Basic Control of a Generic Air-Breathing Hypersonic Vehicle

Research Project Report, Course #085851

Omer Wexler

Advisor: Moshe Idan

September 2024

In this work, we focus on the longitudinal control of an air-breathing hypersonic vehicle (AHV). Linear control methods such as PID and LQR are explored to stabilize the system and provide a basic intuition of its different intricacies. A nonlinear feedback linearization controller is implemented based on work made in [1]. Simulations are performed to test those controllers and explain their behavior, as well as discover the strengths and shortcomings of those different methods. Results are compared with previous papers to validate the implemented model and simulations. Conclusions are made to assist with further research and the improvement of this specific project.

Nomenclature

Linear Control Notations

$\delta\vec{u}$ Pertubation in system inputs from nominal values.

$\delta\vec{x}$ Pertubation in system states from nominal values.

$\vec{u}$ Space state vector notation for input.

$\vec{x}$ Space state vector notation for system states.

$\vec{y}$ Linear system output.

A, B The Jacobian matrices used for linearization.

C, D Define the relation between the states and inputs of a linear system to its output.

K_p, K_i, K_d Respective gains for the proportional, integral, and derivative parts of a PID controller.

Q_0 Controllability matrix or nominal pitch rate (context-dependent).

Model Constants and Variables

α Angle of attack

β_i ith thrust fit parameter

δ_e Abstract elevator deflection

γ Flight path angle

$\bar{c}$ Mean aerodynamic chord

Φ Stoichiometrically normalized fuel-to-air ratio

ρ Air density

θ Pitch angle

C_D^0	Constant drag coefficient
$C_D^{\alpha^i}$	ith order coefficient of α contribution to C_D
$C_D^{\delta_e^i}$	ith order coefficient of δ_e contribution to C_D
c_e	Elevator coefficient of moment
C_L^0	Constant lift coefficient
$C_L^{\alpha^i}$	ith order coefficient of α contribution to C_L
C_M^0	Constant pitching moment coefficient
$C_M^{\alpha^i}$	ith order coefficient of α contribution to C_M
C_T^0	Constant thrust coefficient
$C_T^{\alpha^i}$	ith order coefficient of α contribution to C_T
D	Drag
g	Gravitation constant
h	Height
I_y	Longitudinal moment of inertia
L	Lift
M	Pitching moment
m	Vehicle mass
Q	Pitch rate
q	Dynamic pressure
S	Reference area
T	Thrust
V	Air speed (magnitude)
z_T	Thrust to moment coupling coefficient

Other symbols

γ_{ref}	Reference trajectory angle.
$\mathcal{L}_f$	Lie derivative.
$\mathcal{L}_g$	Refers to the control inputs' coefficients in normal form derivatives.
ϕ_A	Root locus asymptote angles.
σ_A	Root locus asymptote center of gravity.
ε	The small perturbation used for numeric linearization and derivative.
$\vec{z}$	Normal form state notation.
A_c	The normal-from decoupling matrix.
j	The square root of -1.
V_{ref}	Reference speed.

1 Introduction

In recent years, the application of AHVs in civil and military fields has become the center of attention of many institutes and countries alike due to its promising capabilities. Alongside its many complex anecdotes, such vehicles are highly unstable and nonlinear in nature, introducing a complex set of challenges that must be overcome to control such aircraft.

This project is the first step of our research regarding air-breathing hypersonic vehicles. Its goals are to understand the basic methods present in controlling such challenging vehicles, study their complex dynamics and behavior, and gain experience in implementing Matlab and Simulink models as well as performing different control-related analyses using these tools.

To achieve these goals, we will follow the footsteps of [1, 2] and attempt to recreate some of their work, while explaining and examining the results and techniques implemented by them to develop our understanding. First, a widespread longitudinal model is reconstructed,

studied, tested, and linearized to provide the infrastructure for this project. Results regarding the model are verified against previous works to confirm it is proper. This model is known to be highly unstable and includes several challenging coupling between the control inputs and forces acting upon the vehicle.

Having derived the linear system we will attempt to stabilize the pitch angle θ using linear control methods acting on δ_e as the control input, including the use of a multiloop PID controller utilizing Φ and an LQ controller, both of which are tested at the trim condition of the aircraft and away from it to examine the performance of these methods. Since the system's linearization is relevant only near the trim conditions, these methods are unlikely to be useful for tracking problems, and so a better alternative will be proposed using nonlinear control.

A nonlinear control law is explained and developed according to work done in [1], where the system is brought to a normal-form representation, some weak couplings are neglected, and dynamic extension is added to achieve full vector relative degree. Dynamic inverse is applied to this system to gain a linear system composed of two integration chains. This system is linear through the entire range of operation, and with the application of an LQ controller, we can attempt tracking commands.

Finally, we will discuss the results of this paper. Then, based on those findings, future work and research will be addressed.

2 The Model

This paper considers the curve fitted model (CFM) proposed in [1]. It is based on a rich truth model (TM) also presented in the same article, but not in full detail. The CFM encapsulates the aerodynamics and thrust via

$$L \approx \frac{1}{2}\rho V^2 S C_L(\alpha, \delta_e), \quad D \approx \frac{1}{2}\rho V^2 S C_D(\alpha, \delta_e), \tag{1a}$$

$$M \approx z_T T + \frac{1}{2}\rho V^2 S \bar{c}[C_{M,\alpha}(\alpha) + C_{M,\delta_e}(\delta_e)], \tag{1b}$$

$$T \approx C_T^{\alpha^3}\alpha^3 + C_T^{\alpha^2}\alpha^2 + C_T^{\alpha}\alpha + C_T^0. \tag{1c}$$

The normalized coefficients are composed of

$$C_L = C_L^{\alpha}\alpha + C_L^{\delta_e}\delta_e + C_L^0, \tag{2a}$$

$$C_D = C_D^{\alpha^2}\alpha^2 + C_D^{\alpha}\alpha + C_D^{\delta_e^2}\delta_e^2 + C_D^{\delta_e}\delta_e + C_D^0, \tag{2b}$$

$$C_{M,\alpha} = C_M^{\alpha^2}\alpha^2 + C_M^{\alpha}\alpha + C_M^0, \quad C_{M,\delta_e} = c_e\delta_e, \tag{2c}$$

$$C_T^{\alpha^3} = \beta_1(h,q)\Phi + \beta_2(h,q), \quad C_T^{\alpha^2} = \beta_3(h,q)\Phi + \beta_4(h,q), \tag{2d}$$

$$C_T^{\alpha} = \beta_5(h,q)\Phi + \beta_6(h,q), \quad C_T^0 = \beta_7(h,q)\Phi + \beta_8(h,q), \tag{2e}$$

while all relevant coefficients are listed in Appendix A.

In addition to the CFM, a control-oriented model (COM) is obtained when neglecting elevator couplings ($C_L^{\delta_e}$), flight altitude, and the flexible states (that will not be mentioned in

detail in this paper). The relevant equations of the COM (in state space) are

$$\vec{x} = \begin{Bmatrix} V & \alpha & Q & \theta \end{Bmatrix}^T, \quad \vec{u} = \begin{Bmatrix} \delta_e & \Phi \end{Bmatrix}^T \tag{3a}$$

$$\vec{f}(\vec{x}) = \begin{Bmatrix} f_1(\vec{x}) \\ f_2(\vec{x}) \\ f_3(\vec{x}) \\ f_4(\vec{x}) \end{Bmatrix} = \begin{Bmatrix} \dot{V} \\ \dot{\alpha} \\ \dot{Q} \\ \dot{\theta} \end{Bmatrix} = \begin{Bmatrix} \frac{1}{m}\Big(T\cos(\alpha) - D\Big) - g\sin(\theta - \alpha) \\ \frac{1}{mV}\Big(-T\sin(\alpha) - L\Big) + Q + \frac{g}{V}\cos(\theta - \alpha) \\ \frac{M}{I_y} \\ Q \end{Bmatrix}. \tag{3b}$$

The CFM is used for simulation purposes, whereas the COM is used for analysis and control design. The CFM is implemented in Matlab and Simulink and validated using the information provided in [1] (trim conditions, step responses, linearization pole-zero configurations, etc.). Results are also compared with [2].

3 Trim Conditions

Trim conditions are found for both models, used for validation and later on - linearization. The trim condition is defined as steady, level flight, requiring

$$\gamma = \theta - \alpha = 0 \Longrightarrow \theta_0 = \alpha_0, \qquad \text{(Zero flight path angle)}, \tag{4a}$$

$$Q_0 = 0, \qquad \text{(Pitch rate is zero)}. \tag{4b}$$

The flight conditions for this trimmed state as detailed in Parker (2007) are listed in Table 1

Table 1: Flight conditions.

Set Variable	Set Value
h_0	85000 [ft]
V_0	7702.0808 [ft/sec]

Using (3) and (4), we are left with the following equations, where the variables we are trying to solve for are $\vec{y} = \begin{bmatrix} \alpha_0 & \delta_{e,0} & \Phi_0 \end{bmatrix}$,

$$\begin{Bmatrix} \dot{V} \\ \dot{\alpha} \\ \dot{Q} \end{Bmatrix} = \begin{Bmatrix} \frac{1}{m}\Big(T\cos(\alpha_0) - D(\alpha_0, \delta_{e,0}, \Phi_0)\Big) \\ \frac{1}{mV_0}\Big(-T\sin(\alpha_0) - L(\alpha_0, \delta_{e,0}, \Phi_0)\Big) + \frac{g}{V_0} \\ \frac{M(\alpha_0, \delta_{e,0}, \Phi_0)}{I_{yy}} \end{Bmatrix} = \overline{0}. \tag{5}$$

Solving this system of equations, one could find the appropriate trim conditions for the COM or CFM. The detailed solution approach is described in Appendix B.

Table 2 details the different trim conditions found in [1, 2], and in this paper. Note that [1] provides only trim conditions found regarding the TM, which are slightly different than those presented here and in [2], hinting at a slight model mismatch (as one would expect).

Table 2: Trim conditions of [1, 2], and what was calculated in this paper.

State/Input	COM (Computed)	CFM (Computed)	COM (Vaknin)	CFM (Vaknin)	Truth Model (Parker)
α_0	3.684°	1.646°	3.65°	1.62°	1.515°
$\delta_{e,0}$	16.368°	12.544°	16.29°	12.49°	11.463°
Φ	0.161	0.268	0.16	0.27	0.2514

Substituting the results from Table 2 into (5), we find the state derivatives at trim are

$$\vec{f}(\vec{y}_{sol}) = \begin{Bmatrix} \dot{V} \\ \dot{\alpha} \\ \dot{Q} \\ \dot{\theta} \end{Bmatrix} = \begin{Bmatrix} 1.518 \cdot 10^{-15} \\ 0 \\ -4.729 \cdot 10^{-17} \\ 0 \end{Bmatrix}$$

for the COM, and

$$\vec{f}(\vec{y}_{sol}) = \begin{Bmatrix} \dot{V} \\ \dot{\alpha} \\ \dot{Q} \\ \dot{\theta} \end{Bmatrix} = \begin{Bmatrix} 0 \\ -8.673 \cdot 10^{-19} \\ -3.638 \cdot 10^{-17} \\ 0 \end{Bmatrix}$$

for the CFM. This indicates that the trim conditions found solve our problem. As we will demonstrate later on, this point is unstable and the aircraft cannot maintain it.

4 Model Validation

The model is implemented in Simulink via the **MATLAB Function** block. First, to validate the implemented model, we simulate the system with the initial conditions specified in Table 3. The comparison will be made with results presented in [2]. The implemented model will be initialized with both conditions to validate what was found and qualitatively compare the open-loop responses.

Table 3: Simulated Trim Conditions, Compared to with [2].

	V_0 [ft/sec]	α_0[°]	Q_0 [°/sec]	θ_0[○]	h_0 [ft]	$\delta_{e,0}$[°]	Φ_0
Vaknin (CFM) [2]	7702.0808	1.6232°	0	1.6232°	85000	12.4895°	0.2665
Found Trim (CFM)	7702.0808	1.6465°	0	1.6465°	85000	12.5447°	0.2682

Figure 1 shows simulation results with said initial conditions, performed on the CFM configuration. The results are qualitatively similar, both diverge at approximately the same point in time and trim the aircraft to some extent (for a short time). As expected by previous articles such as [1], the aircraft's dynamics are unstable, and left uncontrolled, it will diverge. These results imply that the model is indeed valid and can be used to simulate the system.

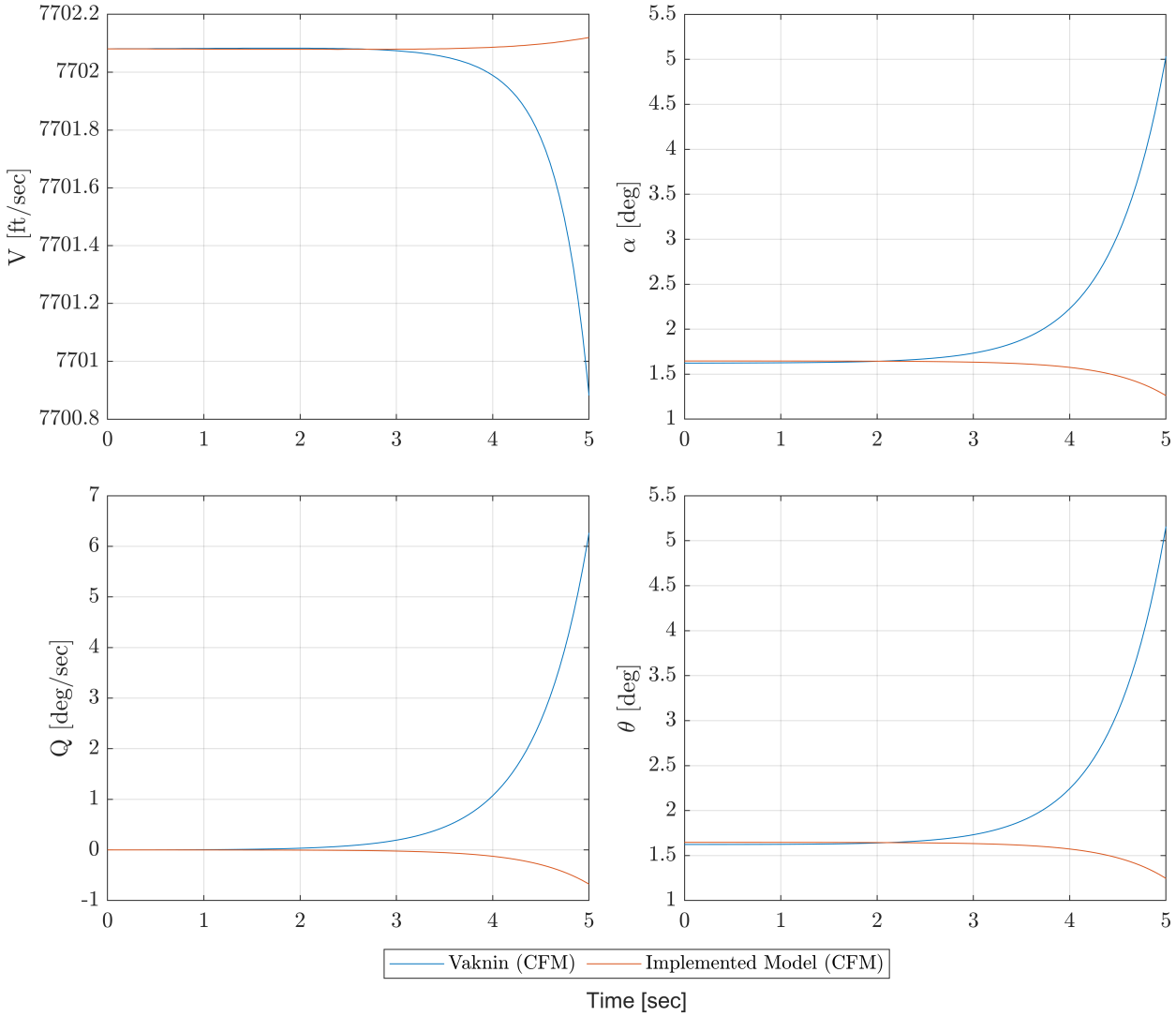


Figure 1: Open loop simulation results.

5 Numeric Linearization

This section will focus on the numeric linearization performed on the CFM. For this, we will use central differences to numerically calculate the values of the Jacobian matrices A and B, which is derived from the equations of motion (3) in the following manner

$$\delta\dot{\vec{x}} = A\delta\vec{x} + B\delta\vec{u}, \tag{6a}$$

$$A_{ij} = \frac{\partial f_i}{\partial x_j}, \quad B_{ij} = \frac{\partial f_i}{\partial u_j}, \tag{6b}$$

$$\frac{\partial \vec{f}}{\partial x_i} = \frac{\vec{f}(\vec{x}_0 + \varepsilon_x\vec{e}_i, u_0) - \vec{f}(\vec{x}_0 - \varepsilon_x\vec{e}_i, u_0)}{2\varepsilon_x}, \quad \varepsilon_x \to 0, \quad (\vec{e}_i)_j = \begin{cases} 1, & j = i \\ 0, & \text{otherwise} \end{cases}, \tag{6c}$$

$$\frac{\partial \vec{f}}{\partial u_i} = \frac{\vec{f}(\vec{x}_0, \vec{u}_0 + \varepsilon_u\vec{k}_i) - \vec{f}(\vec{x}_0, \vec{u}_0 - \varepsilon_u\vec{k}_i)}{2\varepsilon_u}, \quad \varepsilon_u \to 0, \quad (\vec{k}_i)_j = \begin{cases} 1, & j = i \\ 0, & \text{otherwise} \end{cases}. \tag{6d}$$

Applying (6) with $\varepsilon_x = 1 \cdot 10^{-5}$ and $\varepsilon_u = 1 \cdot 10^{-5}$, the results for A and B are

$$A = \begin{bmatrix} -0.00155 & 19.8572 & 0 & -32.2 \\ -1.0798 \cdot 10^{-6} & -0.06968 & 1 & 0 \\ -7.8283 \cdot 10^{-6} & 2.9879 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} -40.7230 & 24.7000 \\ -0.0112 & -9.2182 \cdot 10^{-5} \\ -1.4909 & 0.12394 \\ 0 & 0 \end{bmatrix}. \tag{7}$$

The matrices presented in [2],

$$A = \begin{bmatrix} -0.0015 & 19.95 & 0 & -31.9 \\ -1.06 \cdot 10^{-6} & -0.0697 & 1 & 0 \\ -7.71 \cdot 10^{-6} & 2.981 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} -40.54 & 24.65 \\ -0.0112 & -9.07 \cdot 10^{-5} \\ -1.491 & 0.1237 \\ 0 & 0 \end{bmatrix}, \tag{8}$$

are almost the same as those presented here[1], with negligible difference. There are some reassuring elements to these results, for example, we find that indeed $\dot{\theta} = Q$ according to the last row of A, and that Q directly affects α according to the second row. This further validates our model, hinting it is correctly implemented and captures the system's essence.

Equation (6) applies central differences to derive a linearized model, so the choice of ε matters. To justify the choice made, Fig. 2 shows a sensitivity test for the SVDs of A and B. According to this test, the selected values are past numerical convergence and numerical errors should be suppressed.

[1] Note Vaknin's definition of the input is: $u = [\Phi, \delta_e]$ and here it is: $u = [\delta_e, \Phi]$.

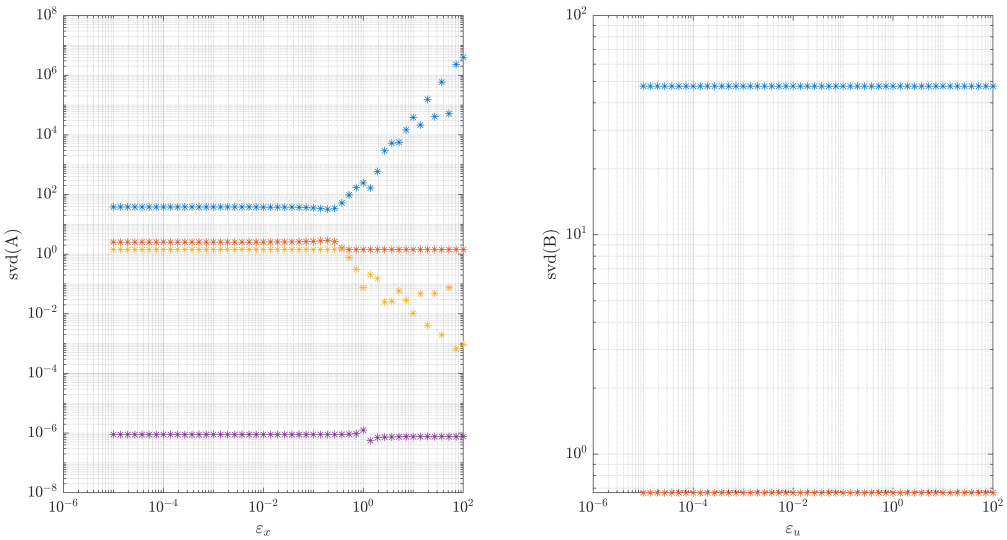
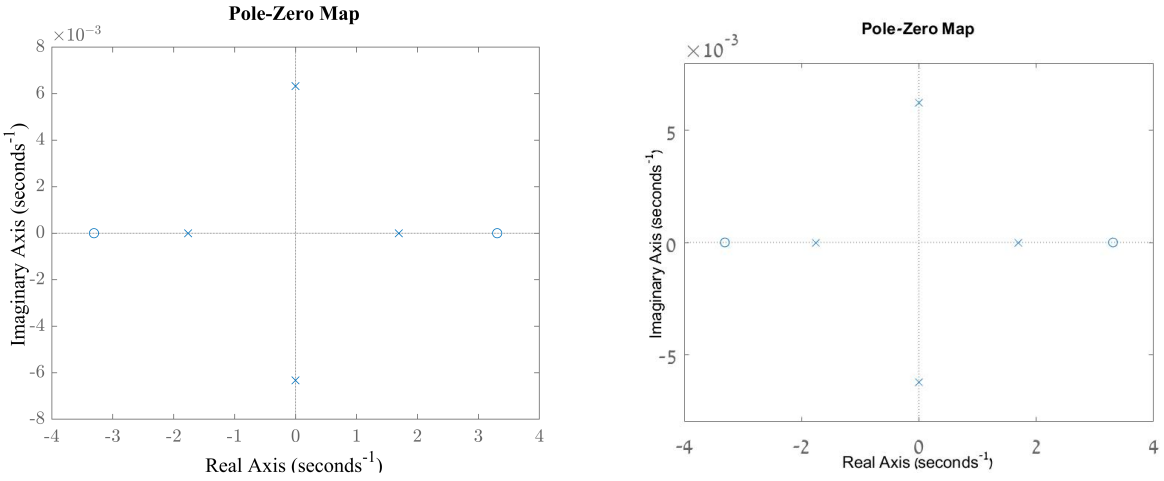


Figure 2: SVDs of A and B as a function of the numerical parameters.

Figure 3 shows the resulting Pole-Zero map of the linearized system, compared to the pole-zero map in [2]. Both maps use the definition $y = [V, \gamma]$ as the output, according to [1]. Figure 4 depicts the pole-zero maps presented by Parker. In Parker's map, there are four additional sets of poles and zeros resulting from the flexible modes, which are neglected here and in Vaknin's work. Besides the flexible modes, all maps match each other well.

The system dynamics as shown in these maps are composed of two phugoid poles with very low damping, two real short-period poles, and two zeros resulting from the pitch dynamics of the system. This system is unstable due to the short-period pole on the right, and non-minimum phase due to the pitch dynamics zero.



Implemented model. Vaknin's Model.

Figure 3: The Pole-Zero Map of the Linearized System, Compared to Vaknin's.

In Fig. 5 are the responses of the linearized and nonlinear systems. The linearized system is unstable as implied by Fig. 3. When initializing it at the trim conditions, we find that while θ and Q are somewhat "stable" at the beginning of the simulation, V and α diverge almost immediately. When compared to the non-linear system, we can see the linear one diverges much faster.

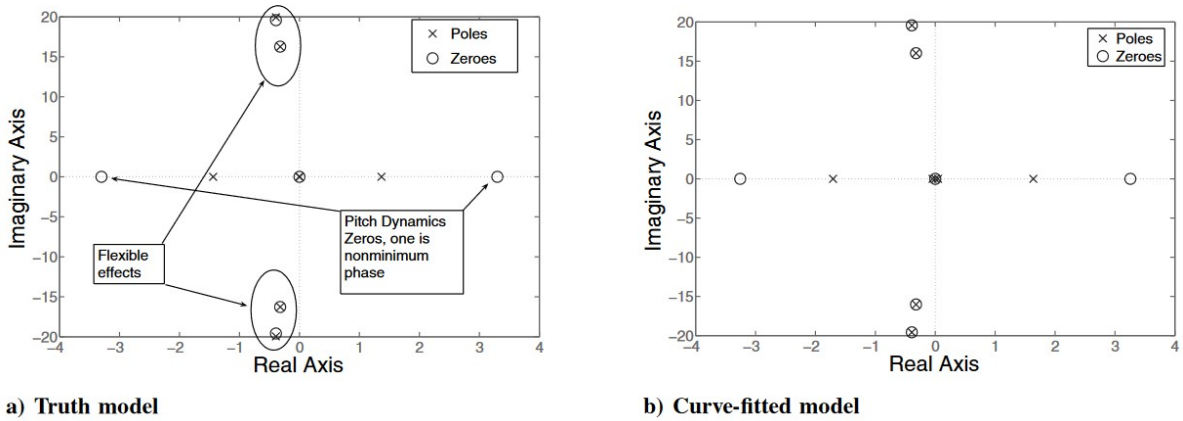


Figure 4: The Pole-Zero maps presented in [1].

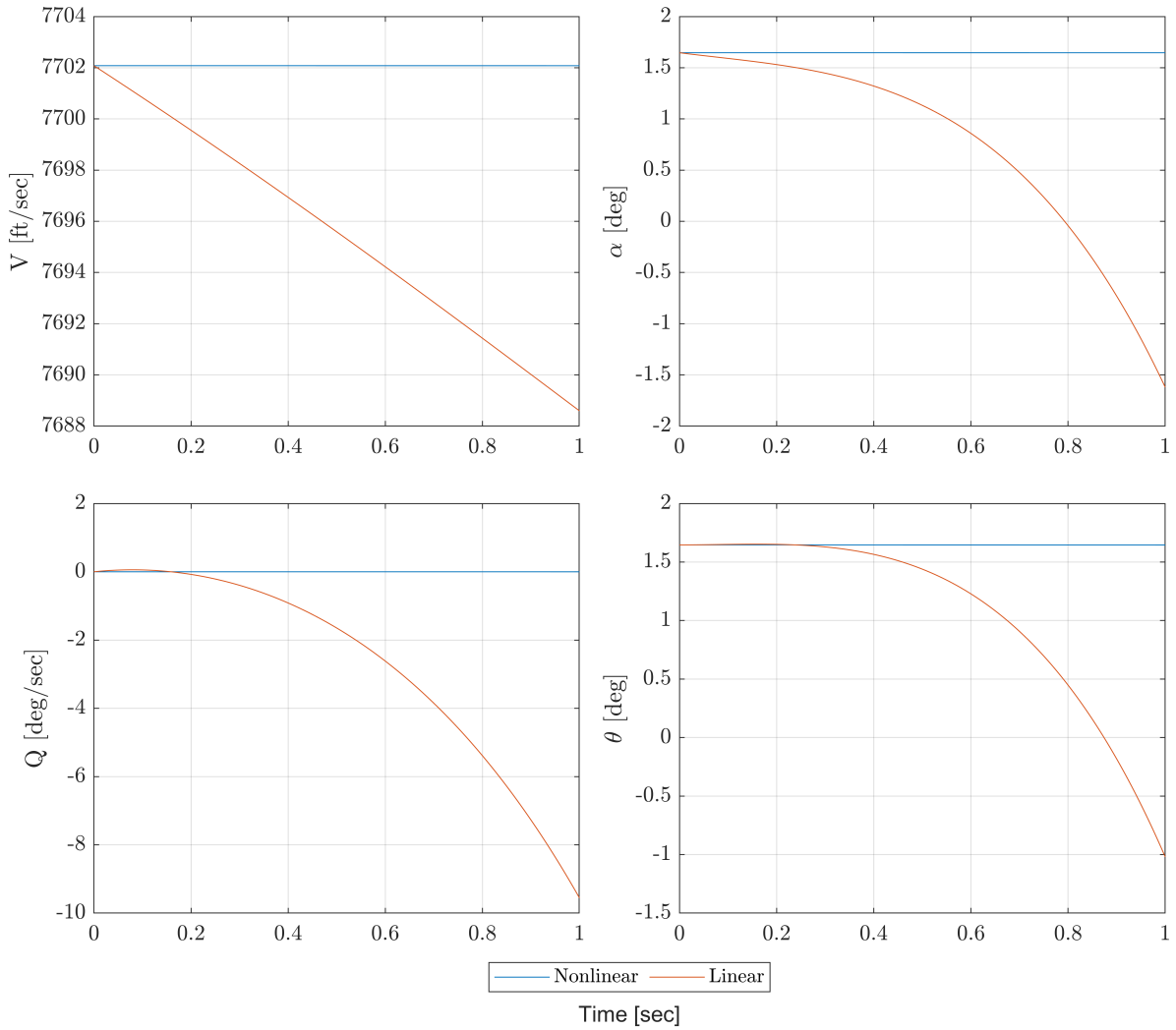


Figure 5: Comparison of the open loop simulation results for V, α, Q, θ, using the linear and non-linear model.

6 Pitch Angle Controller - Classic PID

This section will focus on an attempt to regulate θ using the actuator δ_e, within the linearized system model derived in section 5. First, we must derive the transfer function from θ to δ_e. Later, we will see that a derivative feedback controller is required to stabilize the pitch angle. It will also become apparent that the application of a controller on V is attractive to improve the dynamics of the system, hence the transfer functions for V and Q will also be derived. Our linearized model is of the form

$$\delta\vec{x} = \begin{Bmatrix} \delta V & \delta\alpha & \delta Q & \delta\theta \end{Bmatrix}^T, \tag{9a}$$

$$\delta\dot{\vec{x}} = A\delta\vec{x} + B\delta\vec{u}, \tag{9b}$$

$$\delta\vec{y} = C\delta\vec{x} + D\delta\vec{u}, \tag{9c}$$

$$\delta\vec{x} = \vec{x} - \vec{x}_0, \quad \delta\vec{u} = \vec{u} - \vec{u}_0, \tag{9d}$$

where

$$A = \begin{bmatrix} -0.00155 & 19.8572 & 0 & -32.2 \\ -1.0798 \cdot 10^{-6} & -0.06968 & 1 & 0 \\ -7.8283 \cdot 10^{-6} & 2.9879 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} -40.7230 & 24.7000 \\ -0.0112 & -9.2182 \cdot 10^{-5} \\ -1.4909 & 0.12394 \\ 0 & 0 \end{bmatrix}. \tag{10}$$

Using Matlab, we can define a linear system and obtain the two transfer functions, setting the output as θ and V respectively, and ignoring the irrelevant input. Throughout the following sections, we will not use the $\delta\vec{x}$ notation, but one should keep in mind that all states express deviations from a nominal one (the trim conditions).

First, let us look at the root locus plot of a proportional pitch angle controller, using δ_e, as shown in Fig. 6. Analyzing the plot, we infer that the system is highly unstable, with the two phugoid poles crossing to the right of the imaginary axis with relative ease (at $K \approx 0.5$), and another unstable ORHP pole associated with the short period of the AHV. There is also a NMP zero. This matches our experience with the system's stability (and lack of it).

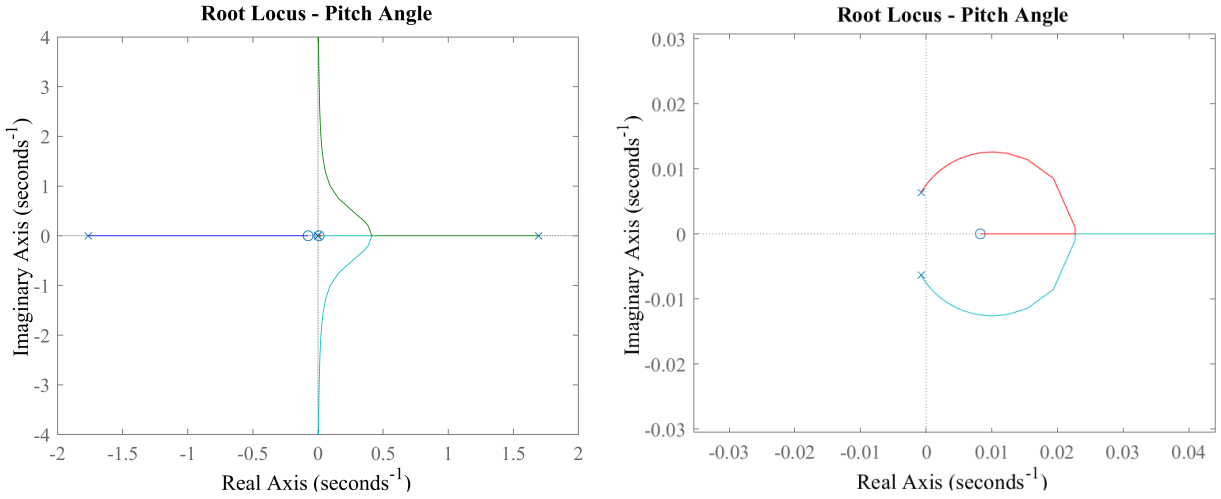


Figure 6: Root Locus Plot of the Pitch Angle Transfer Function (w.r.t δ_e).

Recall the asymptotes of an R-L diagram can be obtained as

$$\sigma_A = \frac{\sum_{n=0}^{P}(p_n) - \sum_{m=0}^{Z}(z_m)}{P - Z}, \quad \phi_A = \frac{2k+1}{P-Z}, k = 0, 1, ... \tag{11}$$

In our case, the system has 4 poles and 2 zeros, therefore the vertical asymptote. We could try and skew the asymptotes in our favor, so there exists a gain for which all poles are stable. This depends on three factors:

1. The breakaway point.
2. Asymptote center of gravity (σ_A).
3. Asymptote angle (ϕ_A).

The asymptote emerges from the center of gravity at the respective angle. This means if we were to change the angle (increase or decrease $P - Z$), the direction would depend on the resulting center of gravity. Another option would be to force a breakaway on the right plane and a break-in at the left, so the unstable pole advances towards a zero on OLHP.

Increasing $P - Z$ would "tilt" the asymptote to the right, further destabilizing the system. Decreasing $P - Z$ is a reasonable option and will be explored later. If we were to preserve the two vertical asymptotes, our only hope is to pull the COG further left, meaning we must add an equivalent amount of poles and zeros such that the zeros are placed in the ORHP and the poles are in the OLHP. This calls for a lead/lad compensator, but those will not be explored here.

First, we will consider classic PID controllers in an attempt to find a suitable candidate or disprove the use of such controllers. Needless to say, a proportional controller won't suffice at all, since the unstable pole is unstable for any gain (according to the R-L asymptotes of the system). Instinctively, we could try using a PI controller of the form

$$C_{PI} = K_P + K_I \frac{1}{s} = \frac{K(s+z)}{s}. \tag{12}$$

This controller maintains a vertical asymptote (since we added the same amount of zeros and poles). Since the asymptote center of gravity is currently at zero (-0.0019), for the unstable pole to cross left of the origin, the center of gravity must be pulled left, meaning we would need z to be positive (since the pole stands at the origin), this is sub-optimal, as already explained.

Our next proposition would be a PID controller. Usually, we would approach such suggestions with caution, due to the derivative part not being causal, but in our case, we can utilize Q. Under the assumption we can perfectly measure the pitch rate, it becomes possible to implement a derivative controller without the need for a pseudo derivative, meaning PID controllers such as

$$C_{PID} = K_P + K_I \frac{1}{s} + K_D s = \frac{K_D s^2 + Kps + K_i}{s} = \frac{K(s+z_1)(s+z_2)}{s} \tag{13}$$

become a feasible option.

This controller adds two zeros with one pole, meaning the overall system has 5 poles and 4 zeros, this is good for us as it forces the asymptote to become horizontal. That way the unstable pole breaks away at the right plane and back in the left plane. Take

$$C_{PID,1} = \frac{(s+1.1)(s+0.9)}{s} \tag{14}$$

for example, this controller results in the R-L diagram illustrated in Fig. 7. Supposedly, this result is good as it allows us to stabilize the unstable pole. Close inspection reveals this

setup causes the phugoid poles to destabilize. The phugoid "red" pole crosses to the ORHP very fast (at $K \approx 8 \cdot 10^{-5}$), which leaves us with an unstable system over long periods. For reference, the required gain to stabilize the right-side pole is $K \approx 0.6$, hence instability is inevitable, later it will be shown that we could improve this situation using a speed controller.

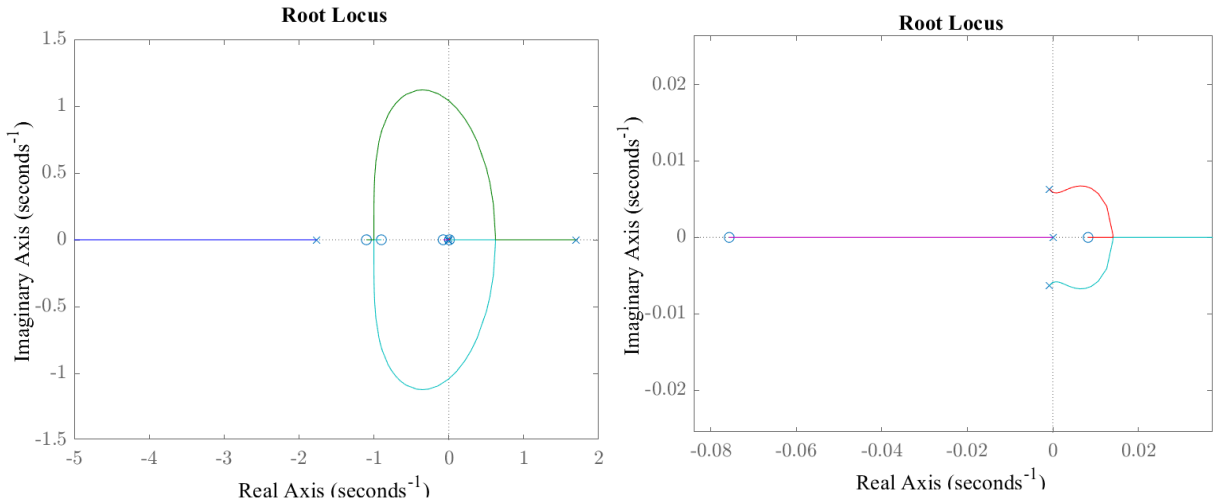


Figure 7: Root Locus of the Controlled System.

This result is inevitable. The added zeros and poles force a breakaway for the unstable pole, meaning another pole must move right to said breakaway point, changing the position of the zeros is of no use. Another solution worth considering is removing the integrator, and try using a PD controller as follows:

$$C_{PD} = K_P + K_D s = K(s + a) \tag{15}$$

Unfortunately, the result is the same - removing the Integrator still entails a right breakaway point, forcing one pole right. We can explain this behavior using the ORHP zero. The real part of R-L is only to the left of an odd number of zeros and poles. Because we have a zero and pole on the right, the real part of R-L is located between the two, and won't go left of the zero, this means the breakaway point also must be right of the origin, and that no matter the controller we won't be able to prevent one of the stable poles to cross over. This conclusion encumbers the use of such controllers, but could still redeemed.

7 Multiloop Controller Design

As evident by section 6, a single controller on θ inevitably results in divergence, and hence we would like to try and add a velocity controller to dampen the phugoid motion. The velocity controller's purpose will be to draw the phugoid poles further from the origin, allowing for greater gains using PID and PD controllers without the phugoid poles becoming unstable.

Figure 8 depicts $\dfrac{V(s)}{\Phi(s)}$ in CL when paired with a proportional controller, which is clearly very different from $\dfrac{\theta(s)}{\delta_e(s)}$. Because the phugoid poles are now drawn left instead of right (since the NMP zero is gone), we can use a proportional controller to move them further from the complex axis. This will improve the system's dynamics and dampen the phugoid motion.

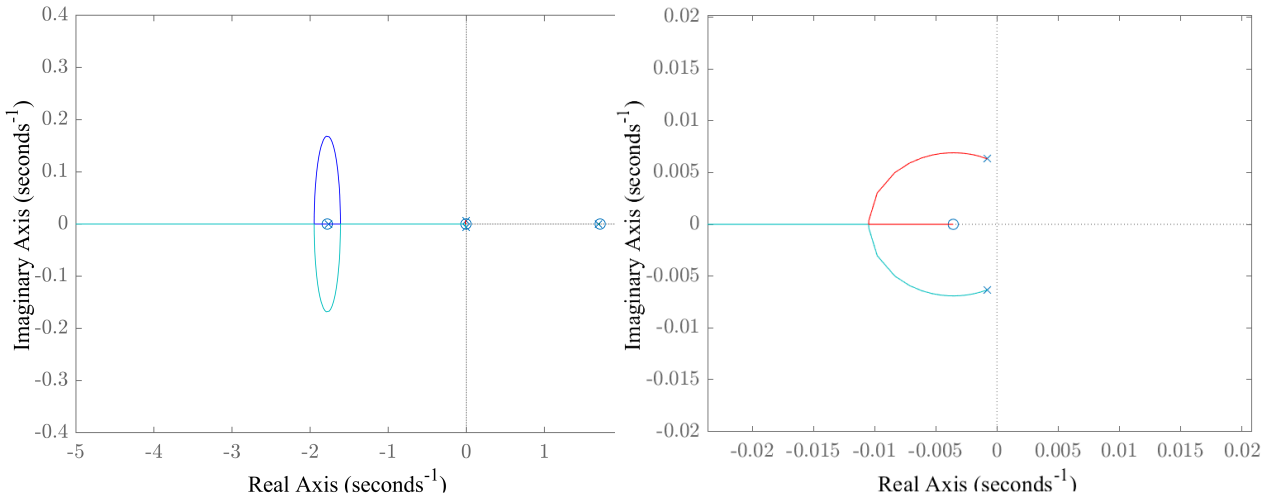


Figure 8: Closed loop root locus for $\dfrac{V(s)}{\Phi(s)}$ with a proportional controller.

For this purpose, we can use the controller

$$\Phi = K_V(0 - V) = -K_V V. \tag{16}$$

Since we would like to "regulate" V, the command in this case is 0. Do keep in mind - this controller will not stabilize the velocity and is not meant to, its purpose is to improve phugoid dynamics.

As for the actual value of K_V - we would ideally like to pull the phugoid poles left, the gain for which the closest of the two is leftmost is the break-away point. Any gain lower will pull the "red" pole right, higher gain will pull the "cyan" pole right as well. Hence the ideal gain is $K_V = 7.6829 \cdot 10^{-4}$ to yield two real stable and equal poles. Note, that it increases the distance of the poles from the origin by orders of magnitude, hopefully allowing us to stabilize θ. While that choice for K_V has no guarantee of working, it is a good starting point.

With (16) in place, the new system could be written as (the δ notation is removed, but not forgotten)

$$\begin{aligned}
&\vec{x} = \begin{Bmatrix} V & \alpha & Q & \theta \end{Bmatrix}^T, \quad \vec{u} = \begin{Bmatrix} \delta_e & u \end{Bmatrix}^T, \\
&\dot{\vec{x}} = A\vec{x} + B\vec{u} = A\vec{x} + \begin{bmatrix} B_1 & B_2 \end{bmatrix} \begin{Bmatrix} \delta_e \\ \Phi \end{Bmatrix} = A\vec{x} + B_1\delta_e - B_2 K_V V = \\
&\quad = A\vec{x} + B_1\delta_e - K_V B_2(\begin{Bmatrix} 1 & 0 & 0 & 0 \end{Bmatrix} \vec{x}) = (A - B_2\vec{K}_V)\vec{x} + B_1\delta_e,
\end{aligned} \tag{17}$$

where $\vec{K}_V = \begin{Bmatrix} K_V & 0 & 0 & 0 \end{Bmatrix}$. This effectively transforms the linear system into a SIMO system.

With the improved phugoid, we can now attempt to stabilize θ once again. Let's begin by examining the new closed-loop poles of $\frac{\theta(s)}{\delta_e(s)}$ when applying a proportional controller, as seen in Fig. 9.

This new system is different than what one might expect. We previously saw that the system should have a vertical asymptote, which is clearly not present. To understand why, we could take a look at the respective transfer function:

$$\frac{\theta(s)}{\delta_e(s)} = \frac{-1.491s^2 - 0.1617s - 0.002207}{s^4 + 0.08853s^3 - 2.987s^2 - 0.05749s - 0.0003099} \tag{18}$$

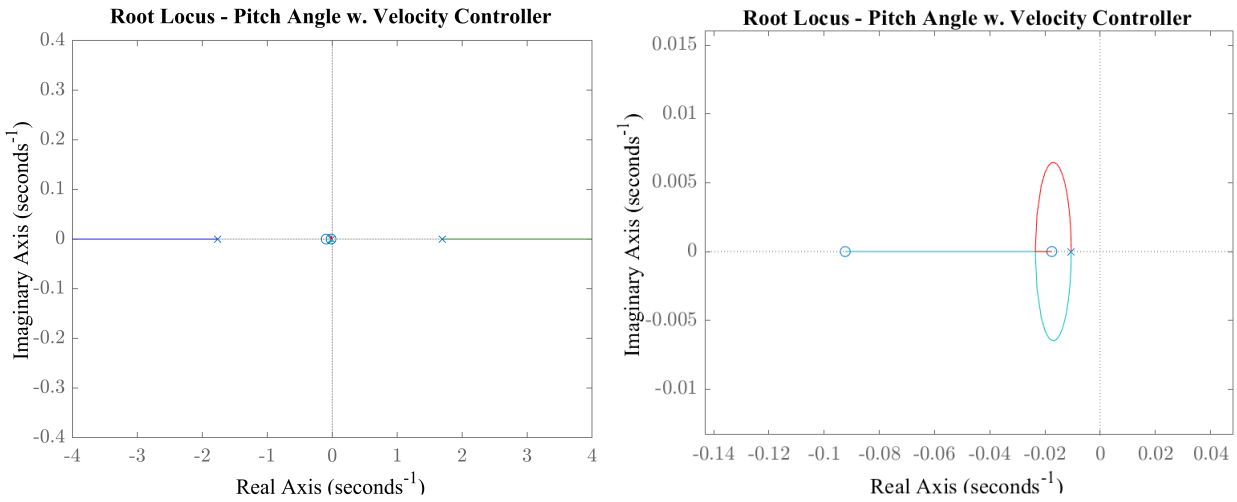


Figure 9: Closed loop root locus for $\frac{\theta(s)}{\delta_e(s)}$ with a proportional controller, when applying said velocity controller.

This transfer function has a negative gain by default. This means we should instead look at a ZARL diagram of the closed loop instead. The closed loop ZARL plot of the system with a proportional controller is shown in figure Fig. 10.

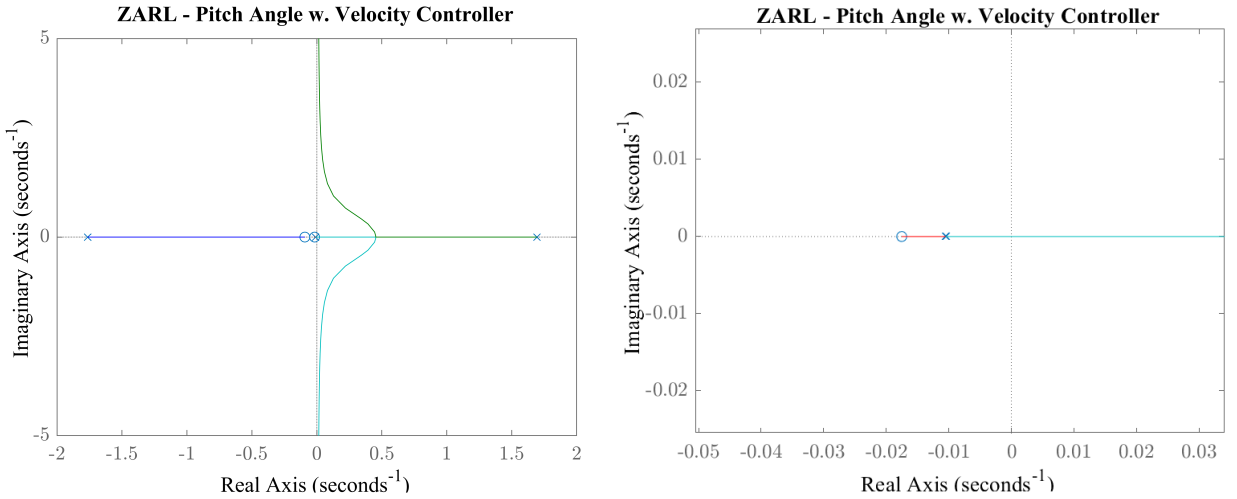


Figure 10: Closed loop ZARL for $\frac{\theta(s)}{\delta_e(s)}$ with a proportional controller, when applying said velocity controller.

These are the loci we are more familiar with for this system. Now we can see that the phugoid poles indeed move leftwards, as expected (since V and θ share the same poles). What we already learned about the system is still valid, and we could try applying a PID controller once again.

A PID controller would add two zeros and one pole, meaning we would expect the asymptote to become horizontal. Since the asymptote is now horizontal, the two rightmost poles (that currently break away into the asymptote) would have to break in somewhere within the left plane where the poles could connect with the zeros. The breakaway is forced because the poles have no other zeros to go to.

Consider the controller

$$C_{PID} = K_P + K_I\frac{1}{s} + K_D s = \frac{K_D s^2 + Kps + K_i}{s} = \frac{K(s+z_1)(s+z_2)}{s}. \tag{19}$$

A ZARL for θ with this controller in a closed loop could be found in Fig. 11. This controller is much better, since the phugoid poles are now always stable, and the new origin pole is drawn to the OLHP alongside the unstable pole. The loci reveal this system now has a threshold gain that guarantees stability (in our case $K \approx 1.35$).

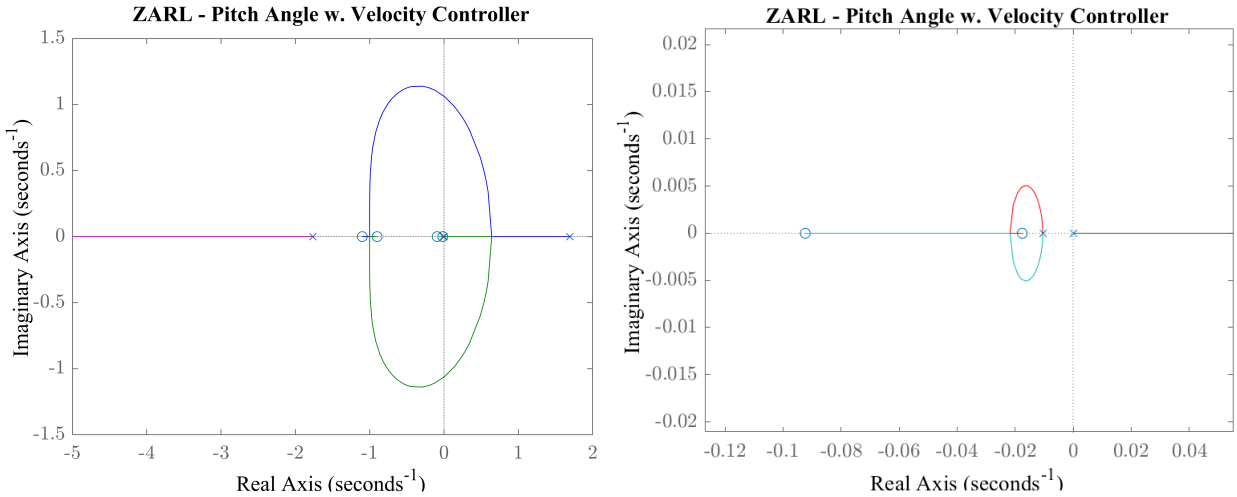


Figure 11: Closed loop ZARL for $\frac{\theta(s)}{\delta_e(s)}$ with a PID controller, when applying said velocity controller.

In fact, we don't really need the integrator and could achieve a very similar result (in terms of stability) using a PD controller. Figure 12 depicts the ZARL of θ in closed loop with the controller

$$C_{PD} = K_P + K_D s = (s + 1.1). \tag{20}$$

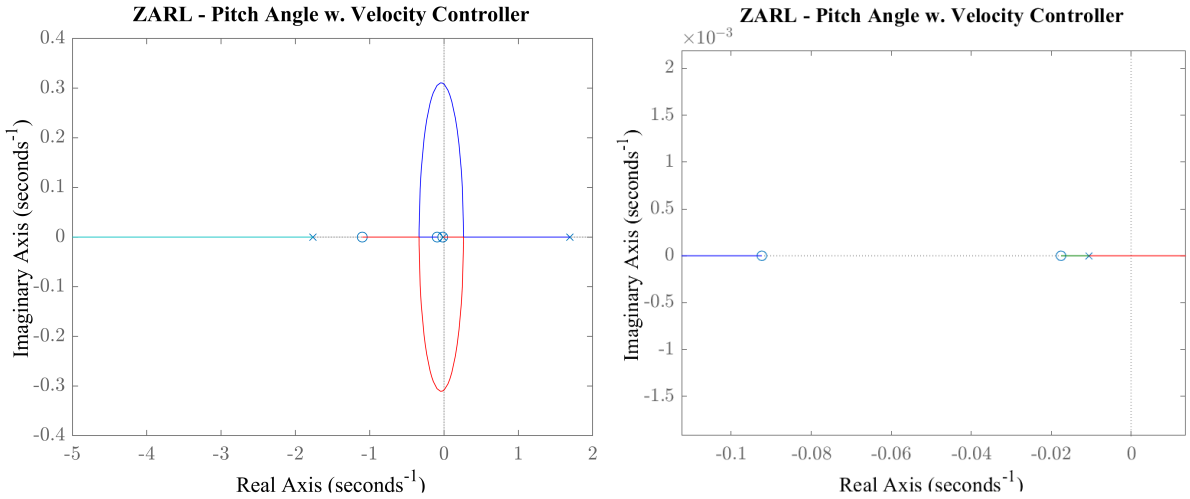


Figure 12: Closed loop ZARL for $\frac{\theta(s)}{\delta_e(s)}$ With a PD controller, when applying said velocity controller.

Further analysis of the system using rltool, with the added PD controller shows that the system can be stabilized. For our purpose, the following controller will be used,

$$C(s) = -5(s+1), \tag{21}$$

which corresponds to the time-domain controller

$$\delta_e = K_Q(-Q + K_\theta(\theta_{com} - \theta)), \tag{22}$$

where $K_Q = -5$, $K_\theta = 1$. Since we are interested in regulating θ, and the linear model has no physical meaning far from trim conditions anyway, we will set θ_{com} to zero. The resulting controller is

$$\delta_e = K_Q(-Q - K_\theta\theta). \tag{23}$$

To summarize, we will apply two controllers, one over V and the other over θ, defined as

$$\delta_e = K_Q(-Q - K_\theta\theta), \quad K_Q = -5, K_\theta = 1, \tag{24a}$$
$$\Phi = -K_V V, \quad K_V = 7.6829 \cdot 10^{-4}. \tag{24b}$$

8 Simulation Results - Multiloop PD Controller

Applying both controllers described in (24), we can simulate the linear and non-linear model's response in different scenarios. Table 4 lists the initial conditions that were simulated.

Figure 13 shows simulation results using the two controllers, where the system is initialized at the trim conditions. Note the controller handles the system well, with the linear simulation steady on the trim condition, while the non-linear system drifts from the trim condition, but is stable, and the deviation is negligible. The control inputs don't deviate too much from nominal values as well.

Table 4: Simulated initial conditions.

State\Input	Value
V_0	7702.0808 [ft/s]
α_0	1.6465°
Q_0	0 [deg/s]
θ_0	1.6465°
h_0	85000 [ft]
$\delta_{e,0}$	12.5447°
Φ_0	0.2682

Additionally, to confirm the stability and ability of the controller to converge, the controller was initiated at near-trim conditions. Results are shown in Fig. 14. The controller handles the deviations well, being capable of regulating the system. Do note the control inputs are quite high, with δ_e reaching values as high as 50°. The deviations from trim conditions (at initialization) are shown in Table 5.

While δ_e exceeds acceptable values, it could be possible to restrict the controller such that convergence may be slower but it will not surpass allowed values. Keep in mind this controller is "unreasonable" as it strives to minimize deviations immediately (since it is a step response), multiple solutions could be proposed to solve this issue, but are irrelevant. Later on, a more advanced controller will be proposed, where these issues could be handled more elegantly with some fine-tuning.

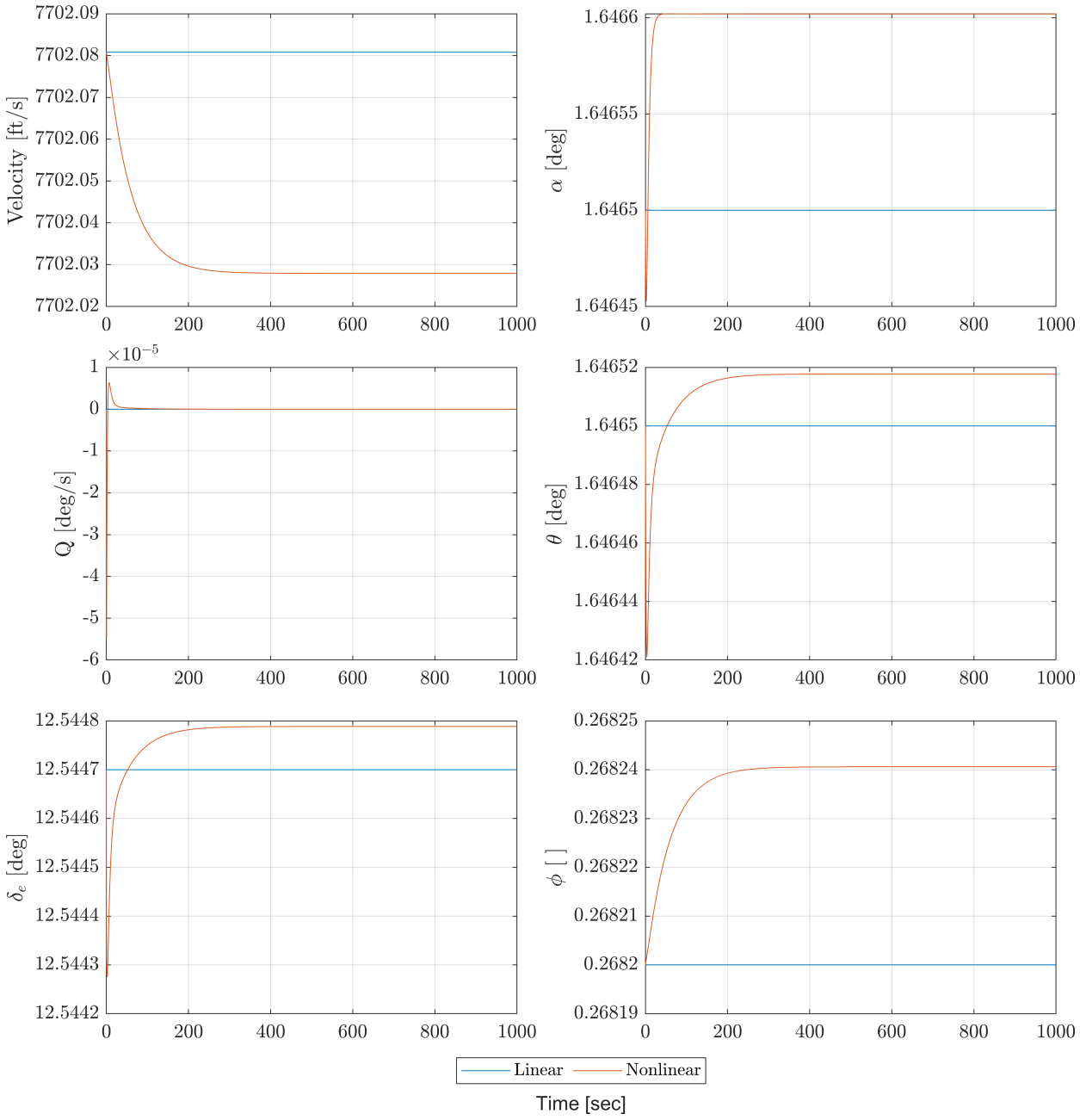


Figure 13: Linear and nonlinear system responses, simulated at trim conditions (as seen in Table 4).

Table 5: Simulated Trim Deviations.

State\Input	Value
ΔV_0	300 [ft/ses]
$\Delta\alpha_0$	5°
ΔQ_0	3 [°/sec]
$\Delta\theta_0$	5°

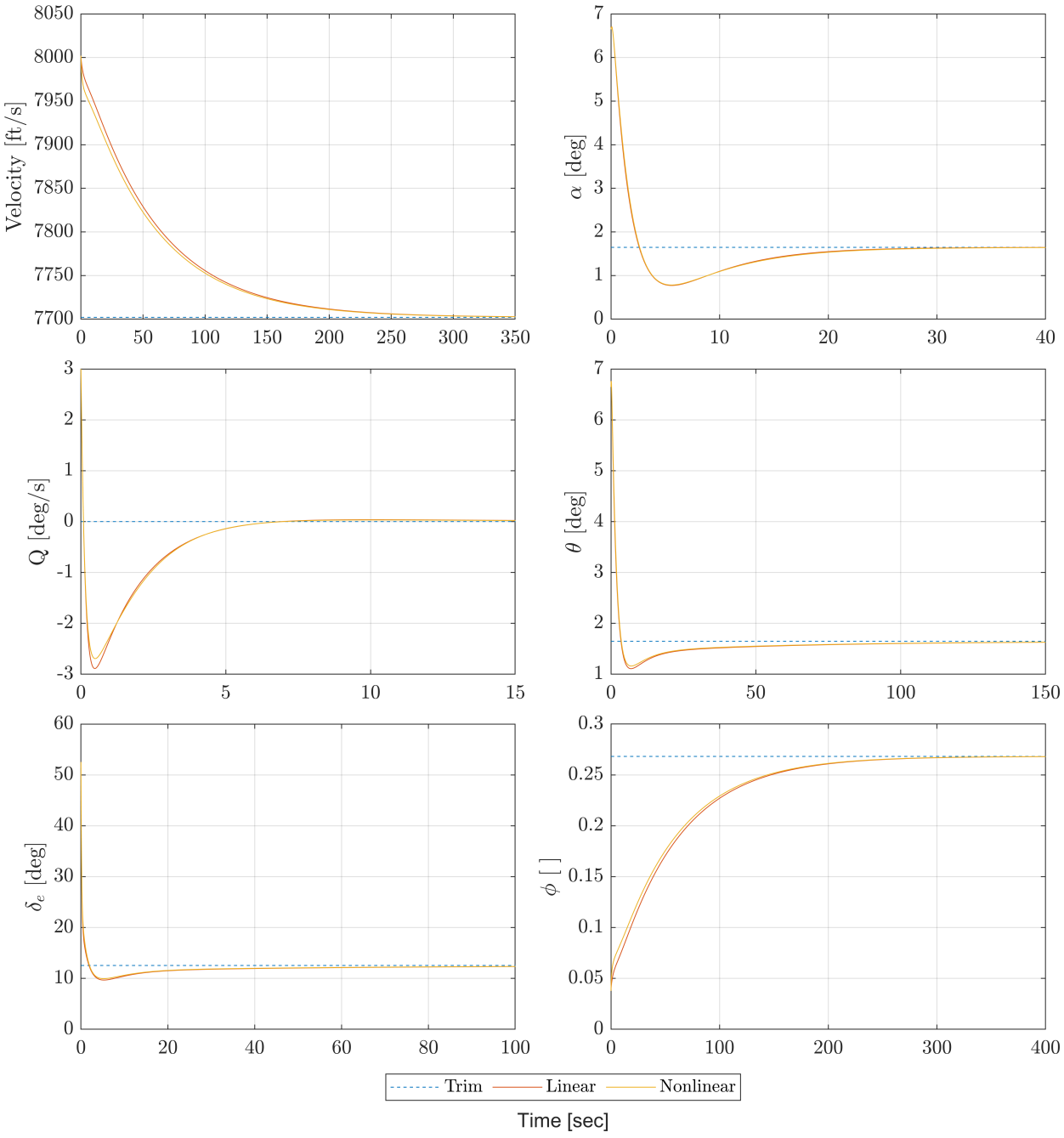


Figure 14: Linear and Non-Linear System Responses, Simulated near Trim Conditions.

9 LQR Controller Design

Consider now a controller designed using LQR. For this purpose we will use the **lqr** method in Matlab, to solve the optimization problem

$$\min_{\vec{x},\vec{u}} \int_0^\infty \left(\vec{x}^T Q \vec{x} + \vec{u}^T R \vec{u} + 2\vec{x}^T N \vec{u}\right) dt, \tag{25}$$

that according to the theory of LQR controllers is solved using the control input

$$\vec{u} = -K\vec{x}. \tag{26}$$

Here K is defined by the solution of the algebraic Ricati equation

$$\begin{aligned} &A^T P + PA - (PB + N)R^{-1}(B^T P + N^T) + Q = 0, \\ &K = R^{-1}(B^T P + N^T). \end{aligned} \tag{27}$$

Controller application requirements are:

- The system must be controllable.
- Q must be a PSD matrix, and P, R a PD matrix.
- The pair (A, Q) must not contain unobservable states.

First, let us check Controllability using the Kalman Test:

$$Q_0 = \begin{bmatrix} B & AB & A^2B & A^3B \end{bmatrix} \tag{28}$$

Performing the Kalman Test, over Φ and δ_e, we discover that the system is controllable by δ_e, with $|Q_{0,\delta_e}| = -1.33 \neq 0$. For Φ the $|Q_{0,\Phi}| = -0.0001$, which is close to zero but is not zero by definition.

Next, we will select Q to be a simple diagonal matrix, with its members chosen according to Bryson's Law. Since we can't accurately determine the maximum values of all the states, we will use values of the maximum deviations we are willing to test, in this case, the deviations in Table 5 should do well. Consequently,

$$Q = \begin{bmatrix} 1.1111 \cdot 10^{-5} & 0 & 0 & 0 \\ 0 & 131.312 & 0 & 0 \\ 0 & 0 & 364.756 & 0 \\ 0 & 0 & 0 & 131.312 \end{bmatrix}. \tag{29}$$

This matrix is indeed PSD. For R we can see that a good value for Φ can be 0.3, where the weight for δ_e will be 5° since we know from the PD controller it tends to have high values, and it wouldn't make sense to apply high deflection angles at hypersonic speeds. These values produce

$$R = \begin{bmatrix} 131.31 & 0 \\ 0 & 11.111 \end{bmatrix} \tag{30}$$

As for N, we will simply use 0 since we have no interest in the mixed weight of x and u. Solving for the given weights, we get the following results,

$$K = \begin{bmatrix} 3.375 \cdot 10^{-4} & -3.392 & -4.219 & -1.868 \\ 7.749 \cdot 10^{-4} & 0.3788 & 0.9952 & 0.8229 \end{bmatrix}, \tag{31}$$

$$P = \begin{bmatrix} 4.385 \cdot 10^{-4} & -0.1793 & -0.01806 & 0.1419 \\ -0.1793 & 1196.023 & 70.596 & -994.151 \\ -0.01806 & 70.596 & 92.868 & 44.750 \\ 0.1419 & -994.151 & 44.750 & 1261.522 \end{bmatrix}. \tag{32}$$

P is indeed PD.

10 Simulation Results - LQR

As before, the controller is simulated for two scenarios - exactly at the trim condition, and offset from it. Tables 4 and 5 describe the tested trim conditions and deviations from them. Figure 15 shows simulation results using the proposed LQR controller when using the trim conditions as the initial condition. Figure 16 shows the same simulation, with the initial conditions skewed by the rates described in Table 5.

The LQR controller too, is capable of regulating the system. Note that when compared to the PD controller, the LQR controller is slightly slower, but uses about 15° less in δ_e. This is a great improvement in terms of applicability, and having tuned the controller we could probably improve upon this result. With that aside, the controller handles the system in a similar manner to the PD controller presented earlier. As previously seen, the linear and nonlinear (CFM) models react similarly, meaning the linear approximation is rather accurate at these deviations.

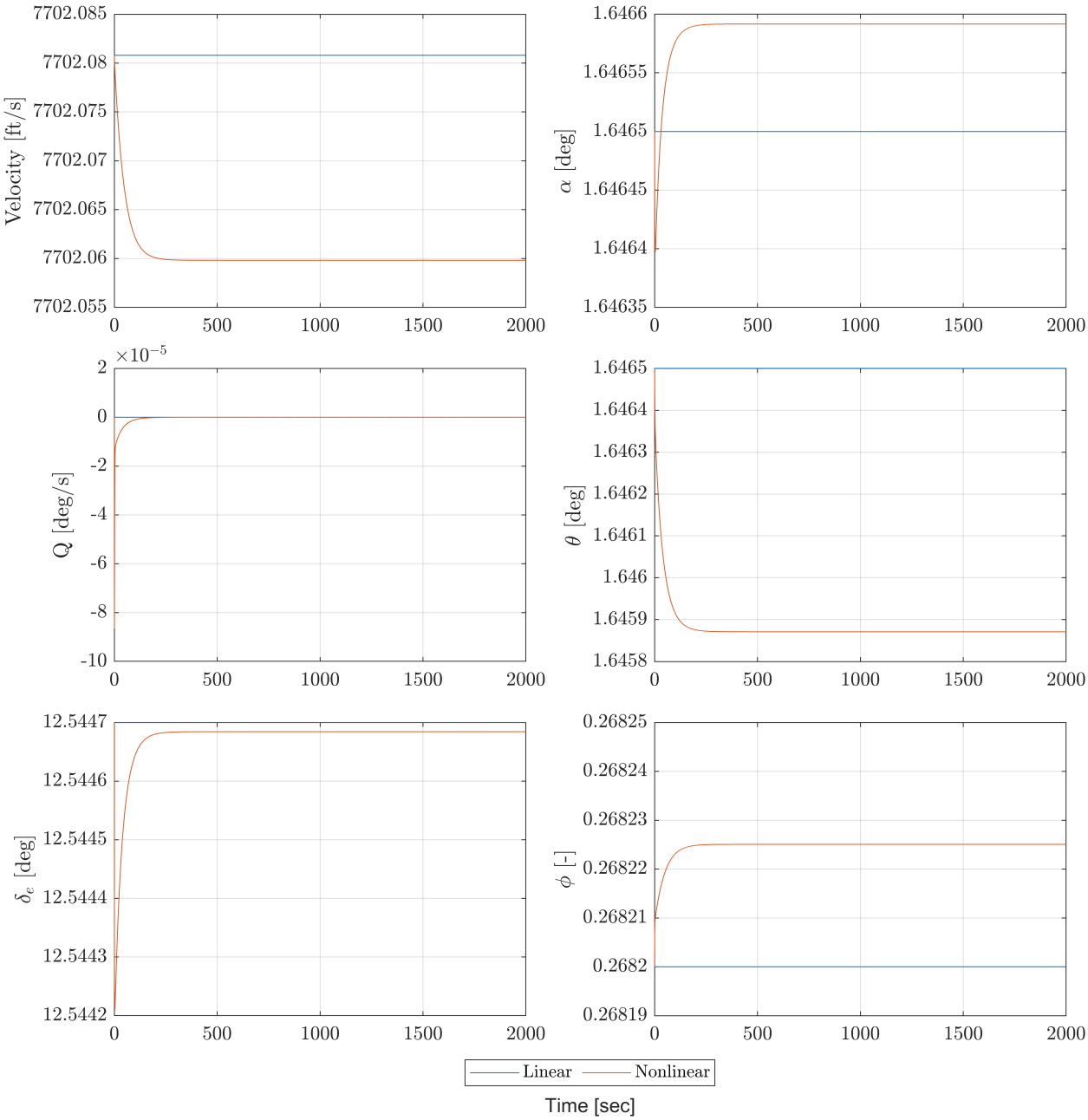


Figure 15: Linear and nonlinear system responses, simulated exactly at the trim conditions, using the LQR controller.

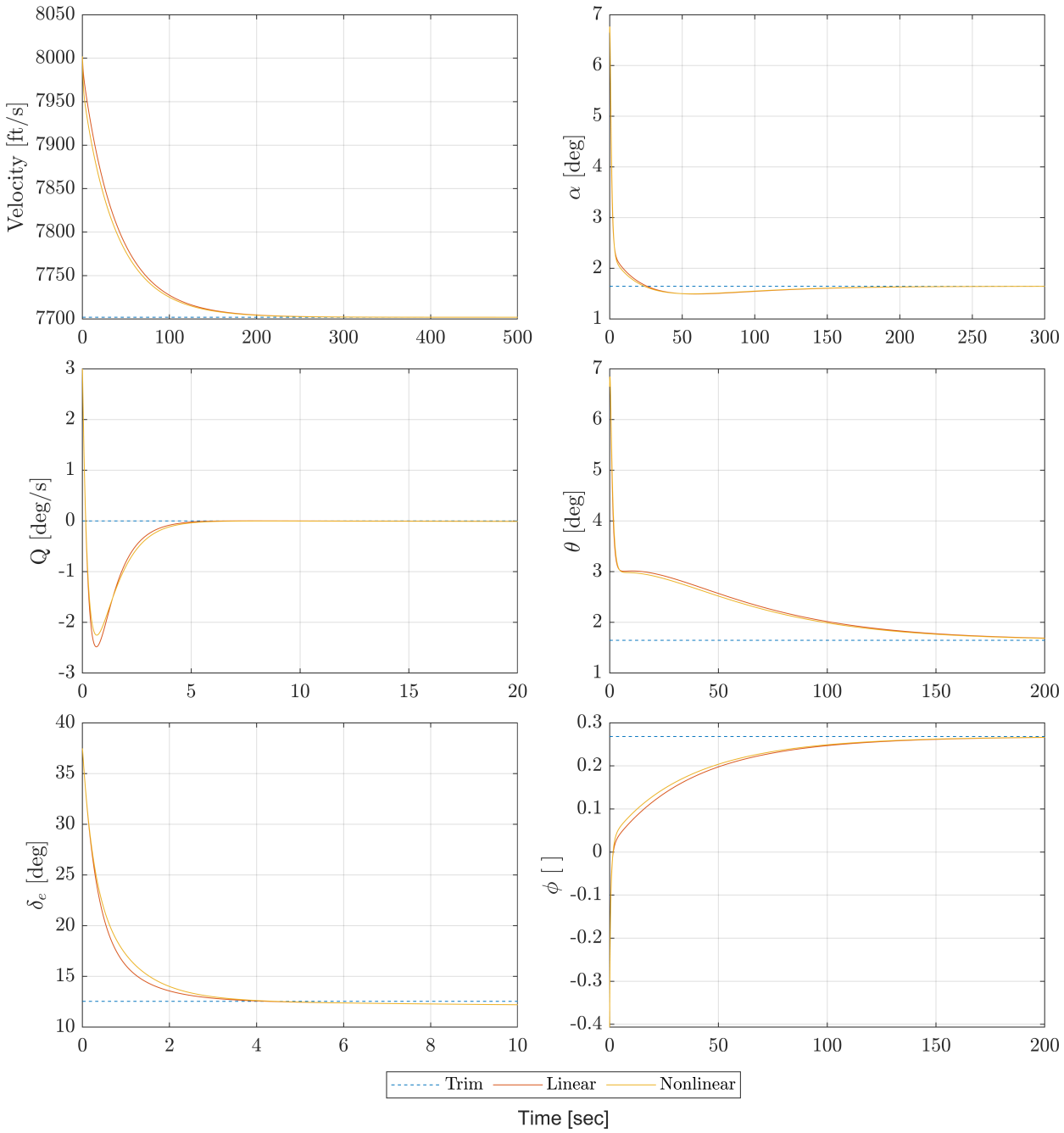


Figure 16: Linear and nonlinear system responses, simulated near the trim conditions, using the LQR controller.

11 Normal Form Model

The remaining sections of this project will focus on recreating the feedback linearization controller developed in [1]. This particular section will detail the model's transformation to normal form, which is made of two integration chains with two control inputs (Φ_c and δ_e). This transformation is advantageous since it will ease the application of dynamic inverse over the model.

With the addition of a dynamic extension (on Φ) to the COM, it consists of a space state with 6 states. The relative degree is 6, this is partly because weak elevator couplings (namely $C_L^{\delta_e}$, $C_D^{\delta_e}$, $C_D^{\delta_e^2}$) were previously neglected. With the dynamic extension in place, (3) effectively becomes

$$\vec{x} = \begin{Bmatrix} V & \alpha & Q & \theta & \Phi & \dot{\Phi} \end{Bmatrix}^T, \quad \vec{u} = \begin{Bmatrix} \delta_e & \Phi_c \end{Bmatrix}^T, \tag{33a}$$

$$\vec{f}(\vec{x}) = \begin{Bmatrix} f_1(\vec{x}) \\ f_2(\vec{x}) \\ f_3(\vec{x}) \\ f_4(\vec{x}) \\ f_5(\vec{x}) \\ f_6(\vec{x}) \end{Bmatrix} = \begin{Bmatrix} \dot{V} \\ \dot{\alpha} \\ \dot{Q} \\ \dot{\theta} \\ \dot{\Phi} \\ \ddot{\Phi} \end{Bmatrix} = \begin{Bmatrix} \frac{1}{m}\Big(T\cos(\alpha) - D\Big) - g\sin(\theta - \alpha) \\ \frac{1}{mV}\Big(-T\sin(\alpha) - L\Big) + Q + \frac{g}{V}\cos(\theta - \alpha) \\ \frac{M}{I_{yy}} \\ Q \\ \dot{\Phi} \\ -2\zeta\omega\dot{\Phi} - \omega^2\Phi + \omega^2\Phi_c \end{Bmatrix}. \tag{33b}$$

Written in space state, (33) becomes

$$\vec{x} = \begin{Bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{Bmatrix}^T, \quad \vec{u} = \begin{Bmatrix} \delta_e & \Phi_c \end{Bmatrix}^T, \tag{34a}$$

$$\dot{\vec{x}} = \begin{Bmatrix} f_1(\vec{x},\vec{u}) \\ f_2(\vec{x},\vec{u}) \\ f_3(\vec{x},\vec{u}) \\ f_4(\vec{x},\vec{u}) \\ f_5(\vec{x},\vec{u}) \\ f_6(\vec{x},\vec{u}) \end{Bmatrix} = \begin{Bmatrix} \frac{1}{m}\Big(T(\vec{x},\vec{u})\cos(x_2) - D(\vec{x},\vec{u})\Big) - g\sin(x_4 - x_2) \\ \frac{1}{mx_1}\Big(-T(\vec{x},\vec{u})\sin(x_2) - L(\vec{x},\vec{u})\Big) + x_3 + \frac{g}{x_1}\cos(x_4 - x_2) \\ \frac{M(\vec{x},\vec{u})}{I_{yy}} \\ x_3 \\ x_6 \\ -2\zeta\omega x_6 - \omega^2 x_5 + \omega^2\Phi_c \end{Bmatrix}. \tag{34b}$$

For the normal form, the input will be defined as $\vec{u} = \begin{Bmatrix} u_V & u_\gamma \end{Bmatrix}^T$ and the output as $\vec{y} = \begin{Bmatrix} V & \gamma \end{Bmatrix}^T$. We will denote the states of the normal as $\vec{z}$. Let us begin the transformation with the following definitions: $z_1 = V$, $z_3 = \gamma$. The remaining 4 states will be inhibited by the Lie derivatives of V and γ,

$$\vec{z} = \begin{Bmatrix} V & \mathcal{L}_f V & \mathcal{L}_f^2 V & \gamma & \mathcal{L}_f \gamma & \mathcal{L}_f^2 \gamma \end{Bmatrix}^T. \tag{35}$$

Using this definition, and neglecting the weak elevator couplings, allows us to perform feedback linearization on the system, the dynamics of which is

$$\dot{z}_1 = z_2, \qquad \dot{z}_2 = z_3, \qquad \dot{z}_3 = \mathcal{L}_f^3 V + \left(\mathcal{L}_{\Phi_c}\mathcal{L}_f^2 V\right)\Phi_c + \left(\mathcal{L}_{\delta_e}\mathcal{L}_f^2 V\right)\delta_e, \tag{36a}$$

$$\dot{z}_4 = z_5, \qquad \dot{z}_5 = z_6, \qquad \dot{z}_6 = \mathcal{L}_f^3 \gamma + \left(\mathcal{L}_{\Phi_c}\mathcal{L}_f^2 \gamma\right)\Phi_c + \left(\mathcal{L}_{\delta_e}\mathcal{L}_f^2 \gamma\right)\delta_e, \tag{36b}$$

i.e., it is in normal form. The notation $\mathcal{L}_{u_i}\mathcal{L}_f^2 V$ (u_i is either Φ_c or δ_e) refers to the coefficients of the control inputs that are obesered in the higher derivatives. The derivation is done using a symbolic Matlab code. To verify its results, we will derive $\mathcal{L}_f^2 V$ manually and compare it with results from the code,

$$\mathcal{L}_f V = \frac{dV}{dt}\vec{f}(\vec{x},\vec{u}) = f_1(\vec{x}) = \frac{1}{m}(T(\vec{x},\vec{u})\cos(x_2) - D(\vec{x},\vec{u})) - g\sin(x_4 - x_2),$$

$$\mathcal{L}_f^2 V = \frac{d\mathcal{L}_f V}{dt} = \frac{\partial \mathcal{L}_f V}{\partial \vec{x}}\vec{f}(\vec{x},\vec{u}) =$$

$$= \begin{Bmatrix} -\frac{1}{m}\rho V S C_D(\alpha) \\ -\frac{1}{m}\rho V^2 S\frac{dC_D}{d\alpha} + g\cos(\theta-\alpha) \\ 0 \\ -g\cos(\theta) \\ \alpha^3\beta_1 + \alpha^2\beta_3 + \alpha\beta_5 + \beta_7 \\ 0 \end{Bmatrix}^T \begin{Bmatrix} \frac{1}{m}\left(T(\vec{x},\vec{u})\cos(x_2) - D(\vec{x},\vec{u})\right) - g\sin(x_4 - x_2) \\ \cdots \\ \cdots \\ \cdots \\ \cdots \\ \cdots \end{Bmatrix}. \tag{37}$$

Making the full calculation would be incredibly long, so, we will only look at the coefficient of f_1 (the first row in (33)). The coefficient of this function according to Matlab should amount to

$$3.8209 \cdot 10^{-6}\, x_1 \left(5.8224\, x_2^2 - 0.0453\, x_2 + 0.0101\right). \tag{38}$$

Note we can see the exact quantities of the model, where x_1 is V and x_2 is α. We easily verify the coefficients by hand, according to the manual derivation of the lie derivative. The actual coefficient should be $-\frac{1}{m}\rho V S C_D(\alpha)$, expanding the coefficient of drag, and substituting values according to the known model as listed in Appendix A, we get

$$\begin{aligned} &\frac{1}{m}\rho S = 3.8209 \cdot 10^{-6}, \\ &C_D^{\alpha^2} = 5.8224, \\ &C_D^{\alpha} = -4.5315 \cdot 10^{-2}, \\ &C_D^{0} = 1.0131 \cdot 10^{-2}. \end{aligned} \tag{39}$$

This validates our normal form model. We can also verify that the derivatives of z_1, z_2, z_4, and z_5 are as stated in (36). This implies a relative degree of 3 (with respect to both inputs) and corresponds to results shown in [1].

12 Feedback Linearization LQ Controller

This section will focus on the implementation of feedback linearization and an LQ controller for the normal-form model. First, feedback linearization is applied to eliminate the non-linearity of the higher derivatives; this will be the inner loop controller. Then, a standard LQ controller serves as the outer loop controller for the now linear system. The complete control scheme is presented in Fig. 17.

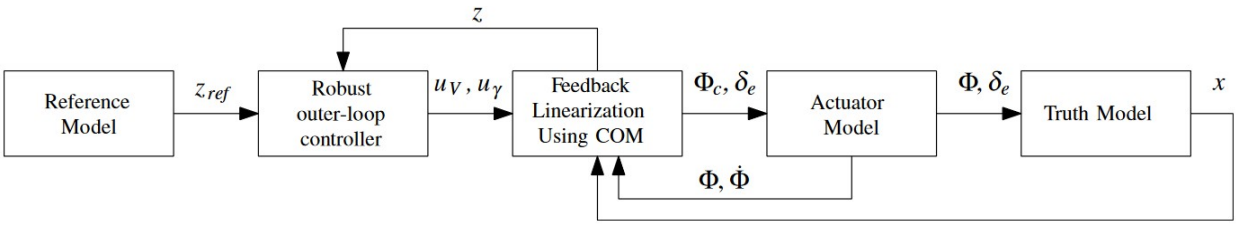


Figure 17: A conceptual block diagram of the controller, taken from [1].

With the system's normal form representation, we have a non-linear dynamic system, formed by two decoupled integration chains. The relative degree is 3, and the system's non-linearity originates from the higher derivatives. Feedback linearization is applicable for this structure, utilizing dynamic inversion.

We are interested in $\dot{z}_3$ and $\dot{z}_6$ for feedback linearization. Specifically, the ideal form of these derivatives would be

$$\begin{Bmatrix} \dot{z}_3 \\ \dot{z}_6 \end{Bmatrix} = \begin{Bmatrix} u_V \\ u_\gamma \end{Bmatrix}. \tag{40}$$

Rewriting (36), we get a matrix representation of the model,

$$\begin{Bmatrix} \dot{z}_3 \\ \dot{z}_6 \end{Bmatrix} = \begin{bmatrix} \mathcal{L}_{\delta_e}\mathcal{L}_f^2 V & \mathcal{L}_{\Phi_c}\mathcal{L}_f^2 V \\ \mathcal{L}_{\delta_e}\mathcal{L}_f^2 \gamma & \mathcal{L}_{\Phi_c}\mathcal{L}_f^2 \gamma \end{bmatrix} \begin{Bmatrix} \delta_e \\ \Phi_c \end{Bmatrix} + \begin{Bmatrix} \mathcal{L}_f^3 V \\ \mathcal{L}_f^3 \gamma \end{Bmatrix}. \tag{41}$$

To get the form seen in (40), we dictate the inner loop controller as

$$\begin{Bmatrix} \delta_e \\ \Phi_c \end{Bmatrix} = A_c(\vec{x})^{-1} \begin{Bmatrix} u_V - \mathcal{L}_f^3 V \\ u_\gamma - \mathcal{L}_f^3 \gamma \end{Bmatrix}, \quad A_c(\vec{x}) = \begin{bmatrix} \mathcal{L}_{\delta_e}\mathcal{L}_f^2 V & \mathcal{L}_{\Phi_c}\mathcal{L}_f^2 V \\ \mathcal{L}_{\delta_e}\mathcal{L}_f^2 \gamma & \mathcal{L}_{\Phi_c}\mathcal{L}_f^2 \gamma \end{bmatrix}. \tag{42}$$

This is applicable only if the decoupling matrix A_c is nonsingular. For this reason, the singularity of A_c was tested along a range of relevant values. It turns out that A_c depends only on states x_1, x_2 and x_5. A_c also depends on the height h, which is not a state, but should be considered. Hence, 10 grids were created (for 10 different values of h), where each grid contains test values for x_1, x_2, and x_5. The tested values for each state are listed in Table 6. Figures 18 and 19 depicts how the maximum condition number and minimum determinant of A_c (over the range of tested values) change with height.

The condition number for inversion[2] measures the sensitivity of the solution proposed in (42) to numerical estimates, when this metric approaches infinity, it could be a sign that A_C is singular. Thus, this number is evaluated for A_c. Since the condition number is of a reasonable order of magnitude ($1{\cdot}10^6$), and the minimum determinant of A_c doesn't approach 0 (even at high altitude), we can safely assume A_c is nonsingular within the range of operation we are interested in.

[2]There are many condition numbers, we are interested in the one for inversion. For more information, read about the *cond* function in Matlab.

Table 6: Relevant range of state values, where non-singularity was tested. Values are based on a range of operations suggested in [1] (for the CFM).

	Minimum Value	Maximum Value	Interval	Number of Points
h [kft]	75	95	2.222e3	10
V (x_1) [ft/s]	4000	10000	60.2020	100
α (x_2) [deg]	-10	10	0.0035	100
Φ (x_5)	0.1	1.2	0.0111	100

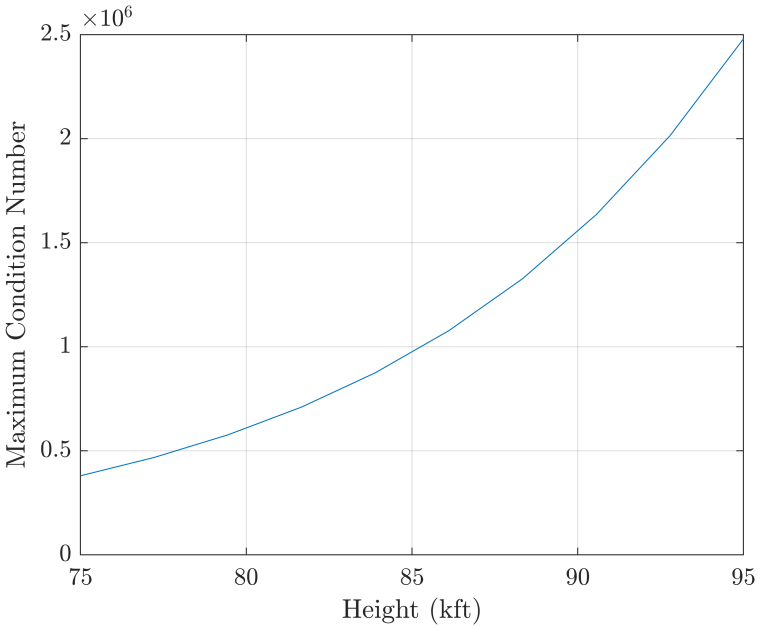


Figure 18: The maximum condition number of A_c along values tested, for different values of h.

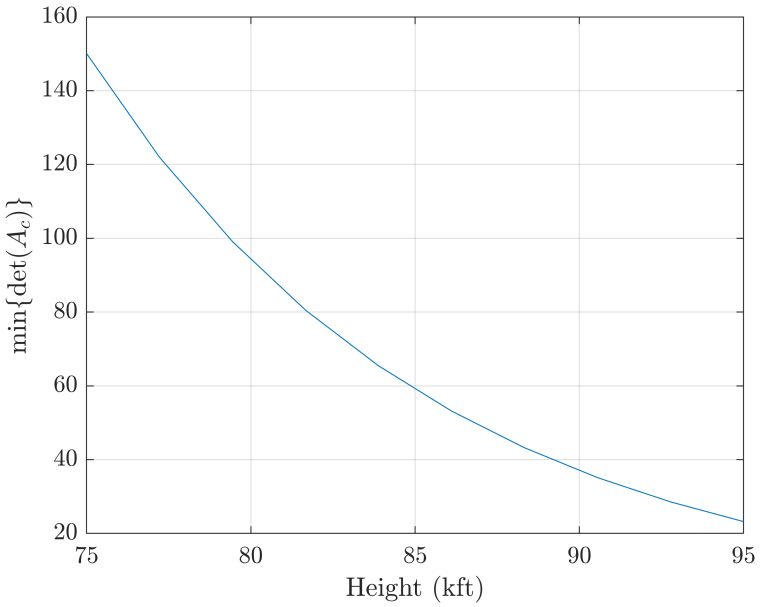


Figure 19: The minimum of $\det(A_c)$ along values tested, for different values of h.

Using (42) in (36), the system becomes linear with respect to the control inputs u_V and u_γ. Now we can design an outer loop controller using linear methods, in our case, LQR. Integral augmentation is applied before designing the LQR-based outer loop, alongside the control law as defined in (42), restructuring the system into the form

$$\dot{z}_{i,1} = z_1 \qquad \dot{z}_1 = z_2 \qquad \dot{z}_2 = z_3 \qquad \dot{z}_3 = u_V \tag{43a}$$

$$\dot{z}_{i,2} = z_4 \qquad \dot{z}_4 = z_5 \qquad \dot{z}_5 = z_6 \qquad \dot{z}_6 = u_\gamma. \tag{43b}$$

To tackle a tracking problem, a reference model will be presented later. From here on, the notation w will refer to system tracking errors relative to the reference model, such as $w_1 = z_1 - z_{1,ref} = V - V_{ref}$. By definition, the dynamics of the tracking errors still preserve the normal-form properties. For example,

$$\dot{w}_1 = \dot{z}_1 - \dot{z}_{1,ref} = z_2 - z_{2,ref} = w_2. \tag{44}$$

Expanding this logic to the entire system (recall the higher derivatives must be somewhat different due to the reference derivatives), we can express the dynamics of the deviations from the reference model as

$$\dot{w}_{i,1} = w_1 \equiv V - V_{ref} \qquad \dot{w}_{i,2} = w_4 \equiv \gamma - \gamma_{ref} \tag{45a}$$

$$\dot{w}_1 = w_2 \equiv \mathcal{L}_f V - \dot{V}_{ref} \qquad \dot{w}_4 = w_5 \equiv \mathcal{L}_f \gamma - \dot{\gamma}_{ref} \tag{45b}$$

$$\dot{w}_2 = w_3 \equiv \mathcal{L}_f^2 V - \ddot{V}_{ref} \qquad \dot{w}_5 = w_6 \equiv \mathcal{L}_f^2 \gamma - \ddot{\gamma}_{ref} \tag{45c}$$

$$\dot{w}_3 = u_V - \dddot{V}_{ref} \qquad \dot{w}_6 = u_\gamma - \dddot{\gamma}_{ref}. \tag{45d}$$

The decoupling of V and γ allows us to look at the system as two separate systems with 4 states. A design for "both systems" will be presented in a decoupled manner. The LQR weights are chosen as

$$Q_V = \mathrm{diag}(10, 1, 1, 1), \qquad R_V = 1, \tag{46a}$$

$$Q_\gamma = I_{4\times 4}, \qquad R_\gamma = 0.1, \tag{46b}$$

according to the weights presented in [1]. As mentioned in previous sections, the optimal controller for both systems (up to the choice of weight matrices) will be

$$u_V - \dddot{V}_{ref} = -K_V \begin{Bmatrix} w_{i,1} \\ w_1 \\ w_2 \\ w_3 \end{Bmatrix} \quad \Rightarrow \quad u_V = \dddot{V}_{ref} - K_V \begin{Bmatrix} \int_0^t [V(\tau) - V_{ref}(\tau)] d\tau \\ V - V_{ref} \\ \dot{V} - \dot{V}_{ref} \\ \ddot{V} - \ddot{V}_{ref} \end{Bmatrix}, \tag{47a}$$

$$u_\gamma - \dddot{\gamma}_{ref} = -K_\gamma \begin{Bmatrix} w_{i,2} \\ w_4 \\ w_5 \\ w_6 \end{Bmatrix} \quad \Rightarrow \quad u_\gamma = \dddot{\gamma}_{ref} - K_\gamma \begin{Bmatrix} \int_0^t [\gamma(\tau) - \gamma_{ref}(\tau)] d\tau \\ \gamma - \gamma_{ref} \\ \dot{\gamma} - \dot{\gamma}_{ref} \\ \ddot{\gamma} - \ddot{\gamma}_{ref} \end{Bmatrix}. \tag{47b}$$

Before calculating K_V and K_γ, one must make sure the system is controllable. Again, by applying the Kalman Test, we can derive Q_0 and determine controllability,

$$Q_0 = \begin{bmatrix} B & AB & A^2B & A^3B \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}. \tag{48}$$

The controllability matrix has full rank, meaning the system is indeed controllable. Assuming the pair (A, Q) doesn't have any unobservable states (for both γ and V), the LQR design (performed in Matlab using *lqr*) yields

$$K_V = \begin{Bmatrix} 3.162 & 6.568 & 6.663 & 3.784 \end{Bmatrix}, \qquad K_\gamma = \begin{Bmatrix} 3.162 & 8.482 & 9.796 & 5.439 \end{Bmatrix}, \tag{49a}$$

$$S_V = \begin{bmatrix} 20.770 & 21.070 & 11.969 & 3.162 \\ 21.070 & 31.794 & 21.698 & 6.568 \\ 11.969 & 21.698 & 18.651 & 6.663 \\ 3.162 & 6.568 & 6.663 & 3.784 \end{bmatrix}, \qquad S_\gamma = \begin{bmatrix} 2.682 & 3.097 & 1.720 & 0.316 \\ 3.097 & 6.589 & 4.298 & 0.848 \\ 1.720 & 4.298 & 4.480 & 0.979 \\ 0.316 & 0.848 & 0.979 & 0.543 \end{bmatrix}. \tag{49b}$$

Both S_V and S_γ are PD matrices. The closed-loop poles of the linearized system with the above control gains are

$$P_V = \begin{Bmatrix} -0.598 + 1.138j \\ -0.598 - 1.138j \\ -1.294 + 0.487j \\ -1.294 - 0.487j \end{Bmatrix}, \qquad P_\gamma = \begin{Bmatrix} -0.705 + 0.725j \\ -0.705 - 0.725j \\ -1.029 + 0.000j \\ -2.999 + 0.000j \end{Bmatrix}. \tag{50}$$

13 Simulation Results - Feedback Linearization LQ Controller

Following the LQR design presented earlier, we must propose a reference model before assessing controller performance. The maneuver used as a reference starts at the trim point: altitude of 85 [kft] and speed of 7702.0808 [ft/s]. The AHV will accelerate at 10 [ft/s^2], to a speed of 8500 [ft/s], and simultaneously climb at an angle of $\gamma = 0.3°$. Upon reaching an altitude of 90 [kft], the AHV will level and maintain speed. The reference model is a corresponding step command filtered through a unity DC gain low-pass filter. The filter's order is chosen to be 5 to ensure all control inputs are smooth, namely, the 4$^{\text{th}}$ state of the filter, i.e., the third derivative of the output, remains smooth regardless of the input. The reference model filter is detailed in Appendix C.

First, a regulation command is given to the controller (initialized at the trim condition). Figures 20 to 23 depict the response of the closed-loop system to a regulation command. Shown are, respectively: state responses, control inputs, absolute errors, and a comparison with the trim condition.

Notice there is some transient response before settling back to the trim condition. The reason for this result is that the control inputs aren't initialized to match the trim condition (specifically δ_e), hence a short transient occurs, after which the controller converges on the trim condition. This problem could be solved in a few ways, initializing the integral augmentation states to some value that can be calculated and will bring δ_e to trim at the first time step of the simulation, or just ignoring the transient and initiating the reference model after the system settles. For this work, we will initialize the reference model at $t = 30$ [sec] so the transient won't interfere with tracking performance.

Overall, regulation performance is good, the absolute errors seen in Fig. 22 are quite negligible, with the velocity error peaking at ≈ 8 [ft/s] and the trajectory angle error peaking at $\approx 0.17°$. It is worth mentioning that due to the transient response, the AHV converges to a level flight ≈ 56 [ft] higher than the trim condition. This doesn't contradict the control design, since the altitude is not controlled actively. Control inputs are well-behaved, and are within a reasonable range of operation. Keep in mind that according to [1], the CFM is

relevant only while $\delta_e \in [-15°, 15°]$, meaning we are slightly violating the appropriate range of δ_e, but in comparison to the performance shown by the other controllers in this paper, it is the best result we can achieve[3].

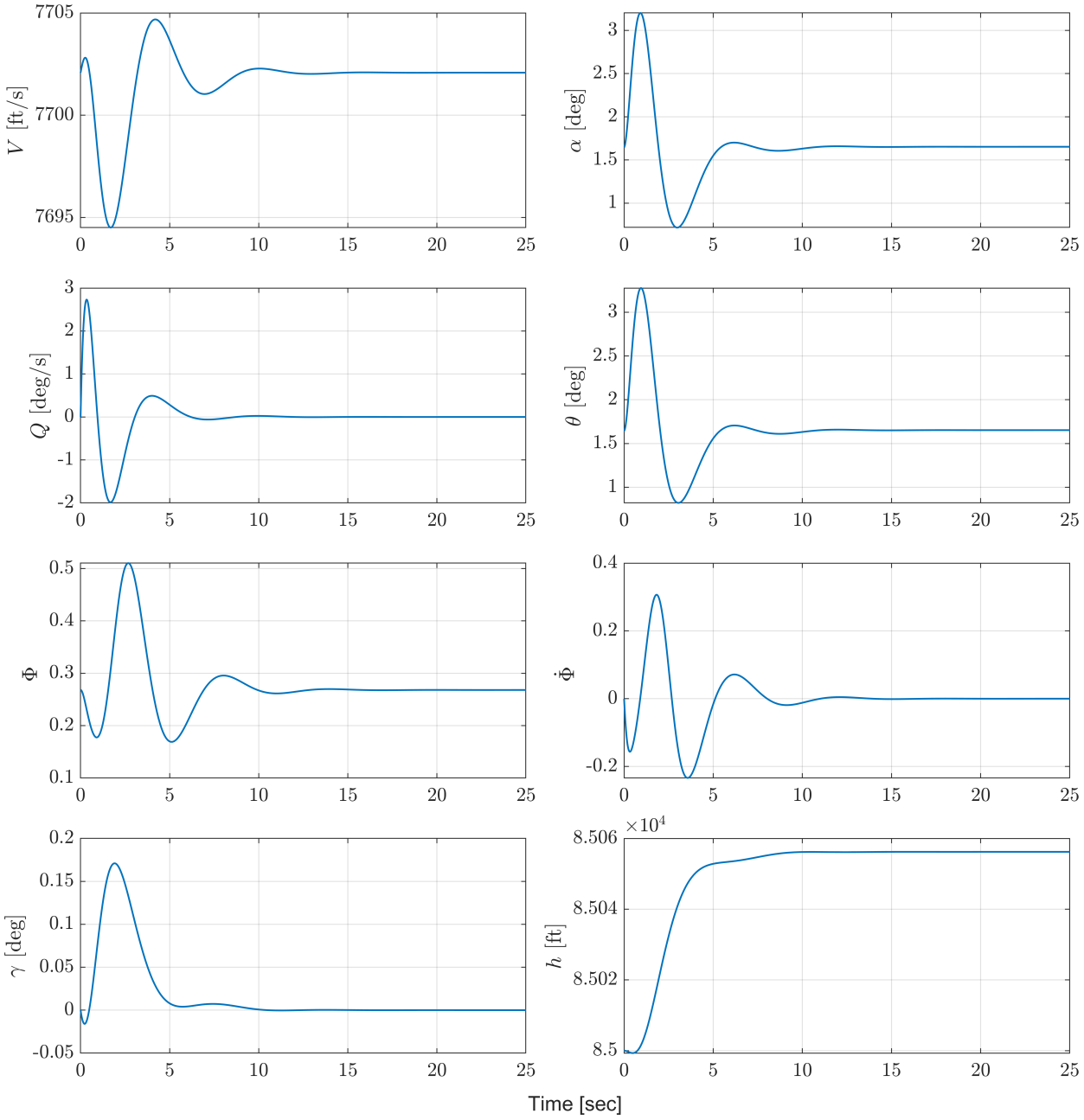


Figure 20: State responses of the system to a regulation command - V, α, Q, θ, Φ, $\dot{\Phi}$, γ, and h.

Figures 24 to 28 present the tracking results. Shown are, respectively: the reference model (as defined earlier), tracking response, control inputs, absolute tracking errors, and a comparison between the response and reference signals.

Here too, we note the initial transient, but, as seen in Fig. 25, the system has enough time to settle before issuing the tracking command. Note that when the engine is commanded to maintain speed, we have a variation in γ that is quickly returned to its commanded value. As for the rest of the states, there are not many interesting conclusions to draw. Looking at

[3]Different requirements, such as tracking and regulation will have major differences, and even regulating different states could yield different results, hence we might see larger violations in δ_e later.

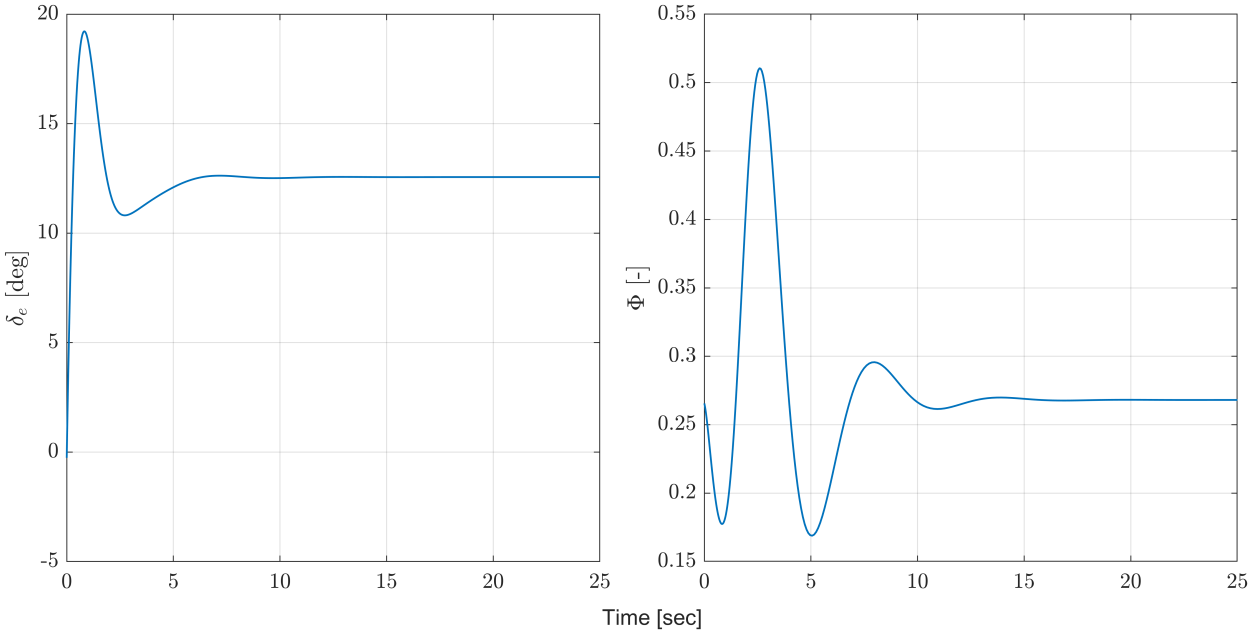


Figure 21: Control inputs for a regulation command.

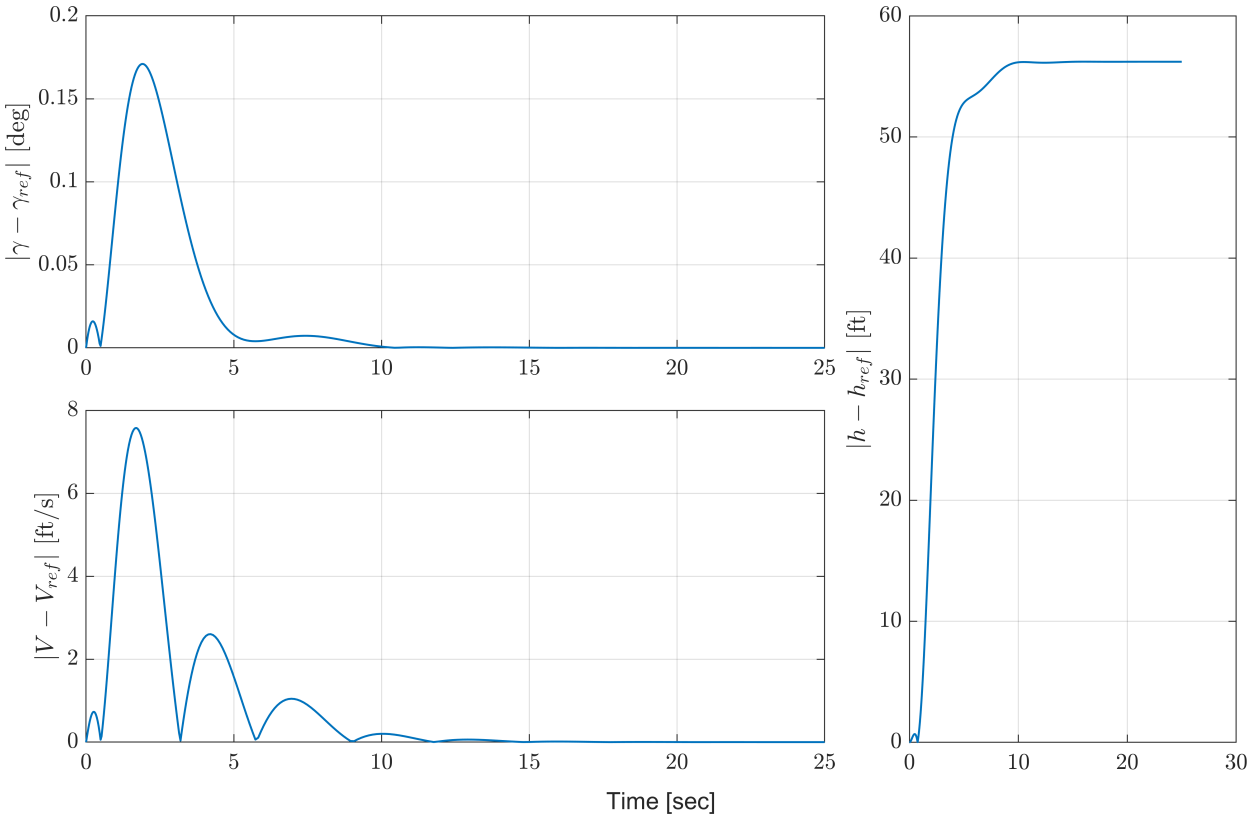


Figure 22: Absolute errors of h, V and γ from the trim state for a regulation command.

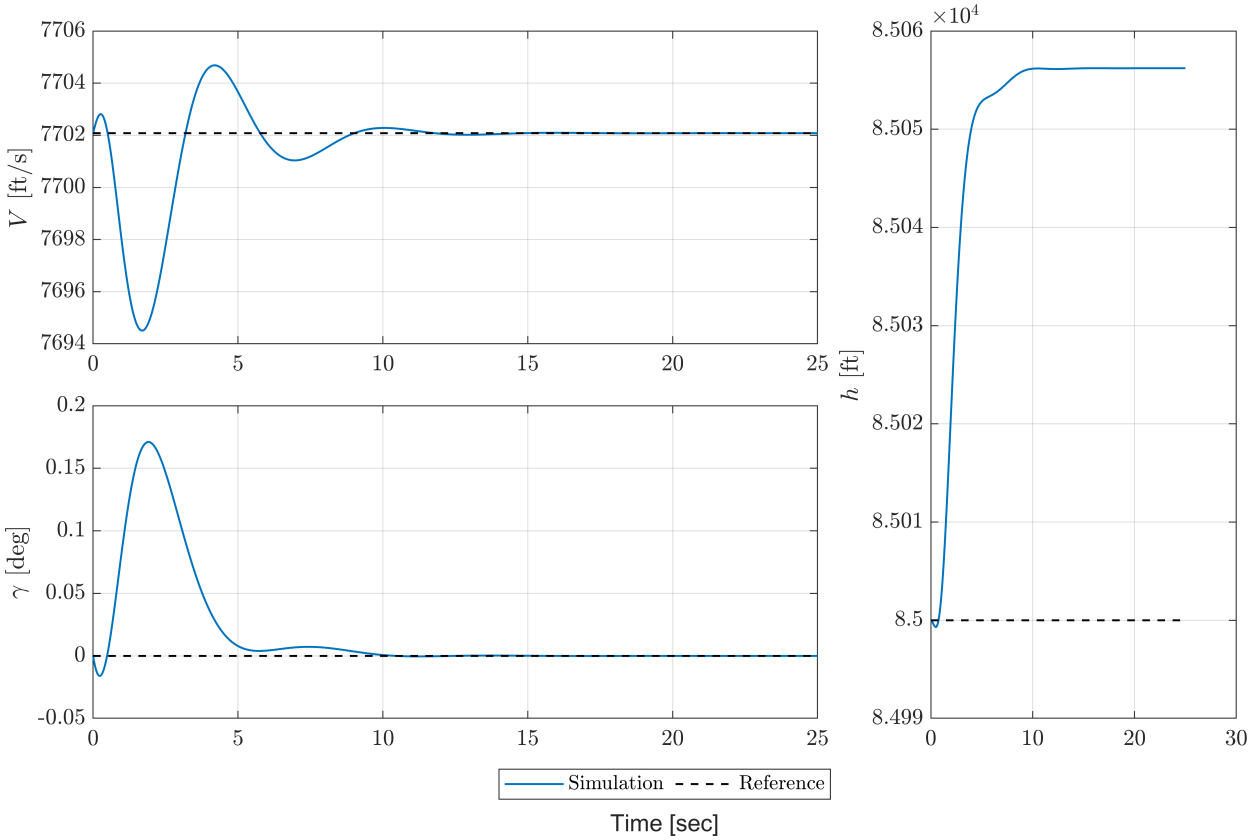


Figure 23: Comparison of h, V and γ with their respective trim values to maintain.

the control inputs, one might notice that we are on the verge of violating the limit of δ_e at around $t = 35$ [sec] (besides the peak introduced during the initial transient response). As for Φ, the input is within the relevant range specified in [1] (which is $\Phi \in [0.1, 1.2]$).

Tracking errors (in their absolute values) indicate prominent performance, with the peak error in γ at $\approx 0.03^\circ$ (which is about 10% from γ_{max}). The maximum error in V is around 2 [ft/s], which is minuscule compared to the velocities we try to achieve. As mentioned before, the altitude is not actively controlled and reaches about 160 [ft] above its initial value.

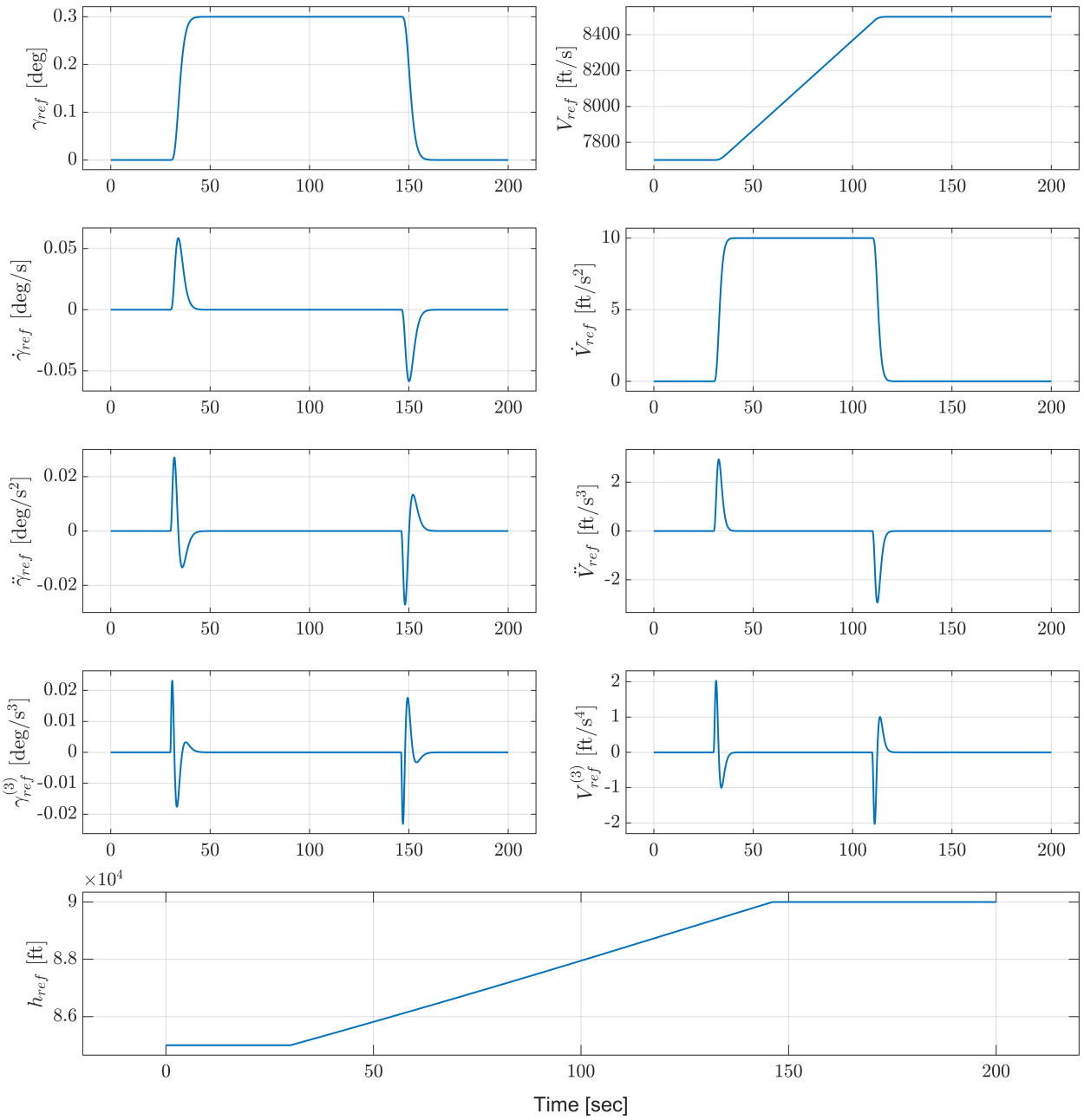


Figure 24: The reference model - V, γ, h, and the relevant derivatives (after passing the filter).

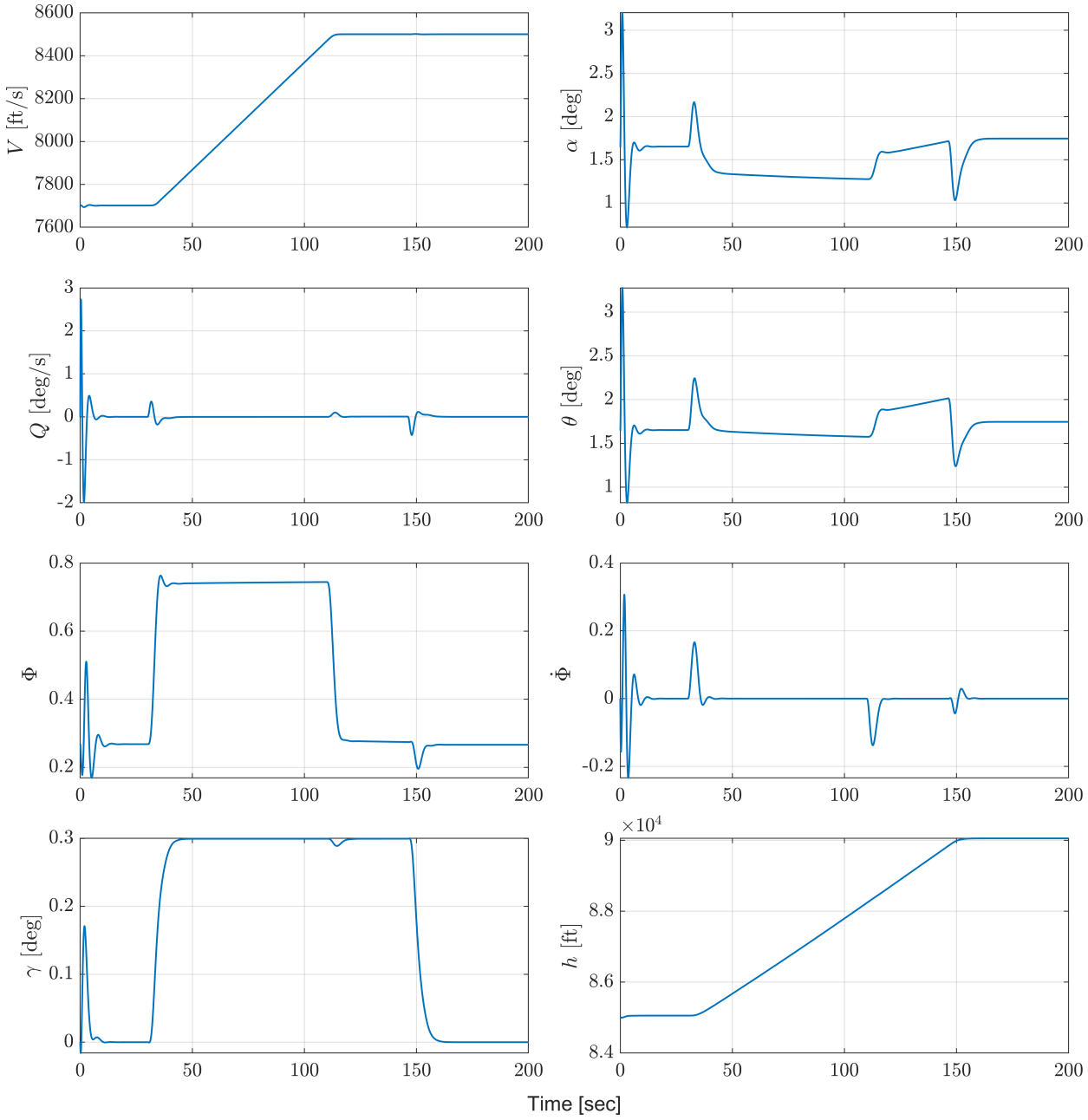


Figure 25: State responses of the system to a tracking command (reference model) - V, α, Q, θ, Φ, $\dot{\Phi}$, γ, and h.

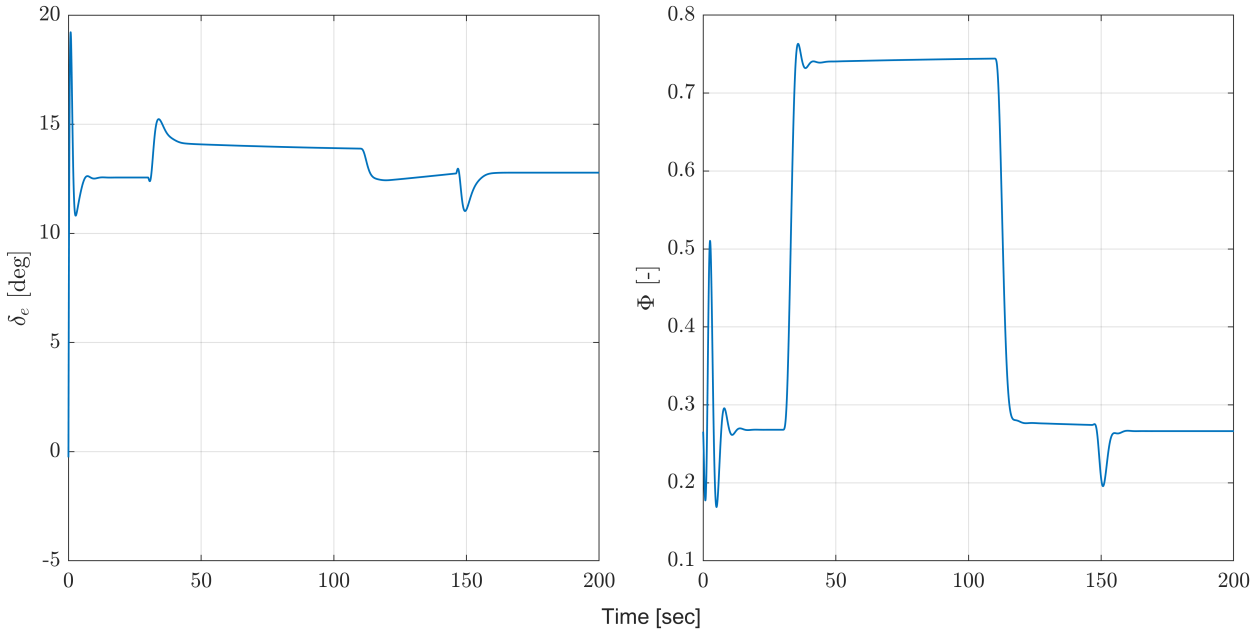


Figure 26: Control inputs for a tracking command.

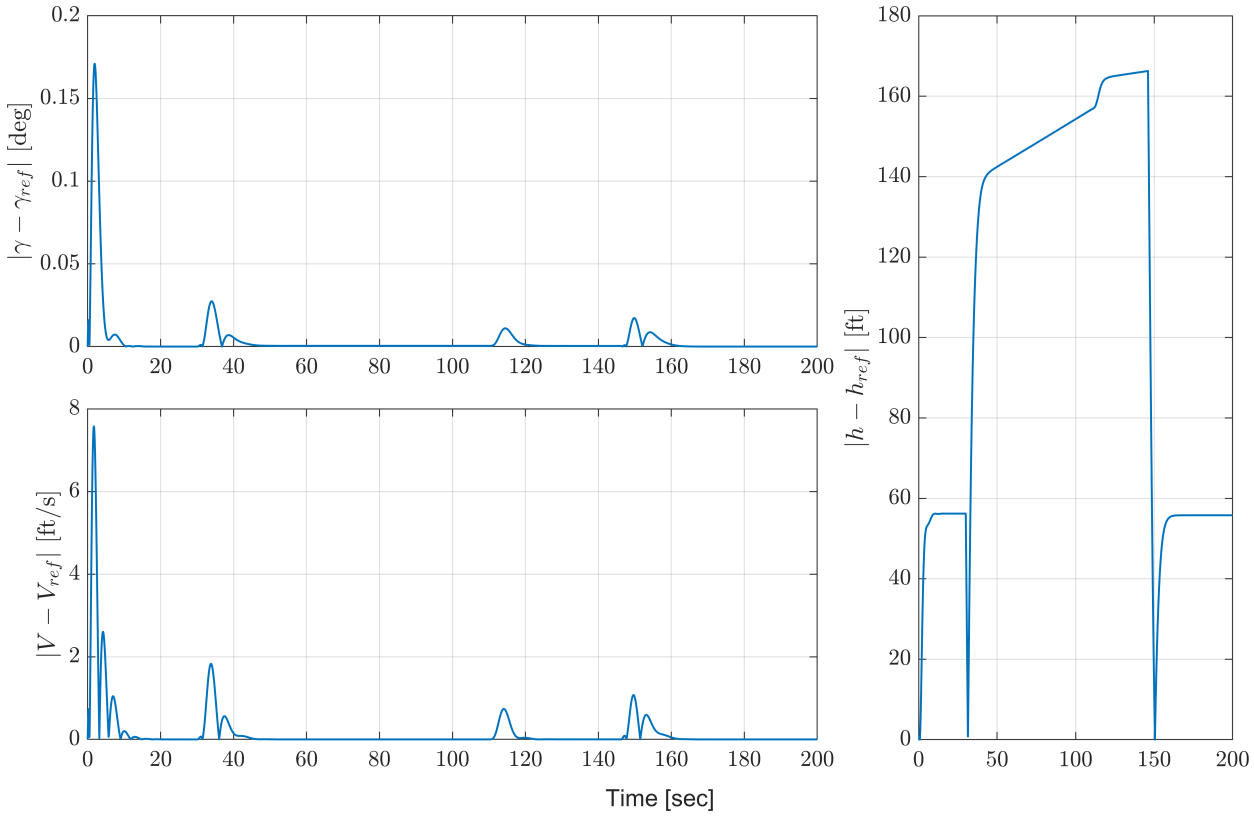


Figure 27: Absolute errors of h, V and γ when tracking the reference model.

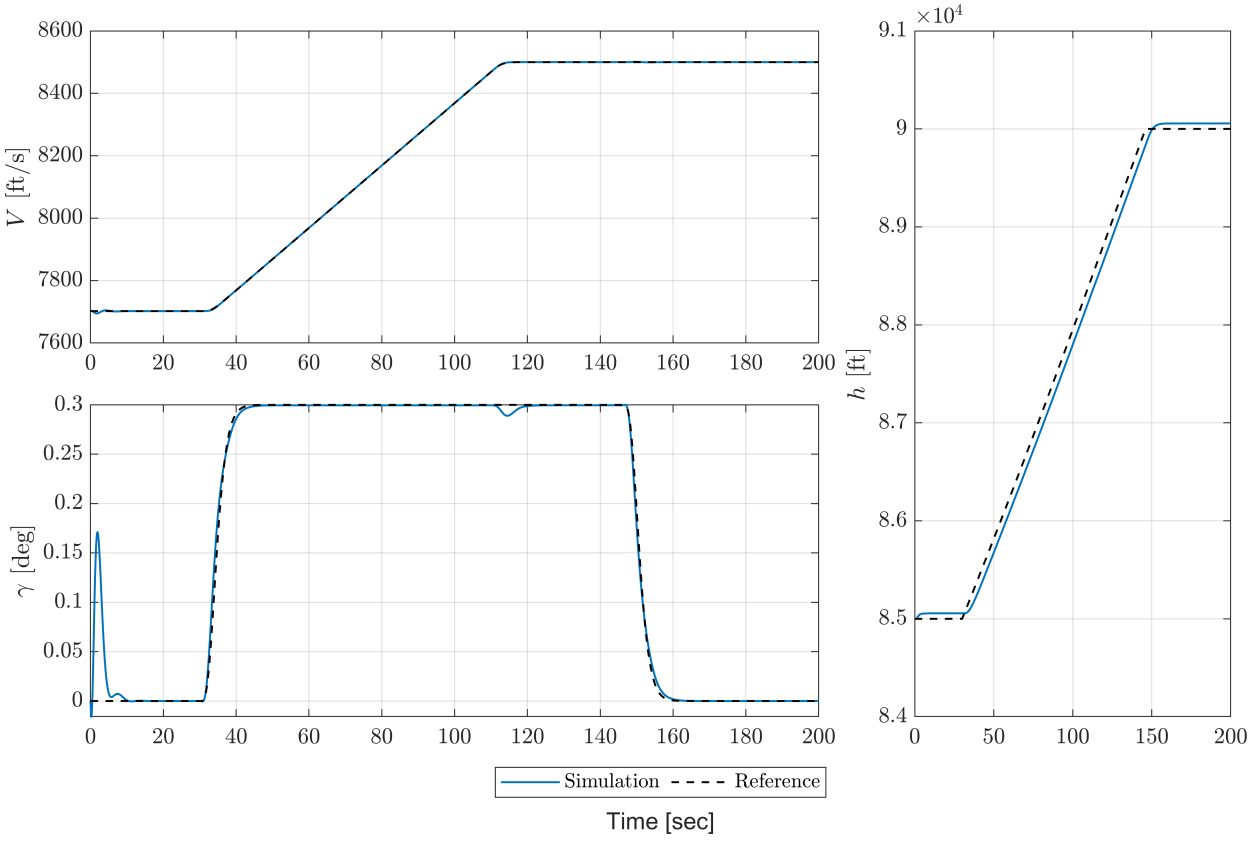


Figure 28: Comparison of h, V and γ with their respective reference signals at tracking.

14 Summary

This work discusses the longitudinal control of an air-breathing hypersonic vehicle. First, a model is implemented based on the one used in [1], numeric linearization and open-loop simulation are performed to validate the model and provide a framework for linear control methods. Later, linear control methods are proposed, namely PID and LQR which are applied to the linearized model. Finally, we implement a nonlinear controller using feedback linearization, comparing the simulation results with [1, 2].

We saw that as for the linear control methods, simply applying a PID controller is not enough to stabilize the system: a multiloop controller is required to dampen the phugoid poles and resolve the NMP zero present in the linearized system. Simulations show that linear control methods can regulate the aircraft and maintain trim conditions. These methods could also stabilize the aircraft when initializing the simulation further from the trim conditions. One of the most dominant shortcomings of linear controllers is their excessive use of the control input δ_e, which violated the relevant range of operation of the model.

The nonlinear controller performance demonstrated in simulations confirms the nonlinear controller implemented is capable of tracking commands. A reference maneuver is proposed and used to assess tracking performance, with a pre-filter to smooth the command and avoid aggressive control inputs. This type of controller exhibits a transient response when attempting to maintain the trim conditions. It requires more intricate initialization to prevent it when starting the simulation. This problem is solved in [2] analytically by calculating the required initial conditions of the integral states, so the initial control input by the controller matches the trim conditions. The resulting tracking errors of this controller are within reasonable values for the trajectory angle and negligible for the velocity. Since altitude was not controlled, its final values were different from the nominal one. As opposed to the linear methods, this controller was far more restrained in its use of control inputs, being on the verge of allowed values. This of course is a function of the reference model, and a more aggressive maneuver could make the controller violate the relevant range of operation.

This project is a part of our research and represents a basic introduction to controlling an AHV. It will assist in further research of our group. Further work could improve the quality of this project and make it more comprehensive, such as attempting to implement lead/lag compensators to enrich the linear control that is discussed in this paper. It is recommended to initialize the integral states of the nonlinear controller to avoid the transient response and test more complex missions and scenarios to better understand controller strengths and weaknesses, it could also prove interesting to test the feedback linearization controller without integral augmentation. Further research in the context of this work will focus on the use of 6-DOF models, incorporating lateral control. In addition, no control input limitations were discussed in this project, further research could include hard or soft constraints on control inputs and aircraft states to ensure applicability in physical systems, which could suffer greatly from unlimited inputs or states.

References

[1] Jason T. Parker et al. "Control-Oriented Modeling of an Air-Breathing Hypersonic Vehicle". In: *Journal of Guidance, Control, and Dynamics* 30.3 (May 2007), pp. 856–869.

[2] Ofir Vaknin and Moshe Idan. "Survey of Control Methods for Hypersonic Vehicles (Project A)". In: *Faculty of Aerospace Engineering, Technion* (Dec. 2022).

Appendix A Coefficient Tables

The following tables refer to the different coefficients that comprise the CFM and COM, according to [1], there are two more tables that refer to the flexible modes of the model, but these are of no interest in this paper.

Table A.1: Miscellaneous coefficient values

Coefficient	Value	Units
S	1.7000×10^{1}	$\text{ft}^2 \cdot \text{ft}^{-1}$
ρ_0	6.7429×10^{-5}	$\text{slugs} \cdot \text{ft}^{-3}$
h_0	8.5000×10^{4}	ft
h_s	2.1358×10^{4}	ft

Table A.2: Lift coefficient values

Coefficient	Value	Units
C_L^{α}	4.6773×10^{0}	rad^{-1}
$C_L^{\delta_e}$	7.6224×10^{-1}	rad^{-1}
C_L^{0}	-1.8714×10^{-2}	——

Table A.3: Drag coefficient values

Coefficient	Value	Units
$C_D^{\alpha^2}$	5.8224×10^{0}	rad^{-2}
C_D^{α}	-4.5315×10^{-2}	rad^{-1}
$C_D^{\delta_e^2}$	8.1993×10^{-1}	rad^{-2}
$C_D^{\delta_e}$	2.7699×10^{-4}	rad^{-1}
C_D^{0}	1.0131×10^{-2}	——

Table A.4: Moment coefficient values

Coefficient	Value	Units
z_T	8.3600×10^{0}	ft
$\bar{c}$	1.7000×10^{1}	ft
$C_{M,\alpha}^{\alpha^2}$	6.2926×10^{0}	rad^{-2}
$C_{M,\alpha}^{\alpha}$	2.1335×10^{0}	rad^{-1}
$C_{M,\alpha}^{0}$	1.8979×10^{-1}	——
c_e	-1.2897×10^{0}	rad^{-1}

Table A.5: Thrust coefficient values

Coefficient	Value	Units
β_1	-3.7693×10^{5}	$\text{lb} \cdot \text{ft}^{-1} \cdot \text{rad}^{-3}$
β_2	-3.7225×10^{4}	$\text{lb} \cdot \text{ft}^{-1} \cdot \text{rad}^{-3}$
β_3	2.6814×10^{4}	$\text{lb} \cdot \text{ft}^{-1} \cdot \text{rad}^{-2}$
β_4	-1.7277×10^{4}	$\text{lb} \cdot \text{ft}^{-1} \cdot \text{rad}^{-2}$
β_5	3.5542×10^{4}	$\text{lb} \cdot \text{ft}^{-1} \cdot \text{rad}^{-1}$
β_6	-2.4216×10^{3}	$\text{lb} \cdot \text{ft}^{-1} \cdot \text{rad}^{-1}$
β_7	6.3785×10^{3}	$\text{lb} \cdot \text{ft}^{-1}$
β_8	-1.0090×10^{2}	$\text{lb} \cdot \text{ft}^{-1}$

Appendix B Trim Condition Solution Approach

The implementation strategy: Commonly used terms such as L, D, M, and T will be implemented as **Matlab Functions** according to the CFM model. These basic building blocks could be used in Simulink using the Matlab Function block or directly implemented in other Matlab functions.

Said functions will depend on various coefficients, as well as relevant states. To simplify, a **Struct** represents global simulation parameters. It will be initialized via a helper function, which if needed could set certain parameters to zero, yielding the COM (conditioned according to function input). The aforementioned **Struct** is passed between functions as a parameter. Table B.1 lists the names of all mentioned coefficients in Matlab.

Table B.1: Matlab names of the difference coefficients.

Name	Relevant Symbol	Property Name
Lift Coefficients	$C_L^{\alpha}, C_L^{\delta_e}, C_L^0$	CLa, CLd, CL0
Drag Coefficients	$C_D^{\alpha^2}, C_D^{\alpha}, C_D^{\delta_e^2}, C_D^{\delta_e}, C_D^0$	CDa2, CDa, CDd2, CDd, CD0
Moment Coefficients	$z_T, \bar{c}, C_{M,\alpha}^{\alpha^2}, C_{M,\alpha}^{\alpha}, C_{M,\alpha}^0, c_e$	zT, mac, CMa2, CMa, CM0, ce
Thrust Coefficients	β_i	bi
Physical Properties	$L_v, I_y y, m, S$	Lv, Iyy, m, S
Flight Conditions	ρ_0, h_0, h_s	rho0, h0, hs

Using the implemented model, a solution to (5) is found via **fsolve**.

Appendix C Simualtion Reference Model Filter

The filter is constructed via a fifth-order transfer function,

$$\begin{aligned}\frac{Y_{ref}}{Y_{ex}} &= \left(\frac{\omega_n^2}{s^2+2\zeta\omega_n s+\omega_n^2}\right)^2 \frac{\omega_n}{s+\omega_n} \\ &= \frac{\omega_n^5}{(s+\omega_n)(s^4+4\zeta\omega_n s^3+(4\zeta^2\omega_n^2+2\omega_n^2)s^2+4\zeta\omega_n^3 s+\omega_n^4)}.\end{aligned} \tag{C.1}$$

It has four imaginary poles - two pairs in the same location, and another real pole with the same natural frequency (w_n - distance from the origin) as the others. Equation (C.1) can be reformulated in controllable canonical form to yield the space state representation

$$\dot{\vec{x}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ -a & -b & -c & -d & -e \end{bmatrix} \vec{x} + \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \omega_n^5 Y_{ex} \end{Bmatrix}, \quad \vec{x} = \begin{Bmatrix} Y_{ref} \\ \dot{Y}_{ref} \\ \ddot{Y}_{ref} \\ \dddot{Y}_{ref} \\ Y_{ref}^{(4)} \end{Bmatrix}, \tag{C.2}$$

where

$$a = \omega_n^5, \qquad b = \omega_n^4(4\zeta+1), \tag{C.3a}$$
$$c = \omega_n^3(4\zeta^2+4\zeta+2), \qquad d = \omega_n^2(4\zeta^2+4\zeta+2), \tag{C.3b}$$
$$e = \omega_n(4\zeta+1). \tag{C.3c}$$

The selected values for the said filter are

$$\zeta_V = 1, \qquad \omega_{n,V} = 1.5 \text{ [rad/s]}, \tag{C.4a}$$
$$\zeta_\gamma = 1, \qquad \omega_{n,\gamma} = 1 \text{ [rad/s]}. \tag{C.4b}$$

This filter is used for simulations presented in section 13. The coefficients a, b, c, d, and e are calculated in Matlab and used to define state-space matrices, which are later fed into Simulink to implement the filter.