ELEE 4200

Autonomous Mobile Robots

Homework No:HW4

Position Estimation with Odometry

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

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

Instructor:
Dr. Mohan Krishnan

October 11, 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

The homework goal is to observe the position models for robot localization based on odometry using wheel encoder data in two different ways. Finally, we should compare the two sets of results in order to understand the limitations associated with the odometry approach. [1]

2 Introduction

There are three goals for the homework:
1. Estimate each point on robot trajectory as (x,y,θ); And plotting the trajectory in two ways:
(1)Using the exact circle equation; Additionally, we should first plot the ideal circle of the trajectory. And then, we should add Gaussian noise to Sr and Sl in order to model the real world;
(2)Using the Borenstein approximation. And we should also add Gaussian noise to Sr and Sl in order to model the real world;

2. Comparing the two methods' results and come out the conclusions.

3. Understand the limitations to use the odometry approach. [1]

3 Methodology

In the requirements, the known information includes the following details:

We assume that we use a differentially-steered robot, and L = 0.25m;

The linear velocity is: v= 0.4 m/s ;

The angular velocity is: $\omega = -0.2$ r/s;

And we assume that when we use wheel encoder to read every pace, the pace is 0.05m.

3.1 Drawing the Ideal (circle)

1.Theoretical Background

We know that we should use the (x,y) coordinates value of the robot to plot the trajectory we want. From Professor's introduction and Homework4 guidance, we know that the Ideal (circle) equations are:

$$x(t) = x_0 + L\frac{s_R + s_L}{s_R - s_L}\left[sin\left(\frac{(s_R - s_L)}{2L} + \theta_0\right) - sin(\theta_0)\right]$$

$$y(t) = y_0 - L\frac{s_R + s_L}{s_R - s_L}\left[cos\left(\frac{(s_R - s_L)}{2L} + \theta_0\right) - cos(\theta_0)\right]$$

Figure 2: Ideal (circle) equations [1]

According to the equations, we know that we should remind the content of the lecture, which is From Velocity to Position(Odometry Models);

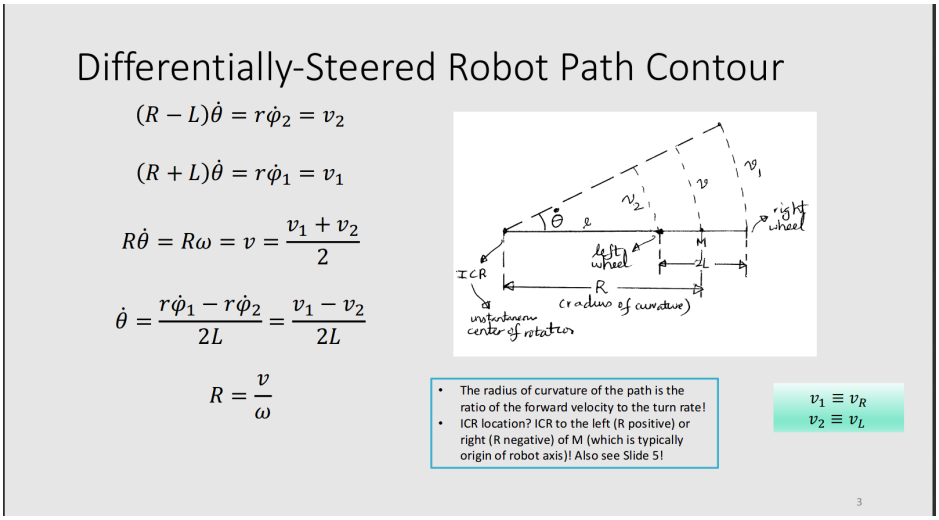


Figure 3: Differentially-Steered Robot Path Contour [2]

So imagine that we connect the right wheel and left wheel as a line, and the center of the line is M. So, as the figure 3 shown, we know that the equation to calculate the center point's velocity is:

$$v = \frac{vL+vR}{2}$$

On the both sides of equation, multiply the δ t, and the result comes out is

$$sM = \frac{sL+sR}{2}$$

And the equation to calculate the angular velocity of the $\dot{\theta}$ is:

$$\dot{\theta} = \frac{vR-vL}{2L}$$

The same theory, multiply the δ t on the both side of the equation, and the result comes out is

$$\theta = \frac{sR-sL}{2L}$$

2.The basic idea

First of all, according to the requirements and the equation $R = \frac{v}{w}$, we can calculate the R of the center point M, which is 2m; And we also know that the half length between the two wheels is L = 0.25m. So the radius of the left wheel is **Rl=R-L=1.75**; and the radius of the right wheel is **Rr=R+L=2.25**.

1. Because the requirements ask us to measure the robot position in every 0.05m, we come up the idea to divide the whole trajectory into

small circular arc pieces, which each piece is 0.05m. And according to the perimeter equation, we can calculate the elements in the whole trajectory. The result is $n = \frac{2*\pi*R}{0.05} \approx 252$.
Therefore, we use the equation that $\theta = \frac{Lab}{R}$,
we can calculate that **each step's change θ is 0.025 rad**. And then, the length for the left wheel's circular arc and the length for the right wheel's circular arc can be easily calculate by the equation $Lab = \theta * R$. And the result is that: In each step, the change of the left wheel's trajectory is **sL = θ*(R-L)=0.0438**;the change of the right wheel's trajectory is **sL = θ*(R+L)=0.0563**.
And because we need to calculate each step's (x,y), and we need to use each step's angle change and sRi with sLi, we create three arrays(which are sR,sL,and theta)for the three elements in order to use it directly in the later calculation, and the length of each array is all equal to n which we calculate at the first step.

2. We assume that the initial position of the robot [x,y,direction]=[0,0,0]; And we create four arrays for x coordinate which name is X,Y coordinate which name is Y,direction angle, which name is theta, and the changing direction angle for each step which name is deltheta. we can store the initial position information in the first element for X,Y and theta; Specifically, X[1,1]=0;Y[1,1]=0,theta[1,1]=0; And according to the first step's calculate, the change of the first step's angle is θ = 0.025 rad, so the first element of the array deltheta is 0.025 rad.

3.According to the equation, every step's position is depended by the previous step. So we should use for loop or while loop to calculate each step's position. And because the first element of the crucial arrays are all known(which are X,Y,sR,sL,theta and deltheta), we can use the Ideal(circle) equations to calculate the coordinates of (x,y).

4.After the for loop calculation(we use the for loop), all the arrays have been fulled with each step's position information(including X,Y,sR,sL,theta and deltheta). So, we can use plot function to draw the ideal circle.

3.2 Drawing the Ideal (circle) with adding noise

1.Theoretical Background

The noise can be created by random function: for example, if we want an array which its range is [c,d], and c is a negative number, and d is a positive number. We can create the noise array by the command above:

noise = (d-c)*rand(row,column)+(c+d)/2 ;

On the other hand, there is another function which we can gain the noise array directly without calculating, it is the command:

noise = normrnd(mean,σ,[row,column]) ;

mean:the mean for the array which ruled by ourselves;
σ:standard difference for the array which ruled by ourselves

2.The basic idea

The whole idea is same as the first section, but when we calculate all the arrays(including X,Y,sR,sL,theta and deltheta). There is an additional step before plot the figure is that we should add noise on both sL and sR noise with different noise array.

But we should pay attention that, in the requirements, it also has the principle that the noise should be in the range from [-0.01,0.01]. So after creating two noise array, we should find the elements which not satisfied the requirements in the array.If the elements higher than 0.01, replacing then by 0.01; If the elements lower than -0.01, replacing then by -0.01;

The specific code is like above figure 4:

```
noise1 = normrnd(0,0.005,[1,n]); %creat the noise array for left wheel
noise1(find(noise1>0.01))=0.01; % judge the noise's elements if higher than 0.01,replace it with 0.01
noise1(find(noise1<-0.01))=-0.01; % judge the noise's elements if lower than 0.01,replace it with -0.01

noise2 = normrnd(0,0.005,[1,n]);  %creat the noise array for right wheel
noise2(find(noise2>0.01))=0.01; % judge the noise's elements if higher than 0.01,replace it with 0.01
noise2(find(noise2<-0.01))=-0.01; % judge the noise's elements if lower than 0.01,replace it with -0.01

Sl2 = Sl2 + noise1; % add the noise on the left wheel
Sr2 = Sr2 + noise2; % add the noise on the right wheel
```

Figure 4: noise creating section

After adding noise, using plot function to draw the circle with adding noise by using the Ideal(circle) equations.

3.3 Borenstein approximation:

1.Theoretical Background
The equations of Borenstein approximation are:

$$x_i = s_M cos(\theta) + x_{i-1};\ \theta: pose\ angle$$

$$y_i = s_M sin(\theta) + y_{i-1};\ \theta: pose\ angle$$

Figure 5: Borenstein approximation equation

2.The basic idea
The same idea as the first section circle, but this time we should use the different equations to draw the circles and we also add noise on both sides wheels. The different parts is :Adding two noise arrays on the sL and sR array, and then take the values of array sL, sR , theta and deltheta into the equations to get the array X and array Y.
Then, plot the figure to get the circle we want.

4 Results

The result figures are locates in the appendix. The result figures are satisfying all the requirements.

5 Discussions and Conclusion

Conclusion and discussion
Comparing the second and the third circle, although we add same noise on the same wheels, the results figure a little bit difference, so we guess that the equation might cause the differences on the figure.

Problem:
One of problem we met in this homework is that in the noise circle, we use the ideal theta. In this condition, the theta is not influenced by the noise. So the circle we get is more "put in order" (as the figure show below). After the guidance from TA, we deal with the problem. The theta is influenced by the noise on SR and SL. After change the theta that calculated by SR and SL, we get the right result.

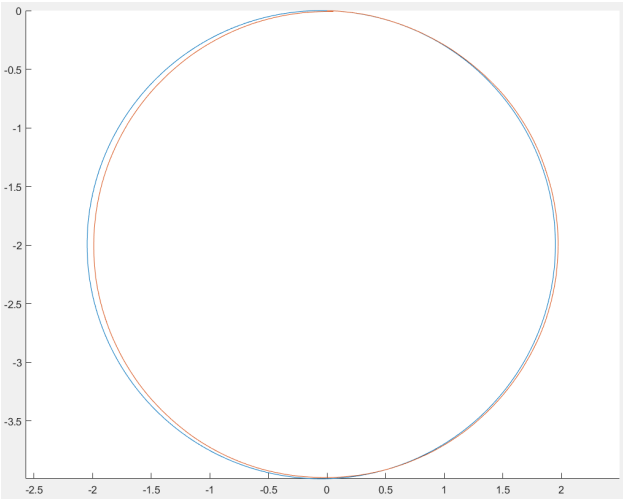


Figure 6: noise circle with ideal theta

References

[1] *Homework 4, Position Estimation with Odometry.*

[2] *power points slides page3,From Velocity to Position (Odometry Models).*

A Appendix

The following are our matlab mile and result's figures.

A.1 List of attached files

1. MATLAB file version1

Contents

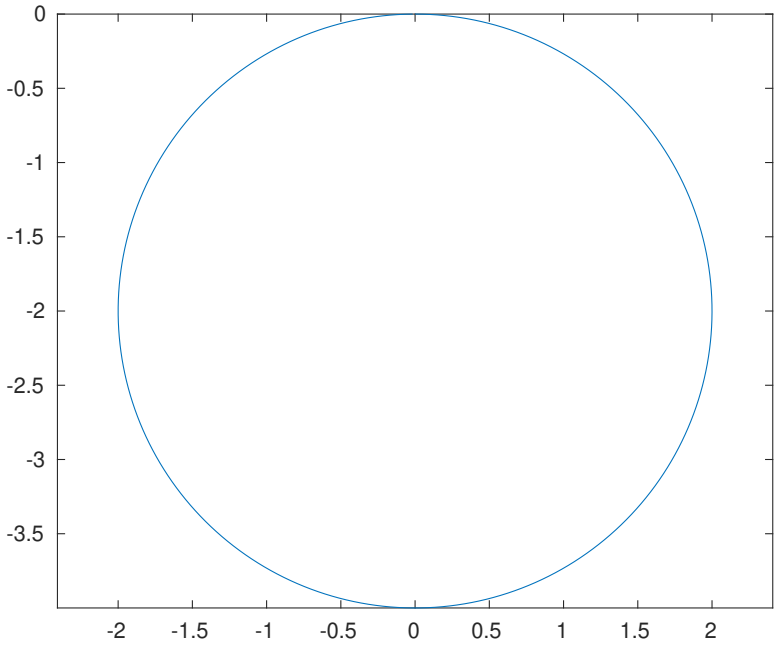
- Ideal(circle) equations:
- adding Gaussian noise
- Borenstein approximation

Ideal(circle) equations:

```
clear all
clc

L = 0.25; % The distance between two wheel is 2L = 0.5
v = 0.4; % The velocity of the robot is 0.4m/s
w = -0.2; % The angular velocity of the robot is -0.2r/s

X0 = 0; % assume that the initial point is (0,0);
Y0 = 0;
theta0 = 0;
% assume that the initial direction is point to the positive x axis;

R = abs(v/w); % calculate the radius of the path
Rl = R-L; % The left wheel's radius
Rr = R+L; % The right wheel's radiu

pace = 0.05; %every step is 0.05m
n = ceil((2*pi*R)/pace); %calculate the element number for the trajectory
theta = pace/R; % calculate the first rotate angle for the robot

Sl1 = zeros(1,n); %creat an array for the left wheel's each step movement
```

```
Sr1 = zeros(1,n); %creat an array for the right wheel's each step movement
Sm1 = zeros(1,n); %creat an array for the center points of the two wheels
deltheta1 = zeros(1,n); %creat an array for each step's changing angle

theta1 = zeros(1,n); %creat an array for each step's direction angle
theta1(1,1)= theta0; %initialize the first element

X1 = zeros(1,n);% creat an array for the x
Y1 = zeros(1,n);% creat an array for the y
X1(1,1)=X0; %initialze the first element as x0
Y1(1,1)=Y0; %initialze the first element as y0

for i1 = 1:n
% calculate the left wheel and right wheel movements at each step
    Sl1(1,i1) = Rl*theta;
    Sr1(1,i1) = Rr*theta;
end

for k1 = 1:n % calculate the Sm and the change of each step's theta
    Sm1(1,k1) = (Sl1(1,k1)+Sr1(1,k1))/2;
    deltheta1(1,k1) = -[(Sr1(1,k1)-Sl1(1,k1))/(2*L)];
    theta1(1,k1+1)=theta1(1,k1)+deltheta1(1,k1);
end

A = zeros(1,n);

for m1=1:n %calculate the coefficient
    A(1,m1)=Sm1(1,m1)/deltheta1(1,m1);
end

for j1= 2:n %calculate the coordinate of every segment
X1(1,j1)=X1(1,j1-1)+A(1,j1-1)*(sin(deltheta1(1,j1-1)+theta1(1,j1-1))
-sin(theta1(1,j1-1)));
Y1(1,j1)=Y1(1,j1-1)-A(1,j1-1)*(cos(deltheta1(1,j1-1)+theta1(1,j1-1))
-cos(theta1(1,j1-1)));
end
```

```
plot(X1,Y1);
axis equal
```

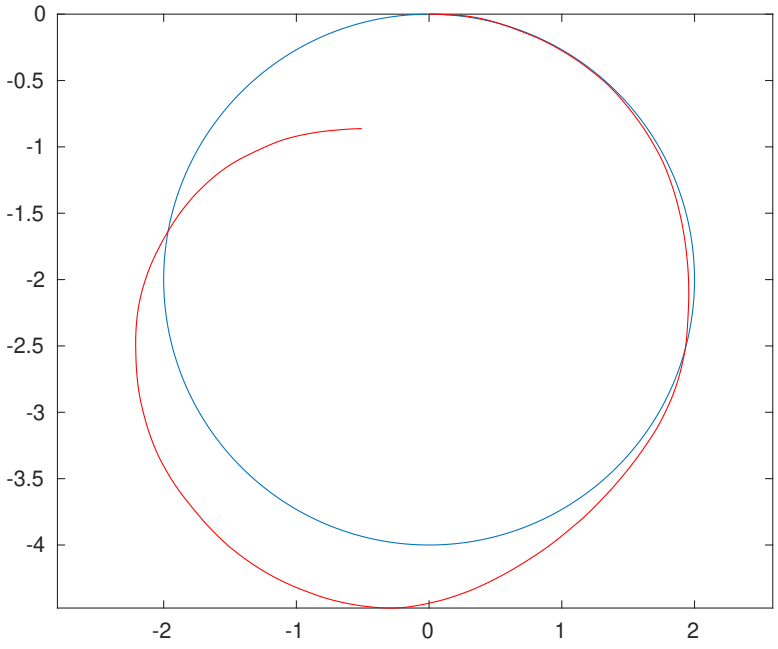


adding Gaussian noise

```
Sl2 = zeros(1,n); %creat an array for the left wheel's each step movement
Sr2 = zeros(1,n); %creat an array for the right wheel's each step movement
Sm2 = zeros(1,n); %creat an array for the center points of the two wheels
deltheta2 = zeros(1,n); %creat an array for each step's changing angle

theta2 = zeros(1,n); %creat an array for each step's direction angle
theta2(1,1)= theta0; %initialize the first element

X2 = zeros(1,n); % creat an array for the x
Y2 = zeros(1,n); % creat an array for the y
X2(1,1)=X0; %initialze the first element as x0
Y2(1,1)=Y0; %initialze the first element as y0

for i2 = 1:n
```

```
% calculate the left wheel and right wheel movements at each step
    Sl2(1,i2) = Rl*theta;
    Sr2(1,i2) = Rr*theta;
end

noise1 = normrnd(0,0.005,[1,n]);
%creat the noise array for left wheel
noise1(find(noise1>0.01))=0.01;
% judge the noise's elements if higher than 0.01,replace it with 0.01
noise1(find(noise1<-0.01))=-0.01;
% judge the noise's elements if lower than 0.01,replace it with -0.01

noise2 = normrnd(0,0.005,[1,n]);
%creat the noise array for right wheel
noise2(find(noise2>0.01))=0.01;
% judge the noise's elements if higher than 0.01,replace it with 0.01
noise2(find(noise2<-0.01))=-0.01;
% judge the noise's elements if lower than 0.01,replace it with -0.01

Sl2 = Sl2 + noise1; % add the noise on the left wheel
Sr2 = Sr2 + noise2; % add the noise on the right wheel

for k2 = 1:n % calculate the Sm and the change of each step's theta
    Sm2(1,k2) = (Sl2(1,k2)+Sr2(1,k2))/2;
    deltheta2(1,k2) = -[(Sr2(1,k2)-Sl2(1,k2))/(2*L)];
    theta2(1,k2+1)=theta2(1,k2)+deltheta2(1,k2);
end

B = zeros(1,n);

for m2=1:n %calculate the coefficient
    B(1,m2)=Sm2(1,m2)/deltheta2(1,m2);
end

for j2= 2:n %calculate the coordinate of every segment
X2(1,j2) = X2(1,j2-1) + B(1,j2-1)*(sin(deltheta2(1,j2-1)+theta2(1,j2-1))
-sin(theta2(1,j2-1)));
Y2(1,j2) = Y2(1,j2-1) - B(1,j2-1)*(cos(deltheta2(1,j2-1)+theta2(1,j2-1))
```

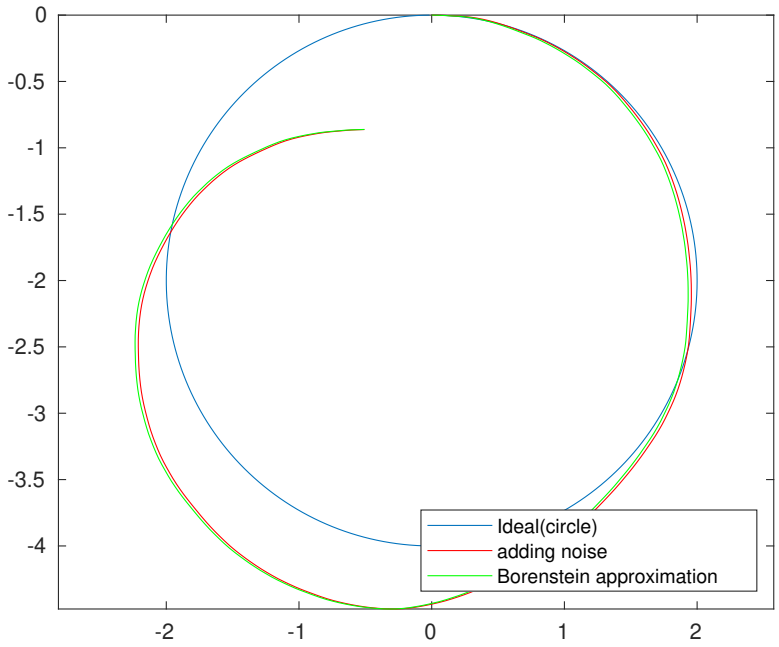
```
-cos(theta2(1,j2-1)));
end

hold on
plot(X2,Y2,'r');
axis equal
```



Borenstein approximation

```
Sl3 = zeros(1,n); %creat an array for the left wheel's each step movement
Sr3 = zeros(1,n); %creat an array for the right wheel's each step movement
Sm3 = zeros(1,n); %creat an array for the center points of the two wheels
deltheta3 = zeros(1,n); %creat an array for each step's changing angle

theta3= zeros(1,n); %creat an array for each step's direction angle
theta3(1,1)= theta0; %initialize the first element

X3 = zeros(1,n); % creat an array for the x
Y3 = zeros(1,n); % creat an array for the y
```

```
X3(1,1)=X0; %initialze the first element as x0
Y3(1,1)=Y0; %initialze the first element as y0

for i3= 1:n
% calculate the left wheel and right wheel movements at each step
    Sl3(1,i3) = Rl*theta;
    Sr3(1,i3) = Rr*theta;
end

Sl3 = Sl3 + noise1; % add the noise on the left wheel
Sr3 = Sr3 + noise2; % add the noise on the right wheel

for k3 = 1:n % calculate the Sm and the change of each step's theta
    Sm3(1,k3) = (Sl3(1,k3)+Sr3(1,k3))/2;
    deltheta3(1,k3) = -[(Sr3(1,k3)-Sl3(1,k3))/(2*L)];
    theta3(1,k3+1)=theta3(1,k3)+deltheta3(1,k3);
end

for j3 = 2:n
%use Borenstein approximation to calculate the coordinate of every segments
X3(1,j3) = X3(1,j3-1) + Sm3(1,j3)*cos(theta3(1,j3));
Y3(1,j3) = Y3(1,j3-1) + Sm3(1,j3)*sin(theta3(1,j3));
end

hold on
plot(X3,Y3,'g');

legend('Ideal(circle)','adding noise ',
'Borenstein approximation','location','southeast');
```

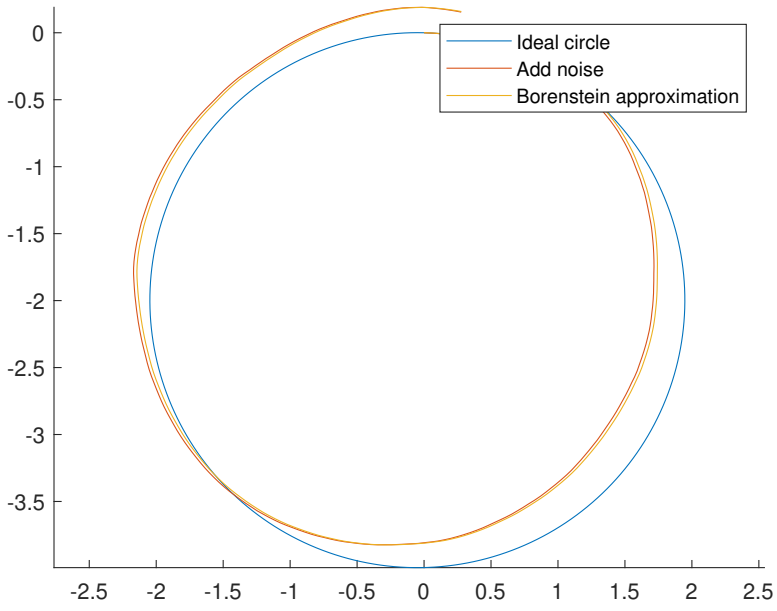


2. MATLAB file version2

```
clc
clear all
v=0.4;%robot's speed is 0.4m/s
w=-0.2;%robot's rotation speed is -0.2 rad/s
L=0.25;%the longth of robot is 2L,0.5m
reading_number=size(0:0.05:2*pi*abs(v/w));%get the number of point
(each point apart 0.05)
cita_change=zeros(1,reading_number(2));%build a  matrix for the ideal theta
cita=-(2*pi)/(reading_number(2)-1);%the ideal theta

syms SR SL;%figure out SL and SR
eq1=(SR-SL)/(2*L)==cita;
eq2=(SR+SL)/2==0.05;
[SR,SL]=solve([eq1,eq2],[SR,SL]);%get SL and SR

SR=double(SR);% change sym to int ,in oder to caculate
SL=double(SL);% change sym to int ,in oder to caculate
x_y(1,1)=0;% initial point ,X
x_y(1,2)=0;% initial point ,Y
```

```
cita_change(1)=cita;
for i=2:1:reading_number(2)
   cita_change(i)=cita_change(i-1)+cita; % the ideal theta for every point
end
hold on
for i=2:1:reading_number(2) %caculate the position of each point
    x_y(i,1)=x_y(i-1,1)+L*((SR+SL)/(SR-SL))*(sin(cita_change(i-1)+((SR-SL)
    /(2*L)))-sin(cita_change(i-1)));
    x_y(i,2)=x_y(i-1,2)-L*((SR+SL)/(SR-SL))*(cos(cita_change(i-1)+((SR-SL)
    /(2*L)))-cos(cita_change(i-1)));
end
plot(x_y(:,1),x_y(:,2))%draw the circle
axis equal
noise_x_y=zeros(reading_number(2),2);%build a matrix for noise circle
noise_x_y(1,1)=0;% initial point ,X
noise_x_y(1,2)=0;% initial point ,Y
random_L=normrnd(0,sqrt(0.000025),1,reading_number(2));
% use normrnd to get radome number
random_R=normrnd(0,sqrt(0.000025),1,reading_number(2));
% use normrnd to get radome number
random_L(random_L>0.01)=0.01;% limit the range of noise
random_L(random_L<-0.01)=-0.01;% limit the range of noise
random_R(random_R>0.01)=0.01;% limit the range of noise
random_R(random_R<-0.01)=-0.01;% limit the range of noise
cita_noise_change(1)=0;% initial the theta include noise
for i=2:1:reading_number(2)%the position of the circle with noise
    random_SR=SR+random_R(i);%get the SR with noise
    random_SL=SL+random_L(i);%get the SL with noise
    cita_noise=(random_SR-random_SL)/(2*L);
    %caculate the theta with noise
    cita_noise_change(i)=cita_noise_change(i-1)+cita_noise;
    %caculate the theta with noise
    noise_x_y(i,1)= noise_x_y(i-1,1)+L*((random_SR+random_SL)/
    (random_SR-random_SL))*(sin(cita_noise_change(i-1)+
    ((random_SR-random_SL)/(2*L)))-sin(cita_noise_change(i-1)));
    noise_x_y(i,2)= noise_x_y(i-1,2)-L*((random_SR+random_SL)/
    (random_SR-random_SL))*(cos(cita_noise_change(i-1)+
    ((random_SR-random_SL)/(2*L)))-cos(cita_noise_change(i-1)));
```

```
end
plot(noise_x_y(:,1),noise_x_y(:,2));%draw the noise circle
B_x_y=zeros(reading_number(2),2);% build a matrix for Borenstein circle
B_x_y(1,1)=0;% initial point ,X
B_x_y(1,2)=0;% initial point ,Y
cita_noise_change(i)=0;%initial the theta include noise
for i=2:1:reading_number(2)%the position of the Borenstein circle with noise
    random_SR=SR+random_R(i);
    %get the SR with noise, the namuber same as noise circle
    random_SL=SL+random_L(i);
    %get the SR with noise, the namuber same as noise circle
    cita_noise=(random_SR-random_SL)/(2*L);
    %caculate the theta with noise, the namuber same as noise circle
    cita_noise_change(i)=cita_noise_change(i-1)+cita_noise;
   %caculate the theta with noise, the namuber same as noise circle
    B_x_y(i,1)=((random_SL+random_SR)/2)
    *cos(cita_noise_change(i-1))+B_x_y(i-1,1);
    B_x_y(i,2)=((random_SL+random_SR)/2)
    *sin(cita_noise_change(i-1))+B_x_y(i-1,2);
end
plot(B_x_y(:,1),B_x_y(:,2))%draw  Borenstein circle
legend('Ideal circle','Add noise','Borenstein approximation')%add legend
```


Ideal circle
Add noise
Borenstein approximation
0
-0.5
-1
-1.5
-2
-2.5
-3
-3.5
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5