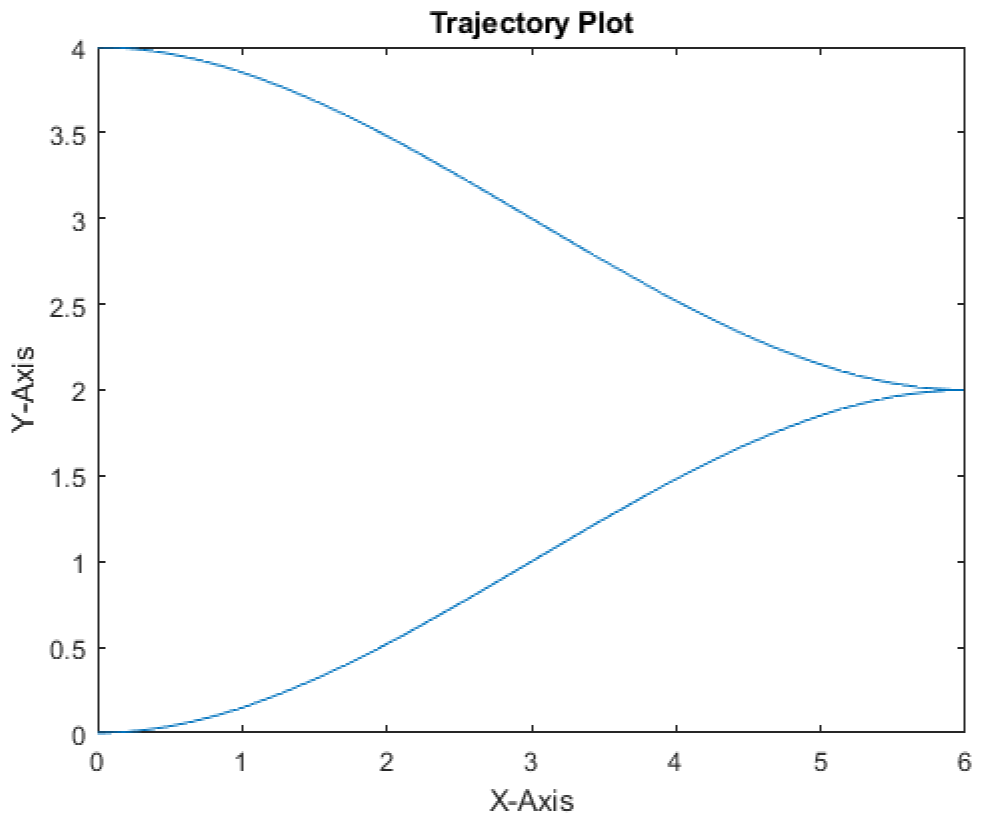
Problem 1:

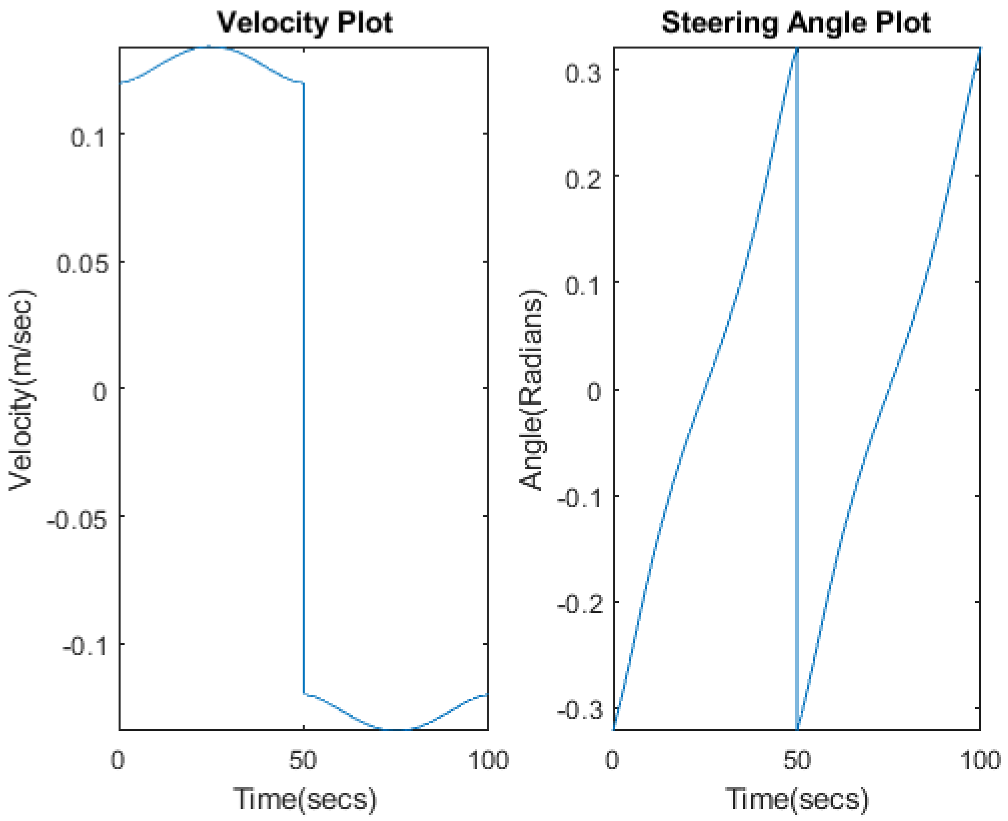
```
%Initial and final condition (x, y, theta)
clc; clear all; close all;
q0 = [0, 4, 0];
qi = [6, 2, 0];
qf = [0, 0, 0];
T0 = 0;
Ti = 50;
Tf = 100;    % time to reach destination

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 = diff(basis);
xsym = a*basis;
ysym = b*basis;
dx = a*dbasis;
dy = b*dbasis;

x0 = subs(xsym,t,T0);
xi = subs(xsym,t,Ti);
xf = subs(xsym,t,Tf);
dx0 = subs(dx,t,T0);
dxi = subs(dx,t,Ti);
dxf = subs(dx,t,Tf);

y0 = subs(ysym,t,T0);
yi = subs(ysym,t,Ti);
yf = subs(ysym,t,Tf);
dy0 = subs(dy,t,T0);
dyi = subs(dy,t,Ti);
dyf = subs(dy,t,Tf);

% compute the jacobian linearization of the vector field.
l=1;
syms v w theta x y phi
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))==0, ...
xi==qi(1), yi==qi(2), dxi*sin(qi(3))-dyi*cos(qi(3))==0],
[a(1),a(2),a(3),a(4),b(1),b(2),b(3),b(4)]);
param1 = matA\matb;
```

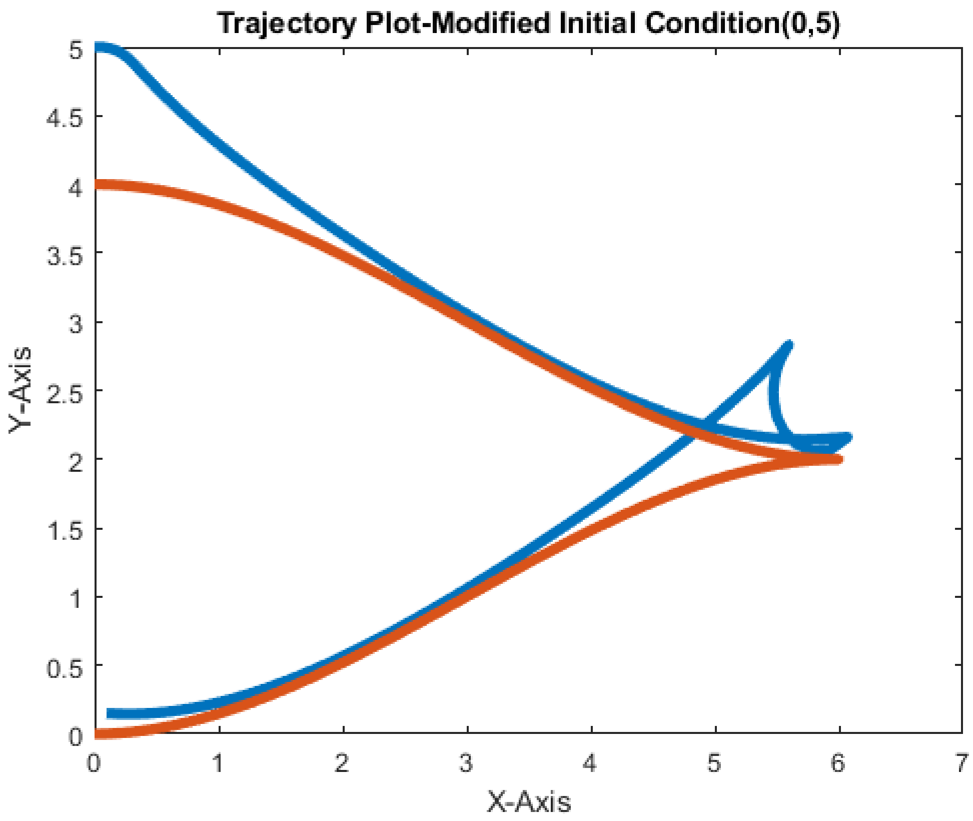
```
[matA,matb] = equationsToMatrix([xi==qi(1), yi==qi(2), dxi*sin(qi(3))-
dyi*cos(qi(3))==0, ...
xf==qf(1), yf==qf(2), dxf*sin(qf(3))-dyf*cos(qf(3))==0],
[a(1),a(2),a(3),a(4),b(1),b(2),b(3),b(4)]);
param2 = matA\matb;

X1 = param1(1:4)'*basis;
Y1 = param1(5:8)'*basis;
X2 = param2(1:4)'*basis;
Y2 = param2(5:8)'*basis;

%Trajectory Plot
% fplot(X1,Y1,[T0,Ti]);
% hold on;
% fplot(X2,Y2,[Ti,Tf]);
% title('Trajectory Plot');
% xlabel('X-Axis');
% ylabel('Y-Axis');

X_vec = piecewise(t<Ti,X1,t>Ti,X2);
Y_vec = piecewise(t<Ti,Y1,t>Ti,Y2);

dX = diff(X_vec,t);
dY = diff(Y_vec,t);

ddX = diff(dX,t);
ddY = diff(dY,t);

theta = atan(dY/dX);
v = dX*cos(theta) + dY*sin(theta);
temp1 = (ddY*dX - ddX*dY) / (dX)^2;
temp2 = l/(v*(sec(theta))^2);
phi = atan(temp1*temp2);

figure(1);
fplot(X_vec,Y_vec,[T0 Tf]);
title('Trajectory Plot');
xlabel('X-Axis');
ylabel('Y-Axis');

% fplot(theta,[T0 Tf]);
% title('Angle Plot');
% xlabel('X-Axis');
% ylabel('Y-Axis');

figure(2);
subplot(1,2,1)
fplot(v,[T0 Tf]);
title('Velocity Plot');
ylabel('Velocity(m/sec)');
xlabel('Time(secs)');

subplot(1,2,2)
fplot(phi,[T0 Tf]);
```

```
title('Steering Angle Plot');
ylabel('Angle(Radians)');
xlabel('Time(secs)');

ode_tracking
```

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

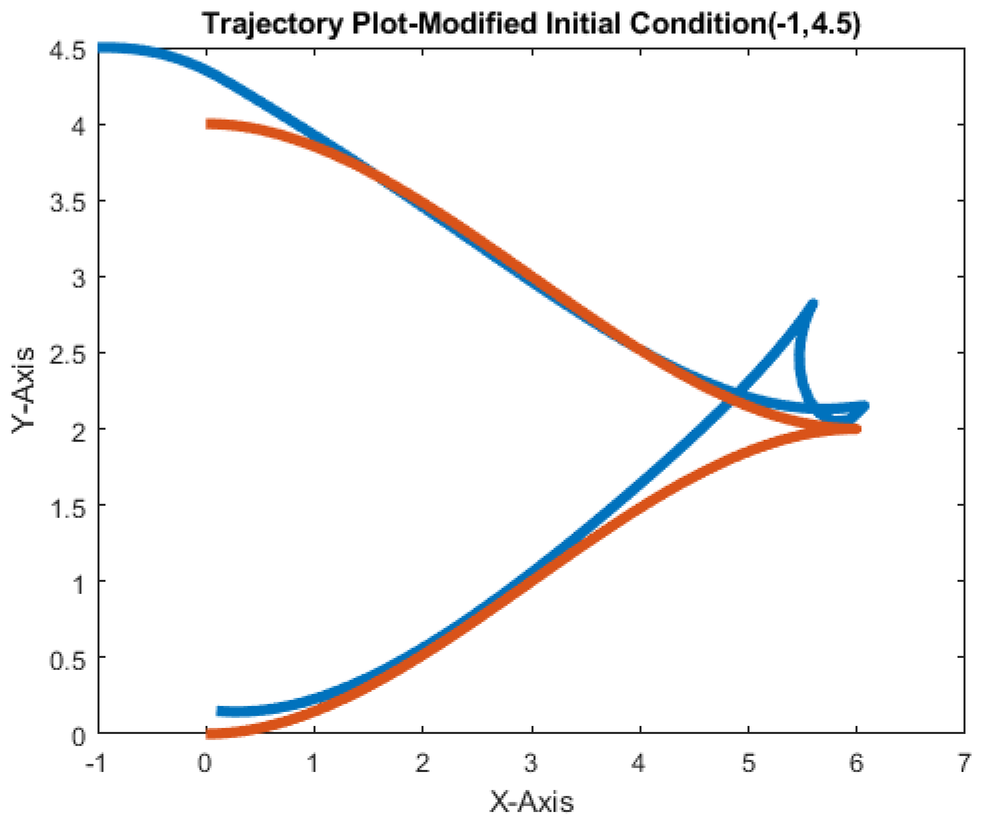



Velocity Plot
Velocity(m/sec)
Time(secs)
0.1
0.05
0
-0.05
-0.1
0
50
100
Steering Angle Plot
Angle(Radians)
Time(secs)
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
0
50
100


Trajectory Plot-Modified Initial Condition(0,5)
Y-Axis
X-Axis
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
0
1
2
3
4
5
6
7

Published with MATLAB® R2017b

```
dt=0.1;
tsteps=[T0:dt:Tf];
N=size(tsteps,2);
% X(:,1)=[0, 2, 0];
X(:,1)=[-1, 4.5, 0];

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];

    if(t<Ti)
        xdes = param1(1:4)'*basis;
        dxdes = param1(1:4)'*dbasis;
        ddxdes = param1(1:4)'*ddbasis;

        ydes = param1(5:8)'*basis;
        dydes = param1(5:8)'*dbasis;
        ddydes = param1(5:8)'*ddbasis;
    else
        xdes = param2(1:4)'*basis;
        dxdes = param2(1:4)'*dbasis;
        ddxdes = param2(1:4)'*ddbasis;

        ydes = param2(5:8)'*basis;
        dydes = param2(5:8)'*dbasis;
        ddydes = param2(5:8)'*ddbasis;
    end
    % 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 = dxdes*cos(thetades) + dydes*sin(thetades);
    dthetades = 1/vf*(ddydes*cos(thetades) - ddxdes*sin(thetades));
    wf = dthetades;
    phi1 = atan2(wf,vf);


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

    B = [ cos(thetades), 0;
          sin(thetades), 0;
          tan(phi1), vf*(1+tan(phi1)^2)];

    Q = eye(3);
    R = eye(2);
    %if any(eig(A-B*K))>=0;
    K = lqr(double(A),double(B),double(Q),double(R));
    % K = [1 1 1; 1 1 1];
    %end

    u = -K*(xvec - xdes_vec) + [vf; phi1];

    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];
    if(t<Ti)
        Xdes(1,i) = param1(1:4)'*basis;
        Xdes(2,i)= param1(5:8)'*basis;
    else
        Xdes(1,i) = param2(1:4)'*basis;
        Xdes(2,i)= param2(5:8)'*basis;
    end
end

figure
plot(X(1,:), X(2,:),'LineWidth', 4);
hold on
plot(Xdes(1,:), Xdes(2,:), 'LineWidth', 4);
title('Trajectory Plot-Modified Initial Condition(-1,4.5)');
xlabel('X-Axis');
ylabel('Y-Axis');
```



Published with MATLAB® R2017b