Project Report
AE 581

To: Dr. Mohammed Ghazy and
Dr. Muhammad Faizan Mysorewala

Date: 2020/12/13

By: Ibrahim Aljalal

Outline

1-Objectives

1.1-Problem statement:

In this project the path of a Front-Wheel Bicycle/Robot will be determined by providing the front wheel angular speed and the steering angle of the robot (note the word robot will be used instead of bicycle in this project since it will not be directly controlled by a human). These two inputs (front wheel angular speed $\omega_s(t)$ in rads/second and steering angle $\alpha(t)$ in degrees) **should be provided as an expression of time** and the software should figure out the x, y and angular rotation of the whole robot **with respect to the inertial/global frame**. In a real case scenario if we want to accurately control the steering angle of the robot, we usually consider using a stepper motor. A stepper motor will basically have a discrete angle as an output (usually a multiple of 1.8 degrees). In our case we will assume it is 1 degree. **Two plots should be given to compare between the ideal case (α as an expression of time) and the real case (same α but its values are multiple of 1).** Note that **α is not restricted to ± 90 degrees** but could make a full rotation. That is because a stepper motor can make a full rotation and there are some types of bicycles which could make 360 degrees steering angle rotation.

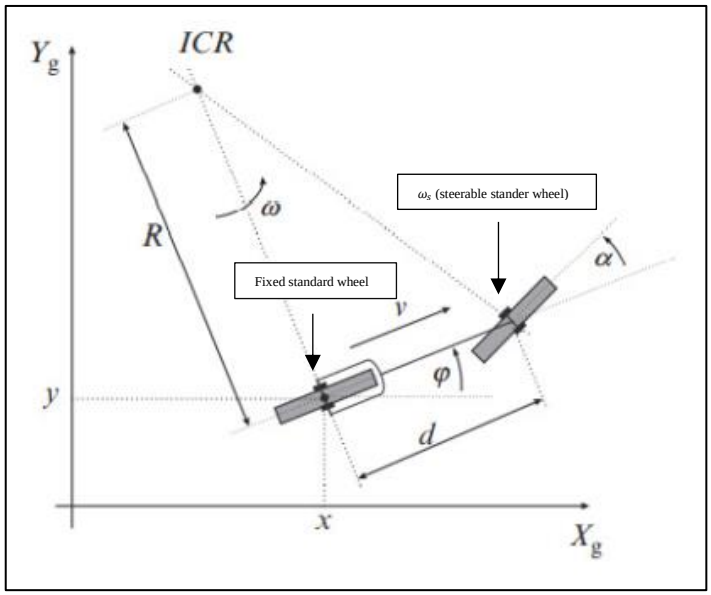


Figure 1: Problem statement illustration.

1.2-Aditional requirements:

1-Write a completely independent software for this project that works in an exe file (choose any icon)

2-The path of the robot should also be simulated by a 3D software. No need to draw the robot just an arrow model (to show the direction/x-local-frame of the robot) should be enough to demonstrate the functionality of the software.

3-In the 3D software it should take multiple inputs of $\alpha(t)$ and $\omega_s(t)$

2-Assumptions

1-There is **no slipping** in the robot wheels either in the direction of the wheel's rotation or the lateral direction (meaning the lateral direction should have zero velocity)

2-The whole **robot is rigid** (except for the movements of the wheels). That also results in the fact that the wheels should only have one point of contact with the ground.

3-The planes of the wheels are **perpendicular with the ground**.

4-The robot will have **two spherical wheels** (not shown in Figure 1) **to balance it** if its speed is slow or zero (note that the spherical wheels do not add any constraints to the robot since they are omnidirectional so we could model the robot as if the spherical wheels are not there)

5-The **ground is planer** (flat)

6-The **kinetic constraints or devices constraints in the robot will not be considered**. For example, $\omega_s = 50 \sin(t)$ could not be accurately generated by the motor or might exceeds its limit.

7-The **error in the discrete integration is small**. To avoid large error in integration we should mainly avoid two types of functions. 1-very high frequency functions such as $f(t) = \sin(50t)$ and 2- functions that sharply increases/decreases with time such as $f(t) = t^8$. Of course, we could decrease dt to capture the changes but that will result in a lot of computation. Also, in real life scenario the input usually should not be that complicated.

3-Specifications and Approach

$r = 2\ cm$ (radius of the two wheels of the robot). **Given information**

$d = 15\ cm$ (distance between the centres of the wheels). **Given information**

$\omega_s = f_{\omega_s}(t)\ rads/second$ (the steering angle of the front wheel). **Input information**

$\alpha = f_{\alpha}(t)$ Degrees (The steering angle of the front wheel). **Input information**

$f(t)$= mathematical expression where t is in seconds.

Main Output:

$x\ (globa)\ in\ cm, y\ (global)\ in\ cm\ and\ \varphi(robot\ angle)\ in\ radins$

The approach used for finding $x, y\ and\ \varphi$ will be the same as the one provided in the example project provided by the instructor

4-Mathematical Models

$v_s = r\omega_s(t)$ (linear velocity of the front wheel)

$v(t) = v_s(t)\cos(\alpha(t))$ (linear velocity of the front wheel in the local x axis of the robot)

$\omega(t) = \frac{v_s(t)}{d} sin(\alpha(t))$ (angular velocity of the whole robot in the global z axis. Not ω_s)

$$x(k+1) = x(k) + v(k)\cos\big(\varphi(k)\big)\,dt$$

$$y(k+1) = y(k) + v(k)\sin\big(\varphi(k)\big)\,dt$$

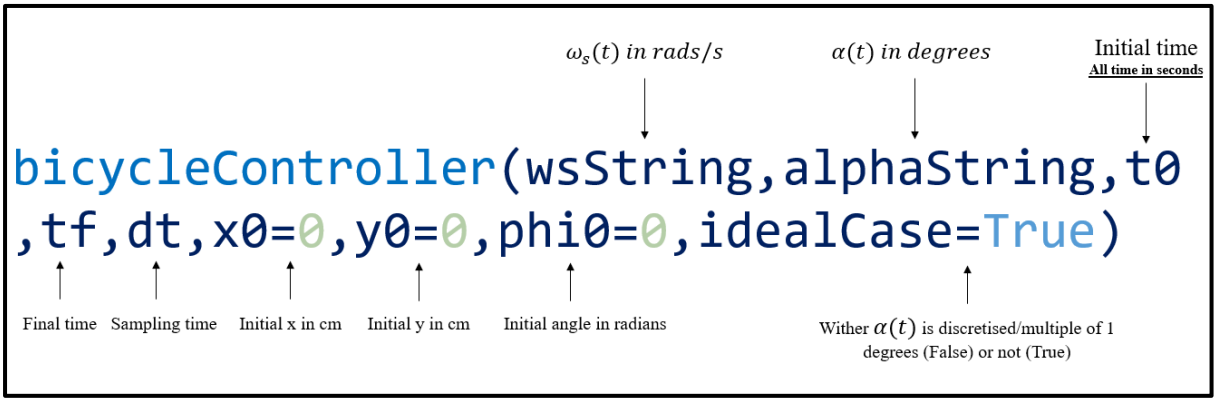
$$\varphi(k+1) = \varphi(k) + \omega(k)\,dt$$

5-Solution (Code)

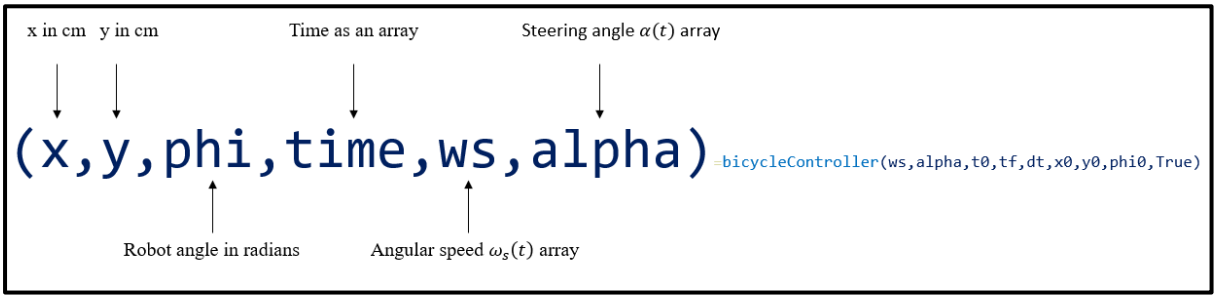
5.1-bicycleController function

Most of this project was made by writing a python code. So, it is more connivant to use the code instead. However, since bicycleController is one of the main functions in this project it will be explained here:

Input:



Output:



5.2-AE581.py library

QR Code:

Or URL:

https://drive.google.com/drive/folders/10lfCmvTrE-5VhvSnPXavYXgv9DEstlqU?usp=sharing

5.3-AE581.exe File

QR Code:

Or URL:

https://drive.google.com/drive/folders/1TVt0TEHlBd1e0lPAyhht08lNzCN3sauv?usp=sharing

5.4-AE581.mkv 3D simulation (in Blender software)

QR Code:

Or URL:

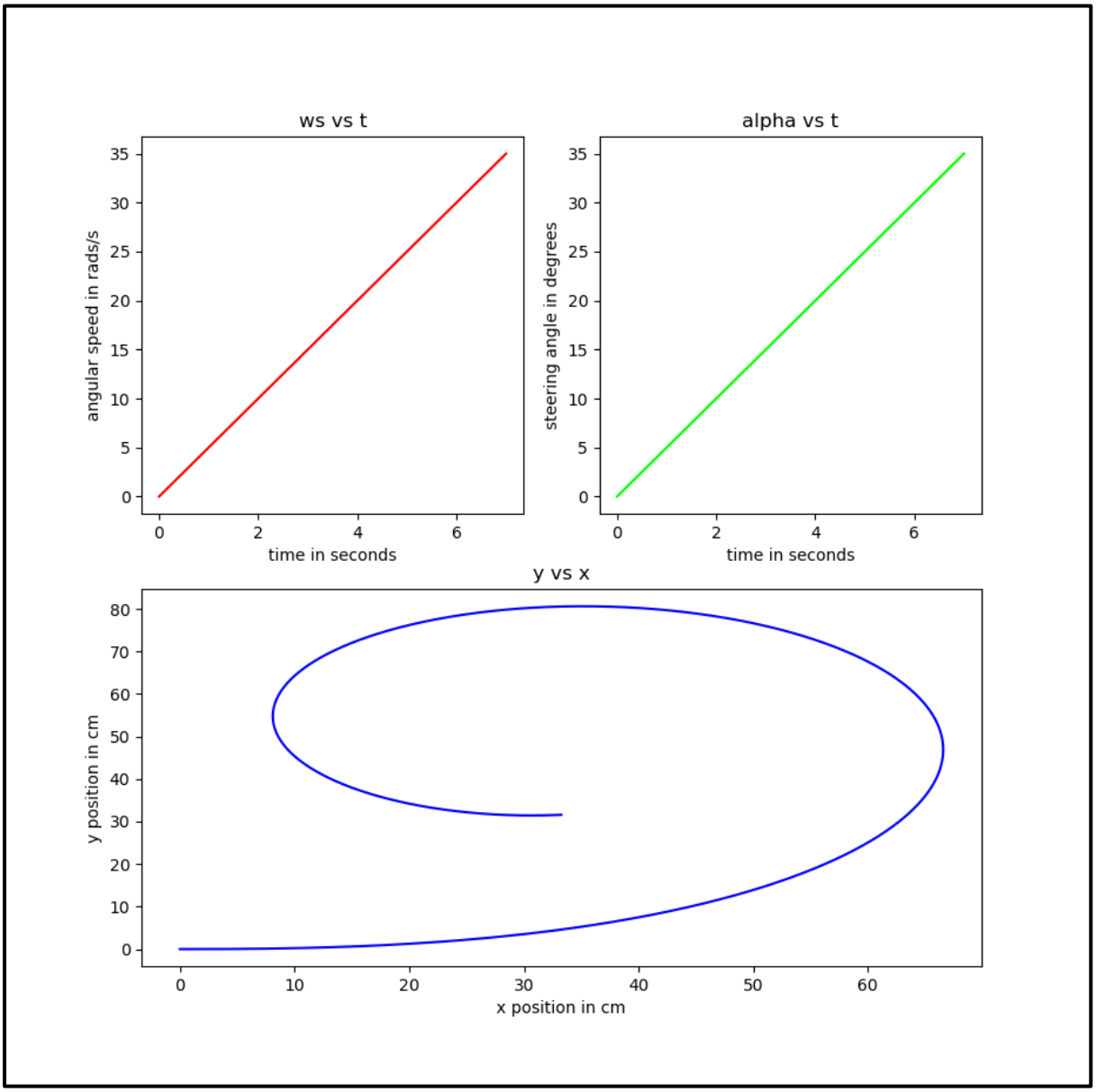
https://drive.google.com/drive/folders/1EbIEYm7_R0kmDSTeHw1_qPPw-3BGzfxD?usp=sharing

6-Results and Analysis

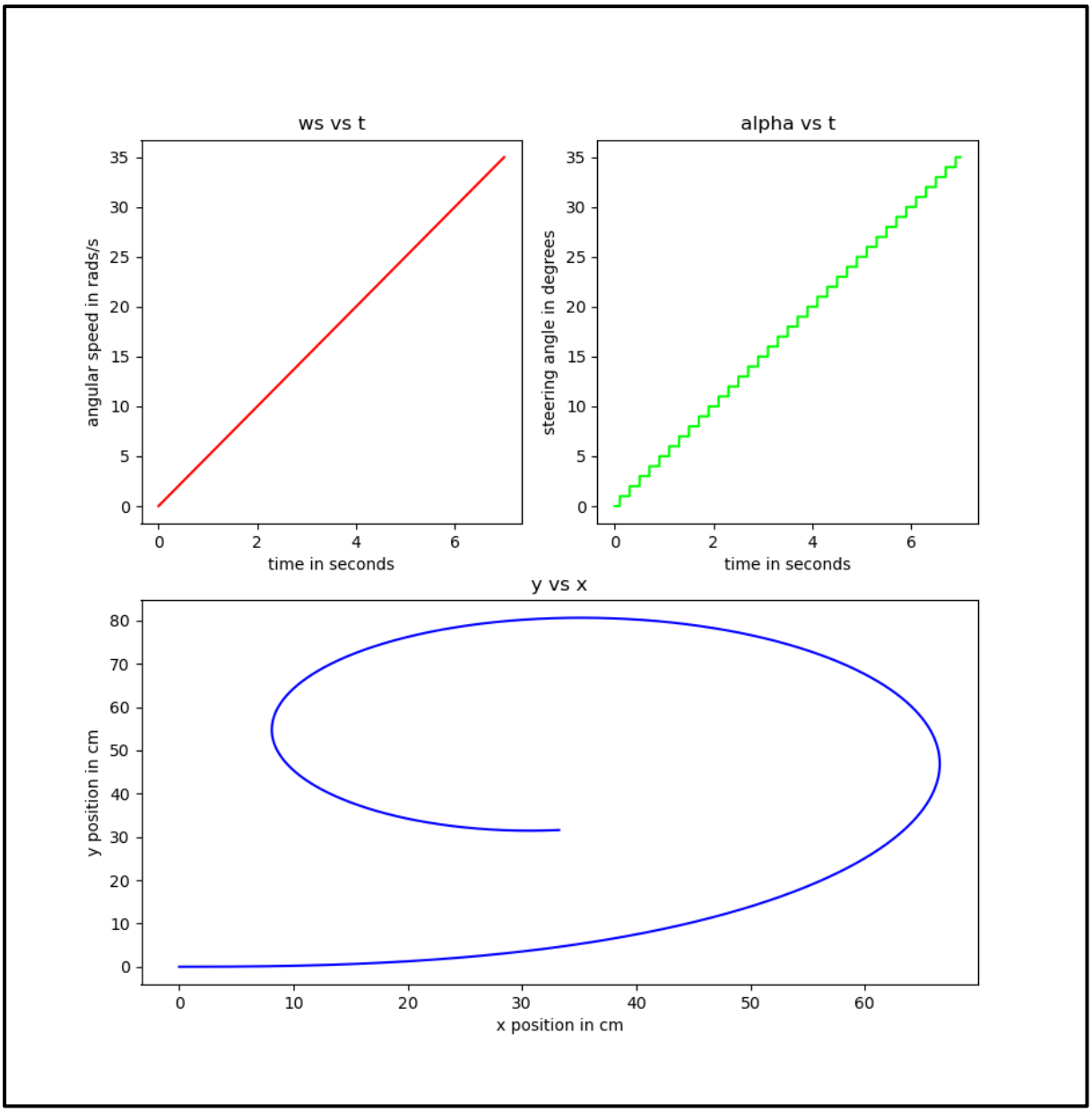
In the fowling results we will be by mainly changing in $\omega_s(t)\ and\ \alpha(t)$

6.1-Results for $\omega_s(t) = "5 * t"\ and\ \alpha(t) = "5 * t"$

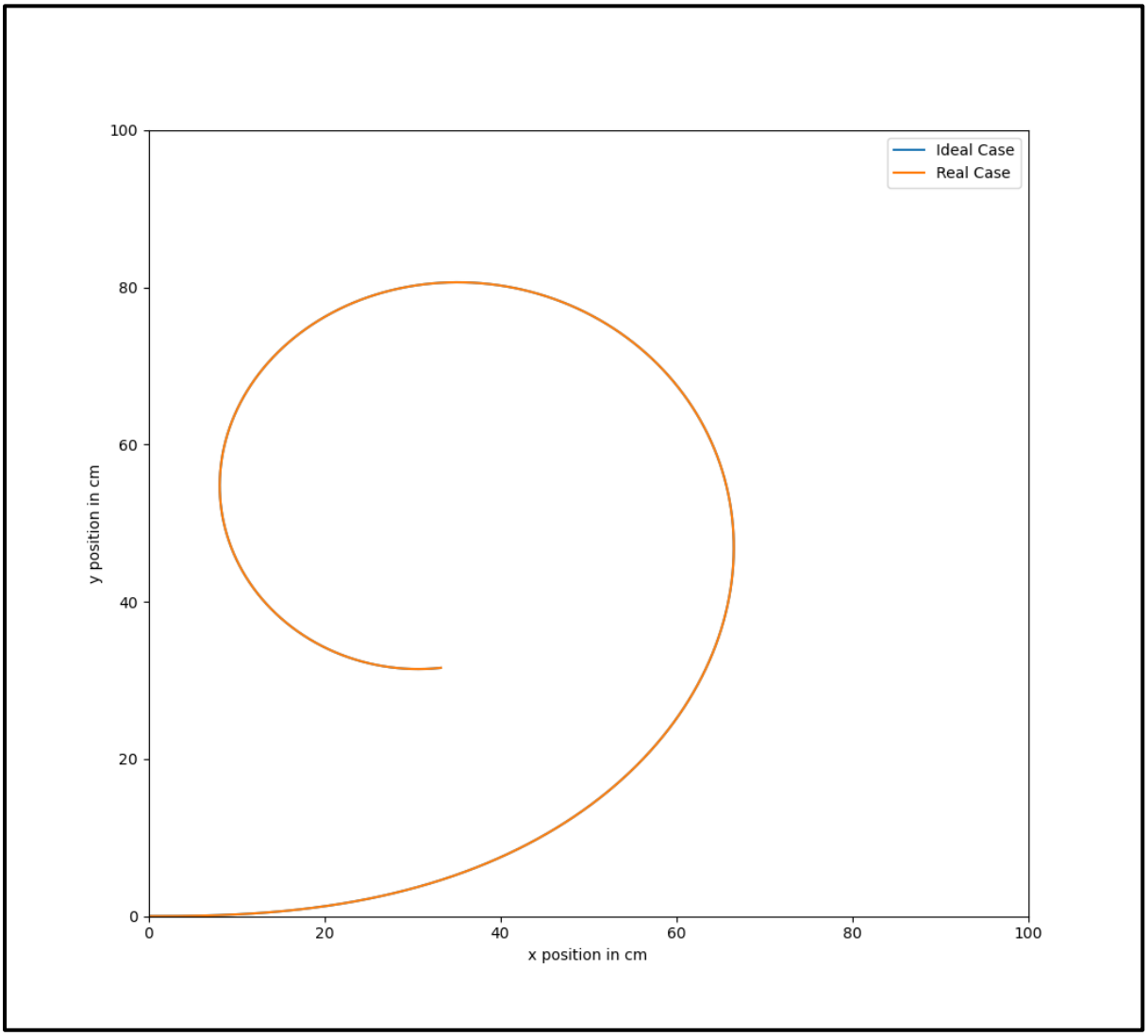
6.1.1-Ideal case:



6.1.2-Actual case:



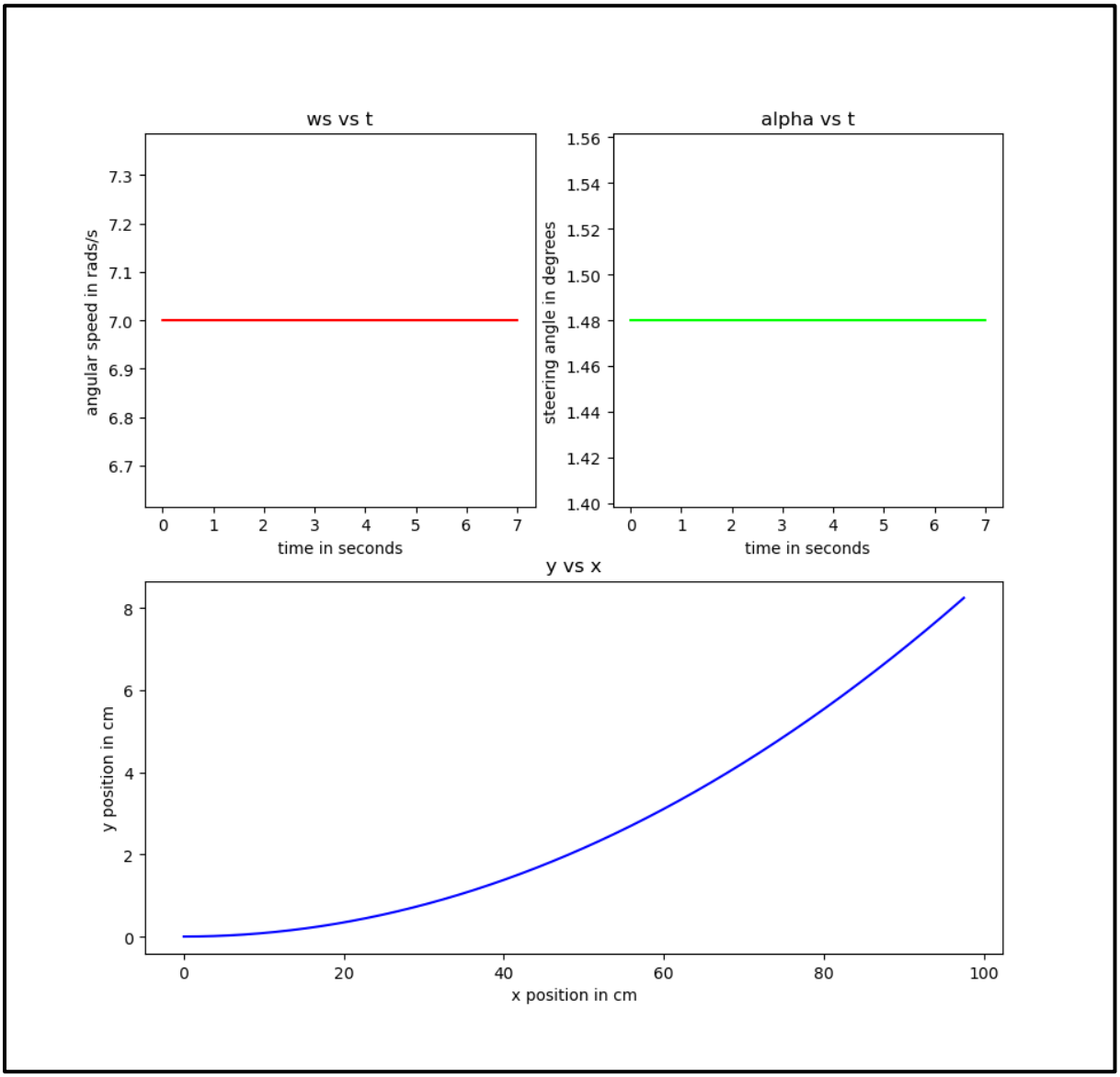
6.1.3-Actual case vs ideal case:



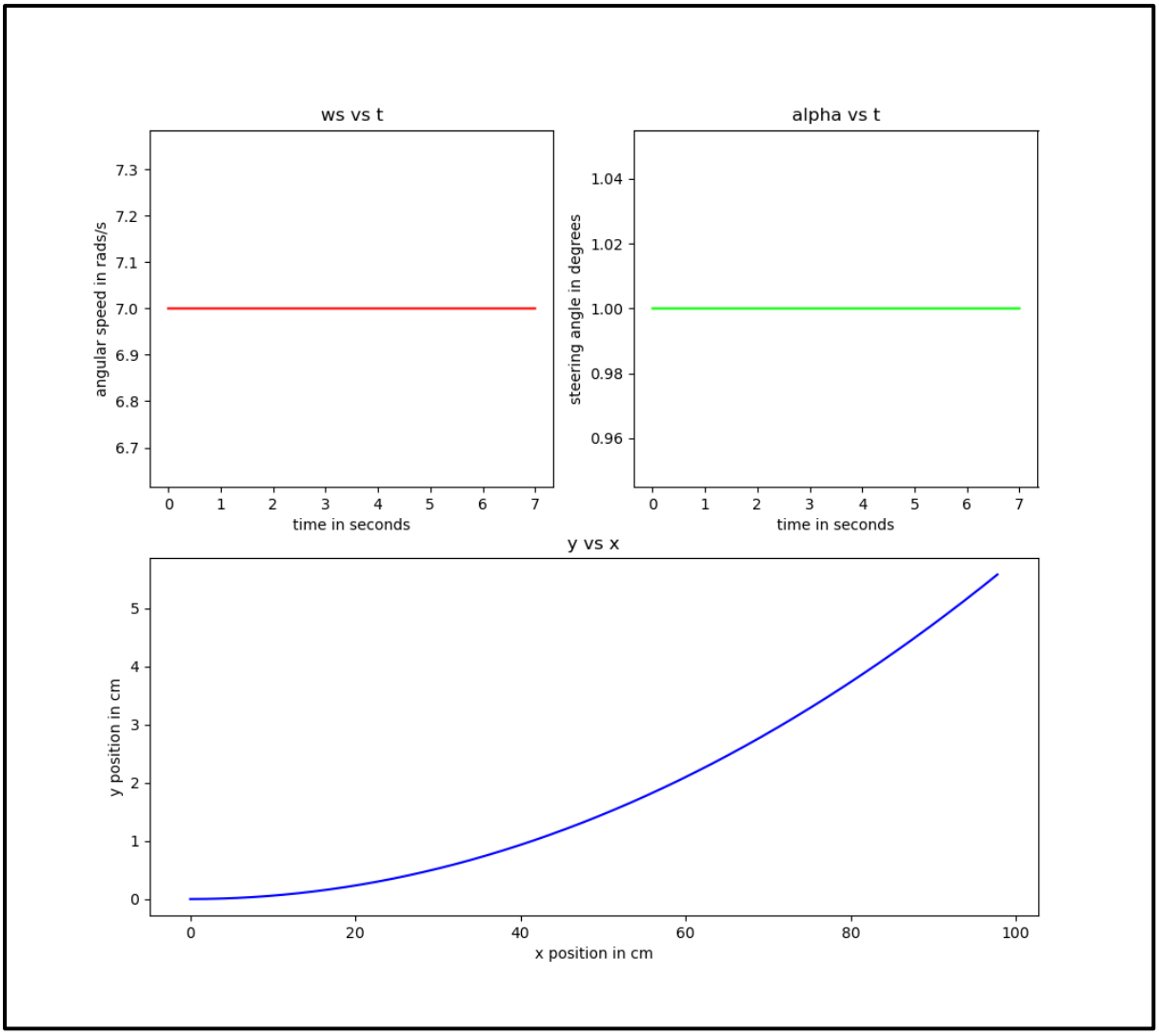
The difference between the ideal and actual final positions is **0.0054 cm.**

6.2-Results for $\omega_s(t) = "7"\ and\ \alpha(t) = "1.48"$

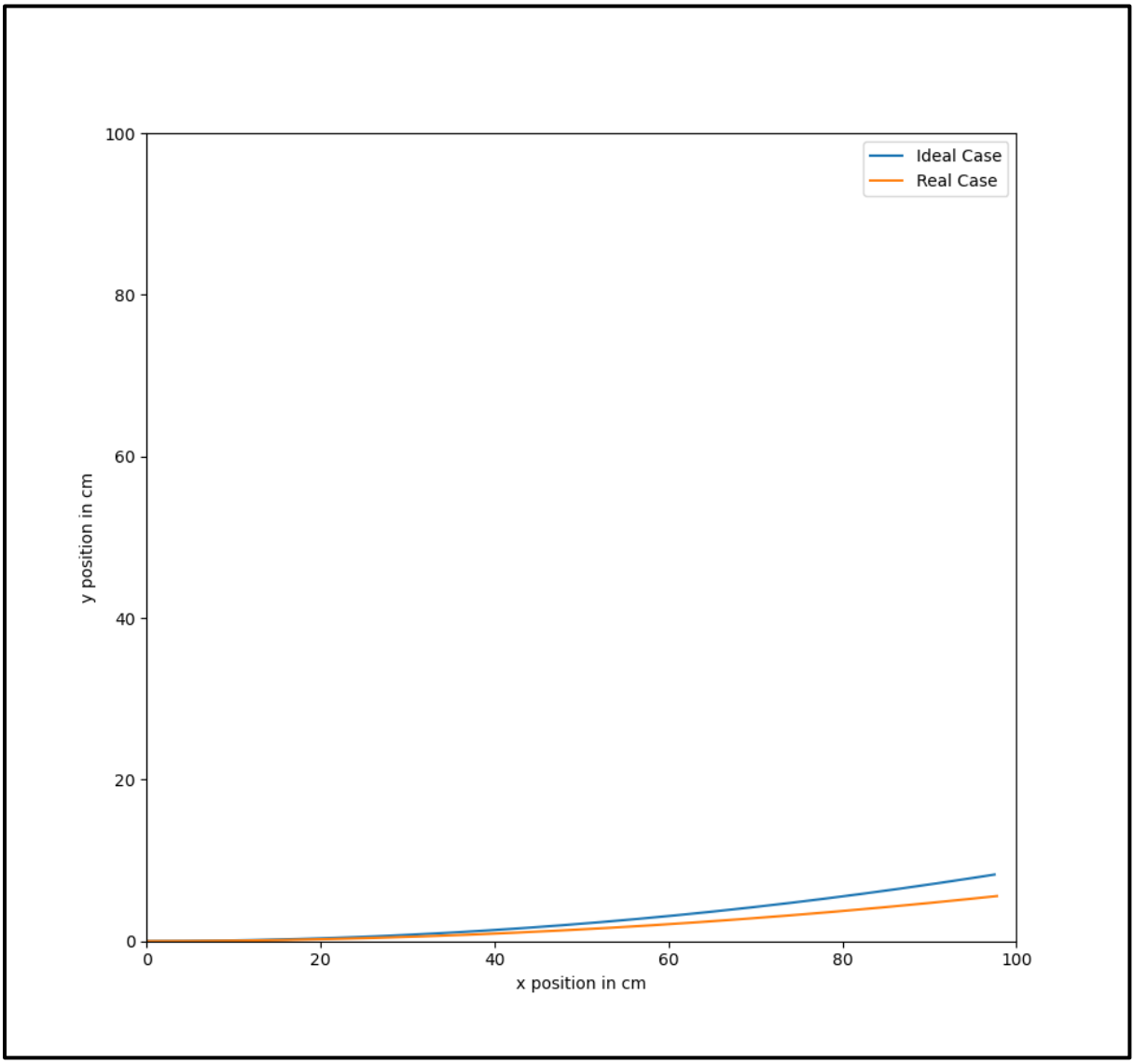
6.2.1-Ideal case:



6.2.2-Actual case:



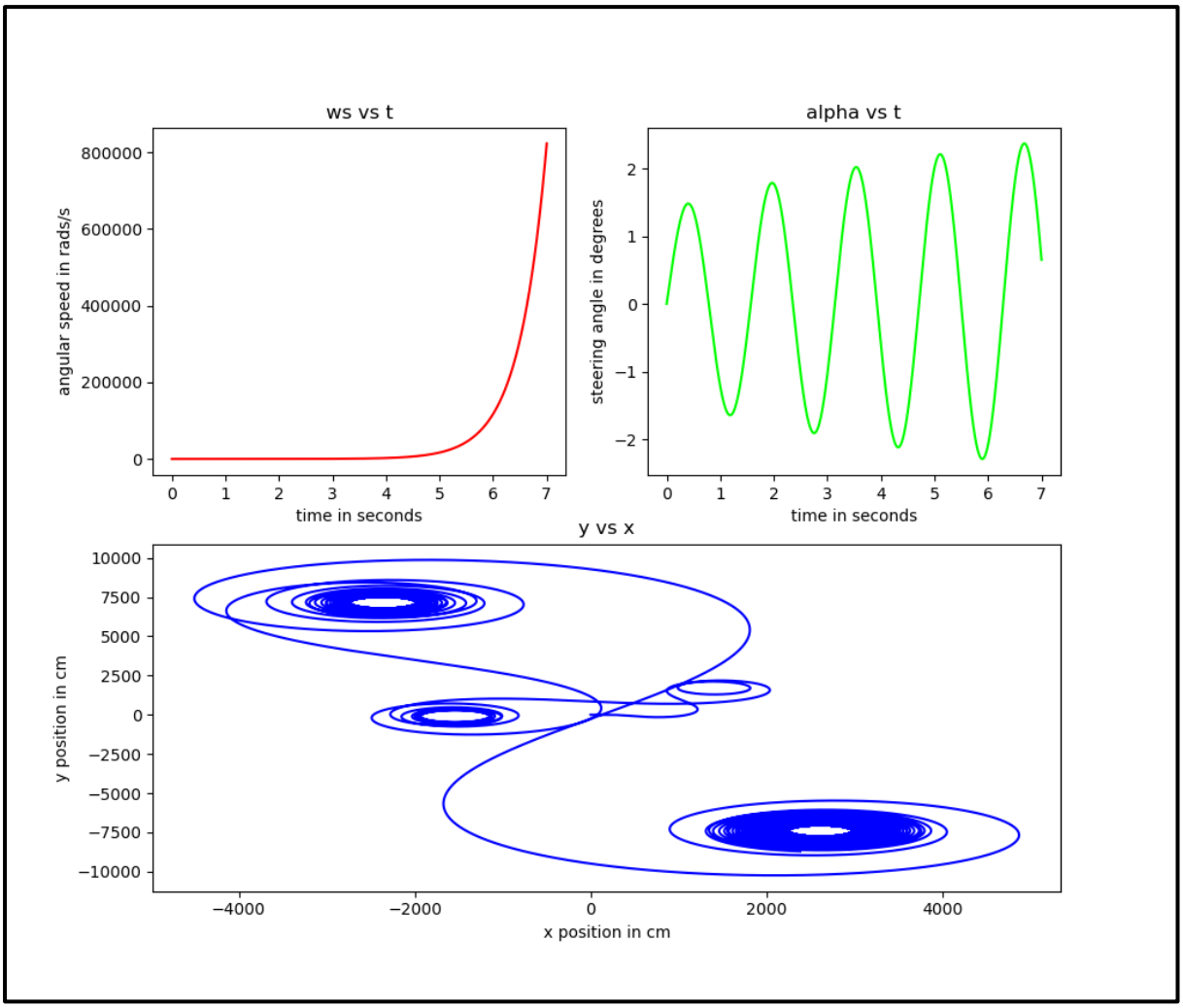
6.2.3-Actual case vs ideal case:



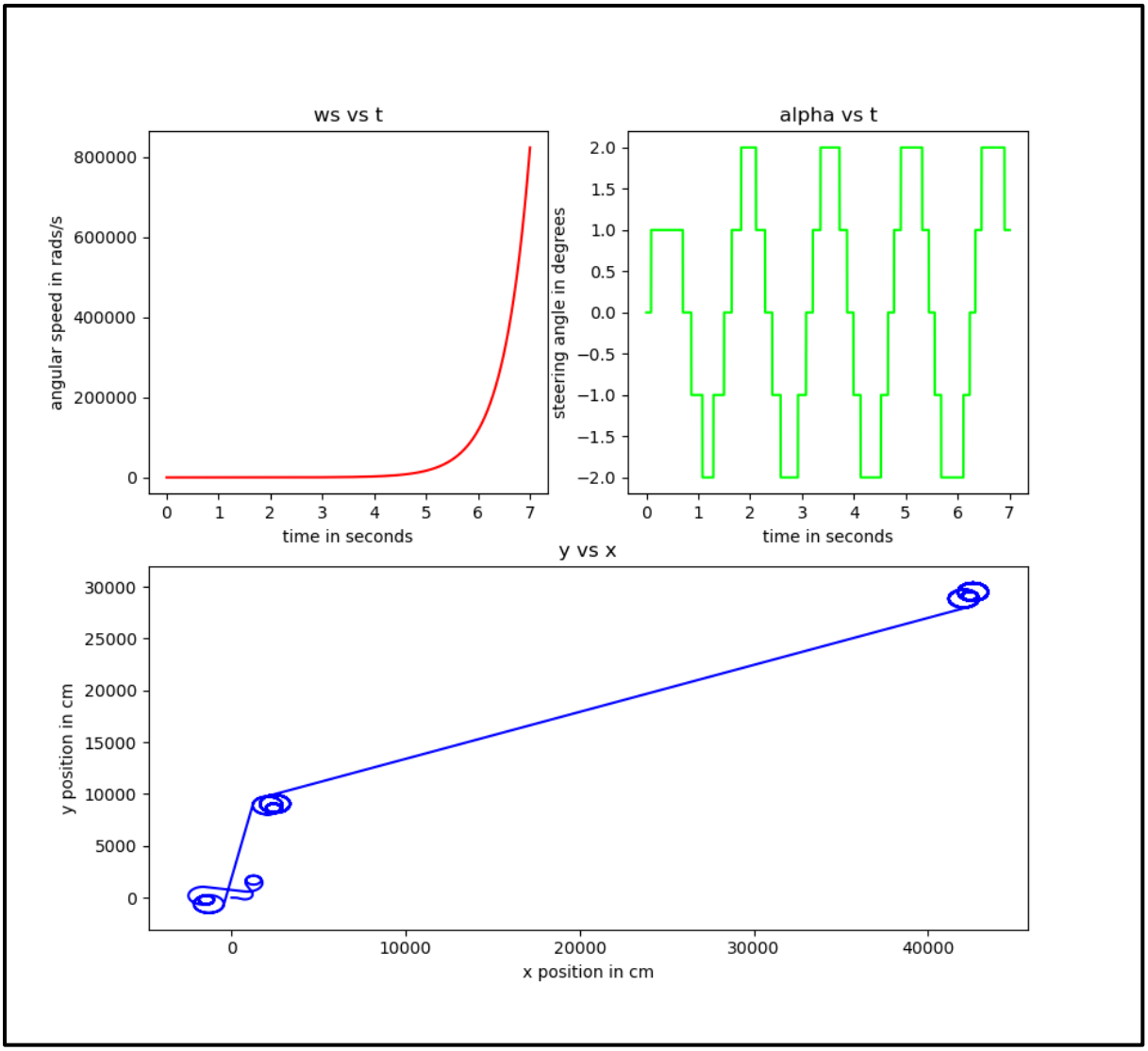
The difference between the ideal and actual final positions **is 2.68 cm.**

6.3-Results for $\omega_s(t) =$ "7 ** t" and $\alpha(t) =$ "log(t + 4) * sin (4 * t)":

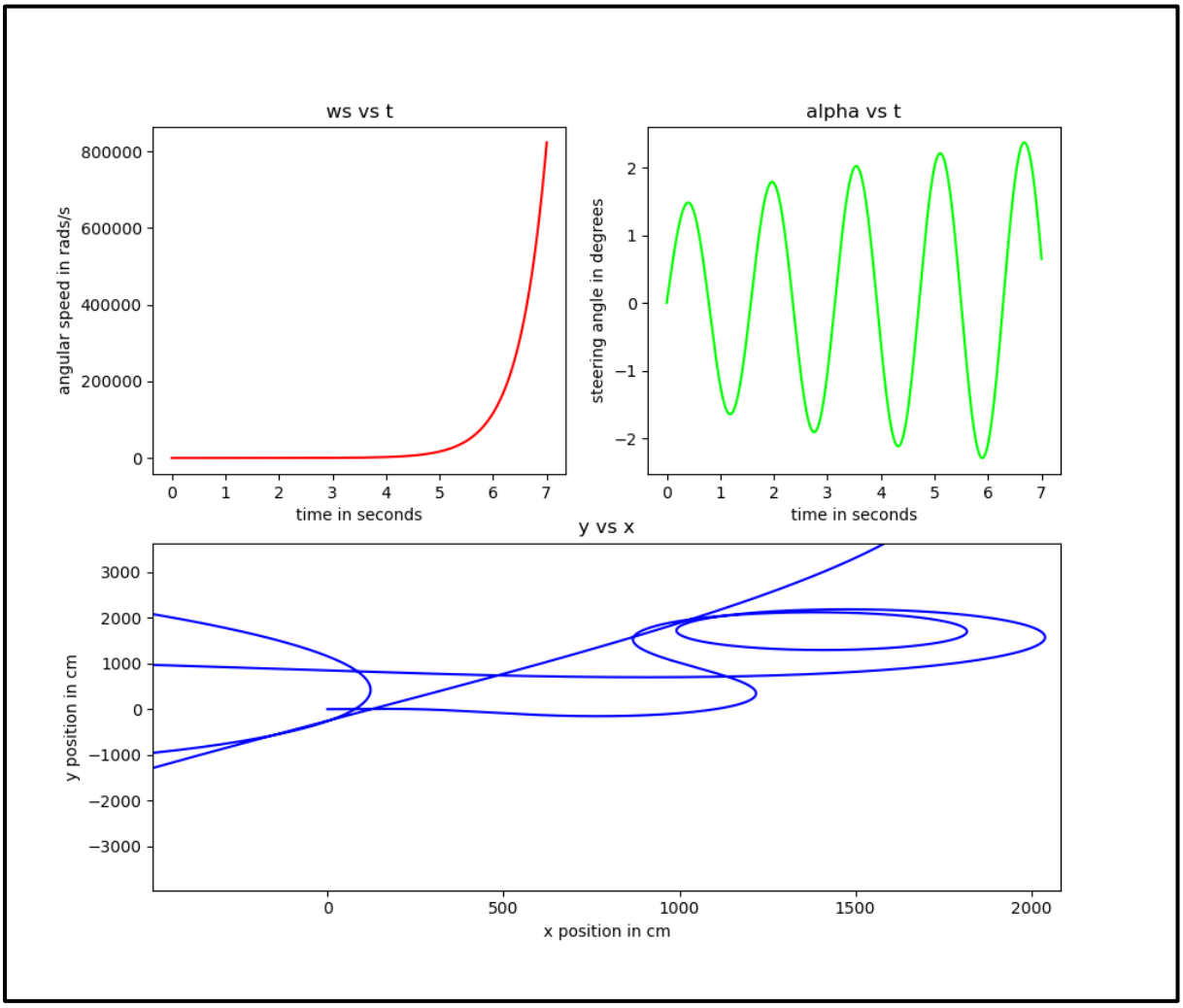
6.3.1-Ideal case:



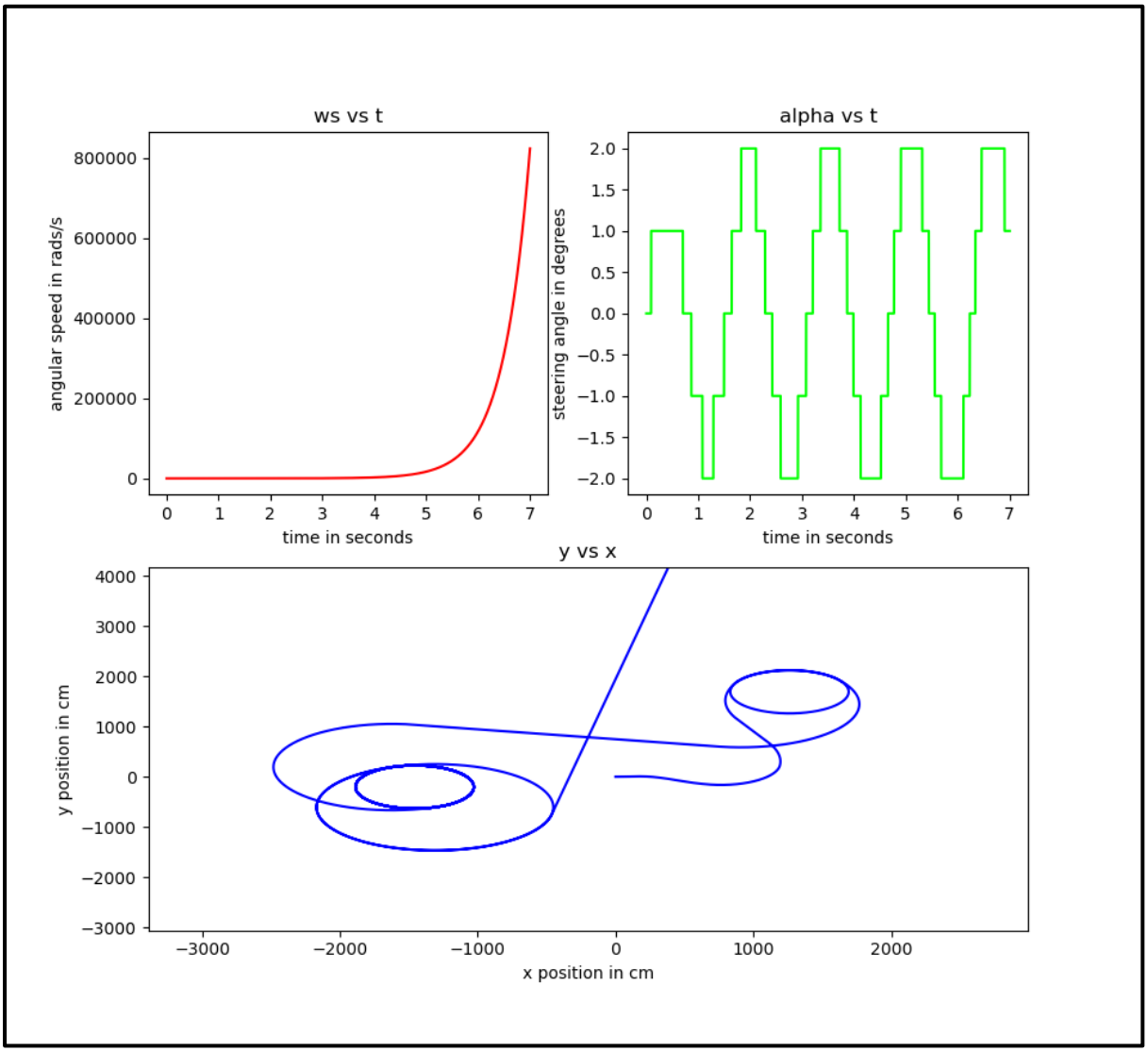
6.3.2-Actual case:



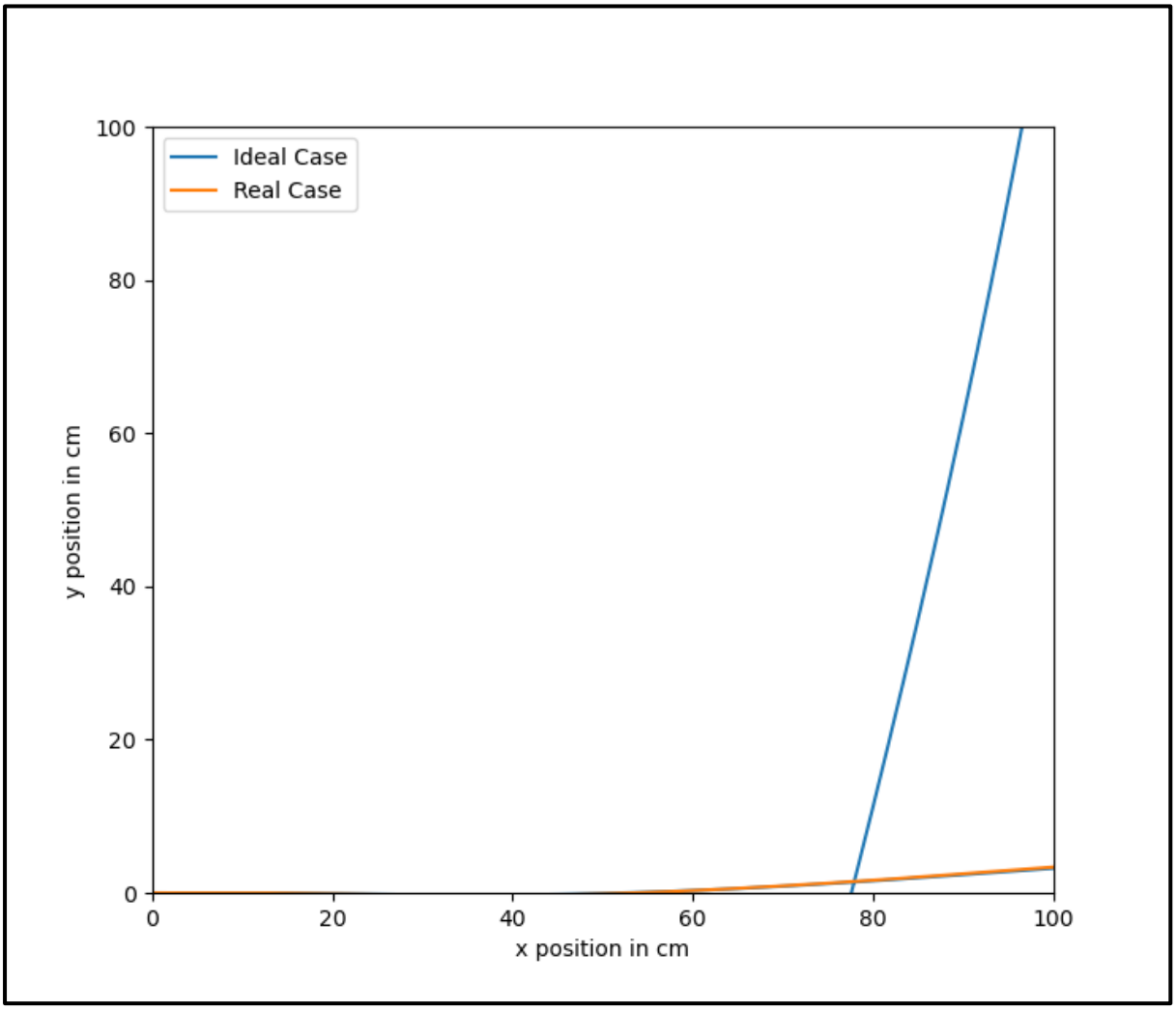
6.3.3-Ideal case (zoomed at the starting point):



6.3.4-Actual case (zoomed at the starting point):



6.3.5-Actual case vs ideal case:



This time unlike the previous experiments. The difference between the ideal and actual path is very clear.

The difference between the ideal and actual final positions **is 54664.73 cm.**

To catch the details of the two complicated expressions as much as possible we made dt unlike the previous two experiments much lower ($\boldsymbol{dt = \frac{1}{1000000} seconds}$). It took the computer about two minutes to show the results

7-Conclusion

One of the main conclusions that we clearly can inspect from these experiments is that if we use a stepper motor to control our robot with 1 degree of increment the error is very low. Since actual steeper motors have a step angle which is somewhat close to 1 degree than using them in a robot is actually a good idea to make very close paths to the ideal case one. Also, by small adjustment of stepper motors the step could be much lower than 1.8 degrees if we use micro-stepping.

The reason for the large and clear difference of the third experiment is mainly because of something which is called **the butterfly effect.** For example, if we have a complex differential equation and if we slightly change the initial conditions of the system (last 0.00001 digit) the result will be completely different than the first case.

References

1-https://www.iconarchive.com/show/matrilineare-icons-by-sora-meliae/Mimes-image-x-ico-icon.html

2- https://www.youtube.com/user/schafer5

3- https://www.youtube.com/user/sentdex

4-https://www.youtube.com/watch?v=UZX5kH72Yx4

5- https://www.blender.org/