College of Engineering
School of Aeronautics and Astronautics

AAE 421
Flight Dynamics and Controls

HW 3
Aircraft Simulink Practice

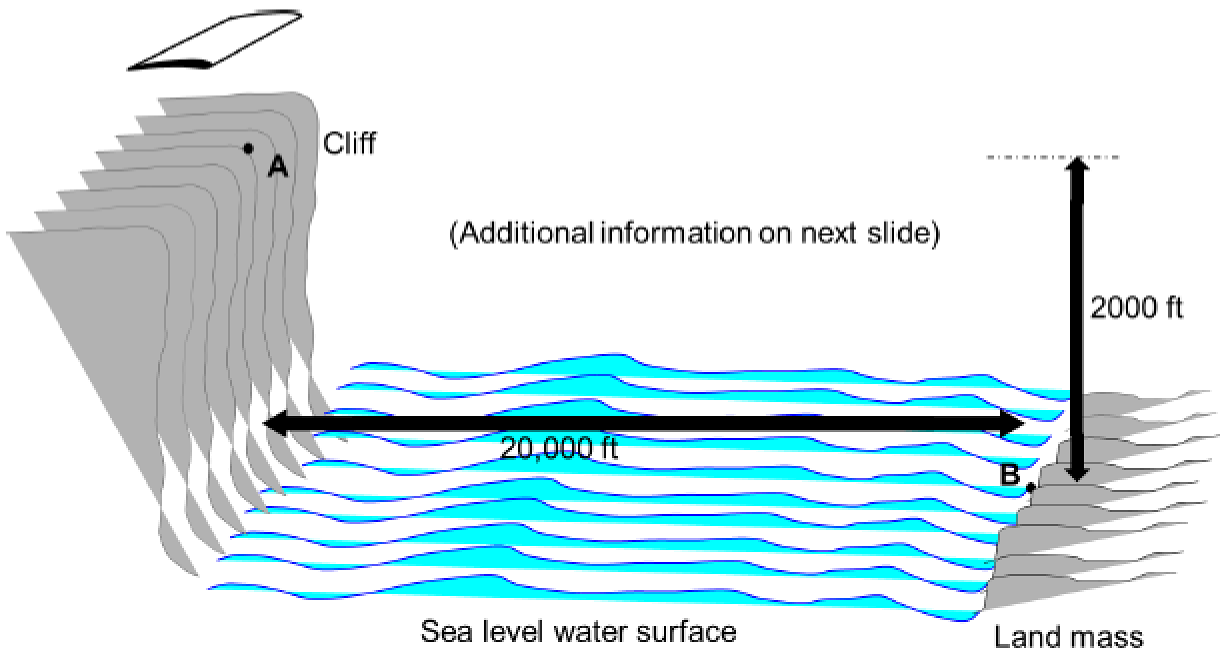
Author:
Tomoki Koike

Supervisor:
Ran Dai

October 15th, 2020
Purdue University
West Lafayette, Indiana

Problem 1. (30pts)

A flying wing glider configuration rests on the flat top of a mountain 2000 ft (MSL) high. The mountain is surrounded with water (sea level) and has steep cliffs on all sides. The nearest land mass is located 20,000 ft in the distance at sea level. Can the glider configuration fly from the mountain top to the land mass without touching water? Assuming initial condition $V_h = 0,\ V_x = 51 ft/s,\ h = 2000 ft,\ x = 0$, find the time history of altitude and range for wing glider from starting point to landing point when $C_L = 0.9$.



The equations for this system are:

$$\dot{V}_x = 0.0153V^2(-C_D cos\gamma - C_L sin\gamma) \tag{1}$$

$$\dot{V}_h = 0.0153V^2(C_L cos\gamma - C_D sin\gamma) - 32.17 \tag{2}$$

$$\dot{h} = V_h \tag{3}$$

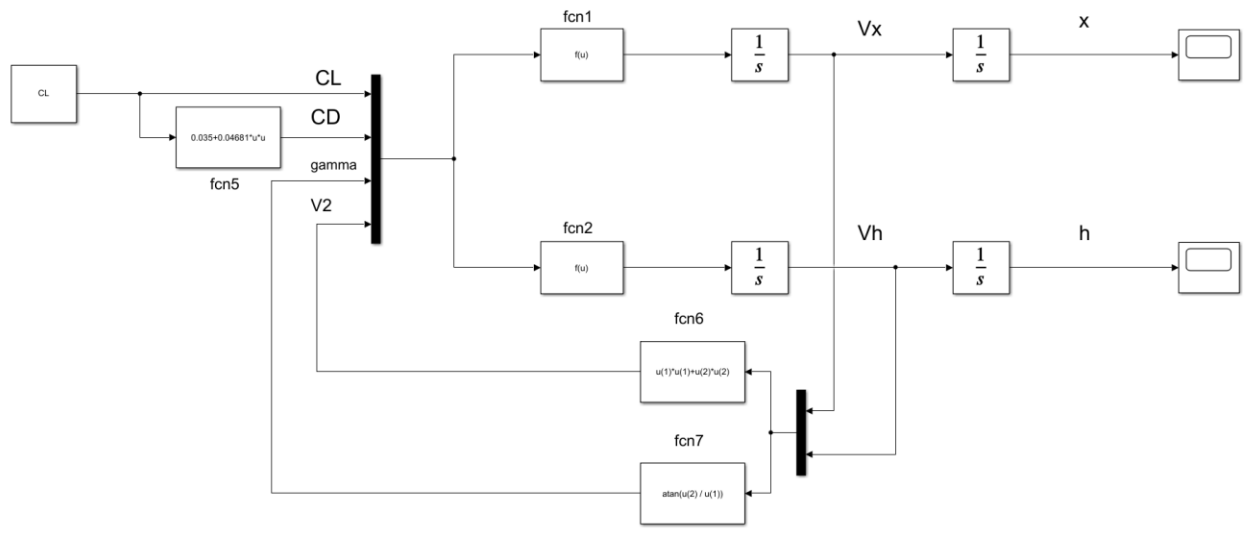
$$\dot{x} = V_x \tag{4}$$

$$C_D = 0.035 + 0.04681C_L^2 \tag{5}$$

$$V^2 = V_x^2 + V_h^2 \tag{6}$$

$$\gamma = arctan\left(\frac{V_h}{V_x}\right) \tag{7}$$

The Simulink model that we have made is the following.



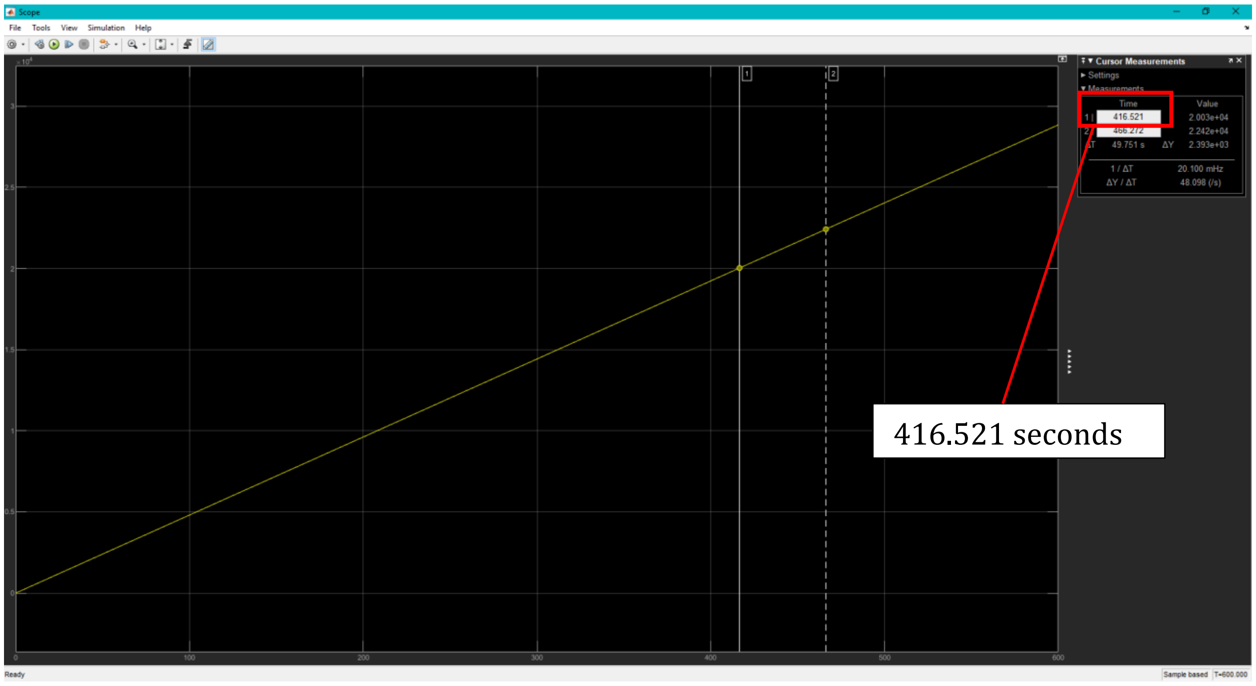
The function blocks indicated as "fcn" have numbers that correspond to the function numbers noted on the previous page inside the problem statement.

The initial conditions and value of the lift coefficient is initialized in the MATLAB file and the simulation is ran for a time of 600 seconds. The code is shown below.

```matlab
% AAE 421 HW3 P1 MATLAB CODE
% TOMOKI KOIKE
clear all; close all; clc;

% Setup initial conditions and input
Vh_i = 0;
Vx_i = 51;  % ft/s
h_i  = 2000;  % ft
x_i  = 0;  % ft
CL   = 0.9;  % lift coefficient

simout = sim('simulink\hw3_p1_glider.slx')
```

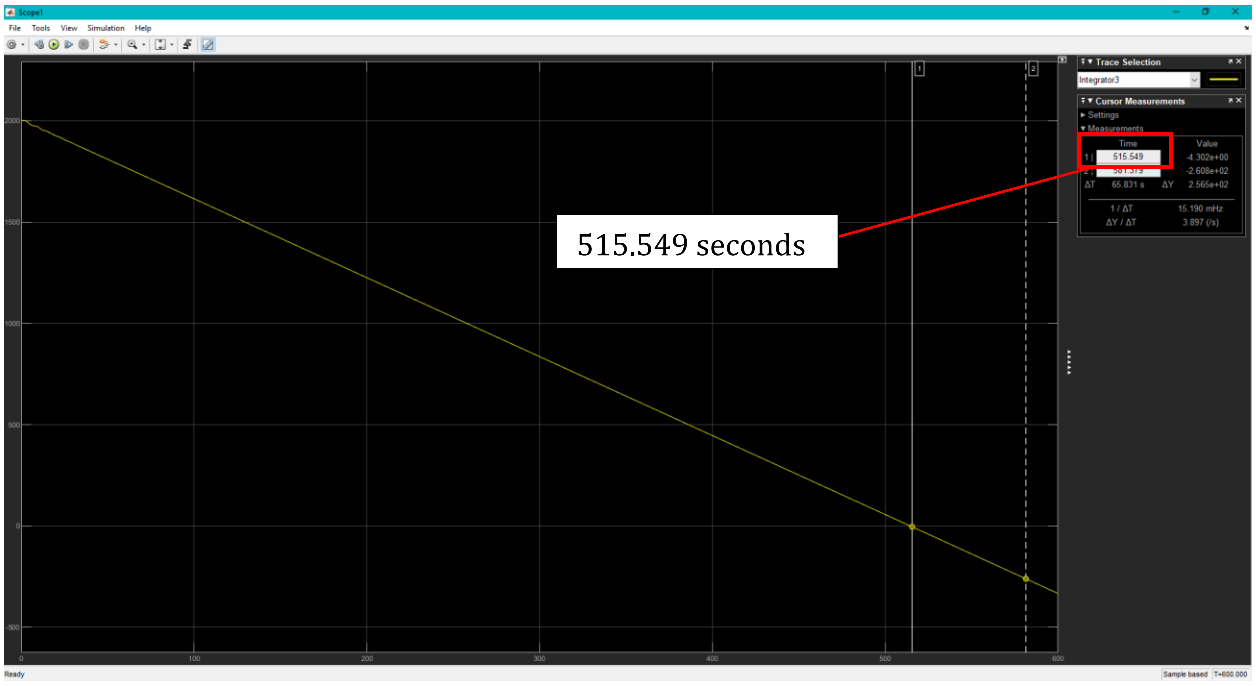
The resulting graph of the simulation are on the next page.

x-plot:



```
Cursor Measurements
▸ Settings
▾ Measurements
        Time            Value
1 |   416.521         2.003e+04
2 |   466.272         2.242e+04
ΔT    49.751 s    ΔY  2.393e+03

      1 / ΔT          20.100 mHz
      ΔY / ΔT         48.098 (/s)
```

h-plot:



```
Trace Selection
Integrator3

Cursor Measurements
▸ Settings
▾ Measurements
        Time            Value
1 |   515.549        -4.302e+00
2 |   581.379        -2.608e+02
ΔT    65.831 s    ΔY  2.565e+02

      1 / ΔT          15.190 mHz
      ΔY / ΔT         3.897 (/s)
```

From the plots, we can see that the glider reaches 20,000 ft at 416.521 s and the glider reaches sea level at 515.549 s. This means that the glider CAN reach the land mass without touching the water.

Problem 2. (40pts)

The equations below represent the longitudinal and lateral-directional equations of motion for the Boeing 727 aircraft. The pitch, roll, yaw variables (p, q, r) have been reinstated in the equations. The inertial orientation and position equations must be included in their full nonlinear form. The reference condition for the airplane is a steady level flight condition: W = 130,000 lb, Trim Velocity = 832 ft/sec, Trim altitude = 27,000 ft.

$$\dot{u} = -0.01008u + 2.425\dot{\alpha} + 7.055\alpha - 2.308q - 32.17\theta - 0.9679\Delta\delta_\theta + 0.004\Delta\delta_T \quad (1)$$

$$\dot{\alpha} = -0.0001u - 0.9967\alpha + 1.0046q + 0.03867(cos\theta cos\phi - 1.0) - 0.03827\Delta\delta_\theta \quad (2)$$

$$\dot{q} = 0.8277\dot{\alpha} - 0.000013u - 3.326\alpha - 1.337q - 2.57\Delta\delta_\theta \quad (3)$$

$$\dot{p} = -2.5612p + 0.7245r - 0.0677\dot{\beta} - 14.0174\beta + 0.4395\Delta\delta_\alpha + 0.3830\Delta\delta_r \quad (4)$$

$$\dot{r} = 0.8476p - 0.4155r + 0.3564\dot{\beta} + 6.6931\beta - 0.1556\Delta\delta_a - 0.7090\Delta\delta_r \quad (5)$$

$$\dot{\beta} = -0.2127\beta - 0.0011p - 0.9957r + 0.03867cos\theta sin\phi + 0.03259\Delta\delta_r \quad (6)$$

$$tan\alpha = \frac{w}{u_0 + u} \quad (7)$$

$$tan\beta = \frac{v}{u_0 + u} \quad (8)$$

$$\dot{\theta} = qcos\phi - rsin\phi \quad (9)$$

$$\dot{\phi} = p + qsin\phi tan\theta + rcos\phi tan\theta \quad (10)$$

$$\dot{\psi} = (qsin\phi + rcos\phi)sec\theta \quad (11)$$

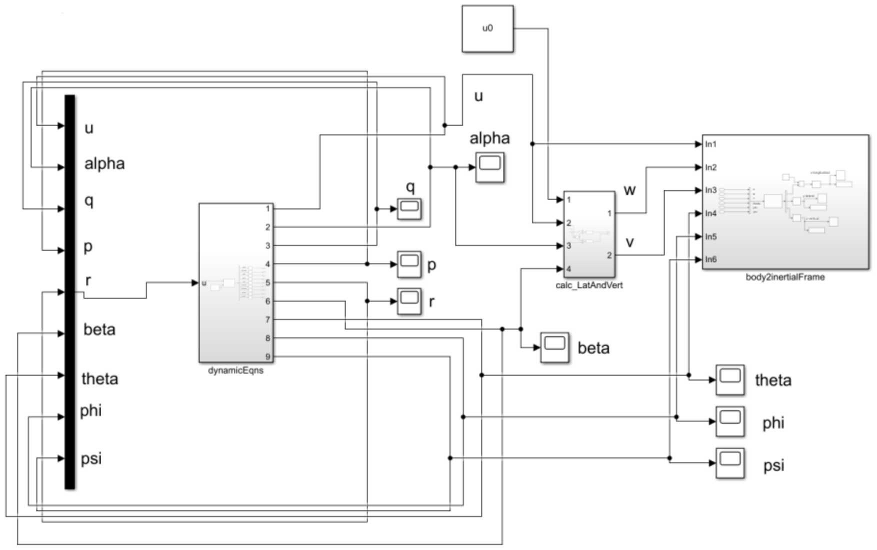
Prepare a SIMULINK simulation circuit for the Boeing B727 aircraft using the equations of motion given above. A trajectory module is required to be made available that will allow system plots of flight path range vs altitude and flight path range vs lateral displacement, or you may incorporate your own. Upon completion of your overall simulation circuit with appropriate debugging perform the following simulation tasks.

(a) All initial conditions set to zero except alpha. Set initial condition for alpha at 0.1 rad. Keep all control inputs set to zero. Show plots for the trajectory of flight path range vs altitude.
(b) All initial conditions set to zero except beta. Set initial condition for beta at 0.1 rad. Keep all control inputs set to zero. Show plots for the trajectory of flight path range vs lateral displacement.

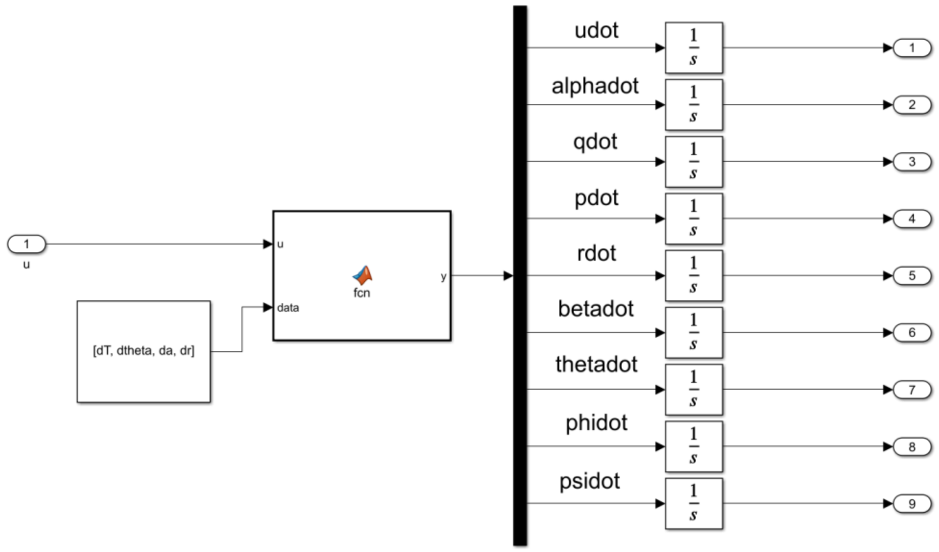
From the problem statement, we know that

$$\begin{aligned} W &= 130{,}000 lb \\ u_0 &= 832 ft/s \\ w_0 &= 27{,}000 ft \end{aligned} .$$

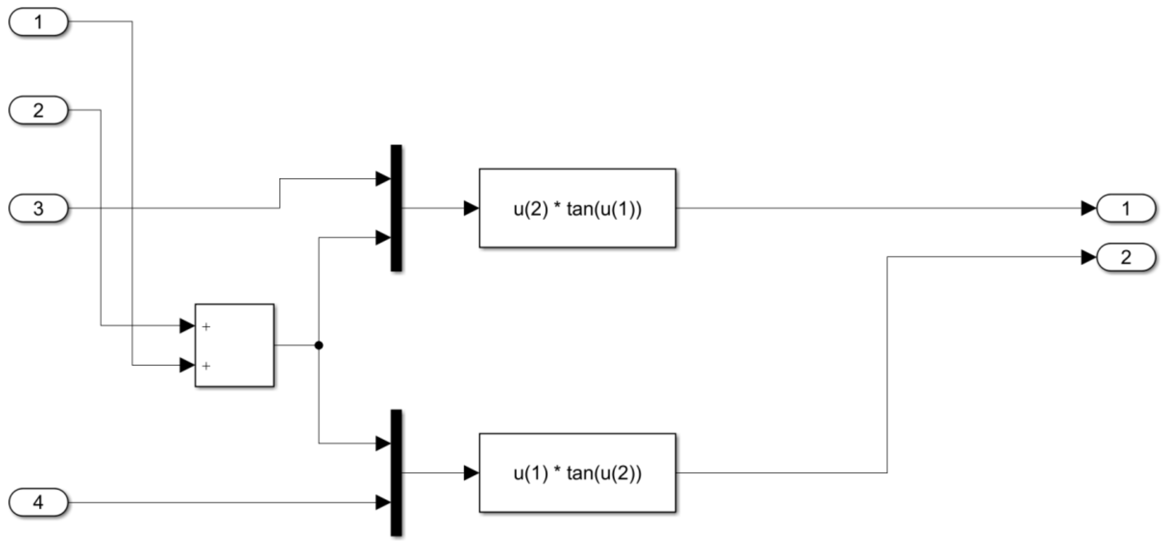
Using this model we conduct the simulation,



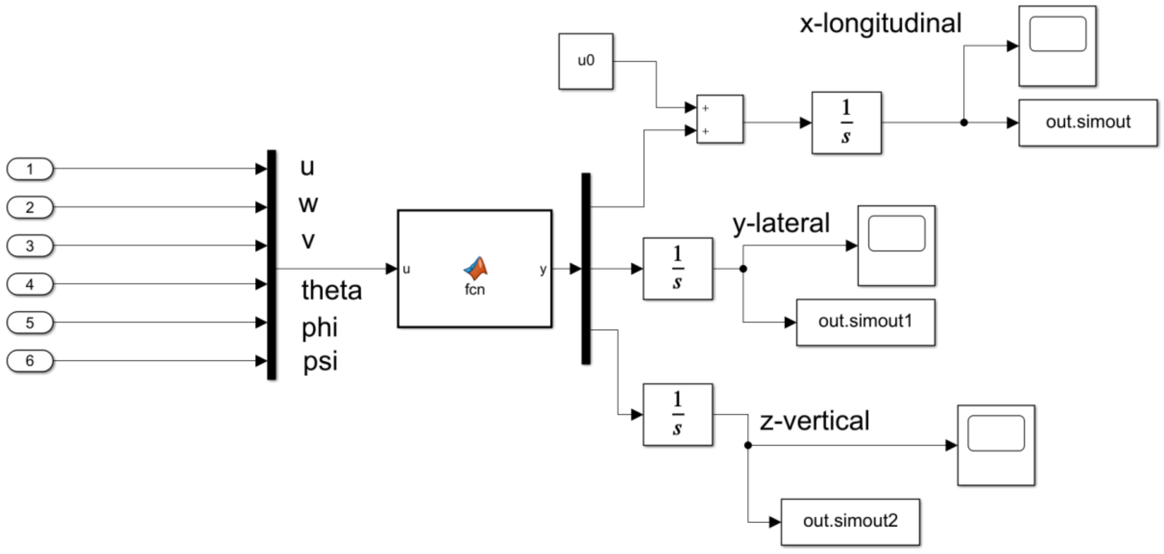
The subsystem "dynamicEqns" contains



The subsystem "calc_LatAndVert" contains



The subsystem "body2inertialFrame" contains



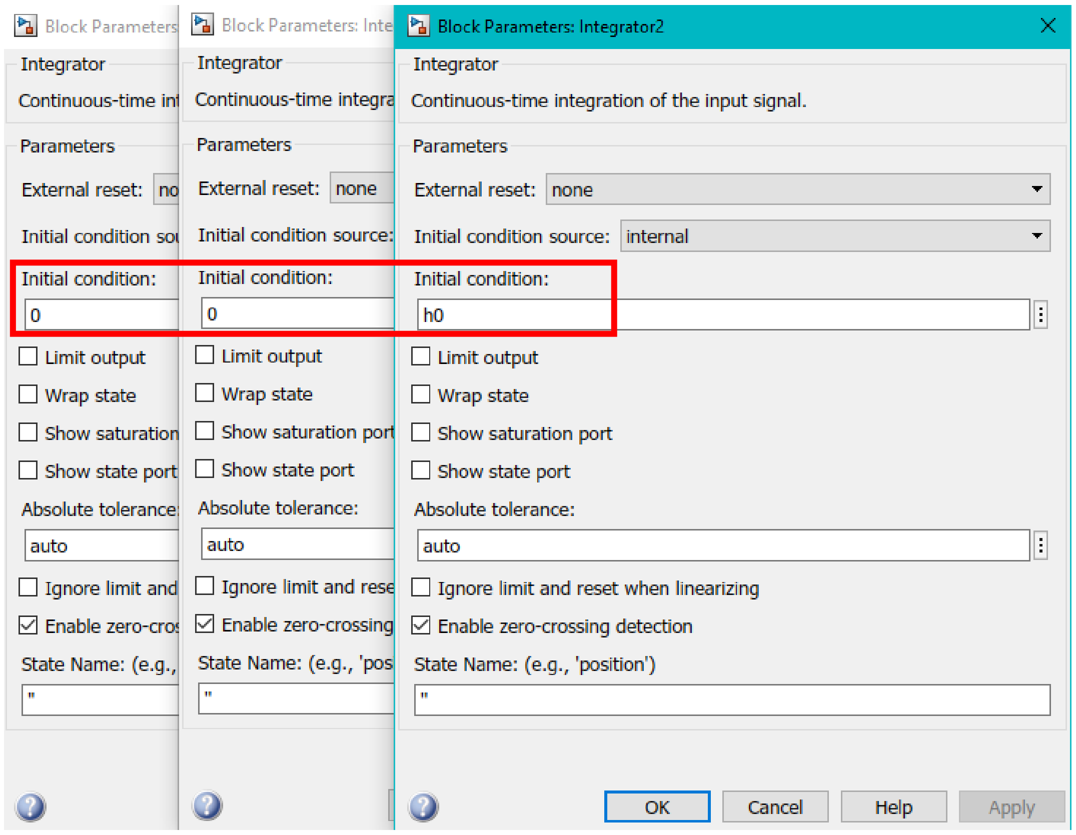
The Embedded MATLAB Block in "dynamicEqns" has the following code:

```
function y = fcn(u, data)
    % Unpack the delta inputs
    dT = data(1); dtheta = data(2); da = data(3); dr = data(4);
    % Unpack the input variables
    uv = u(1); alpha = u(2); q = u(3); p = u(4); r = u(5);
    beta = u(6); theta = u(7); phi = u(8); psi = u(9);

    % Functions
    alphadot = -0.0001*uv -0.9967*alpha +1.0046*q
+0.03867*(cos(theta)*cos(phi)-1.0)...
                -0.03827*dtheta;
    betadot  = -0.2127*beta -0.0011*p -0.9957*r 
+0.03867*cos(theta)*sin(phi)...
                +0.03259*dr;
    udot     = -0.01008*uv +2.425*alphadot +7.055*alpha -2.308*q -
32.17*theta...
                -0.9679*dtheta +0.004*dT;
    qdot     = 0.8277*alphadot -0.000013*uv -3.326*alpha -1.337*q -
2.57*dtheta;
    pdot     = -2.5612*p +0.7245*r -0.0677*betadot -14.0174*beta +0.4395*da 
+0.3830*dr;
    rdot     = 0.8476*p -0.4155*r +0.3564*betadot +6.6931*beta -0.1556*da -
0.7090*dr;
    thetadot = q*cos(phi) - r*sin(phi);
    phidot   = p + q*sin(phi)*tan(theta) + r*cos(phi)*tan(theta);
    psidot   = (q*sin(phi) + r*cos(phi))*sec(theta);

    y = [udot; alphadot; qdot; pdot; rdot; betadot; thetadot; phidot; 
psidot];
end
```

The Embedded MATLAB Block in "body2inertialFrame" has the following code:

```
function y = fcn(u)
    % Rotation transformation from body-frame to inertial-frame
    uv = u(1); wv = u(2); vv = u(3);
    theta = u(4); phi = u(5); psi = u(6);

    R3 = [cos(psi), -sin(psi), 0;
          sin(psi),  cos(psi), 0;
                 0,         0, 1];
    R2 = [ cos(theta), 0, sin(theta);
                    0, 1,          0;
          -sin(theta), 0, cos(theta)];
    R1 = [1,        0,         0;
          0, cos(phi), -sin(phi);
          0, sin(phi),  cos(phi)];
    y = R3*R2*R1 * [uv; vv; wv];
end
```

For the integrators of range, lateral displacement, and altitude set the initial conditions to their trim conditions



Block Parameters: Integrator2

Integrator

Continuous-time integration of the input signal.

Parameters

External reset: none

Initial condition source: internal

Initial condition: 0 | 0 | h0

- [ ] Limit output
- [ ] Wrap state
- [ ] Show saturation port
- [ ] Show state port

Absolute tolerance: auto

- [ ] Ignore limit and reset when linearizing
- [x] Enable zero-crossing detection

State Name: (e.g., 'position')

''

(a).

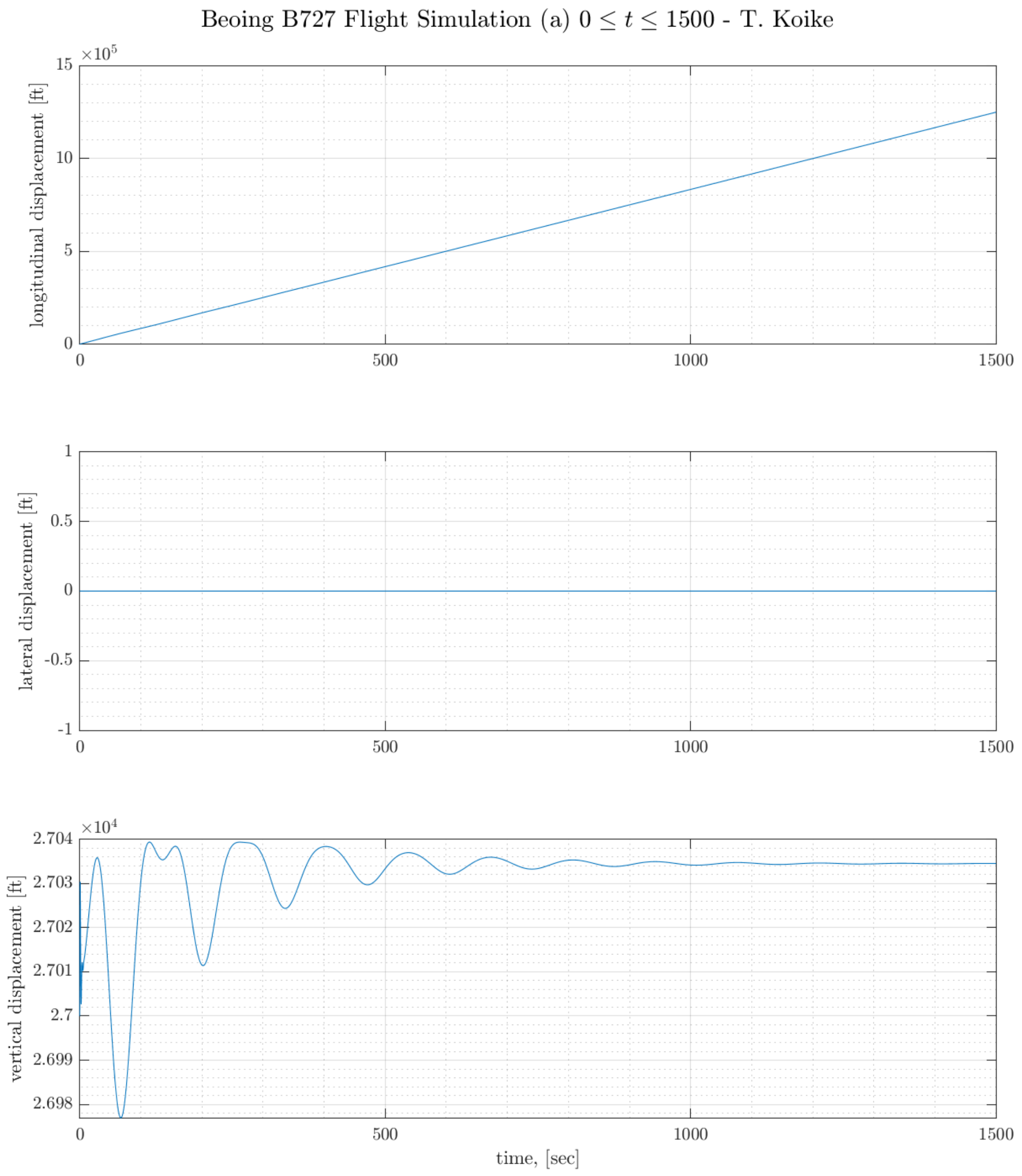
Run the simulation by defining initial conditions and system inputs and get the outputs from the simulation.

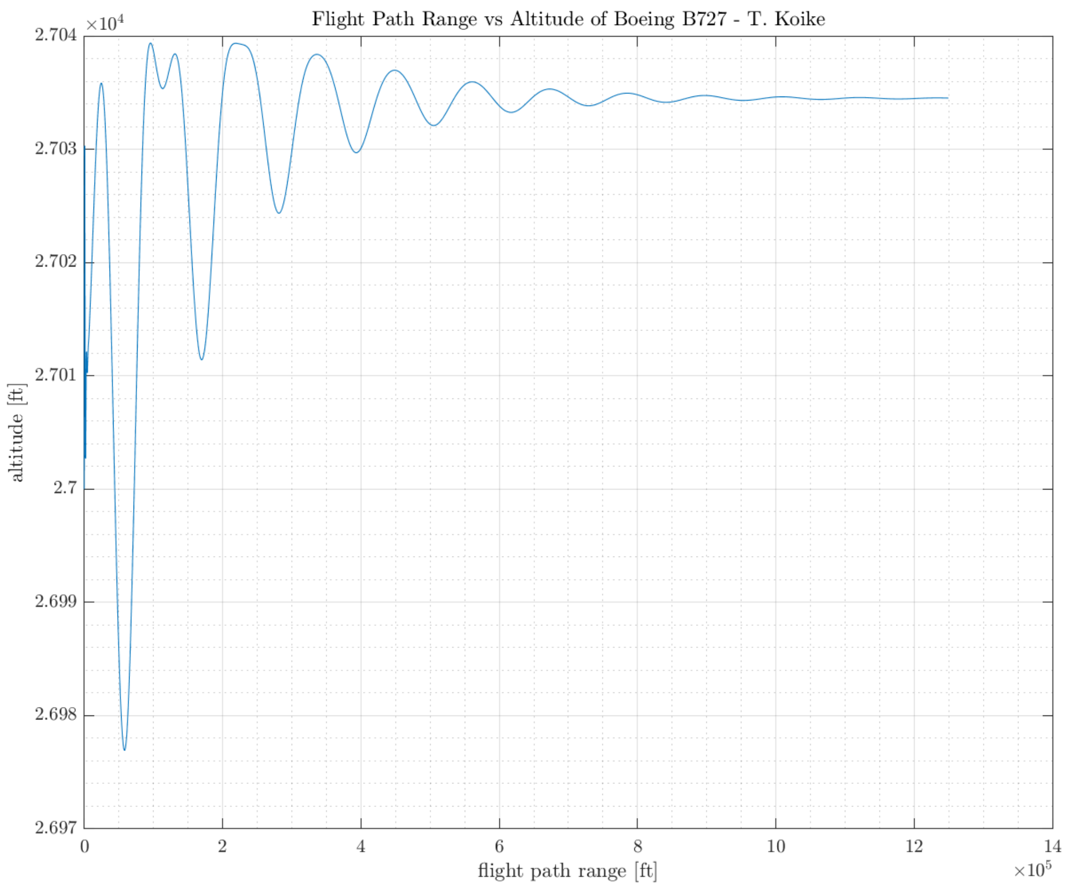
```
% Control inputs
dT = 0; dtheta = 0; da = 0; dr = 0;
u0 = 832;  % trim velocity ft/s
alpha_i = 0.1;  % initial alpha 0.1 rad
beta_i = 0;
h0 = 27000;  % trim altitude
sim_a = sim("hw3_p2_Boeing727.slx");
t = sim_a.simout.time;
long = sim_a.simout.signals.values;
lat  = sim_a.simout1.signals.values;
vert = sim_a.simout2.signals.values;
```

Then plot using the following code

```
fig1 = figure('Renderer',"painters", 'Position', [10 10 900 1000]);
subplot(3,1,1)
grid on; grid minor; box on;
plot(t,long)
ylabel('longitudinal displacement [ft]')
subplot(3,1,2)
grid on; grid minor; box on;
plot(t,lat)
ylabel('lateral displacement [ft]')
subplot(3,1,3)
grid on; grid minor; box on;
plot(t,vert)
ylabel('vertical displacement [ft]')
xlabel('time, [sec]')
sgtitle('Beoing B727 Flight Simulation (a) $0\leq t \leq 10$ - T. Koike')
saveas(fig1, '.png')

% range vs lateral displacement
fig2 = figure('Renderer',"painters", 'Position', [10 10 900 700]);
plot(long, lat)
grid on; grid minor; box on;
title('Flight Path Range vs Lateral Displacement of Boeing B727 - T. Koike')
xlabel('flight path range [ft]')
ylabel('lateral displacement [ft]')
saveas(fig2, fullfile(fdir, "p2_b_rangeVSlat.png"))
```

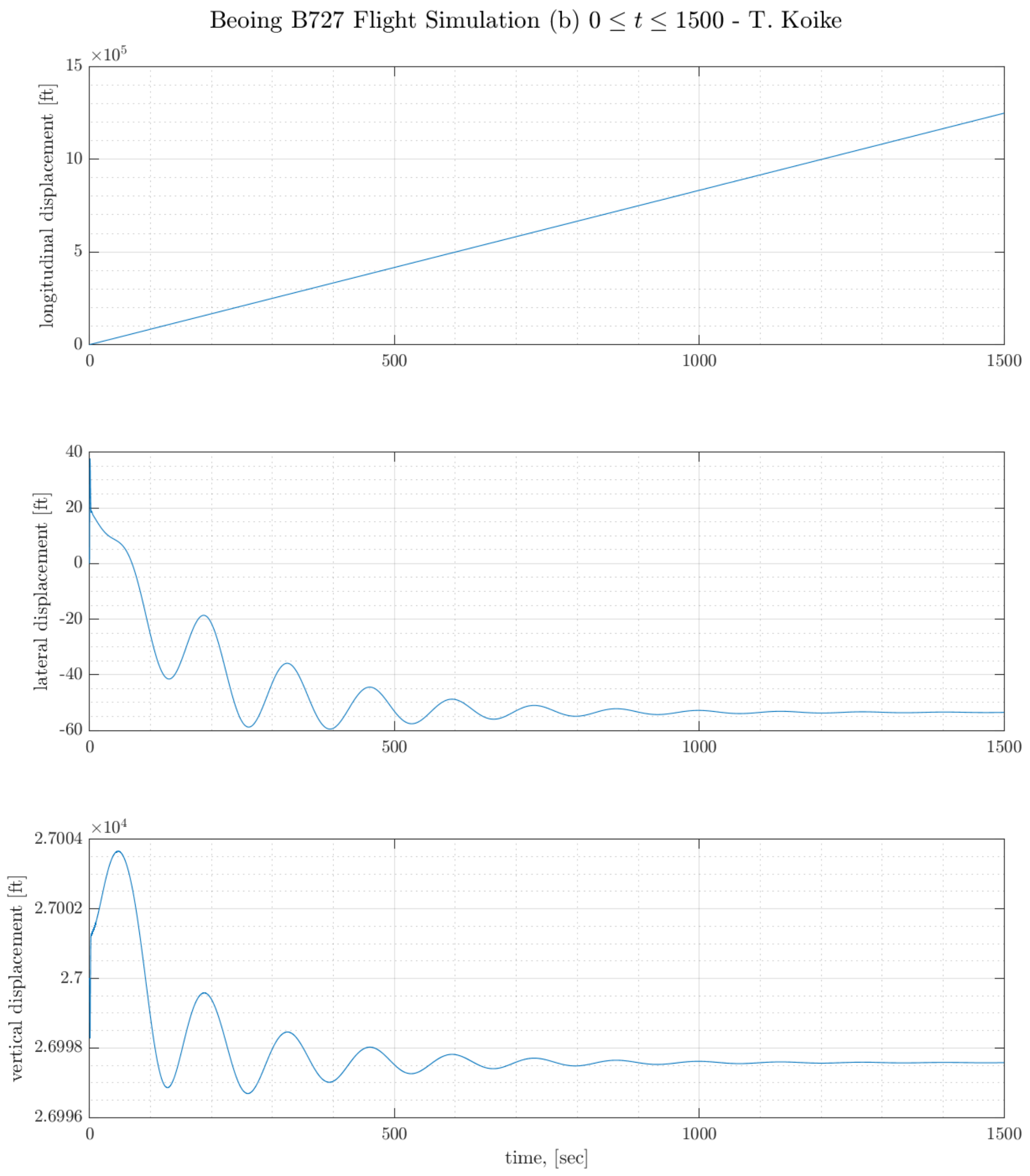
The resulting displacements for (a) are the following

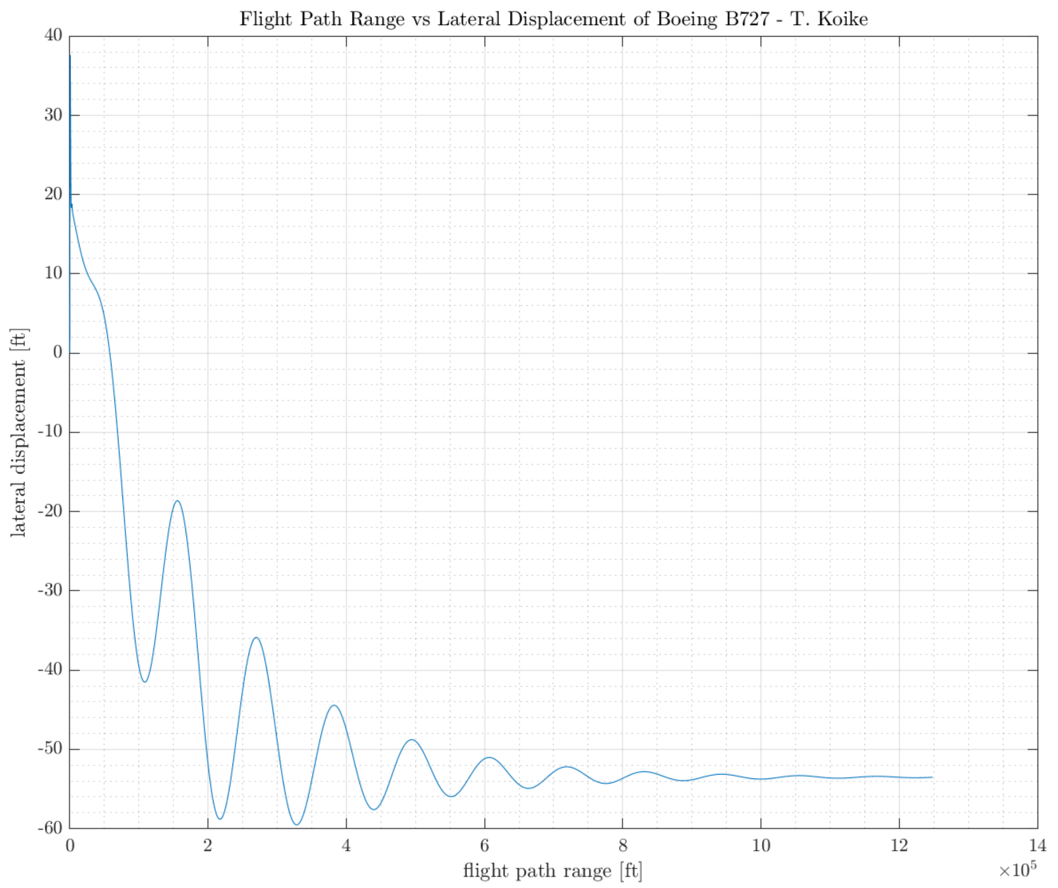



Flight Path Range vs Altitude of Boeing B727 - T. Koike
altitude [ft]
flight path range [ft]
$\times 10^4$
2.704
2.703
2.702
2.701
2.7
2.699
2.698
2.697
$\times 10^5$
0
2
4
6
8
10
12
14

(b).

Repeat the procedures in part (a) but changing the initial conditions. And we obtain the following graph.





The code for this part is the following

```
% Control inputs
dT = 0; dtheta = 0; da = 0; dr = 0;
u0 = 832;  % trim velocity ft/s
alpha_i = 0;  % initial alpha
beta_i = 0.1;  % initial beta
h0 = 27000;  % trim altitude
sim_b = sim("hw3_p2_Boeing727.slx");

t = sim_b.simout.time;
long = sim_b.simout.signals.values;
lat  = sim_b.simout1.signals.values;
vert = sim_b.simout2.signals.values;

% Plotting
fig1 = figure('Renderer',"painters", 'Position', [10 10 900 1000]);
subplot(3,1,1)
plot(t,long)
grid on; grid minor; box on;
ylabel('longitudinal displacement [ft]')
subplot(3,1,2)
```

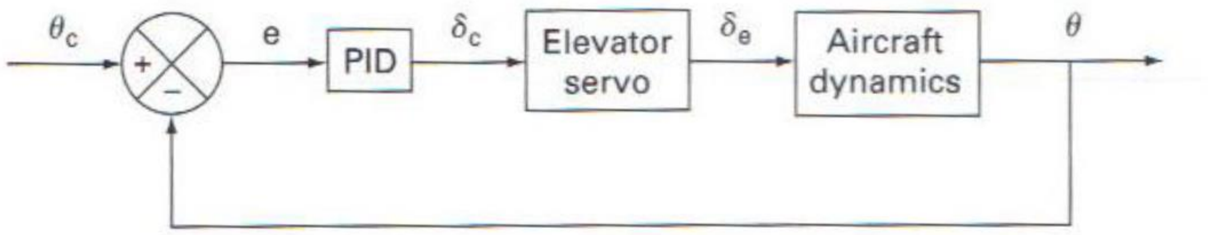
```
plot(t,lat)
grid on; grid minor; box on;
ylabel('lateral displacement [ft]')
subplot(3,1,3)
plot(t,vert)
grid on; grid minor; box on;
ylabel('vertical displacement [ft]')
xlabel('time, [sec]')
sgtitle('Beoing B727 Flight Simulation (b) $0\leq t \leq 1500$ - T. Koike')
saveas(fig1, fullfile(fdir, "p2_b.png"))

% range vs lateral displacement
fig2 = figure('Renderer',"painters", 'Position', [10 10 900 700]);
plot(long, lat)
grid on; grid minor; box on;
title('Flight Path Range vs Lateral Displacement of Boeing B727 - T. Koike')
xlabel('flight path range [ft]')
ylabel('lateral displacement [ft]')
saveas(fig2, fullfile(fdir, "p2_b_rangeVSlat.png"))
```

Problem 3. (30pts)

For an autopilot system shown below with

$$\frac{\Delta\theta}{\Delta\delta_c(s)} = \frac{3}{(s+10)(s^2+2s+5)} \text{ ,}$$

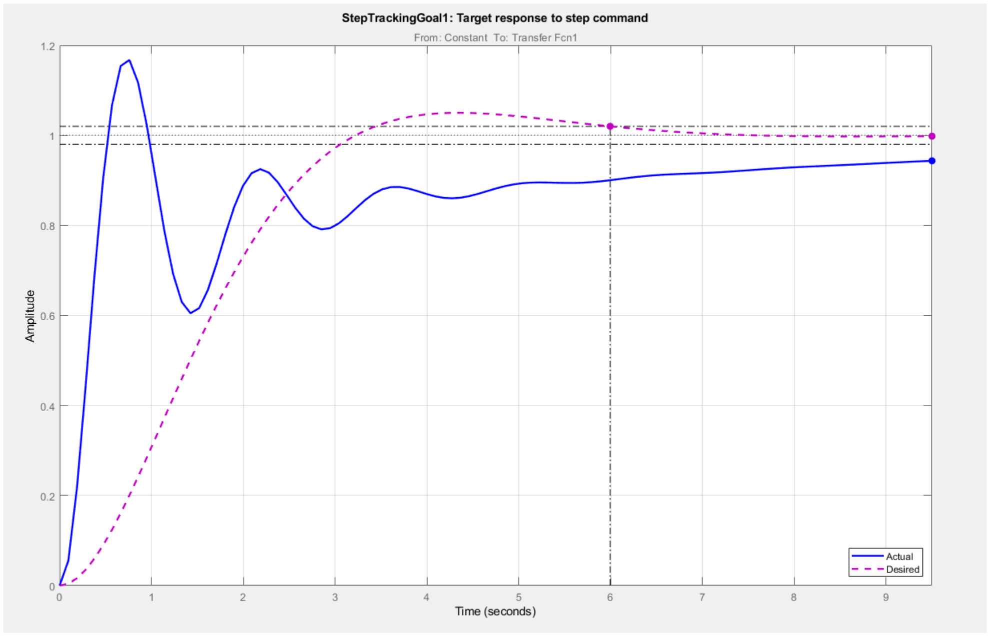
design a PID controller using the Control System Tuner Toolbox such that when tracking the input signal θ_c, the system has a second order approximation with time constant = 1 sec, overshoot less than 5%, and steady state error less than 10%.



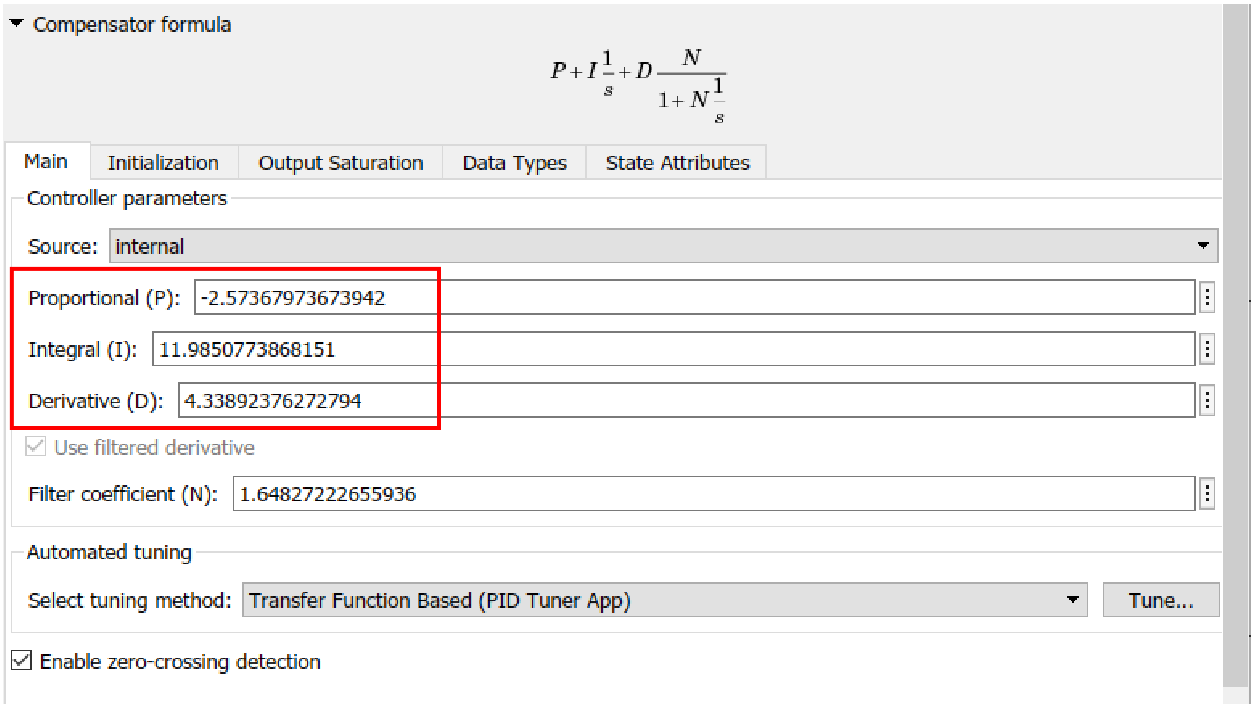
Initially, we set the gains to

K_p	50
K_i	10
K_d	5

Using the Control System Tuner Toolbox with the conditions we get the following plot



Now, tuning the PID controller and updating it in our model we get the following PID gains



$$P + I\frac{1}{s} + D\frac{N}{1 + N\frac{1}{s}}$$

Controller parameters

Source: internal

Proportional (P): -2.57367973673942

Integral (I): 11.9850773868151

Derivative (D): 4.33892376272794

[✓] Use filtered derivative

Filter coefficient (N): 1.64827222655936

Automated tuning

Select tuning method: Transfer Function Based (PID Tuner App)

[✓] Enable zero-crossing detection

K_p	-2.5737
K_i	11.9851
K_d	4.3389

This is the PID controller we want to design.