ELEE 4200

Autonomous Mobile Robots

Homework No:1

MATLAB-ROS Connectivity(OR Let's Jump into the Water!)

Author 1:
Name :Wang Jingya
T.No.: 02157119

Author 2:
Name : Mi liang
T.No.: 02157118

Instructor:
Dr. Mohan Krishnan

September 19, 2019

Grading Rubric - Information for Students

Grading rubric that will broadly apply to assessing performance. That is, assessment exercises that are associated with a predominantly qualitative rather than quantitative character. If, for a particular exercise, there is substantial deviation from the scheme outlined below, I will let you know! Note: Be aware that if your "Quality of Presentation" is poor, it could impact scores assigned for the other two categories! (DO NOT EDIT THIS PAGE)

		Level of accomplishment					
		0	**1**	**2**	**3**	**4**	**5**
Aspect of effort	**Extent of Completion of Technical Requirements**	Achieved none of the objectives or did not submit	Achieved very few of the objectives	Achieved some of the objectives	Achieved most of the objectives	Achieved almost all the objectives	Demonstrated additional initiative beyond what was required.
	Quality of Analysis & Conclusions	Inadequate	Poor	Below average	Average	Good	Insightful!
	Quality of Presentation	Inadequate	Poor	Below average	Average	Good	Exceptional!

Figure 1: Level of accomplishments

- **Student's overall Level/Score (To be entered by Professor/GTA):**.

Contents

List of Figures

List of Tables

1 Abstract

Practising the skill that programming on Matlab, and simulating on Gazebo and RViz within the Matlab environment, in order to develop the ability of using the MATLAB-ROS connectivity.

2 Introduction

- The homework requirements:

  Part A: Identify the linear velocity and angular velocity, and then change them severe times. Finally, conclude the differences.

  Part B:

  1. Programming the code which can make the robot run like a letter "L" in Matlab.

  2. Setting up the Turtlebot in Gazebo in a appropriately way and let the result present as a letter "L"

  3. In Rviz , to identify whether the robot run as a letter "L".

  4. Output the time of the whole process.

  5. Printing screen and then recording the video.

3 Methodology

Part A:
First, use the "Run Section" to write the three codes in one script.
Second, open GAZEBO, RVIZ and build the model in it.
Third, run three code one by one.
Fourth, observe the result in GAZEBO and RVIZ.
Part B:
B.1 First Version:
The idea is :
1.Dividing the picture into different parts.
Two lines and five parts of circles.
2.By simple calculation, get each parts' base data.
Get length, radius and angle by drawing measurement(As the picture

figure 2 below).
3.Get more data by calculate.
Use available data figure out each parts' distance.
Use formula $T = \frac{s}{v}$ to get the time that robot should running.
4.Write initial code.
First, write the basic code for initialise ROS, build and publish topic.
Second, build a function and two global variables to change the data and the robot's movement condition.
Third, add a variables i and instruct "pause" in "while" loop to calculate the time robot running and as an input in function to judge whether change robot's movement condition.
Fourth, put the calculate data into function and fish initial code.
5.Modify values in running.
First analyse robot's motion trajectory and find the parts need to be change.
Third, Calculate again and change the data in function.
Fourth, repeat the third step until get suitable line.
And modify values in latter experiments. B.2.Second version

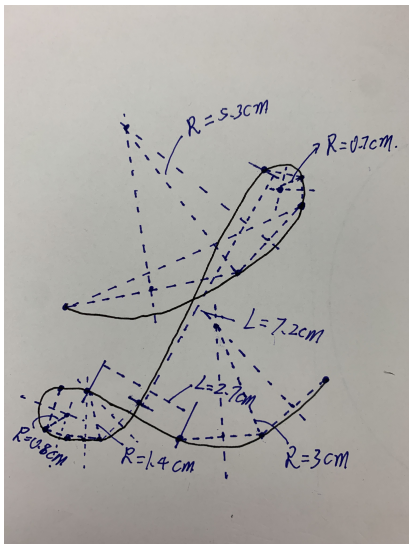


Figure 2: Calculation diagram

The idea is :
Firstly, dividing the letter "L" into three parts;like the figure 3 below.

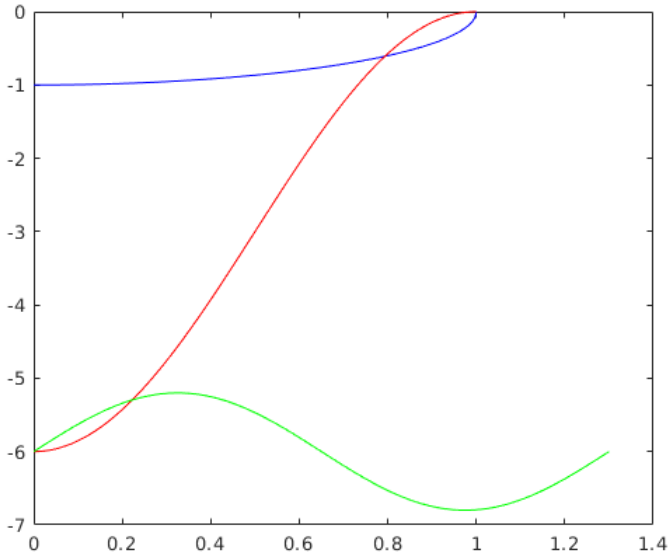


Figure 3: Dividing figure

First blue part is seen as a $1 \div 4$ part of a circle, and the center is (0,0), radius is 1;
This part function is $f(x) = centerpointy - \sqrt{radius^2 - x1.^2}$;
Second red part is seen as the part of a specified cos function, which is the range from 0 to π, and reverse the figure on the x axis, and shift to the -y axis direction 3 .
This part function is $f(x) = -3 * \cos(\pi * x) - 3$;
Third green part is seen as the part of a complete specified sin function, which is the range from 0 to $2 * \pi$
This part function is $f(x) = 0.8 * \sin(20 \div 13 * \pi * x) - 6$;

We use the math knowledge to calculate the range of three functions and the specific equation of them. And using math basic function to draw the letter "L", and making the combination of the shape looks like the shape in the figure. This part code is above, and the plotting figure is the figure 3
Secondly, calculating the first derivative of each function, which is

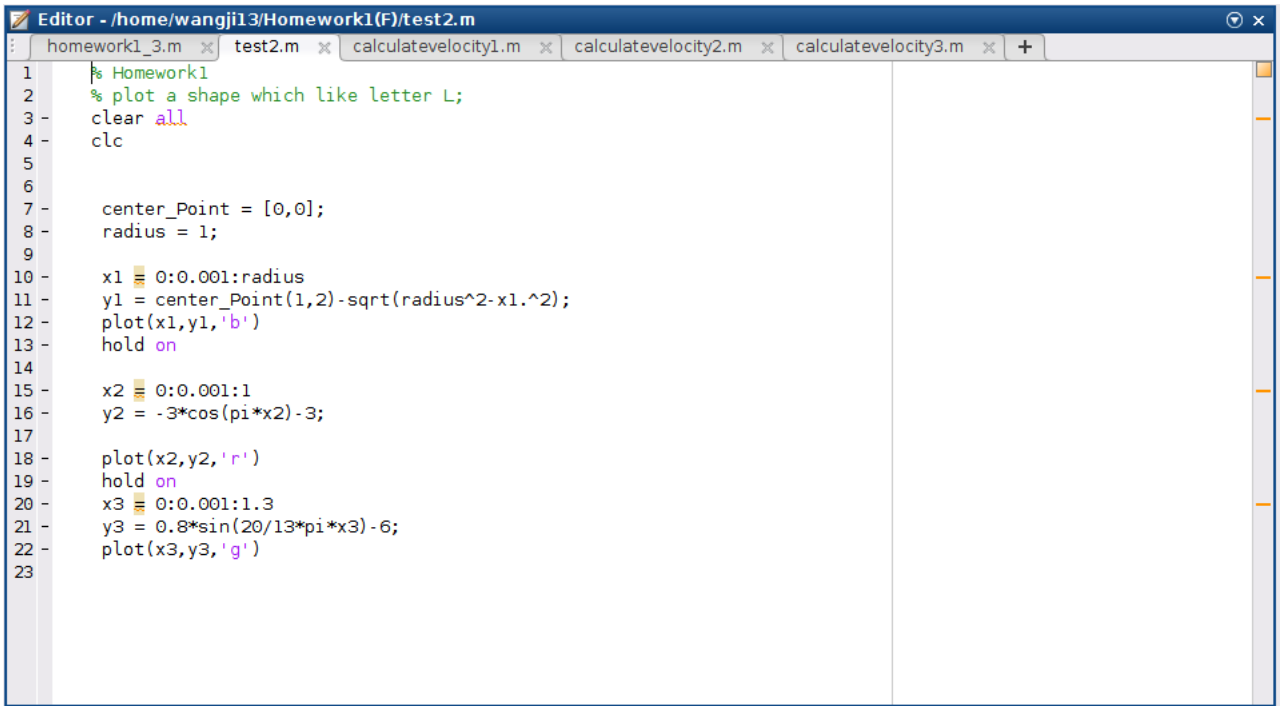


Figure 4: The code of the plotting figure

called k(x);

And the result has two meanings:

1.the velocity of the robot;

2.it also means the slope of each point's differential line segment

According to the two meanings, if we define a δx, and the δy = δx*k(x0);The result can calculate by $linear_velocity = \sqrt{\delta x^2 + \delta y^2}$. Then integrating each step, and the result is a shape which looks like letter "L".

But how to define the angular velocity? Because we identify δx very small, so we divide each step into a very little one, which is same as the differential thought; Therefore, we can use the slope of each point's differential line segment to calculate the θ.

$$\theta = \arctan(k(x0));$$

And then let angular velocity of each step equal to $\delta\theta$ and also equal to θ, which is

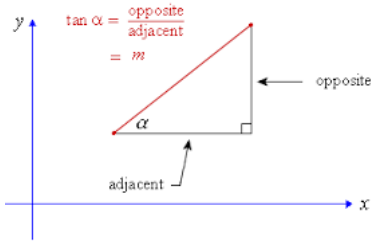


Figure 5: Use slope to calculate the angle[1]

$$\text{angular velocity} = \delta\theta = \theta$$

Fourthly, input every steps velocity to the turtle robot, use tic and toc to record the time.
Finally, open GAZEBO and RViz to simulate the result,run the code, and video the result.

4 Results

Part A In the first and second test, the robot move a perfect circle(As the picture figure below) and by observing the path on the map we can find the radius that robot move. By observe the data we get. We can verified the formula $R = \frac{v}{w}$.

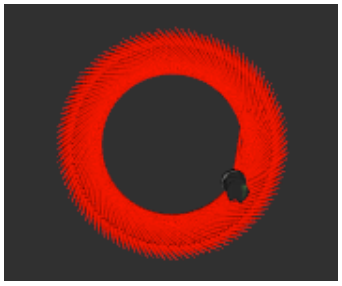
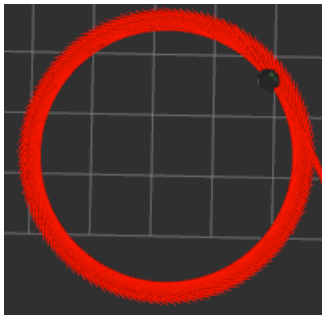
[Text 1]  [Test 2] 

Figure 6: Solution2 's figure1 and figure2

But the third test go wrong(As the picture figure below). At beginning the robot move naturally but in the later the radius of robot moving beginning to decrease and just revolve in the last. The consequence we

get is that palstance(w) is too large and robot can not move naturally in such large palstance.

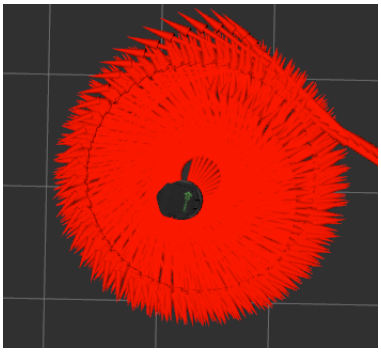


Figure 7: result

Part B
The first version:
In the last, get the perfect result(As the picture figure 8 below), although meet a problem in the process.

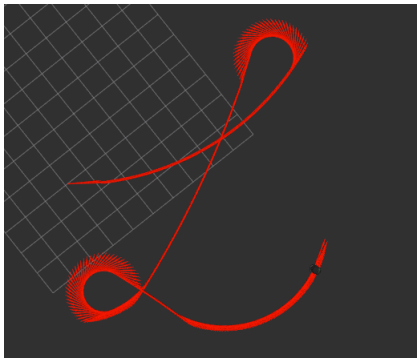


Figure 8: result

The second version:
The second version's result is actually not same as our prediction. And during the debug process, we met many problems, and the result is just satisfied the requirements, but not a perfect letter "L".
The first result of the second version:

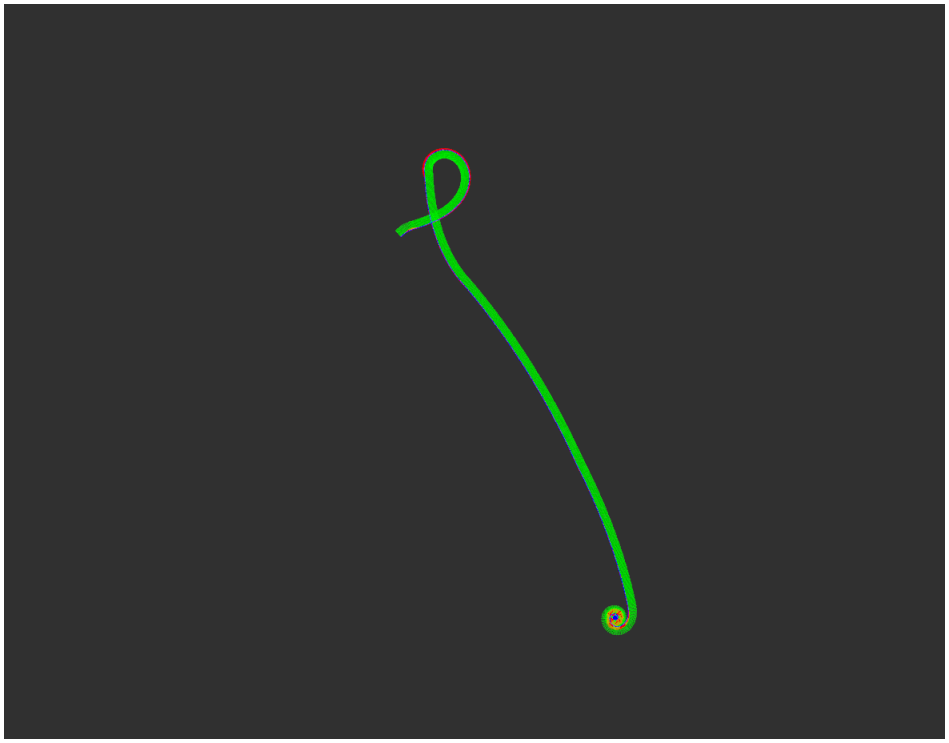


Figure 9: The first result of the second version

At the third function, the robot begin to rotate. According to the teacher's introduction, this is might because that the angular velocity is very big. And we should use coefficient to equal proportion reduce the the angular velocity result.(How to solve it will be explained in the discussion and conclusion part.)

The second result of the second version:

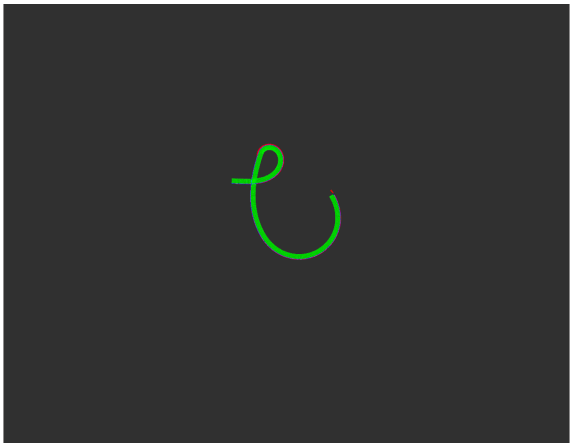


Figure 10: The second result of the second version

The program I even didn't finish to run. Because at the second corner,it turn to the wrong direction.

The third result of the second version:

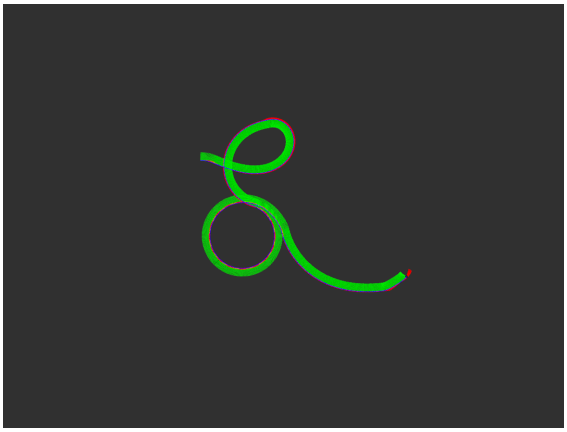


Figure 11: The third result of the second version

The result is the most satisfied the shape of latter "L". And this is the result of observing and change the coefficients.(The detailed part I will be explained in the discussion parts)

5 Discussions and Conclusion

- The homework requirements:

  Part A:

  The goal is to

  In the first and second codes the robot running in good condition, but the last one robot appear problem that in a few seconds, after running the robot began to drift and can't to maintain moving state and just rotation in the last.

  Part B:

  The first Version:

  The first version's result meet expectations perfectly, unless a accidental circle at the beginning(As the picture figure 12 below). There is noting to build this circle in my code. I do lots things to find the problem in my code, but nothing be found. In the later experiment i see the obstacles in "gazebo" accidentally. I think may be the obstacles hinder the robot's move in "rviz" and i delete them. As expected, those obstacles are reason.

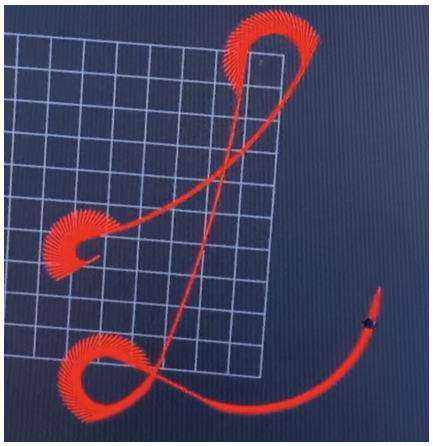


Figure 12: result

B.2 The second Version:
The result is satisfied the goal in the end. But we think the second

version is not a very 'perfect' one.

1.The first problem: The first version of this idea:

```
% Homework1
% plot a shape which like letter L;
clear all
clc

rosshutdown;
ipaddress='localhost';
rosinit(ipaddress);
robot = rospublisher('/mobile_base/commands/velocity');
velmsg = rosmessage(robot);

angular_velocity = 0;


 center_Point = [0,0];
 radius = 1;

 x1 = 0:0.001:radius
 y1 = center_Point(1,2)-sqrt(radius^2-x1.^2);
 k1 = diff(y1);
 mx = 0.1;
 my = mx.*k1;
 linear_velocity1 = sqrt(my.^2+mx);

velmsg.Linear.X = linear_velocity1;
velmsg.Angular.Z= angular_velocity;
send(robot,velmsg);

 x2 = 0:0.001:1
 y2 = -3*cos(pi*x2)-3;
 k2 = diff(y2);
 mx = 0.1;
```

```
33 -     my = mx.*k2;
34 -     linear_velocity2 = sqrt(my.^2+mx);
35
36 -     velmsg.Linear.X = linear_velocity2;
37 -     velmsg.Angular.Z= angular_velocity;
38 -     send(robot,velmsg);
39
40 -     x3 = 0:0.001:1.3
41 -     y3 = 0.8*sin(20/13*pi*x3)-6;
42 -     k3 = diff(y3);
43 -     mx = 0.1;
44 -     my = mx.*k3;
45 -     linear_velocity3 = sqrt(my.^2+mx);
46
47 -     velmsg.Linear.X = linear_velocity2;
48 -     velmsg.Angular.Z= angular_velocity;
49 -     send(robot,velmsg);
50
51
```

Figure 13: First code of the second version

Firstly, we even can't run our code. The first problem is that the data type is of the linear velocity is not suitable for output, which is like the figure 14below
And the final code is like above:

```
        validateattributes(x, {'numeric'}, {'nonempty', 'scalar'}, 'Vector3', 'X');

Error in homework1 (line 25)
velmsg.Linear.X = linear_velocity1;

>>
```

Figure 14: The first problem

Solving the problem: we think we can change the data type of the calculate results into number type before output to use the function that $z = double(result)$
And the solution code is like above:

```
18 -  for x1 = 0:0.001:0.741
19 -      k1=calculatevelocity1(x1);
20 -      my1 = mx*k1;
21 -      linear_velocity1 = sqrt(my1^2+mx);
22 -      velmsg.Linear.X =mx;
23
24 -      z11 = double(linear_velocity1);
25
26 -      velmsg.Linear.Y =z11;
27
28 -      angular1 = atan(k1);
29 -      z12=double(angular1);
30 -
31 -      if z12 <0.5
32 -        velmsg.Angular.Z= z12;
33 -      else
34 -        velmsg.Angular.Z= 0.1*z12;
35 -      end
36 -      send(robot,velmsg);
37 -      x1
38
39 -  end
```

Figure 15: Solution of the first code of the second version

As you can see, the line 24 and line 29, we add two lines to change the data type.Finally, we can run the code.This is just the first function, the same solution to the second and the third function.

2.The second problem:
As you can see in the below figure 16:

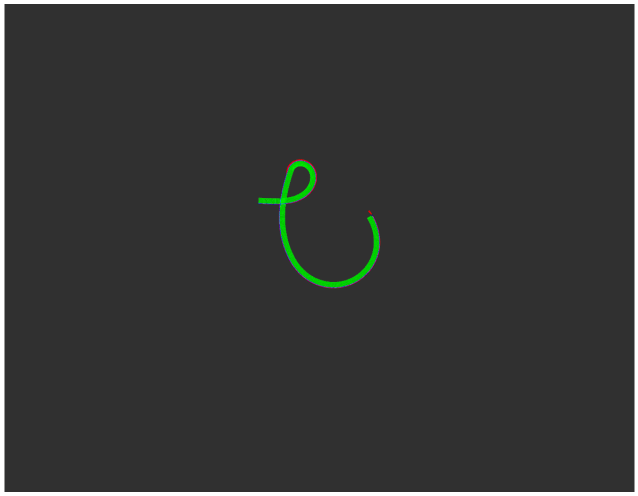


Figure 16: The second result of the second version

The second problem is that because the angular velocity are too big, the robot begin to rotate at the second corner. So we think we should limit the output values. To identify the shape, we consider to multiply the calculate result *angular* with a coefficient (range is from 0 to 1), equal proportion reduce the Z,and make sure the robot are not rotating.
And also, we add a x output and let the robot to find the boundary of where to reduce the Z.All the three function need to change.

The second version of this idea:

```
% Homework1
% plot a shape which like letter L;
clear all
clc
rosshutdown

ipad='localhost';
rosinit(ipad);
robot = rospublisher('/mobile_base/commands/velocity');
velmsg = rosmessage(robot);
%angular_velocity = 0


 center_Point = [0,0];
 radius = 1;
 mx = 1;

for x1 = 0:0.001:0.741
   k1=calculatevelocity1(x1);
   my1 = mx*k1;
   linear_velocity1 = sqrt(my1^2+mx);
   velmsg.Linear.X =mx;

   z11 = double(linear_velocity1);

   velmsg.Linear.Y =z11;

   angular1 = atan(k1);
   z12=double(angular1);

   if z12 <0.5
     velmsg.Angular.Z= z12;
```

```
    else
      velmsg.Angular.Z= 0.1*z12;
    end
    send(robot,velmsg);
    x1

end


for x2 = 0.02:0.001:0.561
    k2 = calculatevelocity2(x2);

    my2 = mx*k2;
    linear_velocity2 = sqrt(my2^2+mx^2);

    velmsg.Linear.X = mx;

    z21 = double(linear_velocity2);
    velmsg.Linear.Y =z21;

    angular2 = atan(k2);

    z22=double(angular2);
    if z22 <0.05
      velmsg.Angular.Z= z22;
    else
      velmsg.Angular.Z= 0.01*z22;
    end
    send(robot,velmsg);
    x2
```

```
53 -    end
54
55 -        angular = pi;
56 -        z=double(angular);
57 -        velmsg.Angular.Z= z;
58 -        send(robot,velmsg);
59
70 -    for x3 = 0.002:0.001:1.3
71 -        k3 = calculatevelocity3(x3);
72 -        my3 = mx*k3;
73 -        linear_velocity3 = sqrt(my3^2+mx^2);
74 -        velmsg.Linear.X = mx;
75 -        z31 = double(linear_velocity3);
76 -        velmsg.Linear.Y =z31;
77 -        angular3 = atan(k3);
78 -        z32=double(angular3);
79 -        if z32 <0.05
80 -         velmsg.Angular.Z= z32;
81 -        else
82 -         velmsg.Angular.Z= 0.01*z32;
83 -        end
84 -        x3
85 -        send(robot,velmsg);
86 -    end
87
```

Figure 17: Second code of the second version

3.The third problem:
At the second corner, the robot turns into the wrong direction. As you can see below,

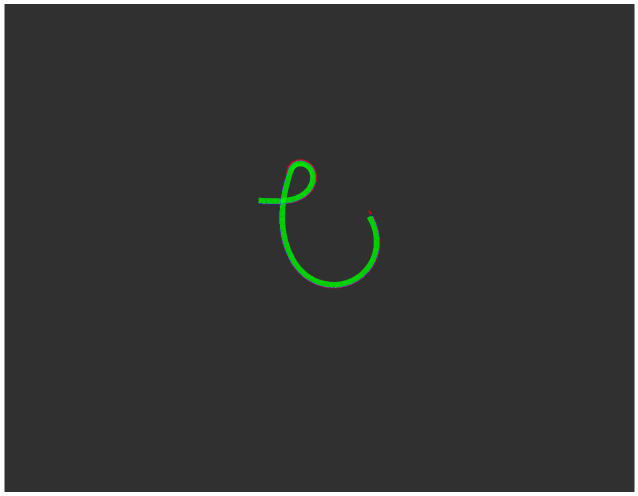


Figure 18: The third result of the second version

The problem confused us the longest time. And we don't know why the right figure function can not output the right direction.Then we asked the TA, and get the answer.
It is because that the $\theta = \arctan(k(x0))$ can get both negative and positive results. It makes the robot to rotate counterclockwise when we want it to rotate clockwise.
And TA suggested that we should also output the angle we already calculate, and to see which one is not fit for our requirement and to change it.
In the end, we find the boundary of where to turn, and use "-" to change it and use $abs()$function to let the number which should stay at positive to keep it symbol.

The whole final code is in the appendix, so I just show the different part between the second version code.

```
    if z22 <0.5&&z22 >-0.5   %range of x2 is though we observe(explain detailed in the report)
        if x2 >=0.35  %range of x2 is though we observe(explain detailed in the report)
        velmsg.Angular.Z= -z22; %change the value into positive
        else
        %if not in the range, stay the positive z22 in initial value,and
        %change the negative one to positive
        velmsg.Angular.Z= abs(z22);
        end
    else
        % In some cases, the angular velocity is very big, so we proportional reduce the value,
        if x2 >=0.35
        % The coefficient is also determined by observe(explain detailed in
        % the report)
        velmsg.Angular.Z= -0.15*z22; %change the value into positive
        else
        %if not in the range, stay the positive z22 in initial value,and
        %change the negative one to positive
        velmsg.Angular.Z= abs(0.15*z22);
        end
    end

    %publish the message[4]
    send(robot,velmsg);
    x2
    z22
end
```

Figure 19: The solution of the third problem

References

[1] slope to calculate the angle. https://www.google.com/search?

A Appendix

The following are guidelines. Use your judgment appropriately
Part A code:

```
%%Section1
% forward Velocity= 0.5
% angular Velocity= 0.5

rosshutdown
ipad='localhost';  %get address
rosinit(ipad);     %initialise
robot = rospublisher('/mobile_base/commands/velocity'); %build a velocity topic
velmsg= rosmessage(robot);  %build a message about velocity topic
forwardVelocity= 0.5;
angularVelocity= 0.5;
velmsg.Linear.X= forwardVelocity;  %change spead
velmsg.Angular.Z=angularVelocity;  %change palstance
while 1    %keep send message
send(robot,velmsg);  %send message
end
%%
%%Section2
% forward Velocity= 0.25;
% angular Velocity= 0.125;

rosshutdown
ipad='localhost';
rosinit(ipad);
robot = rospublisher('/mobile_base/commands/velocity');
velmsg= rosmessage(robot);
forwardVelocity= 0.25;
angularVelocity= 0.125;
velmsg.Linear.X= forwardVelocity;
velmsg.Angular.Z=angularVelocity;
while 1
send(robot,velmsg);
end
%%
%%Section 3
%forwardVelocity= 1
%angularVelocity= 1

rosshutdown
ipad='localhost';
rosinit(ipad);
robot = rospublisher('/mobile_base/commands/velocity');
velmsg= rosmessage(robot);
forwardVelocity= 1;
angularVelocity= 1;
velmsg.Linear.X= forwardVelocity;
velmsg.Angular.Z=angularVelocity;
while 1
send(robot,velmsg);
end
```

Figure 20: Part A code

Part B code: version 1 code:

```
clc;
clear all;
rosshutdown;
ipad='localhost';  %get the address
rosinit(ipad);     %initialise
robot=rospublisher('/mobile_base/commands/velocity'); %build a velocity topic
velmsg=rosmessage(robot);%build a message about topic "rotob"
global m; %control the "while" loop
global linear_velocity;  %use global to change the data in function
global angular_velocity; %use global to change the data in function
m=1;i=1;
while m
function_changedata(i);  % a function to change the robot's move
velmsg.Linear.X=linear_velocity;
velmsg.Angular.Z=angular_velocity;
send(robot,velmsg);     %send message
pause(0.1)  %use i and pause to caculate the times and control
i=i+1;
end
fprintf("the performance period is %d s ",i*0.1); %show the time of robot run
```

Figure 21: Main.m

```
function [] = function_changedata(input1)
global linear_velocity;  %use global to change the data
global angular_velocity; %use global to change the data
global m;
    if input1<200     %first part to move as a part of circle
linear_velocity=0.5;
angular_velocity=0.06;
    end
    if input1>200     %second part to move as a part of circle
linear_velocity=0.25;
angular_velocity=0.3;
    end
    if input1>316     %third part to move as a line
linear_velocity=0.5;
angular_velocity=0;
    end
    if input1>510     %fourth part to move as a part of circle
linear_velocity=0.5;
angular_velocity=-0.14;
    end
    if input1>550     %fiveth part to move as a part of circle
linear_velocity=0.25;
angular_velocity=-0.3;
    end
    if input1>680     %sixth part to move as a line
linear_velocity=0.5;
angular_velocity=0;
    end
    if input1>730     %seventh part to move as a part of circle
linear_velocity=0.5;
angular_velocity=0.14;
    end
    if input1>860     %stop the "while" loop and finish the program
        m=0;
    end
end
```

Figure 22: Function

version 2 code:

```
% Homework1
% Homework introduction: Homework1 plot a shape which like letter L;
clear all
clc

%this section code is to initialize ROS[1]
 rosshutdown
 ipad='localhost';
 rosinit(ipad);

%create a publisher to the velocity topic[2]
 robot = rospublisher('/mobile_base/commands/velocity');

%create a message with the right format to the topic[2]
 velmsg = rosmessage(robot);

 %Assume the first function,which is a circle with (0,0) center and radius is 1
 center_Point = [0,0];
 radius = 1;

 %assume every step's {\delta x}=1
 mx = 1;

 %begin to record the time
 tic

 %The for loop is used to output the first section, and its caculate
 %equation is in calculatevelocity1 function file.
 for x1 = 0:0.001:0.393
    % input every x1 and calculate the slope of each point's differential
    % line segment
    k1=calculatevelocity1(x1);
```

```
33
34          %calculate the each step {\delta x}
35 -        my1 = mx*k1;
36
37          %linear_velocity=\sqrt{\delta x^2+\delta y^2}
38 -        linear_velocity1 = sqrt(my1^2+mx);
39
40          %Attribute the desired values to the message[3]
41 -        velmsg.Linear.X =mx;
42
43          % because the verialble of message only want numberic value, otherwise it will say error happened
44          % I will explain in the report detailed
45 -        z11 = double(linear_velocity1);   %So I change the data type there
46          %Attribute the desired values to the message[3]
47 -        velmsg.Linear.Y =z11;
48
49          %calculate the angle of each point's differential line segment
50 -        angular1 = atan(k1);
51          %The reason is same as above, so change the data type
52 -        z12=double(angular1);
53          %Attribute the desired values to the message[3]
54 -        velmsg.Angular.Z= z12;
55
56          %publish the message
57 -        send(robot,velmsg);
58 -        x1
59 -        z12
60
61 -    end
```

```
63
64 -  for x2 = 0.02:0.001:1
65       % input every x2 and calculate the slope of each point's differential
66       % line segment
67 -     k2 = calculatevelocity2(x2);
68
69       %calculate the each step {\delta x}
70 -     my2 = mx*k2;
71
72       %linear_velocity=\sqrt{\delta x^2+\delta y^2}
73 -     linear_velocity2 = sqrt(my2^2+mx^2);
74
75       %Attribute the desired values to the message[3]
76 -     velmsg.Linear.X = mx;
77       %The same reason,change the data type there
78 -     z21 = double(linear_velocity2);
79       %Attribute the desired values to the message[3]
80 -     velmsg.Linear.Y =z21;
81
82       %The same reason,change the data type there
83 -     angular2 = atan(k2);
84       %Attribute the desired values to the message[3]
85 -     z22=double(angular2);
86
87       %Because sometimes the z22 is negative ,so angule result is not we
88       %want( explain detailed in the report) , so we change the z22 value to positive and
89       %then calculate the angle is want we want. Sometimes, we want z22
90       %negative value ,but it is positive, so we shold also change it to
91       %negative
92       % The most important is: what's the range should we change the positive
```

```
90        %negative value ,but it is positive, so we shold also change it to
91        %negative
92        % The most important is: what's the range should we change the positive
93        % to negative
94
95 -      if z22 <0.5&&z22 >-0.5   %range of x2 is though we observe(explain detailed in the report)
96 -          if x2 >=0.35  %range of x2 is though we observe(explain detailed in the report)
97 -          velmsg.Angular.Z= -z22; %change the value into positive
98 -          else
99            %if not in the range, stay the positive z22 in initial value,and
100           %change the negative one to positive
101 -         velmsg.Angular.Z= abs(z22);
102 -         end
103 -     else
104           % In some cases, the angular velocity is very big, so we proportional reduce the value,
105 -         if x2 >=0.35
106           % The coefficient is also determined by observe(explain detailed in
107           % the report)
108 -         velmsg.Angular.Z= -0.15*z22; %change the value into positive
109 -         else
110           %if not in the range, stay the positive z22 in initial value,and
111           %change the negative one to positive
112 -         velmsg.Angular.Z= abs(0.15*z22);
113 -         end
114 -     end
115
116       %publish the message[4]
117 -     send(robot,velmsg);
118 -     x2
119 -     z22
```

```
120 -   end
121
122     %all the comments is same as the second part
123 -   for x3 = 0.002:0.001:0.4
124 -       k3 = calculatevelocity3(x3);
125
126 -       my3 = mx*k3;
127
128 -       linear_velocity3 = sqrt(my3^2+mx^2);
129
130 -       velmsg.Linear.X = mx;
131
132 -       z31 = double(linear_velocity3);
133 -       velmsg.Linear.Y =z31;
134
135 -       angular3 = atan(k3);
136 -       z32=double(angular3);
137 -       if z32 <0.5&&z32 >-0.5
138 -           if x3 >=0.378
139 -           velmsg.Angular.Z= -z32;
140 -           else
141 -           velmsg.Angular.Z= abs(z32);
142 -           end
143 -       else
144 -           if x3 >=0.378
145 -           velmsg.Angular.Z= -0.1*z32;
146 -           else
147 -           velmsg.Angular.Z= abs(0.1*z32);
148 -           end
149 -       end
```

```
151 -       z32
152 -       send(robot,velmsg);
153 -   end
154
155     %Finished the recording of time and publish the time value
156 -   s = toc;
157 -   fprintf('The total time is %d s',s);
158
159
160     %%reference :
161     % [1] Juliana Vilela, AMR Sections_Matlab+ ROSBasic Concepts, page 11
162     % [2] Juliana Vilela, AMR Sections_Matlab+ ROSBasic Concepts, page 12
163     % [3] Juliana Vilela, AMR Sections_Matlab+ ROSBasic Concepts, page 13
164     % [4] Juliana Vilela, AMR Sections_Matlab+ ROSBasic Concepts, page 13
```

Figure 23: the second version's main file

calculatevelocity1.m

```
function [output1] = calculatevelocity1(x1)
%The function is used to calculate the velocity1
%Assume the first function,which is a circle with (0,0) center and radius is 1
center_Point1 = [0,0];
radius = 0.9;

syms x
f(x) = center_Point1(1,2)-sqrt(radius^2-x^2);
k(x) = diff(f(x));
output1 = k(x1);

end
```

Figure 24: the second version's calculate function

```
function [output2] = calculatevelocity2(x2)
%The function is used to calculate the velocity2
%the part of a specified cos function, which is the range from 0 to $\pi$,
%and reverse the figure on the x axis, and shift to the -y axis direction 3 .
syms x
f(x) = -3*cos(pi*x)-3;
k(x) = diff(f(x));

output2 = k(x2);
end
```

Figure 25: the second version's calculate function

```
function [output3] = calculatevelocity3(x3)
%The function is used to calculate the velocity3
%the part of a complete specified sin function, which is the range from 0 to $2*\pi$
syms x
f(x) = 0.8*sin((4/3)*pi*x)-6;
k(x) = diff(f(x));

output3 = k(x3);
end
```

Figure 26: the second version's calculate function