



Amirkabir University of Technology  
(Tehran Polytechnic)



Department of  
Computer Engineering

دانشگاه صنعتی امیرکبیر

دانشکده مهندسی کامپیوتر و فناوری اطلاعات

---

مبانی اصول علم رباتیک

تمرین سری دوم

---

محمدحسین بدیعی

شماره دانشجویی 9531701

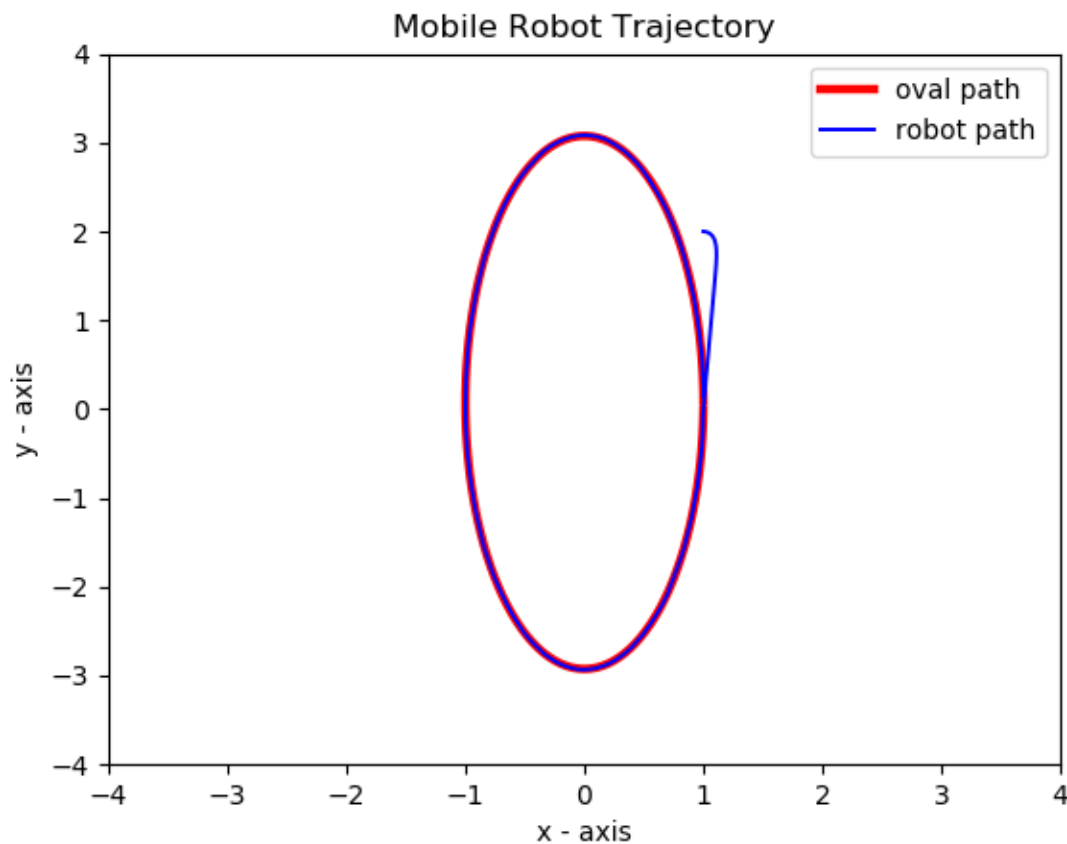
استاد : دکتر مهدی جوانمردی

بهار 1399-1400

ابتدا نتایج را ذکر میکنیم و سپس به توضیحات خواسته شده در صورت سوال می‌پردازیم.

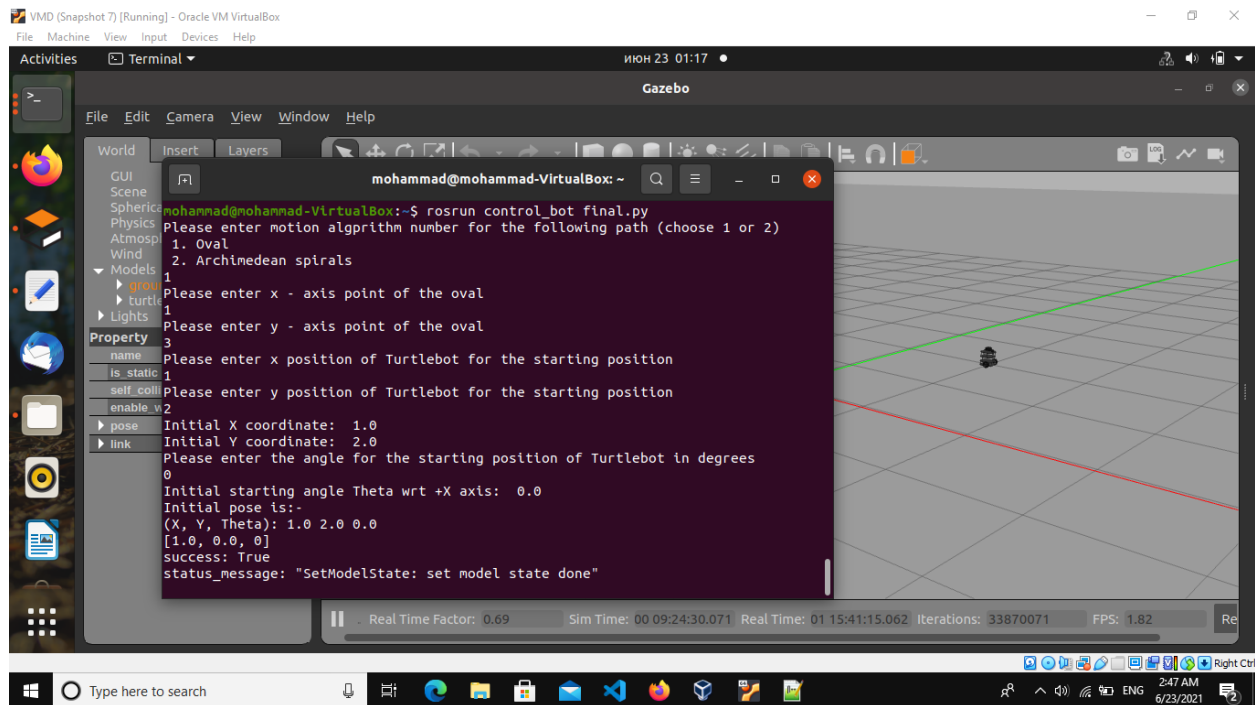
برای رسم نتایج مسیر ربات و فیدبک‌های گرفته شده از مکان ربات را در کنار یکدیگر برای دو شکل بیضی oval و مارپیچی ارشمیدوس spiral رسم کردیم و دو شکل نیز مربوط به خطاهای هر کدام از مسیرها رسم کردیم که برای بیضی این خطا برای بازه 0 تا  $2\pi$  و برای مارپیچی برای بازه 0 تا  $9\pi$  رسم نمودیم.

خروجی برای شکل oval (بیضی) در بازه 0 تا  $2\pi$  برای مسیر طی شده در Gazebo



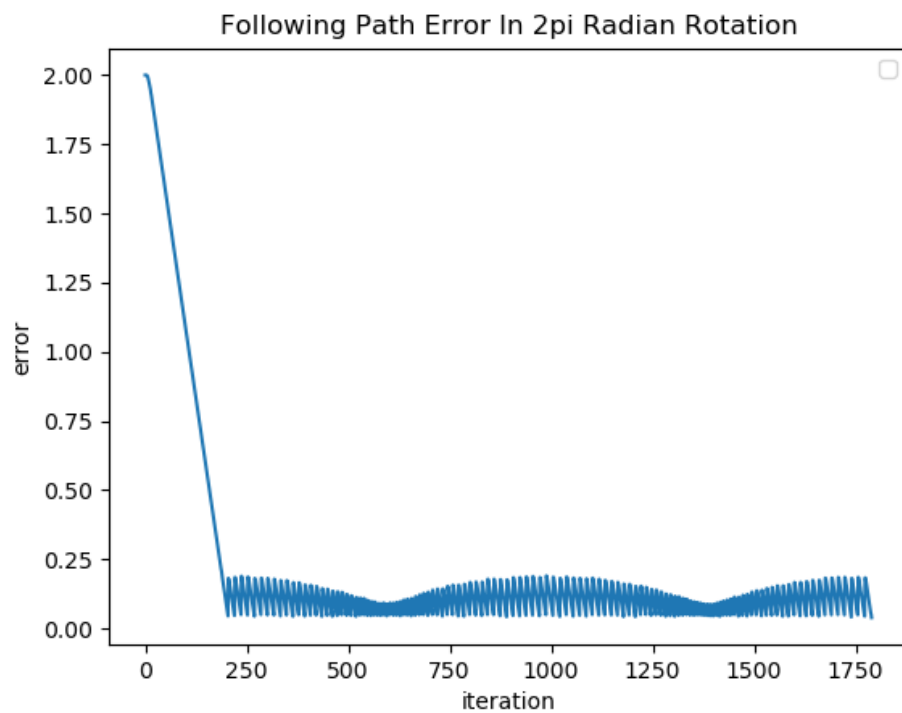
همانطور که مشاهده می‌کنید نقطه‌ی شروع ربات را مطابق با خواسته‌ی مساله و البته به صورت پارامتری به عنوان ورودی قرار دادیم.

ورودی‌های کد به صورت شکل زیر می‌باشد:



تمامی پارامترهای لازم را مطابق شکل فوق از کاربر به عنوان ورودی دریافت کردیم و شکل قبلی را مطابق با این پارامترها که البته خواسته‌ی مسأله نیز بود برای بازه‌ی 0 تا  $2\pi$  برای بیضی رسم نمودیم.

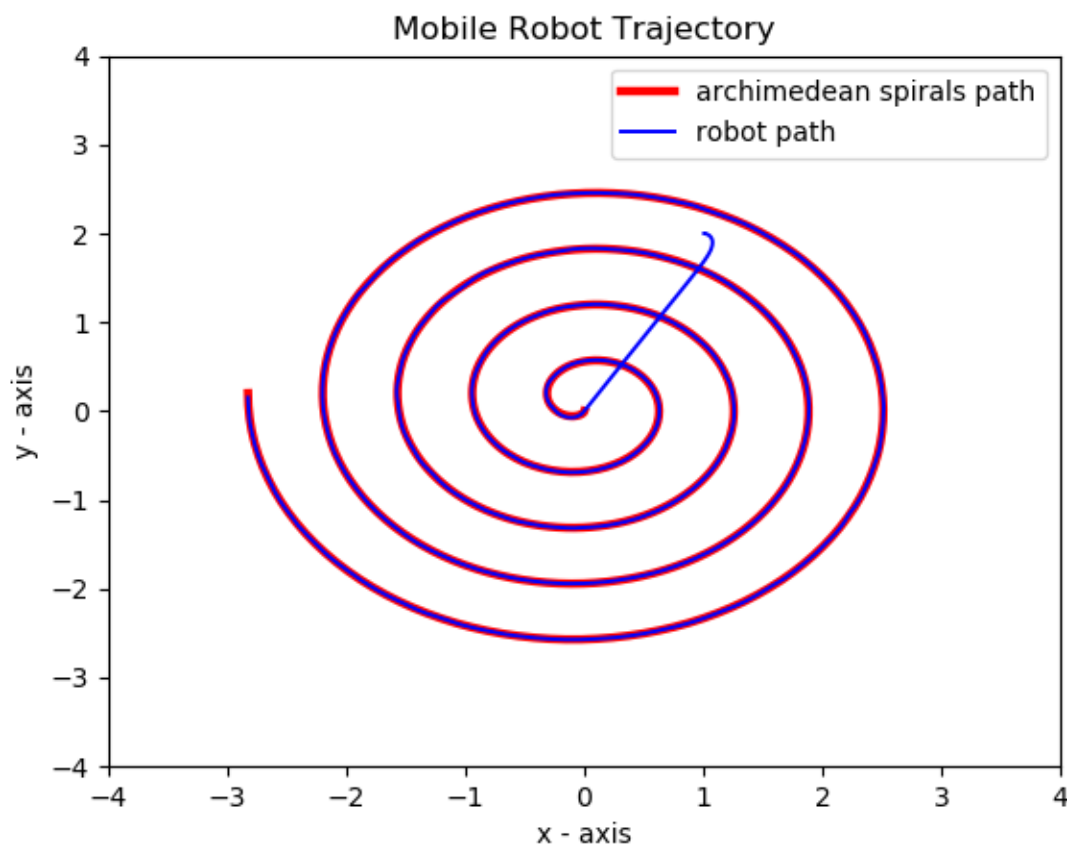
در نهایت خطا در این بازه نیست بر اساس تعداد iteration ها فیدبک گرفتن از position در این بازی به صورت زیر است.



شکل بعدی‌ای که مساله از ما خواسته بود، یک Archimedean spirals می‌باشد که growth factor آن برابر با 0.1 است.

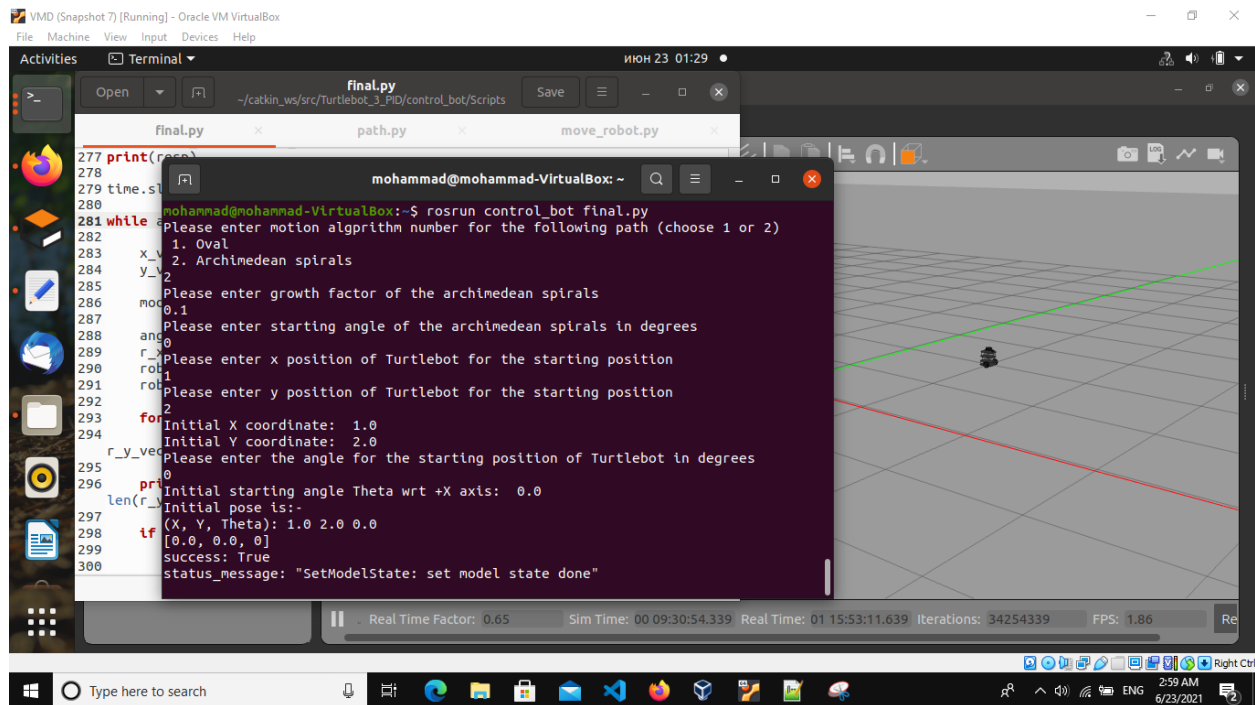
خروجی گرفته شده به صورت زیر می باشد.

توجه بفرمایید که بازه‌ی حرکت ربات از 0 تا  $2\pi$  قرار داده‌ایم.



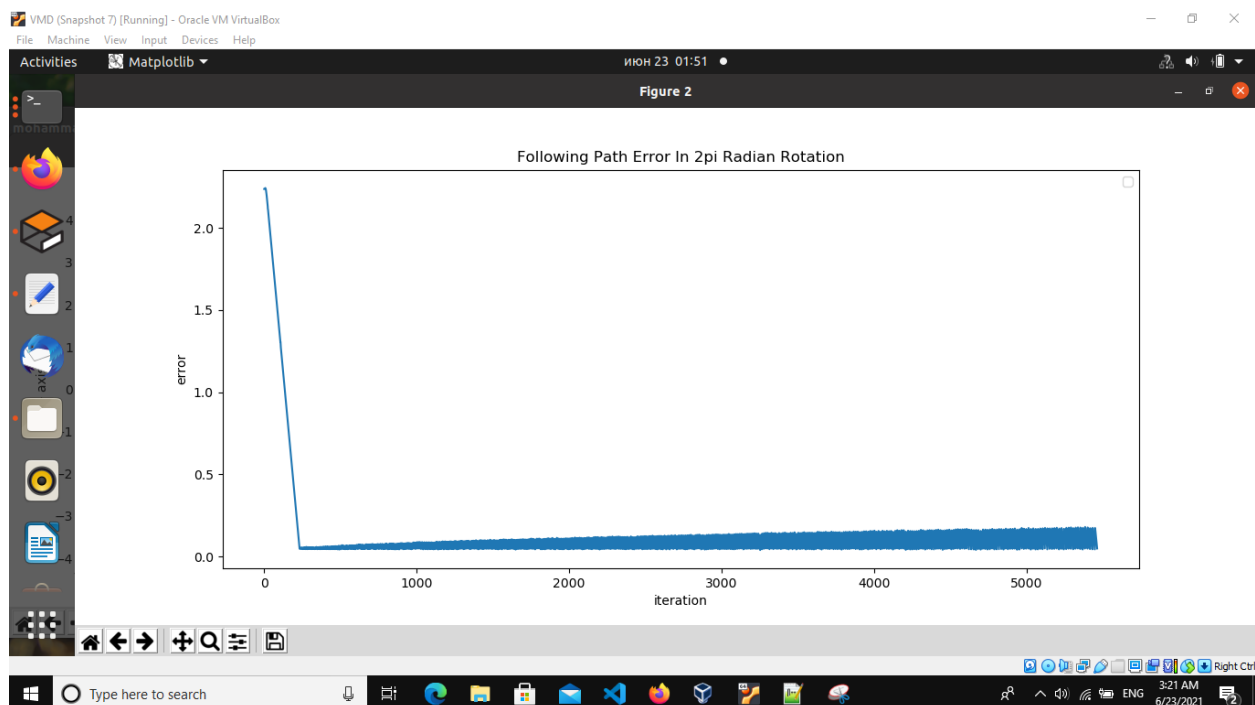
همانطور که مشاهده می‌کنید در این شکل نیز ربات از نقطه‌ی ( 2 , 1 ) طبق خواسته‌ی مسأله شروع به حرکت نموده و خود را به شعاع صفر یعنی  $\text{radius} = \Theta = 0$  رسانده است. البته توجه نمایید که ما حتی  $\Theta$  را به عنوان پارامتر ورودی در کنار growth factor از کاربر دریافت می‌نماییم.

برای مشاهده پارامترهای ورودی‌ای که برای طی کردن مسیر مارپیچ توسط ربات به کد دادیم به شکل زیر توجه بفرمایید. در واقع ابتدا الگوریتم مسیر را که یبضی یا مارپیچ است مشخص کردیم و سپس growth factor و  $\Theta$  اولیه مارپیچ را از کاربر درخواست کردیم و سپس مشخصات شروع ربات که x و y و angle اولیه ربات است که به ترتیب ( 0 , 2 , 1 ) را وارد کردیم.



لذا مشخصات را مانند کد بالا وارد کردیم و ربات بلافاصله در نقطه مورد نظر قرار گرفته و سپس مطابق شکلی که `plot` کردیم در Gazebo حرکت می‌کند.

در نهایت خطا نیز برای بازه‌ی 0 تا  $2\pi$  به صورت زیر می‌باشد. (در شکل زیر به اشتباه  $2\pi$  برچسب خورده و در واقع  $9\pi$  می‌باشد. و هر iteration نیز در واقع ارور موجود در position فیدبک گرفته شده از gazebo است که به نمایش گذاشته ایم.)



در ادامه به سوالات موجود در مساله پاسخ می‌دهیم.

کنترلر را برای ربات با ضرایب زیر تنظیم کردیم و در واقع با این ضرایب نتایج خوبی که شکلش را نشان دادیم حاصل شد.

✓ ضریب تناسبی در کنترلر سرعت  $v(t)$  :  $k_{p-distance} = 1$

✓ ضریب انتگرالی در کنترلر سرعت  $v(t)$  :  $k_{i-distance} = 0.01$

✓ ضریب تناسبی در کنترلر زاویه  $\lambda(t)$  :  $k_{p-angle} = 1$

```
17
18 kp_distance = 1
19 ki_distance = 0.01
20 kp_angle = 1
21
```

سپس از مشخصه‌های ربات در gazebo فیدبک گرفتیم به جهت مقایسه با ورودی مرجه برای سرعت و زاویه و در نهایت به کنترلر سرعت اعمال نمودیم.

در کنترلر سرعت نیز از فورمولی که در اسلاید ذکر شده بود استفاده کردیم و به صورت زیر پیاده نمودیم.

```
distance = sqrt(pow((goal_x - x_start), 2) + pow((goal_y - y_start), 2))
control_signal_distance = kp_distance*distance + ki_distance*total_distance
control_signal_angle = kp_angle*(path_angle - rotation)
```

شرط خروج از while و تنظیم set-point جدید را با ارور فاصله‌ی کمتر از 0.05 اعمال کردیم.

```
70 while distance > 0.05:
71     (position, rotation) = self.get_odom()
72     x_start = position.x
73     y_start = position.y
74
75     self.robot.x_vector.append(np.float32(x_start))
```

Python Tab Width: 8 Ln 78, Col 3

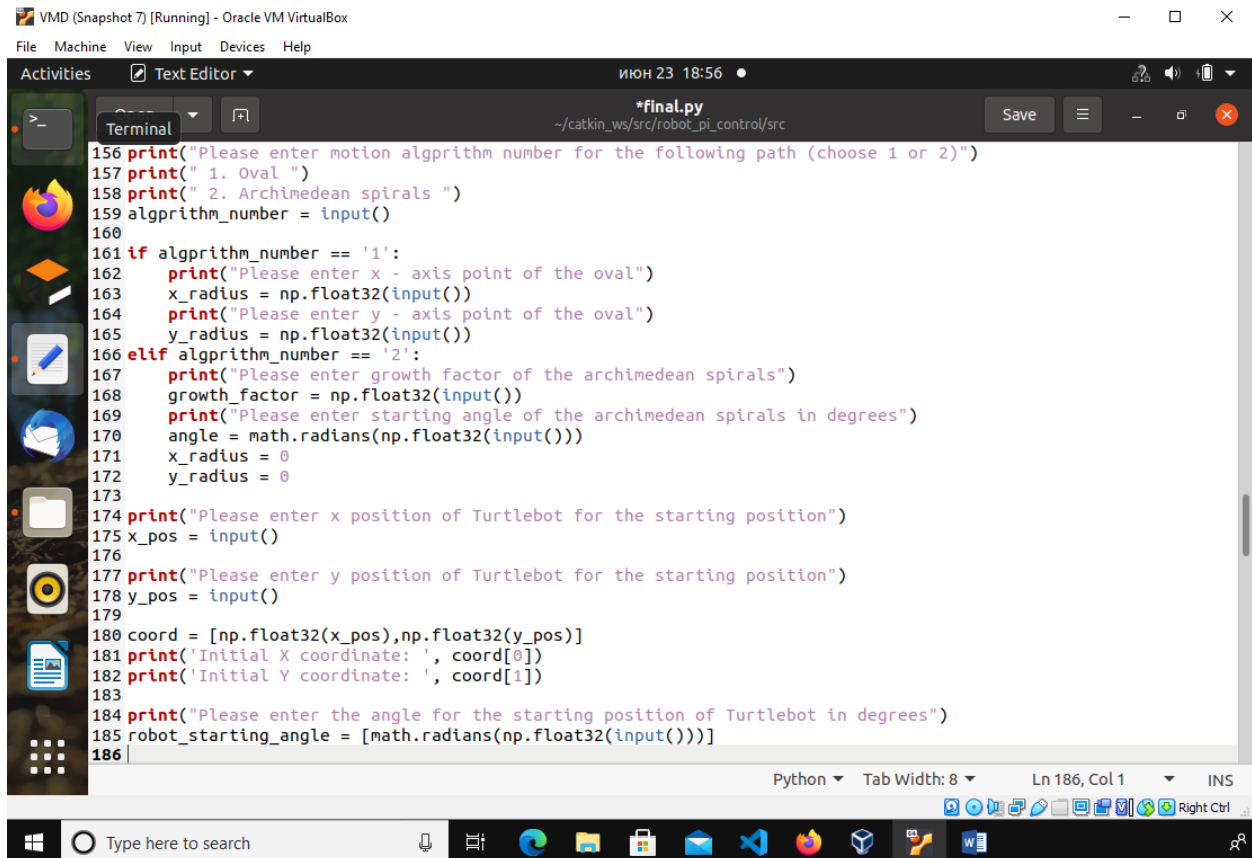
همچنین در هر گام برای رسیدن به هدف نیز previous\_distance و total\_distance را آپدیت می‌نماییم و در هر لحظه به کاربر موقعیت و جهت ربات را نشان می‌دهیم.

```
108 self.end_vec.position(move_end,
109 r.sleep()
110 previous_distance = distance
111 total_distance = total_distance + distance
112 print("Current positin and rotation are: ", (position, rotation))
113
```

همچنین لازم به ذکر است که در هر با خواندن تابع، total distance را ریست می‌کنیم که برای جلوگیری از اشباع بکار می‌رود.

فورموله سازی مسأله دقیقاً مطابق اسلاید ها و البته مراجعی که در ادامه ذکر خواهیم کرد انجام گرفته است.

برای ورودی ها نیز مسأله را کاملاً به صورت پارامتریک تعریف کردیم.



```
156 print("Please enter motion algorithm number for the following path (choose 1 or 2)")
157 print(" 1. Oval ")
158 print(" 2. Archimedean spirals ")
159 algorithm_number = input()
160
161 if algorithm_number == '1':
162     print("Please enter x - axis point of the oval")
163     x_radius = np.float32(input())
164     print("Please enter y - axis point of the oval")
165     y_radius = np.float32(input())
166 elif algorithm_number == '2':
167     print("Please enter growth factor of the archimedean spirals")
168     growth_factor = np.float32(input())
169     print("Please enter starting angle of the archimedean spirals in degrees")
170     angle = math.radians(np.float32(input()))
171     x_radius = 0
172     y_radius = 0
173
174 print("Please enter x position of Turtlebot for the starting position")
175 x_pos = input()
176
177 print("Please enter y position of Turtlebot for the starting position")
178 y_pos = input()
179
180 coord = [np.float32(x_pos), np.float32(y_pos)]
181 print('Initial X coordinate: ', coord[0])
182 print('Initial Y coordinate: ', coord[1])
183
184 print("Please enter the angle for the starting position of Turtlebot in degrees")
185 robot_starting_angle = [math.radians(np.float32(input()))]
186
```

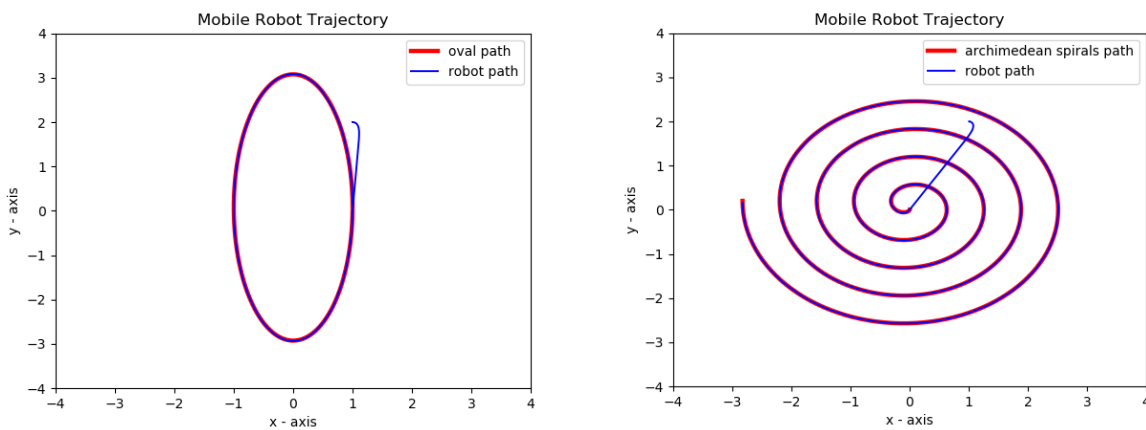
تمامی ورودی ها مطابق با قطعه کد فوق از کاربر دریافت می‌شود. این ورودی ها عبارت است از:

- ✓ کاربر در ابتدا الگوریتم مسیر خود را مشخص می‌کند. (بیضی یا مارپیچ)
  - ✓ در صورتیکه بیضی را انتخاب کند فاصله‌ی شعاعی محور X و نیز محور Y را به صورت جداگانه در ترمینال وارد می‌کند.
  - ✓ در صورتیکه مارپیچ را انتخاب کند growth factor و angle آغازین را برای طی کردن مارپیچ تعیین می‌کند.
  - ✓ در نهایت نیز از کاربر سه مقدار X و Y و  $\Theta$  را برای موقعیت آغازین ربات درخواست می‌کند.
- همچنین لازم به ذکر است که پارامتر سرعت زاویه‌ای ترسیم مسیر 0.05 قرار داده‌ایم که اشکال مسیر را با دقت بالایی در gazebo تعیین کند. در شکل‌هایی که از خروجی matplotlib گرفتیم این موضوع و دقت ایجاد شده کاملاً مشهود است.

در قسمت دوم خواسته شده است که ضرایب مناسب P و I را ارائه کنیم که در ابتدا ذکر کردیم به ضرایب مطلوب زیر رسیدیم.

$k_{p-distance}$	1
$k_{i-distance}$	0.01
$k_{p-angle}$	1

و گفتیم نتایج برای این ضرایب کنترلی نیز به صورت زیر برای پیمودن مسیر توسط ربات در Gazebo درآمد:



می بینیم که نتایج در این حالت بسیار مطلوب است و این ضرایب را می توان به عنوان یک ضرایب مناسب کنترلی ارائه کرد.

در قسمت سوم درباره ی تأثیرات افزایش یا کاهش ضرایب کنترلی بحث کرده و خواسته آن را تشریح کنیم.

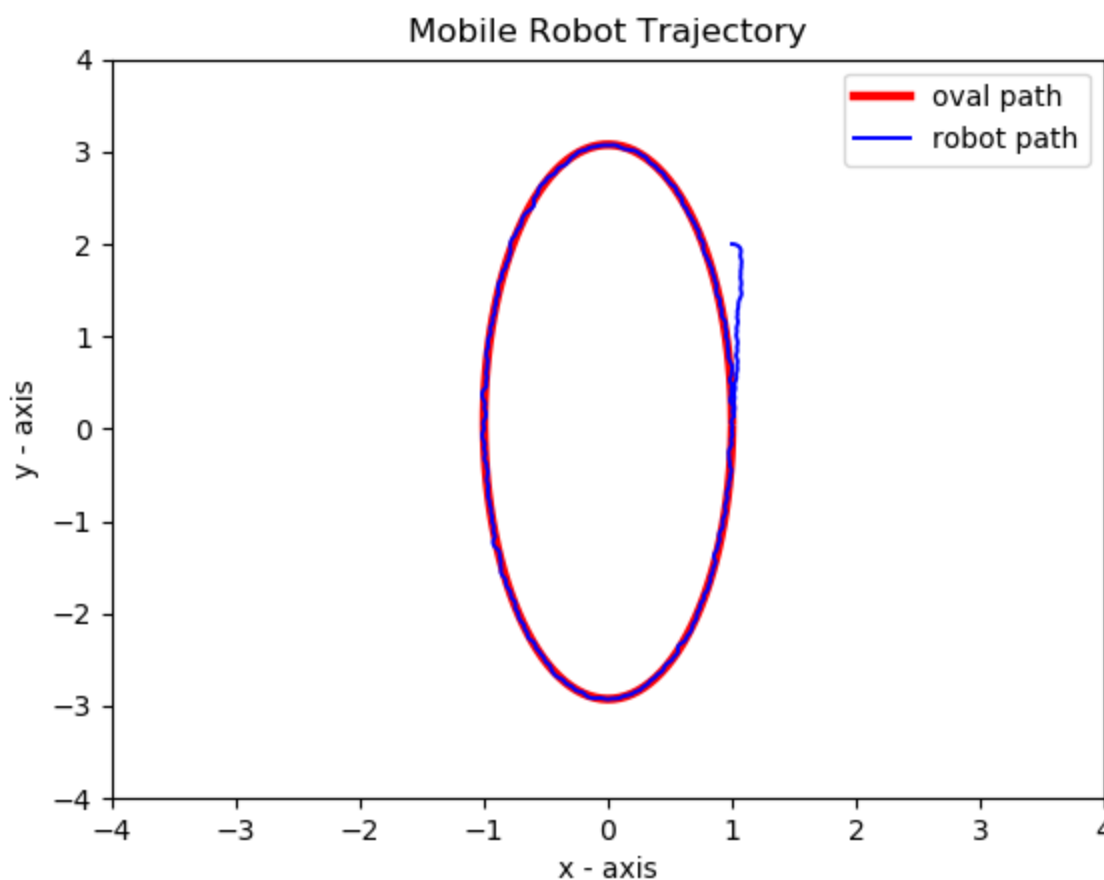
ابتدا از ضریب P شروع می کنیم. ضریب P یک ضریب تناسبی است. در واقع افزایش این ضریب با توجه به معادله ی فرکانسی زیر باعث کاهش خطا می شود. (مثلا یک کنترلر تناسبی مستقل را در نظر بگیرید. و فیدبک را مطابق پیاده سازی در Gazebo یک فیدبک واحد در نظر داریم)

$$Y(s) = (R(s) - Y(s)) * (k_p G(s)) \Rightarrow Y(s) = \frac{k_p G(s)}{1 + k_p G(s)} R(s)$$

$$\Rightarrow Error = Y(s) - R(s) = -\frac{1}{1 + k_p G(s)} R(s)$$



همانطور که از معادله‌ی فرکانسی بالا بر می‌آید، ذاتِ کنترلِ تناسبی، یک کاهنده‌ی خطا است ولی نکته‌ای که حائز اهمیت است این است که افزایشِ ضریبِ تناسبی هم نوسان را بالا می‌برد و هم زمانِ صعود را کاهش می‌دهد. در موردِ اینکه ضریبِ تناسبی تا حدودی نوسان ساز است می‌توانید به معادله‌ی صفحه‌ی قبل توجه کنید. معادله مشخصه‌ی سیستم به صورت  $1 + k_p G(s)$  می‌باشد و ضریبِ تناسبی با افزایشِ خود، اثرِ قطب‌های سیستم (ریشه‌های معادله مشخصه) را زیاد خواهد کرد که نتیجه‌ی افزایشِ ضریبِ جملاتی از چندجمله‌ای معادله‌ی مشخصه می‌باشد. از طرفی ضریبِ تناسبی سرعتِ تغییرات را بالا و زمانِ صعود را کم می‌کند که این هم نتیجه‌ی سرعتِ واکنش دهی نسبت به تغییرات است و دقیقاً مفهومِ گین را برای ورودیِ ارور داشته و واکنش را بالا می‌برد. برای مشاهده‌ی عینیِ مطالبِ فوق، اینجانب ضرایبِ تناسبی را از 1 به 40 افزایش دادم. به شکل‌های گرفته شده از مسیرِ ربات در gazebo که با matplotlib در همان کدِ پایتون در ros رسم نمودیم، توجه بفرمایید.



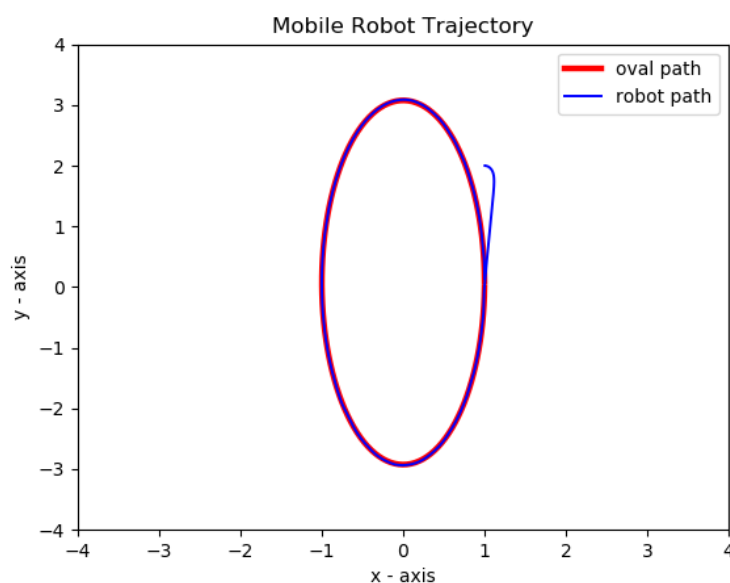
تأثیرِ افزایشِ ضریبِ تناسبی که باعثِ وجود آمدنِ عکس‌العملِ سریع و نوساناتی در سیستم شده است. هر چند که خطا را به خوبی کاهش می‌دهد

این شکل را با شکل قبلی ای که نتایج اصلی خود ارائه کردیم مقایسه کنید. شکل قبلی در واکنش به خطا smooth تر عمل میکرد حال آنکه با افزایشِ ضریبِ  $p$ ، خطوطِ آبی که موقعیتِ ربات در لحظه‌های مختلف را نشان می‌دهد، به سرعت تغییر می‌کند.

و لذا همین عامل است که یک خطای کوچک یک واکنش سریع را به دنبال دارد و باعث نوسانی شدن سیستم و کاهش زمان صعود می‌شود. پس فهمیدیم که افزایش ضریب تناسبی، کاهش خطا و کاهش زمان صعود و افزایش نوسانات را به دنبال دارد و کاهش ضریب تناسبی نیز بالعکس است یعنی افزایش خطا و کاهش نوسان و افزایش زمان صعود را به همراه خواهد داشت.

حال ضریب انتگرالی را بررسی می‌کنیم. یک انتگرال گیر در واقع خطا را می‌تواند به خوبی صفر کند. این را معادلات فرکانسی برآمده از سیستم مانند معادلاتی که برای تناسبی نوشتیم نشان می‌دهد. البته نوع ورودی هم در خطا تاثیر گذار است و ممکن است برای یک سیستم حتی یک کنترلر تناسبی مناسب و کافی باشد و نیازی به PI یا PID هم نداشته باشیم.

حال به سراغ عیب ضریب انتگرالی می‌رویم. هر چند که ضریب انتگرالگیر باعث کاهش خطا به خوبی می‌شود ولی در عین حال به شدت سیستم را نوسانی کرده و عامل overshoot های بسیار زیاد خواهد بود که اصلا مطلوب ما نیست. ولی در این مسیرها باید به یک حالت ویژه توجه کرد. این حالت بدین صورت است که ما طبق خواسته‌ی سوال تنها ضریب کنترلی انتگرالی را برای مسافت و در واقع تنظیم سرعت خطی لحاظ کردیم و تغییر در زاویه نخواهیم دید و مسیر همچنان به خوبی طی می‌شود ولی در سرعت خطی این نوسانات به خوبی مشاهده می‌شد که خواسته‌ی مسأله رسم سرعت خطی نبود. ولی برای بیضی نتیجه برای ضریب انتگرالی 1000 به صورت زیر است.

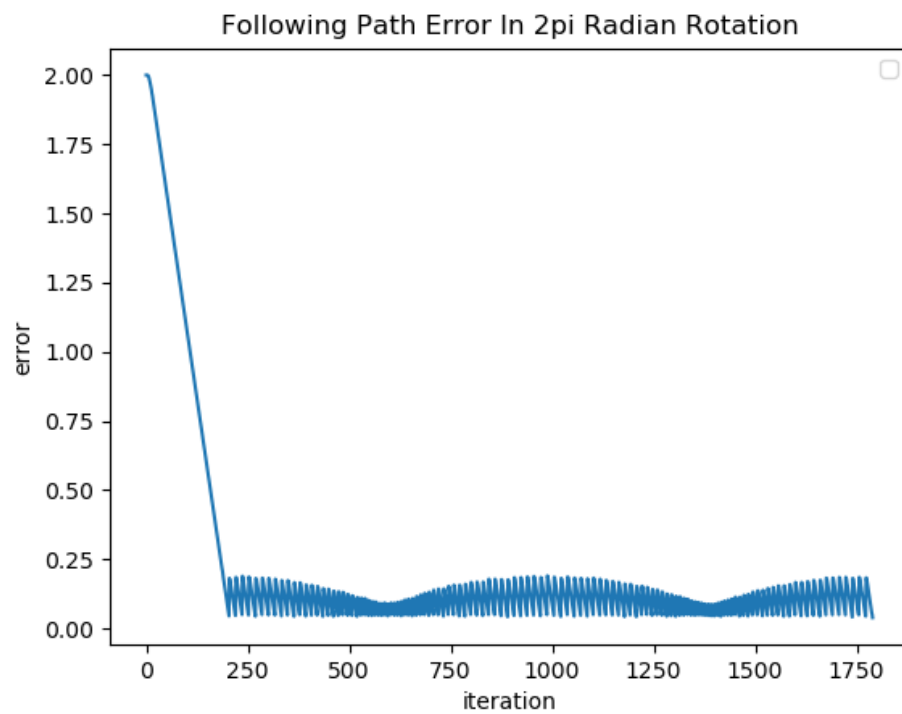


در آخرین بخش هم خطا را خواسته که ما خطا را برای بیضی در بازه‌ی 0 تا  $2\pi$  و برای مارپیچ در بازه‌ی 0 تا  $9\pi$  محاسبه کرده‌ایم و در ابتدا نشان دادیم. فورمول محاسبه خطا به صورت زیر می‌باشد.

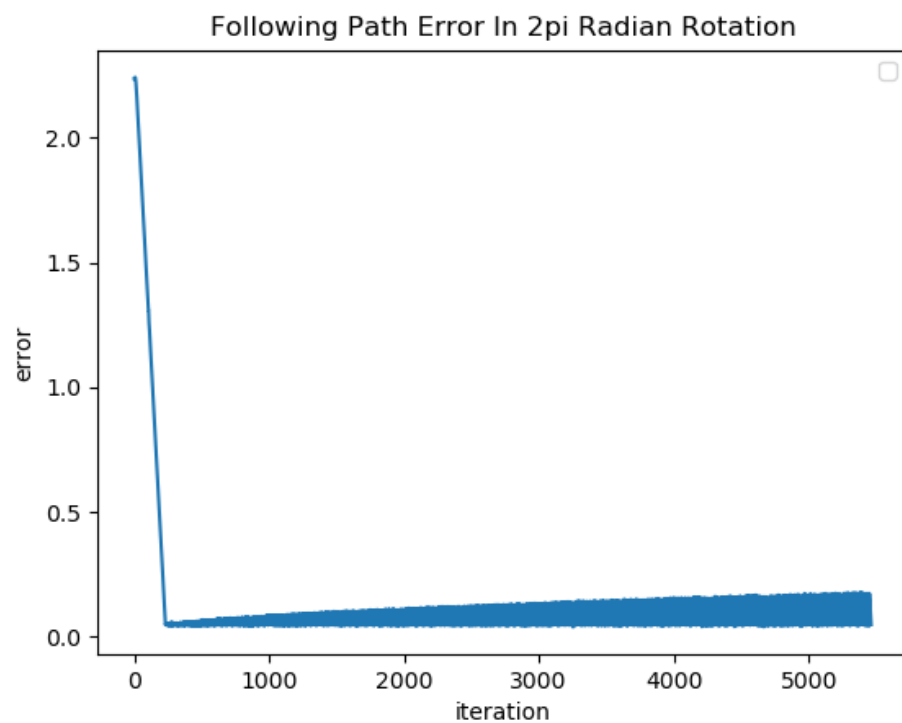
237 `error.append(sqrt((x_final - r_x_vector[i])**2+(y_final - r_y_vector[i])**2))`

لذا نتیجه‌ی ترسیم خطا برای بیضی و مارپیچ را مجددا در اینجا نیز می‌آوریم:

خطای محاسبه شده برای بیضی در بازه‌ی 0 تا  $2\pi$ :

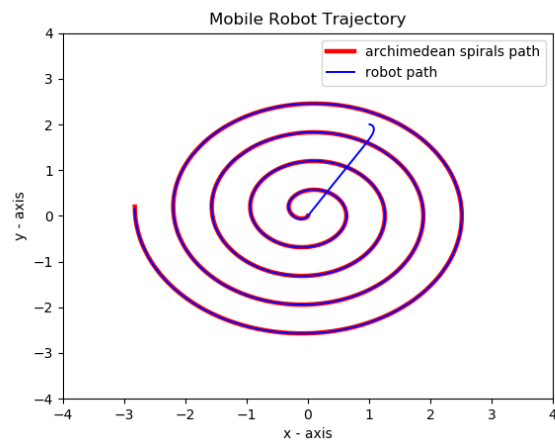
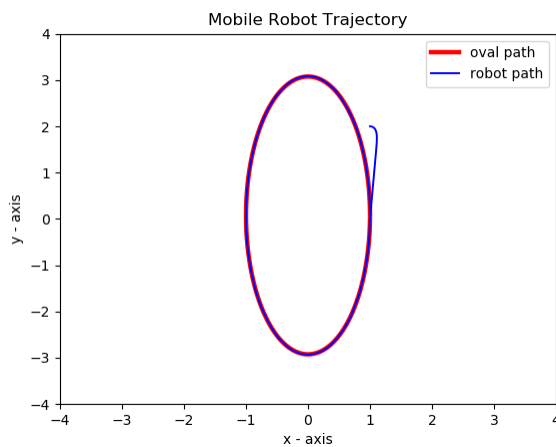


خطای محاسبه شده برای مارپیچ ارشمیدسی در بازه‌ی 0 تا  $9\pi$ :



در مورد خطایی که مشاهده می‌کنید در ابتدا که ربات در نقطه شروع است و بیشترین خطا را دارد. در مورد بیضی نیز خطا مدام به مقدار کمی، کم و زیاد می‌شود و این بدلیل تغییر نقطه هدف است که مدام در حال تغییر است تا بیضی را تشکیل دهد و اما در مورد خطای مارپیچ باید ذکر کرد که ما اشکال را بر حسب زاویه ترسیم می‌نماییم و هر چقدر شعاع افزایش یابد، طول کمان نسبت افزایش می‌یابد. لذا در اینجا همان کم و زیاد شدن ها مانند بیضی وجود دارد و در کنار آن خطای حداکثر نیز بدلیل افزایش طول کمان در حال زیاد شدن است و این دلیل اصلی رفتار خطا در مارپیچی ارشمیدسی است.

در نهایت مجددا خروجی های رسم شده از مکان ربات در gazebo را که با matplotlib ترسیم کردیم ذکر می‌کنیم.



با تشکر - بدیعی