Cairo University Faculty of
Engineering Aerospace
Department

AEROSPACE ENGINEERING

3/12/2022

Task 3

Airframe model

Autopilot -AER 408
Dr.Osama Saaid

Team 4

Name	sec	BN
Mohammed Ahmed Hassan Ahmed	2	37
Ibrahim Thabet Allam	1	1
Mohammed Hatem Mohammed Saeed	2	39
Mohammed Abd El-Mawgoud Ghoneam	2	43
Mohamed Hassan Gad Ali	2	41

Contents

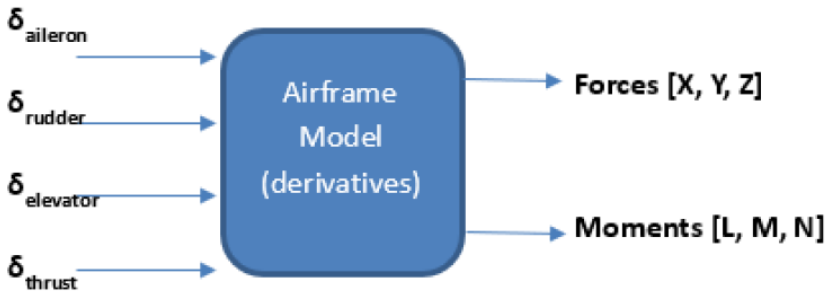
Task #3

Introduction

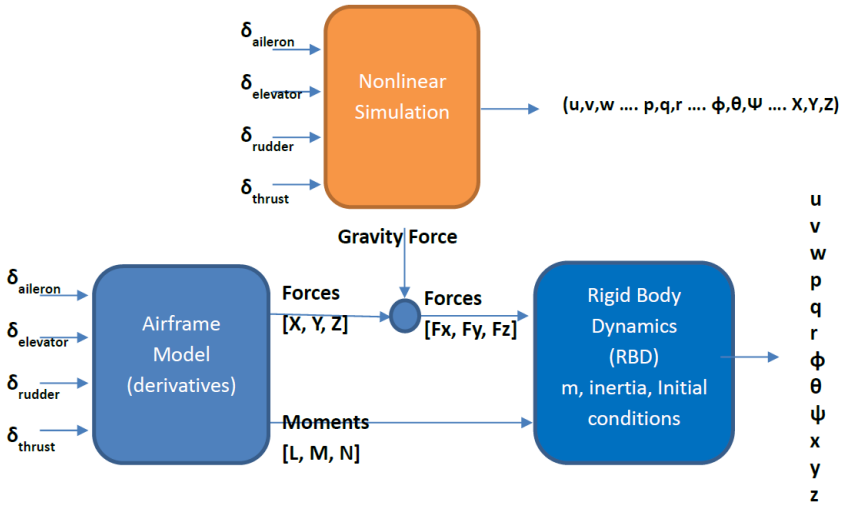
In this task we are concerned about calculating the change in states due to change in control surfaces at certain aircraft data.

We will build an airframe model code that takes the control surfaces deflection values

$$\delta_{aileron}, \delta_{thrust}, \delta_{rudder}, \delta_{elevator}$$



We will combine the airframe model code with the RBD solver code that we made to make a nonlinear simulator



Equations representing the change in the Aerodynamic & thrust forces & moments:

$$\Delta X = \frac{\partial X}{\partial u}\Delta u + \frac{\partial X}{\partial w}\Delta w + \frac{\partial X}{\partial \delta_e}\Delta \delta_e + \frac{\partial X}{\partial \delta_T}\Delta \delta_T$$

$$\Delta Y = \frac{\partial Y}{\partial v}\Delta v + \frac{\partial Y}{\partial p}\Delta p + \frac{\partial Y}{\partial r}\Delta r + \frac{\partial Y}{\partial \delta_r}\Delta \delta_r$$

$$\Delta Z = \frac{\partial Z}{\partial u}\Delta u + \frac{\partial Z}{\partial w}\Delta w + \frac{\partial Z}{\partial \dot{w}}\Delta \dot{w} + \frac{\partial Z}{\partial q}\Delta q + \frac{\partial Z}{\partial \delta_e}\Delta \delta_e + \frac{\partial Z}{\partial \delta_T}\Delta \delta_T$$

$$\Delta L = \frac{\partial L}{\partial v}\Delta v + \frac{\partial L}{\partial p}\Delta p + \frac{\partial L}{\partial r}\Delta r + \frac{\partial L}{\partial \delta_r}\Delta \delta_r + \frac{\partial L}{\partial \delta_a}\Delta \delta_a$$

$$\Delta M = \frac{\partial M}{\partial u}\Delta u + \frac{\partial M}{\partial w}\Delta w + \frac{\partial M}{\partial \dot{w}}\Delta \dot{w} + \frac{\partial M}{\partial q}\Delta q + \frac{\partial M}{\partial \delta_e}\Delta \delta_e + \frac{\partial M}{\partial \delta_T}\Delta \delta_T$$

$$\Delta N = \frac{\partial N}{\partial v}\Delta v + \frac{\partial N}{\partial p}\Delta p + \frac{\partial N}{\partial r}\Delta r + \frac{\partial N}{\partial \delta_r}\Delta \delta_r + \frac{\partial N}{\partial \delta_a}\Delta \delta_a$$

Axes systems

There are three types of body axes

- Principle axes ($I_{xy} = 0$)
- Stability Axes ($w_0 = 0$), X_b $axis$ concides with V_T
- Body axes (fuselage reference axis)

we will transform from stability axes derivatives to body axes

$$L'_\beta = (L_\beta + I_{xz}N_\beta/I_x)G \quad 1/sec^2$$

$$L'_p = (L_p + I_{xz}N_p/I_x)G \quad 1/sec$$

$$L'_r = (L_r + I_{xz}N_r/I_x)G \quad 1/sec$$

$$L'_{\delta_r} = (L_{\delta_r} + I_{xz}N_{\delta_r}/I_x)G \quad 1/sec^2$$

$$L'_{\delta_a} = (L_{\delta_a} + I_{xz}N_{\delta_a}/I_x)G \quad 1/sec^2$$

$$N'_\beta = (N_\beta + I_{xz}L_\beta/I_z)G \quad 1/sec^2$$

$$N'_p = (N_p + I_{xz}L_p/I_z)G \quad 1/sec$$

$$N'_r = (N_r + I_{xz}L_r/I_z)G \quad 1/sec$$

$$N'_{\delta_r} = (N_{\delta_r} + I_{xz}L_{\delta_r}/I_z)G \quad 1/sec^2$$

$$N'_{\delta_a} = (N_{\delta_a} + I_{xz}L_{\delta_a}/I_z)G \quad 1/sec^2$$

$$G = \frac{1}{1 - \frac{I_{xz}^2}{I_x I_z}}$$

Codes

```
clc; clear; close all;
```

Inputs

Forces, Moments and Inertia

```
[Mass, g, I, invI, timeSpan, dt, ICs, ICs_dot0, Vt0, ...
    dControl, SD_Long, SD_Lat, initialGravity] = Input("NT-33A_4.xlsx");
```

Solving

```
steps = (timeSpan(2) - timeSpan(1))/dt;

Result = NaN(12, steps);

Result(:,1) = ICs;

time_V = linspace(0, timeSpan(2), steps+1);
```

```
dForces = [0 ; 0; 0];

dMoments = [0 ; 0; 0];



for i =1:steps
    tic
    Result(:, i+1) = RBDSolver(Result(:, i), dt, (initialGravity + dForces),
dMoments, Mass, I, invI, g);

    [dF, dM] = airFrame(SD_Long, SD_Lat, dControl, ICs, ICs_dot0, Result(:, i+1)
,Vt0, ...
        (initialGravity + dForces), dMoments, Mass, I, invI, g);

    toc
    dForces = vpa(dF');
    dMoments = vpa(dM');

end
```

Plotting

Rearranging Results

```
u = Result(1,:);

v = Result(2,:);

w = Result(3,:);

p = Result(4,:);

q = Result(5,:);

r = Result(6,:);

phi = Result(7,:);

theta = Result(8,:);

psi = Result(9,:);

x = Result(10,:);

y = Result(11,:);

z = Result(12,:);



beta_deg=asin(v/Vt0)*180/pi;

alpha_deg=atan(w./u)*180/pi;
```

```
p_deg=p*180/pi;

q_deg=q*180/pi;

r_deg=r*180/pi;

phi_deg=phi*180/pi;

theta_deg=theta*180/pi;

psi_deg=psi*180/pi;



figure

plot3(x,y,z);

title('Trajectory')
figure
subplot(4,3,1)
plot(time_V,u)
title('u (ft/sec)')
xlabel('time (sec)')
subplot(4,3,2)
plot(time_V,beta_deg)
title('\beta (deg)')
xlabel('time (sec)')
subplot(4,3,3)
plot(time_V,alpha_deg)
title('\alpha (deg)')
xlabel('time (sec)')
subplot(4,3,4)
plot(time_V,p_deg)
title('p (deg/sec)')
xlabel('time (sec)')
subplot(4,3,5)
plot(time_V,q_deg)
title('q (deg/sec)')
xlabel('time (sec)')
subplot(4,3,6)
plot(time_V,r_deg)
title('r (deg/sec)')
xlabel('time (sec)')
subplot(4,3,7)
plot(time_V,phi_deg)
title('\phi (deg)')
xlabel('time (sec)')
subplot(4,3,8)
plot(time_V,theta_deg)
title('\theta (deg)')
xlabel('time (sec)')
subplot(4,3,9)
plot(time_V,psi_deg)
title('\psi (deg)')
xlabel('time (sec)')
subplot(4,3,10)
plot(time_V,x)
title('x (ft)')
xlabel('time (sec)')
subplot(4,3,11)
plot(time_V,y)
title('y (ft)')
xlabel('time (sec)')
```

```
subplot(4,3,12)
plot(time_V,z)
title('z (ft)')
xlabel('time (sec)')
```

```
function [Mass, g, I, invI, timeSpan, dt, ICs, ICs_dot0, Vt0, ...
    dc, SD_Long, SD_Lat, initialGravity] = Input(inputs_filename)
    % Inputs
    % here B2:B61 means read the excel sheet from cell B2 to cell B61
    aircraft_data=xlsread(inputs_filename,'B2:B61');

    % Integration time span & Step
    dt = aircraft_data(1);
    tfinal = aircraft_data(2);
    timeSpan = [0 tfinal];

    % Initial Conditions
    % [u; v; w; p; q; r; phi; theta; epsi; xe0; ye0; ze0]
    % ICs = [10; 2; 0; 2*pi/180; pi/180; 0; 20*pi/180; 15*pi/180; 30*pi/180; 2; 4;
7];
    ICs = aircraft_data(4:15);
    ICs_dot0 = zeros(12,1);
    Vt0 = sqrt(ICs(1)^2 + ICs(2)^2 + ICs(3)^2);    % Vto

    % control actions values
    % D_a, D_r, D_e, D_th
    dc = [ aircraft_data(57:59) * pi/180 ; aircraft_data(60)];

    % gravity, mass % inertia
    Mass = aircraft_data(51);
    g = aircraft_data(52);
    Ixx = aircraft_data(53);
    Iyy = aircraft_data(54);
    Izz = aircraft_data(55);
    Ixz = aircraft_data(56);
    Ixy=0;  Iyz=0;
    I = [Ixx , -Ixy , -Ixz ;...
        -Ixy , Iyy , -Iyz ;...
        -Ixz , -Iyz , Izz];
    invI = inv(I);


    % Stability Derivatives Longitudinal motion
    SD_Long = aircraft_data(21:36);

    % Stability Derivatives Lateral motion
    SD_Lat_dash = aircraft_data(37:50);
    SD_Lat_dash(9) = SD_Lat_dash(9)*Vt0;    % From dimension-less to dimensional
    SD_Lat_dash(10) = SD_Lat_dash(10)*Vt0;  % Form dimension-less to dimensional

    SD_Lat = LateralSD2BodyAxes(SD_Lat_dash, I);

    % initial gravity force
    [S, C, ~] = SCT(ICs(7:9));
    initialGravity = Mass*g*[
        S.theta;
        -S.phi*C.theta;
        -C.phi*C.theta;
    ];

end
function [dForce, dMoment] = airFrame(SD_Long, SD_Lat, dControl, state0,
state0_dot, state, Vt0 ,forces, moments, Mass, I, invI, g)
```

```
    [YV, YB, LB, NB, LP, NP, LR, NR, YDA, YDR, LDA, NDA, LDR, NDR] = feval(@(x)
x{:}, num2cell(SD_Lat));
    [XU, ZU, MU, XW, ZW, MW, ZWD, ZQ, MWD, MQ, XDE, ZDE, MDE, XD_TH, ZD_TH, MD_TH]
= feval(@(x) x{:}, num2cell(SD_Long));
    [Da, Dr, De, Dth] = feval(@(x) x{:}, num2cell(dControl));

    Ixx = I(1,1);
    Iyy = I(2,2);
    Izz = I(3,3);

    state_dot = DOF6(state, forces, moments, Mass, I, invI, g);

    ds = state - state0;
    ds_dot = state_dot - state0_dot;

    beta0 = asin(state0(2)/Vt0);
    beta = asin(state(2)/Vt0);
    dbeta = beta-beta0;

    dX = Mass*(XU*ds(1)+XW*ds(3)+XDE*De+XD_TH*Dth);
    dY = Mass*(YV*ds(2)+YB*dbeta+YDA*ds(1)+YDR*Dr);
    dZ = Mass*(ZU*ds(1)+ZW*ds(3)+ZWD*ds_dot(3)+ZQ*ds(5)+ZDE*De+ZD_TH*Dth);

    dL = Ixx*(LB*dbeta+LP*ds(4)+LR*ds(6)+LDR*Dr+LDA*Da);
    dM = Iyy*(MU*ds(1)+MW*ds(3)+MWD*ds_dot(3)+MQ*ds(5)+MDE*De+MD_TH*Dth);
    dN = Izz*(NB*dbeta+NP*ds(4)+NR*ds(6)+NDR*Dr+NDA*Da);

    dForce = [dX dY dZ];
    dMoment = [dL dM dN];

end
```

Function return the set of 12 state of the six degree of freedom system after sub., with given ICs

```
function F = DOF6(ICs, forces, Moments, Mass, Inertia, invI, g)

    % (Sin, Cos, Tan) of (phi, theta, epsi)
    [S, C, T] = SCT(ICs(7:9));

    Forces = forces + Mass*g*[
        -S.theta;
        S.phi*C.theta;
        C.phi*C.theta;
    ];

    % (u, v, w) dot
    F(1:3, 1) = Forces/Mass - cross(...
        ICs(4:6, 1), ICs(1:3, 1)...
    );

    % (p, q, r) dot
    F(4:6, 1) = invI*(Moments - cross(...
        ICs(4:6, 1), Inertia * ICs(4:6, 1)...
    ));

    % (phi, theta, epsi) dot
    F(7:9, 1) = [
        1, S.phi*T.theta, C.phi*T.theta;
        0, C.phi, -S.phi;
        0, S.phi/C.theta, C.phi/C.theta;
    ] * ICs(4:6, 1);

    % (x, y, z) dot
    F(10:12, 1) = [
        C.theta*C.epsi, (S.phi*S.theta*C.epsi - C.phi*S.epsi),
(C.phi*S.theta*C.epsi + S.phi*S.epsi);
```

```
        C.theta*S.epsi, (S.phi*S.theta*S.epsi + C.phi*C.epsi),
(C.phi*S.theta*S.epsi - S.phi*C.epsi);
        -S.theta, S.phi*C.theta, C.phi*C.theta
    ] * ICs(1:3, 1);

end
function SD_Lat = LateralSD2BodyAxes(SD_Lat_dash, I)

    [YV, YB,LBd,NBd,LPd,NPd,LRd,NRd,YDA,YDR,LDAd,NDAd,LDRd,NDRd] = feval(@(x) x{:},
num2cell(SD_Lat_dash));

    Ixx = I(1);
    Izz = I(9);
    Ixz = -I(3);

    G = 1/(1 - Ixz^2 / Ixx / Izz);

    syms LB LP LR LDR LDA NB NP NR NDR NDA
    eq1 = (LB+Ixz*NB/Ixx)*G == LBd;
    eq2 = (NB+Ixz*LB/Izz)*G == NBd;
    eq3 = (LP+Ixz*NP/Ixx)*G == LPd;
    eq4 = (NP+Ixz*LP/Izz)*G == NPd;
    eq5 = (LR+Ixz*NR/Ixx)*G == LRd;
    eq6 = (NR+Ixz*LR/Izz)*G == NRd;
    eq7 = (LDR+Ixz*NDR/Ixx)*G == LDRd;
    eq8 = (NDR+Ixz*LDR/Izz)*G == NDRd;
    eq9 = (LDA+Ixz*NDA/Ixx)*G == LDAd;
    eq10 = (NDA+Ixz*LDA/Izz)*G == NDAd;

    [A,B] = equationsToMatrix(...
        [eq1, eq2, eq3, eq4, eq5, eq6, eq7, eq8, eq9, eq10],...
        [LB LP LR LDR LDA NB NP NR NDR NDA]);

    X = A\B;

    SD_Lat = [
        YV YB ...
        X(1) X(6) X(2) X(7) ...
        X(3) X(8) ...
        YDA YDR ...
        X(5) X(10) X(4) X(9) ...
    ]';

    SD_Lat = vpa(SD_Lat);

end
```

RBD Solver is function that implements Runge-Kutta-4 Algorithm in order to integrate 6 DOF set of equations with a give Inital Conditions ICs, for single dt time step.

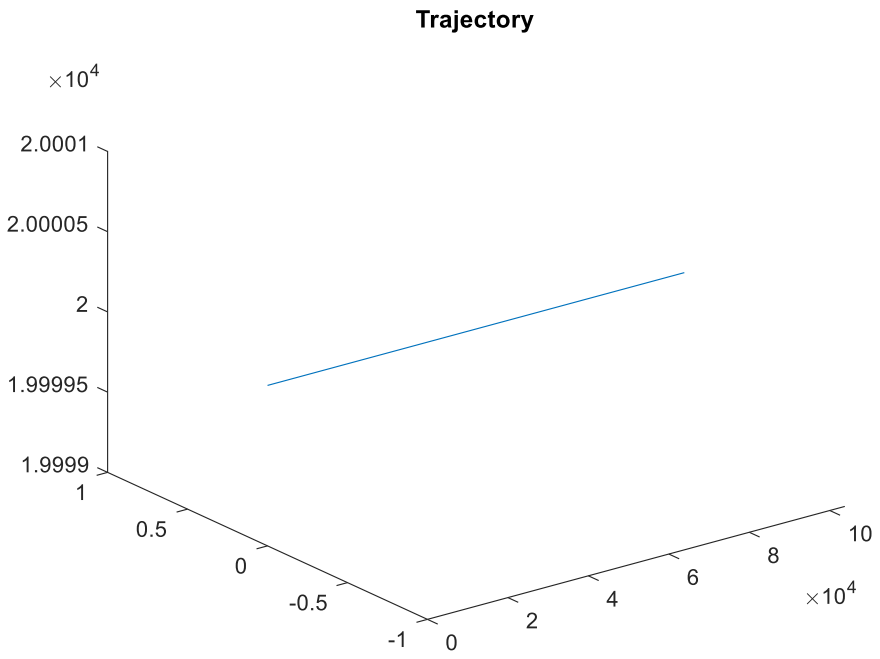
```
function state = RBDSolver(ICs, dt, Force, Moments, Mass, I, invI, g)

    K = zeros(12, 4);

    K(:, 1) = dt*DOF6(ICs ,Force, Moments, Mass, I, invI, g);
    K(:, 2) = dt*DOF6(ICs+0.5*K(:, 1) ,Force, Moments, Mass, I, invI, g);
    K(:, 3) = dt*DOF6(ICs+0.5*K(:, 2) ,Force, Moments, Mass, I, invI, g);
    K(:, 4) = dt*DOF6(ICs+K(:, 3) ,Force, Moments, Mass, I, invI, g);

    state = ICs + (...
        K(:, 1)+...
        2*K(:, 2)+...
        2*K(:, 3)+...
        K(:, 4))/6;

end
```

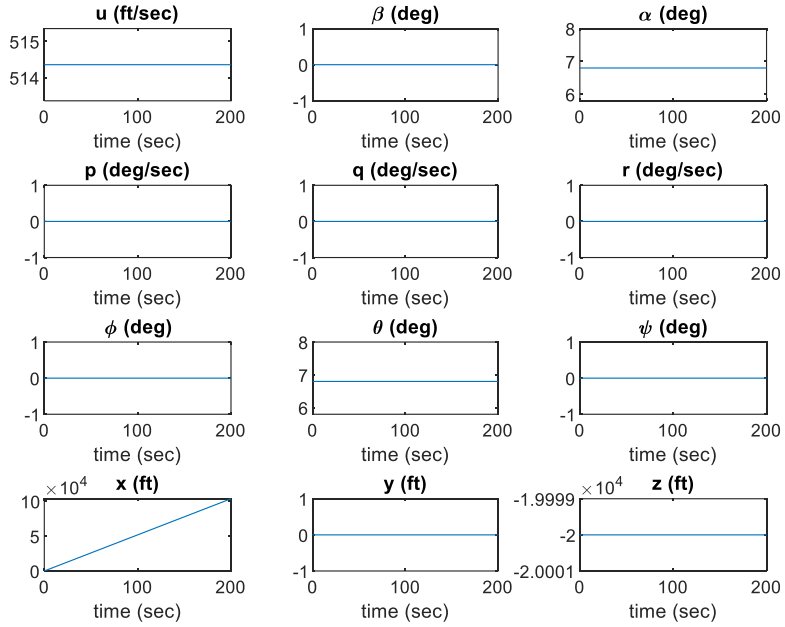
Calculate Sin, Cos ,Tan for any set of three angles and return results in struct form for easy access in code.

```
function [S, C, T] = SCT(ICs)
    S = struct(...
        'phi', sin(ICs(1)),...
        'theta', sin(ICs(2)),...
        'epsi', sin(ICs(3))...
    );
    C = struct(...
        'phi', cos(ICs(1)),...
        'theta', cos(ICs(2)),...
        'epsi', cos(ICs(3))...
    );
    T = struct(...
        'phi', tan(ICs(1)),...
        'theta', tan(ICs(2)),...
        'epsi', tan(ICs(3))...
    );
end
```

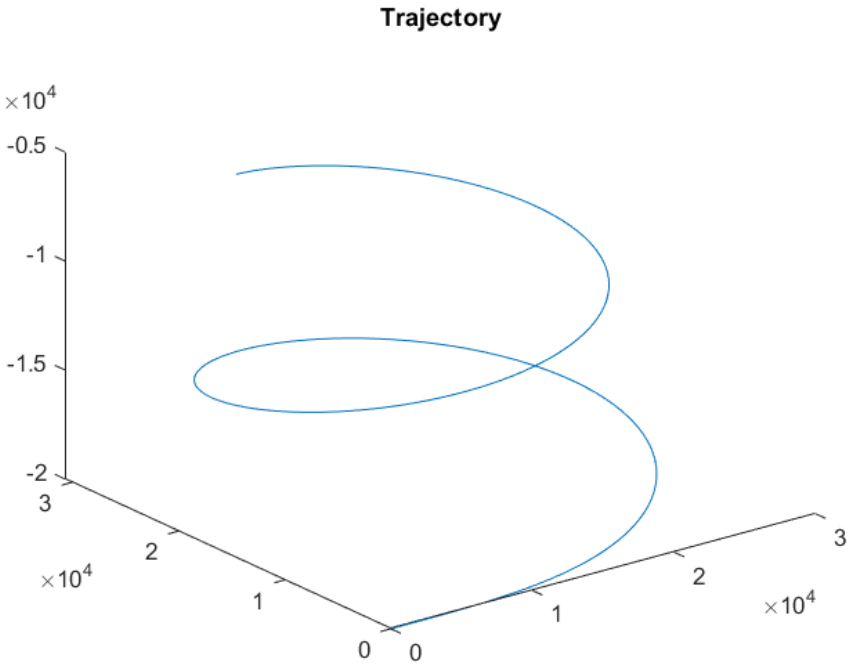
Benchmark test plots B-747 airplane

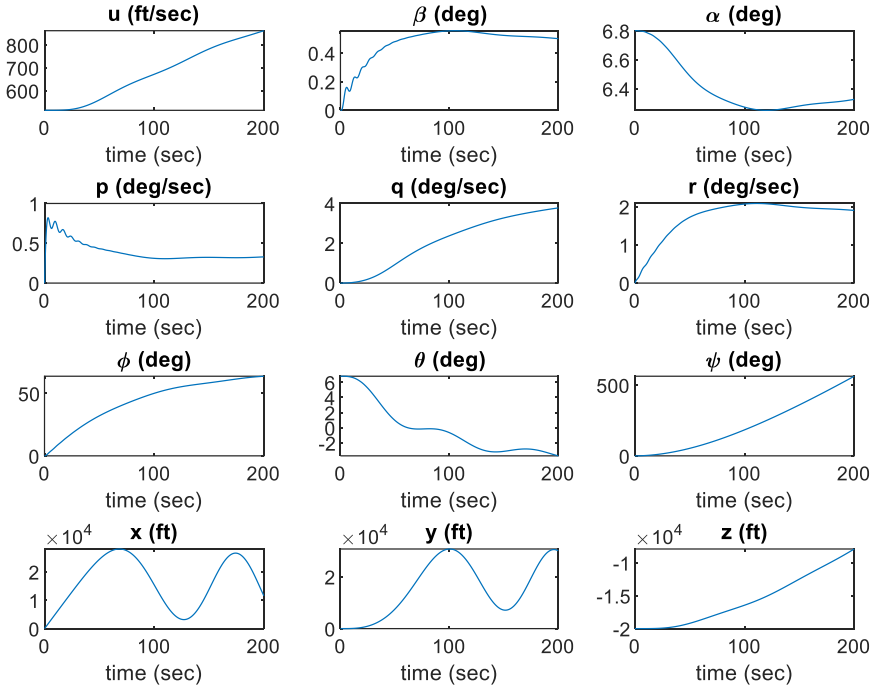
For No input



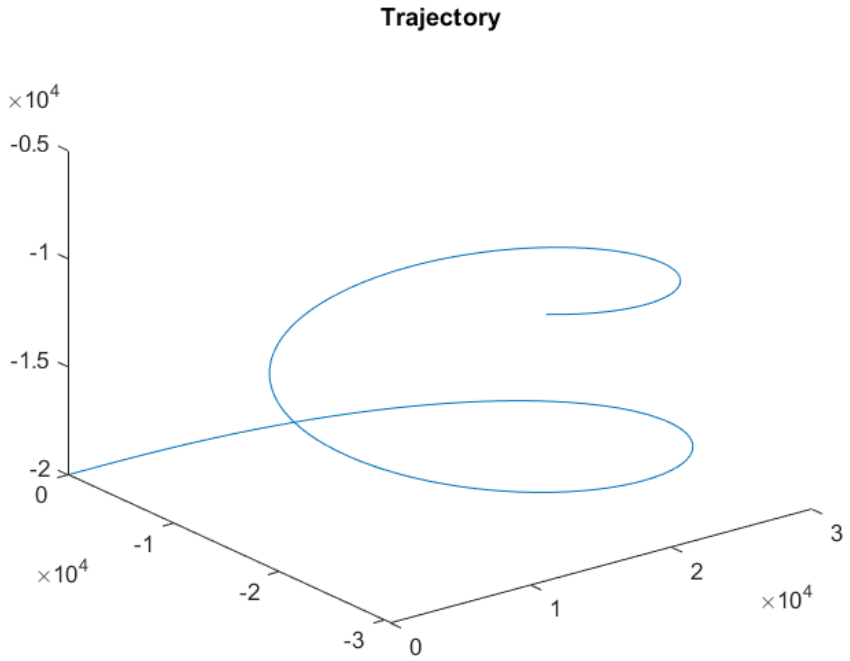


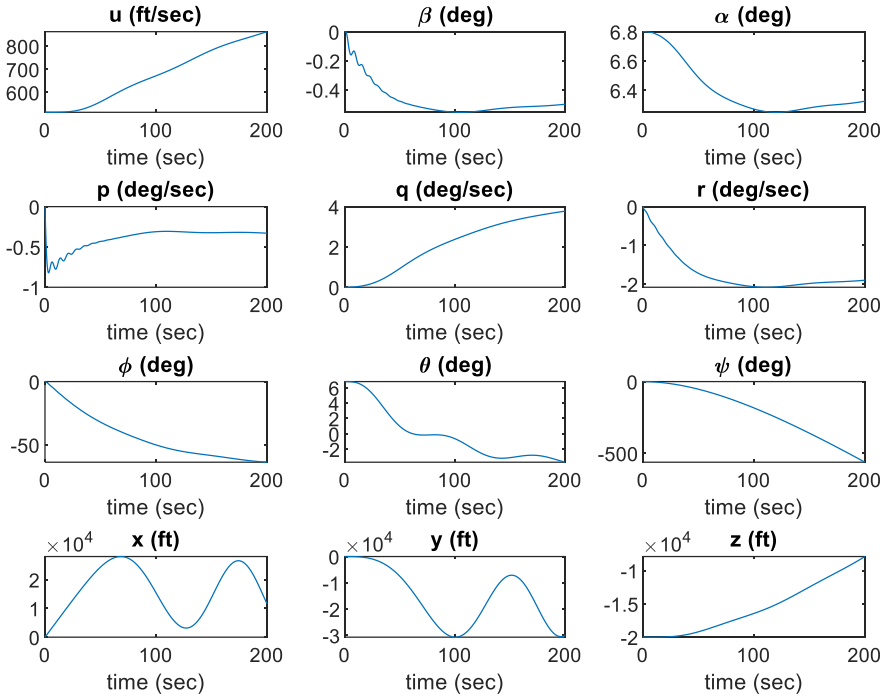
For +ve 5 degree aileron



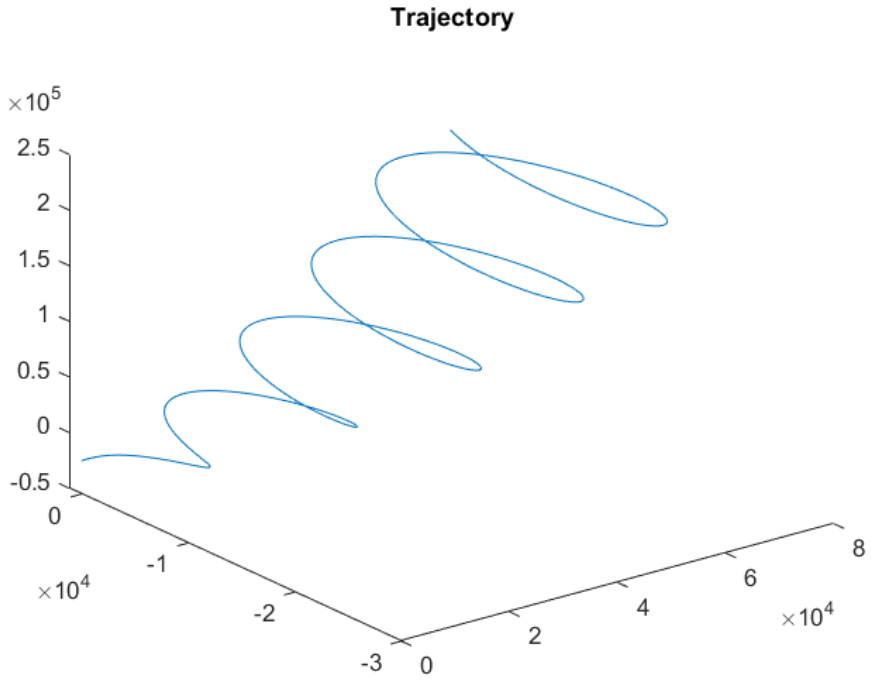


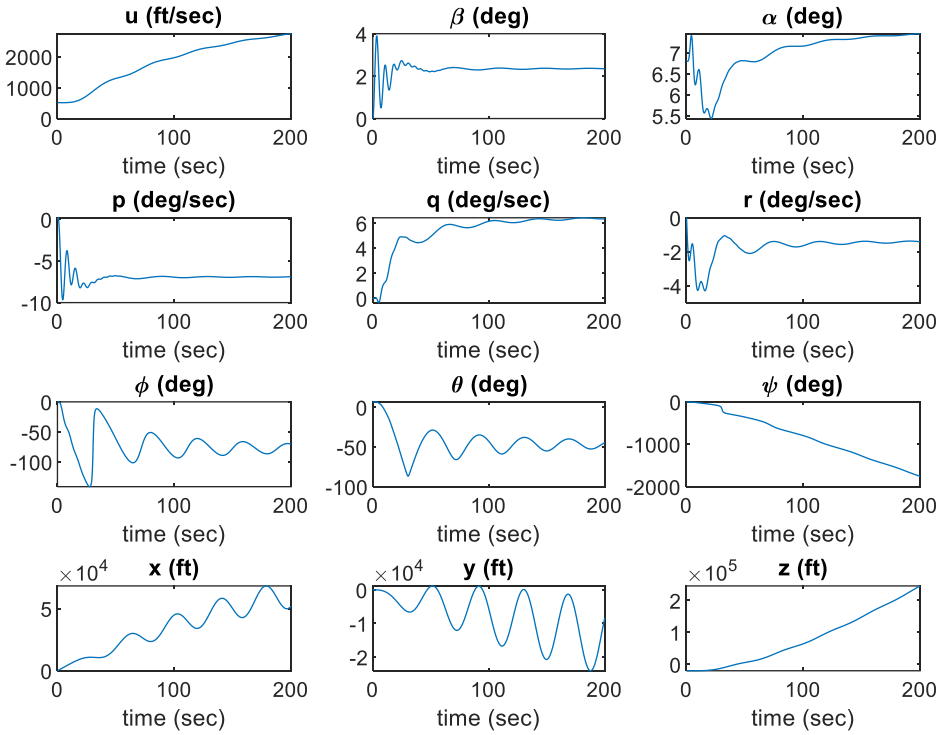
For -ve 5-degree aileron



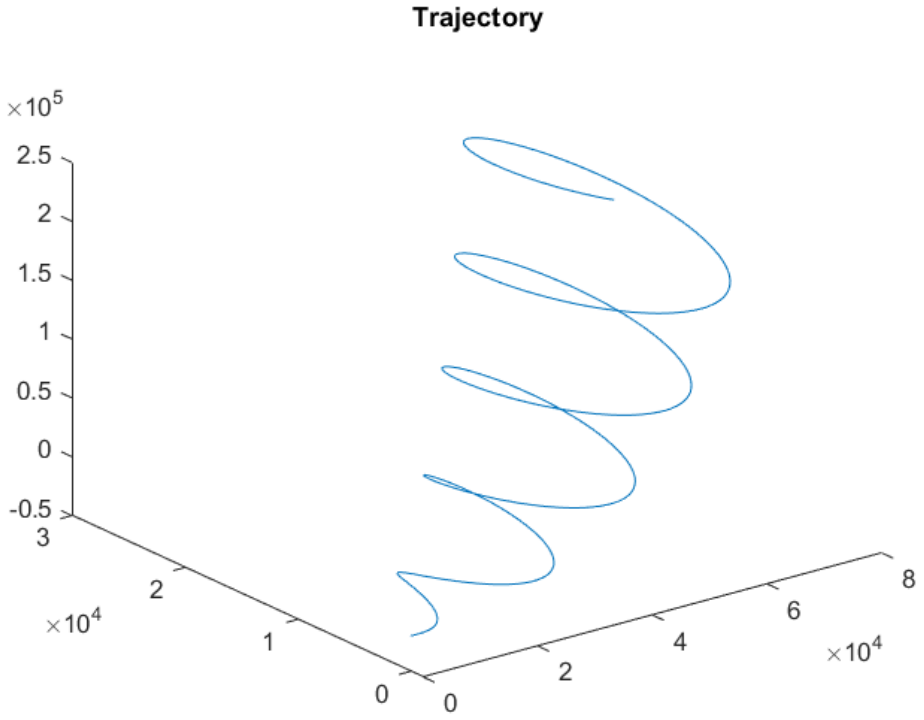


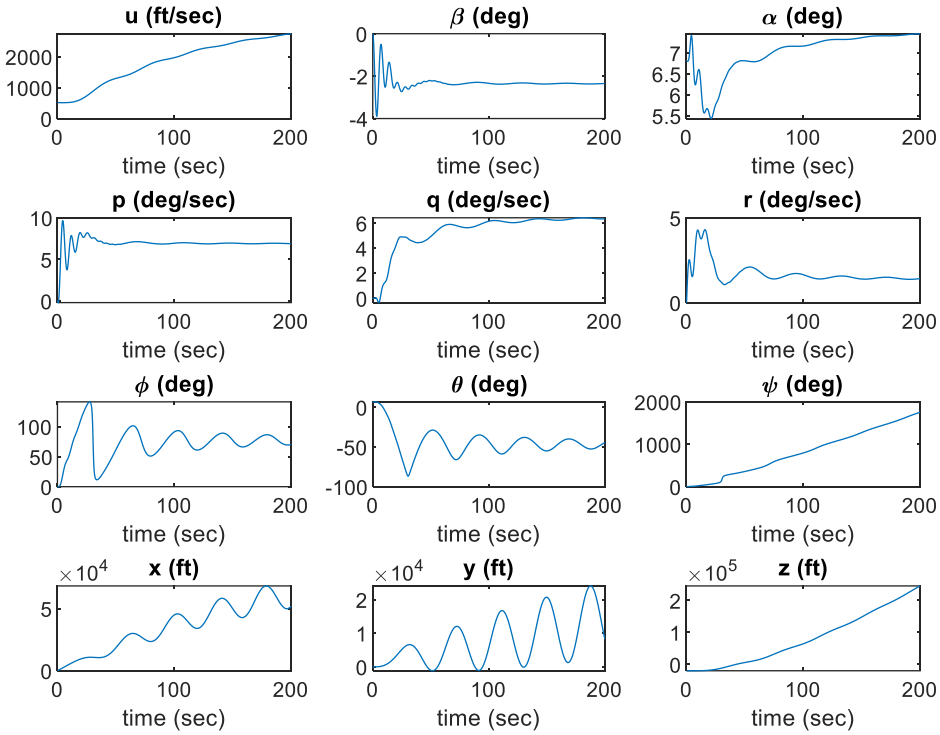
For +ve 5-degree rudder



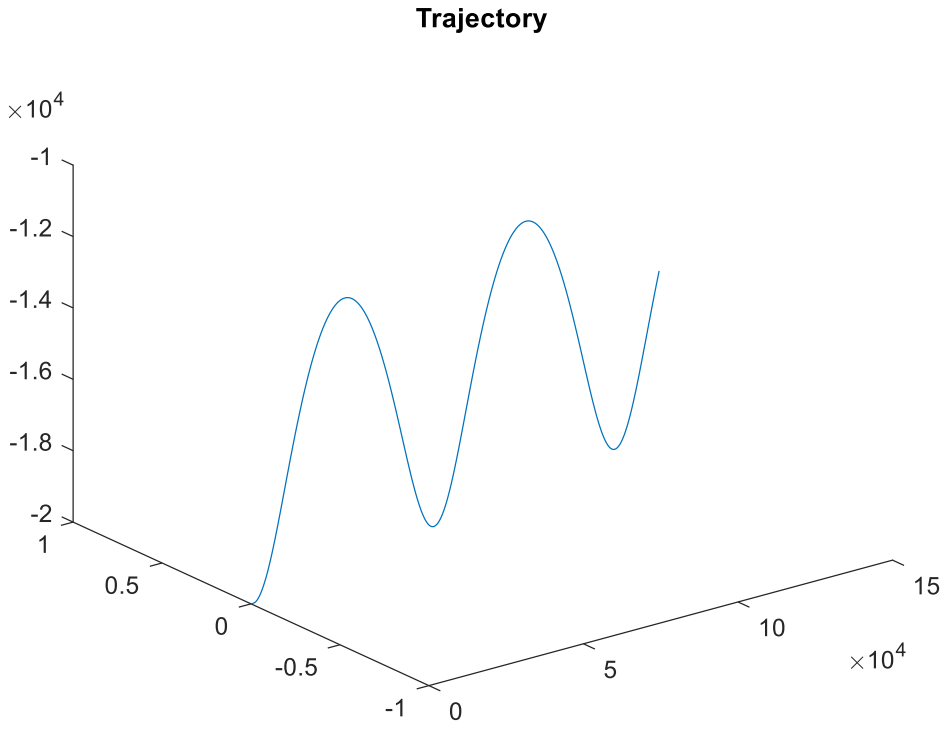


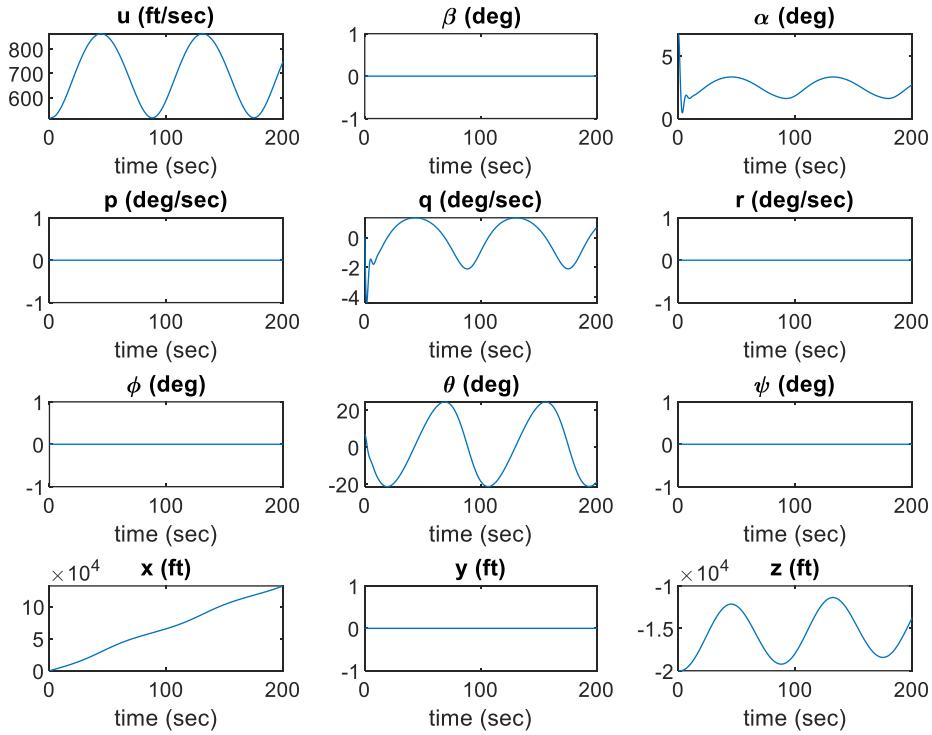
For -ve 5-degree rudder



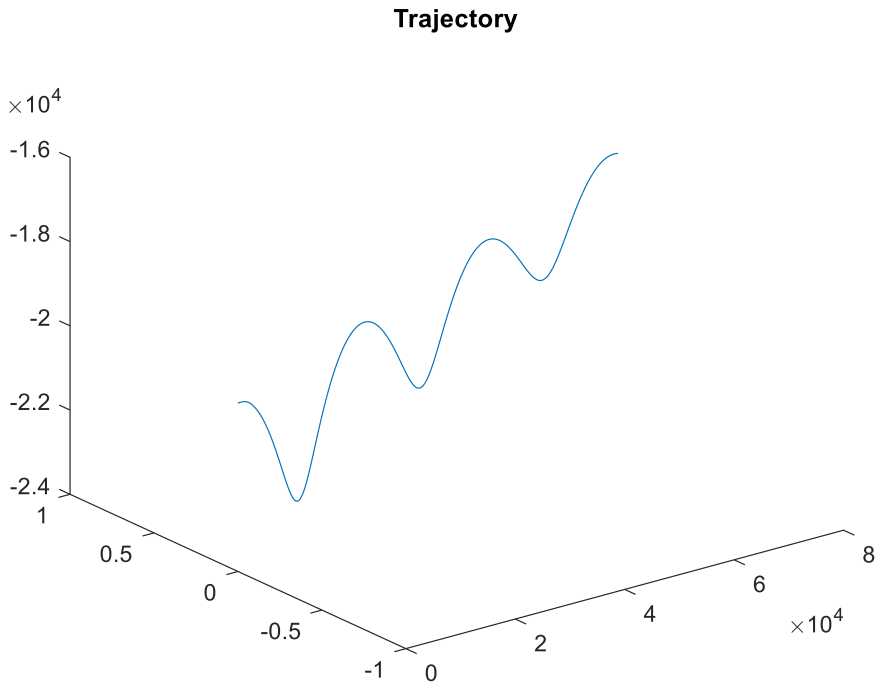


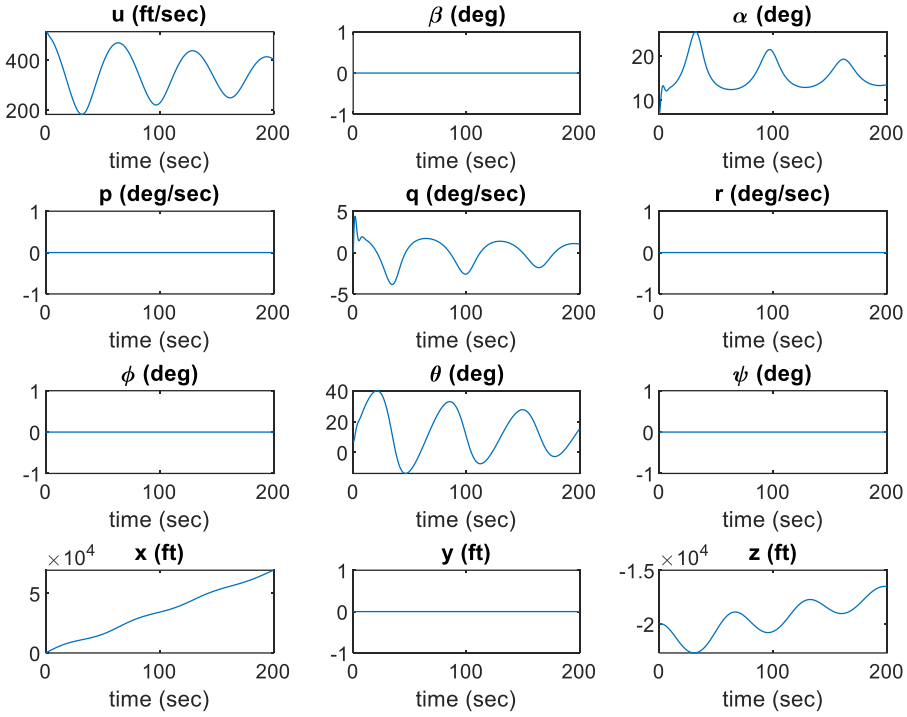
For +ve 5-degree elevator



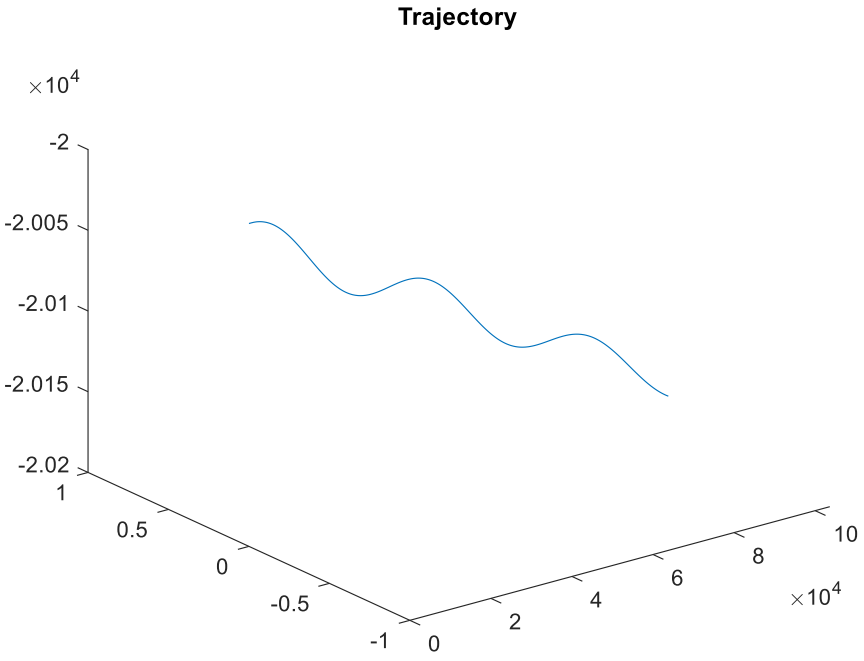


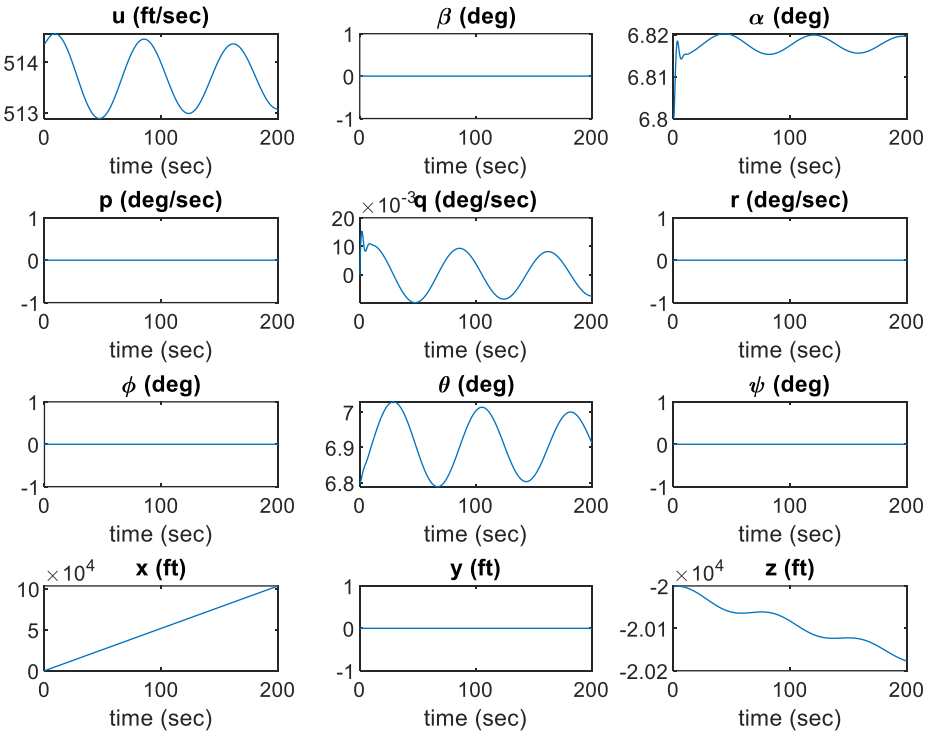
For -ve 5-degree elevator



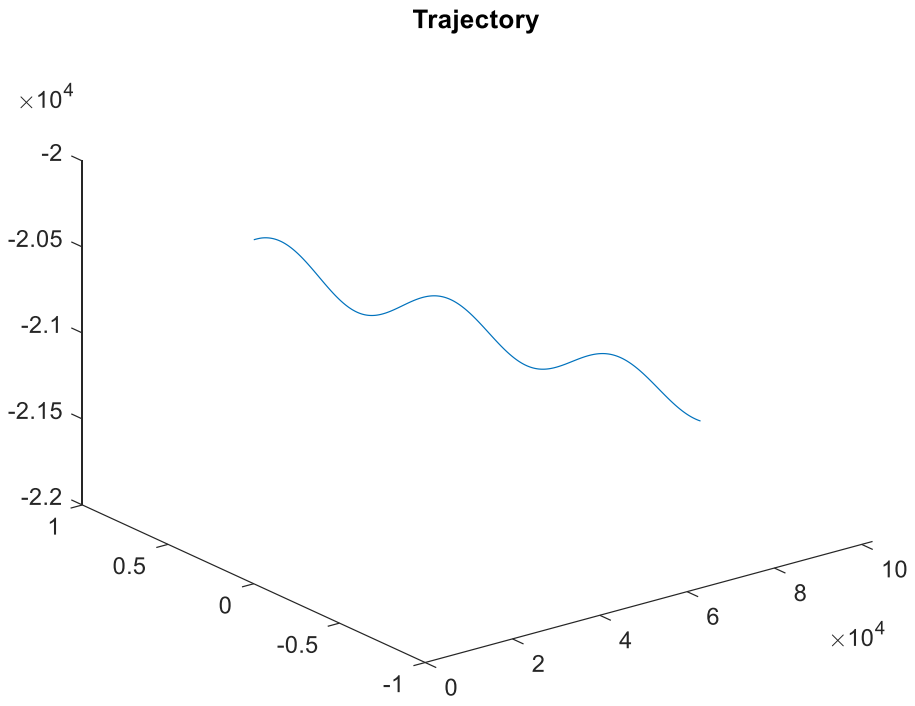


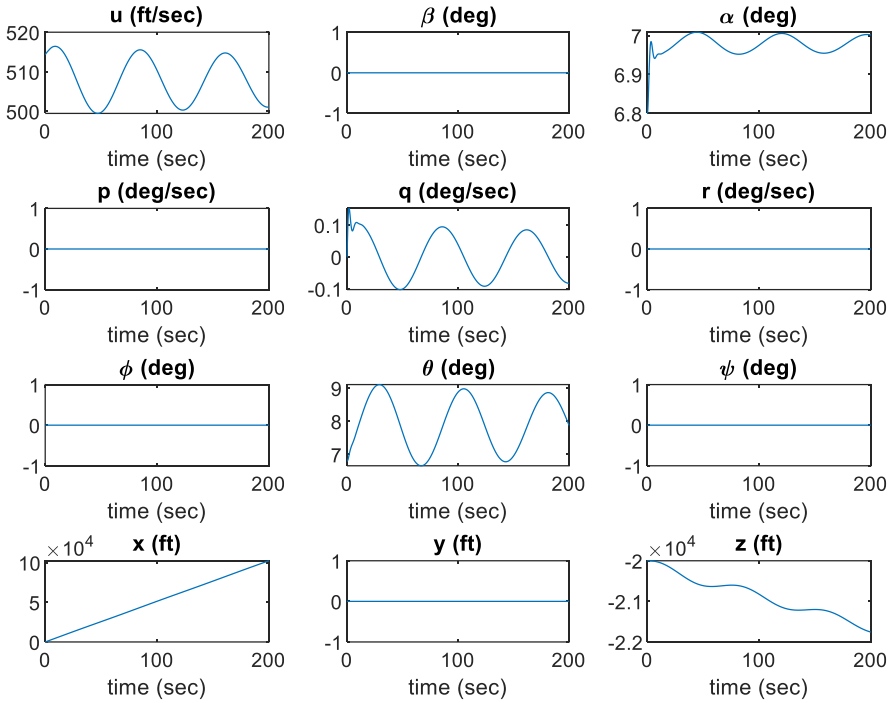
For 1000 in thrust





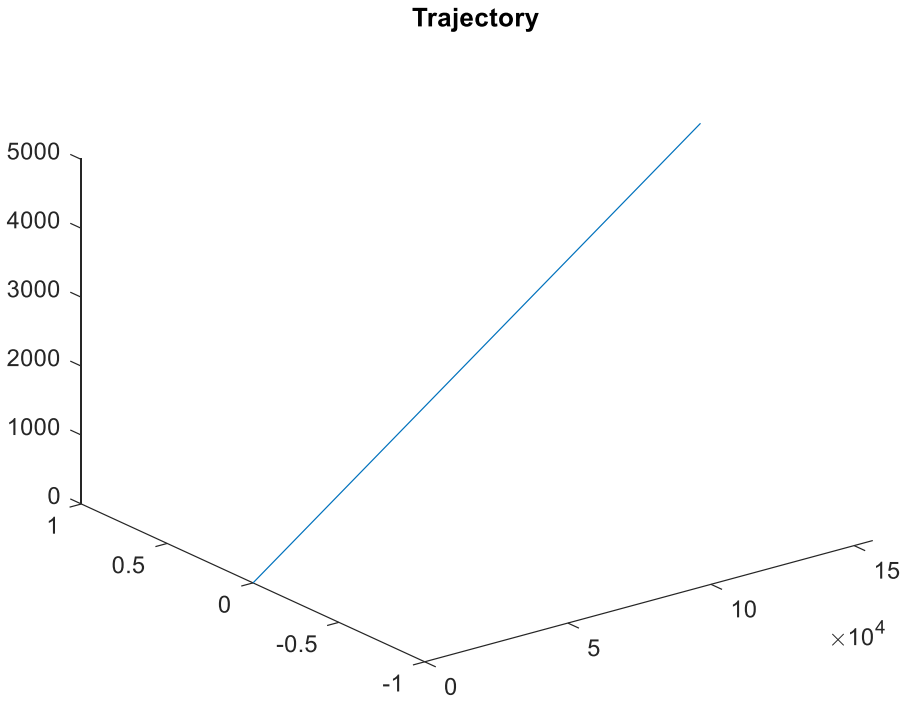
For 10000 in thrust

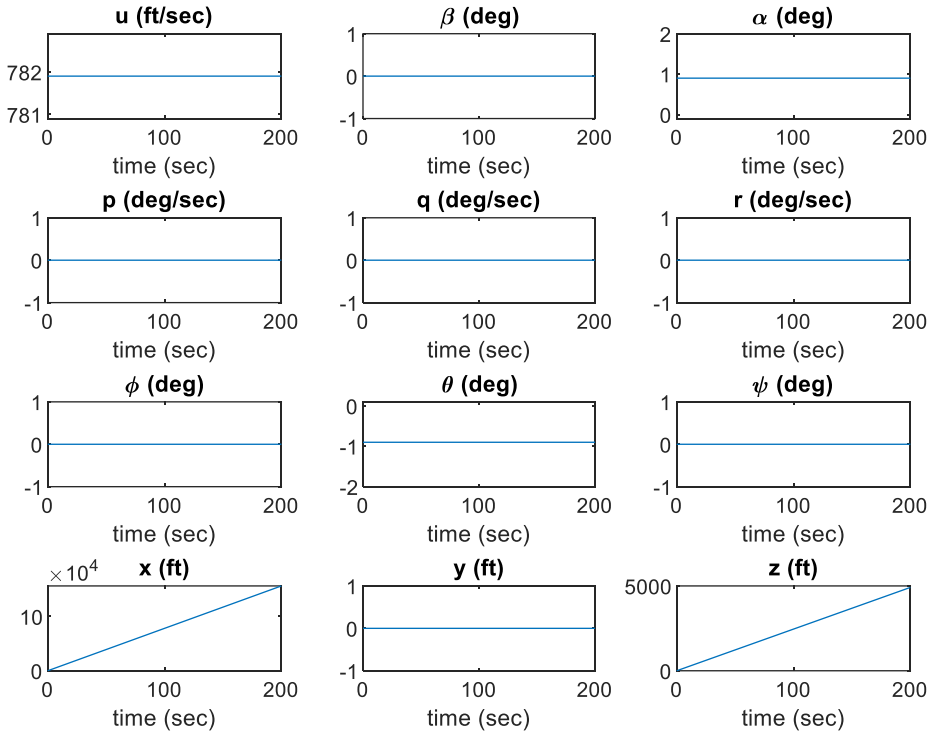




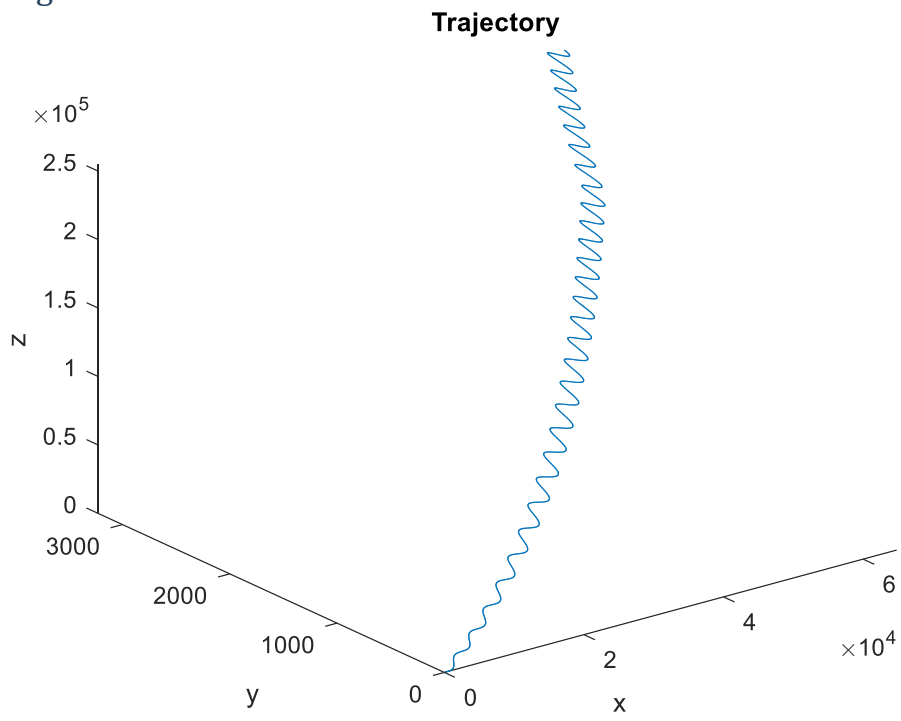
Our airplane Plots (NT-33A)

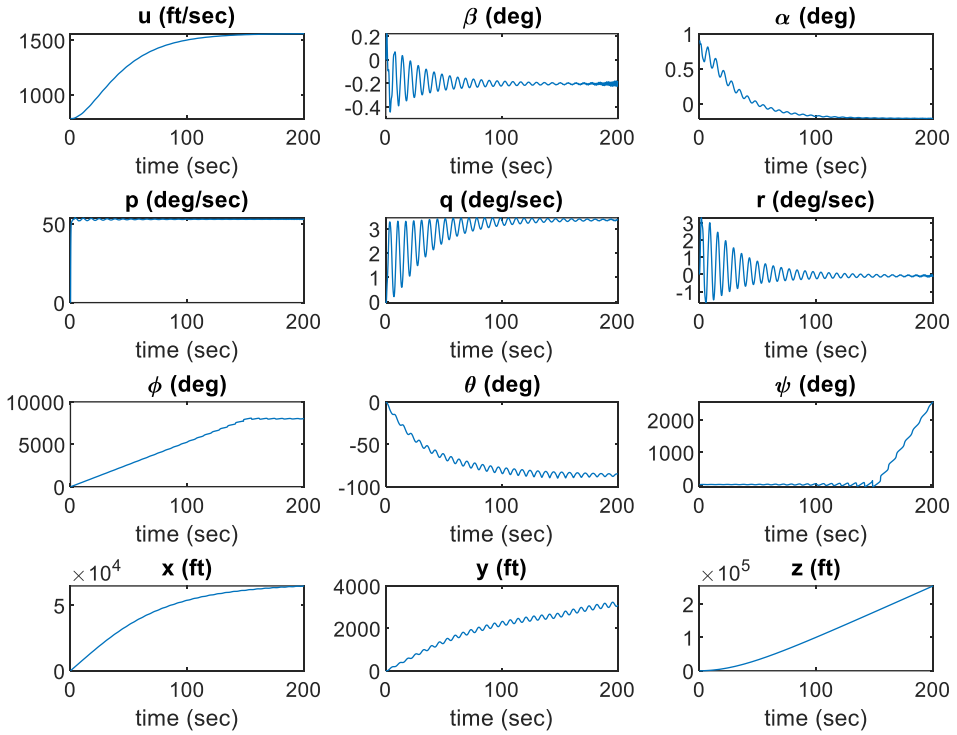
For No input



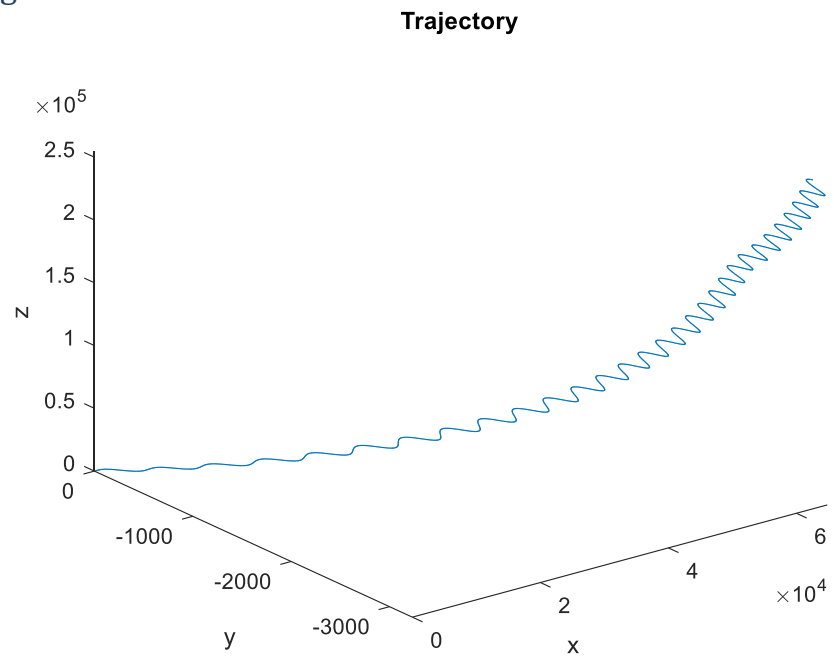


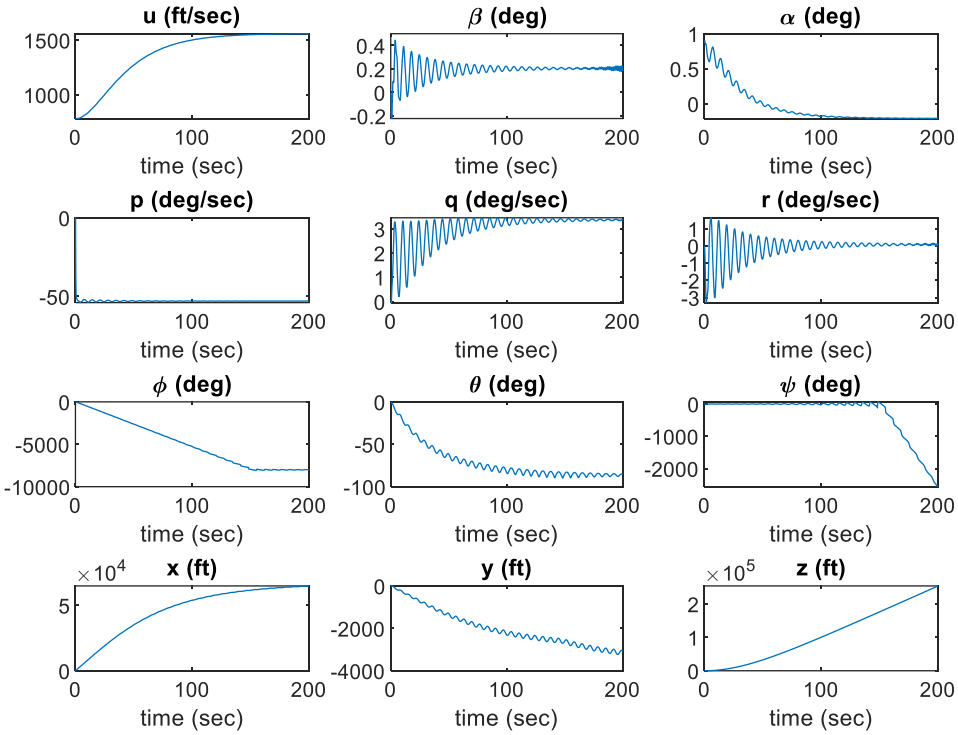
For +ve 5 degree aileron



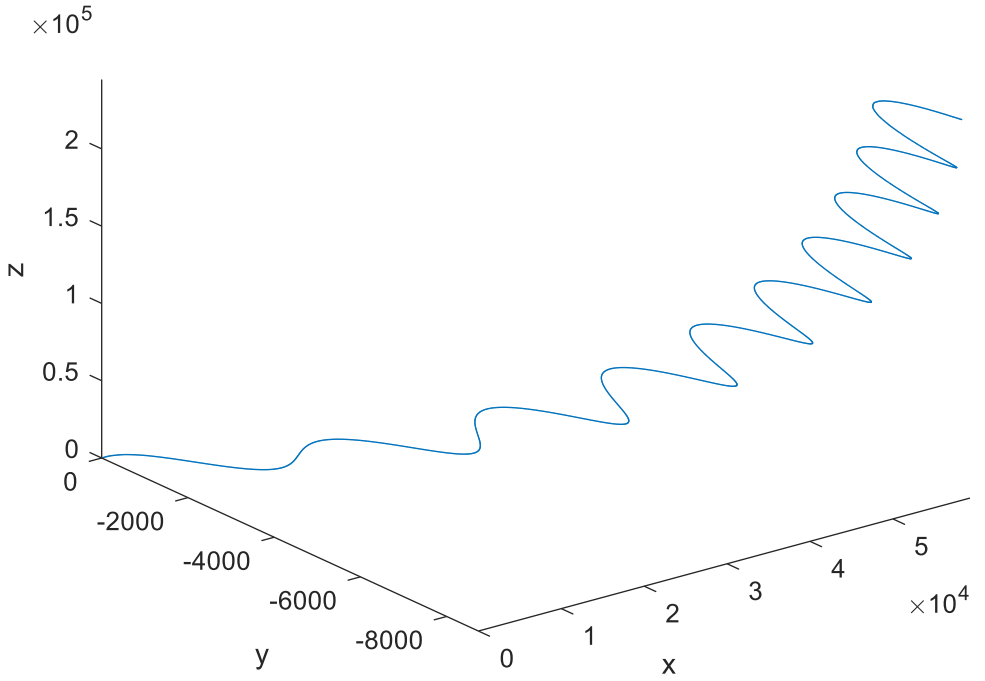


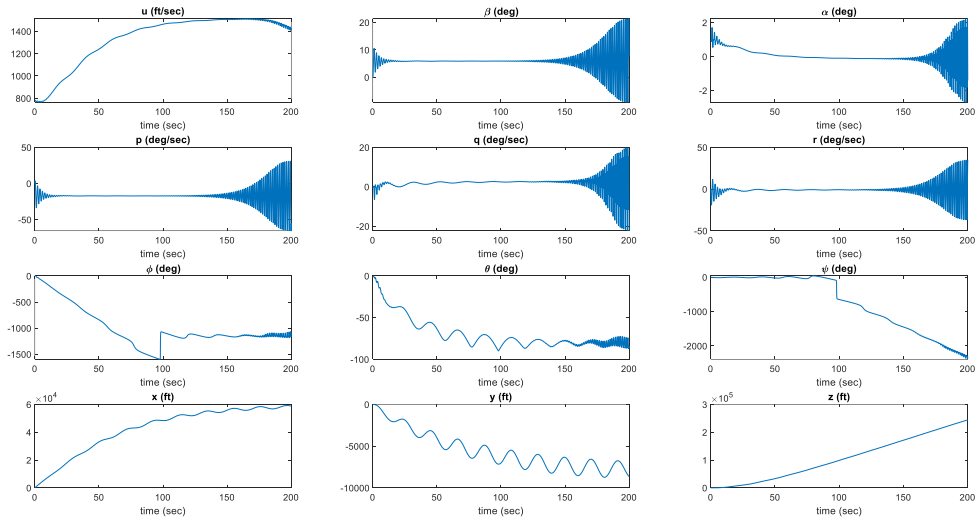
For -ve 5-degree aileron



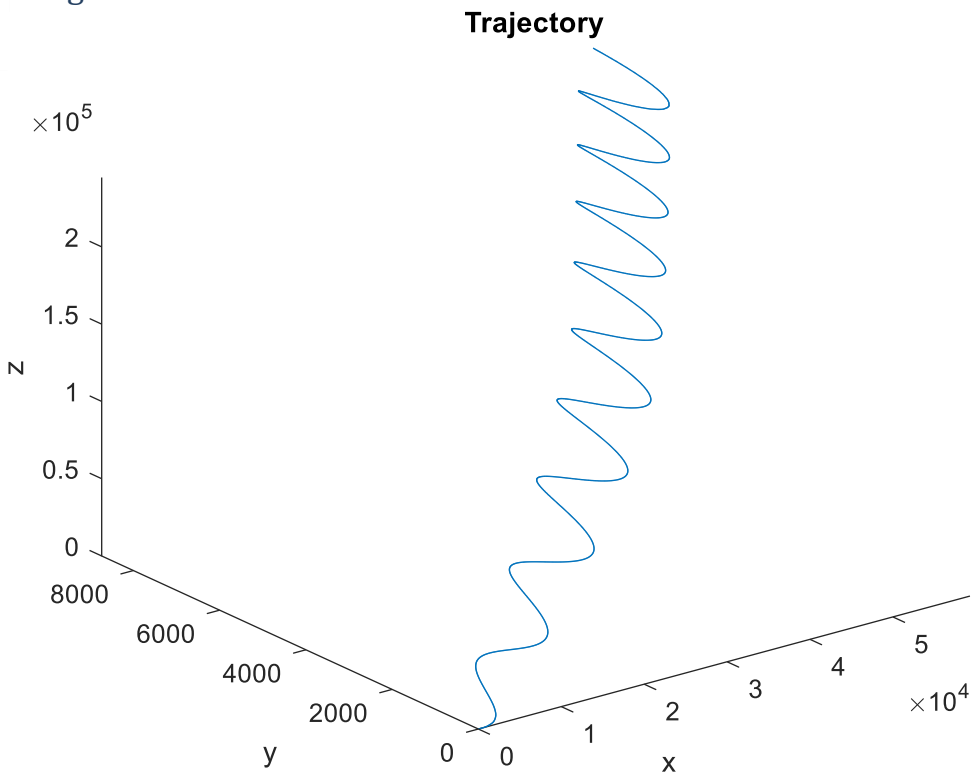

u (ft/sec)
β (deg)
α (deg)
p (deg/sec)
q (deg/sec)
r (deg/sec)
ϕ (deg)
θ (deg)
ψ (deg)
x (ft)
y (ft)
z (ft)
time (sec)

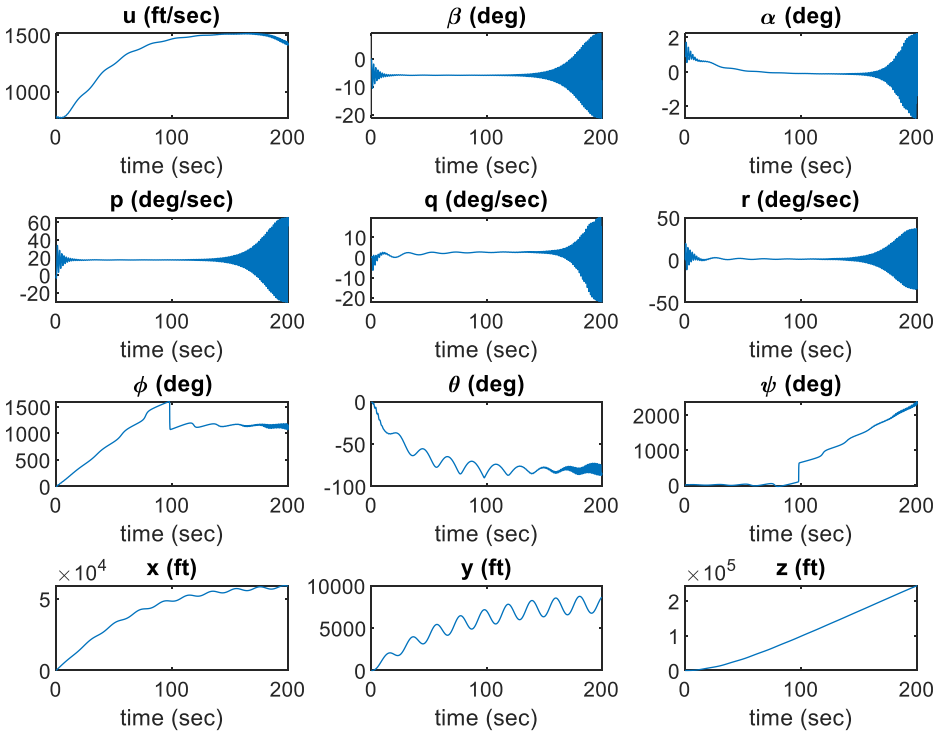
For +ve 5-degree rudder



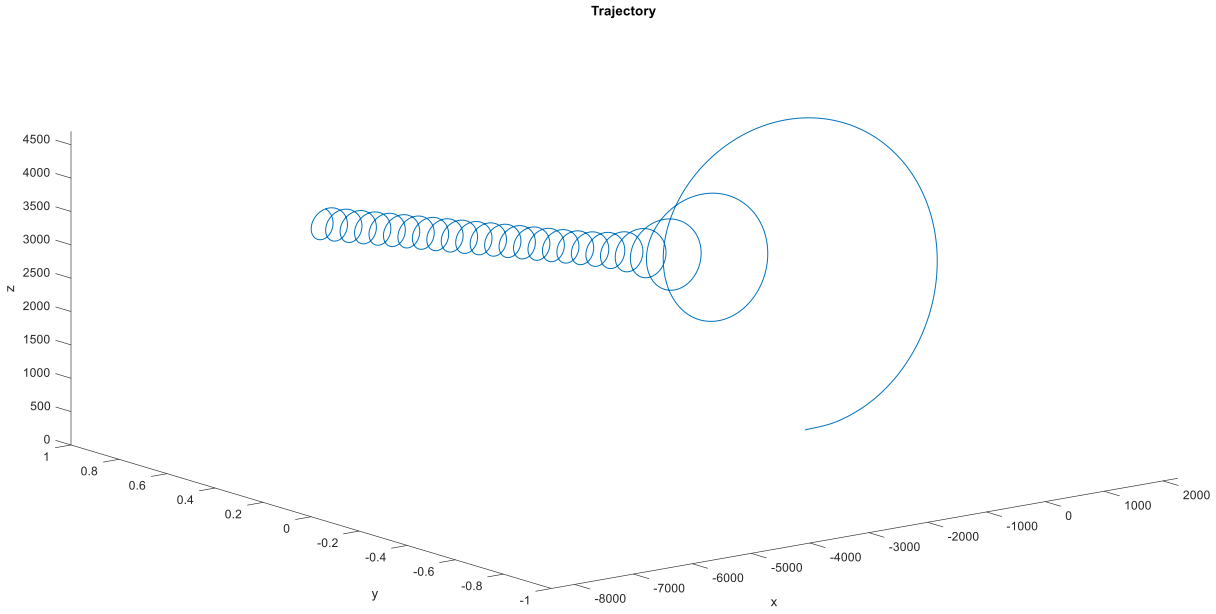


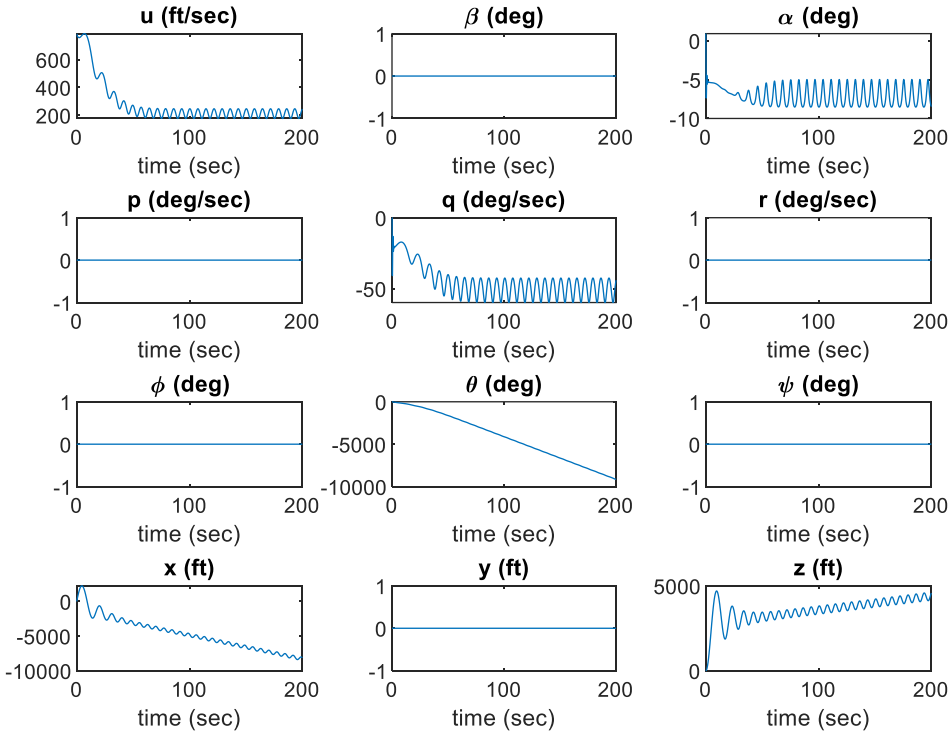
For -ve 5-degree rudder



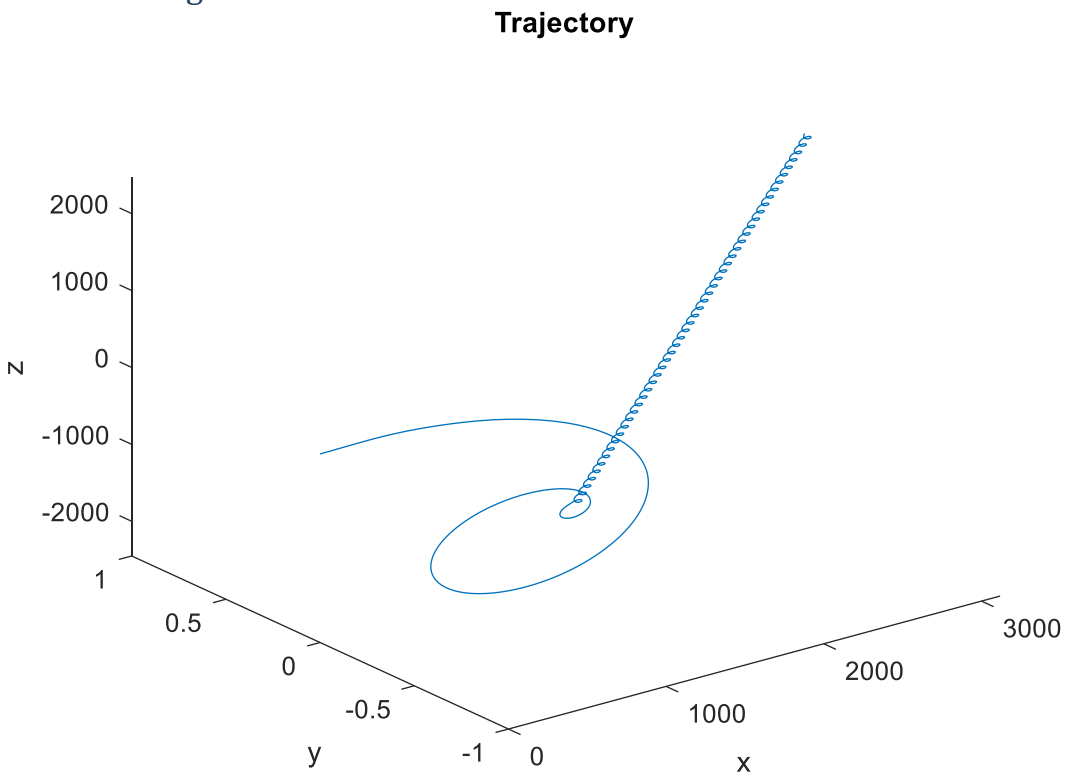


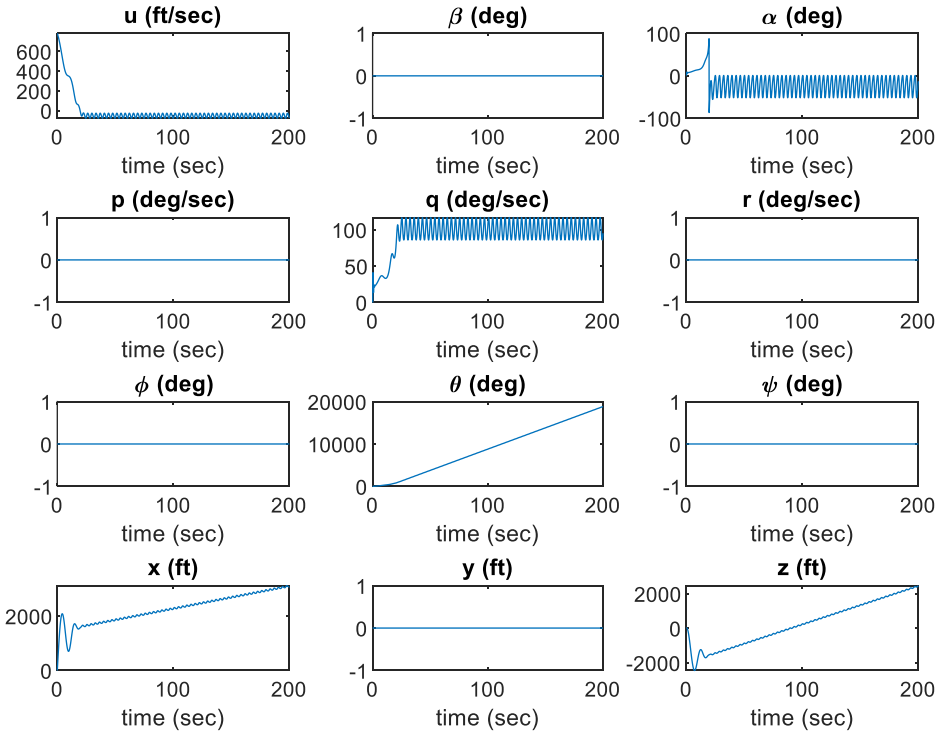
For +ve 5-degree elevator



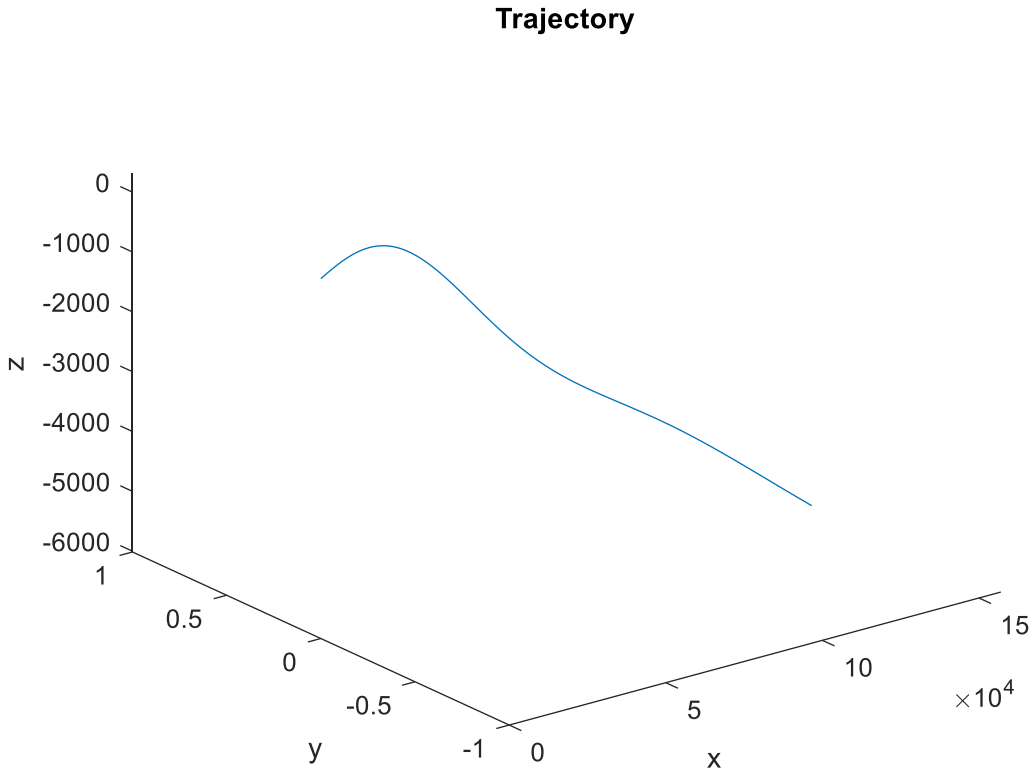


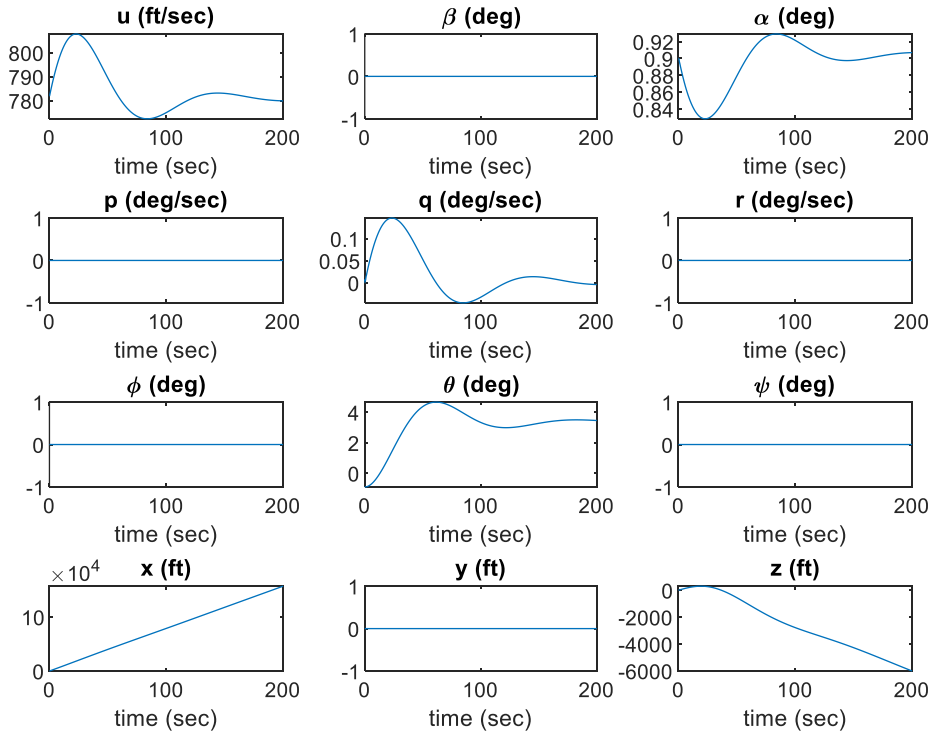
For -ve 5-degree elevator





For 1000 in thrust





For 10000 in thrust