Assignment 5

Table of Contents

Author: Samuel Razumovskiy

Date written: 10/3/19

Date modified: 10/10/19

Purpose: Numerically finding gain values for modeling quadcopter flight with a feed back loop

```
set(groot, 'defaulttextinterpreter', 'latex');
set(groot, 'defaultAxesTickLabelInterpreter', 'latex');
set(groot, 'defaultLegendInterpreter', 'latex');
```

Problem 1

```
clear,clc,close all

k3 = linspace(-10,10,1000);
k6 = linspace(-10,10,1000);

% Moments of inertia
Ix = 5.8e-5;
Iy = 7.2e-5;
Iz = 1e-4;

% Dominating Ratio
dr = 50;

% Zeta calculation
zeta = sqrt(1/(1-(dr-1)^2/(dr+1)^2));

% Omega calculation
om = 2/(zeta-sqrt(zeta^2-1));

% calculations for k1,k2,k4,k5
k1 = zeta*2*om*Ix;
k2 = om^2*Ix;
k4 = zeta*2*om*Iy;
k5 = om^2*Iy;
g= 9.81;
```

```
j = 1;
k = 1;

% Looks for k3 and k6 values that result in all negative eigen values
 and a
% settling time of less than 1.25
for i = 1:numel(k3)
    Ala = [-k1/Ix, -k2/Ix, -k2*k3(i)/Ix; 1, 0, 0; 0, g, 0];
    Alo = [-k4/Iy, -k5/Iy, -k5*k6(i)/Iy; 1, 0, 0; 0, -g, 0];
    if (norm(eig(Ala))>1.25) & (imag(eig(Ala)) == 0)
       lam1(:,j) = eig(Ala);
       if lam1(1,j) < 0 && lam1(2,j) < 0 && lam1(3,j) < 0
          poslam1(j) = k3(i);
          j = j+1;
       end
    end
    if (norm(eig(Alo))>1.25) & (imag(eig(Alo)) == 0)
       lam2(:,k) = eig(Alo);
       if lam2(1,k) < 0 && lam2(2,k) < 0 && lam2(3,k) < 0
          poslam2(k) = k6(i);
          k = k+1;
       end
    end
end

k3 = poslam1(1);
k6 = poslam2(1);

fprintf("k3 was found to be %1.3f\n",k3)
fprintf("k6 was found to be %1.3f\n",k6)
```

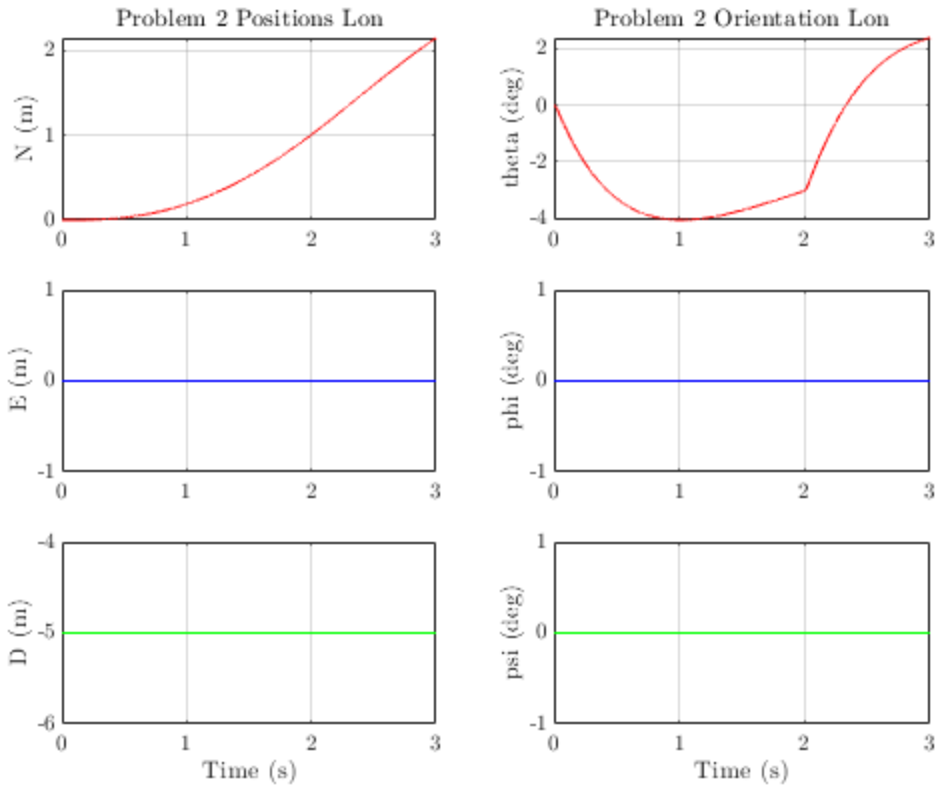
k3 was found to be 0.010
k6 was found to be -0.050

Problem 2

```
% Non changing values
mass = 0.068; % kg
g = 9.81; % m/s^2
weight = mass*g; % N B frame 3x1
radius = 0.060; % m
del_f1 = 0; % N B frame 3x1
del_VE_Er = [4;0;0]; % m/s E frame required velocity
I_B = [6.8e-5,0,0;0,9.2e-5,0;0,0,1.35e-4]; % kg/m^2 B frame 3x3
tspan = [0 3];

% Initial conditions for changing variables
pos_N = 0; pos_E = 0; pos_D = -5; % m E frame
velE_xB = 0; velE_yB = 0; velE_zB = 0; % m/s B frame
y_trans = [pos_N; pos_E; pos_D; velE_xB; velE_yB; velE_zB];

psi = 0; theta = 0; phi = 0; % rad
p = 0; q = 0; r = 0; % rad/s B frame
```

```
y_rot = [psi; theta; phi; p; q; r];

y = [y_trans;y_rot];
opt = odeset('maxstep',0.001);

% Calculating the ideal inertial u velocity required
for u = linspace(1.5,3)
    del_VE_Er = [u;0;0]; % m/s E frame required velocity
    [t,pos] =
 ode45(@(t,y)linearized_FB_quadcopter_ODE(t,y,mass,I_B,g,radius,del_VE_Er,k1,k2,k3
    index = find(t >= 2,1);
    if pos(index,1) >= 1
        break
    end
end

figure
subplot(3,2,1)
plot(t,pos(:,1),'r')
title("Problem 2 Positions Lon")
ylabel('N (m)')
grid on

subplot(3,2,3)
plot(t,pos(:,2),'b')
ylabel('E (m)')
grid on

subplot(3,2,5)
plot(t,pos(:,3),'g')
ylabel('D (m)')
xlabel('Time (s)')
grid on

subplot(3,2,2)
plot(t,pos(:,8).*180./pi,'r')
title("Problem 2 Orientation Lon")
ylabel('theta (deg)')
grid on

subplot(3,2,4)
plot(t,pos(:,9).*180./pi,'b')
ylabel('phi (deg)')
grid on

subplot(3,2,6)
plot(t,pos(:,7).*180./pi,'g')
ylabel('psi (deg)')
xlabel('Time (s)')
grid on

fprintf(['The model moves in the desired direction and reacts to the
 '...
    'change velocity required'])
```

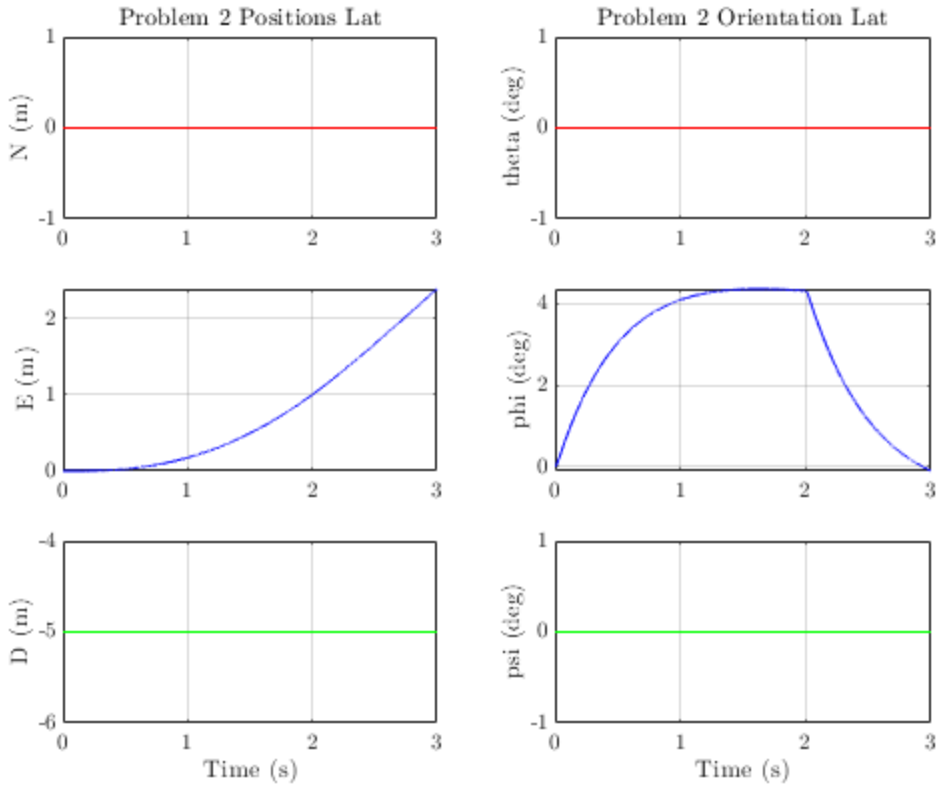
The model moves in the desired direction and reacts to the change velocity required




```
close all
% Calculating the ideal inertial v velocity required
for v = linspace(8,10)
    del_VE_Er = [0;v;0]; % m/s E frame required velocity
    [t,pos] =
 ode45(@(t,y)linearized_FB_quadcopter_ODE(t,y,mass,I_B,g,radius,del_VE_Er,k1,k2,k3
    index = find(t >= 2,1);
    if pos(index,2) >= 1
        break
    end
end
figure
subplot(3,2,1)
plot(t,pos(:,1),'r')
title("Problem 2 Positions Lat")
ylabel('N (m)')
grid on

subplot(3,2,3)
plot(t,pos(:,2),'b')
ylabel('E (m)')
grid on
```

```
subplot(3,2,5)
plot(t,pos(:,3),'g')
ylabel('D (m)')
xlabel('Time (s)')
grid on

subplot(3,2,2)
plot(t,pos(:,8).*180./pi,'r')
title("Problem 2 Orientation Lat")
ylabel('theta (deg)')
grid on

subplot(3,2,4)
plot(t,pos(:,9).*180./pi,'b')
ylabel('phi (deg)')
grid on

subplot(3,2,6)
plot(t,pos(:,7).*180./pi,'g')
ylabel('psi (deg)')
xlabel('Time (s)')
grid on

fprintf(['The model moves in the desired direction and reacts to the
 '...
    'change velocity required'])
```

```
The model moves in the desired direction and reacts to the change
 velocity required
```

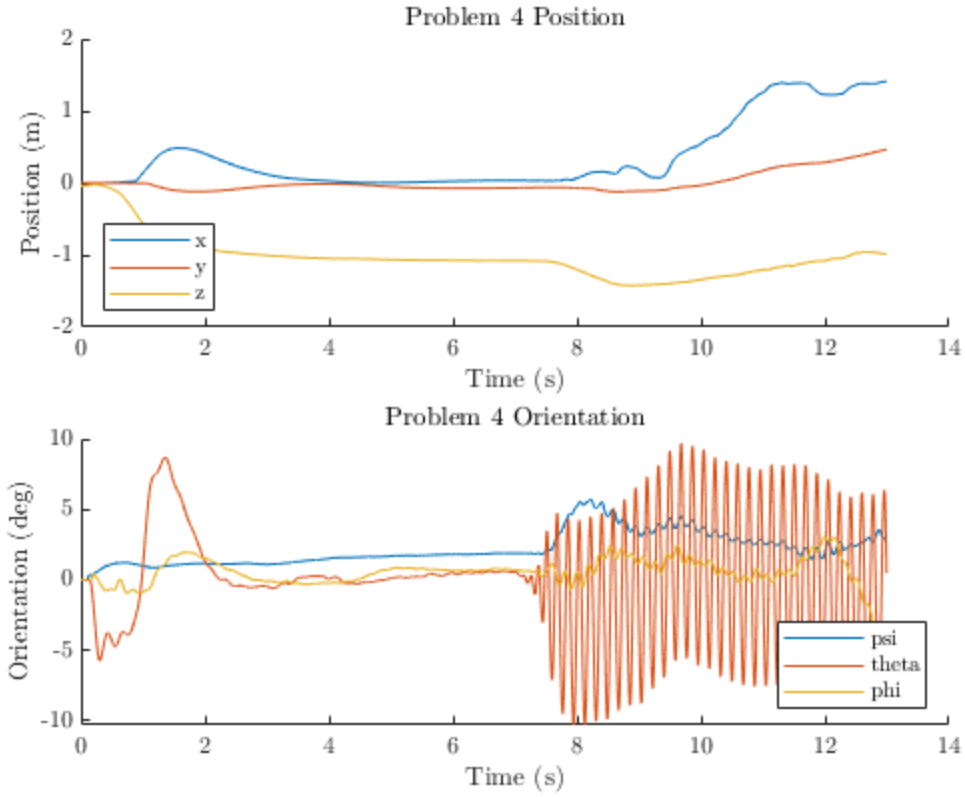


Problem 3

```
clear,clc,close all
load RSdata_two_1549

times = rt_estimatedStates.time(:);
xdata = rt_estimatedStates.signals.values(:,1);
ydata = rt_estimatedStates.signals.values(:,2);
zdata = rt_estimatedStates.signals.values(:,3);
psi = rt_estimatedStates.signals.values(:,4);
theta = rt_estimatedStates.signals.values(:,5);
phi = rt_estimatedStates.signals.values(:,6);

figure(18)
subplot(2,1,1)
hold on
plot(times,xdata)
plot(times,ydata)
plot(times,zdata)
xlabel('Time (s)')
ylabel('Position (m)')
title("Problem 4 Position")
legend('x','y','z','location','southwest')

subplot(2,1,2)
```

```
hold on
plot(times,psi.*180./pi)
plot(times,theta.*180./pi)
plot(times,phi.*180./pi)
xlabel('Time (s)')
ylabel('Orientation (deg)')
title("Problem 4 Orientation")
legend('psi','theta','phi','location','southeast')

fprintf(['The measured beahavior was likely caused by too high of '...
        'k1,k2,k4, and k5 values since the copter seems to be over
 '...
        'over rotating constantly, but it still tried to translate
 '...
        'albiet not very successfully. The high gains cause the '...
        'sinusoidal motions because of accuracy of the sensors on the
 '...
        'copter, since it does not have an instantious reactions to
 '...
        'disturbances it starts to over correct for them instead.'])
```

```
The measured beahavior was likely caused by too high of k1,k2,k4, and
 k5 values since the copter seems to be over over rotating constantly,
 but it still tried to translate albiet not very successfully. The
 high gains cause the sinusoidal motions because of accuracy of
 the sensors on the copter, since it does not have an instantious
 reactions to disturbances it starts to over correct for them instead.
```



Functions Called

The following functions were built and called as part of this assignment.

```
function [dydt] =
 linearized_FB_quadcopter_ODE(t,y,mass,I_B,g,radius,del_VE_Er,k1,k2,k3,k4,k5,k6)
% quadcopter_ODE This function is used to simulate trimmed flight of a
% quadcopter with simplified kinematics and dynamics.

% Position in E frame
del_pos_N = y(1); del_pos_E = y(2); del_pos_D = y(3); % m
% Velocity in B frame
del_u = y(4); del_v= y(5); del_w= y(6); % m/s
del_psi = y(7); del_theta = y(8); del_phi = y(9); % rad
del_p = y(10); del_q = y(11); del_r = y(12); % rad/s
del_VE_B = [del_u;del_v;del_w];

% Transfer matrix from B to E frame coordinates
L_EB =
 [cos(del_theta)*cos(del_psi),sin(del_phi)*sin(del_theta)*cos(del_psi)-
cos(del_phi)*sin(del_psi),...

 cos(del_phi)*sin(del_theta)*cos(del_psi)+sin(del_phi)*sin(del_psi);cos(del_theta)

 sin(del_phi)*sin(del_theta)*sin(del_psi)+cos(del_phi)*cos(del_psi),...
```

```
    cos(del_phi)*sin(del_theta)*sin(del_psi)-
sin(del_phi)*cos(del_psi);...
    -
sin(del_theta),sin(del_phi)*cos(del_theta),cos(del_phi)*cos(del_theta)];

% Velocity required in B frame calculated from E frame velocity
 required
if t >= 2
    del_VE_Er = [0;0;0];
end
del_VE_r = L_EB\del_VE_Er;
% Forces from rotors and aerodynamic drag
del_Xc = 0;
del_Yc = 0;
del_Zc = 0;
del_Ac = [del_Xc;del_Yc;del_Zc];

% Acceleration in B frame coordinates
del_u_d = -g*del_theta;
del_v_d = g*del_phi;
del_w_d = 1/mass*del_Ac(3);

% Velocity in E frame coordinates
VE_E = L_EB*del_VE_B;
vel_NE = VE_E(1);
vel_EE = VE_E(2);
vel_DE = 0;

% Finding the aerodynamic moments from rotors and drag
del_Lc = -k1*del_p + k2*(-del_phi+k3*(del_VE_r(2)-del_v));
del_Mc = -k4*del_q + k5*(-del_theta+k6*(del_VE_r(1)-del_u));
del_Nc = -0.004*del_r;
del_G_c = [del_Lc,del_Mc,del_Nc];

% Finding change in p,q,r
del_p_d = 1/I_B(1,1)*del_G_c(1);
del_q_d = 1/I_B(2,2)*del_G_c(2);
del_r_d = 1/I_B(3,3)*del_G_c(3);

% Finding changes in psi,theta,phi from p,q,r
del_psi_d = del_r;
del_theta_d = del_q;
del_phi_d = del_p;

yd_trans = [vel_NE; vel_EE; vel_DE; del_u_d; del_v_d; del_w_d];
yd_rot  = [del_psi_d; del_theta_d; del_phi_d; del_p_d; del_q_d;
 del_r_d];

dydt = [yd_trans;yd_rot];
end
```

Published with MATLAB® R2019b