Table of Contents


```
clc; close all; clear all;

% State-space matrices
A = [-1.064  1;
      290.26 0];

B = [-0.25;
     -331.40];

C = [-123.34  0;
       0      1];

D = [13.51;
      0];

states  = {'AoA','q'};
inputs  = {'\delta_c'};
outputs = {'Az','q'};

% State-space system
sys = ss(A, B, C, D, ...
    'statename',  states, ...
    'inputname',  inputs, ...
    'outputname', outputs);

% Transfer functions
TFs = tf(sys);
TF  = TFs(2,1);
disp('Poles of TF (output 2 / input 1):');
disp(pole(TF));

% LQR weighting matrices
Q = [0.1 0;
     0   0.1];
R = 0.5;

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

% Closed-loop system
```

```
Acl = A - B*K;
Bcl = B;

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

% Closed-loop TF from input to second output
TFcl = tf(syscl);
TFc  = TFcl(2,1);

% LQG / Kalman filter design
G = eye(2);       % process noise distribution
H = 0*eye(2);     % measurement noise feedthrough (zero)

% Kalman Q,R noise covariances
Qbar = diag(0.00015*ones(1,2));  % process noise cov
Rbar = diag(0.55*ones(1,2));     % measurement noise cov

% Define noisy system: inputs = [control u; process noise w]
sys_n = ss(A, [B G], C, [D H]);

% Kalman filter
[kest, L, P] = kalman(sys_n, Qbar, Rbar, 0);

% Observer dynamics
Aob = A - L*C;
%display obs eignen
fprintf('Observer eigenvalues\n');
disp(eig(Aob));
```

```
Poles of TF (output 2 / input 1):
  -17.5773
   16.5133

Eigenvalues of A-BK:
   -2.5132
 -150.1351

Feedback gain K:
   -1.4290   -0.4563

Observer eigenvalues
  -18.2152
  -15.9376
```

Noise TC

```
dT1 = 0.75;
dT2 = 0.25;
```

Missile parameters

```
R = 6371e3; % earth radius
Vel =1021.08; %speed (m/s)
meters2feet = 3.2811; %meters to feet

%intital loc
LAT_INIT = 34.329;
LON_INIT = -119.4573;
ELEV_INIT = 10000; % m from see level

%target loc
LAT_TARGET = 34.6588;
LON_TARGET = -118.769745;
ELEV_TARGET = 795; %m from sea level
%Obstacle location between target
LAT_OBS = 34.61916;
LON_OBS = -118.8429;

d2r = pi/180; %convert to radians

%further conversion to radians

l1 = LAT_INIT*d2r;
u1 = LON_INIT*d2r;
l2 = LAT_TARGET*d2r;
u2 = LON_TARGET*d2r;


dl= l2-l1;
du= u2-u1;

%haversine formula

a = sin(dl/2)^2+ cos(l1)*cos(l2)*sin(du/2)^2;

c = 2*atan2(sqrt(a),sqrt(1-a));

d = R*c; %horizontal dis (m)

%range (Pythag theorem)


r = sqrt(d^2*(ELEV_TARGET-ELEV_INIT)^2);

%dubstarcting from due north
yaw_init = azimuth(LAT_INIT,LON_INIT,LAT_TARGET,LON_TARGET);
yaw = yaw_init*d2r;
```

inital flight path angle

```
dh = abs(ELEV_TARGET-ELEV_INIT);
FPA_INIT = atan(dh/d); %rads
```

Published with MATLAB® R2025a