# ME 425 Post Lab Report 3

Demir Demirel, Yunus Emre Madak

***Abstract*— This report include implementation of motion control of Two Wheeled Robot made with Lego Mindstorms$^{TM}$ EV3 and discussion on the test results gathered with MATLAB.**

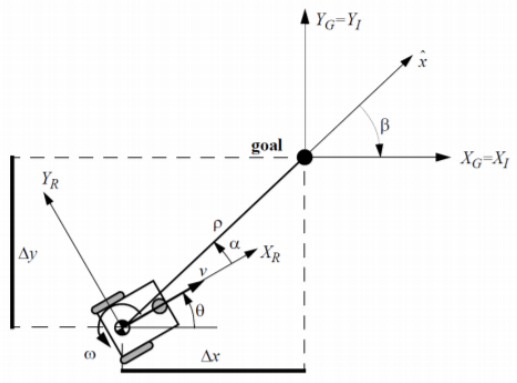
## I. INTRODUCTION

In this lab we were expected to implement the positioning controller derived in the lecture. To conduct we construct a two-wheeled robot by using the building instructions given in Lab2 document. Assuming that the odometry estimations provide the position and the orientation of the robot accurate enough to be work them in our controller. We implement differential-drive robot model and collect position and orientation data during the motion of the robot. We assumed that ;x(0) = 0, y(0) = 0 and $\theta(0) = 0$.

## II. PROCEDURE

In this lab, we have experienced positioning control of differential-drive robot model. For this purpose we use some of the previous labs objectives. For instance odometry estimation is used for the getting know the robots current position and trajectory on the MATLAB environment. The main problem to be encountered is we work on the polar coordinate system while controlling. Therefor the output should be in the cartesian for showing the trajectory. The figure II shows us what would be the values of polar coordinates with the cartesian frame. As it is seen in the figure there is a

Fig. 1. Polar Coordinate Transformation




triangle shape occur between x and y axis and the values of the $\rho$ could be determined with that relation. As Pythagorean Theorem stated, $\rho$ is hypotenuse and $\Delta x$ and $\Delta y$ would be the other edges. From that relation we can get the formula as follows:

$$\rho = \sqrt{\Delta x^2 + \Delta y^2} \tag{1}$$

$\alpha$ is determined with simple trigonometry as $\arctan(y/x)$. But arctan function has a major problem which affect the stable control of the robot. It only gives results for $1^{st}$ and $4^{th}$ quadrants. Therefore we need to use $atan2$ for enable us to calculate $1^{st}$ ,$2^{nd}$,$3^{rd}$ and $4^{th}$ quadrants. As seen in the figure when we apply $atan2(\Delta y, \Delta x)$ gives us the angle of $-beta$. Which is equal to $\alpha + \theta$ . With that intuition we can find out the equations below:

$$\begin{aligned} \alpha &= -\theta + atan2(\Delta y, \Delta x) \\ \beta &= -\theta - \alpha \end{aligned} \tag{2}$$

All the given formulas has a common $\Delta x$ and $\Delta y$ values. Which are basically the distance between goal point and robot's centre of motion in both x, y axis. And it could be obtain by the $\Delta x = x_G - x$ and $\Delta y = y_G - y$.
We get the given values and manipulate them as the form that we need in the paragraphs above. When we come to the control the robot as our target we need to derive a control law. As we working on the robot, movement of the robot could be done with only controlling the velocity and angular velocity. So we need to control v and $\omega$ with the known data which are the position of the robot. In that time the position is not determined by cartesian coordinate system so we were used polar variables $\rho, \alpha, \beta$. When we apply proportional control over these variables the formulas below appears:

$$\begin{aligned} v &= k_\rho \rho \\ \omega &= k_\alpha \alpha + k_\beta \beta \end{aligned} \tag{3}$$

After we get the values of v and $\omega$ we can control both system and observe the trajectory in the cartesian plane with the experience that we have from previous lab assignment; odometry estimation.
When we done with the odometry estimation we can observe the trajectory in MATLAB environment. All of the data that we are going to talk about in the Results section were gathered with the calculations above.

## III. RESULT

In this lab we observed the trajectory of the two wheeled robot with MATLAB plots. We initialize the different initial positions. For instance In the Figure :2 we try to plot 3 different trajectory. Initial positions represent 3 different quadrant which is very convenient to show that atan2 function is working. Even if the initial position in the $2^{nd}$ and $3^{rd}$ quadrant, our robot tend to go origin. Only the trajectory start from $2^{nd}$ quadrant goes to origin. The other two was not goes directly to origin. The reason for this could be the control gains. Even if the controller gains are same the final points are not same. The reason could be it controls only proportional, if it controlled by PID controller

it would be more accurate result. When we look at the

Fig. 2. Initial positions at $1^{st}, 2^{nd}, 3^{rd}$ Quadrants

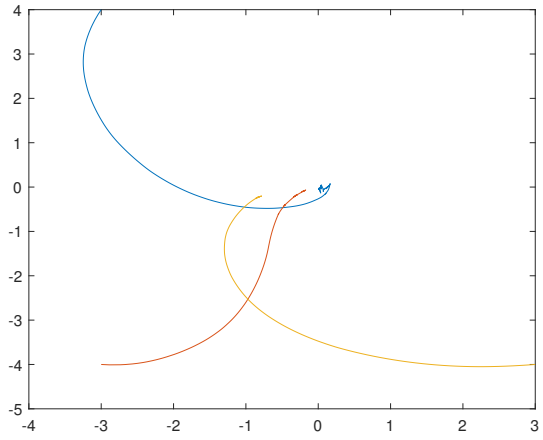


Fig. 3. Initial positions at $1^{st}, 2^{nd}, 3^{rd}$ Quadrants Rho

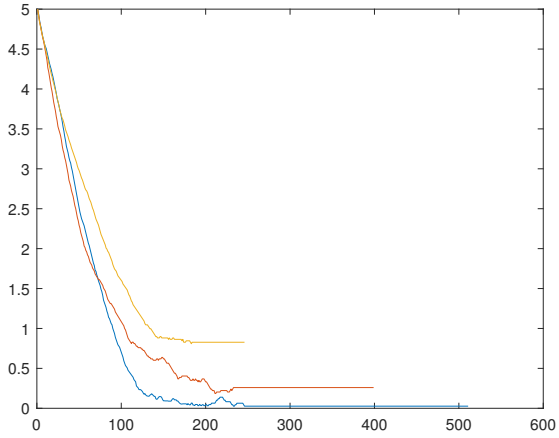


Fig. 4. Initial positions at $1^{st}, 2^{nd}, 3^{rd}$ Quadrants Theta

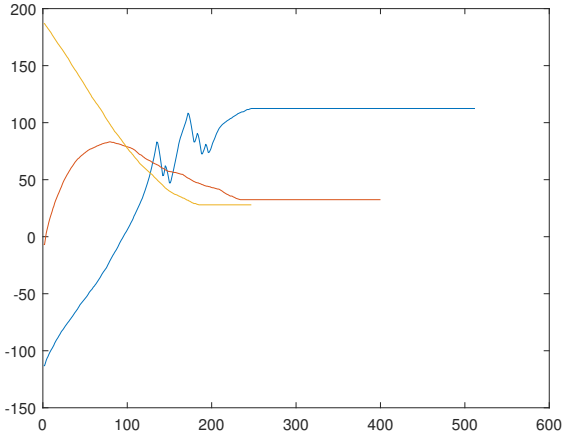


Figure :3 we can easily see the settling points which are very distinct. Blue line goes to 0 as we expect in both graphs and we can see the settling point is very accurate. But when we look at the orange lines the robot stops a bit far from the origin. With comparison of blue and orange line we can understand that it stops at settling point. Also in yellow line there is a undeniable truth since the difference between goal point is very distinct.

### A. CONCLUSION

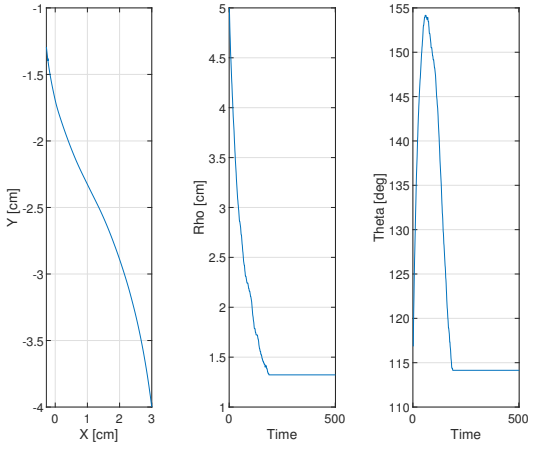
The two wheeled differential drive robot experimental platform LEGO MINDSTORMS EV3 is described in this lab report. The data gathered with odometry estimation from actuators, sensors and control hardware are reviewed in terms of graphs that represent location of the robot, Rho and Theta values. The theoretical expectations was not satisfied well. Applied control technique is not capable of give the stable results. There are mechanical constraints which prevent the accurate motion like friction and stiction.

## IV. INDIVIDUALS

### A. Demir Demirel

In this lab we are asked to do some test with the positioning control algorithms. About the positioning control it is a part of parking problem. Humans parks the cars very accurate but when we come to the autonomous robots it would be not easy as we did. The planning of the route is not really done with some via points as its expected. We plan route only with the error between the goal point. We apply control algorithm as writen in the equation (3). From the equation; $\rho$, $\alpha$ and $\beta$ are calculated from the feedback signal. $k_\rho$, $k_\alpha$ and $k_\beta$ are decided by users. We plot some graphs with different gains for compare the effect of the gains. When we compare the these two position graphs

Fig. 5. $\rho$ = 2 , $\alpha$=5 and $\beta$=-1.5




we can easily said that in figure:5 the control constants were not able to drive the robot to the origin, it settled around (-0.2,-1.3) point but in Figure : 6 the robot tend to go origin. When we pass to the Rho graphs it is obvious that the

Fig. 6. $\rho$ = 5 , $\alpha$=10 and $\beta$=-3

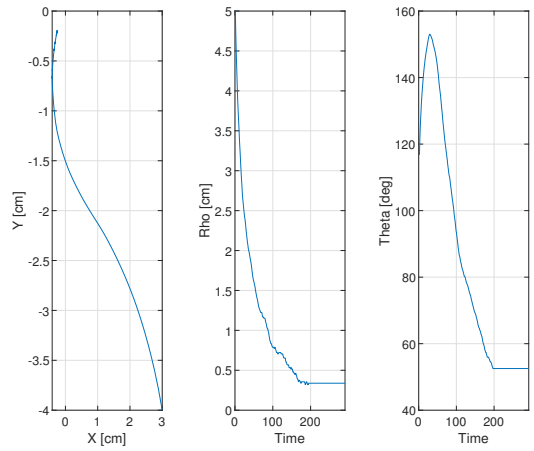
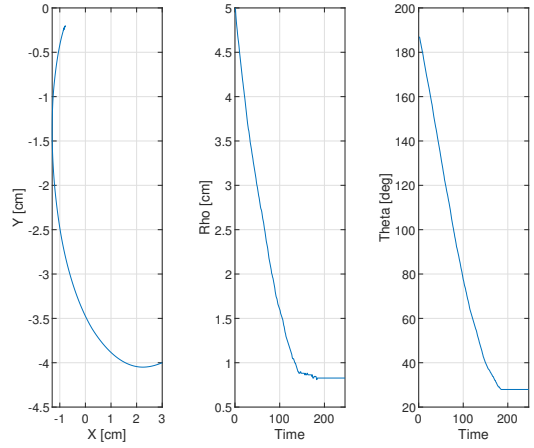



Figure: 5 has settled around 1.3 cm far from origin while Figure: 6 settled around 0.4 cm from origin. So we can say that changing the gains makes the system more accurate in the frame of expectations. And we can determine gains with observing the data.

For the given figures above, we can see the effect of initial position for same initial positions in the condition of $\alpha < (\frac{-pi}{2}, \frac{pi}{2}$. The Figure:7 has initially 180 degree of theta with respect to global origin. For the Figure:8 we start from 130 degree. As we can observe resultant trajectory, and both the graphs goes to the origin point. When we look at the Rho plots, both of the figures has nearly same settling time and it is very similar in Rho plot as well.

Fig. 7. (3,-4) $\alpha$ = -60




When we discuss on the Figure:9 and 10 the initial $\alpha$ values are different. The trajectories are not same but the Rho graph is very similar in both figure. In the Figure 9 the trajectory is much shorter than the trajectory in Figure 10. And when we compare the settling time Figure 9 settles at 170 second while the other was settled in 200 second.

As a result we compare some of the graphs. We applied

Fig. 8. (3,-4) $\alpha$ = 60

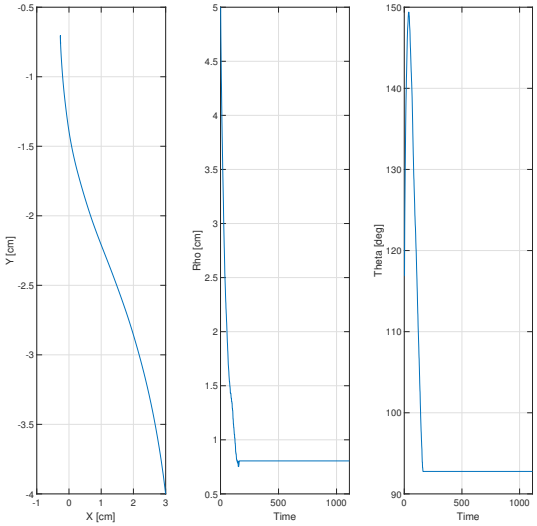



Fig. 9. (-5,-5) $\alpha$ = -40

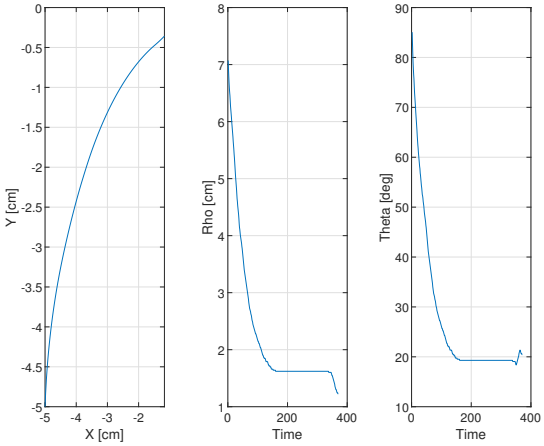
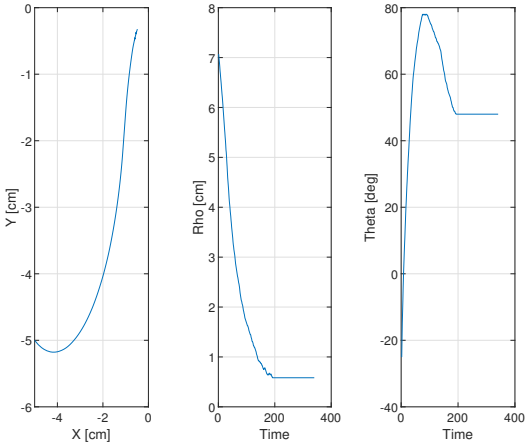



Fig. 10. (-5,-5) $\alpha$ = 70



some control algorithms but these results could be done in better way with a distinct sophisticated control algorithms. For example we can apply different control method when the Rho is less then 2. Or used PI controller for the better solutions.

### B. Yunus Emre Madak

## APPENDIX

Listing 1. main.m file

```
clear all;
close all;
clc;
%% Lego Definitions:
mylego = legoev3('usb'); % Connect lego CPU ...
    via USB.
beep(mylego) % Verify the connection.
motorLeft = motor(mylego,'B'); % Left motor ...
    connection
motorRight = motor(mylego,'C'); % Right ...
    motor connection
motorLeft.Speed = 0; % Zero initial speed
motorRight.Speed = 0; % Zero initial speed
start(motorLeft) % Start left motor
start(motorRight) % Start right motor
%% Parameters:
ΔT = 0.01; % Period
R = 4.32/2; % Wheel radius [cm]
L = 9.5; % Distance between wheels in [cm]
%% Initializations:
initialization % Your initialization file
%% Main Loop:
k=1;
while¬readButton(mylego, 'up')
% ----------------------------
odometryEstimation % Your odometry ...
    estimation file
myController % Your controller file
% ----------------------------
subplot(1,3,1)
plot(X,Y)
xlabel('X [cm]'); ylabel('Y [cm]'); grid on;
subplot(1,3,2)
plot(Rho)
xlabel('Time'); ylabel('Rho [cm]'); grid on;
subplot(1,3,3)
plot(Theta*(180/pi))
xlabel('Time'); ylabel('Theta [deg]'); grid on;
end
stop(motorLeft)
stop(motorRight)
```

Listing 2. initialization.m file

```
X=0;
Y=0;
alpha1=180;
alpha1=degtorad(alpha1);
Theta=atan2(-Y,-X)-alpha1;

krho =3;
kalpha=8;
kbeta = -2;

Rho=[];
```

Listing 3. myController.m file

```
delX = 0 - X(k);
delY = 0 - Y(k);
rho = sqrt(delX*delX + delY*delY);
alpha = - Theta(k) + atan2(delY,delX);
beta = -Theta(k) - alpha;
Rho=[Rho rho];
v=krho*rho;
w= kalpha*alpha + kbeta*beta ;
vR = (2*v + w*L)/2;
vL = (2*v - w*L)/2;

motorLeft.Speed = vL;
motorRight.Speed = vR;
```

Listing 4. odometryEstimation.m file

```
resetRotation(motorLeft);
resetRotation(motorRight);

pause(ΔT);

el = double(readRotation(motorLeft));
er = double(readRotation(motorRight));

Dr= (2*pi*R*er)/360;
Dl= (2*pi*R*el)/360;
Dc= (Dr+Dl)/2;
ΔTheta= (Dr-Dl)/L;

X(k+1) = X(k) + Dc*cos(Theta(k)+ΔTheta/2);
Y(k+1) = Y(k) + Dc*sin(Theta(k)+ΔTheta/2);
Theta(k+1) = Theta(k) + ΔTheta;

k=k+1;
```