Robot Homework 2
11610303 黄玉安

H1.**Discuss the advantages and limits of different wheel drives (for differential drive, Ackerman Drive, Synchronous Drive, XR4000 drive, and MecanumDrive, list their pros and cons in terms of cost, degree of mobility, degree of steerability, degree of maneuverability, workload capacity, motion control complexity）**

Notation: D(x): the degree of x;

Differential drive
pros: D(mobility) = 2, D(maneuverability) = 2; The workload capacity is large, cost and the motion control complexity is low.
cons: D(steerability) = 0;

Ackerman Drive
pros: D(steerability) = 1, D(maneuverability) = 2; The workload capacity is very large, cost and the motion control complexity is low.
cons: D(mobility) = 1;

Synchronous Drive
pros: D(mobility) = 3, D(maneuverability) = 3;The workload capacity is very large.
cons: D(steerability) = 0; The cost and motion control complexity is usually high.

XR4000 drive
pros: D(mobility) = 3, D(maneuverability) = 3; The workload capacity is quite large.
cons: D(steerability) = 0; The cost and the motion control complexity is usually high.

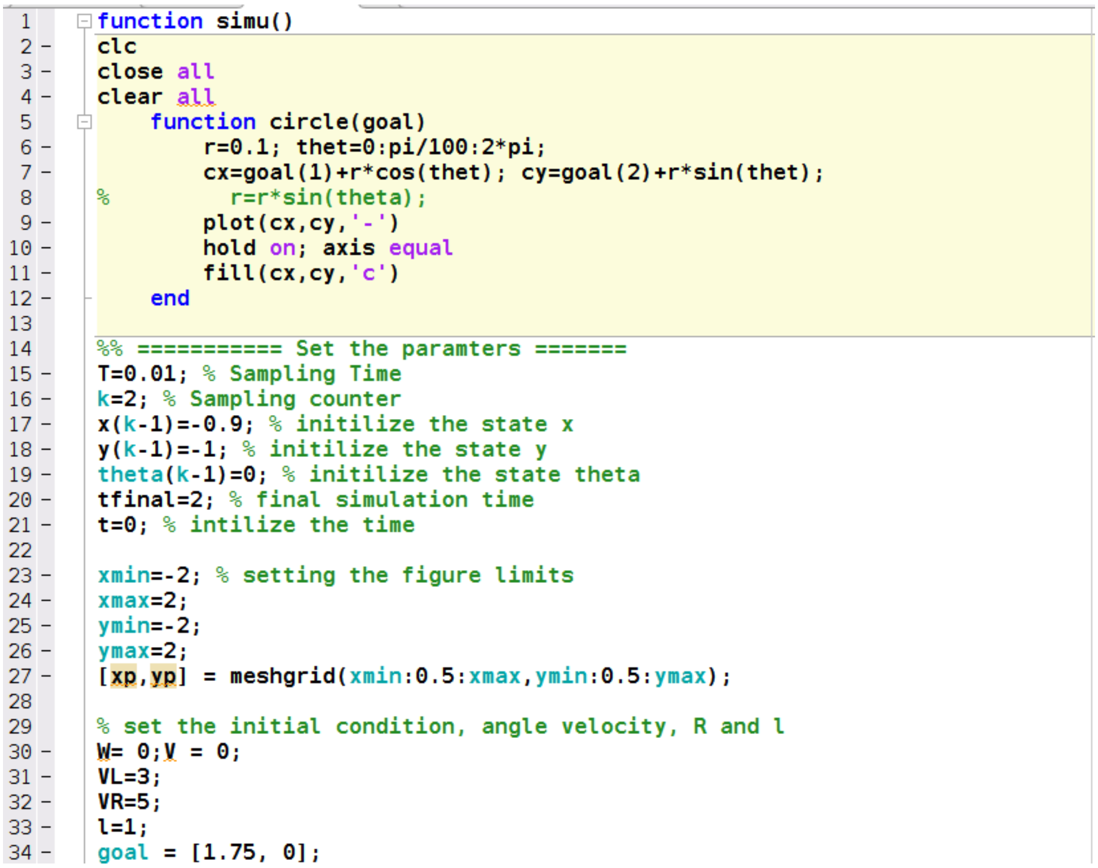
MecanumDrive
pros: D(mobility) = 3, D(maneuverability) = 3; The motion control complexity is small since it can move to any direction.
cons: D(steerability) = 0; The cost is high and the workload capacity is quite small.

H2.**Simulate the kinematics of a robot with a differential drive (plot a motion trajectory, given the initial pose [position orientation] and goal pose [position orientation], by selecting feasible velocities and assuming a goal tolerance; discuss the effects of different selections of goals and velocities; provide codes and plots of simulation results）**

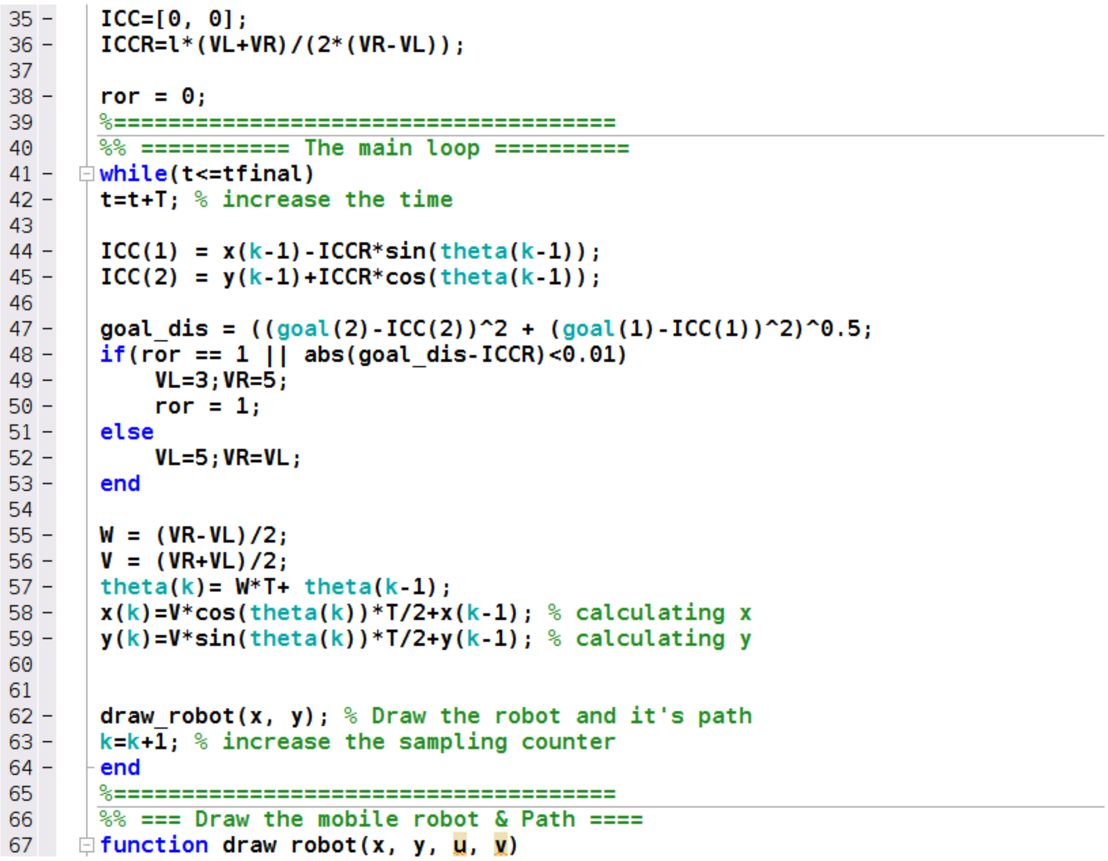
H3.**Simulate the motion control of a robot with a differential drive and show the resulting paths w.r.t. different control gains (Provide codes and plots of simulation results**

The code is showing below(just part, the whole code is provided in the another source file). Function circle(goal) is to draw a goal position which is a blue circle. Line 17~18 is to make sure the initial position and line 34 make sure the goal position.

I let VL(left wheel speed) = 3, VR(right wheel speed) = 5 when rotating and other case they are the same which is 5. The key code to implement differential drive is in line 44~59. It will go straight first and turn direction in proper time.



```
function simu()
clc
close all
clear all
    function circle(goal)
        r=0.1; thet=0:pi/100:2*pi;
        cx=goal(1)+r*cos(thet); cy=goal(2)+r*sin(thet);
%         r=r*sin(theta);
        plot(cx,cy,'-')
        hold on; axis equal
        fill(cx,cy,'c')
    end

%% =========== Set the paramters =======
T=0.01; % Sampling Time
k=2; % Sampling counter
x(k-1)=-0.9; % initilize the state x
y(k-1)=-1; % initilize the state y
theta(k-1)=0; % initilize the state theta
tfinal=2; % final simulation time
t=0; % intilize the time

xmin=-2; % setting the figure limits
xmax=2;
ymin=-2;
ymax=2;
[xp,yp] = meshgrid(xmin:0.5:xmax,ymin:0.5:ymax);

% set the initial condition, angle velocity, R and l
W= 0;V = 0;
VL=3;
VR=5;
l=1;
goal = [1.75, 0];
```

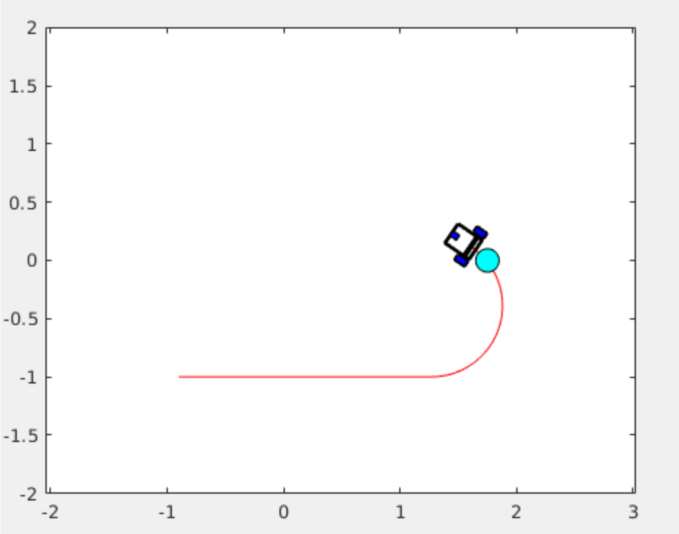


```
ICC=[0, 0];
ICCR=l*(VL+VR)/(2*(VR-VL));

ror = 0;
%=======================================
%% =========== The main loop ==========
while(t<=tfinal)
t=t+T; % increase the time

ICC(1) = x(k-1)-ICCR*sin(theta(k-1));
ICC(2) = y(k-1)+ICCR*cos(theta(k-1));

goal_dis = ((goal(2)-ICC(2))^2 + (goal(1)-ICC(1))^2)^0.5;
if(ror == 1 || abs(goal_dis-ICCR)<0.01)
    VL=3;VR=5;
    ror = 1;
else
    VL=5;VR=VL;
end

W = (VR-VL)/2;
V = (VR+VL)/2;
theta(k)= W*T+ theta(k-1);
x(k)=V*cos(theta(k))*T/2+x(k-1); % calculating x
y(k)=V*sin(theta(k))*T/2+y(k-1); % calculating y


draw_robot(x, y); % Draw the robot and it's path
k=k+1; % increase the sampling counter
end
%=======================================
%% === Draw the mobile robot & Path ====
function draw_robot(x, y, u, v)
```

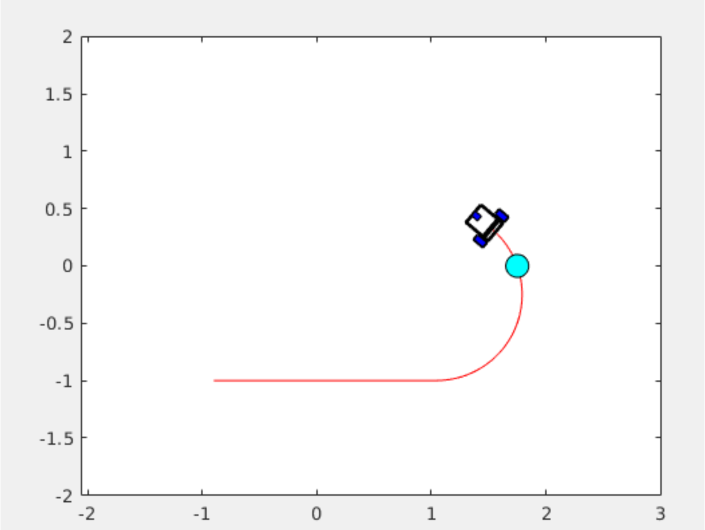
However, the result path for this robot will change when I select different speed of left wheel and keep initial position and goal position not change.

Those figure below shows this result.

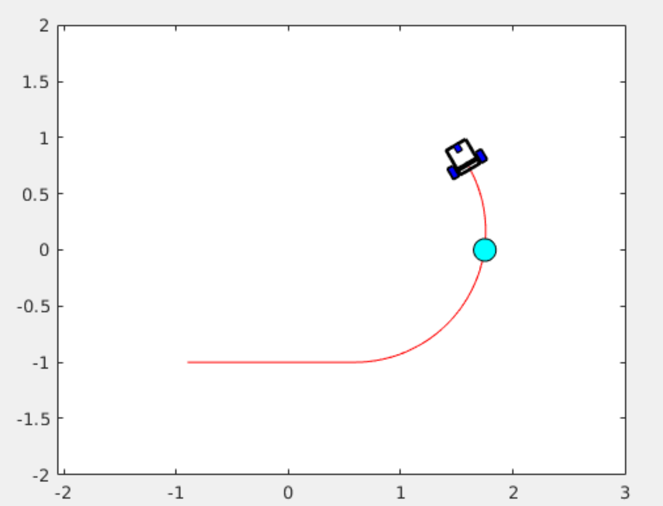
Vl=0.5, VR=5



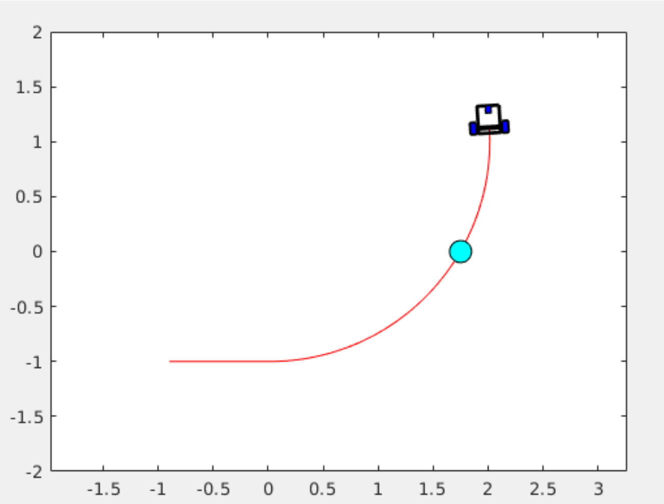
Vl=1, VR=5



Vl=2, VR=5



Vl=3, VR=5


2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5
3