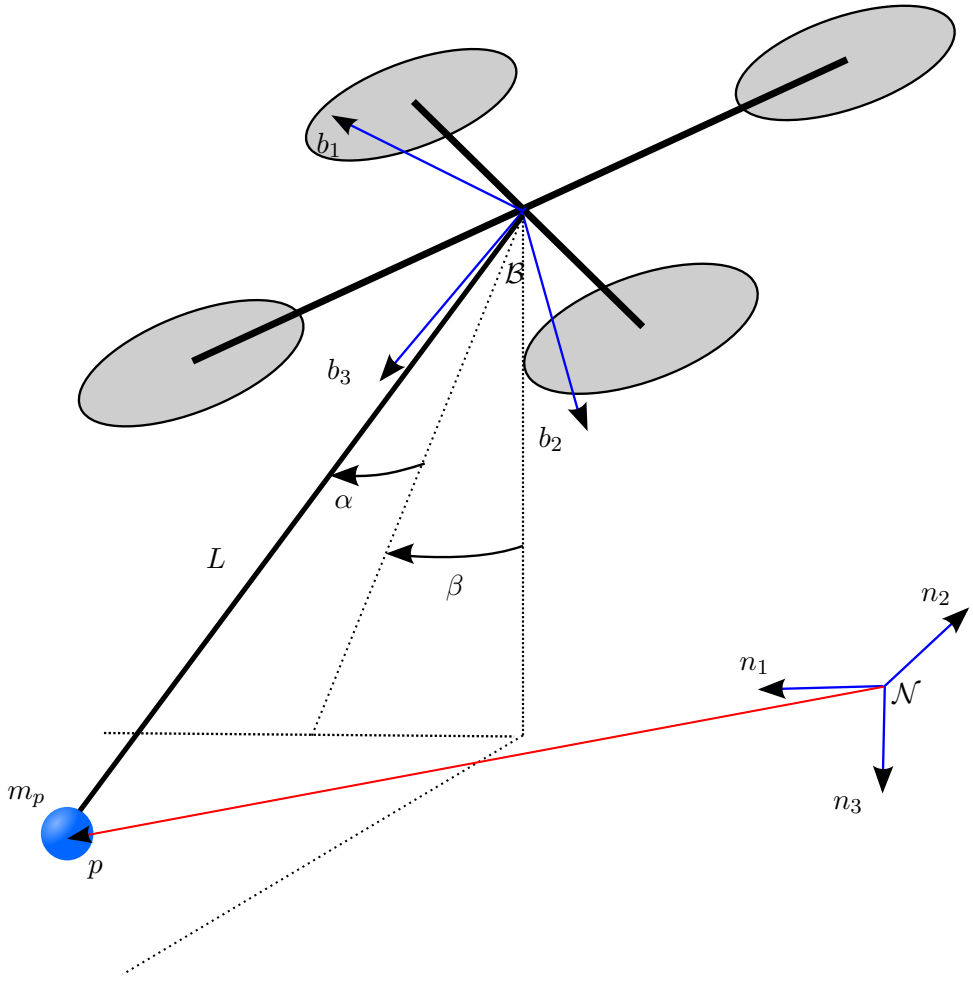


Figure 2.2: SLS system. The suspended load is modelled as a spherical pendulum attached to the UAV CoM.

with $\mathcal{N}$, then rotating about n_2 by an angle β and then followed by a rotation about n_1 by an angle α. Hence, the rotation between $\mathcal{N}$ and $\mathcal{P}$ is

$$R_p^n = R_x(\alpha)R_y(\beta) \tag{2.8}$$

where R_p^n maps points expressed in $\mathcal{P}$ to $\mathcal{N}$. Note that since the rotations are performed relative to the fixed frame $\mathcal{N}$, we perform successive rotations by premultiplying in (2.8). In particular, R_p^n maps $[0,0,L]^T$ expressed in $\mathcal{P}$ into $L[\mathrm{s}_\beta, -\mathrm{s}_\alpha\mathrm{c}_\beta, \mathrm{c}_\alpha\mathrm{c}_\beta]^\top$ in frame $\mathcal{N}$.

We make the common assumption that the pendulum pivots about the UAV CoM and therefore does not exert torque on the UAV. This implies the rotational dynamics of the UAV (2.3c) (and hence (2.6)) and (2.3d) are decoupled from the pendulum dynamics.

Therefore, the kinetic energy of the system is

$$K(q,\dot{q}) = \frac{1}{2}(m_q\|\dot{\bar{p}}\|^2 + m_p\|\dot{p}\|^2) \tag{2.9}$$

And the potential energy of the system is given by

$$V(q) = -n_3 m_q g\bar{p} - n_3 m_p g(\bar{p} + L\mathrm{c}_\alpha\mathrm{c}_\beta) \tag{2.10}$$

We combine (2.9) and (2.10) and replace $\bar{p}$ with p from (2.7), and define the Lagrangian

$$L(q,\dot{q}) = K(q,\dot{q}) - V(q) \tag{2.11}$$

Where $q = [p^T, \alpha, \beta]^\top$

Substituting L into the Euler-Lagrange equations

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_j} - \frac{\partial L}{\partial q_j} = \bar{\tau}_j, \quad 1 \le j \le 5 \tag{2.12}$$

where $\bar{\tau}_j$ are generalized external torques, we obtain the standard 2nd order mechanical system form

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \frac{m_p}{m}([(-\bar{u}Rn_3)^\top, 0, 0]^\top + [d_q^\top, 0, 0]^\top) \tag{2.13}$$

Where M is a positive definite mass matrix

$$M(q) = \left[\begin{array}{c|cc} & 0 & -Lm_q\mathrm{c}_\beta \\ (m_q+m_p)I_3 & Lm_q\mathrm{c}_\beta\mathrm{c}_\alpha & -Lm_q\mathrm{s}_\beta\mathrm{s}_\alpha \\ & Lm_q\mathrm{s}_\alpha\mathrm{c}_\beta & Lm_q\mathrm{s}_\beta\mathrm{c}_\alpha \\ \hline & L^2m_q\mathrm{c}_\beta^2 & 0 \\ * & 0 & L^2m_q \end{array}\right] \tag{2.14}$$

In which the submatrix denoted by * can be determined by symmetry. $C\dot{q}$ models the centrifugal and Coriolis effects, and its (k, j) element can be obtained from

$$C_{kj} = \sum_{i=1}^{5} \left(\frac{\partial M_{kj}}{\partial q_i} + \frac{\partial M_{ki}}{\partial q_j} + \frac{\partial M_{ij}}{\partial q_k} \right) \dot{q}_i$$

Where M_{kj} is the (k, j) element of M. This yields

$$C(q, \dot{q}) = -Lm_q \left[\begin{array}{c|cc} & 0 & -\mathrm{s}_\beta \dot{\beta} \\ & \mathrm{c}_\alpha \mathrm{s}_\beta \dot{\beta} + \mathrm{c}_\beta \mathrm{s}_\alpha \dot{\alpha} & \mathrm{s}_\beta \mathrm{c}_\alpha \dot{\alpha} + \mathrm{c}_\beta \mathrm{s}_\alpha \dot{\beta} \\ 0 & \mathrm{s}_\beta \mathrm{s}_\alpha \dot{\beta} - \mathrm{c}_\beta \mathrm{c}_\alpha \dot{\alpha} & \mathrm{s}_\beta \mathrm{s}_\alpha \dot{\alpha} - \mathrm{c}_\beta \mathrm{c}_\alpha \dot{\beta} \\ & L\mathrm{s}_\beta \mathrm{c}_\beta \dot{\beta} & L\mathrm{s}_\beta \mathrm{c}_\beta \dot{\alpha} \\ & -L\mathrm{c}_\beta \mathrm{s}_\beta \dot{\alpha} & 0 \end{array}\right]$$

G is the gravity vector

$$G(q) = -g \begin{bmatrix} 0 & 0 & (m_q + m_p) & Lm_q \mathrm{c}_\beta \mathrm{s}_\alpha & Lm_q \mathrm{s}_\beta \mathrm{c}_\alpha \end{bmatrix}^\top$$

Pendulum dynamics can then be obtained from (2.13). The drift vector field components are

$$-M^{-1}(C(q, \dot{q})q + G(q)) = \begin{bmatrix} -\frac{\mathrm{s}_\beta(\gamma_\alpha^2 \mathrm{c}_\beta^2 + \gamma_\beta^2) Lm_q}{m_q + m_p} \\ \frac{\mathrm{s}_\alpha \mathrm{c}_\beta(\gamma_\alpha^2 \mathrm{c}_\beta^2 + \gamma_\beta^2) Lm_q}{m_q + m_p} \\ g - \frac{\mathrm{c}_\alpha \mathrm{c}_\beta(\gamma_\alpha^2 \mathrm{c}_\beta^2 + \gamma_\beta^2) Lm_q}{m_q + m_p} \\ 2\gamma_\alpha \gamma_\beta \mathrm{t}_\beta \\ -\gamma_\alpha^2 \mathrm{c}_\beta \mathrm{s}_\beta \end{bmatrix} \tag{2.15}$$

and the input vector field components are

$$M^{-1}B = \begin{bmatrix} -\frac{w_1 + w_2 s_\beta^2}{m_q + m_p} \\ \frac{(w_3 + w_4) c_\beta}{m_q + m_p} \\ \frac{(w_5 - w_6) c_\alpha c_\beta}{m_q + m_p} \\ \frac{w_7}{Lm_q c_\beta} \\ \frac{w_8 + w_9}{Lm_q} \end{bmatrix} \tag{2.16}$$

where

$$
\begin{aligned}
w_1 &= c_\beta s_\beta \left(\left(c_\alpha c_\theta - s_\alpha s_\theta s_\psi \right) c_\phi + c_\psi s_\alpha s_\phi \right) \\
w_2 &= c_\phi s_\theta c_\psi + s_\phi s_\psi \\
w_3 &= c_\beta \left(\left(c_\alpha^2 s_\theta s_\psi + c_\alpha c_\theta s_\alpha - s_\psi s_\theta \right) c_\phi + c_\psi s_\phi s_\alpha^2 \right) \\
w_4 &= s_\beta s_\alpha \left(c_\phi s_\theta c_\psi + s_\phi s_\psi \right) \\
w_5 &= c_\beta ((s_\alpha s_\theta s_\psi - c_\alpha c_\theta) c_\phi - c_\psi s_\alpha s_\phi) \\
w_6 &= s_\beta (c_\phi s_\theta c_\psi + s_\phi s_\psi) \\
w_7 &= c_\psi c_\alpha s_\phi - c_\phi (s_\theta c_\alpha s_\psi + s_\alpha c_\theta) \\
w_8 &= c_\phi \left(s_\beta \left(s_\alpha s_\theta s_\psi - c_\alpha c_\theta \right) + c_\beta c_\psi s_\theta \right) \\
w_9 &= s_\phi \left(c_\beta s_\psi - c_\psi s_\alpha s_\beta \right)
\end{aligned}
$$

2.3 SLS Dynamics

Pendulum dynamics (2.15)-(2.16) can be combined with quadrotor rotational dynamics (2.3d) and (2.6) to obtain a control-affine model with $m = 4$ inputs of (2.5) and $n = 16$ states. By doing that we obtain

$$
\dot{x} = \begin{bmatrix} v \\ \gamma_\alpha \\ \gamma_\beta \\ W(\eta)\omega \\ -\frac{\mathrm{s}_\beta(\gamma_\alpha^2 \mathrm{c}_\beta^2 + \gamma_\beta^2) L m_q}{m_q + m_p} \\ \frac{\mathrm{s}_\alpha \mathrm{c}_\beta(\gamma_\alpha^2 \mathrm{c}_\beta^2 + \gamma_\beta^2) L m_q}{m_q + m_p} \\ g - \frac{\mathrm{c}_\alpha \mathrm{c}_\beta(\gamma_\alpha^2 \mathrm{c}_\beta^2 + \gamma_\beta^2) L m_q}{m_q + m_p} \\ 2\gamma_\alpha \gamma_\beta \mathrm{t}_\beta \\ -\gamma_\alpha^2 \mathrm{c}_\beta \mathrm{s}_\beta \\ -J^{-1} S(\omega) J \omega \end{bmatrix} + \begin{bmatrix} 0_{8\times1} & 0_{8\times3} \\ \bar{g}(x) & 0_{5\times3} \\ 0_{3\times1} & J^{-1} \end{bmatrix} u + \begin{bmatrix} 0_{8\times1} \\ w_p(x)[d_q^\top, 0, 0]^\top \\ 0_{3\times1} \end{bmatrix} \tag{2.17}
$$

where $x = [p^\top, \alpha, \beta, \eta^\top, v^\top, \gamma_\alpha, \gamma_\beta, \omega^\top]^\top \in \mathbb{R}^{16}$, and we have defined $v = \dot{p}, \gamma_\beta = \dot{\beta}, \gamma_\alpha = \dot{\alpha}$. The vector $w_p \in \mathbb{R}^{5\times5}$ is given below and its derivation is in the appendix A. Finally $\bar{g} = M^{-1}B$ as given in (2.16).

We remark that due to parametrization of the orientation of the UAV and pendulum, (2.17) includes singularities when $\mathrm{c}_\beta = \mathrm{c}_\theta = 0$ and hence we define the domain $\mathcal{M}$ of x as a subset of $\mathbb{R}^{16}$ containing 0 and excluding these points.

$$\mathcal{M} = \{x \in \mathbb{R}^{16} \mid \theta \neq \pm 90^\circ, \beta \neq \pm 90^\circ\} \tag{2.18}$$

From a practical point of view, it is unlikely these singularities are of concern as they involve extreme motion not normally encountered during the safe operation of the SLS. Further, $\mathrm{c}_\beta = 0$ (or $\mathrm{c}_\alpha = 0$) is impossible without the pendulum colliding with the UAV.

Our control objective is to achieve asymptotic tracking of a general smooth reference output y_d for pendulum position and yaw angle. Hence, the output is taken as

$$y = h(x) = [p^\top, \psi]^\top \tag{2.19}$$

The disturbance coefficient term w_p is as follows:

$$w_p = \begin{bmatrix} \frac{L' m_{p-q} c_\beta^2 + m_q L}{m_{qpq} L} & \frac{L' m_{p-q} c_\beta s_\alpha s_\beta}{m_{qpq} L} & -\frac{L' m_{p-q} c_\beta c_\alpha s_\beta}{m_{qpq} L} & 0 & -\frac{c_\beta}{m_q L} \\ * & \frac{-L' m_{p-q} s_\alpha^2 c_\beta^2 + m_{qp}(L-1)}{m_{qpq} L} & \frac{L' m_{p-q} s_\alpha c_\alpha c_\beta^2}{m_{qpq} L} & \frac{c_\alpha}{L m_q c_\beta} & -\frac{s_\alpha s_\beta}{m_q L} \\ * & * & \frac{-L' m_{p-q} c_\alpha^2 c_\beta^2 + m_{qp}(L-1)}{m_{qpq} L} & \frac{s_\alpha}{L m_q c_\beta} & \frac{c_\alpha s_\beta}{m_q L} \\ * & \frac{L' m_{p-q} c_\alpha}{L^2 c_\beta m_{pq}} & \frac{L' m_{p-q} s_\alpha}{L^2 c_\beta m_{pq}} & \frac{m_{qp}}{L^2 c_\beta^2 m_{pq}} & 0 \\ -\frac{L' m_{p-q} c_\beta}{L^2 m_{pq}} & -\frac{L' m_{p-q} s_\beta s_\alpha}{L^2 m_{pq}} & \frac{L' m_{p-q} s_\beta c_\alpha}{L^2 m_{pq}} & * & \frac{m_{qp}}{m_{pq} L^2} \end{bmatrix} \tag{2.20}$$

Where we have defined $m_{qp} = m_q + m_p$, $m_{pq} = m_q \times m_p$, $m_{qpq} = m_{qp} \times m_q$, and $L' m_{p-q} = (L-1)m_p - m_q$ to save the space here.

Evaluation of (2.20) at $\alpha = \beta = 0$ is given by

$$w_{p0} = \begin{bmatrix} \frac{L-1}{m_q L} & 0 & 0 & 0 & -\frac{1}{m_q L} \\ 0 & \frac{L-1}{m_q L} & 0 & \frac{1}{m_q L} & 0 \\ 0 & 0 & \frac{1}{m_q + m_p} & 0 & 0 \\ 0 & \frac{(L-1)m_p - m_q}{L^2 m_q m_p} & 0 & \frac{m_q + m_p}{L^2 m_q m_p} & 0 \\ \frac{-m_p L + m_q + m_p}{L^2 m_q m_p} & 0 & 0 & 0 & \frac{m_q + m_p}{L^2 m_q m_p} \end{bmatrix} \tag{2.21}$$

Where we have used w_{p0} as an abbreviation for $w_p|_{\alpha=\beta=0}$.

(2.21) shows that the disturbance on SLS, as modeled in (2.17), is not of a vanishing type.

Chapter 3

Control Using Dynamic Extension Algorithm

This chapter describes how the slung load system (SLS) kinematics derived in Section 2.3 are used to derive a dynamic extension algorithm (DEA)-based control law for the tracking task. That is, given a smooth bounded desired trajectory for output (2.19), denoted by $y_d(t)$, derive a feedback control law for inputs (2.5) to ensure asymptotic convergence of the tracking errors, i.e.,

$$\|y(t) - y_d(t)\| \to 0, t \to \infty \tag{3.1}$$

Two types of controllers are designed in this thesis and both are based on the idea of feedback linearization (FL) [28, 29] depicted in Figure 3.1. As is shown in the figure, this control design is based on finding an appropriate set of coordinates $z = h(x)$ such that the dynamics of the system in the new coordinate are linear. Control design then is a straightforward task using theorems from linear control. We start this chapter by investigating whether the SLS system as modelled in the previous chapter is statically feedback linearizable. This is followed by presenting the essential component of this chapter; DEA. Using DEA, we develop our first controller assuming a perfect knowledge of the system model is available and then in Section 3.3.1, we robustify the proposed controller using the ''integral of error" technique to reject external disturbances locally. in Section 3.5, we compare our results with one of the state-of-the-art controllers from SLS literature.

The control design of this chapter is further robustified in Chapter 4, where we have utilized ideas from reinforcement learning (RL). The motivation for designing such a controller is to reject disturbances on a larger domain as well as parametric uncertainties.

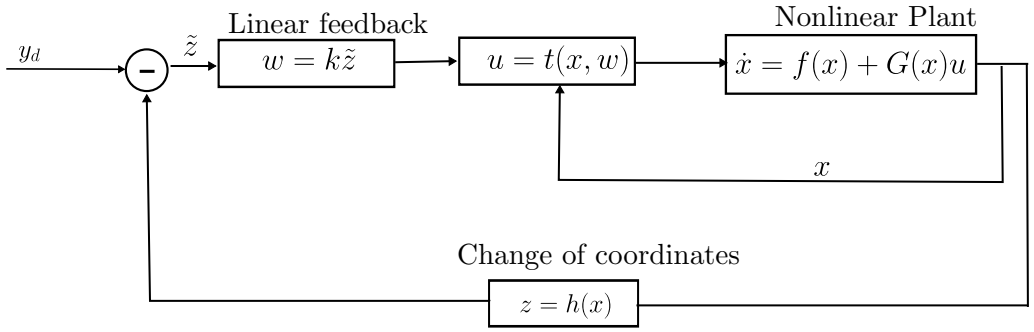


Figure 3.1: Control structure of feedback linearization design in which a change of coordinates converts the nonlinear control task into a linear design.

3.1 Feedback linearizability of the Slung Load System

To simplify the analysis we take the disturbance $d_q = 0_3$ in (2.17). Later when designing the control law in Section 3.3.2 we reintroduce the disturbance.

Consider the system (2.1); the Lie derivative of a function $\lambda : \mathcal{M} \to \mathbb{R}$ along the vector field f is defined by $L_f\lambda(x) = \frac{\partial \lambda}{\partial x} f(x)$. We make use of a vector of indices $r = [r_1, \ldots r_m]$ such that r_i is the largest integer satisfying $L_{g_j} L_f^k h_i(x) = 0, 1 \leq i, j \leq m, k < r_i - 1$,[1] about some $x_0 \in \mathcal{M}$. The existence of these indices does not imply the system has a well-defined relative degree about x_0 as this requires the decoupling matrix

$$A(x) = \begin{bmatrix} L_{g_1} L_f^{r_1 - 1} h_1(x) & \ldots & L_{g_m} L_f^{r_1 - 1} h_1(x) \\ \vdots & \ldots & \vdots \\ L_{g_1} L_f^{r_m - 1} h_m(x) & \ldots & L_{g_m} L_f^{r_m - 1} h_m(x) \end{bmatrix} \tag{3.2}$$

to be nonsingular at x_0.

We know that for the multiple-input multiple-output (MIMO) system (2.1) to be statically feedback linearizable, the above matrix must be nonsingular (see [30] page 220). For the SLS model (2.17), (2.19) we compute $r = [2, 2, 2, 2]$ and

$$A(x) = \begin{bmatrix} a_{11}(x) & 0 & 0 & 0 \\ a_{21}(x) & 0 & 0 & 0 \\ a_{31}(x) & 0 & 0 & 0 \\ 0 & 0 & \frac{\mathrm{s}_\phi}{J_2 \mathrm{c}_\theta} & \frac{\mathrm{c}_\phi}{J_3 \mathrm{c}_\theta} \end{bmatrix} \tag{3.3}$$

[1] We only discuss square systems here, i.e., number of inputs and outputs are the same.

where a_{11}, a_{21}, a_{31} are functions of state. On the region where

$$a_{31}(x) = -\frac{[\mathrm{s}_\beta, -\mathrm{s}_\alpha \mathrm{c}_\beta, \mathrm{c}_\alpha \mathrm{c}_\beta] \cdot Rn_3}{m + m_p} \neq 0 \tag{3.4}$$

we obtain $\mathrm{rank}(A(x)) = 2$. Clearly, relative degree does not exist at any point in $\mathcal{M}$. This implies the SLS cannot be input-output state feedback linearized with output (2.19). However, using the DEA we will construct an extended dynamics with full relative degree (i.e., $\sum_{i=1}^{m} r_i$ for the extended dynamics equals the dimension of the extended state).

Following [31] we outline the DEA for (2.1).

3.2 Dynamic Extension Algorithm

Following we review the DEA steps [30] and superscript variables with $^{\langle i \rangle}$ to keep track of the algorithm iteration.

We begin with Step 1 and assume the decoupling matrix $A^{\langle 0 \rangle} = A$ given by (3.2) has constant rank less than m about $x_0 \in \mathcal{M}$ where $r^{\langle 0 \rangle} = [r_1, \ldots, r_m]$. This implies there exist one or more rows of $A^{\langle 0 \rangle}$ which can be expressed as a linear combination of the others. Let $a_i, 1 \leq i \leq m$ denote the rows of $A^{\langle 0 \rangle}$. We reorder (if necessary) the rows of $A^{\langle 0 \rangle}$ such that its first $k-1$ rows are independent at x_0. The remaining $m-k+1$ rows are dependent and hence a_j for some $j \geq k$ can be written as a linear combination of the first $k-1$ rows. That is, there exists $k-1$ smooth functions $c_1(x), \ldots, c_{k-1}(x)$ such that $a_j(x) = \sum_{i=1}^{k-1} c_i(x) a_i(x)$. Hence, there exists integers $(i_0, j_0), 1 \leq i_0 \leq k-1$ such that c_{i_0} is not identically zero and for Step i we have

$$a_{i_0 j_0}^{\langle i-1 \rangle}(x_0^{\langle i-1 \rangle}) = L_{g_{j_0}^{\langle i-1 \rangle}} L_{f^{\langle i-1 \rangle}}^{r_{i_0}^{\langle i-1 \rangle}-1} h_{i_0}(x_0^{\langle i-1 \rangle}) \neq 0$$

where quantities with $^{\langle i-1 \rangle}$ come from the previous step. That is, $f^{\langle i-1 \rangle}(x^{\langle i-1 \rangle})$, $g_j^{\langle i-1 \rangle}(x^{\langle i-1 \rangle}), 1 \leq j \leq m$ denote the system vector fields from the previous step with $g_j^{\langle 0 \rangle} = g_j$ and $f^{\langle 0 \rangle} = f$ from the original dynamics (2.1). State $x^{\langle i-1 \rangle}$ corresponds to the system from the previous step and $x^{\langle 0 \rangle} = x$ (see (3.6) for $i > 1$). At Step i we define the dynamic state feedback $v^{\langle i \rangle}$ using

$$\begin{aligned} v_j^{\langle i \rangle} &= v_j^{\langle i-1 \rangle}, \quad 1 \leq j \leq m, j \neq j_0 \\ v_{j_0}^{\langle i-1 \rangle} &= \frac{1}{a_{i_0 j_0}^{\langle i-1 \rangle}} (k^{\langle i \rangle} + s^{\langle i \rangle} \xi_i - \sum_{\substack{j=1 \\ j \neq j_0}}^{m} a_{i_0 j}^{\langle i-1 \rangle} v_j^{\langle i-1 \rangle}) \\ \dot{\xi}_i &= v_{j_0}^{\langle i \rangle} \end{aligned} \tag{3.5}$$

where $v^{\langle 0 \rangle} = u$ is the input for (2.1), $\xi_i \in \mathbb{R}$ is a controller state, and $k^{\langle i \rangle}$ and $s^{\langle i \rangle}$ are

any smooth functions such that $k^{\langle i\rangle}(x_0^{\langle i-1\rangle}) = 0$ and $s^{\langle i\rangle}(x_0^{\langle i-1\rangle}) = 1$. At Step i we create a new state vector $x^{\langle i\rangle}$ by concatenating the state of the previous iteration $x^{\langle i-1\rangle}$ with controller state ξ_i:

$$x^{\langle i\rangle} = [x^{\top\langle i-1\rangle}, \xi_i]^\top \in \mathcal{M} \times \mathbb{R}^i, i \geq 1 \tag{3.6}$$

Applying (3.5) to (2.1) defines new system vector fields $f^{\langle i\rangle}(x^{\langle i\rangle}), g_j^{\langle i\rangle}(x^{\langle i\rangle}), 1 \leq j \leq m$ defined on an extended state space $\mathcal{M} \times \mathbb{R}^i$ and with new input $v^{\langle i\rangle} = [v_1^{\langle i\rangle}, \ldots, v_m^{\langle i\rangle}]^\top$. That is,

$$\dot{x}^{\langle i\rangle} = f^{\langle i\rangle}(x^{\langle i\rangle}) + \sum_{j=1}^{m} g_j^{\langle i\rangle}(x^{\langle i\rangle}) v_j^{\langle i\rangle}$$

where, for $i > 0$, $f^{\langle i\rangle}$, and $g^{\langle i\rangle}$ are given by

$$\begin{aligned}
f^{\langle i\rangle} &= [(f^{\langle i-1\rangle} + \frac{k^{\langle i\rangle} + \xi_i s^{\langle i\rangle}}{a_{i_0 j_0}^{\langle i-1\rangle}} g_{j_0}^{\langle i-1\rangle})^\top, 0]^\top \\
g_{j_0}^{\langle i\rangle} &= [0_{1\times(n+i-1)}, 1]^\top \\
g_j^{\langle i\rangle} &= [(g_j^{\langle i-1\rangle} - \frac{g_{j_0}^{\langle i-1\rangle} a_{i_0 j}^{\langle i-1\rangle}}{a_{i_0 j_0}^{\langle i-1\rangle}})^\top, 0]^\top, \quad 1 \leq j \leq m, j \neq j_0
\end{aligned}$$

The DEA iteration proceeds to Step $i+1$ if $A^{\langle i\rangle}$ has constant rank less than m at $[x_0^\top, 0_{1\times i}]^\top$. If rank of $A^{\langle i\rangle}$ is m, the extended dynamics has a well-defined relative degree $r^{<i>}$ and the algorithm terminates.

Achieving a well-defined vector relative degree, one can then follow the same control structure as in Fig.3.1, and control the system using linear design tools. Fig. 3.2 illustrates DEA-based control design assuming the algorithm is terminated at step $^{\langle i\rangle}$.

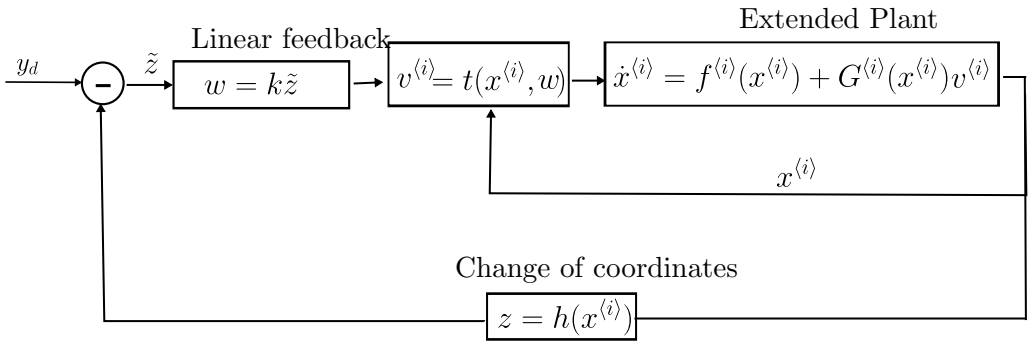


Figure 3.2: DEA Control structure: Nonlinear control affine system is dynamically controlled using state feedback x, extended dynamics (ξ), and reference signal y_d.

3.3 Dynamic Full State Feedback Linearization of the Slung Load System

In this section, we apply the DEA to the SLS model (2.17), (2.19). We show that after four iterations the algorithm converges to extended dynamics with a full relative degree of 20 which is the dimension of the extended state. In Section 3.3.1 we augment the state with the integral of load position and apply a static state feedback linearization to achieve the output tracking control objective.

We pick the point $x_0 = 0$ for practical reasons as we require extended dynamics with a well-defined relative degree in hover. The region on which relative degree is defined includes x_0 and allows for trajectory tracking is discussed further in Section 3.3.3.

Due to the structure of the decoupling matrix (3.2) for the SLS system, we obtain a unique $(i_0, j_0) = (3, 1)$ at each DEA step. In order to simplify the extended system dynamics, the functions k and s are chosen as

$$k^{\langle i\rangle}(x^{\langle i-1\rangle}) = -L_{f^{\langle i-1\rangle}}^{r_3^{\langle i-1\rangle}} h_3(x^{\langle i-1\rangle}), \quad s^{\langle i\rangle}(x^{\langle i-1\rangle}) = 1 \tag{3.7}$$

We remark this choice is important as other choices lead to very large expressions for the extended dynamics which cannot be computed using symbolic math tools in a reasonable amount of time. Using (3.7) we obtain

$$\begin{aligned} f^{\langle i\rangle} &= [(f^{\langle i-1\rangle} + \frac{k^{\langle i\rangle} + \xi_i s^{\langle i\rangle}}{a_{31}^{\langle i-1\rangle}} g_1^{\langle i-1\rangle})^\top, 0]^\top \\ g_1^{\langle i\rangle} &= [0_{1\times(15+i)}, 1]^\top \\ g_j^{\langle i\rangle} &= [g_j^{\langle i-1\rangle\top}, 0]^\top, \text{ for } j = 2, 3, 4 \end{aligned} \tag{3.8}$$

for $i = 1, 2, 3, 4$, where $f^{\langle 0\rangle}(x) = f(x)$ and $g^{\langle 0\rangle}(x) = g(x)$ are from (2.17), and $a_{31}^{\langle i-1\rangle}$ is the $(3, 1)$ entry of $A^{\langle i-1\rangle}$.

As we see from the decoupling matrices

$$A^{\langle i\rangle}(x^{\langle i\rangle}) = \begin{bmatrix} \frac{\mathrm{t}_\beta}{\mathrm{c}_\alpha} & 0 & 0 & 0 \\ -\mathrm{t}_\alpha & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & \frac{\mathrm{s}_\phi}{J_2 \mathrm{c}_\theta} & \frac{\mathrm{c}_\phi}{J_3 \mathrm{c}_\theta} \end{bmatrix}, \quad i = 1, 2, 3$$

The structure of $A^{\langle i\rangle}$ is consistent with $r^{\langle i\rangle} = [i+2, i+2, i+2, 2], 1 \le i \le 3$. At each iteration, we delay the appearance of the thrust input by defining a new controller state. Therefore, it takes another time derivative of the output for the thrust to

appear when using the extended state and extended dynamics.

Using (3.5) we obtain the dynamic state feedback

$$\begin{aligned} v_j^{\langle i \rangle} &= v_j^{\langle i-1 \rangle} = \tau_j, \quad \text{for } j = 2,3,4 \\ v_1^{\langle i-1 \rangle} &= \frac{1}{a_{31}^{\langle i-1 \rangle}} (\xi_i - L_{f^{\langle i-1 \rangle}}^{r_3^{\langle i-1 \rangle}} h_3) \\ \dot{\xi}_i &= v_1^{\langle i \rangle} \end{aligned} \tag{3.9}$$

where $i = 1,2,3,4$. Hence, we observe that only the thrust component of the input is modified.

At Step 4, due to the appearance of τ in the time derivatives of y_1, y_2, we obtain

$$A^{\langle 4 \rangle} = \begin{bmatrix} a_{11}^{\langle 4 \rangle} & a_{12}^{\langle 4 \rangle} & a_{13}^{\langle 4 \rangle} & 0 \\ a_{21}^{\langle 4 \rangle} & a_{22}^{\langle 4 \rangle} & a_{23}^{\langle 4 \rangle} & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & a_{43}^{\langle 4 \rangle} & a_{44}^{\langle 4 \rangle} \end{bmatrix} \tag{3.10}$$

with $\operatorname{rank}(A^{\langle 4 \rangle}(x)) = 4$ on a subset of $\mathcal{M}$ (see (2.18)), we call this set $\mathcal{M}^{\langle 4 \rangle}$ and define it as follows

$$\mathcal{M}^{\langle 4 \rangle} = \{ \mathcal{M} \times \mathbb{R}^4 \subset \mathbb{R}^{20} \mid \operatorname{rank}(A^{\langle 4 \rangle}) = 4 \} \tag{3.11}$$

$\mathcal{M}^{\langle 4 \rangle}$ is described in detail in Section 3.3.3.

Since $r^{\langle 4 \rangle} = [6,6,6,2]$, Step 4 achieves full relative degree of 20 which is the dimension of $x^{\langle 4 \rangle}$.

3.3.1 Static State Feedback Linearization for Robust Output Tracking

In this section, we apply static state feedback linearization to the extended dynamics computed in the previous subsection. Also to improve robustness we augment the extended state with the integral of pendulum position. For convenience we define the input of the extended dynamics as $\bar{\nu} = [v_1^{\langle 4 \rangle}, \tau^\top]^\top$, $\bar{f} = f^{\langle 4 \rangle}, \bar{\mathcal{M}} = \mathcal{M}^{\langle 4 \rangle} \times \mathbb{R}^4 \subset \mathbb{R}^{20}$, $\bar{r} = r^{\langle 4 \rangle}$, and $\bar{A} = A^{\langle 4 \rangle}$.

First, we define tracking error coordinates

$$\begin{aligned} \tilde{z} = [&\int_0^t (h_1 - y_{d1}) d\tau, h_1 - y_{d1}, L_{\bar{f}} h_1 - \dot{y}_{d1}, \dots, \\ &L_{\bar{f}}^5 h_1 - y_{d1}^{(5)}, \dots, \int_0^t (h_3 - y_{d3}) d\tau, \dots, \\ &L_{\bar{f}}^5 h_3 - y_{d3}^{(5)}, h_4 - y_{d4}, L_{\bar{f}} h_4 - \dot{y}_{d4}]^\top \end{aligned} \tag{3.12}$$

We have $\tilde{z} \in \mathbb{R}^{23}$, where 16 components come from the original system dynamics (2.17), four components are added as controller states in the DEA, and three from integral augmentation of the pendulum position.

We can write the error dynamics in $\tilde{z}$-coordinates as

$$\dot{\tilde{z}} = A_c \tilde{z} + B_c(b + \bar{A}\bar{\nu}) \tag{3.13}$$

where

$$\begin{aligned} A_c &= \text{diag}(A_1, A_2, A_3, A_4) \\ B_c &= \begin{bmatrix} e_7 & e_{14} & e_{21} & e_{23} \end{bmatrix} \\ b &= [L_{\bar{f}}^6 h_1, L_{\bar{f}}^6 h_2, L_{\bar{f}}^6 h_3, L_{\bar{f}}^2 h_4]^\top \end{aligned}$$

$e_i \in \mathbb{R}^{23}$ denotes the unit vector in the ith direction, and for $i = 1, 2, 3$ we have

$$A_i = \begin{bmatrix} 0_{6\times 1} & I_6 \\ 0 & 0_{1\times 6} \end{bmatrix} \in \mathbb{R}^{7\times 7}, \text{ and } A_4 = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

Applying the static state linearizing control

$$\bar{\nu} = \bar{A}^{-1}(K\tilde{z} - b + y_d^{(\bar{r})}) \tag{3.14}$$

with $y_d^{(\bar{r})} = [y_{d1}^{(6)}, y_{d2}^{(6)}, y_{d3}^{(6)}, y_{d4}^{(2)}]^\top$ to (3.13) gives

$$\dot{\tilde{z}} = (A_c + B_c K)\tilde{z} \tag{3.15}$$

where $K \in \mathbb{R}^{4\times 23}$ is a control gain chosen such that $A_c + B_c K$ is Hurwitz and suitable tracking error transient performance is obtained.

Designed controller (3.15) for the SLS is depicted in Figure 3.3

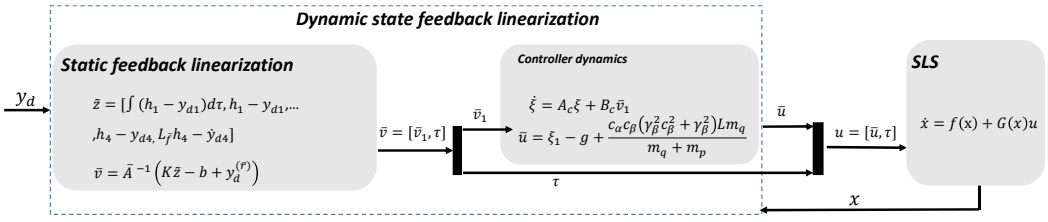


Figure 3.3: output tracking controller designed for the SLS using DEA algorithm

Here we prove that the proposed design can achieve perfect tracking for constant disturbances (e.g., a small neighborhood of α and β) due to the integral augmentation of the state.

3.3.2 Convergence Proof for Constant Disturbances

Here we reintroduce the disturbance term d_q from (2.17) but assume its coefficient w_p is a constant. Also, for simplicity, we neglect the effect of disturbance on pendulum angles (this is the case for long cables ($L \gg 1$), see (2.21)). We show that in this case, the tracking error $y - y_d$ converges exponentially to zero for Hurwitz $A_c + B_c K$.

To begin with, it can be easily shown that[2] (3.15) in terms of coonstant w_p is

$$\dot{\tilde{z}} = (A_c + B_c K)\tilde{z} + E d_q \tag{3.16}$$

Where

$$E = \begin{bmatrix} \frac{L-1}{Lm_q} e_3 & \frac{L-1}{Lm_q} e_{10} & \frac{1}{m_q+m_q} e_{17} \end{bmatrix}$$

For the sake of presentation, we take the practical case of a block diagonal K which implies (3.16) consists of 4 decoupled subsystems. Three are the position errors and are 7-dimensional which forced with input d_{qi}, $i = 1, 2, 3$. The fourth yaw subsystem is 2-dimensional and unforced. Hence, it converges exponentially to zero. Therefore, we focus on the first 3 subsystems. We partition their state $\tilde{z}_i$ into two parts: $\tilde{z}_{ai} = [h_i - y_{di}, \ldots, L_{\bar{f}}^5 h_i - y_{di}^{(5)}]^\top \in \mathbb{R}^6$, $\tilde{z}_{bi} = \int_0^t (h_i - y_{di}) d\tau$, and $\tilde{z}_i = [\tilde{z}_{ai}^\top, \tilde{z}_{bi}]^\top$, $i = 1, 2, 3$. The ith subsystem error dynamics is

$$\begin{bmatrix} \dot{\tilde{z}}_{ai} \\ \dot{\tilde{z}}_{bi} \end{bmatrix} = \begin{bmatrix} A_1 + B_1 K_{1i} & B_1 K_{2i} \\ C_1 & 0 \end{bmatrix} \begin{bmatrix} \tilde{z}_{ai} \\ \tilde{z}_{bi} \end{bmatrix} + \begin{bmatrix} e_2 \\ 0 \end{bmatrix} w_{p0}(i,i) d_q \tag{3.17}$$

where $A_1 \in \mathbb{R}^{6\times 6}$, $B_1 \in \mathbb{R}^{6\times 1}$ are in Brunovky Controller Form, $e_2 = [0, 1, 0, \ldots, 0]^\top \in \mathbb{R}^6$, $K_{1i} \in \mathbb{R}^{1\times 6}$, $K_{2i} \in \mathbb{R}$, and $C_1 = \begin{bmatrix} 1 & 0_{1\times 5} \end{bmatrix}$. Letting $\Lambda_i = A_1 + B_1 K_{1i}$ we compute

$$\begin{bmatrix} \Lambda_i & B_1 K_{2i} \\ C_1 & 0 \end{bmatrix}^{-1} = \begin{bmatrix} \bar{A}_i & \star \\ \star & \star \end{bmatrix}$$

where $\bar{A}_i = \Lambda_i^{-1}(I_6 - B_1(C_1\Lambda_i^{-1}B_1)^{-1}C_1\Lambda_i^{-1})$. Hence, the equilibrium of $y_i - y_{di}$ is

$$\begin{bmatrix} C_1 & 0 \end{bmatrix} \begin{bmatrix} \bar{A}_i & \star \\ \star & \star \end{bmatrix} \begin{bmatrix} e_2 \\ 0 \end{bmatrix} = 0$$

since $C_1 \bar{A}_i = 0$ from

$$C_1 \bar{A}_i = C_1\Lambda_i^{-1} - C_1\Lambda_i^{-1}B_1(C_1\Lambda_i^{-1}B_1)^{-1}C_1\Lambda_i^{-1} = 0$$

[2] To see this, consider (2.21) and note that based on the definition of $\tilde{z}$ in (3.12), disturbances only show up in the velocity channels.

Note that as long as (3.17) is exponentially stable it can be shown that Λ_i and $C_1\Lambda_i^{-1}B_1$ are invertible.

3.3.3 Domain of the Output Tracking Controller

In this section we determine the domain of the linearizing control law (3.14). Since the $\tilde{z}$-coordinates (3.12) are defined using the Lie derivatives of h along $\bar{f}$, some of singular points of the control law are inherited from those in f. These points include $\theta = \pm 90°$, $\beta = \pm 90°$ which are seen in the original dynamics (2.17) and which arise due to the Euler angles. The points $\alpha = \pm 90°$ are singularities that appear in $\bar{f}$ due to the dynamic extension. The expression for $\bar{f}$ is omitted as is it too large. The control is also singular at points where the Jacobian matrix of $\tilde{z}$-coordinates is singular. These points are the same as those where the distribution rank condition in [32] does not hold. Equivalently, we can obtain these points from where the decoupling matrix $\bar{A}$ is singular. These points are

$$\phi = \pm 90° \tag{3.18a}$$

$$[\mathrm{s}_\beta, -\mathrm{s}_\alpha \mathrm{c}_\beta, \mathrm{c}_\alpha \mathrm{c}_\beta] \cdot R n_3 = 0 \tag{3.18b}$$

$$\xi_1 = g - \frac{\mathrm{c}_\alpha \mathrm{c}_\beta (\gamma_\alpha^2 \mathrm{c}_\beta^2 + \gamma_\beta^2) L m}{m + m_p} \tag{3.18c}$$

$$\xi_1 = g \tag{3.18d}$$

We remark that the geometric interpretation of the left hand side (LHS) of (3.18b) (which from (3.4) is a scaling of a_{31}) is the inner product of the direction vector of the pendulum $[\mathrm{s}_\beta, -\mathrm{s}_\alpha \mathrm{c}_\beta, \mathrm{c}_\alpha \mathrm{c}_\beta]$ and the direction of the unmanned aerial vehicle (UAV) thrust vector with both vectors expressed in $\mathcal{N}$. Thus, a physical singularity appears when the direction of the pendulum is perpendicular to the thrust vector. In Step 1 of the DEA we define a controller state $\xi_1 = L_f^2 h_3(x) + a_{31}(x)\bar{u}$ (see (3.9) for $i = 1$). Hence, condition (3.18c) is equivalent to $\bar{u} = 0$. The condition (3.18d) occurs when the downwards linear acceleration of the pendulum $\ddot{p}_3 = g$. This is another physical singularity which appears in many UAV motion controllers, e.g., [33]. Hence, we remark that the domain of the controller is a suitable subset of $\bar{\mathcal{M}}$ which excludes the above-mentioned points. This domain is practically sensible in that singularities only occur in conditions that would not be typical of safe operation.

3.4 Dynamic Extension Algorithm in Symbolic Math

Symbolic math software Maple [34] was used to implement the DEA for a general control-affine system (2.1). The code is available at [1] and was inspired by work in [35]. However, the approach in [35] did not implement the DEA correctly for a

number of reasons. First, it only performed one DEA step. Second, the method for computing indices i_0, j_0 was incorrect. For example, the first and fourth inputs were extended simultaneously in Step 1 when applied to (2.17)-(2.19). Unlike the work of [35] where depreciated packages were used (e.g., `linalg`) our code uses only currently supported packages (e.g., `LinearAlgebra` and their data structures). We use the `CodeGeneration` package to optimize the control law expressions for Matlab simulation. This is important as it allows us to generate an expression for the control in less than a minute on a standard CPU (Intel i9-10900K, 3.70 GHz). As the approach [35] did not perform this optimization, it would yield large expressions which could not be used for simulation.

Another feature of our work is that it allows user input at each DEA step so that functions k and s in (3.5) can be adjusted incrementally. This is important as we have observed that for certain common choices (e.g., $k(x) = 0$ and $s(x) = 1$)), controller expressions are so large that Maple cannot provide a result in a reasonable amount of time. Lastly, our simulations are performed from Maple using Matlab by code generating the simulation files in Maple. Matlab commands are run using Maple's `Matlab` package. The developed code ensures an efficient control design and error-free control law expressions. Four example systems are provided (including the SLS) to demonstrate the toolbox capability. Fig 3.4 shows the first page of the developed script, as it is shown the toolbox is already tested on four different nonlinear systems. Also, Fig 3.5- 3.7 illustrate the selection page for functions k and s in (3.5), intermediary dynamics where users can either continue with their selection or pick different functions, and the third iteration of the DEA on SLS system respectively.

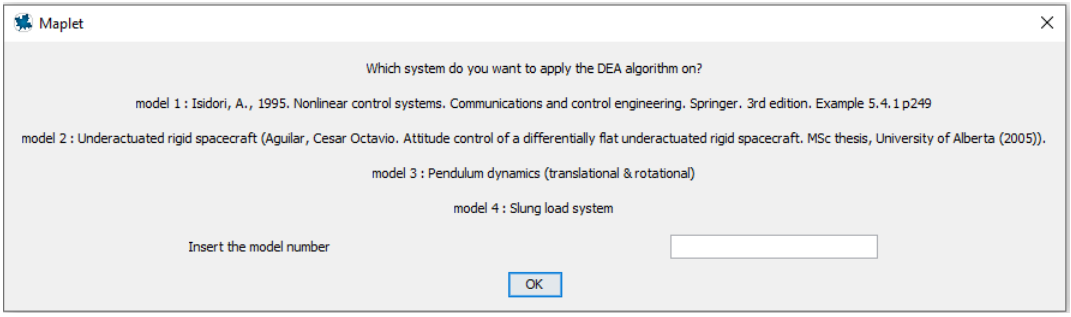


Figure 3.4: First page of the developed Maple script [1]. Although four sample systems are shown here, users can insert any control-affine model.

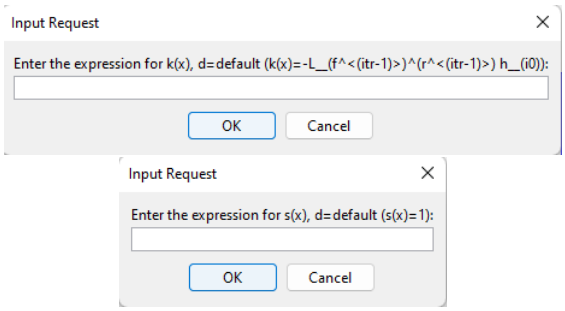


Figure 3.5: Function selection in designed maple script [1].

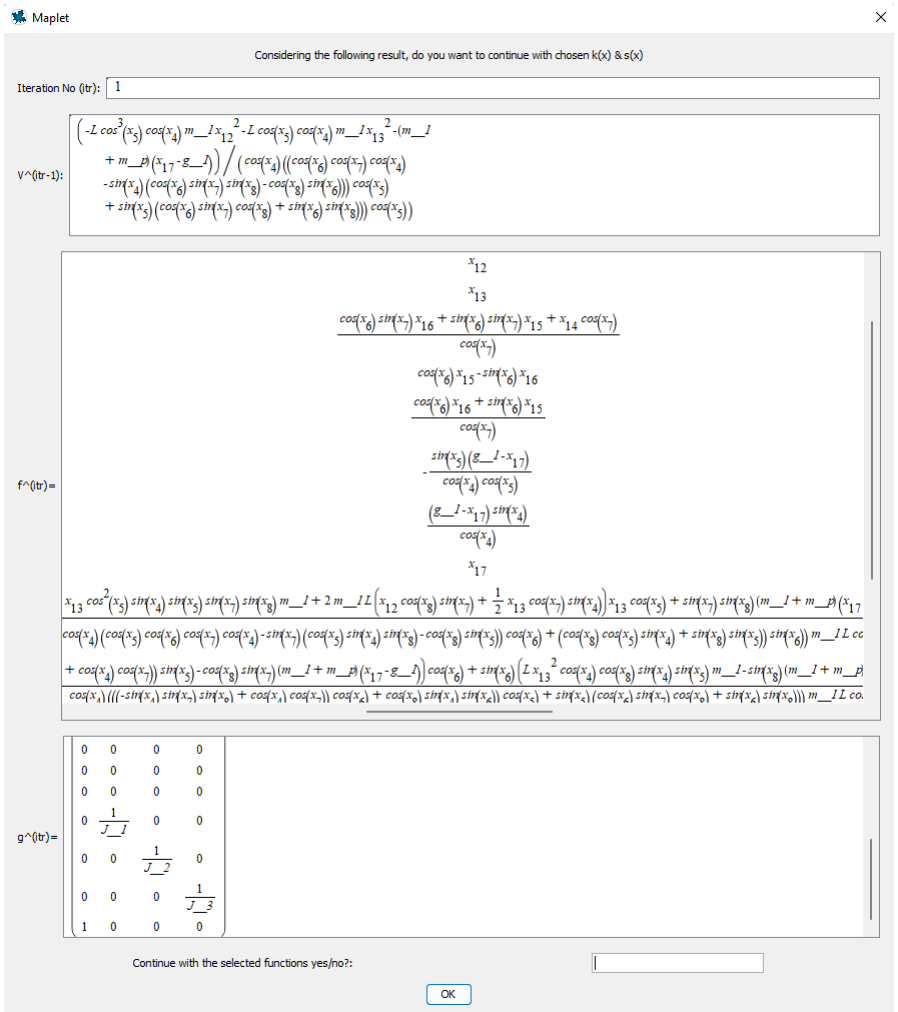


Figure 3.6: Intermediary dynamics resulted from applying the developed script [1] to the SLS.

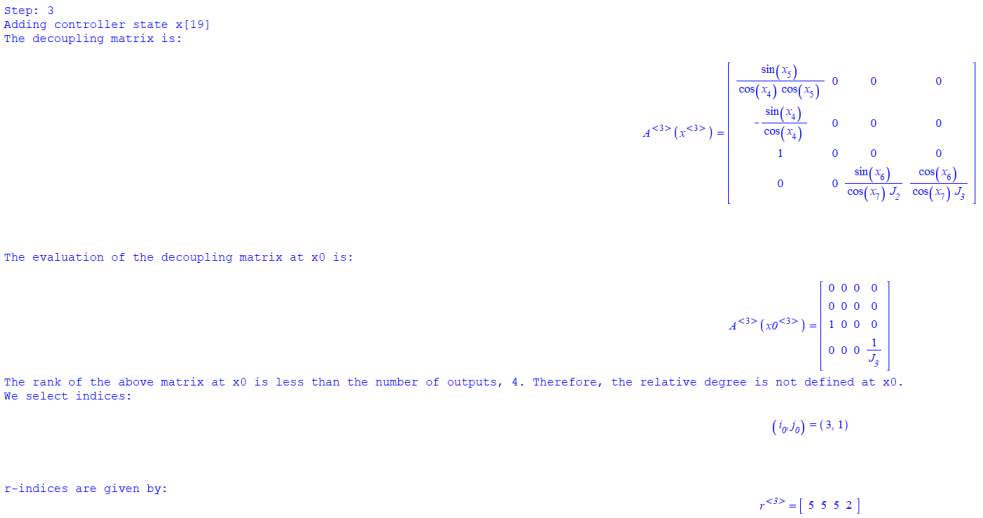


Figure 3.7: Third step of applying the developed script [1] to the SLS.

.

3.5 Simulations

In this section, we validate the proposed control law in simulation and compare its performance with an existing method [2]. We consider simulations for stabilization and tracking of the pendulum position and the yaw angle. The system parameters used in the model and controller are in Table 3.1.

Table 3.1: System parameters.

m_q	$1.6\,\mathrm{kg}$
m_p	$0.16\,\mathrm{kg}$
L	$1\,\mathrm{m}$
J_1	$0.03\,\mathrm{kg}\cdot\mathrm{m}^2$
J_2	$0.03\,\mathrm{kg}\cdot\mathrm{m}^2$
J_3	$0.05\,\mathrm{kg}\cdot\mathrm{m}^2$

We consider three different cases: stabilization (with and without disturbance) and tracking (without disturbance).

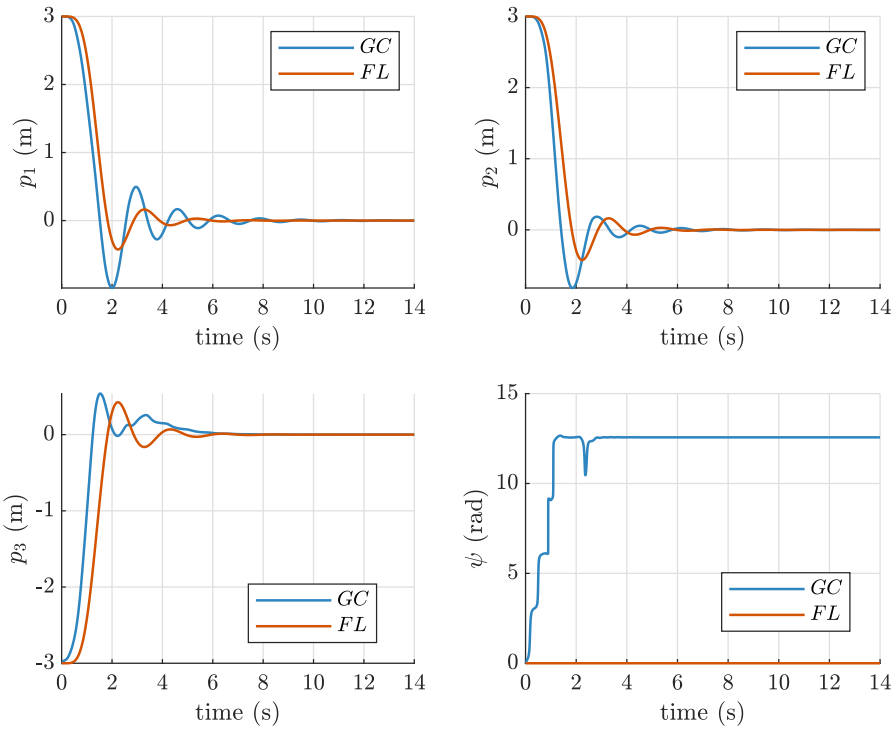
It should be noted that for these simulations, the evaluation of the control law expression takes (at max) 0.6 ms on an Intel i9-10900K CPU running at 3.70 GHz.

Position Stabilization with no Disturbance

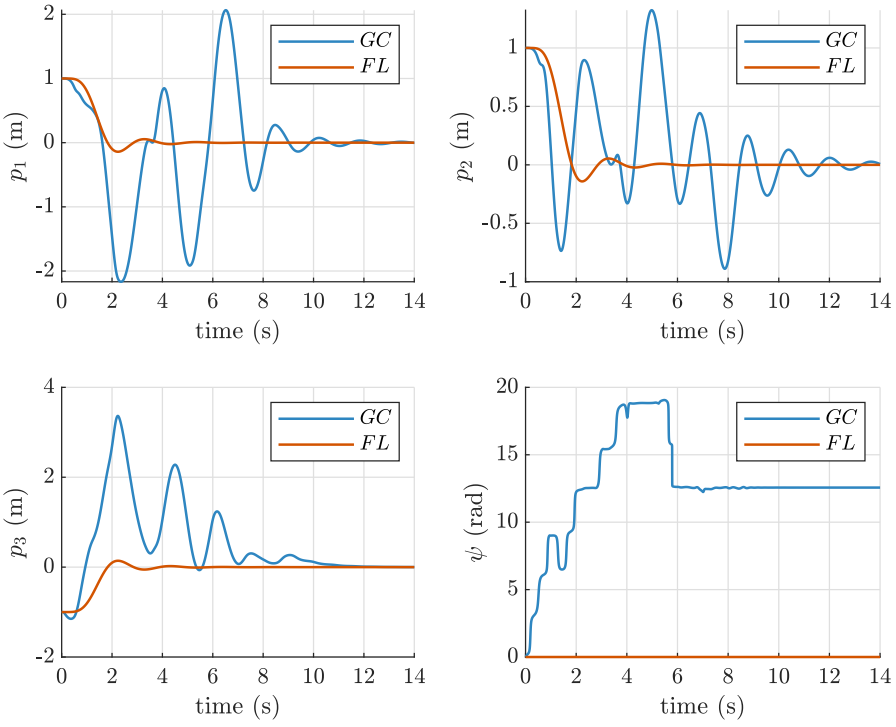
We consider the problem of stabilizing $x = 0$ and choose reference output $y_d = 0$. We initialize the system with $p(0) \neq 0$ and all other states zero (i.e., the SLS is at rest). Since the error dynamics (3.15) are linear time-invariant (LTI), it is straightforward to design the transient performance for y. Using linear system theory we design

K for a 10%-settling time of 3 s and an overshoot of 15%. Fig. 3.8(a) shows the response for p in red where the performance specifications are met as expected. To compare the performance with the proposed design, we consider the same problem using the geometric controller (GC) in [2]. We remark that the GC is a 3-level nested controller and guarantees almost-global *exponential attractiveness* [36, Def. 1], but exponential stability occurs in a limited region. As well, the error dynamics are nonlinear and this makes gain tuning difficult and non-systematic. Hence, the GC gains were chosen by trial-and-error to yield a response that is as close as possible to the desired transient specifications. The trajectories for p are in blue in Fig. 3.8(a).

Since for the proposed design $x = 0$ is exponentially stable (ES), when the initial position is closer to the origin and controller gains are unchanged, we obtain a lower bound on $\|p\|$. Hence, when we initialize $p(0)$ closer to the 0 we expect a smaller transient. This is confirmed in simulation in Fig. 3.8(b). However, since ES is not guaranteed with the GC, the transient performance for y is severely degraded for smaller $p(0)$. We observe that although $\psi(0) = 0$, the UAV performs two full yaw rotations before it converges. As well, the transient performance in position is very oscillatory. The poor transient performance of the GC is attributed to its nonlinear error dynamics whose equilibrium is locally ES. Using [2, Eqns. (39), (45)], for the simulated initial conditions and controller gains, an estimated ES region of attraction (ROA) was computed which shows that ES is only locally ensured and a complex function of controller gain. In particular, the ES ROA does not include $\alpha = \beta = 0$ which implies ES is not ensured for the simulated conditions. This should be compared to the proposed controller whose ES ROA includes $\alpha = \beta = 0$.



(a) Initial position further from origin: $\|p(0)\| = 5.83\,\text{m}$.



(b) Initial position closer to origin: $\|p(0)\| = 2.45\,\text{m}$.

Figure 3.8: Position trajectories for the proposed design (FL), and the design in [2] (GC). The same gains are used for both simulations.

Position tracking with no Disturbance

We consider the desired position

$$y_d(t) = \begin{bmatrix} -1.5\sin(\pi t/4) + 1 \\ 0.75\sin(\pi t/2) \\ 2\sin(\pi t/4) - 3 \\ 0.01t \end{bmatrix} \tag{3.19}$$

Fig. 3.9 shows the tracking error trajectories. We observe the convergence of all components in a reasonable time. The corresponding inputs are given in Fig. 3.10. The trajectories for p, α, β, η are given in Fig 3.11. A 3D position graph of pendulum position after transients decay is given in Figure 3.12 to help visualize the motion.

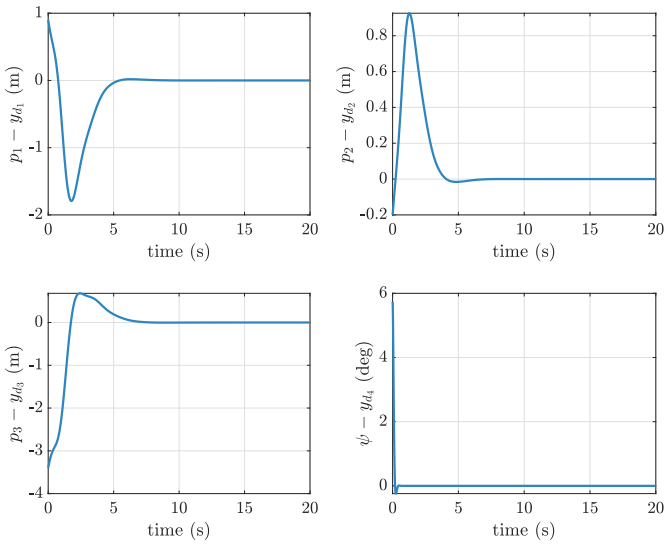


Figure 3.9: Error trajectories for the tracking task.

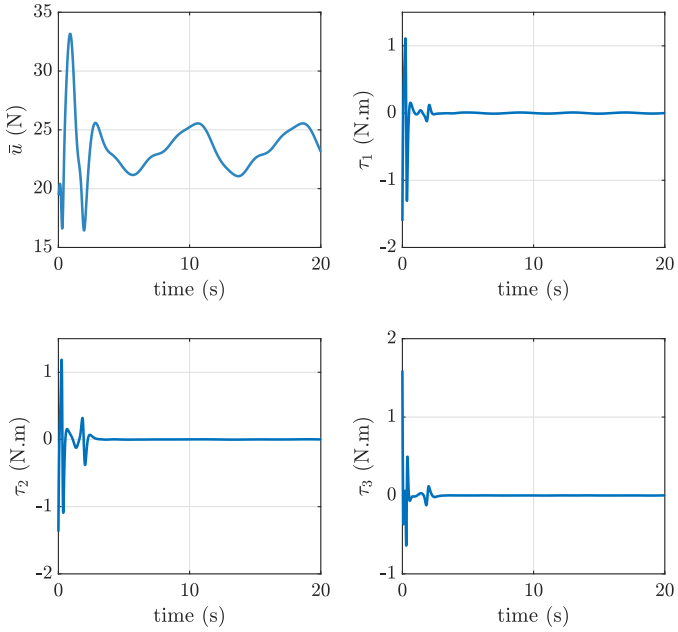


Figure 3.10: Inputs for the tracking task.

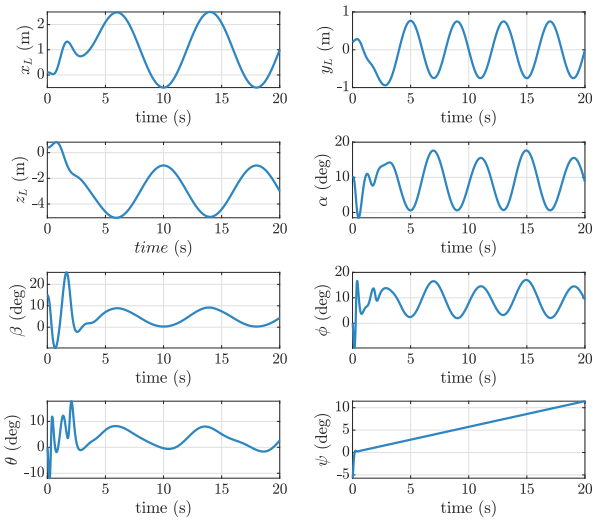


Figure 3.11: Configuration variables for the tracking task.

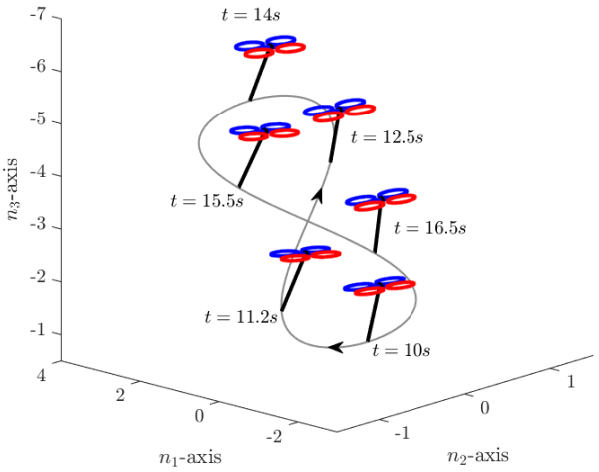


Figure 3.12: Position trajectories for the tracking task.

Position Stabilization with Constant Disturbance

We consider hover as the desired position ($y_d = 0_4$) and disturbance vector $d_q = [1, -1.5, -.2]^\top$ N. Given the ES error dynamics and the integral augmentation technique, we expect convergence which is shown in Fig. 3.13. The inputs are in Fig. 3.14 and p, α, β, η are in Fig. 3.15. As we can see in this figure, pendulum angles α and β converge to 0, however, quadrotor angles ϕ and θ converge to nonzero values which is the expected case to reject the external disturbance.

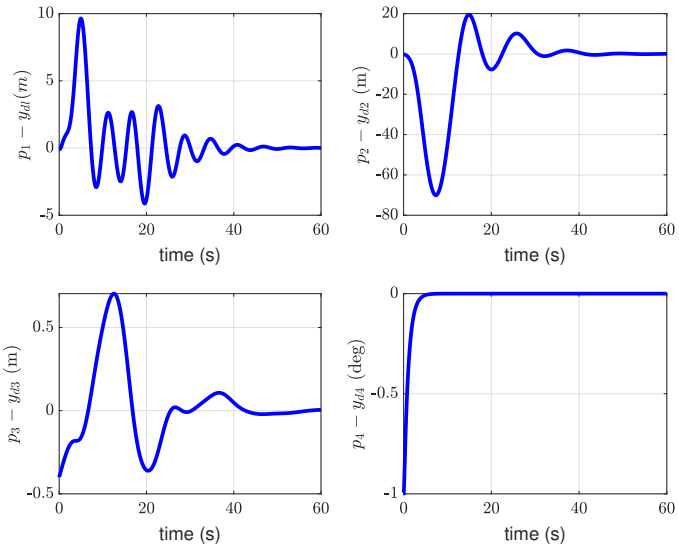


Figure 3.13: Hover stabilization error in presence of constant disturbance.

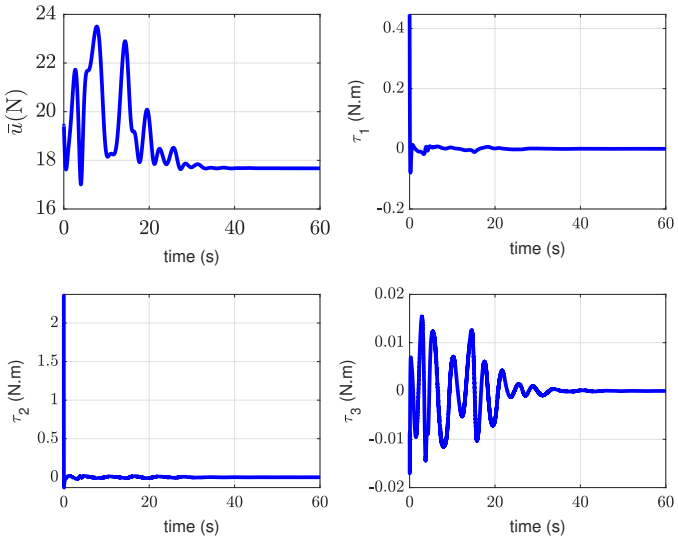


Figure 3.14: Inputs $\bar{u}, \tau$ for the hover stabilization task in presence of constant disturbance.

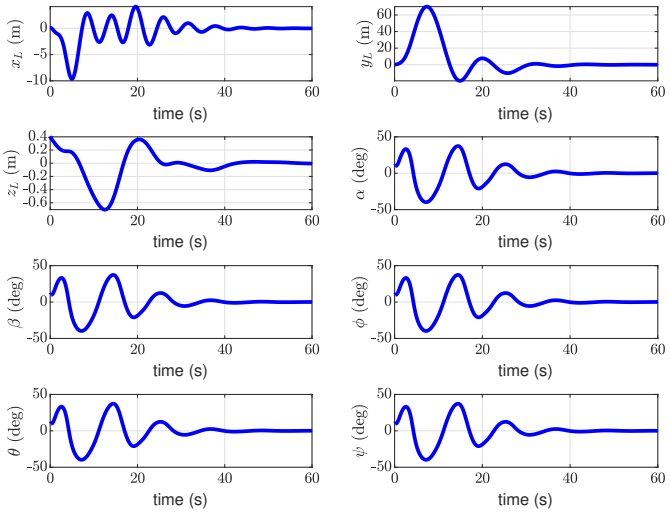


Figure 3.15: Configuration variables for the hover stabilization task in presence of constant disturbance.

Chapter 4

Reinforcement Learning Control

Our goal in this chapter is to control the slung load system (SLS) using designs from reinforcement learning (RL). The purpose of designing such a controller is to reject larger disturbances and parametric uncertainties. We first introduce the necessary elements and algorithms and then formulate our problem within the RL framework. This chapter is concluded by simulating the proposed design on the SLS.

4.1 Reinforcement Learning

The goal of RL is to identify how to map states to actions so that a predefined reward signal is maximized. There are no instructions given to the learner about which actions to take; instead, by trying different actions, one must discover which yields the most cumulative reward.

As stated in [37], learning by reinforcement differs from learning by supervision, which is mostly studied in current machine learning research. In supervised learning, a knowledgeable external supervisor provides labeled examples as a training set. Although this is a fundamental form of learning, it is insufficient for learning from interaction. The reason is that in dynamic problems, it is impossible to find examples of desired behavior that are both correct and representative of all states in which the agent has to act.

Unsupervised learning, which involves discovering structure in unlabeled data collections, is also different from RL. As a result of RL's lack of reliance on examples of correct behavior, one might mistake it for unsupervised learning. However, RL maximizes a reward signal rather than seeking a hidden structure. RL can certainly benefit from uncovering structure in an agent's experiences, but it does not solve the problem of maximizing reward signals on its own. The trade-off between exploration and exploitation is a challenge unique to reinforcement learning [37]. In order to obtain the reward, the agent must *exploit* its previous experience, but it must also

explore new possibilities in order to make better future choices.

Fig. 4.1 attempts to provide a comprehensive representation of available machine learning tasks and their associated categories.

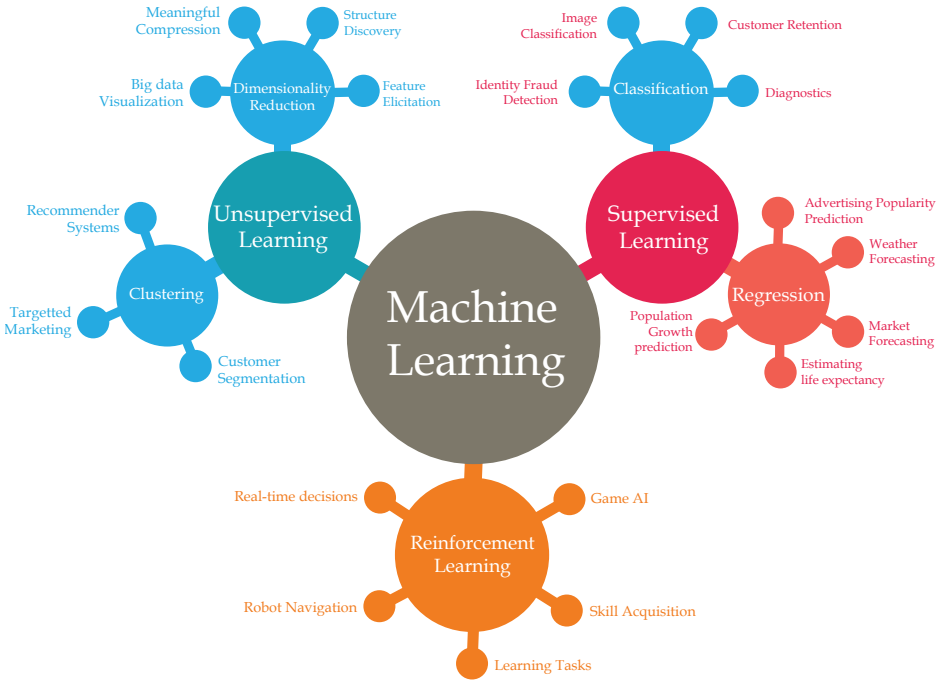


Figure 4.1: Different machine learning tasks and their associated categories [3].

4.1.1 Reinforcement Learning Preliminaries

Closed-loop system in the RL framework is shown in Fig. 4.2. We follow the notation of [37]. The RL *agent* and the *environment* interact during a sequence of discrete time steps $t = 0, 1, 2, \ldots$ as shown in the figure. At each time instant t the RL agent receives information about the environment's *state* $S_t \in \mathcal{S}$ and based on that, selects an *action* $A_t \in \mathcal{A}(s)$. One-time step later, as a result of the action performed in S_t, the RL agent receives a *reward* $R_{t+1} \in \mathcal{R}$, and finds itself in a new state S_{t+1}[1].

Here we assume our problem is of the form of a finite Markov Decision Process (MDP). In this case, the random variables R_t and S_t have well defined discrete probability distributions dependent only on the *preceding* state and action. That is, for particular values of these random variables, $s' \in \mathcal{S}$ and $r \in \mathcal{R}$, there is a probability of those values occurring at time t, given particular values of the preceding state and action:

$$p(s', r|s, a) = \Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\} \tag{4.1}$$

[1]Capital letters for state, reward, and actions emphasizes stochasticity in the environment and agent.

for all $s', s \in \mathcal{S}, r \in \mathcal{R}$ and $a \in \mathcal{A}(s)$.

The definition of a *policy* π is mapping from states to probabilities of each action being taken. The goal of the RL agent is to find a policy that maximizes the total expected reward. We define value functions as follows: The state–value function $v_\pi(s)$ of a state s under a policy π is the expected accumulated reward when starting in s and following policy π. *State-action value* function on the other hand, is the expected return starting from s, taking the action a, and thereafter following policy π. We use the discounted version of these value functions here[2]:

$$v_\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s \right] \tag{4.2a}$$

$$q_\pi(s, a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, a_0 = a \right] \tag{4.2b}$$

where $\gamma < 1$ is the discounting factor.

From control system point of view, the agent is referred to as a *controller*, *Environment* refers to the *open-loop system* or *plant* with all disturbances and *reward* is provided by the reference output. Also, the *policy* is equivalent to the *control law* of a controlled system.

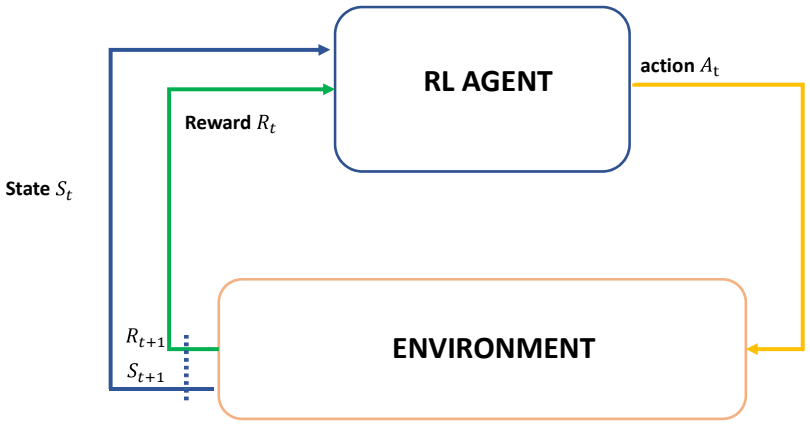


Figure 4.2: The agent–environment interaction in a Markov decision process.

Solving an RL problem means finding a solution with a large v_π. For finite

[2]We used discounted returns here for two reasons: first, as stated in [37], page 254, discounting does not affect policy ordering, and second, in discounted settings, more tools and algorithms have been developed (particularly for the proximal policy optimization (PPO) algorithm that we use later).

MDPs, policy π, with corresponding v_π, is defined to be better than or equal to policy π', with corresponding $v_{\pi'}$, if $v_\pi(s) \geq v_{\pi'}(s)$ for all $s \in \mathcal{S}$. In this case we say $\pi \geq \pi'$. Policies π^* better than or equal to all other policies are called optimal. Optimal policies may not be unique and share the same value function called the optimal value function

$$v^*(s) = \max_\pi v_\pi(s)$$

When the environment dynamics (4.1) are unknown, it is difficult to find the optimal policy and optimal value function. In such cases, a model-free approach is a method of finding the optimal policy without knowing how the environment dynamics are changing. A partial model-free approach is used in this thesis.

Also, in this work, we have used a parametric approach to represent a policy,

$$\pi(a|s;\theta) = \Pr\{A_t = a \mid S_t = s, \theta_t = \theta\} \tag{4.3}$$

where $\theta \in \mathbb{R}^d$ is the policy's parameter vector. Many reasons are cited in the literature to justify why policy gradient methods can generally be more effective, including [37–39]: By storing policy parameters, these methods can learn specific probabilities for taking actions. Also, by learning appropriate levels of exploration, they can approach deterministic policies asymptotically. Continuous action spaces are naturally handled by them, and most importantly, parameterized policy methods have a solid theoretical advantage over action-value methods in the form of the policy gradient theorem, which gives an exact formula for how performance is affected by the policy parameter that does not involve derivatives of the state distribution.

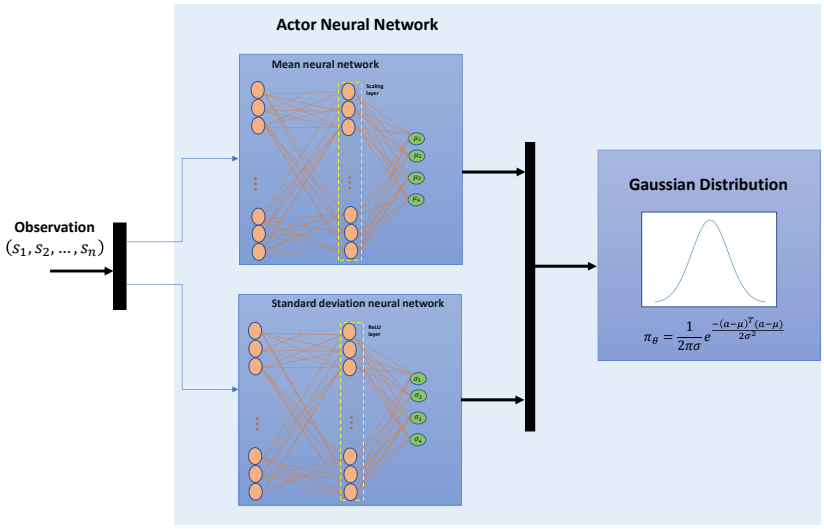
As stated in [37] the remarkable performances of popular RL applications (e.g., AlphaGo, TD-Gammon) owe much of their performance to nonlinear function approximation by multi-layer artificial neural network (ANN)s. For this reason, we have also utilized ANN here, in fact the term deep RL is to indicate that ANN is used as the function approximator in our RL algorithm.

Following, we will introduce actor-critic structure, policy gradient theorem and PPO algorithm that we later use in Section 4.2 as a solution to our RL problem.

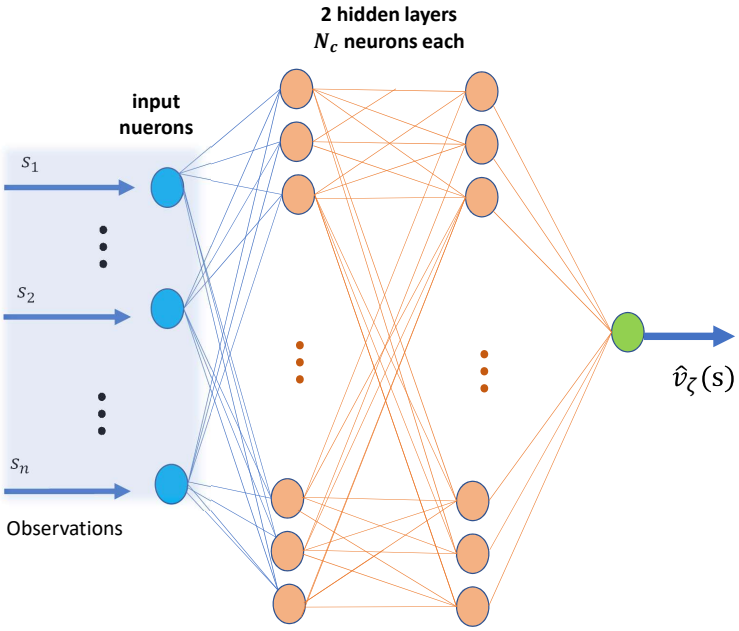
4.1.2 Neural Actor-Critic

The feedforward ANNs used in this thesis are defined here. The notation is based on [40]. As shown in Fig. 4.3a an actor ANN is used to approximate the policy (4.3) while a critic is used to estimate the cost function (4.2a). Without loss of generality, we assume two hidden layers are used in both networks[3]. Observations, states, and actions are specified in Section 4.2.

[3]Later in simulations, we sweep over different hyperparameters, including the number of layers and nodes of the actor-critic network and report the effect of each.



(a) Actor structure



(b) Critic structure

Actor network

As it is customary for the case of continuous actions [37], we consider the following form for the parameterized policy (4.3)

$$\pi(a|s;\theta) = \frac{1}{\sqrt{2\pi_c}\sigma} \exp - \frac{(a-\mu(s,\theta))^\top (a-\mu(s,\theta))}{2(\sigma(s,\theta))^2} \tag{4.4}$$

where μ and σ here are the mean and standard deviation of the normal distribution and $\pi_c \approx 3.14$.

Note that standard deviations must be nonnegative and mean values must fall within the range of the possible actions. Therefore the output layer that returns the standard deviations must be a softplus or ReLU layer to enforce nonnegativity, while the output layer that returns the mean values must be a scaling layer to scale the mean values to the output range [41]. Apart from this and without loss of generality, we assume both $\mu(s,\theta)$ and $\sigma(s,\theta)$ are generated using the same network structure as shown in Fig. 4.3a and only formulate μ here.

Naming activation functions for the i^{th} layer as $g_a^{(i)}$ (superscript a refer to the actor network) we have

$$h_a^{(1)}(s) = g_a^{(1)}\left(\theta^{(1)\top} s + \theta_b^{(1)}\right) \tag{4.5a}$$

$$h_a^{(2)}\left(h_a^{(1)}(s)\right) = g_a^{(2)}\left(\theta^{(2)\top} h_a^{(1)}(s) + \theta_{\text{b}}^{(2)}\right) \tag{4.5b}$$

$$\mu(s) = \theta^{(3)\top} h_a^{(2)}\left(h_a^{(1)}(s)\right) + \theta_{\text{b}}^{(3)} \tag{4.5c}$$

where the first and second layers' outputs are denoted by $h_a^{(1)}$ and $h_a^{(2)}$, respectively, each with dimension N_a, as a design parameter. Thus, we have $\theta^{(1)} \in \mathbb{R}^{N_a \times n}$, $\theta^{(2)} \in \mathbb{R}^{N_a \times N_a}$, $\theta^{(3)} \in \mathbb{R}^{m \times N_a}$ where n is the number of observations and m is the number of actions to be specified later. The biases are denoted by $\theta_{\text{b}}^{(k)}, k = 1,2,3$ with their dimension determined from (4.5a)-(4.5c). Parameter θ consists of $\theta^{(k)}$ and $\theta_{\text{b}}^{(k)}, k = 1,2,3$. As mentioned above $\theta^{(3)}$ and $\theta_b^{(3)}$ are to map actions into an appropriate output range.

It should also be mentioned that the standard deviation σ is sometimes taken as a small constant. A widely used expression for σ is $\sigma = \exp(-\omega)$ where ω is usually chosen on $[0.5, 5]$ ([42]- [43]). However, here we assume σ is tuned during training as in soft actor-critic methods ([44]).

Critic network

The critic ANN is shown in Fig. 4.3b. The network approximates v_{π_θ} by $\hat{v}_\zeta$ which is evaluated using

$$h_c^{(1)}(s) = g_c^{(1)}(\zeta^{(1)\top} s + \zeta_{\mathrm{b}}^{(1)}) \tag{4.6a}$$

$$h_c^{(2)}(h_c^{(1)}(s)) = g_c^{(2)}(\zeta^{(2)\top} h_c^{(1)}(s) + \zeta_{\mathrm{b}}^{(2)}) \tag{4.6b}$$

$$\hat{v}_\zeta(s) = \zeta^{(3)^\top} h_c^{(2)}\left(h^{(1)}(s)\right) + \zeta_{\mathrm{b}}^{(3)} \tag{4.6c}$$

where $\zeta^{(1)} \in \mathbb{R}^{N_c \times n}, \zeta^{(2)} \in \mathbb{R}^{N_c \times N_c}, \zeta^{(3)} \in \mathbb{R}^{1 \times N_c}$. Parameter ζ consists of $\zeta^{(k)}$ and $\zeta_{\mathrm{b}}^{(k)}, k = 1, 2, 3$.

4.1.3 Policy Gradient Theorem

We treat the SLS output tracking problem as an episodic task[4], therefore, the performance measure is defined as the value of the start state of each episode. We follow the same notation as in [37] and assume every episode starts in some particular (non-random) state s_0. Then, the performance is defined as

$$J(\theta) = v_{\pi_\theta}(s_0) \tag{4.7}$$

Where v_{π_θ} is the true value function for π_θ, the policy determined by θ. Considering definition (4.7), and based on the policy gradient theorem (see [37] page 326 for proof), the gradient of J_{π_θ} with respect to θ satisfies

$$\nabla_\theta J(\theta) \propto \sum_s \mu(s) \sum_a q_{\pi_\theta}(s, a) \nabla \pi_\theta(a|s; \theta) \tag{4.8}$$

where distribution μ is the on-policy distribution under π_θ and gradients are column vectors of partial derivatives with respect to the components of θ. In the episodic case, the constant of proportionality is the average length of an episode (which is unknown). Note that proportionality of the sample gradients is all we need because any constant of proportionality can be absorbed into the step size α (see below) which is otherwise arbitrary.

Hence, the gradient ascent update to maximize J_{π_θ} is

$$\theta_{k+1} = \theta_k + \alpha \nabla_\theta J_{\pi_{\theta_k}} = \theta_k + \alpha \sum_{a,s} q_{\pi_\theta}(a, s) \nabla_\theta \pi(a|s; \theta_k) \tag{4.9}$$

where $\alpha > 0$ is the update step size, $q_{\pi_\theta}(a, s)$ is given by (4.2b) and $\pi(a|s; \theta)$ is obtained by (4.4).

[4]Strictly speaking, SLS output tracking is not an episodic task; however, taking a sufficiently large episode length T allows us to approximate a continuing task.

4.1.4 Proximal Policy Optimization

PPO is a model-free, online, and on-policy reinforcement learning method. It is currently considered a state-of-the-art algorithm in the RL community [45] that offers relatively simple implementation and promising results in practice [46]. PPO algorithm is a type of policy gradient training that alternates between sampling data through environmental interaction and optimizing a clipped surrogate objective function using stochastic gradient descent [45]. In order to accelerate the learning process, PPO updates parameters over multiple epochs of mini-batch data, and to improve training stability it limits the size of the policy change at each step.

PPO is a simplified version of trust region policy optimization (TRPO). TRPO is computationally more expensive than PPO, but TRPO tends to be more robust than PPO if the environment dynamics are deterministic and the observation is low dimensional.

A PPO agent, in brief, follows these steps:

- Estimates a probability distribution (4.4) for each action in the action space and randomly selects actions based on that distribution.
- Interacts with the environment for multiple steps using the current policy and then uses a mini-batch experience to update the actor and critic elements (4.5c), (4.6c) using the policy gradient theorem (4.9).

This thesis uses the actor-critic PPO based on the implementation in [41].

Actor-Critic PPO Training Algorithm

1. Initialize the actor $\pi(a|s;\theta)$ (developed in 4.1.2) with some random parameter values θ.
2. Initialize the critic $\hat{v}_\zeta(s)$ (developed in 4.1.2) with random parameter values ζ.
3. Generate N experiences by following the current policy. Let the experience sequence be

$$s_t, a_t, r_{t+1}, s_{t+1}, \ldots, s_{t+N-1}, a_{t+N-1}, r_{t+N}, s_{t+N}$$

   where t is the starting time-step of the current set of N experiences. At the beginning of the training episode, $t = 1$, and for each subsequent set of N experiences in the same training episode, $t \leftarrow t + N$. For each experience sequence that does not contain a terminal state, N is equal to the hyper-parameter *Experience-Horizon*. Otherwise, N is less than Experience-Horizon and s_{t+N} is the terminal state.

4. For each episode step within the set N, $t = t+1, t+2, \ldots, t+N$, compute the return ($G_t$) and generalized advantage function (D_t) [47] using

$$D_t = \sum_{k=t}^{t+N-1} (\gamma\lambda)^{k-t}\delta_k \tag{4.10}$$

$$\delta_k = r_k + b\gamma\hat{v}_\zeta(s_k) \tag{4.11}$$

Here, b is θ if s_{t+N} is a terminal state and 1 otherwise. λ is a smoothing factor[5] and resembles the trace-decay parameter in the $TD(\lambda)$ algorithms [37]. And

$$G_t = D_t + \hat{v}_\zeta(s_t) \tag{4.12}$$

5. Learn from mini-batches of experiences over K epochs. For each learning epoch:

   5.1. Sample a random mini-batch data set of size M from the current set of experiences N. Each element of the mini-batch data set contains a current experience and the corresponding return and advantage function values.

   5.2. Update the critic parameters by minimizing the loss J_{critic} across all sampled mini-batch data.

$$J_{\text{critic}}\,(\zeta) = \frac{1}{M}\sum_{i=1}^{M}(G_i - \hat{v}_\zeta(s_i))^2 \tag{4.13}$$

   Therefore, J_{critic} can be updated given the critic structure (see Section 4.1.2) and ANN updates techniques (e.g., Backpropagation).

   5.3. Normalize the advantage values D_i based on the unnormalized advantages in the current mini-batch.

$$\widehat{D}_i \leftarrow \frac{D_i - \text{mean}\,(D_1, D_2, \ldots, D_M)}{\text{std}\,(D_1, D_2, \ldots, D_M)} \tag{4.14}$$

   5.4. Update the actor parameters by minimizing the actor loss function J_{actor} across all sampled mini-batch data.

$$J_{\text{actor}}(\theta) = \frac{1}{M}\sum_{i=1}^{M}\left(\min\left(k_t(\theta)\cdot\widehat{D}_i, \sigma\left(k_t(\theta)\right)\cdot\widehat{D}_i\right) + w\mathscr{H}_i\left(\theta, s_i\right)\right) \tag{4.15}$$

[5] Also known as *GAE parameter*

Where

$$k_t(\theta) = \frac{\pi(a_t|s_t;\theta)}{\pi(a_t|s_t;\theta_{\text{old}})} \tag{4.16}$$

describes the ratio of policies corresponding to parameters θ and θ_{old} where θ_{old} is the parameter from previous epoch of the optimization.

And $\sigma : \mathbb{R} \to \mathbb{R}$ is the saturation function

$$\sigma(\xi) = \begin{cases} 1+\varepsilon & \xi & > 1+\varepsilon \\ \xi & 1-\varepsilon & \leq \xi \leq 1+\varepsilon \\ 1-\varepsilon & \xi & < 1-\varepsilon \end{cases} \tag{4.17}$$

where the clip ratio $\varepsilon > 0$ is a small value.

Last term $w\mathscr{H}_i(\theta)$ represents the entropy loss function and is described below. This function is added to further encourage exploration.

Assuming $1-\varepsilon \leq k_t(\theta) \leq 1+\varepsilon$, (4.15) simplifies to:

$$J_{\text{actor}}(\theta) = \frac{1}{M}\sum_{i=1}^{M}\left(k_t(\theta)\cdot\widehat{D}_i + w\mathscr{H}_i\left(\theta, s_i\right)\right)$$

with gradient of

$$\nabla_\theta J_{\text{actor}}(\theta) = \frac{1}{M}\sum_{i=1}^{M}\left(\widehat{D}_i\cdot\nabla_\theta k_t(\theta) + w\nabla_\theta\mathscr{H}_i\left(\theta, s_i\right)\right) \tag{4.18}$$

Where we have used the fact that advantage function (4.10) (and hence it's normalized version) is independent of θ. First term in (4.18) demonstrates that PPO is a member of the policy gradient methods (described in Section 4.1.3). Finally, we have the following update rule for actor parameters

$$\theta_{k+1} = \theta_k + \frac{\alpha}{M}\sum_{i=1}^{M}\left(\widehat{D}_i\cdot\frac{\nabla_\theta\pi(a_t|s_t;\theta)}{\pi(a_t|s_t;\theta_{\text{old}})} + w\nabla_\theta\mathscr{H}_i\left(\theta, s_i\right)\right) \tag{4.19}$$

6. Repeat steps 3. through 5. until the training episode reaches a terminal state.

To promote agent exploration, we added the entropy loss function $w\mathscr{H}_i\left(\theta_i, s_i\right)$ to the actor loss (4.15), where w is the entropy loss weight and $\mathscr{H}_i\left(\theta_i, s_i\right)$ is the entropy:

$$\mathscr{H}_i\left(\theta, S_i\right) = \frac{1}{2}\sum_{k=1}^{4}\ln\left(2\pi_c\cdot e\cdot\sigma_{k,i}^2\right) \tag{4.20}$$

where $\sigma_{k,i}$ is the standard deviation (see Section 4.1.2) for action k when in state s_i and $\pi_c \approx 3.14$.

It should be noted that for the PPO algorithm described above, parameters N, γ, λ, K, M, ε, and w are hyperparameters to be tuned.

4.2 SLS Output Tracking as an RL Problem

Our goal in this section is to formulate the output tracking problem (3.1) in the RL framework. The main difference between our approach and conventional RL-based control is that we construct the reward function and observation sets based on the intrinsic linearizability property of the extended SLS.

Due to the main concern of this section being the robustness of the controller, we assume the real system drift vector f and input matrix g (and hence extended $f^{\langle 4\rangle}$ and $g^{\langle 4\rangle}$) are not accessible[6], and we only have access to the nominal plant described by (2.17). So we use the superscript p to refer to actual system (e.g., with f_p, g_p) and n to refer to the nominal system (e.g., f_n, g_n, as given in (2.17)).

We recall from Chapter 3 and (3.13) that the extended SLS model described by dynamics (3.8) (for $i = 4$), inputs $\bar{\nu} = [v_1^{\langle 4\rangle}, \tau^\top]^\top$ and outputs (2.19) is fully[7] feedback-linearizable and thus its output derivative can be written as:

$$y_n^{(\bar{r})} = b_n + \bar{A}_n \bar{\nu}_n \tag{4.21}$$

where $\bar{r} = [6, 6, 6, 2]$, and by $\bar{\nu}_n$ we mean linearizing controller for the nominal plant.

A critical assumption must be made here as in [48–51] before we proceed with the controller design.

Assumption 1. *Extended SLS plant* $(f_p^{\langle 4\rangle}, g_p^{\langle 4\rangle})$ *and model* $(f_n^{\langle 4\rangle}, g_n^{\langle 4\rangle})$ *have the same vector relative degree* $\bar{r} = [6, 6, 6, 2]$ *on* $\mathcal{M}^{\langle 4\rangle}$ *(see 3.11).*

In light of this assumption, there exist linearizing controllers for the plant and the model, which can be described by[8]:

$$\bar{v}_p(x, v_r) = \beta_p(x) + \alpha_p(x) v_r \tag{4.22a}$$

$$\bar{v}_n(x, v_r) = \beta_n(x) + \alpha_n(x) v_r \tag{4.22b}$$

[6] This might seem in contrast to the construction of the extended system where based on (3.9) we need f and g to construct auxiliary inputs, however, as stated in [30]-page 258, we know that for different successful selections of $k(x)$ and $s(x)$ the algorithm always consists of the same number of iterations. Therefore, we assume in this section that functions $k(x)$ and $s(x)$ are independent of true f and g and hence $v_1^{\langle i\rangle}$ for $i = 1, 2, 3$ are accessible and do not require the knowledge of f_p or g_p.

[7] By full feedback linearization we mean the extended system has no zero dynamics.

[8] With slight abuse of notation, from this point, we use x, u, v, f, g, A, b to refer to the quantities for the extended system (e.g., $f \leftarrow f^{\langle 4\rangle}$).

Comparing (4.22b) with (3.14) we can infer

$$\beta_n = -A^{-1}b \tag{4.23a}$$
$$\alpha_n = A^{-1} \tag{4.23b}$$
$$v_r = K\tilde{z} + y_d^{(\bar{r})} \tag{4.23c}$$

While the terms in $\bar{v}_p$ are unknown, they can always be written by

$$\beta_p(x) = \beta_n(x) + \Delta\beta(x) \tag{4.24}$$
$$\alpha_p(x) = \alpha_n(x) + \Delta\alpha(x) \tag{4.25}$$

where $\Delta\beta$ and $\Delta\alpha$ arise due to uncertainties in the SLS model (2.17). An estimate for $\bar{v}_p$ in (4.22a) can then be derived by

$$\bar{v}_p(x, v_r, \theta) = \big(\beta_n(x) + \beta_{\theta_1}(x)\big) + \big(\alpha_n(x) + \alpha_{\theta_2}(x)\big)\, v_r \tag{4.26}$$

where $\beta_{\theta_1} : \mathcal{M}^{\langle 4\rangle} \to \mathbb{R}^4$ is a parameterized estimate for $\Delta\beta$, and $\alpha_{\theta_2} : \mathcal{M}^{\langle 4\rangle} \to \mathbb{R}^{4\times 4}$ is a parameterized estimate for $\Delta\alpha$. The parameters $\theta_1 = \left(\theta_1^1, \theta_1^2, \ldots, \theta_1^{K_1}\right) \in \mathbb{R}^{K_1}$ and $\theta_2 = \left(\theta_2^1, \theta_2^2, \ldots, \theta_2^{K_2}\right) \in \mathbb{R}^{K_2}$ are to be learned and combined into the total set of learned parameters [9] $\theta = (\theta_1, \theta_2) \in \mathbb{R}^{K_1+K_2}$. The learned component is directly added to the controller derived from our nominal model, allowing us to integrate prior knowledge of the plant that the controller designer has.

By injecting (4.26) to SLS we obtain

$$y_p^{(\bar{r})} = \underbrace{b_p(x) + \bar{A}_p(x)\bar{v}_p(x, v_r, \theta)}_{W(x, v_r, \theta)} \tag{4.27}$$

In order to linearize the plant, we have to to find θ^* such that $W(x, v_r, \theta^*) \approx v_r$ for each $x \in \mathcal{M}^{\langle 4\rangle}$ and $\bar{v} \in \mathbb{R}^4$. Thus, we define the point-wise loss $\ell : \mathcal{M}^{\langle 4\rangle} \times \mathbb{R}^4 \times \mathbb{R}^{K_1+K_2} \to \mathbb{R}^+$ by

$$\ell(x, v_r, \theta) = \big\| v_r - W(x, v_r, \theta) \big\|_2^2, \tag{4.28}$$

In fact, ℓ measures the degree to which the learned controller $\bar{v}_p(x, v_r, \theta)$ in (4.26) linearizes the plant at state x when v_r is applied to the linear reference model.

Now, we are ready to formulate our linearization problem in the RL framework:

[9] Parameter θ here is independent of the actor parameter θ we used in (4.4).

$$\begin{aligned} &\min_{\theta\in\Theta} \mathbb{E}_{x_0\sim X, v_{r_k}\sim V_r}\left[\sum_{k=1}^{N} \ell\left(x_k, v_{r_k}, \theta_k\right)\right] \\ \text{subject to:} \quad & x_{k+1} = x_k + F_k\left(x_k, u_k\right), \quad x_0 = x_0 \\ & \bar{v}_{p_k} = \bar{v}_p\left(x_k, v_{r_k}, \theta_k\right) \end{aligned} \tag{4.29}$$

Where F_k is the discrete-time implementation of the SLS extended drift vector in (3.8) (with $i = 4$), and $\bar{v}_p$ is in (4.26). We sample initial conditions x_0 from X, where X models our preference for having an accurate linearizing controller at different points in the SLS state-space. We also sample the virtual input v_{r_k} from V_r at each time-step according to our desired trajectory (e.g., the fig-8 trajectory).

From RL prospective, ℓ in (4.29) is the reward function which we seek to minimize its expected sum over time. Vector $[x_k^\top, v_{r_k}^\top]^\top$ forms our observations[10] at each time-step, and $\bar{v}_{p_k}$ is the input vector which at each step consists of the linearizing feedback for the nominal model (developed in Section 3.3.1) and a learning part generated by the PPO algorithm described in Section 4.1.4.

The challenging part when solving (4.29) is that we can't evaluate W (and hence ℓ) since terms b_p and $\bar{A}_p$ in (4.27) are unknown. Alternatively, we can run experiments on the plant (either in simulations or using SLS hardware) and use numerical derivation techniques to obtain $y_p^{(\bar{r})}$ from the plant output.

Also, as stated in [48], (4.29) is a non-convex problem, so it is highly probable for the solution to be of the type local optimal. However, it is proved in [52] that if the true linearizing controller follows the form

$$\beta_{\theta_1}(x) = \sum_{k=1}^{K_1} \theta_1^k \beta_k(x) \quad \alpha_{\theta_2}(x) = \sum_{k=1}^{K_2} \theta_2^k \alpha_k(x) \tag{4.30}$$

where $\{\beta_k\}_{k=1}^{K_1}$ and $\{\alpha_k\}_{k=1}^{K_2}$ are nonlinear continuous functions, then the cost function in (4.29) becomes quadratic in parameters and therefore a convex problem. In addition, if $\{\beta_k\}_{k=1}^{K_1}$ and $\{\alpha_k\}_{k=1}^{K_2}$ are linearly independent then (4.29) will be strongly convex. Obviously, in such a case it is reliable to use our iterative PPO technique to find its globally optimal solution. An example is provided in [49] where the norm of the tracking error for a double-pendulum controlled by the control structure (4.26) converges to zero. However, we were not able to verify whether the given form (4.30) holds for the uncertain SLS system[11].

[10] The key difference between the observation vector and extended SLS states (which are composed of SLS states and $\bar{u}$ and its derivatives up to order three) is that the observation vector also consists of v_r.

[11] To analyze this, we first calculated the symbolic linearizing control law as a function of all parameters (including pendulum and quadrotor masses and cable length) and then created uncertainty on each parameter (e.g., by inserting $L_p = L_n + \Delta L$), but due to the large numbers of symbolic expressions, we could not verify whether α and β follows the form of (4.30) or whether

Figure 4.4 illustrates the overall structure of the RL-based linearizing controller.

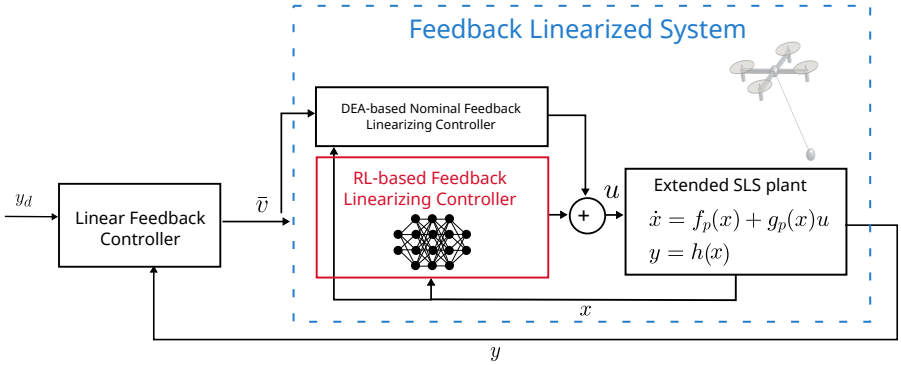


Figure 4.4: RL-based linearizing controller for the SLS that consists of a learning part running by PPO and a nominal part which is built using partial knowledge of the true model.

4.3 Simulations

The purpose of this section is to train the PPO agent described in Section 4.1.4 using actions, states and rewards defined in Section 4.2. For the sake of exact analysis, we assume we have access to the true plant (i.e., f_p and g_p) and use true b_p and $\bar{A}_p$ to calculate $y_p^{(\bar{r})}$ in (4.27) from generated $\bar{v}_p(x, v_r, \theta)$, and then to evaluate the reward ℓ in (4.29), however, we hide this knowledge and use nominal f_m and g_m when constructing the linearizing feedback $\bar{v}_n$ in (4.22b).

Our first step is to determine the best set of hyperparameters by sweeping over feasible values of each parameter while keeping others fixed. We then select the best set and after a training phase, validate our agent against different uncertainties and disturbances.

It is worth noting that this thesis uses the RL toolbox from [41] to perform simulations; this is contrary to the fact that most people use Python for RL simulations. We used MATLAB because it enabled us to employ advanced integration techniques using the Simulink environment and also because the first part of this thesis (Section 3.1) was already done in MATLAB.

In addition, we implemented our training algorithms using computing services provided by the Digital Research Alliance of Canada[12]. This was crucial because some implementations were not possible with existing computers (especially hyperparameter tuning).

bases are linearly independent

[12]Previously called ComputeCanada

4.3.1 Hyperparameter Tuning

This subsection aims to provide the best set of hyperparameters for the PPO algorithm of Section 4.1.4 which achieves the lowest possible expected return for the RL problem formulated in (4.29). Hyperparameters and the corresponding value sets are given in the table 4.1. As this section's goal is to identify the setting which results in the lowest cost within a smaller number of samples, we let our agent freely explore the action space and never stop the training unless it reaches the target number of episodes. Additionally, we terminate each episode once positions and velocities pass a certain threshold to prevent instability during training.

For this training phase, we assume the pendulum mass m_p (see (2.17)) in true SLS model and its nominal counterpart are different. Specifically, we assume the nominal feedback linearizing controller (4.22b) is built using inexact knowledge of the pendulum mass so that the nominal m_p is 0.8 of its true value.

Table 4.1: Hyperparameters used in the proposed PPO algorithm

Hyperparameter	Value Set
Discount factor (γ)	$[0.7, 0.8, 0.9, 0.92, 0.998]$
Mini-batch size (M)	$[10, 20, 40, 64, 100]$
Advantage Normalization Method $(\widehat{D})$	[none, current, moving]
Number of Learning Epochs (K)	$[1, 3, 5, 7, 10]$
Actor & Critic Learning Rates (subset1)$(\alpha_{a,c})$	$[10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}]$
Actor & Critic Learning Rates (subset2)$(\alpha_{a,c})$	$[0.05, 0.08, 0.1, 0.3, 0.5, 0.8, 1]$
GAE parameter (λ)	$[0.3, 0.5, 0.7, 0.97, 0.99]$
Critic Number of Nodes (N_c)	$[30, 64, 100, 128, 200]$
Actor Number of Nodes (N_a)	$[30, 64, 100, 128, 200]$
Critic Number of Layers	$[2, 3, 4, 5]$
Actor Number of Layers	$[2, 3, 4, 5]$
Clip ratio (ε)	$[0.1, 0.3, 0.5, 0.7, 0.9]$
Entropy loss weight(w)	$[10^{-3}, 5 \times 10^{-3}, 10^{-2}, 0.05, 0.1]$
Experience-Horizon (N)	$[200, 300, 400, 500, 600]$

Figures 4.5- 4.7 shows the performance of the PPO agent when the corresponding hyperparameter is changing and others are fixed in terms of the cumulative cost of (4.29). The value set for each hyperparameter is selected based on their feasible values (e.g., λ and $\alpha_{a,c}$ must always be less than one) and what is recommended in the literature. Fluctuations in the figures are because our agent is set to explore the action space (minimum exploitation). By analyzing these figures, we select

hyperparameters:

$$\begin{aligned}&\gamma = 0.998,\ M = 20, \text{normalization= current}, K = 5, \alpha_a = 10^{-3}, \alpha_c = 10^{-4}\\ &N_a = 100, N_c = 100, L_a = L_c = 3, \varepsilon = 0.5, \lambda = 0.99, w = 0.005, N = 500\end{aligned} \tag{4.31}$$

4.3.2 Training

Picking the best value for hyperparameters in (4.31), we are now ready for our final training phase. For the construction of the nominal linearizing controller (4.22b), we assume that the nominal quadrotor mass (m_q) is 0.8 if its actual value. The training result is shown in Fig. 4.8; we removed the first 32000 episodes of the graph as it resulted in no reward/learning. Also, we stopped the training when the average reward passed 25000 (which is the average episode reward for 200 consecutive episodes).

Fig. 4.9 illustrates the average number of steps our agent takes in each episode. As shown in the figure, as learning progresses, our agent stays longer in each episode, which means by the end of the learning process, the agent is able to finish episodes without ever failing[13].

4.3.3 Validation

For the validation phase, we use the trained agent of Section 4.3.2 with different initial conditions and desired trajectory. Fig. 4.10 shows episode reward (ℓ in (4.29), with no average) for the validation phase for 100 episodes each starting with random initial conditions.

Parameter Uncertainty

Fig 4.11 shows the 3D position of our RL agent along with the desired trajectory when the quadrotor mass is 0.8 of its true value. As we can see from Fig. 4.12, bounded tracking error is achieved despite the parameter uncertainty. If we remove the learning part and only use the nominal linearizing controller of Chapter 3 (u_m in 3.10), the result is unstable at $t = 10^{-5}$ second[14].

[13] To see this, note that in our Simulink environment, we limit the length of each learning episode to 800 steps. Our agent is free to perform all these steps unless positions and velocities deviate from their desired counterparts by a certain threshold.

[14] When integrating with ode45.

External Disturbance

Fig. 4.13 shows position errors in case of both constant disturbance and quadrotor mass parameter uncertainty. We can see that the RL-based control has more robustness compared to the nominal linearizing control with integral action[15].

Sample Efficiency

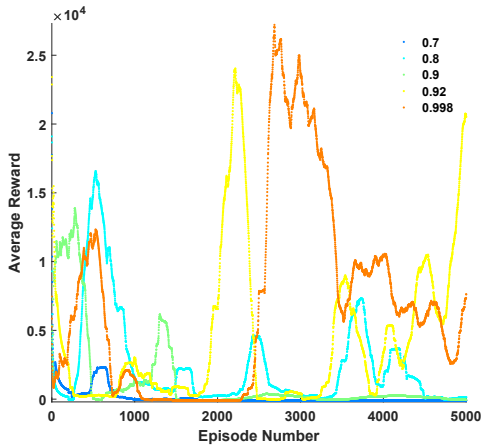
Here we compare our RL problem formulated in (4.29) with a conventional RL formulation for the SLS output tracking task (3.1).

We consider the following RL formulation:

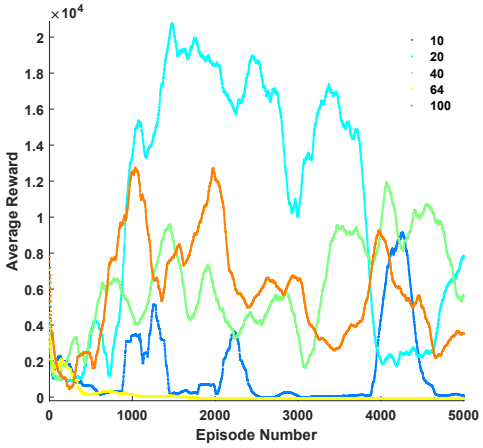
$$\begin{aligned} &\min_{\theta\in\Theta} \mathbb{E}_{x_0\sim X, v_k\sim V}\left[\sum_{k=1}^{N}\left\|y(t)-y_d(t)\right\|_2^2\right] \\ \text{subject to:}\quad &x_{k+1}=x_k+F_k\left(x_k,u_k\right), \quad x_0=x_0 \\ &u_k=u_k(x_k) \end{aligned} \tag{4.32}$$

Compared to (4.29), (4.32) is not based on feedback linearizability of the SLS in that the cost function does not reflect the degree to which the input u linearizes the plant and input u itself does not incorporate the nominal linearizing controller u_m (4.22b) anymore. Fig. 4.14 shows the average reward for the above formulation when we have the same parametric uncertainty as in Fig. 4.8; however, learning has not occurred in Fig. 4.14 for the same agent settings and number of episodes.

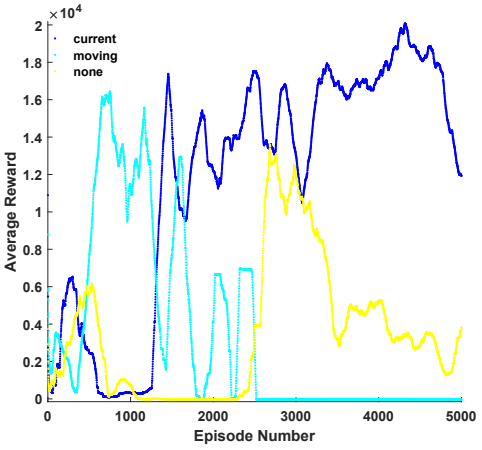
[15] Control using the nominal linearizing control with $d_q = [1, -1.5, -1]^\top$ and complete knowledge of all parameters will result in instability.



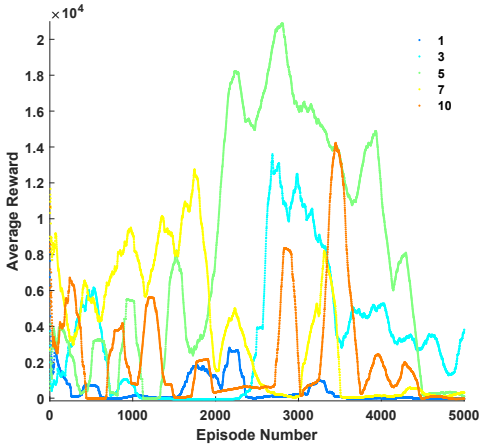
(a) Discount factor γ



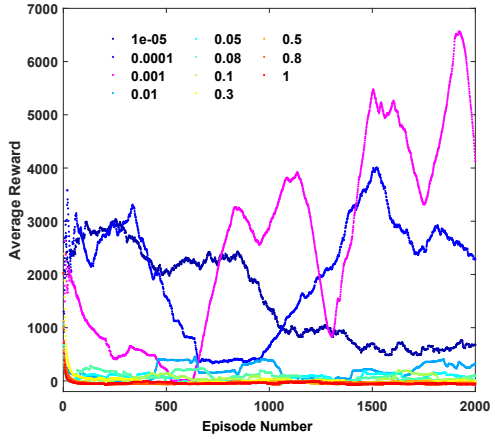
(b) Mini-batch size (M)



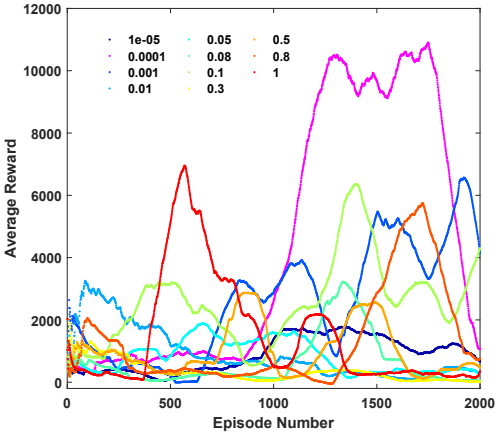
(c) Advantage Normalization Method ($\widehat{D}$)



(d) Number of Learning Epochs (K)

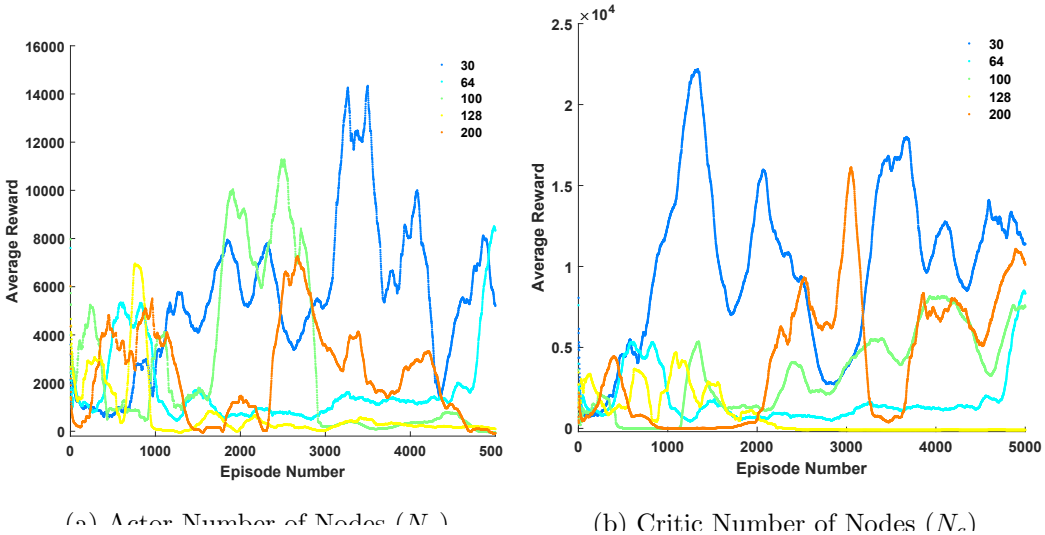


(e) Actor Learning Rate α_a

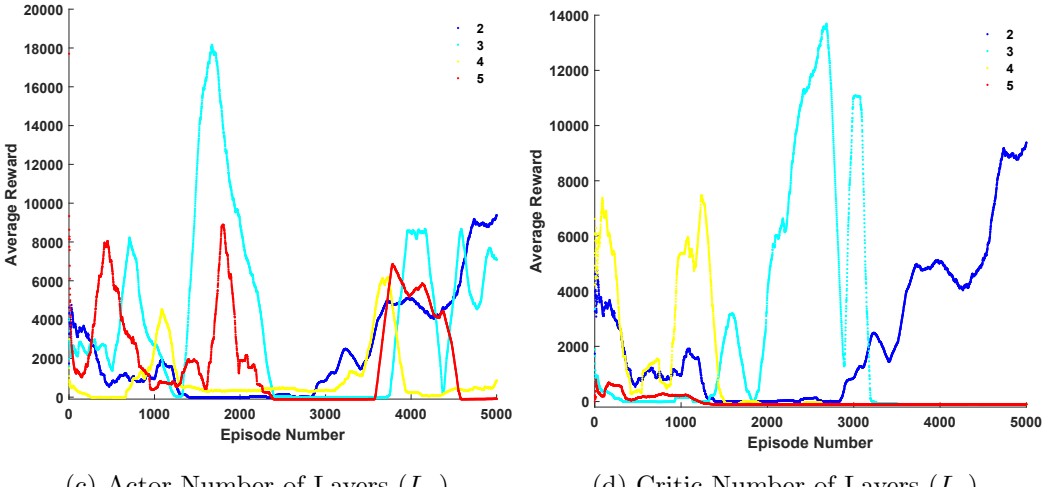


(f) Critic Learning Rate α_c

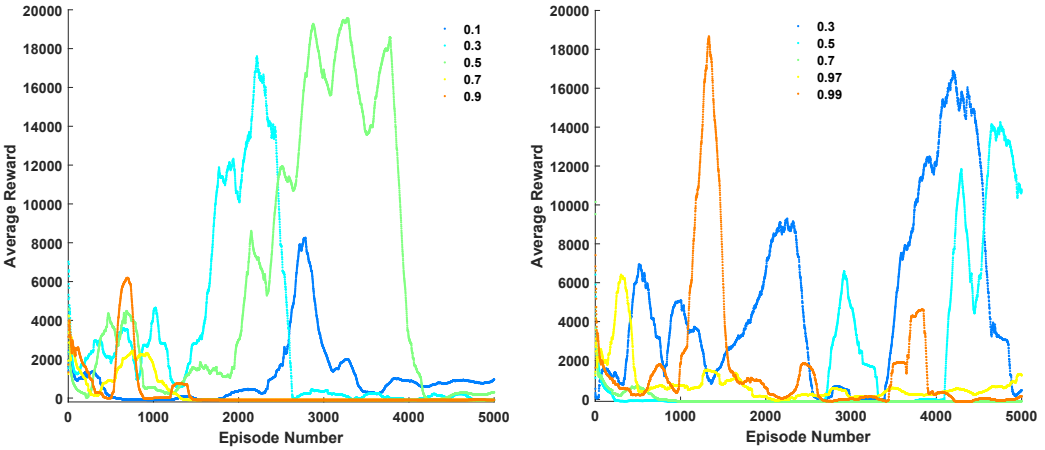
Figure 4.5: Heyeprparameter tuning for the proposed RL algorithm (Set1), when pendulum mass is 0.8 of its true value.



(a) Actor Number of Nodes (N_a) (b) Critic Number of Nodes (N_c)

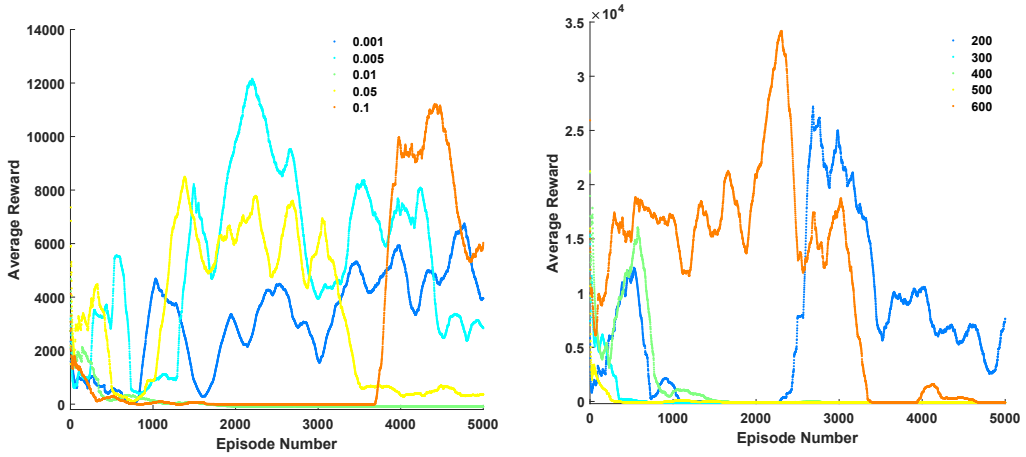


(c) Actor Number of Layers (L_a) (d) Critic Number of Layers (L_c)



(e) Clip ratio (ε) (f) GAE factor (λ)

Figure 4.6: Heyeprparameter tuning for the proposed RL algorithm (Set2), when pendulum mass is 0.8 of its true value.



(a) Entropy loss weight (w) (b) Experience-Horizon (N)

Figure 4.7: Heyeprparameter tuning for the proposed RL algorithm (Set3), when pendulum mass is 0.8 of its true value.

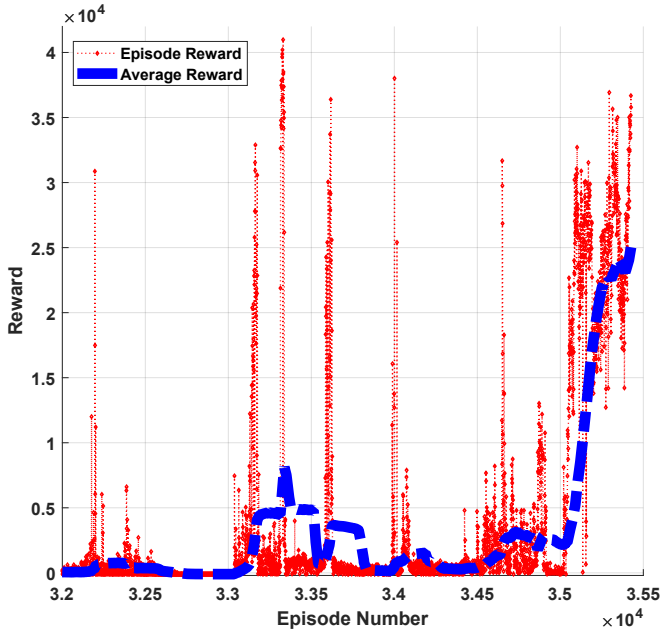


Figure 4.8: Training results (average reward) with hyperparameter set (4.31) when quadrotor mass is 0.8 of its true value.

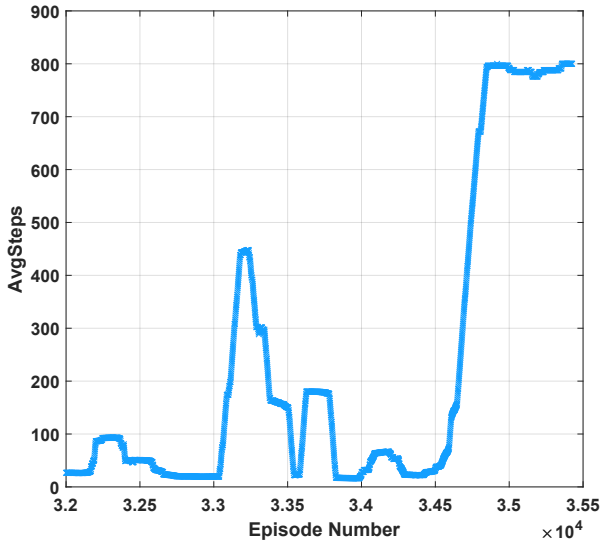


Figure 4.9: Training results (average steps) with hyperparameter set (4.31) when quadrotor mass is 0.8 of its true value.

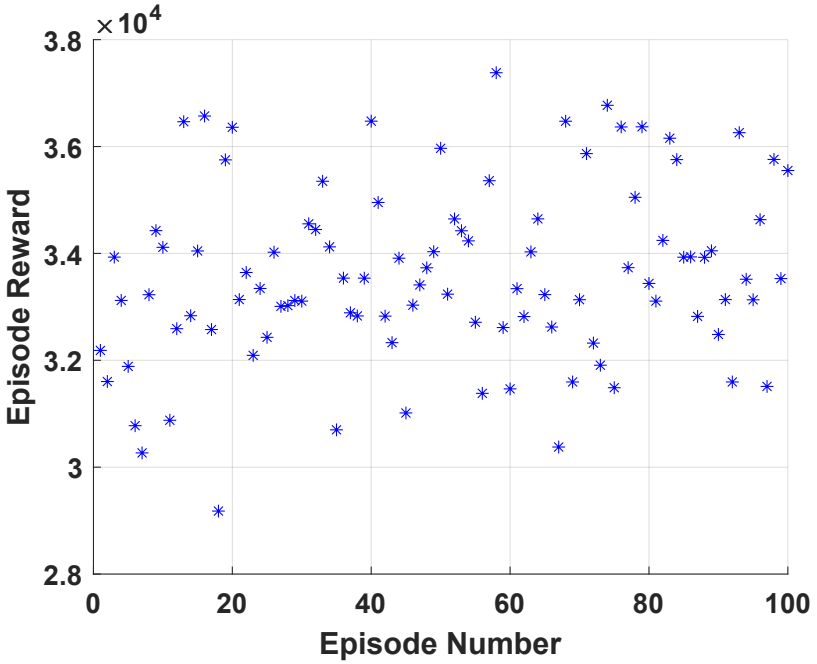


Figure 4.10: Validation result (episode rewards) with hyperparameter set (4.31) when quadrotor mass is 0.8 of its true value.

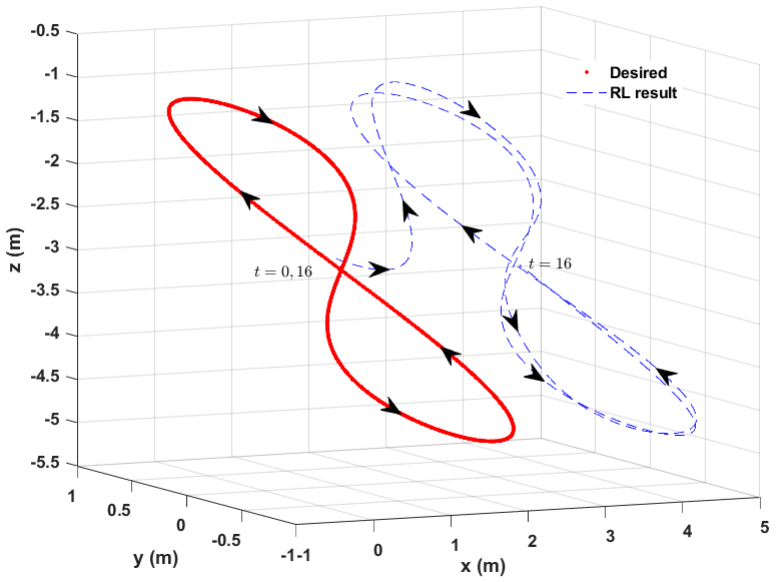


Figure 4.11: Validation result (3D positions) with hyperparameter set (4.31) when quadrotor mass is 0.8 of its true value.

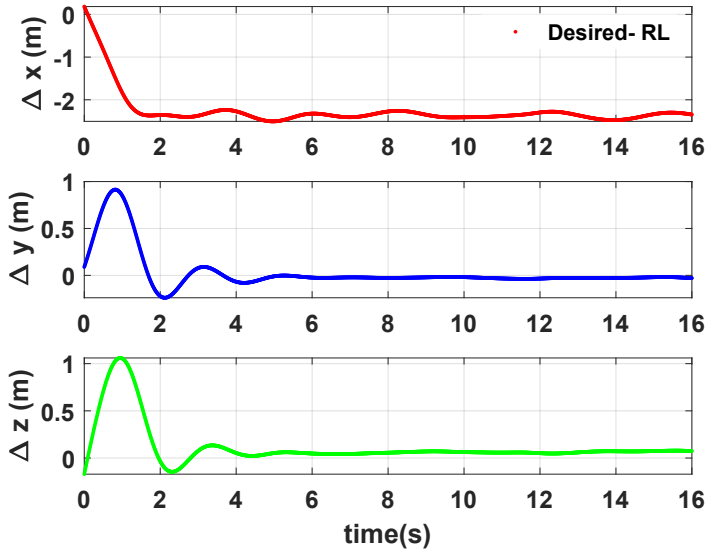


Figure 4.12: Validation result (positions errors) with hyperparameter set (4.31) when quadrotor mass is 0.8 of its true value.

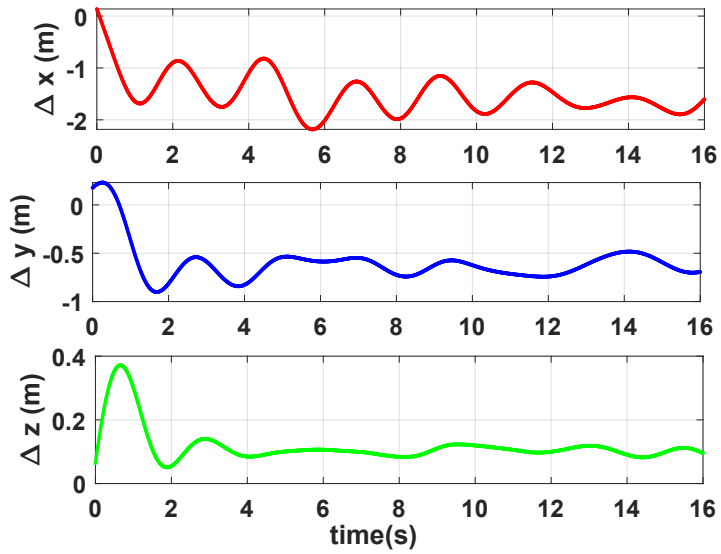


Figure 4.13: Validation result (positions errors) for constant disturbance vector $d_q = [1, -1.5, -1]^\top$ N, when quadrotor mass is 0.8 of its true value.

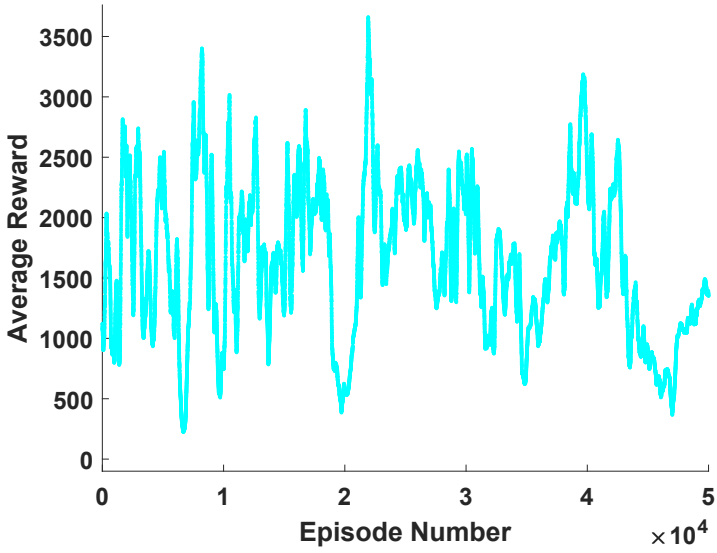


Figure 4.14: Validation result (sample efficiency) for the case of conventional RL when quadrotor mass is 0.8 of its true value.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

This work presented two output tracking controllers for an slung load system (SLS) using dynamic state feedback linearization and reinforcement learning (RL).

In the first design section, assuming perfect knowledge of the SLS model is available, four iterations of the dynamic extension algorithm (DEA) were used to derive the control law. Hence, the dynamic state feedback introduces an additional 4 controller states. These states are related to unmanned aerial vehicle (UAV) thrust and its time derivatives up to order 3. The DEA constructs an extended dynamics with a full relative degree and hence static state feedback linearizable with an auxiliary control. Extended states augmented with integral of pendulum position and static state feedback linearization achieve asymptotic output tracking in the presence of local force disturbances.

The tracking error dynamics are linear time-invariant (LTI) exponentially stable (ES). Linearity is an important property of the design as it facilitates gain selection and stability analysis. Simulations confirm the design's ES property and ease of gain tuning. Performance is compared with an existing design [2]. The proposed design is implemented using symbolic math software which can be applied to generic nonlinear control-affine systems and is available at [1].

Moreover, an RL-based control algorithm is provided which uses proximal policy optimization (PPO) to iterate the learning steps and is built upon the DEA controller in the sense that its reward function measures the degree to which actions contribute to feedback linearization. An exhaustive hyperparameter analysis is done and simulations are provided to demonstrate the applicability of the proposed learning method.

5.2 Future Work

There are a number of ways in which each chapter presented in this thesis can be extended.

Our model can be further extended to include the effect of drag forces and the offset of the pendulum pivot from the UAV center of mass (CoM). To compensate for these disturbances, the first control section (without learning) can be extended by adding a disturbance observer. Adding control objectives that minimize pendulum oscillations and/or schemes that prevent states from approaching singular points can also improve that chapter.

Also, there are several ways in which the last design section can be enriched. While the proposed learning algorithm here is offline, approaches similar to [49] can be used for online implementation, however, we believe safety should be taken into account when designing such online learning schemes. Furthermore, we can use other state-of-the-art RL agents (e.g. deep deterministic policy gradient (DDPG), twin delayed DDPG (TD3)) to see whether better learning can be accomplished in terms of sample efficiency and average rewards.

Finally, experimental validation and software in the loop (SITL) implementation of the proposed controllers are also possible avenues for future work.

Bibliography

[1] A. Mohammadhasani, "Maple codes for dynamic extension algorithm (DEA)," 2021. [Online]. Available: https://github.com/ANCL/DSFBL.git

[2] K. Sreenath, T. Lee, and V. Kumar, "Geometric control and differential flatness of a quadrotor UAV with a cable-suspended load," in *Proc. IEEE Int. Conf. on Desision and Control*, Firenze, Tuscany, 2013, pp. 2269–2274.

[3] https://aws.amazon.com, accessed: 2022-09-21.

[4] G. Cai, J. Dias, and L. Seneviratne, "A survey of small-scale unmanned aerial vehicles: Recent advances and future development trends," *Unmanned Systems*, vol. 2, no. 02, pp. 175–199, 2014.

[5] P. E. I. Pounds, D. R. Bersak, and A. M. Dollar, "Grasping from the air: Hovering capture and load stability," in *Proc. IEEE Int. Conf. on Robotics and Automation*, Toronto, ON, 2011, pp. 2491–2498.

[6] A. Gawel, M. Kamel, T. Novkovic, J. Widauer, D. Schindler, B. P. von Altishofen, R. Siegwart, and J. Nieto, "Aerial picking and delivery of magnetic objects with MAVs," in *Proc. IEEE Int. Conf. on Robotics and Automation*, Singapore, 2017, pp. 5746–5752.

[7] M. Bernard and K. Kondak, "Generic slung load transportation system using small size helicopters," in *Proc. IEEE Int. Conf. on Robotics and Automation*, Kobe, Osaka, 2009, pp. 3258–3264.

[8] P. Kotaru, G. Wu, and K. Sreenath, "Differential-flatness and control of quadrotor (s) with a payload suspended through flexible cable (s)," in *Indian Control Conference*. IEEE, 2018, pp. 352–357.

[9] 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. 3–13, 2001.

[10] A. Irscheid, M. Konz, and J. Rudolph, "A flatness-based approach to the control of distributed parameter systems applied to load transportation with heavy ropes," in *Advanced Control Techniques in Complex Engineering Systems: Theory and Applications*. Springer, 2019, pp. 279–294.

[11] R. M. Murray, "Trajectory generation for a towed cable system using differential flatness," *IFAC Proceedings Volumes*, vol. 29, no. 1, pp. 2792–2797, 1996.

[12] F. A. Goodarzi, D. Lee, and T. Lee, "Geometric stabilization of a quadrotor UAV with a payload connected by flexible cable," in *Proc. American Control Conf*, Portland, OR, 2014, pp. 4923–4930.

[13] P. Kotaru, G. Wu, and K. Sreenath, "Differential-flatness and control of quadrotor(s) with a payload suspended through flexible cable(s)," in *Indian Control Conference*, Kanpur, Uttar Pradesh, 2018, pp. 352–357.

[14] M. Guerrero-Sánchez, R. Lozano, P. Castillo, O. Hernández-González, C. García-Beltrán, and G. Valencia-Palomo, "Nonlinear control strategies for a UAV carrying a load with swing attenuation," *Applied Mathematical Modelling*, vol. 91, pp. 709–722, 2021.

[15] M. E. Guerrero, D. Mercado, R. Lozano, and C. García, "Swing-attenuation for a quadrotor tranporting a cable suspended payload," *ISA transactions*, vol. 68, pp. 433–449, 2017.

[16] K. Klausen, T. I. Fossen, and T. A. Johansen, "Nonlinear control with swing damping of a multirotoruav with suspended load," *J. Intell. Robot. Syst.*, vol. 88, pp. 379–394, 2017.

[17] K. Mohammadi, S. Sirouspour, and A. Grivani, "Control of multiple quadcopters with a cable-suspended payload subject to disturbances," *IEEE/ASME Trans. Mechatronics*, vol. 25, no. 4, pp. 1709–1718, 2020.

[18] K. Sreenath, N. Michael, and V. Kumar, "Trajectory generation and control of a quadrotor with a cable-suspended load-a differentially-flat hybrid system," in *Proc. IEEE Int. Conf. on Robotics and Automation*, Karlsruhe, 2013, pp. 4888–4895.

[19] S. Dai, T. Lee, and D. S. Bernstein, "Adaptive control of a quadrotor uav transporting a cable-suspended load with unknown mass," in *Proc. IEEE Int. Conf. on Desision and Control*. IEEE, 2014, pp. 6149–6154.

[20] L. Qian and H. H. Liu, "Path-following control of a quadrotor uav with a cable-suspended payload under wind disturbances," *IEEE Transactions on Industrial Electronics*, vol. 67, no. 3, pp. 2021–2029, 2019.

[21] G. V. Raffo and M. M. de Almeida, "Nonlinear robust control of a quadrotor uav for load transportation with swing improvement," in *Proc. American Control Conf.* IEEE, 2016, pp. 3156–3162.

[22] J. Zeng and K. Sreenath, "Geometric control of a quadrotor with a load suspended from an offset," in *Proc. American Control Conf.* IEEE, 2019, pp. 3044–3050.

[23] L. Qian and H. H. Liu, "Dynamics and control of a quadrotor with a cable suspended payload," in *Proc. Canadian Electrical and Computer Engineering Conf.* IEEE, 2017, pp. 1–4.

[24] H. Hua, Y. Fang, X. Zhang, and C. Qian, "A new nonlinear control strategy embedded with reinforcement learning for a multirotor transporting a suspended

payload," *IEEE Transactions on Industrial Electronics*, vol. 27, no. 2, pp. 1174–1184, 2021.

[25] H. Xie, "Dynamic visual servoing of rotary wing unmanned aerial vehicles," Ph.D. dissertation, Dept. of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, 2016.

[26] S. Bouabdallah, "Design and control of quadrotors with application to autonomous flying," Ph.D. dissertation, École Polytechnique Fédérale de Luasanne, Lausanne, Switzerland, 2007.

[27] L. Meier, "PX4 autopilot," http://pixhawk.org/ [accessed 01 Jan 2018], Institute for Visual Computing, Swiss Federal Institute of Technology Zurich, 2018. [Online]. Available: http://pixhawk.org/

[28] B. Jakubczyk *et al.*, "On linearization of control systems," 1980.

[29] R. W. Brockett, "Feedback invariants for nonlinear systems," *IFAC Proceedings Volumes*, vol. 11, no. 1, pp. 1115–1120, 1978.

[30] A. Isidori, "Nonlinear control systems. communications and control engineering," *Springer. 3rd edition.*, 1995.

[31] H. Nijmeijer and W. Respondek, "Dynamic input-output decoupling of nonlinear control systems," *IEEE Transactions on Automatic Control*, vol. 33, no. 11, pp. 1065–1070, 1988.

[32] H. Nijmeijer and J. Schumacher, "The regular local noninteracting control problem for nonlinear control systems," *SIAM J. Control Optim.*, vol. 24, no. 6, pp. 1232–1245, 1986.

[33] A. Moeini, A. F. Lynch, and Q. Zhao, "A backstepping disturbance observer control for multirotor UAVs: theory and experiment," *Int. J. Control*, pp. 1–15, 2021, online access.

[34] Maplesoft, a division of Waterloo Maple Inc., "Maple," Waterloo, Ontario. [Online]. Available: https://hadoop.apache.org

[35] G. Fischer, "Nonlincon: symbolic analysis and design package for nonlinear control systems," Master's thesis, Eindhoven University of Technology, Eindhoven, 1994.

[36] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor UAV on SE(3)," in *Proc. IEEE Int. Conf. on Desision and Control*, Atlanta, GA, 2010, pp. 5420–5425.

[37] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction.* MIT Press, 2018.

[38] I. Grondman, L. Busoniu, G. A. Lopes, and R. Babuska, "A survey of actor-critic reinforcement learning: Standard and natural policy gradients," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 1291–1307, 2012.

[39] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in neural information processing systems*, vol. 12, 1999.

[40] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT Press, 2016.

[41] MATLAB, "9.12.0.1956245 (r2022a) update 2," Natick, Massachusetts, 2022.

[42] J. Achiam, "Spinning up in deep reinforcement learning.(2018)," *URL https://spinningup. openai. com*, 2018.

[43] A. Raffin, A. Hill, M. Ernestus, A. Gleave, A. Kanervisto, and N. Dormann, "Stable baselines3," 2019.

[44] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

[45] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[46] N. Heess, D. TB, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. Eslami *et al.*, "Emergence of locomotion behaviours in rich environments," *arXiv preprint arXiv:1707.02286*, 2017.

[47] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[48] T. Westenbroek, D. Fridovich-Keil, E. Mazumdar, S. Arora, V. Prabhu, S. S. Sastry, and C. J. Tomlin, "Feedback linearization for uncertain systems via reinforcement learning," in *Proc. IEEE Int. Conf. on Robotics and Automation*. IEEE, 2020, pp. 1364–1371.

[49] T. Westenbroek, E. Mazumdar, D. Fridovich-Keil, V. Prabhu, C. J. Tomlin, and S. S. Sastry, "Adaptive control for linearizable systems using on-policy reinforcement learning," in *Proc. IEEE Int. Conf. on Desision and Control*. IEEE, 2020, pp. 118–125.

[50] M. Estrada, "Toward the control of non-linear, non-minimum phase systems via feedback linearization and reinforcement learning," Ph.D. dissertation, University of California Berkeley, CA, USA, 2021.

[51] M. Estrada, S. Li, and X. Cai, "Feedback linearization of car dynamics for racing via reinforcement learning," *arXiv preprint arXiv:2110.10441*, 2021.

[52] T. Westenbroek, D. Fridovich-Keil, E. Mazumdar, S. Arora, V. Prabhu, S. S. Sastry, and C. J. Tomlin, "Feedback linearization for unknown systems via reinforcement learning," *arXiv preprint arXiv:1910.13272*, 2019.

****

Appendix A

Pendulum Disturbance

Here we construct the expression for w_p in (2.17) based on the the disturbance acting on the quadrotor in (2.3b) .

If we rewrite the Euler-Lagrange equations (2.13) using quadrotor positions and load angles $\bar{q} = [\bar{p}^\top, \alpha, \beta]^\top$ then d_q (from (2.3b)) shows up (see [16])

$$\bar{M}(\bar{q})\ddot{\bar{q}} + \bar{C}(\bar{q}, \dot{\bar{q}})\dot{\bar{q}} + \bar{G}(\bar{q}) = \bar{B}\bar{u} + [d_q, 0, 0]^\top \tag{A.1}$$

Where $\bar{M}$, $\bar{C}$, $\bar{G}$, and $\bar{B}$ can be obtained the same way as M, C, G, and B by solving (2.12) for $\bar{q}$ instead of q. Using (A.1) and similar to (2.15) we can obtain the disturbed state dynamics this time in $\bar{q}$ coordinates. Let's for simplicity only consider the disturbance terms and ignore other terms (i.e., $\bar{M}^{-1}(\bar{C}\dot{\bar{q}} + \bar{G})$ and $\bar{M}^{-1}\bar{B}u$)

$$\ddot{\bar{q}}_D = \bar{M}^{-1}[d_q, 0, 0]^\top \tag{A.2}$$

Where superscript "D" means we are only looking for the term caused by disturbance. Now, if we replace $\bar{q}$ with q in (A.2) using (2.7) we obtain

$$\ddot{q}_D = \bar{M}^{-1}[d_q, 0, 0]^\top + L[\ddot{\beta}\mathrm{c}_\beta, -\ddot{\alpha}\mathrm{c}_\alpha\mathrm{c}_\beta + \ddot{\beta}\mathrm{s}_\alpha\mathrm{s}_\beta, -\ddot{\alpha}\mathrm{s}_\alpha\mathrm{c}_\beta - \ddot{\beta}\mathrm{c}_\alpha\mathrm{s}_\beta, 0, 0]_D)^\top \tag{A.3}$$

Where the term in bracket is obtained from the last two rows in (A.2). Let's rename rows in $\bar{M}^{-1}$ as $(\bar{m}_1, \bar{m}_2, \ldots, \bar{m}_5)$, then we can rewrite (A.3) as

$$\ddot{q}_D = \underbrace{\begin{bmatrix} \bar{m}_1 + \bar{m}_5\mathrm{c}_\beta \\ \bar{m}_2 - \bar{m}_4\mathrm{c}_\alpha\mathrm{c}_\beta - \bar{m}_5\mathrm{s}_\alpha\mathrm{s}_\beta \\ \bar{m}_3 - \bar{m}_4\mathrm{s}_\alpha\mathrm{c}_\beta - \bar{m}_5\mathrm{c}_\alpha\mathrm{s}_\beta \\ \bar{m}_4 \\ \bar{m}_5 \end{bmatrix}}_{w_p(x)} \begin{bmatrix} d_q \\ 0 \\ 0 \end{bmatrix} \tag{A.4}$$

Therefore w_p in (2.17) is the matrix coefficient above.