

Home Work 1

Name : Mostroor Mofiz Atman

ID : 1921079642

Course : CSE495A

Section : 1

Submitted to :

Dr. Shahreza Siddique (sns1)

Associate Professor

Department of Electrical & Computer Engineering

North South University.

Date : 07-03-2024.

Ans to the QNO-1 (i)

(a) Unicycle Model :

State Space Equation (Differential equation):

$$\dot{x} = v \cos \theta$$

$$\dot{y} = v \sin \theta$$

$$\dot{\theta} = \omega$$

State variables :

(i) x and y : Position coordinates of the robot in a 2D plane.

(ii) θ : Orientation angle of the robot with respect to a reference axis.

Control variables :

(i) v : Linear velocity of the robot.

(ii) ω : Angular velocity of the robot.

(b) Differential drive robot :

State Space Equation (Differential equation):

$$\dot{x} = \frac{v_L + v_R}{2} \cos \theta$$

$$\dot{y} = \frac{v_L + v_R}{2} \sin \theta$$

$\dot{\theta} = \frac{v_R - v_L}{L}$; where L is the distance between the two wheels.

State variables:

- (i) x and y : Position coordinates of the robot in a 2D plane.
- (ii) θ : Orientation angle of the robot with respect to a reference axis.

Control variables:

- (i) v_L : Linear velocity of the left wheel.
- (ii) v_R : Linear velocity of the right wheel.

(c) Simplified Car Model:

State Space Equation (Differential equation):

$$\dot{x} = v \cos \theta$$

$$\dot{y} = v \sin \theta$$

$$\dot{\theta} = \frac{v \tan \phi}{L}, \text{ where } L \text{ is the distance between front and rear axles}$$

$$\dot{v} = a, \text{ where } a \text{ is the acceleration.}$$

State variables:

- (i) x and y : Position coordinates of the car in a 2D plane.

(ii) θ : Orientation angle of the car with respect to a reference axis.

(iii) v : Linear velocity of the car.

Control variables:

ϕ : Steering angle of the front wheels.

6

Ans to the QNO-1(ii)

State variables representation:

State variables represent the current state of the system, such as position, orientation and velocity.

Control variables representation:

Control variables represent the inputs or actions applied to the system to influence its state evolution, such as velocities, accelerations or steering angles.

Ans to the QNO-1(iii)

Unicycle Model	Differential Drive Robot	Simplified Car Model
(1) <u>State variables:</u> x (position), y (position), θ (orientation angle)	(1) <u>State variables:</u> x (position), y (position), θ (orientation angle)	(1) <u>State variables:</u> x (position), y (position), θ (orientation angle), v (linear velocity).
(2) <u>Control variables:</u> v (linear velocity), ω (angular velocity).	(2) <u>Control variables:</u> v_L (left wheel velocity), v_R (right wheel velocity)	(2) <u>Control variables:</u> v (linear velocity), ϕ (steering angle)

<p>(3) Suitable for robots with independent control over linear and angular velocities. Provides simple motion control but lacks precise positioning.</p>	<p>(3) Suitable for robots with independently controlled wheels. Allows for precise motion control including turning in place and following curved paths.</p>	<p>(3) Suitable for car-like robots with steering capabilities. Allows for precise trajectory following and dynamic speed adjustment.</p>
<p>(4) wheeled robots, drones etc are applications.</p>	<p>(4) Mobile robots, robotic vehicles, tractors etc are applications</p>	<p>(4) Autonomous cars, car-like robots etc are applications.</p>

Answer to the Question no. – 2(a, b)

Google Colab Code: Source code file has also attached(Name:
HomeWork1_Question_2(a,_b).ipynb)

```
1 from google.colab import drive
2
3 # Mount Google Drive
4 drive.mount('/content/drive')
5
6 import numpy as np
7 import matplotlib.pyplot as plt
8 from matplotlib.animation import FuncAnimation
9
10 # Define control inputs
11 def control_input(t):
12     v = np.where((t >= 0) & (t <= 10), 1, 0)
13     w = np.where(((t >= 0.5) & (t <= 1.5)) | ((t >= 6) & (t <= 7)), 3, 0)
14     w = np.where(((t >= 2) & (t <= 3)) | ((t >= 4) & (t <= 5)) | ((t >=
15 8) & (t <= 9)), -3, w)
16     return v, w
17
18 # Define differential equations
19 def f(x, y, theta, v, w):
20     x_dot = v * np.cos(theta)
21     y_dot = v * np.sin(theta)
22     theta_dot = w
23     return x_dot, y_dot, theta_dot
24
25 # Simulation parameters
26 dt = 0.1
27 T = 10
28
29 # Initial conditions
30 x = 0
31 y = 0
32 theta = 1
33
34 # Time array
35 t_data = np.arange(0, T + dt, dt)
36 # Control inputs
37 v, w = control_input(t_data)
38
39 # Initialize data storage
40 x_data = np.zeros(len(t_data))
41 y_data = np.zeros(len(t_data))
42 theta_data = np.zeros(len(t_data))
43
44 # Euler's method loop
45 for i in range(len(t_data)):
46     x_dot, y_dot, theta_dot = f(x, y, theta, v[i], w[i])
```

```

47     x += x_dot * dt
48     y += y_dot * dt
49     theta += theta_dot * dt
50     x_data[i] = x
51     y_data[i] = y
52     theta_data[i] = theta
53
54 # Plot the results
55 plt.figure(figsize=(10, 6))
56 plt.subplot(2, 2, 1)
57 plt.plot(x_data, y_data)
58 plt.xlabel('X (m)')
59 plt.ylabel('Y (m)')
60 plt.title('Robot trajectory')
61
62 plt.subplot(2, 2, 2)
63 plt.plot(t_data, x_data)
64 plt.xlabel('Time (s)')
65 plt.ylabel('X (m)')
66 plt.title('X vs time')
67
68 plt.subplot(2, 2, 3)
69 plt.plot(t_data, y_data)
70 plt.xlabel('Time (s)')
71 plt.ylabel('Y (m)')
72 plt.title('Y vs time')
73
74 plt.subplot(2, 2, 4)
75 plt.plot(t_data, theta_data)
76 plt.xlabel('Time (s)')
77 plt.ylabel('Theta (rad)')
78 plt.title('Theta vs time')
79
80 plt.tight_layout()
81 plt.show()
82
83 # Animation
84 fig, ax = plt.subplots()
85 line, = ax.plot([], [], 'bo-', lw=2)
86
87 def init():
88     ax.set_xlim(min(x_data) - 1, max(x_data) + 1)
89     ax.set_ylim(min(y_data) - 1, max(y_data) + 1)
90     ax.set_xlabel('X (m)')
91     ax.set_ylabel('Y (m)')
92     ax.set_title('Robot Motion')
93     return line,
94
95 def update(frame):
96     line.set_data(x_data[:frame], y_data[:frame])
97     return line,
98

```

```

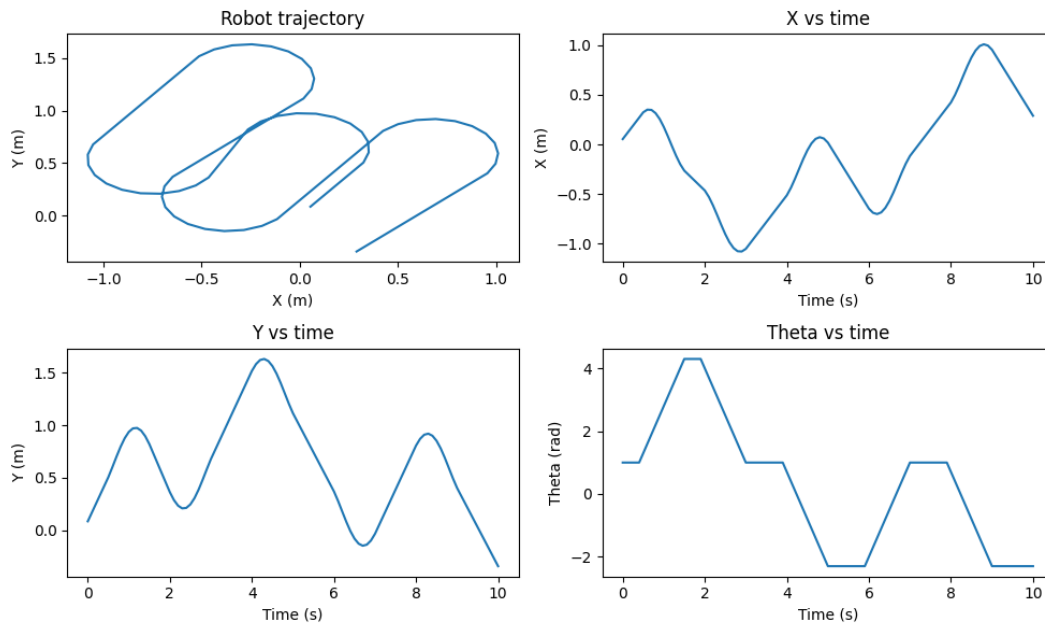
99 ani = FuncAnimation(fig, update, frames=len(t_data), init_func=init,
100 blit=True)
101 ani.save('/content/drive/My Drive/robot_motion.mp4', fps=10)

plt.show()

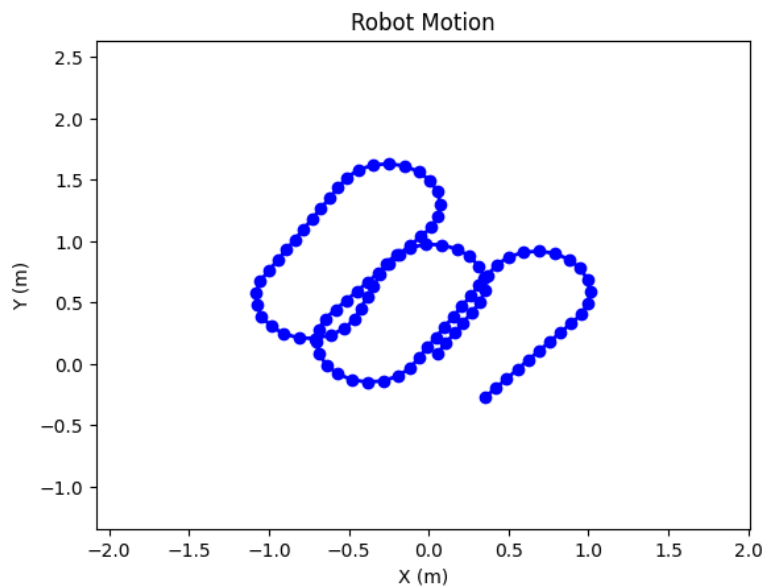
```

Output:

2(a):



2(b): A mp4 file has also attached (Name: *HomeWork1 Question 2(b).mp4*)



Updated version:

$\Delta t = 0.5, 0.1, 0.01, 0.05$.

Google Colab Code: Source code file has also attached(Name:
HomeWork1_Question_2(delta_t_updated).ipynb)

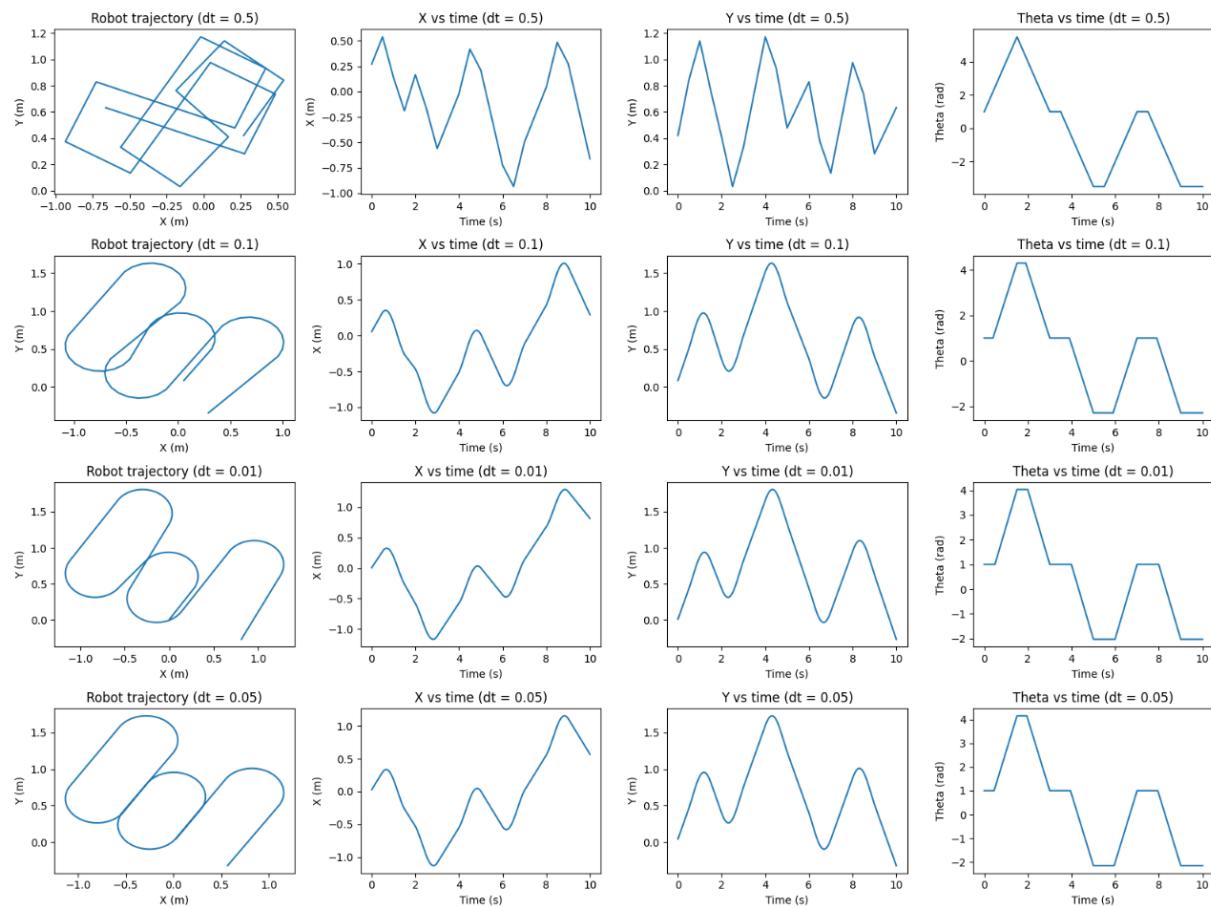
```
1 # If time step  $\Delta t = 0.5, 0.1, 0.01, 0.05$ .
2
3 import numpy as np
4 import matplotlib.pyplot as plt
5
6 # Define control inputs
7 def control_input(t):
8     v = np.where((t >= 0) & (t <= 10), 1, 0)
9     w = np.where(((t >= 0.5) & (t <= 1.5)) | ((t >= 6) & (t <= 7)), 3, 0)
10    w = np.where(((t >= 2) & (t <= 3)) | ((t >= 4) & (t <= 5)) | ((t >= 8)
11    & (t <= 9)), -3, w)
12    return v, w
13
14 # Define differential equations
15 def f(x, y, theta, v, w):
16     x_dot = v * np.cos(theta)
17     y_dot = v * np.sin(theta)
18     theta_dot = w
19     return x_dot, y_dot, theta_dot
20
21 # Simulation parameters
22 dt_values = [0.5, 0.1, 0.01, 0.05]
23 T = 10
24 # Initial conditions
25 x0 = 0
26 y0 = 0
27 theta0 = 1
28
29 # Plotting setup
30 plt.figure(figsize=(16, 12))
31 plot_idx = 1
32
33 for dt in dt_values:
34     # Time array
35     t_data = np.arange(0, T + dt, dt)
36
37     # Control inputs
38     v, w = control_input(t_data)
39
40     # Initialize data storage
41     x_data = np.zeros(len(t_data))
42     y_data = np.zeros(len(t_data))
43     theta_data = np.zeros(len(t_data))
44
```

```

45     # Euler's method loop
46     x = x0
47     y = y0
48     theta = theta0
49     for i in range(len(t_data)):
50         x_dot, y_dot, theta_dot = f(x, y, theta, v[i], w[i])
51         x += x_dot * dt
52         y += y_dot * dt
53         theta += theta_dot * dt
54         x_data[i] = x
55         y_data[i] = y
56         theta_data[i] = theta
57
58     # Plotting
59     plt.subplot(4, 4, plot_idx)
60     plt.plot(x_data, y_data)
61     plt.xlabel('X (m)')
62     plt.ylabel('Y (m)')
63     plt.title(f'Robot trajectory (dt = {dt})')
64
65     plt.subplot(4, 4, plot_idx + 1)
66     plt.plot(t_data, x_data)
67     plt.xlabel('Time (s)')
68     plt.ylabel('X (m)')
69     plt.title(f'X vs time (dt = {dt})')
70
71     plt.subplot(4, 4, plot_idx + 2)
72     plt.plot(t_data, y_data)
73     plt.xlabel('Time (s)')
74     plt.ylabel('Y (m)')
75     plt.title(f'Y vs time (dt = {dt})')
76
77     plt.subplot(4, 4, plot_idx + 3)
78     plt.plot(t_data, theta_data)
79     plt.xlabel('Time (s)')
80     plt.ylabel('Theta (rad)')
81     plt.title(f'Theta vs time (dt = {dt})')
82
83     plot_idx += 4
84
85 plt.tight_layout()
plt.show()

```

Output:



Answer to the Question no. – 3(a, b)

Google Colab Code: Source code file has also attached(Name: *HomeWork1_Question_3(a_b).ipynb*)

```
1 from google.colab import drive
2
3 # Mount Google Drive
4 drive.mount('/content/drive')
5
6 import numpy as np
7 import matplotlib.pyplot as plt
8 from matplotlib.animation import FuncAnimation
9
10 # Define control inputs
11 def control_input(t):
12     w1 = np.where((t >= 4) & (t <= 6), 12, np.where((t >= 6) & (t <= 8),
13 12, 1))
```

```

14     wr = np.where(((t >= 0.5) & (t <= 1.5)) | ((t >= 2) & (t <= 4)), 12,
15 1)
16     return wl, wr
17
18 # Define differential equations
19 def f(x, y, theta, wl, wr):
20     r = 0.1
21     L = 1
22
23     v = (wl + wr) * r / 2
24     w = (wr - wl) * r / L
25
26     x_dot = v * np.cos(theta)
27     y_dot = v * np.sin(theta)
28     theta_dot = w
29     return x_dot, y_dot, theta_dot
30
31 # Simulation parameters
32 dt = 0.1
33 T = 10
34
35 # Initial conditions
36 x = 0
37 y = 0
38 theta = 1
39
40 # Time array
41 t_data = np.arange(0, T + dt, dt)
42 # Control inputs
43 wl, wr = control_input(t_data)
44 # Initialize data storage
45 x_data = np.zeros(len(t_data))
46 y_data = np.zeros(len(t_data))
47 theta_data = np.zeros(len(t_data))
48
49 # Euler's method loop
50 for i in range(len(t_data)):
51     x_dot, y_dot, theta_dot = f(x, y, theta, wl[i], wr[i])
52     x += x_dot * dt
53     y += y_dot * dt
54     theta += theta_dot * dt
55     x_data[i] = x
56     y_data[i] = y
57     theta_data[i] = theta
58
59 # Plot the results
60 plt.figure(figsize=(10, 6))
61
62 plt.subplot(2, 2, 1)
63 plt.plot(x_data, y_data)
64 plt.xlabel('X (m)')
65 plt.ylabel('Y (m)')

```



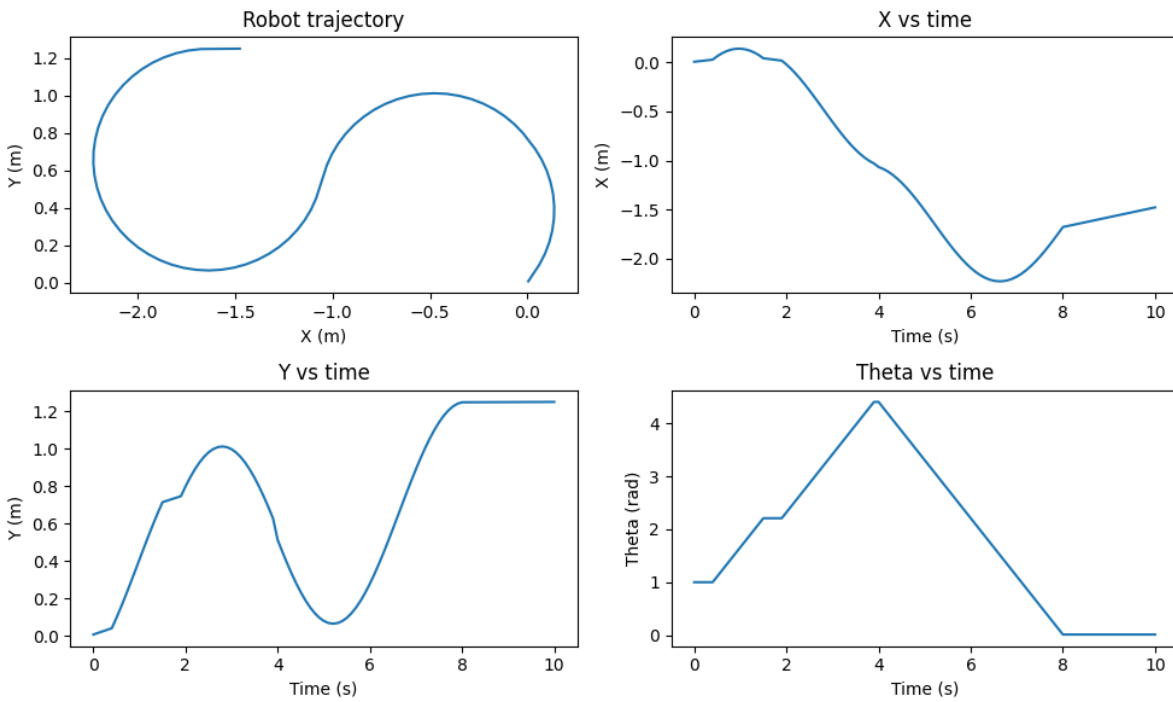
```

66 plt.title('Robot trajectory')
67
68 plt.subplot(2, 2, 2)
69 plt.plot(t_data, x_data)
70 plt.xlabel('Time (s)')
71 plt.ylabel('X (m)')
72 plt.title('X vs time')
73
74 plt.subplot(2, 2, 3)
75 plt.plot(t_data, y_data)
76 plt.xlabel('Time (s)')
77 plt.ylabel('Y (m)')
78 plt.title('Y vs time')
79
80 plt.subplot(2, 2, 4)
81 plt.plot(t_data, theta_data)
82 plt.xlabel('Time (s)')
83 plt.ylabel('Theta (rad)')
84 plt.title('Theta vs time')
85
86 plt.tight_layout()
87 plt.show()
88
89 # Animation
90 fig, ax = plt.subplots()
91 line, = ax.plot([], [], 'bo-', lw=2)
92
93 def init():
94     ax.set_xlim(min(x_data) - 1, max(x_data) + 1)
95     ax.set_ylim(min(y_data) - 1, max(y_data) + 1)
96     ax.set_xlabel('X (m)')
97     ax.set_ylabel('Y (m)')
98     ax.set_title('Robot Motion')
99     return line,
100
101 def update(frame):
102     line.set_data(x_data[:frame], y_data[:frame])
103     return line,
104
105 ani = FuncAnimation(fig, update, frames=len(t_data), init_func=init,
106 blit=True)
107 ani.save('/content/drive/My Drive/robot_trajectory.mp4', fps=10)
108
109 plt.show()

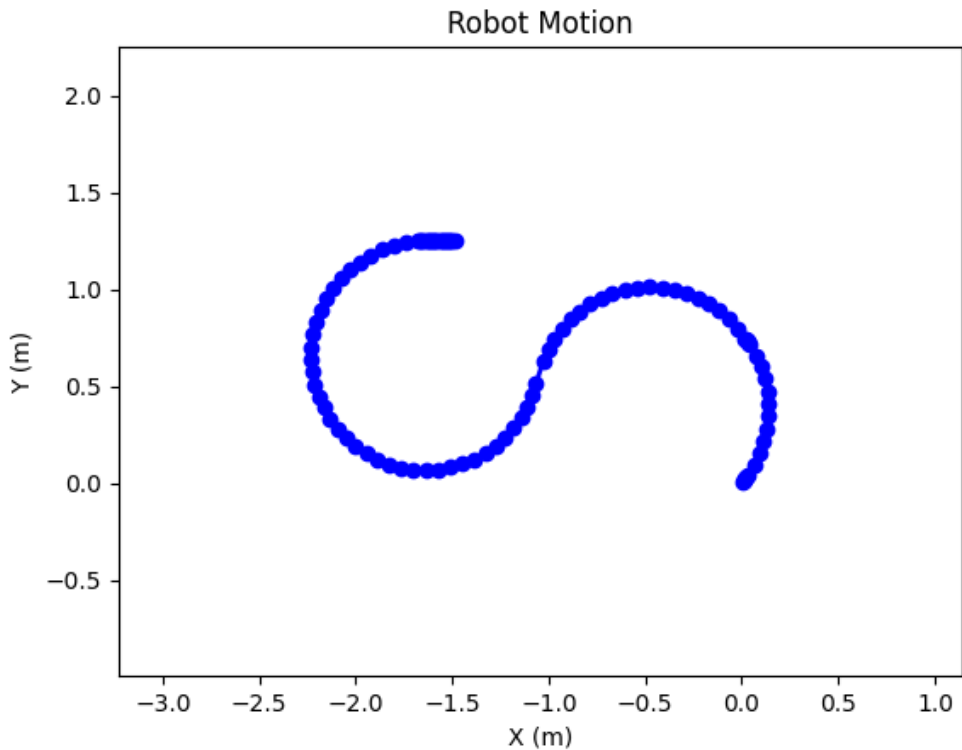
```

Output:

3(a):



3(b): A mp4 file has also attached (Name: *HomeWork1 Question 3(b).mp4*)



Updated version:

$\Delta t = 0.5, 0.1, 0.01, 0.05$.

Google Colab Code: Source code file has also attached(Name:
HomeWork1_Question_3(delta_t_updated).ipynb)

```
1 # If time step  $\Delta t = 0.5, 0.1, 0.01, 0.05$ .
2
3 import numpy as np
4 import matplotlib.pyplot as plt
5
6 # Define control inputs
7 def control_input(t):
8     wl = np.where((t >= 4) & (t <= 6), 12, np.where((t >= 6) & (t <= 8),
9 12, 1))
10     wr = np.where(((t >= 0.5) & (t <= 1.5)) | ((t >= 2) & (t <= 4)), 12,
11 1)
12     return wl, wr
13
14 # Define differential equations
15 def f(x, y, theta, wl, wr):
16     r = 0.1
17     L = 1
18
19     v = (wl + wr) * r / 2
20     w = (wr - wl) * r / L
21
22     x_dot = v * np.cos(theta)
23     y_dot = v * np.sin(theta)
24     theta_dot = w
25     return x_dot, y_dot, theta_dot
26
27 # Simulation parameters
28 dt_values = [0.5, 0.1, 0.01, 0.05]
29 T = 10
30
31 # Initial conditions
32 x = 0
33 y = 0
34 theta = 1
35
36 # Plotting setup
37 plt.figure(figsize=(16, 12))
38 plot_idx = 1
39
40 for dt in dt_values:
41     # Time array
42     t_data = np.arange(0, T + dt, dt)
43
44     # Control inputs
```

```

45     wl, wr = control_input(t_data)
46
47     # Initialize data storage
48     x_data = np.zeros(len(t_data))
49     y_data = np.zeros(len(t_data))
50     theta_data = np.zeros(len(t_data))
51
52     # Euler's method loop
53     for i in range(len(t_data)):
54         x_dot, y_dot, theta_dot = f(x, y, theta, wl[i], wr[i])
55         x += x_dot * dt
56         y += y_dot * dt
57         theta += theta_dot * dt
58         x_data[i] = x
59         y_data[i] = y
60         theta_data[i] = theta
61
62     # Plotting
63     plt.subplot(4, 4, plot_idx)
64     plt.plot(x_data, y_data)
65     plt.xlabel('X (m)')
66     plt.ylabel('Y (m)')
67     plt.title(f'Robot trajectory (dt = {dt})')
68
69     plt.subplot(4, 4, plot_idx + 1)
70     plt.plot(t_data, x_data)
71     plt.xlabel('Time (s)')
72     plt.ylabel('X (m)')
73     plt.title(f'X vs time (dt = {dt})')
74
75     plt.subplot(4, 4, plot_idx + 2)
76     plt.plot(t_data, y_data)
77     plt.xlabel('Time (s)')
78     plt.ylabel('Y (m)')
79     plt.title(f'Y vs time (dt = {dt})')
80
81     plt.subplot(4, 4, plot_idx + 3)
82     plt.plot(t_data, theta_data)
83     plt.xlabel('Time (s)')
84     plt.ylabel('Theta (rad)')
85     plt.title(f'Theta vs time (dt = {dt})')
86
87     plot_idx += 4
88
89     plt.tight_layout()
90     plt.show()

```


Output:

