University of Colorado Boulder

ASEN 3801: Aerospace Vehicle Dynamics and Controls Lab

December 14, 2023

ASEN 3801 LAB 5: Fixed Wing Aircraft Analysis

Team 33

Paige Catena[1]

Anna Krueger[2]

Rishab Pally[3]

Kieran Yochim[4]

Professor: Eric Frew

[1]SID: 110170350
[2]SID: 105867276
[3]SID: 109519936
[4]SID: 109330801

Ann and H.J. Smead
Aerospace Engineering Sciences
UNIVERSITY OF COLORADO **BOULDER**

Problem 1

In problem 1, we were tasked with modifying the "PlotSimulation" function to output control input variables such as elevator, aileron, rudder and throttle. The modified function can be seen in the appendix.

Problem 2

In problem 2, we are implementing the "AircraftEOM" function, which calculates the equations of motion for the fixed-wing aircraft. Next, we implemented the provided "AerodForcesandMoments.m" function to calculate the forces and moments with respect to the aircraft. Additionally, we utilized Matlab's ode45 function to simulate the aircraft equations of motion for the three different sets of initial conditions from parts 1, 2, and 3 of the problem. The Figures seen below are the result of these simulations, and their analysis can be seen in their respective sections.

Section 2.1

For part 1 of problem 2, we set all of the aircraft's initial state and control inputs to zero except for the interial velocity component that is in the body x-direction, which is set equal to 21 m/s, and the height, which is equal to 1609.34 m. The plots for the simulation under these conditions are shown below. From the Angles vs. Time graph, you can see that the pitch angle experienced decaying oscillations with a period of about 10.5 seconds before reaching a steady state value of -0.05 radians. Looking at the angular velocity about the y-axis (q) from the Angular Velocities vs. Time plot, it is clear that q also experienced decaying oscillations with roughly the same period of 10.5 seconds, but eventually came to a steady state value of zero. Since we are not using any control input for this problem, it makes sense that we see a steady value of zero for all of the Controls vs. Time plots. The x position saw a steady increase, the y position remained at zero, and the z position saw a somewhat steady increase of 180m over the 100 second period of time. For these initial conditions, we expected to see the aircraft near, but not at the trim condition because without any controls the aircraft cannot naturally return to its trim condition. Specifically, with a lack of throttle input, there is no force to counteract drag. This will cause the aircraft to slowly descend from its starting height which is shown in the 3D plot below.

Figure: Angles vs. Time

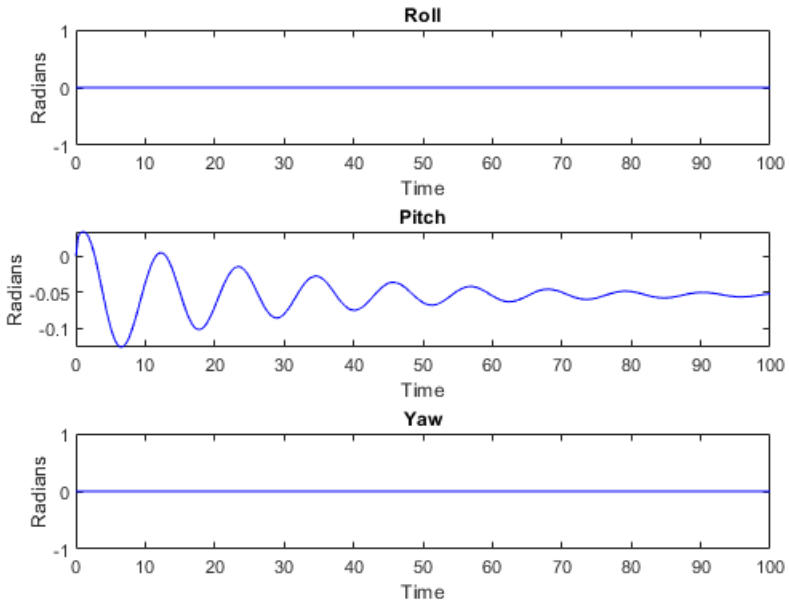


Figure: Angular Velocities vs. Time

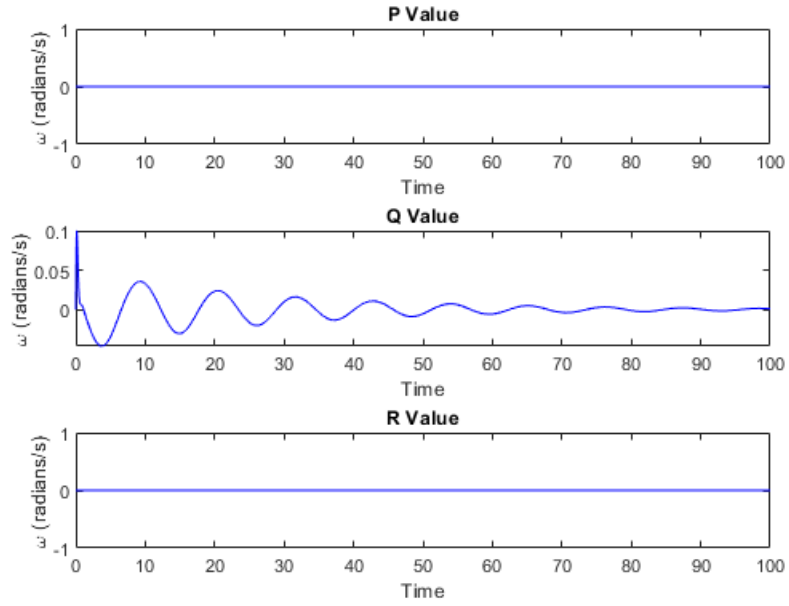


Figure: Controls vs. Time

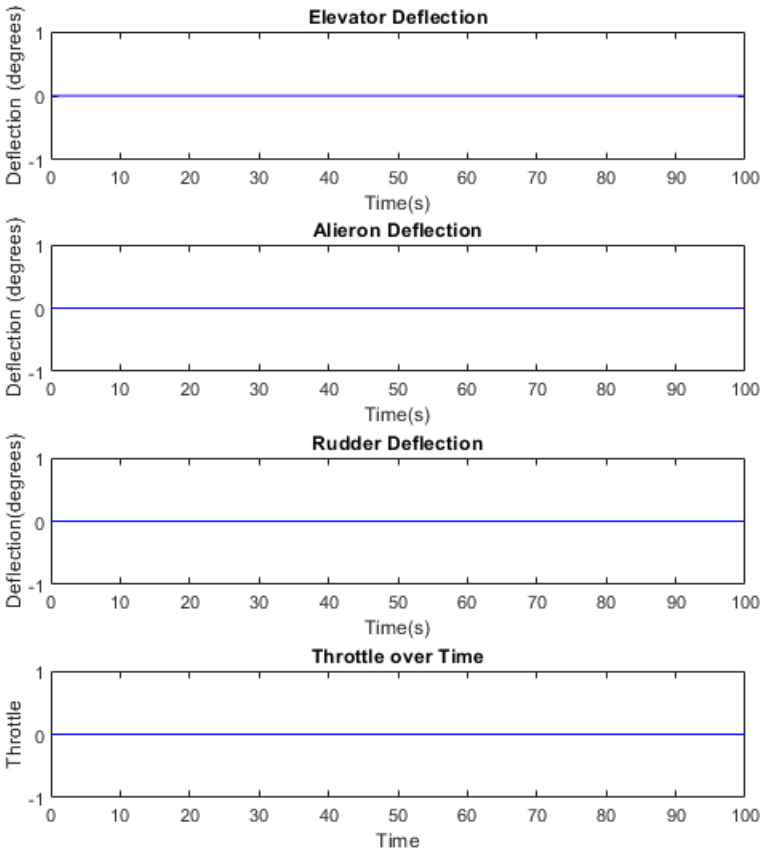


Figure: Position vs Time

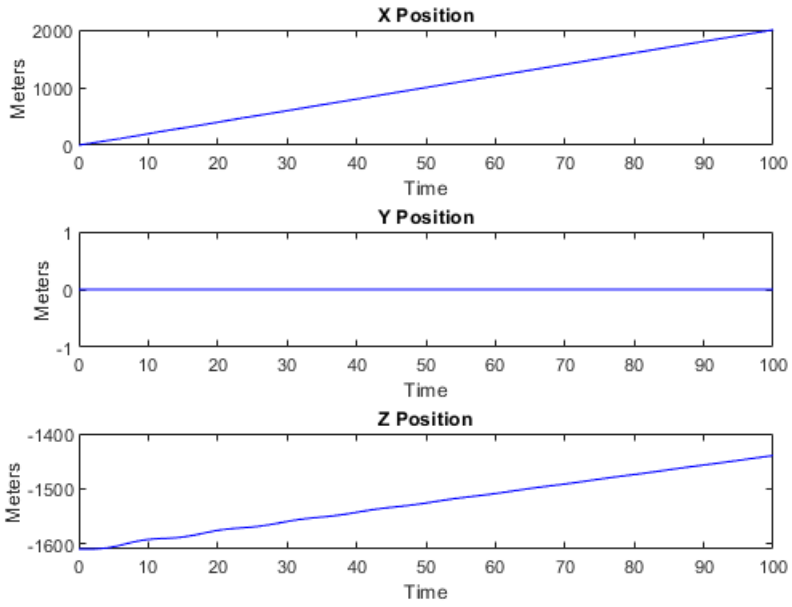


Figure: 3D Aircraft Position vs. Time

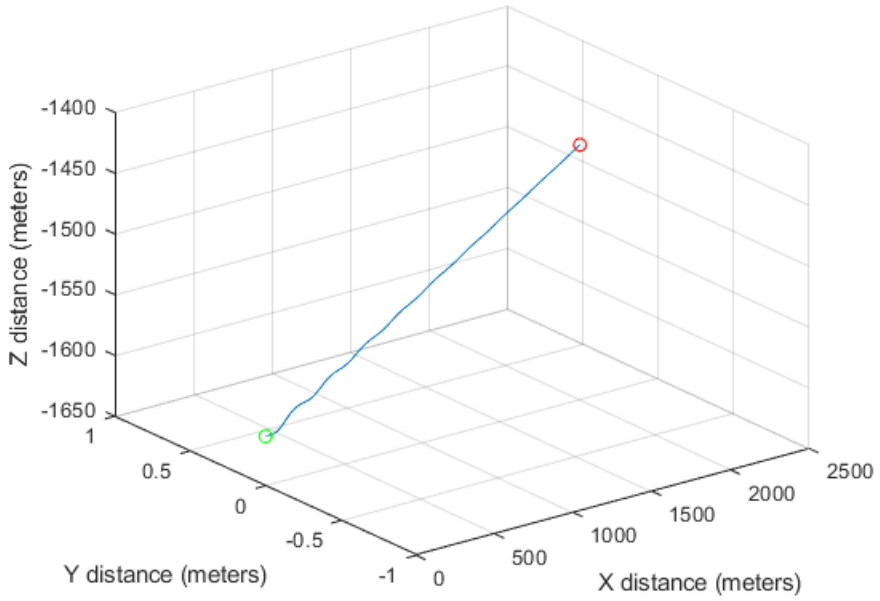
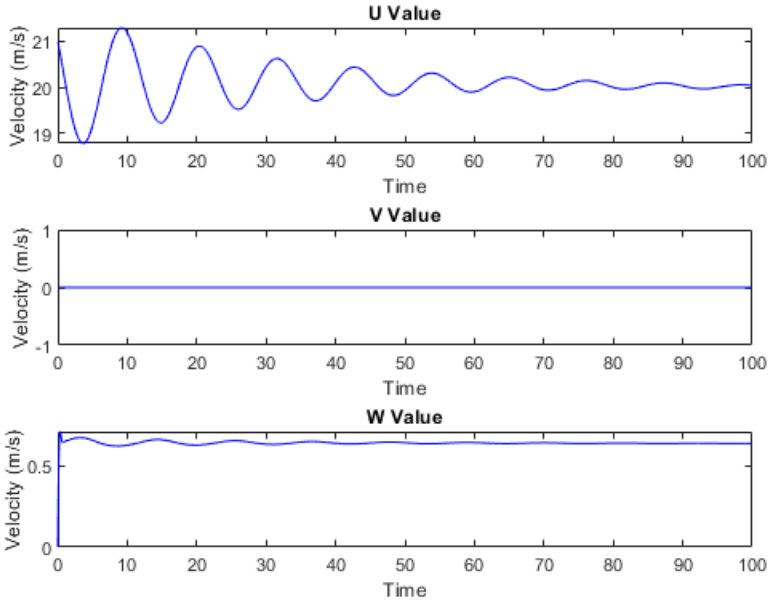


Figure: Velocities vs. Time



Section 2.2

For part 2 of problem 2, we are given a new set of initial state and control inputs. From the plots, you can see that while roll and yaw experience no perturbations and remain at a value of zero, pitch undergoes oscillations with a period around 10.5 seconds. The oscillations in pitch decay with a settling time around 40-50 seconds, and ultimately pitch returns to a steady state value just slightly below 0.2878 radians, which was the trim condition initially given. The only angular velocity that experienced oscillations for this simulation was q, which experienced decaying oscillations with a period of 10.5 seconds and a settling time approximated at 40-50 seconds. After q settled, it remained at a steady state value of zero, which is what we expect from the initial trim condition of $q = 0$. The control inputs remained at a constant value equal to the control inputs given for this problem. For example, the elevator control input was 0.1079 radians, and we saw a constant value of 6.182 degrees (which is equivalent to 0.1079 radians and exactly what we expect). At first glance, the z position plot looks to be increasing similarly to the x position, however after looking closely at the axis, you can see that the z position only increases by 0.02 m, which is actually a very small amount compared to the 2000 m the x position increased in the same 100 second time span. The same can be said for the 3D plot of the Aircraft's position; while at first it looks like there is a lot of movement in the z direction, that is just the scale of the z-axis, and the aircraft's z position is actually nearly constant. Finally, when analyzing the velocities vs. time, we saw decaying oscillations in u and w while v remained at a constant value of zero. The u velocity eventually settled to a steady state value of 20.9915 m/s and the w velocity settled to a value of 0.583725 m/s. This means the error in the velocities (with respect to the trim condition) is less than 0.05 percent, so we are happy with these results.

Figure: Angles vs. Time

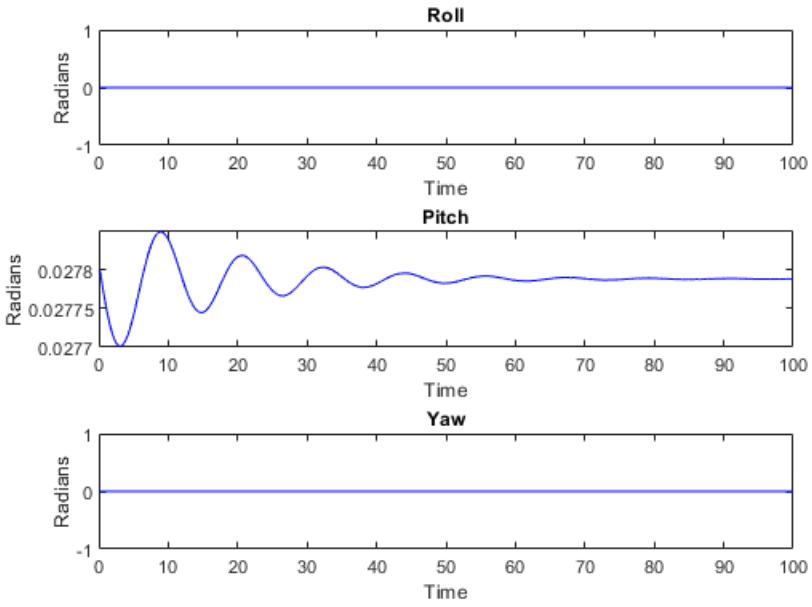


Figure: Angular Velocities vs. Time

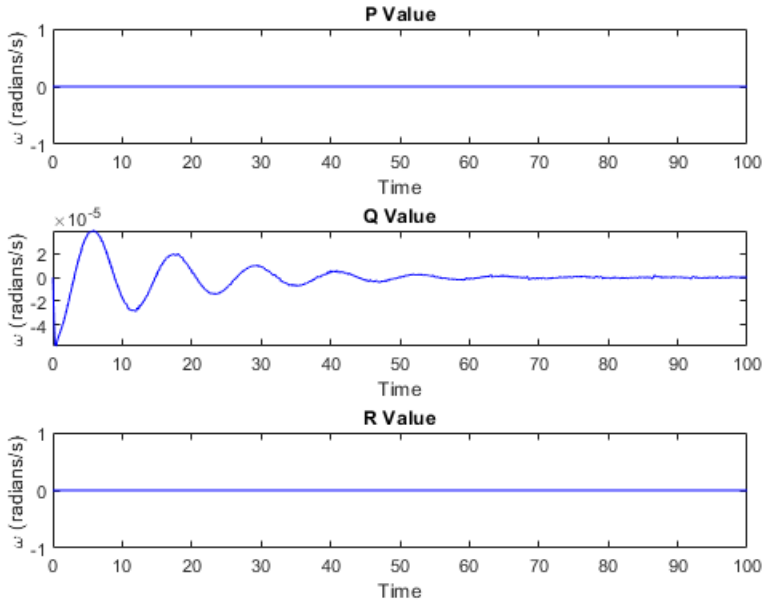


Figure: Controls vs. Time

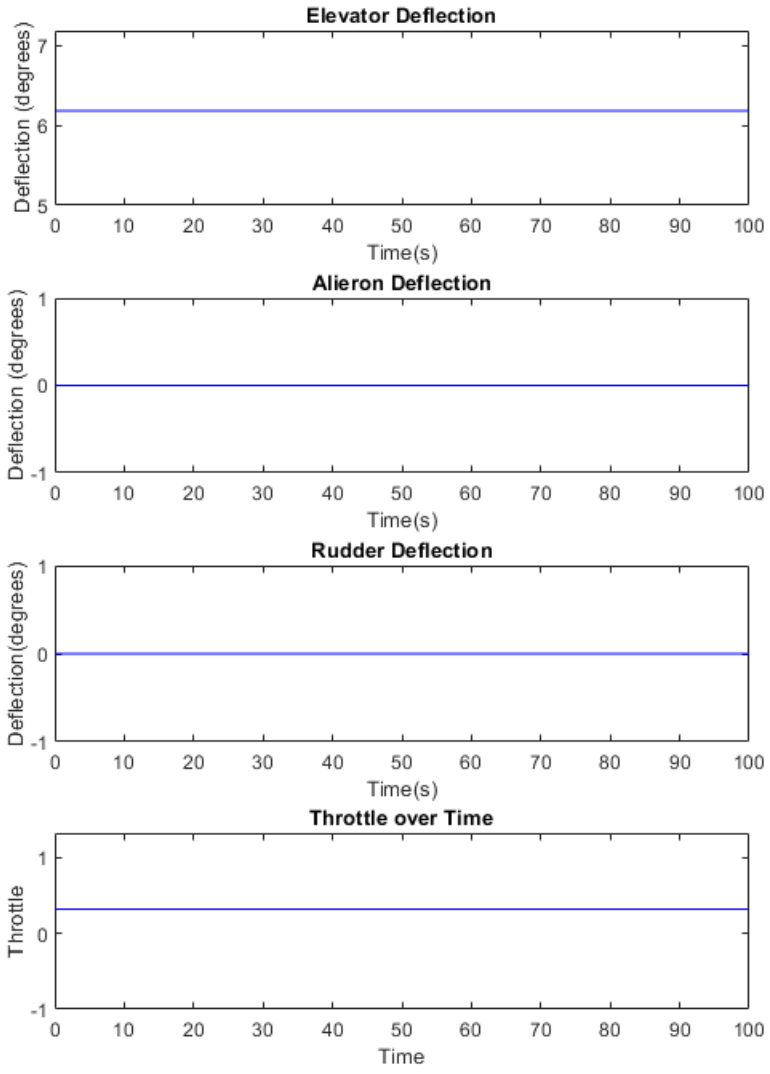


Figure: Position vs. Time

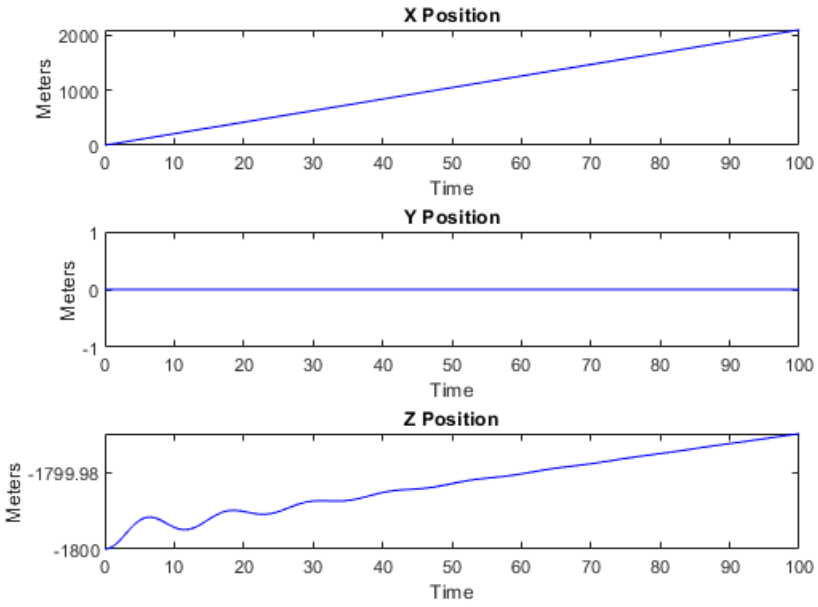


Figure: 3D Aircraft Position vs. Time

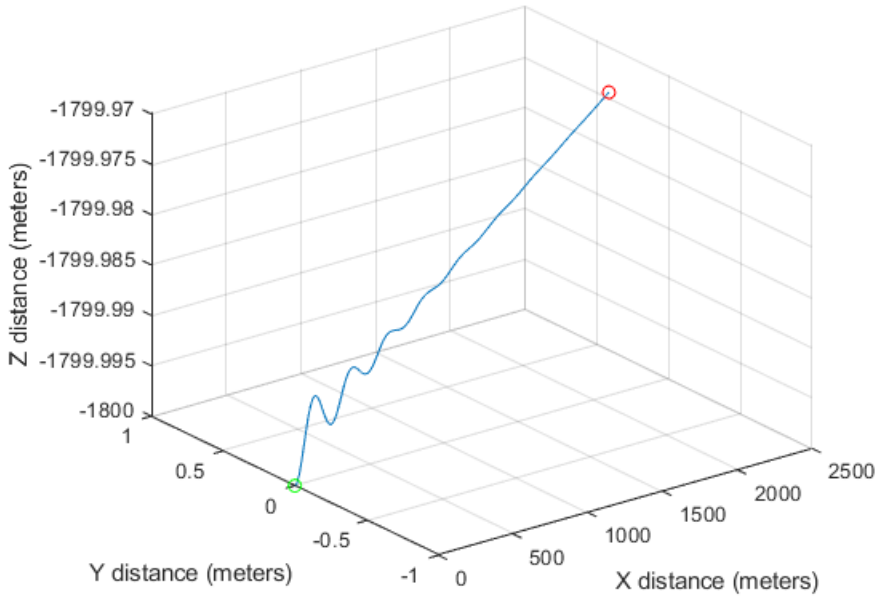
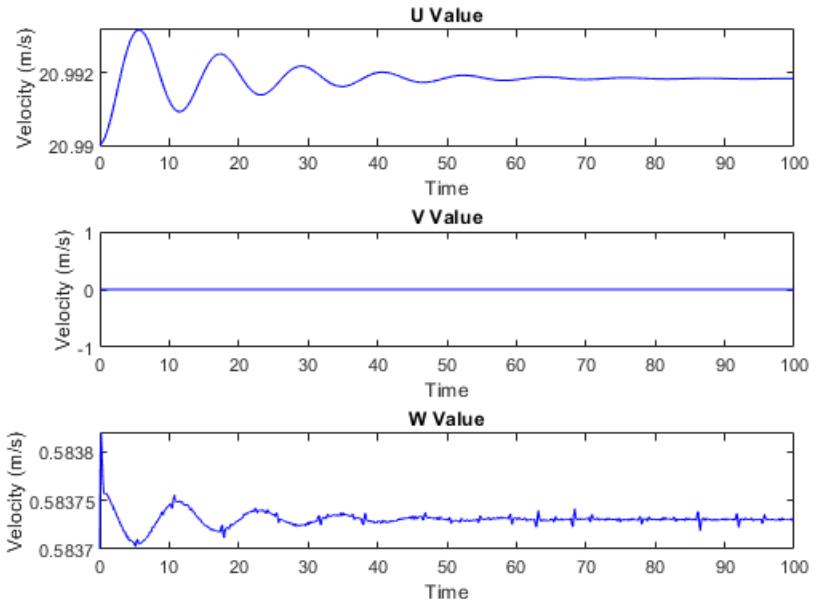


Figure: Velocites vs. Time



Section 2.3

The aircraft starts at a significant altitude of -1800 meters. The initial attitude shows a roll angle of 15°, a pitch of -12° and a yaw of 270°. The u, v, w velocities of the aircraft are 19, 3 and -2 respectively along with a roll rate, pitch rate and yaw rate of 0.08, -0.2 and 0 °/s respectively. The ailerons are deflected to 5°, the elevator is deflected to 2°, the rudder is deflected to -13° and the throttle is set to 0.3. Similar to part 2, we expect a settling time but in this case it is approximately 80 seconds. These initial conditions initially put the aircraft into a banked turn. Upon exiting the banked turn, the spiral mode of the aircraft is excited after about 80 seconds of simulation time. When observing the spiral behavior of the aircraft, the radius stays relatively constant as it spirals to the ground indicating an unstable spiral mode. The control inputs would be expected to remain constant, which they do. The aileron and rudder deflections result in a turn, while the elevator and throttle settings are managing the aircraft's pitch and speed. When looking in closer detail to the results of our analysis, theta, q, and u all experience decaying oscillations with a settling time of about 80 seconds and a period of about 13 seconds. P, r, and v all experience some short period oscillations at the start of the simulation. This simulation shows us that action must be taken with feedback control in order to prevent unstable spirals from interfering with the aircraft's performance.

Figure: Angles vs. Time

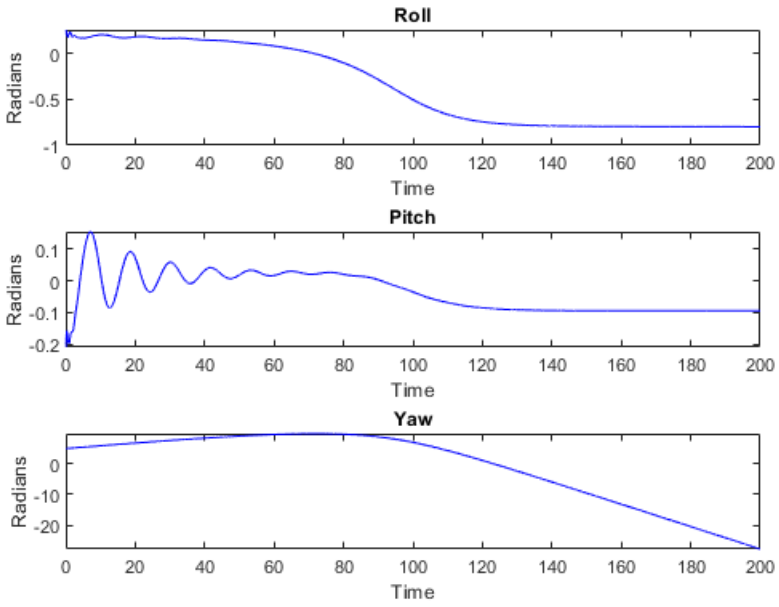


Figure: Angular Velocities vs. Time

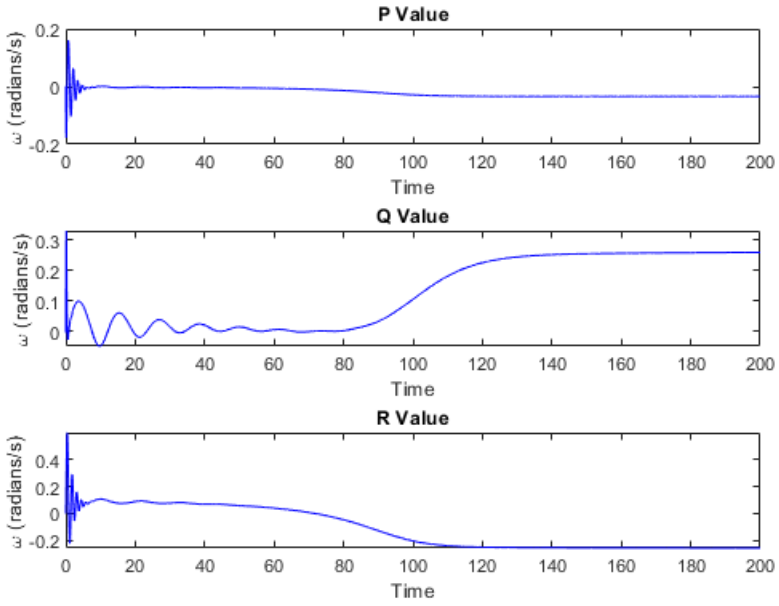


Figure: Controls vs. Time

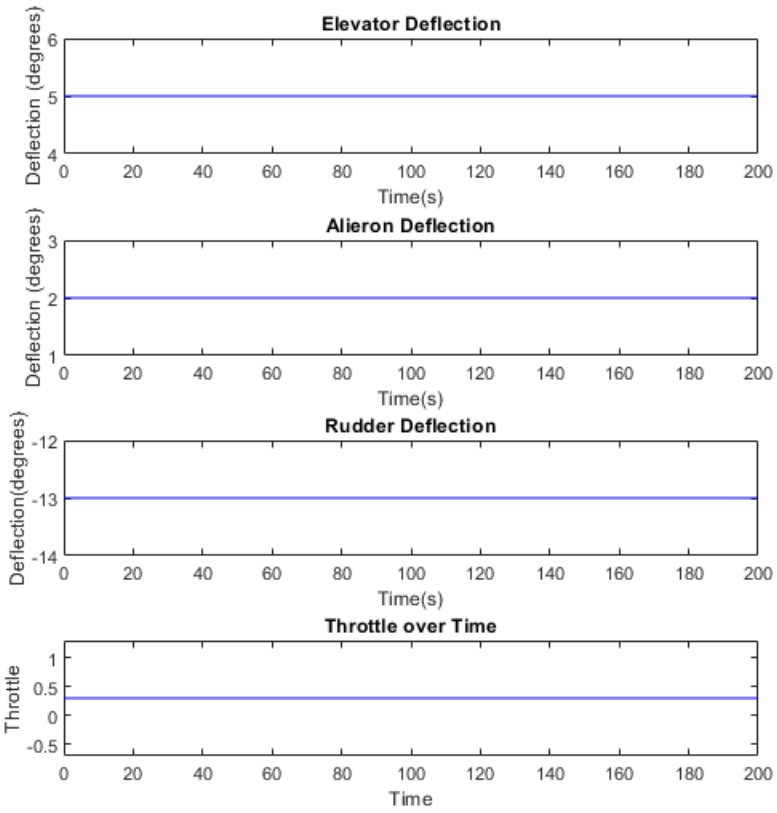


Figure: Position vs. Time

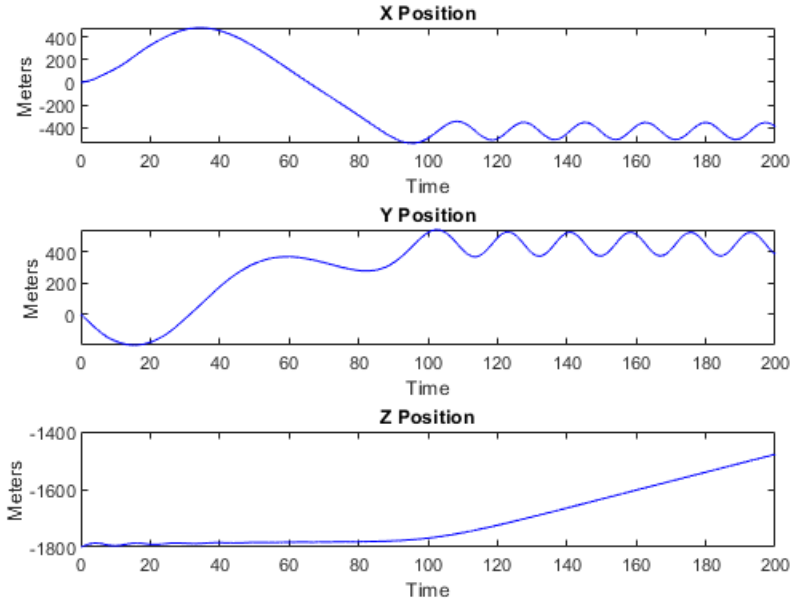


Figure: 3D Aircraft Position vs. Time

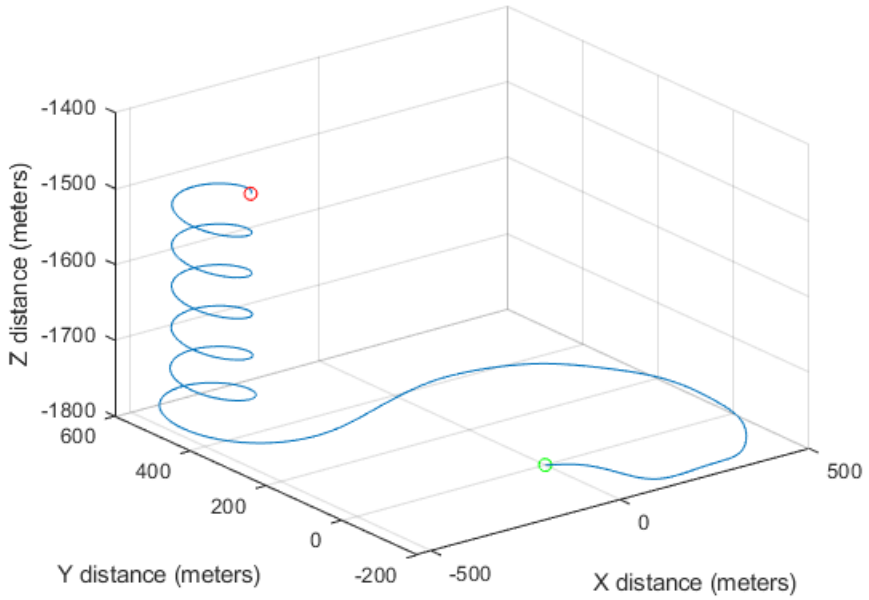
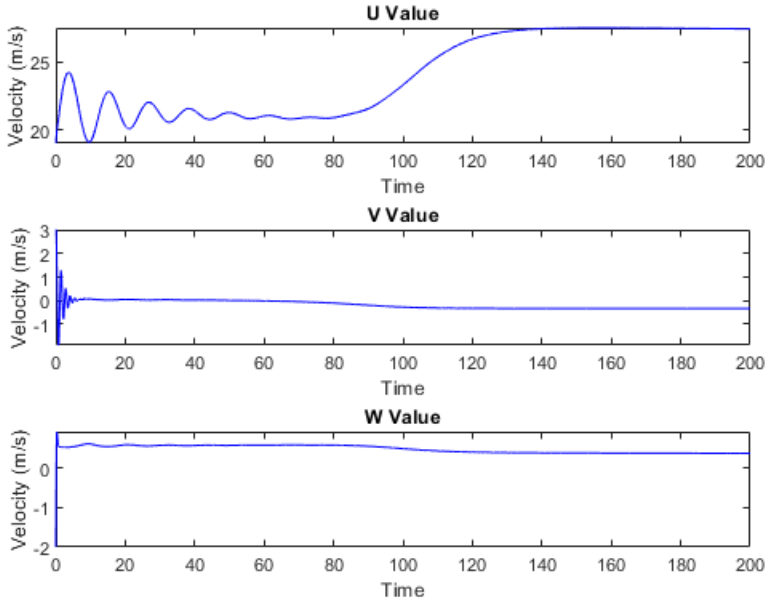


Figure: Velocites vs. Time



Problem 3

In problem 3, we created a function called "AircraftEOMDoublet", which incorporates a doublet in the elevator input. We defined the elevator input with respect to the positive and negative pulse. Using ode45,

we were able to simulate the aircraft equations of motion for a short period duration (3 seconds) and a long period duration (100 seconds). Specifically, we are simulating a 15 degree doublet with a duration doublet time of 0.25 seconds for a total of 3 seconds. Then, we plotted the results and estimated the natural frequency and damping ratio of the short period mode.

Section 3.1

Figure: Angles vs. Time

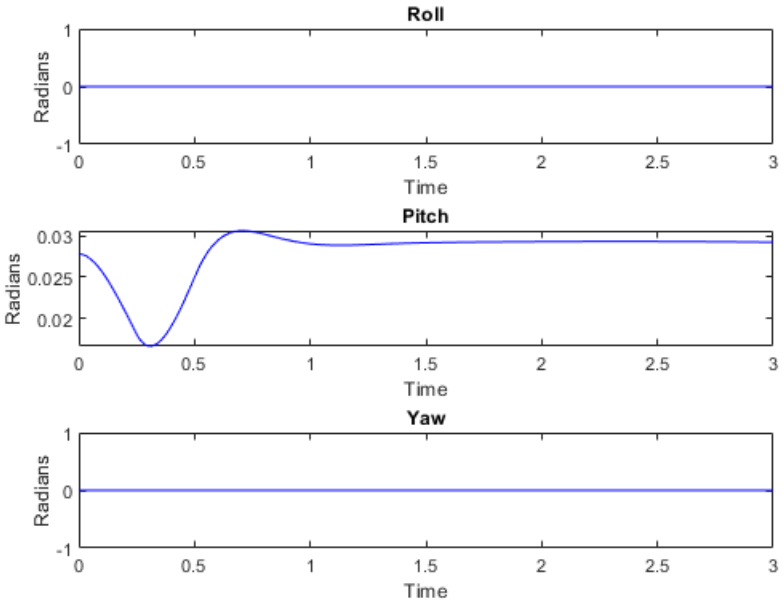


Figure: Angular Velocities vs. Time

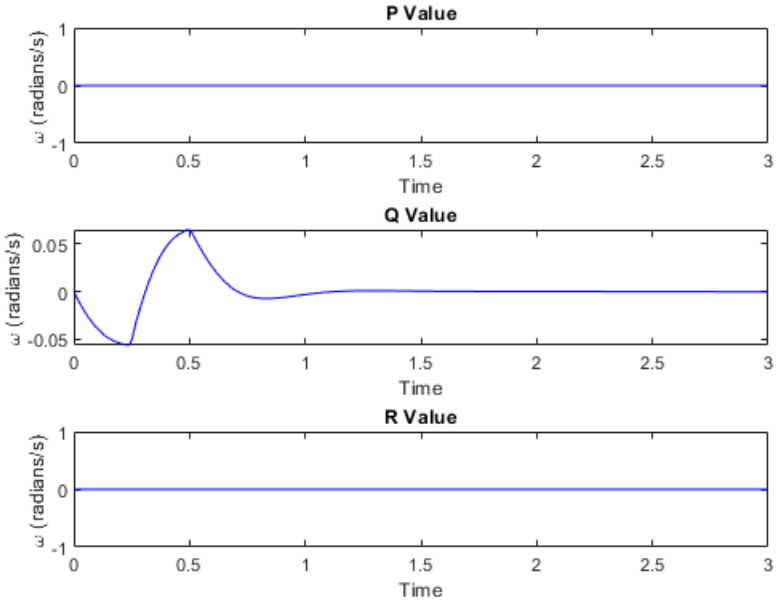


Figure: Controls vs. Time

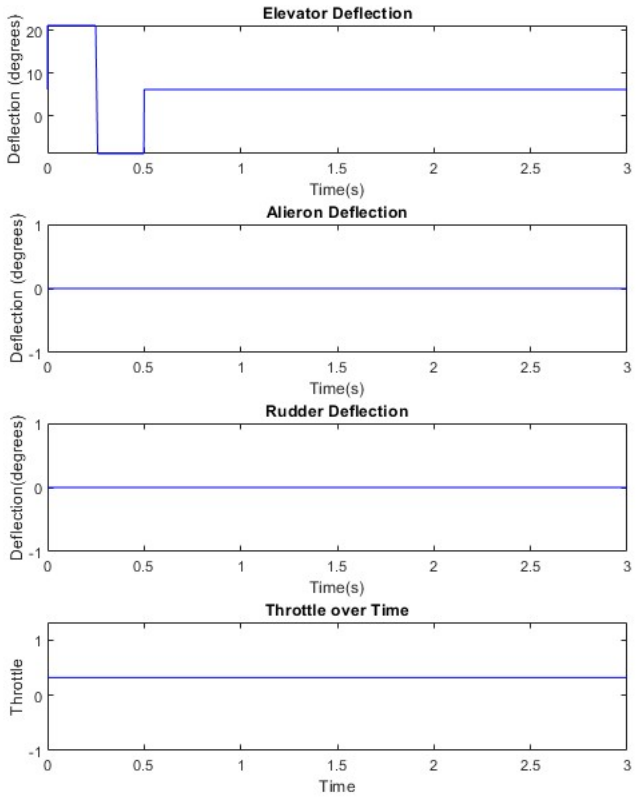


Figure: Positions vs. Time

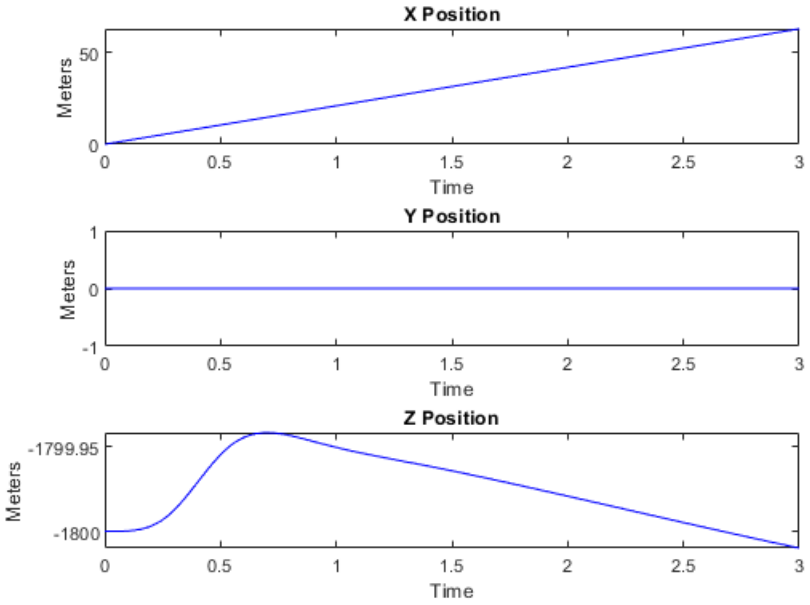


Figure: 3D Aircraft Position vs. Time

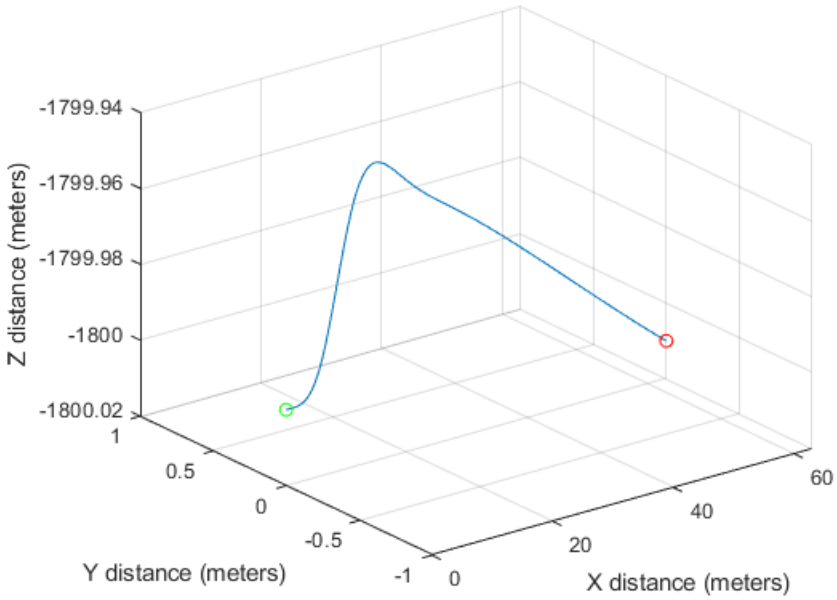
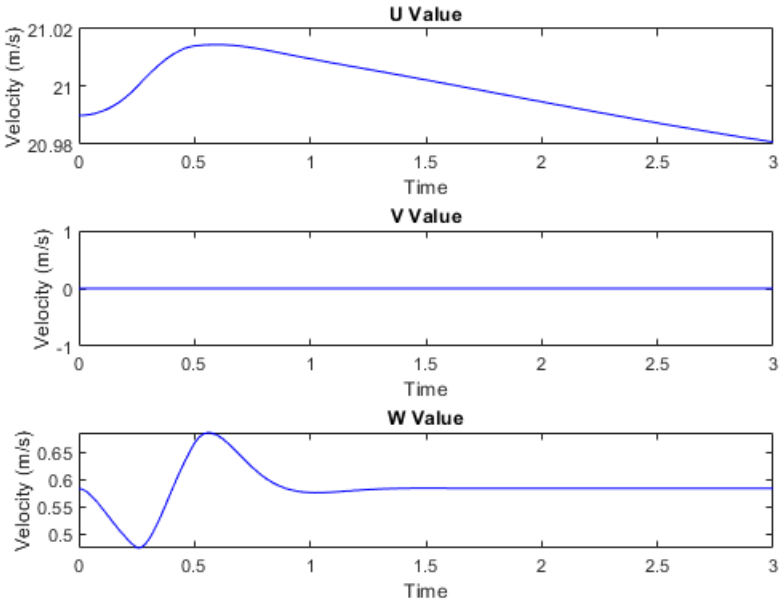


Figure: Velocities vs. Time



In order to estimate the damping ratio and natural frequency we used the pitch component of our results. Using the equations below

$$\zeta = \frac{ln\left(\frac{y_1}{y_2}\right)}{\sqrt{4\pi^2 + \left(ln\left(\frac{y1}{y2}\right)\right)^2}}$$

$$\omega = \frac{ln\left(\frac{y_1}{y_2}\right)}{\zeta\left(t_2 - t_1\right)}$$

We end up estimating $\zeta = 0.69$ and $\omega = 1.98$Hz.

Section 3.2

For this section we used the same initial state vector as section 2.2. We ran the simulation for 100 seconds to record the slow oscillations and identify the respective phugoid modal response, as well as its natural frequency and damping ratio.

Figure: Angles vs. Time

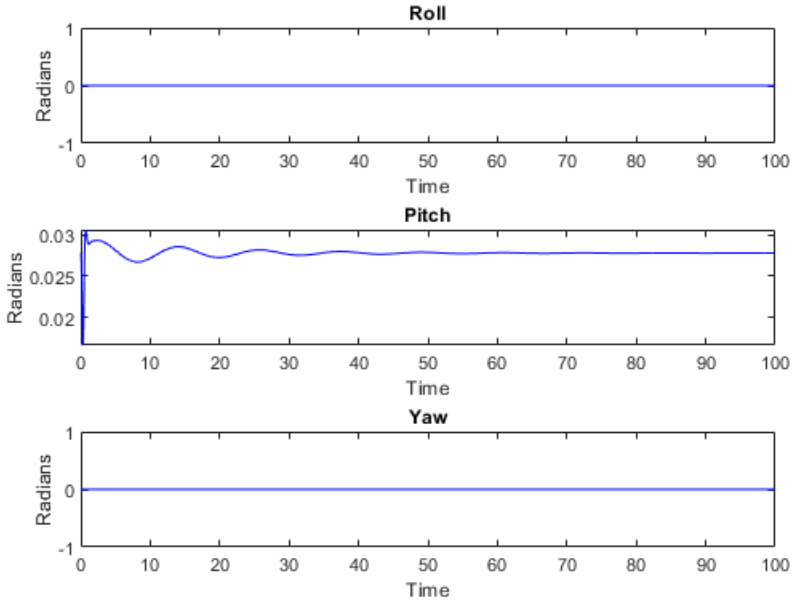


Figure: Angular Velocities vs. Time

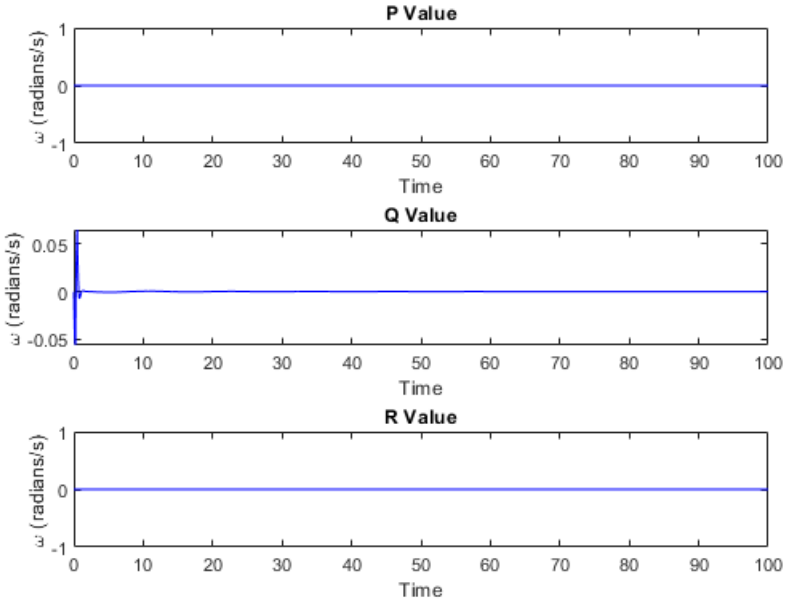


Figure: Controls vs. Time

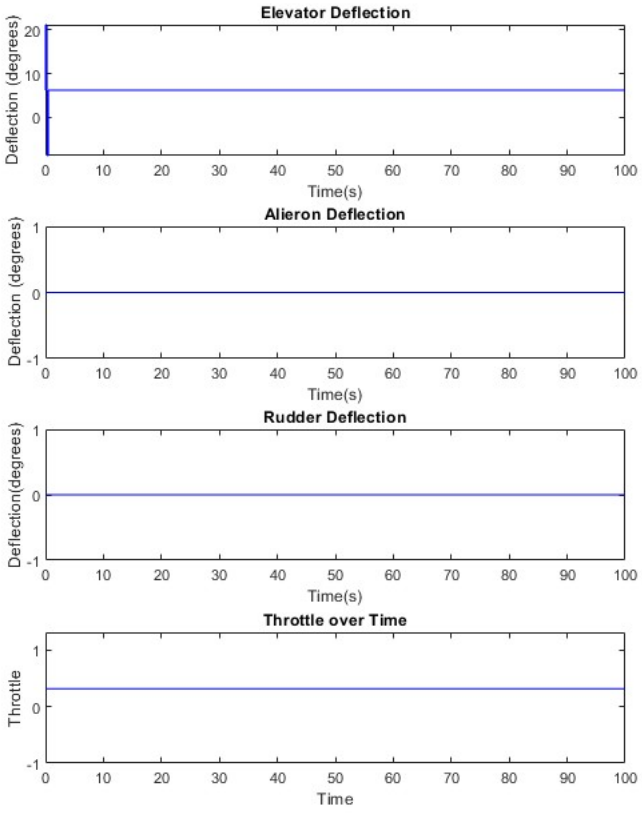


Figure: Position vs. Time

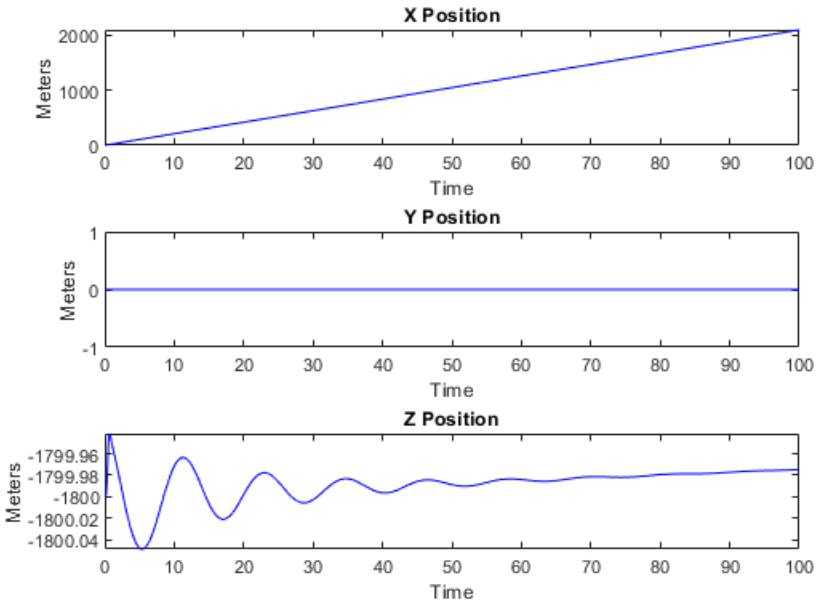


Figure: 3D Aircraft Position vs. Time

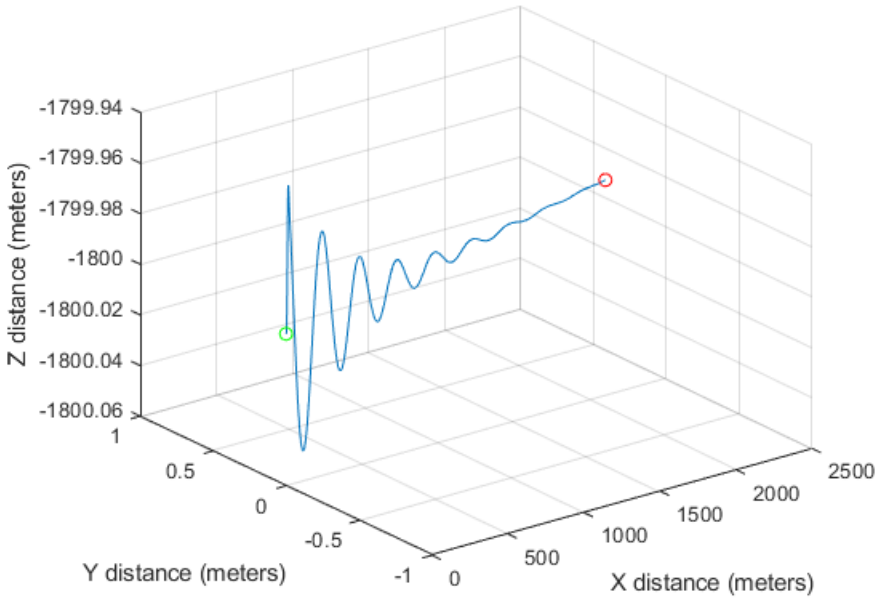
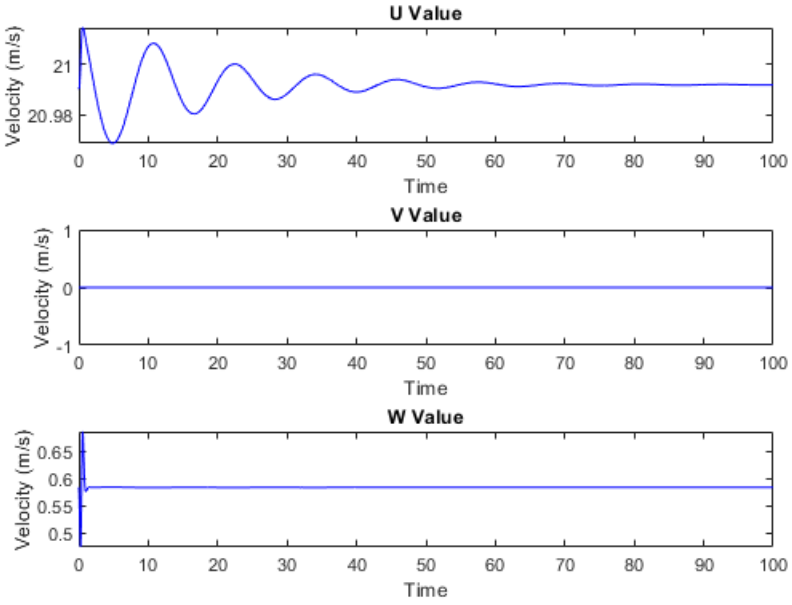


Figure: Velocities vs. Time



In order to estimate the damping ratio and natural frequency we used the inertial velocity, u_E, in the x direction component in the figure above. Using the equations defined in 3.1 we end up estimating $\zeta = .32$ and $\omega = 0.21$Hz.

Member Contributions

Team Participation	Plan	Model	Experiment	Results	Report	Code	ACK
Paige Catena	2	1	1	2	1	1	x
Anna Krueger	1	2	1	1	2	1	x
Rishab Pally	1	2	1	1	2	1	x
Kieran Yochim	1	2	1	1	1	2	x

Matlab Components

Code is attached below.

```
close all
clc
clear
ttwistor;
% Initial state: 12 aircraft parameters and 4 control inputs
initstate = [0;0;-1800;0;0.02780;0;20.99;0;0.5837;0;0;0];
tspan = [0 100];
wind_inertial = [0,0,0]';
de = .1079;  % Elevator deflection
da = 0;  % Aileron deflection
dr = 0;  % Rudder deflection
dt = 0.3182;  % Throttle setting
aircraft_surfaces = [de da dr dt];
doublet_time = 0.25;
doublet_size = 15*(pi/180);

[time,state] = ode45(@(time,state) AircraftEOMDoublet(time, state,
aircraft_surfaces, doublet_size, doublet_time, wind_inertial,
aircraft_parameters),tspan,initstate);
aircraft_state = state;
control_input_array = aircraft_surfaces;
PlotAircraftSim(time,aircraft_state,control_input_array,1:6,'b-')


function PlotAircraftSim(time, aircraft_state_array, control_input_array,
fig, col)
deltae_trim = control_input_array(1);
deltae = zeros(length(time),1);
t=time;
for i=1:length(time)
    if 0<t(i) && t(i)<= 0.25
        deltae(i) = deltae_trim + 15*(pi/180);
    elseif 0.25<t(i) && t(i)<=2*0.25
        deltae(i) = deltae_trim - 15*(pi/180);
    else
        deltae(i) = deltae_trim;
    end
end

figure(fig(1));
subplot(311);
plot(time, aircraft_state_array(:,1), col); hold on;
title('X Position')
xlabel('Time')
ylabel('Meters')
subplot(312);
plot(time, aircraft_state_array(:,2), col); hold on;
title('Y Position')
xlabel('Time')
ylabel('Meters')
subplot(313);
plot(time, aircraft_state_array(:,3), col); hold on;
```

```
title('Z Position')
xlabel('Time')
ylabel('Meters')
figure(fig(2));
subplot(311);
plot(time, aircraft_state_array(:,4), col); hold on;
title('Roll')
xlabel('Time')
ylabel('Radians')
subplot(312);
plot(time, aircraft_state_array(:,5), col); hold on;
title('Pitch')
xlabel('Time')
ylabel('Radians')
subplot(313);
plot(time, aircraft_state_array(:,6), col); hold on;
title('Yaw')
xlabel('Time')
ylabel('Radians')
figure(fig(3));
subplot(311);
plot(time, aircraft_state_array(:,7), col); hold on;
title('U Value')
xlabel('Time')
ylabel('Velocity (m/s)')
subplot(312);
plot(time, aircraft_state_array(:,8), col); hold on;
title('V Value')
xlabel('Time')
ylabel('Velocity (m/s)')
subplot(313);
plot(time, aircraft_state_array(:,9), col); hold on;
title('W Value')
xlabel('Time')
ylabel('Velocity (m/s)')
figure(fig(4));
subplot(311);
plot(time, aircraft_state_array(:,10), col); hold on;
title('P Value')
xlabel('Time')
ylabel('\omega (radians/s)')
subplot(312);
plot(time, aircraft_state_array(:,11), col); hold on;
title('Q Value')
xlabel('Time')
ylabel('\omega (radians/s)')
subplot(313);
plot(time, aircraft_state_array(:,12), col); hold on;
title('R Value')
xlabel('Time')
ylabel('\omega (radians/s)')
figure(fig(5));
subplot(411);
plot(time, rad2deg(deltae), col); hold on;
```

```
title('Elevator Deflection')
xlabel('Time(s)')
ylabel('Deflection (degrees)')
subplot(412);
plot(time, control_input_array(:,2).*(180/pi)*ones(length(time)), col); hold
on;
title('Alieron Deflection')
xlabel('Time(s)')
ylabel('Deflection (degrees)')
subplot(413);
plot(time, control_input_array(:,3).*(180/pi)*ones(length(time)), col); hold
on;
title('Rudder Deflection')
xlabel('Time(s)')
ylabel('Deflection(degrees)')
subplot(414);
plot(time, control_input_array(:,4)*ones(length(time)), col); hold on;
title('Throttle over Time')
xlabel('Time')
ylabel('Throttle')
figure(fig(6))
hold on
grid on
plot3(aircraft_state_array(:,1),aircraft_state_array(:,2),aircraft_state_arra
y(:,3))
plot3(aircraft_state_array(1,1),aircraft_state_array(1,2),aircraft_state_arra
y(1,3),'og')
plot3(aircraft_state_array(end,1),aircraft_state_array(end,2),aircraft_state_
array(end,3),'or')
view(3)
xlabel('X distance (meters)')
ylabel('Y distance (meters)')
zlabel('Z distance (meters)')
end

function xdot = AircraftEOM(time, aircraft_state, aircraft_surfaces,
wind_inertial, aircraft_parameters)
stuff = atmosisa(-aircraft_state(3));
density = stuff(4);
[aero_forces, aero_moments] = AeroForcesAndMoments(aircraft_state,
aircraft_surfaces', wind_inertial, density, aircraft_parameters);
% Unpack the state
x = aircraft_state(1);
y = aircraft_state(2);
z = aircraft_state(3);
pos = [x,y,z]';
phi = aircraft_state(4);
theta = aircraft_state(5);
psi = aircraft_state(6);
ang = [phi,theta,psi]';
u = aircraft_state(7);
v = aircraft_state(8);
w = aircraft_state(9);
vel = [u,v,w]';
```

```
p = aircraft_state(10);
q = aircraft_state(11);
r = aircraft_state(12);
ang_vel = [p,q,r]';
% Calculate position derivatives
M_pos_r1 = [cos(theta)*cos(psi), sin(phi)*sin(theta)*cos(psi) -
cos(phi)*sin(psi), cos(phi)*sin(theta)*cos(psi) + sin(phi)*sin(psi)];
M_pos_r2 = [cos(theta)*sin(psi), sin(phi)*sin(theta)*sin(psi) +
cos(phi)*cos(psi), cos(phi)*sin(theta)*sin(psi) - sin(phi)*cos(psi)];
M_pos_r3 = [-sin(theta), sin(phi)*cos(theta), cos(phi)*cos(theta)];

M_pos = [M_pos_r1; M_pos_r2; M_pos_r3];
pos_dot = M_pos*vel;
% Angle derivatives
angle_dot = [1,sin(phi).*tan(theta),cos(phi)*tan(theta);0,cos(phi),-
sin(phi);0,sin(phi)*sec(theta),cos(phi)*sec(theta)]*ang_vel;
%Velocity derivatives
u_dot = (r*v) - (q*w) + (9.81*-sin(theta))+ (aero_forces(1)./
aircraft_parameters.m);
v_dot = p*w - r*u + 9.81*cos(theta)*sin(phi)+ aero_forces(2)./
aircraft_parameters.m;
w_dot = q*u - p*v + 9.81*cos(theta)*cos(phi)+aero_forces(3)./
aircraft_parameters.m;
vel_dot = [u_dot,v_dot,w_dot]';
% Angular Velocity Derivatives
gamma = aircraft_parameters.Ix*aircraft_parameters.Iz-
aircraft_parameters.Ixz^2;
gamma1 = (aircraft_parameters.Ixz*(aircraft_parameters.Ix-
aircraft_parameters.Iy+aircraft_parameters.Iz))/gamma;
gamma2 = (aircraft_parameters.Iz*(aircraft_parameters.Iz-
aircraft_parameters.Iy)+aircraft_parameters.Ixz^2)/gamma;
gamma3 = aircraft_parameters.Iz/gamma;
gamma4 = aircraft_parameters.Ixz/gamma;
gamma5 = (aircraft_parameters.Iz-aircraft_parameters.Ix)/
aircraft_parameters.Iy;
gamma6 = aircraft_parameters.Ixz/aircraft_parameters.Iy;
gamma7 = (aircraft_parameters.Ix*(aircraft_parameters.Ix-
aircraft_parameters.Iy)+aircraft_parameters.Ixz^2)/gamma;
gamma8 = aircraft_parameters.Ix/gamma;
p_dot = (gamma1*p*q)-(gamma2*q*r)+(gamma3*aero_moments(1))+
(gamma4*aero_moments(3));
q_dot = gamma5*p*r-gamma6*(p^2-r^2)+aero_moments(2)/aircraft_parameters.Iy;
r_dot = gamma7*p*q-gamma1*q*r+gamma4*aero_moments(1)+gamma8*aero_moments(3);
ang_vel_dot = [p_dot,q_dot,r_dot]';

xdot = [pos_dot;angle_dot;vel_dot;ang_vel_dot];
end

function xdot =
AircraftEOMDoublet(time,aircraft_state,aircraft_surfaces,doublet_size,doublet
_time,wind_inertial,aircraft_parameters)

deltae_trim = 0.1079;
deltaa = aircraft_surfaces(2);
```

```
deltar = aircraft_surfaces(3);
deltat = aircraft_surfaces(4);
t = time;
if 0<t && t<= doublet_time
    deltae = deltae_trim + doublet_size;
elseif doublet_time<t && t<=2*doublet_time
    deltae = deltae_trim - doublet_size;
else
    deltae = deltae_trim;
end
aircraft_surfaces = [deltae,deltaa,deltar,deltat];
[T,a,P,density] = atmosisa(-aircraft_state(3));
[aero_forces, aero_moments] = AeroForcesAndMoments(aircraft_state,
aircraft_surfaces', wind_inertial, density, aircraft_parameters);
% Unpack the state
x = aircraft_state(1);
y = aircraft_state(2);
z = aircraft_state(3);
pos = [x,y,z]';
phi = aircraft_state(4);
theta = aircraft_state(5);
psi = aircraft_state(6);
ang = [phi,theta,psi]';
u = aircraft_state(7);
v = aircraft_state(8);
w = aircraft_state(9);
vel = [u,v,w]';
p = aircraft_state(10);
q = aircraft_state(11);
r = aircraft_state(12);
ang_vel = [p,q,r]';
% Calculate position derivatives
M_pos_r1 = [cos(theta)*cos(psi), sin(phi)*sin(theta)*cos(psi) -
cos(phi)*sin(psi), cos(phi)*sin(theta)*cos(psi) + sin(phi)*sin(psi)];
M_pos_r2 = [cos(theta)*sin(psi), sin(phi)*sin(theta)*sin(psi) +
cos(phi)*cos(psi), cos(phi)*sin(theta)*sin(psi) - sin(phi)*cos(psi)];
M_pos_r3 = [-sin(theta), sin(phi)*cos(theta), cos(phi)*cos(theta)];

M_pos = [M_pos_r1; M_pos_r2; M_pos_r3];
pos_dot = M_pos*vel;
% Angle derivatives
angle_dot = [1,sin(phi).*tan(theta),cos(phi)*tan(theta);0,cos(phi),-
sin(phi);0,sin(phi)*sec(theta),cos(phi)*sec(theta)]*ang_vel;
%Velocity derivatives
u_dot = r*v - q*w - 9.81*sin(theta)+ aero_forces(1)./aircraft_parameters.m;
v_dot = p*w - r*u + 9.81*cos(theta)*sin(phi)+ aero_forces(2)./
aircraft_parameters.m;
w_dot = q*u - p*v + 9.81*cos(theta)*cos(phi)+aero_forces(3)./
aircraft_parameters.m;
vel_dot = [u_dot,v_dot,w_dot]';
% Angular Velocity Derivatives
gamma = aircraft_parameters.Ix*aircraft_parameters.Iz-
aircraft_parameters.Ixz^2;
gamma1 = (aircraft_parameters.Ixz*(aircraft_parameters.Ix-
```

```
aircraft_parameters.Iy+aircraft_parameters.Iz))/gamma;
gamma2 = (aircraft_parameters.Iz*(aircraft_parameters.Iz-
aircraft_parameters.Iy)+aircraft_parameters.Ixz^2)/gamma;
gamma3 = aircraft_parameters.Iz/gamma;
gamma4 = aircraft_parameters.Ixz/gamma;
gamma5 = (aircraft_parameters.Iz-aircraft_parameters.Ix)/
aircraft_parameters.Iy;
gamma6 = aircraft_parameters.Ixz/aircraft_parameters.Iy;
gamma7 = (aircraft_parameters.Ix*(aircraft_parameters.Ix-
aircraft_parameters.Iy)+aircraft_parameters.Ixz^2)/gamma;
gamma8 = aircraft_parameters.Ix/gamma;
p_dot = gamma1*p*q-gamma2*q*r+gamma3*aero_moments(1)+gamma4*aero_moments(3);
q_dot = gamma5*p*r-gamma6*(p^2-r^2)+aero_moments(2)/aircraft_parameters.Iy;
r_dot = gamma7*p*q-gamma1*q*r+gamma4*aero_moments(1)+gamma8*aero_moments(3);
ang_vel_dot = [p_dot,q_dot,r_dot]';

%aircraft_surfaces_dot = [0;0;0;0];
% Concatenate
xdot = [pos_dot;angle_dot;vel_dot;ang_vel_dot];

end
function [aero_forces, aero_moments] = AeroForcesAndMoments(aircraft_state,
aircraft_surfaces, wind_inertial, density, aircraft_parameters)
%
%
% aircraft_state = [xi, yi, zi, roll, pitch, yaw, u, v, w, p, q, r]
%   NOTE: The function assumes the veolcity is the air relative velocity
%   vector. When used with simulink the wrapper function makes the
%   conversion.
%
% aircraft_surfaces = [de da dr dt];
%
% Lift and Drag are calculated in Wind Frame then rotated to body frame
% Thrust is given in Body Frame
% Sideforce calculated in Body Frame
```

redefine states and inputs for ease of use

```
ap = aircraft_parameters;

wind_body = TransformFromInertialToBody(wind_inertial,
aircraft_state(4:6,1));
air_rel_vel_body = aircraft_state(7:9,1) - wind_body;

[wind_angles] = WindAnglesFromVelocityBody(air_rel_vel_body);
V = wind_angles(1,1);
beta = wind_angles(2,1);
alpha = wind_angles(3,1);

p = aircraft_state(10,1);
q = aircraft_state(11,1);
r = aircraft_state(12,1);
```

```
de = aircraft_surfaces(1,1);
da = aircraft_surfaces(2,1);
dr = aircraft_surfaces(3,1);
dt = aircraft_surfaces(4,1);

alpha_dot = 0;

%Q = ap.qbar;
Q = 0.5*density*V*V;

sa = sin(alpha);
ca = cos(alpha);
```

determine aero force coefficients

```
CL = ap.CL0 + ap.CLalpha*alpha + ap.CLq*q*ap.c/(2*V) + ap.CLde*de;
%CD = ap.CD0 + ap.CDalpha*alpha + ap.CDq*q*ap.c/(2*V) + ap.CDde*de;
CD = ap.CDpa + ap.K*CL*CL;

CX = -CD*ca + CL*sa;
CZ = -CD*sa - CL*ca;

CY = ap.CY0 + ap.CYbeta*beta + ap.CYp*p*ap.b/(2*V) + ap.CYr*r*ap.b/(2*V) +
ap.CYda*da + ap.CYdr*dr;

%%Thrust = .5*density*ap.Sprop*ap.Cprop*((ap.kmotor*dt)^2 - V^2);
Thrust = density*ap.Sprop*ap.Cprop*(V + dt*(ap.kmotor - V))*dt*(ap.kmotor-
V); %%Changed 5/2/15;New model as described in http://
uavbook.byu.edu/lib/exe/fetch.php?media=shared:propeller_model.pdf
```

determine aero forces from coeffficients

```
X = Q*ap.S*CX + Thrust;
Y = Q*ap.S*CY;
Z = Q*ap.S*CZ;

aero_forces = [X;Y;Z];
```

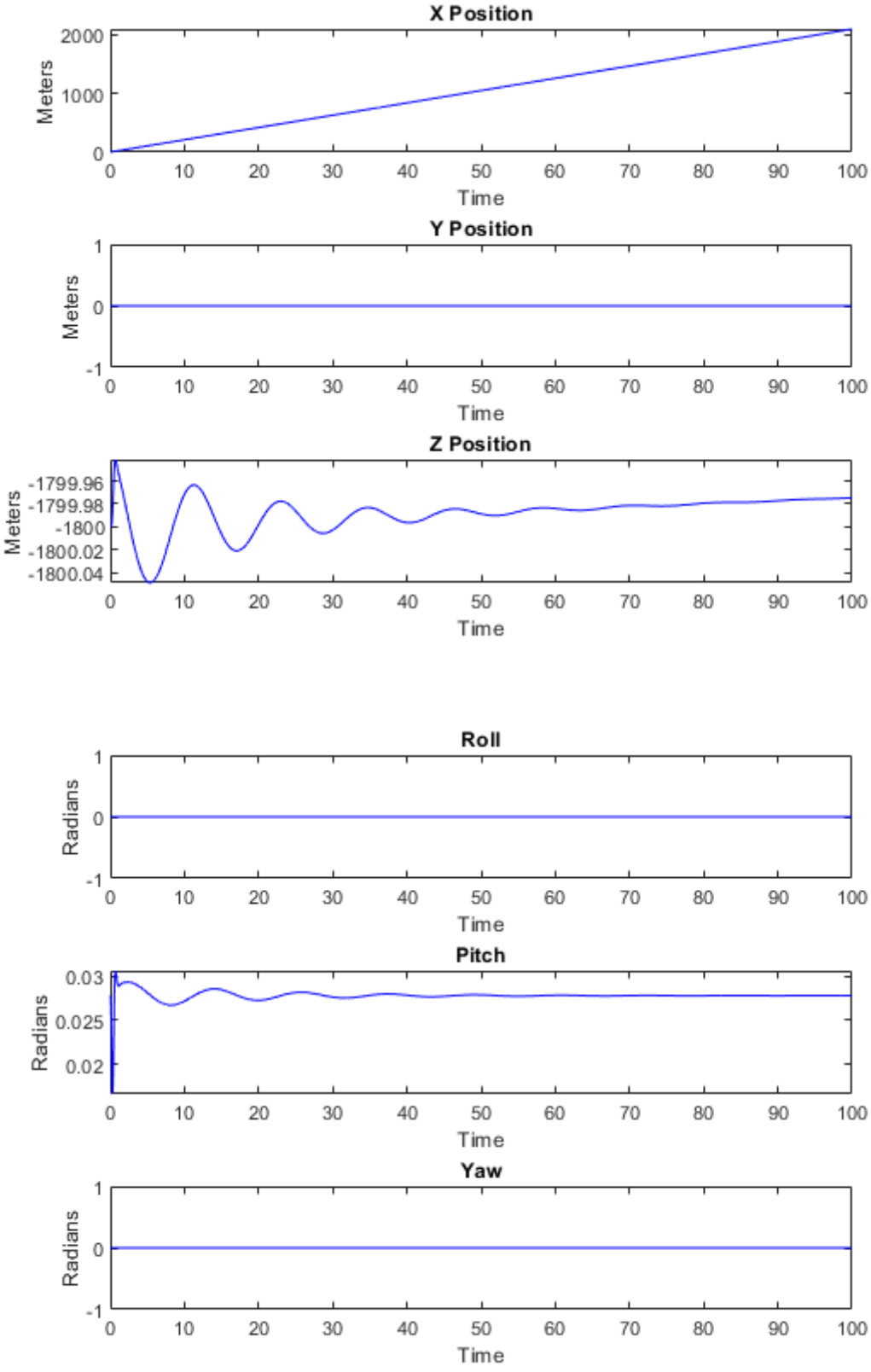
determine aero moment coefficients

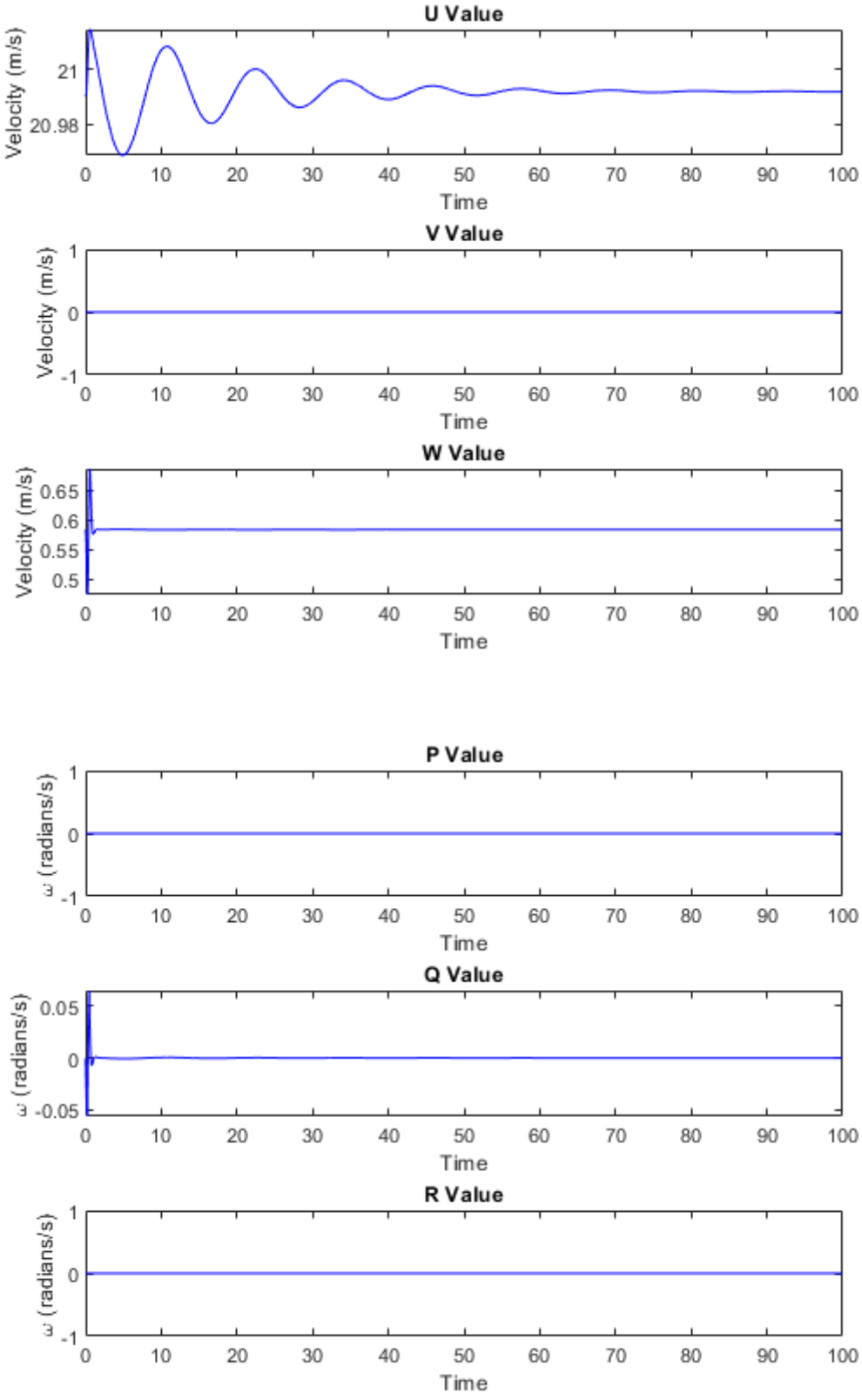
```
Cl = ap.b*[ap.Cl0 + ap.Clbeta*beta + ap.Clp*p*ap.b/(2*V) + ap.Clr*r*ap.b/
(2*V) + ap.Clda*da + ap.Cldr*dr];
Cm = ap.c*[ap.Cm0 + ap.Cmalpha*alpha + ap.Cmq*q*ap.c/(2*V) + ap.Cmde*de];
Cn = ap.b*[ap.Cn0 + ap.Cnbeta*beta + ap.Cnp*p*ap.b/(2*V) + ap.Cnr*r*ap.b/
(2*V) + ap.Cnda*da + ap.Cndr*dr];
```

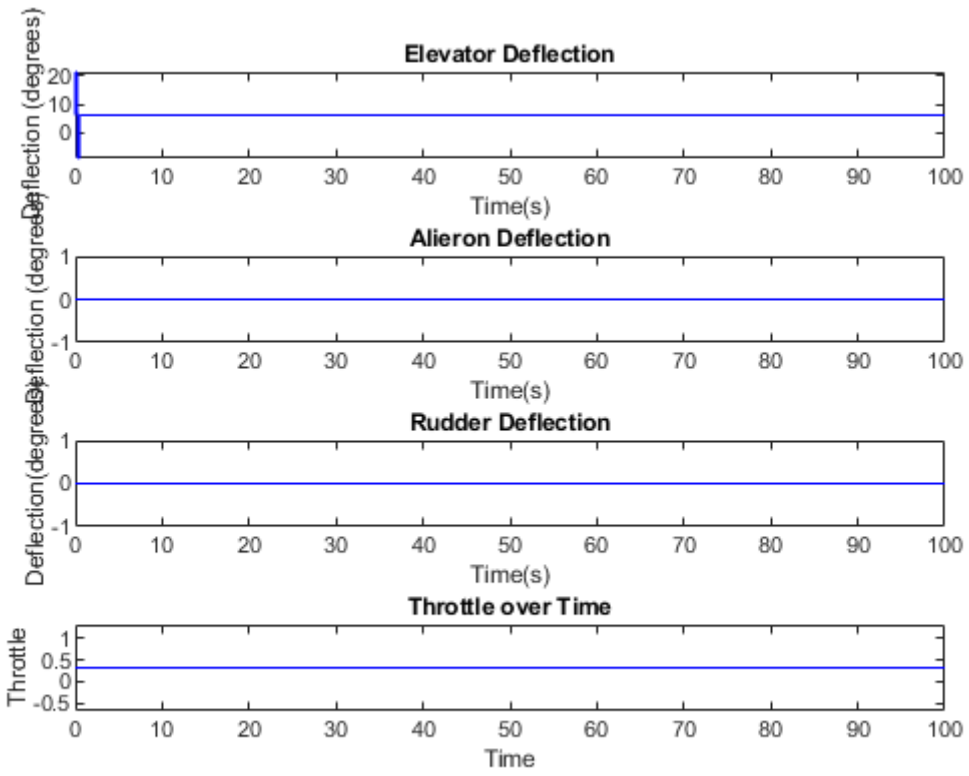
determine aero moments from coeffficients

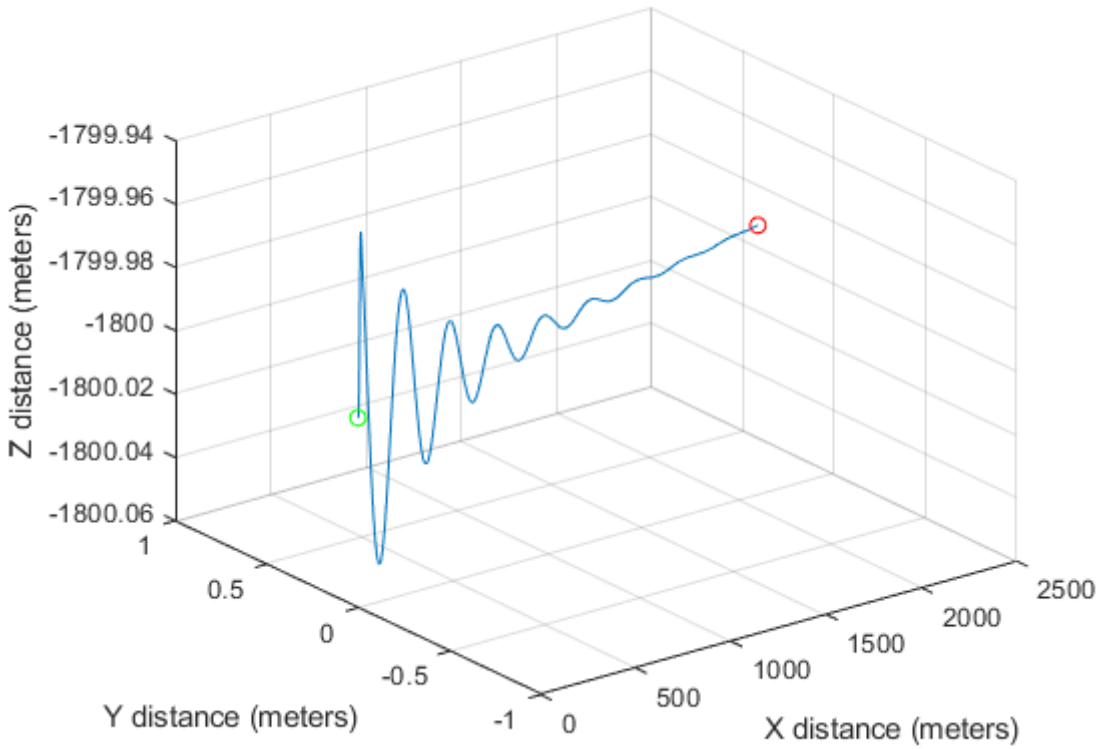
```
aero_moments = Q*ap.S*[Cl; Cm; Cn];%[l;m;n];
end
function wind_body = TransformFromInertialToBody(wind_inertial,rot_ang)
```

```
%Extracting 3-2-1 euler angles
phi = rot_ang(1);
theta = rot_ang(2);
psi = rot_ang(3);

%Calculating each row of DCM

Row1 = [cos(theta)*cos(psi),...
            cos(theta)*sin(psi),...
                -sin(theta)];

Row2 = [sin(phi)*sin(theta)*cos(psi)-cos(phi)*sin(psi),...
            sin(phi)*sin(theta)*sin(psi)+cos(phi)*cos(psi),...
                sin(phi)*cos(theta)];

Row3 = [cos(phi)*sin(theta)*cos(psi)+sin(phi)*sin(psi),...
            cos(phi)*sin(theta)*sin(psi)-sin(phi)*cos(psi),...
                cos(phi)*cos(theta)];

%Concatenating rows to return 3 x 3 DCM
R321 = [Row1; Row2; Row3];

wind_body = R321*wind_inertial;
end
function [wind_angles] = WindAnglesFromVelocityBody(velocity_body)

V = norm(velocity_body);
alpha = atan2(velocity_body(3,1),velocity_body(1,1));
beta = asin(velocity_body(2,1)/V);

wind_angles = [V; beta; alpha];
end
```


X Position
Meters
2000
1000
0
0 10 20 30 40 50 60 70 80 90 100
Time
Y Position
Meters
1
0
-1
0 10 20 30 40 50 60 70 80 90 100
Time
Z Position
Meters
-1799.96
-1799.98
-1800
-1800.02
-1800.04
0 10 20 30 40 50 60 70 80 90 100
Time
Roll
Radians
1
0
-1
0 10 20 30 40 50 60 70 80 90 100
Time
Pitch
Radians
0.03
0.025
0.02
0 10 20 30 40 50 60 70 80 90 100
Time
Yaw
Radians
1
0
-1
0 10 20 30 40 50 60 70 80 90 100
Time


U Value
Velocity (m/s)
21
20.98
0 10 20 30 40 50 60 70 80 90 100
Time
V Value
Velocity (m/s)
1
0
-1
0 10 20 30 40 50 60 70 80 90 100
Time
W Value
Velocity (m/s)
0.65
0.6
0.55
0.5
0 10 20 30 40 50 60 70 80 90 100
Time
P Value
ω (radians/s)
1
0
-1
0 10 20 30 40 50 60 70 80 90 100
Time
Q Value
ω (radians/s)
0.05
0
-0.05
0 10 20 30 40 50 60 70 80 90 100
Time
R Value
ω (radians/s)
1
0
-1
0 10 20 30 40 50 60 70 80 90 100
Time


Elevator Deflection
Deflection (degrees)
Time(s)
Alieron Deflection
Deflection (degrees)
Time(s)
Rudder Deflection
Deflection (degrees)
Time(s)
Throttle over Time
Throttle
Time


Z distance (meters)
Y distance (meters)
X distance (meters)

Published with MATLAB® R2023b