Autopilot

Team 14

June 4, 2025

Submitted to:
Dr. Osamah Mohammedy
Eng. Mahmoud Gamal

Name	Section	B.N.
Ahmed Salah Hammad Ahmed	1	4
Ola Mostafa AbdelMo'ty	1	36
Fouad Tarek Mostafa	1	39
Mohammed Ahmed Roqayah	2	4
Mohammed Osamah AbdelRasheed	2	6
Mohamed El-Shahat Saad Mohammed	2	7

Contents

Part I

Research Review

1 Autopilot Introduction

An autopilot is a system used to control the path of an aircraft, marine craft, or spacecraft without requiring constant manual control by a human operator. Its main objective is to assist the operator's control of the vehicle, allowing the operator to focus on broader aspects of operations, such as monitoring the trajectory, weather, and onboard systems.

The first aircraft autopilot was developed by Sperry Corporation in 1912. The main purpose of early autopilots was to stabilize the aircraft and return it to the desired flight attitude after any disturbance. Such an autopilot was installed in a Glenn H. Curtis flying boat and first tested in late 1912. The autopilot, using gyros to sense the deviation of the aircraft from the desired attitude and servo motors to activate the elevators and ailerons, was developed and built by the Sperry Gyroscope Company of New York under the direction of Dr. E. A. Sperry (Blakelock, 1991, p. 1).

2 Inputs and Outputs of an Autopilot from a control systems perspective

From a control systems perspective, an autopilot system onboard an airplane processes various inputs and generates corresponding outputs:

Inputs: Desired/Commanded States: These include target parameters such as heading, altitude, airspeed, and flight path, which can be set by the pilot or pre-programmed into the flight management system. Sensor Data: Information from onboard sensors, including gyroscopes, accelerometers, magnetometers, GPS, barometers, and airspeed indicators, providing real-time data on the aircraft's current state.

Outputs: Actuator Commands: Signals sent to control surface actuators (e.g., ailerons, elevators, rudder) and throttle controls to adjust the aircraft's attitude, altitude, and speed to match the desired states.

3 Role of the pilot in an airplane

In an airplane equipped with an autopilot, the pilot's role includes:

Monitoring: Continuously overseeing the autopilot's performance to ensure it is operating correctly and making appropriate adjustments as necessary.

Decision-Making: Making strategic decisions regarding flight path, altitude changes, and responses to air traffic control instructions.

Manual Control: Taking manual control of the aircraft during critical phases of flight, such as takeoff and landing, or in situations where autopilot use is not appropriate.

System Management: Managing and programming the flight management system and autopilot settings to ensure alignment with the intended flight plan.

4 Difference between an Autopilot & SAS (stability augmentation system)

An autopilot is a system designed to control the trajectory of an aircraft without requiring constant hands-on control from a human operator. It performs various functions such as maintaining a set altitude, keeping a constant airspeed, and following a pre-set route using navigation aids like VOR (VHF Omnidirectional Range) and ILS (Instrument Landing System). Additionally, it manages the flight plan and optimizes the route through flight management systems.

On the other hand, a Stability Augmentation System (SAS) is intended to enhance the stability of an aircraft, especially in turbulent conditions. It helps stabilize the aircraft against pitch, roll, and yaw disturbances. Key features of SAS include short-term attitude stabilization, augmentation of pilot input to reduce workload, and providing stability during manual control to allow the pilot to focus on other tasks.

The primary differences between Autopilot and SAS are as follows:

1. Functionality: While the Autopilot performs a wide range of functions including navigation and maintaining specific flight parameters, SAS primarily focuses on stabilizing the aircraft.

2. Control: The Autopilot can take over complete control of the aircraft for extended periods, whereas SAS assists the pilot without taking over full control.

3. Complexity: Autopilot systems are generally more complex and sophisticated compared to SAS.

5 Role of the onboard sensors like (GPS, gyroscopes, ..etc.)

Sensor	Quantities Measured	Typical Sampling Rates
GPS Receiver	Provides position (latitude, longitude, altitude) and ground speed.	1 Hz to 10 Hz
Gyroscopes	Measure angular velocity around the aircraft's axes (roll, pitch, yaw).	50 Hz to 1000 Hz
Accelerometers	Measure linear acceleration along the aircraft's axes.	50 Hz to 1000 Hz
Magnetometers	Measure the Earth's magnetic field to determine heading.	10 Hz to 100 Hz
Barometric Altimeter	Measures static air pressure to determine altitude above sea level.	1 Hz to 10 Hz
Airspeed Indicator	Measures dynamic air pressure to determine the aircraft's airspeed.	1 Hz to 10 Hz
Inertial Measurement Unit (IMU)	Combines gyroscopes, accelerometers, and sometimes magnetometers to provide comprehensive motion data.	50 Hz to 1000 Hz

6 Unmeasured States in the airplane equations state space model

If a particular state is not directly measured by a sensor, it can be estimated using sensor fusion techniques and mathematical models. One common method is the use of a Kalman filter, which combines data from multiple sensors and applies a predictive model to estimate the unmeasured state with a certain level of confidence. This approach allows for the reconstruction of unmeasured states based on available sensor data and system dynamics.

7 Fly-by-Wire flight control system

Fly-by-Wire (FBW) is the generally accepted term for those flight control systems which use computers to process the flight control inputs made by the pilot or autopilot, and send corresponding electrical signals to the flight control surface actuators. This arrangement replaces mechanical linkage and means that the pilot inputs do not directly move the control surfaces. Instead, inputs are read by a computer that in turn determines how to move the control surfaces to best achieve what the pilot wants in accordance with which of the available Flight Control Laws is active.

8 Open Source Autopilot Software to be used on UAVs

PX4: PX4 is an open source flight control software for drones and other unmanned vehicles.

ArduPilot: ArduPilot is open source software that runs on a wide range of hardware. ArduPilot enables the creation and use of trusted, autonomous, unmanned vehicle systems for the peaceful benefit of all.

9 Autopilot Hardware that can be used on UAVs

Pixhawk: A popular open-source flight controller hardware platform that supports various autopilot software, including ArduPilot and PX4. It features a range of sensors and interfaces suitable for UAV applications.

Navio2: A hardware add-on for the Raspberry Pi that turns it into a full-featured autopilot. It supports ArduPilot software and provides IMU, barometer, GPS, and other essential sensors for UAV control.

Part II

Flight Mechanics Review

10 General Rigid Body Dynamics (RBD) equations in 3D Space

These equations are divided into translational motion and rotational motion:

10.1 Translational Motion (Newton's Second Law)

$$\sum \vec{F} = m\vec{a} = m\frac{\partial \vec{v}}{\partial t}$$

Where $\sum \vec{F}$: Total external force acting on the body. m: mass of the rigid body. $\vec{a}$: linear acceleration of the center of mass In component form:

$$\begin{aligned} \sum \vec{F_x} &= m\vec{a_x} \\ \sum \vec{F_y} &= m\vec{a_y} \\ \sum \vec{F_z} &= m\vec{a_z} \end{aligned}$$

Which eventually reduce to:

$$\begin{aligned} X - mgS\theta &= m\left(\dot{u} + qw - rv\right) \\ Y + mgC\theta S\phi &= m\left(\dot{v} + ru - pw\right) \\ Z + mgC\theta C\phi &= m\left(\dot{w} + pv - qu\right) \end{aligned} \tag{1}$$

10.2 Rotational Motion (Euler's Equations of Motion)

$$\sum \vec{M} = I\vec{\alpha} + \vec{\omega} \times (I\vec{\omega})$$

Where $\sum \vec{M}$: Total external moment acting on the body about its center of mass. I: Moment of inertia tensor. α: angular acceleration. $\vec{\omega}$: angular velocity. In component form for principal axes:

$$\begin{aligned}
\sum \vec{M}_x &= I_x\dot{\omega}_x - (I_y - I_z)\,\omega_y\omega_z \\
\sum \vec{M}_y &= I_y\dot{\omega}_y - (I_z - I_x)\,\omega_z\omega_x \\
\sum \vec{M}_z &= I_z\dot{\omega}_z - (I_x - I_y)\,\omega_x\omega_y
\end{aligned}$$

Which reduce to:

$$\begin{aligned}
L &= I_x\dot{p} - I_{xz}\dot{r} + qr\,(I_z - I_y) - I_{xz}pq \\
M &= I_y\dot{q} + rp\,(I_x - I_z) + I_{xz}\,(p^2 - r^2) \\
N &= -I_{xz}\dot{p} + I_z\dot{r} + pq\,(I_y - I_x) + I_{xz}qr
\end{aligned} \tag{2}$$

10.3 Kinematic Equations for Position

$$\begin{aligned}
\dot{x}_E &= C\theta C\psi\,u_b + (S\phi S\theta C\psi - C\phi S\psi)\,v_b + (C\phi S\theta C\psi + S\phi S\psi)\,w_b \\
\dot{y}_E &= C\theta S\psi\,u_b + (S\phi S\theta S\psi + C\phi C\psi)\,v_b + (C\phi S\theta S\psi - S\phi C\psi)\,w_b \\
\dot{z}_E &= -S\theta\,u_b + S\phi C\theta\,v_b + C\phi C\theta\,w_b
\end{aligned} \tag{3}$$

10.4 Kinematic Equations for Orientation

The kinematic equations relate angular velocity to orientation changes:

$$\begin{aligned}
\dot{\phi} &= p + q\sin(\phi)\tan(\theta) + r\cos(\phi)\tan(\theta) \\
\dot{\theta} &= q\cos(\phi) - r\sin(\phi) \\
\dot{\psi} &= q\tfrac{\sin(\phi)}{\cos(\theta)} + r\tfrac{\cos(\phi)}{\cos(\theta)}
\end{aligned} \tag{4}$$

Where: p, q, r: Angular velocity components about the body-fixed axes. while ϕ, θ, ψ: Euler angles (roll, pitch, yaw).

The 12 equations of motion consist of:

- 6 Kinetics Equations:
  - 3 translational equations (F_X, F_Y, F_Z); equations 1.
  - 3 Rotational equations (M_X, M_Y, M_Z); equations 2.
- 6 Kinematics Equations:
  - 3 for position $(x\ ,\ y\ ,\ z)$; equations 3.
  - 3 for rotation (ϕ, θ, ψ); equations 4.

11 Equations added to the (RBD) equations to form the Fixed wing Airplanes (EOM)

To adapt the general RBD equations for fixed wing airplanes, additional forces and moments specific to aerodynamics are introduced:

1. Aerodynamic Forces:
   (a) Lift $(L = C_L qS)$: Acts perpendicular to the relative airflow.
   (b) Drag $(D = C_D qS)$: Acts opposite to the relative airflow.
   (c) Side force$(Y = C_Y qS)$: Acts laterally.
2. Aerodynamic Moments:
   (a) Rolling moment$(L = C_l qSb)$
   (b) Pitching moment$(M = C_m qSc)$
   (c) Yawing moment $(N = C_n qSb)$
3. Propulsive Forces: Include thrust forces generated by engines.
4. Gravitational Force: Accounts for weight acting downward (W=mg)

These forces and moments are added to the general RBD equations to form the airplane-specific EOM. Here:

- $C_L\ ,\ C_D\ ,\ C_Y, C_l, C_m, C_n$: Aerodynamic coefficients.
- $q = \frac{1}{2}\rho V^2$: Dynamic Pressure.
- S, b, c: Wing area, span, and chord length.

12 Assumptions introduced while deriving the airplane equations of motion

- Rigid Body Assumption: The airplane is treated as a rigid body with no deformation.
- Small Angle Approximations: For small perturbations from equilibrium flight conditions (linearized models).
- Steady Aerodynamics: Assumes aerodynamic forces depend only on instantaneous velocity and orientation (ignores unsteady effects).
- Flat Earth Assumption: Neglects curvature of the Earth for short-range flights.
- Constant Mass Distribution: Assumes no significant fuel consumption or payload changes during flight.
- Symmetry: Assumes lateral symmetry about the longitudinal axis.

13 Classification of an aircraft's equations of motion

Order — The equations of motion for an aircraft are typically $1^{st.}$ order differential equations because they involve the second derivative of position and orientation with respect to time (acceleration).

Type — The EOMs are ordinary differential equations (ODEs) rather than partial differential equations (PDEs) since they depend on time derivatives rather than spatial derivatives.

Linearity — The general form of aircraft EOMs is nonlinear due to aerodynamic forces and moments, which depend on velocity, angles, and control inputs in a nonlinear manner.
However, for small perturbations around an equilibrium point (such as level flight), they can be linearized to simplify analysis.

Coupling — The aircraft EOMs are generally coupled, meaning that motion in one axis (e.g., pitch) affects motion in another (e.g., roll or yaw).
However, under certain assumptions (such as decoupling longitudinal and lateral dynamics), the equations can be approximated as uncoupled for analysis.

14 Difference between the (Body axes) and the (earth or inertial axes)

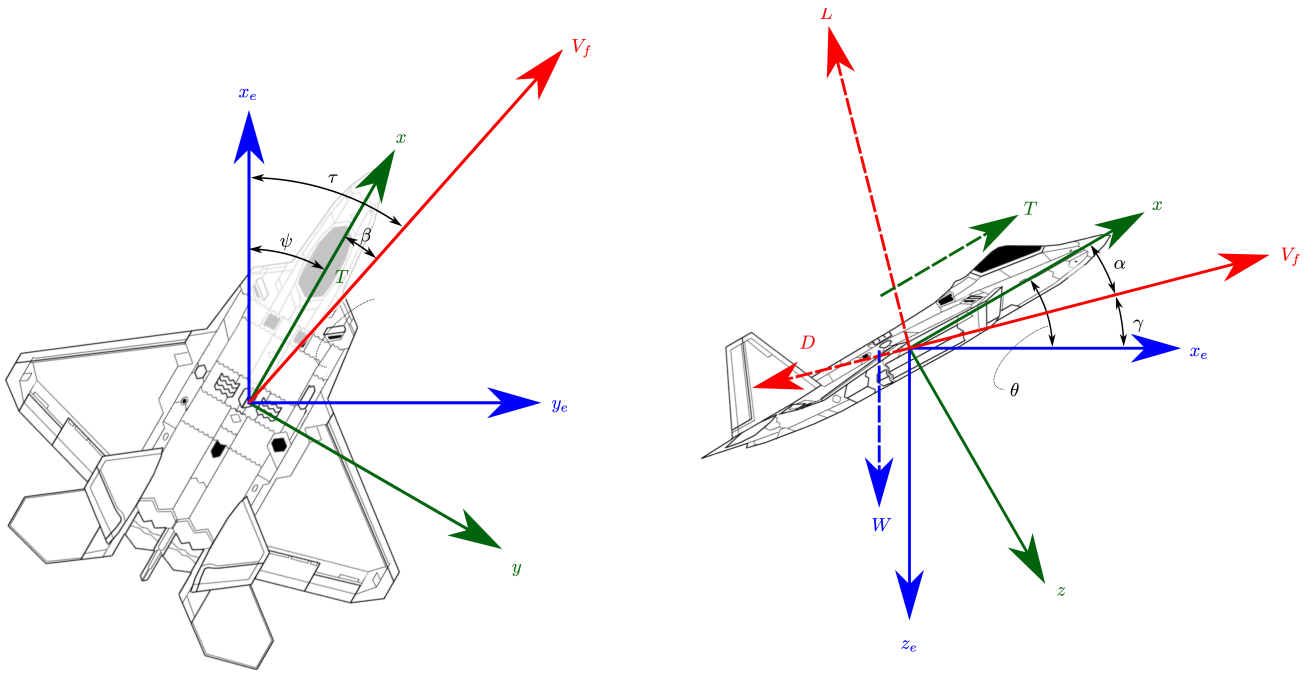


Figure 1: Body and Inertial Axes, Images ref.: [1]

1. Body Axes (Body-Fixed Frame)

   (a) Definition: A coordinate system that moves with the aircraft, with its origin fixed at the aircraft's center of gravity (CG).

   (b) Axes Orientation:

      i. X-axis (Forward Axis): Points along the aircraft's nose (longitudinal axis).

      ii. Y-axis (Lateral Axis): Points towards the right wing (perpendicular to the X-axis).

      iii. Z-axis (Vertical Axis): Points downward, perpendicular to the X-Y plane (opposite to the lift direction in level flight).

   (c) Characteristics:

      i. Rotates and translates with the aircraft.

      ii. Used to express aerodynamic forces, moments, and control inputs.

      iii. Orientation changes as the aircraft maneuvers.

      iv. Wind tunnel data and aerodynamic coefficients are often expressed in body axes.

2. Earth Axes (Inertial or Navigation Frame)

   (a) Definition: A fixed coordinate system that does not move with the aircraft; it is typically referenced to the Earth or an inertial frame.

   (b) Axes Orientation:

      i. X-axis: Points North (or along a chosen reference direction).

      ii. Y-axis: Points East.

      iii. Z-axis: Points downward toward the center of the Earth (opposite to altitude increase).

   (c) Characteristics:

      i. Used to describe absolute motion relative to the Earth.

      ii. Does not change with aircraft movement.

      iii. Used in navigation, trajectory analysis, and flight planning.

      iv. Important in inertial navigation systems (INS) and GPS-based positioning.

15 Difference between the pitch angle (θ) and the angle of attack (α), and between the sideslip angle (β) and the heading angle (ψ):

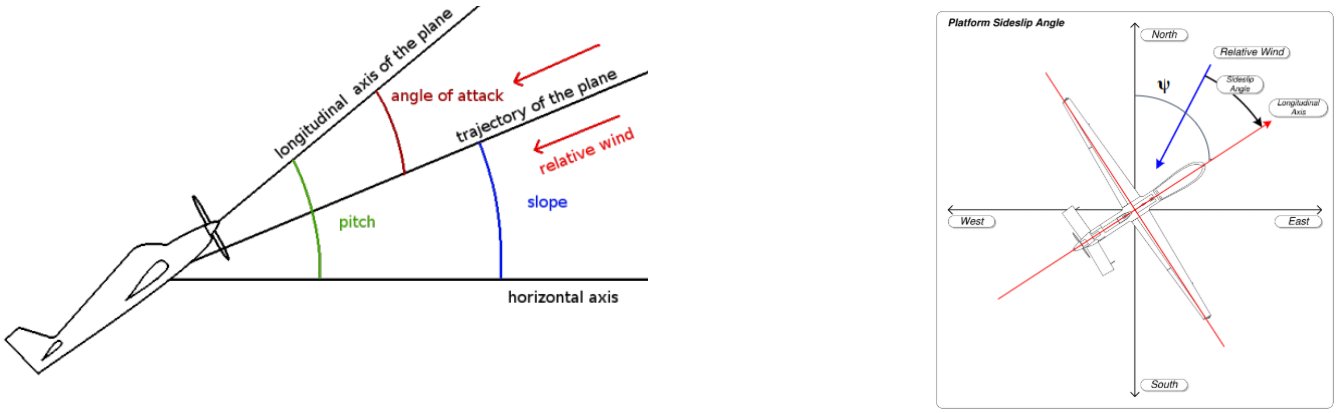


Figure 2: Image source: AeroSkyTechAeroSkyTech

Pitch angle (θ) The angle between the aircraft's longitudinal axis and the Earth **(inertial)** horizontal plane, relation: $\theta = \alpha + \gamma$.

AoA (α)Orientation The angle between the aircraft's longitudinal axis and the relative wind (freestream airflow direction).

Sideslip angle (β) The angle between the aircraft's longitudinal axis and its relative wind (freestream airflow direction) in the lateral plane, relation: $\beta = \tan^{-1}\left(\frac{v}{u}\right)$.

Heading angle (ψ) The angle between the aircraft's nose direction (longitudinal axis) and true north.

	Euler angles	Direction Cosines	Quaternions	Axis-Angle
Advantages	- Only 3 variables. - Directly relates to how pilots and engineers describe aircraft orientation.	- No singularities: Works for all orientations. - Efficient when applying successive rotations.	- No singularities: Works for all orientations. - Faster than DCM in 3D rotations. - Only 4 values instead of 9 (as in DCM).	- Requires only 4 values. - Often used in computer graphics, robotics and even space shuttles.
Disadvantages	- Singularity: A loss of one degree of freedom occurs at certain angles (e.g., pitch ±90°). - Converting between different frames requires multiple matrix operations.	- Contains 9 elements but only 3 are independent. - More storage and calculations compared to other methods.	- Harder to visualize compared to Euler angles. - Quaternions must be kept unit-length to avoid drift in calculations.	- Requires conversion to matrices or quaternions for application in transformations. - Composing multiple rotations is not as straightforward.

Part III

Numerical solution of ODEs

16 Some ODEs numerical solving algorithms

16.1 Euler Method

A 1^{st} order scheme is used to integrate the differential equations, using only one approximation from Taylor's series.

$$y_{n+1} = y_n + h \cdot f(t_n,\, y_n)$$

16.2 Finite Difference Methods

Approximates the derivatives in a set of differential equations using a specific scheme, to transform the set of ODEs into a set of algebraic equations, that can be either in an explicit or implicit form.

16.3 Finite Volume Methods

While similar to FDM but it approximates the equations over control volumes rather than nodes, this ensures conservation of mass, momentum and energy, this is the most common method in PDEs solution such as CFD because it's too complicated to be used to solve ordinary differential equations.

16.4 Finite Element Methods

You can either formulate the finite element method from Weighted Residuals Method, or a Variational Method, this method is mostly used in structural analyses where the system is governed also by PDEs, because the method is also too complicated to be used to solve ordinary DEs.

16.5 Runge-Kutta Methods

Uses simple schemes to obtain a very high accuracy for most dynamical systems, even some (but not all) chaotic systems are usually modeled very well using Runge-Kutta schemes, RK method can commonly use a 2^{nd} or 4^{th} order to solve the differential equations, the Runge-Kutta method is usually used in professional mathematical solvers such as Matlab and Simulink.

17 Algorithm for solving airplane EOM

A **fixed time step** 4^{th} **order Runge-Kutta scheme** is very sufficient to solve the airplane equations of motion.

17.1 Initial Conditions

Typically we need 12 initial conditions for solving the 12 equations of motion for the aircraft, which are: $[x,\, y,\, z,\, u,\, v,\, w,\, p,\, q,\, r,\, \phi,\, \theta,\, \psi]^T$, noticing that $(x,\, y,\, z)$ do not affect the dynamics of the airplane at all, and also noticing that $(\phi,\, \theta,\, \psi)$ are the Euler angles, and their initial values actually **do not matter**, what matters is their change during dynamics, so their initial conditions may be taken as $(0,\, 0,\, 0)$ or as maybe as some other reference directions, such as the true north for ψ.

17.2 Inputs needed at each iteration $(n+1)$

The inputs needed at each iteration are:

1. The twelve states at the previous iteration $[x_n,\, y_n,\, z_n,\, u_n,\, v_n,\, w_n,\, p_n,\, q_n,\, r_n,\, \phi_n,\, \theta_n,\, \psi_n]^T$.
2. Forces and Moments on the airplane.
3. Moments of Inertia are needed but it is constant and does not change during solution.

17.3 Outputs Calculated at each iteration $(n+1)$

First, the derivatives of the twelve states are calculated: $\left[\dot{x},\, \dot{y},\, \dot{z},\, \dot{u},\, \dot{v},\, \dot{w},\, \dot{p},\, \dot{q},\, \dot{r},\, \dot{\phi},\, \dot{\theta},\, \dot{\psi}\right]^T$ from a mix of the (n), $(n+1/2)$ and $(n+1)$ time steps. Which are then going to be used to relate the states from time stamp n to time stamp $n+1$:

$$[x_{n+1},\, y_{n+1},\, z_{n+1},\, u_{n+1},\, v_{n+1},\, w_{n+1},\, p_{n+1},\, q_{n+1},\, r_{n+1},\, \phi_{n+1},\, \theta_{n+1},\, \psi_{n+1}]^T\,.$$

17.4 Solution Algorithm

If N is the total no. of time steps:

```
for n in 1:N-1

    h = t(n+1) - t(n);
    K1 = h*f_dot(t(n), y(n));
    K2 = h*f_dot(t(n)+h/2, y(n)+0.5*K1);
```

```
K3 = h*f_dot(t(n)+h/2, y(n)+0.5*K2);
K4 = h*f_dot(t(n)+h, y(n)+K3);
y(n+1) = y(n) + 1/6*(K1 + 2*K2 + 2*K3 + K4);
```

18 Testing of the numerical solver on simple ODEs

The algorithm RK4 was implemented and tested on a set of simple differential equations:

$$\begin{aligned} \frac{dy_1}{dt} &= \sin(t) + \cos(y_1) + \cos(y_2) \\ \frac{dy_2}{dt} &= \sin(t) + \sin(y_2) \end{aligned}$$

with initial conditions:

$$\begin{Bmatrix} y_1(t=0) \\ y_2(t=0) \end{Bmatrix} = \begin{Bmatrix} 2 \\ 1 \end{Bmatrix}$$

Time span: $[0,\ 20]$, with 100 intervals in between for discretizing the equation.

The two ODEs were solved using our RK4 algorithm, and it was validated using Matlab's `ode45` function, the results were very satisfactory:

```
Mean Squared Error in 1st state: 1.660598e-03, in 2nd state: 4.820051e-07
```

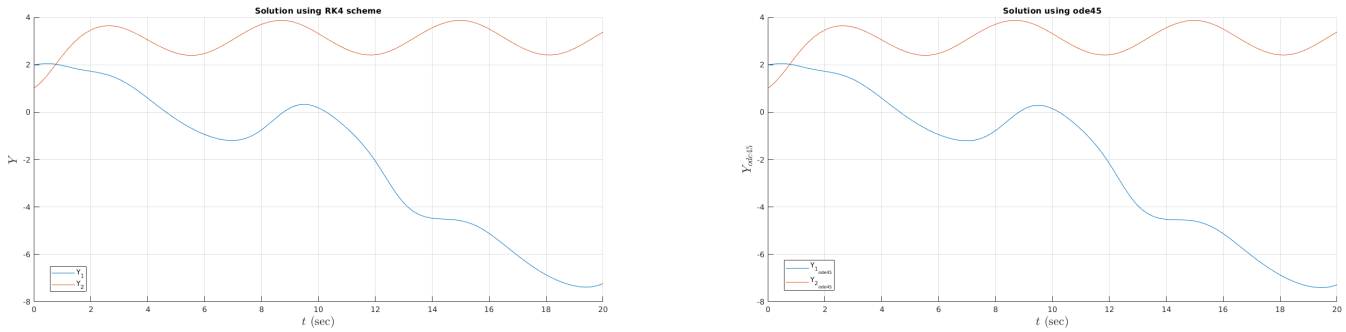


Figure 3: Comparison between RK4 and ode45

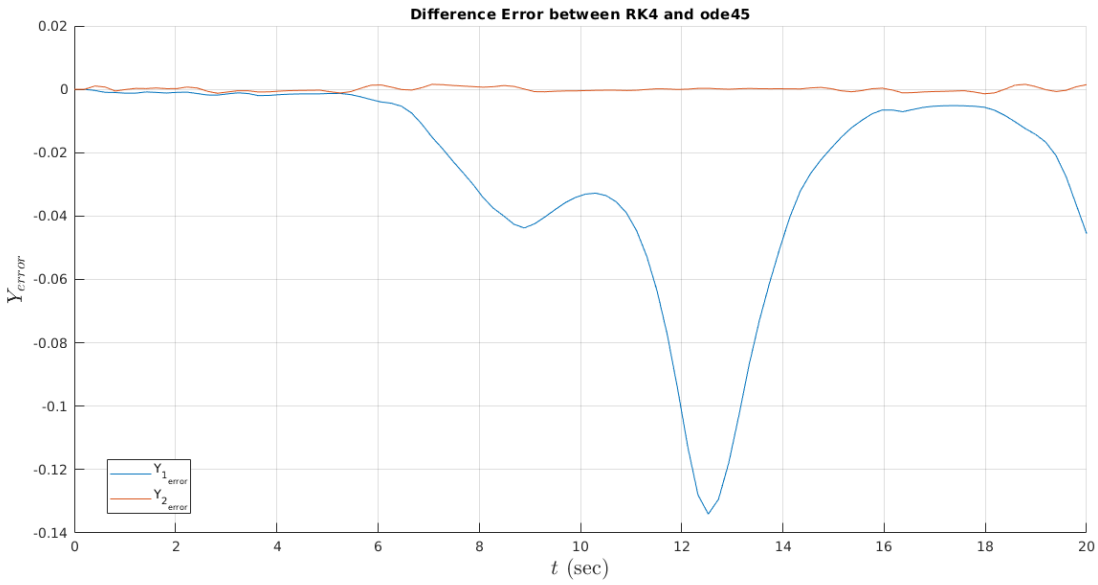


Figure 4: Comparison between RK4 and ode45

Part IV

Nonlinear 6DOFs Airplane Simulator

19 Solving EOM using MATLAB & SIMULINK

The code was based on the our 4th. order fixed step Runge-Kutta method, it is now used to solve the 12 nonlinear, coupled ordinary differential equations of motion of aircraft, which are written in their vectorial (this is to use Matlab's power in manipulating arrays) state space model as:

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \frac{1}{m} \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = [I]^{-1} \left(\begin{bmatrix} L \\ M \\ N \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times [I] \begin{bmatrix} p \\ q \\ r \end{bmatrix} \right)$$

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & S\phi T\theta & C\phi T\theta \\ 0 & C\phi & -S\phi \\ 0 & \frac{S\phi}{C\theta} & \frac{C\phi}{C\theta} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

$$\begin{bmatrix} \dot{x}_e \\ \dot{y}_e \\ \dot{z}_e \end{bmatrix} = [R(\psi) R(\theta) R(\phi)] \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

As apparent from the equations, we used a "ZYX" sequence for Euler angles, which is a very typical convention in aerospace applications.

A, similar technique was used in SIMULINK software, the settings of the simulation on Simulink needed to be changed a little, in order to produce results at the same time intervals as same as the matlab code, so we set the simulation to be a 4^{th} order Runge-Kutta with fixed step size dt (defined in matlab workspace, as well as final time t_f), the resulting time array indeed was same as our time array defined in matlab, but only differed in a few timestamps due to truncation error.

We used the block **6DOF Euler Angles** to solve the exact same equations as written in the previous section, we only need to read the initial conditions, and simulation data from matlab workspace.

20 Comparing Results

There are so many formulae to compare the error/difference between two signals, or more generally, two arrays, among them are:

1. Mean Squared Error:

   This is the most common and easiest way to compare two arrays, it can be calculated using the

formula: **MSE** $= \frac{1}{N}\sum (y_1 - y_2)^2$, a modification to it is the Root MSE, given by equation: **RMSE** $= \sqrt{\frac{1}{N}\sum (y_1 - y_2)^2}$.

2. Mean Absolute Error:
   MAE $= \frac{1}{N}\sum |y_1 - y_2|$.

3. Huber Loss:
   It is used a lot in machine learning and neural networks to calculate cost functions, $L_\delta(y_1, y_2) = \begin{cases} \frac{1}{2}(y_1 - y_2)^2 & \text{if } |y_1 - y_2| \leq \delta \\ \delta|y_1 - y_2| - \frac{1}{2}\delta^2 & \text{otherwise} \end{cases}$.

4. Minkowski Distance:
   $D_p(y_1,y_2) = (\sum |y_1 - y_2|^p)^{\frac{1}{p}}$, when $p = 1$ it becomes the Manhattan distance, when $p = 2$ it becomes the Euclidean distance, other values of p make it more general and abstract.

5. Chebyshev Distance:
   The Chebyshev distance is the maximum absolute difference between the corresponding elements of the two vectors. It is defined as: $D_p(y_1,y_2) = \max|y_1 - y_2|$. This metric is useful when the largest difference between any pair of corresponding elements is the most important factor. [2]

6. Correlation Coefficient:
   $r = \frac{\sum(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum(X_i - \bar{X})^2} \cdot \sqrt{\sum(Y_i - \bar{Y})^2}}$, where $\bar{x}$ and $\bar{y}$ are the means of the vectors. This measures the linear relationship between the two vectors. [3]

In this example we certainly used MSE to compare the results from matlab and Simulink, and it has shown a very great accuracy since the value $MSE = 6.933581\mathbf{e} - 21$.

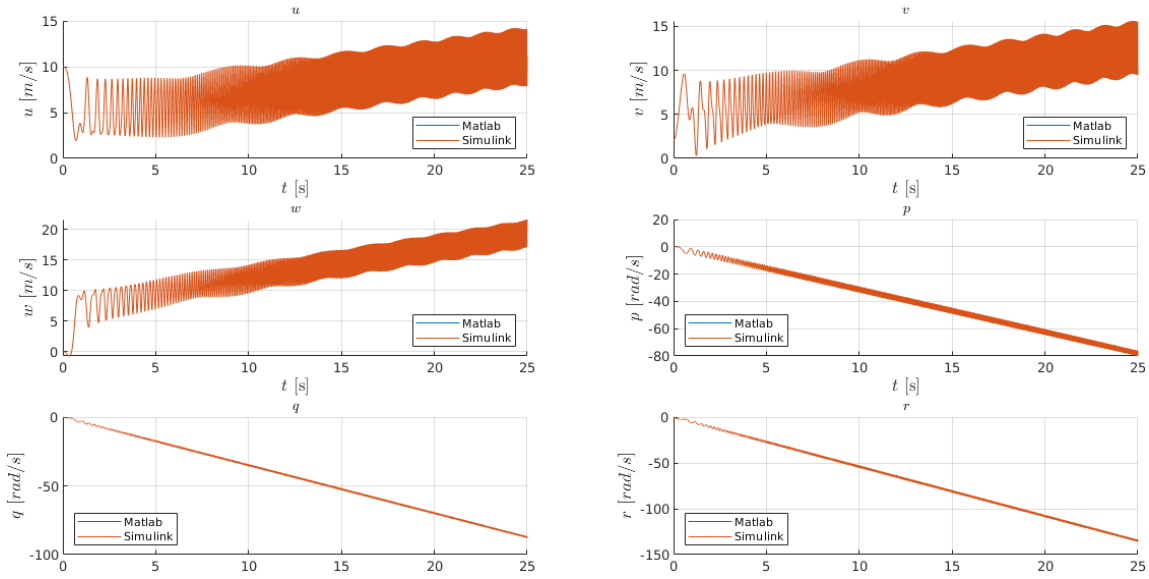


Figure 5: Solution of the states: $[u, v, w, p, q, r]$ by both matlab and Simulink

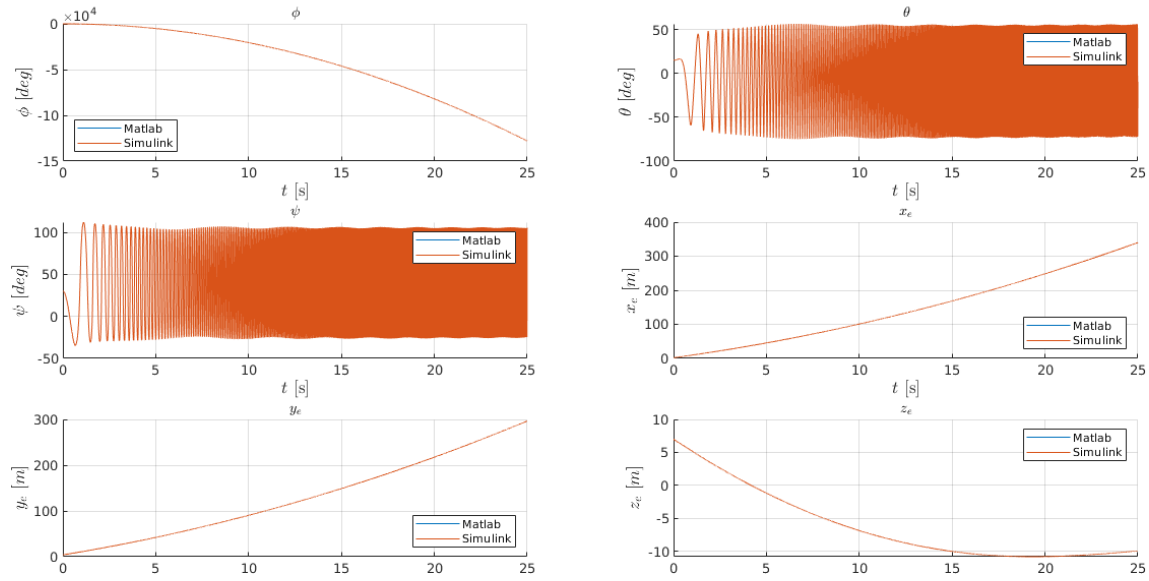


Figure 6: Solution of the states: $[\phi, \theta, \psi, x_e, y_e, z_e]$ by both matlab and Simulink

21 Airframe Model

The airframe model was implemented in both matlab and Simulink, in order to model the forces and moments acting on the airplane, we used a first order Taylor series expansion, in which we can write the forces as a linear combination of the corresponding states as well as control actions, the coefficients in this linear combination are the stability and control derivatives.

Those derivatives can be obtained in multiple ways, among them are: aerodynamics analysis software, CFD or wind tunnels, in our case we are working on an airplane that exists in the NACA 2144 Report, so the stability derivatives are quite ready to be used, only after keeping an eye for units and the dimensionalization process...

$$
\begin{aligned}
\Delta X &= X_u u + X_w w + X_{\delta_e}\delta_e + X_{\delta_{th}}\delta_{th} \\
\Delta Y &= Y_v v + Y_p p + Y_r r + Y_{\delta_a}\delta_a + Y_{\delta_r}\delta_r \\
\Delta Z &= Z_u u + Z_w w + Z_{\dot{w}}\dot{w} + Z_q q + Z_{\delta_e}\delta_e + Z_{\delta_{th}}\delta_{th} \\
\Delta L &= L_v v + L_p p + L_r r + L_{\delta_a}\delta_a + L_{\delta_r}\delta_r \\
\Delta M &= M_u u + M_w w + M_{\dot{w}}\dot{w} + M_q q + M_{\delta_e}\delta_e + M_{\delta_{th}}\delta_{th} \\
\Delta N &= N_v v + N_p p + N_r r + N_{\delta_a}\delta_a + N_{\delta_r}\delta_r
\end{aligned}
$$

22 Validating our nonlinear 6DOF Simulator on Boeing 747 Flight Condition 5

After coding the nonlinear simulator in both MATLAB and SIMULINK, at first a problem appeared in the response of many of the states at which fast oscillations appear (the problem affected some states of both the longitudinal and lateral related states), such as the response in the next figure, we found out that this is due to the fact that we forgot to multiply the value of $\dot{w}$ by dt inside the Runge-Kutta integrator, this lead to an increased damping-like behaviour.

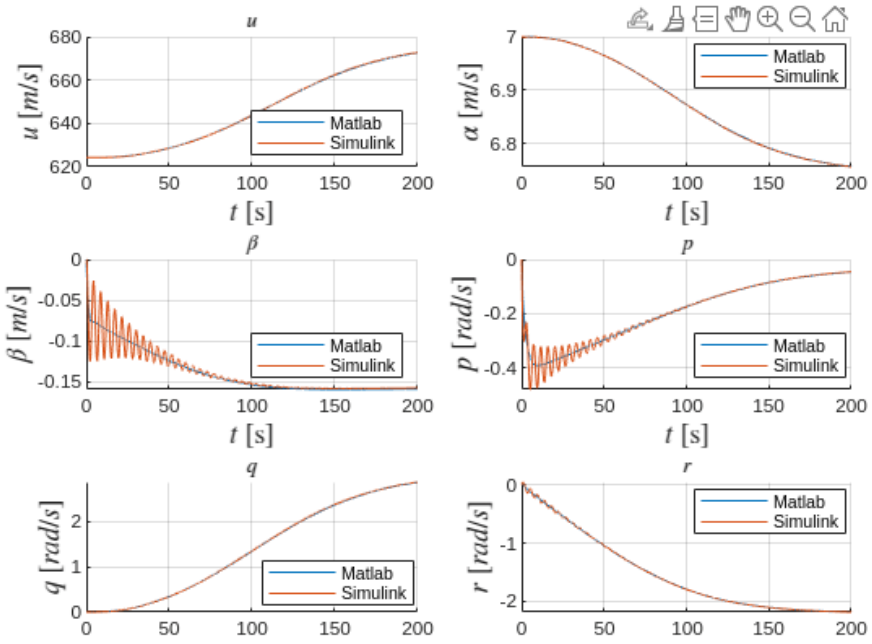


Figure 7: Wrong response due to wrong calculation of $\dot{w}$ in the MATLAB code, the SIMULINK response here is correct

Those are the results for Boeing flight condition 5, in order to compare with the results in the benchmark test, and also in order to compare the matlab code the Simulink one, in order to make sure they produce the same results.

22.1 No Input

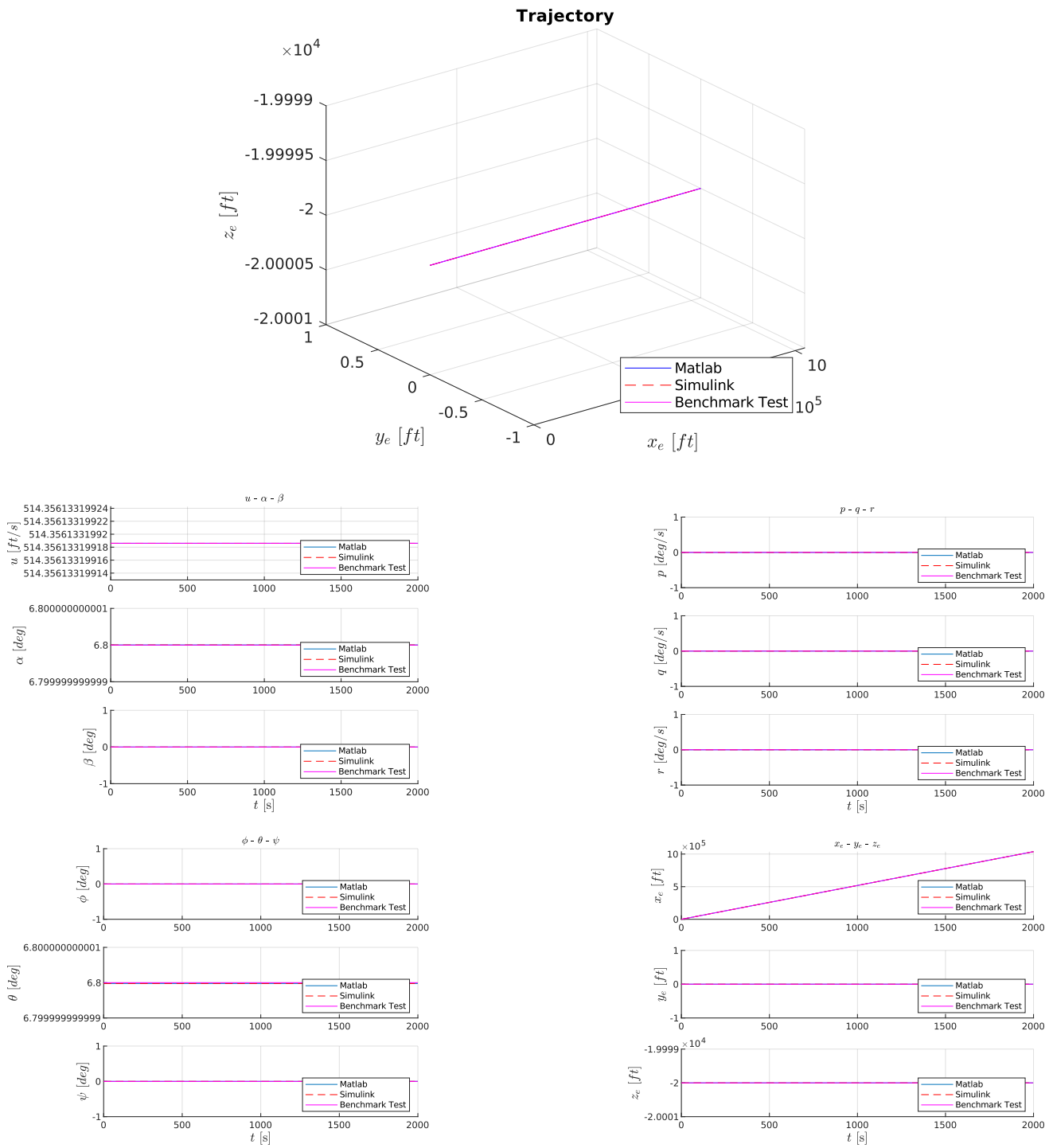


Figure 8: Response of the airplane Boeing 747 FC 5 to no control inputs

22.2 $dA = +5^\circ$

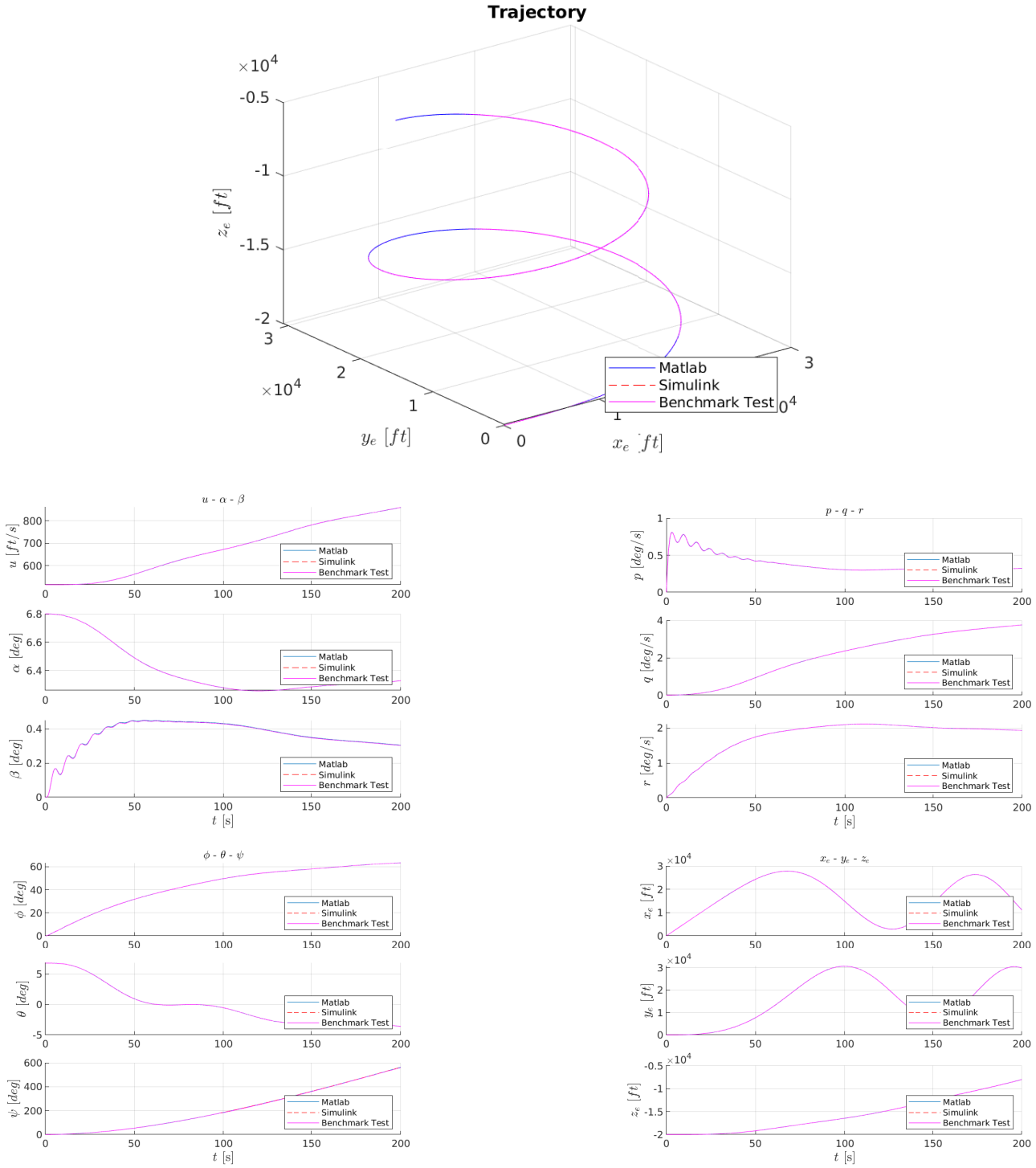


Figure 9: Response of the airplane Boeing 747 FC 5 to $+5^\circ$ aileron

22.3 $dA = -5°$

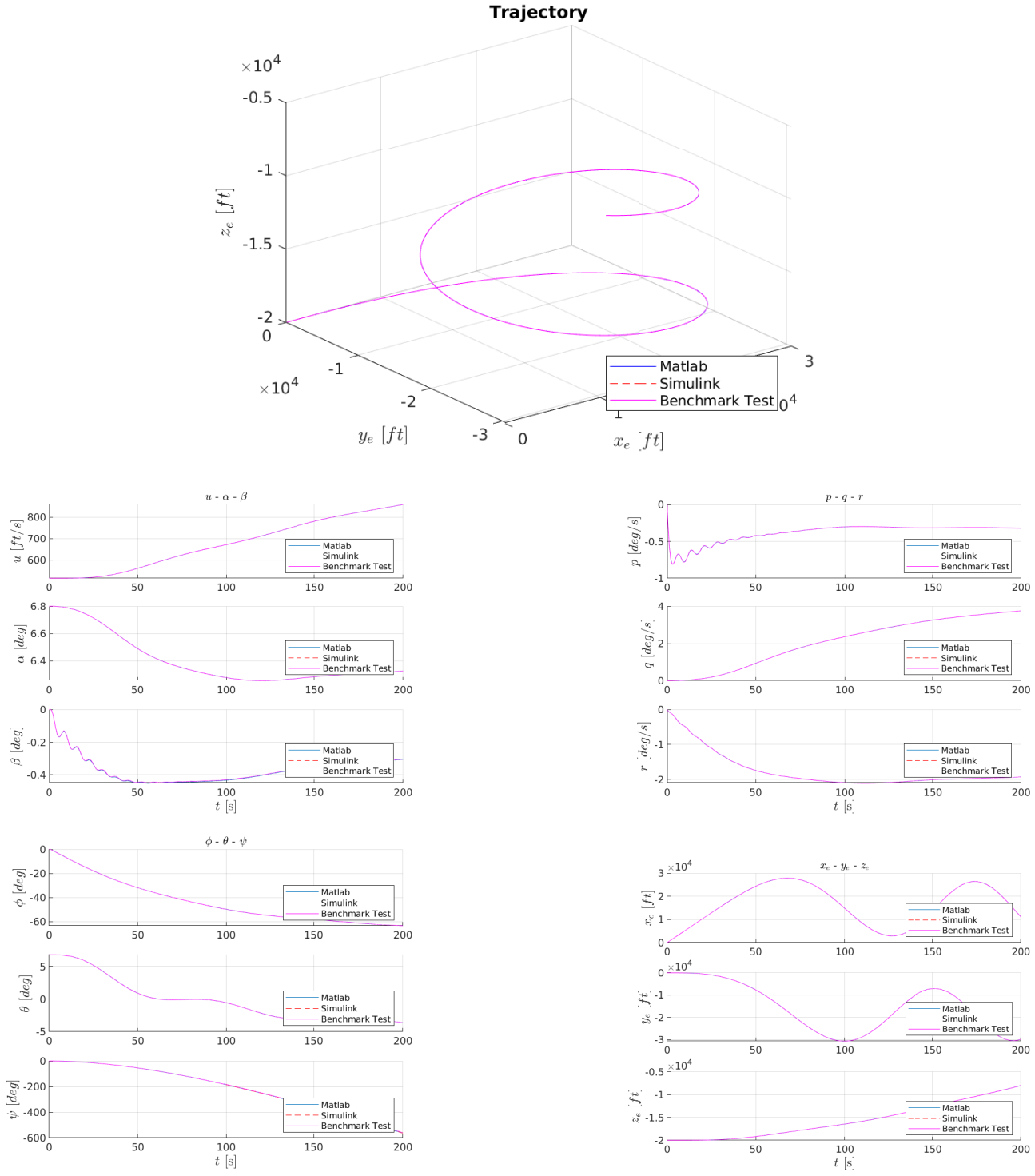


Figure 10: Response of the airplane Boeing 747 FC 5 to $-5°$ aileron

22.4 $dE = +5°$

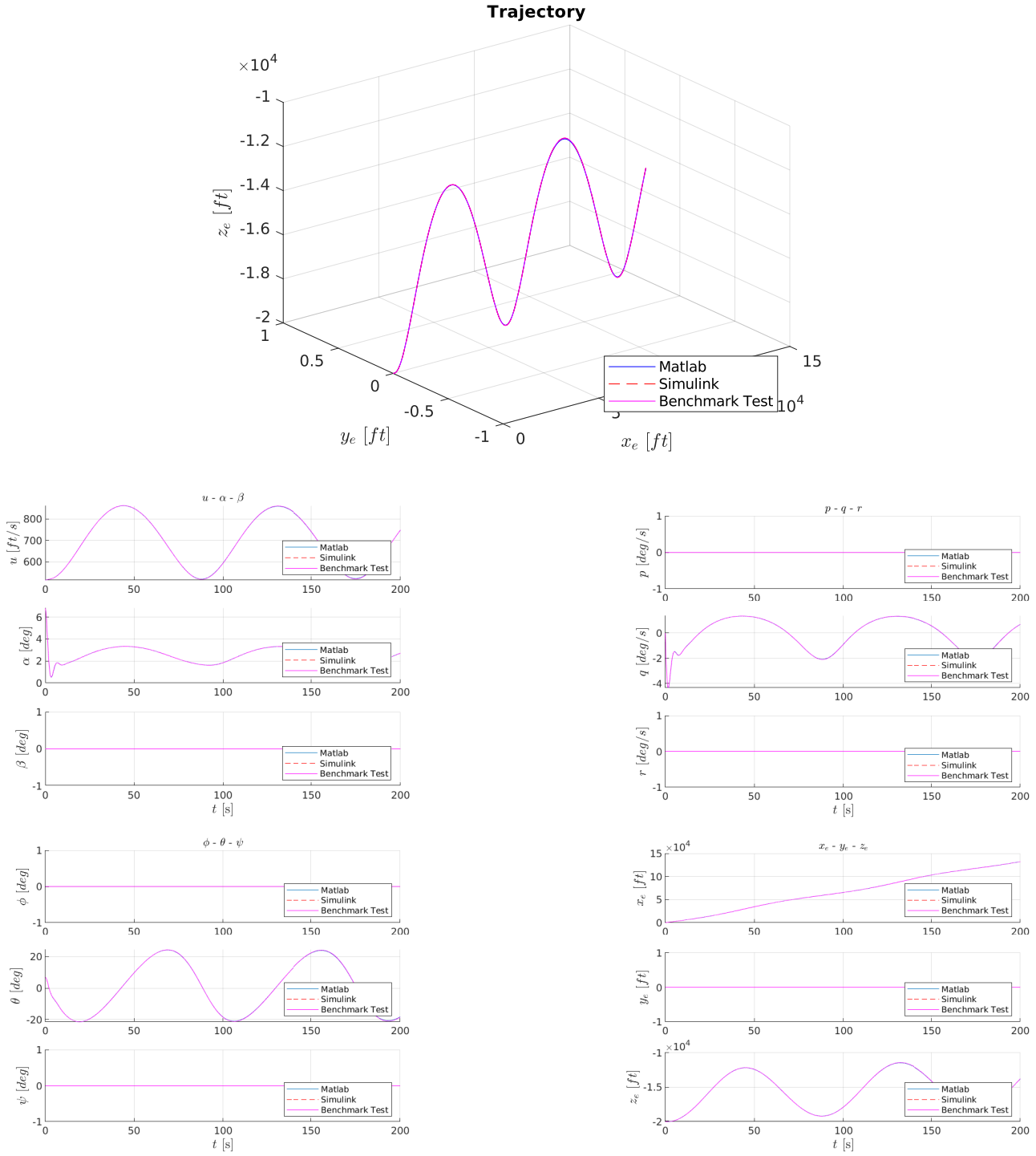


Figure 11: Response of the airplane Boeing 747 FC 5 to $+5°$ elevator

22.5 $dE = -5°$

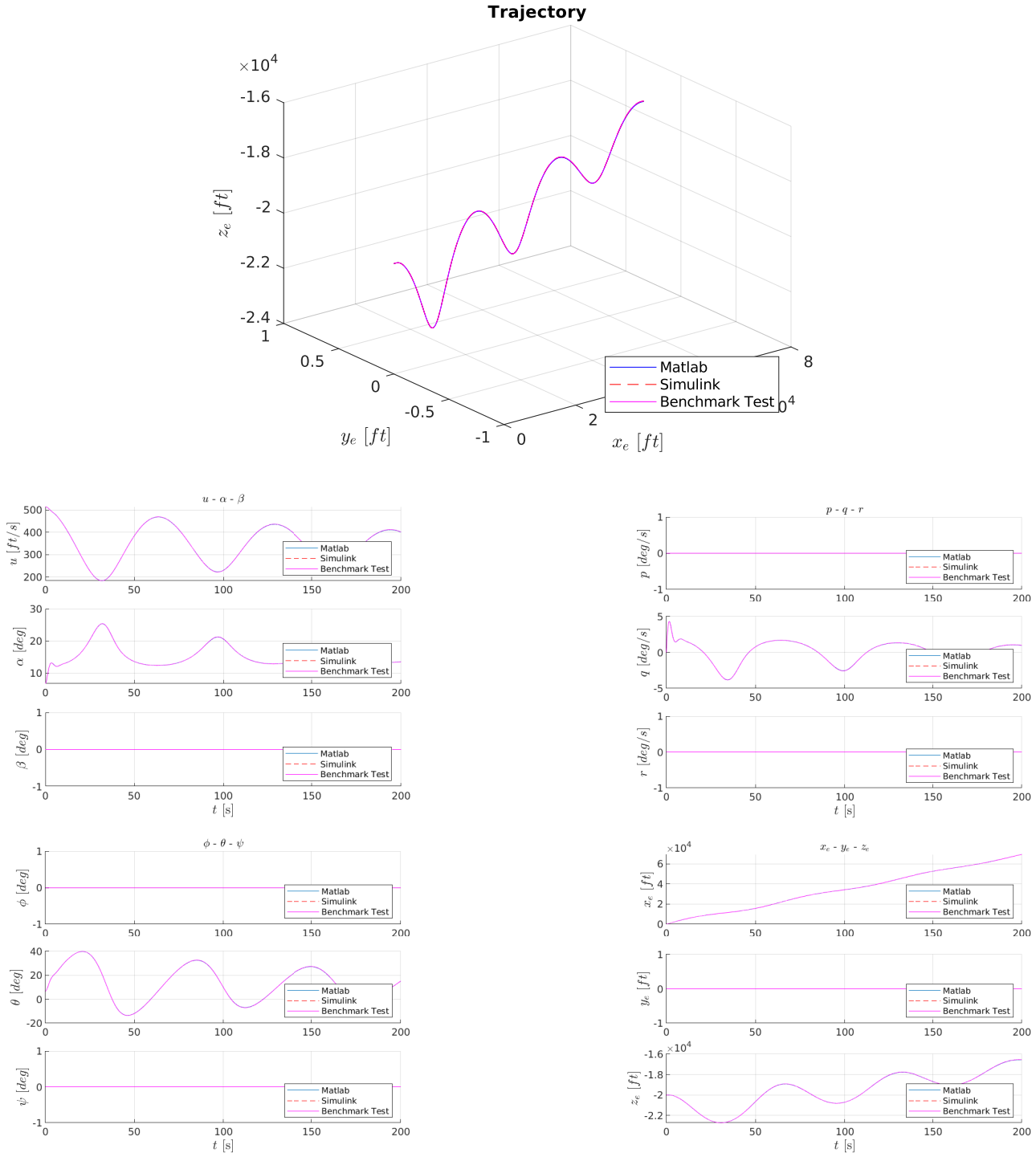


Figure 12: Response of the airplane Boeing 747 FC 5 to $-5°$ elevator

22.6 $dTH = 1000$

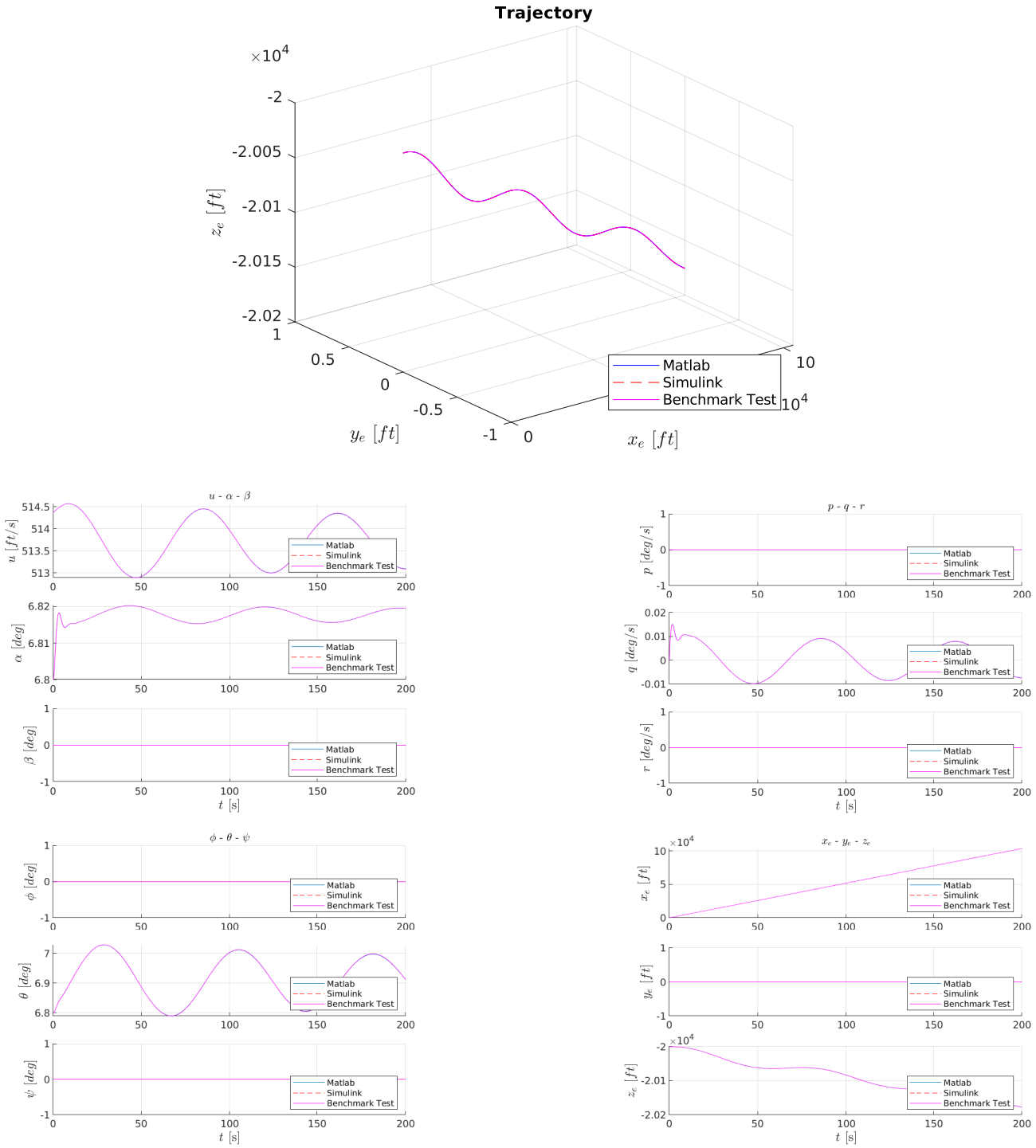


Figure 13: Response of the airplane Boeing 747 FC 5 to 1000 throttle

22.7 $dTH = 10000$

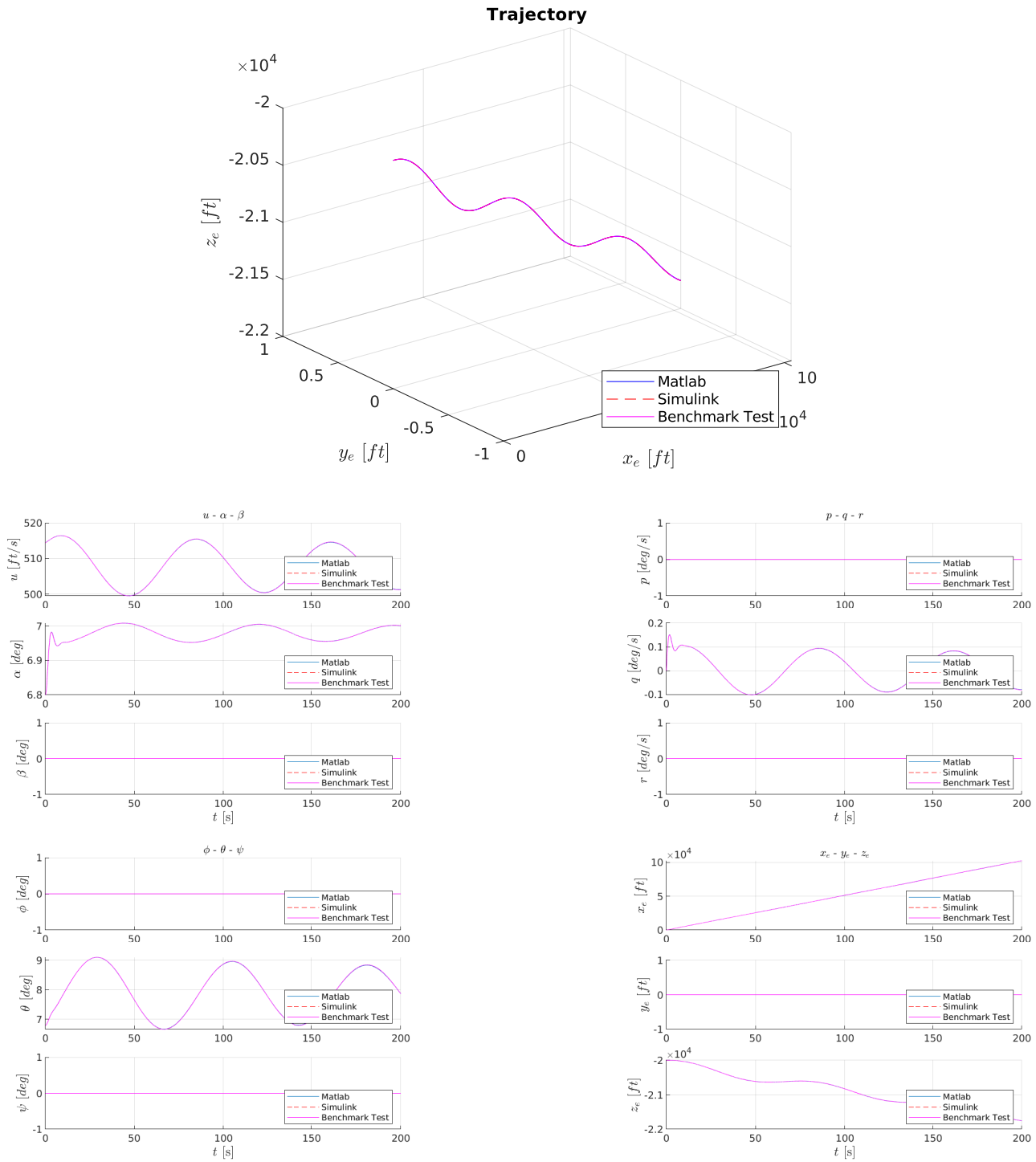


Figure 14: Response of the airplane Boeing 747 FC 5 to 10000 throttle

22.8 $dR = +5^\circ$

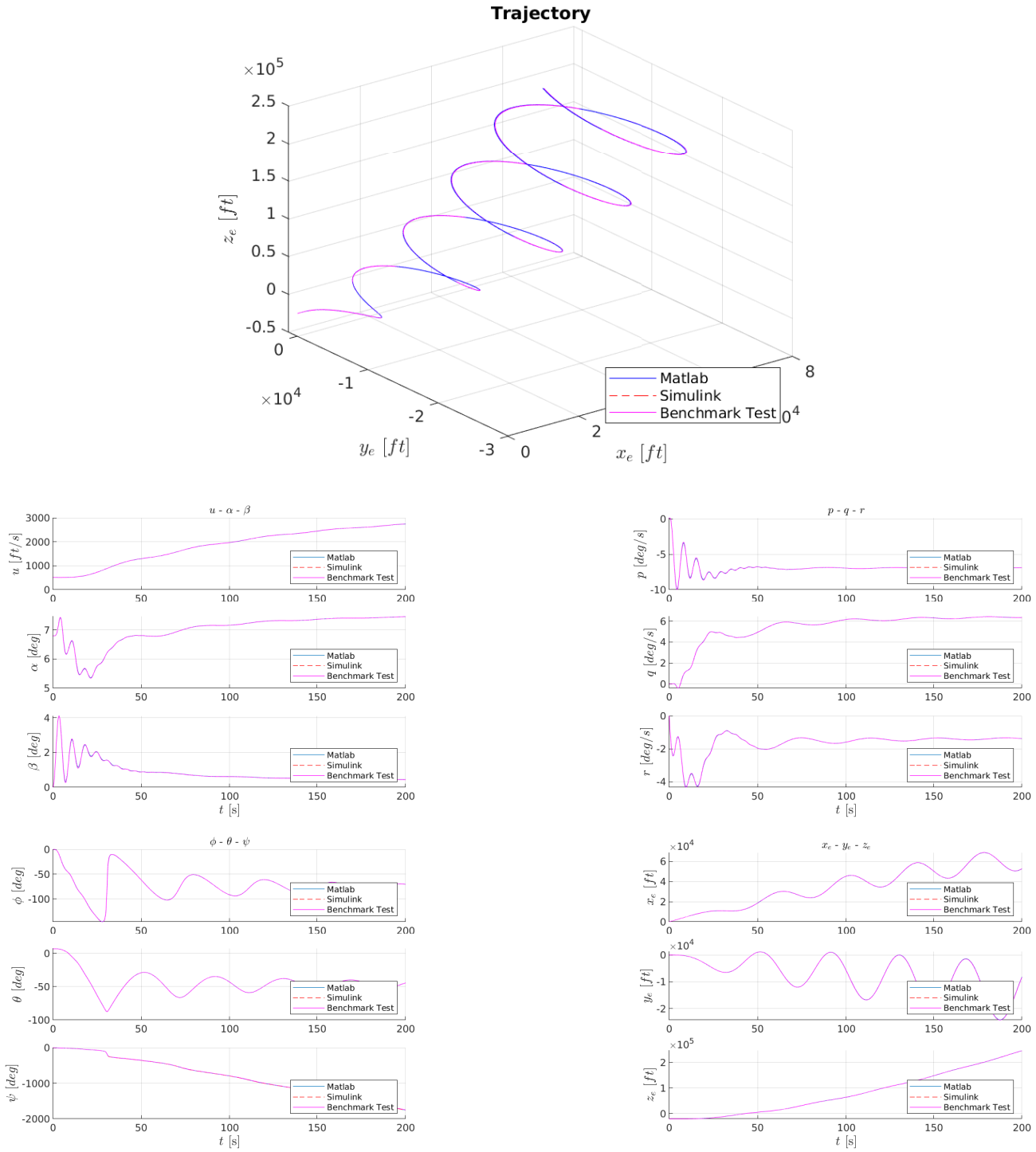


Figure 15: Response of the airplane Boeing 747 FC 5 to $+5^\circ$ rudder

22.9 $dR = -5^\circ$

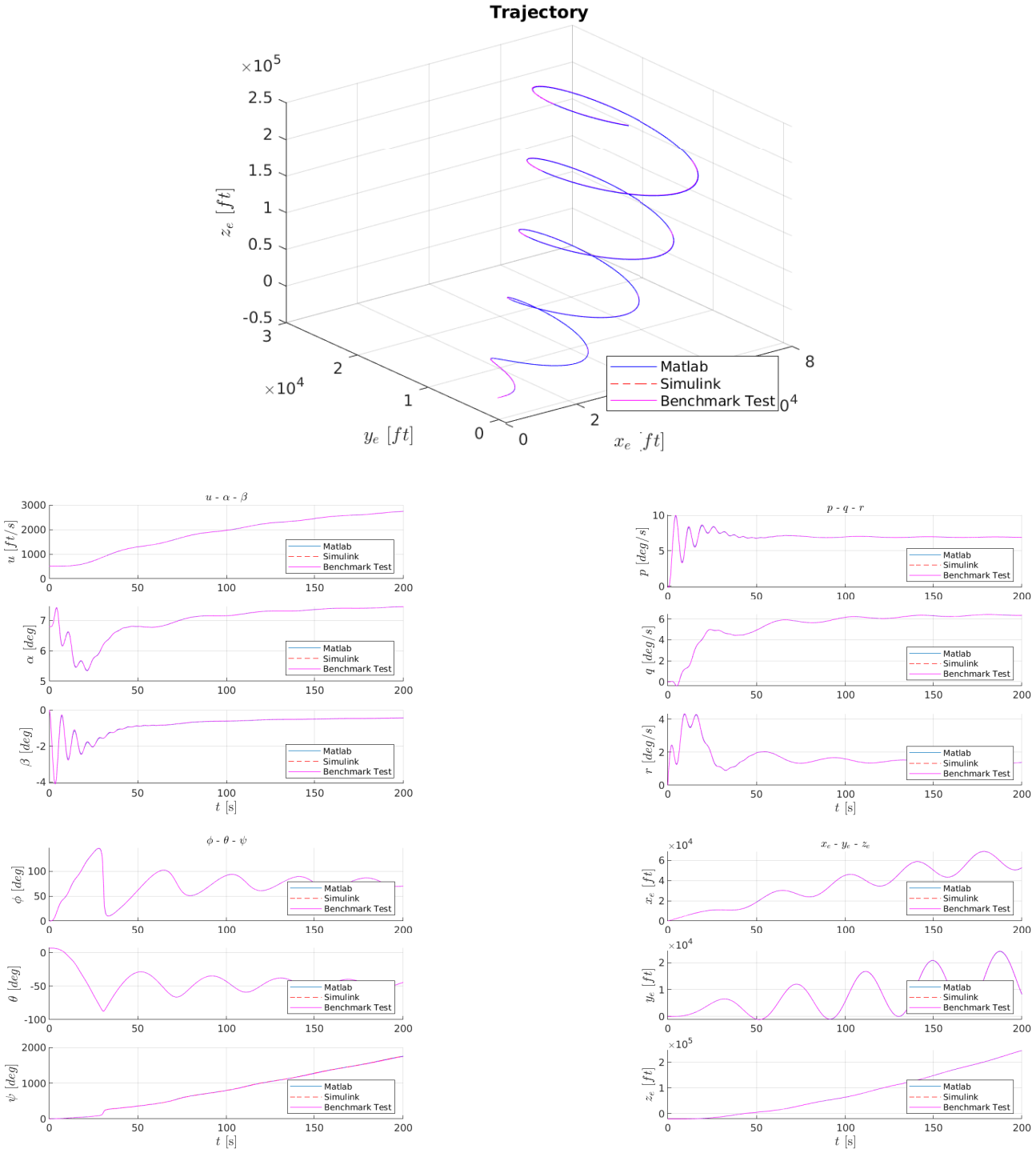


Figure 16: Response of the airplane Boeing 747 FC 5 to -5° rudder

23 Dynamics of Lockheed Jetstar FC 9

23.1 No Input

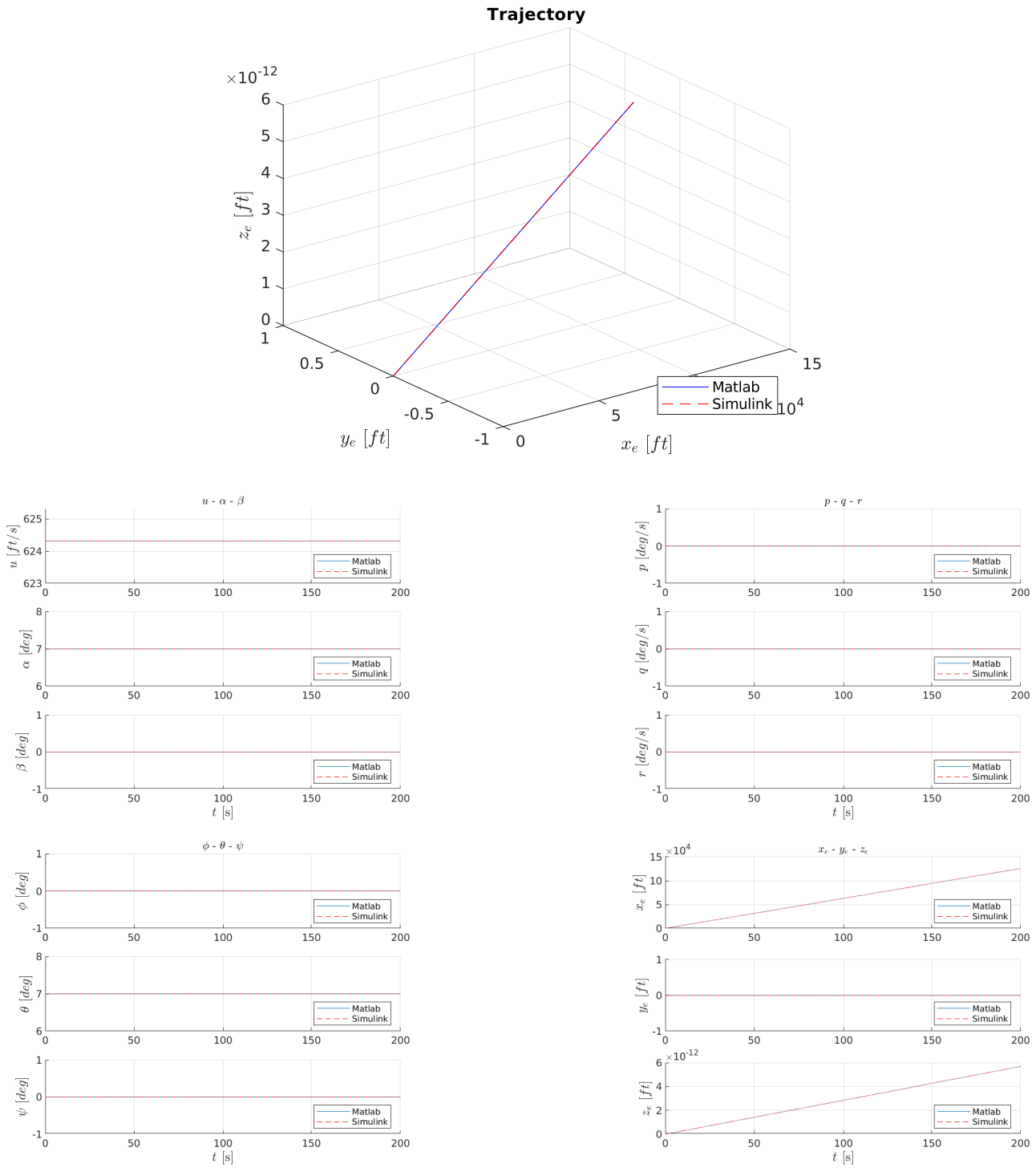


Figure 17: Response of the airplane Boeing 747 FC 5 to no control inputs

23.2 $dA = +5°$

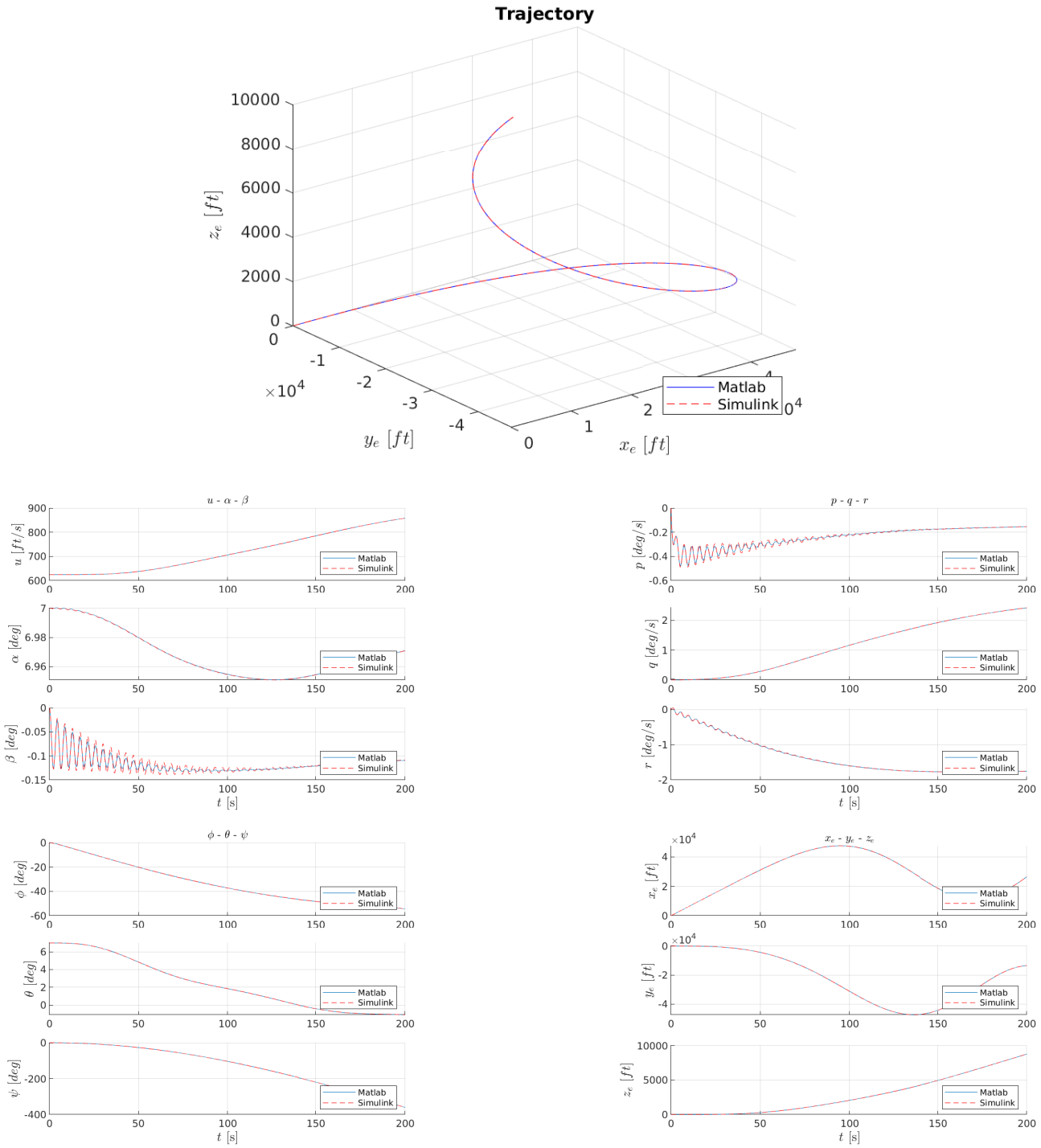


Figure 18: Response of the airplane Boeing 747 FC 5 to $+5°$ aileron

23.3 $dA = -5°$

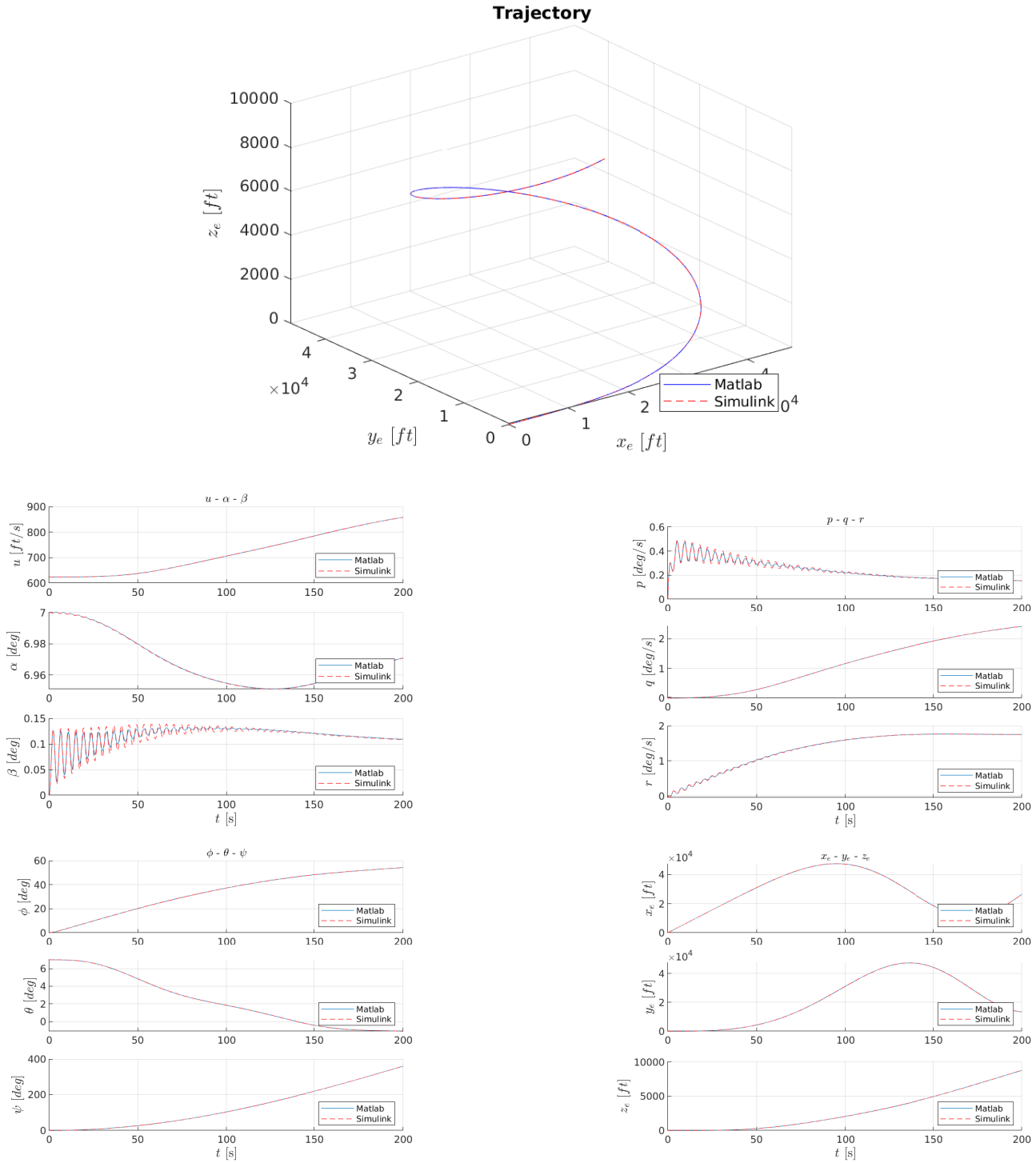


Figure 19: Response of the airplane Boeing 747 FC 5 to $-5°$ aileron

23.4 $dE = +5°$

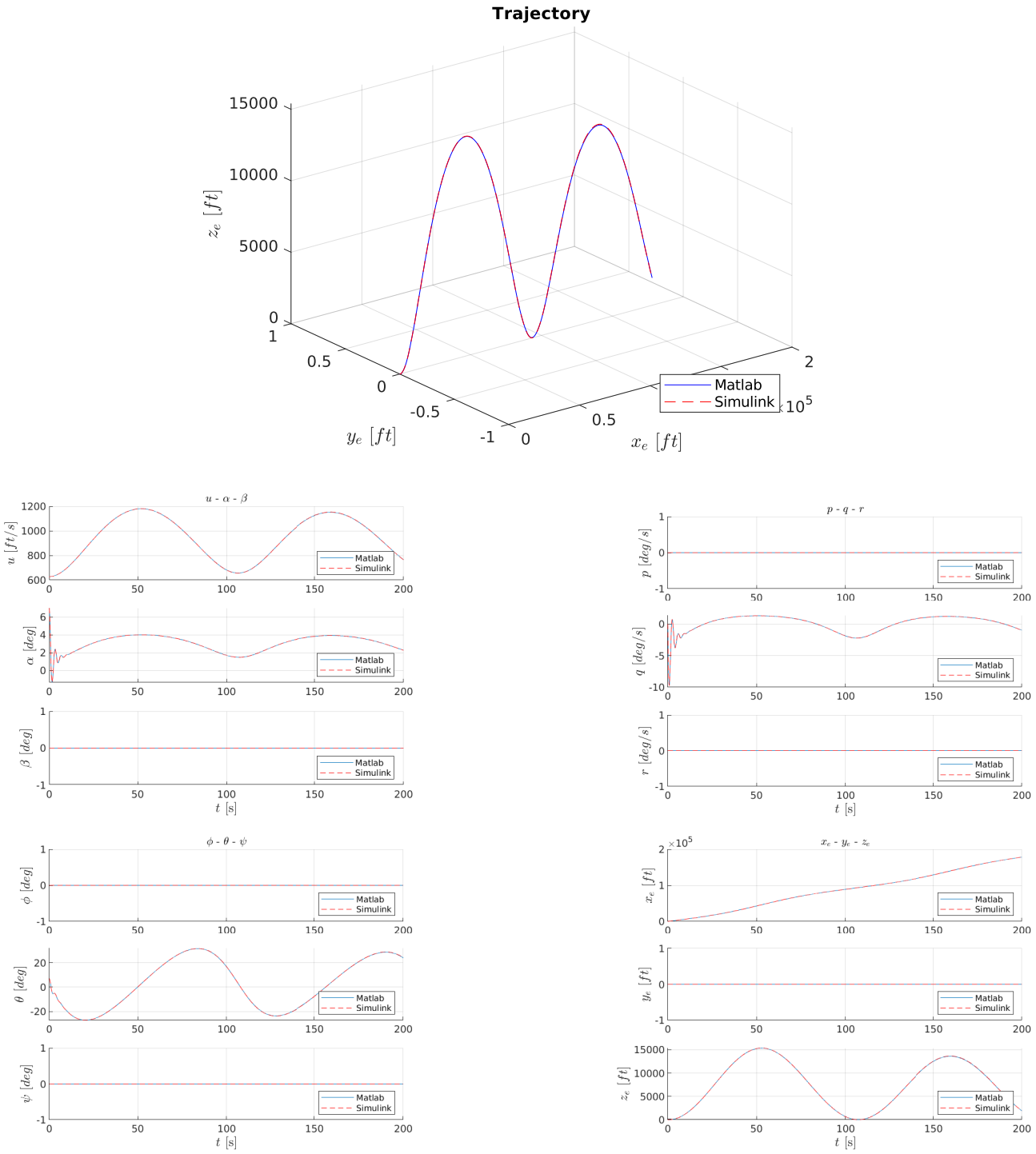


Figure 20: Response of the airplane Boeing 747 FC 5 to $+5°$ elevator

23.5 $dE = -5°$

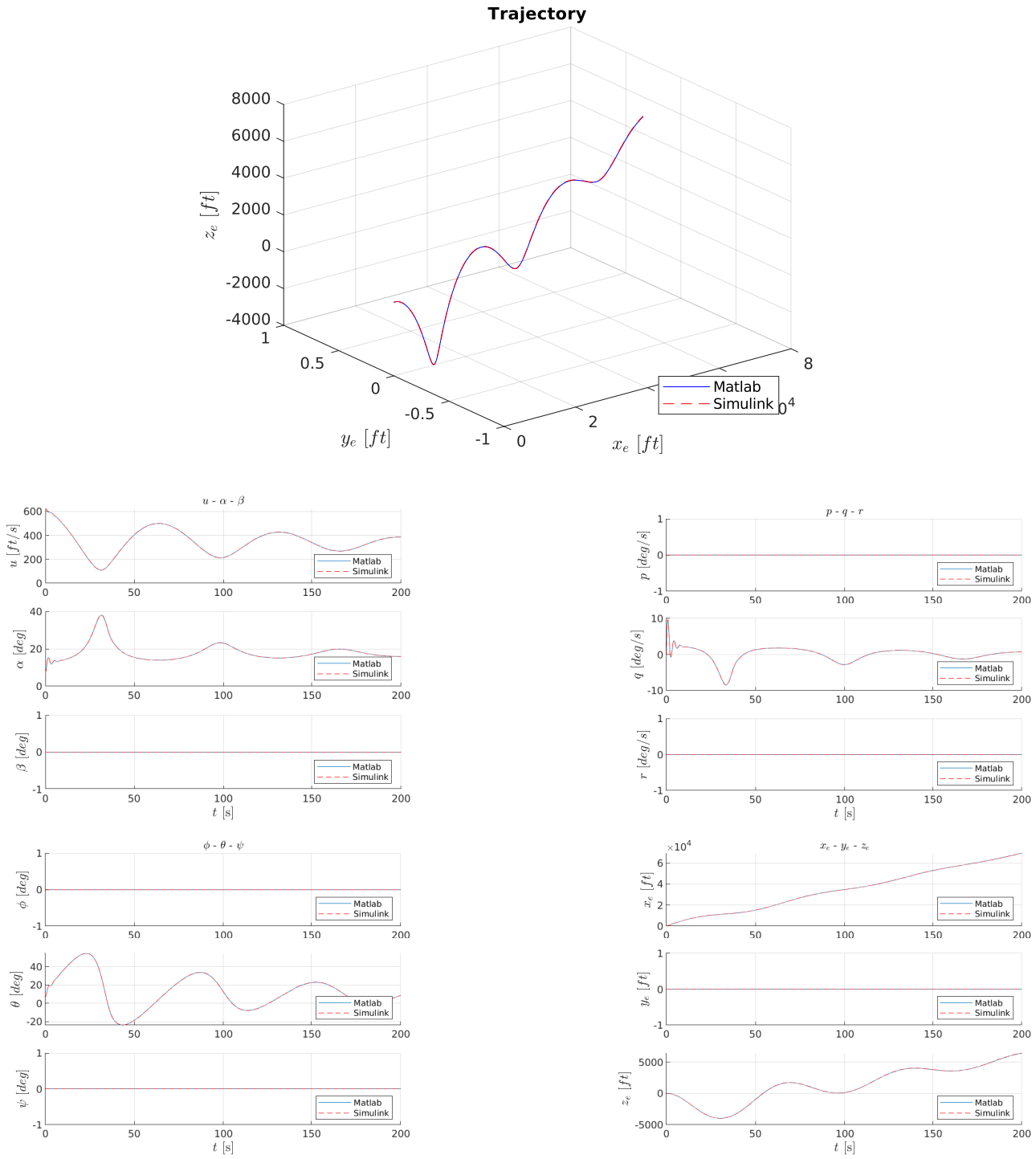


Figure 21: Response of the airplane Boeing 747 FC 5 to $-5°$ elevator

23.6 $dTH = 1000$

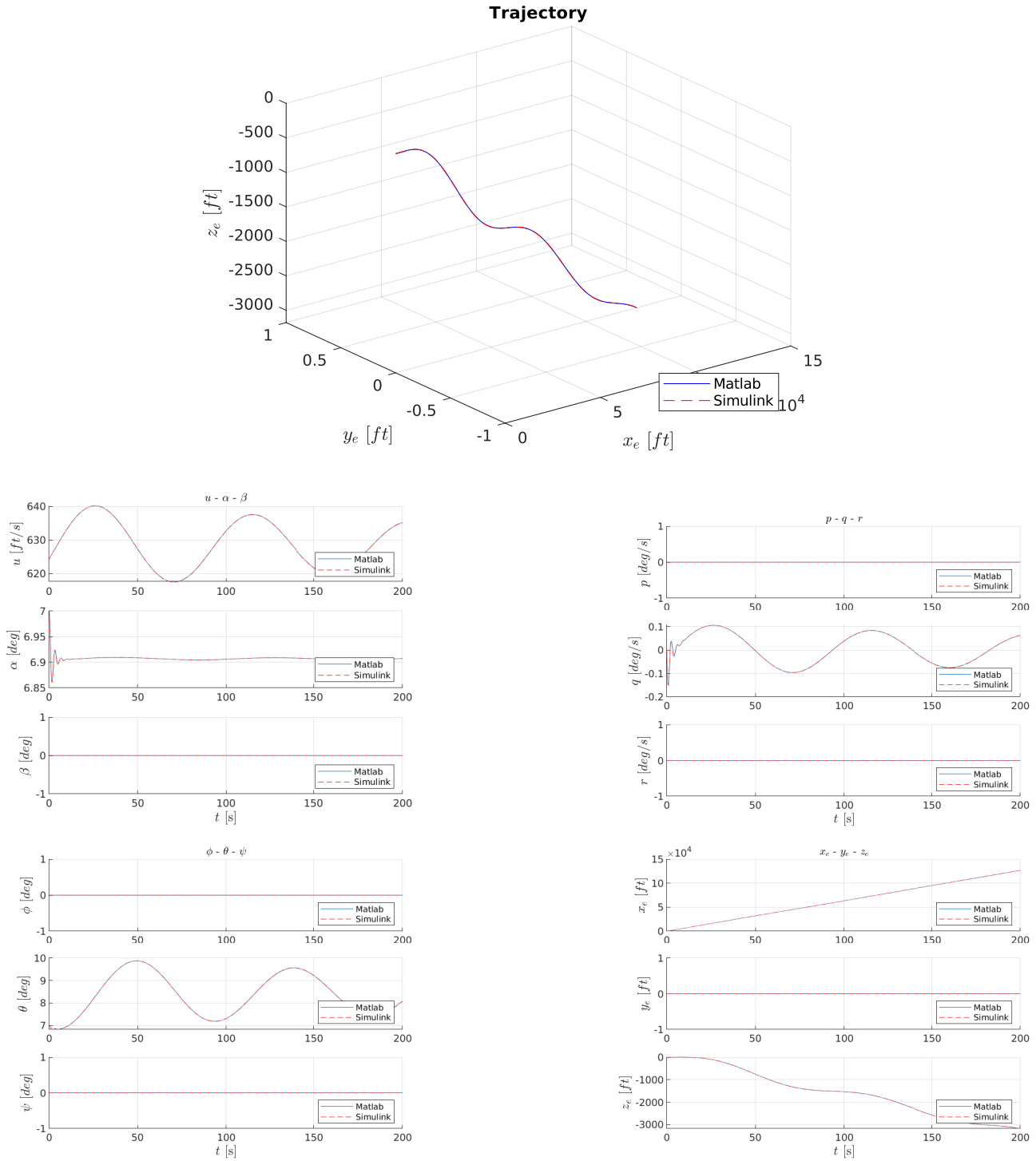


Figure 22: Response of the airplane Boeing 747 FC 5 to 1000 throttle

23.7 $dTH = 10000$

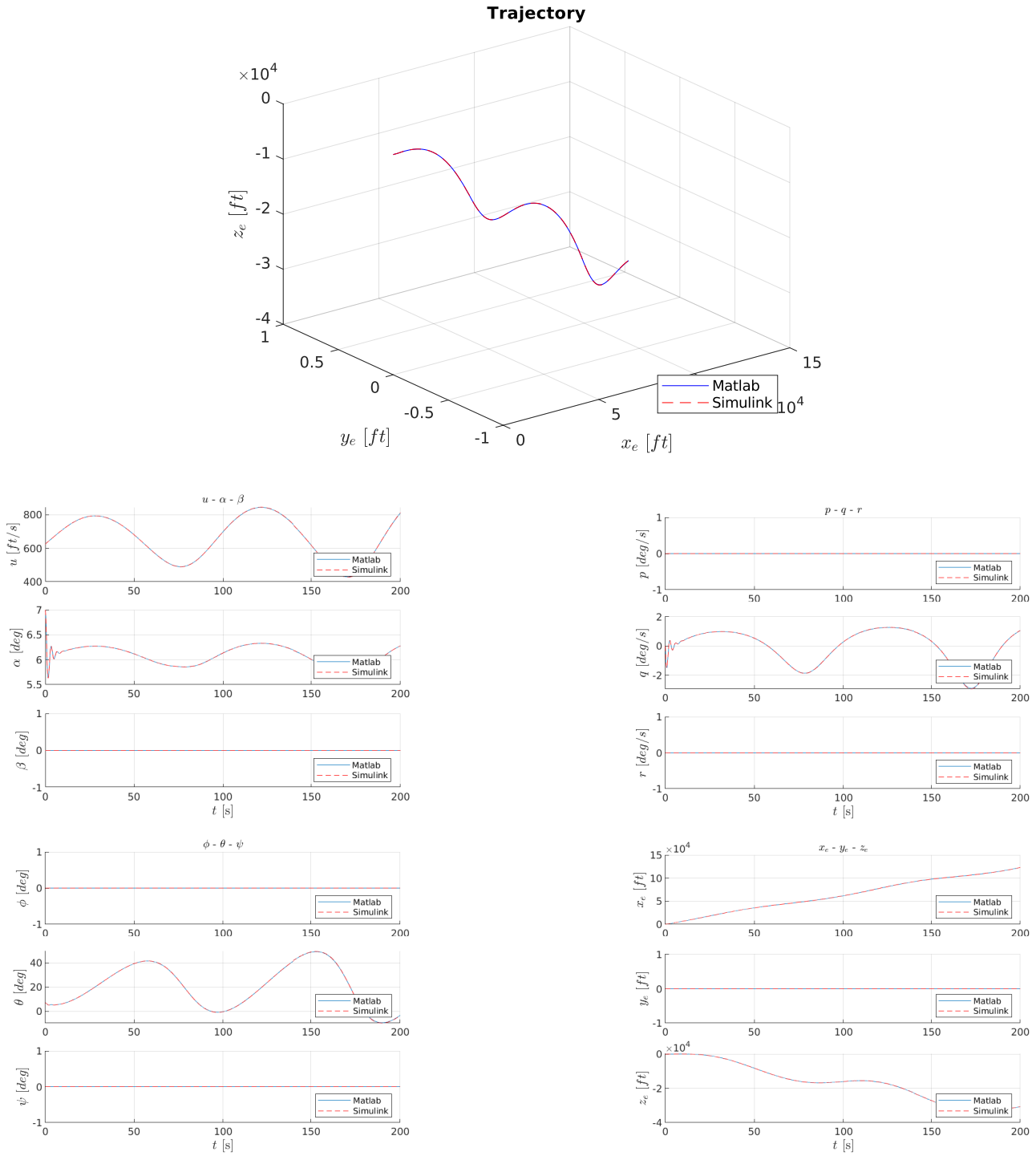


Figure 23: Response of the airplane Boeing 747 FC 5 to 10000 throttle

23.8 $dR = +5°$

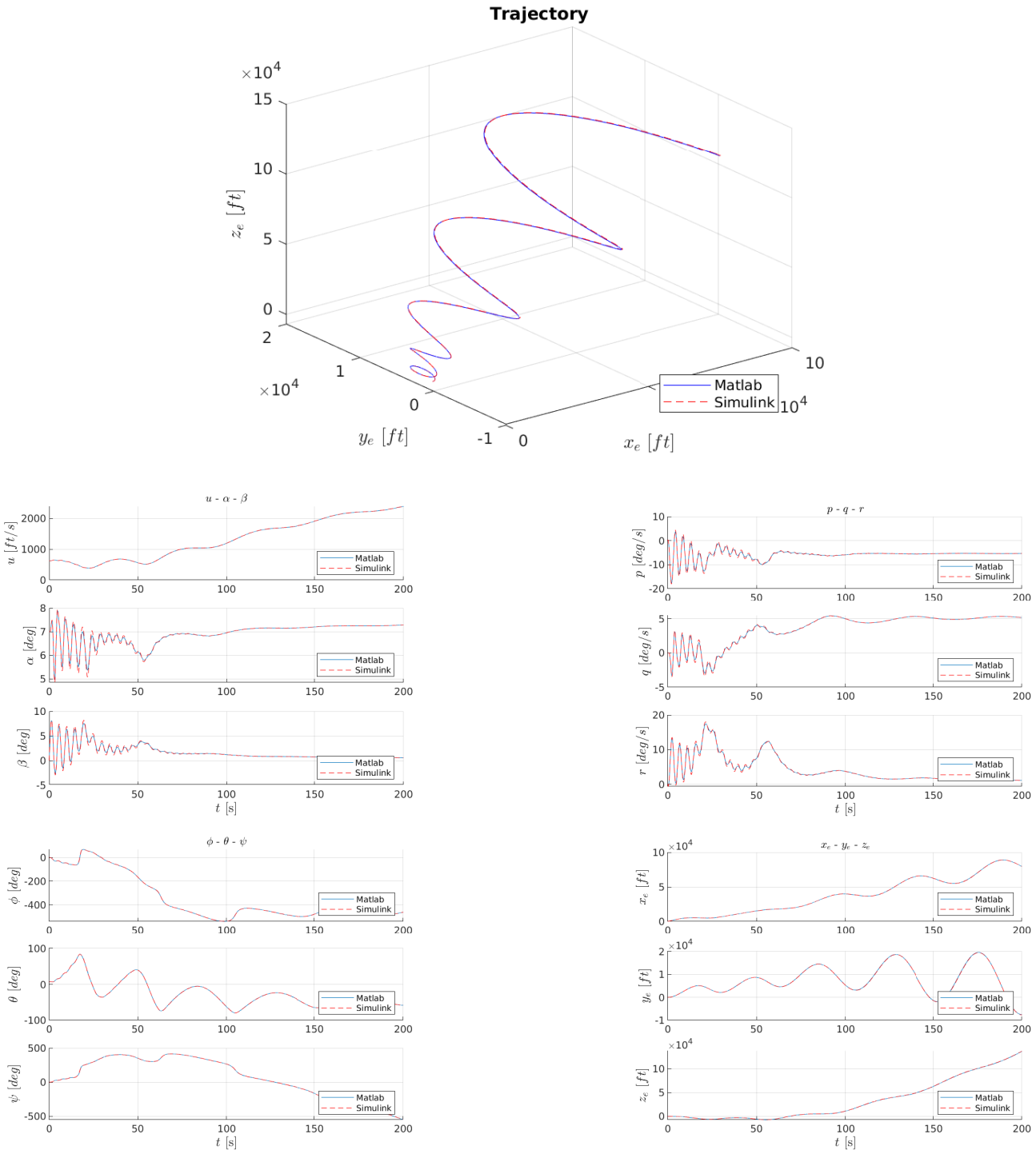


Figure 24: Response of the airplane Boeing 747 FC 5 to $+5°$ rudder

23.9 $dR = -5°$

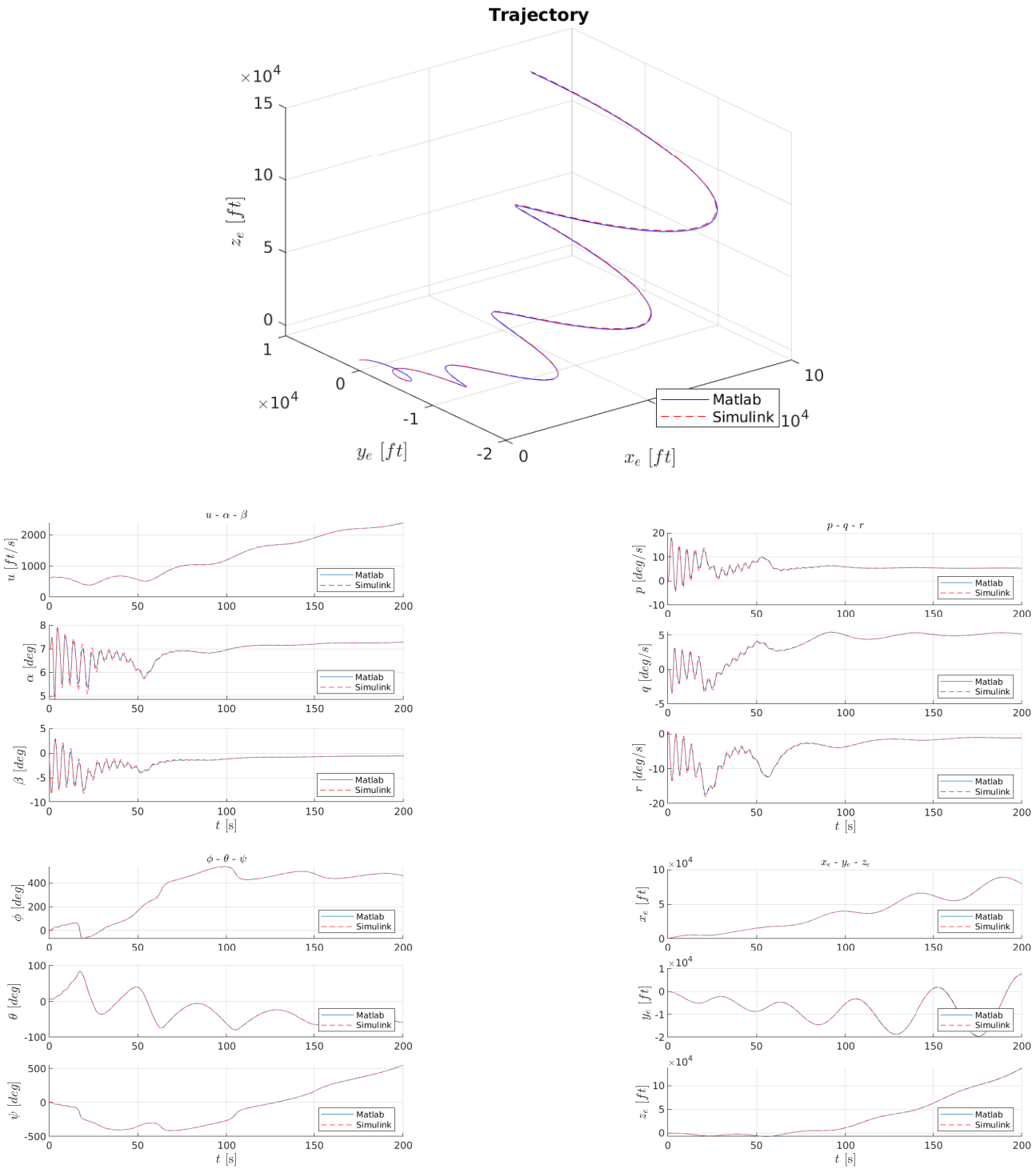


Figure 25: Response of the airplane Boeing 747 FC 5 to $-5°$ rudder

24 Extracting the Stability Derivatives of our airplane: Lockheed Jetstar; Flight Condition 9

We took the flight conditions as well as stability & control derivatives from the NACA report 2144, and we checked them using the `Excel_Validation_App.p` ...

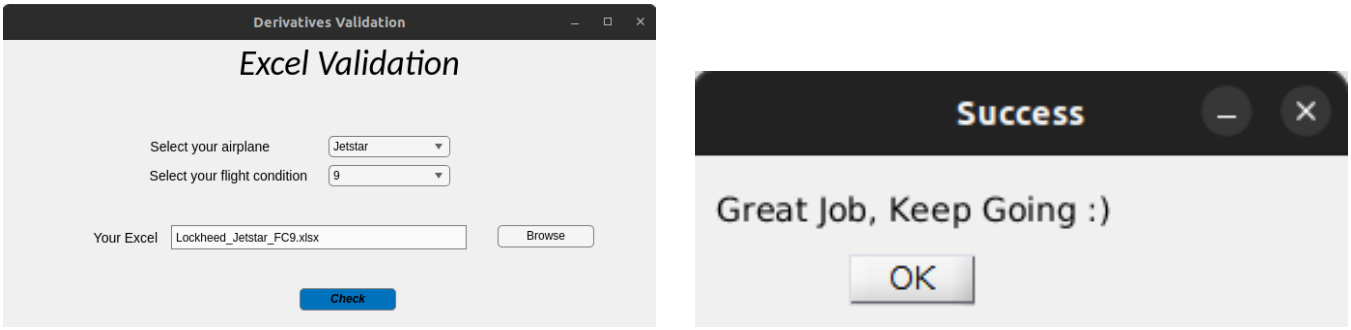


Figure 26: Excel Validation App

TABLE VII-2

JETSTAR DIMENSIONAL, MASS, AND FLIGHT CONDITION PARAMETERS

F/C #	1	2	3	4	5	6	7	8	9	10
H(FT)	SL	SL	SL	SL	20 K	20 K	20 K	40 K	40 K	40 K
M(-)	.200	.230	.400	.525	.350	.550	.750	.500	.650	.800
VTO(FPS)	224.	257.	447.	586.	363.	570.	778.	484.	629.	774.
VTO(KTAS)	133.	152.	265.	347.	215.	338.	461.	287.	373.	459.
VTO(KCAS)	132.	152.	265.	347.	158.	252.	348.	146.	193.	243.
W(LBS)	23905.	38205.	38205.	38205.	38205.	38205.	38205.	38205.	38205.	38205.
C.G.(MGC)	.250	.250	.250	.250	.250	.250	.250	.250	.250	.250
IX (SLUG-FT SQ)	42275.	118779.	118779.	118779.	118779.	118779.	118779.	118779.	118779.	118779.
IY (SLUG-FT SQ)	126106.	135876.	135876.	135876.	135876.	135876.	135876.	135876.	135876.	135876.
IZ (SLUG-FT SQ)	160113.	243518.	243518.	243518.	243518.	243518.	243518.	243518.	243518.	243518.
IXZ(SLUG-FT SQ)	5470.	5061.	5061.	5061.	5061.	5061.	5061.	5061.	5061.	5061.
EPSILON(DEG)	-2.65	-2.32	-2.32	-2.32	-2.32	-2.32	-2.32	-2.32	-2.32	-2.32
Q(PSF)	59.4	78.4	237.	408.	83.5	206.	383.	69.0	117.	177.
QC(PSF)	60.0	79.4	247.	437.	86.0	222.	440.	73.4	129.	207.
ALPHA(DEG)	6.50	11.2	4.00	2.70	9.90	4.50	2.60	11.4	7.00	4.20
GAMMA(DEG)	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
LXP(FT)	22.2	22.2	22.2	22.2	22.2	22.2	22.2	22.2	22.2	22.2
LZP(FT)	-2.40	-2.40	-2.40	-2.40	-2.40	-2.40	-2.40	-2.40	-2.40	-2.40
ITH(DEG)	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
XI(DEG)	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
LTH(FT)	-.820	-.820	-.820	-.820	-.820	-.820	-.820	-.820	-.820	-.820

TABLE VII-3

JETSTAR LONGITUDINAL DIMENSIONAL DERIVATIVES

(BODY AXIS SYSTEM)

F/C #	1	2	3	4	5	6	7	8	9	10
H	SL	SL	SL	SL	20 K	20 K	20 K	40 K	40 K	40 K
M	.200	.230	.400	.525	.350	.550	.750	.500	.650	.800
XU *	-.0166	-.00456	-.0102	-.0136	-.00324	-.00697	-.0157	-.00353	-.00168	-.211E-5
ZU *	-.175	-.103	-.0593	-.0805	-.0804	-.0436	-.0212	-.0614	-.0408	-.0348
MU *	.00131	.00175	.000549	.000727	.00102	.000815	-.000470	.000902	.000747	-.00425
XW	.108	.164	.118	.103	.111	.0918	.0689	.0858	.0498	.0266
ZW	-1.01	-.723	-1.24	-1.65	-.565	-.881	-1.33	-.354	-.475	-.655
MW	-.00991	-.00902	-.0146	-.0201	-.00665	-.0107	-.0154	-.00401	-.00561	-.00760
ZWD	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
ZQ	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
MWD	-.000910	-.000834	-.000848	-.000906	-.000447	-.000482	-.000574	-.000207	-.000237	-.000280
MQ	-.546	-.582	-1.03	-1.33	-.439	-.724	-1.09	-.279	-.380	-.506
XDE	1.97	2.78	3.02	3.51	2.62	2.96	3.34	2.49	2.66	2.54
ZDE	-17.2	-14.0	-43.2	-74.6	-15.0	-37.5	-73.5	-12.4	-21.7	-34.6
MDE	-2.26	-2.80	-8.38	-14.6	-2.95	-7.47	-14.5	-2.47	-4.27	-6.78
XDTH	.00135	.000842	.000842	.000842	.000842	.000842	.000842	.000842	.000842	.000842
ZDTH	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
MDTH	-.650E-5	-.604E-5	-.604E-5	-.604E-5	-.604E-5	-.604E-5	-.604E-5	-.604E-5	-.604E-5	-.604E-5

TABLE VII-7

JETSTAR LATERAL-DIRECTIONAL DIMENSIONAL DERIVATIVES

(BODY AXIS SYSTEM)

F/C #	1	2	3	4	5	6	7	8	9	10
h	SL	SL	SL	SL	20 K	20 K	20 K	40 K	40 K	40 K
M	.200	.230	.400	.525	.350	.550	.750	.500	.650	.800
YV	-.140	-.100	-.175	-.229	-.0756	-.119	-.167	-.0469	-.0618	-.0781
YB	-31.2	-25.8	-78.0	-134.	-27.5	-67.8	-130.	-22.7	-38.9	-60.5
LB'	-4.05	-3.42	-5.27	-7.28	-3.23	-4.43	-4.93	-2.75	-2.93	-2.27
NB'	1.34	1.10	3.30	5.47	1.21	2.99	5.63	1.02	1.75	2.66
LP'	-1.85	-.752	-1.30	-1.75	-.582	-.935	-1.34	-.380	-.492	-.635
NP'	-.245	-.173	-.164	-.187	-.121	-.119	-.137	-.0840	-.0758	-.0682
LR'	.517	.234	.181	.170	.169	.124	.0868	.105	.0936	.0551
NR'	-.190	-.172	-.261	-.333	-.125	-.178	-.252	-.0804	-.0994	-.120
Y*DA	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
L'DA	2.21	1.04	3.14	5.71	1.10	2.88	5.83	.929	1.71	2.64
N'DA	-.00557	-.0864	-.0767	-.0524	-.0770	-.0759	-.0624	-.0716	-.0831	-.0720
Y*DR	.0340	.0244	.0424	.0557	.0184	.0289	.0371	.0114	.0144	.0162
L'DR	1.11	.533	1.61	2.77	.568	1.40	2.43	.444	.766	1.21
N'DR	-.644	-.580	-1.81	-3.12	-.618	-1.55	-2.66	-.511	-.836	-1.16

Part V

Linearization of EOM & Mode Approximations

We started off with the nonlinear 6DOFs Rigid Body Dynamics Equations of motion of aircraft, given in section 10. For stability and control analysis purposes, we now start linearizing the equations using the Small Disturbance Theory, it is assumed that the motion of the airplane consists of small deviations from a reference condition of steady flight, we do this by introducing a perturbation term plus the reference value for each variable in the equations $var = var_0 + \Delta var$, we make some assumptions:

1. Only linear terms are not neglected in the equations
2. Effects of spinning rotors are negligible
3. Wind velocity is zero $V^E = V$
4. Consider $u_0 \neq 0$, $\theta_0 \neq 0$, otherwise $\text{var}_0 = 0$
5. Consider $\dot{\text{var}}_0 = 0$
6. $\cos(\Delta\text{angle}) = 1$, also $\sin(\Delta\text{angle}) = \Delta\text{angle}$
7. Using first order approximation from taylor expansion: $\tan(\text{angle}_0 + \Delta\text{angle}) \approx \tan(\text{angle}_0) + \Delta\text{angle}\ \sec^2(\text{angle}_0)$, also $\sec(\text{angle}_0 + \Delta\text{angle}) = \sec(\text{angle}_0) + \Delta\text{angle}\ \sec(\text{angle}_0)\tan(\text{angle}_0)$.

Equations become:

$$
\begin{aligned}
X_0 + \Delta X - mg\sin(\theta_0 + \Delta\theta) &= m\left(\cancelto{0}{\dot{u}_0} + \Delta\dot{u} + \left(\cancelto{0}{q_0} + \Delta q\right)(w_0 + \Delta w) - \left(\cancelto{0}{r_0} + \Delta r\right)\left(\cancelto{0}{v_0} + \Delta v\right)\right) \\
Y_0 + \Delta Y + mg\cos(\theta_0 + \Delta\theta)\sin(\phi_0 + \Delta\phi) &= m\left(\cancelto{0}{\dot{v}_0} + \Delta\dot{v} + \left(\cancelto{0}{r_0} + \Delta r\right)(u_0 + \Delta u) - \left(\cancelto{0}{p_0} + \Delta p\right)(w_0 + \Delta w)\right) \\
Z_0 + \Delta Z + mg\cos(\theta_0 + \Delta\theta)\cos(\phi_0 + \Delta\phi) &= m\left(\cancelto{0}{\dot{w}_0} + \Delta\dot{w} + \left(\cancelto{0}{p_0} + \Delta p\right)\left(\cancelto{0}{v_0} + \Delta v\right) - \left(\cancelto{0}{q_0} + \Delta q\right)(u_0 + \Delta u)\right)
\end{aligned}
$$

$$
\begin{aligned}
L_0 + \Delta L &= I_x\left(\cancelto{0}{\dot{p}_0} + \Delta\dot{p}\right) - I_{zx}\left(\cancelto{0}{\dot{r}_0} + \Delta\dot{r}\right) + 0 \\
M_0 + \Delta M &= I_y\Delta\dot{q} + 0 \\
N_0 + \Delta N &= I_z\Delta\dot{r} - I_{zx}\Delta\dot{p} + 0
\end{aligned}
$$

$$\dot{\phi}_0 + \Delta\dot{\phi} = p_0 + \Delta p + ((q_0 + \Delta q)\sin(\phi_0 + \Delta\phi) + (r_0 + \Delta r)\cos(\phi_0 + \Delta\phi))\tan(\theta_0 + \Delta\theta)$$

$$\dot{\theta}_0 + \Delta\dot{\theta} = (q_0 + \Delta q)\cos(\phi_0 + \Delta\phi) - (r_0 + \Delta r)\sin(\phi_0 + \Delta\phi)$$

$$\dot{\psi}_0 + \Delta\dot{\psi} = ((q_0 + \Delta q)\sin(\phi_0 + \Delta\phi) + (r_0 + \Delta r)\cos(\phi_0 + \Delta\phi))\sec(\theta_0 + \Delta\theta)$$

$$\dot{x}_{E_0} + \Delta\dot{x}_E = (u_0 + \Delta u)\cos(\theta_0 + \Delta\theta)\cos(\Delta\psi) + 0 + \Delta w\sin(\theta_0 + \Delta\theta)$$

$$\dot{y}_{E_0} + \Delta\dot{y}_E = u_0\cos(\theta_0 + \Delta\theta)\sin(\Delta\psi) + \Delta v * 1 + 0$$

$$\dot{z}_{E_0} + \Delta\dot{z}_E = -(u_0 + \Delta u)\sin(\theta_0 + \Delta\theta) + 0 + \Delta w\cos(\phi_0 + \Delta\phi)\cos(\theta_0 + \Delta\theta)$$

Simplifying:

$$X_0 + \Delta X - mg(\sin(\theta_0)\cos(\Delta\theta) + \sin(\Delta\theta)\cos(\theta_0)) = m(\Delta\dot{u} + w_0\Delta q)$$

$$Y_0 + \Delta Y + mg(\cos(\theta_0)\cos(\Delta\theta) - \sin(\theta_0)\sin(\Delta\theta))\sin(\Delta\phi) = m(\Delta\dot{v} + u_0\Delta r - w_0\Delta p)$$

$$Z_0 + \Delta Z + mg(\cos(\theta_0)\cos(\Delta\theta) - \sin(\theta_0)\sin(\Delta\theta))\cos(\Delta\phi) = m(\Delta\dot{w} - u_0\Delta q)$$

$$L_0 + \Delta L = I_x\Delta\dot{p} - I_{zx}\Delta\dot{r}$$

$$M_0 + \Delta M = I_y\Delta\dot{q}$$

$$N_0 + \Delta N = I_z\Delta\dot{r} - I_{zx}\Delta\dot{p}$$

$$\begin{aligned}
\Delta\dot{\phi} &= \Delta p + (\Delta q \sin(\Delta\phi) + \Delta r \cos(\Delta\phi))(\tan(\theta_0) + \Delta\theta \sec^2(\theta_0)) \\
\Delta\dot{\theta} &= \Delta q \cos(\Delta\phi) - \Delta r \sin(\Delta\phi) \\
\Delta\dot{\psi} &= (\Delta q \sin(\Delta\phi) + \Delta r \cos(\Delta\phi))(\sec(\theta_0) + \Delta\theta \sec(\theta_0)\tan(\theta_0))
\end{aligned}$$

$$\begin{aligned}
\dot{x}_{E_0} + \Delta\dot{x}_E &= (u_0 + \Delta u)(\cos(\theta_0)\cos(\Delta\theta) - \sin(\theta_0)\sin(\Delta\theta))\cos(\Delta\psi) + \Delta w(\sin(\theta_0)\cos(\Delta\theta) + \sin(\Delta\theta)\cos(\theta_0)) \\
\dot{y}_{E_0} + \Delta\dot{y}_E &= u_0(\cos(\theta_0)\cos(\Delta\theta) - \sin(\theta_0)\sin(\Delta\theta))\sin(\Delta\psi) + \Delta v \\
\dot{z}_{E_0} + \Delta\dot{z}_E &= -(u_0 + \Delta u)(\sin(\theta_0)\cos(\Delta\theta) + \sin(\Delta\theta)\cos(\theta_0)) + \Delta w(\cos(\theta_0)\cos(\Delta\theta) - \sin(\theta_0)\sin(\Delta\theta))
\end{aligned}$$

Further simplification:

$$\begin{aligned}
X_0 + \Delta X - mg(\sin(\theta_0) + \Delta\theta\cos(\theta_0)) &= m(\Delta\dot{u} + w_0\Delta q) \\
Y_0 + \Delta Y + mg\cos(\theta_0)\Delta\phi &= m(\Delta\dot{v} + u_0\Delta r - w_0\Delta p) \\
Z_0 + \Delta Z + mg(\cos(\theta_0) - \sin(\theta_0)\Delta\theta) &= m(\Delta\dot{w} - u_0\Delta q)
\end{aligned}$$

$$\begin{aligned}
L_0 + \Delta L &= I_x\Delta\dot{p} - I_{zx}\Delta\dot{r} \\
M_0 + \Delta M &= I_y\Delta\dot{q} \\
N_0 + \Delta N &= I_z\Delta\dot{r} - I_{zx}\Delta\dot{p}
\end{aligned}$$

$$\begin{aligned}
\Delta\dot{\phi} &= \Delta p + \Delta r\tan(\theta_0) \\
\Delta\dot{\theta} &= \Delta q \\
\Delta\dot{\psi} &= \Delta r\sec(\theta_0)
\end{aligned}$$

$$\begin{aligned}
\dot{x}_{E_0} + \Delta\dot{x}_E &= (u_0 + \Delta u)(\cos(\theta_0)) - u_0 \sin(\theta_0)\Delta\theta + \Delta w \sin(\theta_0) \\
\dot{y}_{E_0} + \Delta\dot{y}_E &= u_0 \cos(\theta_0)\Delta\psi + \Delta v \\
\dot{z}_{E_0} + \Delta\dot{z}_E &= -(u_0 + \Delta u)(\sin(\theta_0)) - u_0\Delta\theta\cos(\theta_0) + \Delta w \cos(\theta_0)
\end{aligned}$$

Using the reference steady state forces, we can eliminate the reference forces at equilibrium condition:

$$X_0 - mg\sin\theta_0 = 0$$

$$Y_0 = 0$$

$$Z_0 + mg\cos\theta_0 = 0$$

$$L_0 = M_0 = N_0 = 0$$

$$\dot{x}_{E_0} = u_0\cos\theta_0,\ \dot{y}_{E_0} = 0,\ \dot{z}_{E_0} = -u_0\sin\theta_0$$

Substitute into the above equations and drop the Δ notation off the states:

$$\begin{aligned}
\Delta X - mg\cos\theta_0\ \theta &= m(\dot{u} + w_0\Delta q) \\
\Delta Y + mg\cos\theta_0\ \phi &= m(\dot{v} + u_0 r - w_0\Delta p) \\
\Delta Z - mg\sin\theta_0\ \theta &= m(\dot{w} - u_0 q)
\end{aligned}$$

$$\begin{aligned}
\Delta L &= I_x\dot{p} - I_{zx}\dot{r} \\
\Delta M &= I_y\dot{q} \\
\Delta N &= I_z\dot{r} - I_{zx}\dot{p}
\end{aligned}$$

$$\begin{aligned}
\dot{\phi} &= p + r\tan\theta_0 \\
\dot{\theta} &= q \\
\dot{\psi} &= r\sec\theta_0
\end{aligned}$$

$$\begin{aligned}
\dot{x}_E &= u\cos\theta_0 - u_0\sin\theta_0\ \theta + w\sin\theta_0 \\
\dot{y}_E &= u_0\cos\theta_0\ \psi + v \\
\dot{z}_E &= -u\sin\theta_0 - u_0\cos\theta_0\ \theta + w\cos\theta_0
\end{aligned}$$

Using the first order taylor series approximation for forces (used before in the airframe model):

$$\begin{aligned}
\Delta X &= X_u u + X_w w + X_{\delta_e}\delta_e + X_{\delta_{th}}\delta_{th} \\
\Delta Y &= Y_v v + Y_p p + Y_r r + Y_{\delta_a}\delta_a + Y_{\delta_r}\delta_r \\
\Delta Z &= Z_u u + Z_w w + Z_{\dot{w}}\dot{w} + Z_q q + Z_{\delta_e}\delta_e + Z_{\delta_{th}}\delta_{th} \\
\Delta L &= L_v v + L_p p + L_r r + L_{\delta_a}\delta_a + L_{\delta_r}\delta_r \\
\Delta M &= M_u u + M_w w + M_{\dot{w}}\dot{w} + M_q q + M_{\delta_e}\delta_e + M_{\delta_{th}}\delta_{th} \\
\Delta N &= N_v v + N_p p + N_r r + N_{\delta_a}\delta_a + N_{\delta_r}\delta_r
\end{aligned}$$

Substitute into the above equations, and rearranging, we get some equations decoupled from the others, the first set of equations are called longitudinal dynamics equations (dynamics in the plane of symmetry of the airplane), and the second set is the lateral dynamics equations (dynamics out of the plane of symmetry).

The equations in terms of the derivatives of the states are:

$$\begin{aligned}
\dot{u} &= \frac{\Delta X}{m} - w_0 q - g\cos\theta_0\ \theta \\
\dot{v} &= \frac{\Delta Y}{m} - g\cos\theta_0\ \phi + w_0 p - u_0 r \\
\dot{w} &= \frac{\Delta Z}{m} - g\sin\theta_0\ \theta + u_0 q
\end{aligned}$$

solving for $\dot{p}$ and $\dot{r}$:

$$\dot{r} = \frac{1}{I_{zx}}\left(I_x\dot{p} - \Delta L\right)$$

therefore:

$$\Delta N = \frac{I_z}{I_{zx}}\left(I_x\dot{p} - \Delta L\right) - I_{zx}\dot{p}$$

hence:

$$\dot{p} = \frac{I_{zx}}{I_z I_x - I_{zx}^2}\Delta N + \frac{I_z}{I_z I_x - I_{zx}^2}\Delta L$$

and

$$\dot{r} = \frac{I_x}{I_z I_x - I_{zx}^2}\Delta N + \frac{I_{zx}}{I_z I_x - I_{zx}^2}\Delta L$$

$$\begin{aligned}
\dot{p} &= \frac{I_z}{I_z I_x - I_{zx}^2}\Delta L + \frac{I_{zx}}{I_z I_x - I_{zx}^2}\Delta N \\
\dot{q} &= \frac{\Delta M}{I_y} \\
\dot{r} &= \frac{I_{zx}}{I_z I_x - I_{zx}^2}\Delta L + \frac{I_x}{I_z I_x - I_{zx}^2}\Delta N
\end{aligned}$$

and the Euler angles equations as well as the inertial velocity equations are same as they are.

25 Linearized Longitudinal Dynamics

$$\frac{d}{dt}\begin{bmatrix} u \\ w \\ q \\ \theta \end{bmatrix} = \begin{bmatrix} \frac{X_u}{m} & \frac{X_w}{m} & -w_0 & -g\cos\theta_0 \\ \frac{Z_u}{m-Z_{\dot{w}}} & \frac{Z_w}{m-Z_{\dot{w}}} & \frac{Z_q+mu_0}{m-Z_{\dot{w}}} & \frac{-mg\sin\theta_0}{m-Z_{\dot{w}}} \\ \frac{1}{I_y}\left(M_u + M_{\dot{w}}\frac{Z_u}{m-Z_{\dot{w}}}\right) & \frac{1}{I_y}\left(M_w + M_{\dot{w}}\frac{Z_w}{m-Z_{\dot{w}}}\right) & \frac{1}{I_y}\left(M_q + M_{\dot{w}}\frac{Z_q+mu_0}{m-Z_{\dot{w}}}\right) & -\frac{M_{\dot{w}}}{I_y}\frac{mg\sin\theta_0}{m-Z_{\dot{w}}} \\ 0 & 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} u \\ w \\ q \\ \theta \end{bmatrix} + \begin{bmatrix} \frac{X_{\delta_e}}{m} & \frac{X_{\delta_{th}}}{m} \\ \frac{Z_{\delta_e}}{m-Z_{\dot{w}}} & \frac{Z_{\delta_e}}{m-Z_{\dot{w}}} \\ \frac{1}{I_y}\left(M_{\delta_e} + M_{\dot{w}}\frac{Z_{\delta_e}}{m-Z_{\dot{w}}}\right) & \frac{1}{I_y}\left(M_{\delta_{th}} + M_{\dot{w}}\frac{Z_{\delta_{th}}}{m-Z_{\dot{w}}}\right) \\ 0 & 0 \end{bmatrix}\begin{bmatrix} \delta_e \\ \delta_{th} \end{bmatrix} \tag{5}$$

with inertial velocity equations:

$$\dot{x}_E = u\cos\theta_0 + w\sin\theta_0 - u_0\sin\theta_0\ \theta$$

$$\dot{z}_E = -u\sin\theta_0 + w\cos\theta_0 - u_0\cos\theta_0\ \theta$$

For our specific airplane (Jetstar):

$$\frac{d}{dt}\begin{bmatrix} u \\ w \\ q \\ \theta \end{bmatrix} = \begin{bmatrix} -0.0017 & 0.0498 & -76.6558 & -31.9342 \\ -0.0408 & -0.4750 & 624.3115 & -3.9210 \\ 7.5667e-04 & -0.0055 & -0.5280 & 9.2928e-04 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} u \\ w \\ q \\ \theta \end{bmatrix} + \begin{bmatrix} 2.6600 & 8.4200e-04 \\ -21.7000 & 0 \\ -4.2649 & -6.0400e-06 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \delta_e \\ \delta_{th} \end{bmatrix}$$

26 Linearized Lateral Dynamics

$$\frac{d}{dt}\begin{bmatrix} v \\ p \\ r \\ \phi \end{bmatrix} = \begin{bmatrix} \frac{Y_v}{m} & \frac{Y_p}{m} + w_0 & \frac{Y_r}{m} - u_0 & g\cos\theta_0 \\ \frac{L_v}{I'_x} + I'_{zx}N_v & \frac{L_p}{I'_x} + I'_{zx}N_p & \frac{L_r}{I'_x} + I'_{zx}N_r & 0 \\ I'_{zx}L_v + \frac{N_v}{I'_z} & I'_{zx}L_p + \frac{N_p}{I'_z} & I'_{zx}L_r + \frac{N_r}{I'_z} & 0 \\ 0 & 1 & \tan\theta_0 & 0 \end{bmatrix} \begin{bmatrix} v \\ p \\ r \\ \phi \end{bmatrix} + \begin{bmatrix} \frac{Y_{\delta_a}}{m} & \frac{Y_{\delta_r}}{m} \\ \frac{L_{\delta_a}}{I'_x} + I'_{zx}N_{\delta_a} & \frac{L_{\delta_r}}{I'_x} + I'_{zx}N_{\delta_r} \\ I'_{zx}L_{\delta_a} + \frac{N_{\delta_a}}{I'_z} & I'_{zx}L_{\delta_r} + \frac{N_{\delta_r}}{I'_z} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \delta_a \\ \delta_r \end{bmatrix} \tag{6}$$

notice that this is the convention used in Etkin [4], which is found to be so confusing anyway:

$$I'_x = \frac{I_z}{I_z I_x - I_{zx}^2}$$

$$I'_z = \frac{I_x}{I_z I_x - I_{zx}^2}$$

$$I'_{zx} = \frac{I_z I_x - I_{zx}^2}{I_{zx}}$$

and equations:

$$\dot{\psi} = r \sec \theta_0$$

$$\dot{y}_E = u_0 \cos \theta_0 \ \psi + v$$

For our specific airplane (Jetstar):

$$\frac{d}{dt}\begin{bmatrix} v \\ p \\ r \\ \phi \end{bmatrix} = \begin{bmatrix} -0.0618 & 0.1219 & -0.9925 & 0.0508 \\ -0.0047 & -0.4920 & 0.0936 & 0 \\ 0.0028 & -0.0758 & -0.0994 & 0 \\ 0 & 1 & 0.1228 & 0 \end{bmatrix}\begin{bmatrix} v \\ p \\ r \\ \phi \end{bmatrix} + \begin{bmatrix} 0 & 1.7100 \\ -0.0831 & 0.7660 \\ 0.0144 & -0.8360 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} \delta_a \\ \delta_r \end{bmatrix}$$

27 Longitudinal Mode Approximations

The numerical solutions for the the full linearized system does not give much physical insight into their genesis, but if we consider some simpler modes are only a combination of mass-spring-damper and mass-spring systems, this might give an insight on what parameters in the model equivalently represent the parameters of the mass-spring-damper system, so we now need approximate analytical solutions, this can be achieved in two ways:

1. to assume that the mode is either only due to two small real eigenvalues and neglect the larger (faster) ones $D\lambda + E = 0$ or if complex eigenvalues exist then $C\lambda^2 + D\lambda + E = 0$.

2. to consider the states that are most dominant in a specific response as a separate mode.

The method used here is the second method to obtain mode approximations, which depends on the physical intuition of what states affect the mode and what states don't. Please note that all stability derivatives in this section are the ones normalized by m, I_x, I_y, or I_z.

27.1 Short Period Approximations

The short period is a mode where speed is relatively constant while the pitch and pitch rate are changing rapidly, therefore by assuming $\Delta u = 0$, Etkin [4] ignored $Z_{\dot{w}}$ because it is small relative to m and ignored Z_q because it is small relative to mu_0, but this is not done here, so:

$$\begin{bmatrix} \dot{w} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} \frac{Z_w}{1-Z_{\dot{w}}} & \frac{Z_q+u_0}{1-Z_{\dot{w}}} \\ M_w + M_{\dot{w}}\frac{Z_w}{1-Z_{\dot{w}}} & M_q + M_{\dot{w}}\frac{Z_q+u_0}{1-Z_{\dot{w}}} \end{bmatrix} \begin{bmatrix} w \\ q \end{bmatrix} + \begin{bmatrix} \frac{Z_{\delta_e}}{1-Z_{\dot{w}}} & \frac{Z_{\delta_{th}}}{1-Z_{\dot{w}}} \\ M_{\delta_a} + \frac{M_{\dot{w}}Z_{\delta_e}}{1-Z_{\dot{w}}} & M_{\delta_{th}} + \frac{M_{\dot{w}}Z_{\delta_{th}}}{1-Z_{\dot{w}}} \end{bmatrix} \begin{bmatrix} \delta_e \\ \delta_{th} \end{bmatrix}$$

27.2 Long Period (Phugoid) Approximations

In this mode approximation we assume that the values of both $\dot{w}$ and q are relatively small, so we can ignore the stability derivatives $Z_{\dot{w}}$, $M_{\dot{w}}$ and M_q, and also assume $\dot{q} = 0$

$$\frac{d}{dt}\begin{bmatrix} u \\ \theta \end{bmatrix} = \begin{bmatrix} X_u & -g\cos(\theta_0) \\ \frac{-Z_u}{Z_q+u_0} & \frac{g\sin(\theta_0)}{Z_q+u_0} \end{bmatrix} \begin{bmatrix} u \\ \theta \end{bmatrix} + \begin{bmatrix} X_{\delta_e} & X_{\delta_{th}} \\ \frac{-Z_{\delta_e}}{Z_q+u_0} & \frac{-Z_{\delta_{th}}}{Z_q+u_0} \end{bmatrix} \begin{bmatrix} \delta_e \\ \delta_{th} \end{bmatrix}$$

27.3 Frequency Domain

Here we analyze all the linear systems we obtained in the frequency domain, we notice that the bode plots, as well as the root locus plots are much more consistent (with respect to how much the approximations align with the full linearized model) than appears in the responses of the systems.

Notice the notation used here:

- "lin" means full linearized model
- "lp" means long period mode
- "sp" means short period

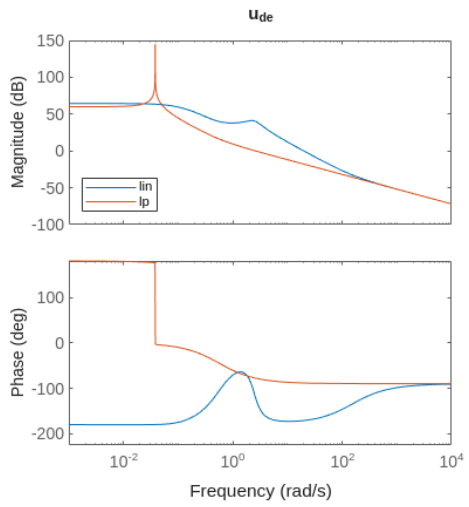
Transfer Functions

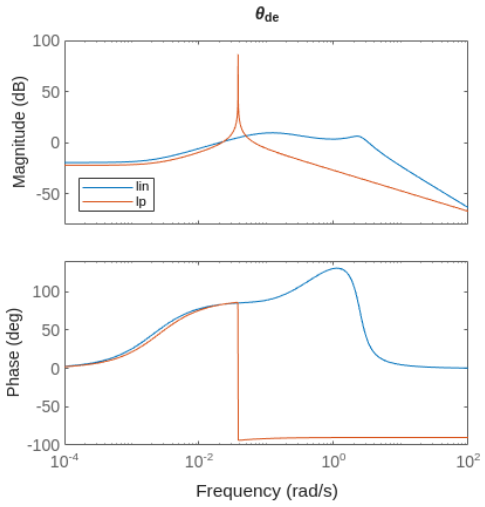
```
u_de =

        2.54 s^3 + 386.4 s^2 + 334 s + 136.6
  ---------------------------------------------
  s^4 + 1.387 s^3 + 5.963 s^2 - 0.2114 s - 0.08247

Continuous-time transfer function.
Model Properties

w_de =

       -34.6 s^3 - 5251 s^2 + 2.548 s - 2.827
  ---------------------------------------------
  s^4 + 1.387 s^3 + 5.963 s^2 - 0.2114 s - 0.08247

Continuous-time transfer function.
Model Properties

q_de =

   -6.77 s^3 - 4.257 s^2 - 0.00888 s - 1.748e-20
  ---------------------------------------------
  s^4 + 1.387 s^3 + 5.963 s^2 - 0.2114 s - 0.08247

Continuous-time transfer function.
Model Properties

theta_de =

           -6.77 s^2 - 4.257 s - 0.00888
  ---------------------------------------------
  s^4 + 1.387 s^3 + 5.963 s^2 - 0.2114 s - 0.08247

Continuous-time transfer function.
Model Properties

u_dth =

  0.000842 s^3 + 0.00151 s^2 + 0.00552 s + 0.0001142
  --------------------------------------------------
   s^4 + 1.387 s^3 + 5.963 s^2 - 0.2114 s - 0.08247

Continuous-time transfer function.
Model Properties

w_dth =

       -0.004692 s^2 - 0.002775 s + 1.688e-06
  ---------------------------------------------
  s^4 + 1.387 s^3 + 5.963 s^2 - 0.2114 s - 0.08247

Continuous-time transfer function.
Model Properties

q_dth =

  -6.04e-06 s^3 - 7.587e-06 s^2 - 2.163e-06 s + 1.269e-24
  -------------------------------------------------------
     s^4 + 1.387 s^3 + 5.963 s^2 - 0.2114 s - 0.08247

Continuous-time transfer function.
Model Properties

theta_dth =

      -6.04e-06 s^2 - 7.587e-06 s - 2.163e-06
  ---------------------------------------------
  s^4 + 1.387 s^3 + 5.963 s^2 - 0.2114 s - 0.08247
```

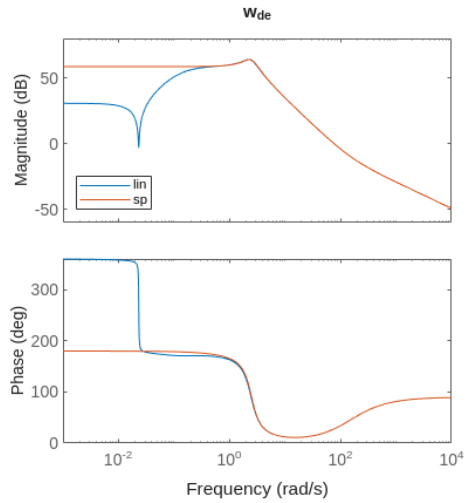
```
Continuous-time transfer function.
Model Properties

u_de_lp =

        2.54 s - 1.438
  --------------------------
  s^2 + 2.11e-06 s + 0.001447

Continuous-time transfer function.
Model Properties

u_dth_lp =

          0.000842 s
  --------------------------
  s^2 + 2.11e-06 s + 0.001447

Continuous-time transfer function.
Model Properties

theta_de_lp =

     0.04482 s + 0.0001146
  --------------------------
  s^2 + 2.11e-06 s + 0.001447

Continuous-time transfer function.
Model Properties

theta_dth_lp =

           3.796e-08
  --------------------------
  s^2 + 2.11e-06 s + 0.001447

Continuous-time transfer function.
Model Properties

w_de_sp =

     -34.6 s - 5251
  --------------------
  s^2 + 1.387 s + 6.203

Continuous-time transfer function.
Model Properties

w_dth_sp =

        -0.004662
  --------------------
  s^2 + 1.387 s + 6.203

Continuous-time transfer function.
Model Properties

q_de_sp =

     -6.77 s - 4.246
  --------------------
  s^2 + 1.387 s + 6.203

Continuous-time transfer function.
Model Properties

q_dth_sp =

  -6.04e-06 s - 4.017e-06
  -----------------------
```

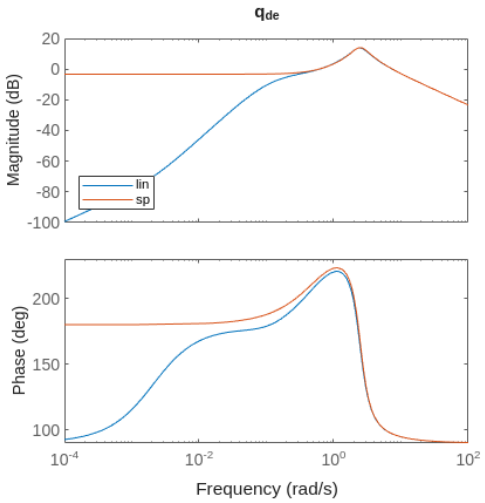
```
   s^2 + 1.387 s + 6.203

Continuous-time transfer function.
Model Properties
```

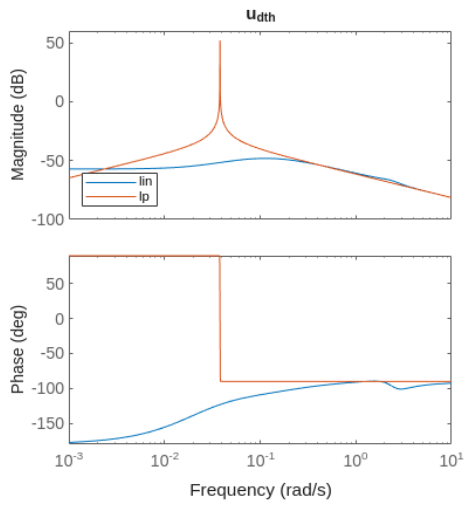
bode Plots

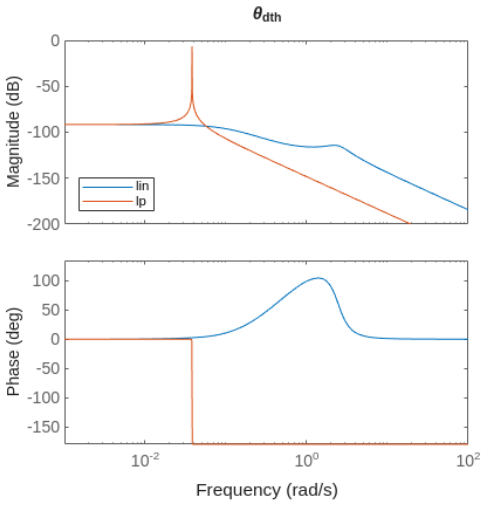


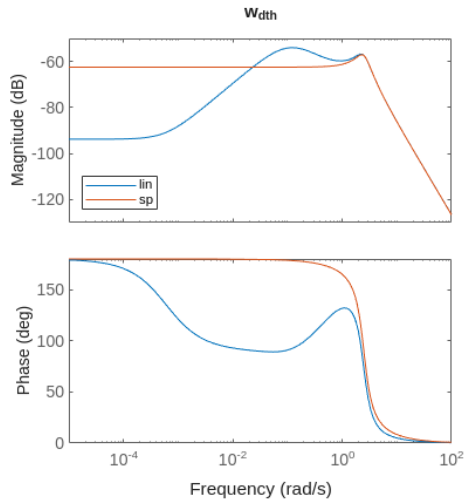


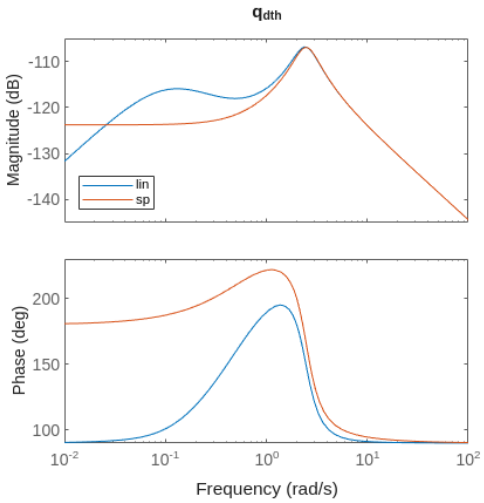




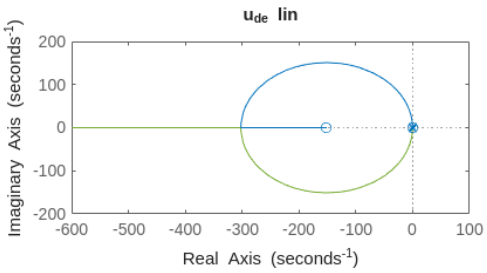


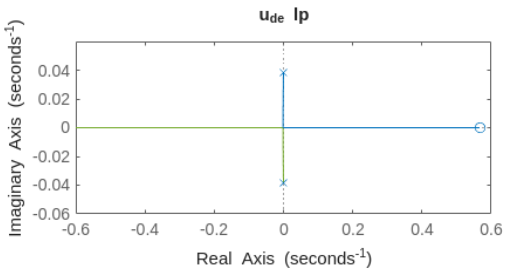


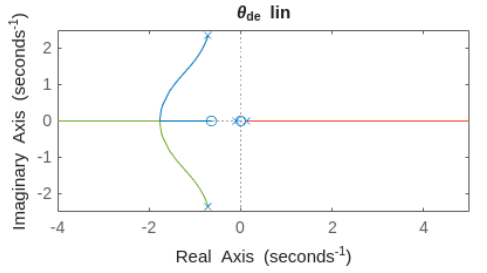


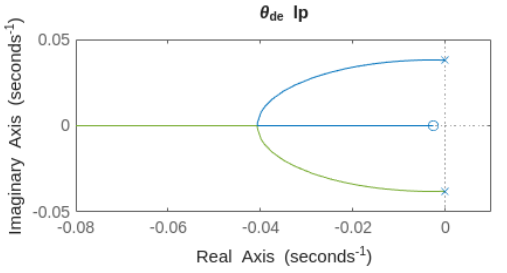


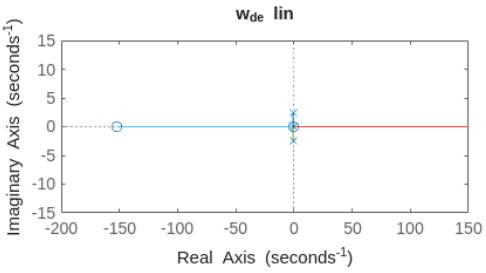
Root Locus Plots

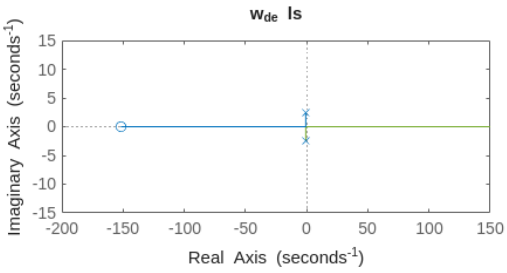


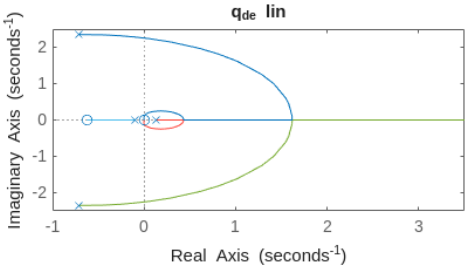


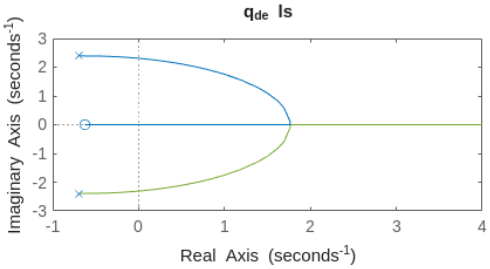


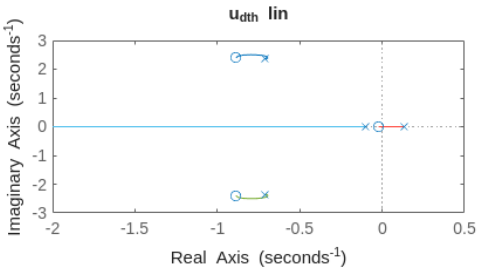


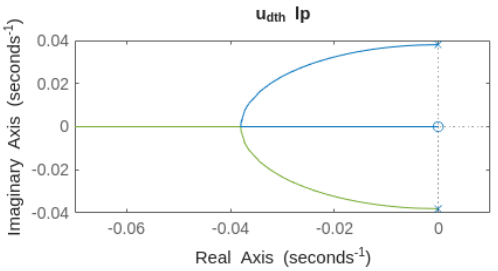

w_{de} lin
Imaginary Axis (seconds^{-1})
Real Axis (seconds^{-1})

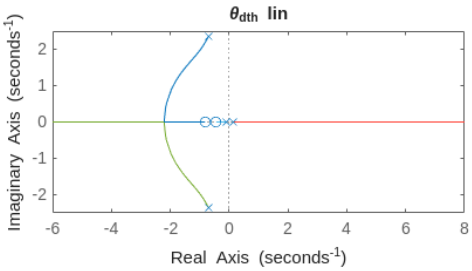

w_{de} ls
Imaginary Axis (seconds^{-1})
Real Axis (seconds^{-1})

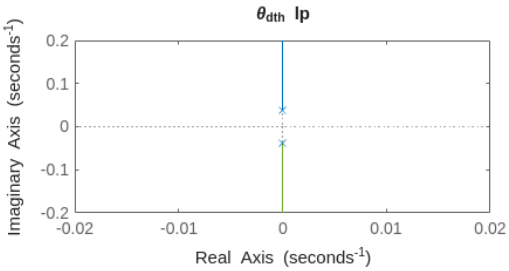

q_{de} lin
Imaginary Axis (seconds^{-1})
Real Axis (seconds^{-1})

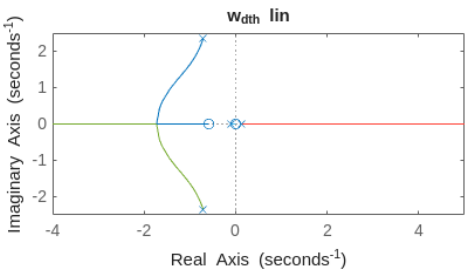

q_{de} ls
Imaginary Axis (seconds^{-1})
Real Axis (seconds^{-1})

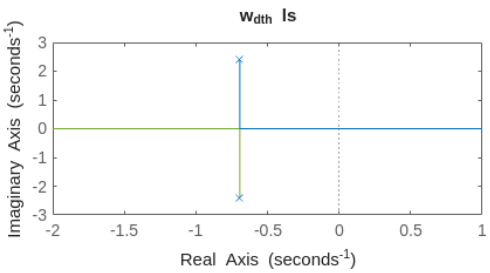

u_{dth} lin
Imaginary Axis (seconds^{-1})
Real Axis (seconds^{-1})

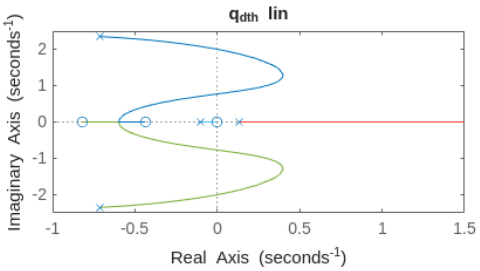

u_{dth} lp
Imaginary Axis (seconds^{-1})
Real Axis (seconds^{-1})

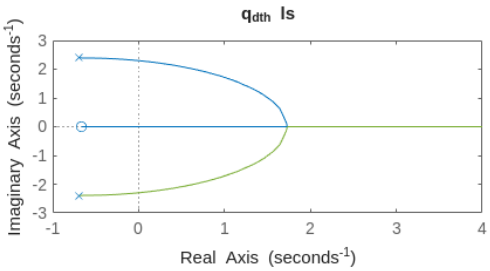

θ_{dth} lin
Imaginary Axis (seconds^{-1})
Real Axis (seconds^{-1})


θ_{dth} lp
Imaginary Axis (seconds^{-1})
Real Axis (seconds^{-1})


w_{dth} lin
Imaginary Axis (seconds^{-1})
Real Axis (seconds^{-1})


w_{dth} ls
Imaginary Axis (seconds^{-1})
Real Axis (seconds^{-1})


q_{dth} lin
Imaginary Axis (seconds^{-1})
Real Axis (seconds^{-1})
2
1
0
-1
-2
-1
-0.5
0
0.5
1
1.5


q_{dth} ls
Imaginary Axis (seconds^{-1})
Real Axis (seconds^{-1})
3
2
1
0
-1
-2
-3
-1
0
1
2
3
4

27.4 Response Comparison for Different Control Inputs

Unlike the frequency domain plots, it didn't appear that the responses of the approximations are going along with the nonlinear model at all, but it is noted that the variables w and q due to short period approximation are not bad; the general behaviour of the response is accepted.

27.4.1 $dE = 1^\circ$

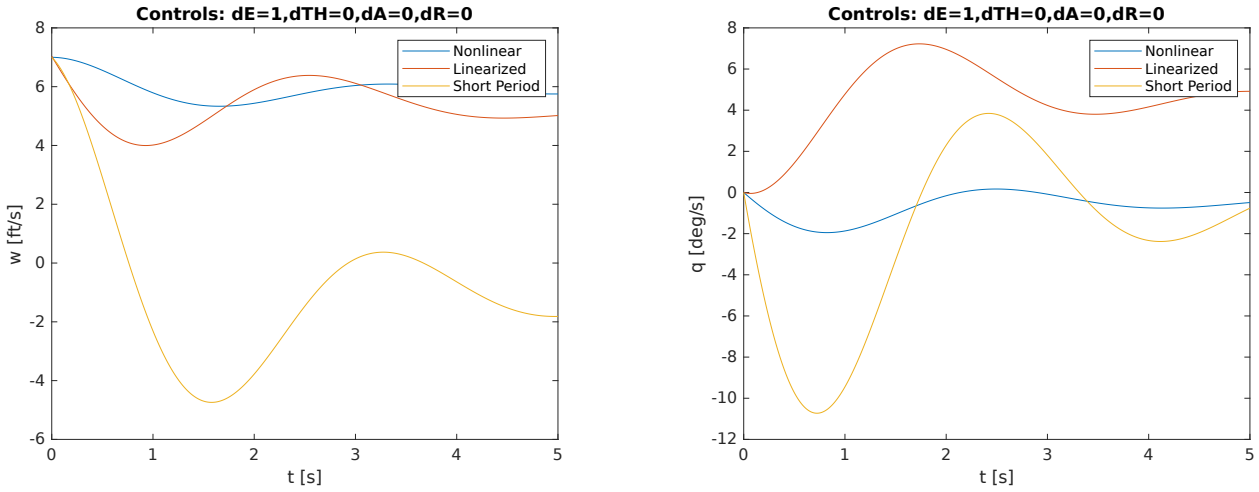


Figure 27

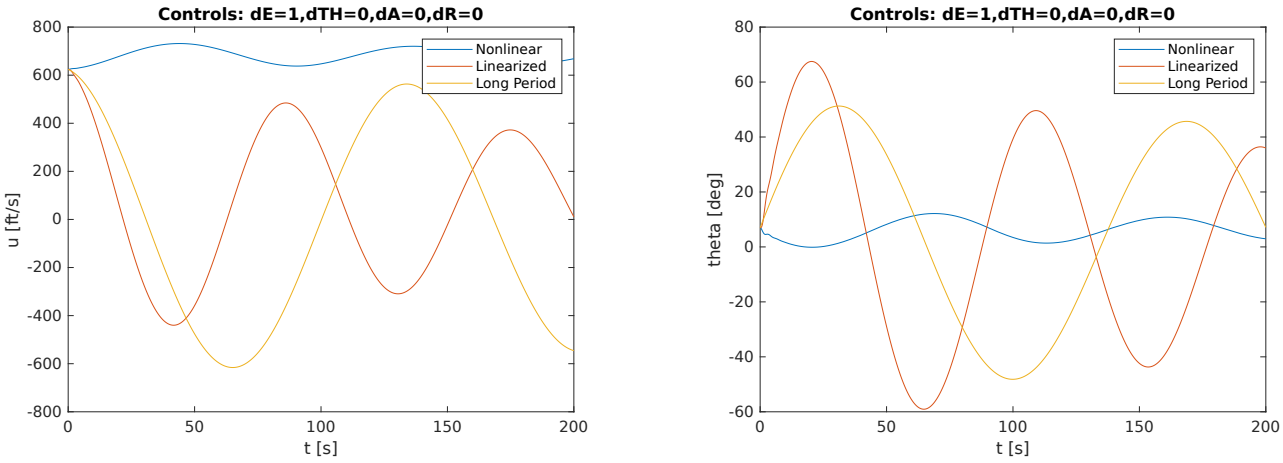
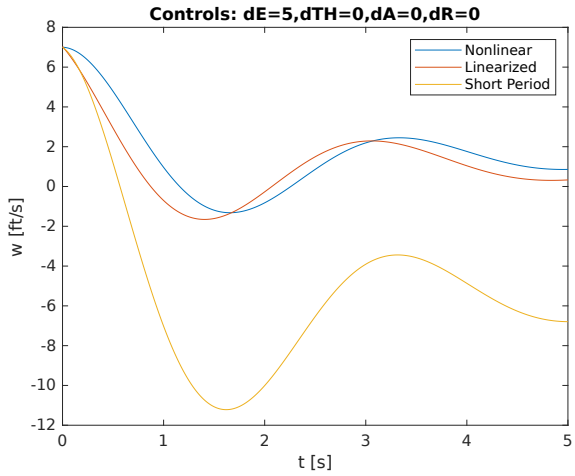


Figure 28

27.4.2 $dE = 5°$



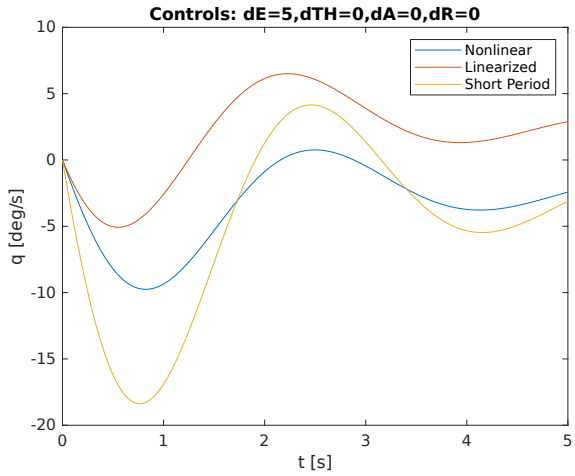
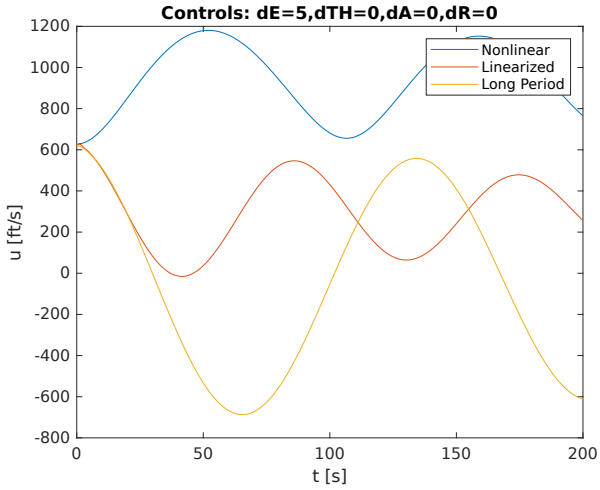


Figure 29



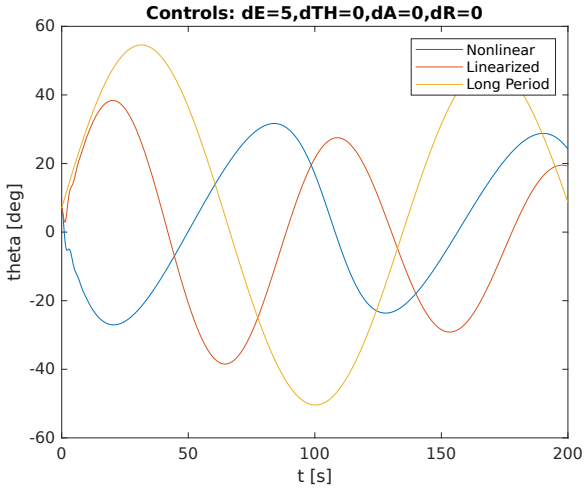
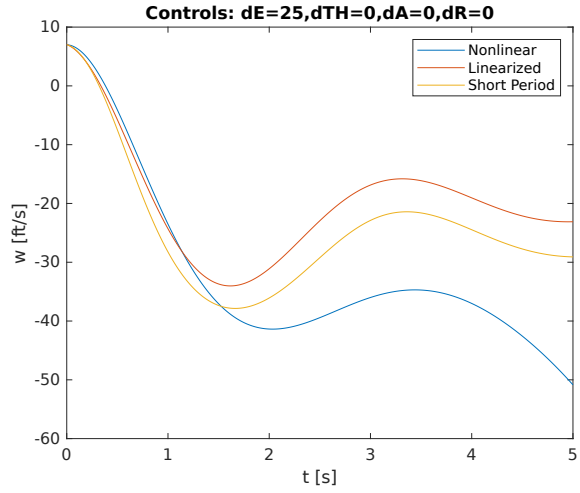


Figure 30

27.4.3 $dE = 25°$



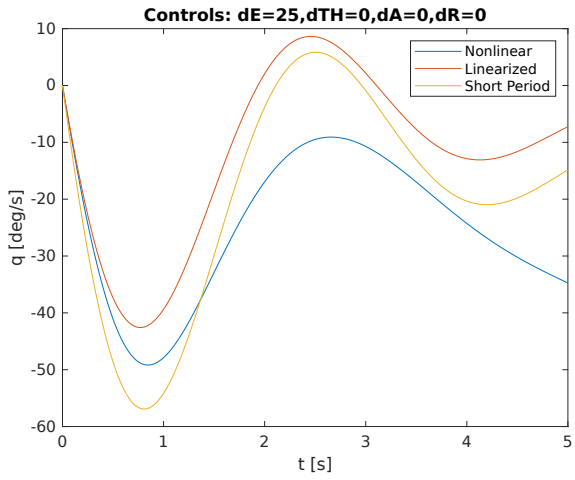
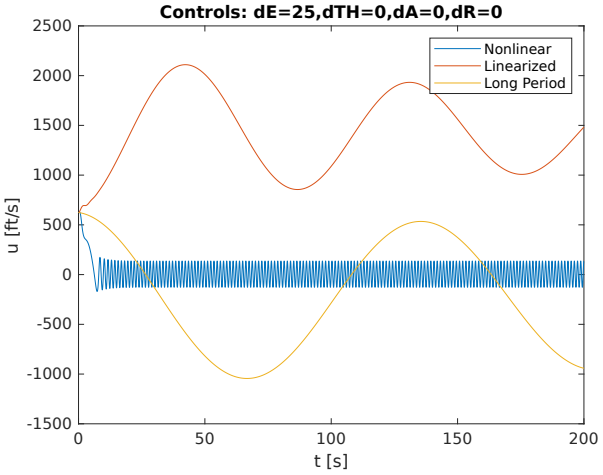


Figure 31



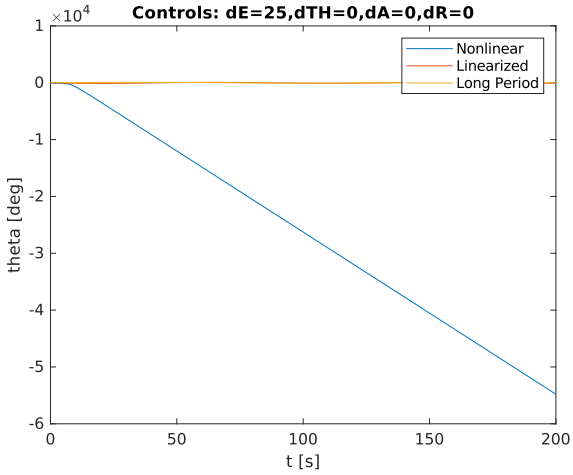
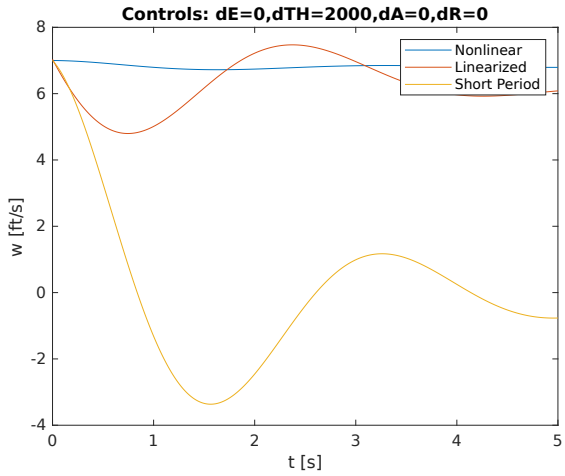


Figure 32

27.4.4 $dTH = 2000$



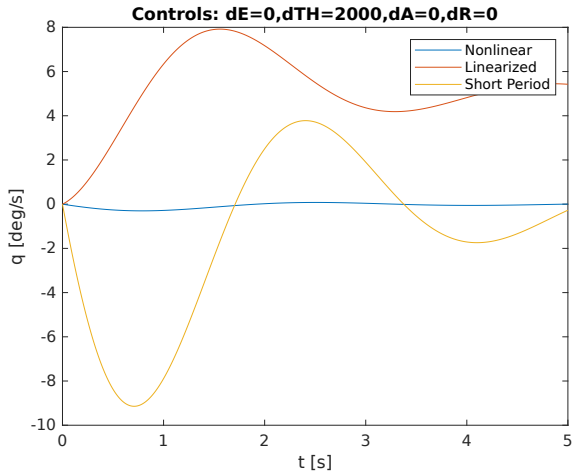
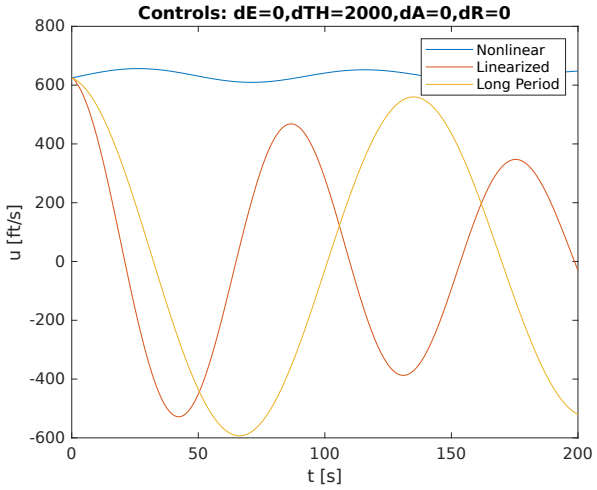


Figure 33



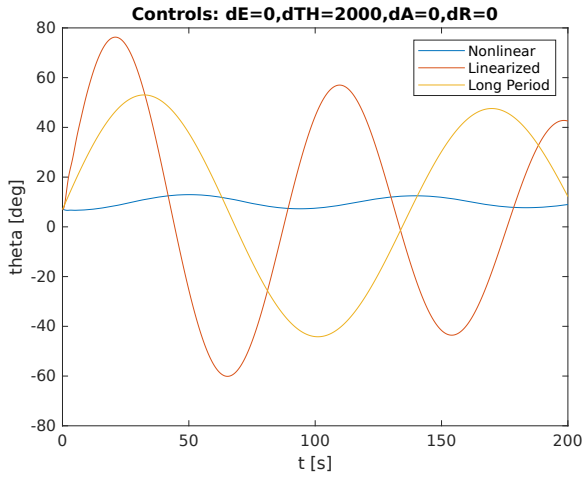


Figure 34

27.4.5 $dTH = 10000$

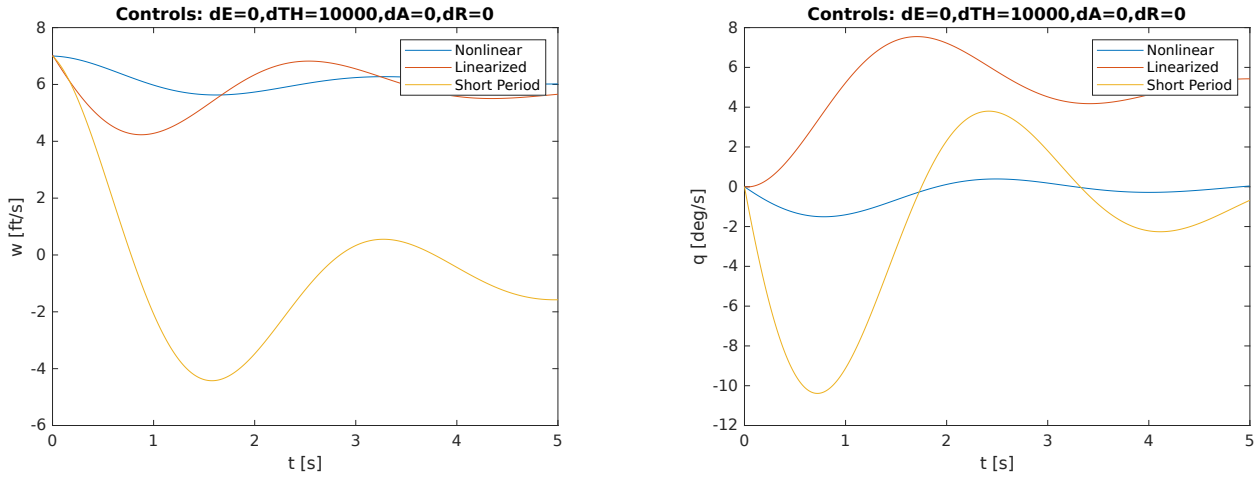


Figure 35

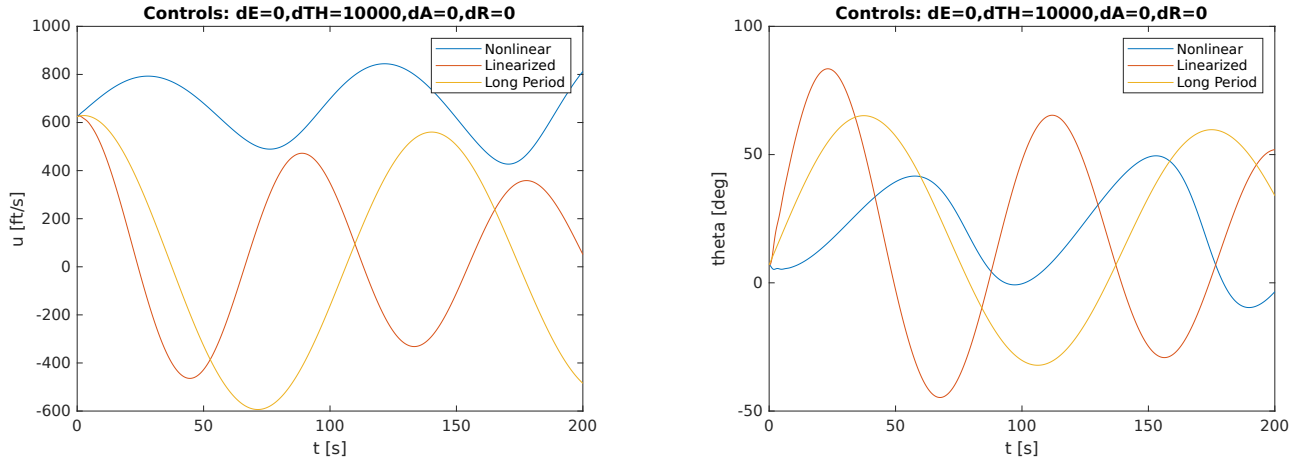


Figure 36

28 Lateral Modes Approximations

28.1 1DOF Approximation (Roll Mode)

$$\dot{p} = L'_p p + L'_{\delta_a} \delta_a$$

28.2 2DOF Approximation (Dutch Roll Mode)

$$\begin{bmatrix} \dot{v} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} Y_v & -u_0 - Y_r \\ N'_v & N'_r \end{bmatrix} \begin{bmatrix} v \\ r \end{bmatrix} + \begin{bmatrix} Y_{\delta_a} & Y_{\delta_r} \\ N'_{\delta_a} & N'_{\delta_r} \end{bmatrix} \begin{bmatrix} \delta_a \\ \delta_r \end{bmatrix}$$

28.3 3DOF Approximation (Dutch Roll Mode)

$$\begin{bmatrix} \dot{v} \\ \dot{p} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} Y_v & 0 & -u_0 - Y_r \\ L'_v & L'_p & 0 \\ N'_v & 0 & N'_r \end{bmatrix} \begin{bmatrix} v \\ p \\ r \end{bmatrix} + \begin{bmatrix} Y_{\delta_a} & Y_{\delta_r} \\ L'_{\delta_a} & L'_{\delta_r} \\ N'_{\delta_a} & N'_{\delta_r} \end{bmatrix} \begin{bmatrix} \delta_a \\ \delta_r \end{bmatrix}$$

28.4 3DOF Approximation (Spiral Mode)

$$\begin{bmatrix} \dot{p} \\ \dot{r} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} L'_p & L'_r & 0 \\ N'_p & N'_r & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} p \\ r \\ \phi \end{bmatrix} + \begin{bmatrix} L_{\delta_r} \\ N'_{\delta_r} \\ 0 \end{bmatrix} \begin{bmatrix} \delta_r \end{bmatrix}$$

28.5 Lateral Modes Frequency Domain

Notice the notation used here:

- "lin" means full linearized model
- "1DOF" means the 1degrees of freedom Rolling mode
- "2DOF" means the 2 degrees of freedom Dutch-Roll mode
- "3DOF_DR" means the 3 degrees of freedom Dutch-Roll mode
- "3DOF" means the 3 degrees of freedom Spiral mode

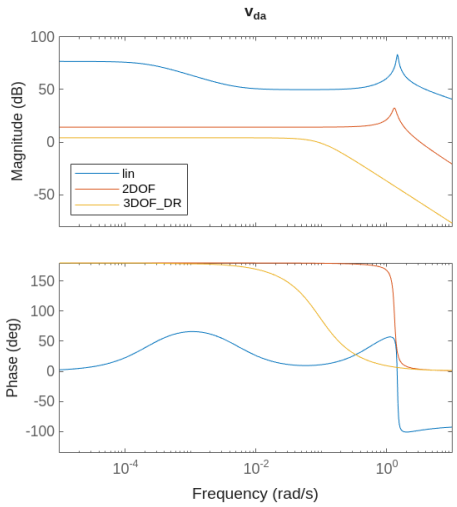
Transfer Functions

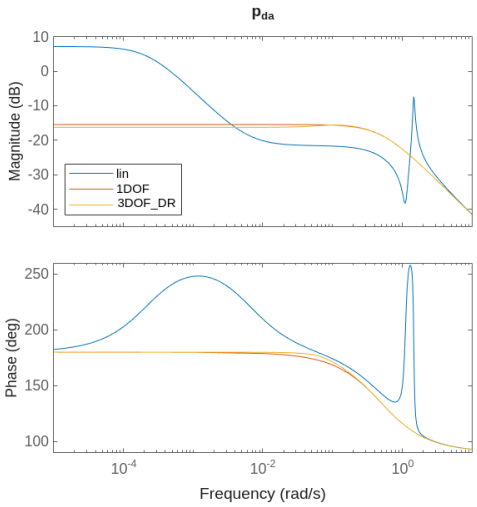
```
v_da =

        1076 s^3 + 1217 s^2 + 374.3 s - 1.908
  ------------------------------------------------
  s^4 + 0.6532 s^3 + 2.187 s^2 + 1.232 s - 0.0002815

Continuous-time transfer function.
Model Properties

p_da =

   -0.0831 s^3 - 0.01205 s^2 - 0.1029 s + 0.0006435
  ------------------------------------------------
  s^4 + 0.6532 s^3 + 2.187 s^2 + 1.232 s - 0.0002815

Continuous-time transfer function.
Model Properties

r_da =

   0.0144 s^3 + 0.01427 s^2 - 0.01175 s - 0.005241
  ------------------------------------------------
  s^4 + 0.6532 s^3 + 2.187 s^2 + 1.232 s - 0.0002815

Continuous-time transfer function.
Model Properties

phi_da =

           -0.08133 s^2 - 0.0103 s - 0.1043
  ------------------------------------------------
  s^4 + 0.6532 s^3 + 2.187 s^2 + 1.232 s - 0.0002815

Continuous-time transfer function.
Model Properties

v_dr =

        1076 s^3 + 1217 s^2 + 374.3 s - 1.908
  ------------------------------------------------
  s^4 + 0.6532 s^3 + 2.187 s^2 + 1.232 s - 0.0002815

Continuous-time transfer function.
Model Properties

p_dr =

      0.766 s^3 - 4.965 s^2 - 1.319 s + 0.006913
  ------------------------------------------------
  s^4 + 0.6532 s^3 + 2.187 s^2 + 1.232 s - 0.0002815

Continuous-time transfer function.
Model Properties

r_dr =

      -0.836 s^3 + 2.471 s^2 + 1.688 s - 0.0563
  ------------------------------------------------
  s^4 + 0.6532 s^3 + 2.187 s^2 + 1.232 s - 0.0002815

Continuous-time transfer function.
Model Properties

phi_dr =

             0.6634 s^2 - 4.662 s - 1.112
  ------------------------------------------------
  s^4 + 0.6532 s^3 + 2.187 s^2 + 1.232 s - 0.0002815
```

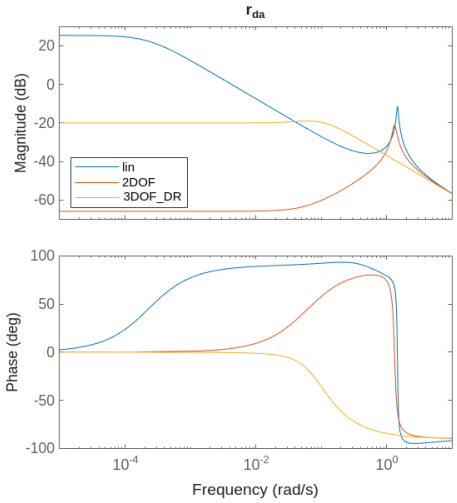
```
Continuous-time transfer function.
Model Properties

v_da_2DOF =

          -8.99
  ----------------------
  s^2 + 0.1612 s + 1.743

Continuous-time transfer function.
Model Properties

r_da_2DOF =

   0.0144 s + 0.0008899
  ----------------------
  s^2 + 0.1612 s + 1.743

Continuous-time transfer function.
Model Properties

v_dr_2DOF =

      1076 s + 628.8
  ----------------------
  s^2 + 0.1612 s + 1.743

Continuous-time transfer function.
Model Properties

r_dr_2DOF =

     -0.836 s + 2.941
  ----------------------
  s^2 + 0.1612 s + 1.743

Continuous-time transfer function.
Model Properties

v_da_3DOF_DR =

           -0.0144
  -------------------------
  s^2 + 0.1612 s + 0.008925

Continuous-time transfer function.
Model Properties

p_da_3DOF_DR =

    -0.0831 s^2 - 0.0134 s - 0.0006746
  ---------------------------------------
  s^3 + 0.6532 s^2 + 0.08824 s + 0.004391

Continuous-time transfer function.
Model Properties

r_da_3DOF_DR =

    0.0144 s + 0.0008899
  -------------------------
  s^2 + 0.1612 s + 0.008925

Continuous-time transfer function.
Model Properties

v_dr_3DOF_DR =

       1076 s + 107.7
  -------------------------
```

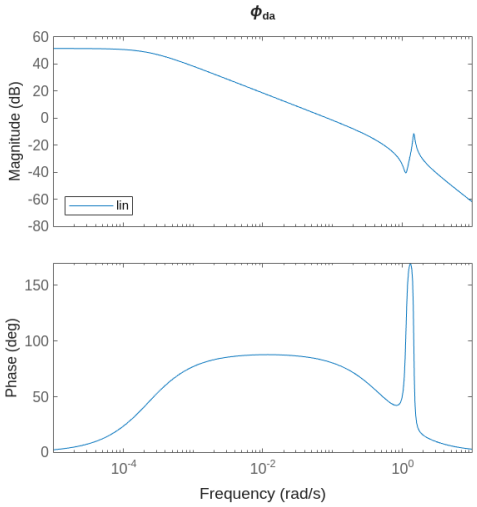
```
  s^2 + 0.1612 s + 0.008925

Continuous-time transfer function.
Model Properties

p_dr_3DOF_DR =

       0.766 s^2 - 4.887 s - 0.4951
  ---------------------------------------
  s^3 + 0.6532 s^2 + 0.08824 s + 0.004391

Continuous-time transfer function.
Model Properties

r_dr_3DOF_DR =

      -0.836 s + 2.941
  -------------------------
  s^2 + 0.1612 s + 0.008925

Continuous-time transfer function.
Model Properties

p_dr_3DOF_SP =

    0.766 s - 0.002109
  ----------------------
  s^2 + 0.5914 s + 0.056

Continuous-time transfer function.
Model Properties

r_dr_3DOF_SP =

    -0.836 s - 0.4694
  ----------------------
  s^2 + 0.5914 s + 0.056

Continuous-time transfer function.
Model Properties

phi_dr_3DOF_SP =

      0.766 s - 0.002109
  -------------------------
  s^3 + 0.5914 s^2 + 0.056 s

Continuous-time transfer function.
Model Properties
```

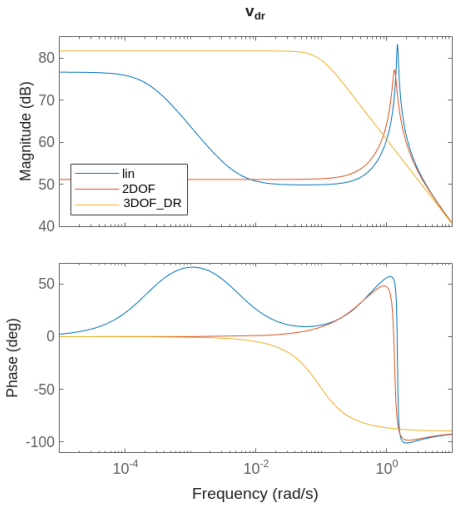
bode Plots

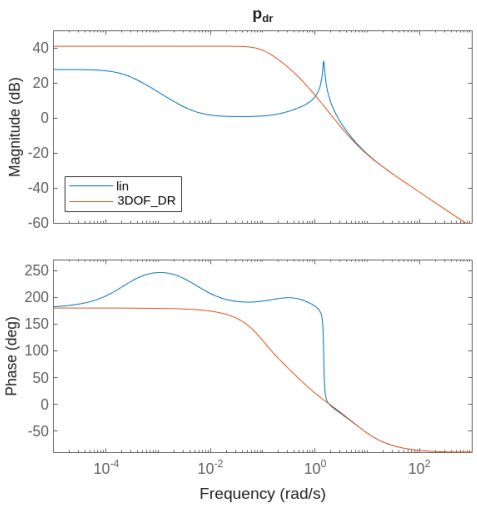

v_{da}
Magnitude (dB)
Phase (deg)
Frequency (rad/s)
lin
2DOF
3DOF_DR

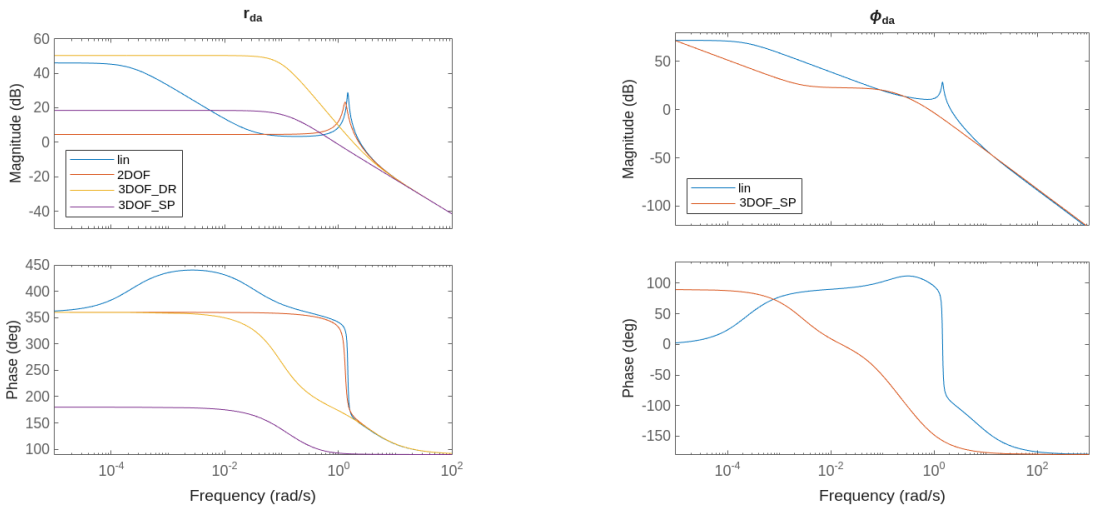

p_{da}
Magnitude (dB)
Phase (deg)
Frequency (rad/s)
lin
1DOF
3DOF_DR


r_{da}
Magnitude (dB)
Phase (deg)
Frequency (rad/s)
lin
2DOF
3DOF_DR

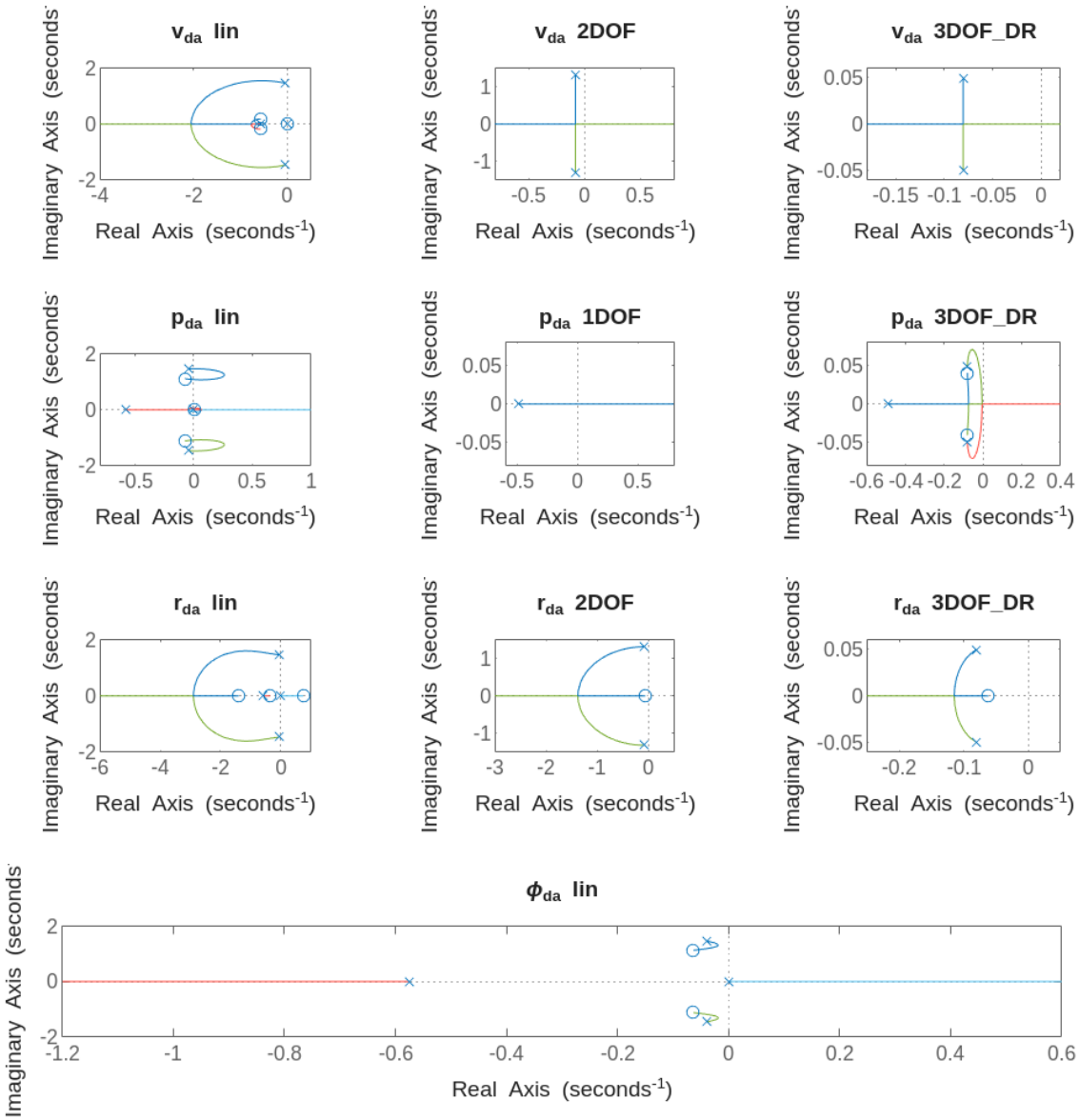

ϕ_{da}
Magnitude (dB)
Phase (deg)
Frequency (rad/s)
lin

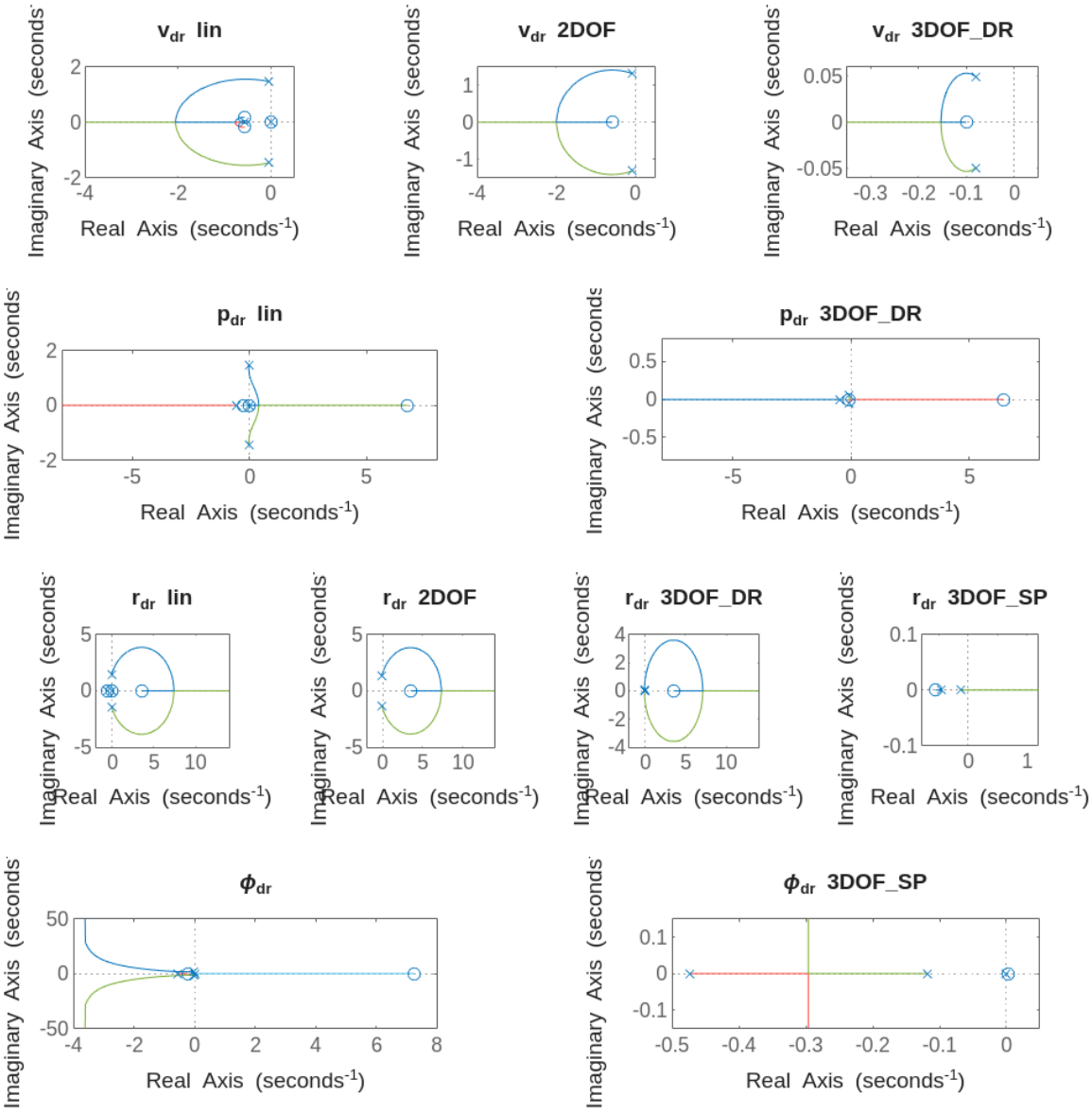

v_{dr}
Magnitude (dB)
Phase (deg)
Frequency (rad/s)
lin
2DOF
3DOF_DR


p_{dr}
Magnitude (dB)
Phase (deg)
Frequency (rad/s)
lin
3DOF_DR



Root Locus Plots

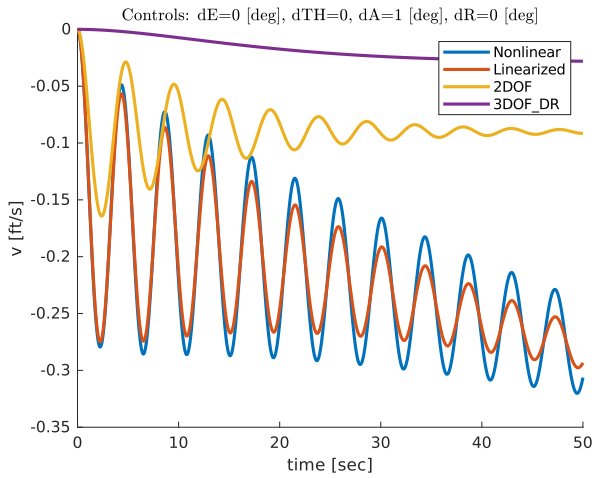



v_{dr} lin
v_{dr} 2DOF
v_{dr} 3DOF_DR
p_{dr} lin
p_{dr} 3DOF_DR
r_{dr} lin
r_{dr} 2DOF
r_{dr} 3DOF_DR
r_{dr} 3DOF_SP
ϕ_{dr}
ϕ_{dr} 3DOF_SP
Imaginary Axis (seconds^{-1})
Real Axis (seconds^{-1})

28.6 Response Comparison for Different Control Inputs

Out of all the responses, it seems that the 2DOF Dutch-Roll mode approximation is very good in most cases, at least better than all other approximations.

28.6.1 $dA = 1°$



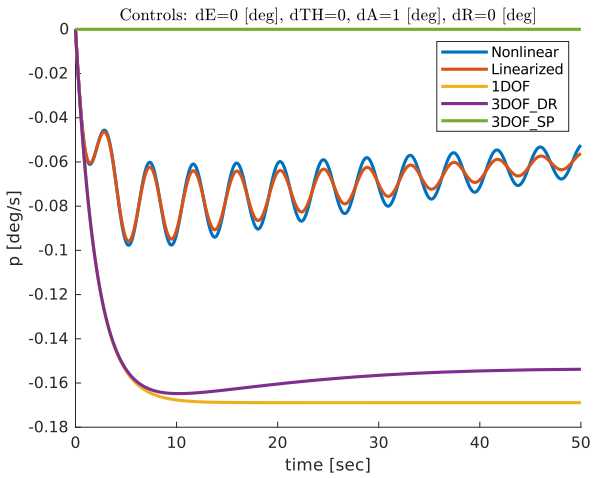
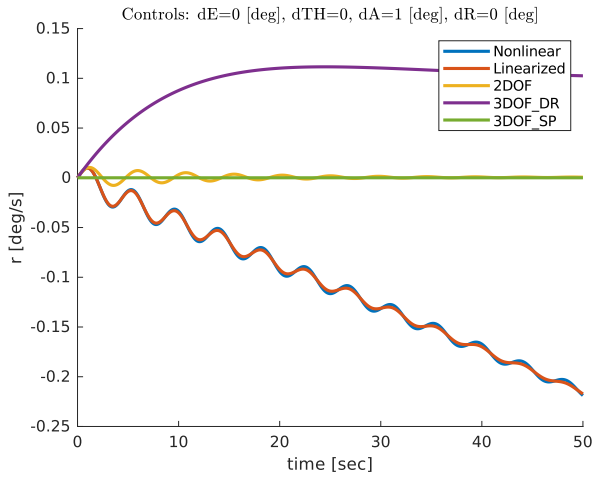


Figure 37



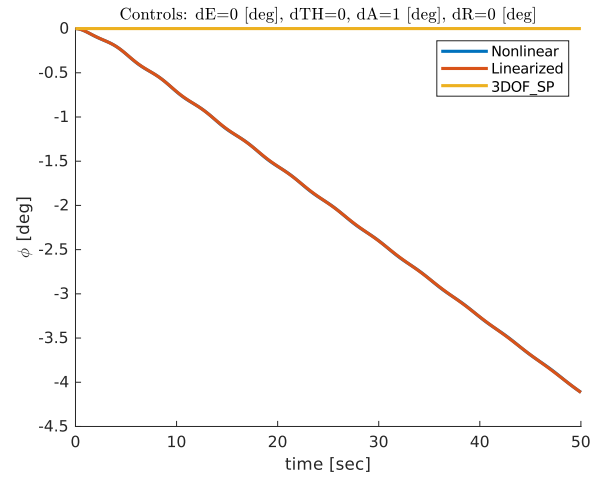


Figure 38

28.6.2 $dA = 5°$

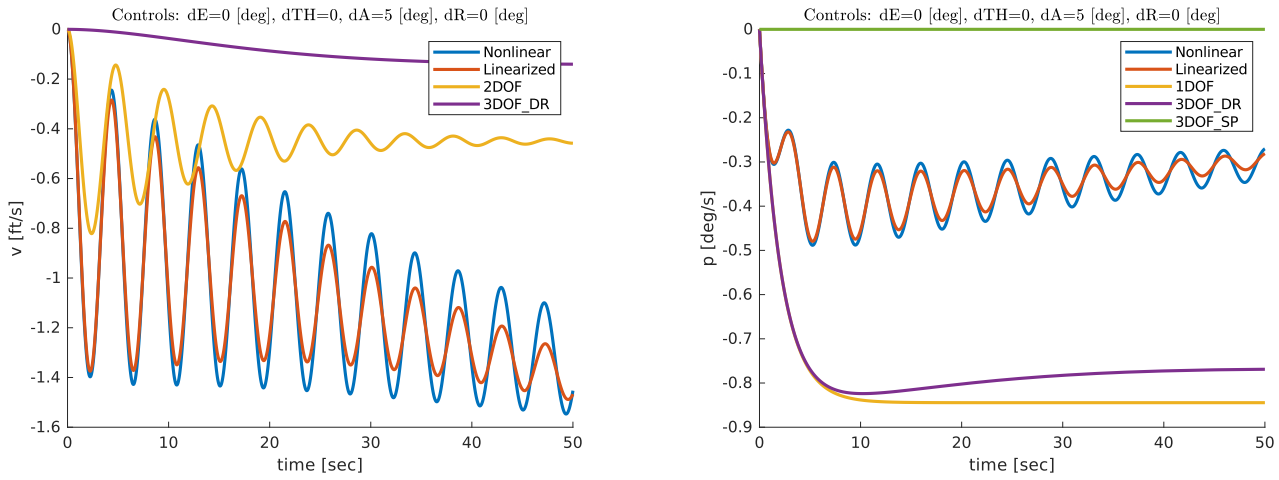


Figure 39

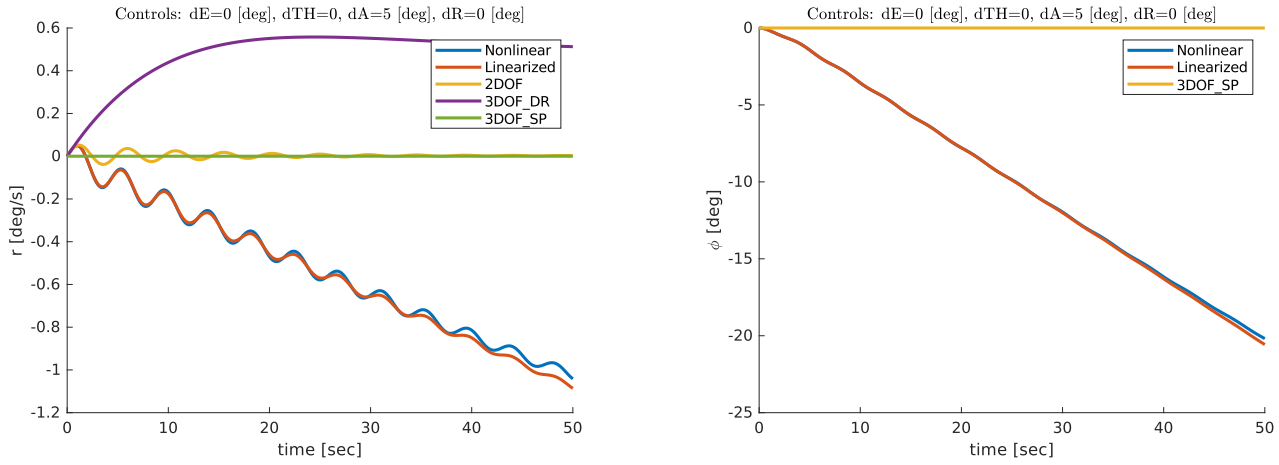


Figure 40

28.6.3 $dA = 10°$

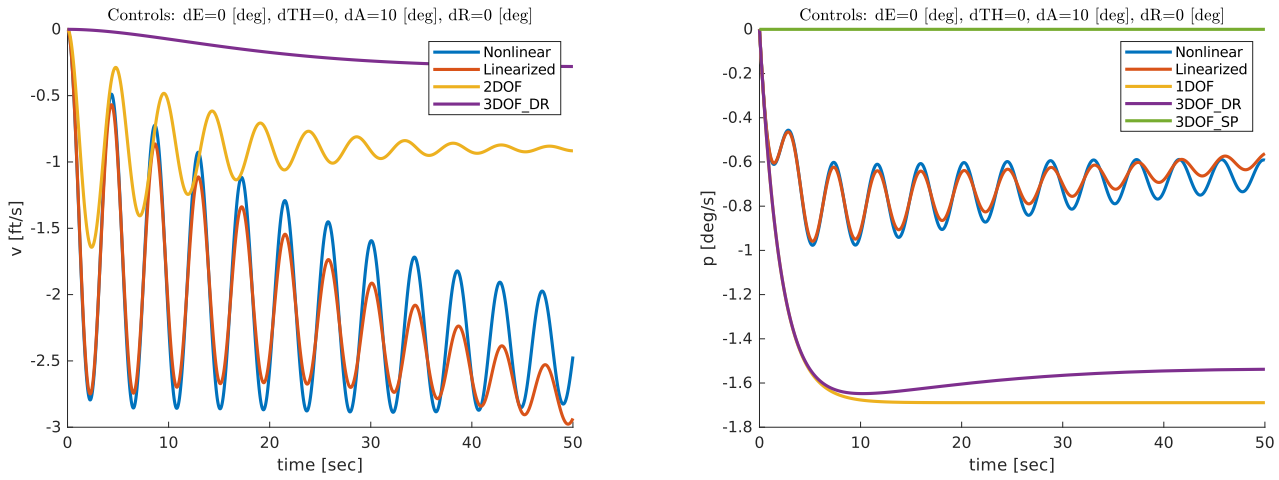


Figure 41

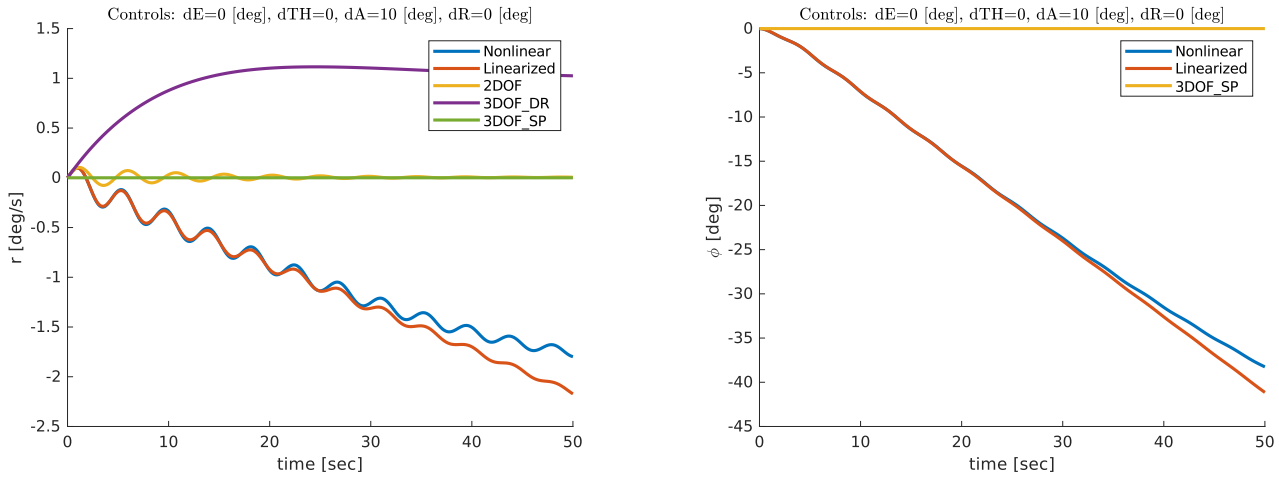
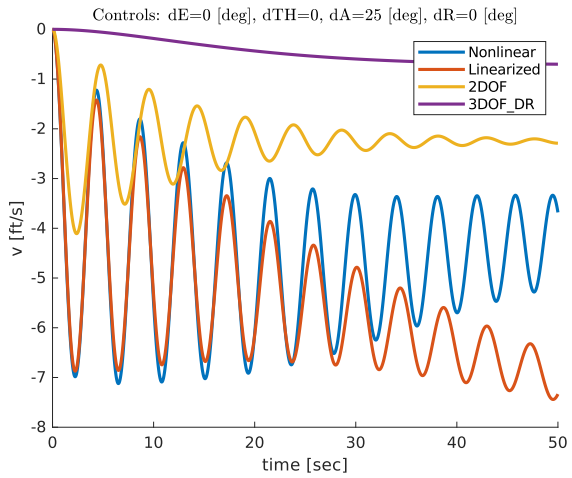


Figure 42

28.6.4 $dA = 25°$



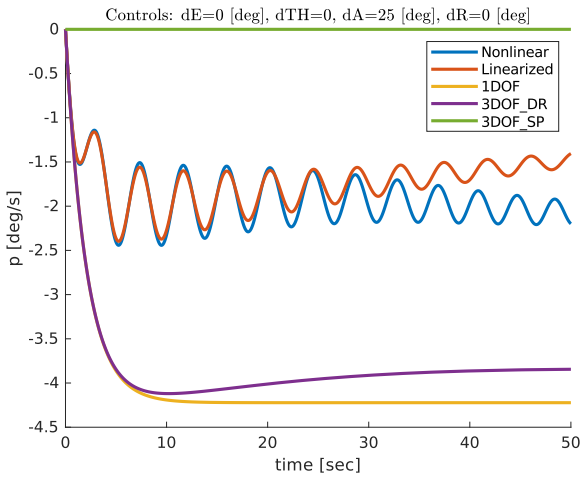
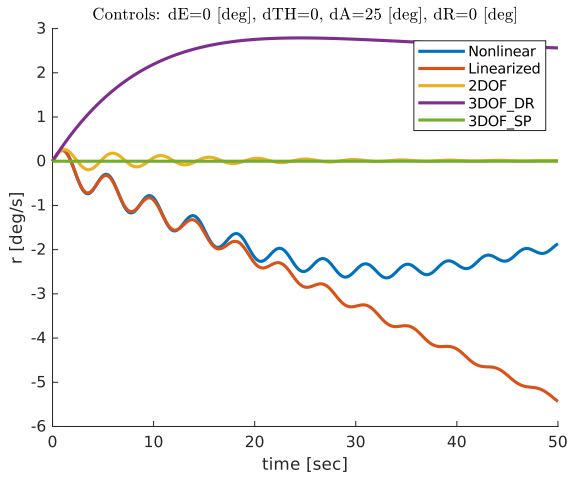


Figure 43



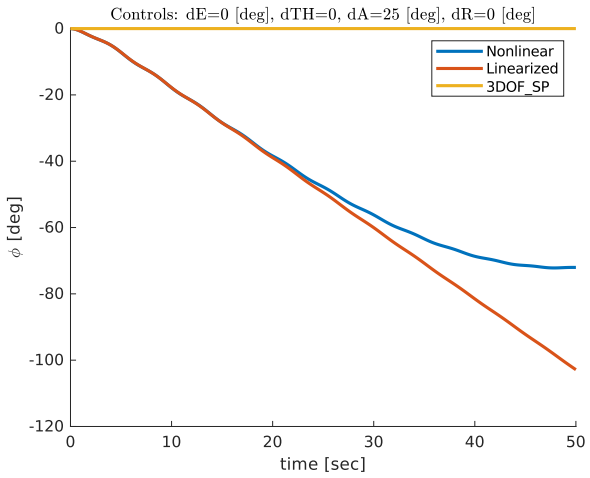


Figure 44

28.6.5 $dR = 1^\circ$

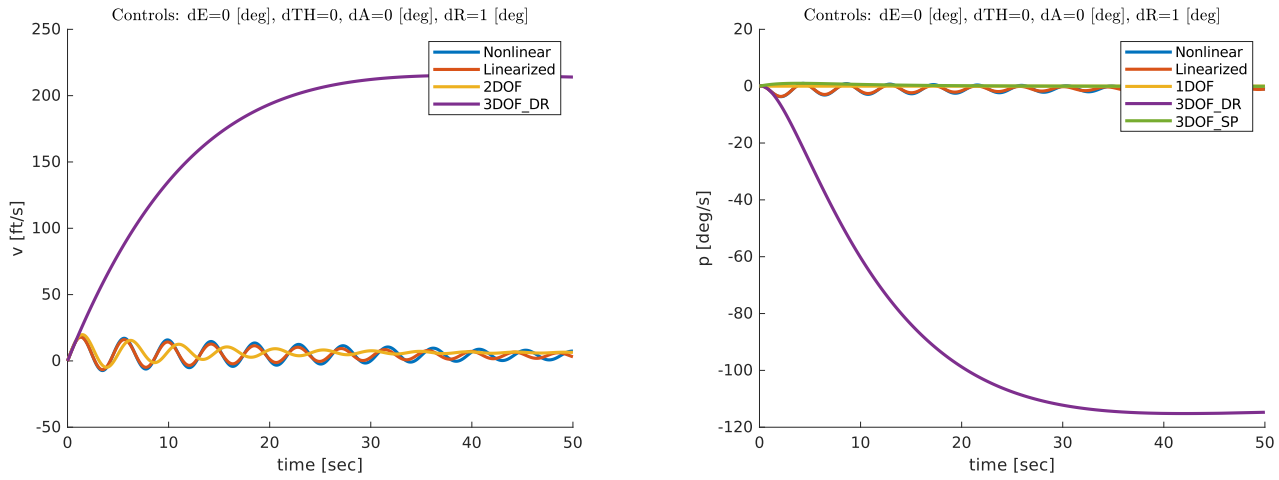


Figure 45

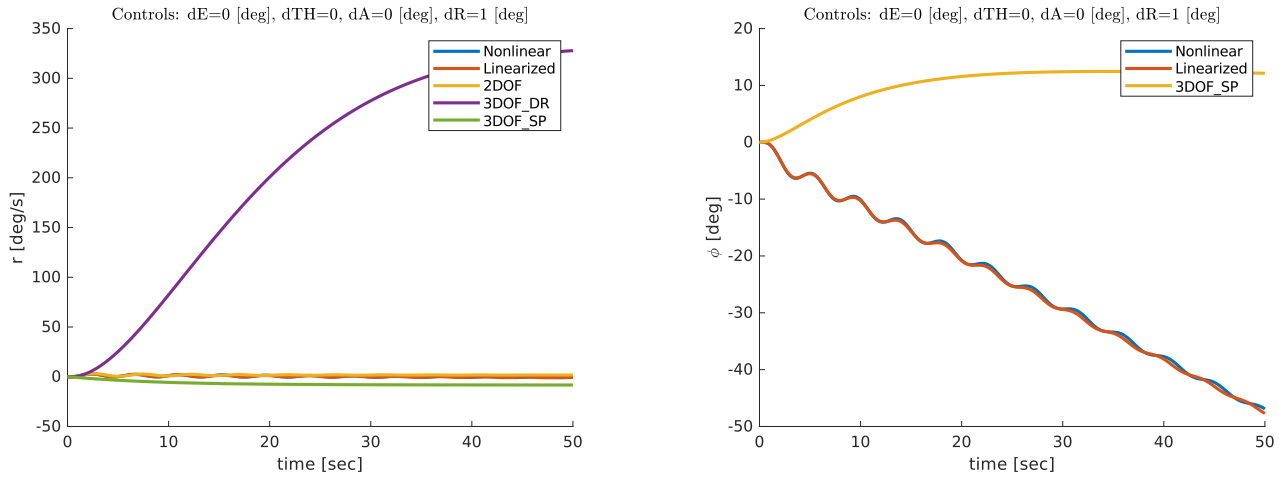
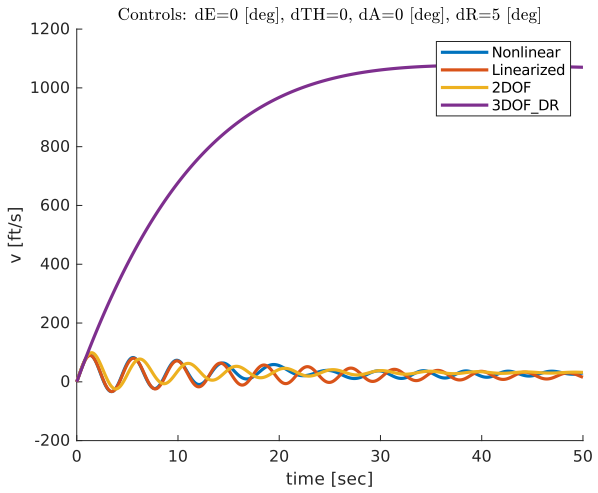


Figure 46

28.6.6 $dR = 5°$



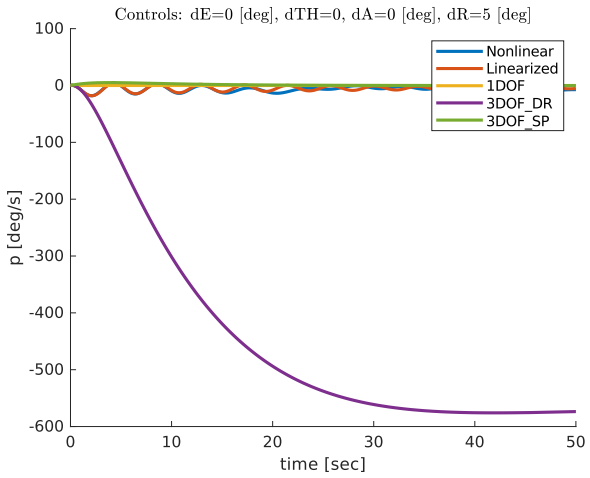
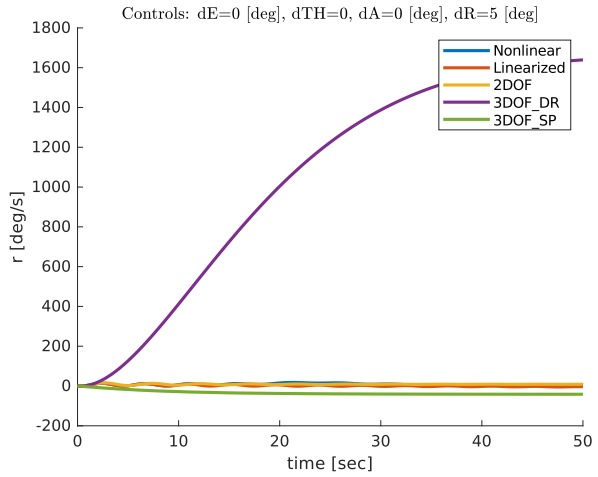


Figure 47



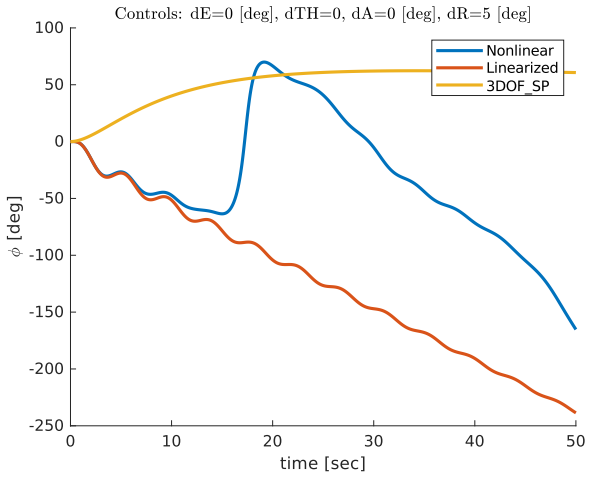
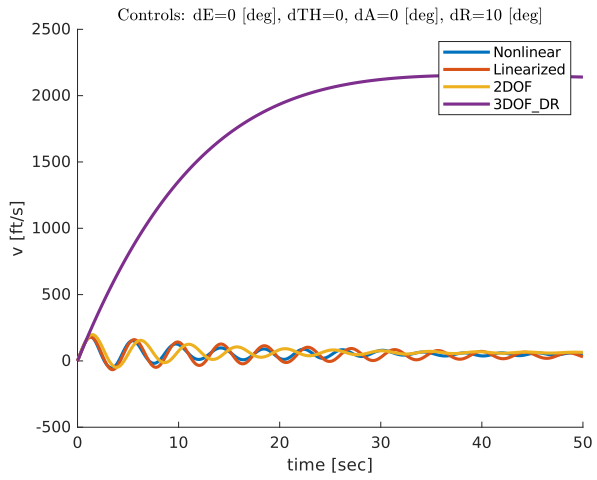


Figure 48

28.6.7 $dR = 10°$



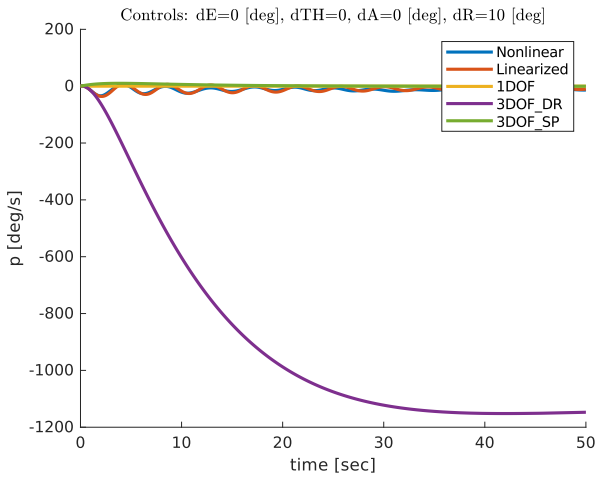
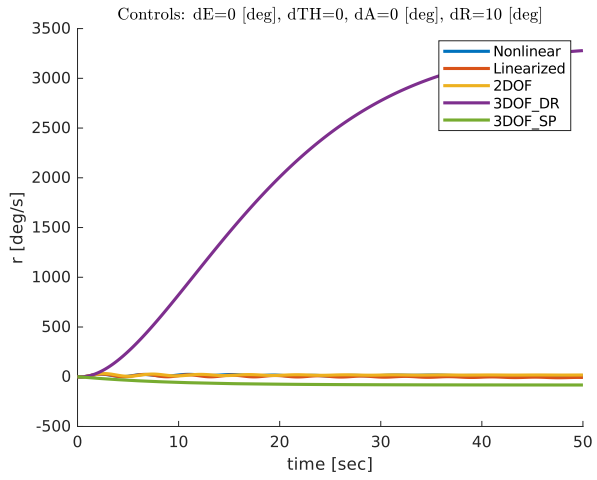


Figure 49



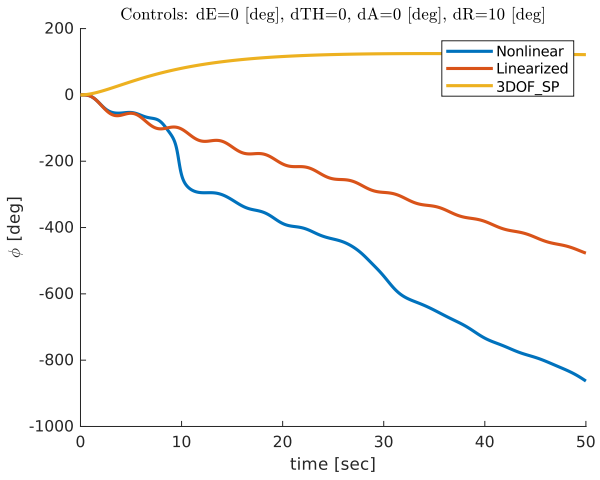
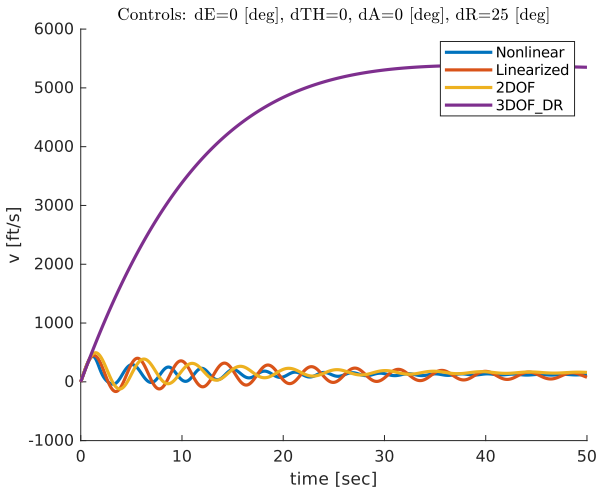


Figure 50

28.6.8 $dR = 25°$



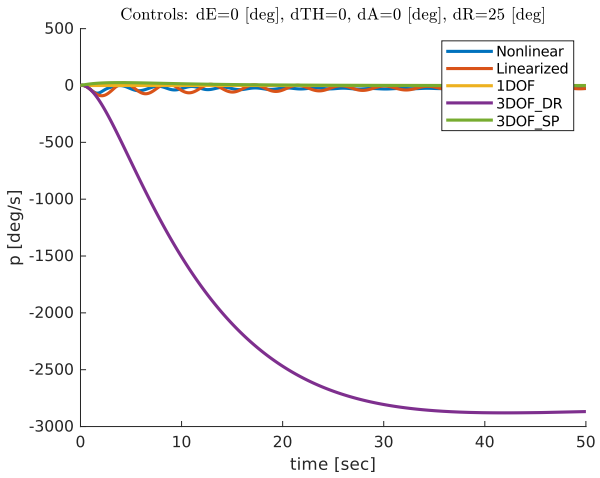
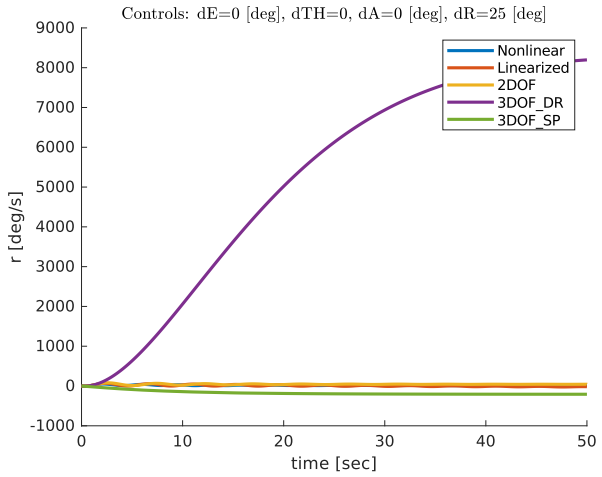


Figure 51



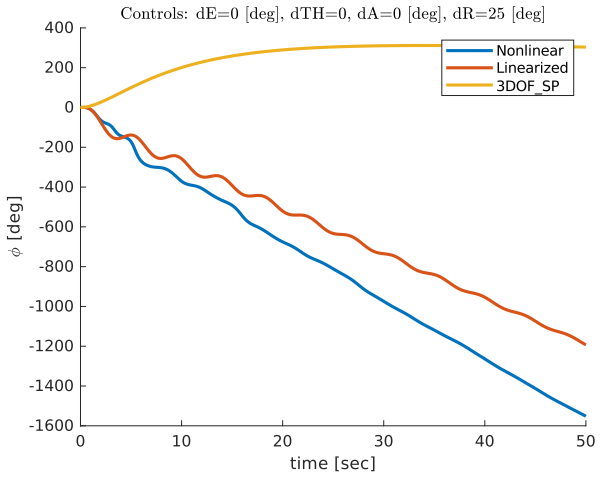


Figure 52

Part VI

Longitudinal Autopilot

Among the many configurations that can be used for the longitudinal autopilot, it is very often to control the elevator deflection through the pitch angle command (**Elevator from Pitch** as an inner control loop), and an outer loop **Pitch from Altitude** loop. Both of them are used along with a simultaneous airspeed control loop, we used **Throttle from Airspeed**.

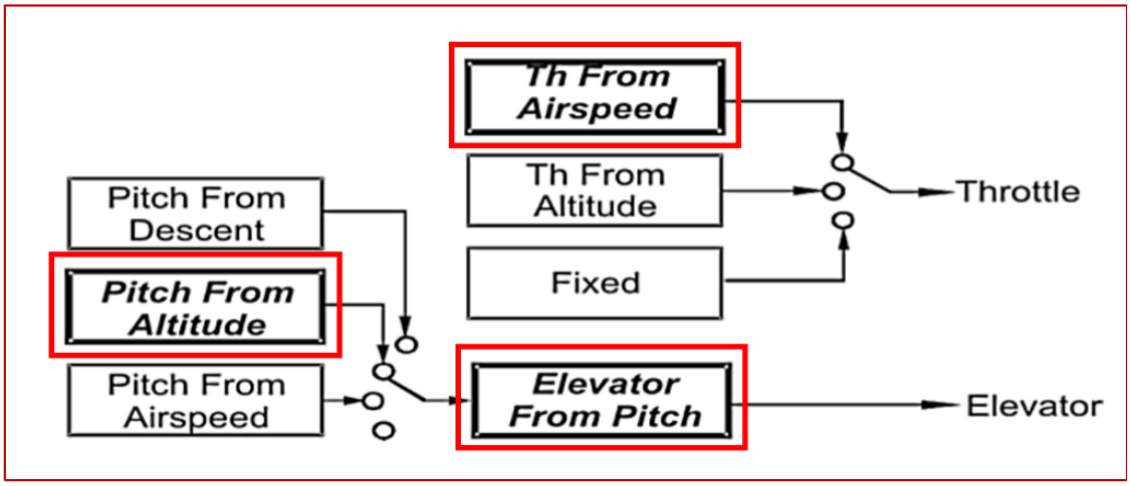


Figure 53: Autopilot Diagram

29 Elevator from Pitch

We used SISOTOOL to tune the PID controller parameters through the root locus of the transfer function $\frac{\theta}{\delta_e}$ multiplied by the servo transfer function $\frac{10}{s+10}$, wee obtained a very good response and characteristics for the closed loop system, a settling time $T_s = 10.4\,sec$, $\%O = 12.4\%$ and zero steady state error for step input. The controller was then implemented on the full linearized model as well as the nonlinear model, and results were checked to be more than good.

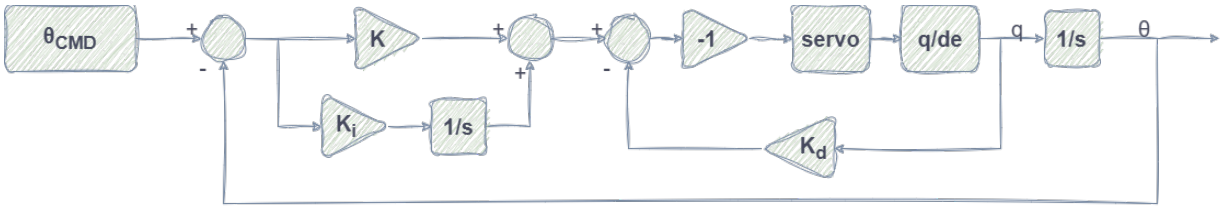


Figure 54: Control Loop to calculate elevator command

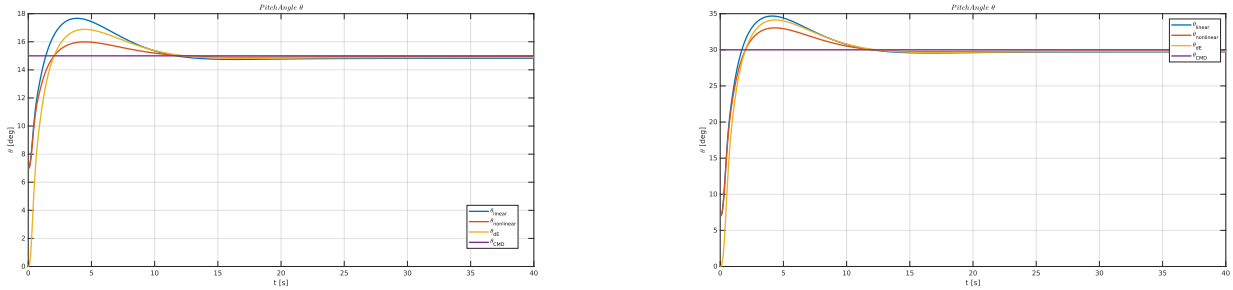


Figure 55: Response at $\theta_{CMD} = 15°$ on the left and $\theta_{CMD} = 30°$ on the right

30 Pitch from Altitude

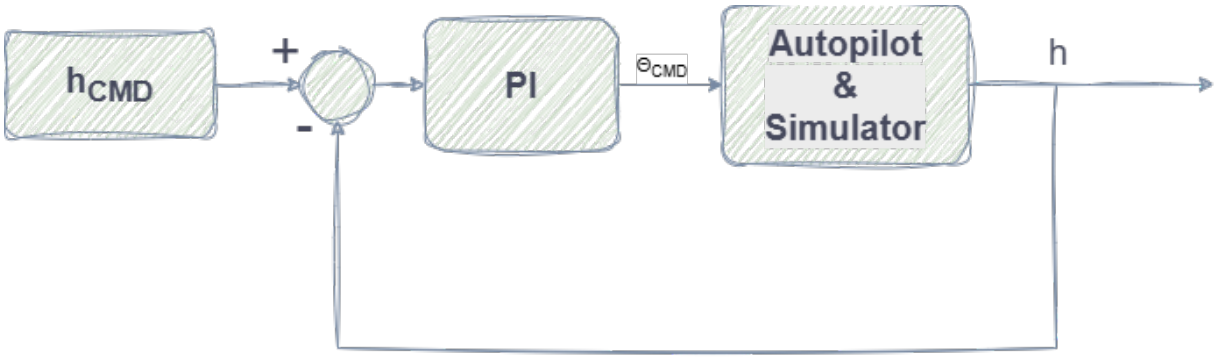


Figure 56: Control Loop to calculate pitch command

31 Throttle from Airspeed

First, to calculate throttle at idle condition, we used this data from the NACA report:

$\rho = 5.87 - 4\, slugs/ft^3$, $CD_\alpha = 0.7\ rad^{-1}$, $CD = CD_\alpha * \alpha_0 = 0.0855$, $S = 542.5\, ft^2$

So idle thrust is: $Thrust_0 = \frac{1}{2}\rho\, V_{to}^2\, S\, CD = 5.3874e{+}03\ lbf$, while the maximum thrust is $Thrust_{max} = 14800\, lbf$; $3700\, lbf$ for each one of four engines.

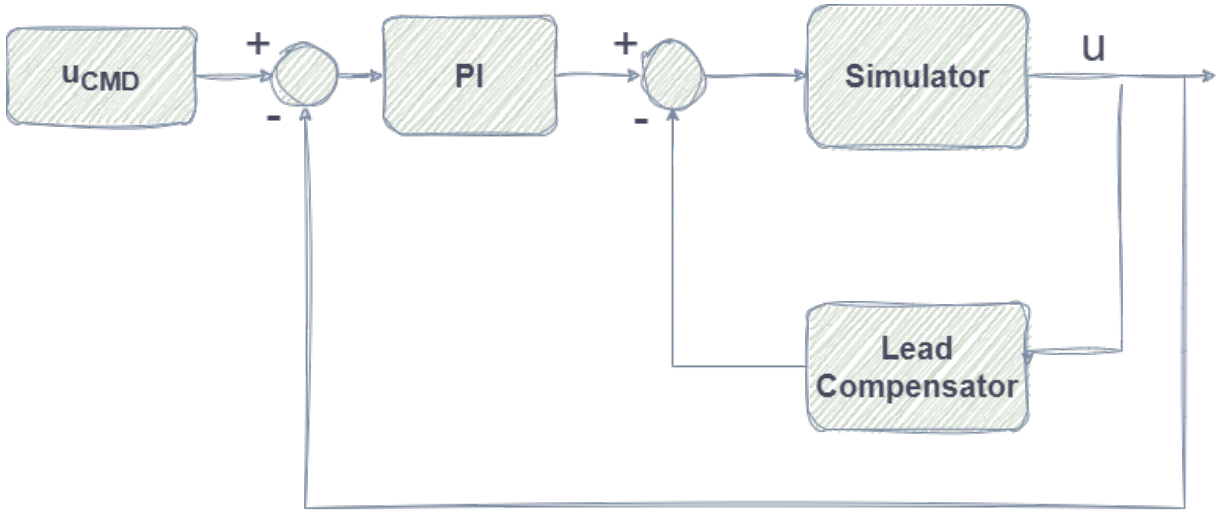


Figure 57: Control Loop to calculate throttle command

Part VII

Lateral Autopilot

The lateral autopilot is responsible for controlling the aircraft's motion within the lateral-directional plane. Specifically, it regulates the rudder and ailerons to achieve a coordinated turn. When integrated with the longitudinal autopilot, the system enables the execution of a coordinated level turn. For the design of the controllers in a MIMO (Multi-Input Multi-Output) framework, we adopt a methodology based on our prior studies of LTI (Linear Time-Invariant) SISO (Single-Input Single-Output) systems. Utilizing classical control techniques such as linear PID controllers and compensators, we structure the control loops through a design strategy known as successive loop closure.

During turning, airplanes might have a different heading than the direction tangent to the turning trajectory, this can cause acceleration in the lateral direction, which not only causes discomfort for passengers,

but a skidding airplane might also cause high loss of lift on one wing, which causes loss of altitude, and the airplane goes into a bad and dangerous spiral mode.

This necessitates the need for turn coordination, this is done using the rudder control action, which controls the coordination using a Yaw–Damper autopilot, and of course, the ailerons control the roll rate of the airplane.

32 Yaw Damper

We need a controller in the feedback signal of the yaw rate, to act as a high pass filter, this is to eliminate the DC gain of the signal of the yaw rate, and only calculate the error due to the variations in the yaw rate, which should preferably be completely zero.

32.1 Cooper-Harper Rating and Flying Qualities

According to the Cooper-Harper rating scale and flying qualities criteria, our airplane falls under Category B, as it is a conventional aircraft. For the Dutch Roll mode, the following requirements must be satisfied: $\zeta > 0.08s$, $\omega_d > 0.4$, $\zeta\omega_d > 0.15$. Physical Limits of Actuators (Saturation) The physical limit for the rudder actuator should be defined as: Delta max = $\pm 15°$.

32.2 Implementation

We chose the transfer function: $HPF = \frac{0.091937s}{(s+0.23)}$, then the feedback signal is positively fed to the commanded yaw rate (which, as said, should be zero), and the output is then the control action δ_r.

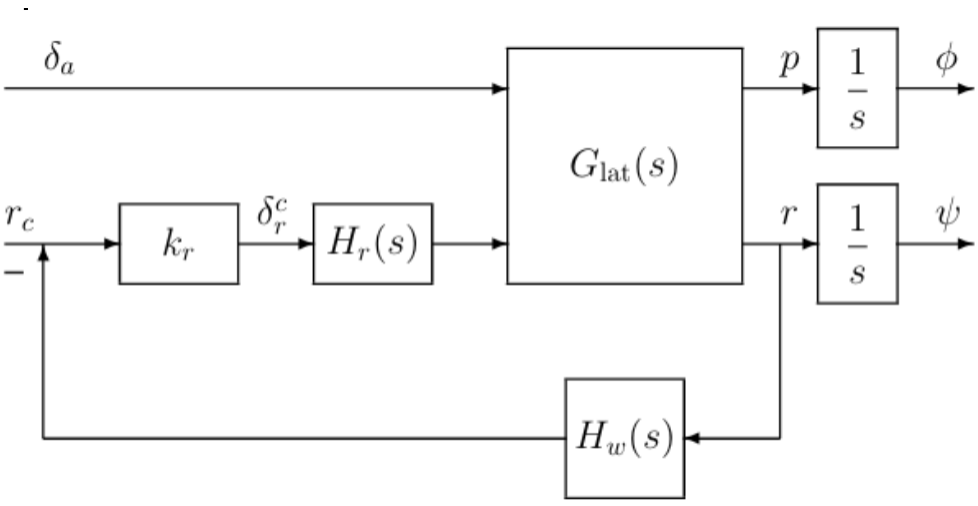


Figure 58: Yaw Damper Loop Architecture

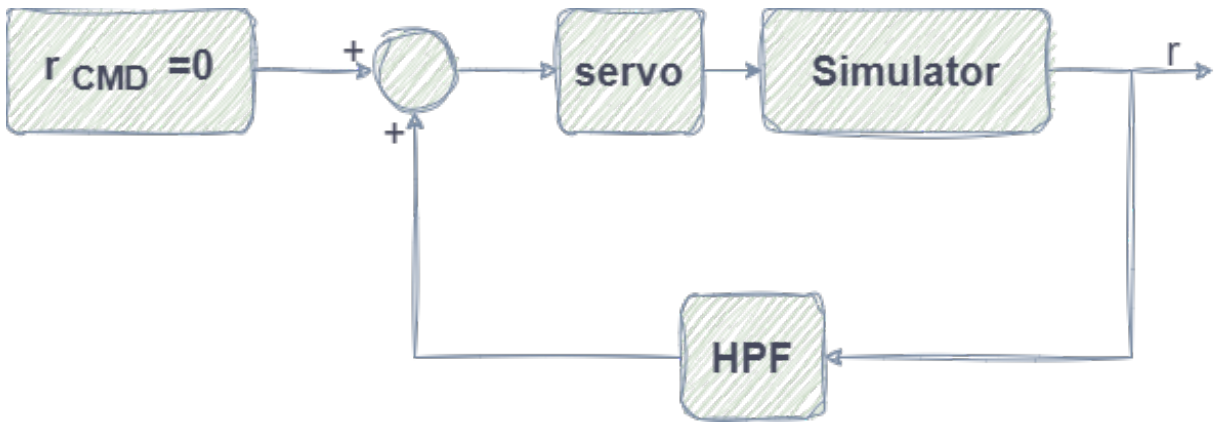


Figure 59: Control Loop to calculate rudder command

32.3 Testing Yaw Damper Controller

Throughout the previous design process, we focused on a single transfer function that relates one specific state: the yaw rate, with the rudder as the input. To validate the performance of our designed controller within a more realistic setting, a simulation was conducted using the full MIMO (Multi-Input Multi-Output) system represented by the lateral state-space model of the aircraft. The result of the test is very unusual:

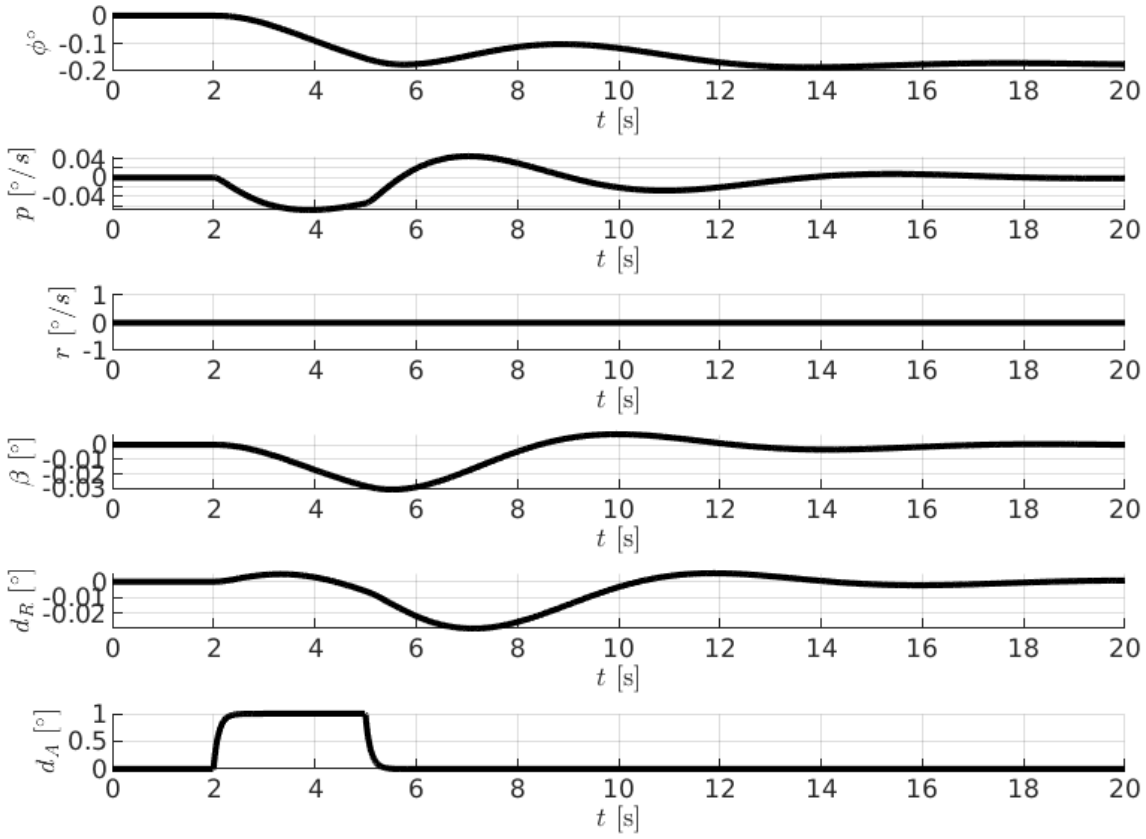


Figure 60: Yaw Damper Test

32.4 Conclusion

- The Closed Loop response is acceptable compared to SISO; the difference is because of the effect of other states.
- The control action is acceptable also compared to which from SISO and guarantee the range of rudder. $\left(\delta_{r,max} = 14^\circ\right)$

33 Roll Control Autopilot

After we have implemented the yaw damper transfer function, we now needed to re-linearize the aircraft system, taking into account the coordinated dynamics, then we only needed a proportional gain for the

design of the roll autopilot, but it was needed to operate on the complementary root locus.

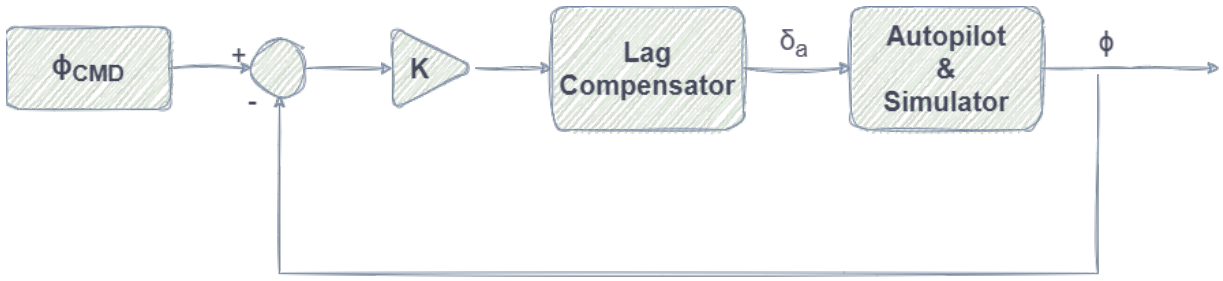


Figure 61: Control Loop to calculate aileron command

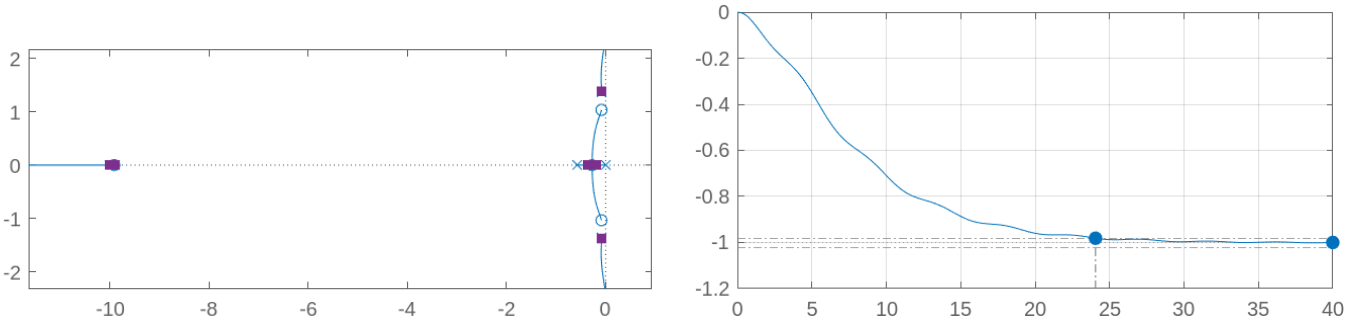


Figure 62: Design of the roll controller, on the left is the complementary root locus plot of the open loop system, on the right is the response of the whole new transfer function to a unit step input.

Here is the response to a commanded roll angle of 20° after implementing the controller:

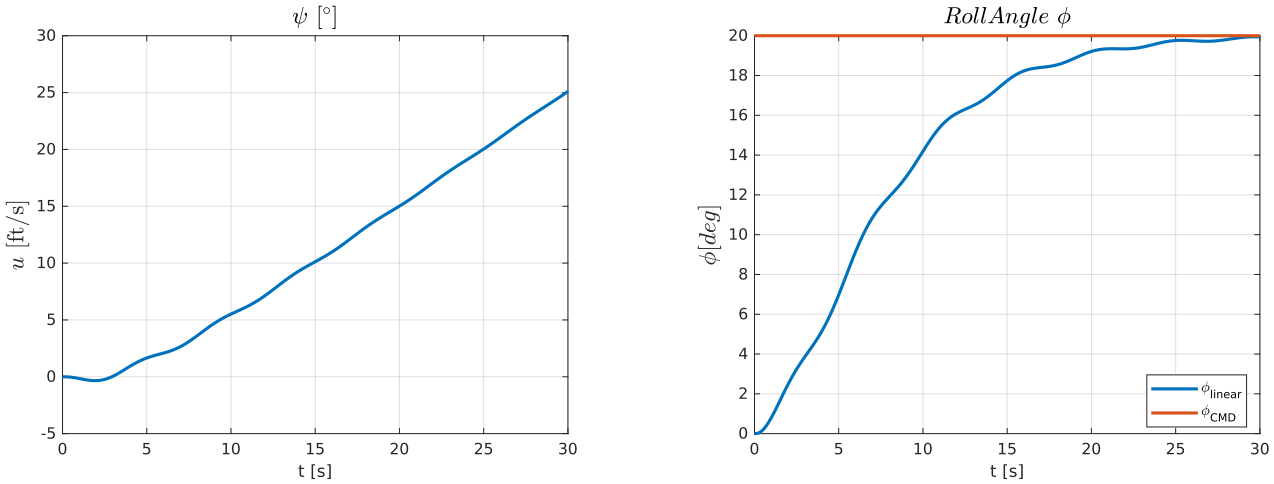


Figure 63: Response to 20° roll angle

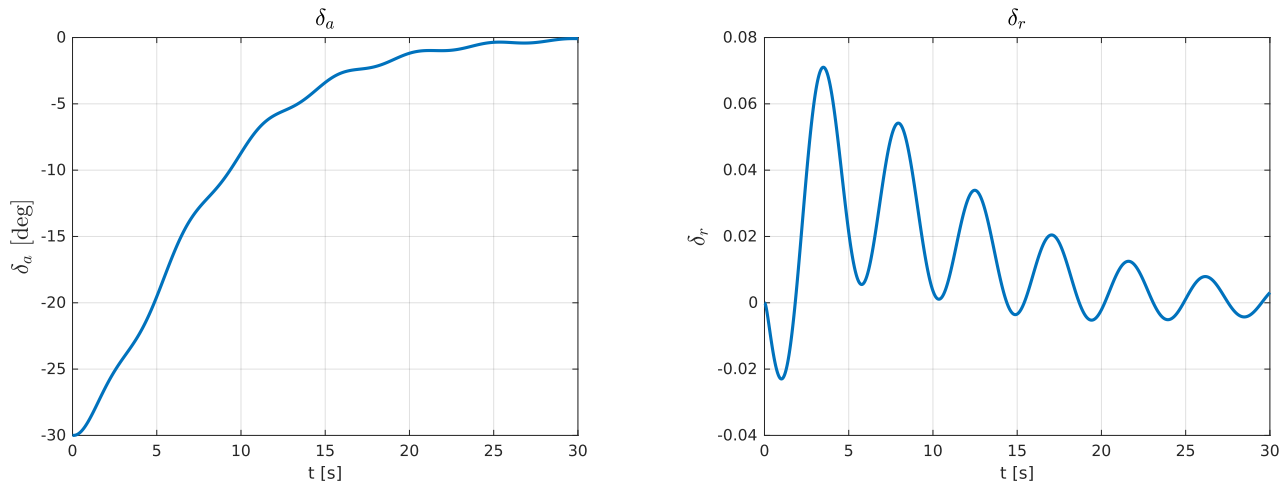


Figure 64: Response to 20° roll angle

What we notice now is that the sideslip angle is not yet zero:

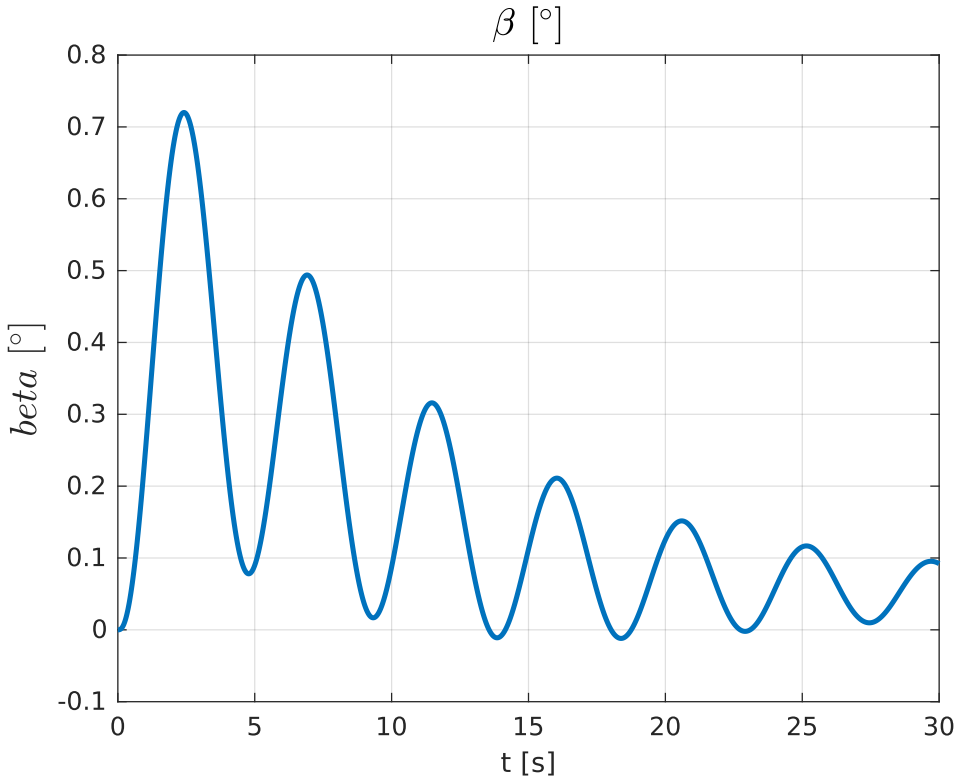


Figure 65: Response to 20° roll angle

This is because the commanded roll angle isn't calculated from the heading rate, but was rather chosen arbitrarily, but for every heading rate, there's an optimum roll angle, which is calculated using the Roll from Heading Autopilot coming next.

34 Roll from Heading Autopilot

This is modelled as a simple lag as follows:

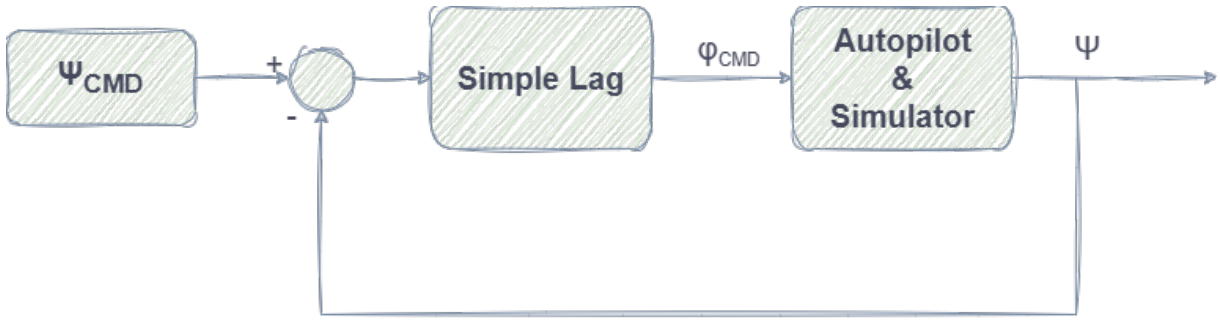


Figure 66: Control Loop to calculate roll angle command

Part VIII

Testing the Autopilot on Nonlinear Simulator

35 Elevator from Pitch Autopilot

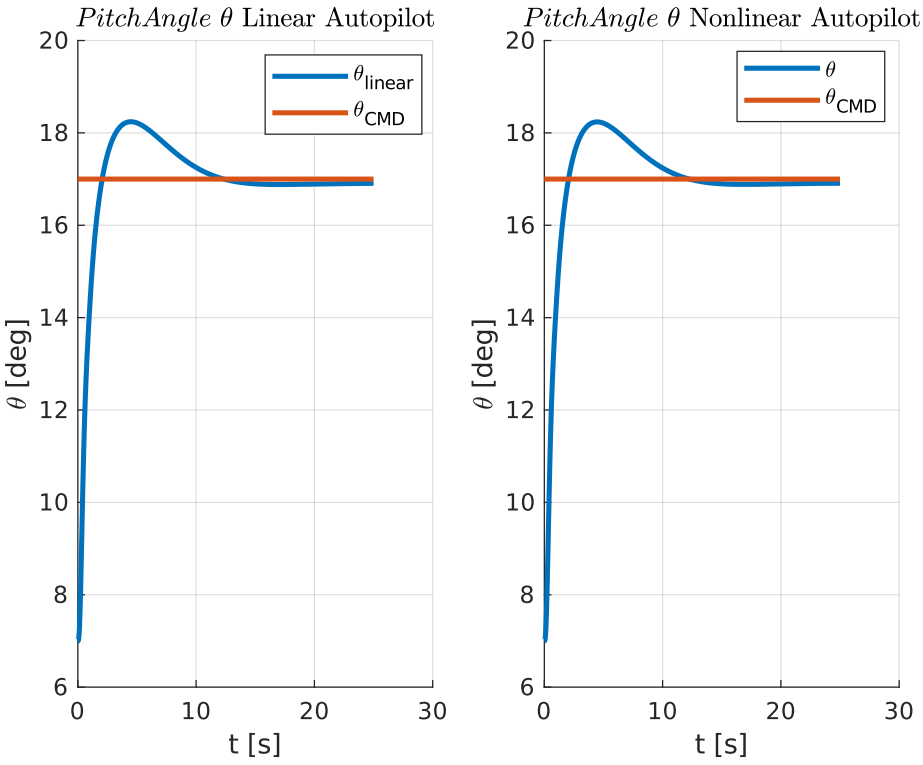


Figure 67: Response to pitch command 10°

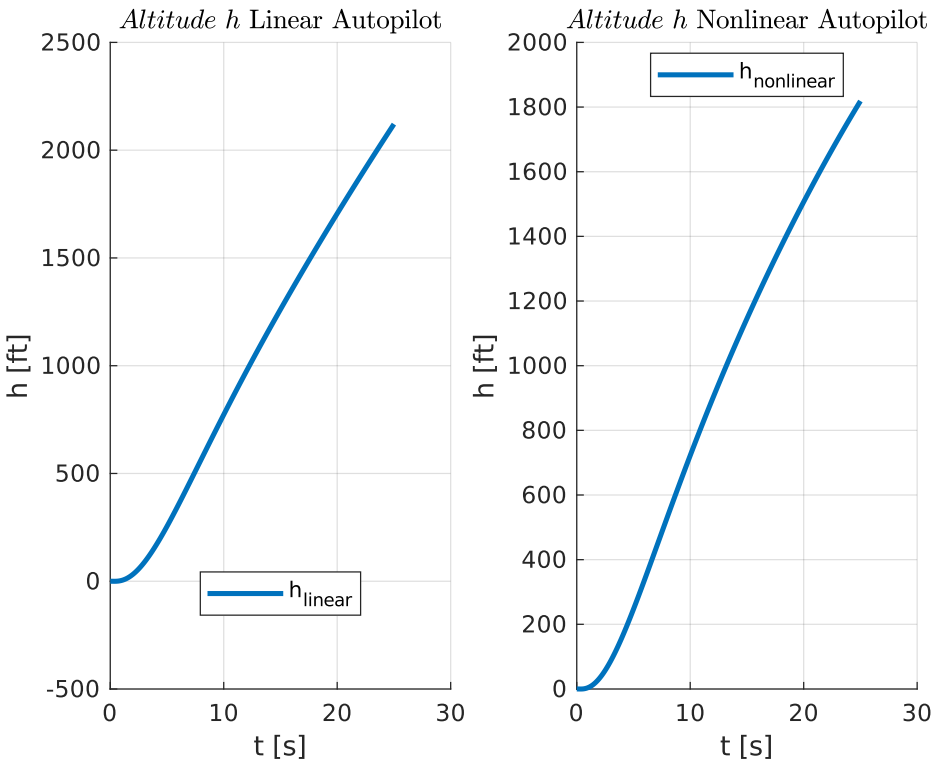


Figure 68: Response to pitch command 10°

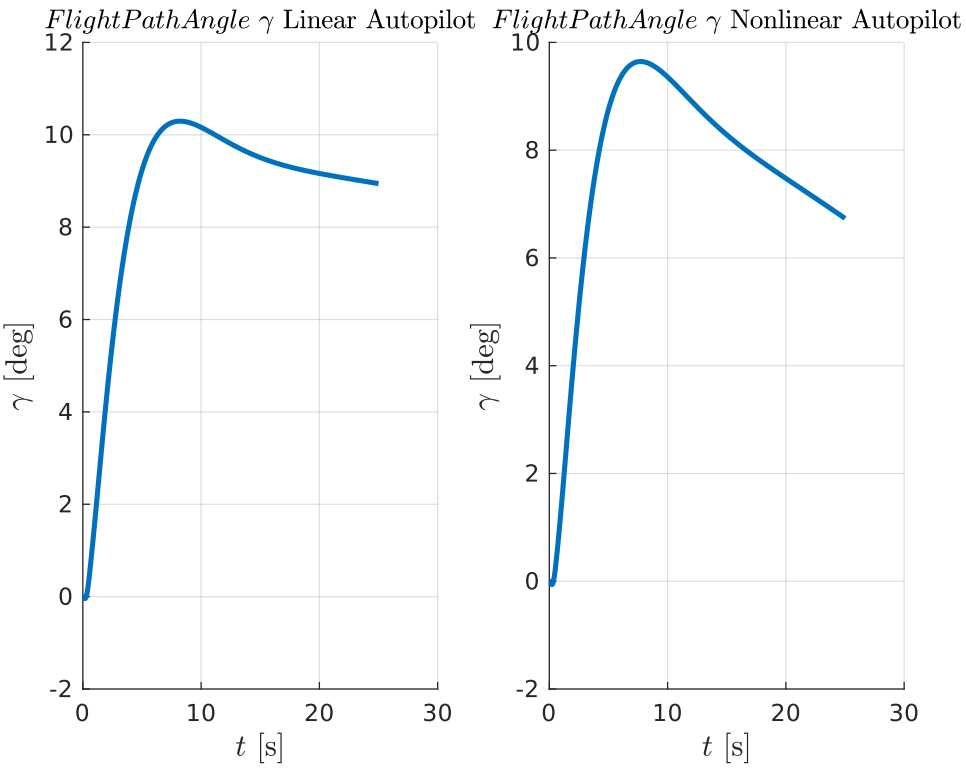


Figure 69: Response to pitch command 10°

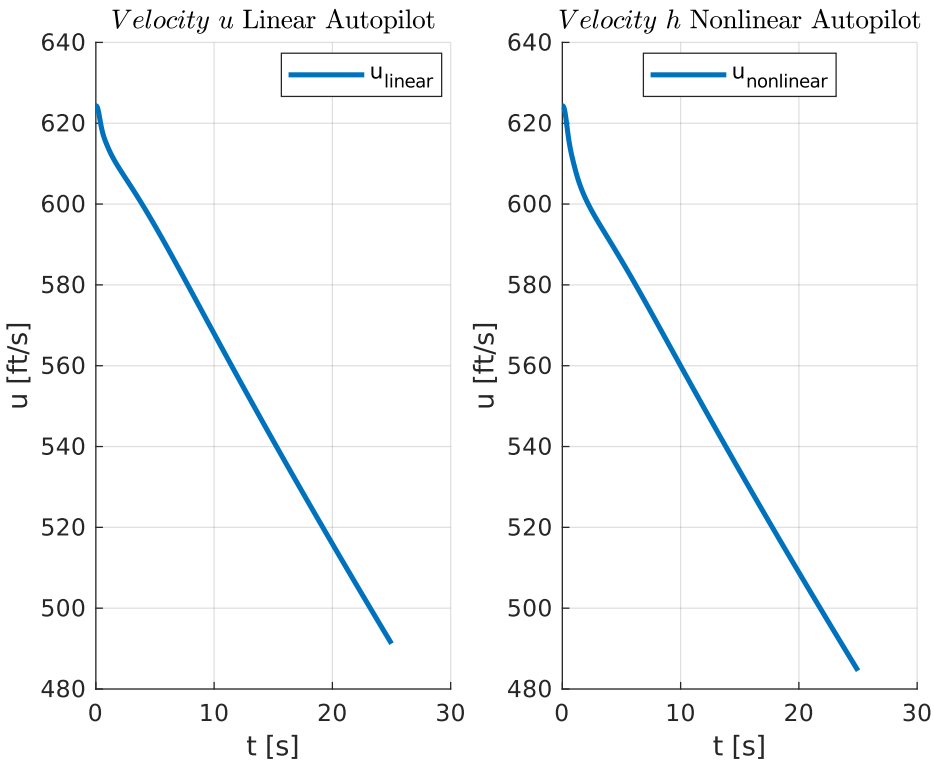


Figure 70: Response to pitch command 10°

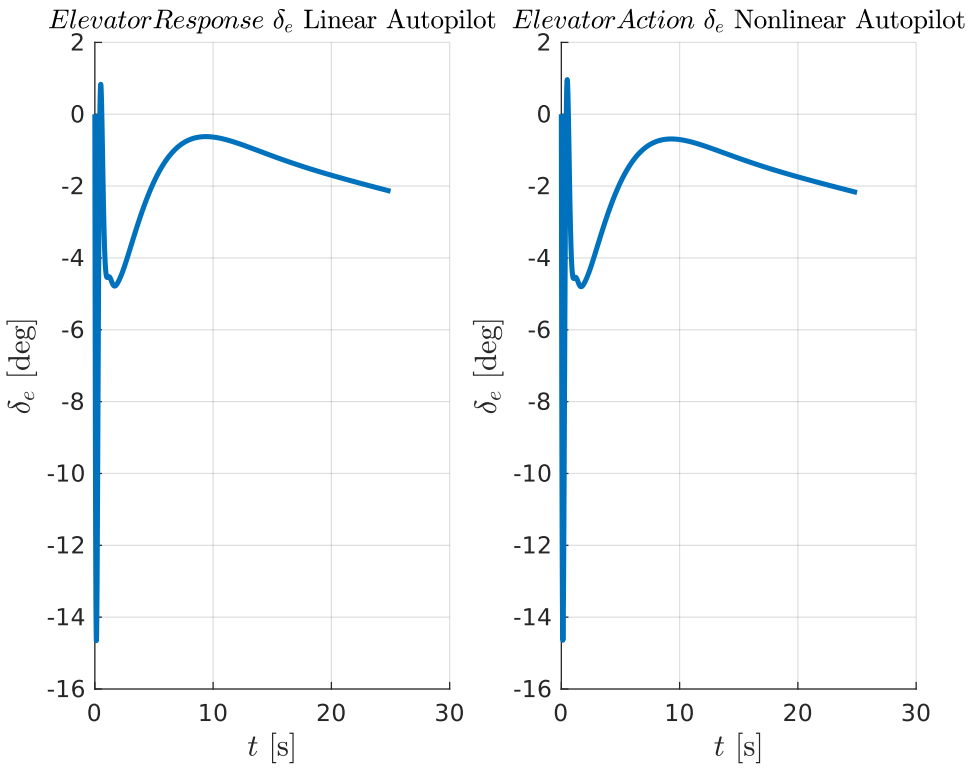


Figure 71: Response to pitch command 10°

36 Elevator from Pitch + Airspeed Autopilot

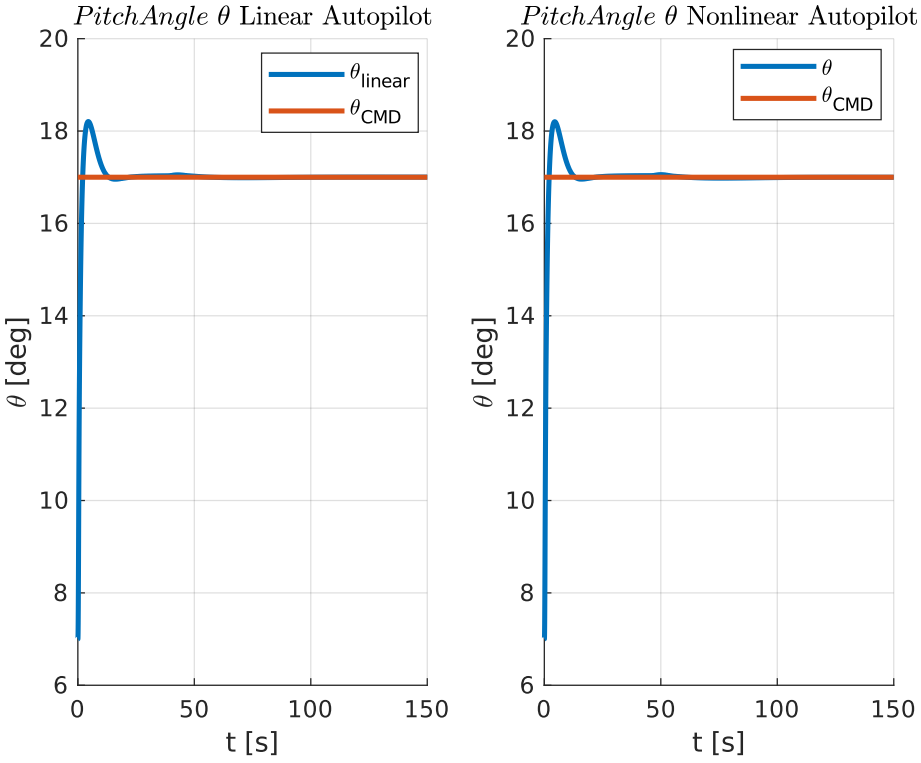


Figure 72: Response to pitch command 10°

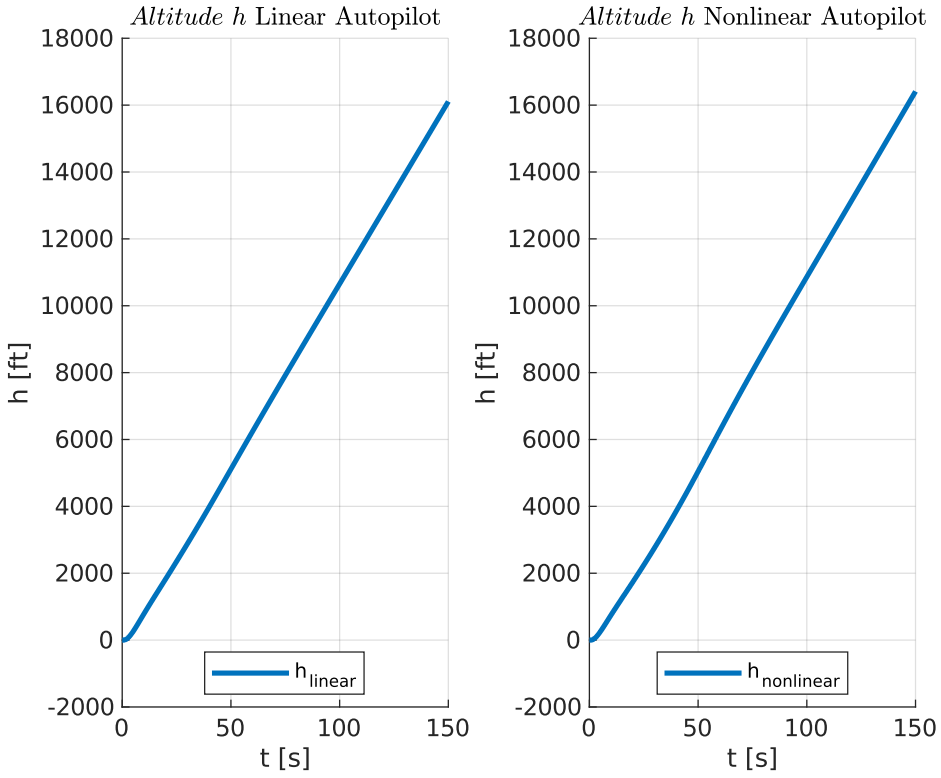


Figure 73: Response to pitch command 10°

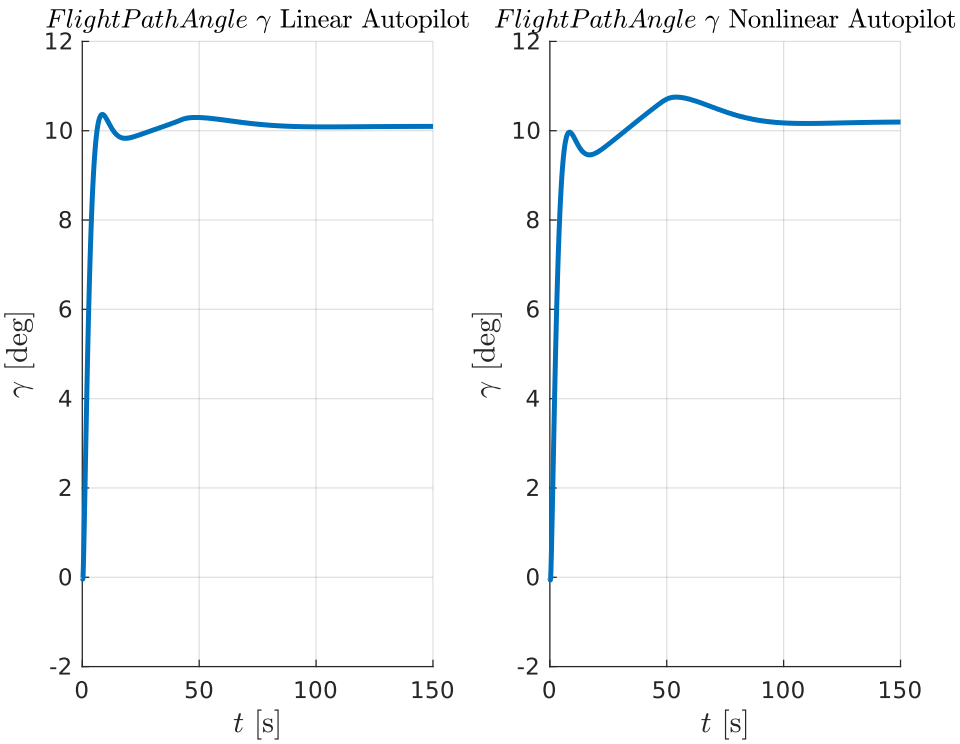


Figure 74: Response to pitch command 10°

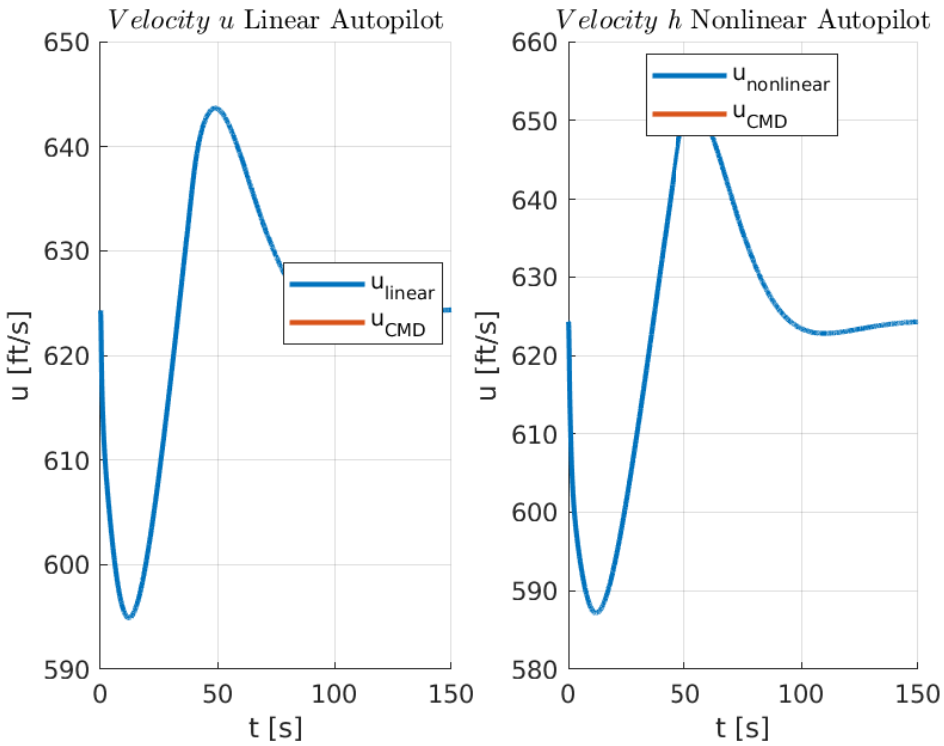


Figure 75: Response to pitch command 10°

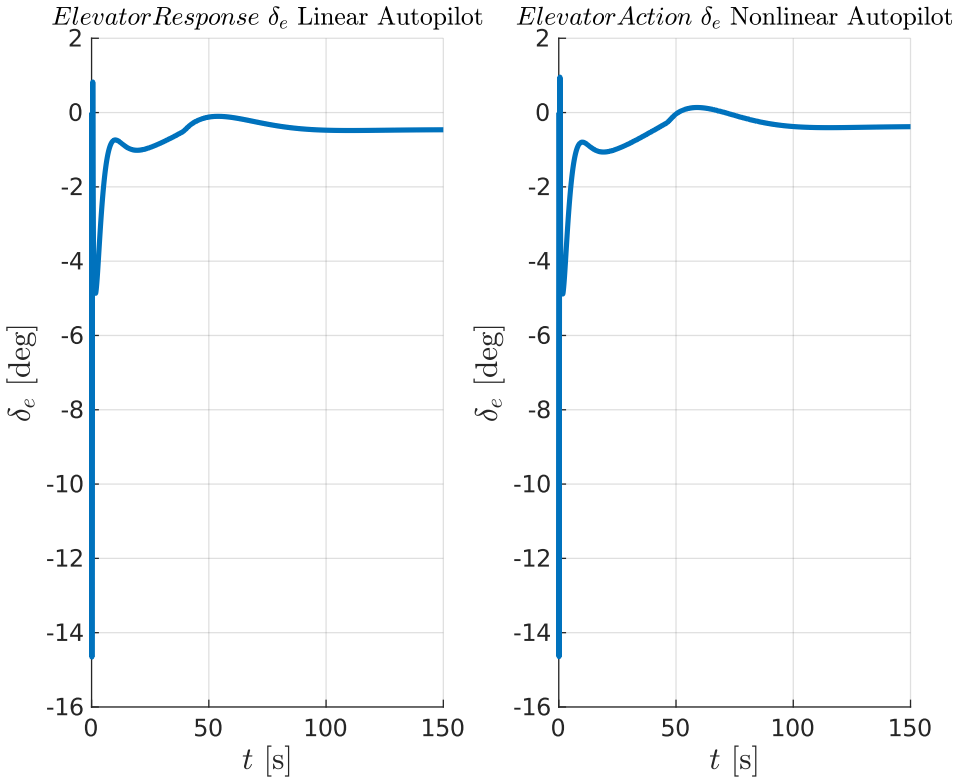


Figure 76: Response to pitch command 10°

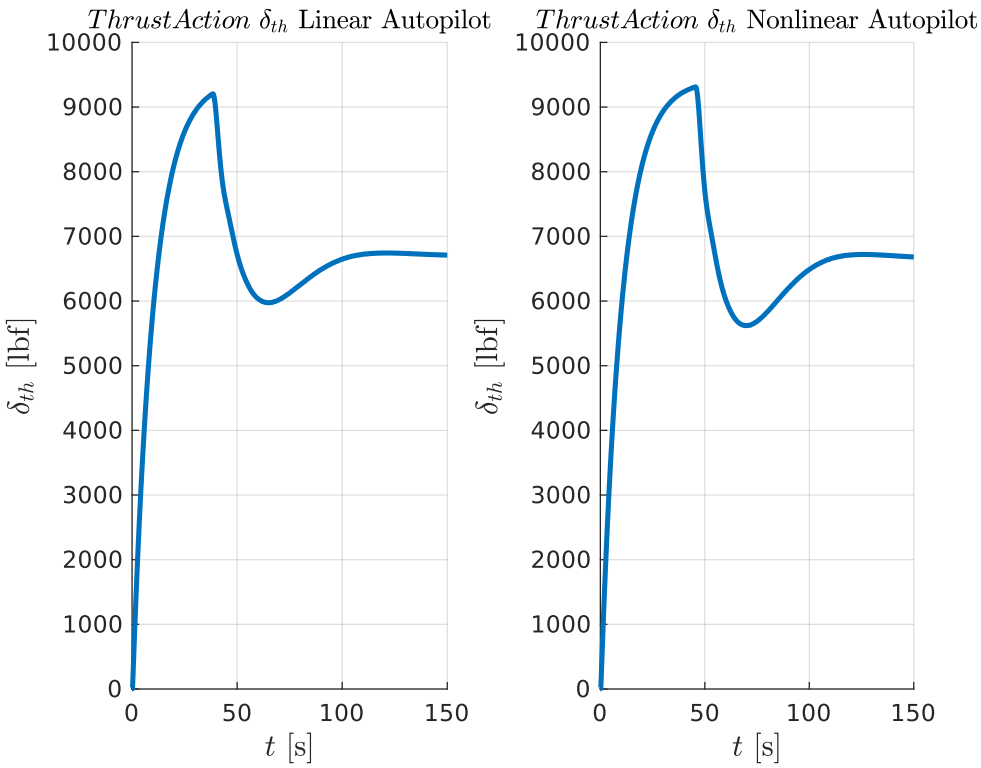


Figure 77: Response to pitch command 10°

37 Altitude Hold Autopilot

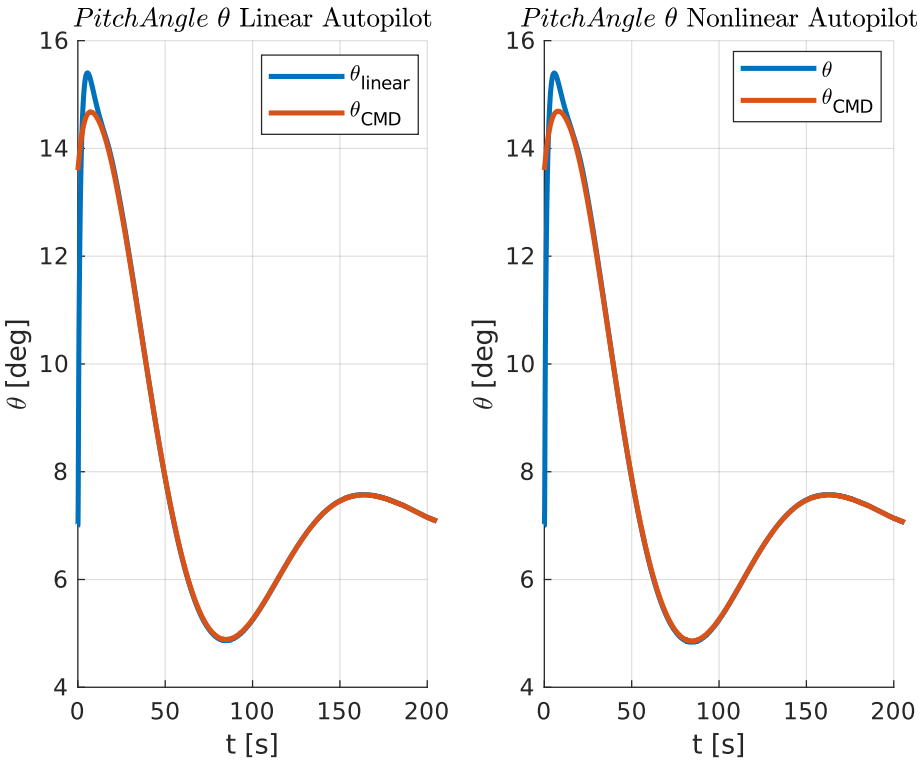


Figure 78: Response to altitude command 2000 ft

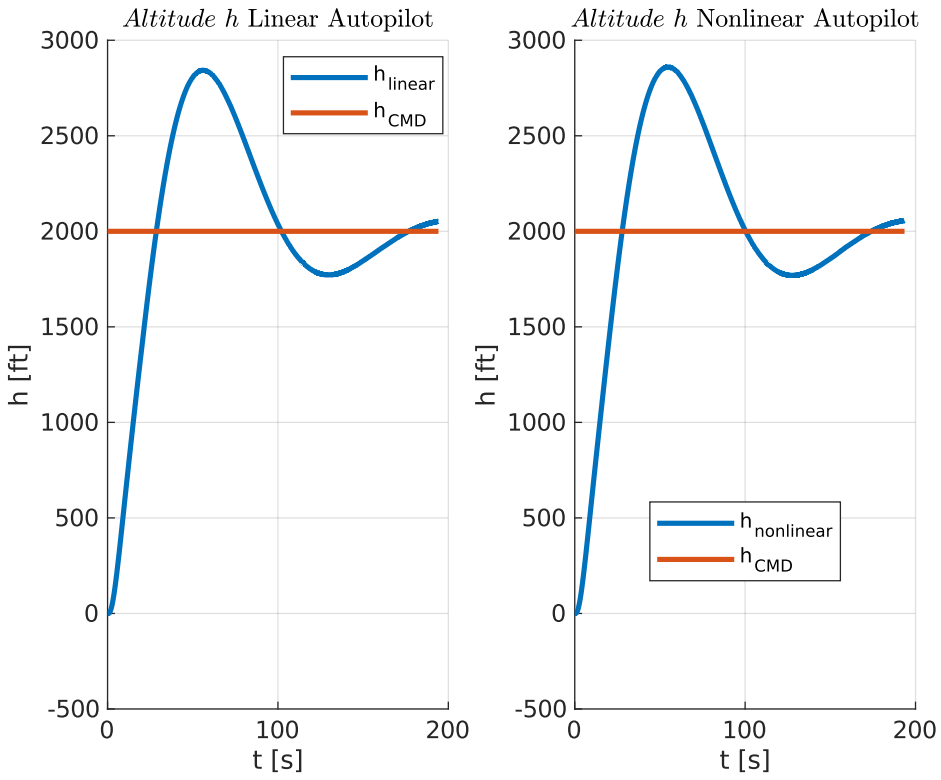


Figure 79: Response to altitude command 2000 ft

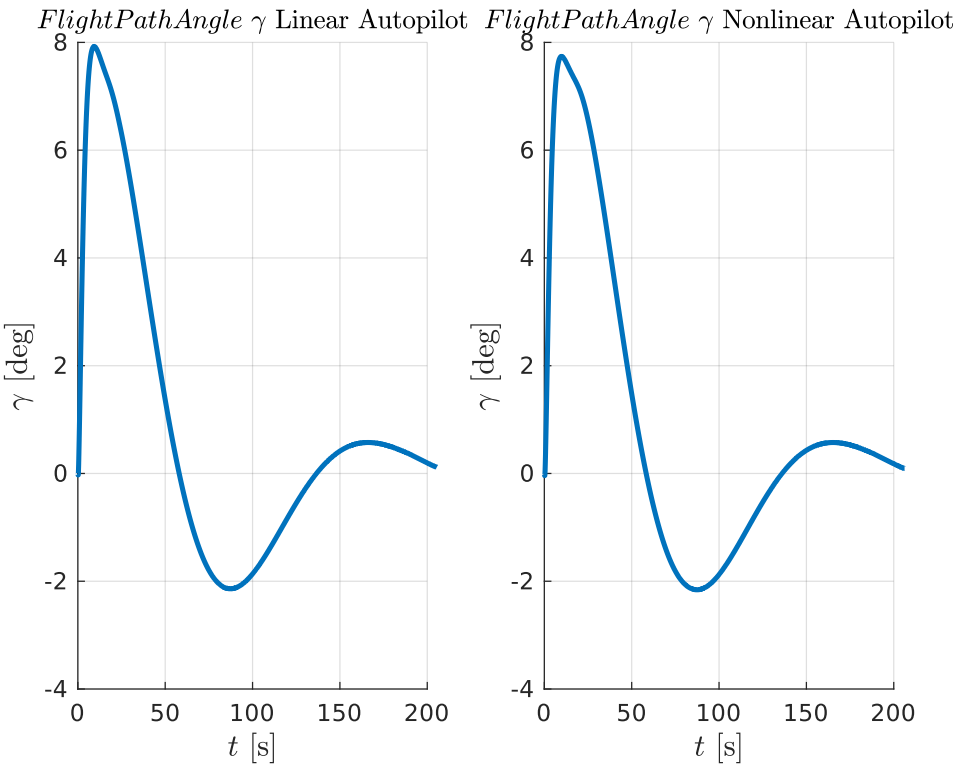


Figure 80: Response to altitude command 2000 ft

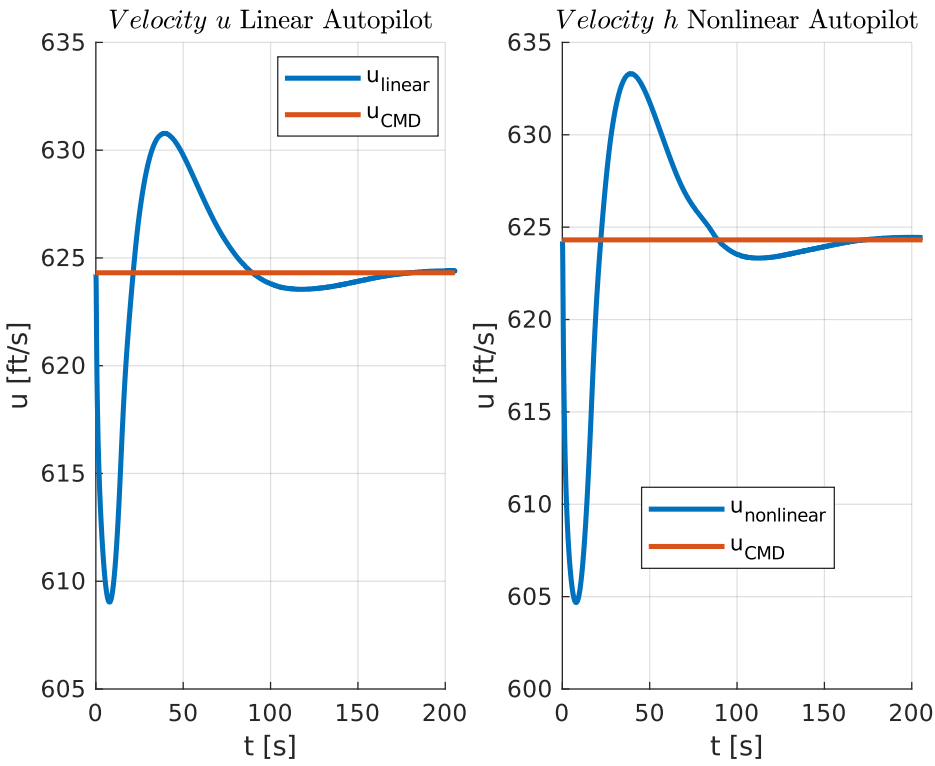


Figure 81: Response to altitude command 2000 ft

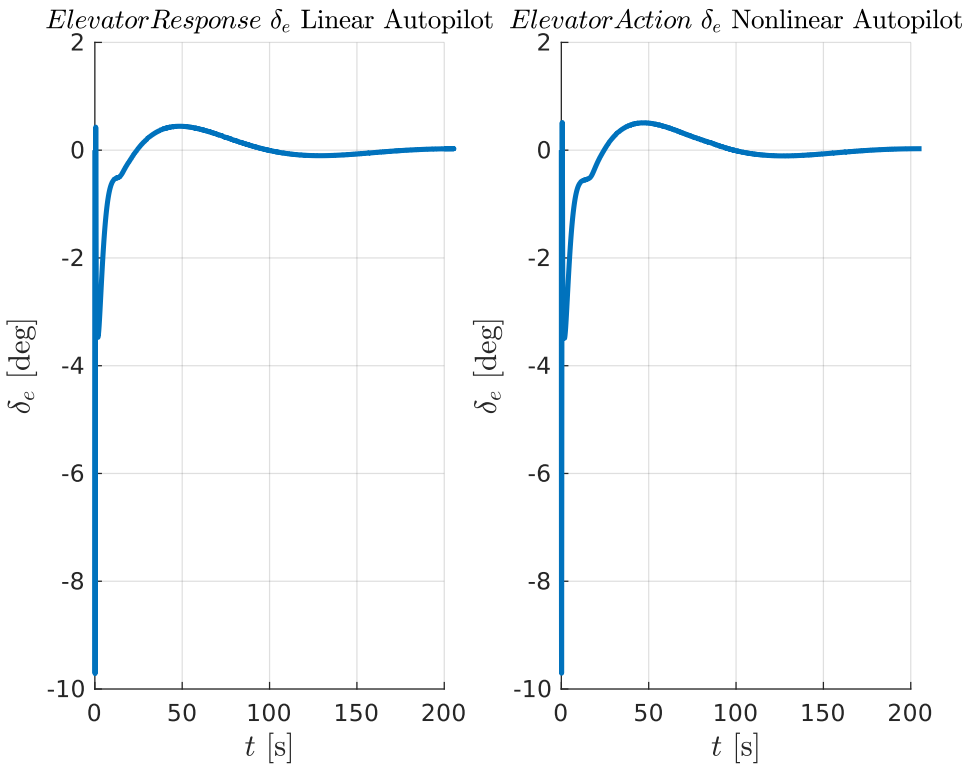


Figure 82: Response to altitude command 2000 ft

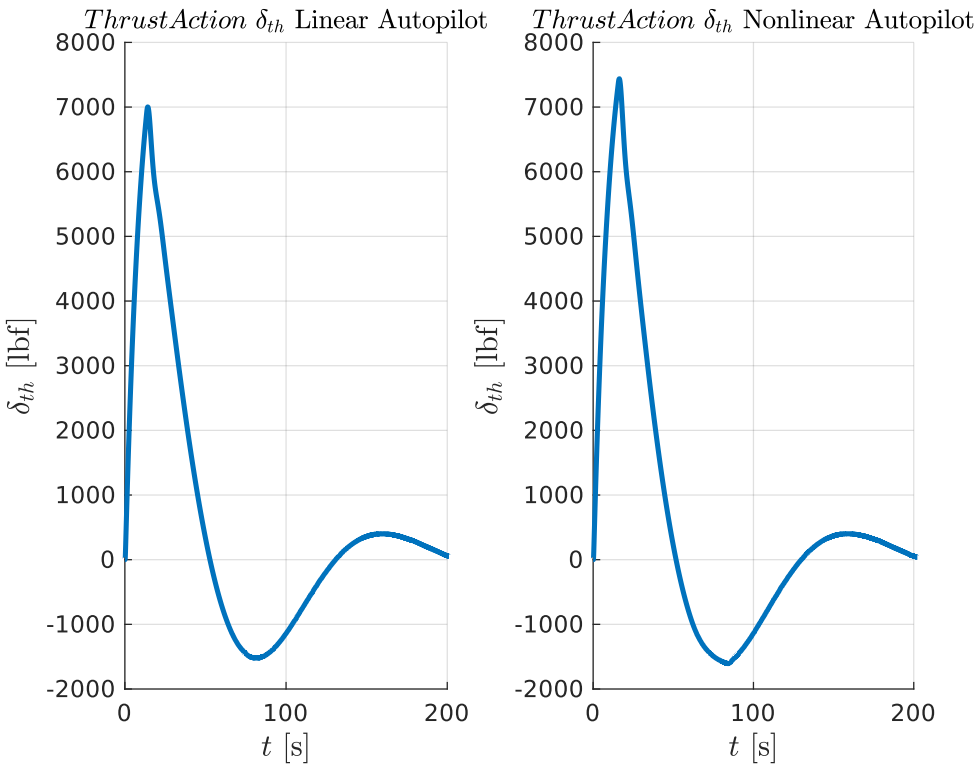


Figure 83: Response to pitch command 10°

38 Lateral Autopilot

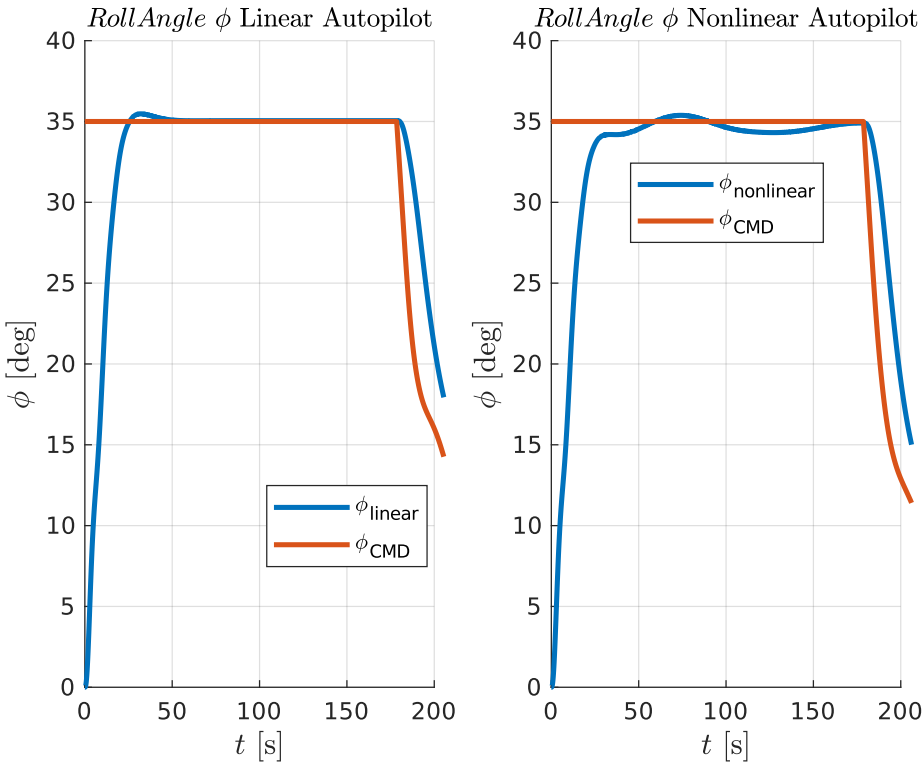


Figure 84: Response to heading angle command 360°

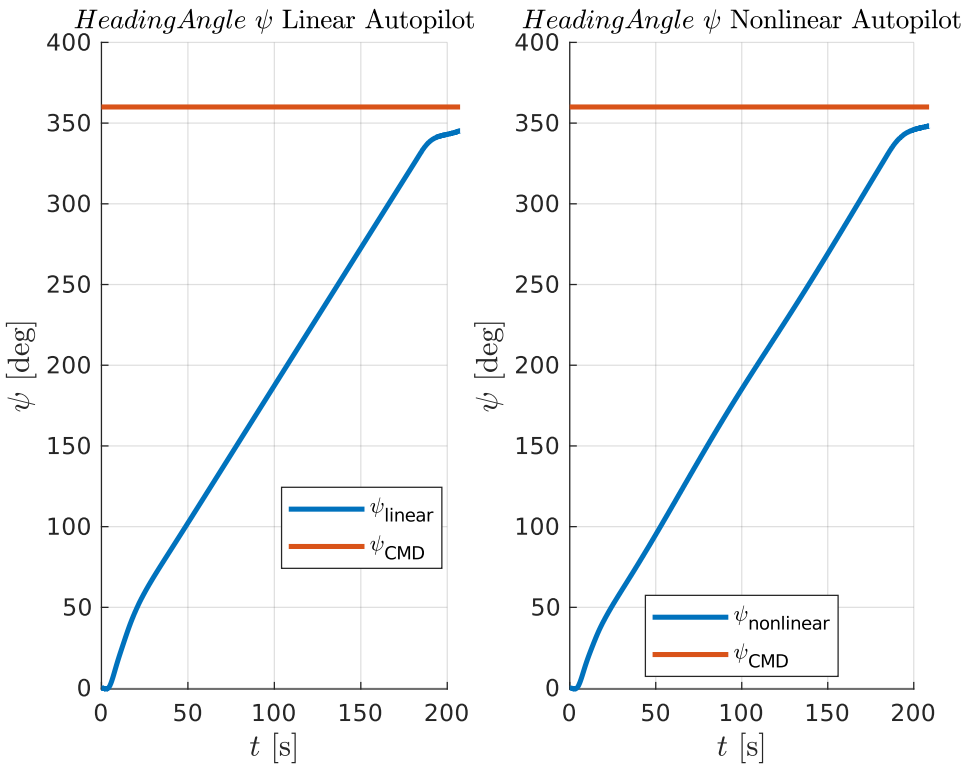


Figure 85: Response to heading angle command 360°

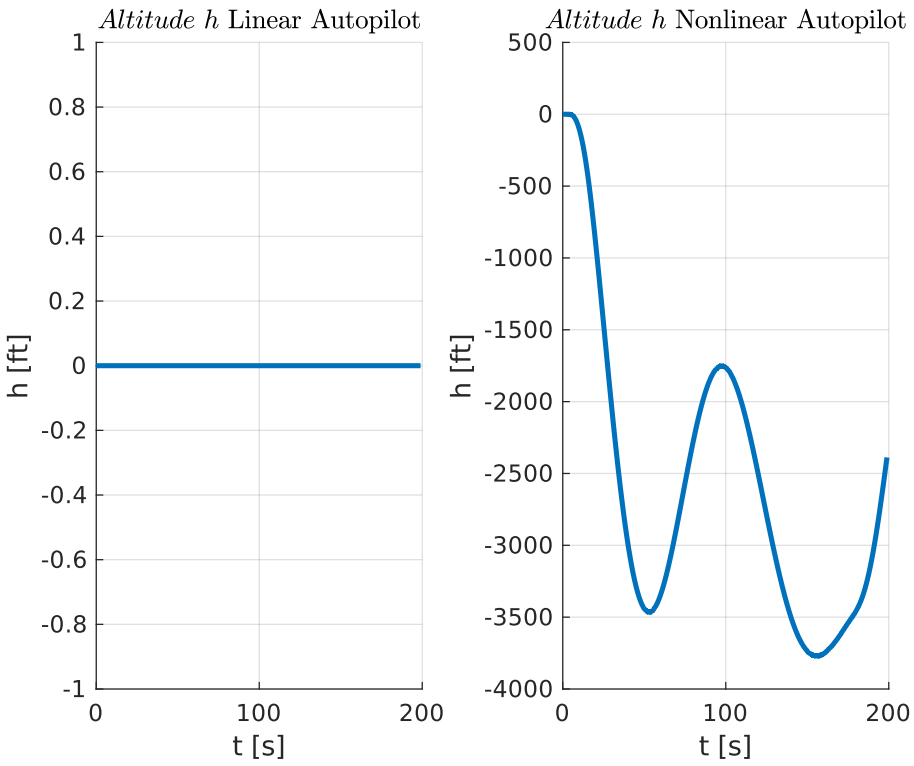


Figure 86: Response to heading angle command 360°

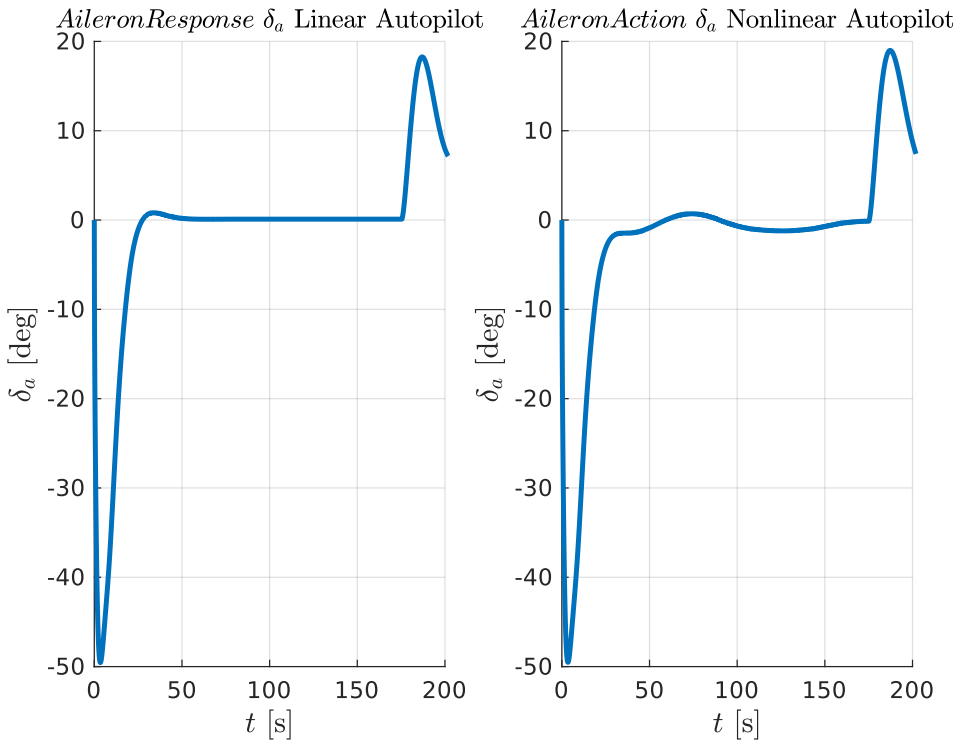


Figure 87: Response to heading angle command 360°

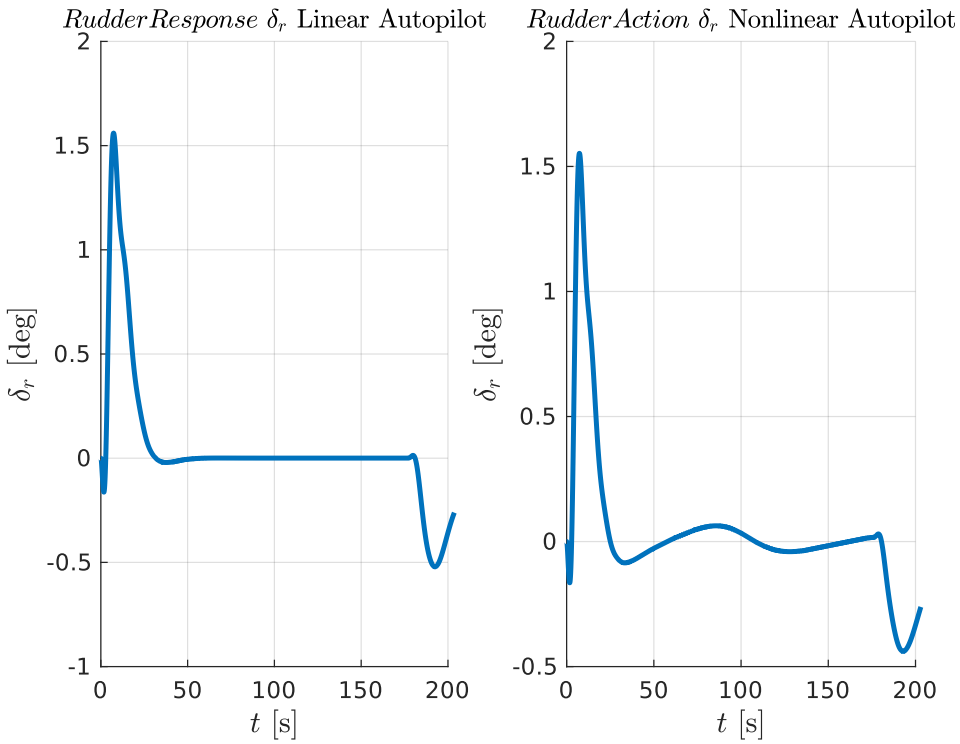


Figure 88: Response to heading angle command $360°$

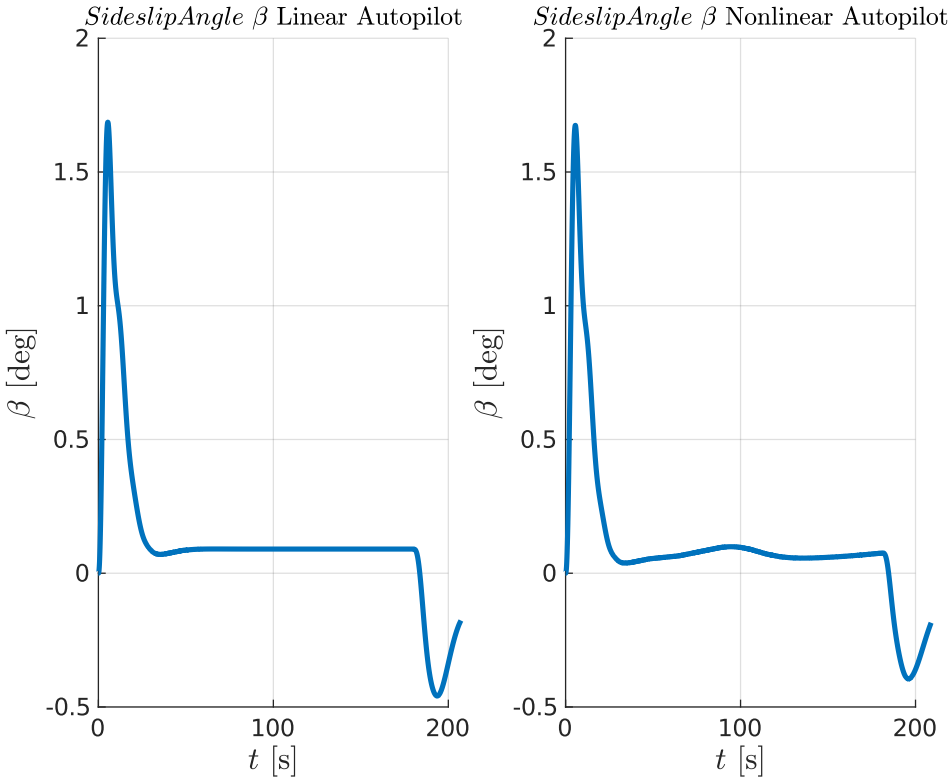


Figure 89: Response to heading angle command 360°

39 Longitudinal + Lateral Autopilot

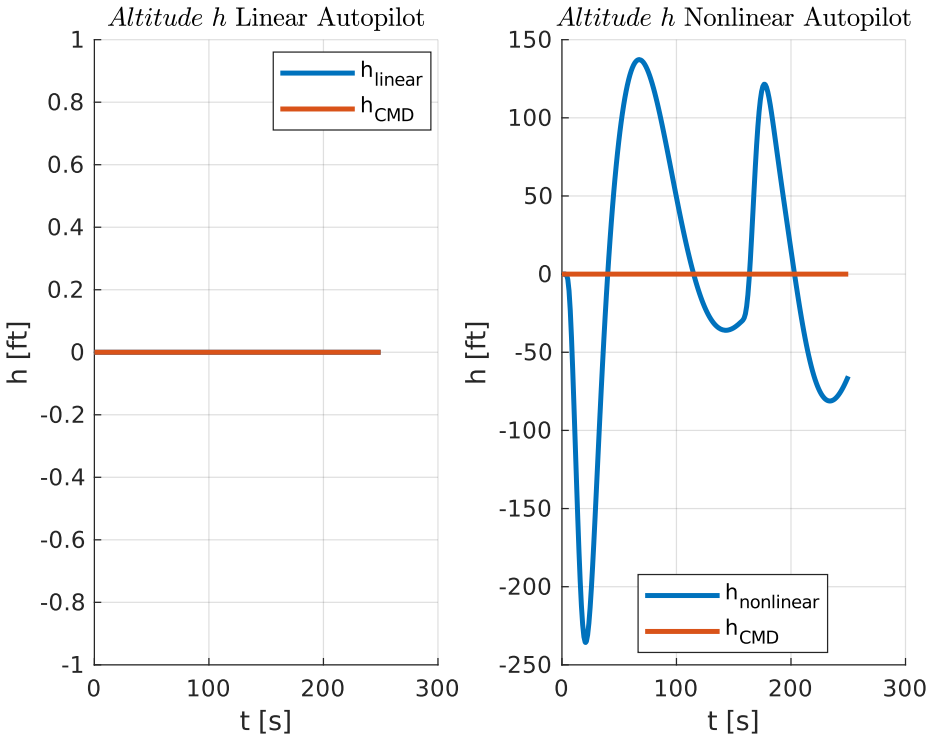


Figure 90: Response to heading angle command $360°$

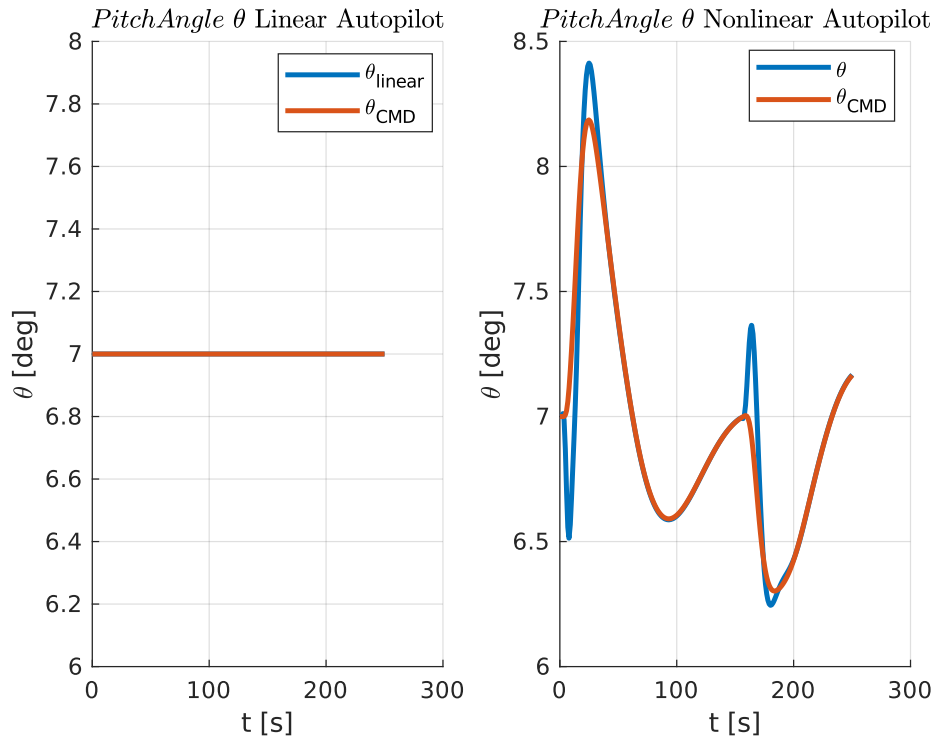


Figure 91: Response to heading angle command 360°

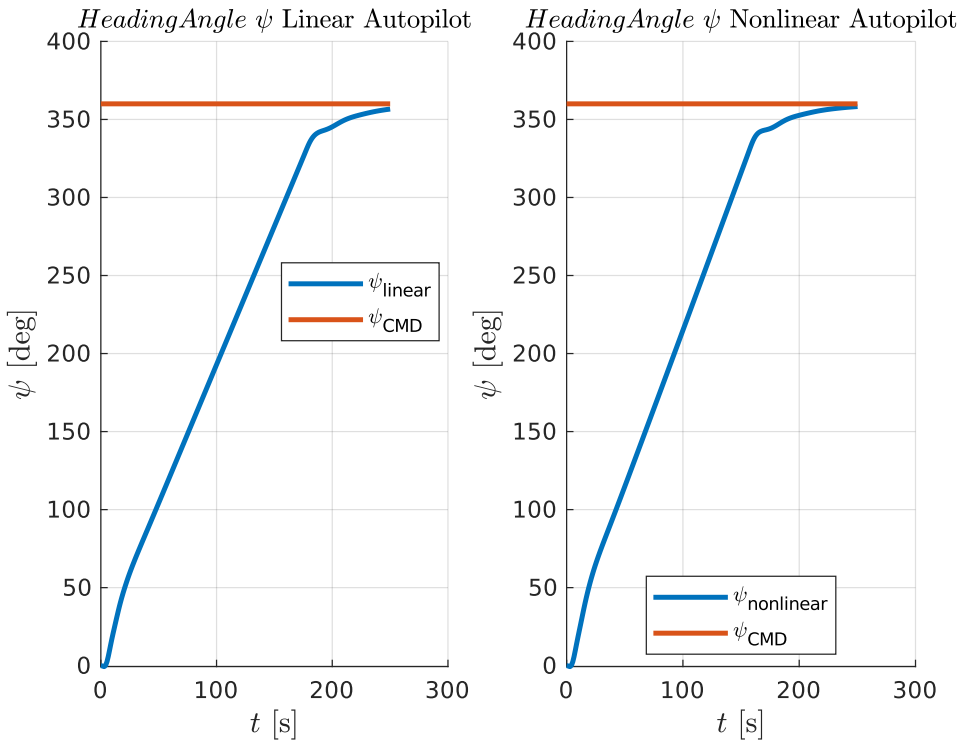


Figure 92: Response to heading angle command 360°

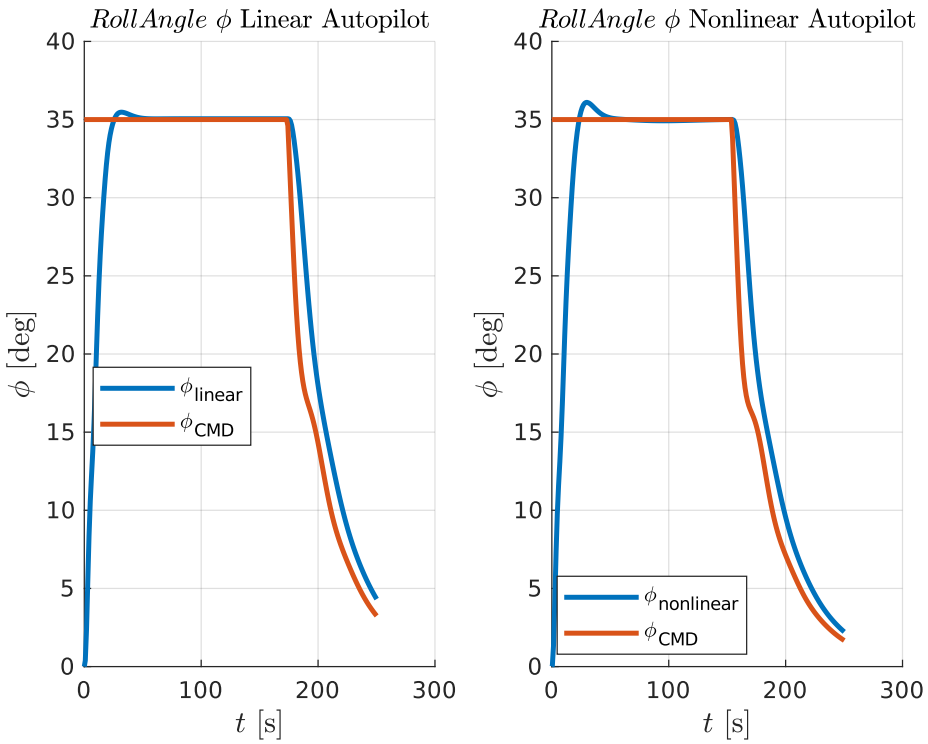


Figure 93: Response to heading angle command 360°

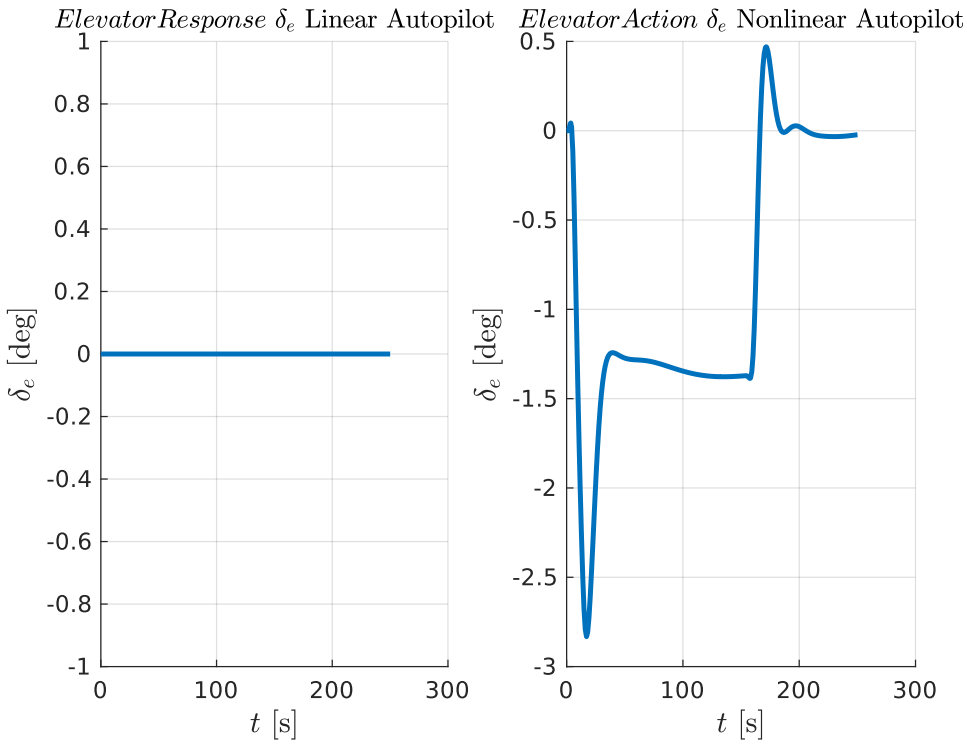


Figure 94: Response to heading angle command 360°

(Clb*Cnr)/(Clr*Cnb) = 1.577984 (>1 if spirally stable)

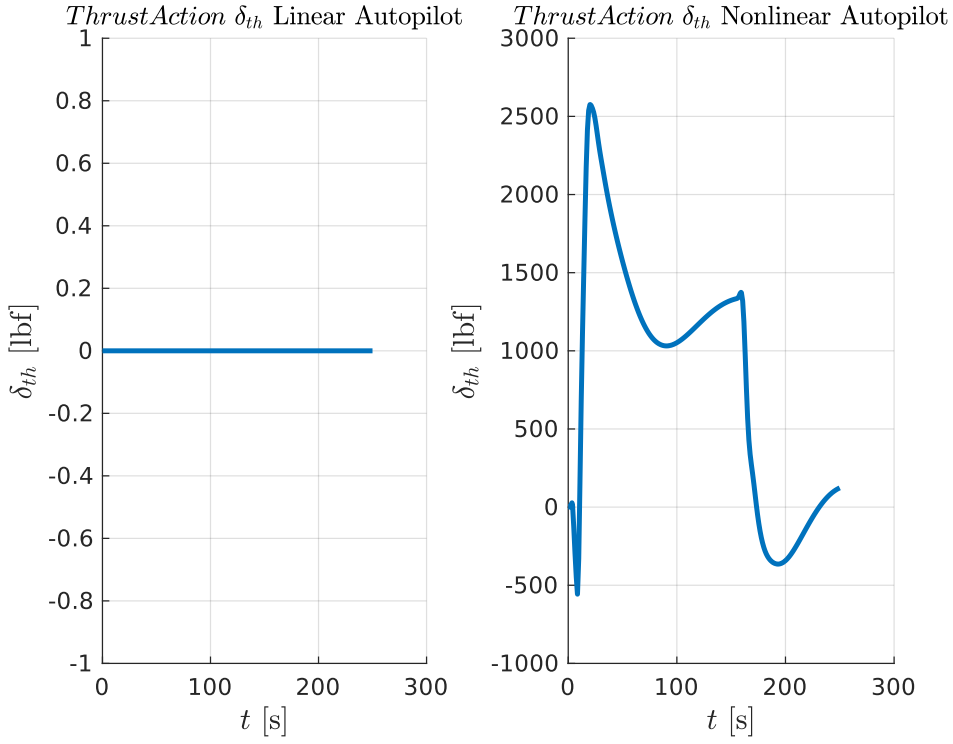


Figure 95: Response to heading angle command 360°

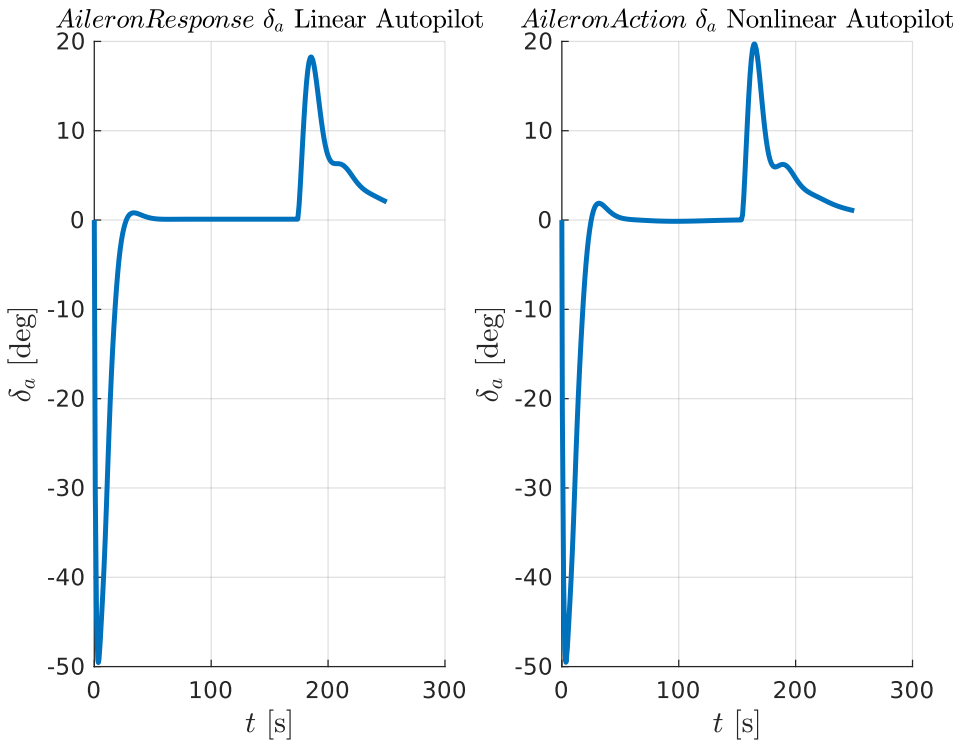


Figure 96: Response to heading angle command 360°

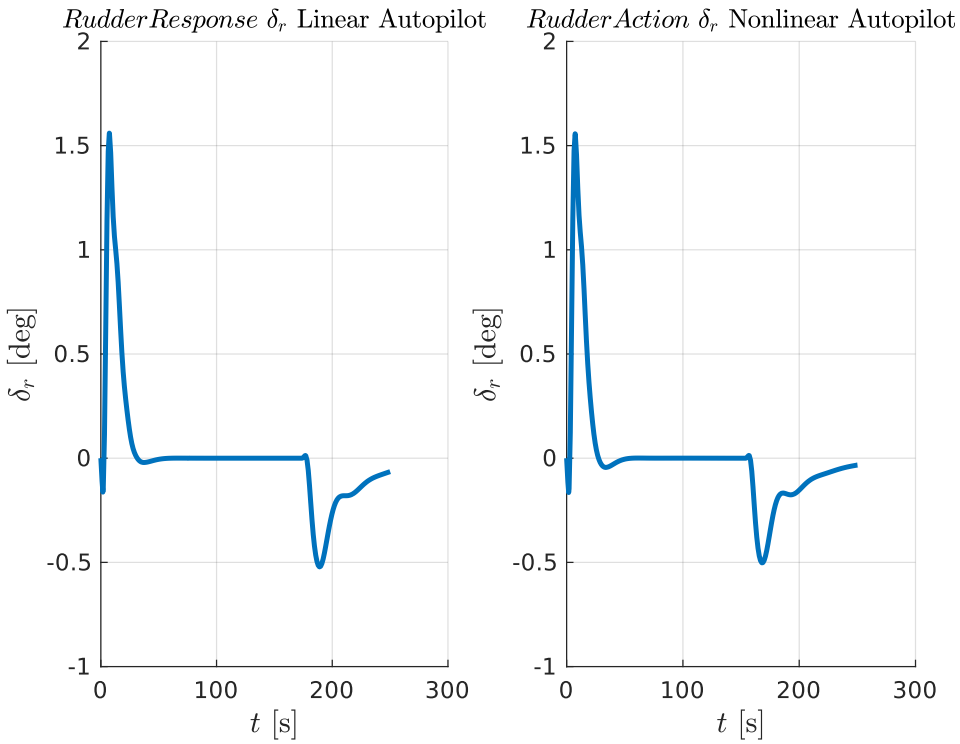


Figure 97: Response to heading angle command $360°$

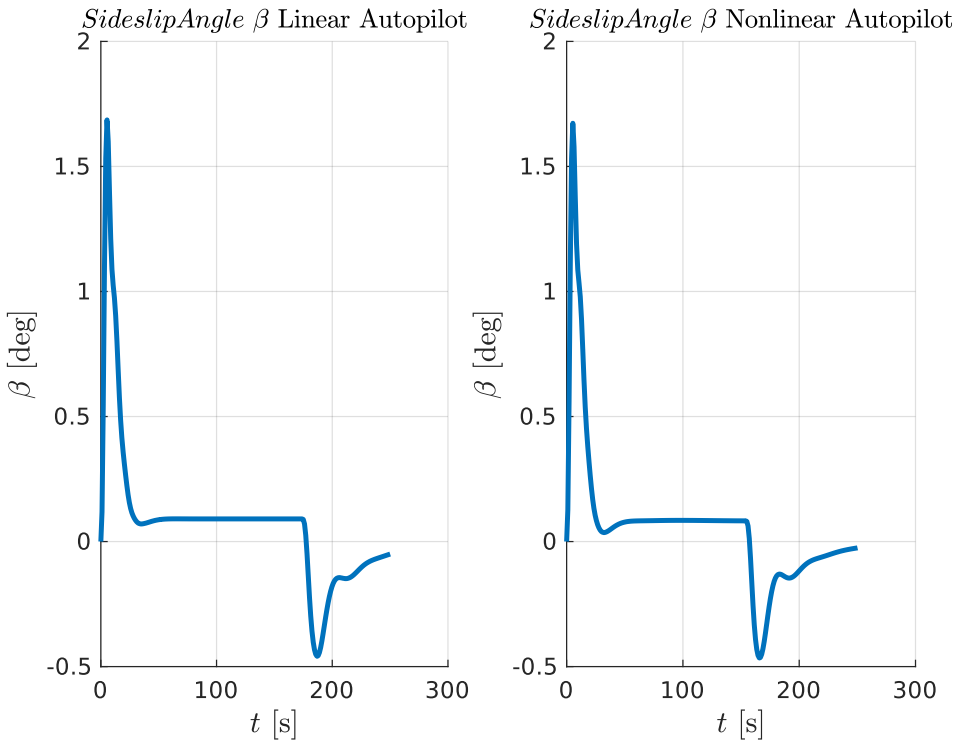


Figure 98: Response to heading angle command 360°

40 Complete Mission

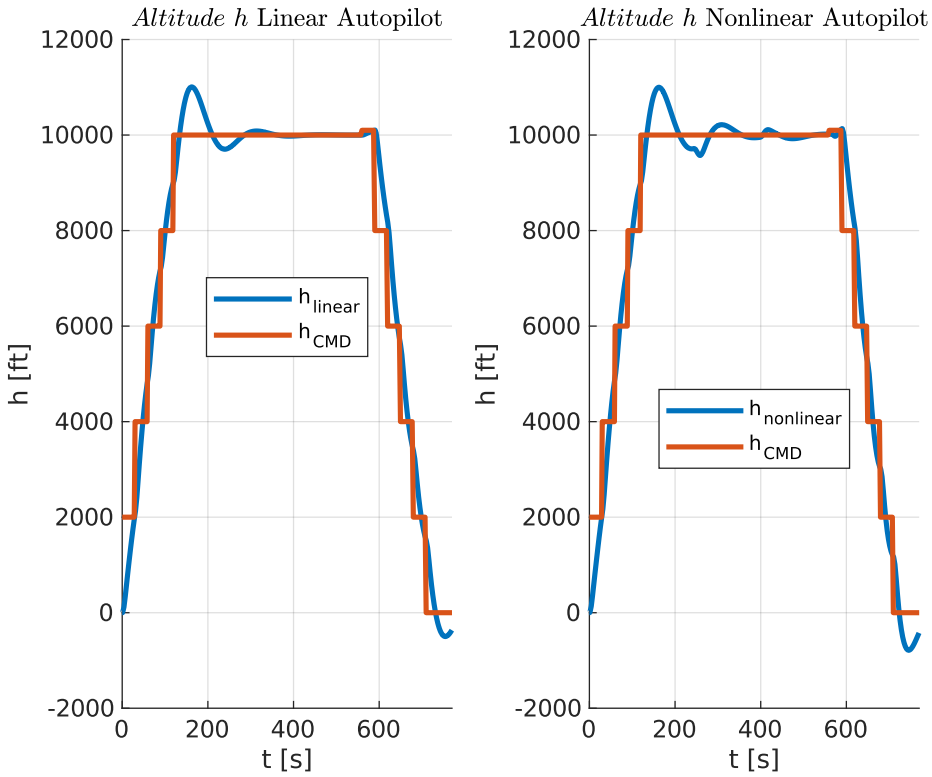


Figure 99: Response due to specific mission, notice the little command of additional 100 ft at time ~ 590 sec

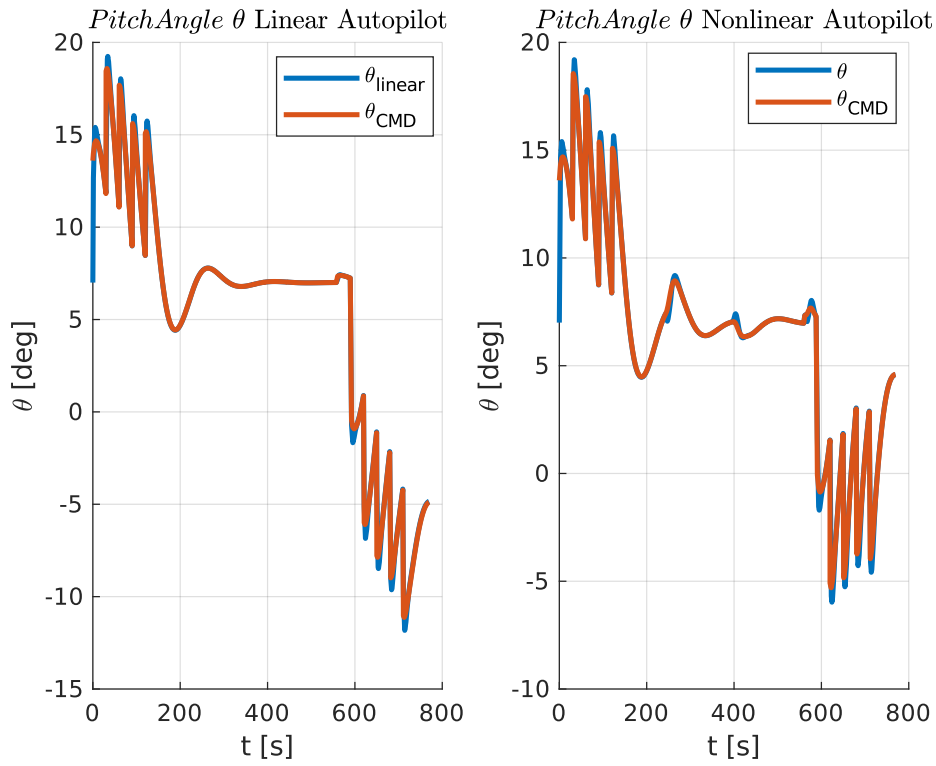


Figure 100: Response due to specific mission

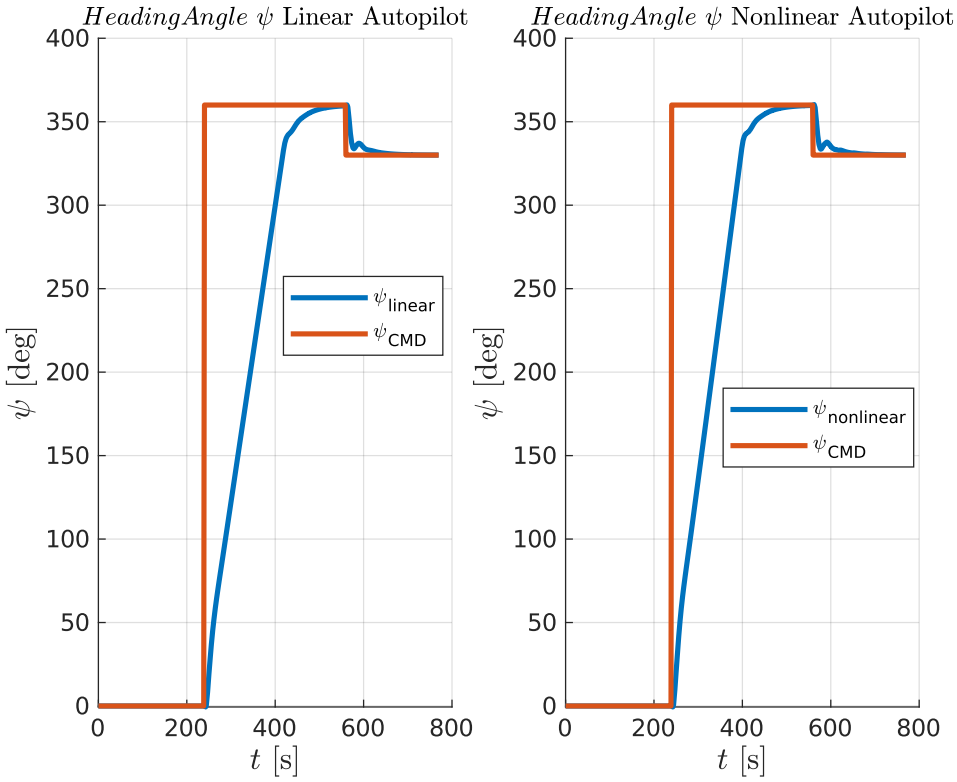


Figure 101: Response due to specific mission

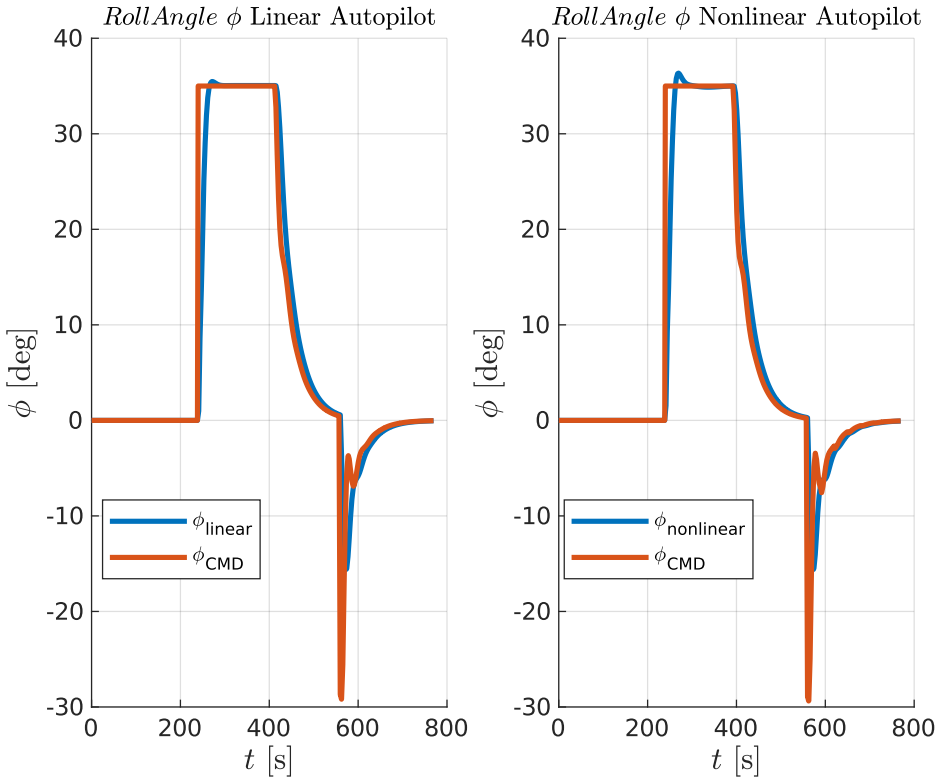


Figure 102: Response due to specific mission

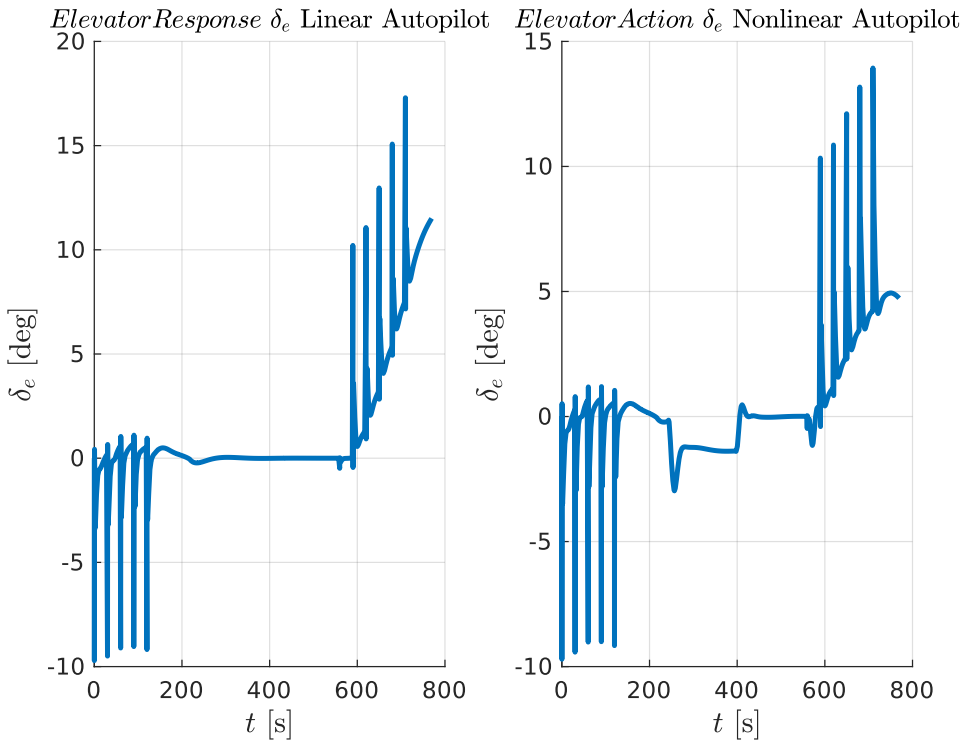


Figure 103: Response due to specific mission

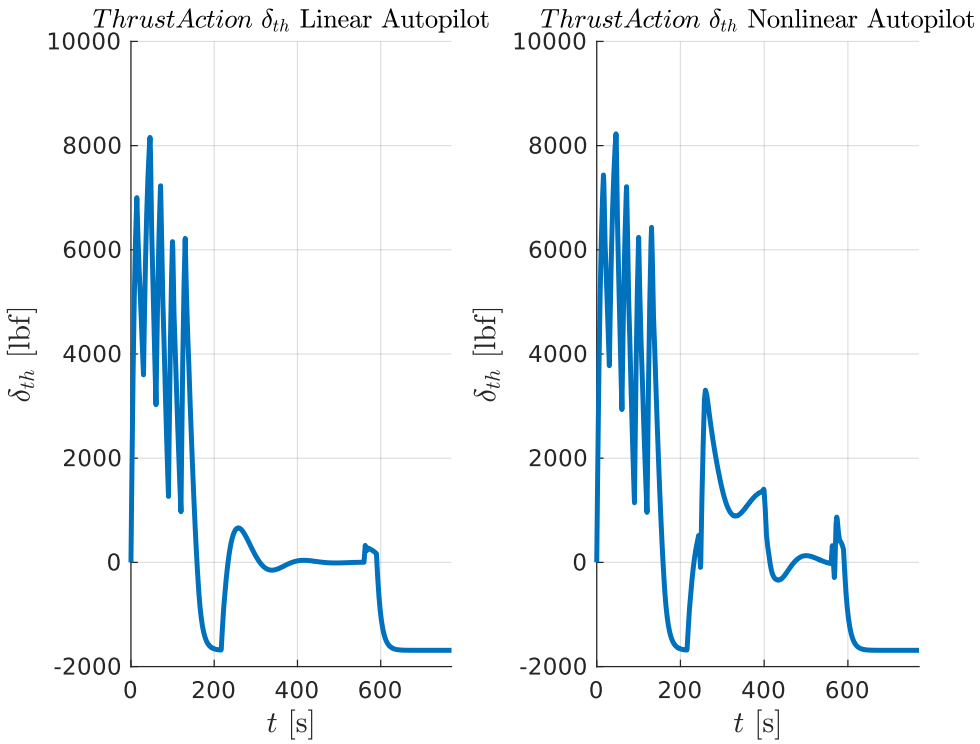


Figure 104: Response due to specific mission

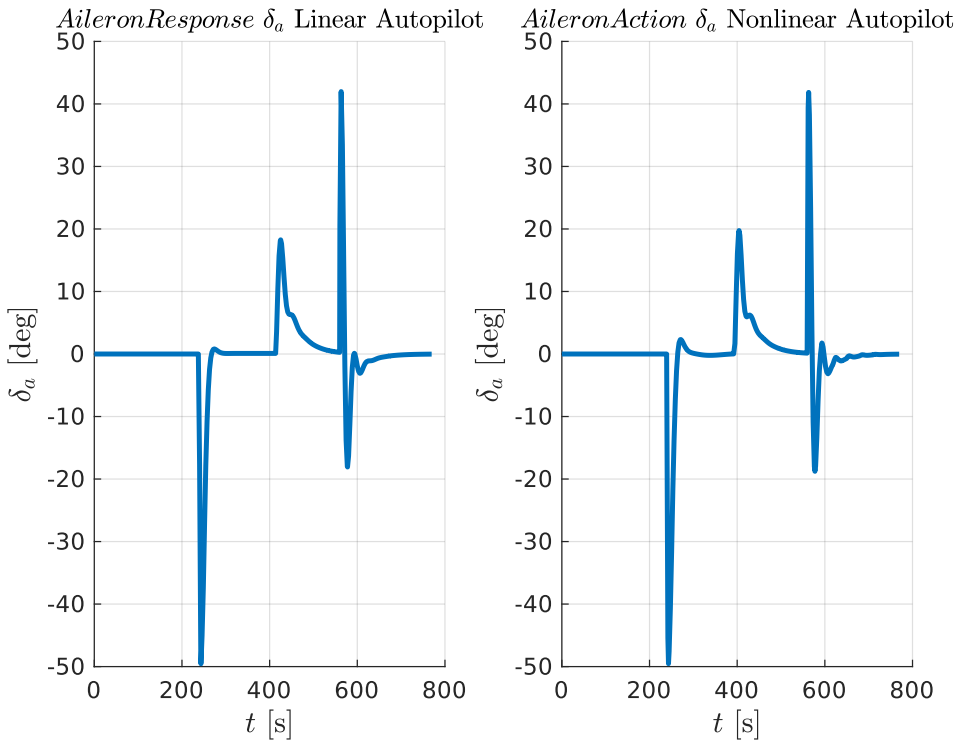


Figure 105: Response due to specific mission

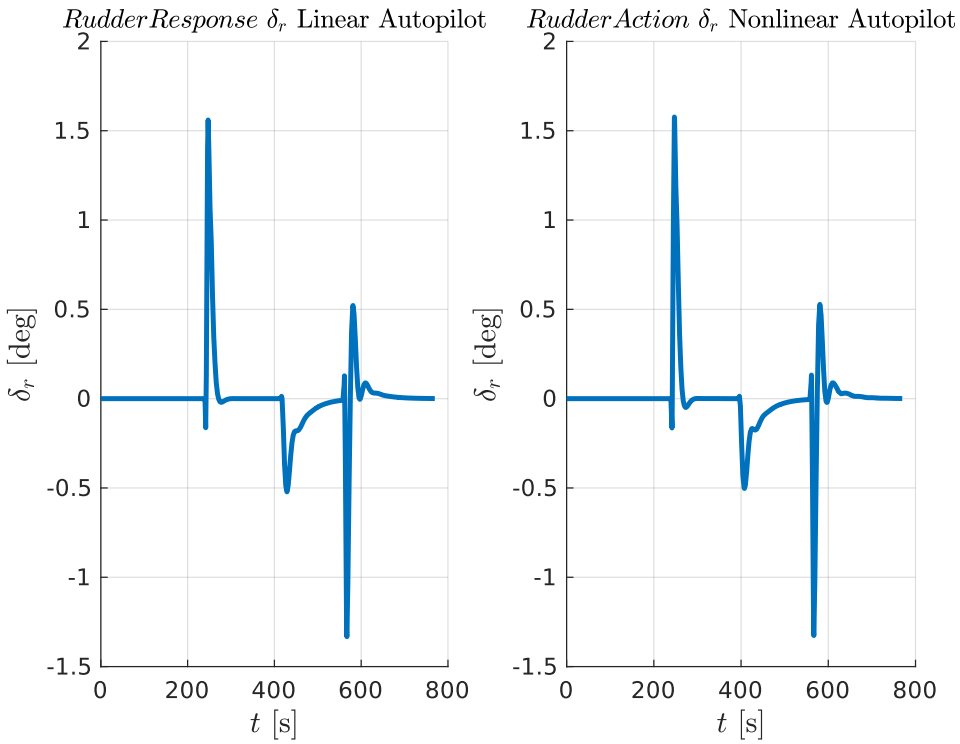


Figure 106: Response due to specific mission

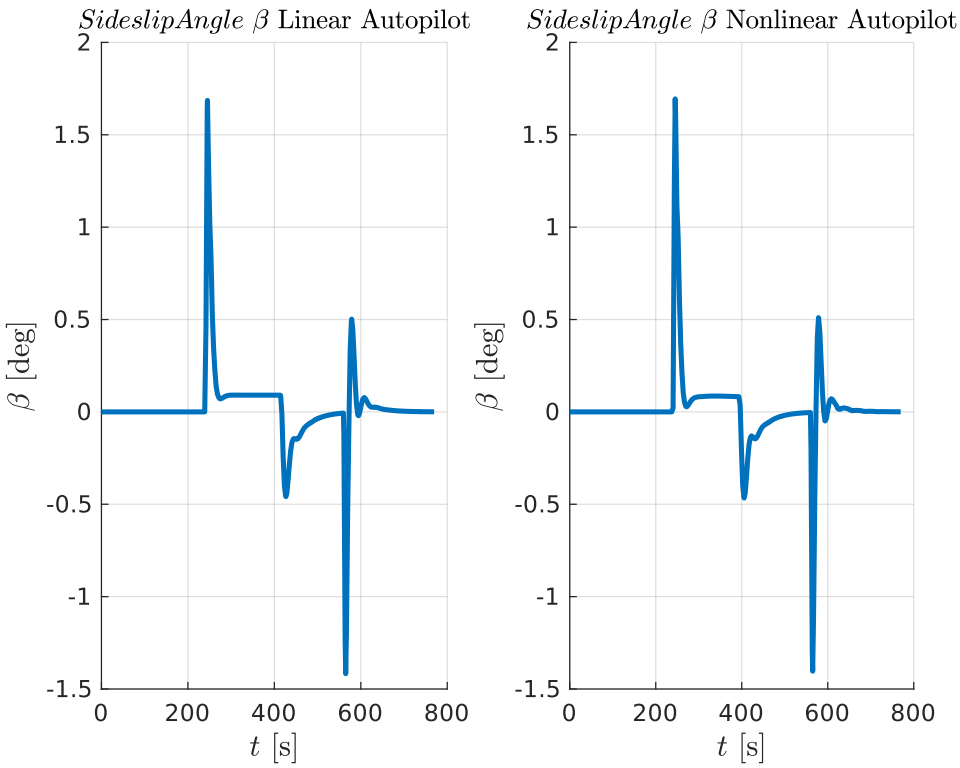


Figure 107: Response due to specific mission

From results, the airplane did a great job in finishing the total mission in less than 13 minutes, it also achieved a very good climb rate, better than the required one, it is about 4000 ft/s, which is standard of the Jetstar. Although the lateral autopilot is slow, it still achieved the mission very well, and the airplane almost achieved a fair enough coordinated level turn.

Appendix

Simulink Code

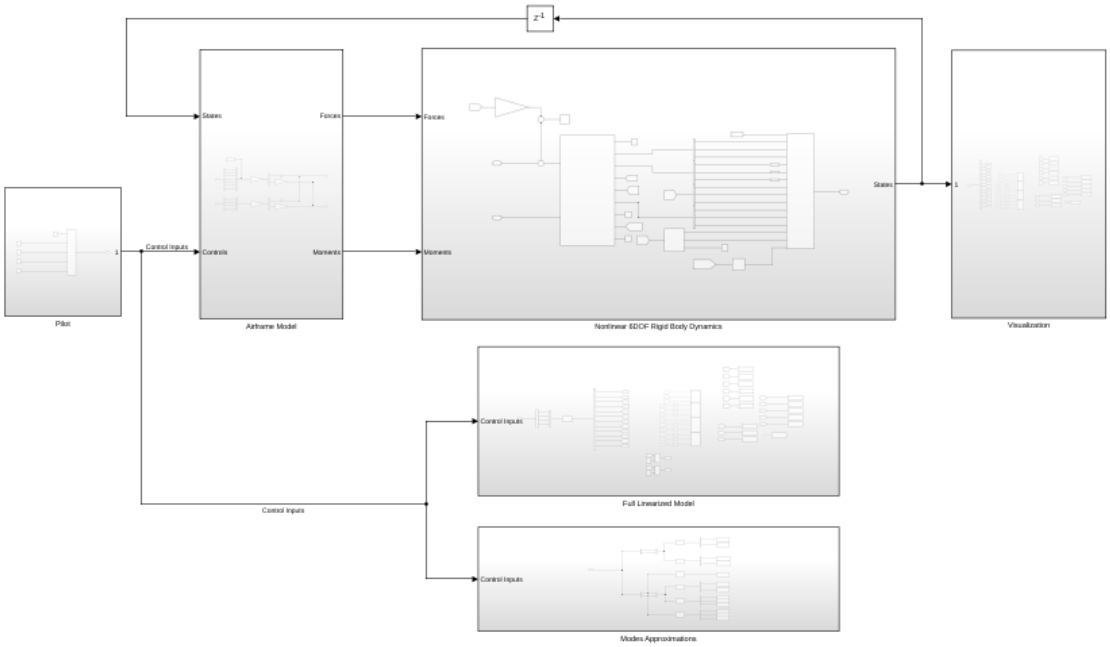


Figure 108: Simulink Subsystems

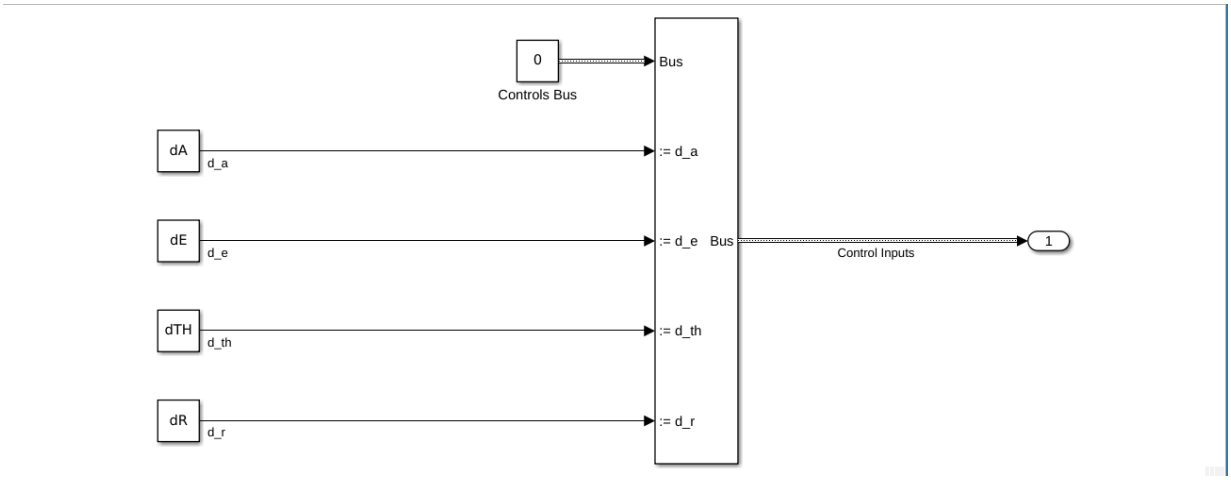


Figure 109: Pilot Subsystem

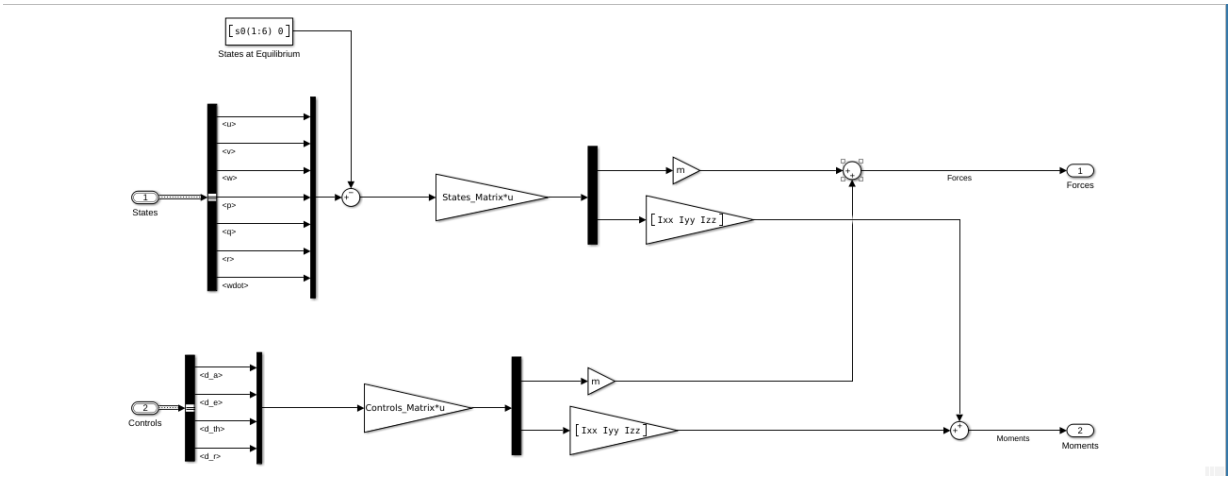


Figure 110: Airframe Model

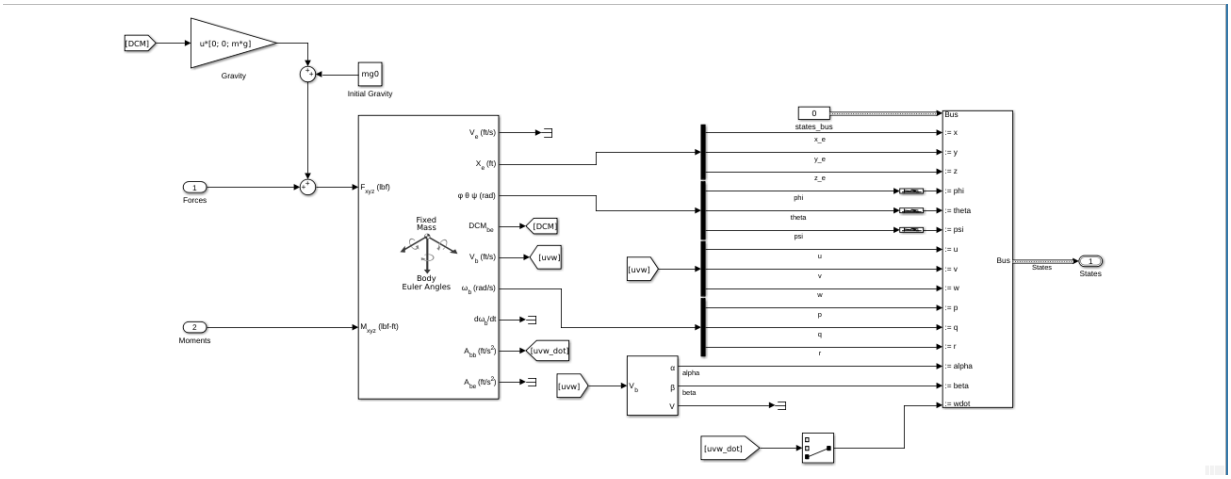


Figure 111: RBD Solver Subsystem

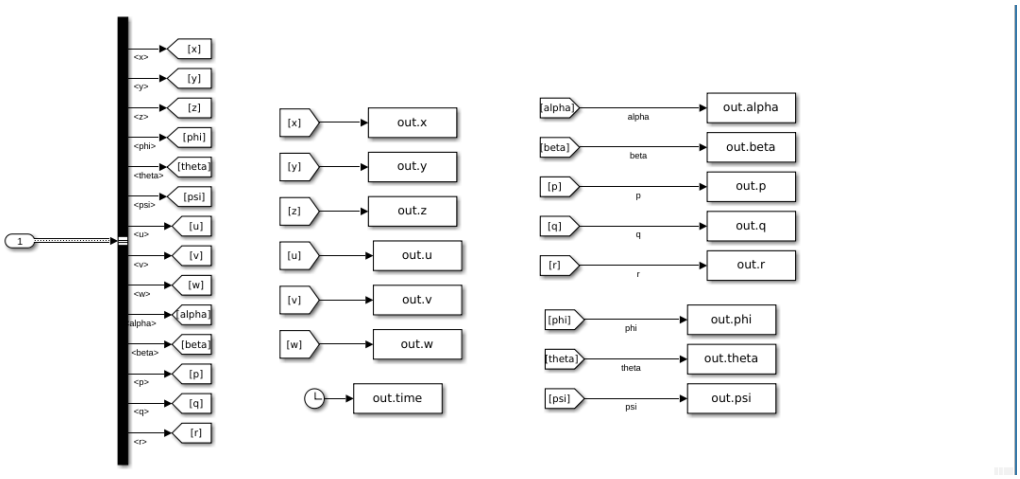


Figure 112: Visualization & Post-Processing Subsystem

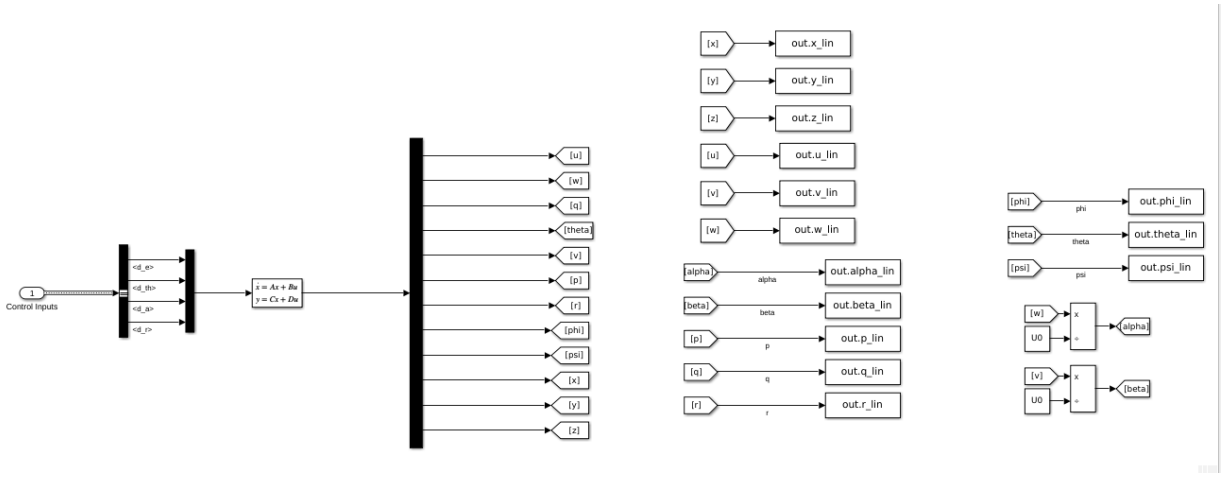


Figure 113: Full Linearized Model Subsystem

Matlab Code

Listing 1: nonlinear_simulator.m

```
% clear; clc; close all;

%% Problem Setup
if(~exist("tf", "var"))
```

```
    tfinal = 200; % final time
    % n_iter = 20001;
    dt = 0.01; % time step
end
t = (0:dt:tfinal).'; % Time Vector

if(~exist("controls", "var"))
    controls = [5*pi/180; 0; 0; 0];
end
dA = controls(1);
dE = controls(2);
dTH = controls(3);
dR = controls(4);

aircraft_data_reader;
% Bus Creation
States_Bus_Creator;
Controls_Bus_Creator;
initial_states_bus.u = s0(1);
initial_states_bus.v = s0(2);
initial_states_bus.w = s0(3);
initial_states_bus.p = s0(4);
initial_states_bus.q = s0(5);
initial_states_bus.r = s0(6);
initial_states_bus.phi = s0(7);
initial_states_bus.theta = s0(8);
initial_states_bus.psi = s0(9);
initial_states_bus.x = s0(10);
initial_states_bus.y = s0(11);
initial_states_bus.z = s0(12);
initial_states_bus.alpha = s0(8);
initial_states_bus.beta = 0;
initial_states_bus.wdot = 0;

%% Simulink
tic;
```

```
Sim_states = sim("model.slx");
toc;
fprintf("Finished Solving SIMULINK Code\n");

%% Our RK4 Solver
tic;
[t, States] = RK4(@(t, states, wdot) Fdot(t, states, wdot, s0, controls,
    States_Matrix, Controls_Matrix, m, g, I, invI, mg0), t, s0);
toc;
fprintf("Finished Solving MATLAB Code\n");

% %% Benchmark Test
% load('Benchmark_B747_FC5.mat');

%% Results
Sim_states.u=Sim_states.u.Data.';Sim_states.v=Sim_states.v.Data.';Sim_states.w=
    Sim_states.w.Data.';
Sim_states.alpha=Sim_states.alpha.Data.';Sim_states.beta=Sim_states.beta.Data.';
Sim_states.p=Sim_states.p.Data.';Sim_states.q=Sim_states.q.Data.';Sim_states.r=
    Sim_states.r.Data.';
Sim_states.phi=Sim_states.phi.Data.';Sim_states.theta=Sim_states.theta.Data.';
    Sim_states.psi=Sim_states.psi.Data.';
Sim_states.x=Sim_states.x.Data.';Sim_states.y=Sim_states.y.Data.';Sim_states.z=
    Sim_states.z.Data.';

Sim_states.time = Sim_states.time.Data;
```

Listing 2: RK4.m

```
function [t, states] = RK4(f_dot, t, states_0)

states = nan(length(states_0), length(t));
states(:, 1) = states_0;
wdot = 0;

for n = 1:length(t)-1
    h = t(n+1) - t(n);
```

```
9
10      K1 = f_dot(t(n), states(:, n), wdot);
11      wdot = K1(3);
12      K2 = f_dot(t(n)+h/2, states(:, n)+0.5*K1*h, wdot);
13      wdot = K2(3)*.5;
14      K3 = f_dot(t(n)+h/2, states(:, n)+0.5*K2*h, wdot);
15      wdot = K3(3)*.5;
16      K4 = f_dot(t(n)+h, states(:, n)+K3*h, wdot);
17      wdot = K4(3);
18      states(:, n+1) = states(:, n) + h/6*(K1 + 2*K2 + 2*K3 + K4);
19
20  end
21
22  end % endfunction
```

Listing 3: Fdot.m

```
1   function [states_dot] = Fdot(t, states, wdot, states_0, controls, States_Matrix,
        Controls_Matrix, mass, gravity, I, invI, mg0)
2
3   states_dot = nan(size(states));
4
5   u = states(1);
6   v = states(2);
7   w = states(3);
8   p = states(4);
9   q = states(5);
10  r = states(6);
11  phi = states(7);
12  theta = states(8);
13  psi = states(9);
14  x = states(10);
15  y = states(11);
16  z = states(12);
17
18  J = [1, sin(phi)*tan(theta), cos(phi)*tan(theta);
19       0, cos(phi), -sin(phi);
```

```
20      0, sin(phi)/cos(theta), cos(phi)/cos(theta)];
21
22  R = eul2rotm([psi, theta, phi], "ZYX");
23
24  d_states = states - states_0; % perturbation in states around equilibrium point
25  % Compute Total Forces
26  [F, M] = forces_moments(t, d_states, controls, States_Matrix, Controls_Matrix,
        wdot);
27  F = F*mass + R.' * [0;0;mass*gravity] + mg0;
28  M = M.*[I(1,1);I(2,2);I(3,3)] + 0;
29
30  % state space model
31  states_dot(1:3) = 1/mass*F - cross([p; q; r], [u; v; w]); % u,v,w
32  states_dot(4:6) = invI*(M - cross([p; q; r], I*[p; q; r])); % p,q,r
33  states_dot(7:9) = J * [p; q; r]; % phi,theta,psi
34  states_dot(10:12) = R * [u; v; w]; % x,y,z
35
36  end
```

Listing 4: forces_moments.m

```
1  function [F, M] = forces_moments(t, d_states, controls, states_matrix,
       controls_matrix, wdot)
2
3  %% This is force per unit mass and moment per [I(xx/yy/zz)]
4  FM = states_matrix*[d_states(1:6); wdot] + controls_matrix*controls;
5  F = FM(1:3);
6  M = FM(4:6);
7
8  end
```

Listing 5: aircraft_data_reader.m

```
1  % the aerodynamic forces and moments can be expressed as a function of all the
       motion variables [Nelson] page 63
2  % clc
3  % clear
```

```
% close all

%% Excel Sheets Data
% filename = 'Boeing_FC5'; %% put here the location of your excel sheet
filename = 'Lockheed_Jetstar_FC9'; %% put here the location of your excel sheet
% filename = 'Jetstar_FC10'; %% put here the location of your excel sheet

aircraft_data=xlsread(filename,'B2:B61'); %#ok here B2:B61 means read the excel
    sheet from cell B2 to cell B61

% initial conditions
s0 = aircraft_data(4:15);
sdot0 = zeros(12,1);
Vto = sqrt(s0(1)^2 + s0(2)^2 + s0(3)^2);

U0 = s0(1);
V0 = s0(2);
W0 = s0(3);
P0 = s0(4);
Q0 = s0(5);
R0 = s0(6);
PHI0 = s0(7);
THETA0 = s0(8);
PSI0 = s0(9);
X0 = 0;
Y0 = 0;
Z0 = -500; % this is to have a good visualization in flightgear
ALPHA0 = THETA0;

% control actions values
% da = aircraft_data(57);
% dr = aircraft_data(58);
% de = aircraft_data(59);
% dth = aircraft_data(60);
% dc = [ aircraft_data(57:59) * pi/180 ; aircraft_data(60)];

```

```
% gravity, mass & inertia
m = aircraft_data(51);
g = aircraft_data(52);
Ixx = aircraft_data(53);
Iyy = aircraft_data(54);
Izz = aircraft_data(55);
Ixz = aircraft_data(56); Ixy=0; Iyz=0;
I = [Ixx , -Ixy , -Ixz ;
    -Ixy , Iyy , -Iyz ;
    -Ixz , -Iyz , Izz];
invI=inv(I);

% stability derivatives Longitudinal motion
SD_Long = aircraft_data(21:36);
SD_Long_final = SD_Long;

% stability derivatives Lateral motion
SD_Lat_dash = aircraft_data(37:50);
G=1/(1-Ixz^2/(Ixx*Izz));
dash_mat=[G, G*Ixz/Ixx; G*Ixz/Izz, G];
dash_mat_inv=inv(dash_mat);

LB_DASH = SD_Lat_dash(3);
NB_DASH = SD_Lat_dash(4);
LP_DASH = SD_Lat_dash(5);
NP_DASH = SD_Lat_dash(6);
LR_DASH = SD_Lat_dash(7);
NR_DASH = SD_Lat_dash(8);
LDA_DASH = SD_Lat_dash(11);
NDA_DASH = SD_Lat_dash(12);
LDR_DASH = SD_Lat_dash(13);
NDR_DASH = SD_Lat_dash(14);

B=dash_mat\[LB_DASH; NB_DASH];
P=dash_mat\[LP_DASH; NP_DASH];
R=dash_mat\[LR_DASH; NR_DASH];
```

```
75  DA=dash_mat\[LDA_DASH; NDA_DASH];
76  DR=dash_mat\[LDR_DASH;NDR_DASH];
77
78  YDA=SD_Lat_dash(9)*Vto;
79  YDR=SD_Lat_dash(10)*Vto;
80  SD_Lat=[SD_Lat_dash(1);SD_Lat_dash(2);B(1);B(2);P(1);P(2);R(1);R(2);YDA;YDR;DA(1)
        ;DA(2);DR(1);DR(2)];
81  % YV YB LB NB LP NP LR NR Y_DA Y_DR LDA NDA LDR NDR
82  SD_Lat_final = [ SD_Lat(1) ; SD_Lat(3) ; SD_Lat(4) ; SD_Lat(5) ; SD_Lat(6) ;...
83                   SD_Lat(7) ; SD_Lat(8) ; SD_Lat(9:10) ; SD_Lat(11) ; SD_Lat(12) ;...
84                    SD_Lat(13) ; SD_Lat(14) ];
85
86  % initial gravity force
87  mg0 = m*g * [ sin(s0(8)) ; -cos(s0(8))*sin(s0(7)) ; -cos(s0(8))*cos(s0(7)) ];
88
89  XU = SD_Long_final(1);
90  ZU = SD_Long_final(2);
91  MU = SD_Long_final(3);
92  XW = SD_Long_final(4);
93  ZW = SD_Long_final(5);
94  MW = SD_Long_final(6);
95  ZWD = SD_Long_final(7);
96  ZQ = SD_Long_final(8);
97  MWD = SD_Long_final(9);
98  MQ = SD_Long_final(10);
99  XDE = SD_Long_final(11);
100 ZDE = SD_Long_final(12);
101 MDE = SD_Long_final(13);
102 XDTH = SD_Long_final(14);
103 ZDTH = SD_Long_final(15);
104 MDTH = SD_Long_final(16);
105
106 YV = SD_Lat_final(1);
107 LB = SD_Lat_final(2);
108 NB = SD_Lat_final(3);
109 LP = SD_Lat_final(4);
```

```
NP = SD_Lat_final(5);
LR = SD_Lat_final(6);
NR = SD_Lat_final(7);
YDA = SD_Lat_final(8);
YDR = SD_Lat_final(9);
LDA = SD_Lat_final(10);
NDA = SD_Lat_final(11);
LDR = SD_Lat_final(12);
NDR = SD_Lat_final(13);

LV_DASH = LB_DASH / Vto;
NV_DASH = NB_DASH / Vto;
LV = LB / Vto;
NV = NB / Vto;

YP = 0;
YR = 0;

States_Matrix = [XU, 0, XW, 0, 0, 0, 0;
                 0, YV, 0, YP, 0, YR, 0;
                ZU, 0, ZW, 0, ZQ, 0, ZWD;
                 0, LV, 0, LP, 0, LR, 0;
                MU, 0, MW, 0, MQ, 0, MWD;
                 0, NV, 0, NP, 0, NR, 0];

Controls_Matrix = [0, XDE, XDTH, 0;
                   YDA, 0, 0, YDR;
                   0, ZDE, ZDTH, 0;
                   LDA, 0, 0 LDR;
                   0, MDE, MDTH, 0;
                   NDA, 0, 0, NDR];


rho = 5.87e-4; % (slugs/ft^3)
CD_alpha = 0.7;
CD = CD_alpha * ALPHA0;
```

```
Qinf = 117; % (PSF)
S = 542.5; % (FT^2)
Thrust0 = 0.5*rho*Vto^2*S*CD; % (LBF)
Thrustmax = 14800; % (LBF)
```

Listing 6: pre_simulink.m

```
clear
close all
clc

%% Problem Setup
aircraft_data_reader;

%% Linearized Matrices
A_longitudinal=[XU,XW,-W0,-g*cos(THETA0), 0, 0;...
       ZU/(1-ZWD),ZW/(1-ZWD),(ZQ+U0)/(1-ZWD),-g*sin(THETA0)/(1-ZWD), 0, 0;...
       MU+MWD*ZU/(1-ZWD),MW+MWD*ZW/(1-ZWD),MQ+MWD*(ZQ+U0)/(1-ZWD),-MWD*g*sin(
           THETA0)/(1-ZWD), 0, 0;...
       0,0,1,0, 0, 0;
       cos(THETA0), sin(THETA0), 0, -U0*sin(THETA0), 0, 0;
       -sin(THETA0), cos(THETA0), 0, -U0*cos(THETA0), 0, 0];
B_longitudinal=[XDE,XDTH;
       ZDE/(1-ZWD),ZDTH/(1-ZWD);
       MDE+MWD*ZDE/(1-ZWD),MDTH+MWD*ZDTH/(1-ZWD);
       0,0;
       0,0;
       0,0];
C_longitudinal=eye(size(A_longitudinal));
D_longitudinal=zeros(size(B_longitudinal));

% %% Short Period
% A_longitudinal_sp = [A_longitudinal(2,2),A_longitudinal(2,3);A_longitudinal
    (3,2),A_longitudinal(3,3)];
% B_longitudinal_sp = [B_longitudinal(2,1),B_longitudinal(2,2);B_longitudinal
    (3,1),B_longitudinal(3,2)];
% C_longitudinal_sp = eye(2);
```

```
% D_longitudinal_sp = zeros(2,2);
%
% %% Long Period
% A_longitudinal_lp = [A_longitudinal(1,1),A_longitudinal(1,4);-ZU/(ZQ+U0) ,0];
% B_longitudinal_lp = [B_longitudinal(1,1),B_longitudinal(1,2);-ZDE/(ZQ+U0),-ZDTH
   /(ZQ+U0)];
% C_longitudinal_lp = eye(2);
% D_longitudinal_lp = zeros(2,2);

%% Lateral Dynamics
A_lateral = [YV , (W0+YP) , -U0+YR, g*cos(THETA0), 0, 0;
       LV_DASH , LP_DASH , LR_DASH , 0 , 0, 0;
       NV_DASH , NP_DASH , NR_DASH , 0 , 0, 0;
       0 , 1 , tan(THETA0) , 0 , 0, 0;
       0 , 0 , 1/cos(THETA0) ,0 ,0, 0;
       1, 0, 0, 0, U0*cos(THETA0), 0];
B_lateral = [YDA , YDR;
        LDA_DASH , LDR_DASH;
        NDA_DASH , NDR_DASH;
        0 , 0;
        0 , 0;
        0, 0]; % check this, might need fixing
C_lateral=eye(size(A_lateral));
D_lateral=zeros(size(B_lateral));

% %% 1DOF Roll Mode
% A_lateral_1DOF = LP_DASH;
% B_lateral_1DOF = LDA_DASH;
% C_lateral_1DOF = eye(1);
% D_lateral_1DOF = zeros(1,1);
%
% %% 2DOF Dutch Roll Mode
% A_lateral_2DOF = [YV ,-U0+YR; %- tan(THETA0)*(YP+W0)/U0
% NV_DASH , NR_DASH];
% B_lateral_2DOF = [YDA , YDR; NDA_DASH , NDR_DASH;];
% C_lateral_2DOF = eye(2);
```

```
% D_lateral_2DOF = zeros(2,2);
%
% %% 3DOF Dutch Roll Mode
% A_lateral_3DOF_DR = [YV, 0, -1; LV_DASH, LP_DASH, 0; NV_DASH, 0, NR_DASH];
% B_lateral_3DOF_DR = [YDA, YDR; LDA_DASH, LDR_DASH; NDA_DASH, NDR_DASH];
% C_lateral_3DOF_DR = eye(size(A_lateral_3DOF_DR));
% D_lateral_3DOF_DR = zeros(size(B_lateral_3DOF_DR));
%
% %% 3DOF Spiral Mode
% A_lateral_3DOF_SP = [LP_DASH, LR_DASH, 0; NP_DASH, NR_DASH, 0; 1, 0, 0];
% B_lateral_3DOF_SP = [LDR_DASH; NDR_DASH; 0];
% C_lateral_3DOF_SP = eye(size(A_lateral_3DOF_SP));
% D_lateral_3DOF_SP = zeros(size(B_lateral_3DOF_SP));

%% Full Linearized Equations
% A_mat = [A_longitudinal, zeros(size(A_longitudinal,1),size(A_lateral,2));
% zeros(size(A_lateral,1),size(A_longitudinal,2)), A_lateral];
% B_mat = [B_longitudinal, zeros(size(B_longitudinal,1),2);
% zeros(size(B_lateral,1), 2), B_lateral];
% C_mat = eye(size(A_mat));
% D_mat = zeros(size(B_mat));

%% Frequency Domain
long_SS = ss(A_longitudinal,B_longitudinal,C_longitudinal,D_longitudinal);
long_TF = tf(long_SS);
u_de = long_TF(1, 1);
w_de = long_TF(2,1);
q_de = long_TF(3,1);
theta_de = long_TF(4,1);
u_dth = long_TF(1,2);
w_dth = long_TF(2,2);
q_dth = long_TF(3,2);
theta_dth = long_TF(4,2);

lat_SS = ss(A_lateral,B_lateral,C_lateral,D_lateral);
lat_TF = tf(lat_SS);
```

```
v_da = lat_TF(1, 2);
p_da = lat_TF(2,1);
r_da = lat_TF(3,1);
phi_da = lat_TF(4,1);
v_dr = lat_TF(1,2);
p_dr = lat_TF(2,2);
r_dr = lat_TF(3,2);
phi_dr = lat_TF(4,2);

% run('frequency_domain_long.mlx');
% run('frequency_domain_lat.mlx');

%% Longitudinal Autopilot
servo = tf(10,[1 10]);
integrator = tf(1,[1 0]);
differentiator = tf([1 0],1);
engine_timelag = tf(0.1,[1 0.1]);

%% Elevator from Pitch
OL_theta_thetaCMD = -servo*theta_de;

load("./Autopilot Data/ElevatorFromPitch.mat");
K_ElevFromPitch = tf(C1_ElevFromPitch).num{1}(1);
Ki_ElevFromPitch = tf(C1_ElevFromPitch).num{1}(2);
Kd_ElevFromPitch = tf(C2_ElevFromPitch).num{1}(1);

%% Throttle from Airspeed
OL_u_uCMD = u_dth * engine_timelag * servo;
load("./Autopilot Data/ThrottleFromAirspeed.mat");

%% Pitch from Altitude Altitude
h_theta = -tf(1, [1,0]) * (w_de/theta_de - U0);
OL_h_hCMD = h_theta * CL_theta_thetaCMD;
load("./Autopilot Data/PitchFromAltitude.mat");

%% Yaw Damper
```

```
OL_r_rCMD = servo * r_dr;
load("./Autopilot Data/yawDamper.mat");

%% Roll Controller
OL_coordinated_ss = feedback(series(append(1,servo), lat_SS, [1,2], [1,2]),
    C_yawDamper, 2, 3, 1);
OL_coordinated_tf = tf(OL_coordinated_ss);
OL_phi_phiCMD = minreal(servo * OL_coordinated_tf(4, 1));
load("./Autopilot Data/AileronFromRoll.mat");

%% Simulation Time
tfinal = 20; % final time
dt = 0.01;

%% Bus Creation
Bus_Creator;
initial_states_bus.u = s0(1);
initial_states_bus.v = s0(2);
initial_states_bus.w = s0(3);
initial_states_bus.p = s0(4);
initial_states_bus.q = s0(5);
initial_states_bus.r = s0(6);
initial_states_bus.phi = s0(7);
initial_states_bus.theta = s0(8);
initial_states_bus.psi = s0(9);
initial_states_bus.x = s0(10);
initial_states_bus.y = s0(11);
initial_states_bus.z = s0(12);
initial_states_bus.alpha = s0(8);
initial_states_bus.beta = 0;
initial_states_bus.wdot = 0;

%% Run Simulink
tic;
out = sim("model.slx");
toc;
```

```
fprintf("Finished Solving SIMULINK Code\n");

% save_system("model.slx","model_19a.slx", "ExportToVersion","R2019a");

%% Plot
% fig1 = figure;
% plot(out.time.Data, rad2deg(out.theta_lin.Data), 'LineWidth', 2);
% hold on; grid on;
% % plot(out.time.Data, rad2deg(out.theta.Data), 'LineWidth', 2);
% plot(out.time.Data, rad2deg(out.theta_CMD.Data), 'LineWidth', 2);
% plot(out.time.Data, rad2deg(out.another_theta.Data), 'LineWidth', 2);
% xlabel("t [s]"); ylabel("\theta [deg]");
% legend("\theta_{linear}", "\theta_{CMD}", "\theta_{dE}", 'Location', 'best');
% title("$Pitch Angle$ $\theta$", 'interpreter', 'latex');
%
% fig2 = figure;
% plot(out.time.Data, out.u_lin.Data, 'LineWidth', 2);
% hold on; grid on;
% % plot(out.time.Data, out.u.Data, 'LineWidth', 2);
% plot(out.time.Data, out.u_CMD.Data, 'LineWidth', 2);
% plot(out.time.Data, out.another_u.Data, 'LineWidth', 2);
% xlabel("t [s]"); ylabel("u [ft/s]");
% title("$u$", 'interpreter', 'latex');
% legend("u_{linear}", "u_{CMD}", "u_{dE}", 'Location', 'best')
%
% fig3 = figure;
% plot(out.time.Data, rad2deg(out.de.Data), 'LineWidth', 2);
% hold on; grid on;
% xlabel("t [s]"); ylabel("dE [deg]");
% title("$dE$", 'interpreter', 'latex');
%
% fig4 = figure;
% plot(out.time.Data, out.dth.Data, 'LineWidth', 2);
% hold on; grid on;
% xlabel("t [s]"); ylabel("dTH");
% title("$dTH$", 'interpreter', 'latex');
```

```
%
% fig5 = figure;
% plot(out.time.Data, rad2deg(out.phi_lin.Data), 'LineWidth', 2);
% hold on; grid on;
% plot(out.time.Data, rad2deg(out.phi_CMD.Data), 'LineWidth', 2);
% xlabel("t [s]"); ylabel("$\phi [deg]$", 'interpreter', 'latex', "FontSize", 15)
    ;
% legend("\phi_{linear}", "\phi_{CMD}", 'Location', 'best');
% title("$Roll Angle$ $\phi$", 'interpreter', 'latex', "FontSize", 15);
%
% fig6 = figure;
% plot(out.time.Data, rad2deg(out.psi_lin.Data), 'LineWidth', 2);
% hold on; grid on;
% plot(out.time.Data, rad2deg(out.psi_CMD.Data), 'LineWidth', 2);
% xlabel("t [s]"); ylabel("$\psi$ $[^\circ]$", 'interpreter', 'latex', "FontSize
    ", 15);
% title("$\psi$ $[^\circ]$", 'interpreter', 'latex', "FontSize", 15);
% legend("\psi_{linear}", "\psi_{CMD}", 'Location', 'best')
%
% fig7 = figure;
% plot(out.time.Data, rad2deg(out.da.Data), 'LineWidth', 2);
% hold on; grid on;
% xlabel("t [s]"); ylabel("$\delta_a$ [deg]", 'interpreter', 'latex', "FontSize",
     15);
% title("$\delta_a$", 'interpreter', 'latex', "FontSize", 15);
%
% fig8 = figure;
% plot(out.time.Data, rad2deg(out.dr.Data), 'LineWidth', 2);
% hold on; grid on;
% xlabel("t [s]"); ylabel("$\delta_r$", 'interpreter', 'latex', "FontSize", 15);
% title("$\delta_r$", 'interpreter', 'latex', "FontSize", 15);
%
% fig9 = figure;
% plot(out.time.Data, rad2deg(out.beta_lin.Data), 'LineWidth', 2);
% hold on; grid on;
% xlabel("t [s]"); ylabel("$beta$ $[^\circ]$", 'interpreter', 'latex', "FontSize
```

```
    ", 15);
% title("$\beta$ $[^\circ]$", 'interpreter', 'latex', "FontSize", 15);

% saveas(fig1, "theta", 'svg')
% saveas(fig2, "u", 'svg')
% saveas(fig3, "dE", 'svg')
% saveas(fig4, "dTH", 'svg')
% saveas(fig5, "phi", 'svg')
% saveas(fig6, "psi", 'svg')
% saveas(fig7, "dA", 'svg')
% saveas(fig8, "dR", 'svg')
% saveas(fig9, "beta", 'svg')
% somefig = figure;
% subplot(6,1,1);
% grid on; hold on;
% plot(out.time.Data, rad2deg(out.phi_lin.Data), 'k', "LineWidth",2);
% xlabel("$t$ [s]", 'Interpreter', 'latex');
% ylabel("$\phi^\circ$", 'Interpreter', 'latex');
% subplot(6,1,2);
% grid on; hold on;
% plot(out.time.Data, rad2deg(out.p_lin.Data), 'k', "LineWidth",2);
% xlabel("$t$ [s]", 'Interpreter', 'latex');
% ylabel("$p$ $[^\circ/s]$", 'Interpreter', 'latex');
% subplot(6,1,3);
% grid on; hold on;
% plot(out.time.Data, rad2deg(out.r.Data), 'k', "LineWidth",2);
% xlabel("$t$ [s]", 'Interpreter', 'latex');
% ylabel("$r$ $[^\circ/s]$", 'Interpreter', 'latex');
% subplot(6,1,4);
% grid on; hold on;
% plot(out.time.Data, rad2deg(out.beta_lin.Data), 'k', "LineWidth",2);
% xlabel("$t$ [s]", 'Interpreter', 'latex');
% ylabel("$\beta$ $[^\circ]$", 'Interpreter', 'latex');
% subplot(6,1,5);
% grid on; hold on;
% plot(out.time.Data, rad2deg(out.dr_lin.Data), 'k', "LineWidth",2);
```

```
% xlabel("$t$ [s]", 'Interpreter', 'latex');
% ylabel("$d_R$ $[^\circ]$", 'Interpreter', 'latex');
% subplot(6,1,6);
% grid on; hold on;
% plot(out.time.Data, rad2deg(out.da_lin.Data), 'k', "LineWidth",2);
% xlabel("$t$ [s]", 'Interpreter', 'latex');
% ylabel("$d_A$ $[^\circ]$", 'Interpreter', 'latex');
% saveas(somefig, "yawDamperTest", 'png')

% fig10 = figure;
% subplot(1,2,1);
% grid on; hold on;
% plot(out.time.Data, rad2deg(out.theta_lin.Data), 'LineWidth', 2);
% plot(out.time.Data, rad2deg(out.theta_CMDlin.Data).*ones(size(out.time.Data)),
    'LineWidth', 2);
% xlabel("t [s]"); ylabel("\theta [deg]");
% legend("\theta_{linear}", "\theta_{CMD}", 'Location', 'best');
% title("$Pitch Angle$ $\theta$ Linear Autopilot", 'interpreter', 'latex');
% subplot(1,2,2);
% grid on; hold on;
% plot(out.time.Data, rad2deg(out.theta.Data), 'LineWidth', 2);
% plot(out.time.Data, rad2deg(out.theta_CMD.Data).*ones(size(out.time.Data)), '
    LineWidth', 2);
% xlabel("t [s]"); ylabel("\theta [deg]");
% legend("\theta", "\theta_{CMD}", 'Location', 'best');
% title("$Pitch Angle$ $\theta$ Nonlinear Autopilot", 'interpreter', 'latex');
%
% fig11 = figure;
% subplot(1,2,1);
% grid on; hold on;
% plot(out.time.Data, out.altitude_lin.Data, 'LineWidth', 2);
% % plot(out.time.Data, out.h_CMDlin.Data, 'LineWidth', 2);
% xlabel("t [s]"); ylabel("h [ft]");
% % legend("h_{linear}", "h_{CMD}", 'Location', 'best');
% title("$Altitude$ $h$ Linear Autopilot", 'interpreter', 'latex');
% subplot(1,2,2);
```

```
% grid on; hold on;
% plot(out.time.Data, out.altitude.Data, 'LineWidth', 2);
% % plot(out.time.Data, out.h_CMD.Data, 'LineWidth', 2);
% xlabel("t [s]"); ylabel("h [ft]");
% % legend("h_{nonlinear}", "h_{CMD}", 'Location', 'best');
% title("$Altitude$ $h$ Nonlinear Autopilot", 'interpreter', 'latex');
%
% fig12 = figure;
% subplot(1,2,1);
% grid on; hold on;
% plot(out.time.Data, rad2deg(out.phi_lin.Data), 'LineWidth', 2);
% plot(out.time.Data, rad2deg(out.phi_CMDlin.Data), 'LineWidth', 2);
% xlabel("$t$ [s]", 'Interpreter', 'latex', 'FontSize',12); ylabel("$\phi$ [deg
    ]", 'Interpreter', 'latex', 'FontSize',12);
% legend("\phi_{linear}", "\phi_{CMD}", 'Location', 'best');
% title("$Roll Angle$ $\phi$ Linear Autopilot", 'interpreter', 'latex');
% subplot(1,2,2);
% grid on; hold on;
% plot(out.time.Data, rad2deg(out.phi.Data), 'LineWidth', 2);
% plot(out.time.Data, rad2deg(out.phi_CMD.Data), 'LineWidth', 2);
% xlabel("$t$ [s]", 'Interpreter', 'latex', 'FontSize',12); ylabel("$\phi$ [deg
    ]", 'Interpreter', 'latex', 'FontSize',12);
% legend("\phi_{nonlinear}", "\phi_{CMD}", 'Location', 'best');
% title("$Roll Angle$ $\phi$ Nonlinear Autopilot", 'interpreter', 'latex');
%
% fig13 = figure;
% subplot(1,2,1);
% grid on; hold on;
% plot(out.time.Data, rad2deg(out.psi_lin.Data), 'LineWidth', 2);
% plot(out.time.Data, rad2deg(out.psi_CMDlin.Data), 'LineWidth', 2);
% xlabel("$t$ [s]", 'Interpreter', 'latex', 'FontSize',12); ylabel("$\psi$ [deg
    ]", 'Interpreter', 'latex', 'FontSize',12);
% legend("\psi_{linear}", "\psi_{CMD}", 'Location', 'best');
% title("$Heading Angle$ $\psi$ Linear Autopilot", 'interpreter', 'latex');
% subplot(1,2,2);
% grid on; hold on;
```

```
341 % plot(out.time.Data, rad2deg(out.psi.Data), 'LineWidth', 2);
342 % plot(out.time.Data, rad2deg(out.psi_CMD.Data), 'LineWidth', 2);
343 % xlabel("$t$ [s]", 'Interpreter', 'latex', 'FontSize',12); ylabel("$\psi$ [deg
      ]", 'Interpreter', 'latex', 'FontSize',12);
344 % legend("\psi_{nonlinear}", "\psi_{CMD}", 'Location', 'best');
345 % title("$Heading Angle$ $\psi$ Nonlinear Autopilot", 'interpreter', 'latex');
346 %
347 % fig14 = figure;
348 % subplot(1,2,1);
349 % grid on; hold on;
350 % plot(out.time.Data, rad2deg(out.beta_lin.Data), 'LineWidth', 2);
351 % xlabel("$t$ [s]", 'Interpreter', 'latex', 'FontSize',12); ylabel("$\beta$ [deg
      ]", 'Interpreter', 'latex', 'FontSize',12);
352 % title("$Sideslip Angle$ $\beta$ Linear Autopilot", 'interpreter', 'latex');
353 % subplot(1,2,2);
354 % grid on; hold on;
355 % plot(out.time.Data, rad2deg(out.beta.Data), 'LineWidth', 2);
356 % xlabel("$t$ [s]", 'Interpreter', 'latex', 'FontSize',12); ylabel("$\beta$ [deg
      ]", 'Interpreter', 'latex', 'FontSize',12);
357 % title("$Sideslip Angle$ $\beta$ Nonlinear Autopilot", 'interpreter', 'latex');
358
359 % fig15 = figure;
360 % subplot(1,2,1);
361 % grid on; hold on;
362 % plot(out.time.Data, rad2deg(out.de_lin.Data), 'LineWidth', 2);
363 % xlabel("$t$ [s]", 'Interpreter', 'latex', 'FontSize',12); ylabel("$\delta_e$ [
      deg]", 'Interpreter', 'latex', 'FontSize',12);
364 % title("$Elevator Response$ $\delta_e$ Linear Autopilot", 'interpreter', 'latex
      ');
365 % subplot(1,2,2);
366 % grid on; hold on;
367 % plot(out.time.Data, rad2deg(out.de.Data), 'LineWidth', 2);
368 % xlabel("$t$ [s]", 'Interpreter', 'latex', 'FontSize',12); ylabel("$\delta_e$ [
      deg]", 'Interpreter', 'latex', 'FontSize',12);
369 % title("$Elevator Action$ $\delta_e$ Nonlinear Autopilot", 'interpreter', 'latex
      ');
```

```
370 %
371 % fig16 = figure;
372 % subplot(1,2,1);
373 % grid on; hold on;
374 % plot(out.time.Data, out.dth_lin.Data, 'LineWidth', 2);
375 % xlabel("$t$ [s]", 'Interpreter', 'latex', 'FontSize',12); ylabel("$\delta_{th}$
        [lbf]", 'Interpreter', 'latex', 'FontSize',12);
376 % title("$Thrust Action$ $\delta_{th}$ Linear Autopilot", 'interpreter', 'latex')
        ;
377 % subplot(1,2,2);
378 % grid on; hold on;
379 % plot(out.time.Data, out.dth.Data, 'LineWidth', 2);
380 % xlabel("$t$ [s]", 'Interpreter', 'latex', 'FontSize',12); ylabel("$\delta_{th}$
        [lbf]", 'Interpreter', 'latex', 'FontSize',12);
381 % title("$Thrust Action$ $\delta_{th}$ Nonlinear Autopilot", 'interpreter', '
        latex');
382
383 % fig17 = figure;
384 % subplot(1,2,1);
385 % grid on; hold on;
386 % plot(out.time.Data, rad2deg(out.da_lin.Data), 'LineWidth', 2);
387 % xlabel("$t$ [s]", 'Interpreter', 'latex', 'FontSize',12); ylabel("$\delta_a$ [
        deg]", 'Interpreter', 'latex', 'FontSize',12);
388 % title("$Aileron Response$ $\delta_a$ Linear Autopilot", 'interpreter', 'latex')
        ;
389 % subplot(1,2,2);
390 % grid on; hold on;
391 % plot(out.time.Data, rad2deg(out.da.Data), 'LineWidth', 2);
392 % xlabel("$t$ [s]", 'Interpreter', 'latex', 'FontSize',12); ylabel("$\delta_a$ [
        deg]", 'Interpreter', 'latex', 'FontSize',12);
393 % title("$Aileron Action$ $\delta_a$ Nonlinear Autopilot", 'interpreter', 'latex
        ');
394 %
395 % fig18 = figure;
396 % subplot(1,2,1);
397 % grid on; hold on;
```

```
% plot(out.time.Data, rad2deg(out.dr_lin.Data), 'LineWidth', 2);
% xlabel("$t$ [s]", 'Interpreter', 'latex', 'FontSize',12); ylabel("$\delta_r$ [deg]", 'Interpreter', 'latex', 'FontSize',12);
% title("$Rudder Response$ $\delta_r$ Linear Autopilot", 'interpreter', 'latex');
% subplot(1,2,2);
% grid on; hold on;
% plot(out.time.Data, rad2deg(out.dr.Data), 'LineWidth', 2);
% xlabel("$t$ [s]", 'Interpreter', 'latex', 'FontSize',12); ylabel("$\delta_r$ [deg]", 'Interpreter', 'latex', 'FontSize',12);
% title("$Rudder Action$ $\delta_r$ Nonlinear Autopilot", 'interpreter', 'latex');

% fig19 = figure;
% subplot(1,2,1);
% grid on; hold on;
% plot(out.time.Data, rad2deg(out.gamma_lin.Data), 'LineWidth', 2);
% xlabel("$t$ [s]", 'Interpreter', 'latex', 'FontSize',12); ylabel("$\gamma$ [deg]", 'Interpreter', 'latex', 'FontSize',12);
% title("$Flight Path Angle$ $\gamma$ Linear Autopilot", 'interpreter', 'latex');
% subplot(1,2,2);
% grid on; hold on;
% plot(out.time.Data, rad2deg(out.gamma.Data), 'LineWidth', 2);
% xlabel("$t$ [s]", 'Interpreter', 'latex', 'FontSize',12); ylabel("$\gamma$ [deg]", 'Interpreter', 'latex', 'FontSize',12);
% title("$Flight Path Angle$ $\gamma$ Nonlinear Autopilot", 'interpreter', 'latex');

% fig20 = figure;
% subplot(1,2,1);
% grid on; hold on;
% plot(out.time.Data, out.u_lin.Data, 'LineWidth', 2);
% plot(out.time.Data, out.u_CMDlin.Data, 'LineWidth', 2);
% xlabel("t [s]"); ylabel("u [ft/s]");
% legend("u_{linear}", "u_{CMD}", 'Location', 'best');
% title("$Velocity$ $u$ Linear Autopilot", 'interpreter', 'latex');
% subplot(1,2,2);
```

```
% grid on; hold on;
% plot(out.time.Data, out.u.Data, 'LineWidth', 2);
% plot(out.time.Data, out.u_CMD.Data, 'LineWidth', 2);
% xlabel("t [s]"); ylabel("u [ft/s]");
% legend("u_{nonlinear}", "u_{CMD}", 'Location', 'best');
% title("$Velocity$ $h$ Nonlinear Autopilot", 'interpreter', 'latex');

% saveas(fig10, "theta", 'svg')
% saveas(fig11, "h", 'svg')
% saveas(fig12, "phi", 'svg')
% saveas(fig13, "psi", 'svg')
% saveas(fig14, "beta", 'svg')
% saveas(fig15, "dE", 'svg')
% saveas(fig16, "dTH", 'svg')
% saveas(fig17, "dA", 'svg')
% saveas(fig18, "dR", 'svg')
% saveas(fig19, "gamma", 'svg')
% saveas(fig20, "u", 'svg')
```

References

[1] Smith H. Aircraft Flight Mechanics; 2023. Available from: https://aircraftflightmechanics.com/EoMs/EulerTransforms.html.

[2] Wikipedia. Chebyshev Distance;. Available from: https://en.wikipedia.org/wiki/Chebyshev_distance.

[3] Wikipedia. Correlation Coefficient;. Available from: https://en.wikipedia.org/wiki/Pearson_correlation_coefficient.

[4] Etkin B, Reid LD. Dynamics of Flight: Stability and Control. 3rd ed. New York: John Wiley & Sons; 1996.

[5] Blakelock JH. Automatic Control of Aircraft and Missiles. 2nd ed. wiley; 1991.

[6] Navigation U. Autopilot Definition;. Available from: https://www.uavnavigation.com/products/autopilot-definition.

[7] SpinningWing. Helicopter SAS and SCAS;. Available from: https://www.spinningwing.com/the-helicopter/sas-scas.

[8] Polak RF. SAS, Autopilots and Flight Directors. Helicopter Maintenance Magazine; 2010. Available from: https://helicoptermaintenancemagazine.com/article/sas-autopilots-and-flight-directors-what%E2%80%99s-name.

[9] Skybrary. Fly-By-Wire;. Available from: https://skybrary.aero/articles/fly-wire.

[10] Ahmed W. Eng. Wessam Tutorials. youtube; 2022. Available from: https://youtube.com/playlist?list=PL1YGQ3yUkoJliWUON8XPtA-jS682yt8_r&si=nDNXhPnH1RijAkXC.

[11] Elewah MG. Eng. Mahmoud Elewah Tutorials. youtube; 2024. Available from: https://youtu.be/PYLd5rSsy_g?si=Xihlcl6dcln6n_4M.

[12] Wikipedia. Minkowski Distance;. Available from: https://en.wikipedia.org/wiki/Minkowski_distance.