**Capstone: Software in the loop simulation of slung load system motion control**

by

Shiyu He

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

Master of Engineering

in

Control System

Department of Electrical and Computer Engineering
University of Alberta

# Abstract

This project introduces a novel algorithm to generate the quasi-static feedback (QSF) controller for differentially flat nonlinear systems. This design exactly linearizes the closed-loop system in new state coordinates. The linearizing feedback has an important static dependence on the state and does not require dynamics in the controller which can save considerable computing resources when the system is complicated. To further demonstrate this quasi-static feedback algorithm, we give an example of the kinematic car, then derive the output tracking controller, and simulate it in Matlab. In the Next step, we apply the algorithm to a Slung Load System (SLS) which is a flat system consisting of a multirotor drone and suspended payload. After linearization, a straightforward output tracking control is used to ensure that error dynamics in the design coordinates are linear and exponentially stable. The control design is tested in an open-source PX4 Software-in-the-Loop (SITL) simulation. An automatic software pipeline is presented for transforming the symbolic controller expression into a C++ executable that can run on a real-world autopilot.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1

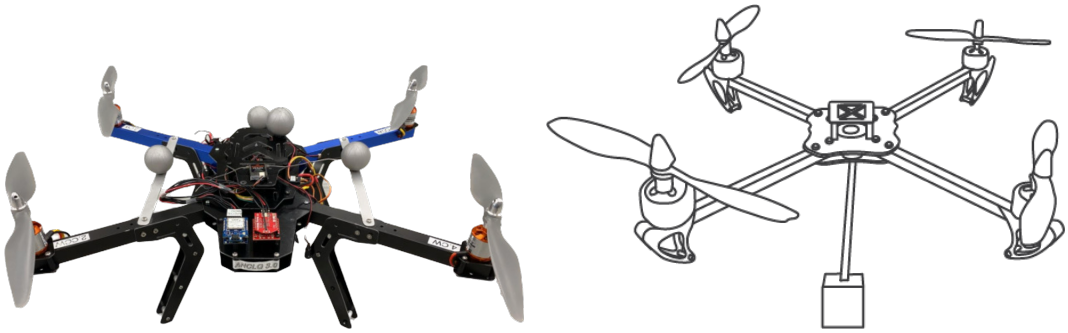
# Introduction

With the development of sensors, high-speed processors and Global Positioning Systems, unmanned aerial vehicles (UAV) technology has more and more mature. Because of its high flexibility and strong movement capability, its usage scenarios are becoming extensive. Nowadays it can be used in agriculture, aerial photography, reconnaissance and so on. Amount all kinds of UAVs, quadrotor is the most common type that has four rotors. Because of its four rotors, the quadrotors are allowed to have smaller propeller diameters and hence smaller vehicle size, which is an advantage compared with other flying vehicles like helicopters[1]. The picture of the quadrotor is shown in Figure 1.1a.

## 1.1 Slung Load System

Quadrotors often are utilized in transferring the payload from one location to another. For example, it can deliver packages in urban areas; provide supplies in conflict zones, among others. Under those usage scenarios, ground vehicles cannot be utilized or are more time-consuming than drones. However, Slung Load System (SLS), which is composed of a quadrotor, cable and load, can be employed. As a matter of fact, load transportation using quadrotors attends commercial, militarian, and civilian interests[2]. The schematic diagram of SLS is shown in Figure 1.1b.

Despite the fact that SLSs are a desirable option for load transportation, the load may be subjected to large oscillations due to external disturbances or the acceleration of the vehicle



(a) ANCL-Q3 quadrotor vehicle (b) Slung load system

Figure 1.1: Quadrotor and slung load system

itself, which will make the whole system unstable. How to achieve a high-performance and rigorous motion control system is a challenge for the researcher.

## 1.2 Software in the Loop

Software-in-the-loop (SITL) is a method of testing and validating code in a simulation environment in order to quickly and cost-effectively catch bugs and improve the quality of the code. Typically, SITL testing is conducted in the early stages of the software development process, while the more complex, expensive hardware-in-the-loop (HITL) testing is done in later stages[3]. This project focuses on the SITL simulation based on PX4 firmware and Robotic Operating System (ROS) while the model was developed in Gazebo. The software and firmware used in this project are the following:

**PX4** is an open-source flight control software powering all kinds of vehicles from racing and cargo drones to ground vehicles. It is hosted by Dronecode, a non-profit organization, developed by world-class developers from industry and academia, and supported by an active worldwide community[4]. The PX4 platform offers powerful compatibility and extensibility: It provides an interface (MAVLink) between the flight controller and companion computers to run them as a distributed system, and the cross-platform API supports writing software packages that can be executed on the microcontroller or on the ROS/Linux[5]. It supports SITL simulation with plenti-

ful simulators such as Gazebo and jMAVSim. The communication between the flight stack and the simulator is accomplished by MAVLink API as well.

**Gazebo** is an open-source 3D robotics simulator. It integrated the Open Dynamics Engine (ODE) physics engine which is composed of the rigid body dynamics simulation engine and collision detection engine so that it can efficiently simulate the dynamic interaction between bodies in space. Besides, a major feature of Gazebo is the ability to easily modify or create robots, actuators, sensors, and even the world environment[6].

**Robot Operating System (ROS)** is an open-source robotics middleware suite and provides a structured communications layer above the host operating systems[7]. With its help, researchers and developers can build and reuse code between robotics applications. ROS can be used with PX4 for drone application development, especially in the automated vehicle control and computer vision field. Combined with Gazebo, ROS makes the simulation of robots easily achieved.

## 1.3 Overview of Project

In **chapter 1**, we introduced the concept of SLSs and give the definition of SITL simulation and the corresponding software and framework used in it.

In **chapter 2**, the theoretical background has been presented. We give the definition of differential flatness which is an important property of the nonlinear system since the flat system is particularly advantageous for solving trajectory planning. Besides, we also give the basic concepts of three different kinds of state feedback linearization which is a common strategy employed in nonlinear control to control nonlinear.

In **chapter 3**, we separately presented the basic quadrotor model and the slung load system which include a rod and a pendulum based on the classic quadrotor model. In addition, we introduced the input parameterization for simulation.

In **chapter 4**, we first introduced the mathematical definition of Lie Derivative, relative degree, etc. Then the quasi-static feedback algorithm was defined which can generate the Quasi-static feedback controller for a flat nonlinear system.

In **chapter 5**, we used an example of the kinematic car to demonstrate how to utilize the quasi-static feedback algorithm. Then we design the controller for it to achieve trajectory tracking and provide the corresponding Matlab simulation result.

In **chapter 6**, we utilized the quasi-static feedback algorithm to design the controller for SLSs and used it to realize trajectory tracking.

In **chapter 7**, we give the parameters used in SITL simulation and present the simulation results about stabilization and trajectory tracking.

In **chapter 8**, we summarize this project.

In **Appendix A**, we provide more detail about the SLS model mentioned in chapter 3, and in **Appendix B** we give the maple code used to design the quasi-static feedback controller for kinematic car mentioned in chapter 5.

# Chapter 2

# Theoretical Background

## 2.1 Differential Flatness

Differentially flat systems have been introduced by Fliess [8], and its applications including planar vertical take-off and landing aircraft (PVTOL) [9], chemical reactor models [10] and mobile wheeled robots [11]. The practical definition of differential flatness is following:

The control system

$$\dot{x} = f(x,u), x \in \mathbb{R}^n, u \in \mathbb{R}^m \tag{2.1}$$

is said to be differentially flat or flat if there exist smooth maps $h$, $\mathscr{A}$ and $\mathscr{B}$, with $\mathscr{A}$ and $\mathscr{B}$ locally surjective, defined on open neighborhoods of $\mathbb{R}^n \times (\mathbb{R}^m)^{\rho+1}$, $(\mathbb{R}^m)^{r+1}$, and $(\mathbb{R}^m)^{r+2}$ respectively, such that:

$$y = h(x,u,\dot{u},...,u^{(\rho)}) \tag{2.2a}$$

$$x = \mathscr{A}(y,\dot{y},...,y^{(r)}) \tag{2.2b}$$

$$u = \mathscr{B}(y,\dot{y},...,y^{(r)},y^{(r+1)}), \tag{2.2c}$$

where $\rho$ and $r$ are positive integers, and the components of $y$ are differentially independent, i.e., they satisfy no differential equation of the form:

$$P(y,\dot{y},...,y^{(r)}) = 0.$$

That is to say a system with $m$ inputs is flat if the state $x$ and input $u$ can be parameterized by (fictitious) output functions $y_1$,...,$y_m$. The reason that y and its derivatives need to

be differentially unrelated is that the components of y can be independently designed to parameterize the state and input. This property is remarkable since trajectory tracking by open- or closed-loop control is relatively easy for flat systems and there are indeed many processes that can be modeled as flat systems[12].

## 2.2 State Feedback Linearization

The problem of transforming a nonlinear system into a linear controllable system by state feedback and change of coordinates is known as the feedback linearization problem (FBLP). As we all know, Jacobian Linearization can also linearize the model and is easy to use. However, it is important to note that the Jabocian model only represents the model at the equilibrium point $(x_0, u_0)$. Therefore, the control strategy based on a linearized model may yield unsatisfactory performance and robustness at other operating points. But the Feedback Linearization can produce a linear model that is an exact representation of the original nonlinear model over a large set of operating conditions.

### 2.2.1 Static and Dynamic Feedback Linearization

Dynamic Feedback Linearization (DFBL) can be achieved by the control law which is a dynamic function of the system state vector, which means the control unit itself has a dynamic behavior; while the control law of Static Feedback Linearization (SFBL) is a static function of the states. Next, we introduce the mathematical expression of DFBL:

Given a nonlinear system

$$\dot{x} = f(x,u), x \in \mathsf{M} \subset \mathbb{R}^n, u \subset \mathbb{R}^m \tag{2.3}$$

where $\mathsf{M}$ is an open set. Let $x_0 \subset \mathsf{M}$. The DFBL about $x_0$ is to find:

1. A dynamic state feedback which has the following form:

$$\begin{aligned} \dot{\xi} &= a(x,\xi,v) \\ u &= b(x,\xi,v), \end{aligned} \tag{2.4}$$

   where $\xi \subset \mathbb{R}^q$ named compensator state and $v \subset \mathbb{R}^m$ is the new control input, and $a$, $b$ are smooth functions locally defined on an open neighbourhood of $\mathbb{R}^n \times \mathbb{R}^q \times \mathbb{R}^m$ containing $(x_0, \xi_0, v_0)$.

2. A coordinate change has the form $z = \Phi(x,\xi)$ which was locally defined on an open neighbourhood of $\mathbb{R}^n \times \mathbb{R}^q$ containing$(x_0, \xi_0)$.

As the result, the closed-loop system can be represented as

$$\dot{x} = f(x, b(x,\xi,v))$$

$$\dot{\xi} = a(x,\xi,v),$$

which is equivalent to a linear controllable system: $\dot{z} = Fz + Gv$ after applying the transformation $\Phi$, where the pair $(F,G)$ is controllable. If the above condition holds, then the system is *dynamic feedback linearizable* (DFBL) at $x_0$; If the above condition holds with $q = 0$, i.e., there are no compensator states $\xi$, then the system is *static feedback linearizable* (SFBL) at $x_0$, and its control law will have the form:

$$u = b(x,v) \tag{2.5}$$

### 2.2.2 Quasi-static Feedback Linearization

Quasi-static Feedback was first introduced by Delaleau and Fliess [13]. Consider a system:

$$\dot{x} = f(x) + \sum_{i=1}^{m} g_i(x) u_i \tag{2.6a}$$

$$y_i = h_i(x), \quad 1 \le i \le m \tag{2.6b}$$

with vector fields $f, g_i : \mathscr{M} \to \mathbb{R}^n$ and output functions $h_i : \mathscr{M} \to \mathbb{R}$ defined on an open subset $\mathscr{M} \subset \mathbb{R}^n$. The Quasi-static Feedback (QSF) is taken as $u = u(x, v, \dot{v}, \ddot{v}, \ldots, v^{(\rho)})$, where $v$ is auxiliary with a dimension $m$. QSF assigns input $v$ to be a time derivative of $y$

$$y^{(r_i)} = v_i, \quad 1 \leq i \leq m \tag{2.7}$$

where $y^{(i)}$ indicates the $i$th time derivative of $y$. Specifically, as we will show in 6.2, when considering output tracking, $v$ is a linear function of the output tracking error and its time derivatives, so it is generally not necessary to differentiate numerically[12].

The goal of QSF is to statically linearize flat systems that do not satisfy the conditions for static state feedback linearization. Compared with dynamic state feedback which is another option when the system is not SFBL, QSF gives a static function of the state, in other words, it requires no state augmentation. This feature leads to a more uncomplicated control design which is critical for onboard implementation such as autopilot where computing resources are limited[14].

# Chapter 3

# Quadrotor Modelling

This Chapter presents the dynamic model for the SLS.

## 3.1 Rotation Dynamic of the UAV

In this project, we choose a commonly used nonlinear rigid body model for the UAV which was introduced in [15]. There are two reference frames used in modeling namely the navigation frame $\mathscr{N}$ and the body frame $\mathscr{B}$. The navigation frame is assumed inertial and has its origin fixed to the center of a specific space. The corresponding basis is the orthonormal set of vectors $\{n_1, n_2, n_3\}$ which are oriented north, east and down respectively. The body frame has its origin fixed to the quadrotor's center of mass (CoM) and its basis $\{b_1, b_2, b_3\}$ is oriented forward, right, and down, respectively. The model of the quadrotor was shown in Figure 3.1.

To transform the vector from frame $\mathscr{B}$ to $\mathscr{N}$, the following equation can be used:

$$R_b^n(\eta) = R_3(\psi)R_2(\theta)R_1(\phi) = \begin{bmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\psi c_\phi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix} \tag{3.1}$$

Where the rotation matrix $R \in SO(3) = \left\{R \in \mathbb{R}^{3\times 3} : RR^T = I, det(R) = 1\right\}$, $\eta = [\phi, \theta, \psi]^T$ which named ZYX Euler angles and represent roll-pitch-yaw respectively (see Figure 1.1b), $c_x = \cos(x)$ and $s_x = \sin(x)$. The inverse of the $R_b^n$ is used to convert the vector from the navigation frame $\mathscr{N}$ to body frame $\mathscr{B}$.

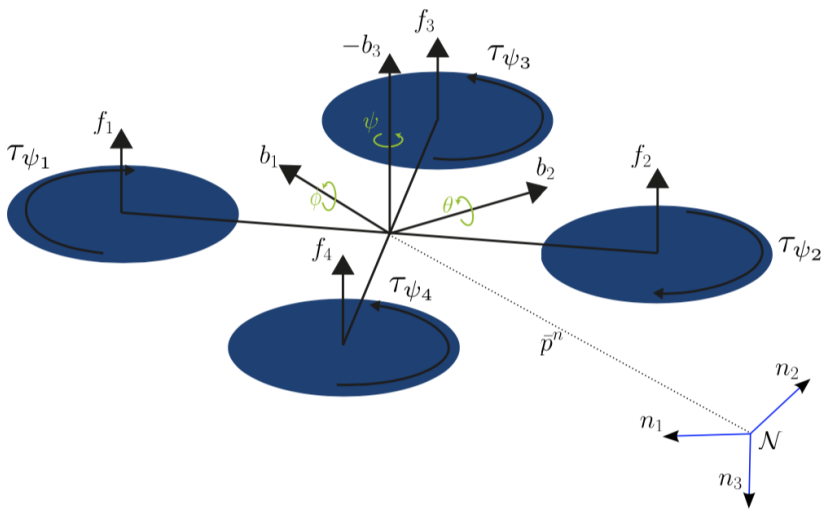



Figure 3.1: Model of quadrotor based on [16]

The Newton-Euler formulation is used to achieve translational and rotational dynamics of the quadrotor:

$$\dot{p}_Q = v_Q \tag{3.2a}$$

$$m_Q \dot{v}_Q = m_Q g n_3 - R\bar{u} n_3 \tag{3.2b}$$

$$\dot{R} = RS(\omega) \tag{3.2c}$$

$$J\dot{\omega} = -\omega \times J\omega + \tau \tag{3.2d}$$

where $p_Q \in \mathbb{R}^3$ is drone's position in $\mathscr{N}$, $v_Q \in \mathbb{R}^3$ is drone's linear velocity in $\mathscr{N}$, $g$ is the gravitational acceleration, $m_Q$ is the mass of drone, $R \in SO(3)$ is the rotation matrix from $\mathscr{N}$ to $\mathscr{B}$ which defined in (3.1), $n_3 \in \mathbb{R}^3$ is the third basis vector of $\mathscr{N}$, $\bar{u} \in \mathbb{R}_{\geq 0}$ is the total propeller thrust from four motors, $\omega \in \mathbb{R}^3$ is the angular velocity of drone in $\mathscr{B}$, $J = \text{diag}(J_1, J_2, J_3) \in \mathbb{R}^{3\times 3}$ is the drone's inertia about CoM. $\tau \in \mathbb{R}^3$ is propeller torque expressed in $\mathscr{B}$ The skew operator $S(\cdot) : \mathbb{R}^3 \to so(3)$ in (3.6c) 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 $so(3) = \{M \in \mathbb{R}^{3\times 3} : M^T = -M\}$

## 3.2 SLS Dynamic

Compared with the quadrotor model, the slung load system has a pendulum that pivots about the UAV CoM. The model and corresponding notation have been shown in Figure 3.2.

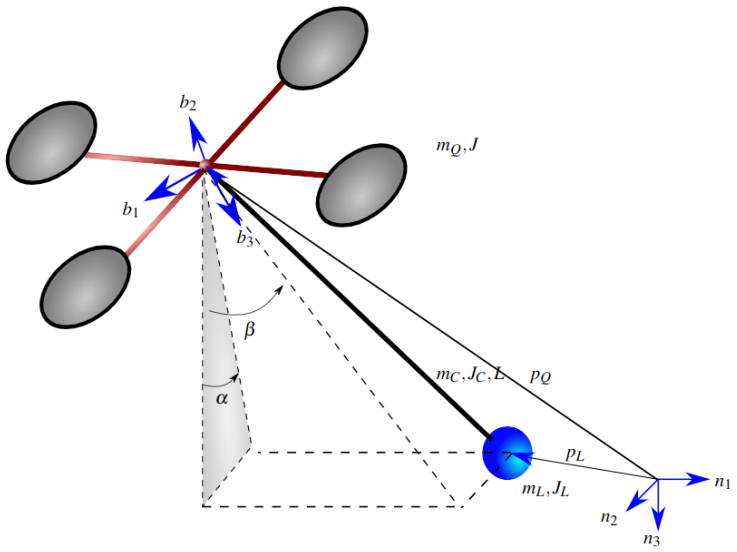



Figure 3.2: The SLS is modeled as a spherical pendulum and multirotor drone.

We denote pendulum position $p_L$, pendulum attitude $q$, and drone attitude $R$. The SLS configuration variable $[p_L, R, q] \in SE(3) \times \mathbb{S}^2$. $q$ is an unit vector expressed in $\mathcal{N}$ and parameterized with angles $\alpha$ and $\beta$ where $\alpha$ is a rotation angle about $n_1$ and $\beta$ is about $n_2$. We have

$$q = R_{n_1}(\alpha) R_{n_2}(\beta) n_3 = [s_\beta, -s_\alpha c_\beta, c_\alpha c_\beta]^T$$

where $R_{n_1}$ and $R_{n_2}$ are elementary rotation matrices about $n_1$ and $n_2$ axis, respectively. Load position $p_L$ and quadrotor position $p_Q$ can be related by following equation:

$$p_L = p_Q + Lq = p_Q + L[s_\beta, -s_\alpha c_\beta, c_\alpha c_\beta]^T \tag{3.3}$$

where $L$ is rod length. Differentiate equation (3.3), we can get:

$$v_L = v_Q + L\dot{q}$$

$$\dot{v}_L = \dot{v}_Q + L\ddot{q} \tag{3.4}$$

To improve accuracy, the model includes rod mass and inertia compared with [14]. The relation between the rod CoM $p_p$ and drone position $p_Q$ is

$$p_p = p_Q + \frac{1}{2}Lq \tag{3.5}$$

By differentiating, we can get:

$$v_p = v_Q + \frac{1}{2}L\dot{q}, \quad \dot{v}_p = \dot{v}_Q + \frac{1}{2}L\ddot{q}$$

As for the Drone part for the SLS, the model introduced in (3.2) should be modified by adding rod tension on the RHS of equation (3.6b). Then the model will be:

$$\dot{p}_Q = v_Q \tag{3.6a}$$

$$m_Q\dot{v}_Q = m_Q g n_3 - R\bar{u}n_3 + Tq \tag{3.6b}$$

$$\dot{R} = RS(\omega) \tag{3.6c}$$

$$J\dot{\omega} = -\omega \times J\omega + \tau \tag{3.6d}$$

Where $T \geq 0$ denotes rod tension. For the rotational dynamics (3.6c), when calculate the differential of rotation matrix $\dot{R}$, the rate of ZYX Euler angle $\eta = [\phi, \theta, \psi]^T \in \mathbb{R}^3$ can be found in terms of the angular velocity $\omega$ by following relation:

$$\dot{\eta} = W(\eta)\omega \tag{3.7}$$

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 $t_\theta = \tan\theta$.

Next, we consider the pendulum dynamics of the SLSs, which are:

$$\dot{p}_L = v_L \tag{3.8a}$$

$$m_L \dot{v}_L = -Tq + (m_L + m_C) g n_3 \tag{3.8b}$$

$$\dot{q} = \omega_L \times q \tag{3.8c}$$

$$\begin{aligned} J_L \dot{\omega}_L = &-\omega_L \times J_L \omega_L + Lq \times (m_L g n_3 - m_L \dot{v}_Q) \\ &+ \frac{1}{2} Lq \times (m_C g n_3 - m_C \dot{v}_Q) \end{aligned} \tag{3.8d}$$

where (3.8a) and (3.8b) are translational dynamics while (3.8c) and (3.8d) are rotational dynamics, $J_L$ is the inertia of load about the drone's CoM. $m_L$ is the mass of the load. Using Steiner's Theorem [17], the $J_L$ can be represented as a function of the relative position between the load and drone:

$$J_L = m_L L^2 (I - qq^T) + J_C + m_C \frac{L^2}{4} (I - qq^T) \tag{3.9}$$

where $m_C$ is the mass of rod and $J_C$ is the inertia matrix. We assume the rod is thin enough so that its radius is equal to 0. So the $J_C$ relative to its CoM given by

$$J_C = \text{diag}(\frac{1}{12} m_C L^2, \frac{1}{12} m_C L^2, 0)$$

To eliminate the rod tension $T$, combining the equations (3.8b) and (3.6b) yields:

$$m_Q \dot{v}_Q + m_L \dot{v}_L = (m_Q + m_L + m_C) g n_3 - R\bar{u} n_3 \tag{3.10}$$

Substituting for $\dot{v}_Q$ in (3.10) using (3.4) and the derivative of (3.8c) $\ddot{q} = \dot{\omega}_L \times q + \omega_L \times \dot{q}$ , we get the translational dynamics for the load:

$$\begin{aligned} (m_L + m_Q) \dot{v}_L &= (m_L + m_Q + m_C) g n_3 - R\bar{u} n_3 + m_Q L \ddot{q} \\ &= (m_L + m_Q + m_C) g n_3 - R\bar{u} n_3 + m_Q L (\dot{\omega}_L \times q + \omega_L \times \dot{q}) \end{aligned} \tag{3.11}$$

Then we combine (3.11) and (3.4) and get:

$$\dot{v}_Q = -\frac{m_L}{m_Q + m_L} L \ddot{q} + (1 + \frac{m_C}{m_L + m_Q}) g n_3 - \frac{R\bar{u} n_3}{m_Q + m_L} \tag{3.12}$$

Substituting (3.9), and (3.12) into (3.8d), we obtain the rotational dynamics of the load expressed in $\mathcal{N}$:

$$J_L\dot{\omega}_L = -\omega_L \times J_L\omega_L + Lq \times \Big((m_L + \frac{m_C}{2}) \\ (gn_3 + \frac{m_L}{m_Q+m_L}L\ddot{q} - (1+\frac{m_C}{m_L+m_Q})gn_3 + \frac{R\bar{u}n_3}{m_Q+m_L})\Big) \quad (3.13)$$

In addition, we have the relation between $\omega_L$ and $\gamma_\alpha, \gamma_\beta$

$$\omega_L = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \gamma_\alpha + \begin{bmatrix} 0 \\ c_\alpha \\ s_\alpha \end{bmatrix} \gamma_\beta \quad (3.14)$$

and

$$\dot{\omega}_L = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \dot{\gamma}_\alpha + \begin{bmatrix} 0 \\ c_\alpha \\ s_\alpha \end{bmatrix} \dot{\gamma}_\beta + \begin{bmatrix} 0 \\ -s_\alpha \\ c_\alpha \end{bmatrix} \gamma_\alpha\gamma_\beta \\ = R_{n_1}(\alpha) \begin{bmatrix} \dot{\gamma}_\alpha \\ \dot{\gamma}_\beta \\ \gamma_\alpha\gamma_\beta \end{bmatrix} \quad (3.15)$$

Substituting (3.14) and (3.15) into (3.11) and (3.13), and solving for $\dot{v}_L, \dot{\gamma}_\alpha, \dot{\gamma}_\beta$, we obtain the SLS dynamics

$$\dot{x} = \begin{bmatrix} v \\ \gamma_\alpha \\ \gamma_\beta \\ W(\eta)\omega \\ -s_\beta(\gamma_\alpha^2 c_\beta^2 + \gamma_\beta^2)LM_0 \\ s_\alpha c_\beta(\gamma_\alpha^2 c_\beta^2 + \gamma_\beta^2)LM_0 \\ g - c_\alpha c_\beta(\gamma_\alpha^2 c_\beta^2 + \gamma_\beta^2)LM_0 \\ 2\gamma_\alpha\gamma_\beta t_\beta \\ -\gamma_\alpha^2 c_\beta s_\beta \\ -J^{-1}S(\omega)J\omega \end{bmatrix} + \begin{bmatrix} 0_{8\times1} & 0_{8\times3} \\ \bar{g}(x) & 0_{5\times3} \\ 0_{3\times1} & J^{-1} \end{bmatrix} u \quad (3.16)$$

where $x = \left[p_L^T, \alpha, \beta, \eta^T, v_L^T, \gamma_\alpha, \gamma_\beta, \omega^T\right]^T \in \mathbb{R}^{16}$, $u = [\bar{u}, \tau^T]^T \in \mathbb{R}^4$, and $M_0 = (2m_Q + m_L)/(2(m_Q + m_L + m_C))$. The expression for $\bar{g}$ has been shown in Appendix A. The drift vector field of (3.16) has singularities at $c_\beta = c_\theta = 0$ due to parametrizations used for the orientation of the drone and pendulum. therefore, domain $\mathscr{M}$ of $x$ is taken to be a subset of $\mathbb{R}^{16}$ including 0 but excluding these singularities. Below we will notice those singularities appear in the controller. However, from a practical point of view, those points are not likely involved in typical SLS motion. Similarly, $c_\beta = 0$ is not physically possible since it means the rod collides with the frame of the drone.

## 3.3 System Inputs

The output of the controller is total thrust $\bar{u}$ and torque $\tau$ in units of N and N$\cdot$m, respectively. The relation between $[\bar{u}, \tau]$ and rotor speed $\Omega$ is

$$\begin{bmatrix} \bar{u} \\ \tau \end{bmatrix} = C_T \begin{bmatrix} 1 & 1 & 1 & 1 \\ -\ell_1 & \ell_3 & \ell_1 & -\ell_3 \\ \ell_2 & -\ell_4 & \ell_2 & -\ell_4 \\ C_M & C_M & -C_M & -C_M \end{bmatrix} \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix} \tag{3.17}$$

where $\Omega_i, 1 \leq i \leq 4$ is the ith rotor speed, $C_T$ is the rotor thrust constant, and $C_M$ is the torque constant. However, PX4 expects a normalized thrust $\tilde{u} \in [0,1]$ and torque $\tilde{\tau} \in [-1,1]$. PX4 converts $\tilde{u}, \tilde{\tau}$ with its Mixer into normalized rotor speed commands $\tilde{\Omega}_i, 1 \leq i \leq 4$, using parameters describing the geometry of the multirotor's frame. In PX4, this mapping is given by $\left[\tilde{\Omega}_1^2, \tilde{\Omega}_2^2, \tilde{\Omega}_3^2, \tilde{\Omega}_4^2\right]^T = K_P\,[\tilde{u}, \tilde{\tau}]^T$, where $K_P$ called mixer which is a $4 \times 4$ matrix. We give those parameter values in chapter 7.

Gazebo receives a normalized rotor speed command $[\tilde{\Omega}_1, \tilde{\Omega}_2, \tilde{\Omega}_3, \tilde{\Omega}_4]$ which needs further transform to obtain a rotor speed command by the following equation:

$$\Omega_i = C_\Omega \tilde{\Omega}_i, \quad 1 \leq i \leq 4 \tag{3.18}$$

where $C_\Omega$ is the scaling constant. In the last step, Gazebo uses a classical quadratic rotor model to generate individual propeller thrust $C_T\Omega^2$ and torque $C_M C_T \Omega^2$ from rotor

speed command $\Omega$. With the assumption the above modeling is known exactly, we can scale the controller output before feeding it to PX4 so that Gazebo applies the desired values of $\bar{u}$ and $\tau$. Given output from the controller $\bar{u}, \tau = [\tau_1, \tau_2, \tau_3]^T$ we scale $[\tilde{u}, \tilde{\tau}^T]$ as $[\bar{u}/S_1, \tau_1/S_2, \tau_2/S_3, \tau_3/S_4]$.

We can summarize the above discussion with:

$$\begin{bmatrix} \bar{u}_G \\ \tau_G \end{bmatrix} = C_T C_\Omega^2 \begin{bmatrix} 1 & 1 & 1 & 1 \\ -\ell_1 & \ell_3 & \ell_1 & -\ell_3 \\ \ell_2 & -\ell_4 & \ell_2 & -\ell_4 \\ C_M & C_M & -C_M & -C_M \end{bmatrix} K_P \begin{bmatrix} \bar{u}/S_1 \\ \tau_1/S_2 \\ \tau_2/S_3 \\ \tau_3/S_4 \end{bmatrix} \tag{3.19}$$

where $[\bar{u}_G, \tau_G]$ is the total thrust and torque in Gazebo.

# Chapter 4

# Quasi-static Feedback Controller Design

To achieve quasi-static feedback linearization of the SLS system, we introduced the controller design method based on Quasi-static Feedback Algorithm which was first presented by Zifei [14].

## 4.1 Mathematical Definition and notation

We first provide the definition of **Lie Derivative** and **Relative degree**:

### 4.1.1 Lie Derivative

The **Lie derivative** of a function $\lambda : \mathscr{M} \to \mathbb{R}$ along the vector field $f$ is defined by $L_f\lambda(x) = \frac{\partial \lambda}{\partial x} f(x)$. It is commonly used to represent the time derivative of the output along the solution of the system. For example, consider a nonlinear system

$$\dot{x} = f(x) + g(x)u, \quad y = h(x)$$

where $f$, $g$, and $h$ are sufficiently smooth in domain $\mathscr{M}$. Calculating the time derivative of the output along the solution of the system yields that

$$\dot{y} = \frac{\partial h}{\partial x}[f(x) + g(x)u] = L_f h(x) + L_g h(x)u$$

### 4.1.2 Relative degree

The **Relative degree** of the SISO system

$$\dot{x} = f(x) + g(x)u, x \in \mathbb{R}^n \quad y = h(x)$$

is $r, 1 \leq r \leq n$ in $U_0 \subset \mathscr{M}$, if $\forall x \in U_0$, the following two conditions are met:

1. $L_g L_f^{i-1} h(x) = 0, i = 1, 2, \cdots, r-1$
2. $L_g L_f^{r-1} h(x) \neq 0$

### 4.1.3 Control Characteristic Indices & Decoupling Matrix

Consider multivariable nonlinear systems with as many inputs ($m$) as outputs

$$\begin{aligned} \dot{x} &= f(x) + \sum_{m}^{i=1} g_i(x)u_i = f(x) + G(x)u, \quad x \in \mathbb{R}^n \\ y_i &= h_j(x), \quad 1 \leq j \leq m \end{aligned} \tag{4.1}$$

with $f, g_1, \cdots g_m$ smooth vector fields, $h_1, \cdots, h_m$ smooth real valued functions, $rank\ G(0) = m$ and $rank\{dh_1(0), \cdots, dh_m(0)\} = m$. A set of $m$ integers $[r_1, \cdots, r_m]$ called **control characteristic indices** [18] are associated with system (4.1) in $U_0$, a neighborhood of the origin, as follows $(1 \leq i \leq m)$:

$$L_{g_j} L_f^k h_i(x) = 0, \quad 1 \leq j \leq m, 0 \leq k \leq r_i - 2, \forall x \in U_0$$

$$L_{g_j} L_f^{r_i - 1} h_i(x) \neq 0, \quad \text{for some } j, 1 \leq j \leq m, \forall x \in U_0$$

We say that $r_i = \infty$ if

$$L_{g_j} L_f^k h_i(x) = 0, \quad \forall x \in U_0, \forall k \geq 0$$

The above definition is given about the origin: it may be given around any point $\bar{x} \in \mathbb{R}^n$ such that $rank\ G(\bar{x}) = m$ and $rank\{dh_1(\bar{x}), \cdots, dh_m(\bar{x})\} = m$.

If $r_i \leq \infty, 1 \leq i \leq m$, then each $r_i$ is equal to the least order of the time derivative of the output $y_i$ which is directly affected at least by some input $u_j, 1 \leq j \leq m$.

If $r_i \leq \infty, 1 \leq i \leq m$, an $m \times m$ matrix, called the **decoupling matrix** [18], is defined as

$$D(x) = \begin{bmatrix} L_{g_1} L_f^{r_1-1} h_1(x) & \dots & L_{g_m} L_f^{r_1-1} h_1(x) \\ \vdots & \cdots & \vdots \\ L_{g_1} L_f^{r_m-1} h_m(x) & \dots & L_{g_m} L_f^{r_m-1} h_m(x) \end{bmatrix} \tag{4.2}$$

One should note that the existence of indices $[r_1, \cdots, r_m]$ does not imply a well-defined relative degree since that requires the decoupling matrix is nonsingular at $x_0$.

## 4.2 Quasi-Static Feedback Algorithm (QSFA)

In this section, the variables used superscript $^{\langle i \rangle}$ to keep track of the algorithm iteration. We begin the QSFA at Step 0 and assume the decoupling matrix $D^{\langle 0 \rangle} = D$ given by (4.2) has a constant rank less than $m$ about $x_0 \in \mathscr{M}$ where $r^{\langle 0 \rangle} = [r_1^{\langle 0 \rangle}, \dots, r_m^{\langle 0 \rangle}] = r$. Define $s^{\langle i \rangle} = rank\ (D^{\langle i \rangle}(x_0))$, and $y^{\langle 0 \rangle} = y^{(r^{\langle 0 \rangle})} = [y_1^{(r_1^{\langle 0 \rangle})}, \dots, y_m^{(r_m^{\langle 0 \rangle})}]^T$.

**Step** 0**:** From the definition of $r^{\langle 0 \rangle}$ we can express $y^{(r^{\langle 0 \rangle})}$ as

$$y^{\langle 0 \rangle} = y^{(r^{\langle 0 \rangle})} = a_0(x) + D^{\langle 0 \rangle} u \tag{4.3}$$

We reorder and decompose $y^{\langle 0 \rangle}$ as

$$y^{\langle 0 \rangle} = \begin{bmatrix} \tilde{y}^{\langle 0 \rangle} \\ \hat{y}^{\langle 0 \rangle} \end{bmatrix} \tag{4.4}$$

where $\tilde{y}^{\langle 0 \rangle}$ corresponds to the first $s^{\langle 0 \rangle}$ independent rows of $D^{\langle 0 \rangle}$, and $\hat{y}^{\langle 0 \rangle}$ corresponds to the dependent rows of $D^{\langle 0 \rangle}$. We introduce auxiliary input $v_1 = \tilde{y}^{\langle 0 \rangle}$. Since the last $m - s^{\langle 0 \rangle}$ rows of $D^{\langle 0 \rangle}$ are linearly dependent on the first $s^{\langle 0 \rangle}$ rows, we have

$$\tilde{y}^{\langle 0 \rangle} = \tilde{a}_0(x) + \tilde{b}_0(x) u = v_1 \tag{4.5}$$

$$\hat{y}^{\langle 0 \rangle} = \hat{y}^{\langle 0 \rangle}(x, v_1) \tag{4.6}$$

where $\hat{y}^{\langle 0 \rangle}$ is affine in $v_1$ and $\tilde{b}_0(x)$ are the first $s^{\langle 0 \rangle}$ independent rows of $D^{\langle 0 \rangle}$. We remark that in (4.6) we have eliminated $u$ using (4.5). This ensures time derivatives of $u$ do not appear $\hat{y}^{\langle 0 \rangle}$ below and we have a static relation between $u$ and $v$ and its time derivatives.

**Step 1:** We compute index $r^{\langle 1\rangle}$ using the definition of $r$ but with output $\hat{y}^{\langle 0\rangle}$. We have

$$\dot{\hat{y}}^{\langle 0\rangle} = \hat{y}^{(r^{\langle 0\rangle}+1)} = \frac{\partial \hat{y}^{\langle 0\rangle}}{\partial x}[f(x)+\sum_{i=1}^{m} g_i(x)u_i] + \frac{\partial \hat{y}^{\langle 0\rangle}}{\partial v_1}\dot{v}_1 \tag{4.7}$$

$$= L_f \hat{y}^{\langle 0\rangle} + \frac{\partial \hat{y}^{\langle 0\rangle}}{\partial v_1}\dot{v}_1 + \sum_{i=1}^{m} L_{g_i}\hat{y}^{\langle 0\rangle}u_i$$

$$= a_1(x,\dot{v}_1) + b_1(x,v_1)u \tag{4.8}$$

where

$$b_1 = \begin{bmatrix} L_{g_1}\hat{y}_1^{\langle 0\rangle} & \dots & L_{g_m}\hat{y}_1^{\langle 0\rangle} \\ \vdots & \dots & \vdots \\ L_{g_1}\hat{y}_{m-s^{\langle 0\rangle}}^{\langle 0\rangle} & \dots & L_{g_m}\hat{y}_{m-s^{\langle 0\rangle}}^{\langle 0\rangle} \end{bmatrix} \in \mathbb{R}^{(m-s^{\langle 0\rangle})\times m} \tag{4.9}$$

Three cases are possible depending on the value of $b_1$. If $b_1$ is identically zero, we must continue taking time derivatives of $\dot{\hat{y}}^{\langle 0\rangle}$. If any row of $b_1$ is linearly dependent on the rows of $\tilde{b}_0(x)$ in (4.5), it should be expressed as a function of $(x,v_1)$ as in (4.6), and then time derivatives of these rows are computed. If all rows of $b_1$ are linearly independent of the rows of $\tilde{b}_0(x)$ in (4.5), we define $r^{\langle 1\rangle} = [r_1^{\langle 1\rangle},\dots,r_{m-s^{\langle 0\rangle}}^{\langle 1\rangle}]$, so that $y^{\langle 1\rangle} = (\hat{y}^{\langle 0\rangle})^{(r^{\langle 1\rangle})}$. Every row of the corresponding decoupling matrix $D^{\langle 1\rangle}$ is linearly independent of the rows of $\tilde{b}_0$. We decompose $y^{\langle 1\rangle}$ as

$$y^{\langle 1\rangle} = \begin{bmatrix} \tilde{y}^{\langle 1\rangle} \\ \hat{y}^{\langle 1\rangle} \end{bmatrix} \tag{4.10}$$

where $\tilde{y}^{\langle 1\rangle}$ corresponds to the first $s^{\langle 1\rangle}$ independent rows of $D^{\langle 1\rangle}$. We introduce auxiliary input $v_2 = \tilde{y}^{\langle 1\rangle}$. Similar to (4.5) and (4.6), $y^{\langle 1\rangle}$ can be written as

$$\tilde{y}^{\langle 1\rangle} = \tilde{a}_1(x,v_1,\dots,v_1^{(r^{\langle 1\rangle})}) + \tilde{b}_1(x,v_1,\dots,v_1^{(r^{\langle 1\rangle}-1)})u = v_2 \tag{4.11}$$

$$\hat{y}^{\langle 1\rangle} = \hat{y}^{\langle 1\rangle}(x,v_1,\dots,v_1^{(r^{\langle 1\rangle})},v_2) \tag{4.12}$$

As shown in (4.11), $\tilde{a}_1, \tilde{b}_1$ and $\hat{y}^{\langle 1\rangle}$ are functions of $x$ and auxiliary input $v$. To track this dependence conveniently, we introduce set $S_n^T$ which contains components of the auxiliary input and their time derivatives. The largest component of auxiliary input is denoted $n$ and $T = [T_1,\dots,T_n] \in \mathbb{N}^n$ is the highest order of derivatives for each input component. That is,

$$S_n^T = \{v_1,\dots,v_1^{(T_1)},v_2,\dots,v_2^{(T_2)},\dots,v_n,\dots,v_n^{(T_n)}\} \tag{4.13}$$

Hence, (4.11) and (4.12) can be written more compactly as

$$\tilde{y}^{\langle 1\rangle} = \tilde{a}_1(x, S_1^{T^{\langle 1\rangle}}) + \tilde{b}_1(x, S_1^{T^{\langle 1\rangle}-1})u = v_2 \tag{4.14}$$

$$\hat{y}^{\langle 1\rangle} = \hat{y}^{\langle 1\rangle}(x, S_1^{T^{\langle 1\rangle}}, v_2) \tag{4.15}$$

**Step $k$.** Suppose that in Steps 0 through $k$, $y^{\langle 0\rangle}, \ldots, y^{\langle k\rangle}$ have been defined so that

$$\begin{aligned}
\tilde{y}^{\langle 0\rangle} &= \tilde{a}_0(x) + \tilde{b}_0(x)u \\
&\vdots \\
\tilde{y}^{\langle k\rangle} &= \tilde{a}_k(x, S_k^{T^{\langle k\rangle}}) + \tilde{b}_k(x, S_k^{T^{\langle k\rangle}-1})u = v_{k+2} \\
\hat{y}^{\langle k\rangle} &= \hat{y}_k^{\langle k\rangle}(x, S_k^{T^{\langle k\rangle}}, v_{k+1})
\end{aligned}$$

where $\hat{y}^{\langle k\rangle}$ is affine in $v_{k+1}$.

Define $y^{\langle k+1\rangle} = (\hat{y}^{\langle k\rangle})^{(r^{\langle k+1\rangle})}$ which can be decomposed as before:

$$y^{\langle k+1\rangle} = \begin{bmatrix} \tilde{y}^{\langle k+1\rangle} \\ \hat{y}^{\langle k+1\rangle} \end{bmatrix} \tag{4.16}$$

where $\tilde{y}^{\langle k+1\rangle}$ are the first $s^{\langle k+1\rangle}$ independent rows of $D^{\langle k+1\rangle}$. Introducing auxiliary inputs $v_{k+2} = \tilde{y}^{\langle k+1\rangle}$, (4.16) can be written as

$$\begin{aligned}
\tilde{y}^{\langle k+1\rangle} &= \tilde{a}_{k+1}(x, S_{k+1}^{T^{\langle k+1\rangle}}) + \tilde{b}_k(x, S_{k+1}^{T^{\langle k+1\rangle}-1})u \\
\hat{y}^{\langle k+1\rangle} &= \hat{y}_{k+1}^{\langle k\rangle}(x, S_{k+1}^{T^{\langle k+1\rangle}}, v_{k+2})
\end{aligned}$$

If $s^{\langle 0\rangle} + s^{\langle 1\rangle} + \cdots + s^{\langle k+1\rangle} = m$, the algorithm terminates. Otherwise, we take the time derivative of $\hat{y}^{\langle k+1\rangle}$ and perform the next iteration.

Defining $[\tilde{y}^{\langle 0\rangle}, \ldots, \tilde{y}^{\langle k+1\rangle}]^T = [v_1, \ldots, v_{k+2}]^T = v \in \mathbb{R}^m$, and when the algorithm terminates, we have an invertible relation between the original input $u$ and auxiliary input $v$:

$$v = \begin{bmatrix} \tilde{a}_0(x) \\ \vdots \\ \tilde{a}_{k+1}(x, S_{k+1}^{T^{\langle k+1\rangle}}) \end{bmatrix} + D^\circ u \tag{4.17}$$

where $D^\circ$ is the final decoupling matrix with $\mathit{rank}\,(D^\circ) = m$.

# Chapter 5

# Exapmle of QSFA: Kinematic Car

In this Chapter, we give a kinematic car example utilizing the QSFA to achieve quasi-static feedback linearization. We will briefly introduce the model and then focus on the implementation of QSFA. The first chapter of the textbook **Flatness-Based Control** by Joachim Rudolph [19] gives more detail about the model.

## 5.1 Model of Kinematic Car

Consider a 4-wheel kinematic car shown in Figure 5.1 which needs to follow a smooth curve $\mathscr{C}$ starting at point $A$ and ending at point $B$. Let $Y$, the center of the rear axle, be the point chosen from the car to trace the path. And $\vec{Y}$ is the vector pointing from the origin of the stationary reference frame to $Y$. Similarly, we introduce $\vec{F}$ for the point $F$ which is on the mid-point of the front axle. In Cartesian coordinates, we might describe the vector by its components as

$$\vec{Y}(t) = \begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix}$$

Moreover, let $\theta$ define the orientation of the car with respect to the $y_1$-axis, and introduce $\phi$ as the angle of the front wheels with respect to the longitudinal axis of the car. Finally, let the constant parameter $l$ stand for the distance between the front and the rear axle centers, i.e., $l = \left|\vec{F} - \vec{Y}\right|$. By assuming rolling without slipping of the rear wheels, and denoting $v$

as the speed of the point $Y$, we have

$$\dot{\vec{Y}} = v\vec{\tau} = v \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix} \tag{5.1}$$

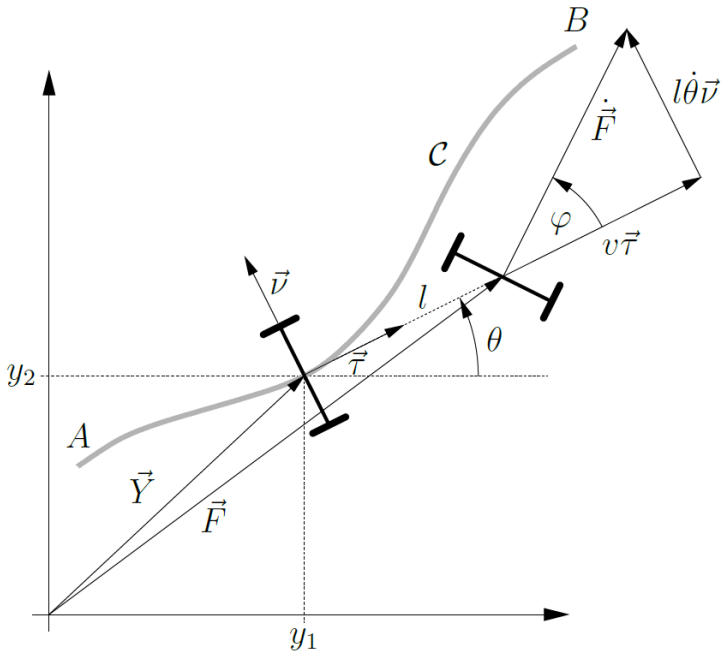



Figure 5.1: The car in a reference frame fixed in the plane, together with a moving frame defined by the unit tangent $\vec{\tau}$ and normal $\vec{\nu}$ to $\mathscr{C}$ at $\mathscr{Y}$.

Here $\vec{\tau}$ is the normalized vector tangent to the curve $\mathscr{C}$ at $\mathscr{Y}$. Similarly, rolling without slipping of the front wheels implies a constraint on the possible motion of the point $F$. This point is given by

$$\vec{F} = \vec{Y} + l\vec{\tau} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} + l \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix} \tag{5.2}$$

In order to calculate its derivative, we first calculate the derivative of $\vec{\tau}$, which is easy using its Cartesian representation:

$$\dot{\vec{\tau}} = \dot{\theta} \begin{bmatrix} -\sin(\theta) \\ \cos(\theta) \end{bmatrix} = \dot{\theta}\vec{\nu} \tag{5.3}$$

where we have introduced $\vec{\nu}$, which is the inward-pointing normal of $\mathscr{C}$ at $\mathscr{Y}$, as drawn in Figure 5.1. Now (5.1) and (5.3) may be used to express the velocity of the point $F$ as

$$\dot{\vec{F}} = v\vec{\tau} + l\dot{\theta}\vec{\nu}$$

which is also shown in Figure 5.1. If the front wheels are rolling without slipping, the velocity vector $\dot{\vec{F}}$ points in the direction of the wheels. That means, its orientation is given by the angle $\phi$. It directly follows that

$$\tan(\phi) = \frac{l\dot{\theta}}{v}$$

at $v \neq 0$. Combining this with (5.1), the two kinematic constraints introduced by assuming rolling without slipping, yield the following model in a stationary Cartesian reference frame:

$$\dot{y_1} = v\cos(\theta) \tag{5.4}$$

$$\dot{y_2} = v\sin(\theta) \tag{5.5}$$

$$\dot{\theta} = v\frac{\tan(\phi)}{l} \tag{5.6}$$

Given the above relation, we can derive the state space model of the kinematic car,

$$\dot{x} = f(x) + g(x)u = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} x + \begin{bmatrix} \cos(x_3) & 0 \\ \sin(x_3) & 0 \\ \tan(x_4)/l & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{5.7}$$

where state is $x = [y_1, y_2, \theta, \phi]^T$, input is $u = [v, \gamma_\phi]^T$ and output is $y = [y_1, y_2]^T$ .

## 5.2 Controller Design Based on QSFA

Suppose we already know that the kinematic car system is a flat system and the output $y = [y_1, y_2]^T$ is flat output, which was demonstrated in Rudolph's book [19], we can use QSFA to derive the feedback law $u = u(x, \delta, \dot{\delta}, \ddot{\delta}, \ldots,)$. One should note that to avoid the conflict between the notation of velocity and auxiliary input, we use $v$ to represent the velocity and use $\delta_i$ to represent the auxiliary input.

**Step 0**: Based on the flat output $y = [y_1, y_2]^T$, and $f(x), g(x)$ we can calculate the control **control characteristic indices** for the system. Since $L_g h_1 = [L_{g_1} h_1, L_{g_2} h_1] = [\cos(\theta), 0]$

and $L_g h_2 = [L_{g_1} h_2, L_{g_2} h_2] = [\sin(\theta), 0]$, based on the definition in section 4.1.3, the control characteristic indices $r^{\langle 0 \rangle} = [r_1^{\langle 0 \rangle}, r_2^{\langle 0 \rangle}] = [1, 1]$ and the decoupling matrix is

$$D^{\langle 0 \rangle} = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \end{bmatrix} \tag{5.8}$$

and $rank(D^{\langle 0 \rangle}) = 1$. Hence, $y^{\langle 0 \rangle} = [\dot{y_1}, \dot{y_2}] = [v\cos\theta, v\sin\theta]$ can be decomposed as $\tilde{y}^{\langle 0 \rangle} = \dot{y_1} = v\cos\theta$ and $\hat{y}^{\langle 0 \rangle} = \dot{y_2} = v\sin\theta$. Then we introduce auxiliary input $\delta_1$

$$\delta_1 = \tilde{y}^{\langle 0 \rangle} = \tilde{a}_0(x) + \tilde{b}_0(x)u = v\cos\theta \tag{5.9}$$

so that $\tilde{a}_0(x) = 0$ and $\tilde{b}_0(x) = [\cos\theta, 0]$. Then $\hat{y}^{\langle 0 \rangle}$ can be written as a function of $(x, \delta_1)$

$$\hat{y}^{\langle 0 \rangle} = v\sin\theta = \frac{\delta_1 \sin\theta}{\cos\theta} \tag{5.10}$$

Based on the equation (4.7). we take the time derivative of $\hat{y}^{\langle 0 \rangle}$ and obtain the $b_1$ term in $\dot{\hat{y}}^{\langle 0 \rangle}$:

$$b_1(x, \delta_1) = \begin{bmatrix} \frac{\delta_1 \tan(\phi)(\tan\theta + 1)}{l} & 0 \end{bmatrix} \tag{5.11}$$

based on $\dot{\hat{y}}^{\langle 0 \rangle}$ , we notice the $b_1(x, \delta_1)$ vector are linear dependent on $\tilde{b}_0(x)$, So we need to compute $\ddot{\hat{y}}^{\langle 0 \rangle}$ and check the $b_1$ terms again:

$$\tilde{b}_1(x, \delta_1) = \begin{bmatrix} \frac{3\tan^2(\phi)\tan(\theta)\delta_1^2}{\cos(\theta)^3 l^2} & \frac{\delta_1^2}{l\cos(\phi)^2\cos(\theta)^3} \end{bmatrix} \tag{5.12}$$

In this time the $b_1(x, \delta_1)$ is not dependent on $\tilde{b}_0(x)$, so $r^{\langle 1 \rangle} = [2]$, which mean the $\hat{y}^{\langle 0 \rangle}$ need to do the second derivative to achieve an $b_1$, which is independent on $\tilde{b}_0(x)$. With the $r^{\langle 1 \rangle} = [2]$, we can do the next iteration.

**Step 1**: Since $y^{\langle 1 \rangle} = (\hat{y}^{\langle 0 \rangle})^{(r^{\langle 1 \rangle})}$ and $r^{\langle 1 \rangle} = [2]$, we need to compute $\ddot{\hat{y}}^{\langle 0 \rangle} = \ddot{\dot{y}}_2(x, \delta_1, \dot{\delta}_1, \ddot{\delta}_1)$ and decompose it to independent rows $\tilde{y}^{\langle 1 \rangle}$ and dependent rows $\hat{y}^{\langle 1 \rangle}$ and assign the auxiliary input $\delta_2$ to $\tilde{y}^{\langle 1 \rangle}$. Given this example, there is only one row left, so we need to assign

$\delta_2 = \dddot{y}_2(x, \delta_1, \dot{\delta}_1, \ddot{\delta}_1)$, which has the following form:

$$\begin{aligned}\delta_2 = \tilde{a}_1(x, \delta_1, \dot{\delta}_1, \ddot{\delta}_1) + \tilde{b}_1(x, \delta_1)u = \tilde{y}^{\langle 1\rangle} = \dddot{y}_2(x, \delta_1, \dot{\delta}_1, \ddot{\delta}_1) = \\ \dot{\delta}_1\left(\frac{2\tan(\phi)\tan^2(\theta)\delta_1}{l\cos(\theta)}\right) + \ddot{\delta}_1\tan(\theta) + \frac{\delta_1^2\gamma_\phi}{l\cos(\phi)^2\cos(\theta)^3} \\ \frac{\sin(\phi)\left(\dot{\delta}_1 l\cos^2(\theta)\cos(\phi) + 3\sin(\phi)\sin(\theta)\delta_1^2\right)v}{l^2\cos(\phi)^2\cos(\theta)^4}\end{aligned} \tag{5.13}$$

and the decoupling matrix is

$$D^{\langle 1\rangle} = \begin{bmatrix} \frac{\sin(\phi)\left(\dot{\delta}_1 l\left(\cos^2(\theta)\right)\cos(\phi)+3\sin(\phi)\sin(\theta)\delta_1^2\right)}{l^2\cos(\phi)^2\cos(\theta)^4} & \frac{\delta_1^2}{l\cos(\phi)^2\cos(\theta)^3} \end{bmatrix} \tag{5.14}$$

which is linear independent of row in $D^{\langle 0\rangle}$. Now we get an invertible relation between $u = [v, \gamma_\phi]$ and $\delta = [\delta_1, \delta_2]^T$ based on (5.9) and (5.13):

$$\delta = \tilde{a} + D^\circ u = \begin{bmatrix} \tilde{a}_0(x) \\ \tilde{a}_1(x, \delta_1, \dot{\delta}_1, \ddot{\delta}_1) \end{bmatrix} + D^\circ u \tag{5.15}$$

where

$$D^\circ = \begin{bmatrix} \tilde{b}_0(x) \\ \tilde{b}_1(x, \delta_1) \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 \\ \frac{\sin(\phi)\left(\dot{\delta}_1 l\left(\cos^2(\theta)\right)\cos(\phi)+3\sin(\phi)\sin(\theta)\delta_1^2\right)}{l^2\cos(\phi)^2\cos(\theta)^4} & \frac{\delta_1^2}{l\cos(\phi)^2\cos(\theta)^3} \end{bmatrix} \tag{5.16}$$

And the $u = [v, \gamma_\phi]$ has the following form:

$$\begin{aligned}\gamma_\phi = \frac{1}{l\,\delta_1^2\cos(\theta)^2}\Big(\delta_2 l^2\cos^2(\phi)\cos^5(\theta) - \ddot{\delta}_1\sin(\theta)l^2\cos^2(\phi)\cos^4(\theta) \\ - 3\cos^2(\theta)\sin(\phi)\cos(\phi)l\delta_1\dot{\delta}_1 + 3\cos^2(\phi)\sin(\theta)\delta_1^3 - 3\delta_1^3\sin(\theta)\Big)\end{aligned} \tag{5.17}$$

$$v = \frac{\delta_1}{\cos(\theta)} \tag{5.18}$$

**Coordinate Transformation**: The coordinate transformation $T(x)$ has the following form:

$$z = T(x) = \begin{bmatrix} y_1(x) \\ y_2(x) \\ \dot{y}_2(x, \delta_1) \\ \ddot{y}_2(x, \delta_1, \dot{\delta}_1) \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \delta_1\tan(\theta) \\ \dot{\delta}_1\tan(\theta) + \frac{\tan(\phi)\delta_1^2(\tan^2(\theta)+1)}{l\cos(\theta)} \end{bmatrix} \tag{5.19}$$

And the linearized system has the following dynamics:

$$\dot{z} = \begin{bmatrix} \dot{y_1} \\ \dot{y}_2 \\ \ddot{y}_2 \\ \dddot{y}_2 \end{bmatrix} = \begin{bmatrix} \delta_1 \\ z_3 \\ z_4 \\ \delta_2 \end{bmatrix} = A_c z + B_c \delta = A_c z + B_c(\tilde{a} + D^{\circ} u) \tag{5.20}$$

$$A_c = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad B_c = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$$

It is easy to draw the conclusion that the linearized system are controllable.

## 5.3 Kinematic Car Output Tracking

As we said at 2.2.2, The advantage of the QSF is that if the feedback is used for the stable tracking of a known reference trajectory, the time derivatives of auxiliary input $(\delta_1, \delta_2)$ are expressed via derivatives of the reference trajectory $y_d$ and, therefore no numerical differentiation is required, which can save computing resources. So next we will show how to achieve trajectory tracking based on quasi-static feedback [12].

First, we need to define the tracking error:

$$e = [e_1, e_2]^T = [y_1 - y_{d1}, y_2 - y_{d2}, \dot{y}_2 - \dot{y}_{d2}, \ddot{y}_2 - \ddot{y}_{d2}]^T \tag{5.21}$$

where $y_{di}, 1 \leq i \leq 2$ are desired outputs, $e_1 = [y_1 - y_{d1}], e_2 = [y_2 - y_{d2}, \dot{y}_2 - \dot{y}_{d2}, \ddot{y}_2 - \ddot{y}_{d2}]$. Then we substitute the feedback $[\delta_1, \delta_2]^T = [\dot{y}_{d1}, \dddot{y}_{d2}]^T + Ke$, in (5.15) and get the invert relation, i.e., the controller for output tracking:

$$u = D^{\circ -1}(Ke - \tilde{a} + \begin{bmatrix} \dot{y}_{d1} \\ \dddot{y}_{d2} \end{bmatrix}) \tag{5.22}$$

$$K \in \mathbb{R}^{m \times \Sigma_i r_i}, \quad i = 1, \cdots, m$$

where $K$ is a block matrix; $m$ is the output number, in this example $m = 2$, $r_1 = r^{\langle 0 \rangle} = 1, r_2 = r^{\langle 0 \rangle} + r^{\langle 1 \rangle} = 3$ and $K \in \mathbb{R}^{2\times 4}$ and . And each blocks $K_i \in \mathbb{R}^{1\times r_i}$ chosen so that the linear controllable differential equation

$$e_i^{(r_i)} = K_i e_i \tag{5.23}$$

is asymptotically stable. And based on (5.23) we can calculate the corresponding $\delta_i, \dot{\delta}_i, \cdots, \delta_i^{(k)}$. In this case, we can use $\dot{e_1} = K_1 e_1$ to calculate $\delta_1$, $\dot{\delta}_1$ and $\ddot{\delta}_1$

$$\delta_1 = \dot{y}_{d1} + K_1(y_1 - y_{d1}) \tag{5.24}$$

$$\dot{\delta}_1 = \ddot{y}_{d1} + K_1(\dot{y}_1 - \dot{y}_{d1}) = \ddot{y}_{d1} + K_1(\delta_1 - \dot{y}_{d1}) \tag{5.25}$$

$$\ddot{\delta}_1 = \dddot{y}_{d1} + K_1(\ddot{y}_1 - \ddot{y}_{d1}) = \dddot{y}_{d1} + K_1(\dot{\delta}_1 - \ddot{y}_{d1}) \tag{5.26}$$

$$\tag{5.27}$$

### 5.3.1 Matlab Simulation

We simulate the trajectory tracking of the Kinematic car with the QSF controller in Matlab. The desired trajectory is $y_{d1} = \sin(\frac{t}{300}), y_{d2} = \cos(\frac{t}{300}), t \in [0, 400]$ and the initical conition is $[0.5, 0.5, 0, 0]$. The tracking error of $e_{y_1}$ and $e_{y_2}$ were shown in Figure 5.2. The states $[y_1, y_2, \theta, \phi]$ was shown in Figure 5.3. The inputs $v$ and $\gamma_\phi$ were shown in Figure 5.4.

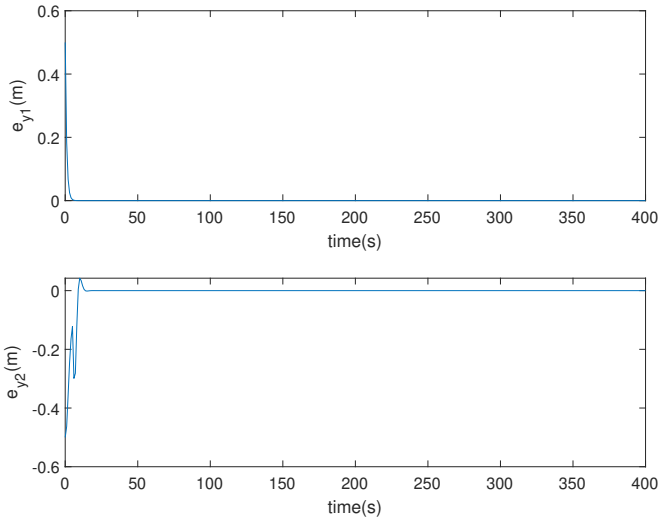



Figure 5.2: Kinematic car output tracking simulation: tracking error $e_i = y_i - y_d i$.

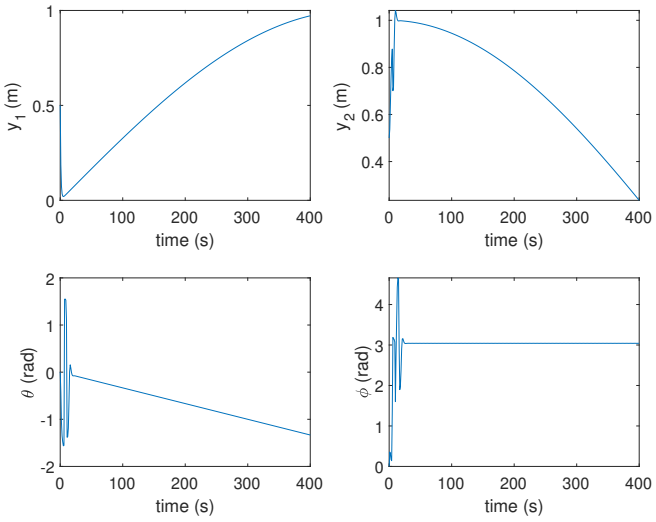



Figure 5.3: Kinematic car output tracking simulation: State of the system $y_1, y_2, \theta, \phi$

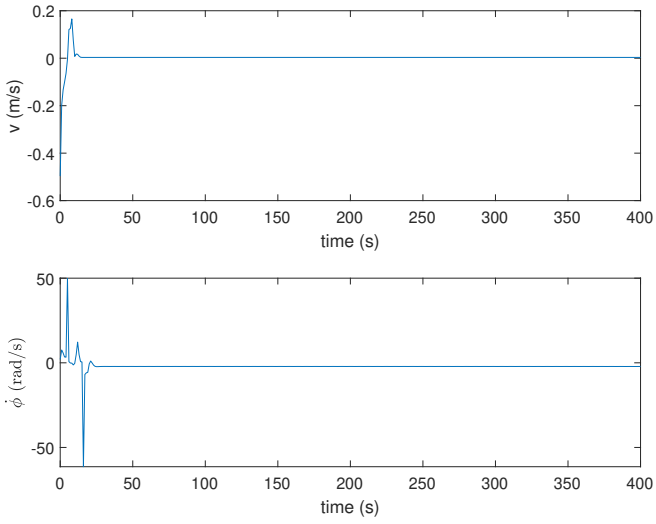



Figure 5.4: Kinematic car output tracking simulation: Input of the system $v, \gamma_\phi$

# Chapter 6

# Controller Design for SLSs

## 6.1 SLS Controller Design Using QSFA

In this section, the QSFA is applied to the SLS model (3.16) and flat output

$$y = [p_L^T, \psi]^T \tag{6.1}$$

As expected, the QSFA yields a QSF which exactly linearizes the SLS dynamics. **Step** 0**.** According to (4.2), we have

$$D^{\langle 0\rangle} = \begin{bmatrix} d_{11}^{\langle 0\rangle} & 0 & 0 & 0 \\ d_{21}^{\langle 0\rangle} & 0 & 0 & 0 \\ d_{31}^{\langle 0\rangle} & 0 & 0 & 0 \\ 0 & 0 & \frac{s_\phi}{J_2 c_\theta} & \frac{c_\phi}{J_3 c_\theta} \end{bmatrix} \tag{6.2}$$

where $d_{11}^{\langle 0\rangle}, d_{21}^{\langle 0\rangle}, d_{31}^{\langle 0\rangle}$ are functions of state and we have $r^{\langle 0\rangle} = [2,2,2,2]^T$ and $rank\ (D^{\langle 0\rangle}) = 2$ on a subset of $\mathscr{M}$ where

$$d_{31}^{\langle 0\rangle}(x) = -\frac{[s_\beta, -s_\alpha c_\beta, c_\alpha c_\beta] \cdot Rn_3}{m_Q + m_L + m_C} \neq 0 \tag{6.3}$$

As in (4.3), we obtain

$$y^{\langle 0\rangle} = a_0(x) + D^{\langle 0\rangle} u \tag{6.4}$$

Hence, $y^{\langle 0\rangle}$ is decomposed as $\tilde{y}^{\langle 0\rangle} = [\ddot{y}_3, \ddot{y}_4]^T$, and $\hat{y}^{\langle 0\rangle} = [\ddot{y}_1, \ddot{y}_2]^T$. We introduce auxiliary input $v_1 = [\ddot{y}_3, \ddot{y}_4]^T$ so that

$$v_1 = \tilde{y}^{\langle 0\rangle} = \tilde{a}_0(x) + \tilde{b}_0(x) u \tag{6.5}$$

Then, $\hat{y}^{\langle 0\rangle}$ can be written as

$$\hat{y}^{\langle 0\rangle} = \begin{bmatrix} -\left(g - [1,0]^T v_1\right) t_\beta / c_\alpha \\ \left(g - [1,0]^T v_1\right) t_\alpha \end{bmatrix} \tag{6.6}$$

**Step** 1. Taking a time derivative of $\hat{y}^{\langle 0\rangle}$ we obtain

$$\dot{\hat{y}}^{\langle 0\rangle} = \begin{bmatrix} \frac{\dot{v}_1 s_\beta}{c_\beta c_\alpha^2} - \frac{\gamma_\beta (g - v_1)}{c_\beta^2 c_\alpha} - \frac{\gamma_\alpha s_\alpha s_\beta (g - v_1)}{c_\beta c_\alpha} \\ -\dot{v}_1 t_\alpha + \frac{\gamma_\alpha (g - v_1)}{c_\alpha^2} \end{bmatrix} \tag{6.7}$$

Since the input does not appear in (6.7), we take another time derivative of $\hat{y}^{\langle 0\rangle}$ to get

$$\ddot{\hat{y}}^{\langle 0\rangle} = a_1(x, v_1, \dot{v}_1, \ddot{v}_1) + b_1(x, v_1, \dot{v}_1) u \tag{6.8}$$

where $a_1(x, v_1, \dot{v}_1, \ddot{v}_1) \in \mathbb{R}^{2\times 1}$, $b_1(x, v_1, \dot{v}_1) \in \mathbb{R}^{2\times 4}$ are functions of $x$, auxiliary input $v$, and its time derivative. Matrix $b_1$ has the form

$$b_1(x, v_1, \dot{v}_1) = \begin{bmatrix} b_{11} & 0 & 0 & 0 \\ b_{21} & 0 & 0 & 0 \end{bmatrix} \tag{6.9}$$

We observe that both rows of (6.9) are linearly dependent on the third row of $D^{\langle 0\rangle}$. Hence, $\ddot{\hat{y}}^{\langle 0\rangle}$ can be expressed using $v_1$. As a result, (6.8) can be written as $\ddot{\hat{y}}^{\langle 0\rangle} = \ddot{\hat{y}}^{\langle 0\rangle}(x, v_1, \dot{v}_1, \ddot{v}_1)$ with $u$ eliminated.

Similarly we eliminate $u$ in $\left(\hat{y}^{\langle 0\rangle}\right)^{(3)}$ to obtain a function $\left(\hat{y}^{\langle 0\rangle}\right)^{(3)}(x, v_1, \dot{v}_1, \ddot{v}_1, v_1^{(3)})$. Calculating $\left(\hat{y}^{\langle 0\rangle}\right)^{(4)}$ gives the decoupling matrix $D^{\langle 1\rangle}$

$$D^{\langle 1\rangle} = \begin{bmatrix} d_{11}^{\langle 1\rangle} & d_{12}^{\langle 1\rangle} & d_{13}^{\langle 1\rangle} & 0 \\ d_{21}^{\langle 1\rangle} & d_{22}^{\langle 1\rangle} & d_{23}^{\langle 1\rangle} & 0 \end{bmatrix} \tag{6.10}$$

where $d_{ij}^{\langle 1\rangle}$ are functions of $(x, v_1, \dot{v}_1, \ldots, v_1^{(3)})$ with $rank\ (D^{\langle 1\rangle}) = 2$ and its rows are linear independent of all rows of $D^{\langle 0\rangle}$. Hence, $r^{\langle 1\rangle} = [4,4]$. Defining $v_2 = \left(\hat{y}^{\langle 0\rangle}\right)^{(r^{\langle 1\rangle})} = y^{\langle 1\rangle}$, we have

$$v_2 = y^{\langle 1\rangle} = \tilde{a}_1(x, \dot{v}_1, \ldots, v_1^{(4)}) + D^{\langle 1\rangle} u \tag{6.11}$$

Combining (6.5), (6.11), we have a invertible relation between $u$ and $v$

$$v = \begin{bmatrix} \tilde{a}_0(x) \\ \tilde{a}_1(x, v_1, \dot{v}_1, \ldots, v_1^{(4)}) \end{bmatrix} + D^\circ u \tag{6.12}$$

where

$$D^\circ = \begin{bmatrix} d_{31}^{\langle 0 \rangle} & 0 & 0 & 0 \\ 0 & 0 & \frac{s_\phi}{J_2 c_\theta} & \frac{c_\phi}{J_3 c_\theta} \\ d_{11}^{\langle 1 \rangle} & d_{12}^{\langle 1 \rangle} & d_{13}^{\langle 1 \rangle} & 0 \\ d_{21}^{\langle 1 \rangle} & d_{22}^{\langle 1 \rangle} & d_{23}^{\langle 1 \rangle} & 0 \end{bmatrix} \tag{6.13}$$

is the final decoupling matrix with $rank\ (D^\circ) = 4$.

## 6.2 SLS Output Tracking

By setting $v = [v_1, v_2]^T = [y_3^{(2)}, y_4^{(2)}, y_1^{(6)}, y_2^{(6)}]^T$ and using (6.12), a linearizing QSF is obtained. The tracking error is defined as

$$\begin{aligned} \tilde{z} = [\tilde{z}_1, \ldots, \tilde{z}_{16}] = [y_1 - y_{d1}, \ldots, y_1^{(5)} - y_{d1}^{(5)}, y_2 - y_{d2}, \ldots, y_2^{(5)} - y_{d2}^{(5)}, \\ y_3 - y_{d3}, \dot{y}_3 - \dot{y}_{d3}, y_4 - y_{d4}, \dot{y}_4 - \dot{y}_{d4}]^T \end{aligned} \tag{6.14}$$

where $y_{di}, 1 \leq i \leq 4$ are desired outputs. Variables $y_1^{(3)}, \ldots, y_1^{(5)}$, and $y_2^{(3)}, \ldots, y_2^{(5)}$ are functions of $x, v_1, \dot{v}_1, \ldots, v_1^{(4)}$, while the remaining outputs and their derivatives in (6.14) can be expressed as a function of $x$.

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

$$\dot{\tilde{z}} = A_c \tilde{z} + B_c(\tilde{a} + D^\circ u) \tag{6.15}$$

where

$$\begin{aligned} A_c &= \text{blockdiag}(A_1, A_2, A_3, A_4) \\ B_c &= \begin{bmatrix} e_6 & e_{12} & e_{14} & e_{16} \end{bmatrix} \\ \tilde{a} &= [\tilde{a}_0(x), \tilde{a}_1(x, v_1, \dot{v}_1, \ldots, v_1^{(4)})]^T \end{aligned}$$

$e_i \in \mathbb{R}^{16}$ denotes the ith unit vector, and

$$A_1 = A_2 = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \text{ and } A_j = \begin{bmatrix} 0_{5\times 1} & I_5 \\ 0 & 0_{1\times 5} \end{bmatrix} \in \mathbb{R}^{6\times 6}$$

with $j = 3,4$.

Applying the linearizing control $u = D^{\circ -1}(K\tilde{z} - \tilde{a} + y_d^{(\bar{r})})$ with $y_d^{(\bar{r})} = [y_{d1}^{(6)}, y_{d2}^{(6)}, y_{d3}^{(2)}, y_{d4}^{(2)}]^T$ to (6.15) we obtain

$$\dot{\tilde{z}} = (A_c + B_c K)\tilde{z} \tag{6.16}$$

where $K \in \mathbb{R}^{4\times 16}$ is a control gain chosen so that $A_c + B_c K$ is Hurwitz and transient error tracking is satisfactory. Because $\dot{v}_1, \ldots, v_1^{(4)}$ are calculated using $y_3 - y_{d3}, \dot{y}_3 - \dot{y}_{d3}, y_4 - y_{d4}, \dot{y}_4 - \dot{y}_{d4}$ and their time derivatives, the controller depends only on $x$ and the reference trajectory. Hence, it is a static state feedback. The expressions for $v_1, \dot{v}_1, \ddot{v}_1$ are

$$v_1 = [\ddot{y}_{d3}, \ddot{y}_{d4}]^T - k_1[\tilde{z}_{13}, \tilde{z}_{15}]^T - k_2[\tilde{z}_{14}, \tilde{z}_{16}]^T \tag{6.17a}$$

$$\dot{v}_1 = [y_{d3}^{(3)}, y_{d4}^{(3)}]^T - k_1[\tilde{z}_{14}, \tilde{z}_{16}]^T - k_2(v_1 - [\ddot{y}_{d3}, \ddot{y}_{d4}]^T) \tag{6.17b}$$

$$\ddot{v}_1 = [y_{d3}^{(4)}, y_{d4}^{(4)}]^T - k_1(v_1 - [\ddot{y}_{d3}, \ddot{y}_{d4}]^T) - k_2(\dot{v}_1 - [y_{d3}^{(3)}, y_{d4}^{(3)}]^T) \tag{6.17c}$$

where $k_1, k_2$ are control gains from $K$. Similar expressions can be obtained for $v_1^{(3)}, v_1^{(4)}$.

## 6.3 QSF Domain

In this section, we describe the domain where the QSF is well-defined. The control is singular at $\theta = \pm 90^\circ$ and $\beta = \pm 90^\circ$ as these singularities occur in (3.16) due to Euler angles. At points where the Jacobian matrix of the $\tilde{z}$-coordinates is singular, the QSF cannot be evaluated. These points can be obtained from the singularities of $D^\circ$ in (6.13).

We obtain

$$\phi = \pm 90^{\circ} \tag{6.18a}$$

$$[s_\beta, -s_\alpha c_\beta, c_\alpha c_\beta] \cdot R n_3 = 0 \tag{6.18b}$$

$$\ddot{p}_3 = g - \frac{c_\alpha c_\beta (\gamma_\alpha^2 c_\beta^2 + \gamma_\beta^2) L m}{m_Q + m_L} \tag{6.18c}$$

$$\ddot{p}_3 = g \tag{6.18d}$$

The LHS of (6.18b) (which from (6.3) is $d_{31}$ scaled) can be geometrically interpreted as the inner product of $q$ and the direction of thrust. Thus, when the pendulum is perpendicular to the thrust, a physical singularity occurs. Condition (6.18c) corresponds to $\bar{u} = 0$. When the pendulum's downward linear acceleration $\ddot{p}_3 = g$, we obtain (6.18d). This is another physical singularity that appears in drone motion control. Based on the above, we can conclude that the controller's domain is a practical subset of $\mathscr{M}$ since the above points are atypical of SLS operation.

# Chapter 7

# SITL Simulation

This section presents Software-In-The-Loop (SITL) simulation of the QSF. SITL emulates an autopilot environment so that performance can be exhaustively verified in a safe and realistic setting. For instance, it ensures the design is robust to unmodeled effects such as controller saturation, multi-rate sampling, and computational delay. Such real-world effects can degrade the performance of the design or make the theory impossible to implement (*e.g.*there may not be enough processing power or memory). We choose the open source PX4 SITL framework [5] with the Gazebo simulator using the open dynamics engine (ODE) as the physics engine [6]. Gazebo was chosen for the simulator because of its simple multi-body model format. Besides, it is open-source and currently recommended by the PX4 developers for SITL. Unlike the Matlab simulation where the SLS model directly uses the differential equations (DEs), a Gazebo model requires no DE model but rather is based on the geometry and inertial properties of the system's bodies. The PX4-Gazebo SITL simulation leverages the RotorS simulation plugins [20].

The structure of the simulation environment is shown in Figure 7.1. There are three components: the Gazebo simulator, the PX4 autopilot code, and the ROS Offboard Control. Gazebo contains a multibody dynamics engine, a 3D graphics interface, a 3DR Iris quadrotor-based SLS model description file, external disturbances, and plugins adopted from RotorS that simulate onboard sensors, *e.g.*Global Positioning System (GPS), Inertial Measurement Unit (IMU), and magnetometer. The PX4 autopilot software is emulated to

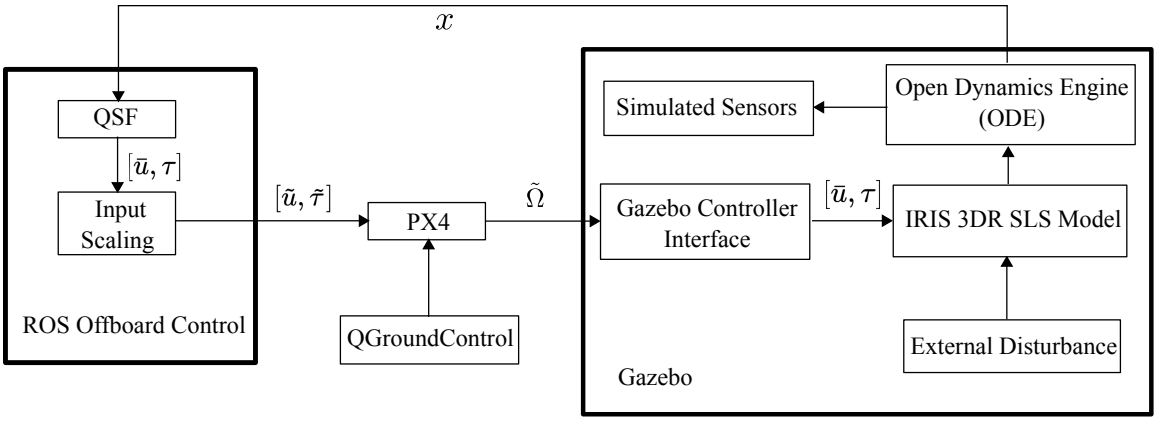



Figure 7.1: PX4-SITL System.

run on a desktop. The ROS Offboard Control contains our QSF and coordinate transformations that transform the Gazebo world frame to the navigation frame $\mathcal{N}$ used in the QSF.

The video of the stabilization and time-varying tracking discussed in this subsection is at https://youtu.be/fcOKLIxspCw.

## 7.1 Parameters Setting

In this section, we give the parameters of rotors and SLS used in the SITL simulation

### Rotor Parameters

The geometry parameters of the 3DR Iris quadrotor mentioned at 3.3 are $\ell = [0.22, 0.13, 0.2, 0.13]$ m. And the rotor thrust constant in the mixer is $C_T = 5 \times 10^{-6}\text{Ns}^2$; and torque constant is $C_M = 0.05\text{m}$. So the equ (3.17) can be written as

$$\begin{bmatrix} \bar{u} \\ \tau \end{bmatrix} = 5 \times 10^{-6} \begin{bmatrix} 1 & 1 & 1 & 1 \\ -0.22 & 0.2 & 0.22 & -0.2 \\ 0.13 & -0.13 & 0.13 & -0.13 \\ 0.05 & 0.05 & -0.05 & -0.05 \end{bmatrix} \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix} \tag{7.1}$$

The PX4 mapping from normalized thrust $\tilde{u}$ and torque $\tilde{\tau}$ to normalized rotor speed

command $\tilde{\Omega}_i, 1 \leq i \leq 4$ is:

$$\begin{bmatrix} \tilde{\Omega}_1^2 \\ \tilde{\Omega}_2^2 \\ \tilde{\Omega}_3^2 \\ \tilde{\Omega}_4^2 \end{bmatrix} = \begin{bmatrix} 1 & -0.4377 & 0.7071 & 0.9091 \\ 1 & 0.4377 & -0.7071 & 1 \\ 1 & 0.4377 & 0.7071 & -0.9091 \\ 1 & -0.4377 & -0.7071 & -1 \end{bmatrix} \cdot \begin{bmatrix} \tilde{u} \\ \tilde{\tau} \end{bmatrix}$$

Since Gazebo will receive the normalized rotor speed from PX4 but need the classical rotor speed to simulate the quadrotor, we introduced the scaling constant in (3.18) and $C_\Omega = 1000$ rad/s. Finally, the scaling for the output from the QSF controller used in equ (3.19) is $S_1 = 20, S_2 = 1.8383, S_3 = 1.8383, S_4 = 0.9546$, which can compensate the effect introduced by normalized thrust and torque used in PX4.

### Slung Load System Parameters

The parameters of SLS has been shown below:

Table 7.1: System parameters.

| | |
|---|---|
| $m$ | 1.6 kg |
| $m_p$ | 0.16 kg |
| $m_c$ | 0.05 kg |
| $L$ | 1 m |
| $J_1$ | 0.03 kg $\times$ m$^2$ |
| $J_2$ | 0.03 kg $\times$ m$^2$ |
| $J_3$ | 0.05 kg $\times$ m$^2$ |

## 7.2 Stabilization

For QSF stabilization, we test the controller without disturbances. The set point for the QSF is $y_d = [0, 0, -10, 0]$. The quadrotor takes off with the PX4 built-in motion controller and is commanded to a position away from the set point. The QSF is activated once the SLS is at rest. The configuration variables are in Figure. 7.2 with the QSF activated at

$t = 138s$. Good stabilization performance is achieved with a well-damped transition to the setpoint in about $4s$. Corresponding input trajectories are given in Figure. 7.3 which are physically realizable and unsaturated.

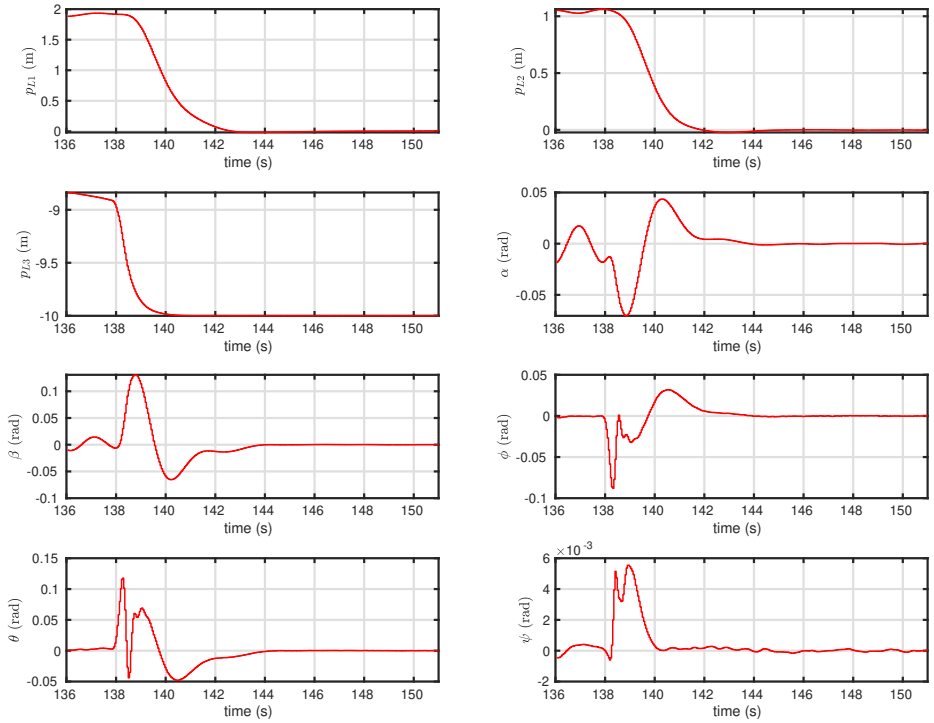



Figure 7.2: Stabilization SITL simulation: system states $p, \alpha, \beta, \eta$.

## 7.3 Trajectory Tracking

The proposed QSF is capable of tracking complex reference trajectories. We consider the 'figure-8' reference

$$y_d(t) = \begin{bmatrix} 3\sin(\pi t/4) \text{ m} \\ 1.5\sin(\pi t/2) \text{ m} \\ 0.5\sin(\pi t/4) - 10 \text{ m} \\ 0.02t \text{ rad} \end{bmatrix} \tag{7.2}$$

One of the advantages of the QSF is its guaranteed tracking performance for any bounded smooth trajectory for the flat output. As shown in Figure. 7.4 and Figure. 7.5, the tracking

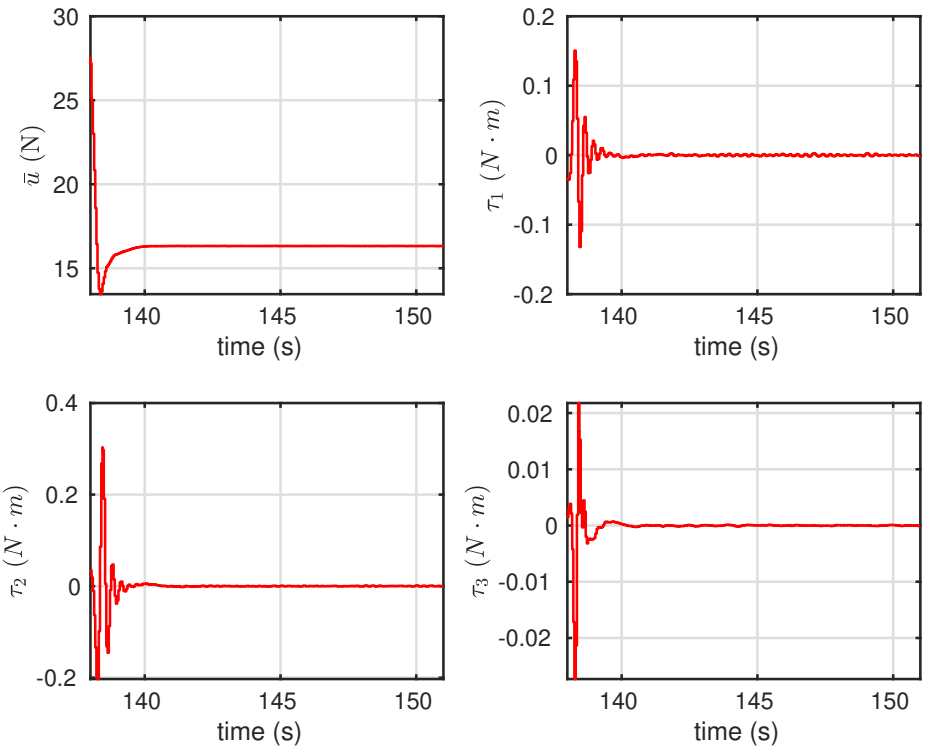



Figure 7.3: Stabilization SITL simulation: inputs $\bar{u}, \tau$.

error converges to zero exponentially with a good transient performance. The control input remains unsaturated as shown in Figure. 7.6. No wind disturbance is applied.

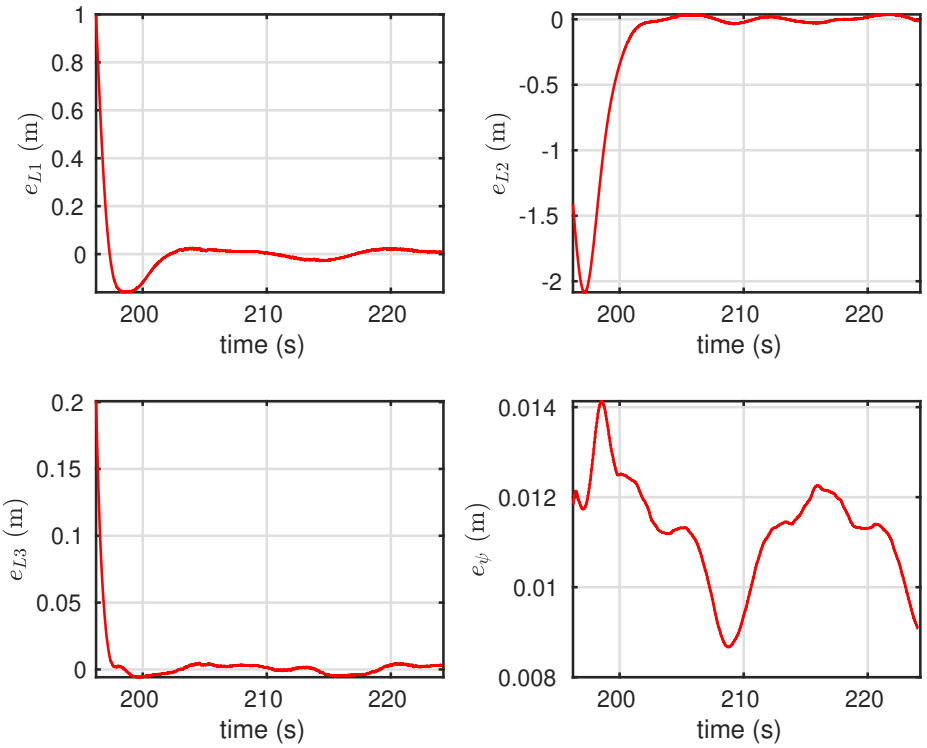



Figure 7.4: Time-varying output tracking SITL simulation: output tracking error $y - y_d$.

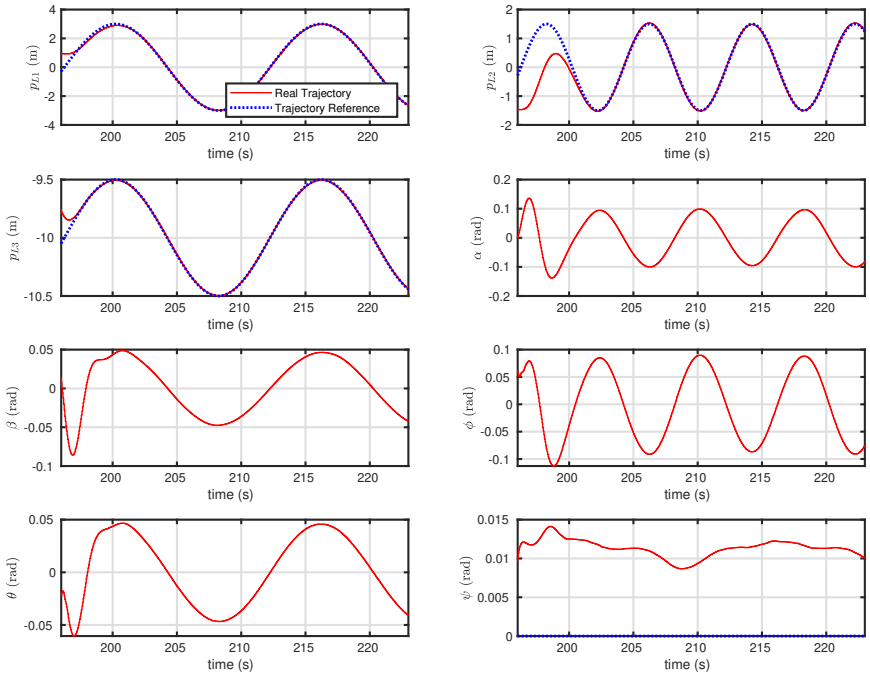



Figure 7.5: Time-varying output tracking SITL simulation: system states $p, \alpha, \beta, \eta$.

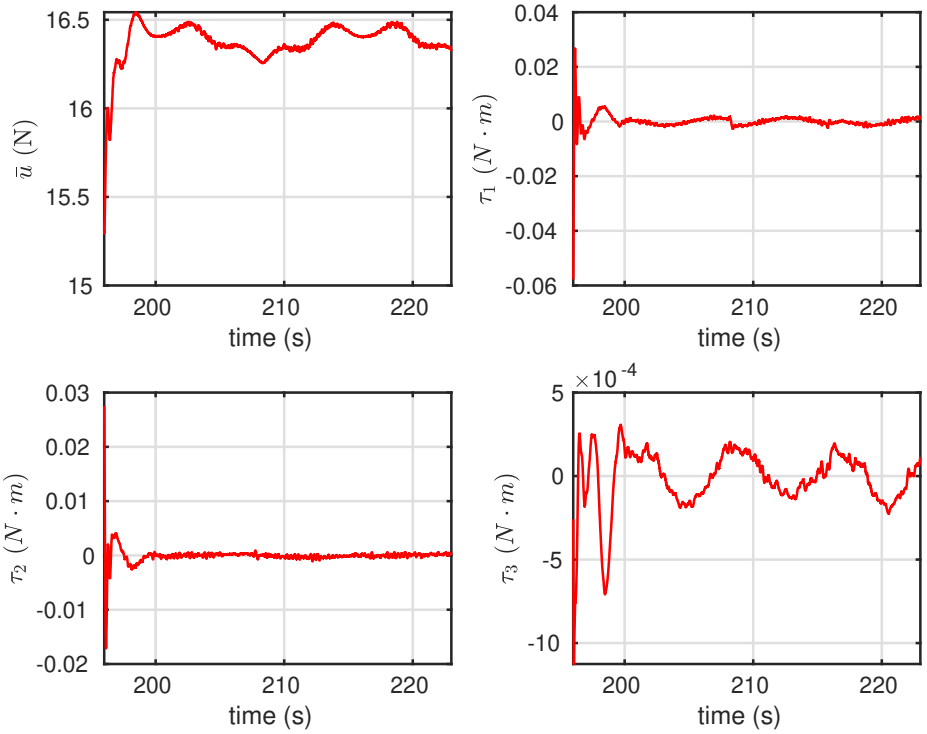



Figure 7.6: Time-varying output tracking SITL simulation: inputs $\bar{u}, \tau$.

# Chapter 8

# Conclusion

A novel algorithm for designing a QSF for a general flat system is presented in this project. The QSF achieves LTI exponentially stable tracking error dynamics on a well-defined and practical region of state space. Although QSF is investigated in [21–23], the design procedure is only demonstrated by specific examples. This quasi-static feedback algorithm is the first to provide a systematic algorithm for QSF. The procedure for applying the QSFA is described in detail for the SLS.

In order to demonstrate the proposed QSF can be implemented on a real autopilot, we present a software pipeline for implementing a PX4/Gazebo SITL simulation. SITL simulation is a critical step in developing flyable controllers as it proves the design is robust to unmodelled effects. The developed SITL framework code is available publicly and can be used for further development of advanced model-based control. Future work includes compensation of rotor drag and external forces due to wind disturbance.

# Bibliography


[1] H. M. Omar, R. Akram, S. M. Mukras, and A. A. Mahvouz, "Recent advances and challenges in controlling quadrotors with suspended loads," *Alexandria Engineering Journal*, 2022.

[2] D. K. Villa, A. S. Brandao, and M. Sarcinelli-Filho, "A survey on load transportation using multirotor uavs," *Journal of Intelligent & Robotic Systems*, vol. 98, no. 2, pp. 267–296, 2020.

[3] *What is software-in-the-loop testing?* [Online]. Available: https://www.aptiv.com/en/insights/article/what-is-software-in-the-loop-testing.

[4] *Open source autopilot for drones*, 2021. [Online]. Available: https://px4.io/.

[5] L. Meier, D. Honegger, and M. Pollefeys, "Px4: A node-based multithreaded open source robotics framework for deeply embedded platforms," in *2015 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2015, pp. 6235–6240.

[6] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, IEEE, vol. 3, 2004, pp. 2149–2154.

[7] M. Quigley *et al.*, "Ros: An open-source robot operating system," in *ICRA workshop on open source software*, Kobe, Japan, vol. 3, 2009, p. 5.

[8] M. Fliess, J. Lévine, P. Martin, and P. Rouchon, "Flatness and defect of non-linear systems: Introductory theory and examples," *International journal of control*, vol. 61, no. 6, pp. 1327–1361, 1995.

[9] P. Martin, S. Devasia, and B. Paden, "A different look at output tracking: Control of a vtol aircraft," *Automatica*, vol. 32, no. 1, pp. 101–107, 1996.

[10] R. Rothfuss, J. Rudolph, and M. Zeitz, "Flatness based control of a nonlinear chemical reactor model," *Automatica*, vol. 32, no. 10, pp. 1433–1439, 1996.

[11] G. Oriolo, A. De Luca, and M. Vendittelli, "Wmr control via dynamic feedback linearization: Design, implementation, and experimental validation," *IEEE Transactions on control systems technology*, vol. 10, no. 6, pp. 835–852, 2002.

[12] E. Delaleau and J. Rudolph, "Control of flat systems by quasi-static feedback of generalized states," *International Journal of Control*, vol. 71, no. 5, pp. 745–765, 1998.

[13] E Delaleau and M Fliess, "An algebraic interpretation of the structure algorithm with an application to feedback decoupling," in *Nonlinear Control Systems Design 1992*, Elsevier, 1993, pp. 179–184.

[14] Z. Jiang, M. Al Lawati, A. Mohammadhasani, and A. F. Lynch, "Flatness-based motion control of a uav slung load system using quasi-static feedback linearization," in *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, 2022, pp. 361–368.

[15] A. Mohammadhasani, "Intelligent control of a quadrotor with suspended load," 2022.

[16] H. Xie, "Dynamic visual servoing of rotary wing unmanned aerial vehicles," 2016.

[17] K. M. Lynch and F. C. Park, *Modern robotics*. Cambridge University Press, 2017.

[18] R. Marino and P. Tomei, *Nonlinear Control Design: Geometric, Adaptive, and Robust* (Prentice-Hall information and system sciences series). Prentice Hall, 1995, ISBN: 9780133426359. [Online]. Available: https://books.google.ca/books?id=HQprQgAACAAJ.

[19] J. Rudolph and S. Verlag, *Flatness-Based Control: An Introduction* (Berichte aus der Steuerungs- und Regelungstechnik). Shaker, 2021, ISBN: 9783844078930. [Online]. Available: https://books.google.ca/books?id=LnxDzgEACAAJ.

[20] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "Robot operating system (ros): The complete reference (volume 1)," in A. Koubaa, Ed. Cham: Springer International Publishing, 2016, ch. RotorS—A Modular Gazebo MAV Simulator Framework, pp. 595–625, ISBN: 978-3-319-26054-9. DOI: 10.1007/978-3-319-26054-9_23. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-26054-9_23.

[21] J. Rudolph and E. Delaleau, "Some remarks on quasi-static feedback of generalized states," in *IFAC Proceedings Volumes*, vol. 27, 1994, pp. 51–56. DOI: https://doi.org/10.1016/S1474-6670(17)47621-X.

[22] E. Delaleau and J. Rudolph, "Some examples and remarks on quasi-static feedback of generalized states," vol. 34, no. 8, pp. 993–999, Aug. 1998. DOI: 10.1016/s0005-1098(98)00047-8.

[23] O. Fritsch, P. D. Monte, M. Buhl, and B. Lohmann, "Quasi-static feedback linearization for the translational dynamics of a quadrotor helicopter," Montreal, QC, Canada, Jun. 2012. DOI: 10.1109/acc.2012.6314682.

# Appendix A: SLS Model Detail

The expression for $\bar{g}$ appearing in the input vector field $g_1$ of (3.16) is

$$\bar{g}(x) = \begin{bmatrix} (\xi_1\xi_{11}\xi_{12}+\xi_2)\xi_{10} \\ ((\xi_3+\xi_4)c_\beta\xi_{12}\xi_{10}+2m_C\xi_5)\xi_{11} \\ ((\xi_6+\xi_7)c_\alpha c_\beta\xi_{12}\xi_{10}-2m_C c_\theta c_\phi)\xi_{11} \\ \frac{(12m_L+6m_C)\xi_8\xi 11}{c_\beta L} \\ \frac{(12m_L+6m_C)\xi_9\xi_{11}}{L} \end{bmatrix}$$

where

$$\xi_1 = c_\beta\left(s_\beta\left(\left(c_\alpha c_\theta - s_\alpha s_\theta s_\psi\right)c_\phi + c_\psi s_\alpha s_\phi\right) - c_\beta\xi_2\right)$$

$$\xi_2 = c_\phi s_\theta c_\psi + s_\phi s_\psi$$

$$\xi_3 = c_\beta\left(\left(c_\alpha^2 s_\theta s_\psi + c_\alpha c_\theta s_\alpha - s_\psi s_\theta\right)c_\phi + c_\psi s_\phi s_\alpha^2\right)$$

$$\xi_4 = s_\beta s_\alpha\left(c_\phi s_\theta c_\psi + s_\phi s_\psi\right)$$

$$\xi_5 = -s_\psi s_\theta c_\phi + c_\psi s_\phi$$

$$\xi_6 = c_\beta((-s_\alpha s_\theta s_\psi + c_\alpha c_\theta)c_\phi + c_\psi s_\alpha s_\phi)$$

$$\xi_7 = s_\beta(c_\phi s_\theta c_\psi + s_\phi s_\psi)$$

$$\xi_8 = c_\psi c_\alpha s_\phi - c_\phi(s_\theta c_\alpha s_\psi + s_\alpha c_\theta)$$

$$\xi_9 = s_\beta\left((-s_\alpha s_\theta s_\psi + c_\alpha c_\theta)c_\phi + c_\psi s_\alpha s_\phi\right) + c_\beta\left(c_\phi s_\theta c_\psi + s_\phi s_\psi\right)$$

$$\xi_{10} = \frac{1}{m_Q + m_L + m_C}$$

$$\xi_{11} = \frac{1}{12m_Q m_L + 4m_Q m_L + 4m_L m_C + m_C^2}$$

$$\xi_{12} = 12m_Q m_L + 6m_Q m_L + 6m_L m_C + 3m_C^2$$

# Appendix B: Maple Code for Kinematic Car Example

```
> restart
> with(DifferentialGeometry):
  with(Tools):
  with(LinearAlgebra):
> sysf := Vector[column](4,[0,0,0,0])
```

$$sysf := \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{1}$$

```
> sysg := Matrix(4, 2, [[cos(theta),0],[sin(theta),0],[tan(phi)/l,
  0],[0,1]]);
  sysg1 := sysg[..,1]:
  sysg2 := sysg[..,2]:
```

$$sysg := \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ \frac{\tan(\phi)}{l} & 0 \\ 0 & 1 \end{bmatrix} \tag{2}$$

```
> u:=Vector[column](2,[v,dot_phi])
```

$$u := \begin{bmatrix} v \\ dot\_phi \end{bmatrix} \tag{3}$$

```
> x:=[y1,y2,theta,phi];
  DGsetup(x, M0):
  basis_vf:=[D_y1,D_y2,D_theta,D_phi]:
```

$$x := [y1, y2, \theta, \phi] \tag{4}$$

```
M0 > f:=DGzip(sysf,basis_vf);
     g__1 := DGzip(sysg1,basis_vf,"plus"):
     g__2 := DGzip(sysg2,basis_vf,"plus"):
     h__1 := y1:
     h__2 := y2:
     g := [g__1,g__2];
     h := [h__1,h__2];
```

$$f := 0\, D\_y1$$

$$g := \left[\cos(\theta)\, D\_y1 + \sin(\theta)\, D\_y2 + \frac{\tan(\phi)\, D\_\theta}{l}, D\_\phi\right]$$

$$h := [y1, y2] \tag{5}$$

# Calculate Control Charateristic Indices ( not relative degree) and Decoupling Matrix

```
M0 > Lgh1:=simplify([
     LieDerivative(g__1, h__1),
     LieDerivative(g__2, h__1)]);
     Lgh2:=simplify([
```

```
     LieDerivative(g__1, h__2),
     LieDerivative(g__2, h__2)]);
     D_iter0 := Matrix(2,2,[Lgh1,Lgh2])
```

$$Lgh1 := [\cos(\theta), 0]$$

$$Lgh2 := [\sin(\theta), 0]$$

$$D\_iter0 := \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \end{bmatrix} \quad (6)$$

result: r<0>=[1,1];

# Iteration 0:

1. Calculate time derivative of output y=[h1,h2]

```
M0 > y1d := LieDerivative(f,h__1)+ u[1]*Lgh1[1] + u[2]*Lgh2[2];
     y2d := LieDerivative(f,h__2)+ u[1]*Lgh2[1] + u[2]*Lgh2[2];
```

$$y1d := v\cos(\theta)$$

$$y2d := v\sin(\theta) \quad (7)$$

decompose the derivative of ydot to ydot_tilde(y1d in this case) and ydot_hat(y2d in this case)

```
M0 > yd_tilde_iter0 := (D_iter0.u)[1];
     yd_hat_iter0 := (D_iter0.u)[2];
```

$$yd\_tilde\_iter0 := v\cos(\theta)$$

$$yd\_hat\_iter0 := v\sin(\theta) \quad (8)$$

3. set auxillary input v__1 = yd_tilde ,transfer yd_hat to func of v__1 and x

```
M0 > v__1 = yd_tilde_iter0;
     yd_hat_iter0 := subs( v=v__1/cos(theta), yd_hat_iter0)
```

$$v_1 = v\cos(\theta)$$

$$yd\_hat\_iter0 := \frac{v_1\sin(\theta)}{\cos(\theta)} \quad (9)$$

4. take the time derivative of yd_hat_iter0 to find the r<1> which used in the next iteration

```
M0 > Lf_yd_hat_iter0 := LieDerivative(f,yd_hat_iter0);
     dydhat_dv1 :=diff(yd_hat_iter0,v__1);
     Lg1_ydhat:=LieDerivative(g__1,yd_hat_iter0);
     Lg2_ydhat:=LieDerivative(g__2,yd_hat_iter0);
     b1:=Vector[row]([Lg1_ydhat,Lg2_ydhat]); #there is no u in
     b1 BUT HAVE V
```

```
dot_yd_hat := simplify(yd_hat_iter0 + dydhat_dv1*v1dot +
b1.u); #equ22
```

$$Lf\_yd\_hat\_iter0 := 0$$

$$dydhat\_dv1 := \frac{\sin(\theta)}{\cos(\theta)}$$

$$Lg1\_ydhat := \frac{\tan(\phi)\left(\frac{v_1 \sin(\theta)^2}{\cos(\theta)^2} + v_1\right)}{l}$$

$$Lg2\_ydhat := 0$$

$$b1 := \begin{bmatrix} \frac{\tan(\phi)\left(\frac{v_1 \sin(\theta)^2}{\cos(\theta)^2} + v_1\right)}{l} & 0 \end{bmatrix}$$

$$dot\_yd\_hat := \frac{l \sin(\theta) \cos(\phi) \left(v_1 + v1dot\right) \cos(\theta) + v\, v_1 \sin(\phi)}{\cos(\phi)\, l \cos(\theta)^2} \quad \textbf{(10)}$$

5. After first time derivative, we notice the b1 vector are linear dependent on b0_tilde, So we have to do time derivative on yd_hat_iter0 2 times.
However, To simplify the calculation, we can just do the time derivative on b1*u, which can be seen as a function of (x,v1), to check if the b1(x,v1) of the 2nd derivative of yd_hat_iter0 can be linear independent to b0_tilde

```
M0 > b1u_dependent := simplify(subs(v=v__1/cos(theta),b1.u));
     Lgb1u:= Vector[row](2,simplify([LieDerivative(g__1,
     b1u_dependent),LieDerivative(g__2,b1u_dependent)]));
```

$$b1u\_dependent := \frac{v_1^2 \sin(\phi)}{l \cos(\phi) \cos(\theta)^3}$$

$$Lgb1u := \begin{bmatrix} \frac{3 v_1^2 \sin(\phi)^2 \sin(\theta)}{l^2 \cos(\phi)^2 \cos(\theta)^4} & \frac{v_1^2}{l \cos(\phi)^2 \cos(\theta)^3} \end{bmatrix} \quad \textbf{(11)}$$

the Lgb1u is not dependent on b0_tilde.; so **r<1>=[2]** which mean the yd_hat_iter0 do twist derivative to achieve an indepent b1

# Iteration 1

Since the input v__1 is related with state, we need to reconfig f to add v__1 as generalized state.

Then, do 2 more time derivative on yd_hat_iter0(y2d in (x, v1)) and assign auxillary input v__2 to it.

```
M0 > x_iter1 := [y1,y2,theta,phi,v__1,v1__d];
     DGsetup(x_iter1, M1):
     basis_vf:=[D_y1,D_y2,D_theta,D_phi,D_v__1,D_v1__d];
     f_iter1:=DGzip(<sysf,v1__d,v1__dd>,basis_vf);
     g__1_iter1 := DGzip(<sysg1,0,0>,basis_vf,"plus");
     g__2_iter1 := DGzip(<sysg2,0,0>,basis_vf,"plus");
     g_iter1:=[g__1_iter1,g__2_iter1]:
```

$$x\_iter1 := [y1, y2, \theta, \phi, v_1, v1_d]$$

$$basis\_vf := [D\_y1, D\_y2, D\_\theta, D\_\phi, D\_v_1, D\_v1_d]$$

$$f\_iter1 := v1_d D\_v_1 + v1_{dd} D\_v1_d$$

$$g_{1\_iter1} := \cos(\theta)\ D\_y1 + \sin(\theta)\ D\_y2 + \frac{\tan(\phi)\ D\_\theta}{l}$$

$$g_{2\_iter1} := D\_\phi \tag{12}$$

we represnet y2d in (x,v__1)

```
M1 > y2d := subs(v=v__1/cos(theta),y2d)
```

$$y2d := \frac{v_1 \sin(\theta)}{\cos(\theta)} \tag{13}$$

```
M1 > Lg1Lfh2_1:=LieDerivative(g__1_iter1,y2d);
     Lg2Lfh2_1:=LieDerivative(g__2_iter1,y2d);
     y2dd:=LieDerivative(f_iter1,y2d) + Lg1Lfh2_1*u[1] +
     Lg2Lfh2_1*u[2]:
     y2dd:=subs(v=v__1/cos(theta),y2dd)
```

$$Lg1Lfh2\_1 := \frac{\tan(\phi)\left(\frac{v_1 \sin(\theta)^2}{\cos(\theta)^2} + v_1\right)}{l}$$

$$Lg2Lfh2\_1 := 0$$

$$y2dd := \frac{v1_d \sin(\theta)}{\cos(\theta)} + \frac{\tan(\phi)\left(\frac{v_1 \sin(\theta)^2}{\cos(\theta)^2} + v_1\right) v_1}{l \cos(\theta)} \tag{14}$$

```
M1 > Lg1Lfh2_iter1:=simplify(LieDerivative(g__1_iter1,y2dd));
     Lg2Lfh2_iter1:=simplify(LieDerivative(g__2_iter1,y2dd));
     y2ddd :=(LieDerivative(f_iter1,y2dd) + Lg1Lfh2_iter1*u[1] +
```

```
Lg2Lfh2_iter1*u[2]);
y2ddd :=subs(v=v__1/cos(theta),y2ddd);
v__2= y2ddd
```

$$Lg1Lfh2\_iter1 := \frac{\sin(\phi)\left(v1_d\, l\cos(\theta)^2\cos(\phi) + 3\sin(\theta)\sin(\phi)\,v_1^2\right)}{l^2\cos(\phi)^2\cos(\theta)^4}$$

$$Lg2Lfh2\_iter1 := \frac{v_1^2}{l\cos(\phi)^2\cos(\theta)^3}$$

$$y2ddd := v1_d\left(\frac{\tan(\phi)\left(\frac{\sin(\theta)^2}{\cos(\theta)^2}+1\right)v_1}{l\cos(\theta)} + \frac{\tan(\phi)\left(\frac{v_1\sin(\theta)^2}{\cos(\theta)^2}+v_1\right)}{l\cos(\theta)}\right) + \frac{v1_{dd}\sin(\theta)}{\cos(\theta)} + \frac{\sin(\phi)\left(v1_d\, l\cos(\theta)^2\cos(\phi) + 3\sin(\theta)\sin(\phi)\,v_1^2\right)v}{l^2\cos(\phi)^2\cos(\theta)^4} + \frac{v_1^2\, dot\_phi}{l\cos(\phi)^2\cos(\theta)^3}$$

$$y2ddd := v1_d\left(\frac{\tan(\phi)\left(\frac{\sin(\theta)^2}{\cos(\theta)^2}+1\right)v_1}{l\cos(\theta)} + \frac{\tan(\phi)\left(\frac{v_1\sin(\theta)^2}{\cos(\theta)^2}+v_1\right)}{l\cos(\theta)}\right) + \frac{v1_{dd}\sin(\theta)}{\cos(\theta)} + \frac{\sin(\phi)\left(v1_d\, l\cos(\theta)^2\cos(\phi) + 3\sin(\theta)\sin(\phi)\,v_1^2\right)v_1}{l^2\cos(\phi)^2\cos(\theta)^5} + \frac{v_1^2\, dot\_phi}{l\cos(\phi)^2\cos(\theta)^3}$$

$$v_2 = v1_d\left(\frac{\tan(\phi)\left(\frac{\sin(\theta)^2}{\cos(\theta)^2}+1\right)v_1}{l\cos(\theta)} + \frac{\tan(\phi)\left(\frac{v_1\sin(\theta)^2}{\cos(\theta)^2}+v_1\right)}{l\cos(\theta)}\right) + \frac{v1_{dd}\sin(\theta)}{\cos(\theta)} + \frac{\sin(\phi)\left(v1_d\, l\cos(\theta)^2\cos(\phi) + 3\sin(\theta)\sin(\phi)\,v_1^2\right)v_1}{l^2\cos(\phi)^2\cos(\theta)^5} + \frac{v_1^2\, dot\_phi}{l\cos(\phi)^2\cos(\theta)^3} \quad \textbf{(15)}$$

Now we check the rank of the D_\circ

```
M1 > D_circ:=Matrix(2,2,[[Lgh1[1],Lgh1[2]],[Lg1Lfh2_iter1,
     Lg2Lfh2_iter1]]);
     Rank(D_circ)
```

$$D\_circ := \begin{bmatrix} \cos(\theta) & 0 \\ \frac{\sin(\phi)\left(v1_d\, l\cos(\theta)^2\cos(\phi) + 3\sin(\theta)\sin(\phi)\,v_1^2\right)}{l^2\cos(\phi)^2\cos(\theta)^4} & \frac{v_1^2}{l\cos(\phi)^2\cos(\theta)^3} \end{bmatrix}$$

$$2 \quad \textbf{(16)}$$

Now we get v__1( x , u[1] ) and v__2(x,v1,dot_v__1,u[2]) , we can represent the u[1] and u[2] in v,dot_v

```
M1 > u_inv := simplify(solve({v__1=y1d,v__2=y2ddd},{u[1],u[2]}))
```

$$u\_inv := \left\{ dot\_phi = \frac{1}{\cos(\theta)^2\, l\, v_1^{\,2}} \left( v_2\, l^2 \cos(\phi)^2 \cos(\theta)^5 - v1_{dd} \sin(\theta)\, l^2 \cos(\phi)^2 \cos(\theta)^4 \right. \right. \quad (17)$$

$$\left. \left. - 3 \sin(\phi) \cos(\theta)^2 \cos(\phi)\, l\, v1_d\, v_1 + 3 \cos(\phi)^2 \sin(\theta)\, v_1^{\,3} - 3\, v_1^{\,3} \sin(\theta) \right), v = \frac{v_1}{\cos(\theta)} \right\}$$

Corordinates Transformation, From original nonlinear system to linear system Z

First we set up the frame of linear system

```
M1 > (* assume(z__1,'real');
      assume (G>0);
      assume(z__2,'real');
      assume(z__3,'real');
      assume(z__4,'real');
      assume(z__5,'real');
      assume(z__6,'real');
      assume *)
```

```
M1 > z := [z__1, z__2, z__3, z__4,z__5,z__6];
      DGsetup(z,N):
      basis_vfz:=[D_z__1,D_z__2,D_z__3,D_z__4,D_z__5,D_z__6];
```

$$z := [z_1, z_2, z_3, z_4, z_5, z_6]$$

$$basis\_vfz := [D\_z_1, D\_z_2, D\_z_3, D\_z_4, D\_z_5, D\_z_6] \quad (18)$$

```
N > T := simplify(Transformation(M1, N,
     [z__1 = h__1,
     z__2 = h__2,
     z__3 = v__1*sin(theta)/cos(theta), # = y2_d or h2_d, i.e.,
     2nd output's 1st time derivative in (x,v1)
     z__4 = y2dd,  # 2nd output's 2nd time derivative, a function
     in (x,v1,dot_v1); generated from 1st iteration
     z__5 = v__1,
     z__6 = v1__d
     ]));
```

$$T := \left[ z_1 = y1, z_2 = y2, z_3 = \frac{v_1 \sin(\theta)}{\cos(\theta)}, z_4 = \frac{v1_d \sin(\theta)\, l \cos(\theta)^2 \cos(\phi) + v_1^{\,2} \sin(\phi)}{\cos(\phi)\, l \cos(\theta)^3}, z_5 = v_1, z_6 \right. \quad (19)$$

$$\left. = v1_d \right]$$

```
N > _EnvExplicit := true;
    Tinv := InverseTransformation(T);
```

$$_EnvExplicit := true$$

(20)

$$Tinv := \left[ y1 = z_1, y2 = z_2, \theta = \arctan\left( \frac{z_3}{\sqrt{z_3^2 + z_5^2}}, \frac{z_5}{\sqrt{z_3^2 + z_5^2}} \right), \phi = -\arctan\left( \frac{l\left(z_3 z_6 - z_4 z_5\right)}{\left(z_3^2 + z_5^2\right)^{3/2}} \right), v_1 = z_5, v1_d = z_6 \right] \tag{20}$$

```
M1 > LinSys := simplify(Pushforward(T, Tinv, simplify(subs
     (u_inv, f_iter1 +  u[1]*g__1_iter1 + u[2]*g__2_iter1))));
```

$$LinSys := z_5 D\_z_1 + z_3 D\_z_2 + z_4 D\_z_3 + v_2 D\_z_4 + z_6 D\_z_5 + v1_{dd} D\_z_6 \tag{21}$$

```
N > Ts:=Vector[column](4,[h__1,
    h__2,
    v__1*sin(theta)/cos(theta), # = y2_d or h2_d, i.e., 2nd
    output's 1st time derivative in (x,v1)
    y2dd]); # 2nd output's 2nd time derivative, a function in (x,
    v1,dot_v1); generated from 1st iteration
```

$$Ts := \begin{bmatrix} y1 \\ y2 \\ \frac{v_1 \sin(\theta)}{\cos(\theta)} \\ \frac{v1_d \sin(\theta)}{\cos(\theta)} + \frac{\tan(\phi)\left(\frac{v_1 \sin(\theta)^2}{\cos(\theta)^2} + v_1\right) v_1}{l \cos(\theta)} \end{bmatrix} \tag{22}$$

Now we give the linear state space model; And check the controllability

```
N > A:= Matrix(4,4,[[0,0,0,0],[0,0,1,0 ],[0,0,0,1],[0,0,0,0]]);
    B:=Matrix(4,2,[[1,0],[0,0],[0,0],[0,1]])
```

$$A := \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$B := \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \tag{23}$$

```
N > <B|A.B|MatrixPower(A,2).B|MatrixPower(A,3).B>;
    Rank(<B|A.B|A.A.B|A.A.A.B>)
```

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}_4 \tag{24}$$