On Point AERONAUTICS

A Simple System for Simulating the Flight Path of an Aircraft

Aerodynamics Stability and Control

Bret L VanHorne
3-15-2022

Table of Contents

The MATLAB and Simulink files for this simulation system can be downloaded from the On Point Aeronautics website at: https://onpointaeronautics.com .

Introduction

One of the most important responsibilities of an aerodynamics stability and control engineer is to simulate the flight path of an aircraft executing stability maneuvers. The simulations provide information for the flight crew and hopefully bring to light any stability problems.

For this type of simulation, the engineer is primarily concerned with the natural aerodynamic stability and dynamics of the airframe itself. Simulink is the most common simulation tool. There are several readily available Simulink aircraft simulations and templates, but they tend to focus more on the control systems and often use third-party visualizations that allow a person to "fly" an aircraft on the computer. These simulations carry an excess amount of "software bloat" that the stability and control engineer does not need or want.

In the following pages, a simplified Simulink system is described that was designed and built expressly for the aerodynamics stability and control engineer. The system will trim an aircraft and fly prescribed flight maneuvers using direct inputs to the control surfaces and direct commands for thrust.

A Simple Simulink System

A folder-based MATLAB project was used to organize the scripts and models needed for the simulation.

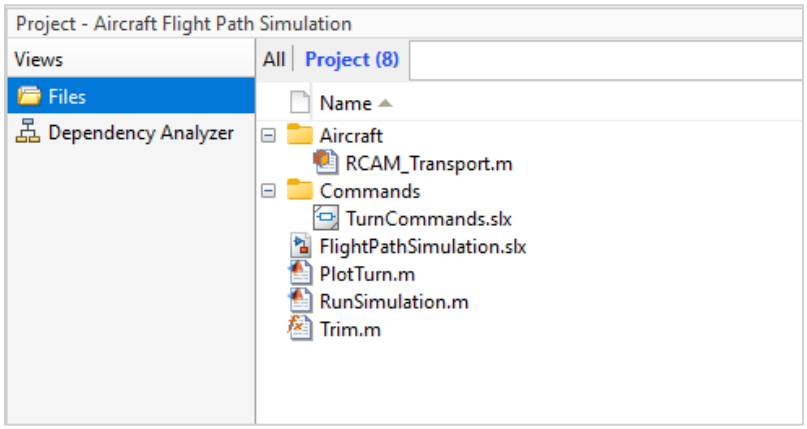

Figure 1. Project Organization.

The system was designed so that all the information specific to an aircraft could be encapsulated in a separate MATLAB class. This class provides the aircraft constants, a method for calculating the aerodynamic coefficients, and a method for calculating the propulsive forces and moments. It also provides a method to limit control surface deflections and enable control surface mixes (i.e. aileron to rudder) (see example in Figure 2).

```
classdef RCAM_Transport
    % Model of a transport aircraft.
    % Based on the Research Civil Aircraft Model (RCAM) from the Group for
    % Aeronautical Research and Technology in Europe (GARTEUR)(1995).
    % @author: Bret L VanHorne, 3/2022
    % Copyright (C) 2022 On Point Aeronautics. All rights reserved.

    properties(Constant) ...

    methods

        function [Controls] = ControlRigging(obj, controls) ...

        function [Coeff_s] = AeroCoefficients(obj, conditions, states, controls) ...

        function [Thrust_b] = EngineThrust(obj, throttle, conditions) ...

    end
end
```

Figure 2. Example Aircraft Model.

This arrangement allows flexibility as well as portability for many engineering tasks. Having the aircraft modeled as a MATLAB class allows the engineer to programmatically define the non-linear aspects of the aerodynamics and control surface interdependencies. It permits calculations outside of Simulink for trim points and other items of interest. This is especially useful when there is a need to calculate a wide range of trim points. Such is the case when analyzing the static longitudinal stability or minimum control speeds for an aircraft.

Similarly, the commands and control laws are encapsulated in a referenced subsystem that can be swapped out. As work progresses, the stability and control engineer can collect a library of aircraft models and command subsystems that can be mixed and matched.

Top-Level System View

The top level of the system is organized into four subsystems. Signal buses are used to flow state data from the six-degree-of-freedom (6-DOF) integration subsystem back to the subsystems for flight conditions, control inputs, and aircraft forces and moments (see Figure 3).

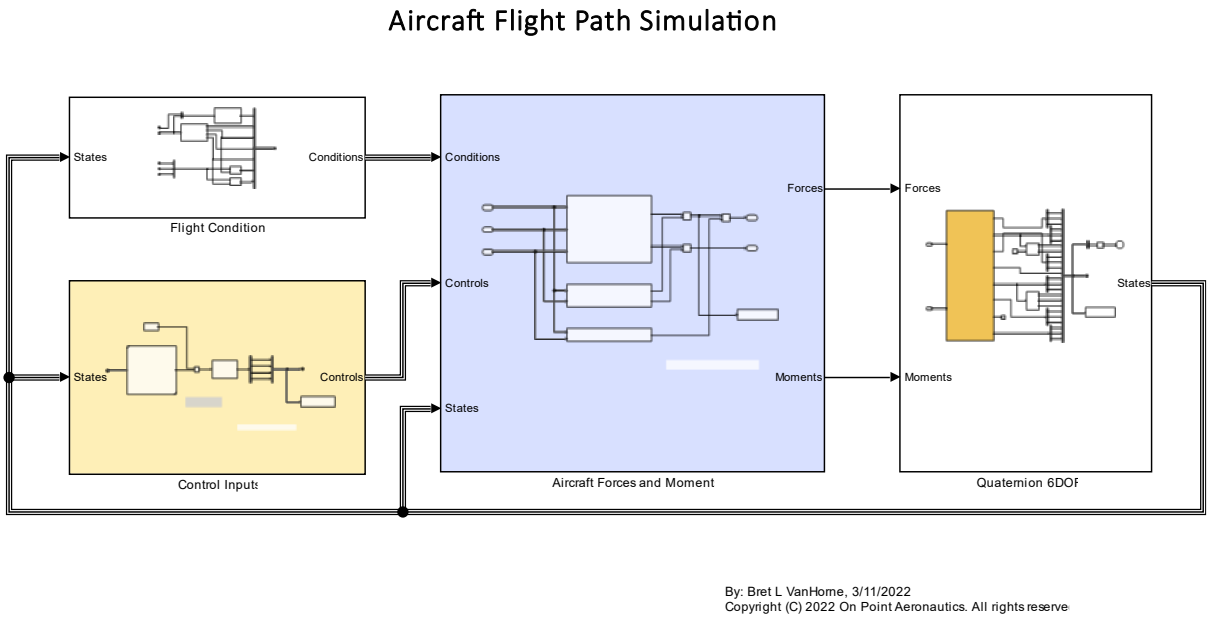

Figure 3. Top-level Model.

Flight Conditions

The flight conditions are computed using blocks from the Aerospace Blockset. Atmospheric quantities are derived from the 1976 COESA-extended U.S. Standard Atmosphere. The acceleration of gravity is derived from the WGS 84 Taylor Series model. The Mach number and Dynamic Pressure are also computed. The output of this subsystem is collected into a signal bus.

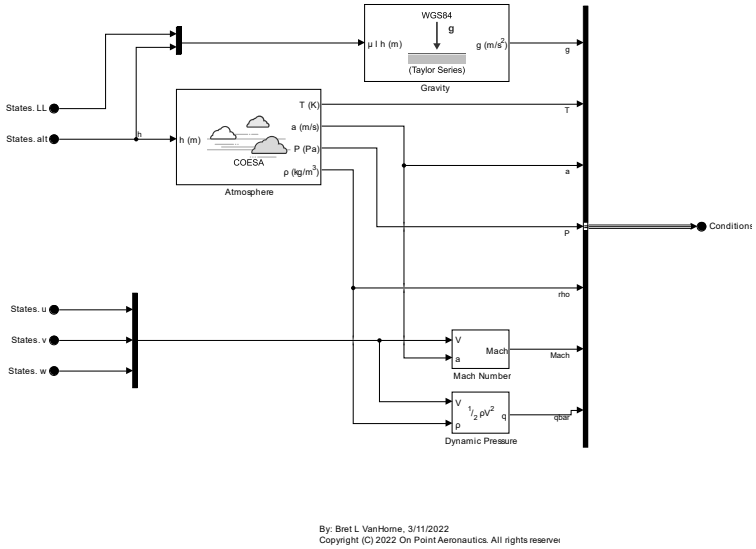

Figure 4. Flight Conditions.

Aircraft Forces and Moments

The subsystem that calculates the aircraft forces and moments is divided into three more subsystems that calculate the aerodynamic forces and moments, the propulsive forces and moments, and the force of gravity acting upon the center of gravity of the aircraft (CG).

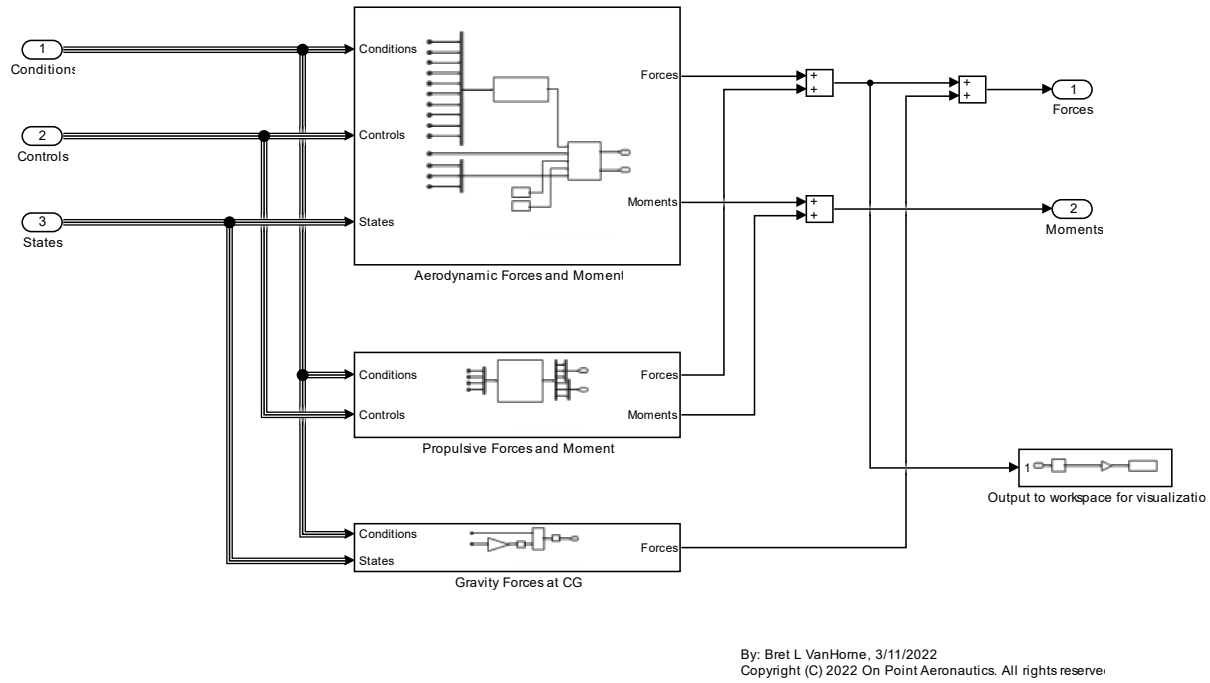

Figure 5. Aircraft Forces and Moments

Aerodynamic Forces and Moments

The subsystem that calculates the aerodynamic forces and moments is configured to make calls to the aircraft model using an “Interpreted MATLAB Function” block that retrieves the aerodynamic coefficients. The forces and moments are then calculated using a block from the Aerospace Blockset (Figure 6).

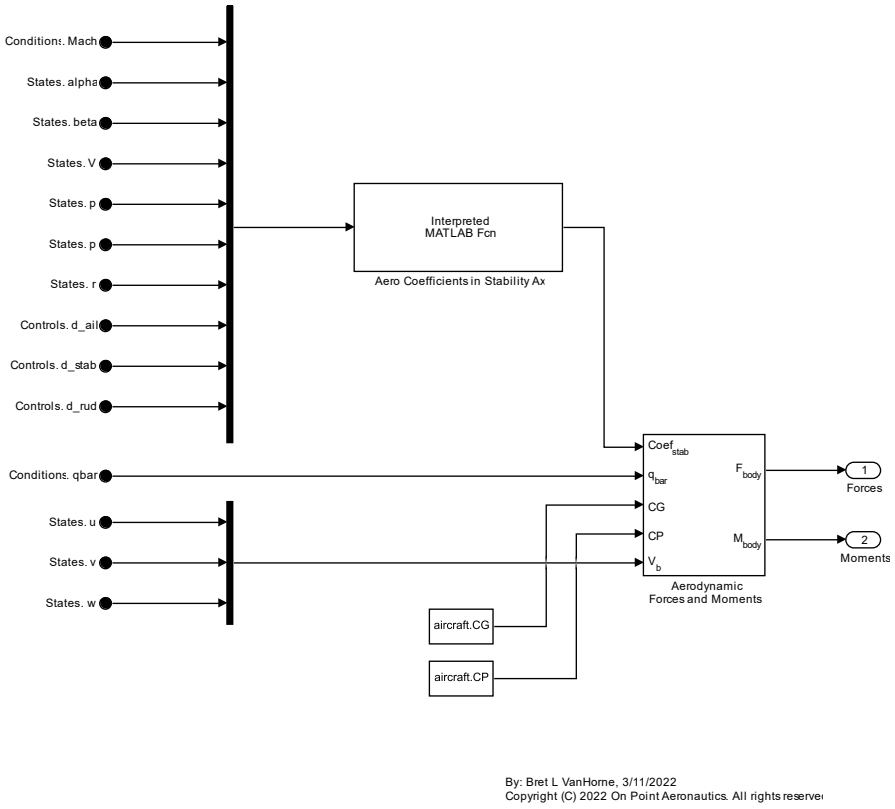

Figure 6. Aerodynamic Forces and Moments

Propulsive Forces and Moments

The subsystem for calculating the propulsive forces and moments is configured to make calls to the aircraft model using an “Interpreted MATLAB Function” block to retrieve the thrust forces and moments in the body axis.

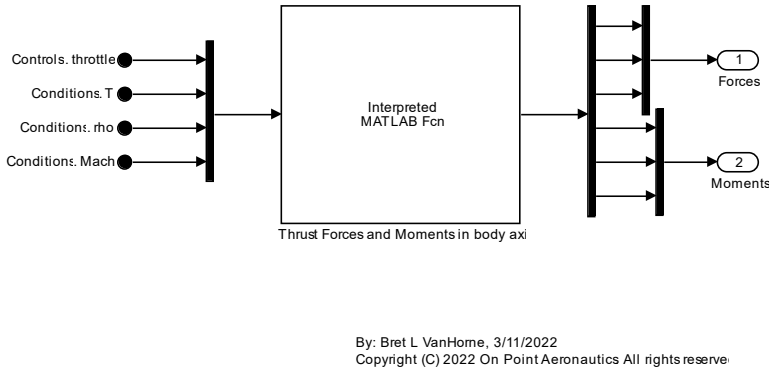

Figure 7. Propulsive Forces and Moments.

Control Inputs

The subsystem for control inputs collects and sums the trim condition along with any incremental control inputs from the “Commands” subsystem. These are fed through the aircraft control rigging method to apply limits to the deflections and enable mixing. The output from this subsystem is collected into a signal bus.

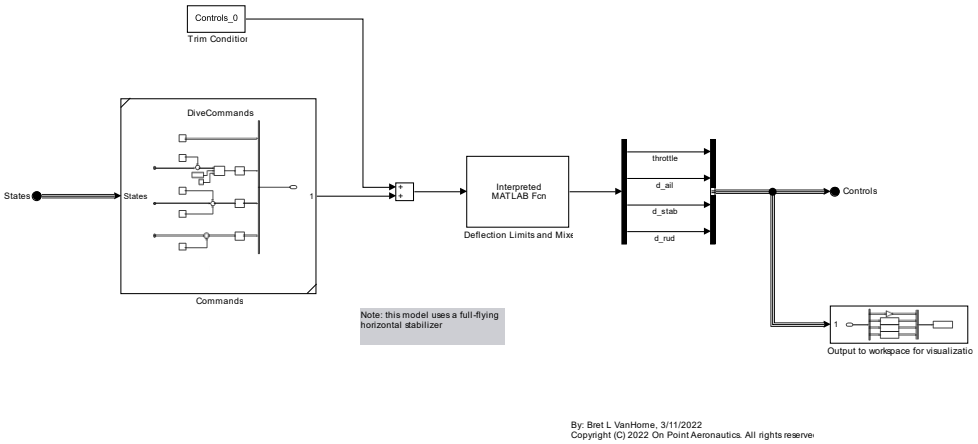

Figure 8. Control Inputs.

Aircraft Trim

A separate MATLAB function was built to trim the aircraft for a given flight condition. A simple numerical strategy was employed to find the roots of the equations for forces and moments. In the case of steady and level flight, the angle of attack and sideslip primarily balance the weight of the aircraft. The control deflections for aileron, elevator, and rudder primarily balance the rolling, pitching, and yawing moments respectively. In an iterative fashion, the function uses the secant method to converge on a solution.

```
% Trim routines
% These trim routines use a simple secant numerical method to find the
% "roots" where the sum of the forces and moments are zero.
% @author: Bret L VanHorne, 3/2022
% Copyright (C) 2022 On Point Aeronautics. All rights reserved

function [States, Euler, Controls] = Trim(routine, aircraft, altitude, airspeed, throttle)

    switch routine
        case '3_DOF'
            [States, Euler, Controls] = threeDOF(aircraft, altitude, airspeed);
        case '6_DOF'
            [States, Euler, Controls] = sixDOF(aircraft, altitude, airspeed);
        case 'TerminalVelocity'
            [States, Euler, Controls] = TerminalVelocity(aircraft, altitude, throttle);
    end

end

function [States, Euler, Controls] = threeDOF(aircraft, altitude, airspeed) ...

function [States, Euler, Controls] = sixDOF(aircraft, altitude, airspeed) ...

function [States, Euler, Controls] = TerminalVelocity(aircraft, altitude, throttle) ...
```

Figure 9. Trim Routines.

Quaternion 6-DOF Integration and States

The 6-DOF block from the Aerospace Blockset is used for quaternion integration. The transformation from Flat Earth to LLA (Latitude-Longitude-Altitude) and the transformation from body velocities to angle of attack and sideslip are included in this subsystem and the values are added to the outgoing signal bus.

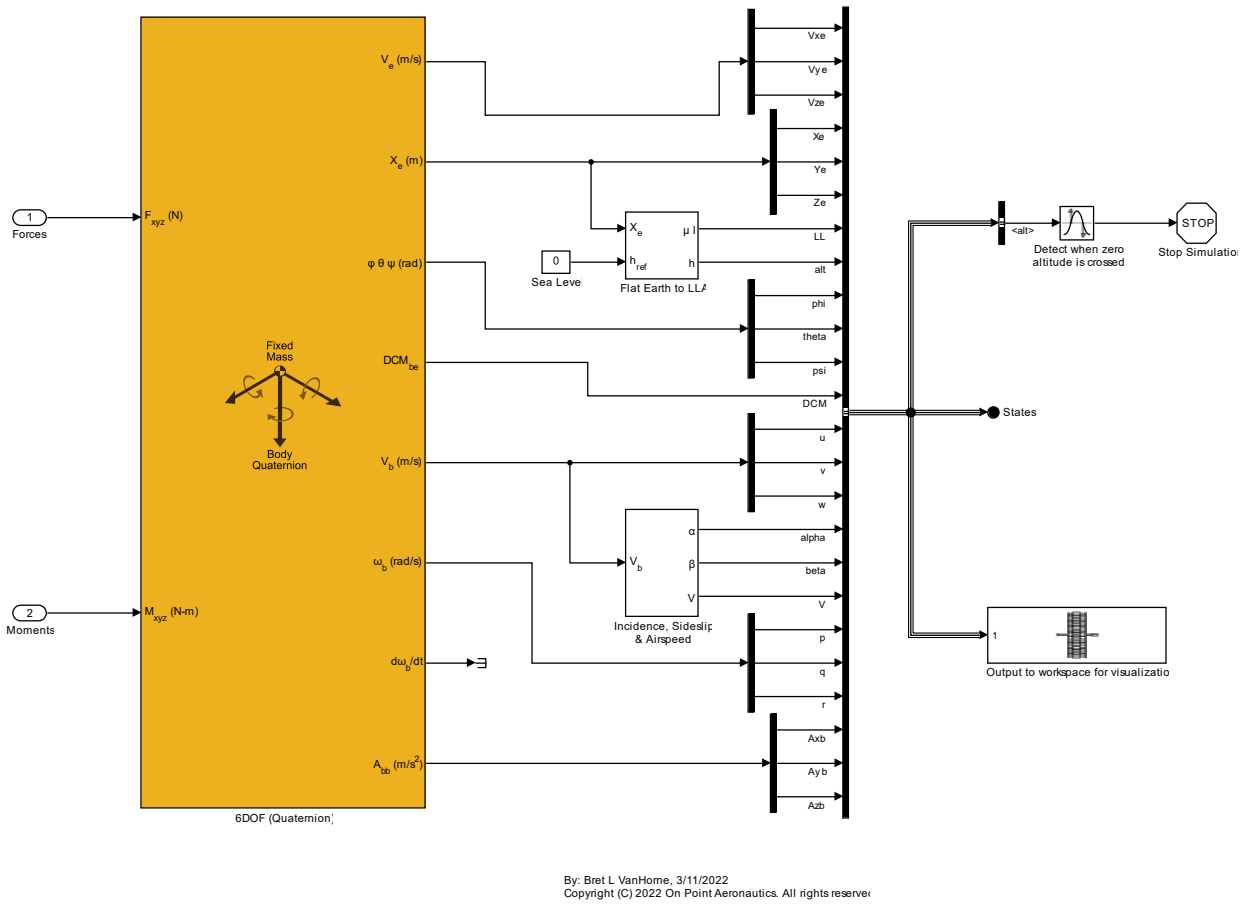

Figure 10. Quaternion 6-DOF Integration.

Running Simulations

Running a simulation begins by first deciding which aircraft and commands will be used. To swap out the "Commands" subsystem, navigate to the Control Inputs subsystem in Simulink. Click on the "Commands" subsystem block. Browse and select the file that contains the commands you would like to use and then click "Save" on the "Simulation" tab (Figure 11).

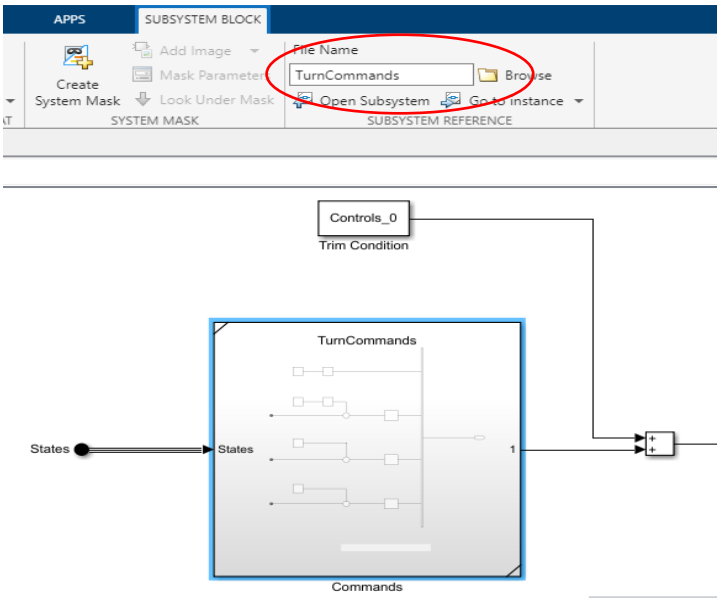

Figure 11. Choosing the simulation commands.

Next, open the "RunSimulation.m" script. Instantiate the aircraft you would like to use naming it "aircraft." Set the trim flight conditions, set the length of time for the simulation, and run the script.

```
% Run the 6-DOF flight path simulation.
% @author: Bret L VanHorne, 3/2022
% Copyright (C) 2022 On Point Aeronautics. All rights reserved.

% Initialize the environment
clear
clc
close all

% Instantiate the aircraft
aircraft = RCAM_Transport;

% Trim the aircraft (20000 feet, 300 kts)
[States_0, Euler_0, Controls_0] = Trim('3_DOF', aircraft, 6096, 154.333, 0);

% Run the simulation
TF = 50; % seconds
simOutput = sim('FlightPathSimulation.slx');

% Plot the results
PlotTurn
```

Figure 12. Script for running the simulation.

Windup Turn

As an example, here is a command subsystem that executes a windup turn. Ramp sources are used to command inputs for throttling up and banking over. The pitch attitude is also commanded nose up to maintain altitude. Simple PID controllers are used to track the commands.

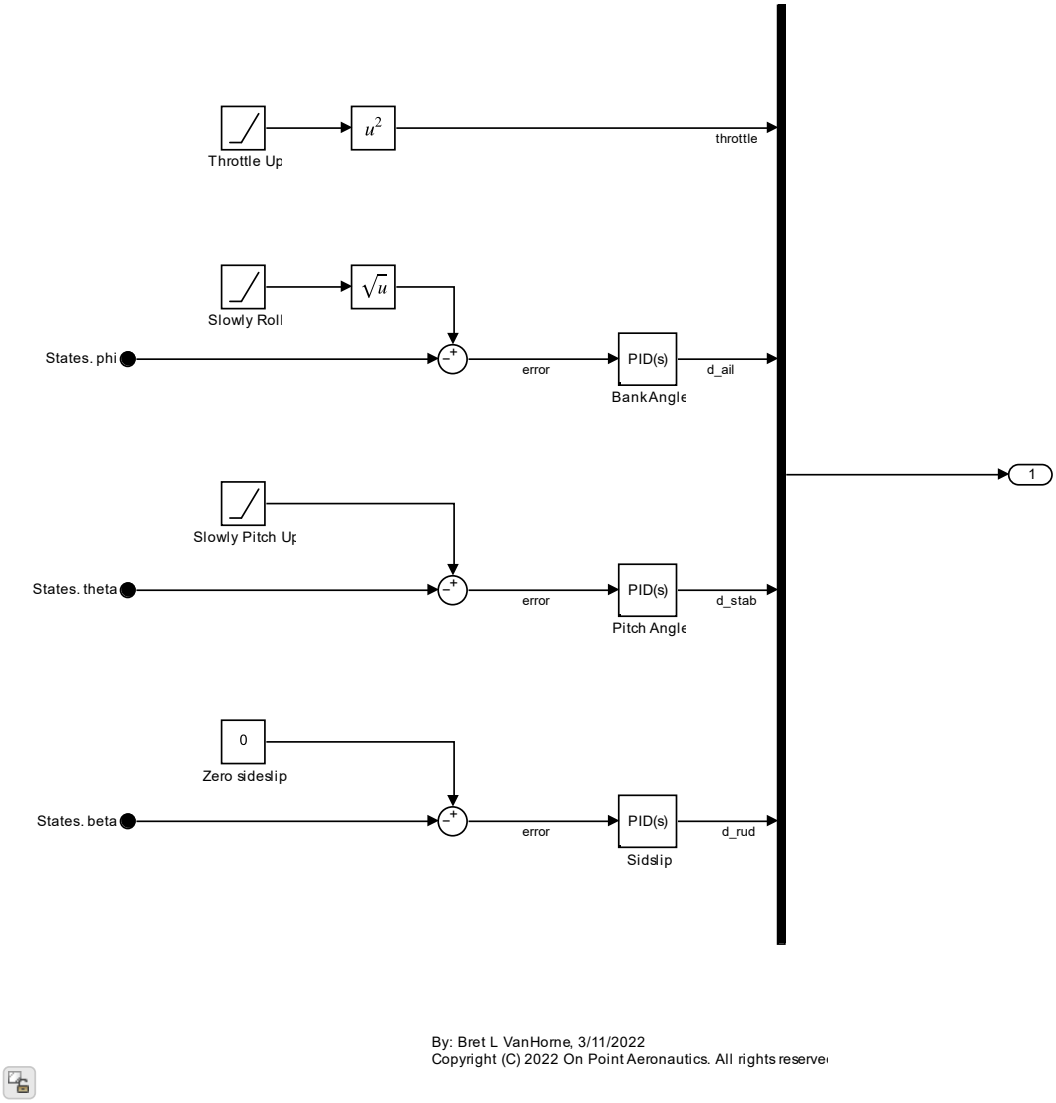

Figure 13. Turn commands and control laws.

If the simulation is successful, the results will be plotted (Figure 14).

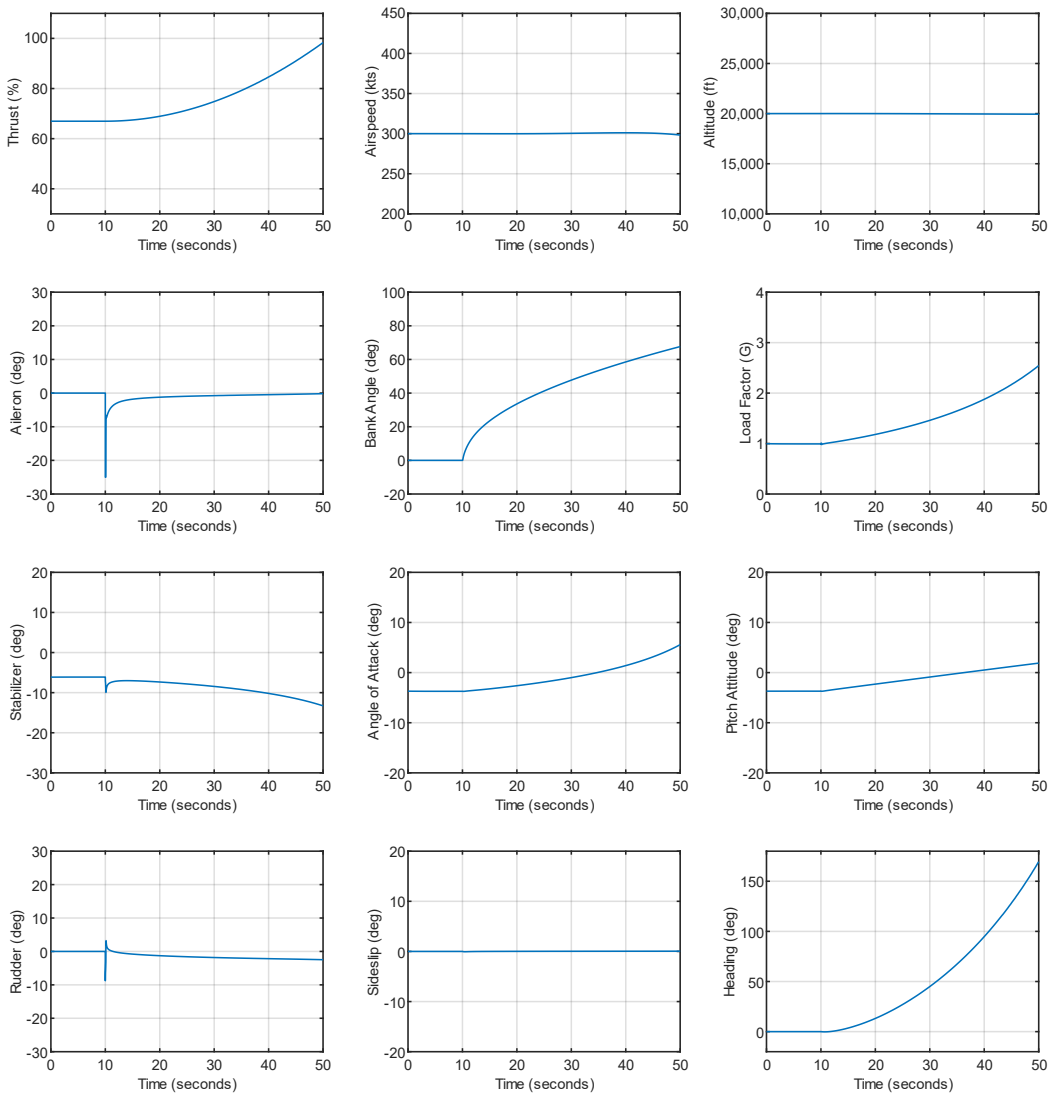

Figure 14. Flight path traces.

Conclusions

A simple Simulink system was created for simulating the flight path of an aircraft and was designed expressly for the aerodynamics stability and control engineer. This system makes use of the Aerospace Toolbox and the Aerospace Blockset. The system was designed to encapsulate the aircraft definitions in a MATLAB class. The command-and-control subsystem was also designed to be encapsulated in a single swappable file. This arrangement allows flexibility as well as portability for many engineering tasks both inside and outside of Simulink.