Question 1a: After editing the Matrix Equations:

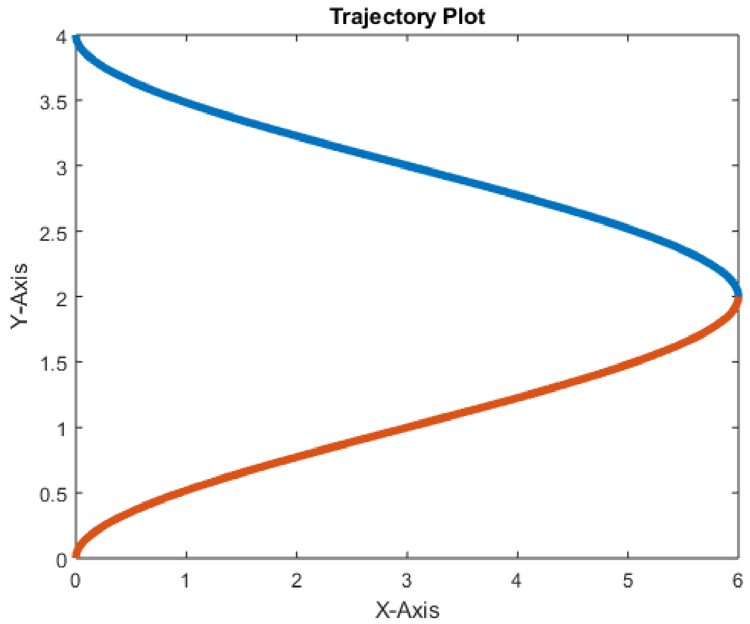
Assignment 2
Robot Control
- Aishwary Jagetia

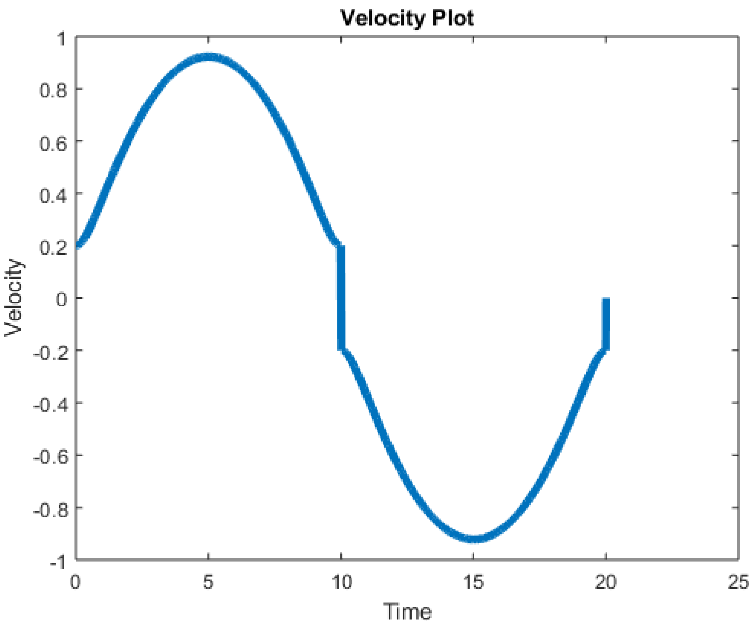
```
clear,clc;

% Initial and final condition (x, y, theta)
q0 = [0, 4, 0];
% q0 = [0, 5, 0];
% q0 = [-1, 4.5, 0];
v0 = 0;   %Velocity at q0 position is 0
qf = [6, 2, 0];
vf = 0;   %Velocity at qf position is 0
q1 = [0, 0, 0];
v1 = 0;   %Velocity at q1 position is 0
Tf = 10;                                % time to reach destination

syms t
a = sym('a', [1,4]); % the parameters of trajectory for x
b = sym('b', [1,4]); % the parameters of trajectory for y
basis = [1; t; t^2; t^3];
dbasis = [0; 1; 2*t; 3*t^2];
xsym = a*basis;
ysym = b*basis;
dx = a*dbasis;
dy = b*dbasis;

x0 = subs(xsym,t,0);
xf = subs(xsym,t,Tf);
dx0 = subs(dx,t,0);
dxf = subs(dx,t,Tf);

y0 = subs(ysym,t,0);
yf = subs(ysym,t,Tf);
dy0 = subs(dy,t,0);
dyf = subs(dy,t,Tf);

% compute the jacobian linearization of the vector field.
l=1;
syms v phi theta x y
f= [v*cos(theta); v*sin(theta); (v/l)*tan(phi)];
dfdx = jacobian(f,  [x;y;theta]);
dfdu = jacobian(f,  [v;phi]);

% solve linear equations for finding the coefficients in the feasible
% trajectories.

% initial and terminal condition: with velocity equals zero.
% [matA,matb] = equationsToMatrix([x0==q0(1), y0==q0(2), dx0*sin(q0(3))- dy0*cos(q0(3))==v0, ...
% xf==qf(1), yf==qf(2), dxf*sin(qf(3))- dyf*cos(qf(3))==vf],[a(1),a(2),a(3),a(4),b(1),b(2),b(3),b(4)]);
% param = matA\matb;
% avec = double(param(1:4)');
% bvec = double(param(5:end)');

[matA,matb] = equationsToMatrix([x0==q0(1), y0==q0(2), dx0*cos(q0(3))+ dy0*sin(q0(3))==v0, dx0*cos(q0(3))+ dy0*sin(q0(3))== v0, ...
xf==qf(1), yf==qf(2), dxf*cos(qf(3))+ dyf*sin(qf(3))==vf, dxf*cos(qf(3))+ dyf*sin(qf(3))==vf],[a(1),a(2),a(3),a(4),b(1),b(2),b(3),b(4)]);
param = matA\matb;
avec = double(param(1:4)');
bvec = double(param(5:end)');

% now apply the feedback controller
[Xdes1,X1, vf_1, phi_1] = ode_tracking(Tf,avec, bvec, 1);

% initial and terminal condition: with velocity equals zero.
% [matA,matb] = equationsToMatrix([x0==qf(1), y0==qf(2), dx0*sin(qf(3))- dy0*cos(qf(3))==v0, ...
% xf==q1(1), yf==q1(2), dxf*sin(q1(3))- dyf*cos(q1(3))==vf],[a(1),a(2),a(3),a(4),b(1),b(2),b(3),b(4)]);
% param = matA\matb;
% avec = double(param(1:4)');
% bvec = double(param(5:end)');

[matA,matb] = equationsToMatrix([x0==qf(1), y0==qf(2), dx0*cos(qf(3))+ dy0*sin(qf(3))==v0, dx0*cos(qf(3))+ dy0*sin(qf(3))== v0, ...
xf==q1(1), yf==q1(2), dxf*cos(q1(3))+ dyf*sin(q1(3))==vf, dxf*cos(q1(3))+ dyf*sin(q1(3))==vf],[a(1),a(2),a(3),a(4),b(1),b(2),b(3),b(4)]);
param = matA\matb;
avec = double(param(1:4)');
bvec = double(param(5:end)');

% now apply the feedback controller
[Xdes2,X2, vf_2, phi_2] = ode_tracking(Tf,avec, bvec, -1);

figure
plot(Xdes1(1,:), Xdes1(2,:), 'LineWidth', 4);
hold on
plot(Xdes2(1,:), Xdes2(2,:), 'LineWidth', 4);
hold on
title('Trajectory Plot');
```

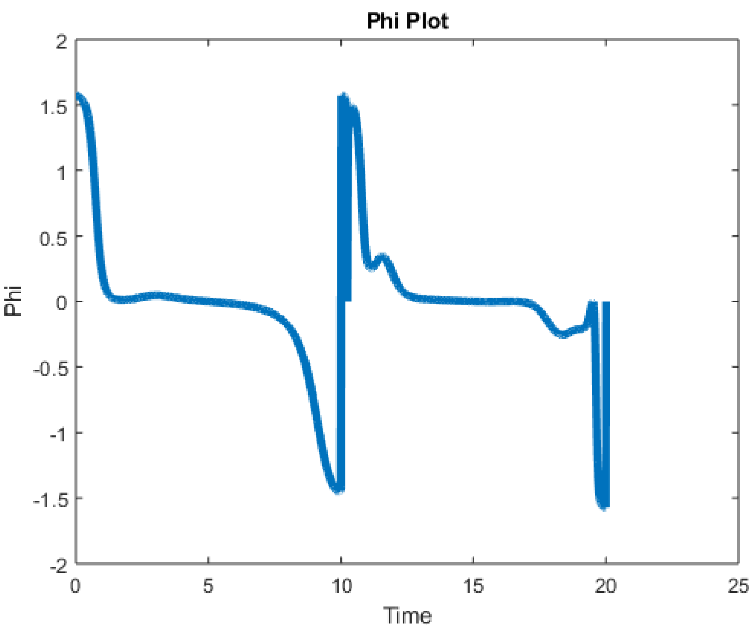
```
xlabel('X-Axis');
ylabel('Y-Axis');
hold off


dt=0.01;
t=[0:dt:(2*Tf)+dt];

figure
plot(t, [vf_1 vf_2], 'LineWidth', 4);
title('Velocity Plot');
xlabel('Time');
ylabel('Velocity');

figure
plot(t, [phi_1 phi_2], 'LineWidth', 4);
title('Phi Plot');
xlabel('Time');
ylabel('Phi');

function [Xdes, X, Vf, Phi] = ode_tracking(Tf,avec, bvec, s)
% evaluate the desired state.
dt=0.01;
tsteps=[0:dt:Tf];
N=size(tsteps,2);
X = zeros(3,N);
Vf = zeros(1,N);
Phi = zeros(1,N);
% with some initial error
% with no-initial error
X(:,1)=[0, 2, pi/6];

Xdes = zeros(3,N);

    for i=1:N-1
        xvec = X(:,i);
        x = xvec(1);
        y = xvec(2);
        theta = xvec(3);
        theta= wrapTo2Pi(theta);
        %theta = theta - 2*pi*floor(theta/(2*pi));

        l=1;
        t=tsteps(i);
        basis = [1; t; t^2; t^3];
        dbasis = [0; 1; 2*t; 3*t^2];
        ddbasis = [0; 0;2; 6*t];
        xdes = avec*basis;
        dxdes = avec*dbasis;
        ddxdes = avec*ddbasis;

        ydes = bvec*basis;
        dydes = bvec*dbasis;
        ddydes = bvec*ddbasis;

        % compute sin(theta_d)

        thetades = atan2(dydes, dxdes);
        Xdes(:,i)= [xdes;ydes;thetades];

        % The desired state.
        xdes_vec = [xdes; ydes; thetades];

        % compute the feedforward in the input.
        vf = s*(dxdes*cos(thetades) + dydes*sin(thetades));
        % dthetades = 1/vf*(ddydes*cos(thetades) - ddxdes*sin(thetades));
        dthetades = ((dxdes*ddydes) - (dydes*ddxdes))/((dxdes^2)+(dydes^2));
        wf = dthetades;
%         phi = atan2(l*thetades / vf);
        b = ((dxdes*ddydes) - (dydes*ddxdes))/(dxdes^2);
        c = 1/(vf*(sec(theta))^2);
        phi = atan2(b*c,1);
        Vf(:,i)= [vf];
        Phi(:,i)= [phi];
        % A =  [ 0, 0,  vf*cos(thetades);
        %  0, 0, -vf*sin(thetades);
        %  0, 0,              0];
        % B = [ sin(thetades),                      0;
        %  cos(thetades),                    0;
        %  tan(deltaf), vf*(tan(deltaf)^2 + 1)];


        A = [ 0, 0, -vf*sin(thetades);
```

```
            0, 0,  vf*cos(thetades);
            0, 0,             0];

        B = [ cos(thetades), 0;
            sin(thetades), 0;
            0, 1];

        Q= eye(3);
        R = eye(2);
        %if any(eig(A-B*K))>=0;
        K= lqr(A,B,Q,R);
        %end

        u = -K*(xvec - xdes_vec) + [vf; wf];

        dxvec = [u(1)*cos(theta);u(1)*sin(theta);u(2)];
        %
        % % without noise
         X(:,i+1)= dxvec*dt+ X(:,i);

        % with noise
        %X(:,i+1)= dxvec*dt+ X(:,i) +0.05*randn(1);
    end

    for i=1:N;
        t=tsteps(i);
        basis = [1; t; t^2; t^3];
        dbasis = [0; 1; 2*t; 3*t^2];
        ddbasis = [0; 0;2; 6*t];
        Xdes(1,i) = avec*basis;
        Xdes(2,i)= bvec*basis;
    end
end
```

```
Warning: Solution is not unique because the system is rank-deficient.
Warning: Solution is not unique because the system is rank-deficient.
```


Trajectory Plot
Y-Axis
X-Axis
0
0.5
1
1.5
2
2.5
3
3.5
4
0
1
2
3
4
5
6


Velocity Plot
Velocity
Time
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
5
10
15
20
25


Phi Plot
Phi
Time
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
5
10
15
20
25

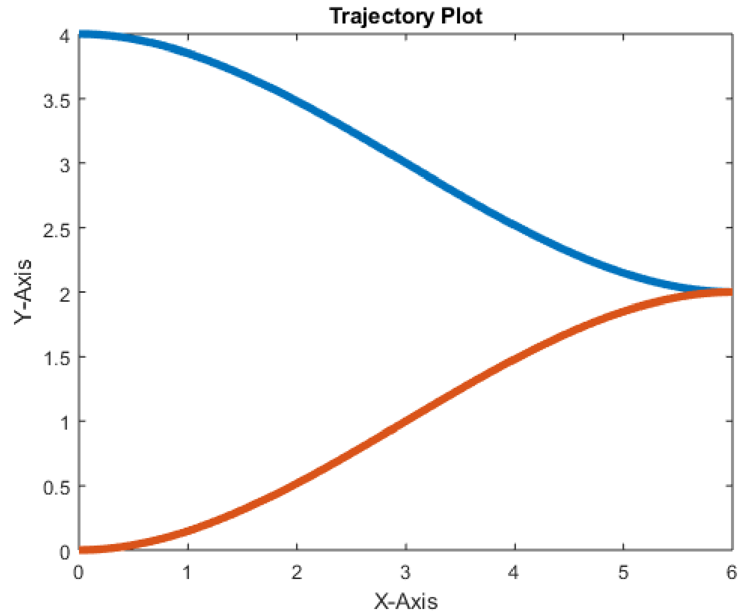
Question 1a: When changes in the Matrix Equations where not made

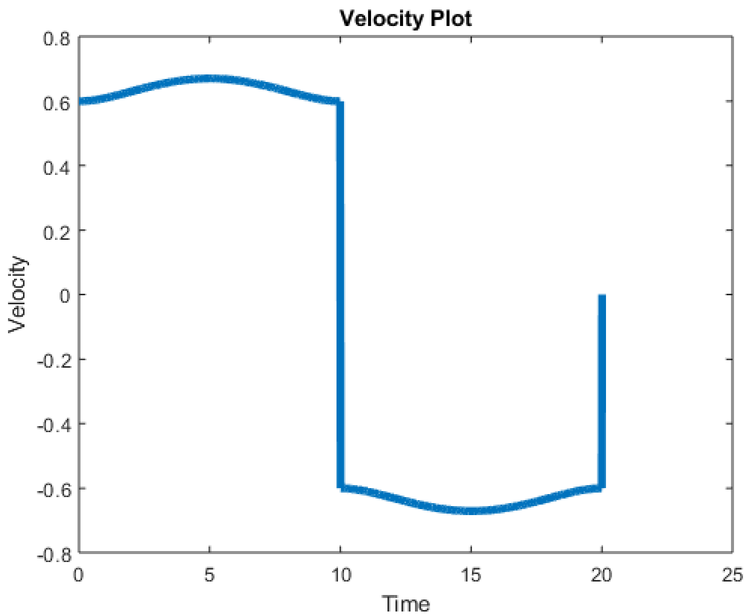
```
clear,clc;

% Initial and final condition (x, y, theta)
q0 = [0, 4, 0];
% q0 = [0, 5, 0];
% q0 = [-1, 4.5, 0];
v0 = 0;   %Velocity at q0 position is 0
qf = [6, 2, 0];
vf = 0;   %Velocity at qf position is 0
q1 = [0, 0, 0];
v1 = 0;   %Velocity at q1 position is 0
Tf = 10;                                % time to reach destination

syms t
a = sym('a', [1,4]); % the parameters of trajectory for x
b = sym('b', [1,4]); % the parameters of trajectory for y
basis = [1; t; t^2; t^3];
dbasis = [0; 1; 2*t; 3*t^2];
xsym = a*basis;
ysym = b*basis;
dx = a*dbasis;
dy = b*dbasis;

x0 = subs(xsym,t,0);
xf = subs(xsym,t,Tf);
dx0 = subs(dx,t,0);
dxf = subs(dx,t,Tf);

y0 = subs(ysym,t,0);
yf = subs(ysym,t,Tf);
dy0 = subs(dy,t,0);
dyf = subs(dy,t,Tf);

% compute the jacobian linearization of the vector field.
l=1;
syms v phi theta x y
f= [v*cos(theta); v*sin(theta); (v/l)*tan(phi)];
dfdx = jacobian(f,  [x;y;theta]);
dfdu = jacobian(f,  [v;phi]);

% solve linear equations for finding the coefficients in the feasible
% trajectories.

% initial and terminal condition: with velocity equals zero.
[matA,matb] = equationsToMatrix([x0==q0(1), y0==q0(2), dx0*sin(q0(3))- dy0*cos(q0(3))==v0, ...
xf==qf(1), yf==qf(2), dxf*sin(qf(3))- dyf*cos(qf(3))==vf],[a(1),a(2),a(3),a(4),b(1),b(2),b(3),b(4)]);
param = matA\matb;
avec = double(param(1:4)');
bvec = double(param(5:end)');

% [matA,matb] = equationsToMatrix([x0==q0(1), y0==q0(2), dx0*cos(q0(3))+ dy0*sin(q0(3))==v0, dx0*cos(q0(3))+ dy0*sin(q0(3))== v0, ...
% xf==qf(1), yf==qf(2), dxf*cos(qf(3))+ dyf*sin(qf(3))==vf, dxf*cos(qf(3))+ dyf*sin(qf(3))==vf],[a(1),a(2),a(3),a(4),b(1),b(2),b(3),b(4)]);
% param = matA\matb;
% avec = double(param(1:4)');
% bvec = double(param(5:end)');

% now apply the feedback controller
[Xdes1,X1, vf_1, phi_1] = ode_tracking(Tf,avec, bvec, 1);

% initial and terminal condition: with velocity equals zero.
[matA,matb] = equationsToMatrix([x0==qf(1), y0==qf(2), dx0*sin(qf(3))- dy0*cos(qf(3))==v0, ...
xf==q1(1), yf==q1(2), dxf*sin(q1(3))- dyf*cos(q1(3))==vf],[a(1),a(2),a(3),a(4),b(1),b(2),b(3),b(4)]);
param = matA\matb;
avec = double(param(1:4)');
bvec = double(param(5:end)');

% [matA,matb] = equationsToMatrix([x0==qf(1), y0==qf(2), dx0*cos(qf(3))+ dy0*sin(qf(3))==v0, dx0*cos(qf(3))+ dy0*sin(qf(3))== v0, ...
% xf==q1(1), yf==q1(2), dxf*cos(q1(3))+ dyf*sin(q1(3))==vf, dxf*cos(q1(3))+ dyf*sin(q1(3))==vf],[a(1),a(2),a(3),a(4),b(1),b(2),b(3),b(4)]);
% param = matA\matb;
% avec = double(param(1:4)');
% bvec = double(param(5:end)');

% now apply the feedback controller
[Xdes2,X2, vf_2, phi_2] = ode_tracking(Tf,avec, bvec, -1);

figure
plot(Xdes1(1,:), Xdes1(2,:), 'LineWidth', 4);
hold on
plot(Xdes2(1,:), Xdes2(2,:), 'LineWidth', 4);
hold on
title('Trajectory Plot');
xlabel('X-Axis');
```

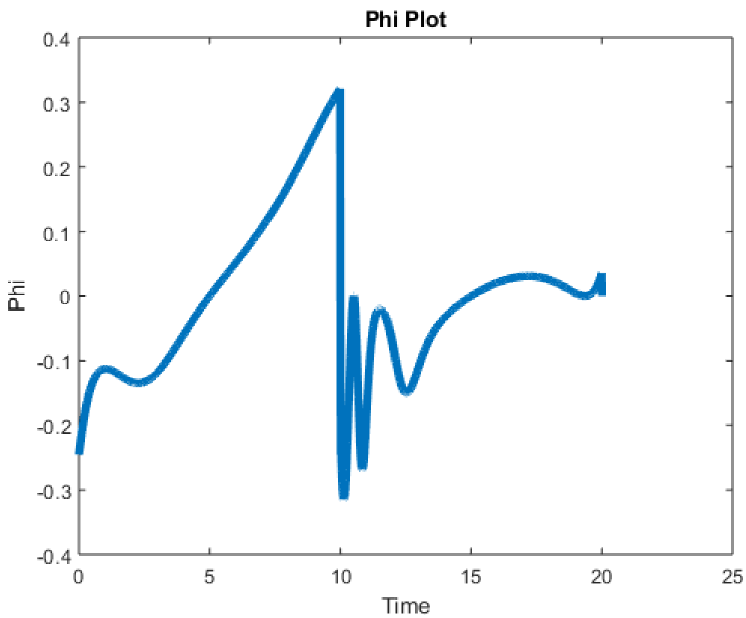
```
ylabel('Y-Axis');
hold off


dt=0.01;
t=[0:dt:(2*Tf)+dt];

figure
plot(t, [vf_1 vf_2], 'LineWidth', 4);
title('Velocity Plot');
xlabel('Time');
ylabel('Velocity');

figure
plot(t, [phi_1 phi_2], 'LineWidth', 4);
title('Phi Plot');
xlabel('Time');
ylabel('Phi');

function [Xdes, X, Vf, Phi] = ode_tracking(Tf,avec, bvec, s)
% evaluate the desired state.
dt=0.01;
tsteps=[0:dt:Tf];
N=size(tsteps,2);
X = zeros(3,N);
Vf = zeros(1,N);
Phi = zeros(1,N);
% with some initial error
% with no-initial error
X(:,1)=[0, 2, pi/6];

Xdes = zeros(3,N);

    for i=1:N-1
        xvec = X(:,i);
        x = xvec(1);
        y = xvec(2);
        theta = xvec(3);
        theta= wrapTo2Pi(theta);
        %theta = theta - 2*pi*floor(theta/(2*pi));

        l=1;
        t=tsteps(i);
        basis = [1; t; t^2; t^3];
        dbasis = [0; 1; 2*t; 3*t^2];
        ddbasis = [0; 0;2; 6*t];
        xdes = avec*basis;
        dxdes = avec*dbasis;
        ddxdes = avec*ddbasis;

        ydes = bvec*basis;
        dydes = bvec*dbasis;
        ddydes = bvec*ddbasis;

        % compute sin(theta_d)

        thetades = atan2(dydes, dxdes);
        Xdes(:,i)= [xdes;ydes;thetades];

        % The desired state.
        xdes_vec = [xdes; ydes; thetades];

        % compute the feedforward in the input.
        vf = s*(dxdes*cos(thetades) + dydes*sin(thetades));
        % dthetades = 1/vf*(ddydes*cos(thetades) - ddxdes*sin(thetades));
        dthetades = ((dxdes*ddydes) - (dydes*ddxdes))/((dxdes^2)+(dydes^2));
        wf = dthetades;
%         phi = atan2(l*thetades / vf);
        b = ((dxdes*ddydes) - (dydes*ddxdes))/(dxdes^2);
        c = 1/(vf*(sec(theta))^2);
        phi = atan2(b*c,1);
        Vf(:,i)= [vf];
        Phi(:,i)= [phi];
        % A =  [ 0, 0,  vf*cos(thetades);
        %  0, 0, -vf*sin(thetades);
        %  0, 0,             0];
        % B = [ sin(thetades),                    0;
        %  cos(thetades),                    0;
        %  tan(deltaf), vf*(tan(deltaf)^2 + 1)];


        A = [ 0, 0, -vf*sin(thetades);
            0, 0,  vf*cos(thetades);
            0, 0,             0];
```

```
        B = [ cos(thetades), 0;
            sin(thetades), 0;
            0, 1];

        Q= eye(3);
        R = eye(2);
        %if any(eig(A-B*K))>=0;
        K= lqr(A,B,Q,R);
        %end

        u = -K*(xvec - xdes_vec) + [vf; wf];

        dxvec = [u(1)*cos(theta);u(1)*sin(theta);u(2)];
        %
        % % without noise
         X(:,i+1)= dxvec*dt+ X(:,i);

        % with noise
        %X(:,i+1)= dxvec*dt+ X(:,i) +0.05*randn(1);
    end

    for i=1:N;
        t=tsteps(i);
        basis = [1; t; t^2; t^3];
        dbasis = [0; 1; 2*t; 3*t^2];
        ddbasis = [0; 0;2; 6*t];
        Xdes(1,i) = avec*basis;
        Xdes(2,i)= bvec*basis;
    end
end
```

```
Warning: Solution is not unique because the system is rank-deficient.
Warning: Solution is not unique because the system is rank-deficient.
```


Trajectory Plot
Y-Axis
X-Axis
4
3.5
3
2.5
2
1.5
1
0.5
0
0
1
2
3
4
5
6


Velocity Plot
Velocity
Time
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
5
10
15
20
25


Phi Plot
Phi
Time
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
0
5
10
15
20
25

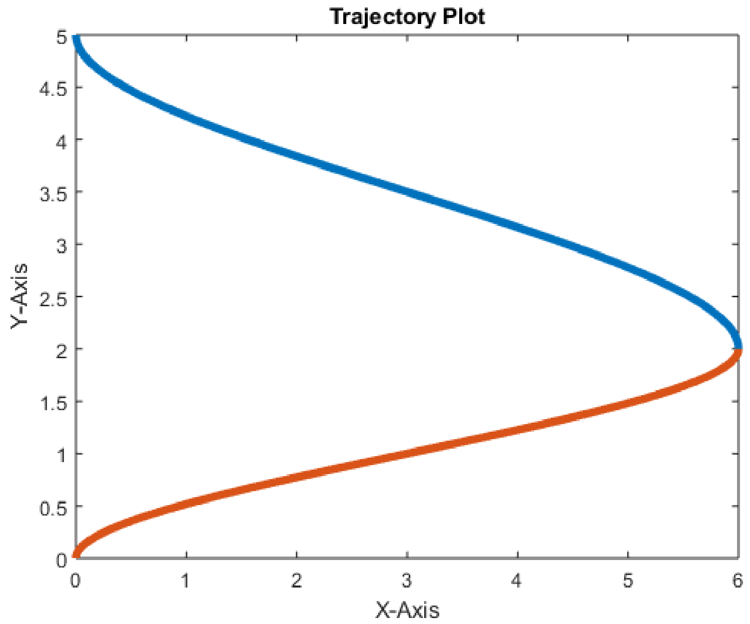
Question 1b: For Initial point [0, 5, 0]

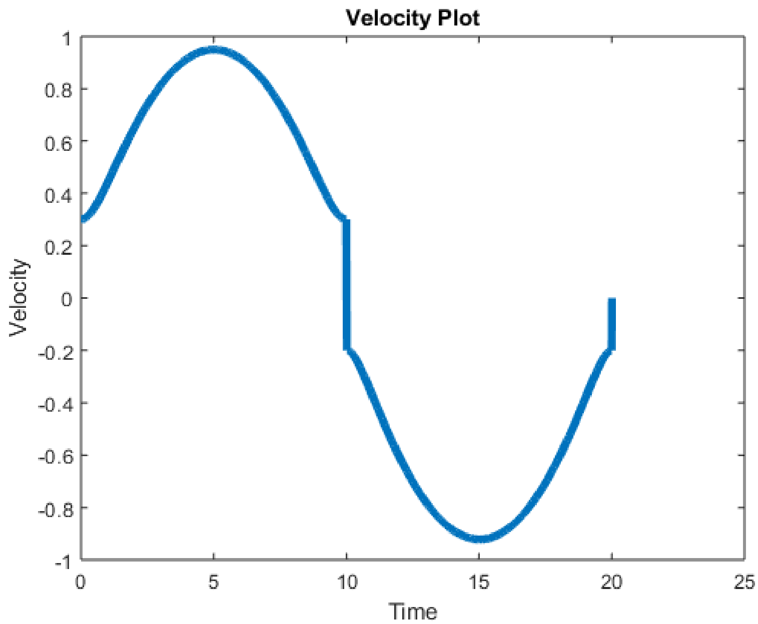
```
clear,clc;

% Initial and final condition (x, y, theta)
% q0 = [0, 4, 0];
q0 = [0, 5, 0];
% q0 = [-1, 4.5, 0];
v0 = 0;   %Velocity at q0 position is 0
qf = [6, 2, 0];
vf = 0;   %Velocity at qf position is 0
q1 = [0, 0, 0];
v1 = 0;   %Velocity at q1 position is 0
Tf = 10;                                % time to reach destination

syms t
a = sym('a', [1,4]); % the parameters of trajectory for x
b = sym('b', [1,4]); % the parameters of trajectory for y
basis = [1; t; t^2; t^3];
dbasis = [0; 1; 2*t; 3*t^2];
xsym = a*basis;
ysym = b*basis;
dx = a*dbasis;
dy = b*dbasis;

x0 = subs(xsym,t,0);
xf = subs(xsym,t,Tf);
dx0 = subs(dx,t,0);
dxf = subs(dx,t,Tf);

y0 = subs(ysym,t,0);
yf = subs(ysym,t,Tf);
dy0 = subs(dy,t,0);
dyf = subs(dy,t,Tf);

% compute the jacobian linearization of the vector field.
l=1;
syms v phi theta x y
f= [v*cos(theta); v*sin(theta); (v/l)*tan(phi)];
dfdx = jacobian(f,  [x;y;theta]);
dfdu = jacobian(f,  [v;phi]);

% solve linear equations for finding the coefficients in the feasible
% trajectories.

% initial and terminal condition: with velocity equals zero.
% [matA,matb] = equationsToMatrix([x0==q0(1), y0==q0(2), dx0*sin(q0(3))- dy0*cos(q0(3))==v0, ...
% xf==qf(1), yf==qf(2), dxf*sin(qf(3))- dyf*cos(qf(3))==vf],[a(1),a(2),a(3),a(4),b(1),b(2),b(3),b(4)]);
% param = matA\matb;
% avec = double(param(1:4)');
% bvec = double(param(5:end)');

[matA,matb] = equationsToMatrix([x0==q0(1), y0==q0(2), dx0*cos(q0(3))+ dy0*sin(q0(3))==v0, dx0*cos(q0(3))+ dy0*sin(q0(3))== v0, ...
xf==qf(1), yf==qf(2), dxf*cos(qf(3))+ dyf*sin(qf(3))==vf, dxf*cos(qf(3))+ dyf*sin(qf(3))==vf],[a(1),a(2),a(3),a(4),b(1),b(2),b(3),b(4)]);
param = matA\matb;
avec = double(param(1:4)');
bvec = double(param(5:end)');

% now apply the feedback controller
[Xdes1,X1, vf_1, phi_1] = ode_tracking(Tf,avec, bvec, 1);

% initial and terminal condition: with velocity equals zero.
% [matA,matb] = equationsToMatrix([x0==qf(1), y0==qf(2), dx0*sin(qf(3))- dy0*cos(qf(3))==v0, ...
% xf==q1(1), yf==q1(2), dxf*sin(q1(3))- dyf*cos(q1(3))==vf],[a(1),a(2),a(3),a(4),b(1),b(2),b(3),b(4)]);
% param = matA\matb;
% avec = double(param(1:4)');
% bvec = double(param(5:end)');

[matA,matb] = equationsToMatrix([x0==qf(1), y0==qf(2), dx0*cos(qf(3))+ dy0*sin(qf(3))==v0, dx0*cos(qf(3))+ dy0*sin(qf(3))== v0, ...
xf==q1(1), yf==q1(2), dxf*cos(q1(3))+ dyf*sin(q1(3))==vf, dxf*cos(q1(3))+ dyf*sin(q1(3))==vf],[a(1),a(2),a(3),a(4),b(1),b(2),b(3),b(4)]);
param = matA\matb;
avec = double(param(1:4)');
bvec = double(param(5:end)');

% now apply the feedback controller
[Xdes2,X2, vf_2, phi_2] = ode_tracking(Tf,avec, bvec, -1);

figure
plot(Xdes1(1,:), Xdes1(2,:), 'LineWidth', 4);
hold on
plot(Xdes2(1,:), Xdes2(2,:), 'LineWidth', 4);
hold on
title('Trajectory Plot');
```

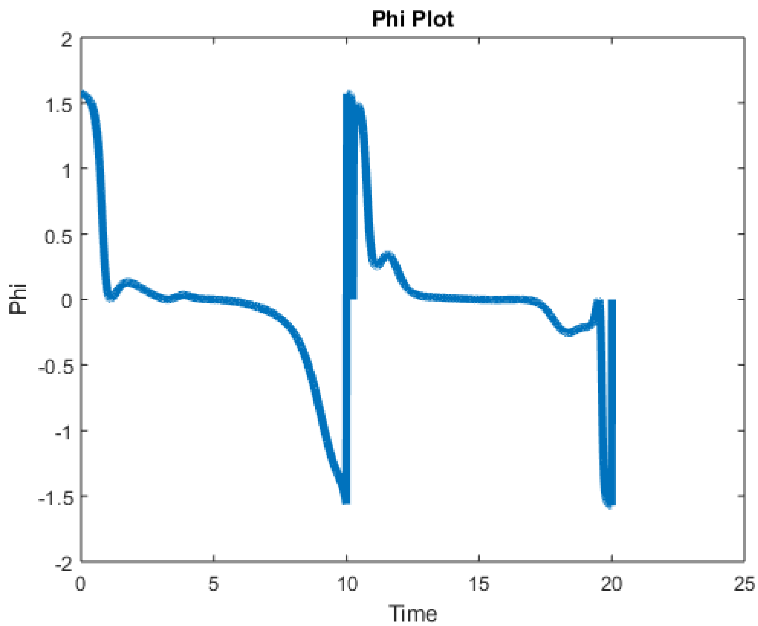
```
xlabel('X-Axis');
ylabel('Y-Axis');
hold off


dt=0.01;
t=[0:dt:(2*Tf)+dt];

figure
plot(t, [vf_1 vf_2], 'LineWidth', 4);
title('Velocity Plot');
xlabel('Time');
ylabel('Velocity');

figure
plot(t, [phi_1 phi_2], 'LineWidth', 4);
title('Phi Plot');
xlabel('Time');
ylabel('Phi');

function [Xdes, X, Vf, Phi] = ode_tracking(Tf,avec, bvec, s)
% evaluate the desired state.
dt=0.01;
tsteps=[0:dt:Tf];
N=size(tsteps,2);
X = zeros(3,N);
Vf = zeros(1,N);
Phi = zeros(1,N);
% with some initial error
% with no-initial error
X(:,1)=[0, 2, pi/6];

Xdes = zeros(3,N);

    for i=1:N-1
        xvec = X(:,i);
        x = xvec(1);
        y = xvec(2);
        theta = xvec(3);
        theta= wrapTo2Pi(theta);
        %theta = theta - 2*pi*floor(theta/(2*pi));

        l=1;
        t=tsteps(i);
        basis = [1; t; t^2; t^3];
        dbasis = [0; 1; 2*t; 3*t^2];
        ddbasis = [0; 0;2; 6*t];
        xdes = avec*basis;
        dxdes = avec*dbasis;
        ddxdes = avec*ddbasis;

        ydes = bvec*basis;
        dydes = bvec*dbasis;
        ddydes = bvec*ddbasis;

        % compute sin(theta_d)

        thetades = atan2(dydes, dxdes);
        Xdes(:,i)= [xdes;ydes;thetades];

        % The desired state.
        xdes_vec = [xdes; ydes; thetades];

        % compute the feedforward in the input.
        vf = s*(dxdes*cos(thetades) + dydes*sin(thetades));
        % dthetades = 1/vf*(ddydes*cos(thetades) - ddxdes*sin(thetades));
        dthetades = ((dxdes*ddydes) - (dydes*ddxdes))/((dxdes^2)+(dydes^2));
        wf = dthetades;
%         phi = atan2(l*thetades / vf);
        b = ((dxdes*ddydes) - (dydes*ddxdes))/(dxdes^2);
        c = 1/(vf*(sec(theta))^2);
        phi = atan2(b*c,1);
        Vf(:,i)= [vf];
        Phi(:,i)= [phi];
        % A =  [ 0, 0,  vf*cos(thetades);
        %  0, 0, -vf*sin(thetades);
        %  0, 0,              0];
        % B = [ sin(thetades),                     0;
        %  cos(thetades),                    0;
        %  tan(deltaf), vf*(tan(deltaf)^2 + 1)];


        A = [ 0, 0, -vf*sin(thetades);
```

```
            0, 0,  vf*cos(thetades);
            0, 0,             0];

        B = [ cos(thetades), 0;
            sin(thetades), 0;
            0, 1];

        Q= eye(3);
        R = eye(2);
        %if any(eig(A-B*K))>=0;
        K= lqr(A,B,Q,R);
        %end

        u = -K*(xvec - xdes_vec) + [vf; wf];

        dxvec = [u(1)*cos(theta);u(1)*sin(theta);u(2)];
        %
        % % without noise
         X(:,i+1)= dxvec*dt+ X(:,i);

        % with noise
        %X(:,i+1)= dxvec*dt+ X(:,i) +0.05*randn(1);
    end

    for i=1:N;
        t=tsteps(i);
        basis = [1; t; t^2; t^3];
        dbasis = [0; 1; 2*t; 3*t^2];
        ddbasis = [0; 0;2; 6*t];
        Xdes(1,i) = avec*basis;
        Xdes(2,i)= bvec*basis;
    end
end
```

```
Warning: Solution is not unique because the system is rank-deficient.
Warning: Solution is not unique because the system is rank-deficient.
```


Trajectory Plot
Y-Axis
X-Axis
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
0
1
2
3
4
5
6


Velocity Plot
Velocity
Time
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
5
10
15
20
25


Phi Plot
Phi
Time
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
5
10
15
20
25

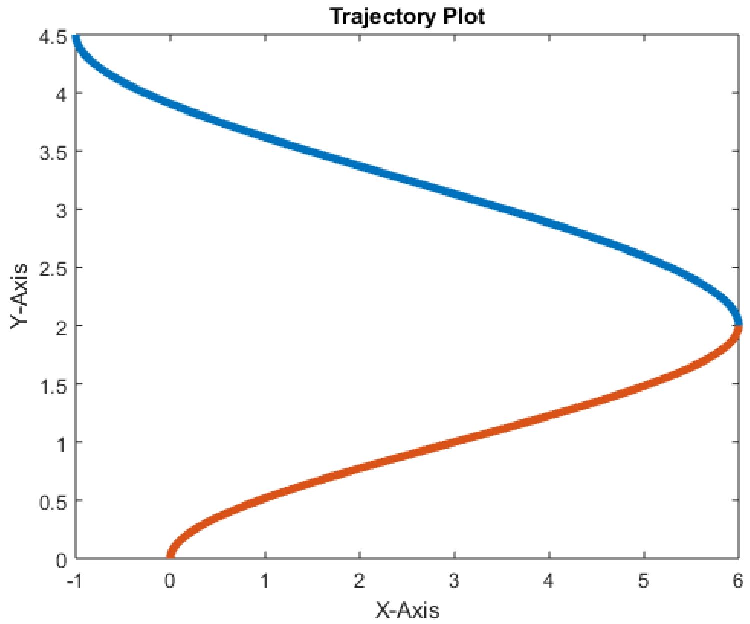
Question 1b: For initial position [-1, 4.5, 0]

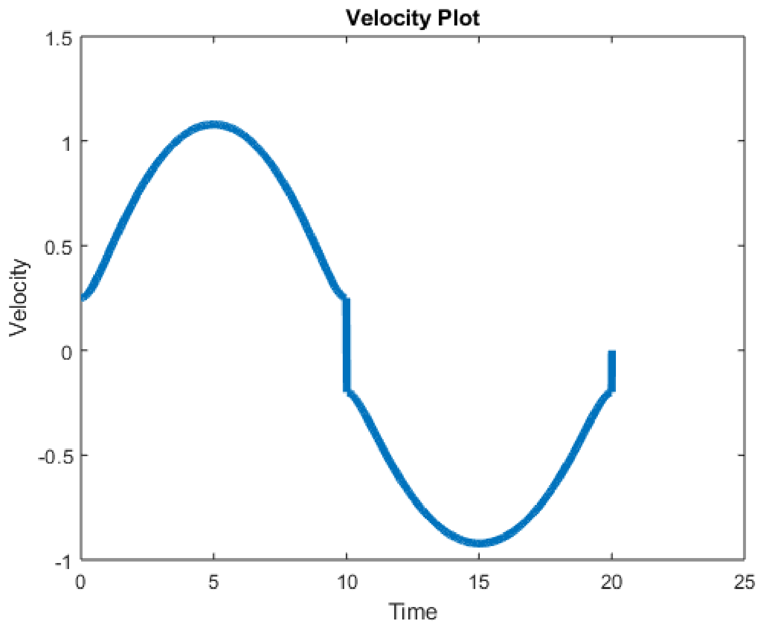
```
clear,clc;

% Initial and final condition (x, y, theta)
% q0 = [0, 4, 0];
% q0 = [0, 5, 0];
q0 = [-1, 4.5, 0];
v0 = 0;   %Velocity at q0 position is 0
qf = [6, 2, 0];
vf = 0;   %Velocity at qf position is 0
q1 = [0, 0, 0];
v1 = 0;   %Velocity at q1 position is 0
Tf = 10;                                % time to reach destination

syms t
a = sym('a', [1,4]); % the parameters of trajectory for x
b = sym('b', [1,4]); % the parameters of trajectory for y
basis = [1; t; t^2; t^3];
dbasis = [0; 1; 2*t; 3*t^2];
xsym = a*basis;
ysym = b*basis;
dx = a*dbasis;
dy = b*dbasis;

x0 = subs(xsym,t,0);
xf = subs(xsym,t,Tf);
dx0 = subs(dx,t,0);
dxf = subs(dx,t,Tf);

y0 = subs(ysym,t,0);
yf = subs(ysym,t,Tf);
dy0 = subs(dy,t,0);
dyf = subs(dy,t,Tf);

% compute the jacobian linearization of the vector field.
l=1;
syms v phi theta x y
f= [v*cos(theta); v*sin(theta); (v/l)*tan(phi)];
dfdx = jacobian(f,  [x;y;theta]);
dfdu = jacobian(f,  [v;phi]);

% solve linear equations for finding the coefficients in the feasible
% trajectories.

% initial and terminal condition: with velocity equals zero.
% [matA,matb] = equationsToMatrix([x0==q0(1), y0==q0(2), dx0*sin(q0(3))- dy0*cos(q0(3))==v0, ...
% xf==qf(1), yf==qf(2), dxf*sin(qf(3))- dyf*cos(qf(3))==vf],[a(1),a(2),a(3),a(4),b(1),b(2),b(3),b(4)]);
% param = matA\matb;
% avec = double(param(1:4)');
% bvec = double(param(5:end)');

[matA,matb] = equationsToMatrix([x0==q0(1), y0==q0(2), dx0*cos(q0(3))+ dy0*sin(q0(3))==v0, dx0*cos(q0(3))+ dy0*sin(q0(3))== v0, ...
xf==qf(1), yf==qf(2), dxf*cos(qf(3))+ dyf*sin(qf(3))==vf, dxf*cos(qf(3))+ dyf*sin(qf(3))==vf],[a(1),a(2),a(3),a(4),b(1),b(2),b(3),b(4)]);
param = matA\matb;
avec = double(param(1:4)');
bvec = double(param(5:end)');

% now apply the feedback controller
[Xdes1,X1, vf_1, phi_1] = ode_tracking(Tf,avec, bvec, 1);

% initial and terminal condition: with velocity equals zero.
% [matA,matb] = equationsToMatrix([x0==qf(1), y0==qf(2), dx0*sin(qf(3))- dy0*cos(qf(3))==v0, ...
% xf==q1(1), yf==q1(2), dxf*sin(q1(3))- dyf*cos(q1(3))==vf],[a(1),a(2),a(3),a(4),b(1),b(2),b(3),b(4)]);
% param = matA\matb;
% avec = double(param(1:4)');
% bvec = double(param(5:end)');

[matA,matb] = equationsToMatrix([x0==qf(1), y0==qf(2), dx0*cos(qf(3))+ dy0*sin(qf(3))==v0, dx0*cos(qf(3))+ dy0*sin(qf(3))== v0, ...
xf==q1(1), yf==q1(2), dxf*cos(q1(3))+ dyf*sin(q1(3))==vf, dxf*cos(q1(3))+ dyf*sin(q1(3))==vf],[a(1),a(2),a(3),a(4),b(1),b(2),b(3),b(4)]);
param = matA\matb;
avec = double(param(1:4)');
bvec = double(param(5:end)');

% now apply the feedback controller
[Xdes2,X2, vf_2, phi_2] = ode_tracking(Tf,avec, bvec, -1);

figure
plot(Xdes1(1,:), Xdes1(2,:), 'LineWidth', 4);
hold on
plot(Xdes2(1,:), Xdes2(2,:), 'LineWidth', 4);
hold on
title('Trajectory Plot');
```

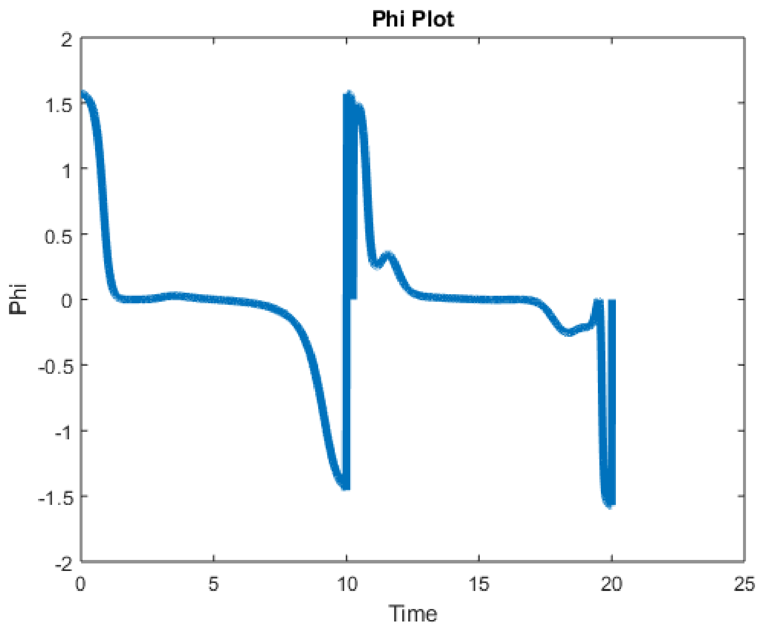
```
xlabel('X-Axis');
ylabel('Y-Axis');
hold off


dt=0.01;
t=[0:dt:(2*Tf)+dt];

figure
plot(t, [vf_1 vf_2], 'LineWidth', 4);
title('Velocity Plot');
xlabel('Time');
ylabel('Velocity');

figure
plot(t, [phi_1 phi_2], 'LineWidth', 4);
title('Phi Plot');
xlabel('Time');
ylabel('Phi');

function [Xdes, X, Vf, Phi] = ode_tracking(Tf,avec, bvec, s)
% evaluate the desired state.
dt=0.01;
tsteps=[0:dt:Tf];
N=size(tsteps,2);
X = zeros(3,N);
Vf = zeros(1,N);
Phi = zeros(1,N);
% with some initial error
% with no-initial error
X(:,1)=[0, 2, pi/6];

Xdes = zeros(3,N);

    for i=1:N-1
        xvec = X(:,i);
        x = xvec(1);
        y = xvec(2);
        theta = xvec(3);
        theta= wrapTo2Pi(theta);
        %theta = theta - 2*pi*floor(theta/(2*pi));

        l=1;
        t=tsteps(i);
        basis = [1; t; t^2; t^3];
        dbasis = [0; 1; 2*t; 3*t^2];
        ddbasis = [0; 0;2; 6*t];
        xdes = avec*basis;
        dxdes = avec*dbasis;
        ddxdes = avec*ddbasis;

        ydes = bvec*basis;
        dydes = bvec*dbasis;
        ddydes = bvec*ddbasis;

        % compute sin(theta_d)

        thetades = atan2(dydes, dxdes);
        Xdes(:,i)= [xdes;ydes;thetades];

        % The desired state.
        xdes_vec = [xdes; ydes; thetades];

        % compute the feedforward in the input.
        vf = s*(dxdes*cos(thetades) + dydes*sin(thetades));
        % dthetades = 1/vf*(ddydes*cos(thetades) - ddxdes*sin(thetades));
        dthetades = ((dxdes*ddydes) - (dydes*ddxdes))/((dxdes^2)+(dydes^2));
        wf = dthetades;
%         phi = atan2(l*thetades / vf);
        b = ((dxdes*ddydes) - (dydes*ddxdes))/(dxdes^2);
        c = 1/(vf*(sec(theta))^2);
        phi = atan2(b*c,1);
        Vf(:,i)= [vf];
        Phi(:,i)= [phi];
        % A =  [ 0, 0,  vf*cos(thetades);
        %  0, 0, -vf*sin(thetades);
        %  0, 0,             0];
        % B = [ sin(thetades),                     0;
        %  cos(thetades),                    0;
        %  tan(deltaf), vf*(tan(deltaf)^2 + 1)];


        A = [ 0, 0, -vf*sin(thetades);
```

```
            0, 0,  vf*cos(thetades);
            0, 0,             0];

        B = [ cos(thetades), 0;
            sin(thetades), 0;
            0, 1];

        Q= eye(3);
        R = eye(2);
        %if any(eig(A-B*K))>=0;
        K= lqr(A,B,Q,R);
        %end

        u = -K*(xvec - xdes_vec) + [vf; wf];

        dxvec = [u(1)*cos(theta);u(1)*sin(theta);u(2)];
        %
        % % without noise
         X(:,i+1)= dxvec*dt+ X(:,i);

        % with noise
        %X(:,i+1)= dxvec*dt+ X(:,i) +0.05*randn(1);
    end

    for i=1:N;
        t=tsteps(i);
        basis = [1; t; t^2; t^3];
        dbasis = [0; 1; 2*t; 3*t^2];
        ddbasis = [0; 0;2; 6*t];
        Xdes(1,i) = avec*basis;
        Xdes(2,i)= bvec*basis;
    end
end
```

```
Warning: Solution is not unique because the system is rank-deficient.
Warning: Solution is not unique because the system is rank-deficient.
```


Trajectory Plot
Y-Axis
X-Axis
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
-1
0
1
2
3
4
5
6


Velocity Plot
Velocity
Time
1.5
1
0.5
0
-0.5
-1
0
5
10
15
20
25


Phi Plot
Phi
Time
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
5
10
15
20
25

Assignment 2

-Aishwarya Jogdia

2] Dynamics for the system are:

$$m\ddot{x} = F_1 \cos\theta - F_2 \sin\theta$$
$$m\ddot{y} = F_1 \sin\theta + F_2 \cos\theta - mg$$
$$J\ddot{\theta} = rF_1$$

Given system:

$$\left.\begin{aligned} z_1 &= x - (J/mr)\sin\theta \\ z_2 &= y + (J/mr)\cos\theta \end{aligned}\right\} \text{—} (1)$$

To be differential flatness

$$x = \beta(z, \dot{z}, \dots, z^{(\alpha)})$$
$$y = \gamma(z, \dot{z}, \dots, z^{(\alpha)})$$

Differentiating eq^ns (1)

$$\left.\begin{aligned} \dot{z}_1 &= \dot{x} - (J/mr)\cos\theta\,\dot{\theta} \\ \dot{z}_2 &= \dot{y} - (J/mr)\sin\theta\,\dot{\theta} \end{aligned}\right\} \text{—} (2)$$

Re-Differentiating eq^n (2)

$$\left.\begin{aligned} \ddot{z}_1 &= \ddot{x} + (\dot{\theta})^2 (J/mr)\sin\theta - (J/mr)\cos\theta\,\ddot{\theta} \\ \ddot{z}_2 &= \ddot{y} - (\dot{\theta})^2 (J/mr)\cos\theta - (J/mr)\sin\theta\,\ddot{\theta} \end{aligned}\right\} \text{—} (3)$$

From the dynamics system substituting $\ddot{x}, \ddot{y}, \ddot{\theta}$

$$\ddot{z}_1 = (F_1/m)\cos\theta - (F_2/m)\sin\theta - (F_1/m)\cos\theta + (J(\dot{\theta})^2/mr)\sin\theta \quad \text{—} (4)$$

$$\ddot{z}_2 = (F_1/m)\sin\theta + (F_2/m)\cos\theta - g - (F_1/m)\sin\theta - (J(\dot{\theta})^2/mr)\cos\theta \quad \text{—} (5)$$

Multiplying eq^n (4) by $\cos\theta$
& eq^n (5) by $\sin\theta$

$$\ddot{z}_1 \cos\theta = \left(F_1/m\right)\cos^2\theta - \left(F_2/m\right)\sin\theta\cos\theta - \left(F_1/m\right)\cos^2\theta + \left(J(\dot{\theta})^2/mr\right)\cos\theta\sin\theta \quad \text{— (6)}$$

$$\ddot{z}_2 \sin\theta = \left(F_1/m\right)\sin^2\theta + \left(F_2/m\right)\cos\theta\sin\theta - g\sin\theta - \left(F_1/m\right)\sin^2\theta - \left(J(\dot{\theta})^2/mr\right)\cos\theta\sin\theta \quad \text{— (7)}$$

Substracting (6) + (7)

$$\ddot{z}_1\cos\theta + \ddot{z}_2\sin\theta = \left(\cancel{F_1/m}\right) - \left(\cancel{F_1/m}\right) - g\sin\theta$$

$$= -g\sin\theta.$$

OR

$$\ddot{z}_1\cos\theta = -\ddot{z}_2\sin\theta - g\sin\theta$$

OR

$$\tan\theta = \left(\frac{-\ddot{z}_1}{\ddot{z}_2 + g}\right)$$

OR

$$\theta = \tan^{-1}\left(\frac{-\ddot{z}_1}{\ddot{z}_2 + g}\right) = \text{function of } (z, \dot{z} \cdots z^{(n)})$$

From equation (1) as we can write θ in terms of z

$$\left.\begin{aligned} x &= z_1 + \left(J/mr\right)\sin\theta \\ y &= z_2 - \left(J/mr\right)\cos\theta \end{aligned}\right\} = \text{function of } (z, \dot{z} \cdots z^{(n)})$$

Also, $\left.\begin{aligned} F_1 &= \frac{J\ddot{\theta}}{r} \\ F_2 &= \frac{F_1 \cos\theta - m\ddot{x}}{\sin\theta} \end{aligned}\right\} = \text{function of } (z, \dot{z}, \ldots z^{(a)})$

Hence all the state and input variables are functions of (z_1, z_2) flat output and their higher-order derivatives.