باسمه تعالی

دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر

اصول علم ربات

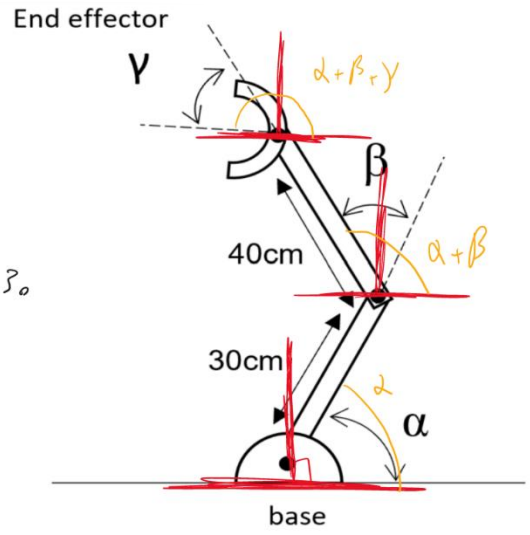
تمرین سری دوم

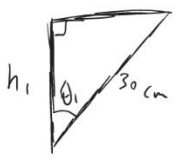
محمد جواد رجبی

9831025

بخش تئوری

سوال اول

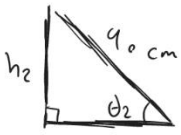




$\theta_1 = 90 - \alpha$

$\Rightarrow h_1 = \cos(\theta_1) \cdot 30\,cm$

$\Rightarrow h_1 = \cos(90-\alpha) \cdot 30 \xrightarrow{*} h_1 = \sin(\alpha) \cdot 30$



$\theta_2 = 180 - \alpha - \beta$, $h_2 = \sin(\theta_2) \cdot 90\,cm$

$\Rightarrow h_2 = \sin(180 - \alpha - \beta) \cdot 90$

$\Rightarrow h_2 = \sin(\alpha+\beta) \cdot 90$

$y_e = y_0 + h_1 + h_2$

$$\Rightarrow y_e = y_0 + 30 \cdot \sin(\alpha) + 90 \cdot \sin(\alpha+\beta)$$

$x_e = x_0 + l_1 - l_2$

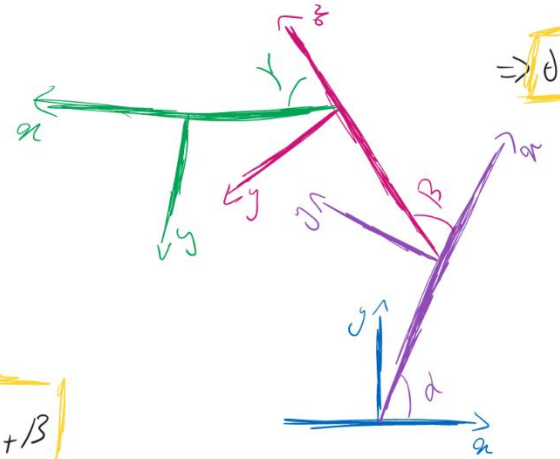
$l_1 = \sin(\theta_1) \cdot 30 = \sin(90-\alpha) \cdot 30$

$\Rightarrow l_1 = \cos(\alpha) \cdot 30$

$l_2 = \cos(\theta_2) \cdot 90 = \cos(180 - \alpha - \beta) \cdot 90$

$\Rightarrow l_2 = -\cos(\alpha+\beta) \cdot 90$

$$\Rightarrow x_e = x_0 + 30 \cdot \cos(\alpha) + 90 \cdot \cos(\alpha+\beta)$$



$\Rightarrow \theta = \alpha + \beta + \gamma$

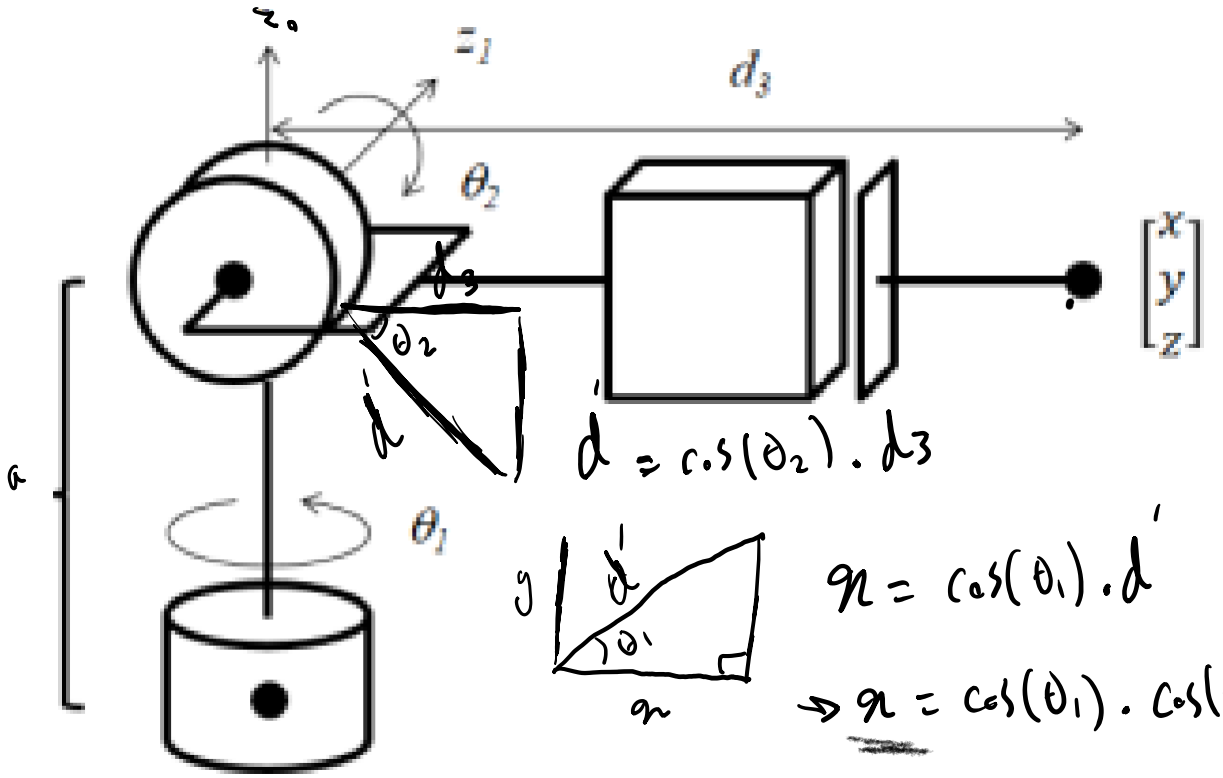
سوال دوم

$${}^{b}P = R_{ab} \cdot {}^{a}P \quad , \quad \underset{3\times3}{R_{ab}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \quad , \quad \underset{3\times1}{{}^{a}P} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

$$\Rightarrow {}^{b}P = \begin{bmatrix} 1 \\ -3 \\ 2 \end{bmatrix}$$

سوال سوم

- forward :



$d = \cos(\theta_2) \cdot d_3$

$x = \cos(\theta_1) \cdot d'$

$\Rightarrow x = \cos(\theta_1) \cdot \cos(\theta_2) \cdot d_3$

$y = \sin(\theta_1) \cdot \cos(\theta_2) \cdot d_3$ ← y به همین شکل برای

$z = a - \sin(\theta_2) \cdot d_3$ و z داریم

- inverse :

$* \; x = d_3 \cdot \cos(\theta_1) \cdot \cos(\theta_2)$ (1)

$* \; y = d_3 \cdot \sin(\theta_1) \cdot \cos(\theta_2)$ (2)

$* \; z = a - d_3 \cdot \sin(\theta_2)$ (3)

تقسیم رابطه ② بر ①

$$\Rightarrow \frac{y}{x} = \frac{d_3 \cdot \sin(\theta_1) \cdot \cos(\theta_2)}{d_3 \cdot \cos(\theta_1) \cdot \cos(\theta_2)}$$

$$\Rightarrow \frac{y}{x} = \frac{\sin(\theta_1)}{\cos(\theta_1)} = \tan(\theta_1) \Rightarrow \theta_1 = \operatorname{atan2}(y, x)$$

توان 2 رابطه‌های ①، ② و ③ را با هم جمع می‌کنیم $\Rightarrow$

$$x^2 = d_3^2 \cdot \cos^2(\theta_2) \cdot \cos^2(\theta_1)$$

$$y^2 = d_3^2 \cdot \cos^2(\theta_2) \cdot \sin^2(\theta_1)$$

$$\Rightarrow x^2 + y^2 = (d_3 \cdot \cos(\theta_2))^2 \cdot \underbrace{(\sin^2(\theta_2) + \cos^2(\theta))}_{1}$$

$$\Rightarrow d_3 \cdot \cos(\theta_2) = \sqrt{x^2 + y^2}$$

رابطه ③ $\quad d_3 \cdot \sin(\theta_2) = a - 2$

$$\Rightarrow \tan(\theta_2) = \frac{a-2}{\sqrt{x^2+y^2}}$$

$$\Rightarrow \underline{\theta_2} = \operatorname{atan2}(a-2, \sqrt{x^2+y^2})$$

$$x^2 + y^2 = d_3^2 \cdot \cos^2(\theta_2)$$

توان 2 رابطه ③ $\Rightarrow (a-2)^2 = d_3^2 \cdot \sin^2(\theta_2)$

با جمع $\Rightarrow$

$$x^2 + y^2 + (a-2)^2 = d_3^2 \underbrace{(\cos^2(\theta_2) + \sin^2(\theta_2))}_{1}$$

$$\Rightarrow \underline{d_3} = \sqrt{x^2 + y^2 + (a-2)^2}$$

بخش شبیه‌سازی

گام اول

در این گام قرار است قرار است کدی بزنیم که ربات ما دور یک مستطیل از قبل تعیین شده حرکت کند و برای این قسمت فرض بر این است که ربات تنها میتواند به صورت مستقیم حرکت کند یا ۹۰ درجه به چپ بچرخد و ربات نیز از مرکز مستطیل یعنی نقطه (۰و۰) شروع به حرکت می کند.

برای این کار نیاز است ابتدا ربات ۹۰ درجه به چپ بچرخد و سپس به صورت مستقیم حرکت کند تا به مستطیل برسد یعنی نقطه (۲و۰) و بعد دوباره ۹۰ درجه بچرخد تا بر روی ضلع مستطیل قرار بگیرد و روی آن حرکت کند. همانطور که مشاهده میکنید این کار در تابع start() انجام می شود.

```
def start(self):

    rospy.sleep(1)
    self.turn_90_left()
    self.stop()
    self.go_straight()

    while True:
        robot_pose = self.get_position()
        tmp_distance = self.Euclidean_distance((robot_pose.x,robot_pose.y),(0,2))

        if tmp_distance <= self.stop_distance :
            self.stop()
            break

    self.turn_90_left()
    self.stop()

    self.state = self.GO
```

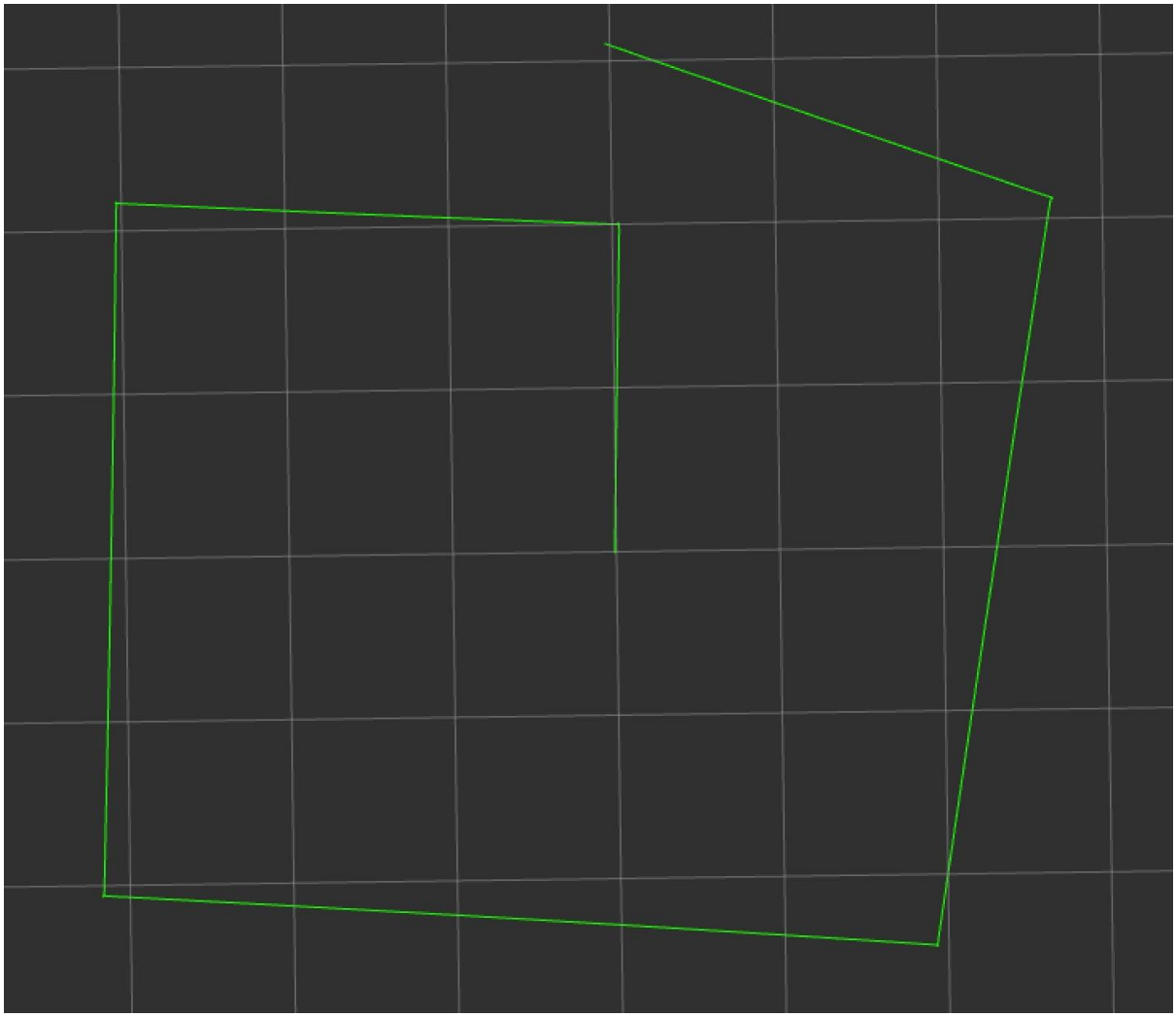
بعد از قرار گرفتن بر روی مستطیل کافی است که به صورت مستقیم حرکت کنیم و تنها وقتی که به راس های مستطیل که نقاطی مشخص هستند، برسیم باید حالت خود را عوض کرده و ۹۰ درجه بچرخیم و بعد از اتمام چرخش دوباره به حالت حرکت مستقیم ادامه دهیم. حرکت بر اساس حالت ربات در تابع run() انجام می شود و تغییر از حالت حرکت مستقیم به چرخش هم توسط تاپیک odometry انجام می شود به این صورت که وقتی به راس ها نزدیک می شویم یعنی اگر فاصله از نزدیک ترین راس از یک حد مشخصی کمتر شود این تغییر حالت رخ میدهد که البته پیچیدگی های بیشتری نیز دارد که در تصویر و کد مشاهده خواهید کرد و همچنین تغییر حالت از چرخش به حرکت مستقیم نیز بعد کامل شدن چرخش در تابع run() انجام می شود.

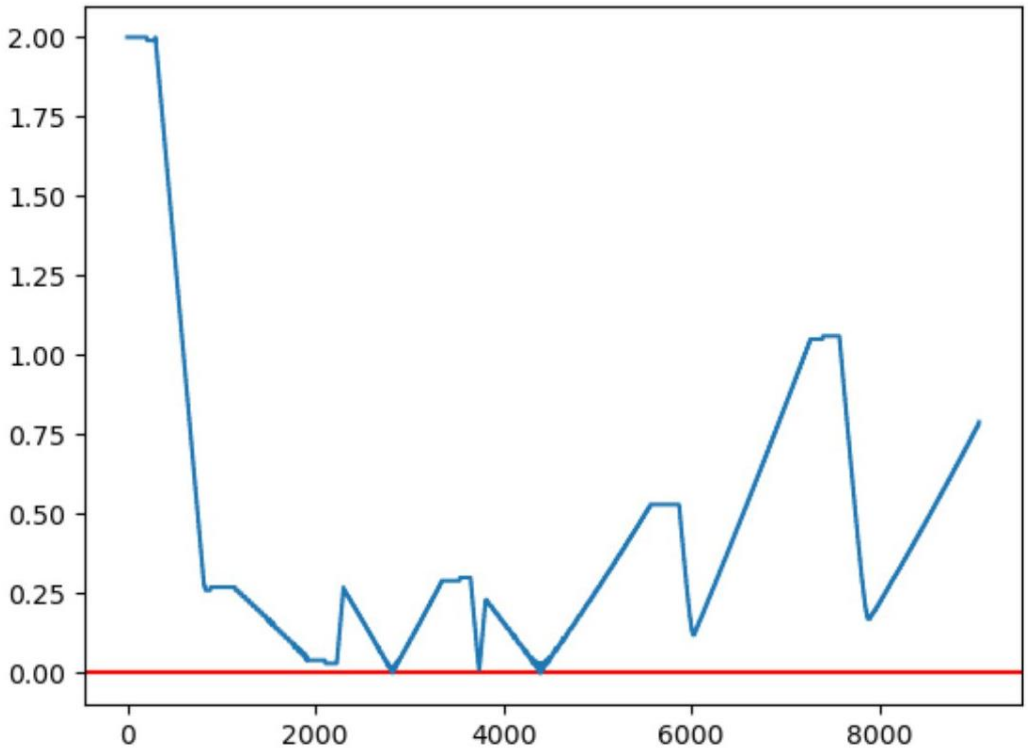
```python
def run(self):

    while not rospy.is_shutdown():

        if self.state == self.GO:
            self.go_straight()
            continue

        self.stop()

        self.turn_90_left()
        self.stop()

        self.state = self.GO
```

```python
def odom_callback(self, msg):

    pose = msg.pose.pose.position
    myposition = (pose.x, pose.y)
    border_distance = []

    for i in range(len(self.border_points)):
        border_distance.append(self.Euclidean_distance(myposition, self.border_points[i]))

    # print(border_distance[self.corner])
    self.errors.append(border_distance[self.corner])

    if border_distance[self.corner] >= self.far_distance:
        return


    if border_distance[self.corner] <= self.stop_distance:
        self.state = self.ROTATE
        self.next_corner()
        print("small")
        return

    if (self.corner == 0 and (pose.x <= -3 or pose.y > 2.5)) or (self.corner == 1 and (pose.y <= -2 or pose.x < -3.5)):
        self.state = self.ROTATE
        self.next_corner()
```

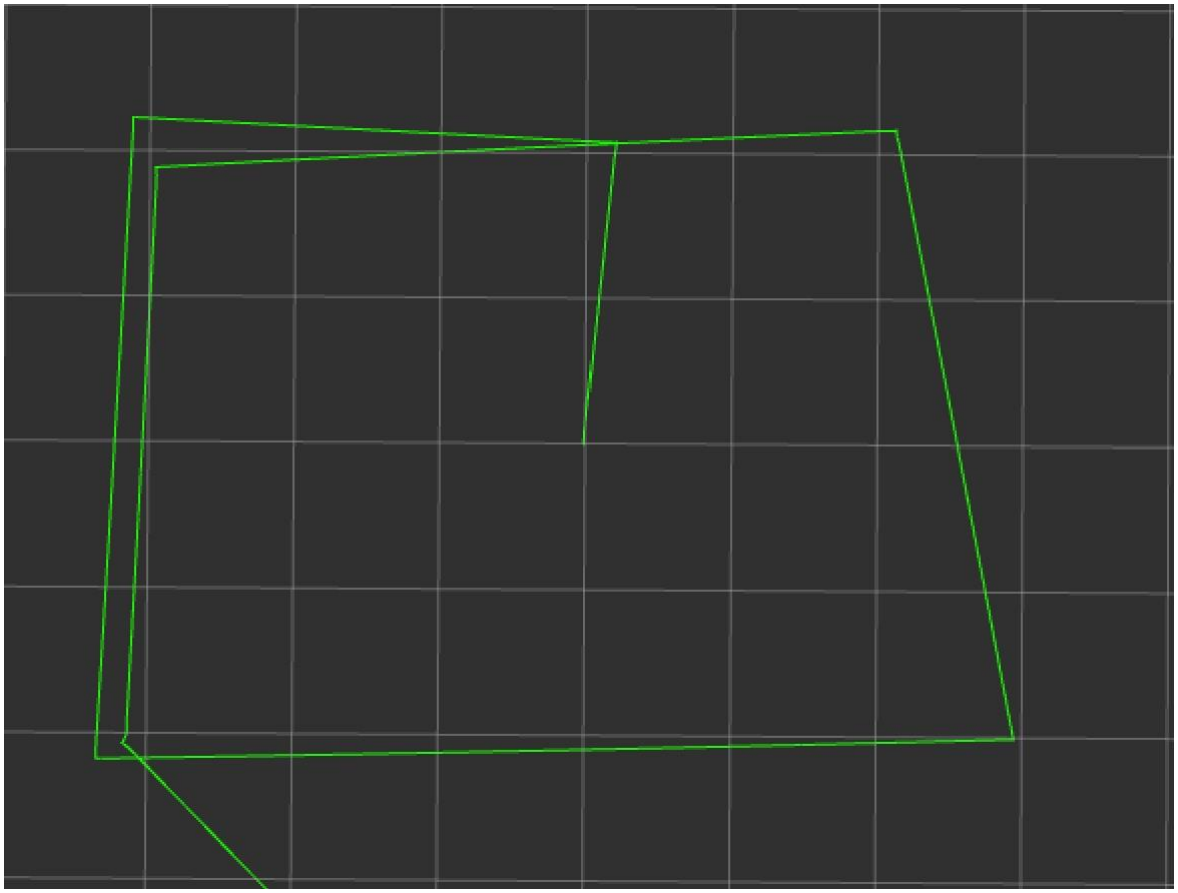
نکته ایی که برای این گام وجود دارد این است که در چرخش ۹۰ درجه ربات دقیق نیست و متاسفانه حتی در مواردی با ۴۵ درجه اختلاف می چرخد و همین موضوع باعث می شود ربات حرکت های نامنظمی داشته باشد و نتواند رو مستطیل حرکت و عکس های آورده شده در صفحه بعد نیز پس از تکرار های زیاد رخ داده است و متاسفانه تضمینی وجود ندارد که در اجرا گرفتن های بعدی چنین شکلی دوباره رخ دهد زیرا همانطور که گفته شد چرخش های ربات انحراف بسیار زیادی دارد

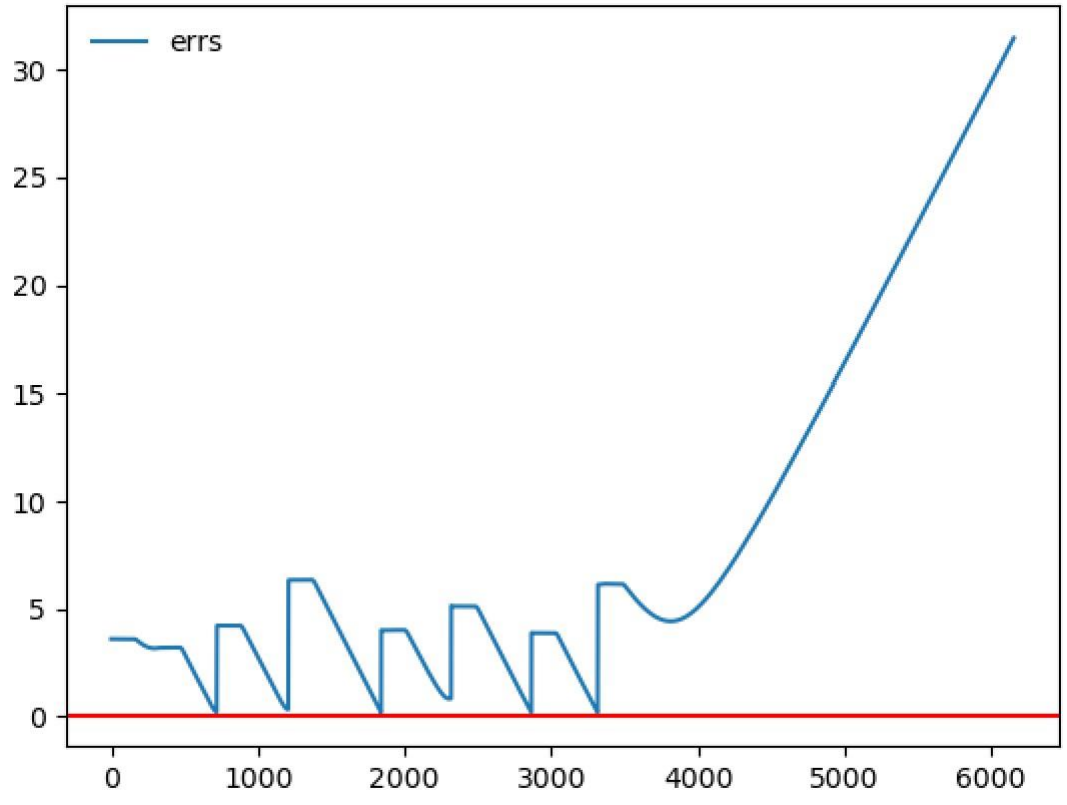
سرعت خطی ۰.۲ :





سرعت خطی ۰.۴ :





همانطور که مشاهده میکنید میزان خطا در چرخش یا به عبارتی انحراف آن به طور مستقیم به سرعت خطی مربوط نیست ولی خب در میزان تشدید آن تاثیر زیادی دارد. برای مثال اگر که ابعاد مستطیل بزرگ تر بود میدید که حتی یک انحراف کوچک در سرعت بالا چه انحراف زیادی را در پی خواهد داشت.

گام دوم

در این بخش قرار است که مسئله گام قبل را بوسیله کنترلر PID حل کنیم، برای اینکار نیاز است تا دو کنترلر، یکی برای کنترل سرعت زاویه ایی و یکی برای کنترل سرعت خطی داشته باشیم و همچنین روش کار به این صورت است که هدف خود را نزدیک ترین نقطه مستطیل به ربات قرار میدهیم و سپس خطای خطی که فاصله ربات تا آن نقطه هدف را بدست میاوریم و سپس خطای زاویه ایی که میزان انحراف زاویه این دو نقطه است را بدست میاوریم و بعد آن را در فرمول معروف PID قرار میدهیم و میزان مقادیر کنترلی را برای ربات بدست می‌آوریم.

نکته ایی که در این کنترلر بسیار اهمیت دارد، پیدا کردن ضرایب مناسب برای PID است که باید با آزمون و خطا آن ها را بدست آورد. همچنین در روشی به کار برده شده برای این گام، اگر که نقطه هدف از یک فاصله مشخصی نزدیک تر بود مهم است که نقطه انتخابی حتما در جلو ربات باشد زیرا که در غیر این صورت ربات به صورت مدام یک نقطه از پشت و جلو انتخاب کند و باعث شود که ربات به جای دنبال کردن ضلع مستطیل به دور خود بچرخد.

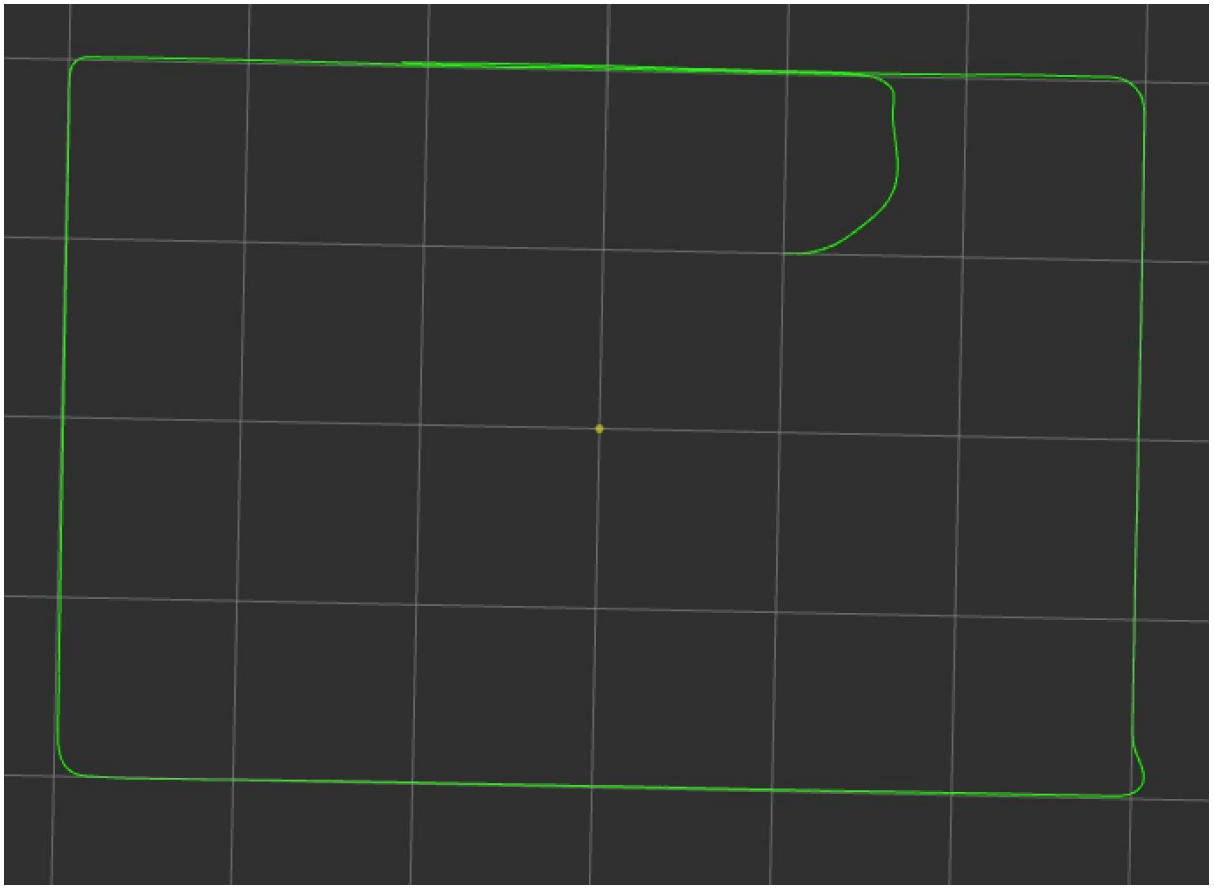
```
linear_error = self.get_linear_error()
angular_error = self.get_angular_error()


self.errors.append(linear_error)
sum_angular_error += (angular_error * self.dt)
sum_linear_error += (linear_error * self.dt)

# calculate PID for linear speed
P = self.kp_l * linear_error
I = self.ki_l * sum_linear_error
D = self.kd_l * (linear_error - prev_linear_error)
twist.linear.x =  P + I + D

# calculate PID for angular speed
P = self.kp_a * angular_error
I = self.ki_a * sum_angular_error
D = self.kd_a * (angular_error - prev_angular_error)
twist.angular.z =  P + I + D

prev_angular_error = angular_error
prev_linear_error = linear_error

self.cmd_publisher.publish(twist)
```

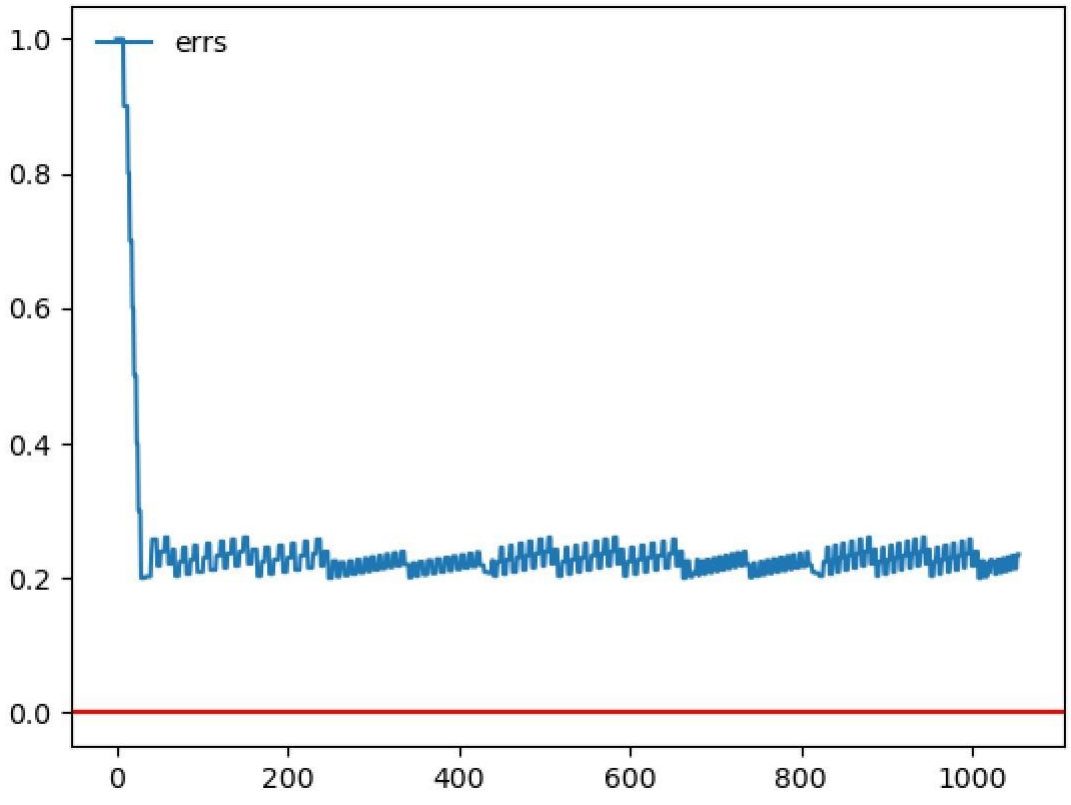
همانطور که در صورت سوال ذکر شده است نقطه شروع ربات، نقطه (۱و۱) است.

شکل اول برای ضرایب :

pid for linear = 0.75 , 0.01 , 2.25

pid for angular = 2.45 , 0.001 , 1.1

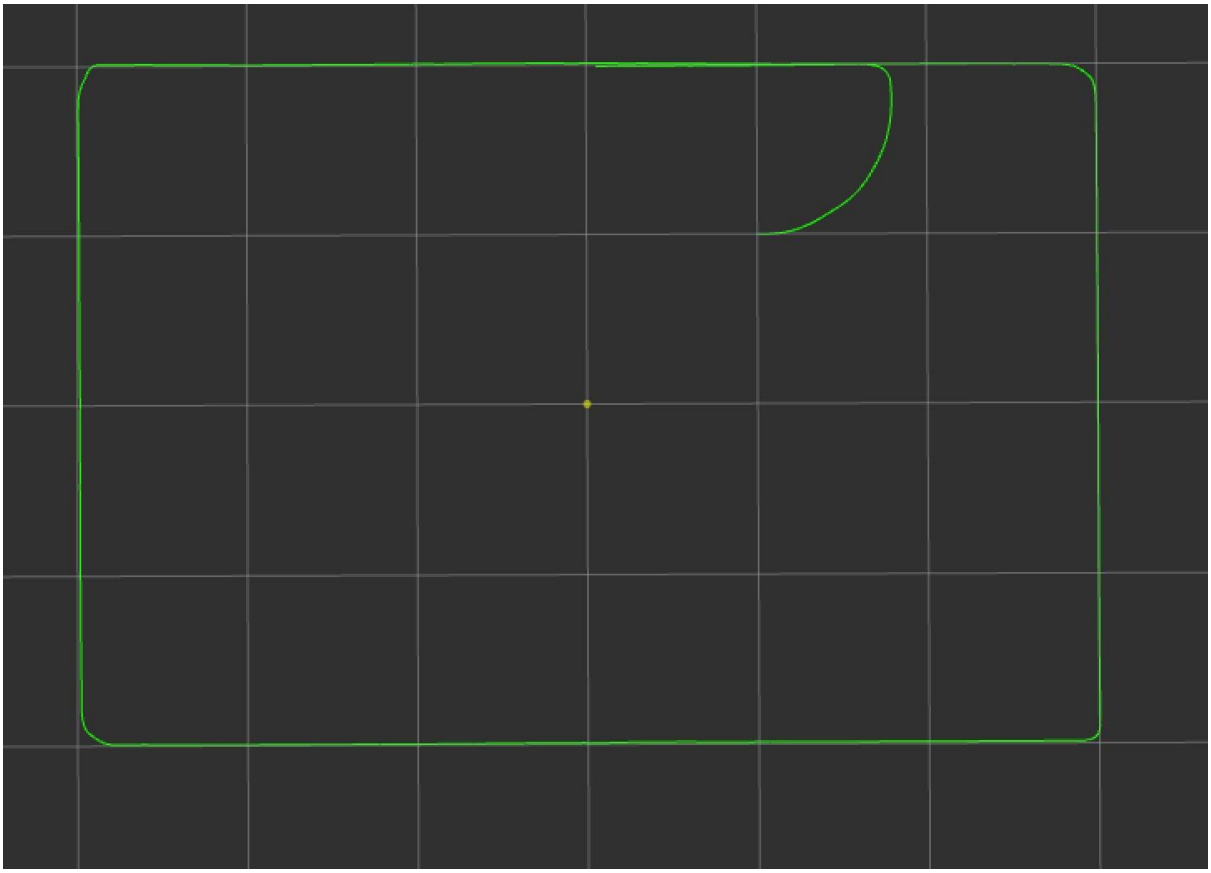


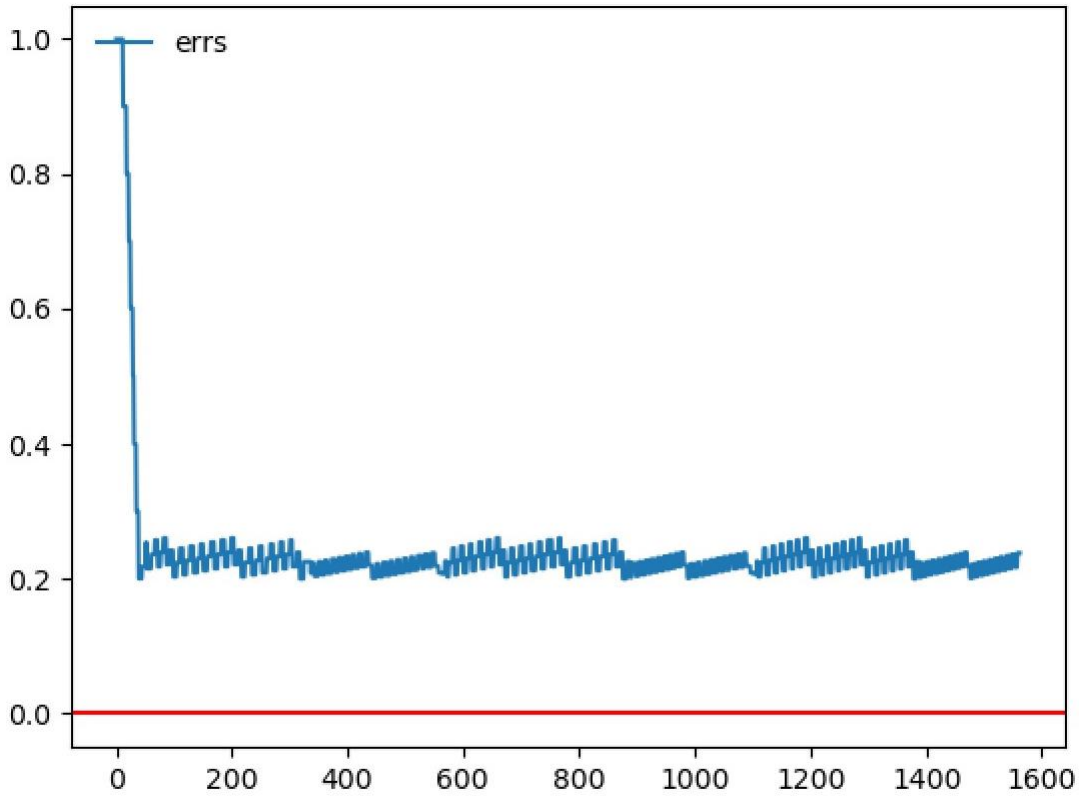


شکل دوم برای ضرایب :

pid for linear = 0.75 , 0.01 , 2.5

pid for angular = 2.45 , 0.01 , 1.1

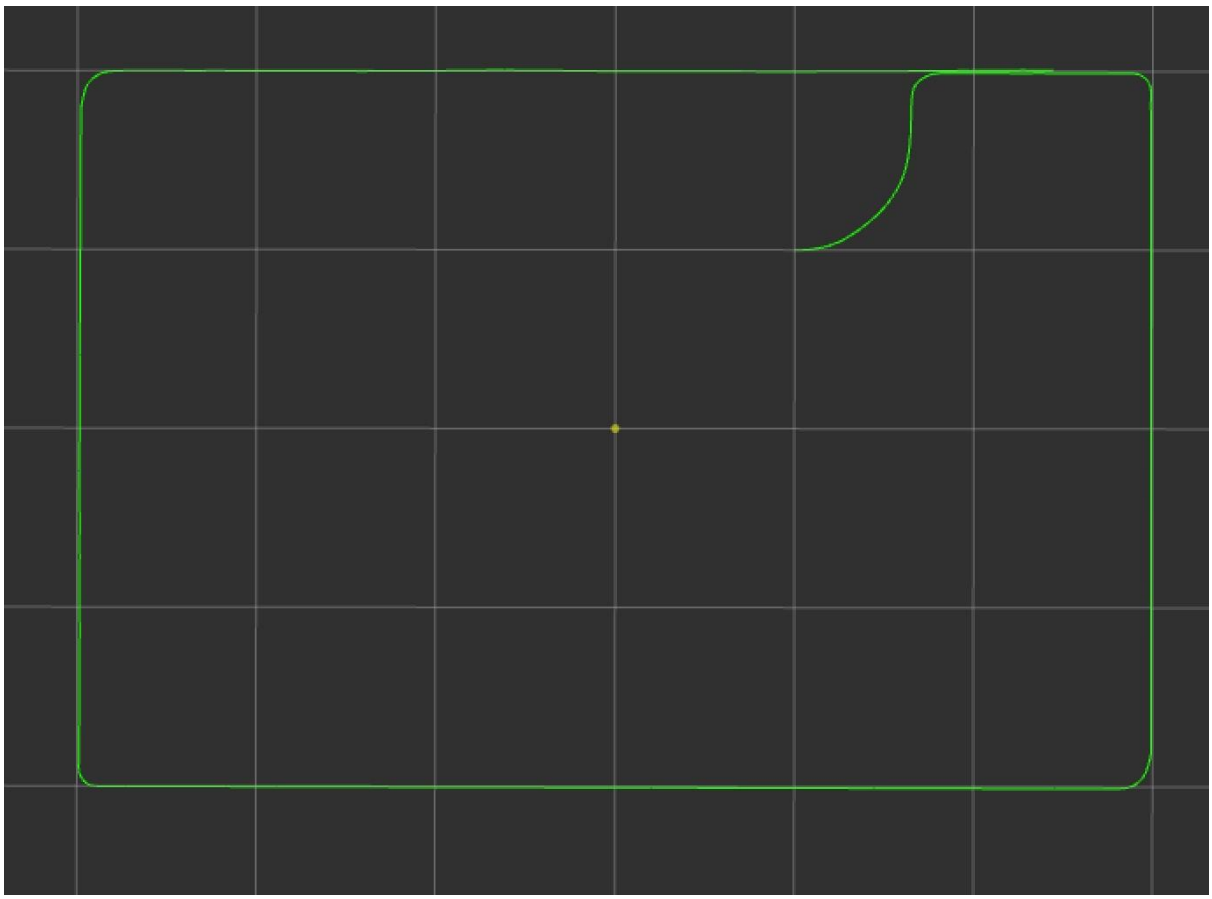


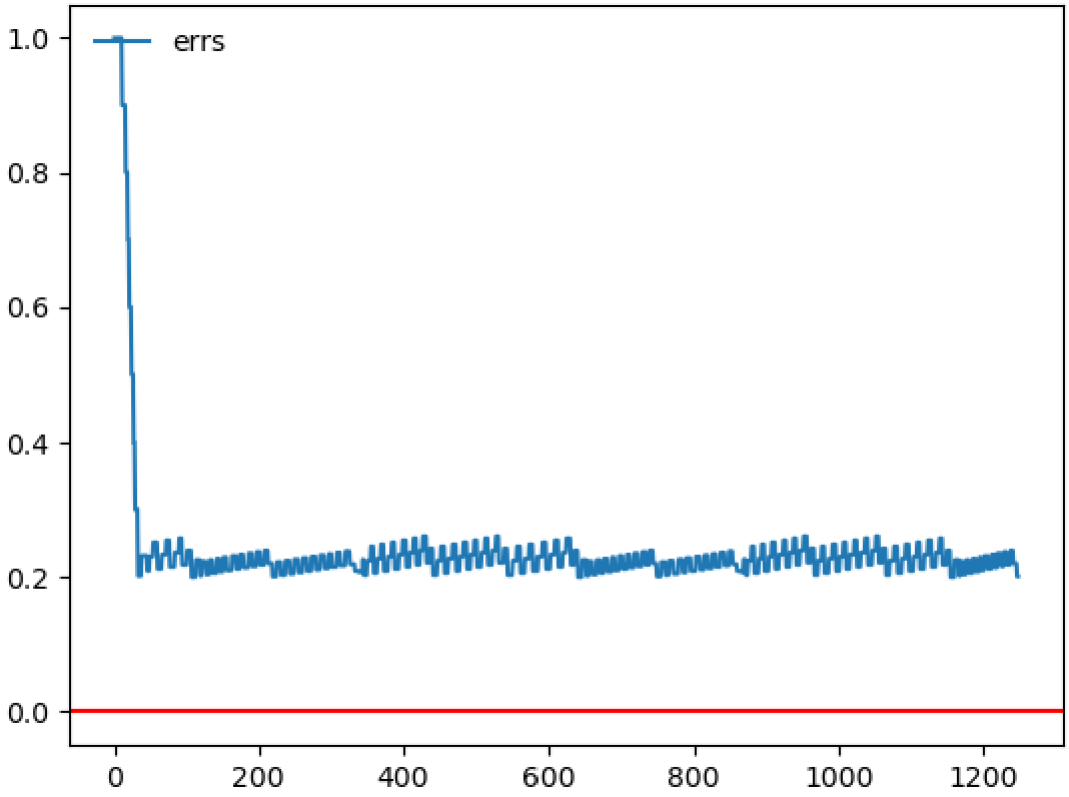


شکل سوم برای ضرایب :

pid for linear = 0.75 , 0.001 , 2.5

pid for angular = 2.6 , 0.001 , 0.9





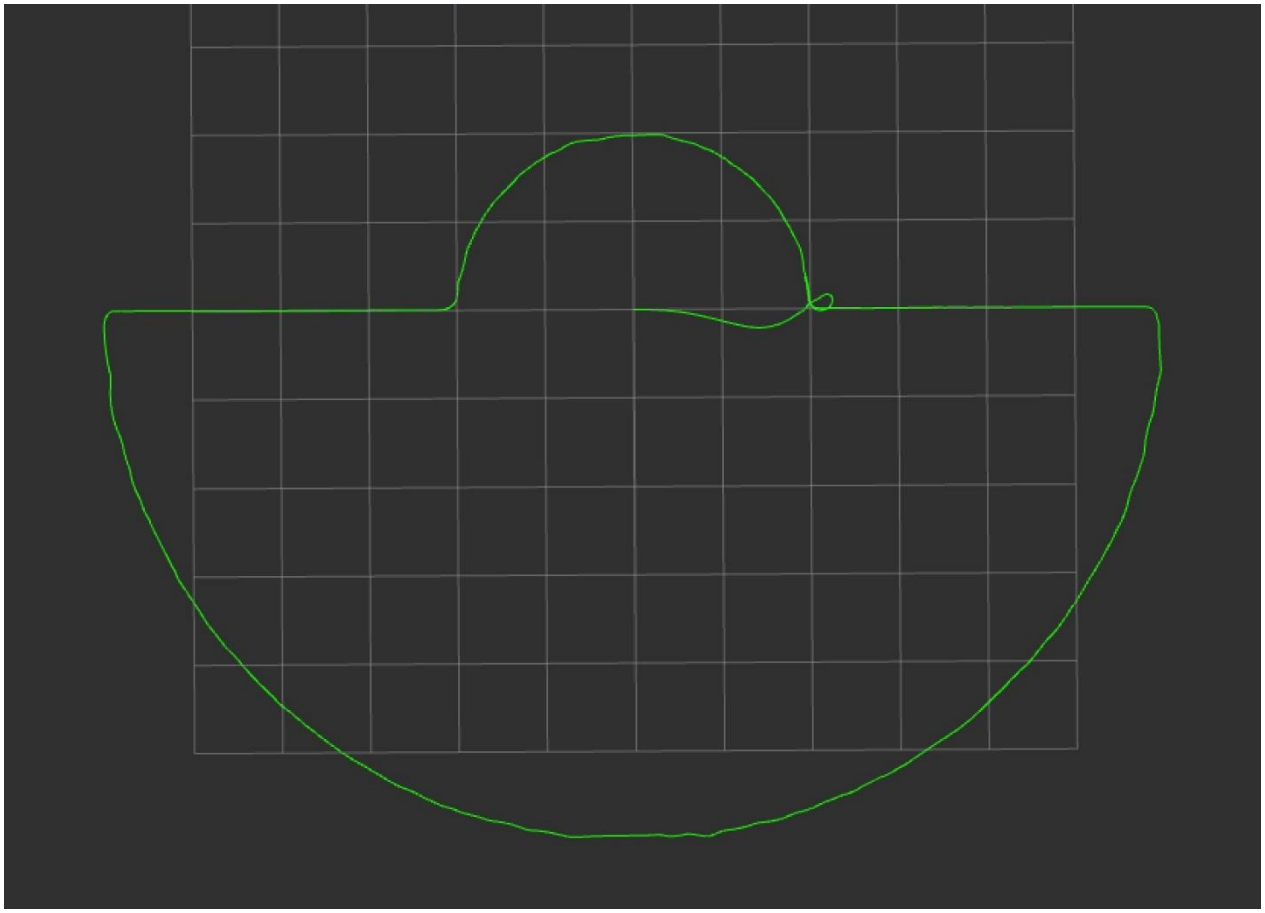
قبل از توضیح چگونگی بدست آوردن ضرایب بهتر است نحوه کار هر قسمت از این کنترلر را بررسی کنیم، همانطور که از درس آموختیم قسمت p نسبت به خطای هر لحظه یا به عبارتی به خطای حال واکنش نشان میدهد پس هر چه قدر مقدار آن بشتر باشد زودتر خطا را کم کرده و به هدف نزدیک می‌شود. قسمت i نیز نسبت گذشته خطاها واکنش نشان می‌دهد که در واقع می‌شود انتگرال خطاها تا لحظه حال، این ضریب باعث می‌شود هنگامی که به هدف رسیدیم و مقدار خطا بسیار کوچک و تقریبا صفر شد، ربات متوقف نشود حالت اش حفظ شود، مثلا در این سوال روی ضلع مستطیل حرکت کند. قسمت d نیز در واقع مشتق خطای آن لحظه است که می توان گفت این قسمت نگاهی به آینده دارد و هنگامی که داریم به هدف نزدیک می شویم با توجه به اینکه خطا کم و مشتق آن منفی می شود و در عمل باعث می شود که ما در نزدیکی هدف حرکتی نرم داشت باشیم و از overshoot اجتناب کنیم.

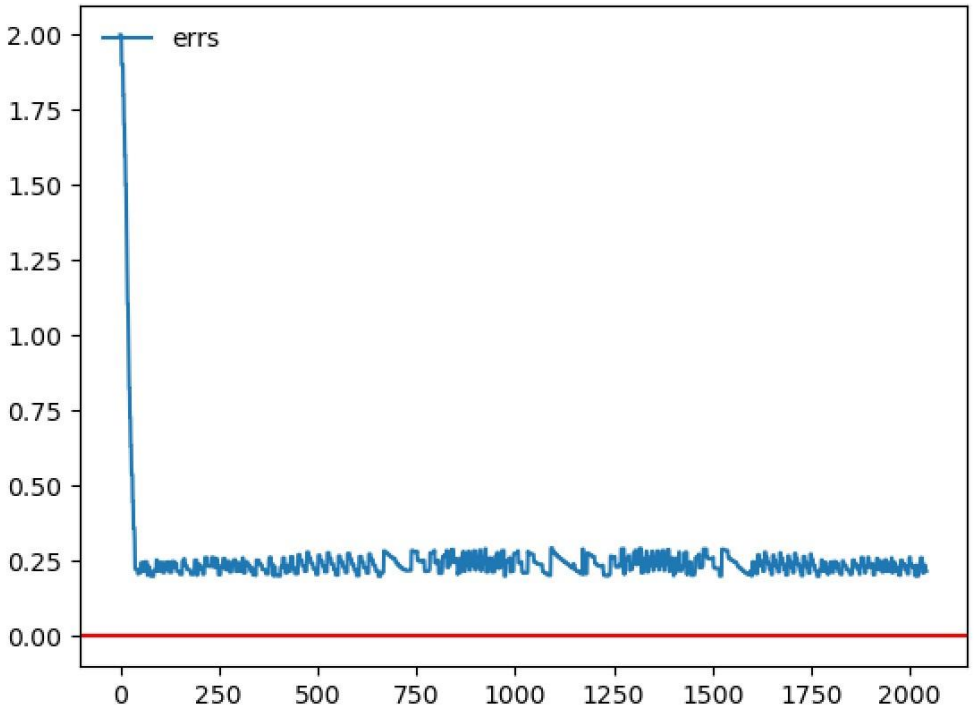
برای بدست آوردن ضرایب بهتر است ابتدا به صورت تصادفی آن‌ها را مقداردهی کنیم و نتایج را بررسی کنیم و هر بار تغییرات لازم را انجام دهیم تا به حرکت نسبتا خوبی مثلا تصویر اول برسیم. سپس با بررسی دقیق تر حرکت ربات باید متوجه شویم که کدام ضریب باید تغییر کند تا ربات حرکت بهتری داشته باشد، برای مثال در همان تصویر اول مشاهده می‌کنیم که در بعضی از راس ها ربات هنگام چرخیدن overshoot داشته است پس برای همین باید مقدار ضریب d سرعت خطی آن را افزایش بدهیم و همچنین برای مثال در تصویر دوم مشاهده میکنیم که ربات هنگام چرخش در راس ها می توانست بهتر عمل کند، در واقع هنگام نزدیک شد به راس ها بهتر است با سرعا زاویه ایی بیشتر بچرخد تا حرکت نرم‌تری بدست بیاد برای همین برای تصویر سوم ضریب p را افزایش و ضریب d را کاهش می‌دهیم و در نتیجه می‌بینیم که ربات در تصویر سوم حرکت بسیار بهتر داشته است.

گام سوم

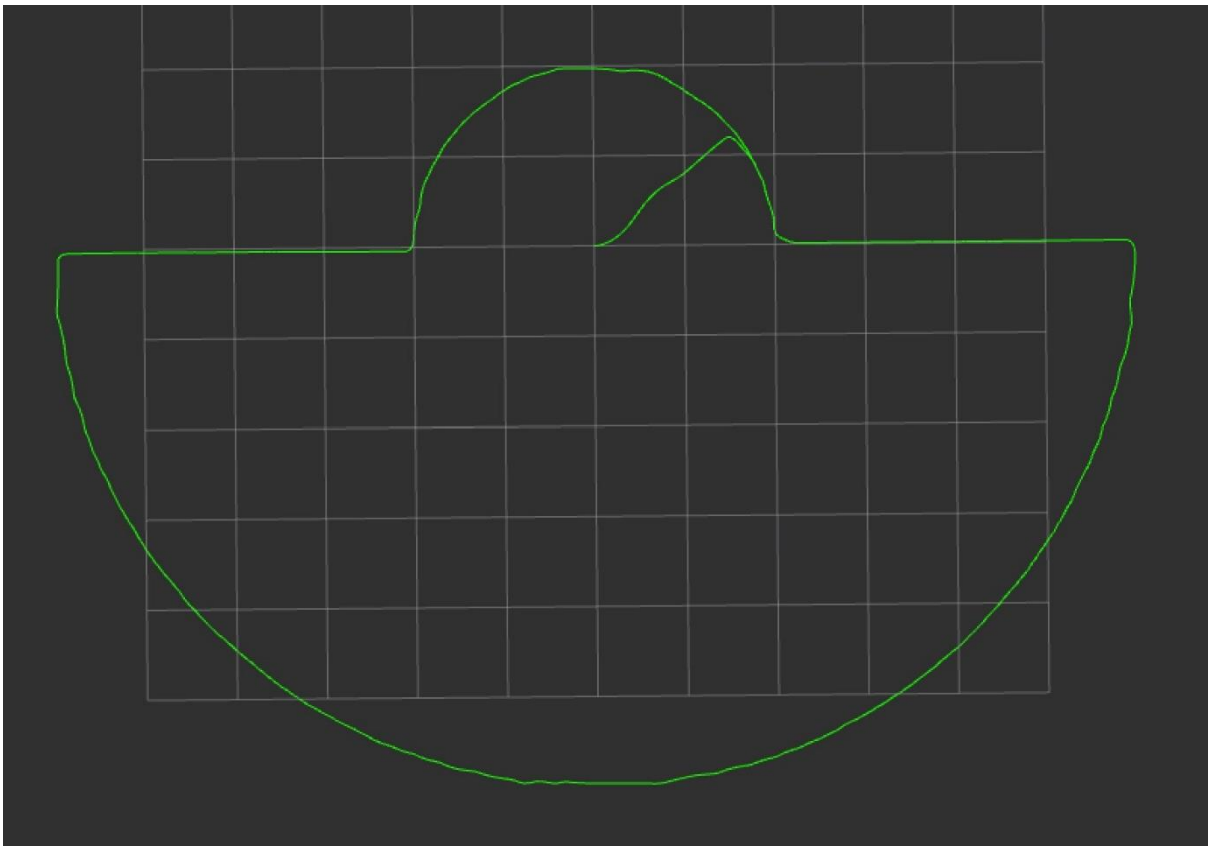
در این گام قصد داریم تا بوسیله کنترلی که در گام قبلی طراحی کردیم، ربات را در مسیر های پیچیده تری کنترل و حرکت دهیم.

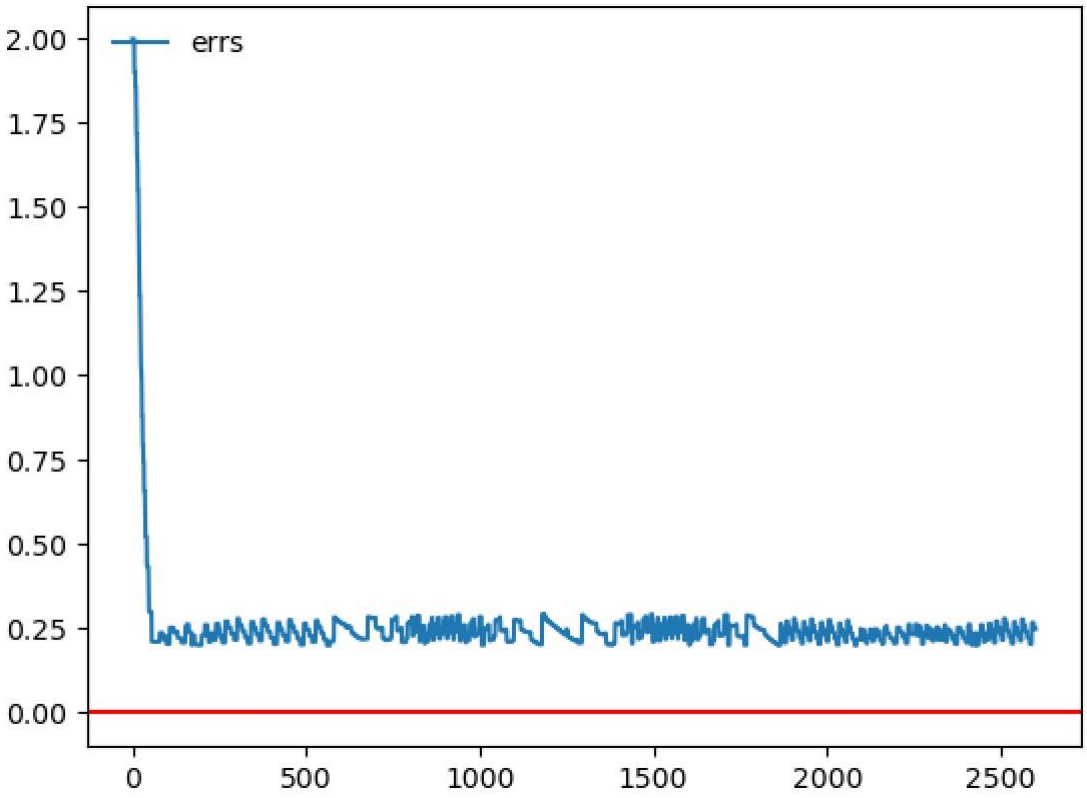
ترکیب دو نیم دایره : تصویر اول



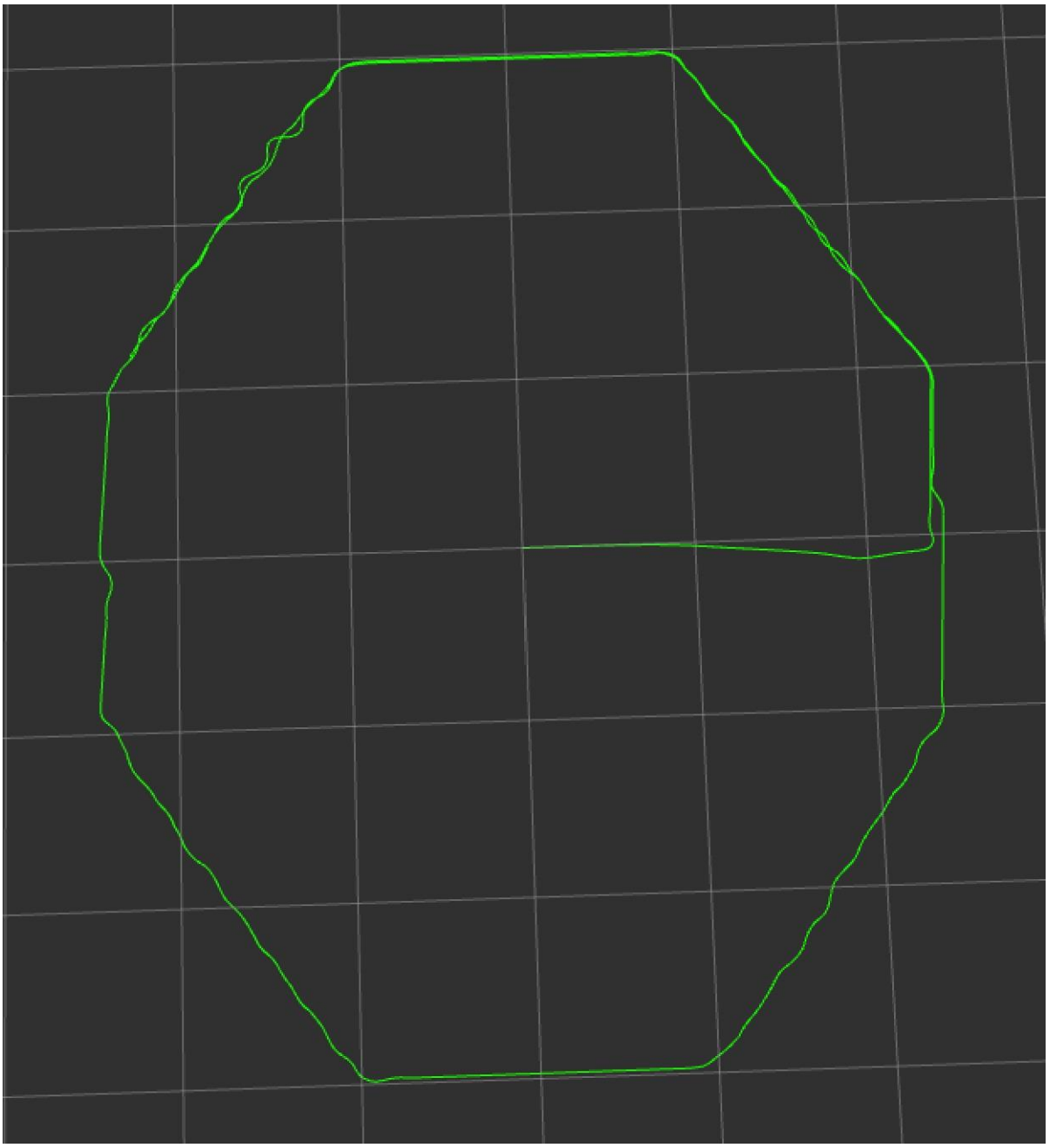


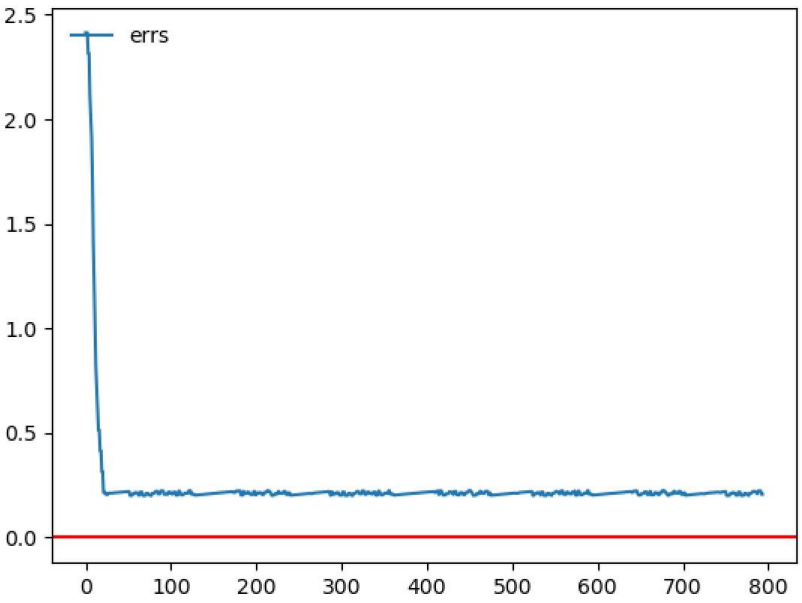
ترکیب دو نیم دایره : تصویر دوم (بدست آوردن ضریب های بهتر)



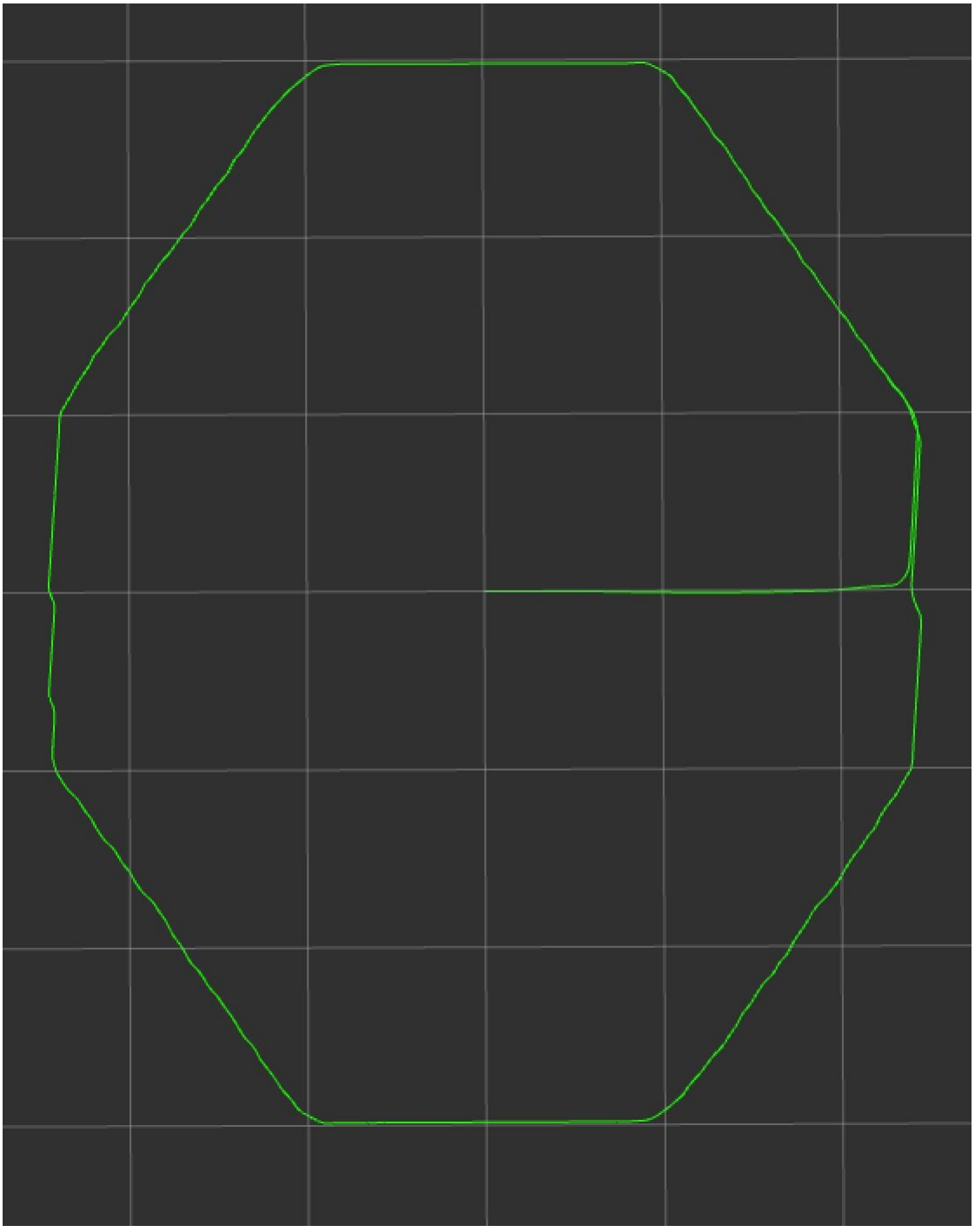


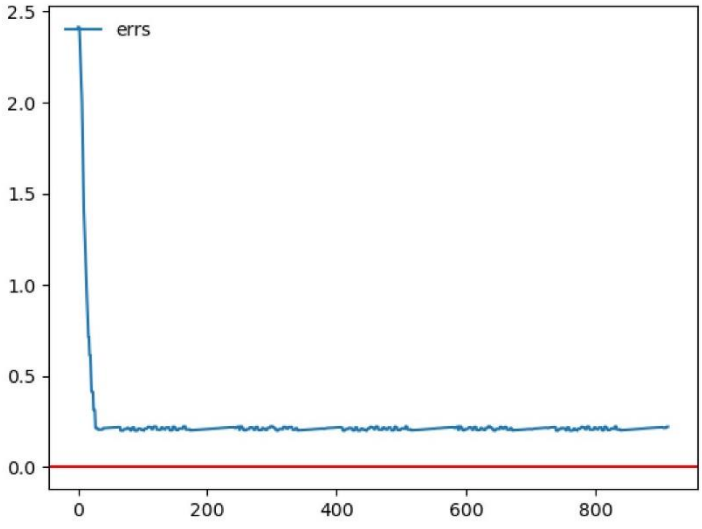
هشت ضلعی منتظم به ضلع دو: تصویر اول





هشت ضلعی منتظم به ضلع دو : تصویر دوم (با ضرایب بهتر)





مارپیچ لگاریتمی

متاسفانه به درستی ربات حرکت نمی کرد