Refining F-16 Flight Dynamics

A Practical Approach to Classical Control

Practical Assignment AE4301P: Exercise Automatic Flight Control System Design

Richard Adam & **Evan Sayer**

Professor: Dr. E. Smeur
Course: AE4301P

February 2025

Delft University of Technology
Delft, The Netherlands

1 Introduction

Modern high-performance aircraft such as the General Dynamics F-16 require sophisticated flight control laws to stabilize multiple coupled motion modes while accommodating pilot commands and mission objectives. The work presented in this report addresses the design of classical controllers for the F-16, focusing on both linearization-based analysis and iterative tuning under realistic operating constraints. In particular, the discussion begins with augmenting the model to capture normal acceleration at different fuselage stations, revealing important non-minimum-phase behavior that arises from the aircraft's geometry and sensor placement. The open-loop dynamics are then examined to see how eigenmodes like the short-period, phugoid, Dutch roll, and spiral each contribute to overall handling qualities.

With these insights, a pitch rate command system is devised that satisfies typical fighter-aircraft criteria for short-period damping, specifically the Control Anticipation Parameter (CAP) and the Gibson dropback criterion. Emphasis is placed on selecting appropriate gains to ensure agile response without leading to large overshoot or undue pilot workload. The resulting closed-loop system is further validated under gust-like disturbances and parameter variations to demonstrate robustness.

Finally, the report examines a practical extension to low-speed flight: the implementation of a glideslope and flare controller that guides the aircraft during approach and landing. The proposed structure illustrates how linear design principles translate into reliable performance near critical phases of flight. It utilizes nested control loops, coordinating pitch, thrust, velocity, and altitude hold modes. Simulation results from MATLAB and Simulink confirm that the proposed methods achieve stable, well-damped maneuvers. Together, these steps illustrate a cohesive framework for integrating trimming, open-loop characterization, feedback controller synthesis, and final approach automation in a demanding fighter-aircraft environment.

2 Normal Acceleration Model and Trimmed Linearization

2.1 Augmenting the Model with an Accelerometer

Equation (5.6) in the assignment specifies the normal acceleration at a point located x_a feet forward of the aircraft center of gravity (cg). In the body-fixed axes, this measured acceleration a_n is written as

$$a_n \;=\; -\,\frac{a_z \;-\; \dot{q}\,x_a}{g_D}, \tag{1}$$

where a_z is the vertical acceleration at the cg, $\dot{q}$ is the pitch-rate derivative, and g_D is standard gravitational acceleration. Implementing (1) in the Simulink file `LIN_F16Block_With_an` augments the standard F-16 states with an additional output channel for normal acceleration. Although the core state matrices $\mathbf{A}$ and $\mathbf{B}$ remain unchanged, the output matrix $\mathbf{C}$ and the feedthrough matrix $\mathbf{D}$ are expanded to reflect the newly introduced measurement of a_n. This approach allows us to evaluate how elevator commands affect normal acceleration at different fuselage stations x_a.

2.2 Trimming and Linearizing at the Chosen Flight Condition

Before linearizing, it is crucial to trim the model at the target flight condition: $h = 15000 ft$ and $V = 500 ft/s$. A steady-state solution is obtained by adjusting thrust, elevator, ailerons, rudder, and angle of attack to minimize the cost function associated with deviations from level flight. As shown in the MATLAB output, the low-fidelity F-16 model converges to near-zero cost (on the order of 10^{-29}), demonstrating that the aircraft is indeed in steady, level flight. Linearization around an untrimmed point would yield inconsistent or misleading linear models, so this trim step is indispensable.

Once the aircraft is trimmed, the script `FindF16Dynamics_15000_500` employs the `linmod` operation to extract a linear state-space representation for each chosen x_a. These linear models capture the local dynamics in the vicinity of the trimmed equilibrium, providing a standard form:

$$\begin{aligned} \dot{\mathbf{x}} \;&=\; \mathbf{A}\,\mathbf{x} \;+\; \mathbf{B}\,\mathbf{u}, \\ \mathbf{y} \;&=\; \mathbf{C}\,\mathbf{x} \;+\; \mathbf{D}\,\mathbf{u}, \end{aligned} \tag{2}$$

where $\mathbf{x}$ and $\mathbf{u}$ denote the state and input vectors, respectively, and $\mathbf{y}$ includes the normal acceleration output, a_n. In particular, for $x_a = 0$ (the cg location), the linearized output equation that maps the state vector to a_n can be seen directly in the MATLAB symbolic expression:

$$a_n \;=\; 0.00399\,V_t \;+\; 9.93\,\alpha \;+\; 0.0208\,\big(\text{elevator\_state}\big) \;-\; 3.24\times 10^{-5}\,h \;+\; 0.966\,q \;-\; 9.68\times 10^{-6}\,\theta,$$

which indicates that α, q, and to a lesser extent V_t and θ drive the cg normal acceleration.

2.3 Elevator-to-Normal-Acceleration Dynamics and Transfer Function Zeros

From the linearized state-space model, one can compute the transfer function

$$G_{a_n}(s) \;=\; \frac{a_n(s)}{\delta_e(s)}$$

relating elevator deflection δ_e to a_n. Table 1 summarizes the zero locations of this transfer function for several accelerometer positions x_a. Notably, in each case there is at least one zero in the right-half-plane (RHP) or near the origin, indicating potential non-minimum-phase (NMP) behavior. A zero in the RHP causes the short-term response to move opposite the final steady-state direction when the elevator is abruptly deflected, producing an undershoot for a nose-up command.

x_a (ft)	Zeros (approx.)
0	$\{\ 9.76,\ -5.39,\ -0.32 \pm 2.74i,\ -2.12,\ 0.0094,\ -0.0113,\ -0.0059,\ 0\ \}$
5	$\{\ 40.52,\ -8.48,\ -0.32 \pm 2.74i,\ -2.12,\ 0.0094,\ -0.0113,\ -0.0059,\ 0\ \}$
5.9	$\{\ 6852,\ -10.5,\ -0.3 \pm 0.0027i,\ -0.0021,\ 0,\ -0.0000,\ -0.0000,\ 0\ \}$
6	$\{-298.61,\ -10.88,\ -0.32 \pm 2.74i,\ -2.12,\ 0.0001,\ -0.0001,\ -0.0001,\ 0\ \}$
7	$\{-13.80 \pm 9.65i,\ -0.32 \pm 2.74i,\ -2.12,\ 0.0094,\ -0.0113,\ -0.0059,\ 0\ \}$
15	$\{-1.94 \pm 5.51i,\ -0.32 \pm 2.74i,\ -2.12,\ 0.0094,\ -0.0113,\ -0.0059,\ 0\ \}$

Table 1: Zeros of the elevator-to-a_n transfer function at different accelerometer positions.

2.4 Step Response to a Nose-Up Elevator Command

Figure 1 presents the simulated step responses to a negative elevator command (nose-up) for $x_a \in \{0,\ 5,\ 5.9,\ 6,\ 7,\ 15\}$ ft. These plots (`step_xa_0`, `step_xa_5`, `step_xa_5_9`, `step_xa_6`, `step_xa_7`, `step_xa_15`) illustrate that the initial acceleration often deflects below the steady-state level due to the non-minimum-phase zero. Positioning the accelerometer further forward accentuates the influence of $\dot{q}\,x_a/g_D$ in (1) and shifts the zero locations, typically intensifying the undershoot.

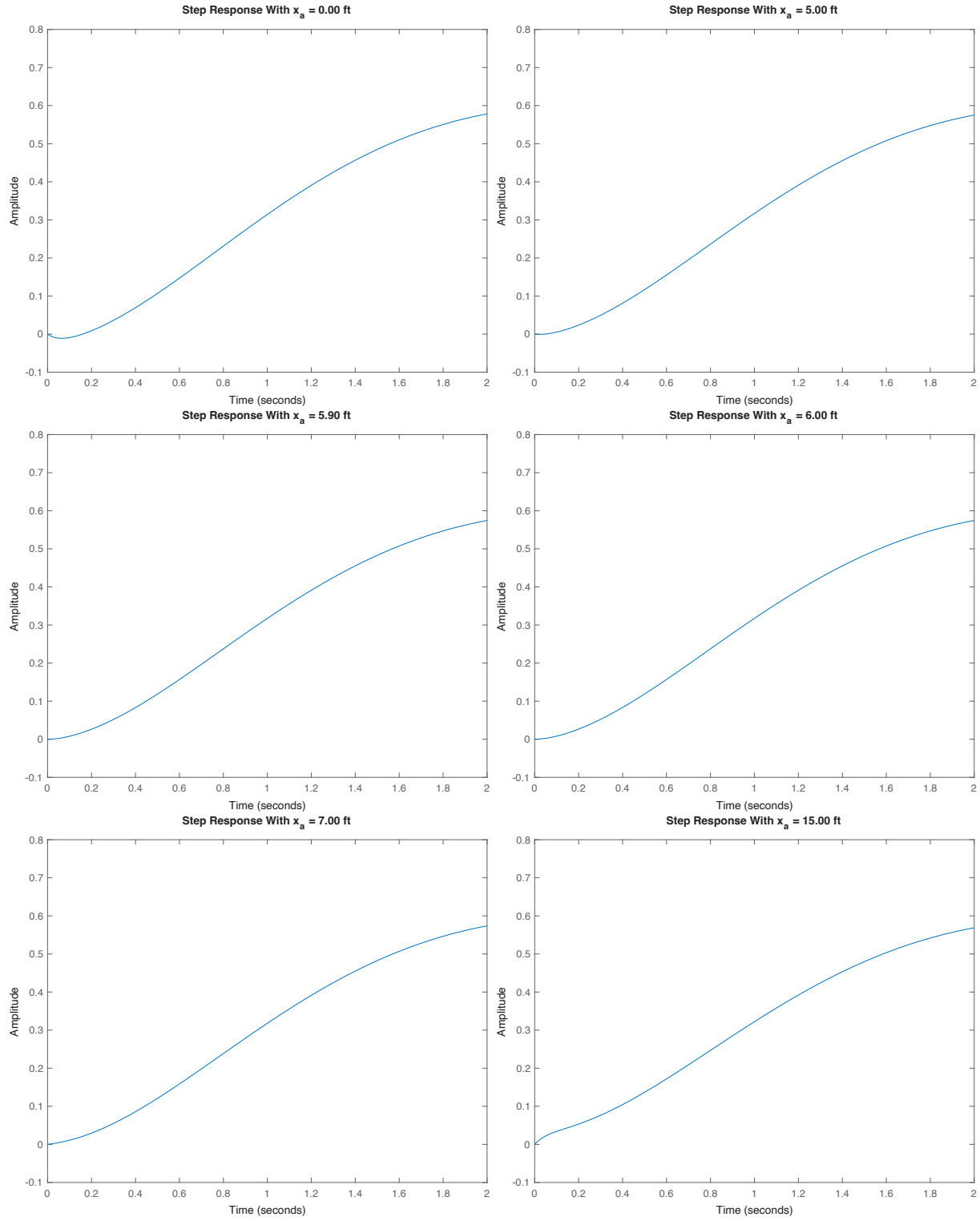

Figure 1: Negative step responses (nose-up) in elevator deflection for $x_a = 0, 5, 5.9, 6, 7,$ and $15\,\text{ft}$

.

This short-term reversal in a_n stems from the need to reorient the aircraft's flight path and angle of attack before generating the net positive lift corresponding to a nose-up command. Such non-minimum-phase behavior makes control design challenging, as naive feedback loops may overcompensate for the initial

dip and create oscillations or instability.

2.5 Instantaneous Center of Rotation and Sensor Placement

Another consideration is the *instantaneous center of rotation*, which typically lies near the cg for small pitch perturbations. Placing the pilot or the sensor well ahead of this center exaggerates rotational accelerations, whereas placing it slightly aft mitigates them. In most aircraft, the cockpit is located near but slightly behind the cg to reduce pilot-induced oscillations and improve comfort. Similarly, situating accelerometers close to a node of the primary fuselage bending mode ensures that structural vibrations do not contaminate the normal-acceleration measurement. This strategy maintains accurate feedback signals, especially under high-g loads where airframe flex can otherwise skew sensor readings.

2.6 Observations and Key Implications

Incorporating an additional normal-acceleration output in the low-fidelity F-16 model at $h = 15000ft$ and $V = 500ft/s$ reveals several important aspects of the aircraft's behavior. The linearized form of the model clarifies how angle of attack, pitch rate, and other states couple into the measured a_n at the center of gravity. Moreover, examining the transfer function from the elevator input to a_n for different accelerometer locations x_a highlights the presence of a right-half-plane zero, which produces an undershoot during nose-up maneuvers.

This undershoot arises because the aircraft momentarily reorients itself, resulting in an initial decrease in normal acceleration before the net lift can settle at the new, higher level. As the accelerometer moves farther forward from the center of gravity, the zero dynamics shift, often amplifying this non-minimum-phase effect. The notion of an instantaneous center of rotation provides additional insight, since placing the sensor (or pilot) well away from that center accentuates rotational accelerations. Finally, situating accelerometers near fuselage bending-mode nodes ensures that structural vibrations do not corrupt the measurement, thereby improving the fidelity of feedback signals under high-g loads. Together, these findings underscore the importance of careful sensor placement and appropriate compensator design when dealing with non-minimum-phase characteristics in high-performance aircraft.

3 Open Loop Analysis

A reduced-order representation of the aircraft's open loop behavior can be obtained by splitting the linearized state-space model into longitudinal and lateral subsystems. The longitudinal subsystem here includes pitch angle θ, pitch rate q, total velocity V_t, and angle of attack α, with thrust and elevator deflection as inputs. In parallel, the lateral subsystem contains roll angle ϕ, sideslip angle β, roll rate p, and yaw rate r, with aileron and rudder deflections as inputs. Each subsystem exhibits distinct eigenmodes that can be categorized into periodic (oscillatory) and aperiodic (non-oscillatory) behaviors. In the longitudinal channel, the system primarily displays the phugoid and short-period modes, while the lateral channel exhibits the Dutch roll, aperiodic roll, and spiral modes. Together, these modes define the inherent flying qualities of the unaugmented aircraft.

3.1 Periodic Eigenmotions

3.1.1 Phugoid

The phugoid mode arises from the exchange of kinetic and potential energy in the longitudinal plane, resulting in low-frequency oscillations. For the selected flight condition of $h = 10000ft$ and $V = 350ft/s$, the linearized system yields phugoid poles at $-0.0053 \pm 0.1144i$. From these poles, a damping ratio $\zeta = 0.5005$ and a natural frequency $\omega_n = 0.1145$ rad/s are computed, corresponding to a relatively lightly damped oscillation. A half-amplitude decay time of approximately 12.09 s suggests that the oscillations persist for an extended period before diminishing. Figure 2 illustrates how the velocity and pitch angle respond when a step deflection is applied to the elevator; the low-frequency oscillatory nature is evident in both responses.

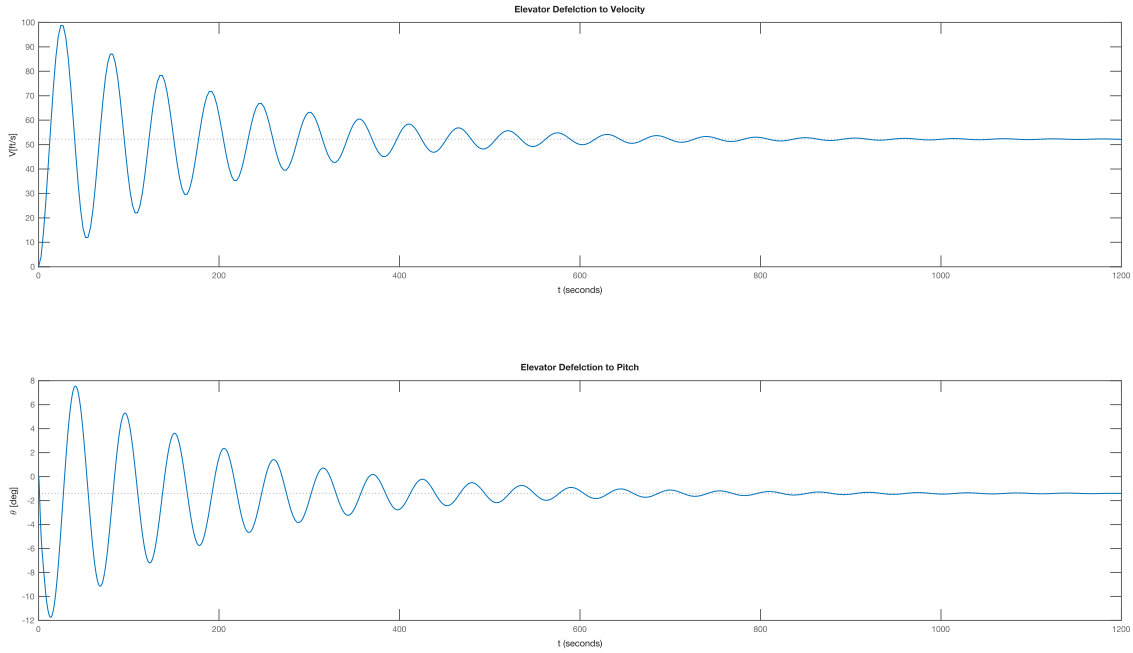

Figure 2: Velocity and pitch angle responses to an elevator step deflection (phugoid mode).

3.1.2 Short-Period

Unlike the low-frequency phugoid, the short-period mode is a higher-frequency, heavily damped longitudinal oscillation involving primarily the angle of attack and pitch rate. The poles appear at $-0.6648 \pm 1.0567i$, yielding $\zeta = 0.5907$ and $\omega_n = 1.2484$ rad/s. A half-amplitude decay time of about 0.94 s indicates a much

faster transient than the phugoid. Step responses of α and q to elevator deflection, as shown in Figure 3, confirm a swift oscillation that subsides within a second or two. Such rapid damping is beneficial for pilot control, though it can still pose handling challenges if the damping is insufficient at higher dynamic pressures.

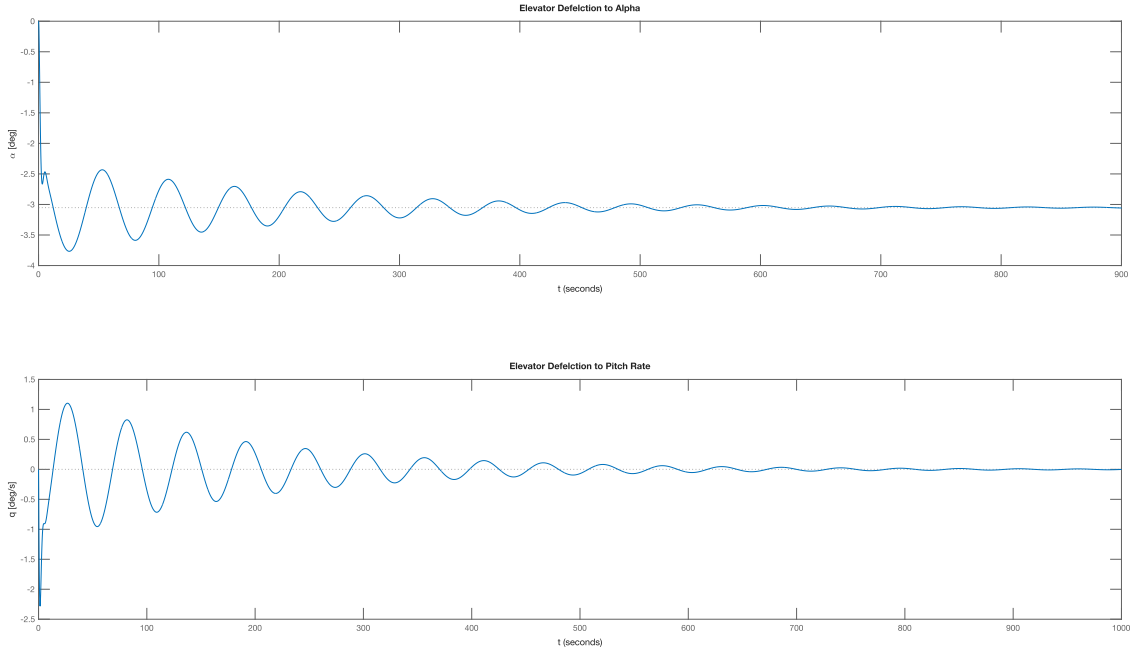

Figure 3: Angle of attack and pitch rate responses to an elevator step (short-period mode).

3.1.3 Dutch Roll

The principal periodic lateral motion is the Dutch roll, characterized by a coupled roll-yaw oscillation. Its poles are located at $-0.3532 \pm 2.3773i$, corresponding to $\zeta = 0.5055$ and $\omega_n = 2.4033$ rad/s, with a half-amplitude decay time of roughly 0.57 s. Although not as rapid as the short-period in the longitudinal plane, the Dutch roll is nonetheless a relatively fast lateral oscillation. Figure 4 shows impulse responses of ϕ, β, p, and r when the aileron is momentarily deflected. In practice, ensuring adequate damping of Dutch roll is critical to provide directional stability and reduce sideslip excursions.

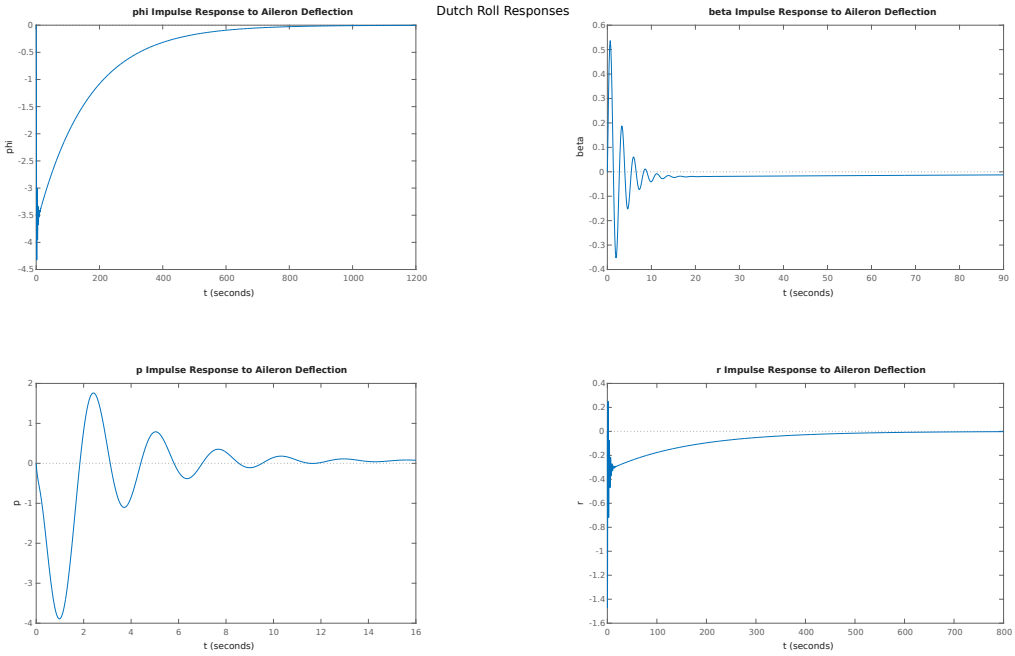

Figure 4: Typical Dutch roll impulse responses in roll and yaw states.

3.2 Aperiodic Motions

3.2.1 Aperiodic Roll

An aileron deflection excites a purely real (non-oscillatory) pole in the lateral channel, corresponding to the aperiodic roll mode. In this flight condition, the aperiodic roll pole is located at -1.4165, giving a time constant of about $0.706\,\mathrm{s}$ and a half-amplitude decay time of $0.4893\,\mathrm{s}$. As shown in Figure 5, the roll rate response to a step in the aileron is relatively quick and decays in under a second, reflecting the natural damping provided by the wing and fuselage dihedral effect.

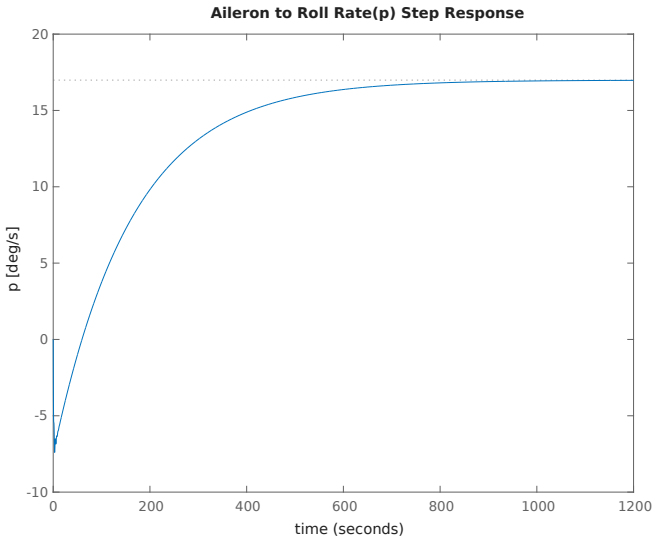

Figure 5: Roll rate step response to an aileron input (aperiodic roll mode).

3.2.2 Spiral

A second real pole in the lateral subsystem is the spiral mode, located at -0.0061. This very small negative real part indicates a weakly stable but extremely slow roll divergence or convergence. A time constant of around 164 s and a half-amplitude decay time of 113 s imply that the aircraft will gradually roll off or restore itself over a long period if disturbed. Figure 6 highlights how an initial roll angle of 20° evolves over time in the absence of pilot or control inputs. Although such a slow response is easily corrected by the pilot or a stability augmentation system, it can be hazardous if left unchecked for extended durations.

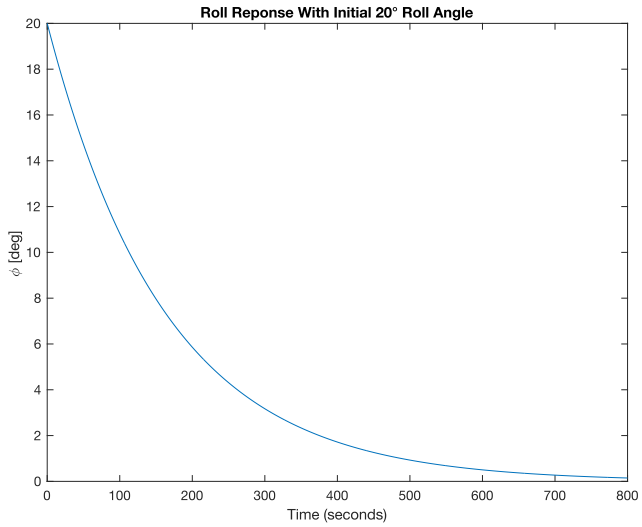

Figure 6: Lateral time response to an initial roll angle (spiral mode).

3.3 Overall Assessment of Open Loop Dynamics

Examination of the eigenvalues in both the longitudinal and lateral subsystems indicates that the aircraft is stable at the chosen flight condition, since all real parts are negative. Nevertheless, the slow damping

of the phugoid, the moderate damping of the Dutch roll, and the sluggish nature of the spiral mode may not meet the handling-quality standards typically prescribed for high-performance aircraft. In particular, MIL-F-8587C specifications often require shorter settling times or higher damping ratios to ensure acceptable levels of pilot workload. Hence, while the open loop dynamics exhibit inherent stability, they are unlikely to satisfy more stringent flight performance criteria without additional control or augmentation systems.

4 Pitch Rate Control System Design

A pitch rate command system can greatly enhance the low-speed handling qualities of an aircraft like the F-16 by enabling precise short-period responses. Military specifications, such as the Control Anticipation Parameter (CAP) and the Gibson dropback/phase-rate criteria, define acceptable short-period frequencies, damping ratios, and transient characteristics to ensure adequate flying qualities. The following sections describe how the short-period model is reduced from the full linearized dynamics, how a controller is devised to meet the derived requirements, and how compliance with CAP/Gibson criteria is verified.

4.1 Reduced Short-Period Model Construction

The first step is to isolate the short-period dynamics, consisting primarily of the angle of attack α and the pitch rate q. Because the actuator dynamics can significantly alter the response, a "four-state" model may include them, whereas a simpler "two-state" model omits the actuator. In this study, the reduced system is obtained by retaining only the relevant states from the larger state-space matrices.

To illustrate the fidelity of each model, Figure 7 compares the pitch rate step responses for:

- The four-state model with actuator dynamics.
- A purely two-state short-period model (no actuator).
- A second-order system derived directly from the identified short-period poles $(\omega_{n,sp}, \zeta_{sp})$.

Although the short-period motion dominates the initial transient, higher-order effects such as phugoid become evident at later times. Nevertheless, over the first few seconds, all three models show similar frequency and damping characteristics.

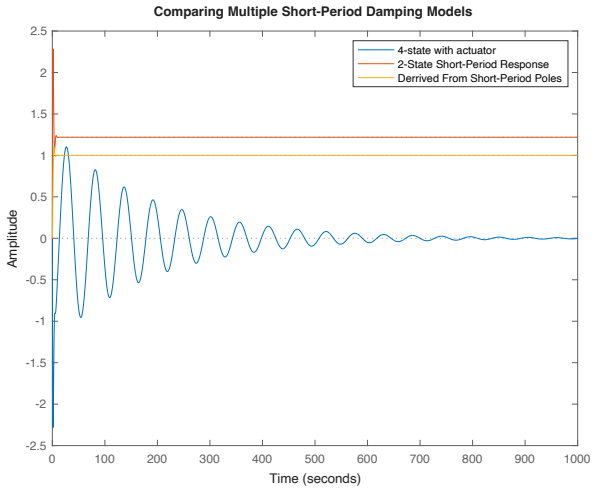

Figure 7: Comparison of short-period responses: four-state (with actuator), two-state (no actuator), and a second-order pole approximation.

4.2 Mapping CAP and Gibson Criteria to Frequency-Domain Requirements

The CAP and Gibson criteria, although expressed in load-factor or pitch-rate terms, must be translated into more conventional frequency-domain specifications for controller design. In particular, the short-period natural frequency $\omega_{n,sp}$, damping ratio ζ_{sp}, and time constant T_{θ_2} should satisfy:

$$\omega_{n,sp}(V,h) = 0.03\,V, \quad \frac{1}{T_{\theta_2}(V,h)} = 0.75\,\omega_{n,sp}(V,h), \quad \zeta_{sp}(V,h) = 0.5,$$

where V is the airspeed in m/s. Meeting these conditions ensures that the short-period mode is neither too sluggish (which would violate CAP) nor overly oscillatory (which would violate Gibson's dropback criterion).

In practice, once $\omega_{n,sp}$ and ζ_{sp} are fixed, the CAP can be computed from

$$\text{CAP} = \frac{g\,\omega_{n,sp}^2\,T_{\theta_2}}{V}, \tag{3}$$

and the Gibson dropback quantity $\text{DB} = T_{\theta_2} - \frac{2\zeta_{sp}}{\omega_{n,sp}}$ can be used to evaluate the negative dip in pitch rate once the step input is removed.

4.3 Procedure for Achieving Desired $\omega_{n_{sp}}$ and ζ_{sp}

To shape the short-period poles to the target frequency and damping ratio, a pitch rate command system is constructed around the reduced model. A typical approach involves:

1. *Pitch Rate Feedback* (K_q): Adjusts short-period stiffness and sets the mode's natural frequency.
2. *Angle-of-Attack Feedback* (K_α): Allows additional phase lead or direct control over the damping ratio.
3. *Lead-Lag Prefilter or Pole-Zero Cancellation*: Fine-tunes the response without destabilizing the internal poles.
4. *Feed-Forward Gain*: Ensures that the steady-state pitch rate tracks the pilot's command.

Figure 8 depicts a block diagram of such a system, showing how the elevator actuator, pitch rate feedback loop, and optional prefilter interact.

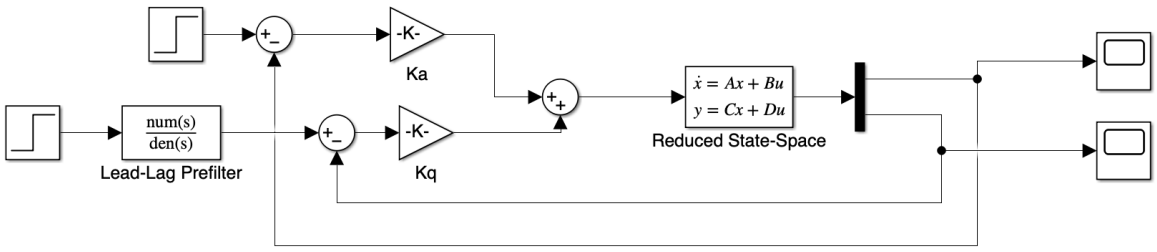

Figure 8: Block diagram for pitch rate command system, including feedback paths and lead-lag/prefilter if required.

4.4 Time Constant T_{θ_2} and Prefilter Placement

A crucial caveat is that T_{θ_2} often arises from aerodynamic and structural properties rather than from purely pilot-selectable gain settings. Hence, simply moving poles via classic feedback may not suffice to adjust T_{θ_2}. A lead-lag prefilter or a carefully placed pole-zero cancellation outside the core feedback loop is typically needed to shape the response further. Placing this filter inside the main feedback loop could undesirably alter the closed-loop poles or create spurious zero-pole cancellations, so the filter is applied externally, modifying the command signal in a stable and predictable way.

4.5 Feed-Forward Gain and Steady-State Tracking

Including a feed-forward gain K_{ff} ensures that the final pitch rate matches the pilot's step input in steady state. By applying a fraction of the reference signal directly to the elevator command, the integrator wind-up or large error transients are minimized. Any changes to this feed-forward path should also be verified for stability—an overly large K_{ff} could introduce additional overshoot or degrade damping. In Figure 8, the feed-foward gain is included in the "Lead-Lag Prefilter" block.

4.6 Controller Implementation and Short-Period Pole Placement

Using the structure of a pitch rate gain controller, angle of attack controller, lead-lag prefilter, and feed-forward gain, the Simulink control tuning toolbox was used to achieve the desired system parameters of:

$\omega_{n_{sp}} = 3.2004$
$\zeta_{sp} = 0.5$
$T_\theta = 0.4166$

After the tuning process, the final control system parameters are as follows:

$K_\alpha = -1.03$
$K_q = -0.29$
$K_{ff} = 1.35e-12$
$Lead-Lag\ Pole = -0.5051$
$Lead-Lag\ Zero = -3.9e+12$

The pole is placed such that it cancels the existing closed-loop zero at -0.5051 and replaces it with a zero at -3.9e12, much further into the left-hand plane. This makes the poles of the full system curve further into the left-hand plane toward the zero to achieve the desired system response parameters. The actual step response is compared with the desired step response in Figure 9.

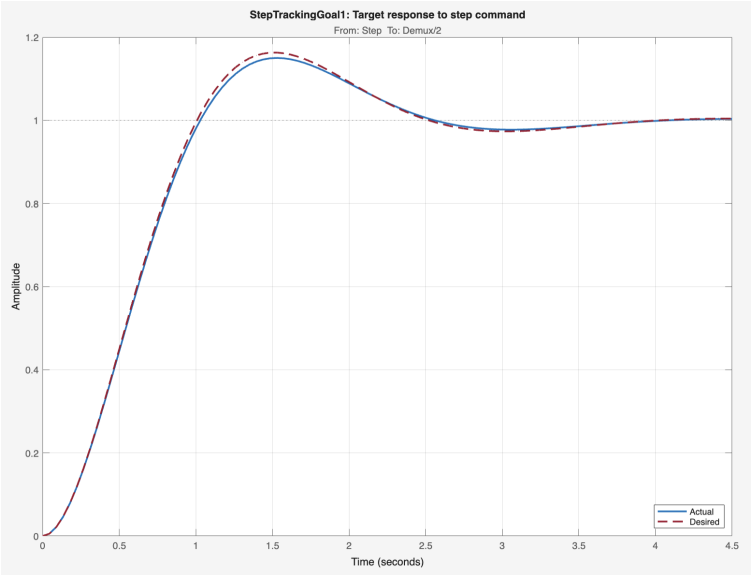

Figure 9: Results of simulink control tuning app with desired and actual system step response.

The tuning matches the desired system almost perfectly, with the only deviation from desired system being a 15% overshoot instead of the desired 16% overshoot. The Nichols plot for the controlled system can be seen in Figure 10, and shows that the controlled system has a high bandwidth and responsiveness.

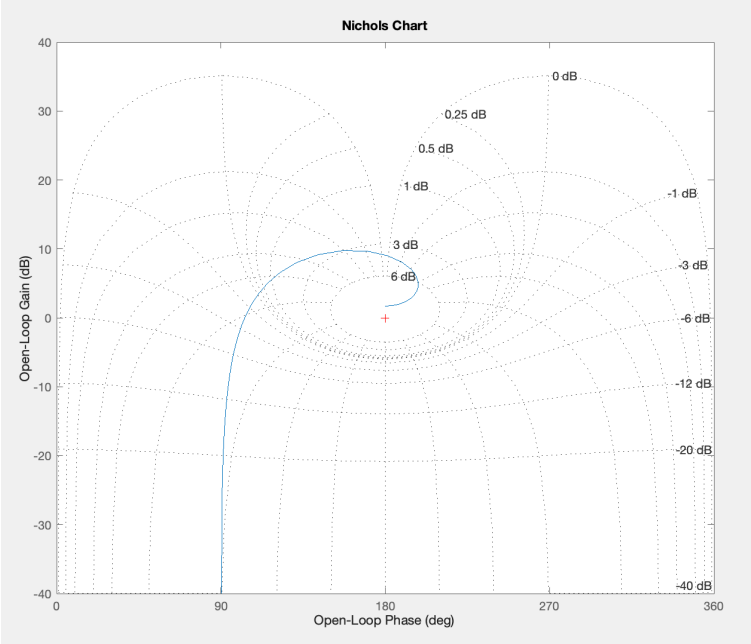

Figure 10: Nichols plot of the controlled short-period subsystem.

Additionally, Figure 11 compares the open-loop and controlled short-period step responses. The commanded pitch rate settles far more quickly in the controlled case and has a much lower overshoot, reflecting the desired $\zeta_{sp} = 0.5$. It also has a much smaller steady-state error than the open-loop system. While this damping is generally adequate, one must also verify that any overshoot meets the Gibson dropback limits.

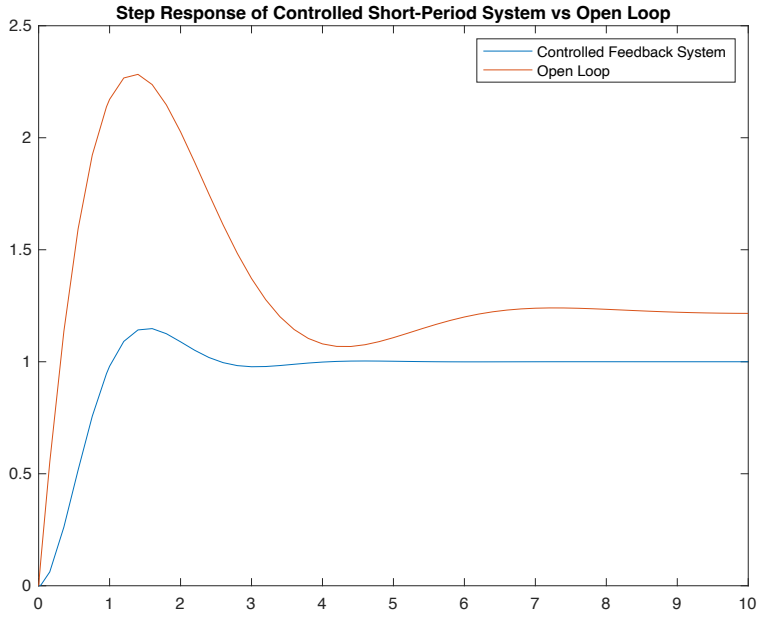

Figure 11: Step response comparison of the open-loop versus controlled short-period mode.

4.7 CAP Calculation and Gibson Dropback Verification

After the system tuner, the short-period parameters approximately satisfy the designed damping, time constant, and natural frequency values above, so the CAP and Gibson dropback can be calculated using

their equations.

With $g = 9.81 m/s^2$ and $V = 106.68 m/s$, the CAP is

$$\text{CAP} = \frac{\omega_{n,sp}^2 \, g \, T_{\theta_2}}{V} = 0.3924,$$

which lies within the typical F-16 envelope for satisfactory maneuverability. This can be seen in Figure 13. Meanwhile, Gibson's dropback parameter can be quantified by

$$\text{DB} = T_{\theta_2} - \frac{2\,\zeta_{sp}}{\omega_{n,sp}} = 0.1098,$$

and the overshoot ratio $\frac{q_m}{q_s}$ from the step response is approximately 15%. Figure 12 illustrates the pitch rate time history once the step input is removed, highlighting a satisfactory position.

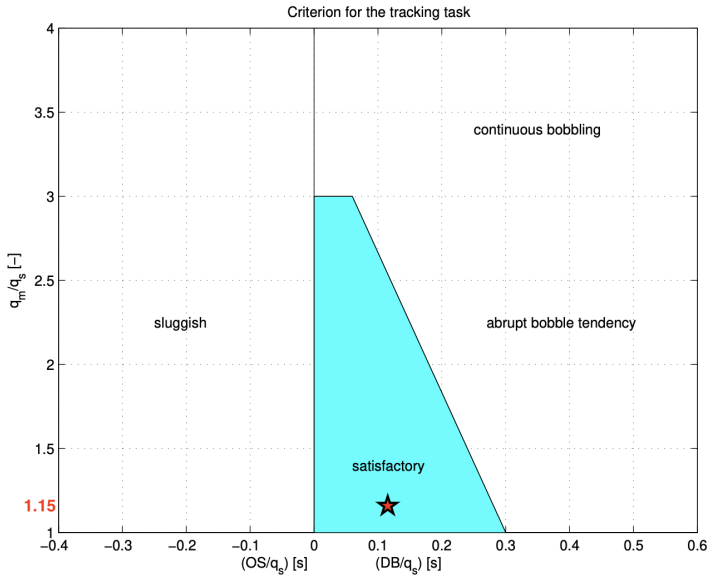

Figure 12: Pitch rate evolution after a commanded step input is removed, showing the overshoot ratio and dropback.

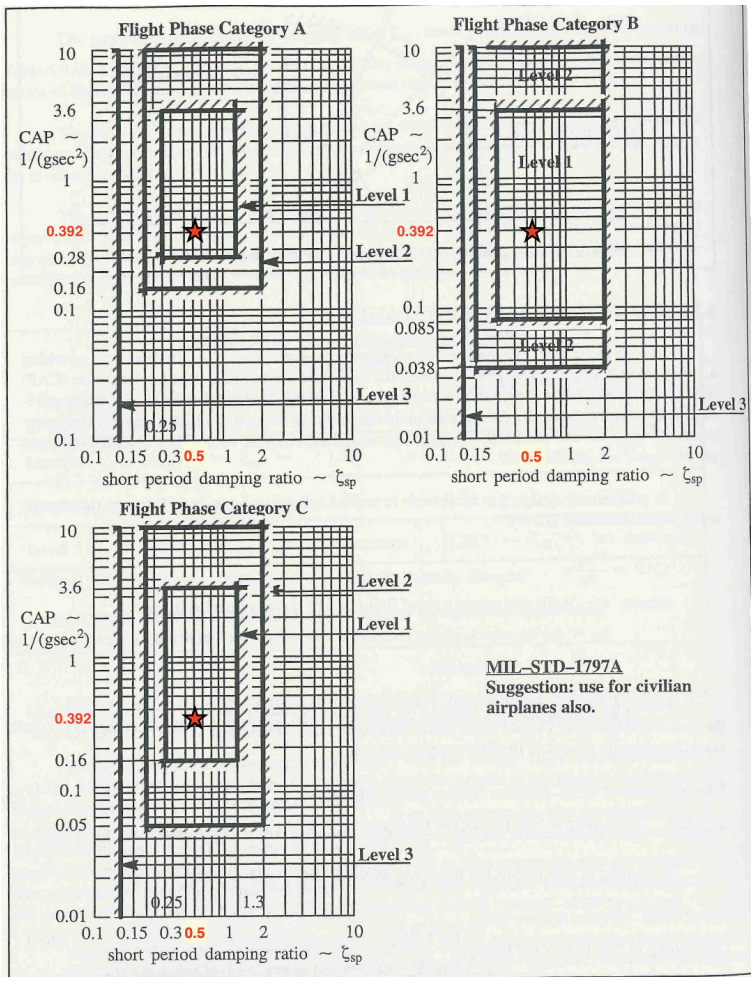

Figure 13: Illustration of allowable CAP regions and the final design's position.

Clearly, the plots show that the proposed pitch-rate command system meets both the Gibson dropback criteria and the CAP requirements for handling.

4.8 Wind Gust Behavior Analysis

An important consideration for pitch-rate command systems is their behavior in wind gusts. To simulate this, an impulse was added to the angle of attack input with a speed of 15ft/s, chosen to meet the "severe gust" criteria of the MIL-F-8785C guidelines. The results are in Figure 14.

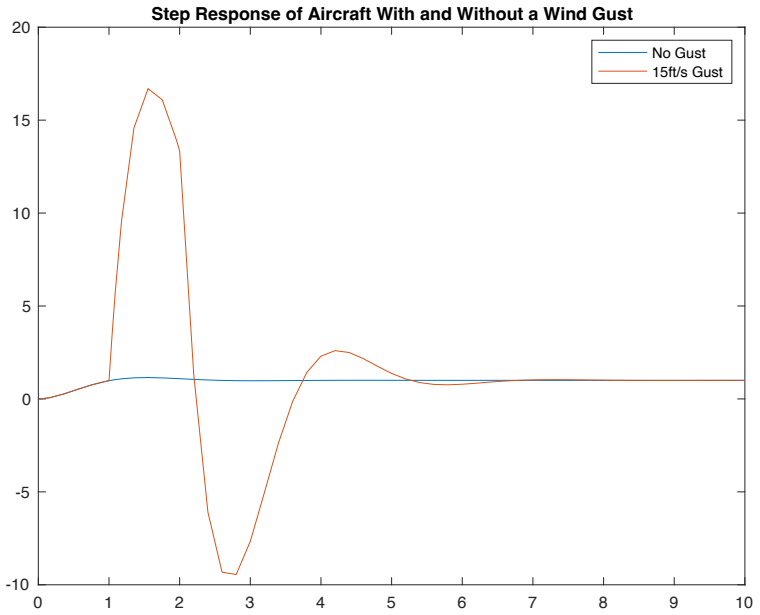

Figure 14: Step response of aircraft both with and without a 15ft/s wind gust added.

The controller is not destabilized by the severe wind gust, but its damping is somewhat slow, which could be an area of further analysis in a more comprehensive control design.

4.9 Reflections on the Pitch Rate Controller Performance

In developing this pitch rate command system, the reduced short-period model was first assembled to isolate the dominant longitudinal dynamics with and without actuator effects. Comparing time responses from a four-state actuator model, a two-state short-period model, and a second-order pole approximation confirmed that the early pitch rate behavior is well-represented by the short-period poles, whereas the phugoid mode primarily affects longer-term settling. These observations facilitated recasting the CAP and Gibson criteria in terms of the short-period frequency $\omega_{n,sp}$, damping ratio ζ_{sp}, and time constant T_{θ_2}.

Both pitch rate and angle of attack proportional control were combined with a lead-lag prefilter and feed-forward gain to achieve the desired system damping, natural frequency, and time constant. This approach successfully met the CAP requirement, indicating that the aircraft's nose follows the stick input with satisfactory agility. Gibson's dropback criteria was also met, indicating that the pitch rate has an appropriate overshoot.

Overall, the final design demonstrates robust pitch rate control, even under severe gust conditions, while adhering to the CAP requirement and maintaining manageable dropback. Should a stricter overshoot limit be required, further tuning or additional lead-lag compensation could be added. In practice, this would entail balancing agility against the tighter constraints imposed by Gibson's dropback criteria.

5 Glideslope and Flare Control System Design

This section describes the development of an automatic glideslope following and flare control system for the F-16 at a reduced flight condition of $h = 5000$ ft and $V = 300$ ft/s. Because this flight condition represents one of the slowest and lowest operating points where the high-performance model remains valid, special attention is given to ensuring that both the glideslope capture and subsequent flare maneuvers meet required performance and saturation constraints.

5.1 Inner Loop Design

The glideslope controller relies on a stable foundation of inner-loop pitch rate, pitch attitude, and velocity control. Before tackling the glideslope tracking problem, these individual loops were created and tuned under simplified conditions corresponding to level, constant-velocity flight. Subsequently, their gains were refined to accommodate the descent and flare phases. Table 2 summarizes the final gains for each of the three inner loops.

Loop	**Pitch Rate**	**Pitch Attitude**	**Velocity**
Gain	-60	15	100

Table 2: Inner-loop gains for pitch rate, pitch attitude, and velocity loops.

Figure 15 illustrates the architecture, with a reference pitch attitude feeding into the pitch-attitude controller, which in turn generates a reference pitch rate. This reference is then passed to the pitch-rate controller for high-bandwidth stabilization. The velocity loop acts in parallel to maintain or adjust airspeed as required for glideslope or flare maneuvers.

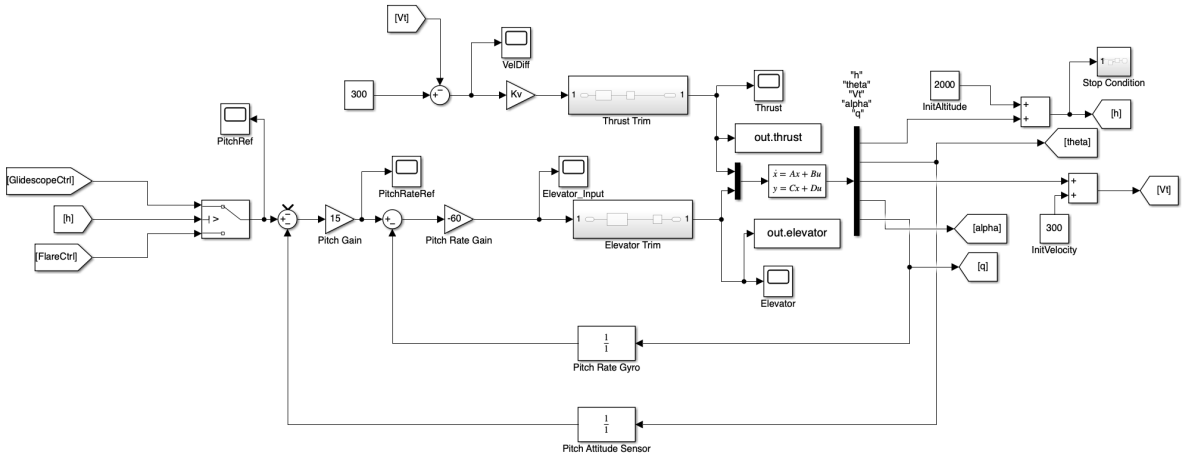

Figure 15: Inner-loop structure for the glideslope Simulink controller.

5.2 Glideslope Controller Design and Analysis

As seen in Figure 16, a glideslope angle of 3° is targeted, meaning that from an altitude of approximately 3000 ft, the aircraft intercepts the runway such that the intercept occurs around 2000 ft above ground level,

roughly 10 s into the descent simulation. To facilitate this approach, a reduced-order model retaining the states for altitude h, true airspeed V_t, angle of attack α, pitch attitude θ, and pitch rate q was extracted from the full state-space matrices.

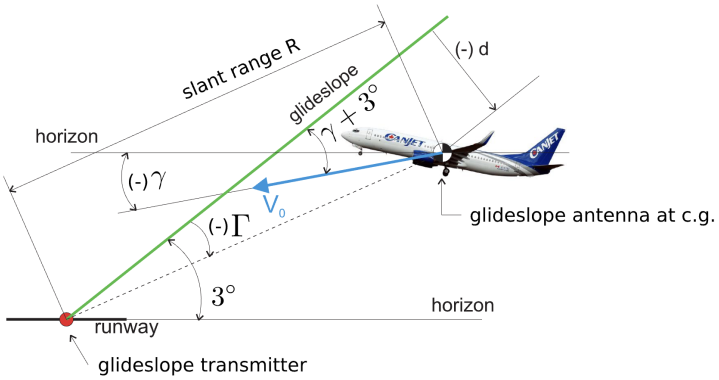

Figure 16: A typical aircraft glideslope.

Before constructing the glideslope controller, the engine and elevator were each modeled as a transfer function and subjected to saturation limits. As indicated in the problem statement, the engine dynamics are represented by

$$H_{\text{engine}}(s) = \frac{1}{s+1}, \quad \text{saturation limits: } \{1000\,\text{lb}, 19000\,\text{lb}\},$$

while the elevator actuator dynamics are given by

$$H_{\text{elevator}}(s) = \frac{20.2}{s+20.2}, \quad \text{saturation limits: } \{-25^\circ, 25^\circ\}.$$

Because the aircraft was trimmed at this flight condition, the saturation blocks must account for deviations from the trimmed thrust and elevator deflection, rather than the absolute values.

After integrating these actuator models within a Simulink subsystem, the glideslope controller was configured to achieve a gradual descent along a 3° glidepath. It generates reference signals for the pitch-attitude and velocity loops by translating altitude deviations into the required pitch attitude commands. The resulting glideslope-specific control loop is presented in Figure 17.

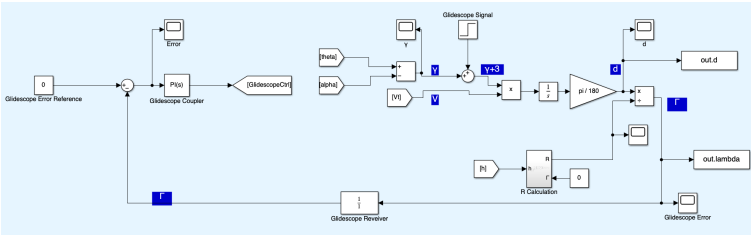

Figure 17: Simulink Control Diagram for Glideslope.

Again, a separate velocity feedback loop helps ensure that the aircraft does not overspeed or slow excessively. The final Simulink diagram includes the reduced model and subsystem blocks for the engine and elevator, each with their respective saturation limits.

In testing, the glideslope descent is initiated from level flight by intercepting the glidepath at $h \approx 2000$ ft above ground and holding a consistent 3° slope. The time response plots in Figures 18 through 22 depict the altitude, velocity, pitch angle, and angle of attack evolutions during this glide. Figure 23 further shows the deviation from the glideslope signal itself. The results confirm that the descent path is consistent with the chosen slope and that the limits on elevator and thrust deflections are not exceeded.

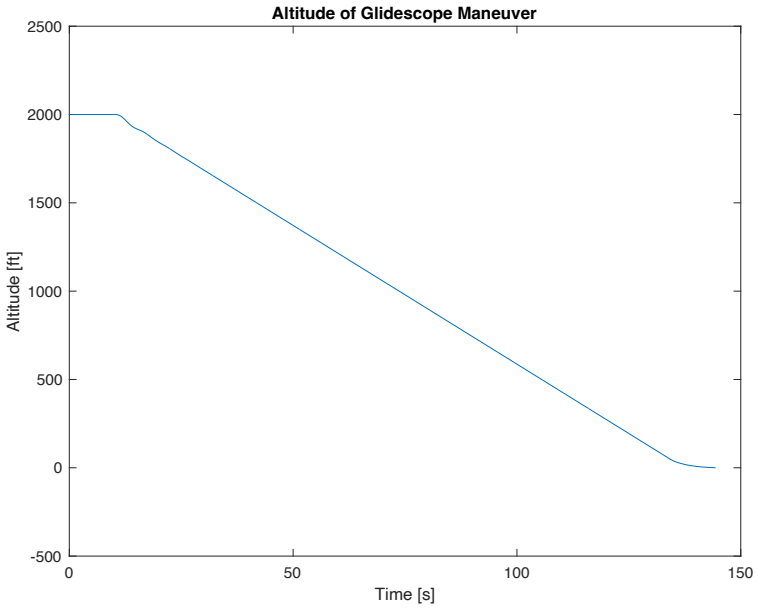

Figure 18: Altitude profile during glideslope descent.

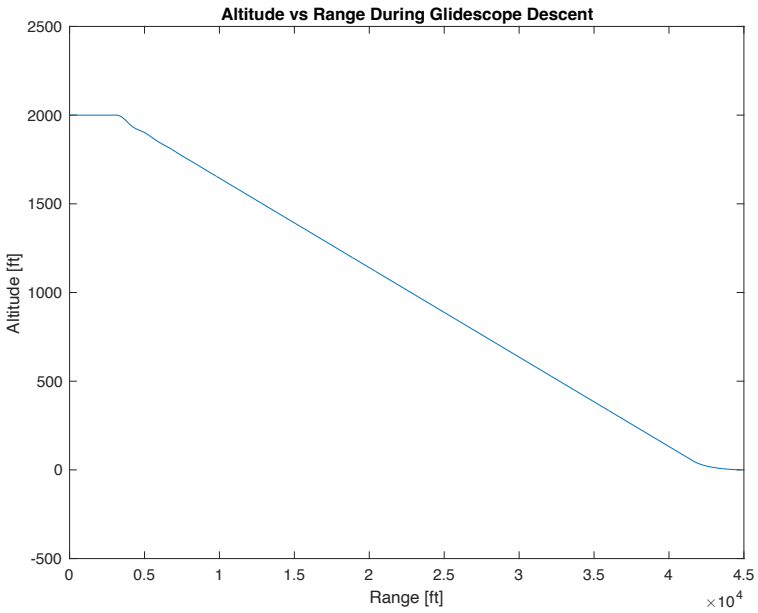

Figure 19: Range with respect to altitude during glideslope descent.

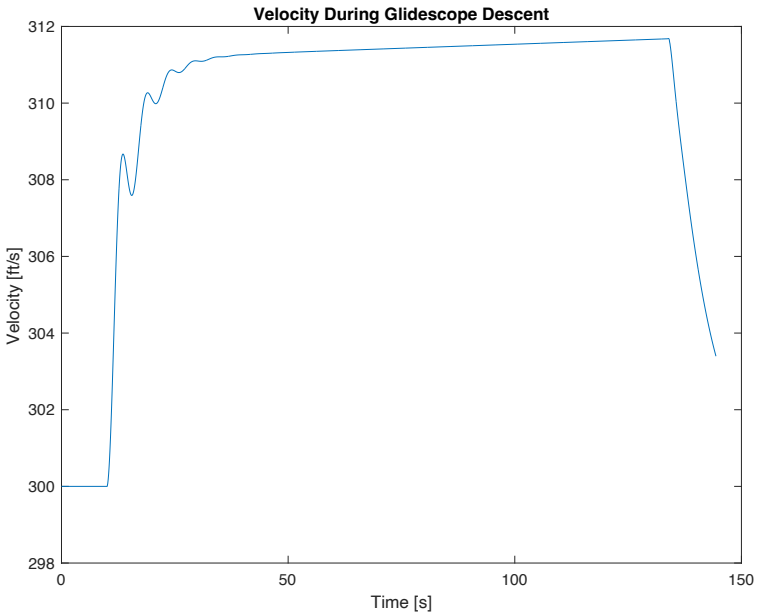

Figure 20: Velocity history throughout the glideslope intercept.

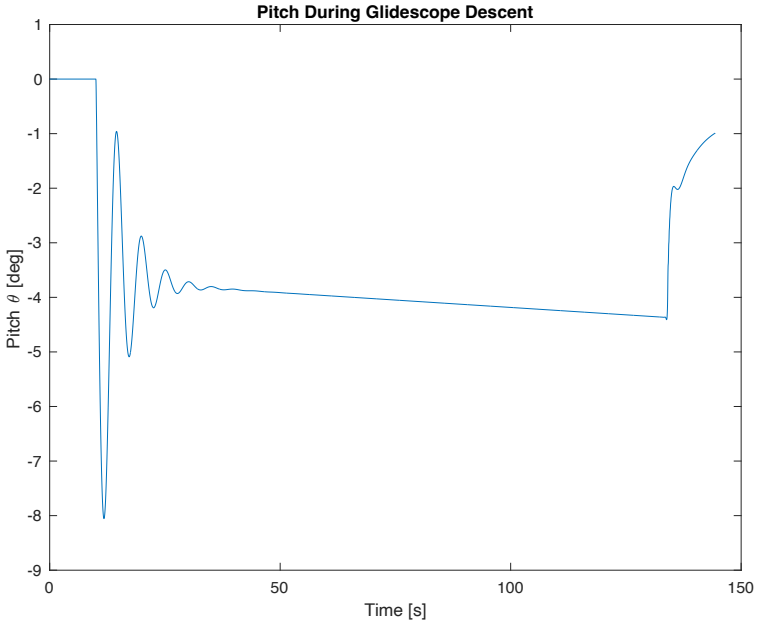

Figure 21: Pitch angle evolution as the aircraft descends on the glidepath.

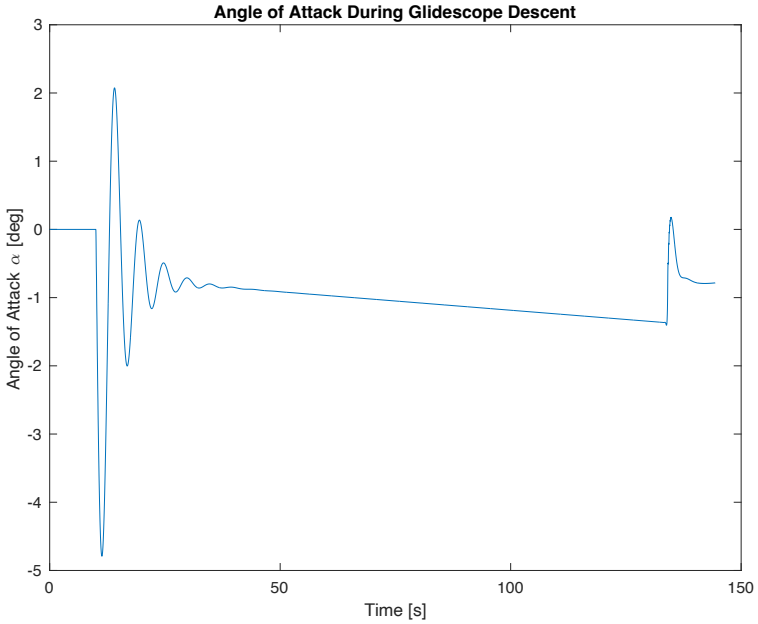

Figure 22: Angle of attack during the glideslope approach.

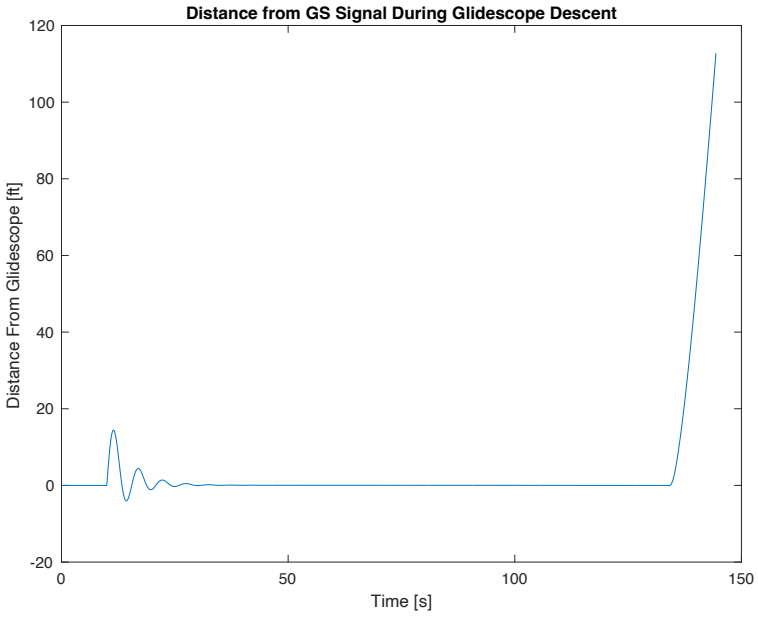

Figure 23: Deviation from the glideslope reference signal.

It is important to note that the F-16 settles onto its glideslope path within approximately 15 seconds of intercepting the glideslope signal, with only minor oscillations occurring in the transient as shown in the graph of distance from gllidescope signal.

5.3 Flare Controller Design and Analysis

Upon reaching a specified flare initiation point, the final descent must be converted into a gentle touchdown, with a vertical speed ideally constrained between 2 ft/s and 3 ft/s. The geometry of the flare path can be

determined by combining boundary conditions on altitude and vertical speed with an exponential decay law, which can be seen in Figure 24.

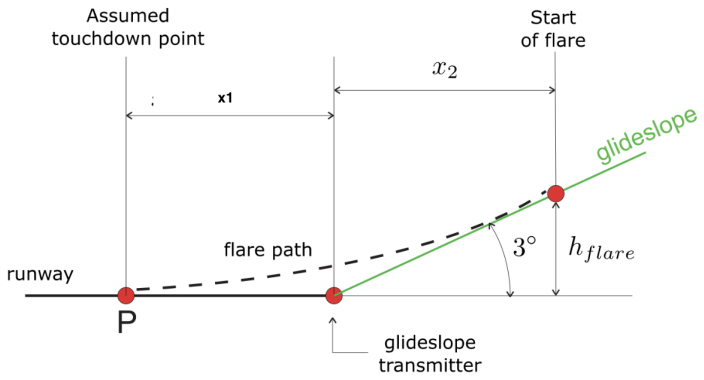

Figure 24: Display of Flare Maneuver Geometry and Parameters.

A symbolic solver was employed to find parameters that satisfy altitude continuity, vertical rate continuity, and the desired touchdown speed. The equations can be written as:

$$h(t) \;=\; h_0\, e^{-t/\tau}, \qquad \dot{h}(t) \;=\; -\frac{h_0}{\tau}\, e^{-t/\tau}.$$

By selecting τ and the final time T appropriately, we ensure that $\dot{h}(T) \approx -v_{\text{touch}} \approx -2.5\,\text{ft/s}$. In practice, the controller transitions from the glideslope logic to a flare logic at a predetermined flare altitude of a few hundred feet. The pitch-loop reference signal is adjusted to follow this exponentially decaying altitude profile. The design of the flare controller, which uses a lead compensator with a time-constant of 1.5, is shown in Figure 25.

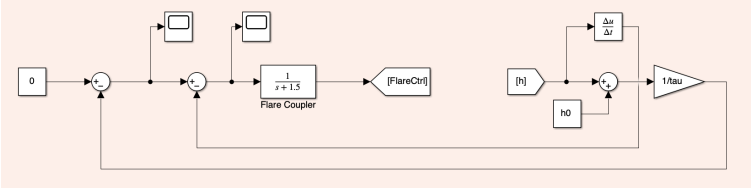

Figure 25: Simulink Diagram of Flare Controller.

To demonstrate the flare maneuver, the system is simulated until touchdown, with the altitude, vertical speed, and pitch angle monitored. The altitude of the aircraft as a function of the distance from the starting point can be seen in Figure 26 alongside the position of the runway.

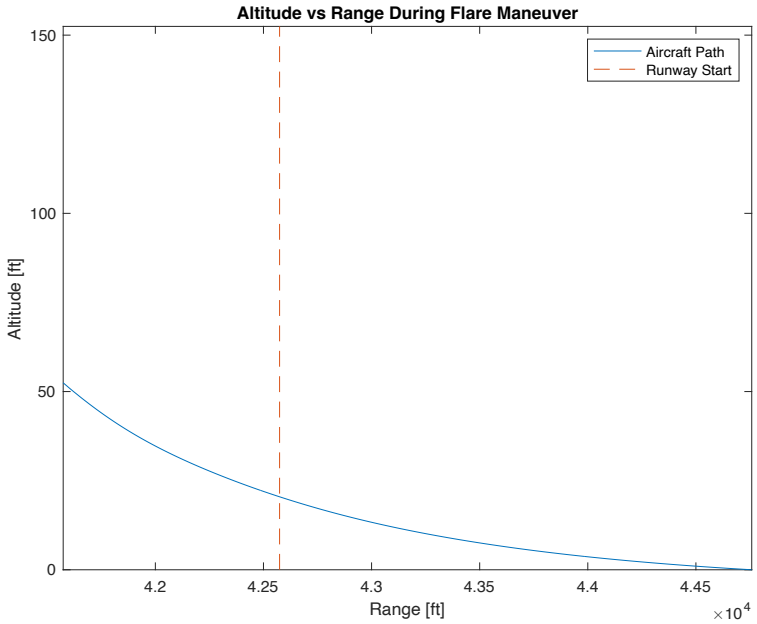

Figure 26: Display of Simulated Flare Maneuver Range and Altitude Plot.

This clearly shows that the F-16 is landing about 2000ft from the start of the runway, which would place it at about a quarter of an 8000ft runway. This is generally acceptable for aircraft considering typical runways can range from 8000ft to 13000ft, but for higher-precision landings on shorter runways a more aggressive flare geometry may be necessary. The vertical velocity can be seen in Figure 27

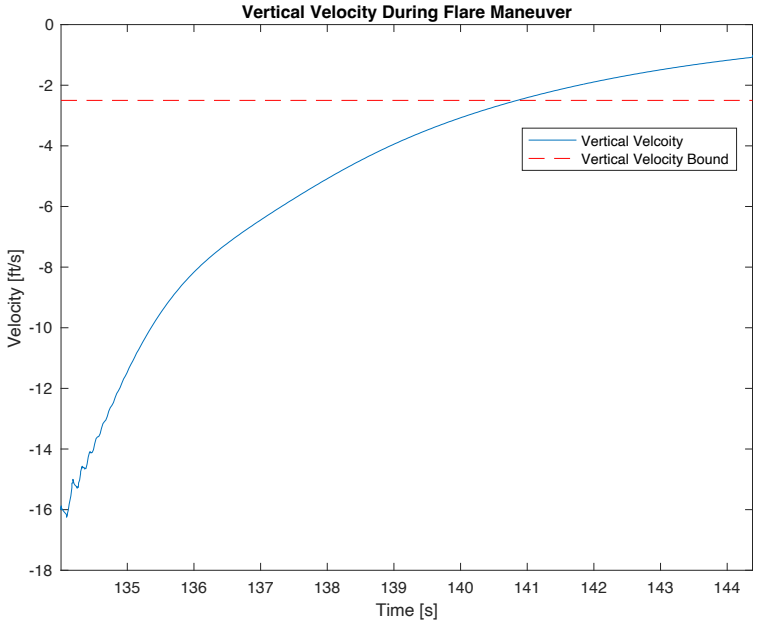

Figure 27: Display of Simulated Flare Maneuver Vertical Velocity and Time Plot.

The results show that the final descent rate is within the bound of -2.5 ft/s, fulfilling the requirement for a soft touchdown. The pitch, angle of attack, and linear velocity can be seen in Figure 28.

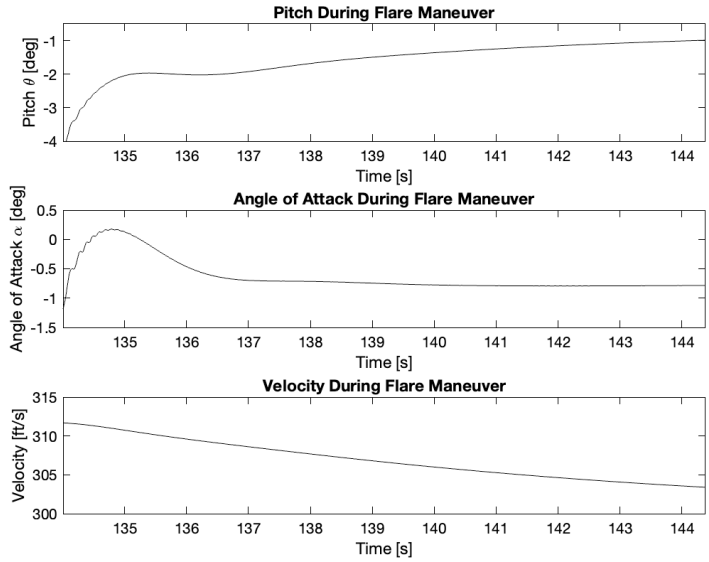

Figure 28: Display of Various Flight Parameters During Simulated Flare Maneuver.

The fact that these plots show negative pitch and angle of attack means the plane is slightly nose-up during landing, which is correct for a landing maneuver. It is also important to note that in all of these graphs, there are no abrupt transients that would stress the airplane and contribute to pilot error.

5.4 Maintaining Airspeed Around 300ft/s

The velocity plot in Figure 28 shows that the velocity is maintained around 300ft/s. Although a landing airspeed of 300 ft/s is somewhat high for an F-16 landing, this velocity was chosen to remain consistent with the validated linear models at this flight condition. Throughout the glideslope and flare, the controller commands modest adjustments in thrust to hold airspeed near this value, preventing energy deficits or excessive sink rates. The velocity loop's gain ensures quick corrections if airspeed strays from the set point, balancing the thrust request against elevator inputs during steep or abrupt sink-rate demands.

5.5 Discussion of Results

Overall, the glideslope controller successfully guides the aircraft from level flight onto a 3° descent path, maintaining stable altitude and velocity profiles. Transition into the flare phase is achieved by modifying the pitch reference to follow an exponential decay in altitude, culminating in a gentle touchdown at < 2.5 ft/s. Saturation blocks for the engine and elevator limit deflection deviations from the trimmed condition, ensuring that these commands remain physically feasible. Figures illustrating altitude, velocity, angle of attack, and pitch angle confirm that the system tracks the intended glidepath and flare geometry without risking overshoot or abrupt transitions. Any real-world implementation would include further validation in more comprehensive flight simulators, along with gain scheduling to accommodate a broader range of landing weights, wind conditions, and runway elevations.

The procedures described here highlight the core steps of glideslope capture and automatic flare control: trimming at a valid slow flight condition, forming a reduced-order model to target the most relevant states, designing and tuning the inner loops for pitch rate and velocity, and finally implementing an outer-loop logic for glideslope and flare path tracking. This layered approach ensures both stability and performance are met in each phase of the landing sequence.

6 Conclusion

This report has presented a comprehensive methodology for modeling, trimming, and controlling an F-16 aircraft under varying flight conditions and mission requirements. The study began by augmenting the standard low-fidelity model with an additional accelerometer measurement, revealing the importance of careful sensor placement and illuminating the non-minimum-phase characteristics that arise when normal acceleration feedback is considered. Trimming at the desired operating points was shown to be critical for extracting meaningful linear models, and the subsequent open-loop analysis provided insight into how phugoid, short-period, Dutch roll, aperiodic roll, and spiral modes each contribute to the overall dynamics of the unaugmented aircraft.

A pitch rate command system was then derived to satisfy both the CAP and Gibson criteria. By isolating the short-period mode, it was possible to design a proportional-plus-integral or lead-lag-based controller that aligns the natural frequency, damping ratio, and time constant with the specifications. While the system attained a short-period damping ratio meeting the CAP requirement and exhibited minimal dropback, practical constraints, such as pilot preference or increased agility demands, may still permit pitch rate overshoots somewhat larger than traditional Gibson guidelines recommend. This underscores the iterative nature of real-world control design, where performance trade-offs must balance agility and handling qualities.

Finally, a glideslope and flare controller was implemented to demonstrate an end-to-end path-following solution. By combining a reduced-order approach with state feedback for pitch rate and velocity, the F-16 was guided along a stable three-degree glidepath before transitioning to a gentle flare. Throughout these maneuvers, actuator and engine limitations, as well as moderate disturbances, were taken into account, ensuring that the designed controllers remained robust and feasible. Taken together, these results highlight how layered control architectures—ranging from inner-loop stabilization to outer-loop guidance—can integrate effectively to meet rigorous flight-envelope and pilot workload requirements. Ultimately, the methods outlined here illustrate how modern control design, combined with thorough trimming and linearization techniques, can achieve a high degree of precision and robustness across multiple flight phases for a high-performance aircraft such as the F-16.

References

[1] E. Smeur, *Practical Assignment AE4301P: Exercise Automatic Flight Control System Design*. Delft University of Technology, Delft, The Netherlands, 2024.

[2] M. V. Cook, *Flight Dynamics Principles*. Arnold, New York, USA, 1997.

[3] Nguyen, Ogburn, Gilbert, Kibler, Brown, and Deal, *Simulator Study of Stall / Post-Stall Characteristics of a Fighter Airplane with Relaxed Longitudinal Static Stability*. Technical Report 1538, NASA, December 1979.

[4] K. Ogata, *System Dynamics*. Prentice Hall, USA, 1998.

[5] J. Roskam, *Airplane Flight Dynamics and Automatic Flight Controls, Part I*. University of Kansas, Lawrence, USA, 1995.

[6] J. Roskam, *Airplane Flight Dynamics and Automatic Flight Controls, Part II*. University of Kansas, Lawrence, USA, 1995.

[7] R. S. Russell, *Non-linear F-16 Simulation using Simulink and Matlab*. Technical Report version 1.0, University of Minnesota, June 22, 2003.

[8] B. L. Stevens and F. L. Lewis, *Aircraft Control and Simulation*. John Wiley & Sons, Hoboken, New Jersey, third edition, 2016.

A MATLAB Code

Part 1-4: Setup and Declaration of Flight Conditions

Based on the specifications in the assignment, we are given:

```
alt = 10000; %ft
speed = 350; %ft

%Ensure there is a nlplant mex file for your system
mex nlplant.c
```

Part 5: Trim and Linearize the Model

Using Low-Fidelity Model For the to Find a_n characteristics Getting the Transfer Function for elevator input to x_a

```
a_n_index = 19;
x_as = [0, 5, 5.9, 6, 7, 15];
elev_index = 2;

model = 'LIN_F16Block_With_an';
open_system(model);

% Path to the block you want to modify
blockPath = [model '/a_x'];


for idx=1:size(x_as, 2)

x_a=x_as(idx);
set_param(blockPath, 'Value', num2str(x_a));

close_system(model, 1);
open_system(model);

sprintf("Values for x_{a} = %3.2f ft", x_a);

%With given an, find the
FindF16Dynamics_15000_500

%Grab State-Space Vectors
A = SS_lo.A;
B = SS_lo.B;
C = SS_lo.C;
D = SS_lo.D;

%Create state-space equations for a_n and elevator
A_an = A;
B_an = B(:, elev_index);
C_an = C(a_n_index, :);
D_an = D(a_n_index, elev_index);
```

```
if x_a == 0

end
%Convert ss->tf and show it
ss_an = ss(A_an,B_an,C_an, D_an);
disp('Transfer function from elevator to a_n:');
an_tf = tf(ss_an)
disp("Transfer function's Zeros:");
zero(an_tf)

[tf_num, tf_denom] = ss2tf(A_an,B_an,C_an, D_an);
an_tf = tf(tf_num, tf_denom);

%Find Dependencies and display ss-eqns
syms npos epos h phi theta psi Vt alpha beta p q r thrust_state
elevator_state aileron_state rudder_state Thrust_cmd Elevator_cmd
Aileron_cmd Rudder_cmd
input_names = [npos; epos; h; phi; theta; psi; Vt; alpha; beta; p; q; r;
thrust_state;  elevator_state; aileron_state; rudder_state];
output_names = [Thrust_cmd; Elevator_cmd; Aileron_cmd ;Rudder_cmd];

dependencies = input_names(not (C_an==0));
ss_an_elev = vpa(C_an * input_names + D(elev_index, :) * output_names, 3)

%Display Negative Step Response
figure()
times = linspace(0, 2, 200);
u_step = ones(size(times));
lsim(-an_tf, u_step, times)
ylim([-.1, .8])
xlim([0,2])
title1 = sprintf("Step Response With x_{a} = %3.2f ft", x_a);
title(title1)
grid on

end
```

Part 6a: Open Loop Analysis of Low-Fidelity System

```
%Switch to Given Flight Condition
FindF16Dynamics_10000_350
```

Part 6b (getting Eigenmotion Responses)

Getting Lateral State-Space Matrices

```
%Grab State-Space Vectors
A = SS_lo.A;
```

```
B = SS_lo.B;
C = SS_lo.C;
D = SS_lo.D;

%Lateral States:
%
state_indeces_lat = [4 9 10 12 15 16];
state_names_lat = string(SS_lo.StateName(state_indeces_lat));
input_indeces_lat = [3 4];
input_names_lat = string(SS_lo.InputName(input_indeces_lat));

A_lat_reduced = A(state_indeces_lat, state_indeces_lat);
B_lat_reduced = B(state_indeces_lat, input_indeces_lat);
C_lat_reduced = C(state_indeces_lat, state_indeces_lat);
D_lat_reduced = zeros(size(B_lat_reduced));

A_ac_lat = A_lat_reduced(1:4, 1:4);
B_ac_lat = A_lat_reduced(1:4, 5:6);
C_ac_lat = C_lat_reduced(1:4, 1:4);
D_ac_lat = zeros(4,2);

lat_tfs = tf(ss(A_lat_reduced, B_lat_reduced, C_lat_reduced,
D_lat_reduced)); %W/ actuators
lat_tfs_noAc = tf(ss(A_ac_lat, B_ac_lat, C_ac_lat, D_ac_lat)); % w/o
actuators

%Getting Longitudinal State-Space Matrices

state_indeces_long = [5 7 8 11 13 14];
state_names_long = string(SS_lo.StateName(state_indeces_long));
input_indeces_long = [1 2];
input_names_long = string(SS_lo.InputName(input_indeces_long));

A_long_reduced = A(state_indeces_long, state_indeces_long);
B_long_reduced = B(state_indeces_long, input_indeces_long);
C_long_reduced = C(state_indeces_long, state_indeces_long);
D_long_reduced = zeros(size(B_long_reduced));

A_ac_long = A_long_reduced(1:4, 1:4);
B_ac_long = A_long_reduced(1:4, 5:6);
C_ac_long = C_long_reduced(1:4, 1:4);
D_ac_long = zeros(4,2);
long_tfs = tf(ss(A_long_reduced, B_long_reduced, C_long_reduced,
D_long_reduced));
long_tfs_noAc = tf(ss(A_ac_long, B_ac_long, C_ac_long, D_ac_long)); %Using
%Reduced Signal

%Extracting Longitudinal Eigenmodes
longitudinal_eig = eig(A_ac_long)
lateral_eig = eig(A_ac_lat)
```

```
%Derriving damping ratios, half-amplitude times for lateral and
%longitudinal eigenmodes
phugoid = min(longitudinal_eig); %Note that min/max measure magnitude in
imaginary #s
phugoid_z = abs(phugoid)/(2 * abs(imag(phugoid)))
phugoid_omega = abs(phugoid)
phugoid_t12 = log(0.5)/(- phugoid_z * phugoid_omega)
%Plotting Phugoid: Elevator to V and to theta
v_elevator = long_tfs(2, 2);
figure()
subplot(2, 1, 1)
step(v_elevator)
title("Elevator Defelction to Velocity")
xlabel("t")
ylabel("V[ft/s]")

theta_elevator = long_tfs(1, 2);
subplot(2, 1, 2)
step(theta_elevator)
title("Elevator Defelction to Pitch")
xlabel("t")
ylabel("\theta [deg]")

%Short-Period Characteristics and Plots
short_period = max(longitudinal_eig);
sp_z = abs(short_period)/(2 * abs(imag(short_period)))
sp_omega = abs(short_period)
sp_t12 = log(0.5)/(- sp_z * sp_omega)
%Plotting Short-Period: Elevator to alpha and elevator to q
alpha_elevator = long_tfs(3, 2);
figure()
subplot(2, 1, 1)
step(alpha_elevator)
title("Elevator Defelction to Alpha")
xlabel("t")
ylabel("\alpha [deg]")

q_elevator = long_tfs(4, 2);
subplot(2, 1, 2)
step(q_elevator)
title("Elevator Defelction to Pitch Rate")
xlabel("t")
ylabel("q [deg/s]")

%Dutch-Roll Characteristics and Plots
dutch_roll = lateral_eig(~imag(lateral_eig)==0); dutch_roll=dutch_roll(1);
dr_z = abs(dutch_roll)/(2 * abs(imag(dutch_roll)))
dr_omega = abs(dutch_roll)
dr_t12 = log(0.5)/(- dr_z * dr_omega)
```

```
%Plotting Dutch Roll: Lateral Rudder Impulse Responses for all states
figure()
for idx=1:4
    tf_i = lat_tfs(idx, 2);
    subplot(2, 2, idx)
    impulseplot(tf_i)
    xlabel("t")
    ylabel(state_names_lat(idx))
    title(sprintf("%s Impulse Response to Aileron Deflection",
state_names_lat(idx)))
end
sgtitle("Dutch Roll Responses")
%Aperiodic Roll Characteristics and Plots
aperiodic_roll = min(lateral_eig(imag(lateral_eig)==0));
aperiodic_thalf = log(0.5)/aperiodic_roll
%Plotting Aperiodic Roll: Aileron to Roll rate output
aileron_to_p = lat_tfs(3,1);
figure()
step(aileron_to_p)
title("Aileron to Roll Rate(p) Step Response")
xlabel("time")
ylabel("p [deg/s]")

%Spiral Roll Characteristics and Plots
spiral_roll = max(lateral_eig(imag(lateral_eig)==0));
spiral_thalf = log(0.5)/spiral_roll
s = tf('s'); spiral_tf = 1/(s-spiral_roll);

%Plotting Spiral Roll: Time Response to Initial Roll Angle
x0 = 20 * pi / 180; %20deg roll
t = linspace(0, 800, 1000); % 350 seconds

%use initial() function to simulate system with initial condition
[y, t_out, x] = initial(ss(spiral_tf), x0, t);

%Plot time response
figure()
plot(t_out, y(:, 1) * 180 / pi); % Convert radians to degrees
xlabel('Time (seconds)');
ylabel('\phi [deg]');
title('Roll Reponse With Initial 20° Roll Angle');
```

Part 7: Designing Pitch Rate Feedback Controller

```
%Step 1: Constructing Reduced Transfer Function For Pitch-Rate Feedback
%States: Angle of Attack and pitch rate
state_indeces_long = [8 11 14];
state_names_long = string(SS_lo.StateName(state_indeces_long));
input_indeces_long = 2;
input_names_long = string(SS_lo.InputName(input_indeces_long));
```

```
%Get the full 2-state transfer function with actuator dynamics
A_long_p4 = A(state_indeces_long, state_indeces_long);
B_long_p4 = B(state_indeces_long, input_indeces_long);
C_long_p4 = C(state_indeces_long, state_indeces_long);
D_long_p4 = zeros(size(state_indeces_long, 1), size(input_indeces_long,1));

%Reduce to only Angle of Attack and Pitch Rate
A_ac_p4 = A_long_p4([1,2],[1,2]);
B_ac_p4 = A_long_p4([1,2],3);
C_ac_p4 = C_long_p4([1,2],[1,2]);
D_ac_p4 = C_long_p4([1,2],3);
lat_tfs_p4 = tf(ss(A_ac_p4, B_ac_p4,C_ac_p4,D_ac_p4))

%constructing the transfer function from Short-Period Pole
s = tf("s");
sp_response_poles = sp_omega^2/(s^2 + 2 * sp_z * sp_omega * s + sp_omega^2)

q_elev_p4 = lat_tfs_p4(2);
alpha_elev_p4 = lat_tfs_p4(1);

%Load Numerator & Denominator for simulink
[num_sp, den_sp] = tfdata(q_elev_p4, 'v');
%Step 2: Comparing pitch rate of 4-state+actuator with 2-state without
actuator
figure()
step(q_elevator)
hold on
step(-q_elev_p4)
step(sp_response_poles)
hold off
legend("4-state with actuator", "2-State Short-Period Response", "Derrived
From Short-Period Poles")
title("Comparing Multiple Short-Period Damping Models")
%From Plot and control system logic, phugoid is dominant pole, as its
%response takes much longer to settle than the short-period.

%ACTUAL ANSWER TO Q7:
%{
From the simulink diagram, you can see that zero-placement is how you move
the poles to exactly where they need to be for damping. The current
closed-loop zero is too close to the imaginary axis, so the poles bend
toward the imaginary axis instead of bending away from the imaginary axis.
By cancelling the real zero close to the imaginary axis and replacing it
with a real negative zero on the order of 10^12, the poles can be tuned to
the exact specifications
%}
```

```
%Alternative: Create and tune a PI Controller
```

```
%THIS IS NOT USED. WE USE THE SIMULINK MODEL INSTEAD
alt = 10000; %ft
speed = 350; %ft

V = speed * 0.3048; %m/s
cn1 = -num_sp(2); cn2 = -num_sp(3); cd1 = den_sp(2); cd2 = den_sp(3);
w_nsp_req = 0.03 * V
time_const = 1/(0.75 * w_nsp_req)
z_sp_req = 0.5
desiredPole = -z_sp_req * w_nsp_req + 1i *  w_nsp_req * sqrt(1-z_sp_req^2);
a = real(desiredPole); b = imag(desiredPole);
Amat = [1, cn1, 0; (a^2+b^2), 0, cn2; 2*a, cn2, cn1]
B = [cd1+2*a; 0;cd2-(a^2 + b^2)]
X = inv(Amat) * B; Kp = X(2); Ka = X(3);

%Fine-tuning gains manually to achieve control parameters
Kp = Kp + (-0.25);
Ka = Ka + (-0.65);

figure()
rlocus(-q_elev_p4)
sgrid(z_sp_req, w_nsp_req)
title("Root Locus of Short-Period Response")


%Q3: Create PI controller for pitch rate: (note kp = -2.2 ki=-6.7 were
%manually tuned values)
%Kp2 = (1.3 - 2 * a - 10)/3.76; %Manually Calculated for pole placement
%Ka2 = (-(a^2 + b^2) * 10 )/1.9; %Manually Calculated for pole placement
pitch_controller= Kp + Ka / tf('s');
[num_pc, den_pc] = tfdata(pitch_controller, 'v');
controlled_cl_tf = feedback(pitch_controller * q_elev_p4, 1, -1) %
controlled closed-loop transfer function

%Plot the step response of feedback vs vs open-loop
figure()
step(controlled_cl_tf)
hold on
step(-q_elev_p4)
hold off
legend("PI Controlled Feedback System", "Open Loop")
title("Step Response of PI Controlled Short-Period System vs Open Loop")

%Q4: Check Designed PI Controller for Damping and Natural Frequency
cl_poles = pole(controlled_cl_tf);
sprintf("Information about closed-loop transfer function damping, natural
Frequency, and tiem constant")
damp(cl_poles)

%{
```

```
Q5: The pole/zero-cancellation or lead-lag prefilter must be placed outside
of
the feedback loop because it is trying to effect the closed-loop poles and
zeros. Placing it inside of the feedback loop would not cancel the pole
correctly, as the filter would be affected by the feedback dynamics. In
addition, the purpose of the lead-lag filter would be to adjust the signal
pre-feedback.
%}


Kff = 0;

%Q7: CAP, assuming damping and natural frequency of the complex poles
g = 9.81; %m/s^2
tau = 0.497; %s (from damp() function above)
wsp = 3.22; %rad/s
z = 0.625;
CAP = wsp^2 * g * tau /V

%For Gibson Criterion
DB_qss = tau - 2 * z/wsp
info = stepinfo(controlled_cl_tf);
qm_qs = info.Overshoot

%Finally: Save workspace for Simunlink Model
save("PitchController")
```

Part 8: Designing Glidescope/Flare Controller

```
%Step 1: Trim and Linearize about alt=

FindF16Dynamics_5000_300
alt = 5000;
v = 300;

A = SS_lo.A;
B = SS_lo.B;
C = SS_lo.C;
D = SS_lo.D;
```

Step 2: Reduce to necessary states:

alt h, airspeed v, angle of attack a, pitch attitude angle theta, and pitch rate q

```
state_indeces_long = [3 5 7 8 11 13 14];
state_names_long = string(SS_lo.StateName(state_indeces_long));
input_indeces_long = [1, 2];
input_names_long = string(SS_lo.InputName(input_indeces_long));
```

```
%Get the full 2-state transfer function with actuator dynamics
A_long_p8 = A(state_indeces_long, state_indeces_long);
B_long_p8 = B(state_indeces_long, input_indeces_long);
C_long_p8 = C(state_indeces_long, state_indeces_long);
D_long_p8 = zeros(size(state_indeces_long, 1), size(input_indeces_long,1));

%Reduce to only state, not actuator
A_ac_p8 = A_long_p8(1:5,1:5);
B_ac_p8 = A_long_p8(1:5,6:7);
C_ac_p8 = C_long_p8(1:5,1:5);
D_ac_p8 = C_long_p8(1:5,6:7);

glidescope_statespace = ss(A_ac_p8, B_ac_p8,C_ac_p8,D_ac_p8);
glidescope_tf = tf(glidescope_statespace);
% Step 3-4 in Simulink:
%Step 5: Additional Variables for the Controller
gamma_to_theta = 1- glidescope_tf(4,2)/glidescope_tf(2,2);
[num_gt, denom_gt] = tfdata(gamma_to_theta, "v");

Kv = 100; %Velocity Gain. Manually Tuned

%Flare Geometry Constants
x1 = 1000; %ft (Chosen Somewhat Arbitrarily)
v_touch = 2.5; %ft/s

syms hf h0 x2 tau T
eq1 = hf - h0 == x2 * tand(3);
eq2 = -v * sind(3) == -hf / tau;
eq3 = hf * exp(-T/tau) - h0 == 0;
eq4 = -(hf/tau) * exp(-T/tau) == -v_touch;
eq5 = x1 + x2 == v * T;

sol = vpasolve([eq1, eq2, eq3, eq4, eq5], [hf, h0, x2, tau, T]);

hf = double(sol.hf);
h0= double(sol.h0);
x2= double(sol.x2);
tau= double(sol.tau);
T = double(sol.T);

%Finally, Save Workspace so you can load it in later
save("glidescope")
```

Making plots from glidescope info

```
%first load the workspace and run the simulation
load("glidescope")
file = "Glidescope_Flare_ActuallyGood";
open_system(file)
```

```
%Now gather the outputs from the SimOutput struct
outputs = sim(file);
tout = outputs.tout;
close_system(file, 1)

%State Variables/Derivatives
alpha = outputs.alpha.Data;
pos = outputs.pos.Data; pos = max(pos) - pos;
h = outputs.h.Data;
q = outputs.q.Data;
theta = outputs.theta.Data;
vt = outputs.vt.Data;
vy = outputs.vy.Data;

%Glidesscope-Specific Variables
d = outputs.d.Data;
lambda = outputs.lambda.Data;

%Control efforts
elevator = outputs.elevator.Data;
thrust = outputs.thrust.Data;

%Some things we need to calculate for graphing:
%1) time of switchover
indSwitch = h-hf == min(abs(h-hf));
tSwitch = tout(indSwitch);

%2) Runway x Val(assumes runway starts at glidescope transmitter
runwayDist = pos(indSwitch) + hf/tand(3);

%Pt 1: Glidescope
%1: Altitude vs Time
figure()
plot(tout, h)
title("Altitude of Glidescope Maneuver")
xlabel("Time [s]")
ylabel("Altitude [ft]")
exportgraphics(gca,'Figures/GS_Alt_v_Time.pdf')

%2: Speed vs Time
figure()
plot(tout, vt)
title("Velocity During Glidescope Descent")
xlabel("Time [s]")
ylabel("Velocity [ft/s]")
exportgraphics(gca,'Figures/GS_Speed_v_Time.pdf')

%3: Pitch vs Time
figure()
plot(tout, theta)
```

```
title("Pitch During Glidescope Descent")
xlabel("Time [s]")
ylabel("Pitch \theta [deg]")
exportgraphics(gca,'Figures/GS_Pitch_v_Time.pdf')

%4: AOA vs Time
figure()
plot(tout, alpha)
title("Angle of Attack During Glidescope Descent")
xlabel("Time [s]")
ylabel("Angle of Attack \alpha [deg]")
exportgraphics(gca,'Figures/GS_AOA_v_Time.pdf')

%5: Distance from GS vs Time
figure()
plot(tout, d)
title("Distance from GS Signal During Glidescope Descent")
xlabel("Time [s]")
ylabel("Distance From Glidescope [ft]")
exportgraphics(gca,'Figures/GS_Dist_v_Time.pdf')

%6: Range vs Altitude
figure()
plot(pos, h)
title("Altitude vs Range During Glidescope Descent")
xlabel("Range [ft]")
ylabel("Altitude [ft]")
exportgraphics(gca,'Figures/GS_Range_v_Alt.pdf')

%Pt 2: Flare
figure()
plot(pos, h)
hold on
plot([runwayDist, runwayDist], [0, hf+100], "--")
hold off
title("Altitude vs Range During Flare Maneuver")
xlabel("Range [ft]")
ylabel("Altitude [ft]")
ylim([0, hf+100])
xlim([pos(indSwitch), pos(end)])
legend("Aircraft Path", "Runway Start")
exportgraphics(gca,'Figures/Flare_Range_v_Alt.pdf')

%Vertical velocity during flare maneuver
figure()
plot(tout, vy)
title("Vertical Velocity During Flare Maneuver")
xlabel("Time [s]")
ylabel("Velocity [ft/s]")
xlim([tout(indSwitch), tout(end)])
```

```
exportgraphics(gca,'Figures/Flare_Vert_Velocity.pdf')

%Mixed Plot of Pitch, AOA, and Velocity
figure()
subplot(3, 1, 1)
plot(tout, theta)
title("Pitch During Flare Maneuver")
xlabel("Time [s]")
ylabel("Pitch \theta [deg]")
xlim([tout(indSwitch), tout(end)])
ylim([-4, -0.5])

subplot(3, 1, 2)
plot(tout, alpha)
title("Angle of Attack During Flare Maneuver")
xlabel("Time [s]")
ylabel("Angle of Attack \alpha [deg]")
xlim([tout(indSwitch), tout(end)])
ylim([-1.5, 0.5])

subplot(3, 1, 3)
plot(tout, vt)
title("Velocity During Flare Maneuver")
xlabel("Time [s]")
ylabel("Velocity [ft/s]")
xlim([tout(indSwitch), tout(end)])
ylim([300, 315])
exportgraphics(gca,'Figures/Flare_Mixed_Plots.pdf')
```

B Glideslope and Flare Full Simulink Diagram