HW #8

Submitted by Jesse Austin Stringfellow, Due Nov. 20, 2019

Contents

- Problem #1
- Problem 2 part a)
- Problem 2, Part b)
- Problem 2, Part c)

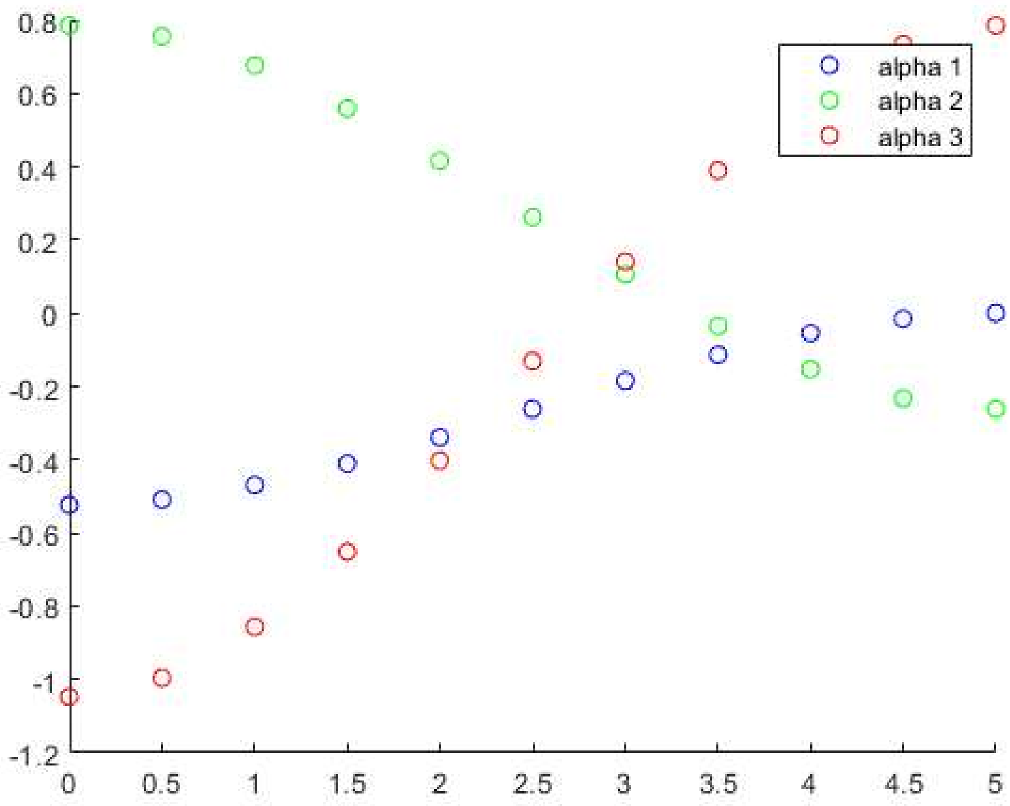
Problem #1

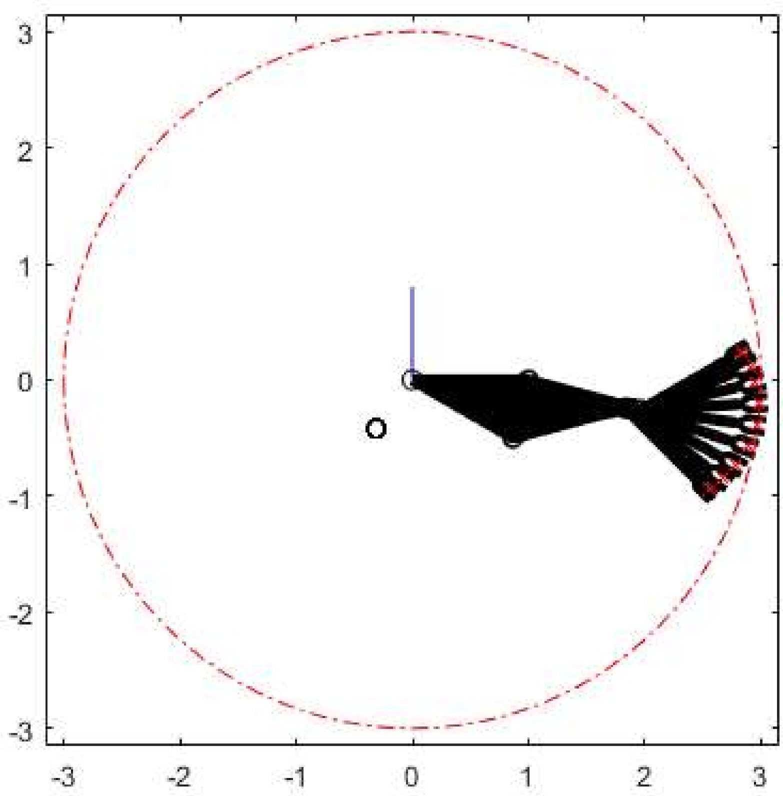
```
l1 = 1;
l2 = 1;
l3 = 1;
links = [l1,l2,l3];
alpha0 = [-pi/6;pi/4;-pi/3];
alpha5 = [0;-pi/12;pi/4];
tf = 5;
poly = ones(3,4);
for i = 1:3
    a0 = alpha0(i);
    a1 = 0;
    a2 = 3*(alpha5(i)-alpha0(i))/(tf^2);
    a3 = 2*(alpha0(i)-alpha5(i))/(tf^3);
    poly(i,:) = [a0 a1 a2 a3];
end
alpha1oft = 'The trajectory polynomial for alpha one is: \n alpha1(t) = %.3f+(%.0f*t)+(%.3f*t
^2)+(%.3f*t^3)';
alpha1oft = sprintf(alpha1oft,poly(1,1),poly(1,2),poly(1,3),poly(1,4))

alpha2oft = 'The trajectory polynomial for alpha two is: \n alpha2(t) = %.3f+(%.0f*t)+(%.3f*t
^2)+(%.3f*t^3)';
alpha2oft = sprintf(alpha2oft,poly(2,1),poly(2,2),poly(2,3),poly(2,4))

alpha3oft = 'The trajectory polynomial for alpha three is: \n alpha3(t) = %.3f+(%.0f*t)+(%.3f
*t^2)+(%.3f*t^3)';
alpha3oft = sprintf(alpha3oft,poly(3,1),poly(3,2),poly(3,3),poly(3,4))

tt = 0:.5:tf

trajalpha1 = polyval(flip(poly(1,:)), 0:.5:5)
trajalpha2 = polyval(flip(poly(2,:)), 0:.5:5)
trajalpha3 = polyval(flip(poly(3,:)), 0:.5:5)
figure(1)
scatter(tt,trajalpha1,'b')
hold on
scatter(tt,trajalpha2,'g')
scatter(tt,trajalpha3,'r')
legend({'alpha 1','alpha 2','alpha 3'})
hold off
alphas = [trajalpha1;trajalpha2;trajalpha3];
```

```
figure(2)
for i = 1:length(tt)
    planarR3_display(alphas(:,i)',links);
    hold on
end
```

```
alpha1oft =

    'The trajectory polynomial for alpha one is:
      alpha1(t) = -0.524+(0*t)+(0.063*t^2)+(-0.008*t^3)'


alpha2oft =

    'The trajectory polynomial for alpha two is:
      alpha2(t) = 0.785+(0*t)+(-0.126*t^2)+(0.017*t^3)'


alpha3oft =

    'The trajectory polynomial for alpha three is:
      alpha3(t) = -1.047+(0*t)+(0.220*t^2)+(-0.029*t^3)'


tt =

  Columns 1 through 7

         0    0.5000    1.0000    1.5000    2.0000    2.5000    3.0000

  Columns 8 through 11

    3.5000    4.0000    4.5000    5.0000


trajalpha1 =

  Columns 1 through 7

   -0.5236   -0.5089   -0.4691   -0.4105   -0.3393   -0.2618   -0.1843

  Columns 8 through 11

   -0.1131   -0.0545   -0.0147    0.0000


trajalpha2 =

  Columns 1 through 7

    0.7854    0.7561    0.6765    0.5592    0.4168    0.2618    0.1068

  Columns 8 through 11

   -0.0356   -0.1529   -0.2325   -0.2618
```

```
trajalpha3 =

  Columns 1 through 7

   -1.0472   -0.9959   -0.8566   -0.6514   -0.4021   -0.1309    0.1403

  Columns 8 through 11

    0.3896    0.5948    0.7341    0.7854

Warning: P-file C:\Users\Austin Stringfellow\Desktop\ECE 4560
2019\planarR3_display.p is older than MATLAB file C:\Users\Austin
Stringfellow\Desktop\ECE 4560 2019\planarR3_display.m.
C:\Users\Austin Stringfellow\Desktop\ECE 4560 2019\planarR3_display.p may be
obsolete and may need to be regenerated.
Type "help pcode" for information about generating P-files.
```





Problem 2 part a)

```
close all, clear all, clc
to = 0; % Initial time
tv = 2.5; % Intermediate point time
tf = 5; % Final time
pox = 0;poy = 0; % Initial pos
pvx = .1;pvy = .3; % Intermediate pos
pfx = -.4;pfy = .7; % Final pos
t1 = tv; t2 = tf -tv; % These are both 2.5 so...
t = t1;
A = [1 0 0 0 0 0 0 0; ...
    1 t t^2 t^3 0 0 0 0; ...
    0 0 0 0 1 0 0 0; ...
    0 0 0 0 1 t t^2 t^3; ...
    0 1 0 0 0 0 0 0; ...
    0 0 0 0 0 1 2*t 3*t^2; ...
    0 1 2*t 3*t^2 0 -1 0 0; ...
    0 0 2 6*t 0 0 -2 0]; % Per class
% A(t)*a_bar = p_bar ---> a_bar = (A^-1(t))*p_bar
invA = pinv(A) %Pseudoinverse per class
invA = invA(:,1:4) %creates a matrix of the first 4 columns since the ...
                    % last 4 are multiplied by zeros

a_bar_x = invA * [pox; pvx; pvx; pfx]
% This contains the a0:a3 and b0:b3 terms for the cubic poly describing the
% movement through x
a_bar_y = invA * [poy; pvy; pvy; pfy]
% This contains the a0:a3 and b0:b3 terms for the cubic poly describing the
```

```
% movement through y
```

```
invA =

  Columns 1 through 7

    1.0000    0.0000    0.0000   -0.0000   -0.0000    0.0000   -0.0000
    0.0000   -0.0000   -0.0000    0.0000    1.0000   -0.0000   -0.0000
   -0.3600    0.3600    0.1200   -0.1200   -0.7000    0.1000   -0.2000
    0.0800   -0.0800   -0.0480    0.0480    0.1200   -0.0400    0.0800
   -0.0000    0.0000    1.0000   -0.0000   -0.0000    0.0000   -0.0000
   -0.3000    0.3000   -0.3000    0.3000   -0.2500   -0.2500   -0.5000
    0.2400   -0.2400   -0.2400    0.2400    0.2000   -0.2000    0.4000
   -0.0480    0.0480    0.0800   -0.0800   -0.0400    0.1200   -0.0800

  Column 8

    0.0000
    0.0000
   -0.1250
    0.0500
    0.0000
    0.3125
   -0.2500
    0.0500


invA =

    1.0000    0.0000    0.0000   -0.0000
    0.0000   -0.0000   -0.0000    0.0000
   -0.3600    0.3600    0.1200   -0.1200
    0.0800   -0.0800   -0.0480    0.0480
   -0.0000    0.0000    1.0000   -0.0000
   -0.3000    0.3000   -0.3000    0.3000
    0.2400   -0.2400   -0.2400    0.2400
   -0.0480    0.0480    0.0800   -0.0800


a_bar_x =

    0.0000
   -0.0000
    0.0960
   -0.0320
    0.1000
   -0.1200
   -0.1440
    0.0448


a_bar_y =

    0.0000
   -0.0000
```

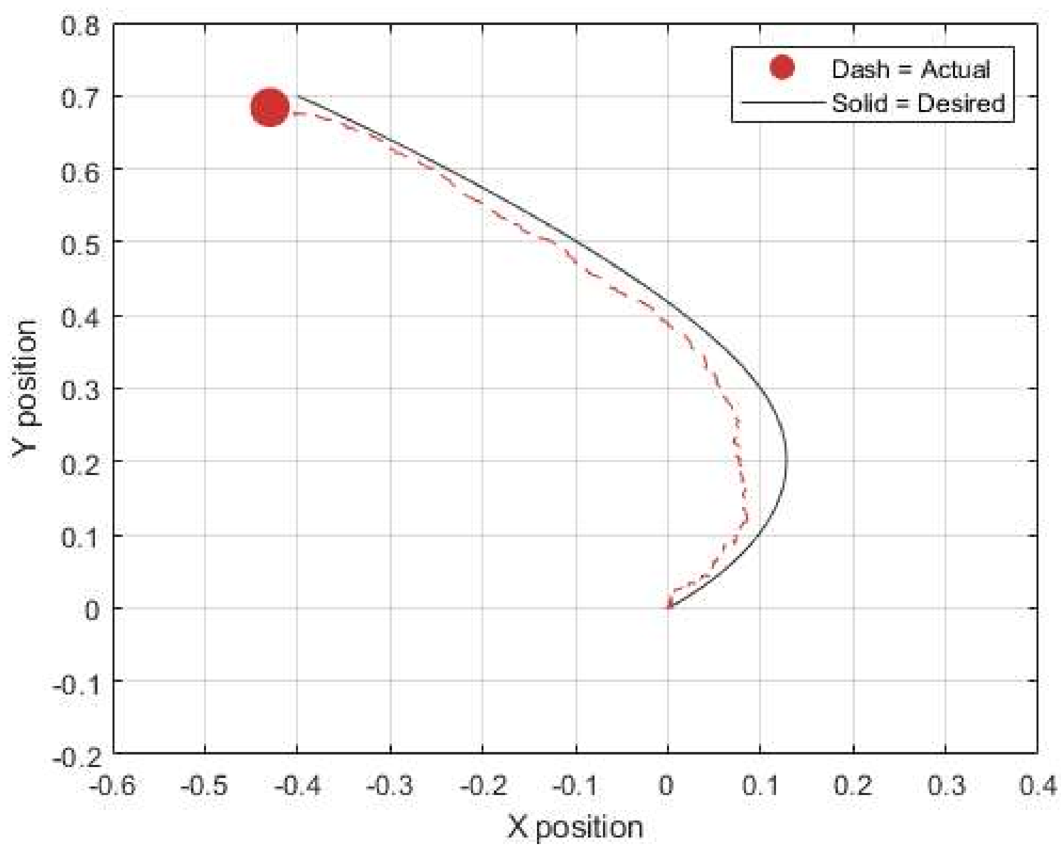
```
    0.0600
   -0.0048
    0.3000
    0.2100
    0.0240
   -0.0176
```

Problem 2, Part b)

CODE COPIED AND MODIFIED FROM CLASS CANVAS PAGE HW8 - Problem 2

```
dt = 0.005;tf = 5;T = 0:dt:tf;
% Noise switch set to 0/1
noisy = 1;
% Trajectory key points
x0 = 0;y0 = 0;xv = 0.1;yv = 0.3;xf = -0.4;yf = 0.7;
% Define trajectory and velocity - COMPLETE
tt1 = 0:dt:tf/2;
tt2 = tf/2:dt:tf;
px1 = (.096 .*tt1.^2) - (.032.*tt1.^3);
px2 = .1 - (.12.*tt1)-(.144.*tt1.^2) +(.0448.*tt1.^3);
px = [px1(1:end-1) px2];
py1 = (.06.*tt1.^2) -(.0048*tt1.^3);
py2 = .3+(.21.*tt1)+(.024.*tt1.^2) -(.0176.*tt1.^3);
py = [py1(1:end-1) py2];
% Open plot
figure, set(gcf, 'color', 'white'), hold on
% Initialize robot plot
rb_pl = plot(x0, y0, 'o', 'markersize',14, 'markerfacecolor',[0.8,0.2,0.2], 'markeredgecolor'
,'none');
% Plot trajectory
tr_pl = plot(px, py, '-', 'color',[0.2,0.2,0.2]);
grid on, box on
axis([-0.6 0.4 -0.2 0.8])
% Set robot's initial conditions
xt = x0;
yt = y0;
% Initialize empty vector to store robot's trajectory
actual_tr = zeros(2,size(T,1));
k = 0;
for t = T
    k = k+1;
    % Current velocity values - COMPLETE
    if t <= 2.5
        vx = (2*.096 *t) - (3*.032*t.^2);
        vy = (2*.06*t) -(3*.0048*t.^2);

    else
        t2 = t - 2.5;
        vx = (-.12)-(2*.144*t2) +(3*.0448*t2.^2);
        vy = (.21)+(2*.024*t2) -(3*.0176*t2.^2);
    end
    if noisy
        mu_x = 0.001*randn(1);
```

```
        mu_y = 0.001*randn(1);
    else
        mu_x = 0;
        mu_y = 0;
    end
% Unicycle dynamics
    xt = xt + dt*vx + mu_x;
    yt = yt + dt*vy + mu_y;
% Record actual robot's trajectory
    actual_tr(:,k) = [xt;yt];
% Update plot
    set(rb_pl, 'xdata', xt, 'ydata', yt)
    pause(0.01)
end
plot(actual_tr(1,:), actual_tr(2,:), '--', 'color',[0.8,0.2,0.2]);
xlabel('X position')
ylabel('Y position')
legend({'Dash = Actual','Solid = Desired'})
```



Problem 2, Part c)

Plot shows how the system responds when noise is added. With the noise added it can be seen that the robot does not stay on path as well. This could be fixed by adding sensors to the robot such as an IMU or using sensors like rotary encoders in the robot in order to collect data real-time that can be used in a feedback loop to provide better control.

Published with MATLAB® R2019b