Table of Contents


```
I = 3e-4;
m = 2;
xr = [10; -10; 0; 0; 0; 0];
```

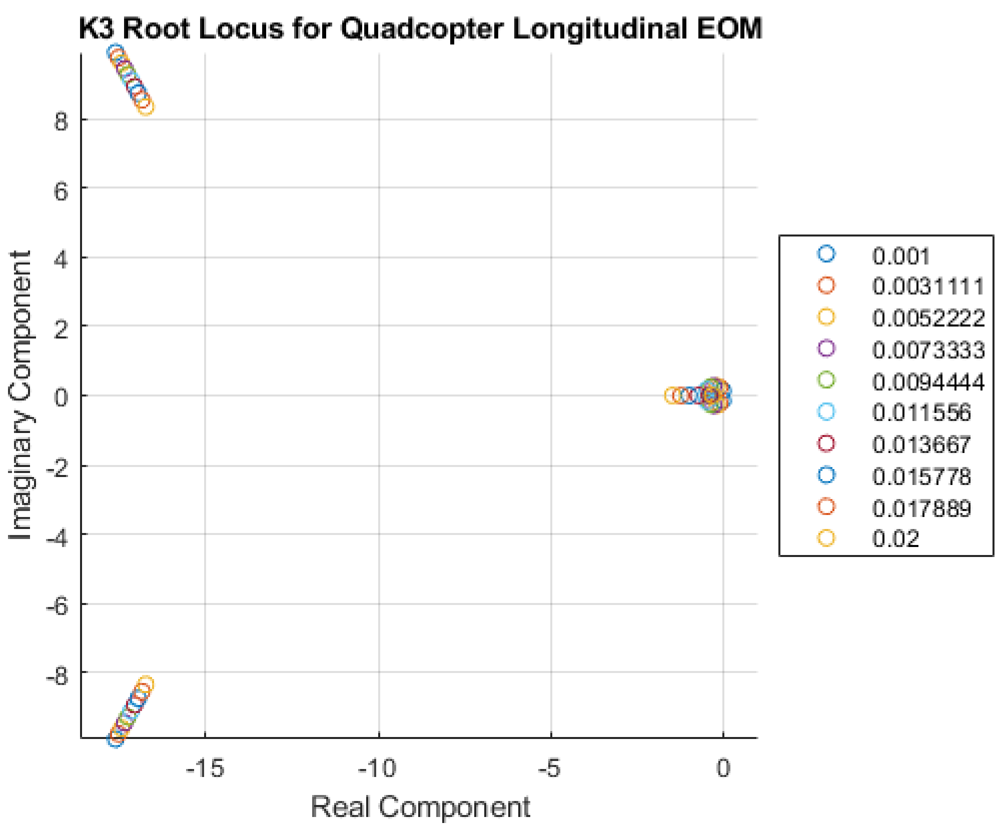
Task 1

```
%These gains were chosen using pole assignment derived from the eigenvalues
%of the 2x2 matrix containing only the delta_z and delta_w terms of our
%given EOM.  The equations used were k1 = (zeta*omega_n)*2*mass and
% k2 = ((omega_n*sqrt(1-zeta^2))^2 + k1^2/(4*mass^2))*mass.  The damping
% coefficient used was 0.7, and the period used was 0.5 seconds, with the
% natural frequency calculated as 2pi/period.

k5 = 17.5929;
k6 = 78.9568;
```

Task 2

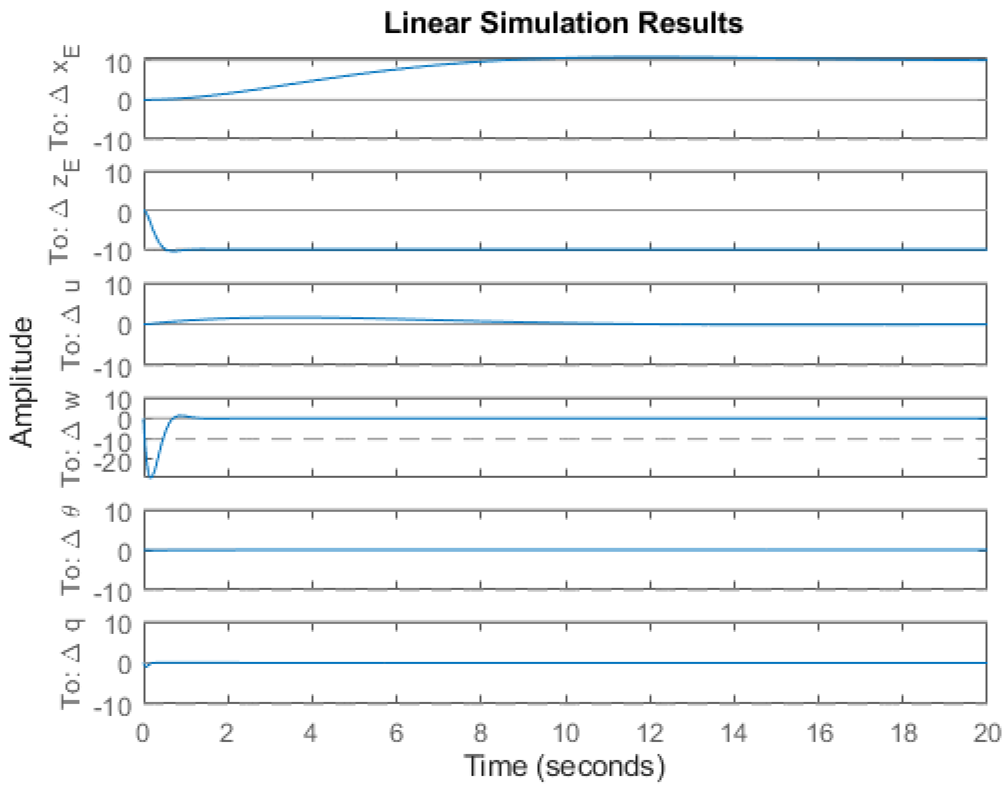
```
%The k1 and k2 gains were chosen using pole assignment derived from the
eigenvalues
%of the 2x2 matrix containing only the delta_q and delta_theta terms of our
%given EOM.  The equations used were k1 = (zeta*omega_n)*2*inertia and
% k2 = ((omega_n*sqrt(1-zeta^2))^2 + k1^2/(4*inertia^2))*inertia.  The
damping
% coefficient used was 0.87, and the period used was 0.155 seconds, with the
% natural frequency calculated as 2pi/period.

%k4 was chosen to be an order of magnitude faster than k1 and k2, using the
%equation k4 < 1/(((2*pi)/(zeta*omega_n))*10).  Since this equation is an
%inequality, the resulting value was reduced by 5%.

% a root locus was plotted for k3, which did not strongly resemble the root
% locus plot shown in class.  However, the values were all to the left of
% the imaginary axis, indicating stability of the system. A value of 0.0052
% was chosen, since its value on the real axis was furthest to the left.

k1 = 0.01058;
k2 = 0.12324;
```

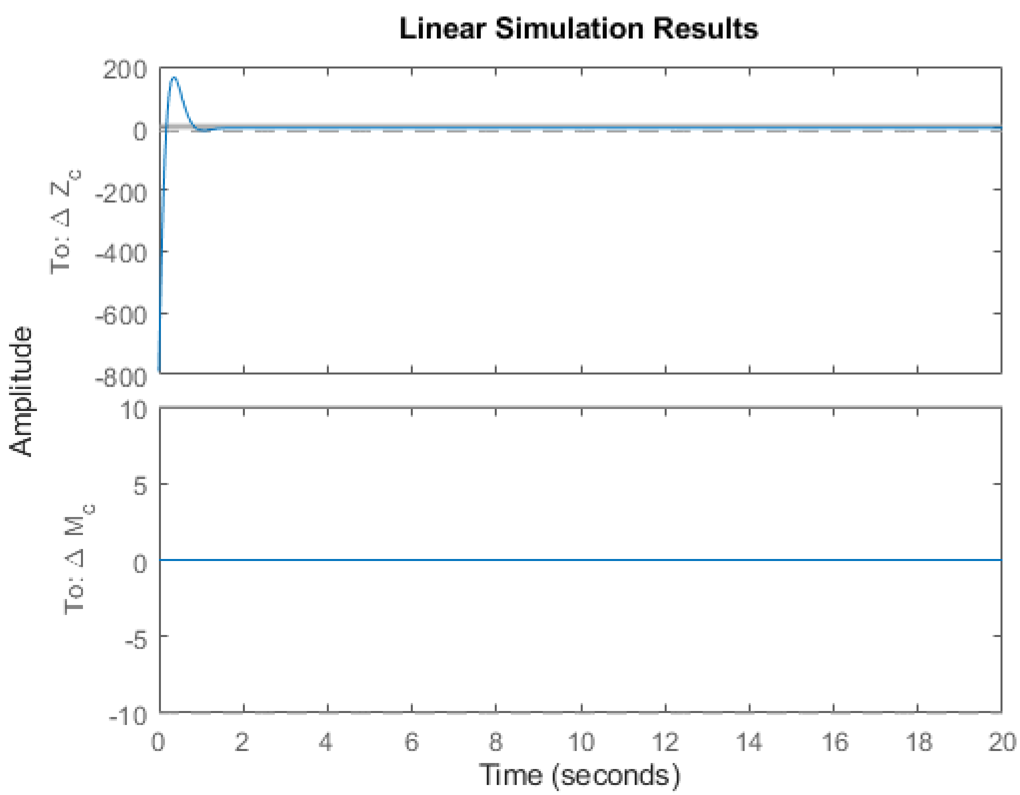
```
k4 = 0.26661;

% Plot root locus for k3 here
k3vec = linspace(0.001,0.02,10); %possible k3 values
figure(); grid on; hold on;
for i = 1:length(k3vec) %loop through possible k3 values
    k3 = k3vec(i);
    %define A matrix according to EOM and control law
    A = [0 1 0 0; 0 0 -9.81 0; 0 0 0 1; (k3*k4)/I k3/I -k2/I -k1/I];
    vals = eig(A);  %get eigenvalues
    scatter(real(vals),imag(vals))  %plot on complex plane
end
title("K3 Root Locus for Quadcopter Longitudinal EOM");
xlabel("Real Component");
ylabel("Imaginary Component");
legend(string(k3vec),'Location','eastoutside');
axis equal;
hold off;

k3 = 0.0052;

kvec = [k1;k2;k3;k4;k5;k6]; %put k-values in vector to feed into functions
```



Task 3

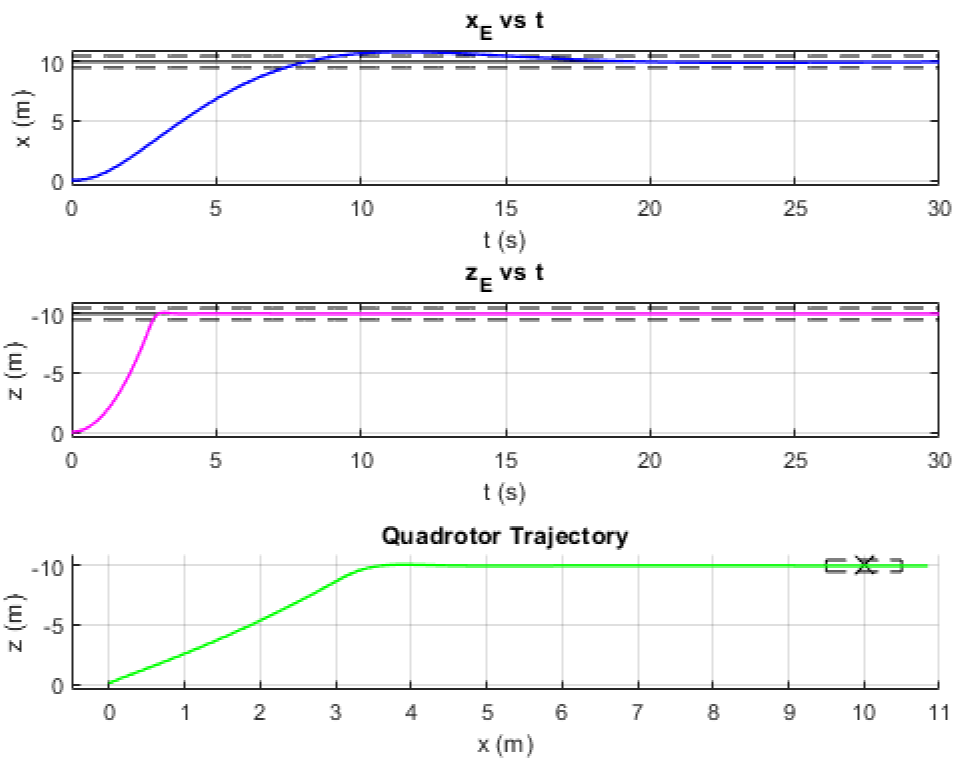
```
% modify the matrices below
Acl = [
    0 0 1 0 0 0;
    0 0 0 1 0 0;
    0 0 0 0 -9.81 0;
    0 -k6/m 0 -k5/m 0 0;
    0 0 0 0 0 1;
    (k3*k4)/I 0 k3/I 0 -k2/I -k1/I;
];
Bcl = [
    0 0 0 0 0 0;
    0 0 0 0 0 0;
    0 0 0 0 0 0;
    0 k6/m 0 0 0 0;
    0 0 0 0 0 0;
    -(k3*k4)/I 0 0 0 0 0;
];

figure(2)
T = 0:0.02:20;
ref = repmat(xr, 1, length(T));
sys = ss(Acl, Bcl, eye(6), zeros(6,6), 'OutputName', {'\Delta x_E', '\Delta
z_E', '\Delta u', '\Delta w', '\Delta \theta', '\Delta q'});
lsim(sys, ref, T) % outputs the state

figure(3)
C_control = [
    0 -k6 0 -k5 0 0;
    k3*k4 0 k3 0 -k2 -k1
];
D_control = [
    0 k6 0 0 0 0;
    -k3*k4 0 0 0 0 0
];
csys = ss(Acl, Bcl, C_control, D_control, 'OutputName', {'\Delta Z_c',
'\Delta M_c'});
lsim(csys, ref, T) % outputs the controls
```


Linear Simulation Results
Amplitude
To: Δ x_E
To: Δ z_E
To: Δ u
To: Δ w
To: Δ θ
To: Δ q
Time (seconds)


Linear Simulation Results
Amplitude
To: Δ Z_c
To: Δ M_c
Time (seconds)

Task 4

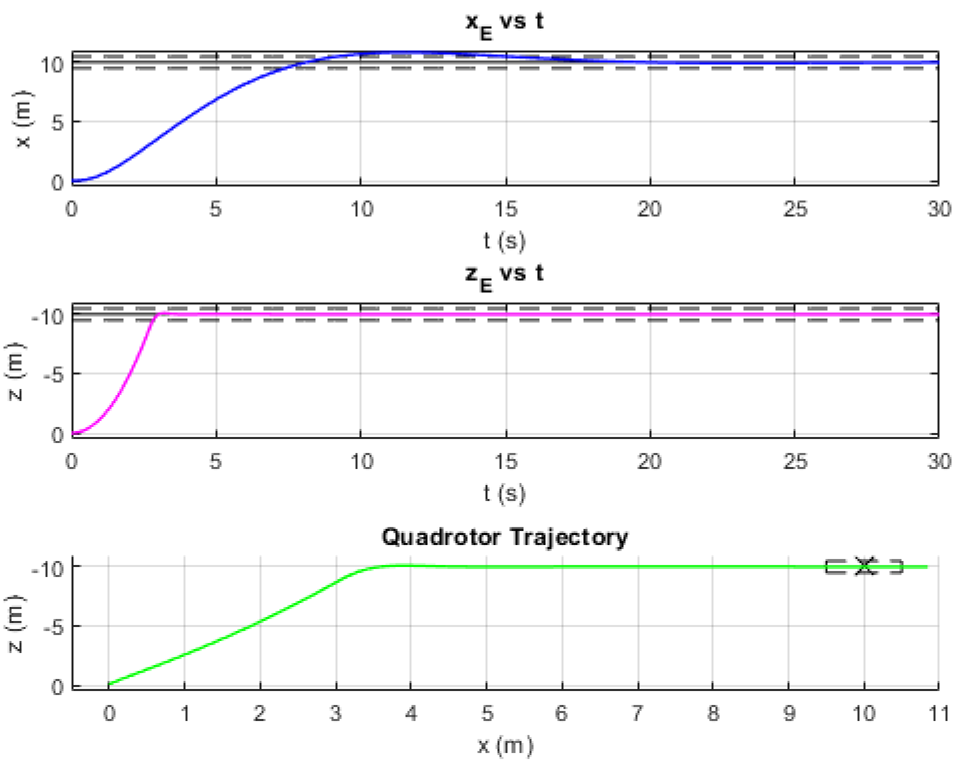
```
%the gain values were input into quadrotorLinearControls by modifying the
%input variables to take in a vector of k-values.  lsim takes in the Acl and
Bcl
% matrices derived from the linearized EOM, and outputs the dynamic response
% of the control system.  However, simulateQuadrotor uses ode45 to
numerically
% integrate the EOM, and the quadrotorDynamics function it uses for the EOM
%sets a maximum value on Z_c and M_c, the control force and moment.  This
%causes the result to be different from lsim's dynamic response.
figure(4)
[t, x] = simulateQuadrotor(@(t, x) quadrotorLinearControls(t, x, xr, kvec));
displayTrajectory(t, x)

%figure(5)
%animateQuadrotor(t, x)
```



Task 5

```
%The full nonlinear EOM for the system has sufficient cross-coupling that
%isolating the vertical and longitudinal dynamics as we have done here is
%impossible.  This task is thus to implement nonlinear controls.  However,
%review of class lectures and notes did not make clear to me how this is to
%be accomplished.  Z_c can incorporate a summation of all four rotors, so
```

```
%gain values can be applied to each one, and M_c can incorporate terms for
%the moment arms which are ignored in the linear model.  However, for a
%first approximation, the linear model was copied into the
%quadrotorControls function to get the report.m file to run completely.
%When it was discovered that this produced an adequate score to receive
%full credit on the assignment, no further work was performed, in order to
%prioritize more clearly expressed and quantified deliverables from other
%classes.
figure(6)
[t, x] = simulateQuadrotor(@(t, x) quadrotorControls(t, x, xr, kvec));
displayTrajectory(t, x)

% figure(7)
% animateQuadrotor(t, x)
```



Task 6

```
evaluate(@(t, x) quadrotorControls(t, x, xr, kvec),
'shane.billingsley@colorado.edu')

Target time: 20.000.
Your time: 15.020.

score =

  125.9000
```

```
Results saved to submission.json
Please submit this to gradescope!

ans =

    'submission.json'
```

Published with MATLAB® R2023b