ENG5076 Simulation Of Aerospace Systems Assignment

Joshua Yandoc - 3065837Y

December 2024

1 Introduction

The overall goal of this assignment is to simulate the lateral beam autopilot guidance system of an aircraft and validate the model, testing how long it takes the control system to go from one heading angle direction, to one where the aircraft's heading is in line with the centre of the runway. As well as testing and validating the model through coded simulation as well as a block model simulation, the assignment also investigates how performance of the system might vary in accuracy and efficiency in cases where certain input or intial paramters are altered.

2 Methodology

2.1 Mathematical Modelling & Continuous Time Simulation

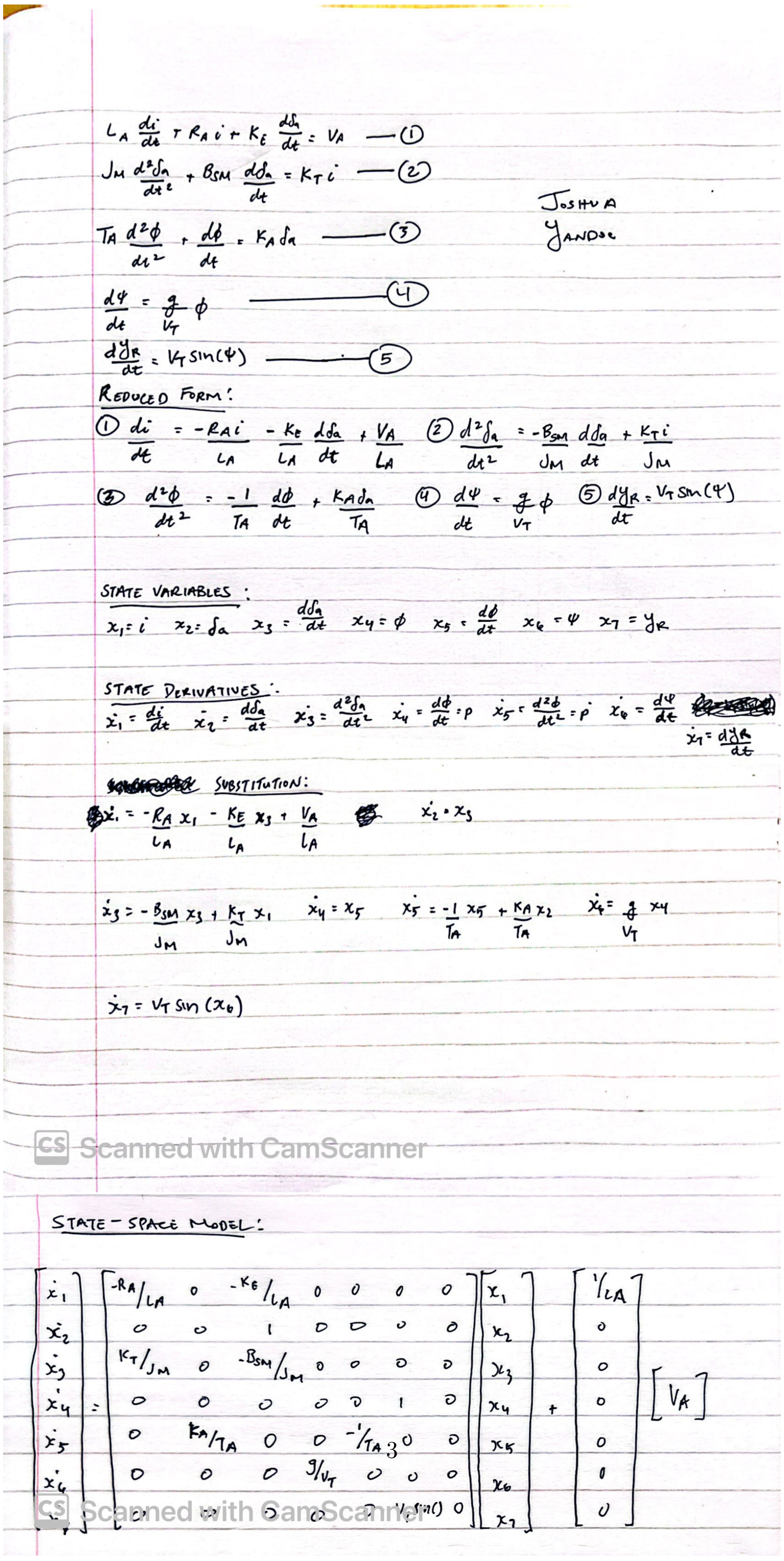
The first step in deriving the state space model for the guidance system started with defining what equations needed to be defined that modelled the exact system being described. It is important to realise early on that this consisted of two sets of coupled equations, one that described the electrical component of the system, and one that described the mechanical component, plus an additional equation to define the rate of change of the lateral displacement of the aircraft. From these five equations, a 7th order model could be defined, meaning that there would be 7 corresponding state variables and state-derivative variables to handle. The hand derivation of the state space model is shown below in **Figure 1**

From this point, both the main script that ran the simulation and plotted the results, as well as the corresponding state space derivation function script that dynamically ran the calculations could be written. Using the parameter values given in the assignment document, as well as the initial conditions given, the reference states and simulation parameters could be initialized, with some exceptions.

Certain states as well as the control input(voltage) are not explicitly stated, but it can safely be assumed to all start at 0 for all variables. Logically speaking, if the aircraft begins at a neutral state before the guidance system takes over, no current is being supplied to the ailerons, which also implies that there is no deflection rate, roll rate, or voltage being supplied to the ailerons. Hence, these respective state variables can simply be assumed to have an initial value of 0.

When it comes to employing an appropriate numerical integration solver and step-size, it was decided that the 4th-order Runge-Kutta integration algorithm provided the best trade off of providing the most accurate data, while not having a substantial enough error due to nonlinearities that the simulation would be considered unreliable. In addition, the total run-time of the simulation was not long enough that the efficiency and speed of the calculations would be an issue. In regards to this total run-time, because it was not on the scale of multiple minutes, but rather seconds, a small step-size of 0.01 was chosen. Much like the reasoning for the choice of integration method, the scale of the chosen step-size provided the best trade off between accuracy of the simulation to the potential round-off error and run-time.

During the implementation of the state-derivative function, the coding itself was as simple as copying verbatim the hand-derived state-space model into code. The only notable addition to this function is in regards to having to implement the calculations of the guidance system controls and its subsequent subsystems into the function as well (e.g. functions like commanding heading angle or voltage input). This is because, while all these states and system parameters were initialized in



$$L_A \frac{di}{dt} + R_A i + K_E \frac{d\delta_a}{dt} = V_A \quad \text{—} (1)$$

$$J_M \frac{d^2\delta_a}{dt^2} + B_{SM} \frac{d\delta_a}{dt} = K_T i \quad \text{—} (2)$$

JOSHUA YANDOC

$$T_A \frac{d^2\phi}{dt^2} + \frac{d\phi}{dt} = K_A \delta_a \quad \text{—} (3)$$

$$\frac{d\psi}{dt} = \frac{g}{V_T}\phi \quad \text{—} (4)$$

$$\frac{dy_R}{dt} = V_T \sin(\psi) \quad \text{—} (5)$$

REDUCED FORM:

$$(1)\ \frac{di}{dt} = \frac{-R_A i}{L_A} - \frac{K_E}{L_A}\frac{d\delta_a}{dt} + \frac{V_A}{L_A} \qquad (2)\ \frac{d^2\delta_a}{dt^2} = -\frac{B_{SM}}{J_M}\frac{d\delta_a}{dt} + \frac{K_T i}{J_M}$$

$$(3)\ \frac{d^2\phi}{dt^2} = \frac{-1}{T_A}\frac{d\phi}{dt} + \frac{K_A\delta_a}{T_A} \qquad (4)\ \frac{d\psi}{dt} = \frac{g}{V_T}\phi \qquad (5)\ \frac{dy_R}{dt} = V_T \sin(\psi)$$

STATE VARIABLES:

$$x_1 = i \quad x_2 = \delta_a \quad x_3 = \frac{d\delta_a}{dt} \quad x_4 = \phi \quad x_5 = \frac{d\phi}{dt} \quad x_6 = \psi \quad x_7 = y_R$$

STATE DERIVATIVES:

$$\dot{x}_1 = \frac{di}{dt} \quad \dot{x}_2 = \frac{d\delta_a}{dt} \quad \dot{x}_3 = \frac{d^2\delta_a}{dt^2} \quad \dot{x}_4 = \frac{d\phi}{dt} = p \quad \dot{x}_5 = \frac{d^2\phi}{dt^2} = \dot{p} \quad \dot{x}_6 = \frac{d\psi}{dt} \quad \dot{x}_7 = \frac{dy_R}{dt}$$

SUBSTITUTION:

$$\dot{x}_1 = -\frac{R_A}{L_A}x_1 - \frac{K_E}{L_A}x_3 + \frac{V_A}{L_A} \qquad \dot{x}_2 = x_3$$

$$\dot{x}_3 = -\frac{B_{SM}}{J_M}x_3 + \frac{K_T}{J_M}x_1 \qquad \dot{x}_4 = x_5 \qquad \dot{x}_5 = \frac{-1}{T_A}x_5 + \frac{K_A}{T_A}x_2 \qquad \dot{x}_6 = \frac{g}{V_T}x_4$$

$$\dot{x}_7 = V_T \sin(x_6)$$

STATE-SPACE MODEL:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \\ \dot{x}_7 \end{bmatrix} = \begin{bmatrix} -R_A/L_A & 0 & -K_E/L_A & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ K_T/J_M & 0 & -B_{SM}/J_M & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & K_A/T_A & 0 & 0 & -1/T_A & 0 & 0 \\ 0 & 0 & 0 & g/V_T & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & V_T\sin() & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1/L_A \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} V_A \end{bmatrix}$$

Figure 1: Hand-derivation of state-space model

the main script, these values dynamically change over time as the system operates on a closed-loop feedback system, meaning over every incremental step-size of the simulation, these values would also need to dynamically be updated to produced the necessary data to be used in the next iteration. Both the main script file, as well as the Runge-Kutta integration method function can be found in **Appendix A**.

2.2 Block Diagram and Validation

Much like how the implementation of the state-space model into functioning code was essentially just writing out the equations verbatim, the modelling of the SIMULINK block diagram and its internal subsystems was achieved in the same manner, with the difference being the conception of this simulation model being done visually, whereas the MATLAB script was based off written code. While the system could easily be replicated using all the simple blocks found within SIMULINK's library, therre were three points of note to take into account to ensure the simulated model matched the MATLAB script

1. Realizing that the most intricate and complex subsystems stemmed from the DC motor system, as well as the roll and heading system, as each of these subsystems required the modelling of two coupled equations for each.
2. Just the act of placing the correct corresponding blocks would not reproduce the simulated model from the MATLAB script, as not all initialized parameters and state variables are explicitly represented by blocks. Namely, the initial lateral displacement Y_{R_0}, the initial roll rate, p, and initial heading angle, ψ_0 need to manually be initialized within their respective integrator blocks that output their state variable.
3. For the purposes of visually validating the block model simulation to the coded simulation, scope blocks would need to be placed at various points within the model to verify these state variables, ane while not as important but nonetheless worth noting, converting specific variables to their proper units (i.e. from radiangs back to degrees).

The overall block diagram model, as well as all of its subsystems can be found in **Appendix B**.

2.3 Control System & Implementation

When increasing the coupler gain from the the one provided in the document, the stability of the system actually reversed, and became unstable. Thus, in pursuit of investigating how varying this gain affects stability and efficiency of the system, various different values were tested ranging from 0 to the initial value chosen. For the purposes of having to run these multiple simulations for quick comparisons, gain factor multiples of 15 were tested to determine the most optimal value.

Once this change in gain factor was determined, the introduction of an integral term with associated gain, K_I, was then introduced to further test the performance of this system in an attempt to pinpoint an ideal performance scenario. The results of both system alterations are discussed in the subsequent section.

2.4 Variations & Limitations

The effect of differing longitudinal velocities is discussed in the next section, but with regards to investigating how different actuator specifications and limits impact the performance of the system,

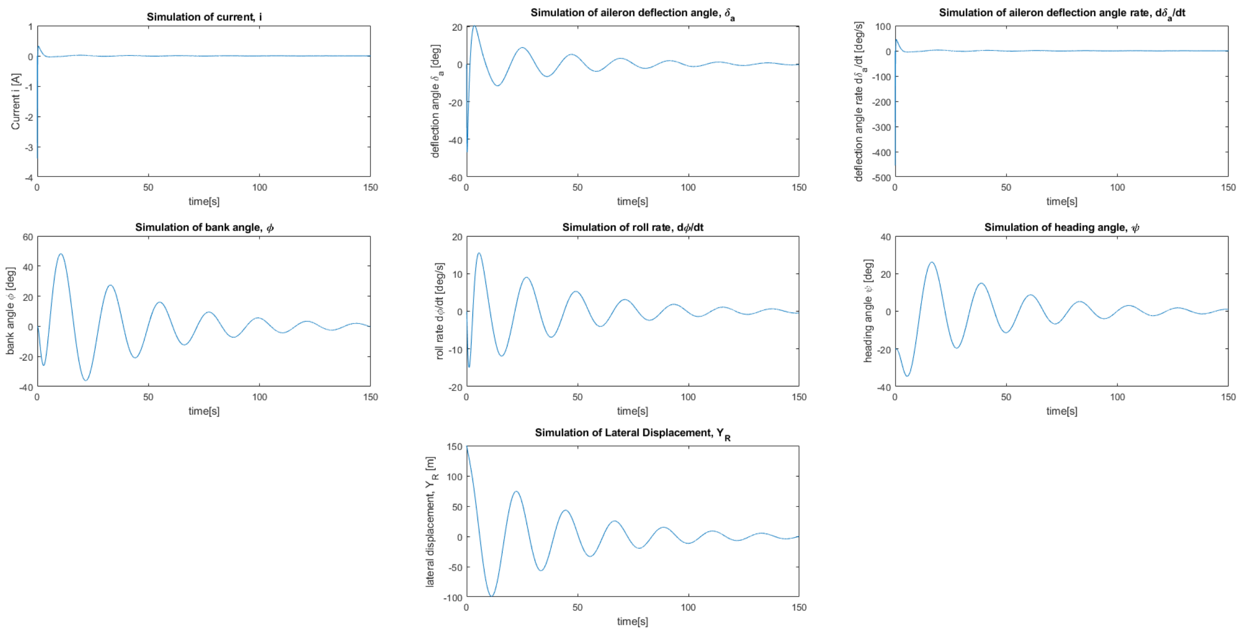


Figure 2: Simulation plots of the state variables over time

it is as simple as just adding the limits as y-intercept curves to their corresponding state variable plots. In doing so, the limits imposed by each actuator on the ailerons can be visually represented, and determining which actuator gives the best performance can easily be assessed. These modified plots are discussed in the next section.

3 Results and Anlysis

3.1 Mathematical Modelling & Continuous Time Simulation

Running the simulation produces the model plots in **Figure 2**, with all state variables (lateral displacement, bank angle, deflection rate, etc) being plotted over time.

Upon initial analysis, the coupler system does its job as intended in guiding the aircraft to have a heading angle to be in line with the centre line. Especially when looking at the plot of deflection angle rate versus time, the simulation tells us that the ailerons are close to maintaining their deflection angle at a constant angle. However, the one crucial factor that makes this coupler system suboptimal, and in fact potentially dangerous, is the fact that in this specific scenario, the bank angle surpasses the threshold considered safe for what pilots should generally be operating under: 45 degrees[1]. Ideally, whlie the idea of having much less adjustments to aileron deflection rates and bank angles would constitute the perfect scenario, this is not realisable, as by definition, this dynamic system is constantly having to adjust its states over time due. Nevermind the fact that this scenario introduces the possibility of having to deal with the human factor, and the implications of

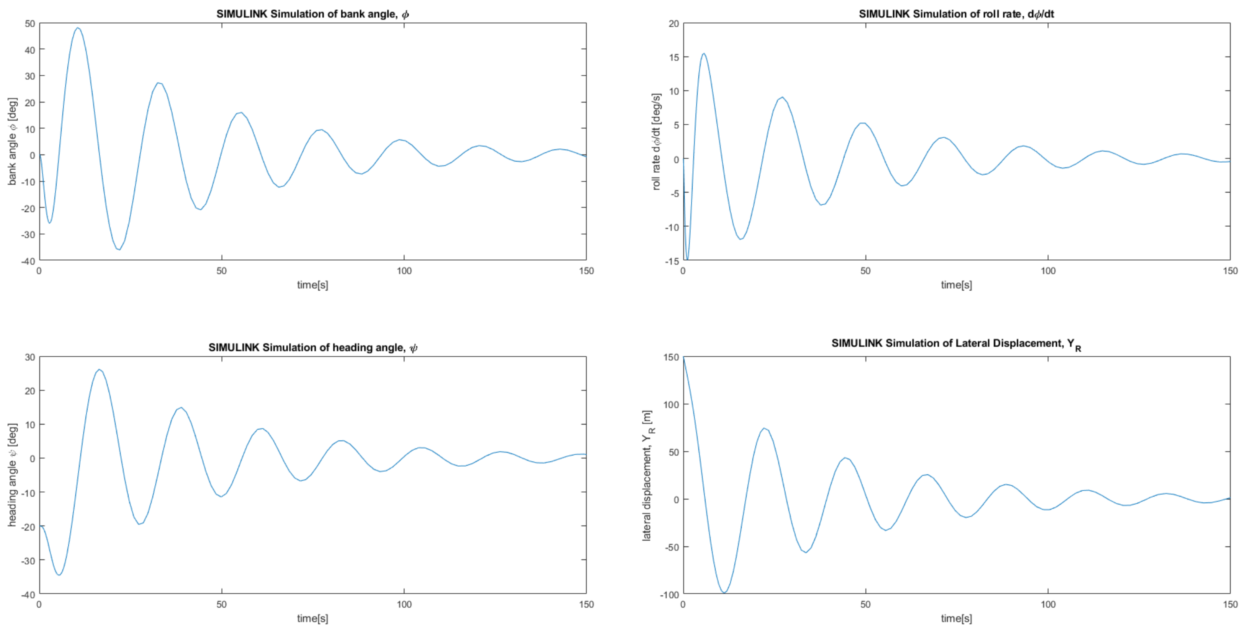


Figure 3: SIMULINK simulation plots over time

having such an underdamped system, that parameters such as bank angle for example may be too steep or too dangerous.

3.2 Bloack Diagram and Validation

After the running the SIMULINK model, the second section of the main script takes that outputted data and then plots a number of the same state variables against the same time domain for the purposes of validating the system. Those plots can be in **Figure 3** below.

Strictly for the purposes of validating the MATLAB simulation model, not all state variables from the SIMULINK model are plotted, as having a handful of the state variables corroborate and validate the MATLAB model is sufficient enough evidence to assess the accuracy of the model in this case. Here, the states chosen were bank angle, roll rate, heading angle, and displacement, as these variables are extrinsic properties that convey the behaviour of the guidance system, and thus the aircraft's behaviour as well.And from the produced plots, it is very clear that there is a direct match in the plots being produced by the SIMULINK model to that produced by the MATLAB model, thus validating the model and its simulation responses as being accurate and realistic.

3.3 Control System & Implementation

Regarding the coupler gain G_c, by lowering the value, the system remained stable, while simultaneously becoming more and more increasingly damped. With a gain of about 30, the simulated

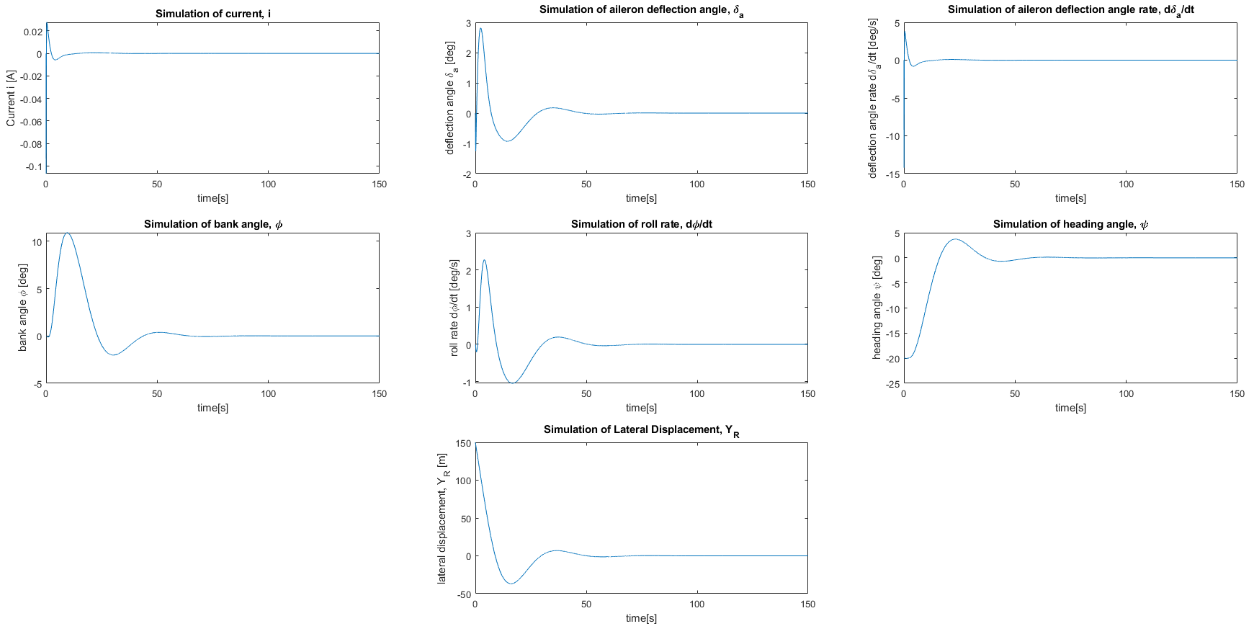


Figure 4: Simulated Guidance System with coupler gain, $G_c = 15$

system was noticeably more efficient, with the coupler having a much more substantial impact on the overshoot, and thus, the resulting oscillations of state variables such as the heading angle, and lateral displacement. In essence, this meant the system error and resulting over-corrective behaviour of the guidance system on the ailerons was mitigated much more. Though, this was far from ideal, as by lowering the gain even more to a value of 15, the system response had very promising results, with having virtually minimal overshoot, and coming to steady state in an efficient, and for that matter, realistic matter. By realistic, this meant the values being outputted, such as deflection angle, roll rate, and bank angle, were fully realisable by the system in practice. The bank angle was also well within the tolerable threshold of safety regulations. The simulated system can be seen in **Figure 4**. Going towards the extreme of that spectrum, a gain of 0 produced an extremely damped system that was not realisable, showing how the coupler system is imperative in helping dampen the system, while efficiently bringing it to steady state.

Surprisingly, in the continuing case study of working to optimize the control guidance system, the introduction of an integral term into this specific system showed no noticeable changes in performance. It was found that the most optimal system utilized an integral gain of 1, which is equivalent to not having an integral term to begin with. The simulated plots are the exact same as **Figure 4**. Comparing these plots with **Figure 5** and **Figure 6**, where the integral term is set to an equal and opposite value of about 0.5 from the $K_I = 1$ value, the higher integral value actually produces a much less stable system overall. While very subtle, it can be seen in the plots for **Figure 6**, for states like deflection angle, roll rate, bank angle and lateral displacement (notice how values like bank angle and roll rate have actually increased). **Figure 5** where $K_I = 0.5$ produces nonsensical data, where states like current start high, before going negative. In both cases, as the gain trends closer and closer to 1, the system becomes much more optimal, efficient, and realisable. Thus, it can be concluded in this case, that the introduction of the integral term serves no purpose in further improving the performance of the lateral beam control system.

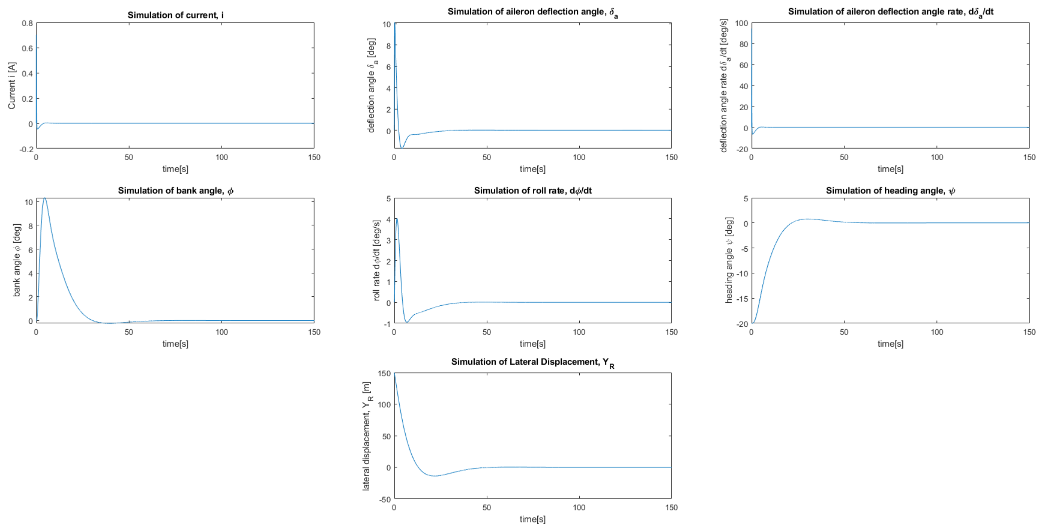


Figure 5: Simulated Guidance System with coupler gain, $G_c = 15$, and integral gain $K_I = 0.5$

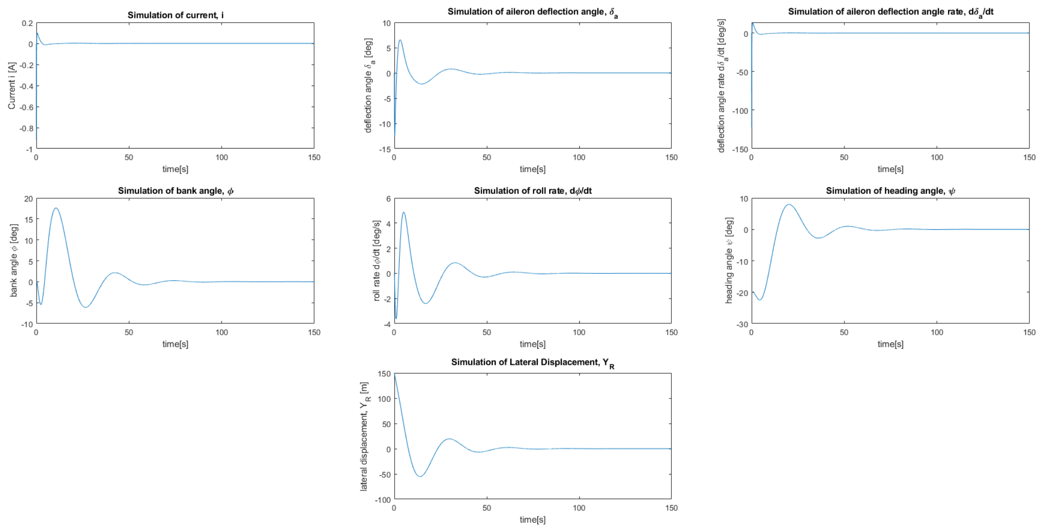


Figure 6: Simulated Guidance System with coupler gain, $G_c = 15$, and integral gain $K_I = 1.5$

3.4 Variations & Limitations

When investigating the effect of different longitudinal velocities on the systems performance, **Figures 7** and **8** correlate to longitudinal velocities of 50 m/s and 60 m/s respectively. The most notable variation in the simulation results relate to the stabilisation of the system and how long it takes to reach steady state. For the lower longitudinal velocity, the system appears to reach steady state much sooner, whereas with the higher velocity, the simulation takes longer to reach steady state. Even within the time domain, it appears to not fully reach steady state, as there are still very noticeable oscillations occurring. However, the overall shape of every plot appears to remain the same, insinuating that heading velocity only really affects how much time an aircraft will need to have a heading angle in line with the centre line. This intuitively makes sense, as the derived lateral displacement, Y_R from **Figure 1** is a function of longitudinal velocity V_T and heading angle, $sin(\psi)$, which means that for higher velocities, the resulting amplitudes would also scale up, requiring much more time for the coupler to dampen the system and reach steady state.

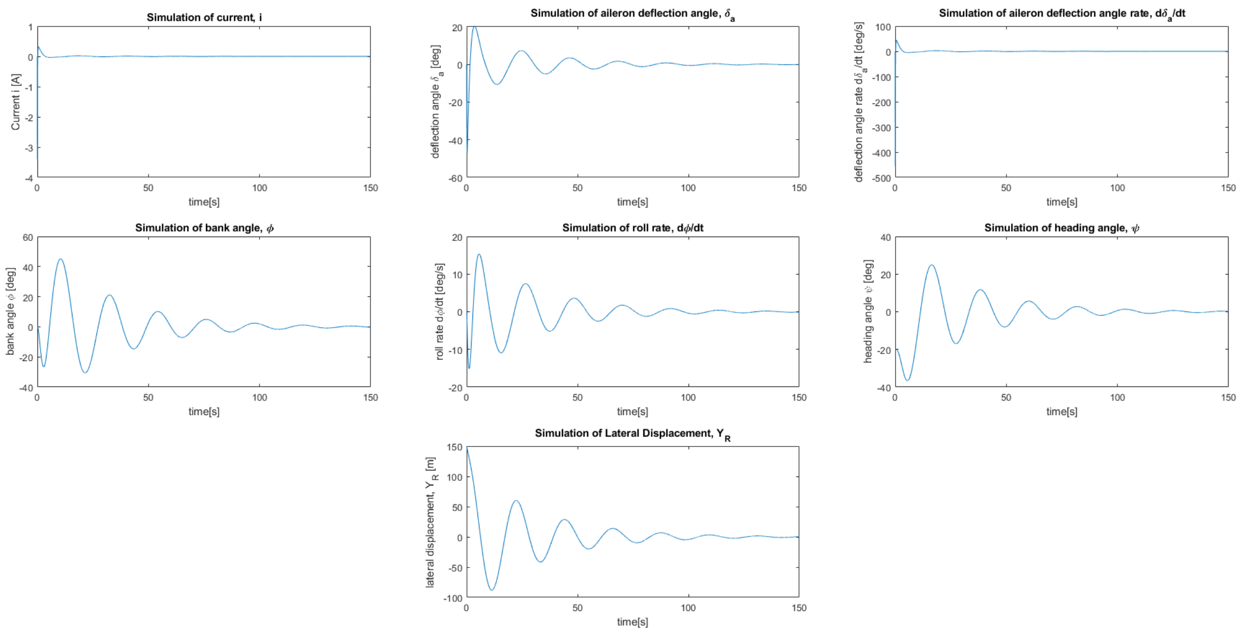


Figure 7: Simulation for an initial longitudinal velocity, $V_T = 50$ m/s

Finally, with regards to imposing different actuators with different limits on the control system, the resulting plots describing the limits to the aileron deflection angle and deflection angle rate are depicted in **Figure 9**. While every actuator in this case would impose substantial restrictions on the aircraft's deflection angle rate, in terms of the deflection angle, the third actuator just barely allows for the aileron to reach its required deflection angle full, at least in this scenario. This means that this specific actuator would be the most optimal choice in providing the best performance, as it most closely allows the system to operate at the full capacity needed, whereas the other options restrict the aileron's output much more.

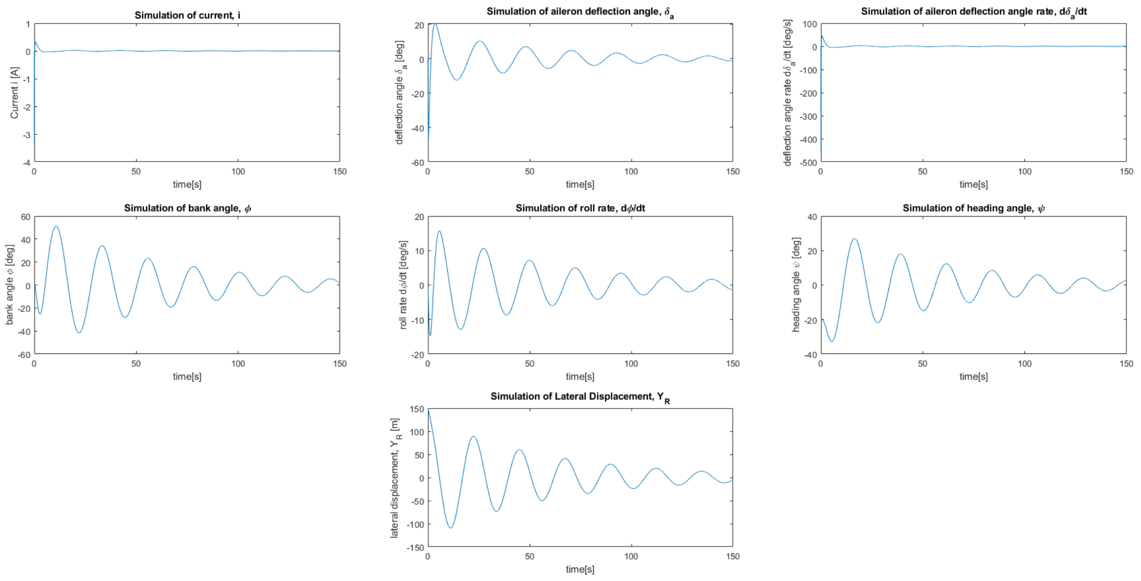


Figure 8: Simulation for an initial longitudinal velocity, $V_T = 60$ m/s

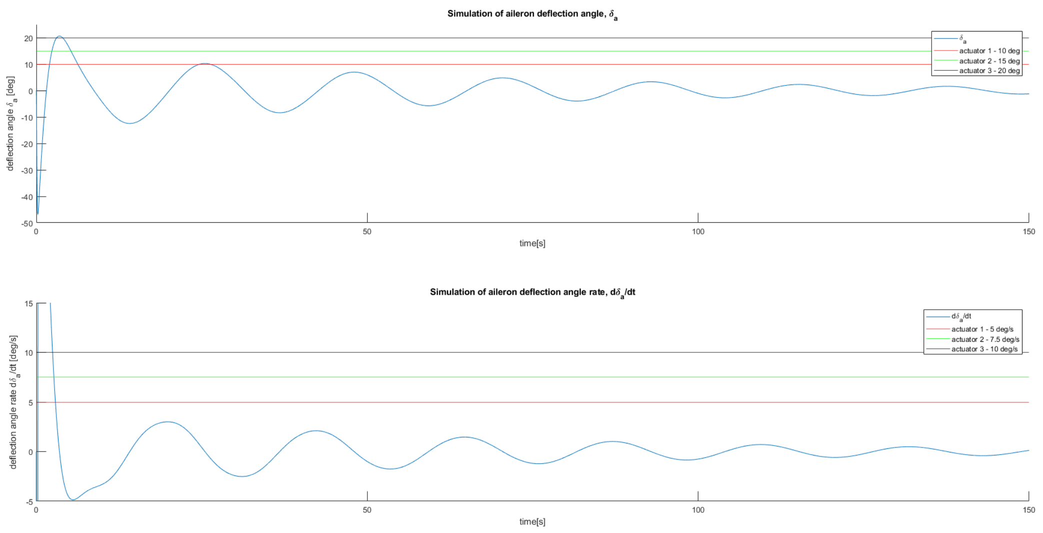


Figure 9: Effect of different actuator limits on aileron deflection angle and deflection angle rate

4 Conclusion

In the end, the validation of control systems through simulated means, either by code or block diagrams, has been shown to provide accurate, reliable feedback. These simulations offer the chance to asses not just specific scenarios under various operating parameters without the costly, and potentially dangerous, need for physical testing, but these models also allow for the assessment and optimization of various qualities like performance handling, system response time, and even the implications that varying components and their limitations will have on the physical system.

5 References

[1] $AirplaneFlyingHandbook(3C)Chapter7Page2, www.faa.gov/sites/faa.gov/files/regulations\_policies /handbooks\_manuals/aviation/airplane\_handbook/08\_afh\_ch7.pdf. Accessed5Dec.2024.$

Appendix

A MATLAB Code

A.1 Main Script

```
% Joshua Yandoc - 3065837Y
% ENG5076 - Assignment
% ILS Lateral Beam Guidance System Simulation

clear
clc

% initialize the parameter values as global variables (i.e. defining
    model
% paramters)
global B_SM g G_c J_M K_A K_D K_E K_P K_R K_T K_V L_A R_A T_A V_T R
   V_A

% Appendix A: Given Parameters for the Lateral Beam Guidance System
B_SM = 0.7;
g = 9.81; % m/s^2
G_c = 45.5;
J_M = 0.006;
K_A = 1.2;
K_D = 0.9;
K_E = 0.9;
K_P = 52.5;
K_R = 1.2;
K_T = 1.7;
K_V = 1.3;
L_A = 0.2; % H
R_A = 10; % Ohms
T_A = 2.0; % s
```

```
% Initial conditions of inputs, states, and state derivatives
psi_0 = deg2rad(-20); % initial heading angle given in [deg], but
    convert to [rad]
phi_0 = deg2rad(0); % initial bank angle given in [deg], but convert
     to [rad]
R_0 = 6000; % initial distance from craft to end of runway [m]
R = R_0;
Y_R0 = 150; % inital lateral distance of craft to centre line [m]
V_T = 55; % initial forward velocity [m/s]
% These initial conditions are not explicity given, but assumed

% current  - initially, aileron is in neutral position as the
    guidance system hasn't provided
% banking angle, roll rate, etc, needed that the actuator will then
% take to control the aileron - hence, aileron isn't moving
% initially, meaning actuator isn't operating, meaning i_0 = 0;
i_0 = 0; % [A]
% if the ailerons aren't operating initially, then deflection angle
    is 0
% deg, which also implies deflection rate is also 0
delta_a_0 = 0; %[rad]
delta_a_dot_0 = 0; %[rad/s]
% if phi_0 is 0, then roll rate, dphi/dt = p, is also 0
p_0 = deg2rad(0); % given in [deg/s], but convert to [rad/s]
% V_A = 0 initially for the same reason as i_0 = 0
% Voltage V_A is also our control input for the system
V_A = 0;


x = [i_0, delta_a_0, delta_a_dot_0, phi_0, p_0, psi_0, Y_R0]; %
    initializing state variable array
xdot = zeros(7,1); % initializing state derivative array

% Defining the simulation parameters
stepsize = 0.01;
endtime = 100;
counter = 0;
comminterval = 0.01;

% Dynamic Segment
for time = 0:stepsize:endtime
    % store the time, state, and state derivative data point at
        every
    % stepsize
    if rem(time, comminterval) ==0
        counter = counter + 1; % incr counter
        tout(counter) = time; % store time
        xout(:,counter) = x; % store state
```

```
            xdout(:, counter) = xdot; % store state deriv.
        end

        % Derivative Section
        xdot = Lateral_Beam_Guidance_System_State_Deriv(x);

        % Integral Section - Using 4th Order Runge-Kutta integration
        k1 = stepsize * Lateral_Beam_Guidance_System_State_Deriv(x);
                            % evaluate derivative k1
        k2 = stepsize * Lateral_Beam_Guidance_System_State_Deriv(x+(k1
            /2)); % evaluate derivative k2
        k3 = stepsize * Lateral_Beam_Guidance_System_State_Deriv(x+(k2
            /2)); % evaluate derivative k3
        k4 = stepsize * Lateral_Beam_Guidance_System_State_Deriv(x+k3);
                         % evaluate derivative k4
        x = x + ((k1 + 2*k2 + 2*k3 + k4)/6);                % averaged
            output


    end

    % convert the state variables with units of [rad] back to units of [
        deg]
    xout(2:6, :) = rad2deg(xout(2:6, :));

    tiledlayout(3,3)
    nexttile
    plot(tout, xout(1,:))
    title('Simulation of current')
    xlabel('time[s]')
    ylabel('Current i [A]')

    nexttile
    plot(tout, xout(2,:))
    title('Simulation of aileron deflection angle')
    xlabel('time[s]')
    ylabel('deflection angle [deg]')

    nexttile
    plot(tout, xout(3,:))
    title('Simulation of aileron deflection angle rate')
    xlabel('time[s]')
    ylabel('deflection angle rate [deg/s]')

    nexttile
    plot(tout, xout(4,:))
    title('Simulation of bank angle')
    xlabel('time[s]')
```

```
ylabel('bank angle [deg]')

nexttile
plot(tout, xout(5,:))
title('Simulation of roll rate')
xlabel('time[s]')
ylabel('roll rate [deg/s]')


nexttile
plot(tout, xout(6,:))
title('Simulation of heading angle')
xlabel('time[s]')
ylabel('heading angle [deg]')

nexttile(8)
plot(tout, xout(7,:))
title('Simulation of Lateral Displacement')
xlabel('time[s]')
ylabel('lateral displacement, Y_R [m]')

%% Plotting the SIMULINK Model

% Get the data from the SIMULINK model
block_time = out.tout;
block_Y_R = out.Y_R;
block_p = out.p;
block_phi = out.phi;
block_psi = out.psi;

figure()
tiledlayout(2,2)

nexttile
plot(block_time, block_phi)
title('Simulation of bank angle, \phi')
xlabel('time[s]')
ylabel('bank angle \phi [deg]')

nexttile
plot(block_time, block_p)
title('Simulation of roll rate, d\phi/dt')
xlabel('time[s]')
ylabel('roll rate d\phi/dt [deg/s]')

nexttile
plot(block_time, block_psi)
title('Simulation of heading angle, \psi')
```

```
xlabel('time[s]')
ylabel('heading angle \psi [deg]')

nexttile()
plot(block_time, block_Y_R)
title('Simulation of Lateral Displacement, Y_R')
xlabel('time[s]')
ylabel('lateral displacement, Y_R [m]')

%% For Q12

% create array for the amplitutde limit and rate limit for each
    aileron
% given in units of [deg] and [deg/s] - convert both to [rad] and [
    rad/s]
Amplitude_limit = [10, 15, 20];
Rate_limit = [5, 7.5, 10];
colors = {'r', 'g', 'k'};

figure()
tiledlayout(2,1)
nexttile
hold on
plot(tout, xout(2,:))
for actuator_number = 1:1:3
    yline(Amplitude_limit(actuator_number), colors{1,
        actuator_number})
end
hold off
ylim([-50 25])
title('Simulation of aileron deflection angle, \delta_a')
xlabel('time[s]')
ylabel('deflection angle \delta_a [deg]')
legend('\delta_a', 'actuator 1 - 10 deg', 'actuator 2 - 15 deg', '
    actuator 3 - 20 deg')

nexttile
hold on
plot(tout, xout(3,:))
for actuator_number = 1:1:3
    yline(Rate_limit(actuator_number), colors{1, actuator_number})
end
hold off
ylim([-5, 15])
title('Simulation of aileron deflection angle rate, d\delta_a/dt')
xlabel('time[s]')
ylabel('deflection angle rate d\delta_a/dt [deg/s]')
legend('d\delta_a/dt', 'actuator 1 - 5 deg/s', 'actuator 2 - 7.5 deg
   /s', 'actuator 3 - 10 deg/s')
```

A.2 Derivative Function Script

```
% Joshua Yandoc - 3065837Y
% ENG5076 - Assignment
% State derivative function for Lateral Beam Guidance System
   Dynamics

function xdot = Lateral_Beam_Guidance_System_State_Deriv(x)

% transfer global parameters from main program
global B_SM g G_c J_M K_A K_D K_E K_P K_R K_T K_V L_A R_A T_A V_T R
   V_A

 % array x contains our state variables - initialize them to their
    proper
 % corresponding variable
 i = x(1); % current [A]
 delta_a = x(2); % aileron deflection angle [rad]
 delta_a_dot = x(3); % aileron deflection angle rate [rad/s]
 phi = x(4); % roll angle converted to [rad]
 p = x(5); % roll rate - dphi/dt converted to [rad/s]
 psi = x(6); % heading angle converted to [rad]
 Y_R = x(7); % lateral displacement [m]


 % output the state derivative equations
 % x(1) dot = di/dt
 xdot(1) = (-(R_A/L_A) * i) - ((K_E/L_A)*delta_a_dot) + (V_A/L_A);
 % x(2) dot = ddelta_a/dt
 xdot(2) = delta_a_dot;
 % x(3) dot = d^2delta_a/dt^2
 xdot(3) = ( -(B_SM/J_M)*delta_a_dot) + ((K_T/J_M)*i);
 % x(4) dot = dphi/dt
 xdot(4) = p; % convert to [rad/s]
 % x(5) dot = d^2phi/dt^2
 xdot(5) = (-(1/T_A)*p) +  ((K_A/T_A)*delta_a);
 % x(6) dot = dpsi/dt
 xdot(6) = (g/V_T)*phi;
 % x(7) dot = dY_R/dt
 xdot(7) = V_T*sin(psi);

 % Calculations of Guidance system controls and its subsystems

 lambda = Y_R/R; % angular error between aircraft and centre line;
    using small angle approx.
 % coupler of the overall Lateral Beam Guidance System; psi_c is
     outputted
 % from coupler into aircraft and lateral autopilot controller to
     output an
 % appropriate heading angle psi, which will influence the lateral
     speed,
 % dY_R/dt (xdot(7))
 psi_c = -G_c * lambda; % commanded heading angle [rad]

 % aircraft and lateral autopilot subsystem dynamic equations
 % outer loop compares the commanded heading, psi_c, and the actual
 % heading, psi; this comparison passes through the directional gyro
     , which
 % is represented by gain K_D. The resulting signal is the required
     (commanded) roll
 % angle, phi_c
 phi_c = K_D * (psi_c - psi); % [rad]
 % once the commanded roll angle is found, it is compared with the
     actual
 % roll angle, phi, through the second loop, where it passes through
      a
 % vertical gyro, represented by gain K_V. This is the commanded
     roll rate,
 % dphi_c/dt = p_c (recall that p = roll rate)
 p_c = K_V * (phi_c - phi); % [rad/s]
 % The final loop uses the outputted commanded roll rate to measure
     an error signal e which uses the
 % aircraft's roll rate through the roll rate gyro, represented by
     gain
 % K_R, and compares it to the commanded roll rate (see fig. 6)
 e = p_c - (p * K_R); % error measures offset of measured roll rate
     from commanded roll rate [rad/s]
 % For the aileron servo-motor subsystem, the aileron angular
     deflection is
 % regulated so that it follows and is compared to the reference
     signal e,
 % and then amplified by gain K_P to produce the input voltage for
     the
 % motor, V_A
 V_A = K_P * (e - delta_a); % [V]

end
```

B SIMULINK Block Model

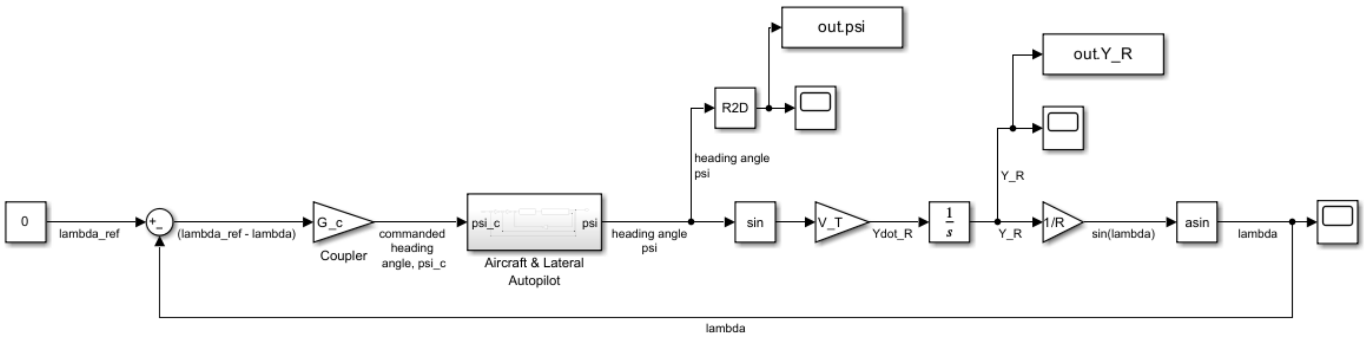


Figure 10: Overall System Block Model of the Guidance System

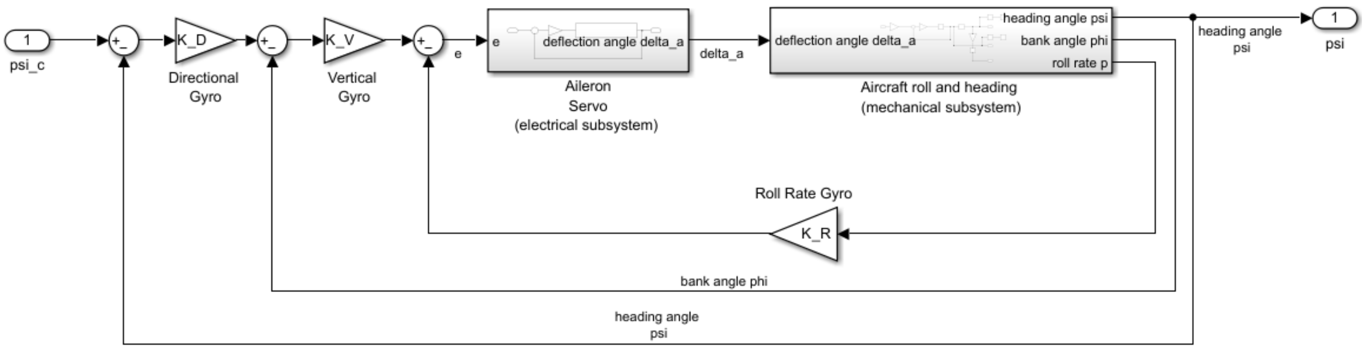


Figure 11: Sub-system Block Model of the lateral autopilot system

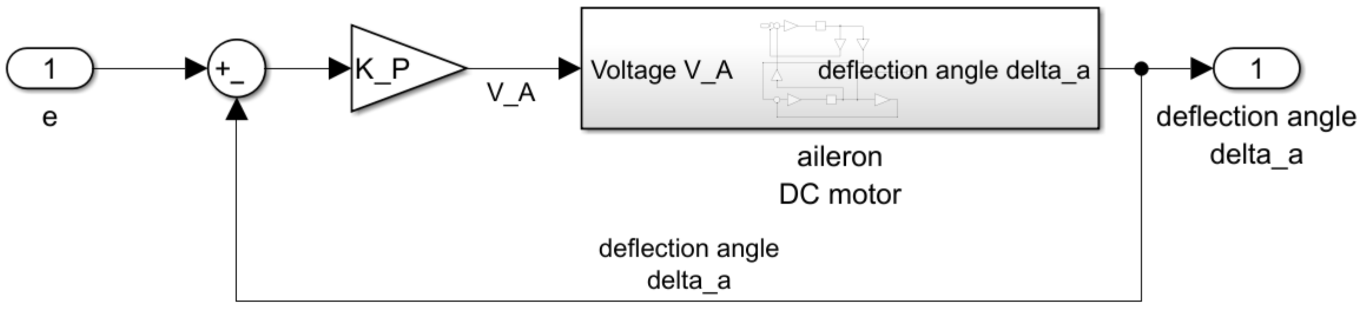


Figure 12: Sub-system Block Model of the aileron servo system

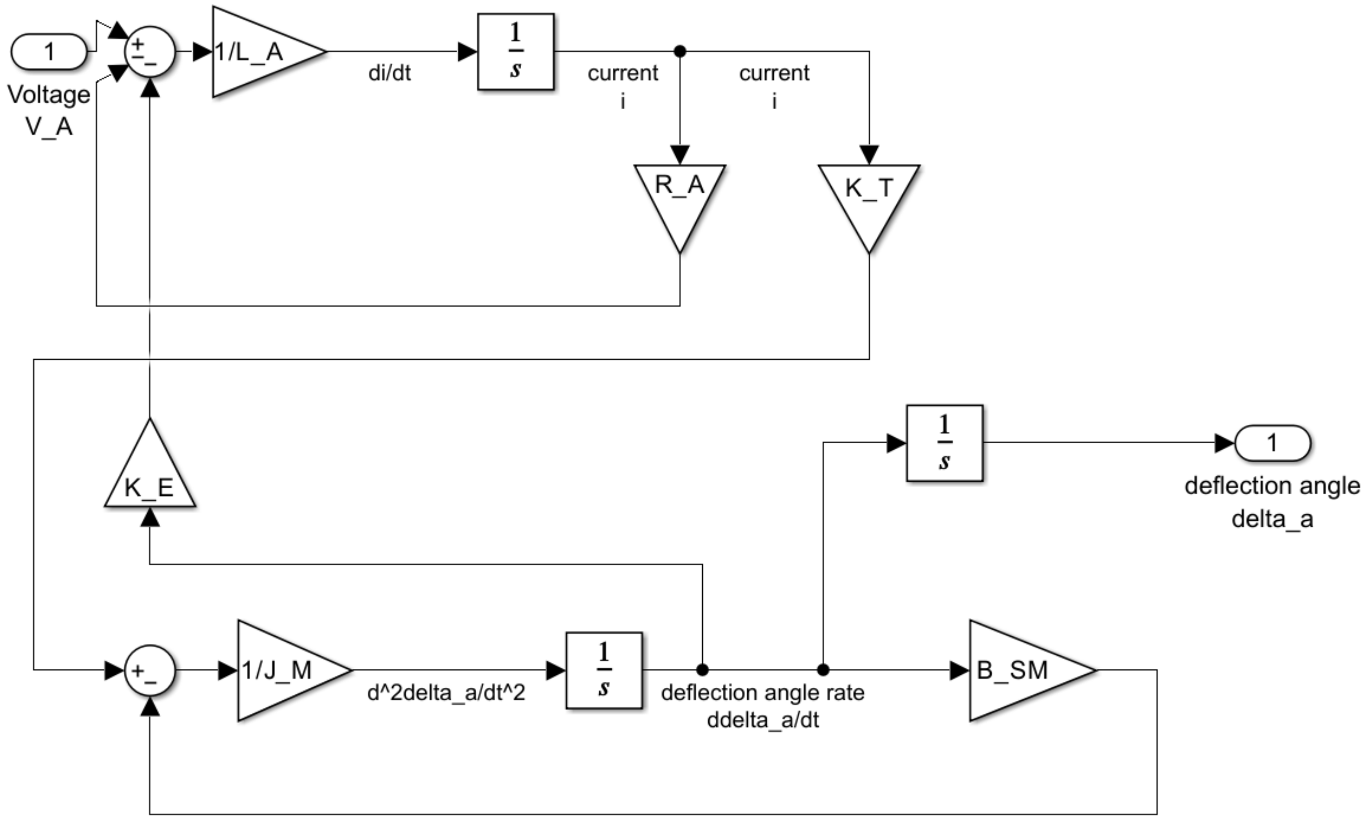


Figure 13: Sub-system Block Model of the aileron DC motor system

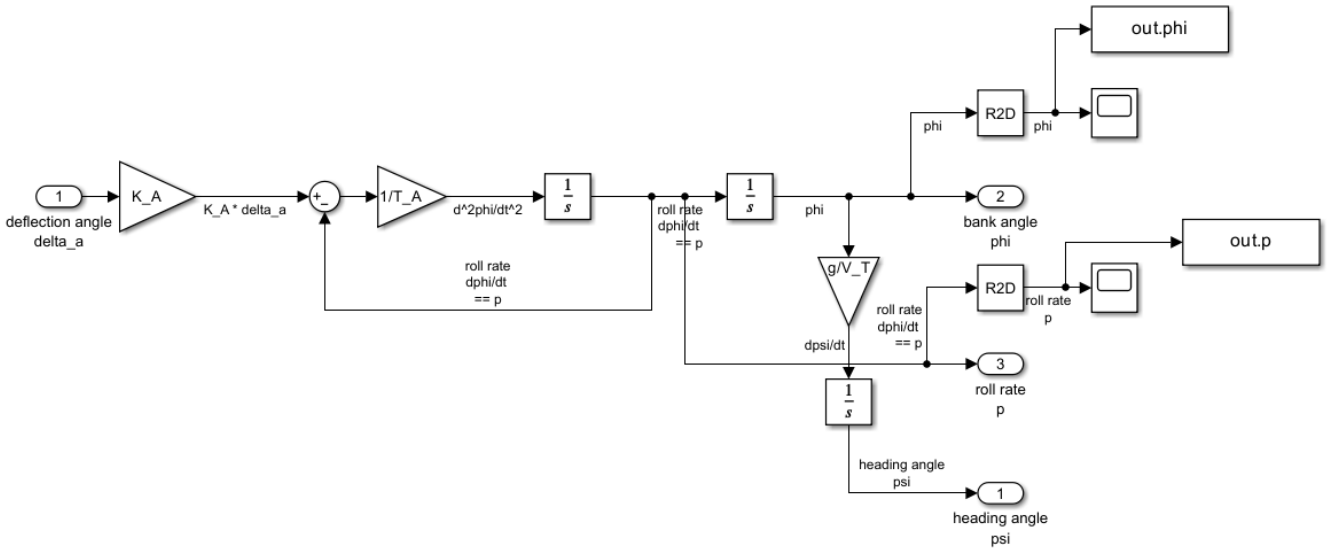


Figure 14: Sub-system Block Model of the roll and heading system