Simulation of Engineering System 3 (ENG3036)

Assignment Report Template Part 1

Student Name: Andy Mannikum

Student Number: 2811078M
Assignment Topic: Instrument Landing System Lateral Beam Guidance System

1.0 Introduction for Part 1

This report will go over the modelling and results of an instrumental landing system (ILS) lateral beam guidance. First, we will go over the state space model derivation, then the model for this system created in MatLab. Initial conditions as well as integration method as well as step size will be justified before demonstrating the results as well
as comparing and analysing between Simulink and MatLab. The Simulink block diagram will then be presented and
finally a brief conclusion will end this report. This report aims to validate the MatLab model as the entire system's response is entirely determined by a set of equations with known values.

Mathematical Modelling & Continuous Time Simulation

1. State Space Model Derivation

From the assignment specification, 5 dynamic equations were identified to be converted into a state space form.

$$\sin\psi \times V_T = \frac{dY_R}{dt} \quad \textit{(eq 1)}$$

$$L_A\frac{di}{dt} + R_A i + K_E\frac{d\delta_a}{dt} = V_A \quad \textit{(eq 2)}$$

$$J_M\frac{d^2\delta_a}{dt^2} + B_{SM}\frac{d\delta_a}{dt} = k_T i \quad \textit{(eq 3)}$$

$$T_A\frac{d^2\phi}{dt^2} + \frac{d\phi}{dt} = k_A\delta_a \quad \textit{(eq 4)}$$

$$\frac{d\psi}{dt} = \frac{g}{V_T}\phi \quad \textit{(eq 5)}$$

Equations 2, 3 and 4 can then rearranged to state space form:

$$\frac{di}{dt} = \frac{V_A - R_A i - K_E\frac{d\delta_a}{dt}}{L_A} \quad \textit{(eq 2.1)}$$

$$\frac{d^2\delta_a}{dt^2} = \frac{k_T i - B_{SM}\frac{d\delta_a}{dt}}{J_M} \quad \textit{(eq 3.1)}$$

$$\frac{d^2\phi}{dt^2} = \frac{k_A\delta_a - \frac{d\phi}{dt}}{T_A} \quad \textit{(eq 4.1)}$$

We can now assign the following state variables:

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

Next, we take the derivative of each state variable:

$xdot_1 = \frac{dY_R}{dt}$ $xdot_2 = \frac{di}{dt}$ $xdot_3 = \frac{d\delta_a}{dt}$ $xdot_4 = \frac{d^2\delta_a}{dt^2}$ $xdot_5 = \frac{d\phi}{dt}$ $xdot_6 = \frac{d^2\phi}{dt^2}$ $xdot_7 = \frac{d\psi}{dt}$

Now we can substitute the state variables and the reduced form into the equations:

$$xdot_1 = \sin x_7 \times V_T \quad \textit{(eq 6)}$$

$$xdot_2 = \frac{V_A - R_A x_2 - K_E x_4}{L_A} \quad \textit{(eq 7)}$$

$$xdot_3 = x_4 \quad \textit{(eq 8)}$$

$$xdot_4 = \frac{k_T x_2 - B_{SM} x_4}{J_M} \quad \textit{(eq 9)}$$

$$xdot_5 = x_6 \quad \textit{(eq 10)}$$

$$xdot_6 = \frac{k_A x_3 - x_6}{T_A} \quad \textit{(eq 11)}$$

$$xdot_7 = \frac{g}{V_T}x_5 \quad \textit{(eq 12)}$$

Please note that this report form must not be more than 6 pages long.

2. *Matlab Model Function*

```
%
% DYNAMIC SEGMENT
%
% The DYNAMIC SECTION is the main section of a simulation program. This is
% evaluated for every time interval during the simulation. Therefore it is
% an interative process.
%dynamics section

tout = [];
lout = [];
for time = 0:stepsize:EndTime
if rem(time,comminterval)==0
i = i+1;            %increment
tout(i) = time;     %store time
Yr(i)=x(1);         %assigns lateral displacement
current(i)=x(2);    %current
delta(i)=x(3);      %delta
deltadot(i)=x(4);   %deltadot
phi(i)=x(5);        %phi
phidot(i)=x(6);     %phidot
psi(i)=x(7);        %psi
lout(i)=lambda;     %store lambda
end
end
% end of storage
%control section
Ro = 6000 - Vt * time; % Inside the loop
psic = GC*(lambdaref - lambda);
lambda = asin(x(1)/Ro);
%derivative section
xdot = model(x,psic);

%integration section
x = rk4int("model",stepsize,x,psic);
% END OF INTEG SECTION
% END OF DYNAMIC SEGMENT
```

Figure 1: Dynamic, control, derivative and integration section in MatLab

Figures 1 through 5 depict the MatLab code with figures 1 and 2 representing the external functions called upon in the derivative and integration sections as seen in figure 4. Figure 3 represents the state space equations seen in section 1.1 (equations 6 through 12).

```
% Define and initialise the model input and any model parameters
% as global variables so that they can be read in the function model.m.

global BSM g GC JM KA KD KE KP KR KT KV LA RA TA Vt;

BSM=0.7; g=9.81; GC=45.5; JM=0.006; KA=1.2; KD=0.9; KE=0.9; KP=52.5; KR=1.2; KT=1.7; KV=1.3; LA=0.2; RA=10; TA=2; Vt=55;

%Define parameters fpr the simulation
stepsize = 0.001;               %Integration step size
comminterval = 0.01;            %Communications interval
EndTime = 200;                   %Duration of the simulation (final time)
i = 0;                          %Initialise counter for data storage
% Initial conditions of all states and state derivatives
Va = 0;
lambdaref = 0;
lambda = 0;
psic = -0.349;
Ro = 6000;
x = [150;0;0;0;0;0;-0.349]; %[lateral velocity,current,angular displacement,rate of angular displacement,roll angle,rate
xdot = [0;0;0;0;0;0;0]; %[lateral acceleration, rate of change of current, rate of change of angular displacement, accele

%
% END OF INITIAL SEGMENT - all parameters initialised
%
```

Figure 2: Initial parameters defining the simulation in MatLab

Please note that this report form must not be more than 6 pages long.

```
function xdot = model(x,psic)

global BSM g GC JM KA KD KE KP KR KT KV LA RA TA Vt;

Va = ((KV*(KD*(psic-x(7))-x(5))-KR*x(6)-x(3))*KP);

xdot(1,1) = sin(x(7))*Vt;                    %yrdot
xdot(2,1) = (Va-(RA*x(2))-(KE*x(4)))/LA;     %idot
xdot(3,1) = x(4);                            %deltadot
xdot(4,1) = ((KT*x(2))-(BSM*x(4)))/JM ;       %deltadotdot
xdot(5,1) = x(6);                            %phidot
xdot(6,1) = ((KA*x(3))-x(6))/TA  ;            %phidotdot
xdot(7,1) = (g*x(5))/Vt;                     %psidot

end
```

Figure 3: State space equations in MatLab

```
function x = rk4int(modelname, h, x, u)
% This function performs one 4th Order Runge-Kutta integration step

k1 = h*feval(modelname, x, u);      % evaluate derivative k1
k2 = h*feval(modelname, x+k1/2, u); % evaluate derivative k2
k3 = h*feval(modelname, x+k2/2, u); % evaluate derivative k3
k4 = h*feval(modelname, x+k3, u);       % evaluate derivative k4

x = x + (k1 + 2*k2 + 2*k3 + k4)/6;      % averaged output
```

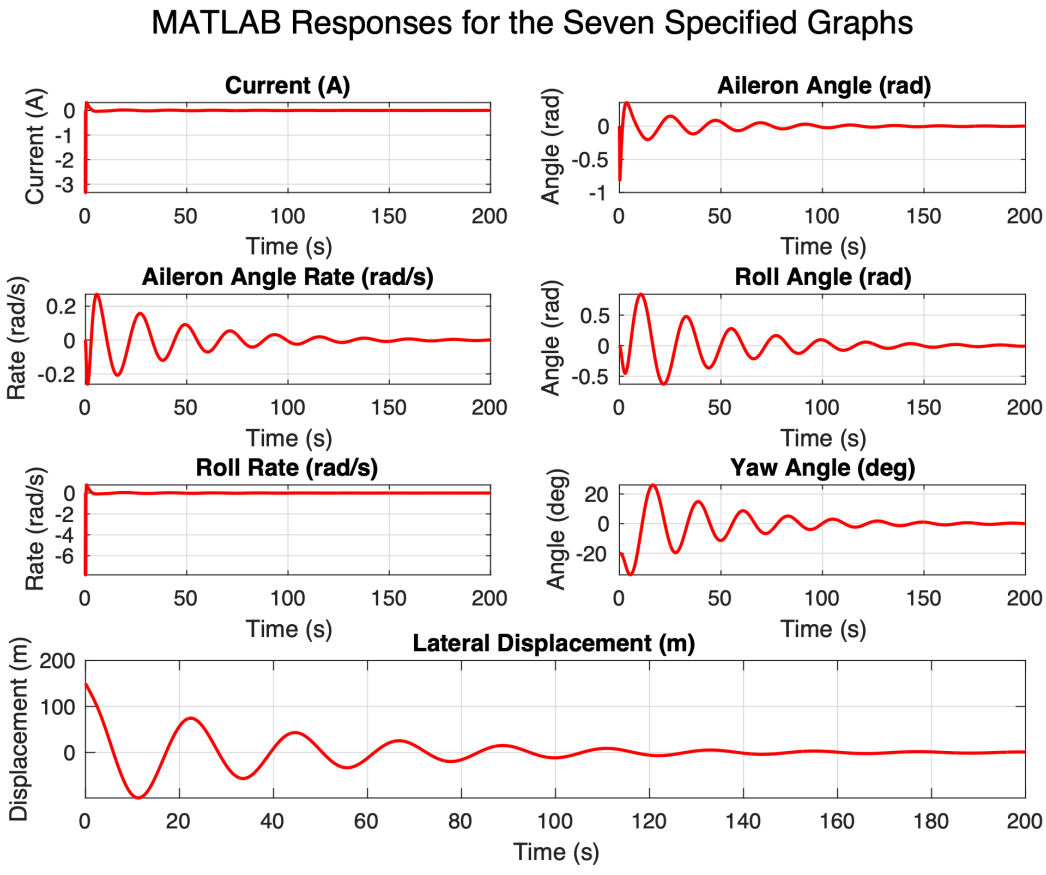
Figure 4: MatLab Runge-kutta integration method

3. Initial Conditions, Numerical Integration Solver & Step-Size Selection

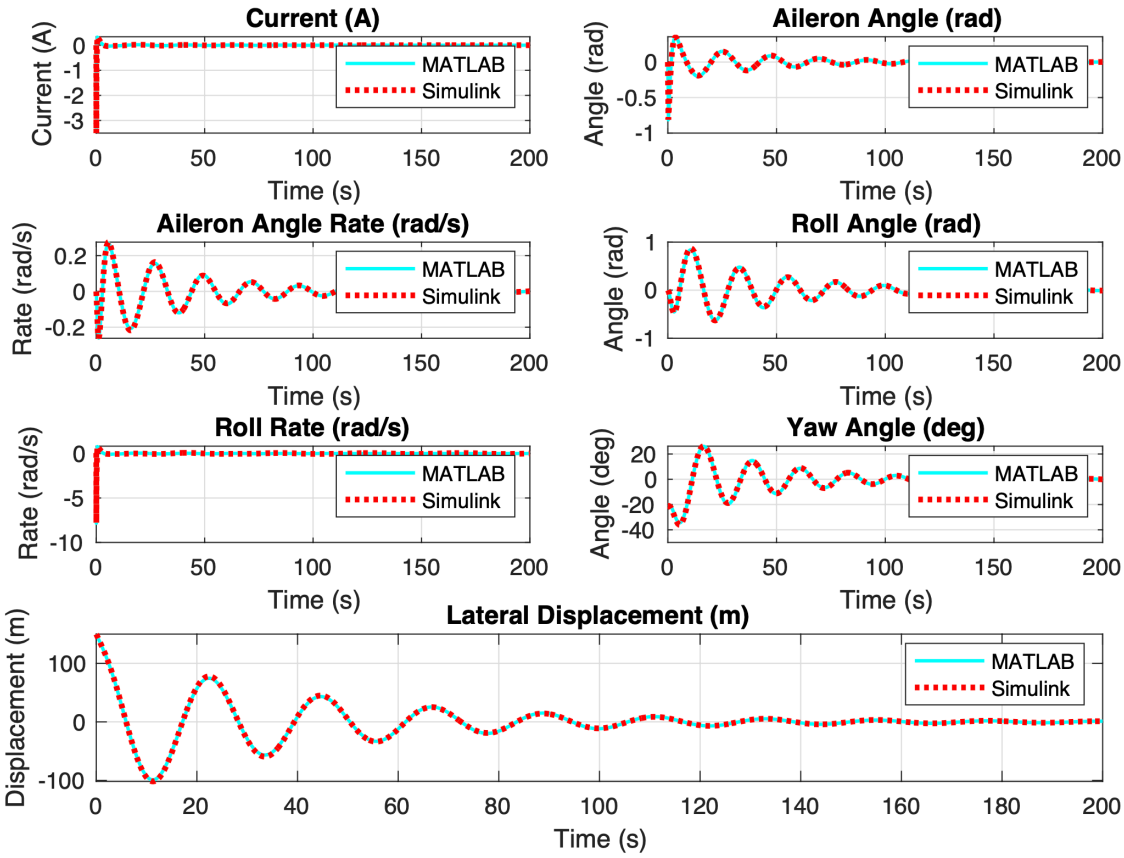
ψo = -20°, ϕo = 0°, *Ro* = 6000m, *YRo* = 150m, *VT* = 55 m/s

The fourth order Runge-Kutta (RK4) method of integration was chosen due to its general higher accuracy compared to Euler's method as well as it generally being more stable. The trade off with Euler's method is that Euler's method tends to be simpler and use less computational resources. Being in a situation where any error and misjudgement can lead to catastrophic accidents, higher accuracy is preferred at the cost of computational resources. Although for comparing with Simulink both methods can be used. For the step size selection, 0.001s was found to be the most appropriate. This was selected due to RK4 being well suited to compute at this size as well as by assuming equation 3 relating $d2\delta a/dt2$ with $d\delta a/dt$ and the current i. The total time was chosen to be 200 seconds as this would replicate well enough the time the plane would take to reach the runway.

4. Simulation Responses and Analysis



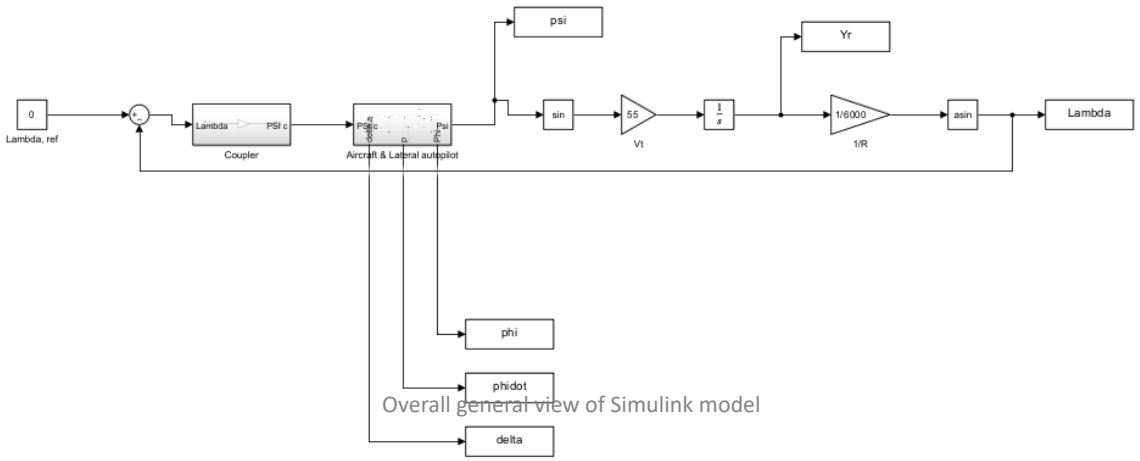
Please note that this report form must not be more than 6 pages long.



The received graphs from both MatLab and Simulink. We can see that almost every graph is near identical with each other validating the models. This accuracy can be achieved through the Simulink settings to be modified to have the same step and integration computation as the MatLab.
We do however witness a large number of oscillations in the reactions of the aircraft. They are slightly dampened as most clear. Ideally, we would want a critically damped system to remove the oscillations.

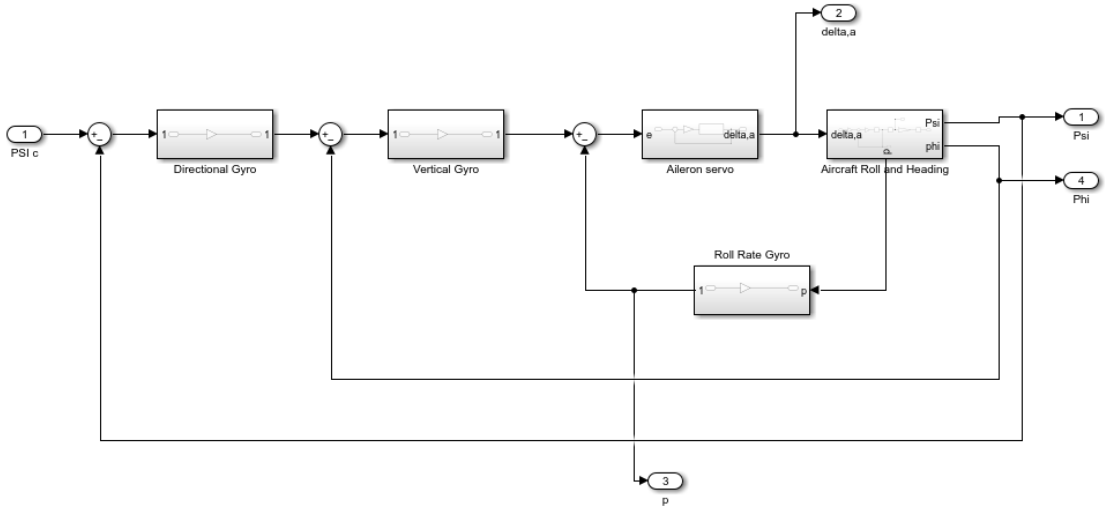
Block Diagram & Validation

5. Simulink Block Diagram

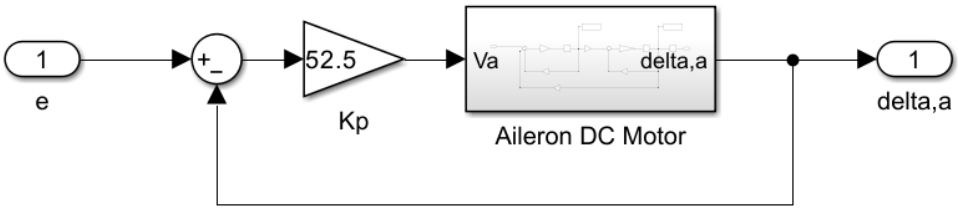


Overall general view of Simulink model

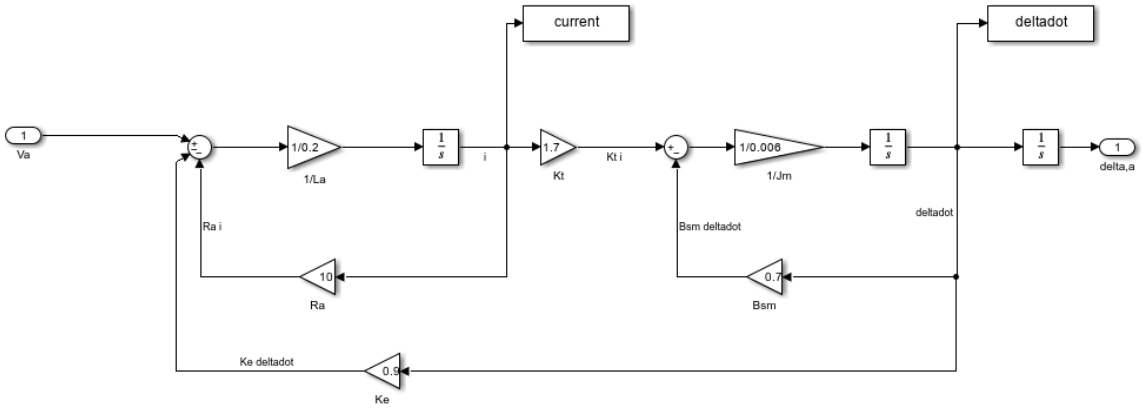
Please note that this report form must not be more than 6 pages long.



Aircraft and Lateral Autopilot subsystem

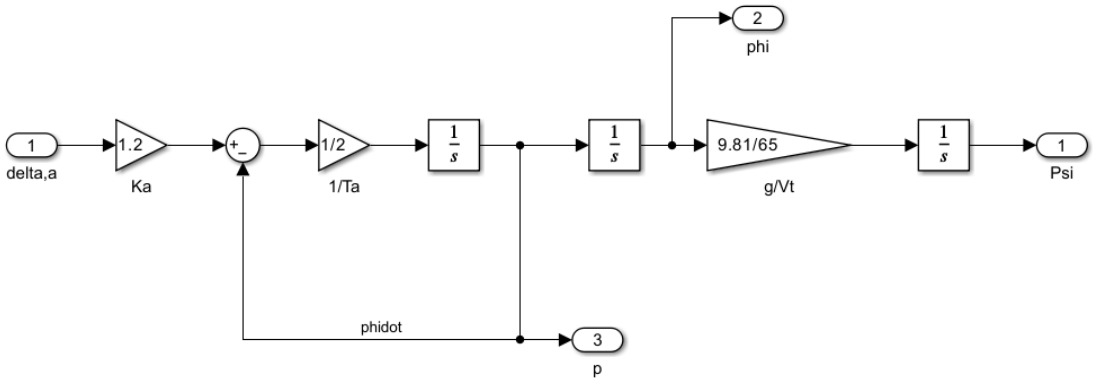


Aileron servo subsytem



Aileron DC motor

Please note that this report form must not be more than 6 pages long.



Aircraft Roll and Heading Subsystem

6. Validation Using Simulink Responses and Analysis

MATLAB and Simulink results showed excellent agreement, confirming the accuracy of the mathematical model. Plots comparing the responses (to be included) demonstrate overlapping curves, validating the model.

7. Conclusions for Part 1

The modeling and simulation of the ILS Lateral Beam Guidance System confirm that the design is effective. The system aligns the aircraft with minimal error and stabilizes the heading angle efficiently. Future improvements could explore adaptive coupler gains for varying weather conditions or dynamic runway alignments.This system does not take into account some factors that can effect this system. Real conditions such as wind and rain can greatly affect this system. The change in distance between the plaene and the ILS glideslope transmitter is also not taken into account, instead just the initial distance is used.

Please note that this report form must not be more than 6 pages long.