```
clc;
```

state space model

```
%state matrices, linearized
% model of the pitch plane (longitudinal dynamics)
A = [-1.064 1; 290.26 0];
B = [-0.25; -331.40];
C = [-123.34 0; 0 1];
D = [-13.51; 0];

%Angle of attack(AoA), q--pitch angular rate
%delta_p-- fin deflection(control) in the pitch direction.
%Az--Normal accelaration (perpendicular to the velocity vector).
states = {'AoA', 'q'};
inputs = {'\delta_p'};
outputs = {'Az', 'q'};



sys = ss(A,B,C,D, 'statename',states,...
     'inputname', inputs,...
     'outputname', outputs);

%%Transfer function model
TFs = tf(sys); % It gives TF for Az and q separately, 2x1 array
TF = TFs(2,1); % TF for i/p (delta_p) to o/p (q)
              % TFs(1,1) for Az
disp(pole(TF)); %gives unstable pole (s= -17.57 and 16.51)
disp(rank(ctrb(A,B))); %2=n. so my system is controllable,
                       %hence, I can develop a controller
                       %to control the system.
```

LQR controller, u = -Kx

```
% weight matrices
Q = [0.1 0; 0 0.2];
R = 0.5;

%LQR optimum controller gain
[K,S,e] = lqr(A,B,Q,R);
fprintf('eigs of A-BK\n');
disp(eig(A-B*K));
fprintf('Feedback gain\n');
disp(K);

%Closed Loop system
Acl = A-B*K;
Bcl = B;

sys_cl = ss(Acl,Bcl,C,D, 'statename',states,...
    'inputname',inputs,...
    'outputname',outputs);
```

```
%Closed Loop TF
TF = tf(sys_cl);
TF_cl = TF(2,1);
pole(TF_cl);
```

State Estimator Design(LQG Kalman filter design)

```
G = eye(2);
H = 0*eye(2); % w doesn't come in o/p, so H=0.

%Kalman Q,R noise matrices
Qcov = diag(0.00015*ones(1,2));
Rcov = diag(0.55*ones(1,2));

% System with disturbance or noisy system
sys_kf = ss(A,[B G],C,[D H]);
% Obtain L&P. since N=0, assuming w and v are uncorrelated.
[Kest,L,P] = kalman(sys_kf,Qcov,Rcov,0);

%closed loop observer gain(kalman gain)
Aob = A-L*C;

%Observer eigs
fprintf('Observer eigs\n');
disp(eig(Aob)) % stable

% Noise time constants
dT1 = 0.75;
dT2 = 0.25;
```

Missile model parameters.

```
R = 6371e3; % Earth radius
vel = 1021.08; %Total missile velocity(m/sec), 3350 ft/sec.
m2f = 3.2808; %meters to feet

%Initial Location in deg and meters
LAT_INIT = 46.8267;
LONG_INIT = 8.8286;
ELEV_INIT = 3267; %Lmeters above sea level

%Obstacle Location in deg
LAT_OBST = 46.9383;
LONG_OBST = 8.7508;

%target location in deg and meters
LAT_TARGET= 46.9886;
LONG_TARGET= 8.7242;
ELEV_TARGET = 822; % m-MS

%convert above coordinates(init and target) values to radians
%Here I apply my own deg_rad function to convert deg to rad
l1 = deg_rad(LAT_INIT);
```

```
u1 = deg_rad(LONG_INIT);
l2 = deg_rad(LAT_TARGET);
u2 = deg_rad(LONG_TARGET);

delta_l = l2-l1;
delta_u = u2-u1;

%Haversine formula, distance between initial and target.
%inputs must be in radians
h = sin(delta_l/2)^2 + cos(l1)*cos(l2)*sin(delta_u)^2;
a = 2*h*atan2(sqrt(h),sqrt(1-h));
d = R*a; %Horizontal distance(in m).

%Initial range of missile
%calculate using pythagoran theorem
r = sqrt(d^2 +(ELEV_TARGET-ELEV_INIT)^2); %range in meters.

%Initial Azimuth(Yaw)
yaw_init = azimuth(LAT_INIT,LONG_INIT,LAT_TARGET,LONG_TARGET);
yaw = deg_rad(yaw_init);%yaw angle need to maintain.

%initial flight path angle(FPA_INIT)
dh = abs(ELEV_INIT-ELEV_TARGET);%vertical height
FPA_INIT = atan(dh/d);%rad
```