Attitude Control for Fixed Wing Aircraft

Bethany Calvert Lily Cothren

April 20th, 2022

1 Introduction

This paper considers the open research problem of designing a feedback controller to minimize effects of oscillatory behaviors around trim conditions for fixed-wing aircraft. First, pre-simulated probes and gyroscopic measurements from [8] are utilized to construct the linearized model for flight dynamics. Literature indicates that the pitch and roll values are significantly under-damped, which causes the unfavorable oscillations around trim conditions [6], [9]. Accordingly, the main goal of the feedback controller will be to increase the damping of these values to drive lateral and longitudinal velocities and pitch and roll rates towards optimal values for a desirable aircraft trajectory. In the following section, we identify the linear model with physical parameters, performance goals; in Section 2, we describe design methods; in Section 3, we showcase numerical verification of our controllers, and finally individually discuss final thoughts prior to concluding summaries.[1]

1.1 Model Formulation

Following [8], we first consider a nonlinear model of aircraft forces and moments produced from non-dimensional stability derivatives, engine and control surface parameters, moments of inertia, and aircraft geometry. Specifically, nonlinear force and moment equations are derived to produce force and moment parameters from state and control inputs. Then, these force and moment parameters are implemented in rigid body equations of motion to obtain new states and their corresponding dynamics [8]. For this nonlinear model, [8] numerically solves the 6 degree-of-freedom nonlinear equations of motion for the aircraft in MATLAB Simulink. These equations are well documented [1],[2],[8],[3]. After these simulations, [6] splits force and moment modeling into those that are gravitational, propulsive, and aerodynamic. Then, Newton's Laws are used to determine another 6 nonlinear equations. Finally, [8] develops a linear model about the trim conditions.

Completing a system identification that quantifies each of these values is extremely complex, time consuming, and beyond the scope of the controls theory of this project. Accordingly, we utilize a predetermined state model with identified matrices from a prior field study [8] per the above discussion. To model flight dynamics for a fixed-wing aircraft, there are twelve state variables that represent the position and velocity along the x, y, and z-axes (six total), and another six variables corresponding to the angle and angular velocity about the x, y, and z-axes. Traditionally, the x, y, and z positions and the yaw angle are not included in aircraft dynamic analysis. The longitudinal state equations include the forward velocity (u), vertical velocity (w), pitch angle (θ), and the pitch rate (q). The lateral state equations include the sideways velocity (v), roll rate (p), roll angle (ϕ),

[1] *Notation.* We denote by $\mathbb{R}, \mathbb{R}_{>0}$ the set of real numbers and the set of positive real numbers, respectively. For vectors $x \in \mathbb{R}^n$, $\|x\|_2$ denotes the Euclidean norm of x.

and the yaw rate (r). The state space equations describing the translational and rotational motion are as follows [8],

$$\begin{bmatrix}\dot{u}\\ \dot{w}\\ \dot{q}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix} X_u & X_w & 0 & -g\cos\Theta_0 \\ \frac{Z_u}{1-Z_{\dot{w}}} & \frac{Z_w}{1-Z_{\dot{w}}} & \frac{u_0+Z_q}{1-Z_{\dot{w}}} & \frac{-g\sin\Theta_0}{1-Z_{\dot{w}}} \\ M_u + \frac{M_{\dot{w}}Z_u}{1-Z_{\dot{w}}} & M_w + \frac{M_{\dot{w}}Z_w}{1-Z_{\dot{w}}} & M_q + \frac{(u_0+Z_q)M_{\dot{w}}}{1-Z_{\dot{w}}} & \frac{-M_{\dot{w}}g\sin\Theta_0}{1-Z_{\dot{w}}} \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} u\\ w\\ q\\ \theta\end{bmatrix} + \begin{bmatrix} X_{\delta_e} & X_{\delta_T} \\ \frac{Z_{\delta_e}}{1-Z_{\dot{w}}} & \frac{Z_{\delta_T}}{1-Z_{\dot{w}}} \\ M_{\delta_e} + \frac{M_{\dot{w}}Z_{\delta_e}}{1-Z_{\dot{w}}} & M_{\delta_T} + \frac{M_{\dot{w}}Z_{\delta_T}}{1-Z_{\dot{w}}} \\ 0 & 0 \end{bmatrix} \begin{bmatrix}\delta_e\\ \delta_T\end{bmatrix} \tag{1}$$

$$\begin{bmatrix}\dot{v}\\ \dot{p}\\ \dot{\phi}\\ \dot{r}\end{bmatrix} = \begin{bmatrix} Y_v & Y_p & g\cos\Theta_0 & Y_r - u_0 \\ \frac{L_v+i_xN_v}{1-i_xi_z} & \frac{L_p+i_xN_p}{1-i_xi_z} & 0 & \frac{L_r+i_xN_r}{1-i_xi_z} \\ 0 & 1 & 0 & 0 \\ \frac{N_v+i_zL_v}{1-i_xi_z} & \frac{N_p+i_zL_p}{1-i_xi_z} & 0 & \frac{N_r+i_zL_r}{1-i_xi_z} \end{bmatrix} \begin{bmatrix} v\\ p\\ \phi\\ r\end{bmatrix} + \begin{bmatrix} Y_{\delta_r} & 0 \\ \frac{L_{\delta_r}+i_xN_{\delta_r}}{1-i_xi_z} & \frac{L_{\delta_a}+i_xN_{\delta_a}}{1-i_xi_z} \\ 0 & 0 \\ \frac{N_{\delta_r}+i_zL_{\delta_r}}{1-i_xi_z} & \frac{N_{\delta_a}+i_zL_{\delta_a}}{1-i_xi_z} \end{bmatrix} \begin{bmatrix}\delta_r\\ \delta_a\end{bmatrix} \tag{2}$$

The definitions of each variable throughout the equations can be found in [8].

From this, we further simplify by considering dynamics without disturbances ($\mathbf{d} \equiv \mathbf{0}$). Then, the linearized state space dynamics decoupled into lateral and longitudinal directions are modeled as,

$$\begin{aligned} \dot{\mathbf{x}}_{lon} &= A_{lon}\mathbf{x}_{lon} + B_{lon}\mathbf{u}_{lon} \\ \dot{\mathbf{x}}_{lat} &= A_{lat}\mathbf{x}_{lat} + B_{lat}\mathbf{u}_{lat}, \end{aligned} \tag{3}$$

where $\mathbf{x}_{lon} := [u/V_a^*,\, w,\, q,\, \theta]^\top$ and $\mathbf{x}_{lat} := [v,\, p,\, r,\, \phi]^\top$ (with V_a^* representing a chosen air speed value) are the aircraft state for the longitudinal and lateral directions, respectively, and $\mathbf{u}_{lon} := [\delta_e,\, \delta_t]^\top$ and $\mathbf{u}_{lat} := [\delta_a,\, \delta_r]^\top$ are the control input for the longitudinal and lateral directions, respectively. The matrices $A_{lon}, A_{lat}, B_{lon}$ and B_{lat} are given as,

$$\begin{gathered} A_{lon} = \begin{bmatrix} X_u & X_w/V_a^* & X_q/V_a^* & -g\cos(\Theta_0)/V_a^* \\ Z_uV_a^* & Z_w & Z_q & -g\sin(\Theta_0) \\ M_uV_a^* & M_w & M_q & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix},\ A_{lat} = \begin{bmatrix} Y_v & Y_p & Y_r & g\cos(\Theta_0) \\ L_v & L_p & L_r & 0 \\ N_v & N_p & N_r & 0 \\ o & 1 & \tan(\Theta_0) & 0 \end{bmatrix}, \\ B_{lon} = \begin{bmatrix} X_{\delta_e}/V_a^* & X_{\delta_t}/V_a^* \\ Z_{\delta_e} & 0 \\ M_{\delta_e} & 0 \\ 0 & 0 \end{bmatrix},\ B_{lat} = \begin{bmatrix} Y_{\delta_a} & Y_{\delta_r} \\ L_{\delta_a} & L_{\delta_r} \\ N_{\delta_a} & N_{\delta_r} \\ 0 & 0 \end{bmatrix}, \\ C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},\ D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \end{gathered} \tag{4}$$

The force and moment coefficients for the longitudinal and lateral state matrices are given in Table 1. The field study uses a well-developed model for a T-Twister fixed wing unmanned autonomous vehicle (UAV). The values for the aircraft geometry are found in Table 2, leading to the state space matrices of (9). Finally, trim conditions are all zero except for those listed in Table 3. With a working model to assess, we use numerical tools and classical state-space control techniques to conduct controls analysis as explained in the next section.

Force Parameter	Value	Units	Moment Parameter	Value	Units
X_u	-0.1271	s^{-1}	L_v	-1.1467	$(m \cdot s)^{-1}$
X_w	0.6409	s^{-1}	L_p	-15.7093	s^{-1}
X_q	-0.9106	$m \cdot s^{-1}$	L_r	2.6774	s^{-1}
Y_v	-0.3714	s^{-1}	M_u	0.1090	$(m \cdot s)^{-1}$
Y_p	0.8254	$m \cdot s^{-1}$	M_w	-2.1148	$(m \cdot s)^{-1}$
Y_r	-17.6451	$m \cdot s^{-1}$	M_q	-3.2853	s^{-1}
Z_u	-0.7655	s^{-1}	N_v	0.6400	$(m \cdot s)^{-1}$
Z_w	-6.3237	s^{-1}	N_p	-1.2356	s^{-1}
Z_q	16.9091	$m \cdot s^{-1}$	N_r	-0.5669	s^{-1}
X_{δ_e}	0.0018	$m \cdot s^{-2}$	L_{δ_a}	-5.3580	s^{-2}
X_{δ_t}	3.3846	$m \cdot s^{-2}$	L_{δ_r}	0.0316	s^{-2}
Y_{δ_a}	-0.0137	$m \cdot s^{-2}$	M_{δ_e}	-1.3996	s^{-2}
Y_{δ_r}	0.0556	$m \cdot s^{-2}$	N_{δ_a}	-0.2566	s^{-2}
Z_{δ_e}	-0.1234	$m \cdot s^{-2}$	N_{δ_r}	-0.1309	s^{-2}

Table 1: Linear model parameters for numerical simulations, [8, Table 2.3].

Geometric Parameter	Symbol	Value	Units	Mass Parameter	Symbol	Value	Units
Wing Span	b	3.067	m	Mass	m	5.74	kg
Wing Area	S	0.6282	m^2	Inertia	I_{xx}	1.2009	$kg \cdot m^s$
Mean Chord	c	.208	m	Inertia	I_{yy}	.9318	$kg \cdot m^s$
-	-	-	-	Inertia	I_{zz}	2.0734	$kg \cdot m^s$
-	-	-	-	Inertia	I_{xz}	.0946	$kg \cdot m^s$

Table 2: Linear model parameters for numerical simulations, [8, Table 2.3].

Trim State	Value	Units
h^*	1800	m
V_a^*	18	m/s
θ^*	0.0515	rad
u^*	17.9762	m/s
w^*	0.9263	m/s
δ_e^*	-0.537	rad
δ_t^*	17.92	fraction of max

Table 3: Trim conditions, [8, Table 2.2].

1.2 Problem Statement and Performance Goals

With the linear model given in (3), we aim to design a linear feedback controller that damps the oscillatory behavior of the pitch and roll rates near trim conditions. Accordingly, we aim to identify suitable matrices $K \in \mathbb{R}^{2\times4}, \bar{N} \in \mathbb{R}^{2\times2}$ to map states $x \in \mathbb{R}^4$ and reference inputs $r \in \mathbb{R}^2$ to control inputs $u \in \mathbb{R}^2$ for both lateral and longitudinal directions such that,

$$u = Kx + \bar{N}r. \tag{5}$$

Additionally, we plan to implement optimal control via a Linear Quadratic Controller and compare the control techniques.

1.3 Design Techniques

Since we consider a Multi-Input Multi-Output (MIMO) system, many frequency-based methods are infeasible to apply. Further, since the states of our model are tightly connected, we cannot decouple the variables and consider each as a subsystem (outside of the expected division between lateral and longitudinal states). Due to these difficulties, we conducted most of our useful analysis in the time domain.

First, we used class notes [3] to construct the matrices K, F for the controller u to drive states to trim conditions. Later, we considered how we might best select the gain matrix K to damp the oscillatory behaviors. Pursuing classical control methods, we studied and implemented the Linear Quadratic Regulator (LQR) method to formulate an optimal K matrix using two different cost functions. The details of these controller designs are given in the following section.

2 Supporting Analysis

In this section, we discuss details of our controller design in two parts. First, the basic controller is constructed to ensure stability of both lateral and longitudinal directions prior to implementing reference tracking, then optimal gain matrices K are chosen with respect to different cost function selections.

2.1 Feedback and Reference Tracking

To complete state feedback control on each of the lateral and longitudinal systems, each system is configured according to its dynamics using the `ss()` in Matlab. Next, the system stability and controllability is assessed. To further control the stability or tune the system response, state feedback control is implemented using the `place()` command to obtain a feedback gain matrix. Next, the reference tracking is implemented to eliminate the steady state error of the response of each state. The output tracks the input by changing the state feedback law to include tracking as stated in 7. $\bar{N}$ is calculated using the following equations:

$$\bar{N} = N_u + KN_x. \tag{6}$$

$$\begin{bmatrix} \mathbf{Nx} \\ \mathbf{Nu} \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix} \tag{7}$$

where $N_x \in \mathbb{R}^{4\times 2}, N_u \in \mathbb{R}^{2\times 2}$. Following the implementation of state feedback tracking, different poles are tested including slow, moderate, fast, and imaginary cases.

2.1.1 Longitudinal Control Discussion

The initial longitudinal dynamics are given as,

$$A_{lon} = \begin{bmatrix} -0.1271 & 0.0356 & -0.0506 & -0.5443 \\ -1.3779 & -6.3237 & 16.9091 & -0.5050 \\ 1.962 & -2.1148 & -3.2853 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, B_{lon} = \begin{bmatrix} 0.001 & 0.1880 \\ -0.1234 & 0 \\ -1.3996 & 0 \\ 0 & 0 \end{bmatrix}. \tag{8}$$

The C and D matrices are found in Sec. 1.1. Due to the size of the C matrix, the only output responses that are analyzed are the forward velocity and the pitch angle. Additionally, notice the dynamics for the pitch angle in the B matrix are zero. This means that the input control from the rudder and aileron do not directly affect the original pitch angle of the system. Physically, this assumption does not make sense, and is a consequence of the decision to pull field data instead of gathering data directly.

At a first glance, the poles of the uncompensated system are given as,

$$\text{poles}_{lon} = [-4.8193 + 5.7960j, -4.8193 - 5.793j, -0.0488 + 0.3776j, -0.0448 - 0.3776j].$$

The dynamics for the uncompensated system are given in Figures 1-2. Clearly, these poles all have negative real part, which means that the uncompensated system is already stable. However, the poles $-0.0448 \pm 0.3776j$ are too close to the origin, so we still apply a proportional matrix K to shift the poles to more stable locations.

After many iterations, we select the poles,

$$\text{poles}_{lon} = [-10, -10, -5, -5],$$

to obtain the dynamics in Figures 3-4. Notably, all values are significantly damped and the Bode diagrams exhibit less changing behaviors. The controller for this step is given as,

$$u = \begin{bmatrix} -1.4 & 1.5 & -8.4 & -35.7 \\ 46.5 & 15.3 & 350.7 & 2868.5 \end{bmatrix} x.$$

After shifting the poles to "stronger" locations, we implement a reference tracking matrix, $F \in \mathbb{R}^{2\times 4}$. The response for this additional term in the controller can be seen in Figures 5-6. Notably, the poles of the system do not change since the matrix A is unchanging. The controller for this step is given as,

$$u = \begin{bmatrix} -1.4 & 1.5 & -8.4 & -35.7 \\ 46.5 & 15.3 & 350.7 & 2868.5 \end{bmatrix} x + \begin{bmatrix} 0 & -35.7 \\ 43.3 & 2870.2 \end{bmatrix} r, \text{ } r \text{ is the reference to follow.}$$

In Figures 7-8, we experiment with placing the poles at different locations after implementing the reference tracking controller. For the fast poles, we implement

$$\text{poles}_{lon} = [-100, -100, -50, -50],$$

and for imaginary poles, we implement

$$\text{poles}_{lon} = [-0.5 + j, -0.5 - j, -0.95, -0.95].$$

As expected, the faster poles exhibit larger overshoot relative to the poles 10 times smaller, and the imaginary poles introduce undershoots, too. For the fast poles, the controller is,

$$u = \begin{bmatrix} -1.4 & 1.5 & -104.8 & -3572.4 \\ 764.4 & -14117.8 & 1598.6 & 285855.3 \end{bmatrix} x + \begin{bmatrix} 0 & -3572.4 \\ 4332.7 & 2870198.6 \end{bmatrix} r, \text{ } r \text{ is the reference to follow.}$$

Finally, we independently studied optimal control methods [7] to better select the gain matrix K. The responses for two different cost selections are given in Figures 11-14. For the fast poles, the controller is,

$$u = \begin{bmatrix} -1.76 & 0.30 & 5.52 & -0.25 \\ 4.96 & 2.13 & -5.38 & -1.48 \end{bmatrix} x + \begin{bmatrix} -0.06 & -0.15 \\ 5.14 & 1.26 \end{bmatrix} r, \text{ } r \text{ is the reference to follow.}$$

In Section 2.2, we discuss the methods for cost function selection.

2.1.2 Lateral Control Discussion

The initial lateral dynamics are as follows:

$$A_{lat} = \begin{bmatrix} -0.3714 & 0.8254 & -17.6451 & 9.7969 \\ -1.1467 & -15.7093 & 2.6774 & 0 \\ 0.6400 & -1.2356 & -0.5669 & 0 \\ o & 1 & 0.0515 & 0 \end{bmatrix}, B_{lat} = \begin{bmatrix} -0.0137 & 0.0556 \\ -5.3580 & 0.0316 \\ -0.2566 & -0.1309 \\ 0 & 0 \end{bmatrix}. \quad (9)$$

The C and D matrices are found in Sec. 1.1. Due to the size of the C matrix, the only output responses that are analyzed are the y-velocity and the roll angle. Additionally, notice the dynamics for the roll angle in the B matrix are zero. This means that the input control from the rudder and aileron do not directly affect the original roll angle of the system. Physically, this assumption does not make sense. This is a consequence of the decision to pull field data instead of gathering data directly. The poles of the uncompensated dynamics are given as,

$$\text{poles}_{lat} = [-15.588, -0.5666 + 3.707j, -0.5666 - 3.707j, 0.0739].$$

The positive pole yields an unstable step and bode response displayed in Figures 18-19. Physically, this response means that the plane will completely lose control of its lateral dynamics once the rudder or aileron are applied.

Upon assessing the instability of the system, the controllability must be investigated to ensure that the system may be stabilized. The controllability matrix was calculated and was full rank, indicating control analysis may be conducted. New poles were selected to be in the right half plane to yield satisfactory transient conditions. After iterating through many pole locations, the following poles were selected:

$$\text{poles}_{lat} = [-1, -1, -.5, -.5].$$

The stabilized step and bode response are displayed in Fig. 20-22.

The bode responses are displayed for the following control strategies, however they do not yield a satisfying frequency response. The response does not yield adequate phase and gain margin that give stability and control intuition due to the complicated dynamics at hand. As each state is not truly decoupled in nature, the corresponding bode plots yield unruly results.

Next, reference tracking is implemented in Fig. 23-26. This does not change the poles of the system, but it does eliminate steady state error. The controller that resulted in placing these poles is

$$u = \begin{bmatrix} 0.1838 & 2.6957 & -0.5315 & -0.0656 \\ -5.0834 & 8.3001 & -5.2138 & 5.9224 \end{bmatrix} x + \begin{bmatrix} 0.0009 & -0.0926 \\ 0.2105 & 0.0896 \end{bmatrix} r.$$

Next, poles are placed very far into the left half plane and extended onto the imaginary planes as well. The fast poles are chosen as

$$\text{poles}_{lat} = [-100, -100, -90, -50].$$

The controller for the response is

$$u = \begin{bmatrix} 7.2458 & -32.2392 & -4.0845 & -1662.2255 \\ 2104.7599 & 29.3360 & -242.2768 & 2538.9549 \end{bmatrix} x + \begin{bmatrix} 0.0009 & -0.0926 \\ 0.2105 & 0.0896 \end{bmatrix} r,$$

with responses displayed in Fig. 27-29. The step response of these poles is extremely fast, and the steady state tracks to zero instead of one. While the response is fast with minimal overshoot, the controller does not actually have a real affect on the lateral velocity and roll angle physically.

Implementing this controller would not be successful in the real world because no controller could actually control the state inputs as successfully physically as it can mathematically. The imaginary poles are chosen as

$$\text{poles}_{lat} = [-1, -7 + .7j, -7 - 7j, -.5].$$

with controller

$$u = \begin{bmatrix} 0.1676 & 1.7225 & -1.4285 & -0.1081 \\ -2.2327 & 12.0618 & -56.9943 & 35.8350 \end{bmatrix} x + \begin{bmatrix} 0.0009 & -0.0926 \\ 0.2105 & 0.0896 \end{bmatrix} r.$$

The responses of the imaginary poles are depicted in 30-33. A significant amount of overshoot is introduced into two of the state responses. As the value of the damping coefficient decreases to a value less than one, the overshoot will increase. If all poles are introduced as complex conjugates, then all of the state responses experience increased overshoot for this system. Also, the reference tracking goes to zero instead of one. Physically, this does not make much sense. Our intuition is that due to the dynamics from the field data that we used, combined with assessing a complex MIMO system with coupled states has affected the controllers influence on the response of the system. This is a larger issue in the lateral modes, as the field study focuses on more specific trim conditions and dynamics for the longitudinal modes.

2.2 Optimal Control via LQR

We reformulate our research question of damping the pitch and roll angles into an optimization problem in the following manner. We aim to determine a linear feedback controller $u = Kx$ where the matrix K satisfies the following optimization problem,

$$u \in \operatorname{argmin}_{u\in\mathbb{R}^2} \int_{t_0}^{t} x(\tau)^\top Q x(\tau) + 2x(\tau)^\top N u(\tau) + u(\tau)^\top R u(\tau) d\tau, \tag{10}$$

where Q is a matrix that penalizes states far from the origin, R penalizes too large of inputs (and thus rewards smaller, more realistic control inputs), and N relates the state the inputs. Without disturbances, this optimization problem is called the Linear Quadratic Regulator problem. Fortunately, once the cost function to be optimized is selected according to the problem's design goals, obtaining the matrix K for the controller $u = Kx$ is a trivial task that can be computed in MATLAB via the function `lqr()`. However, selecting the matrices Q, N, and R can be difficult. Essentially, we need to model what it means to damp the pitch and roll rates for the longitudinal and lateral directions, respectively, within the matrix parameters for quadratic cost functions.

As a first try, we attempted to minimize the square of the dynamics $\dot{x}$, reasoning that minimizing the square magnitude of the dynamics, $\|\dot{x}\|_2^2$, translates to minimal movement in the trajectory of the state. Note that we use the square of $\|\dot{x}\|_2$ to ensure positive definite-ness. For this attempt, our cost function for each direction (longitudinal and lateral) is:

$$\min \int_{t_0}^{t} (\dot{x}(\tau)) d\tau = \min \int_{t_0}^{t} x(\tau)^\top (A^\top A) x(\tau) + 2x(\tau)^\top (A^\top B) u(\tau) + u(\tau)^\top (B^\top B) u(\tau) d\tau. \tag{11}$$

To satisfy positive-definite requirements of the costs, we further multiplied the resulting costs by constants $r_1, r_2, r_3 \geq 0$ against $A^\top A, A^\top B, B^\top B$, respectively, to yield the following matrices for each direction. Using $r_1 = r_2 = 1, r_3 = 0.0001$, we obtain the longitudinal gain matrix

$$K = \begin{bmatrix} -3.7614 & 4.4843 & -12.8680 & -73.1354 \\ 2.7047 & 37.1614 & -40.3538 & -182.7283 \end{bmatrix}.$$

Using $r_1 = r_2 = r_3 = 1$, the lateral gain matrix is given by

$$K = \begin{bmatrix} -0.0149 & -2.1500 & -4.6133 & 1.8797 \\ -2.3436 & 23.6225 & -846.4753 & 461.5295 \end{bmatrix}.$$

The step responses are displayed in Figures 11-12 for the longitudinal direction and in Figures 34-35 for the lateral direction. Initially, the lateral system yields an unstable response. The Q, R, and N matrices are tuned until this response becomes stable. In both of these gain matrices, it is seen that some of the matrix entries impose extremely high gains on the system. While the responses of optimal control are adequate, it is unrealistic to actually implement these control gains into a physical system. The Q, R, and N matrices can be further tuned to fix the system gains.

Secondly, we attempted to script the matrices Q, R, and N to model costs associated with deviation from trim conditions. For simplicity, we set $N \equiv 0$ and $Q \equiv \mathbb{I}$. By setting $Q \equiv \mathbb{I}$, we hope to remove weights associated with stabilizing the state $x_{lon}(t)$ since the longitudinal state is already stable and to focus on how much influence the inputs $u(t)$ can affect. Towards this, we tested values for the matrix R and identified that for some constant $k > 0$, set $Q \equiv k \cdot 100 \cdot \mathbb{I}, R = k \cdot \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix}$, and $N \equiv 0$ so that our cost function is

$$\min \int_{t_0}^{t} J(x(\tau), u(\tau)) d\tau := \min \int_{t_0}^{t} k \cdot 100 \cdot x(\tau)^\top \mathbb{I} x(\tau) + k \cdot u(\tau)^\top R u(\tau) d\tau, \;\; k \in \mathbb{R}_{>0}, \tag{12}$$

corresponds to damped trajectories. Weighting Q extremely high while weighting R extremely low, means that we are influencing the controller to increase the cost of the system performance while lowering the cost of the system controller effort. In turn, we are leaning on our LQR control to provide the lowest cost to the cost function that suites our needs, and we use the K matrix yielded from this optimum trajectory to place the poles of our new system. Therefore, since R is weighted lower in this case, we are leaning on controller effort which will result in an expensive controller in the physical system. Using constant $k = 2$, this outputs longitudinal gain matrix

$$K = \begin{bmatrix} -0.2598 & -4.0382 & -11.9287 & -10.4828 \\ 9.4384 & -0.0709 & 0.0418 & 0.0229 \end{bmatrix}.$$

The corresponding step response for the longitudinal mode is given in Figures 13-14. For the lateral mode, a similar procedure using $k = 2$ yields a lateral gain matrix

$$K = \begin{bmatrix} 0.8800 & -99.6564 & -8.2203 & -112.1439 \\ 99.1664 & 7.0330 & -144.2600 & 79.5078 \end{bmatrix}.$$

The corresponding step response for the lateral mode is given in Figures 36-37.

Comparing the state feedback reference tracking to LQR control, LQR is a much more effective way to achieve an optimal gain matrix. However, high gains are still introduced into the system. These gains could be fixed by adjusting the weighting between Q and R in order to achieve a controller that is more realistic for controlling the physical system. Additionally, LQR is a less time consuming. Since LQR control takes into account the designer's intentions for system performance versus controller effort, there is more intuition on how to iterate through the weighting process. However, for a complicated dynamic system, state feedback offers no intuition on how to iterate through pole locations resulting in a more time consuming process.

3 Results

In this section, we tabulate relevant plots. In particular, we present the step responses, step response information (overshoot percentage, rise time, and settling time), and Bode diagrams for the longitudinal and lateral directions, respectively.

Longitudinal Plots

Uncompensated Longitudinal System

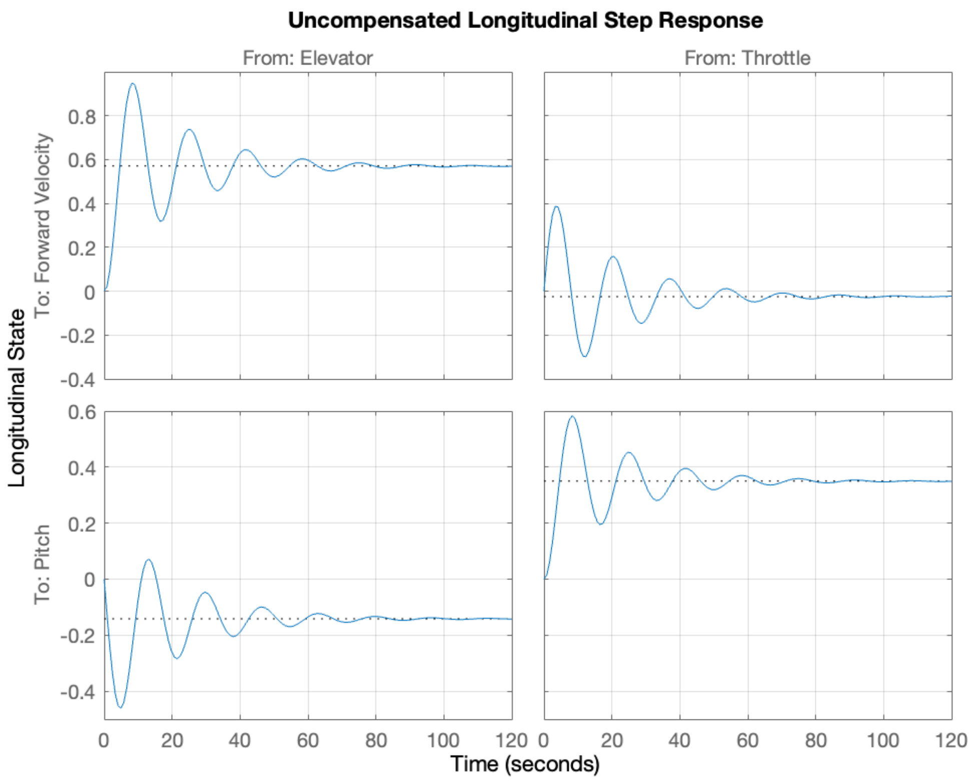

Figure 1: The uncompensated longitudinal step response with information below.

Uncompensated Step Information		
Overshoot (Percent)		
-	From: Elevator	From Throttle
To: Forward Velocity	66.4318	1.1233e+03
To: Pitch Angle	225.7388	66.7001
Rise Time (s)		
-	From: Elevator	From Throttle
To: Forward Velocity	3.0434	0.1505
To: Pitch Angle	0.7552	3.0245
Settling Time (s)		
-	From: Elevator	From Throttle
To: Forward Velocity	76.8211	80.4350
To: Pitch Angle	81.4895	76.7798

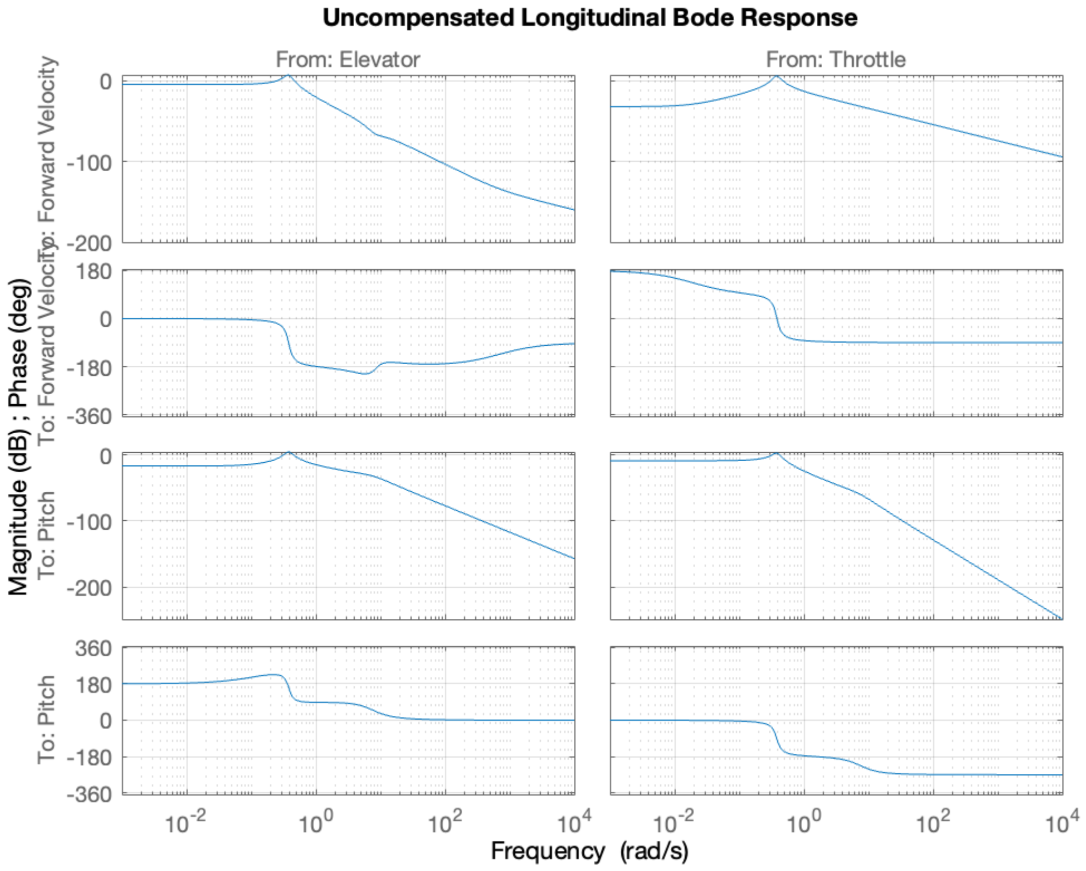

Figure 2: The uncompensated longitudinal Bode diagrams.

Stronger Longitudinal Poles

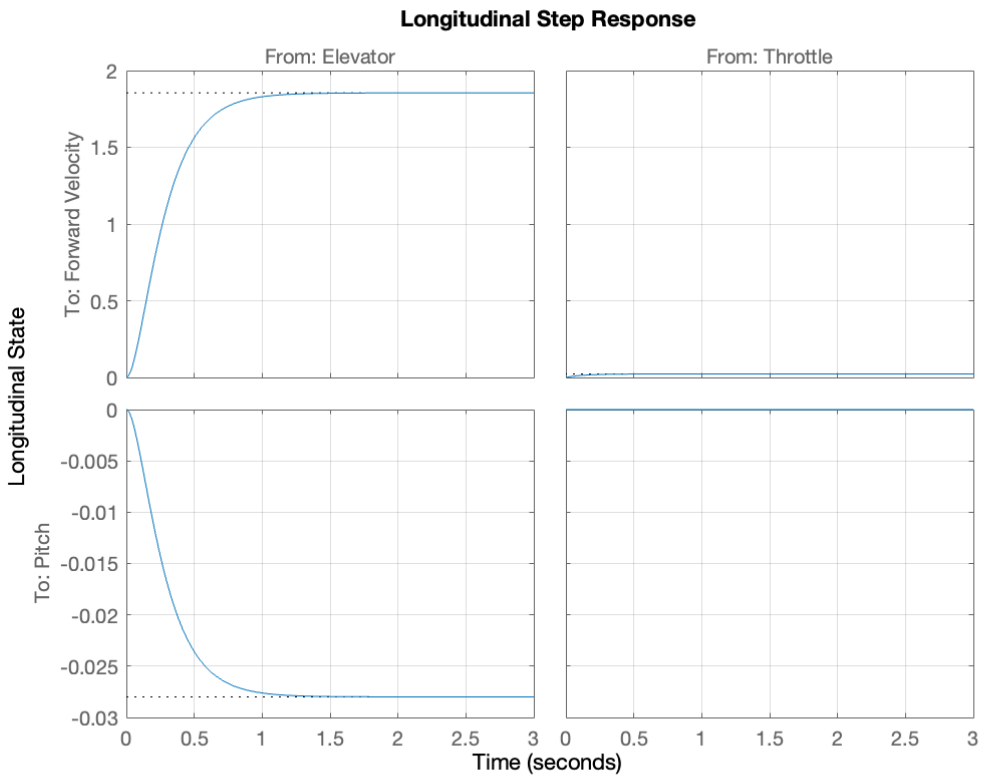

Figure 3: The longitudinal step response with "stronger" poles with information below.

"Stronger" Poles Step Information		
Overshoot (Percent)		
-	From: Elevator	From Throttle
To: Forward Velocity	0	0
To: Pitch Angle	0	0
Rise Time (s)		
-	From: Elevator	From Throttle
To: Forward Velocity	0.5180	0.3077
To: Pitch Angle	0.5180	0.6409
Settling Time (s)		
-	From: Elevator	From Throttle
To: Forward Velocity	0.9200	0.06001
To: Pitch Angle	0.9200	1.5906

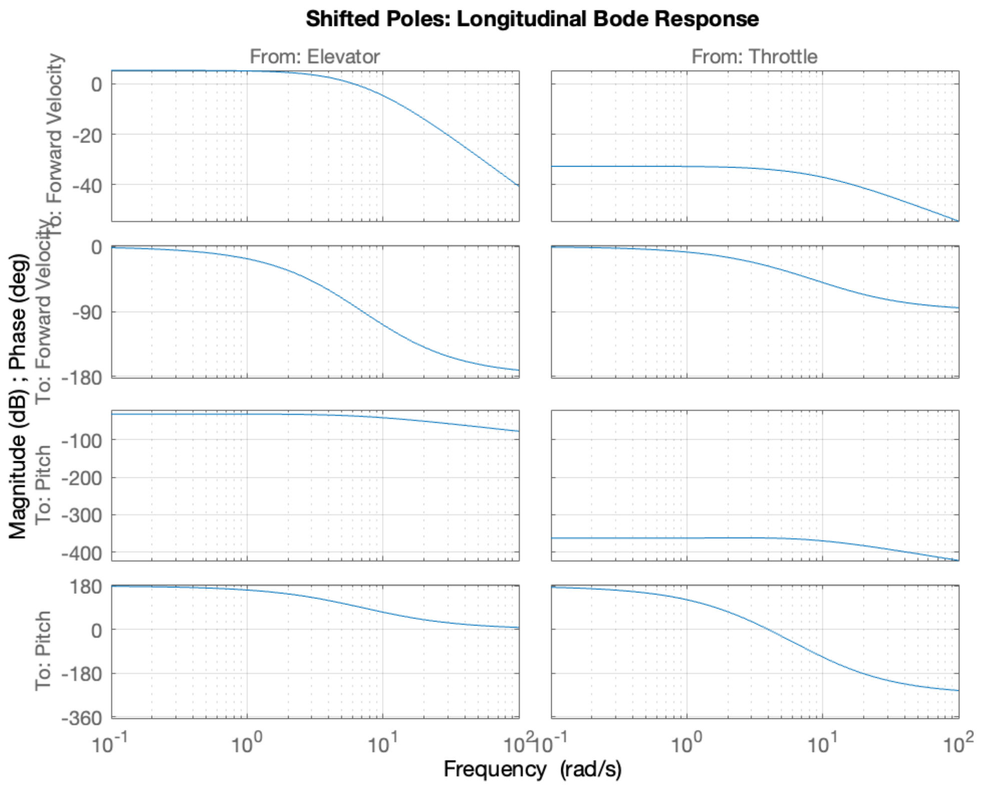

Figure 4: The longitudinal Bode diagrams with “stronger” poles.

Introduce Reference Tracking (RT) for Longitudinal Direction

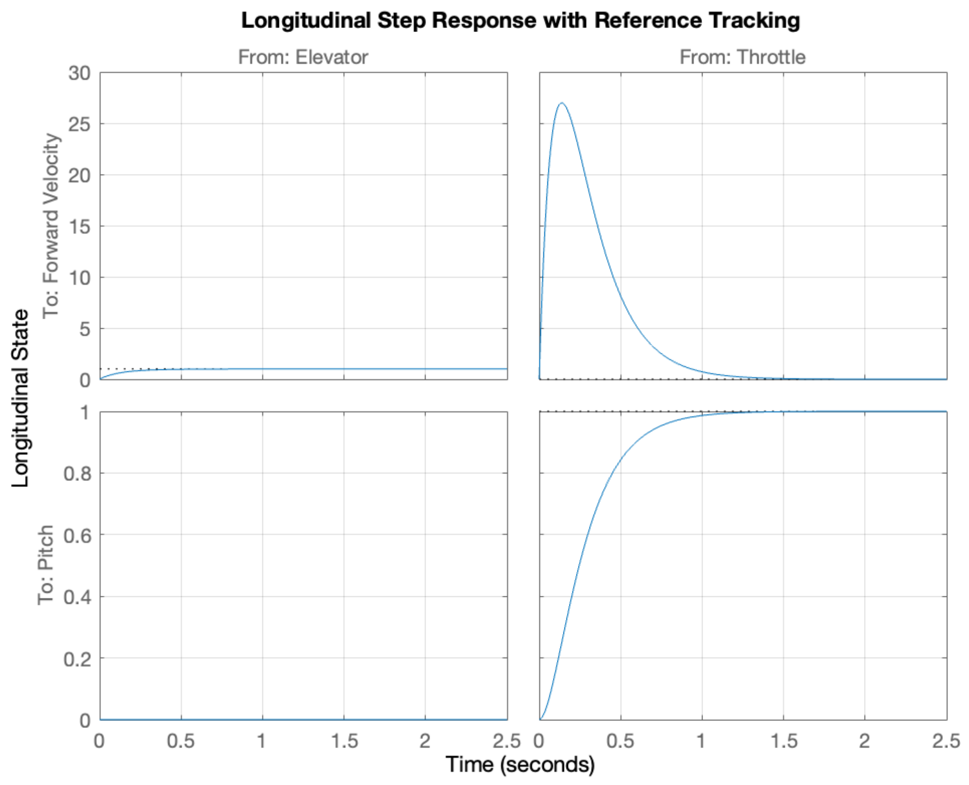

Figure 5: The longitudinal step response with reference tracking with information below.

Reference Tracking Step Information		
Overshoot (Percent)		
-	From: Elevator	From Throttle
To: Forward Velocity To: Pitch Angle	2.6640e+03 0	2.6604e+03 0
Rise Time (s)		
-	From: Elevator	From Throttle
To: Forward Velocity To: Pitch Angle	0.0016 0.5180	0.0016 0.5180
Settling Time (s)		
-	From: Elevator	From Throttle
To: Forward Velocity To: Pitch Angle	1.0606 0.9200	1.0606 0.9200

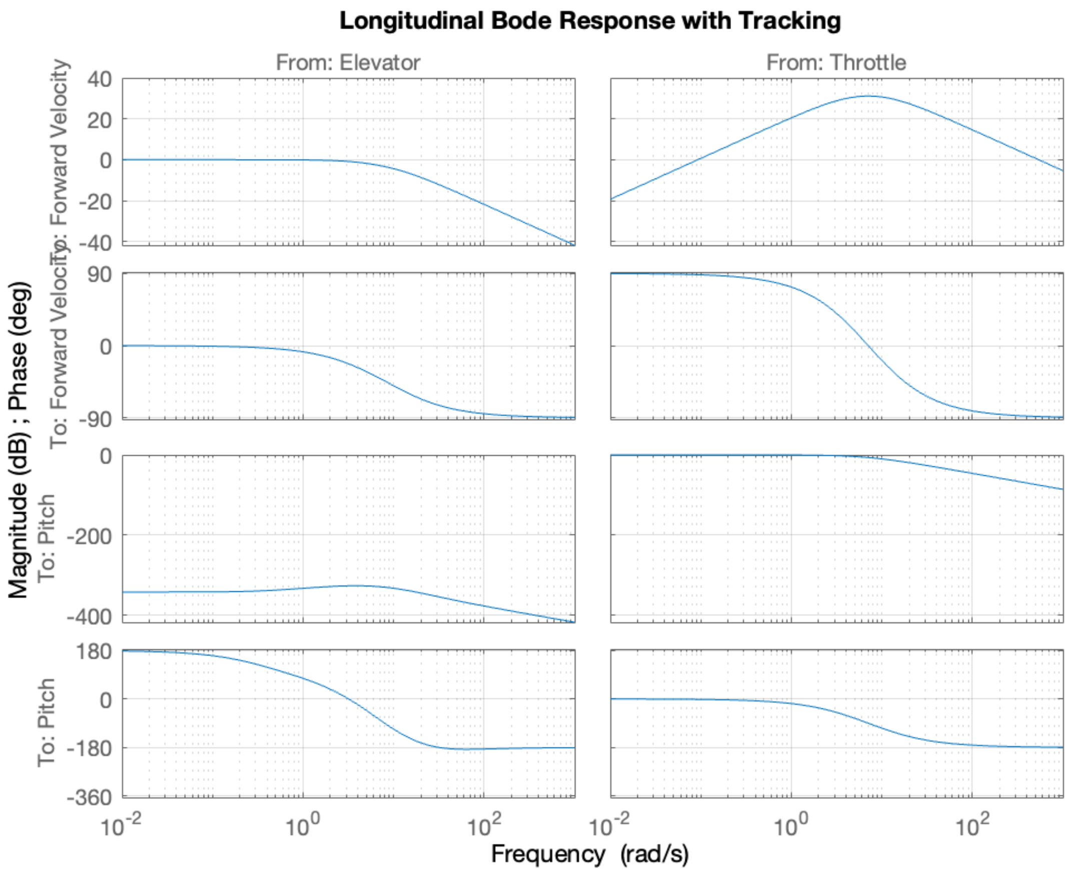

Figure 6: The longitudinal Bode diagrams with reference tracking.

Fast Poles with Reference Tracking (RT)

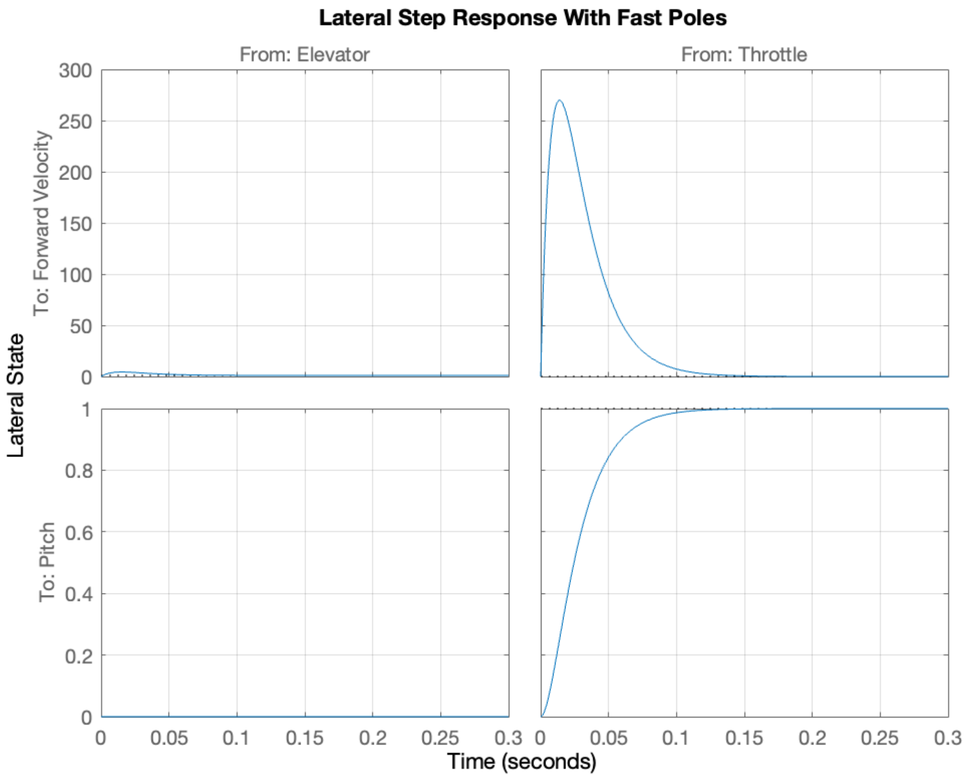

Figure 7: The longitudinal step response with reference tracking for fast poles with information below.

Longitudinal Fast Poles with RT Step Information		
Overshoot (Percent)		
-	From: Elevator	From Throttle
To: Forward Velocity To: Pitch Angle	2.7317e+04 0	2.7317e+04 0
Rise Time (s)		
-	From: Elevator	From Throttle
To: Forward Velocity To: Pitch Angle	1.5645e-05 0.5180	1.5645e-05 0.5180
Settling Time (s)		
-	From: Elevator	From Throttle
To: Forward Velocity To: Pitch Angle	0.1059 0.9200	0.1059 0.9200

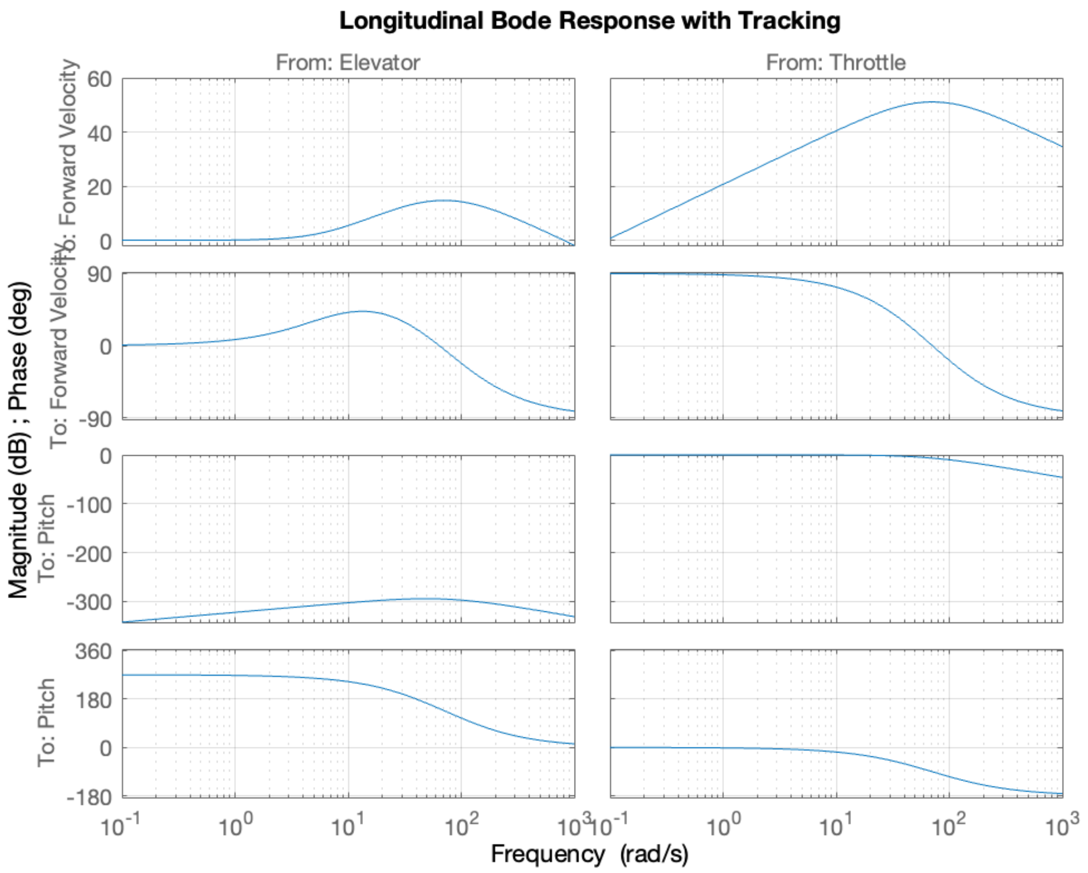

Figure 8: The longitudinal Bode diagrams with reference tracking for fast poles.

Longitudinal Imaginary Poles with Reference Tracking (RT)

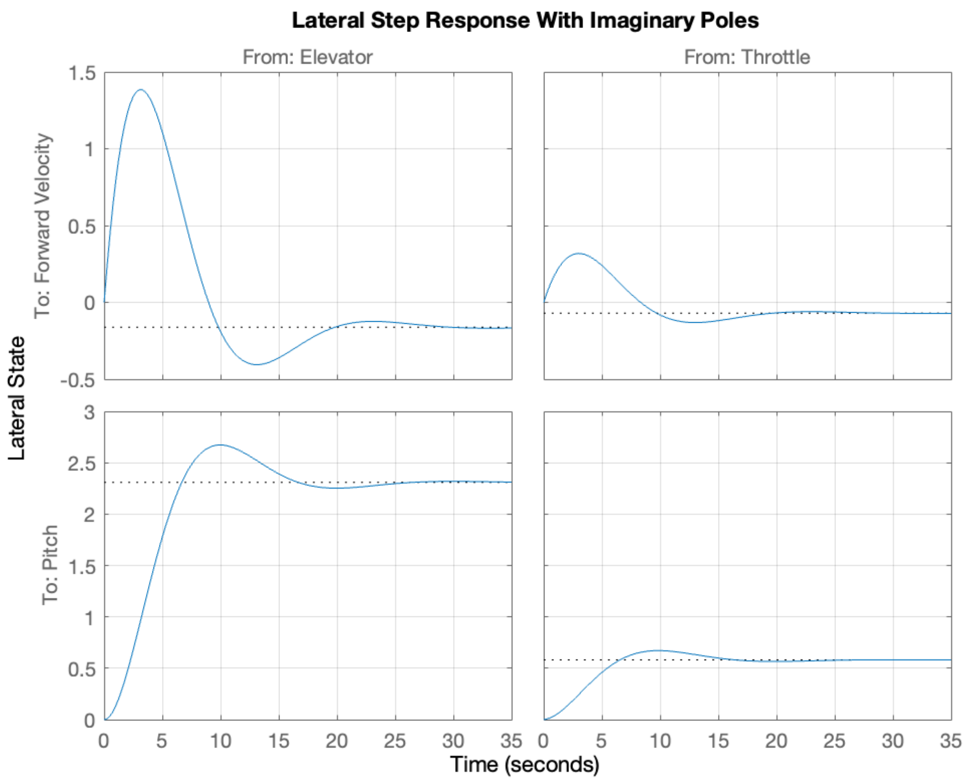

Figure 9: The longitudinal step response with reference tracking for imaginary poles with information below.

Longitudinal Imaginary Poles with RT Step Information		
Overshoot (Percent)		
-	From: Elevator	From Throttle
To: Forward Velocity	131.0142	131.0142
To: Pitch Angle	15.7400	15.7400
Rise Time (s)		
-	From: Elevator	From Throttle
To: Forward Velocity	0.7666	0.7666
To: Pitch Angle	4.5244	4.5244
Settling Time (s)		
-	From: Elevator	From Throttle
To: Forward Velocity	25.0219	25.0219
To: Pitch Angle	21.8714	21.8714

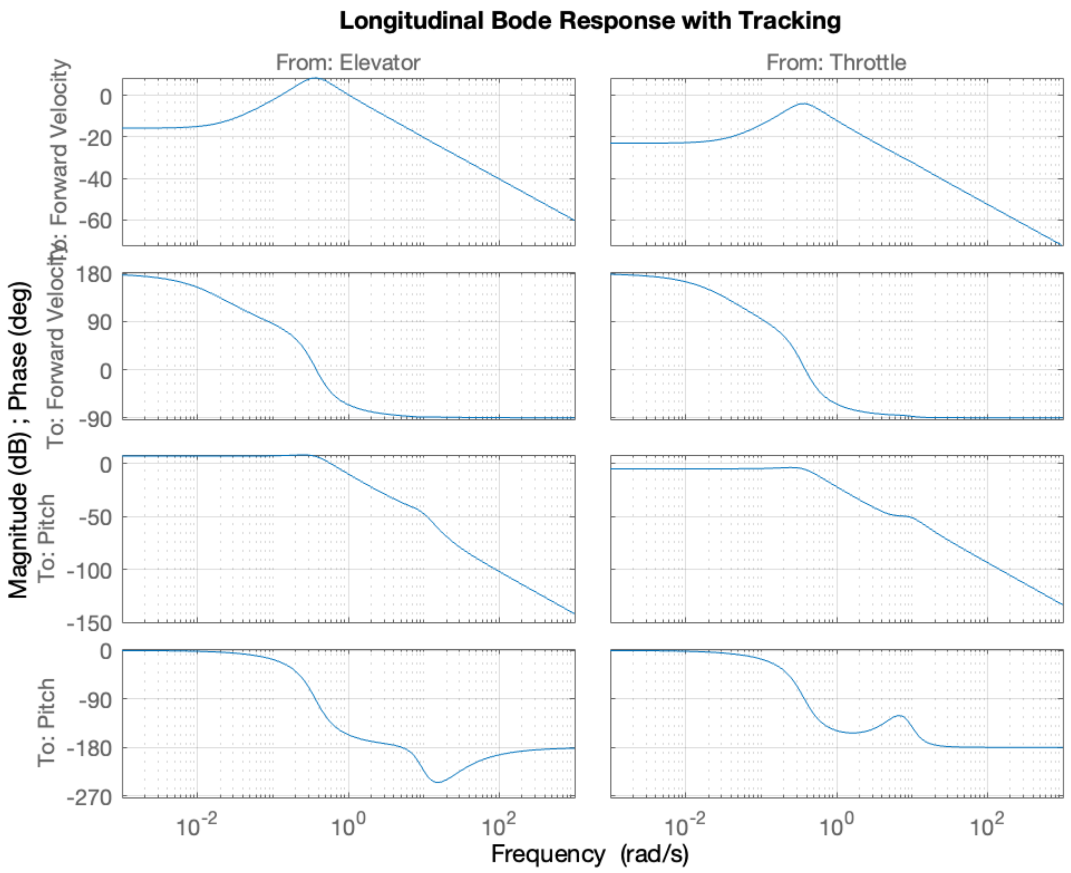

Figure 10: The longitudinal Bode diagrams with reference tracking for imaginary poles.

Reference Tracking Minimizing Dynamics, LQR pt. 1

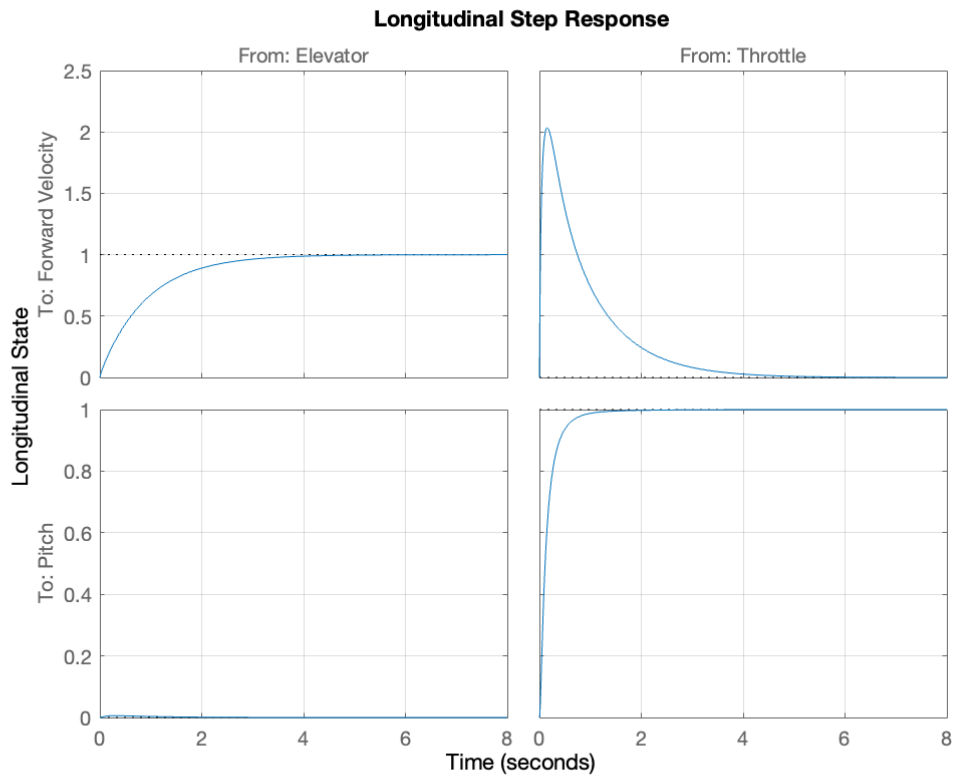

Figure 11: The longitudinal step response with reference tracking using LQR, pt. 1.

RT with LQR, pt. 1 Step Information		
Overshoot (Percent)		
-	From: Elevator	From Throttle
To: Forward Velocity	121.9204	121.9204
To: Pitch Angle	0	0
Rise Time (s)		
-	From: Elevator	From Throttle
To: Forward Velocity	0.0158	0.0158
To: Pitch Angle	0.3563	0.3563
Settling Time (s)		
-	From: Elevator	From Throttle
To: Forward Velocity	3.5545	3.5545
To: Pitch Angle	0.7642	0.7642

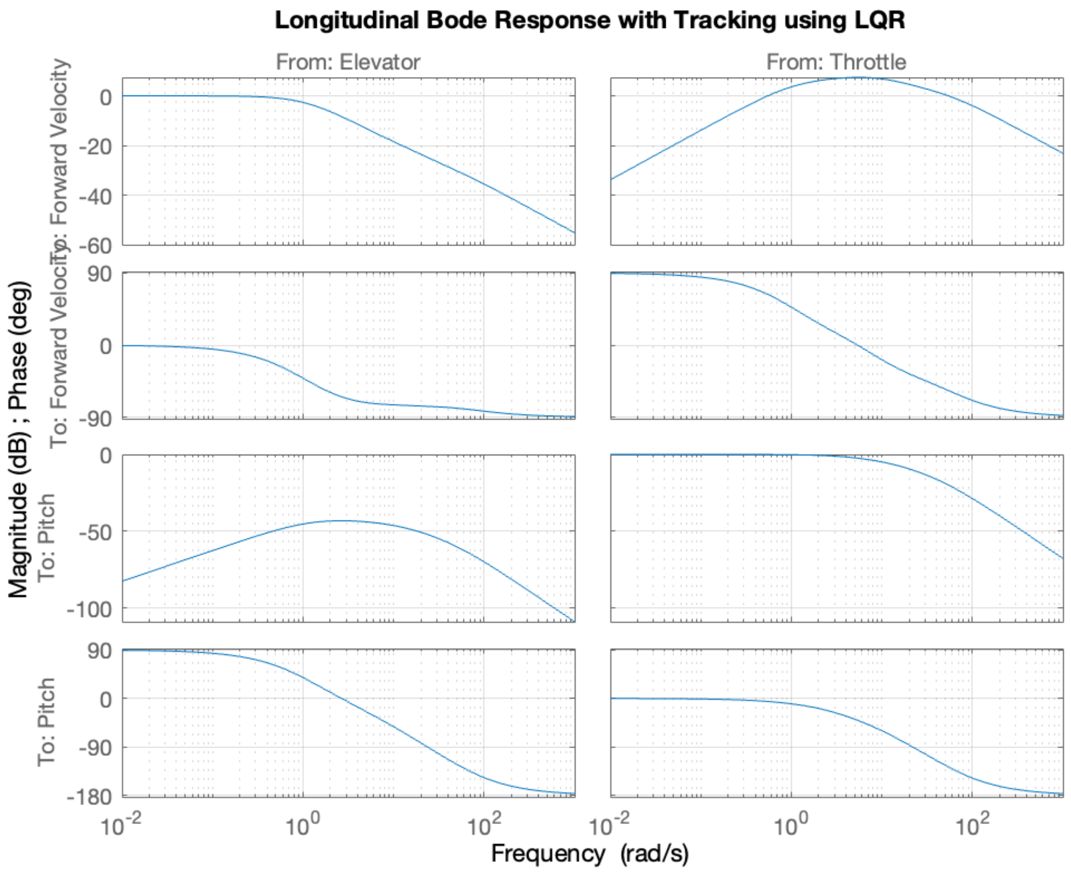

Figure 12: The longitudinal step response and Bode diagrams with reference tracking using LQR, pt. 1.

Reference Tracking using Chosen Costs, LQR pt. 2

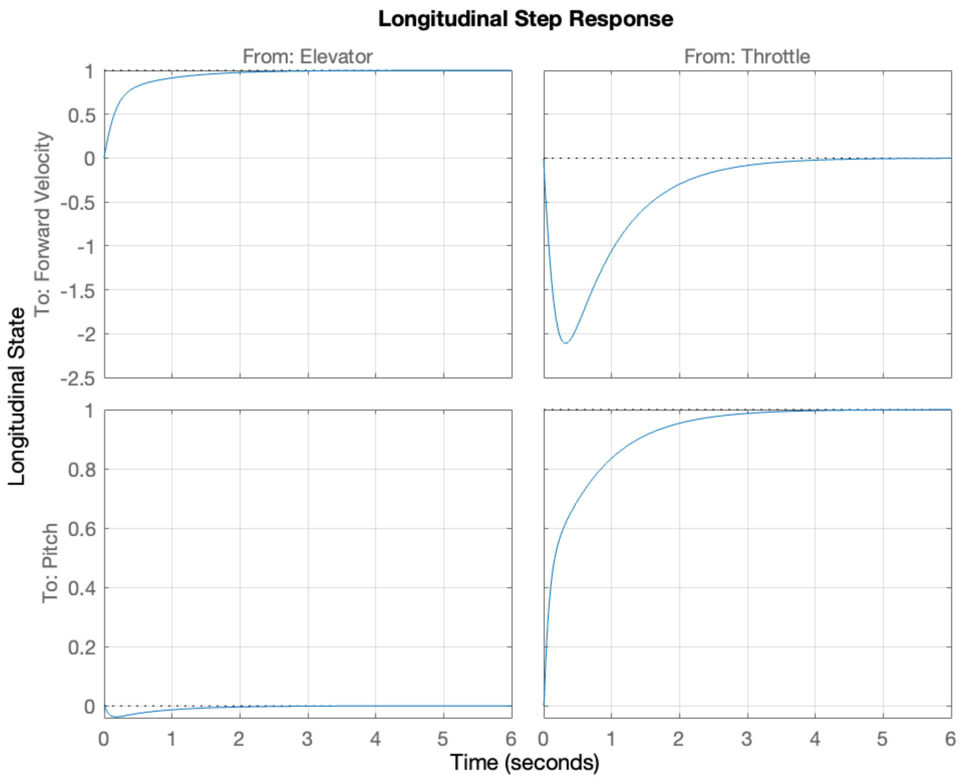

Figure 13: The longitudinal step response with reference tracking using LQR, pt. 2.

RT with LQR, pt. 2 Step Information		
Overshoot (Percent)		
-	From: Elevator	From Throttle
To: Forward Velocity	0	0
To: Pitch Angle	0	0
Rise Time (s)		
-	From: Elevator	From Throttle
To: Forward Velocity	1.7326	1.7326
To: Pitch Angle	1.4305	1.4305
Settling Time (s)		
-	From: Elevator	From Throttle
To: Forward Velocity	3.5028	3.5028
To: Pitch Angle	2.7229	2.7229

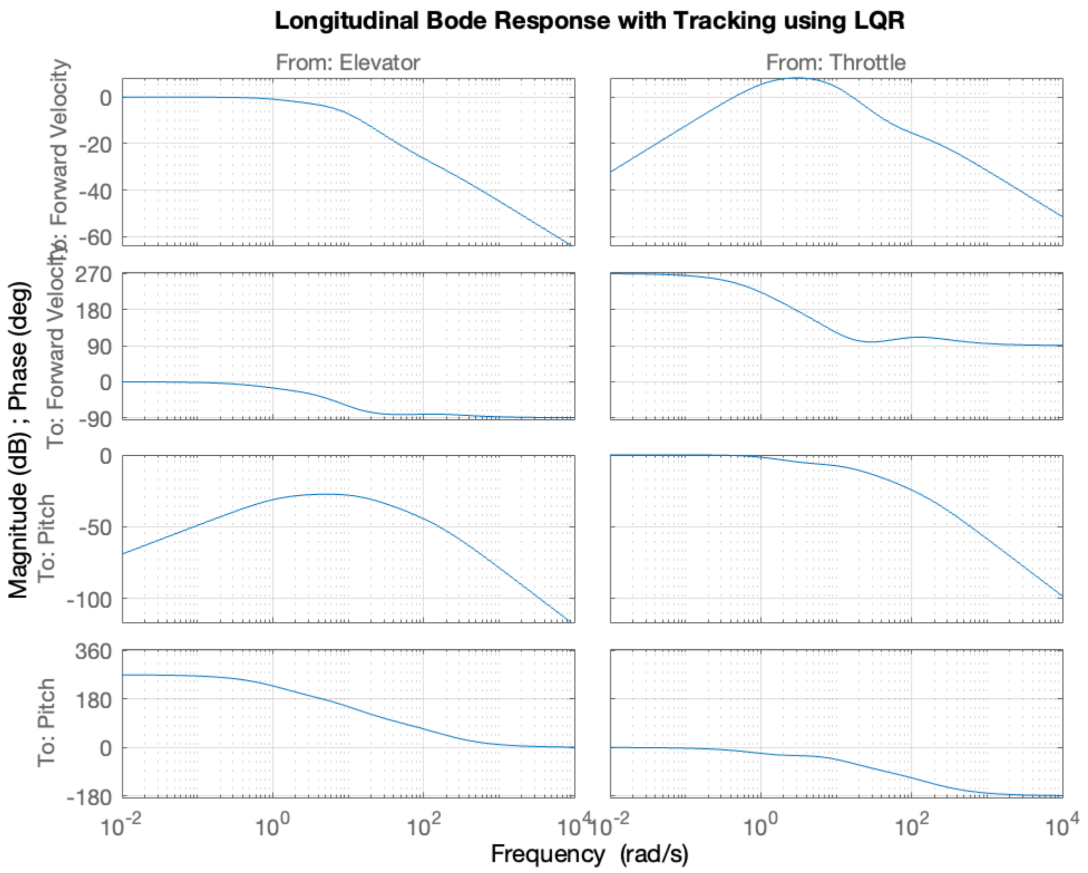

Figure 14: The longitudinal Bode diagrams with reference tracking using LQR, pt. 2.

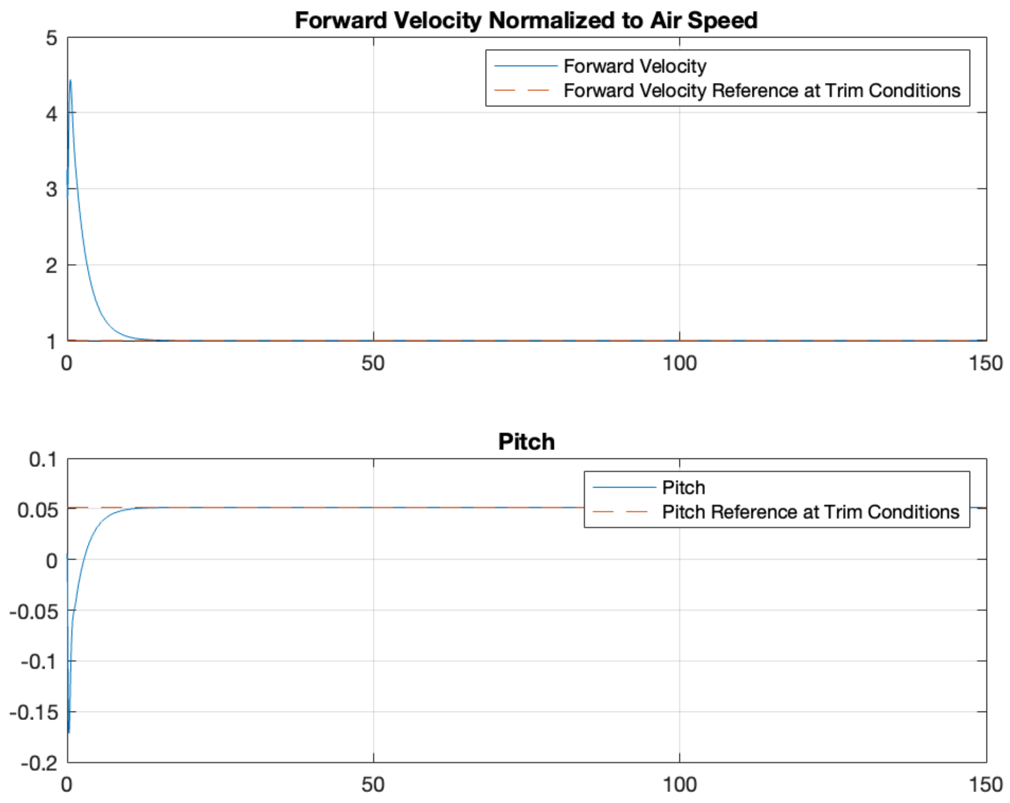

Figure 15: A subfigure

Figure 16: The longitudinal step response using $u = Kx$ with minimized cost $\int_{t_0}^{t} (\dot{x}(\tau))d\tau$.

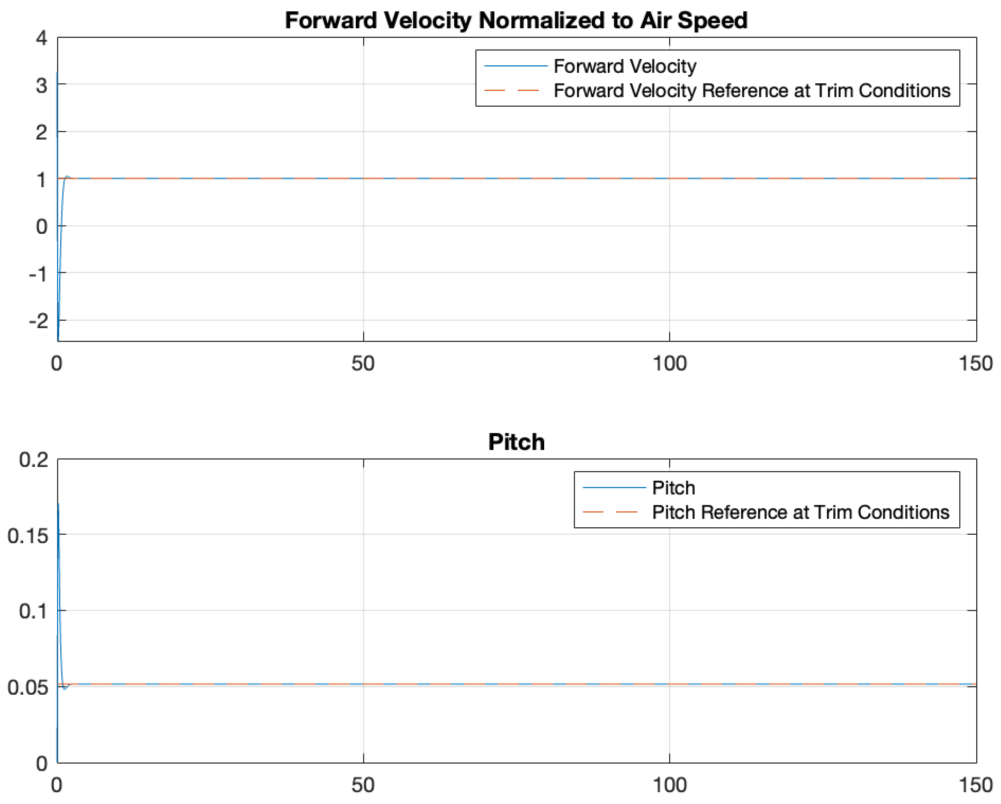

Figure 17: The longitudinal step response using $u = Kx$ with minimized cost $\int_{t_0}^{t} J(x(\tau), u(\tau))d\tau$.

Lateral Plots

Uncompensated Lateral System

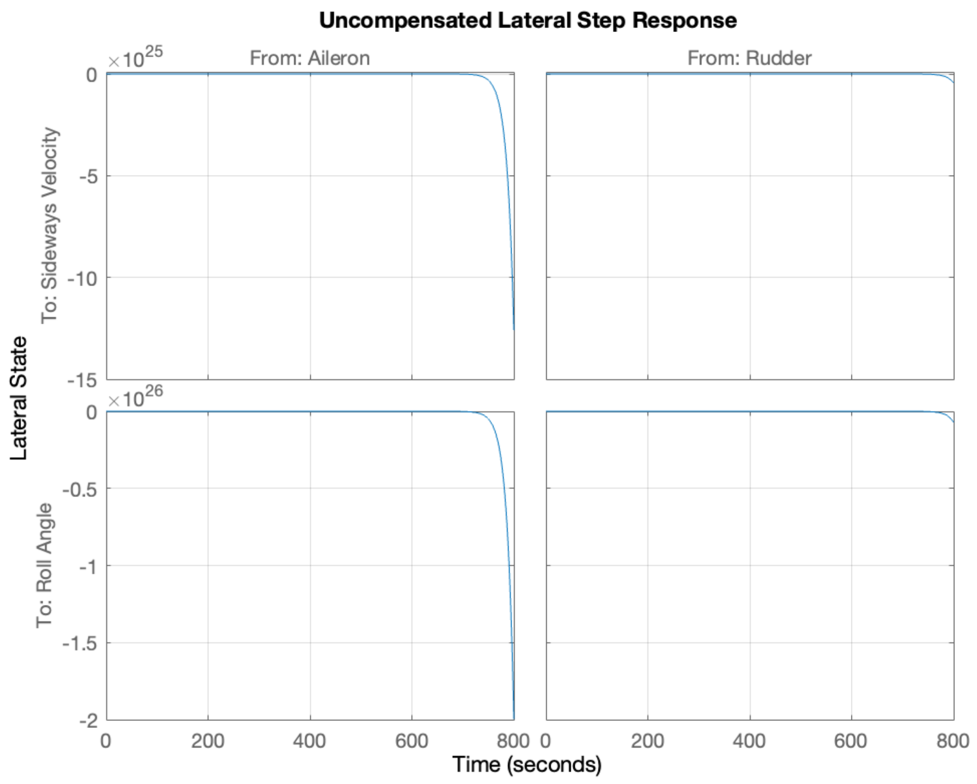

Figure 18: The uncompensated lateral step response with information below.

Uncompensated Step Information		
Overshoot (Percent)		
-	From: Aileron	From Rudder
To: Sideways Velocity	∞	∞
To: Roll Angle	∞	∞
Rise Time (s)		
-	From: Aileron	From Rudder
To: Sideways Velocity	∞	∞
To: Roll Angle	∞	∞
Settling Time (s)		
-	From: Aileron	From Rudder
To: Sideways Velocity	∞	∞
To: Roll Angle	∞	∞

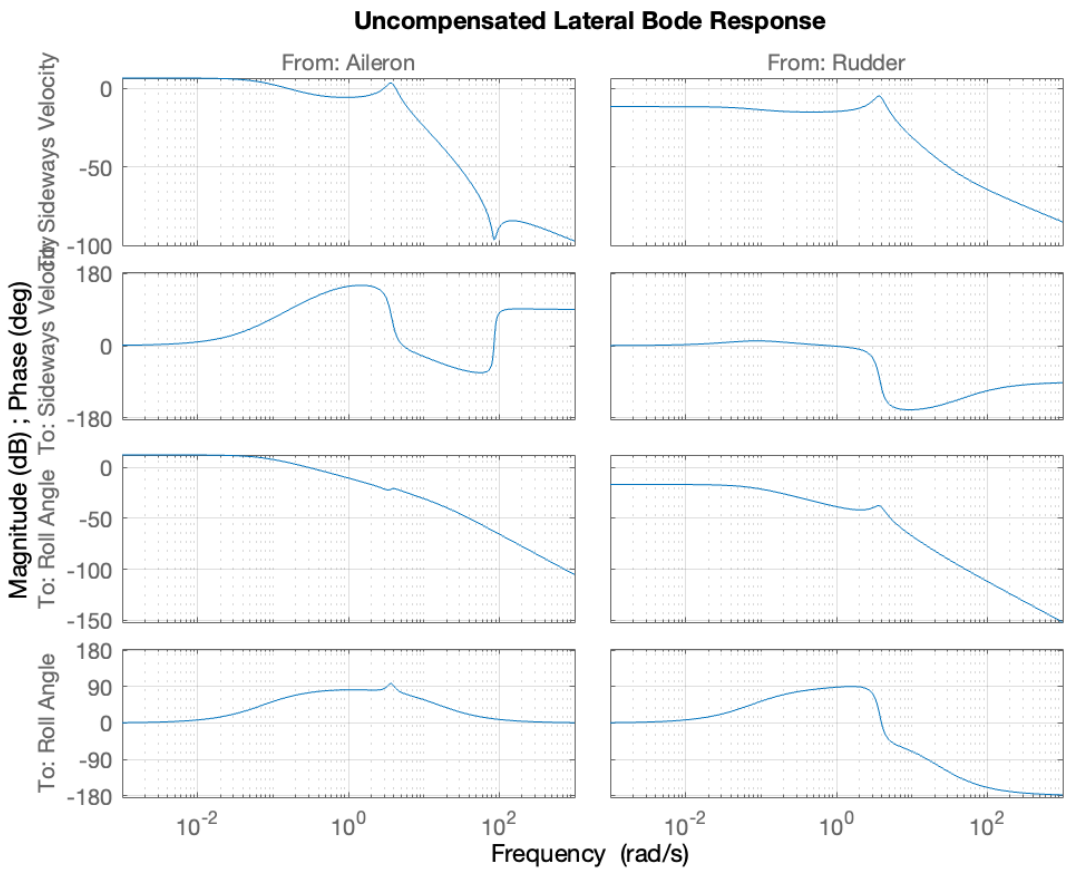

Figure 19: The uncompensated lateral Bode diagrams.

Stable Lateral Poles

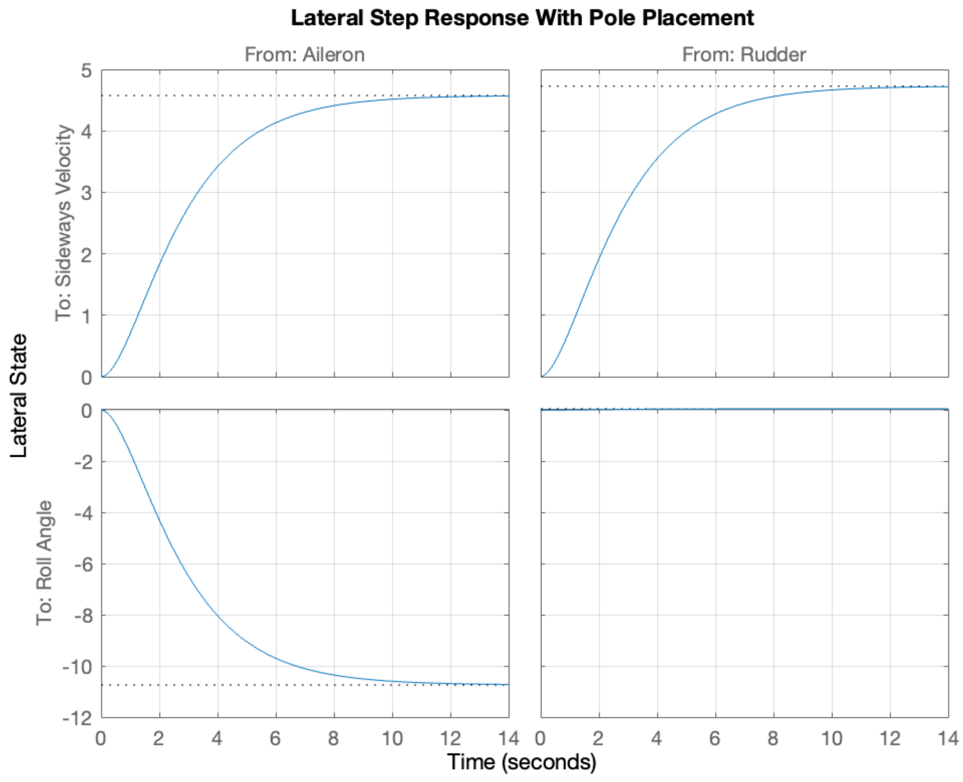

Figure 20: The lateral step response with stable poles with information below.

Stable Poles Step Information		
Overshoot (Percent)		
-	From: Aileron	From Rudder
To: Sideways Velocity	0	0
To: Roll Angle	0	0
Rise Time (s)		
-	From: Aileron	From Rudder
To: Sideways Velocity	5.1802	5.1793
To: Roll Angle	5.1802	5.1802
Settling Time (s)		
-	From: Aileron	From Rudder
To: Sideways Velocity	9.2063	9.1771
To: Roll Angle	9.2005	9.2005

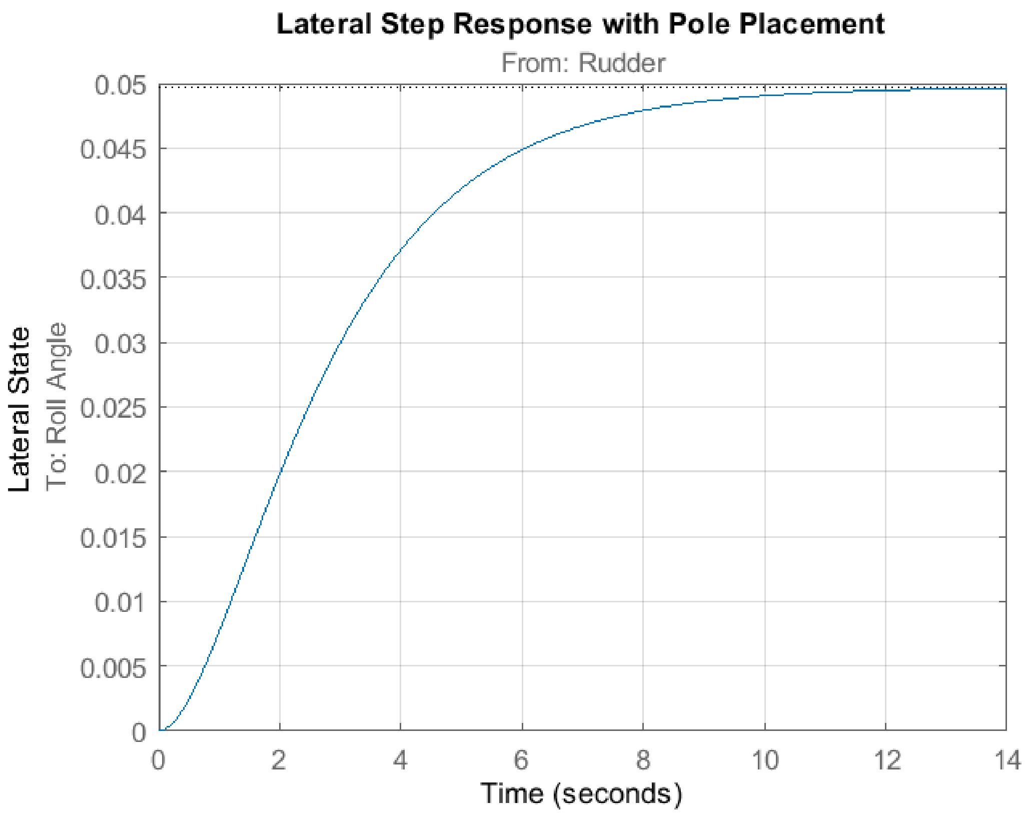

Figure 21: The lateral Bode diagrams with stable poles.

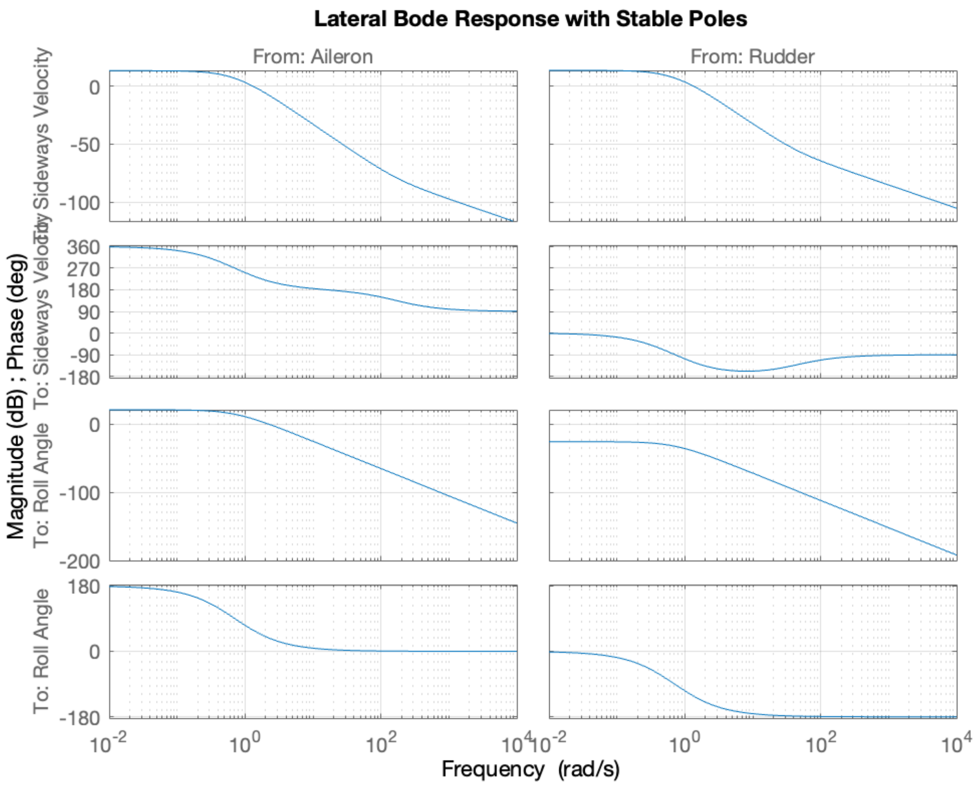

Figure 22: The lateral Bode diagrams with stable poles.

Introduce Reference Tracking (RT) for Lateral Direction

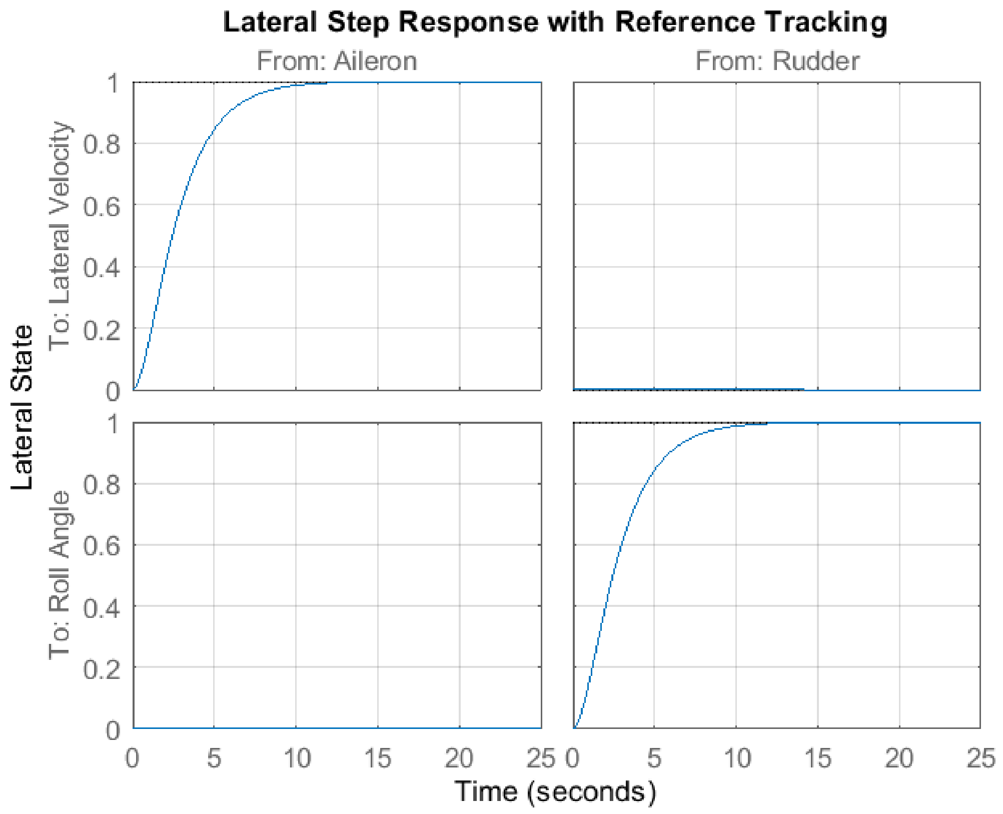

Figure 23: The lateral step response with reference tracking with information below.

Reference Tracking Step Information		
Overshoot (Percent)		
-	From: Aileron	From Rudder
To: Sideways Velocity	0	0
To: Roll Angle	.2861	0
Rise Time (s)		
-	From: Aileron	From Rudder
To: Sideways Velocity	5.1793	NaN
To: Roll Angle	5.6848	5.1820
Settling Time (s)		
-	From: Aileron	From Rudder
To: Sideways Velocity	9.1772	NaN
To: Roll Angle	19.6856	9.2005

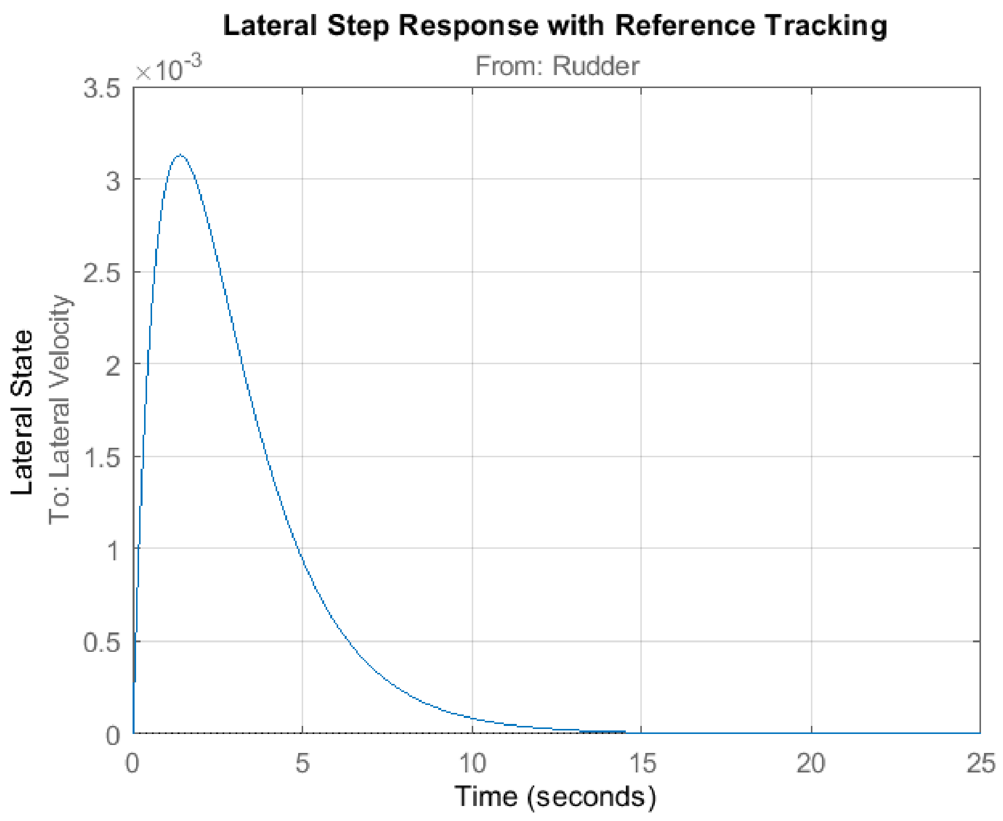

Figure 24: The lateral Bode diagrams with stable poles.

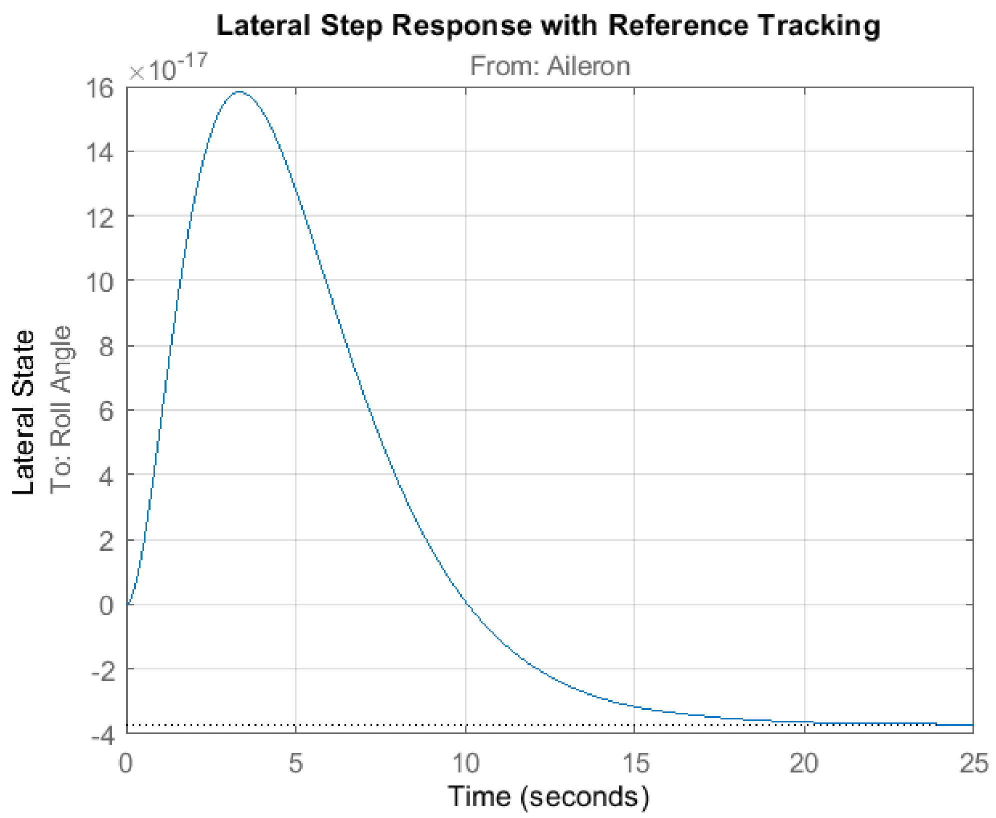

Figure 25: The lateral Bode diagrams with stable poles.

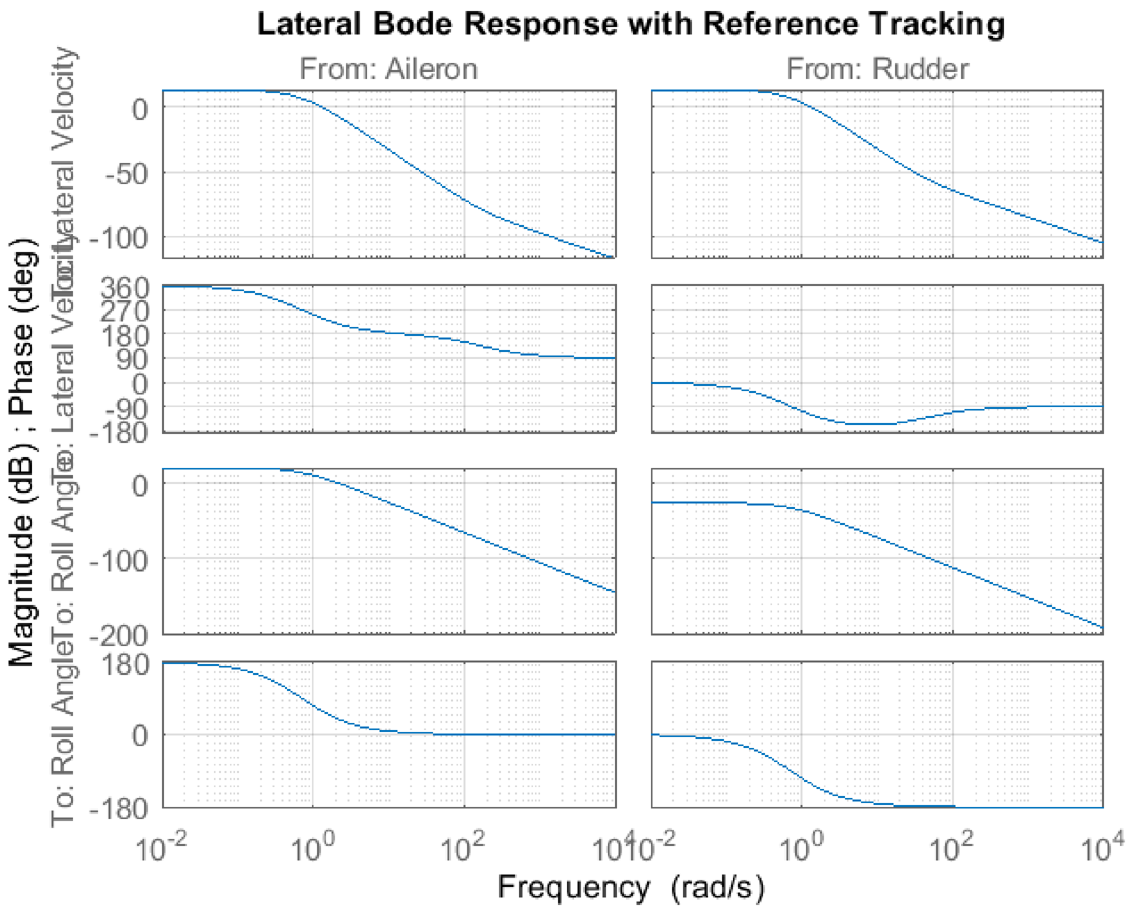

Figure 26: The lateral Bode diagrams with reference tracking.

Fast Poles with Reference Tracking (RT)

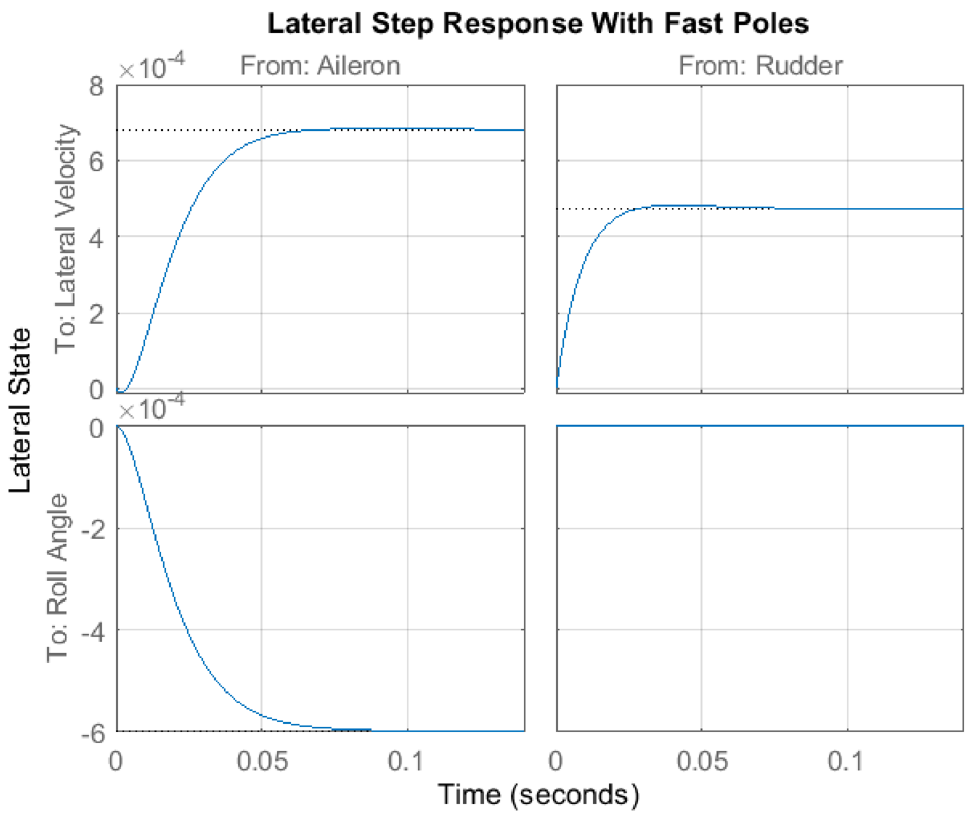

Figure 27: The lateral step response with reference tracking for fast poles with information below.

Lateral Fast Poles with RT Step Information		
Overshoot (Percent)		
-	From: Aileron	From Rudder
To: Sideways Velocity	0.2520	2.1864
To: Roll Angle	0	4.2266
Rise Time (s)		
-	From: Aileron	From Rudder
To: Sideways Velocity	0.0317	0.0159
To: Roll Angle	0.0356	0.0238
Settling Time (s)		
-	From: Aileron	From Rudder
To: Sideways Velocity	0.0546	0.0478
To: Roll Angle	0.0619	0.0832

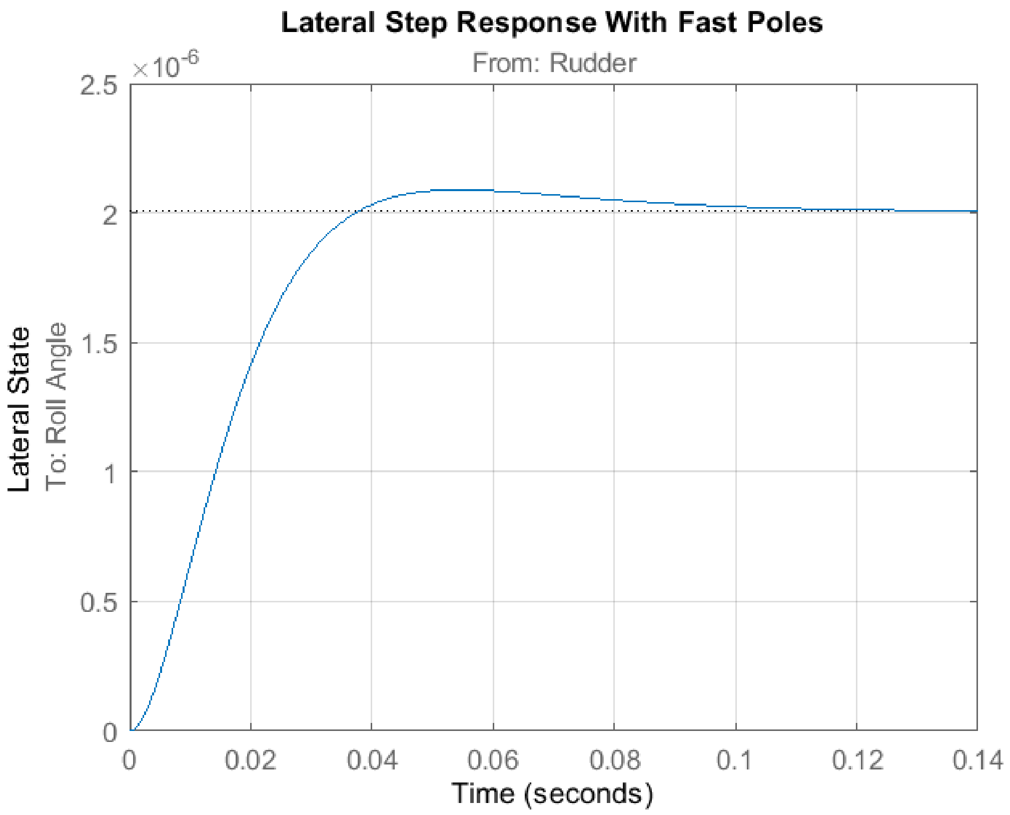

Figure 28: The lateral Bode diagrams with stable poles.

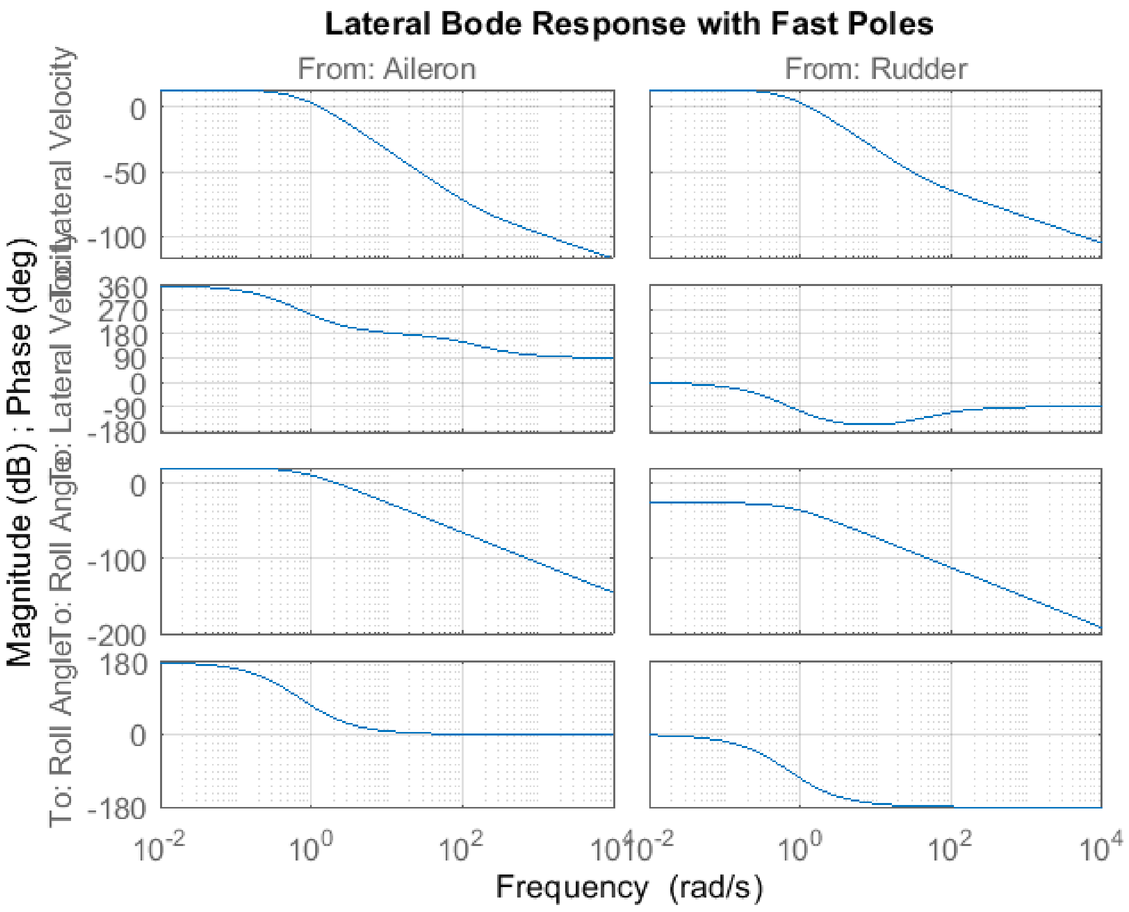

Figure 29: The lateral Bode diagrams with reference tracking for fast poles.

Lateral Imaginary Poles with Reference Tracking (RT)

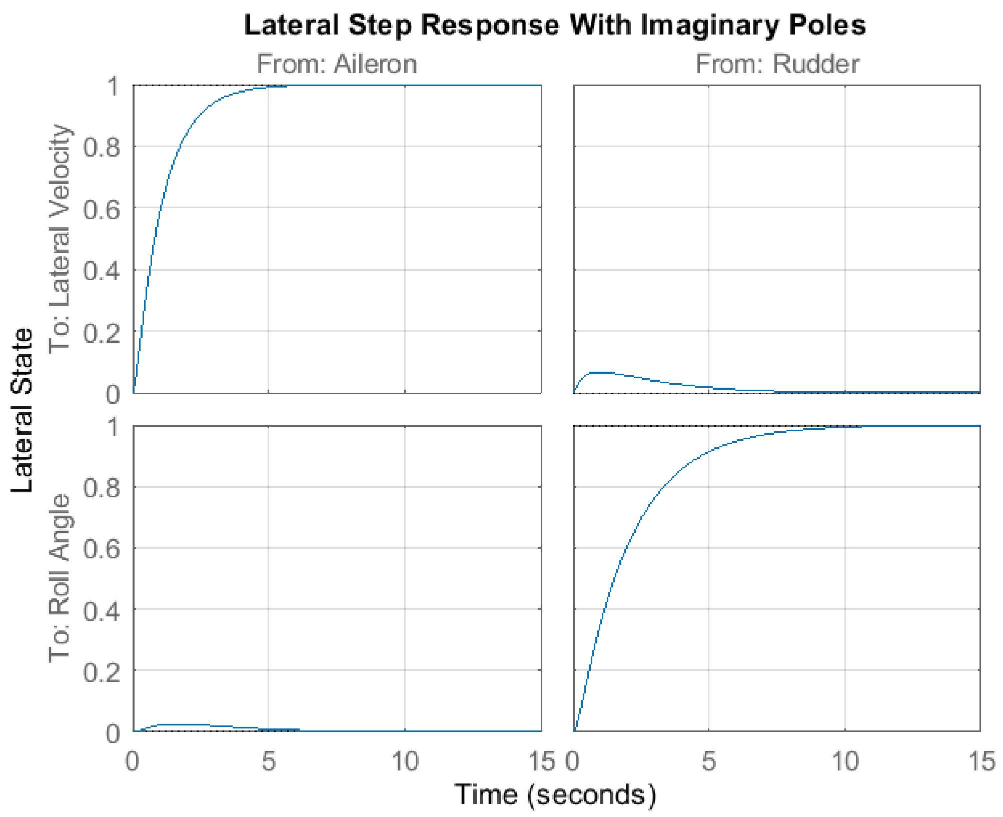

Figure 30: The lateral step response with reference tracking for imaginary poles with information below.

Lateral Imaginary Poles with RT Step Information		
Overshoot (Percent)		
-	From: Aileron	From Rudder
To: Sideways Velocity To: Roll Angle	0 1.1548x10^{17}	5.3409x10^{15} 0
Rise Time (s)		
-	From: Aileron	From Rudder
To: Sideways Velocity To: Roll Angle	2.2717 1.8920x10^{-14}	9.0743x10^{-15} 4.3836
Settling Time (s)		
-	From: Aileron	From Rudder
To: Sideways Velocity To: Roll Angle	4.1595 NaN	NaN 7.9486

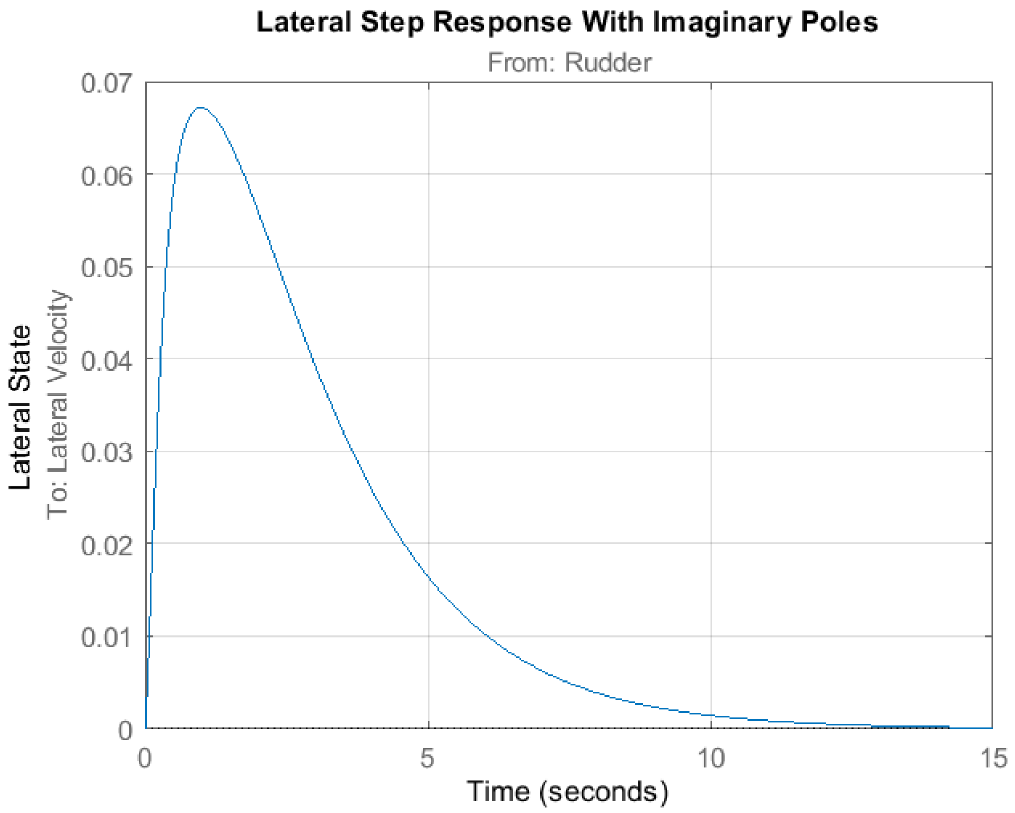

Figure 31: The lateral Bode diagrams with stable poles.

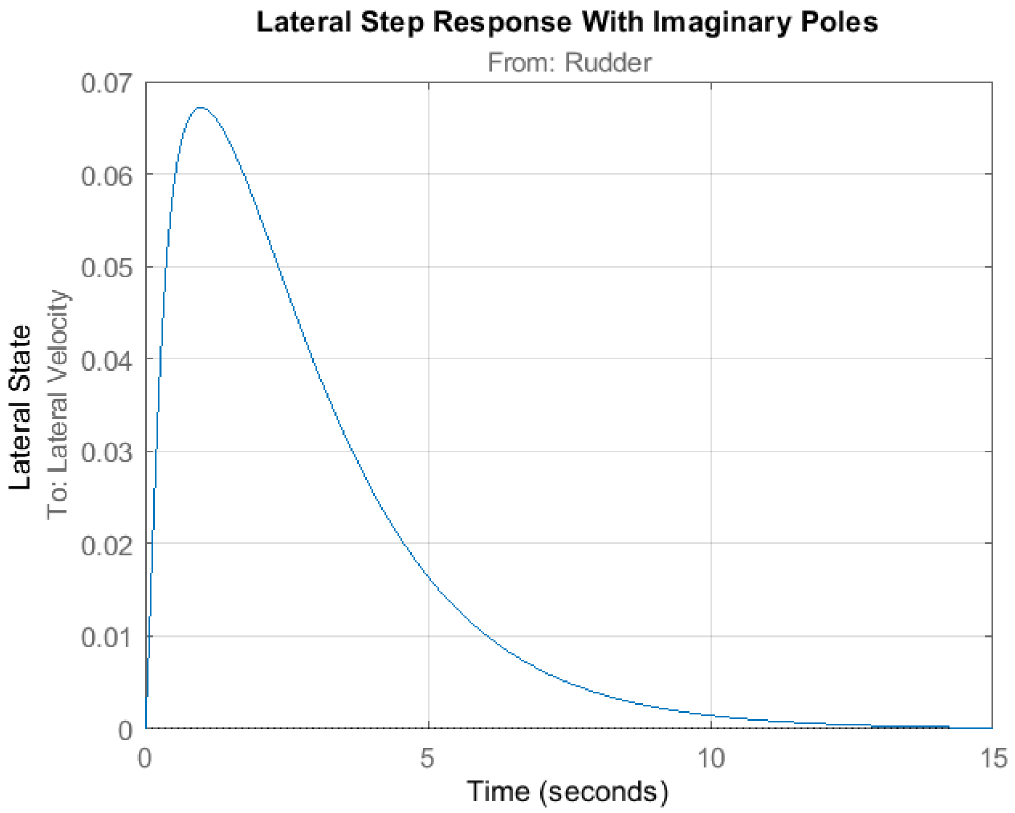

Figure 32: The lateral Bode diagrams with stable poles.

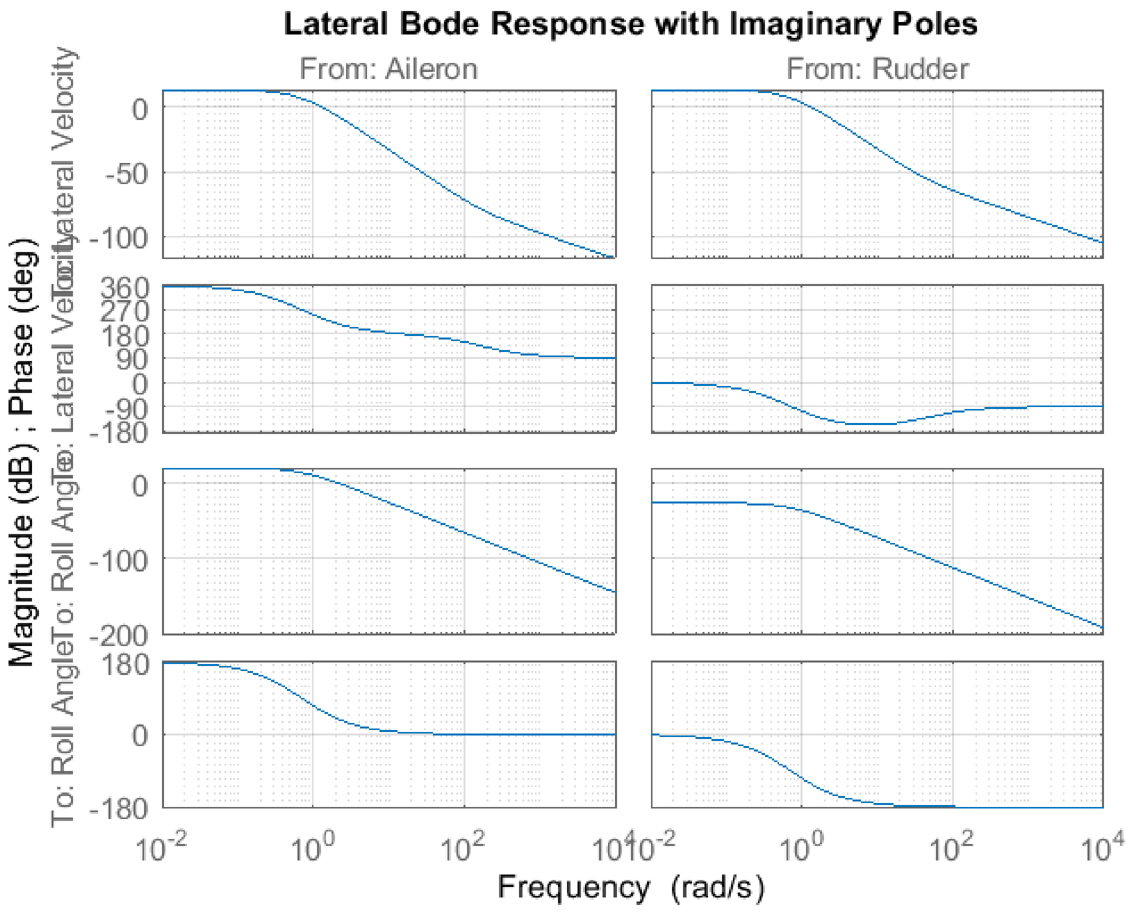

Figure 33: The lateral Bode diagrams with reference tracking for imaginary poles.

Reference Tracking Minimizing Dynamics, LQR pt. 1

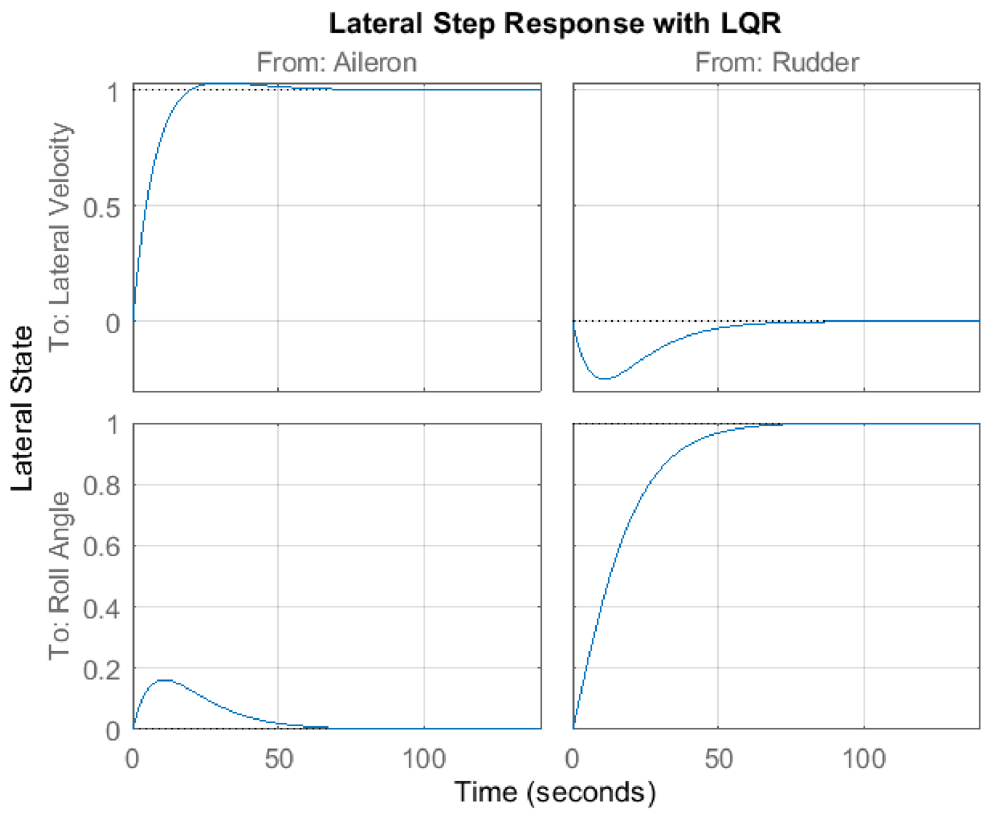

Figure 34: The lateral step response with reference tracking using LQR, pt. 1.

RT with LQR, pt. 1 Step Information		
Overshoot (Percent)		
-	From: Aileron	From Rudder
To: Sideways Velocity To: Roll Angle	3.2004 $2.6689\text{x}10^{16}$	0 0
Rise Time (s)		
-	From: Aileron	From Rudder
To: Sideways Velocity To: Roll Angle	12.4566 $1.3378\text{x}10^{-14}$	NaN 33.3106
Settling Time (s)		
-	From: Aileron	From Rudder
To: Sideways Velocity To: Roll Angle	44.6971 NaN	NaN 56.0613

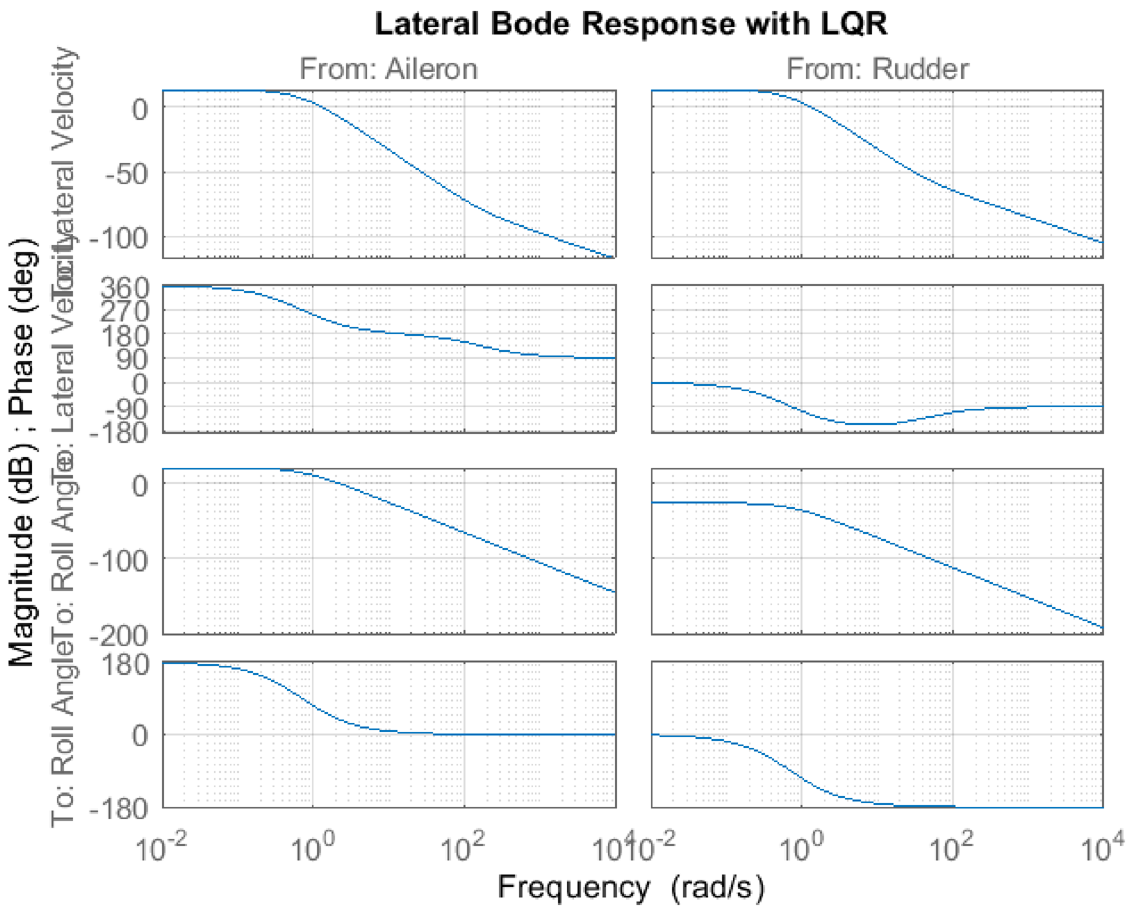

Figure 35: The lateral step response and Bode diagrams with reference tracking using LQR, pt. 1.

Reference Tracking using Chosen Costs, LQR pt. 2

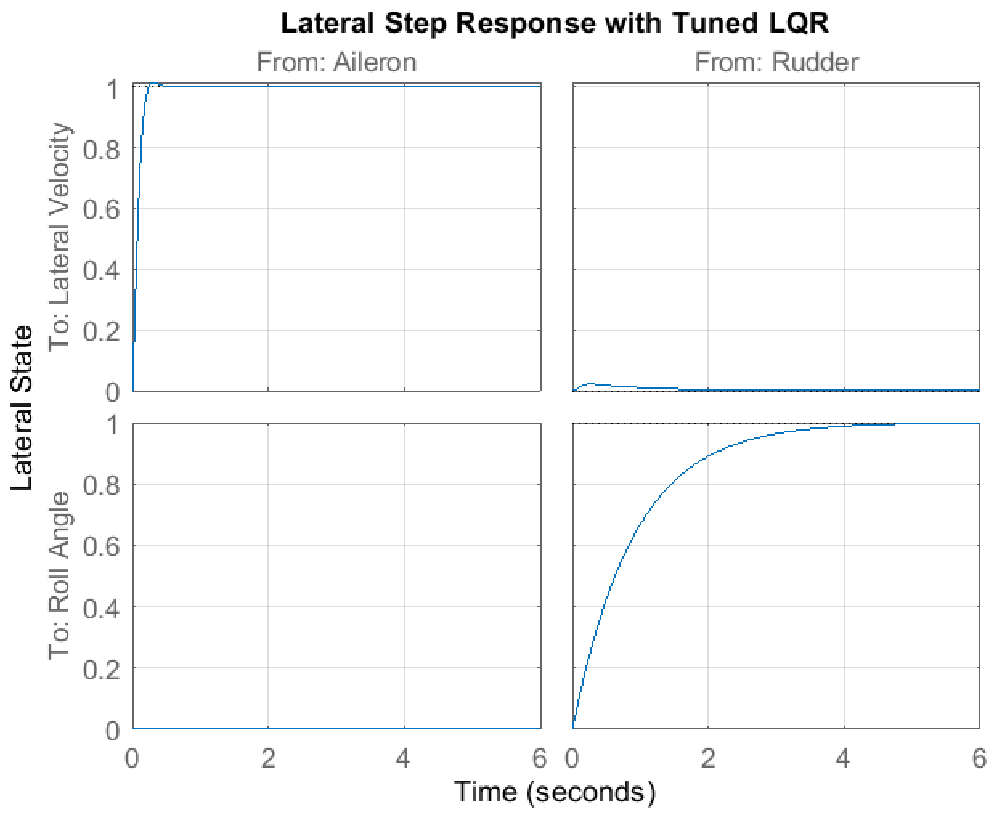

Figure 36: The lateral step response with reference tracking using LQR, pt. 2.

RT with LQR, pt. 2 Step Information		
Overshoot (Percent)		
-	From: Aileron	From Rudder
To: Sideways Velocity	1.4118	0
To: Roll Angle	$1.9696\text{x}10^{16}$	0
Rise Time (s)		
-	From: Aileron	From Rudder
To: Sideways Velocity	0.1483	NaN
To: Roll Angle	$1.1900\text{x}10^{-15}$	1.9839
Settling Time (s)		
-	From: Aileron	From Rudder
To: Sideways Velocity	0.2136	NaN
To: Roll Angle	NaN	3.5340

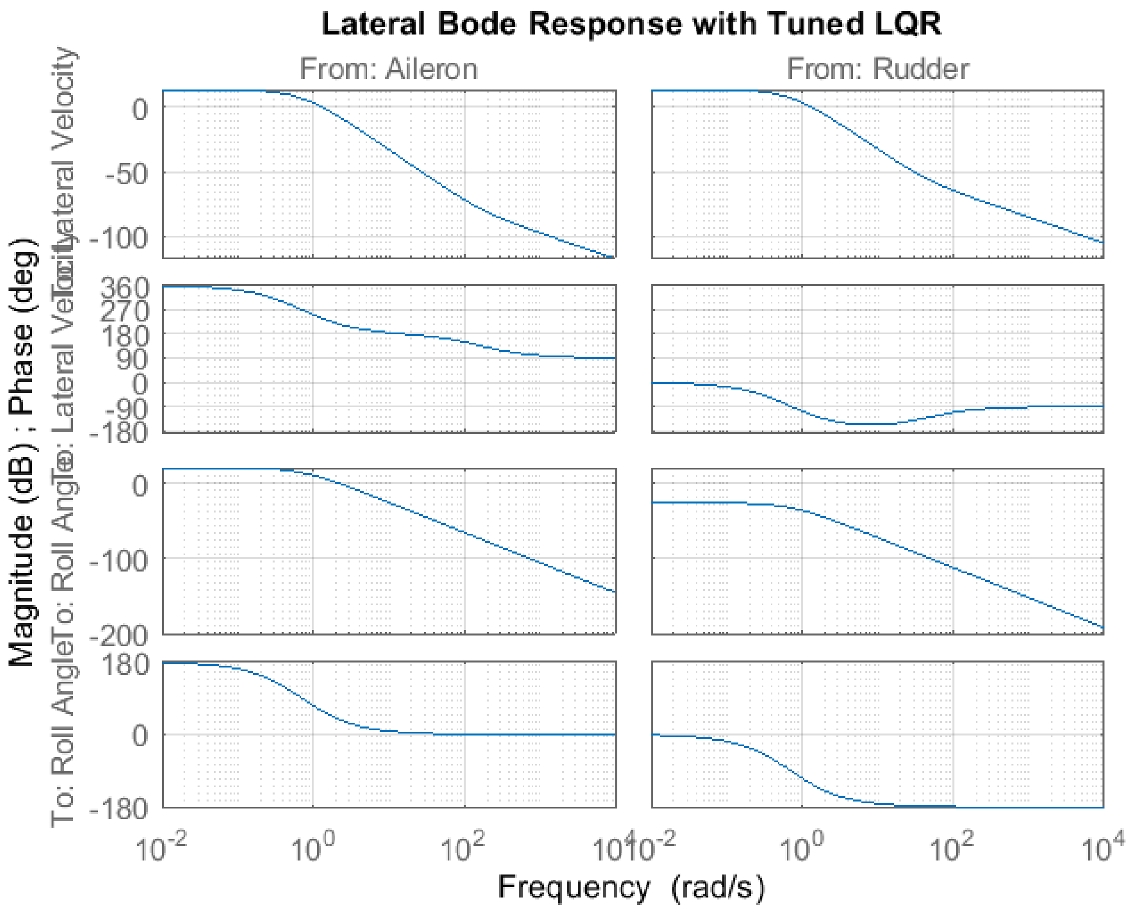

Figure 37: The lateral Bode diagrams with reference tracking using LQR, pt. 2.

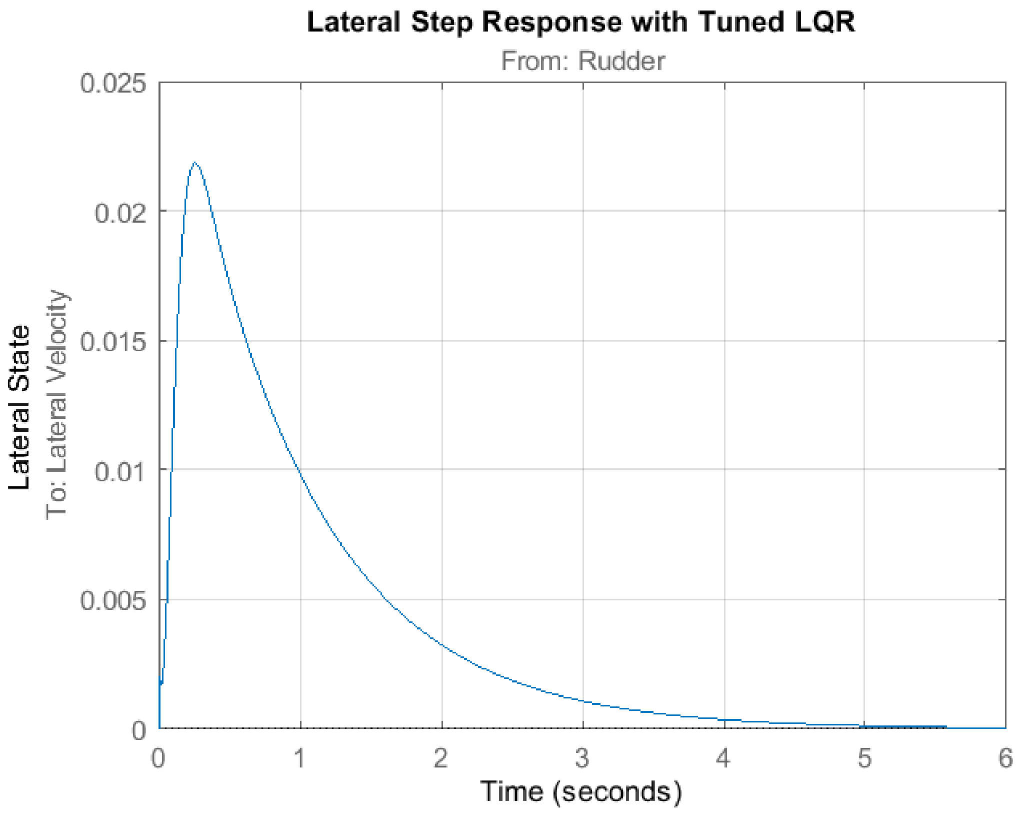

Figure 38: The lateral Bode diagrams with stable poles.

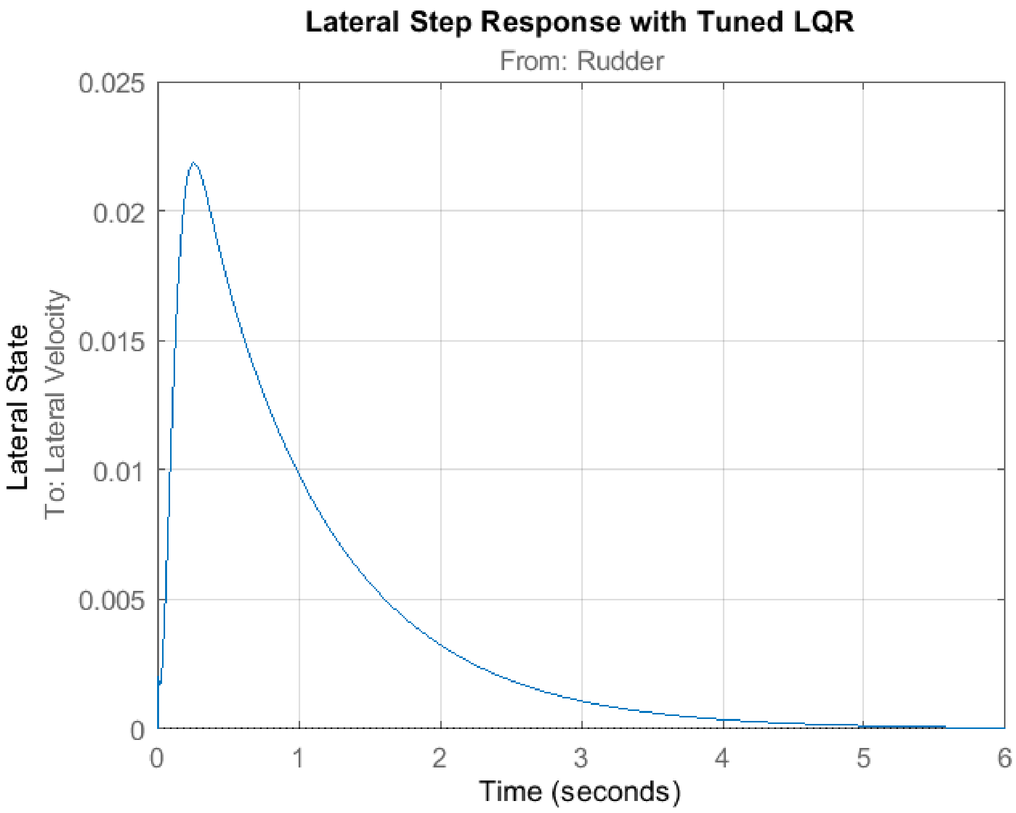

Figure 39: The lateral Bode diagrams with stable poles.

4 Discussion

In this section, I - Lily Cothren - discuss observations of our controllers and difficulties associated with the general project.

4.1 Individual Observation

The tabulated step response information is given in Section 3. These values were calculated using MATLAB's `stepinfo()`. In both lateral and longitudinal directions, we find that for certain controllers that the response from a control input yields the same step information for their respective outputs in two ways. On one hand, we find that either control input gives the same observed output (see the Longitudinal Reference Tracking with LQR, pt. 1 for an illustrative example). On the other, we can observe that either control input gives the same observed output (see the Longitudinal Reference Tracking with LQR, pt. 2 for an illustrative example). Unfortunately, we regret that we were not able to clearly identify why this occurs in our simulations. To the best of our knowledge, we posit this may be due to poorly fitted data for our simulation goals, oversimplification in the physical modeling of the problem, only partially observing certain state values, and restricting ourselves to linear controllers. Because this system (6 dof is a great deal of nonlinearity!) is so complex, we don't think that the matching responses are physically realistic. If they were reasonable, we would claim that the system could be further decoupled and employ frequency-domain techniques (like the root-locus, Bode plots, and Nyquist plots) to design more descriptive and interesting controllers. However, we hesitate on decoupling the system further than into longitudinal and lateral directions because this division was already a large assumption [3].

Given that frequency-based methods that we learned in this course were not at our disposal for our MIMO system, we consulted alternative sources for state-space methods [2], [10], [8]. Unfortunately, the literature presented either provides existence results (as in, there exists a feedback controller to perform x, y, z) or discusses specific nuances for their particular problem, which does not directly translate to our setting. Accordingly, we implemented a variety of gain matrices to view step responses (including an attempt to select an "optimal" K via LQR minimization). Although each controller damped oscillations, over or undershoot was still present in the face of "reasonable" rise times and settling times for at least some of the step responses. As such, the choice of a "best" controller is undetermined because we need to better quantify what type of overshoot or undershoot - if any - is allowable. For example, in the longitudinal direction, the first set of cost functions pertaining to the minimized dynamics shows that the pitch angle is damped (which was our goal), but that the forward velocity overshoots slightly. Is this allowable in a physical system? Moreover, is this controller actually implementable, or are the gain values in K infeasible for actuators?

In future research, many directions are available. First, we are interested in performing our own data generation of environment conditions to use in Simulink as demonstrated in [8]. Second, we are also interested in implementing these studies in a manner that provides more available checks on physical expectations (say, using Vorlax to check that the ailerons only change ± 30 degrees). We may also be interested in a full state feedback controller (instead of matrix C only observing the first and last states, set $C = \mathbb{I}_{4\times 4}$) or scripting observability goals to simulate full state feedback. Finally, it may be fruitful to pursue more advanced controller techniques like gain scheduling [7, Chapter 12-13]. Briefly, gain scheduling is often used in aerospace settings because the mass of the object changes over the trajectory of the flight since fuel is consumed. Gain scheduling is a type of proportional control in which the proportional gain K is a function of some hyper-parameter, α; this enables the control engineers to set different $K(\alpha) \equiv K$ values for different ranges of α to better control the trajectory of the system. We regret that we did not have time for these directions, but hope that future works consider these options with respect to this problem.

4.2 Difficulties

This project utilized myriad course concepts applied in a difficult yet transferable manner. Neither of us were exposed to modeling real systems to this degree (moreover, this was my first engineering class that required any skills for modeling physical systems), which posed some troubles involving the nuances of matching expectations of simulation responses with reality, including what it means to "trim a flight." Even though we ultimately had to use data from someone else's trade study [8], we were careful to check the process for modeling (namely, applying Newton's laws and rounding out the 6 dof equations with another 6 equations for kinetic movement) and eventual linearization about trim conditions. As careful as we were in studying how [8] obtained her model parameters, we find that for the goals of this project, the results of the simulations did not align with reality. In particular, we found it physically unreasonable that in both the lateral and longitudinal controllers that the respective control inputs posed the same step response for our measured output states (see Figures).

Aside, the pole placement problem we posed in this project is tedious to solve if we must iterate through different gain matrix possibilities for K to reduce overshoot in the step responses. Further, typical frequency-based control techniques involving root locuses, Bode plots, and Nyquist plots are difficult (and to our knowledge, impossible) to implement effectively in multi-input, multi-output systems like ours which cannot be decoupled beyond the split into lateral and longitudinal directions. So, even though we displayed the Bode diagrams for each controller, we could not gain intuition from these plots to write controllers. Accordingly, we constricted ourselves to (linear) controllers within the state-space domain to select control inputs. To best select the values for the controller gain matrix K, we considered different pole placements and turned to Optimal Control topics for inspiration.

4.3 Conclusions

Overall, I learned many practical (and harsh) lessons through this project. Much of my research is theoretical in nature, so I was comfortable with the more "math-y" components of this project regarding linearization about trim conditions, why LQR is an interesting and effective tool to choose a gain value, and the brief consideration of gain scheduling. However, this comfort was short-lived. Most of this project was computationally heavy for a real problem, which forced me to learn how one models a complex physical system (it's the same process that we reviewed at the start of this semester, but yields many more equations of motion), how to numerically simulate this system in MATLAB, how to check the frequency domain responses (this topic was also brand-new for me this semester), and how to verify reality with simulations. These latter endeavors were entirely new for me since my formal training is in (pure) mathematics. However straining these were, though, I am grateful to apply these tools in this project because I ultimately want to pursue a career in *engineering.*

In the future, although a majority of my research will be focused on algorithmic development and theoretical verifications, I will employ the skills learned here - coding and checking for physical reasonableness - to supplement future works. Specifically, when I submit publications, I will be better prepared for providing numerical simulations at the end of my papers to verify for the editors the usefulness of my algorithms.

gensymb

5 Discussion

In this section, I - Bethany Calvert - discuss observations of our controllers and difficulties associated with the general project.

5.1 Individual Observation

From the start of the lateral analysis, the data yields difficult and unsatisfactory results. The field study [4] that we pulled data from initializes the system with complex dynamics (see Sec. 1.1). UAVs tend to have large issues with damping perturbations from trim especially with the yaw and roll angles. As analysis continued to be conducted throughout the lateral mode, the system dynamics often yield unruly results. The lateral dynamics are described by a state space MIMO system that analyzes four states. The C matrix asserts that the only state outputs of the system that should be identified are the lateral velocity and roll angle. The original A_{lat} matrix yields the open loop dynamics. The off-diagonal values of this matrix are too large to decouple the system states and analyze each as a SISO system. However, to fulfill the requirement of frequency domain response discussion, the system states are each plotted according to each control input.

The lateral system started unstable with one RHP pole, but the system proved to be controllable, so the analysis moved forward. Checking Fig. 18, the step response does not converge. Additionally, Fig. 19 does not indicate meaningful phase or gain margin values. Next, once all poles are moved and placed into the LHP, the system response is greatly improved. The transient properties of the system are displayed below Fig. 20. The response of the roll angle rudder control is displayed in Fig. 21 on a different axis scaling. The steady state error of the system needed improvement, so a reference tracking loop was implemented.

Upon implementation of the reference tracking loop, the system started to reflect major issues in some of the step responses. Figure 23 displays that only the lateral velocity aileron control and the roll angle rudder control track to a final value of one to exhibit zero steady state error. Figures 24-25 display how the lateral velocity rudder control and the roll angle aileron control both track to a final value of zero. Physically, this response is asserting that the rudder has no affect on lateral velocity, and the aileron has no affect on roll angle. This is an incorrect assertion, and it likely comes from issues in modeling the original dynamics. The bode response of the system is improved, but still does not cross the -180° line on the phase plot for the graphs. Proving, that even with state feedback tracking, the frequency response does not offer great insight on stability of the MIMO system.

Next, different pole values are iterated to accomplish adequate step responses for fast and imaginary poles in Fig.27-33. The fast poles all end at a steady state value of zero. These responses come from extremely high control gains. The poles are working so fast, that in real life they would not actually be able to have a real affect on the system. The roll angle rudder control response is displayed on a different axis scale in Fig.28.

The imaginary poles increase the system over shoot and oscillatory response. When the poles are moved off of the real axis into the imaginary plane, the damping ratio decreases to a value between zero and one. The effect of the damping ratio change increases the system overshoot and settling time. Analyzing the values in the table below Fig.30, the overshoot is increased to an extremely large value while the rise time is decreased to an extremely low value for the lateral velocity aileron control and the roll angle rudder control. These values track to a final value of zero, similarly to the reference tracking pole placement example, and a modified axis response for each output state is displayed in Fig. 31-32. Physically, these response times and overshoot values do not make sense. Even after iterating through many different pole values and combinations, the rise time and overshoot values were not able to be tuned to more reasonable values.

The LQR control lent to a much easier process, because as the designer we were able to assert cost parameters via the cost function to choose the optimal gain matrix. However, similarly to how the imaginary poles affected the system, the original LQR control has an issue with an extremely large overshoot and extremely low rise time. The problem states for the second LQR tuner are displayed on new axes scales in Fig. 38-39. For the other states, the LQR controllers work extremely effectively. The overshoot is kept low while the rise time is fast with a reasonable settling time. These parameters displayed under Fig. 34-36 are slightly improved in the tuning of the LQR controller.

5.2 Difficulties

Aircraft systems are extremely complex MIMO systems. The modeling that takes place to describe the aerodynamic interactions of the aircraft with its environment is very difficult to conduct. With a difficult modeling task on our hands, it only made sense to pull a plant function to conduct analysis. For future research, it would be more conducive to control designers to be able to play a hand in system modeling. Overall, this project mainly took place in the state space domain because it is a much more effective and intuitive way to model MIMO systems. However, the lack of SISO system tools like root locus and bode response made it extremely difficult to gather clear instinct on where poles should be placed for feedback control.

Overall, when problems were introduced it was typically due to issues with the lateral velocity rudder control or the roll angle aileron control. As stated throughout, we believe this problem originates from a combination of modeling errors and assumptions as well as system complexity. Additionally, another issue may be the application of the step input. Future research may take liberties in seeing how the system applies to alternative inputs.

Additionally, huge issues came into play when introducing reference tracking. There is a surprising lack of literature/lectures on state space MIMO strategies for reference tracking. While the class touched base on these state space strategies, the examples were rather simple in comparison to an aircraft system. An alternative strategy of reference tracking using state feedback tracking with integral action was explored. However, ultimately we were unable to trouble shoot this control strategy due to the size of our system. In future project iterations, it would be great to explore tracking via an integrator instead, to see how the system response is affected. Aircraft systems are always in the presence of wind disturbance during flight. Another avenue we would have liked to explore if there was more time is disturbance control. While we built the basic building blocks with reference tracking, trouble shooting the issues within our system prevented us from being able to explore the topic of disturbance control.

5.3 Discussion

In conclusion, this project served as a great avenue to gain physical understanding of control processes. While parts of the lateral dynamics were not ideal, the lateral velocity aileron control and roll angle rudder control exhibited great results from both the state feedback and LQR control. The system was improved from being unstable to providing adequate responses with zero steady state error and small overshoot and rise time values. Often the settling time values tend to be much longer in comparison to the rise time values. This makes sense because aircraft are naturally oscillatory systems, and the oscillations take longer in nature to damp out. Additionally, the quick rise time is a result of turbulent air conditions and fast response time of aircraft to environmental perturbations.

Although a lot of the analysis was hard to interpret, most real world systems exhibit the complex nonlinearity just as this aircraft system does. Applying state space and frequency domain concepts impressed a larger appreciation of SISO and frequency domain tools that give controller design intuition. My future research will center around control of UAV systems modeling, and this served as a useful introduction into aircraft system analysis.

References

[1] 6dof documentation. https://www.mathworks.com/help/aeroblks/6dofeulerangles.html. Accessed: 2022-04-18.

[2] Modeling, simulation, and flight control design of an aircraft with simulink. https://www.mathworks.com/videos/modeling-simulation-and-flight-control-design-of-an-aircraft-with-simulink-81546.html. Accessed: 2022-02-09.

[3] N. Ahmed, ASEN 3128: Aircraft Dynamics, Spring 2022.

[4] G. Brown and A. Roshko, On density effects and large structure in turbulent mixing layers, Journal of Fluid Mechanics, 64 (1974), pp. 775–816.

[5] D. Caughey, Introduction to aircraft stability and control course notes for MAE 5070, 2011.

[6] M. Delisle and T. T. Takahashi, We wonder why we wallow: Impacts of trim error on speed and flight path stability, in AIAA Scitech 2019 Forum, 2019, p. 0559.

[7] H. Khalil, Nonlinear Systems; 3rd ed., Prentice-Hall, Upper Saddle River, NJ, 2002.

[8] A. Maio, Acceleration feedback control for small fixed-wing uas, Theory of Computing Systems Mathematical Systems Theory, (2018), p. 16.

[9] A. Maio, B. Ranganathan, J. S. Humbert, G. Gremillion, and W. Nothwang, Acceleration feedback control for fixed-wing suas, 2018 IEEE Aerospace Conference, (2018), pp. 1–11.

[10] N. Schindeler, Phugoid Damping Control, PhD thesis, Airforce Institute of Technology, 2001.

6 Appendix

6.1 Longitudinal Code

```
clc, clear all, close all
% Longitudinal states are: forward velocity (u), vertical velocity (w),
% pitch angle (theta), and pitch rate (q).
%
% Lateral states are: sideways velocity (v), roll rate(p), roll angle
% (phi), yaw rate (r).

%% Airplane Parameters
span = 3.067 ; %m
area = .6282 ; %m^2
chord = .208 ; %m
mass = 5.74 ; %kg
g = 9.81 ; %m/s^2
Ixx = 1.2009 ; %kg m^2
Iyy = .9318 ; %kg m^2
Izz = 2.0734 ; %kg m^2
Ixz = .0946 ; %kg m^2

%% Trim State
h = 1800 ; %m

%Longitudinal
Va = 18 ; %m/s
theta = .0515 ; %rad
u = 17.9762 ; %m/s
w = .9263 ; %m/s
q = 0 ;
Δ_e = -.537 ; %rad
Δ_t = 17.92 ; %percentage of max
du = 0 ;
```

```
dw = 0 ;
dq = 0 ;

%Lateral
Δ_a = 0 ; %rad
Δ_r = 0 ; %percentage of max
dv = 0 ;
dp = 0 ;
dr = 0 ;
v = 0 ; %m/s
p = 0 ;
r = 0 ;
phi = 0 ;
B = 0 ;

% trim conditions for lon states:
trim_lon_x = [17.9762 / 18; 0.9263; 0; 0.0515];
trim_lon_y = [17.9762 / 18; 0.0515];

%% Linear Model Parameters

%Force Parameters
Xu = -.1271 ; %s^-1
Xw = .6409 ; %s^-1
Xq = -.9106 ; %m/s
Yv = -.3714 ; %s^-1
Yp = .8254 ; %m/s
Yr = -17.6451 ; %m/s
Zu = -.07655 ; %s^-1
Zw = -6.3237 ; %s^-1
Zq = 16.9091 ; %m/s
X_Δ_e = .0018 ; %m/s^2
X_Δ_t = 3.3846 ; %m/s^2
Y_Δ_a = -.0137 ; %m/s^2
Y_Δ_r = .0556 ; %m/s^2
Z_Δ_e = -.1234 ; %m/s^2

%Moment Parameters
Lv = -1.1467 ; %(m*s)^-1
Lp = -15.7093 ; %s^-1
Lr = 2.6774 ; %s^-1
Mu = .1090 ; %(m*s)^-1
Mw = -2.1148 ; %m*s)^-1
Mq = -3.2853 ; %s^-1
Nv = .6400 ; %(m*s)^-1
Np = -1.2356 ; %s^-1
Nr = -.5669 ; %s^-1
L_Δ_a = -5.3580 ; %s^-2
L_Δ_r = .0316 ; %s^-2
M_Δ_e = -1.3996 ; %s^-2
N_Δ_a = -.2566 ; %s^-2
N_Δ_r = -.1309 ; %s^-2

%% State Space Matrices

A_lon = [Xu,     Xw/Va,   Xq/Va,  -g*cos(theta)/Va;
         Zu*Va,   Zw,       Zq,      -g*sin(theta);
         Mu*Va,   Mw,       Mq,             0;
         0,        0,        1,             0]   ;
```

```
A_lat = [Yv, Yp,   Yr,    g*cos(theta);
         Lv, Lp,   Lr,        0;
         Nv, Np,   Nr,        0;
         0,  1,   tan(theta), 0]  ;
B_lon = [X_Δ_e/Va, X_Δ_t/Va;
         Z_Δ_e,         0      ;
         M_Δ_e,         0      ;
            0,              0] ;
B_lat = [Y_Δ_a, Y_Δ_r;
         L_Δ_a, L_Δ_r;
         N_Δ_a, N_Δ_r;
             0,         0] ;
C_lon = [1, 0, 0, 0;
         0, 0, 0, 1] ;
C_lat = [1, 0, 0, 0;
         0, 0, 0, 1] ;
u_lon = [Δ_e; Δ_t] ;
u_lat = [Δ_a; Δ_r] ;
x_lon = [u/Va, w, q, theta]' ;
x_lat = [v, p, r, phi]' ;
ylon = [u/Va; theta] ;
ylat = [v; phi] ;
D = zeros(2,2) ;

states_lon = {'Forward Velocity' 'Vertical Velocity' 'Pitch Angular Velocity' ...
    'Pitch Angle'};
inputs_lon = {'Elevator' 'Throttle'};
outputs_lon = {'Forward Velocity' 'Pitch'};
states_lat = {'Sideways Velocity' 'Roll Angular Velocity' 'Yaw Angular ...
    Velocity' 'Roll Angle'};
inputs_lat = {'Aileron' 'Rudder'};
outputs_lat = {'Sideways Velocity' 'Roll Angle'};
%% State Space Control

longitudinal = ss(A_lon, B_lon, C_lon, D, 'statename', states_lon, 'inputname', ...
    inputs_lon, 'outputname', outputs_lon) ;

poles_lon = eig(A_lon) ;

figure
step(longitudinal)
title('Uncompensated Longitudinal Step Response')
ylabel('Longitudinal State')
grid on

figure
bode(longitudinal)
title('Uncompensated Longitudinal Bode Response')
grid on

%% Proportional Control

% At this point we have a stable system for the longitudinal analysis.
% However, the poles are close enough to the origin that we'll move them anyway.

%Controllability check
Co_lon = ctrb(A_lon,B_lon) ;
rank(Co_lon) ;
%Controllability matrix is full rank, and therefore is controllable. Now
```

```
%pick pole values st the lateral system dynamics are stable
poles_lon = eig(A_lon); k = 10;
corrected_poles_lon = [-1*k, -1*k, -.5*k, -.5*k] ;
[Kp_lon,prec,message] = place(A_lon,B_lon,corrected_poles_lon) %Kp_lon is ...
    proportional controller, prec is measurement of how closely eig(A-BKp) match ...
    specified pole location
A_lon_new = A_lon-B_lon*Kp_lon ;
sys_place = ss(A_lon_new,B_lon,C_lon,D,'statename', states_lon, 'inputname', ...
    inputs_lon, 'outputname', outputs_lon) ;

figure
step(sys_place)
grid on
title('Longitudinal Step Response')
ylabel('Longitudinal State')

figure
bode(sys_place)
title('Shifted Poles: Longitudinal Bode Response')
grid on

%% Feed Forward/Tracking Control
%We need to implement a control term s.t. the output can track the loop
%error from the input. This control should be s.t. u = kx+rF. The F term
%will allow y (output) to track r (input)


Z = [zeros(size(A_lon,1), size(B_lon,2)); eye(size(C_lon,1),size(C_lon,1))] ;
N = inv([A_lon, B_lon; C_lon, D])*Z ;
Nx = [N(1:size(A_lon,1),1:size(B_lon,2))] ;
Nu = [N((size(A_lon,1)+1):(size(A_lon,1)+size(C_lon,1)),1:size(B_lon,2))] ;
N_bar = Nu+Kp_lon*Nx ;


B_lon_new = B_lon*N_bar ;


sys_track = ss(A_lon_new,B_lon_new,C_lon,D,'statename', states_lon, ...
    'inputname', inputs_lon, 'outputname', outputs_lon) ;
poles_lat_track = eig(A_lon-B_lon*Kp_lon)

figure
step(sys_track)
grid on
title('Longitudinal Step Response with Reference Tracking')
ylabel('Longitudinal State')

figure
bode(sys_track)
title('Longitudinal Bode Response with Tracking')
grid on

%% Tuned Proportional Control Fast Response
fast_poles_lon = [-100 -100 -50 -50] ;
[Kp_lon_fast,prec,message] = place(A_lon,B_lon,fast_poles_lon)

N_bar_fast = Nu+Kp_lon_fast*Nx ;

A_lon_fast = A_lon-B_lon*Kp_lon_fast ;
```

```
B_lon_fast = B_lon*N_bar_fast ;

sys_lon_fast = ss(A_lon_fast,B_lon_fast,C_lon,D,'statename', states_lon, ...
    'inputname', inputs_lon, 'outputname', outputs_lon) ;
poles_lon_fast = eig(A_lon-B_lon*Kp_lon_fast)

figure
step(sys_lon_fast)
grid on
title('Lateral Step Response With Fast Poles')
ylabel('Lateral State')

figure
bode(sys_lon_fast)
title('Longitudinal Bode Response with Tracking')
grid on

%% Tuned Proportional Control Imaginary Poles
im_poles_lon = [-.5+j -.5-j -.950 -.950] ;
[Kp_lon_im,prec,message] = place(A_lon,B_lon,im_poles_lon)

N_bar_im = Nu+Kp_lon_im*Nx ;
A_lon_im = A_lon-B_lat*Kp_lon_im ;
B_lon_im = B_lon*N_bar_im ;

sys_lon_im = ss(A_lon_im,B_lon_im,C_lon,D,'statename', states_lon, 'inputname', ...
    inputs_lon, 'outputname', outputs_lon) ;
poles_lon_im = eig(A_lon-B_lon*Kp_lon_im)

figure
step(sys_lon_im)
grid on
title('Lateral Step Response With Imaginary Poles')
ylabel('Lateral State')

figure
bode(sys_lon_im)
title('Longitudinal Bode Response with Tracking')
grid on

%% LQR Control - Changing Dynamics
% Prior control uses trial and error to tune gain. Now, let's use linear
% quadratic control to find optimal gain matrix to minimize \|\dot x \|^2.
%
Q = A_lon_new'*A_lon_new
R = B_lon'*B_lon
N = 0.001*A_lon_new'*B_lon
[Kp_lqr, ARE, poles_lqr] = lqr(sys_place,Q,R,N)
%
N_bar_lqr = Nu+Kp_lqr*Nx ;
A_lon_lqr = A_lon-B_lon*Kp_lqr ;
B_lon_lqr = B_lon*N_bar_lqr ;
%
sys_lon_lqr = ss(A_lon_lqr,B_lon_lqr,C_lat,D,'statename', states_lon, ...
    'inputname', inputs_lon, 'outputname', outputs_lon) ;
poles_lon_lqr = eig(A_lon_lqr)
%
figure
step(sys_lon_lqr)
```

```
grid on
title('Longitudinal Step Response')
ylabel('Longitudinal State')

figure
bode(sys_lon_lqr)
title('Longitudinal Bode Response with Tracking using LQR')
grid on
%% LQR Tuner - choosing costs
%
Q = 200*eye(4) ; %The higher Q and lower R, better OS & rt
R = .02*eye(2) ;
N = 0 ;

[Kp_lqr_new, ARE_new, poles_lqr_new] = lqr(sys_place,Q,R,N)

N_bar_lqr_new = Nu+Kp_lqr_new*Nx ;
A_lon_lqr_new = A_lon-B_lon*Kp_lqr_new ;
B_lon_lqr_new = B_lon*N_bar_lqr_new ;

sys_lon_lqr_new = ss(A_lon_lqr_new,B_lon_lqr_new,C_lon,D,'statename', ...
    states_lon, 'inputname', inputs_lon, 'outputname', outputs_lon) ;
poles_lon_lqr_new = eig(A_lon_lqr_new)


figure
step(sys_lon_lqr_new)
grid on
title('Longitudinal Step Response')
ylabel('Longitudinal State')


figure
bode(sys_lon_lqr_new)
title('Longitudinal Bode Response with Tracking using LQR')
grid on

%% Plotting for checking trim conditions:
xinit = x_dot_lon; t = linspace(0, 150, 10^3); tspan = [0, t(end)];
[t,x] = ode45(@sys_lon_control, tspan, xinit);
x_u = x(:, 1);
x_w = x(:, 2);
x_q = x(:, 3);
x_t = x(:, 4);
plot_lon(t,x_u, x_t);
%
figure;
step(longitudinal_new)
grid on
%% Functions
%
function dx = sys_lon_control(t, x)
% u and y are static, algebraic maps. So, we just need to integrate dx and
% then use its solution, x, to give the output map y.
    global A_lon K_lon
    %
    A = A_lon;
    B = [0.0001    0.1880;
        -0.1234         0;
        -1.3996         0;
         0         0];
    C = [1      0      0      0;
         0      0      0      1];
    D = [0      0;
         0      0];
    % K = K_lon;
    K = K_lon; % [K_lon(:,1), K_lon(:,2), zeros(2,1), zeros(2,1)];

    %
    G_lon = -C*(A-B*K)^-1*B; F= G_lon^-1; r = [0.9987; 0.0515];
    %
    u = -K*x + F*r;
    dx = A*x + B*u;
end
%
function [x_u, x_w, x_q, x_t] = solver_RK4(xdot, t, yinit)
stepsize = t(2) - t(1);
m = length(yinit);
x = zeros(m, length(t) - 1);
x(:, 1) = yinit;
k1 = zeros(1, m); k2 = k1; k3 = k1; k4 = k1;
for ii=1:length(t)-1
    % calculate k values for T4
    k1 = xdot(t(ii),x(:, ii))';
    k2 = xdot(t(ii) + stepsize/2, x(:, ii) + (stepsize/2)*k1)';
    k3 = xdot(t(ii) + stepsize/2, x(:, ii) + (stepsize/2)*k2)';
    k4 = xdot(t(ii) + stepsize,x(:, ii) + stepsize*k3');
    T4 = (1/6)*(k1 + 2*k2 + 2*k3 + k4);
    x(:, ii+1) = x(:, ii) + stepsize*T4;
end
%
x_u = x(1, :);
x_w = x(2, :);
x_q = x(3, :);
x_t = x(4, :);
end
%
function plot_lon(t,x_u, x_t)
figure;
subplot(2,1,1)
plot(t, x_u);
hold on
title('Forward Velocity Normalized to Air Speed')
plot(t, (17.9762/18)*ones(numel(t),1),'LineStyle','--')
legend({'Forward Velocity','Forward Velocity Reference at Trim Conditions'})
hold off
grid on
subplot(2,1,2)
plot(t, x_t);
hold on
title('Pitch')
plot(t, 0.0515.*ones(numel(t),1),'LineStyle','--')
legend({'Pitch','Pitch Reference at Trim Conditions'})
grid on
hold off
end
```

6.2 Lateral Code

```
clc, clear all, close all

%% Airplane Parameters
span = 3.067 ; %m
area = .6282 ; %m^2
chord = .208 ; %m
mass = 5.74 ; %kg
g = 9.81 ; %m/s^2
Ixx = 1.2009 ; %kg m^2
Iyy = .9318 ; %kg m^2
Izz = 2.0734 ; %kg m^2
Ixz = .0946 ; %kg m^2

%% Trim State
h = 1800 ; %m

%Longitudinal
Va = 18 ; %m/s
theta = .0515 ; %rad
u = 17.9762 ; %m/s
w = .9263 ; %m/s
q = 0 ;
Δ_e = -.537 ; %rad
Δ_t = 17.92 ; %percentage of max
du = 0 ;
dw = 0 ;
dq = 0 ;

%Lateral
Δ_a = 0 ; %rad
Δ_r = 0 ; %percentage of max
dv = 0 ;
dp = 0 ;
dr = 0 ;
v = 0 ; %m/s
p = 0 ;
r = 0 ;
phi = 0 ;
B = 0 ;

%% Linear Model Parameters

%Force Parameters
Xu = -.1271 ; %s^-1
Xw = .6409 ; %s^-1
Xq = -.9106 ; %m/s
Yv = -.3714 ; %s^-1
Yp = .8254 ; %m/s
Yr = -17.6451 ; %m/s
Zu = -.07655 ; %s^-1
Zw = -6.3237 ; %s^-1
Zq = 16.9091 ; %m/s
X_Δ_e = .0018 ; %m/s^2
X_Δ_t = 3.3846 ; %m/s^2
Y_Δ_a = -.0137 ; %m/s^2
Y_Δ_r = .0556 ; %m/s^2
```

```
Z_Δ_e = -.1234 ; %m/s^2

%Moment Parameters
Lv = -1.1467 ; %(m*s)^-1
Lp = -15.7093 ; %s^-1
Lr = 2.6774 ; %s^-1
Mu = .1090 ; %(m*s)^-1
Mw = -2.1148 ; %m*s)^-1
Mq = -3.2853 ; %s^-1
Nv = .6400 ; %(m*s)^-1
Np = -1.2356 ; %s^-1
Nr = -.5669 ; %s^-1
L_Δ_a = -5.3580 ; %s^-2
L_Δ_r = .0316 ; %s^-2
M_Δ_e = -1.3996 ; %s^-2
N_Δ_a = -.2566 ; %s^-2
N_Δ_r = -.1309 ; %s^-2

%% State Space Matrices
A_lat = [Yv, Yp,   Yr,    g*cos(theta);
         Lv, Lp,   Lr,        0;
         Nv, Np,   Nr,        0;
         0,  1,  tan(theta), 0]  ;
B_lat = [Y_Δ_a, Y_Δ_r;
         L_Δ_a, L_Δ_r;
         N_Δ_a, N_Δ_r;
             0,         0] ;
C_lat = [1, 0, 0, 0;
         0, 0, 0, 1] ;
% C_lat = eye(4) ;
u_lat = [Δ_a; Δ_r] ;
x_lat = [v, p, r, phi]' ;
ylat = [v; phi] ;
D = zeros(2,2) ;

%% State Space Control

states_lat = {'Lateral Velocity' 'Roll Angular Velocity' 'Yaw Angular Velocity' ...
    'Roll Angle'} ;
inputs_lat = {'Aileron' 'Rudder'} ;
outputs_lat = {'Lateral Velocity' 'Roll Angle'} ;
lateral = ss(A_lat, B_lat, C_lat, D, 'statename', states_lat, 'inputname', ...
    inputs_lat, 'outputname',outputs_lat) ;

[num1,den1] = ss2tf(A_lat, B_lat, C_lat, D,1) ;
[num2,den2] = ss2tf(A_lat, B_lat, C_lat, D,2) ;
yvel_aileron = tf(num1(1,:),den1) ;
yvel_rudder = tf(num2(1,:),den2) ;
roll_aileron = tf(num1(2,:),den1) ;
roll_rudder = tf(num2(2,:),den2) ;

poles_lat = eig(A_lat) ;

% One pole is in RHP, leading to instability in step response

figure
step(lateral)
lateral_info = stepinfo(lateral) ;
title('Uncompensated Lateral Step Response')
```

```
ylabel('Lateral State')

figure
bode(lateral)
title('Uncompensated Lateral Bode Response')

%% Proportional Control

%At this point we have a stable system for the longitudinal analysis.
%However, the lateral analysis contains one unstable pole. Use feedback
%control law u = kx in order to place poles, and change system stability.
%In order to do this, we need to check if the system itself is controllable

%Controllability check
Co_lat = ctrb(A_lat,B_lat) ;
rank(Co_lat) ;
%Controllability matrix is full rank, and therefore is controllable. Now
%pick pole values st the lateral system dynamics are stable
corrected_poles_lat = [-1, -1, -.5, -.5] ;
[Kp_lat,prec,message] = place(A_lat,B_lat,corrected_poles_lat) ; %Kp_lat is ...
    proportional controller, prec is measurement of how closely eig(A-BKp) match ...
    specified pole location
A_lat_new = A_lat-B_lat*Kp_lat ;

states_lat = {'Lateral Velocity' 'Roll Angular Velocity' 'Yaw Angular Velocity' ...
    'Roll Angle'} ;
inputs_lat = {'Aileron' 'Rudder'} ;
outputs_lat = {'Lateral Velocity' 'Roll Angle'} ;
sys_place = ss(A_lat_new,B_lat,C_lat,D, 'statename', states_lat, 'inputname', ...
    inputs_lat, 'outputname',outputs_lat) ;

figure
step(sys_place)
sys_place_info = stepinfo(sys_place) ;
title('Lateral Step Response with Pole Placement')
ylabel('Lateral State')

figure
bode(sys_place)
title('Lateral Bode Response with Pole Placement')

%% Feed Forward/Tracking Control
% We need to implement a control term s.t. the output can track the loop
% error from the input. This control should be s.t. u = kx+rF. The F term
% will allow y (output) to track r (input)


Z = [zeros(size(A_lat,1),size(C_lat,1)); eye(2)] ;
N = inv([A_lat, B_lat; C_lat, D])*Z ;
Nx = [N(1:size(A_lat,1),1:size(C_lat,1))] ;
Nu = [N((size(A_lat,1)+1):(size(A_lat,1)+size(C_lat,1)),1:size(C_lat,1))] ;
N_bar = Nu+Kp_lat*Nx ;


B_lat_new = B_lat*N_bar ;

states_lat = {'Lateral Velocity' 'Roll Angular Velocity' 'Yaw Angular Velocity' ...
    'Roll Angle'} ;
inputs_lat = {'Aileron' 'Rudder'} ;
```

```
outputs_lat = {'Lateral Velocity' 'Roll Angle'} ;
sys_track = ss(A_lat_new,B_lat_new,C_lat,D, 'statename', states_lat, ...
    'inputname', inputs_lat, 'outputname',outputs_lat) ;
[num,den] = ss2tf(A_lat_new,B_lat_new,C_lat,D,1) ;


poles_lat_track = eig(A_lat-B_lat*Kp_lat)  ;

figure
step(sys_track)
sys_track_info = stepinfo(sys_track) ;
title('Lateral Step Response with Reference Tracking')
ylabel('Lateral State')

figure
bode(sys_place)
title('Lateral Bode Response with Reference Tracking')

%% Tuned Proportional Control Fast Response
fast_poles_lat = [-100 -100 -90 -50] ; %Poles are extremely fast. The larger ...
    the split between the poles, the greater the overshoot
[Kp_lat_fast,prec,message] = place(A_lat,B_lat,fast_poles_lat) ;

N_bar_fast = Nu+Kp_lat_fast*Nx ;

A_lat_fast = A_lat-B_lat*Kp_lat_fast ;
B_lat_fast = B_lat*N_bar_fast ;

states_lat = {'Lateral Velocity' 'Roll Angular Velocity' 'Yaw Angular Velocity' ...
    'Roll Angle'} ;
inputs_lat = {'Aileron' 'Rudder'} ;
outputs_lat = {'Lateral Velocity' 'Roll Angle'} ;
sys_lat_fast = ss(A_lat_fast,B_lat,C_lat,D, 'statename', states_lat, ...
    'inputname', inputs_lat, 'outputname',outputs_lat) ;
poles_lat_fast = eig(A_lat-B_lat*Kp_lat_fast) ;

figure
step(sys_lat_fast)
sys_fast_info = stepinfo(sys_lat_fast) ;
title('Lateral Step Response With Fast Poles')
ylabel('Lateral State')

figure
bode(sys_place)
title('Lateral Bode Response with Fast Poles')

%% Tuned Proportional Control Imaginary Poles
im_poles_lat = [-1 -7+.7*j -7-.7*j -.5] ; % The higher the imaginary value, the ...
    worse the overshoot becomes on top right and bottom left
[Kp_lat_im,prec,message] = place(A_lat,B_lat,im_poles_lat)

N_bar_im = Nu+Kp_lat_im*Nx ;
A_lat_im = A_lat-B_lat*Kp_lat_im ;
B_lat_im = B_lat*N_bar_im ;

states_lat = {'Lateral Velocity' 'Roll Angular Velocity' 'Yaw Angular Velocity' ...
    'Roll Angle'} ;
inputs_lat = {'Aileron' 'Rudder'} ;
outputs_lat = {'Lateral Velocity' 'Roll Angle'} ;
```

```matlab
sys_lat_im = ss(A_lat_im,B_lat_im,C_lat,D, 'statename', states_lat, ...
    'inputname', inputs_lat, 'outputname',outputs_lat) ;
poles_lat_im = eig(A_lat-B_lat*Kp_lat_im) ;

figure
step(sys_lat_im)
sys_im_info = stepinfo(sys_lat_im) ;
title('Lateral Step Response With Imaginary Poles')
ylabel('Lateral State')

figure
bode(sys_place)
title('Lateral Bode Response with Imaginary Poles')

%% LQR Control
%Prior control uses trial and error to tune gain. Now, let's use linear
%quadratic control to find optimal gain matrix.


Q = A_lat'*A_lat ;
R = B_lat'*B_lat ;
N = A_lat'*B_lat ;
[Kp_lqr, ARE, poles_lqr] = lqr(sys_place,Q,R,N) ;

N_bar_lqr = Nu+Kp_lqr*Nx ;
A_lat_lqr = A_lat-B_lat*Kp_lqr ;
B_lat_lqr = B_lat*N_bar_lqr ;

states_lat = {'Lateral Velocity' 'Roll Angular Velocity' 'Yaw Angular Velocity' ...
    'Roll Angle'} ;
inputs_lat = {'Aileron' 'Rudder'} ;
outputs_lat = {'Lateral Velocity' 'Roll Angle'} ;
sys_lat_lqr = ss(A_lat_lqr,B_lat_lqr,C_lat,D, 'statename', states_lat, ...
    'inputname', inputs_lat, 'outputname',outputs_lat) ;
poles_lat_lqr = eig(A_lat_lqr) ;

figure
step(sys_lat_lqr)
sys_lqr_info = stepinfo(sys_lat_lqr) ;
title('Lateral Step Response with LQR')
ylabel('Lateral State')

figure
bode(sys_place)
title('Lateral Bode Response with LQR')

%% LQR Tuner
Q = 200*eye(4) ; %The higher Q and lower R, better OS & rt
R = .02*eye(2) ;
N = 0 ;

[Kp_lqr_new, ARE_new, poles_lqr_new] = lqr(sys_place,Q,R,N) ;

N_bar_lqr_new = Nu+Kp_lqr_new*Nx ;
A_lat_lqr_new = A_lat-B_lat*Kp_lqr_new ;
B_lat_lqr_new = B_lat*N_bar_lqr_new ;

states_lat = {'Lateral Velocity' 'Roll Angular Velocity' 'Yaw Angular Velocity' ...
    'Roll Angle'} ;
```

```
inputs_lat = {'Aileron' 'Rudder'} ;
outputs_lat = {'Lateral Velocity' 'Roll Angle'} ;
sys_lat_lqr_new = ss(A_lat_lqr_new,B_lat_lqr_new,C_lat,D, 'statename', ...
    states_lat, 'inputname', inputs_lat, 'outputname',outputs_lat) ;
poles_lat_lqr_new = eig(A_lat_lqr_new) ;


figure
step(sys_lat_lqr_new)
sys_lqrnew_info = stepinfo(sys_lat_lqr_new) ;
title('Lateral Step Response with Tuned LQR')
ylabel('Lateral State')

figure
bode(sys_place)
title('Lateral Bode Response with Tuned LQR')
```