Home Work 1

Name : Mostroor Mofiz Arman

ID : 1921079642

Course : CSE495A

Section : 1

Submitted to :

Dr. Shahrenaz 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):

$$\dot{x} = \frac{V_l + V_r}{2} \cdot \cos\theta$$
$$\dot{y} = \frac{V_l + V_r}{2} \sin\theta$$
$$\dot{\theta} = \frac{V_r - V_l}{L}$$
; 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):

$$\dot{x} = v \cos\theta$$
$$\dot{y} = v \sin\theta$$
$$\dot{\theta} = \frac{v \tan\phi}{L}$$, where L is the distance between front and rear axles
$$\dot{v} = a$$, where a 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) State variables: x(position), y(position), θ(orientation angle)	(1) State variables: x(position), y(position), θ(orientation angle)	(1) State variables: x(position), y(position), θ(orientation angle), v(linear velocity).
(2) Control variables: v(linear velocity), ω(angular velocity).	(2) Control variables: V_L(left-wheel velocity), V_R(right wheel velocity)	(2) Control variables: v(linear velocity), ϕ(steering angle)

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

Answer to the Question no. – 2(a, b)

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

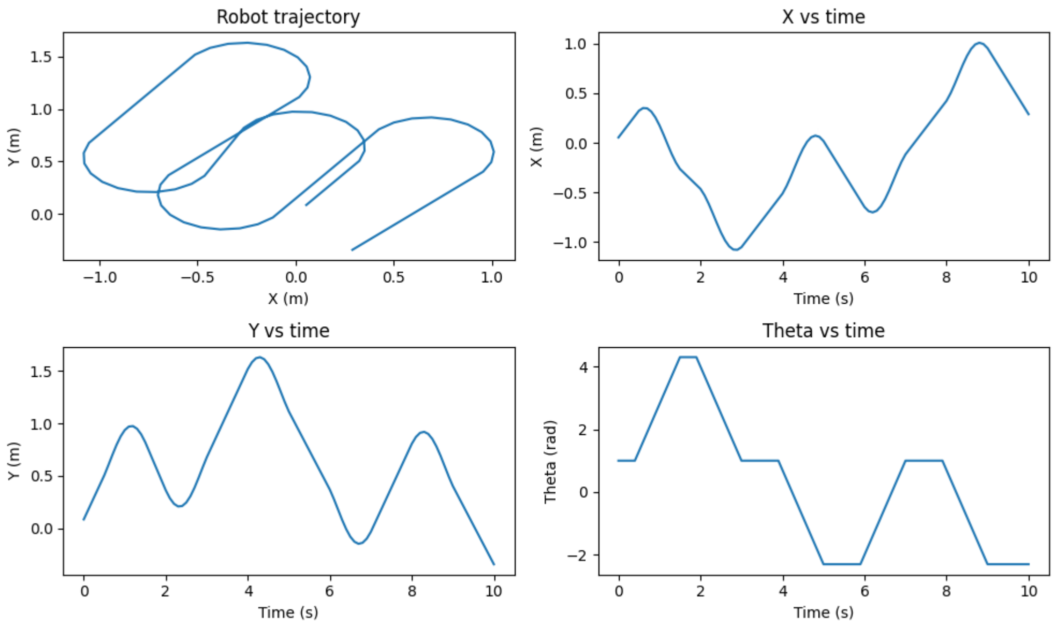
```
from google.colab import drive

# Mount Google Drive
drive.mount('/content/drive')

import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation

# Define control inputs
def control_input(t):
    v = np.where((t >= 0) & (t <= 10), 1, 0)
    w = np.where(((t >= 0.5) & (t <= 1.5)) | ((t >= 6) & (t <= 7)), 3, 0)
    w = np.where(((t >= 2) & (t <= 3)) | ((t >= 4) & (t <= 5)) | ((t >= 8) & (t <= 9)), -3, w)
    return v, w

# Define differential equations
def f(x, y, theta, v, w):
    x_dot = v * np.cos(theta)
    y_dot = v * np.sin(theta)
    theta_dot = w
    return x_dot, y_dot, theta_dot

# Simulation parameters
dt = 0.1
T = 10

# Initial conditions
x = 0
y = 0
theta = 1

# Time array
t_data = np.arange(0, T + dt, dt)
# Control inputs
v, w = control_input(t_data)

# Initialize data storage
x_data = np.zeros(len(t_data))
y_data = np.zeros(len(t_data))
theta_data = np.zeros(len(t_data))

# Euler's method loop
for i in range(len(t_data)):
    x_dot, y_dot, theta_dot = f(x, y, theta, v[i], w[i])
```

```python
    x += x_dot * dt
    y += y_dot * dt
    theta += theta_dot * dt
    x_data[i] = x
    y_data[i] = y
    theta_data[i] = theta

# Plot the results
plt.figure(figsize=(10, 6))
plt.subplot(2, 2, 1)
plt.plot(x_data, y_data)
plt.xlabel('X (m)')
plt.ylabel('Y (m)')
plt.title('Robot trajectory')

plt.subplot(2, 2, 2)
plt.plot(t_data, x_data)
plt.xlabel('Time (s)')
plt.ylabel('X (m)')
plt.title('X vs time')

plt.subplot(2, 2, 3)
plt.plot(t_data, y_data)
plt.xlabel('Time (s)')
plt.ylabel('Y (m)')
plt.title('Y vs time')

plt.subplot(2, 2, 4)
plt.plot(t_data, theta_data)
plt.xlabel('Time (s)')
plt.ylabel('Theta (rad)')
plt.title('Theta vs time')

plt.tight_layout()
plt.show()

# Animation
fig, ax = plt.subplots()
line, = ax.plot([], [], 'bo-', lw=2)

def init():
    ax.set_xlim(min(x_data) - 1, max(x_data) + 1)
    ax.set_ylim(min(y_data) - 1, max(y_data) + 1)
    ax.set_xlabel('X (m)')
    ax.set_ylabel('Y (m)')
    ax.set_title('Robot Motion')
    return line,

def update(frame):
    line.set_data(x_data[:frame], y_data[:frame])
    return line,

```

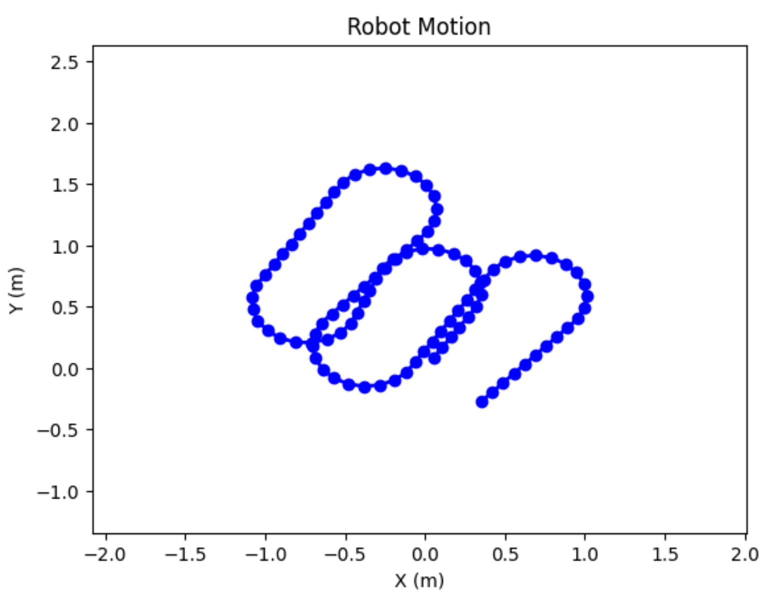
```
ani = FuncAnimation(fig, update, frames=len(t_data), init_func=init,
blit=True)
ani.save('/content/drive/My Drive/robot_motion.mp4', fps=10)

plt.show()
```

Output:

2(a):



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



Updated version:

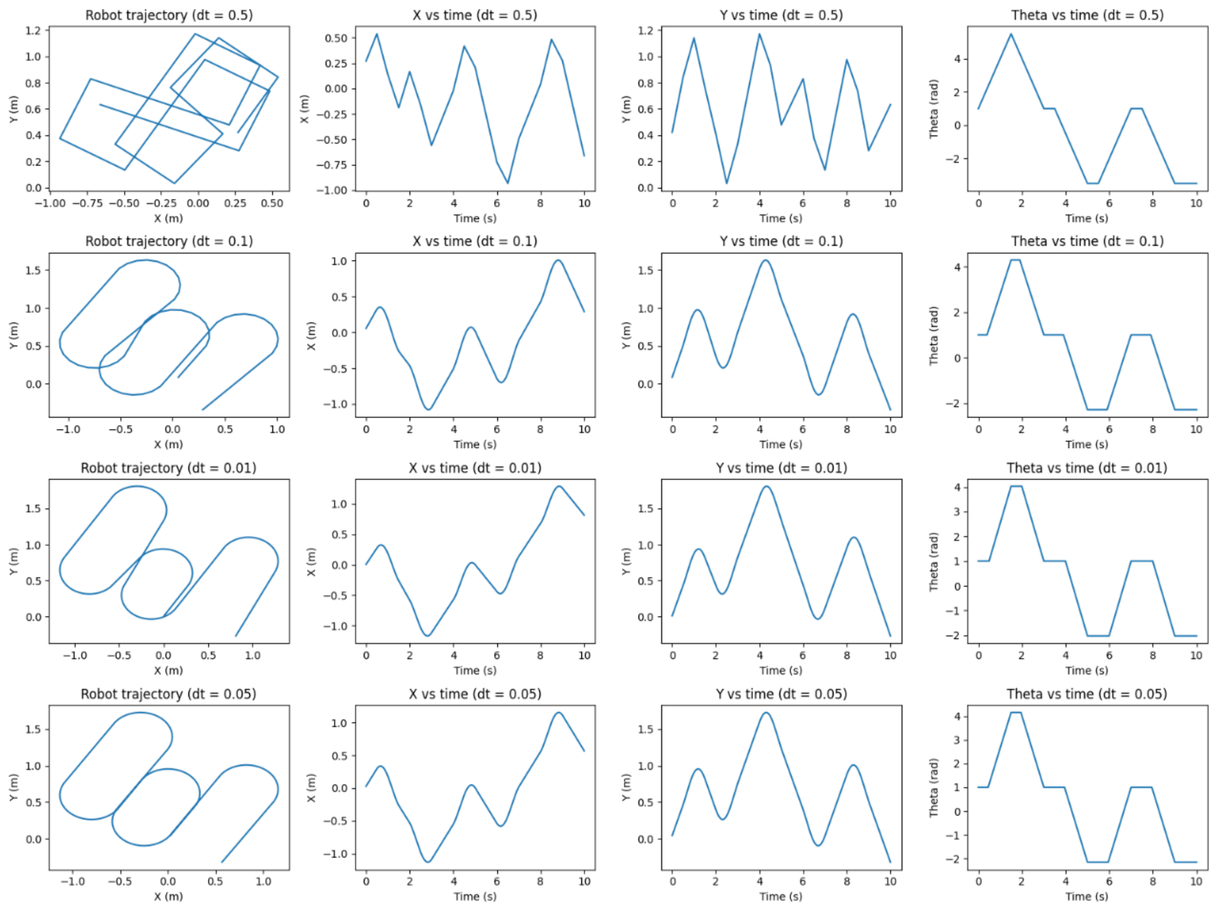
$\Delta t = 0.5, 0.1, 0.01, 0.05.$

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

```
# If time step Δt = 0.5, 0.1, 0.01, 0.05.

import numpy as np
import matplotlib.pyplot as plt

# Define control inputs
def control_input(t):
    v = np.where((t >= 0) & (t <= 10), 1, 0)
    w = np.where(((t >= 0.5) & (t <= 1.5)) | ((t >= 6) & (t <= 7)), 3, 0)
    w = np.where(((t >= 2) & (t <= 3)) | ((t >= 4) & (t <= 5)) | ((t >= 8)
& (t <= 9)), -3, w)
    return v, w

# Define differential equations
def f(x, y, theta, v, w):
    x_dot = v * np.cos(theta)
    y_dot = v * np.sin(theta)
    theta_dot = w
    return x_dot, y_dot, theta_dot

# Simulation parameters
dt_values = [0.5, 0.1, 0.01, 0.05]
T = 10
# Initial conditions
x0 = 0
y0 = 0
theta0 = 1

# Plotting setup
plt.figure(figsize=(16, 12))
plot_idx = 1

for dt in dt_values:
    # Time array
    t_data = np.arange(0, T + dt, dt)

    # Control inputs
    v, w = control_input(t_data)

    # Initialize data storage
    x_data = np.zeros(len(t_data))
    y_data = np.zeros(len(t_data))
    theta_data = np.zeros(len(t_data))

```

```python
    # Euler's method loop
    x = x0
    y = y0
    theta = theta0
    for i in range(len(t_data)):
        x_dot, y_dot, theta_dot = f(x, y, theta, v[i], w[i])
        x += x_dot * dt
        y += y_dot * dt
        theta += theta_dot * dt
        x_data[i] = x
        y_data[i] = y
        theta_data[i] = theta

    # Plotting
    plt.subplot(4, 4, plot_idx)
    plt.plot(x_data, y_data)
    plt.xlabel('X (m)')
    plt.ylabel('Y (m)')
    plt.title(f'Robot trajectory (dt = {dt})')

    plt.subplot(4, 4, plot_idx + 1)
    plt.plot(t_data, x_data)
    plt.xlabel('Time (s)')
    plt.ylabel('X (m)')
    plt.title(f'X vs time (dt = {dt})')

    plt.subplot(4, 4, plot_idx + 2)
    plt.plot(t_data, y_data)
    plt.xlabel('Time (s)')
    plt.ylabel('Y (m)')
    plt.title(f'Y vs time (dt = {dt})')

    plt.subplot(4, 4, plot_idx + 3)
    plt.plot(t_data, theta_data)
    plt.xlabel('Time (s)')
    plt.ylabel('Theta (rad)')
    plt.title(f'Theta vs time (dt = {dt})')

    plot_idx += 4

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*)

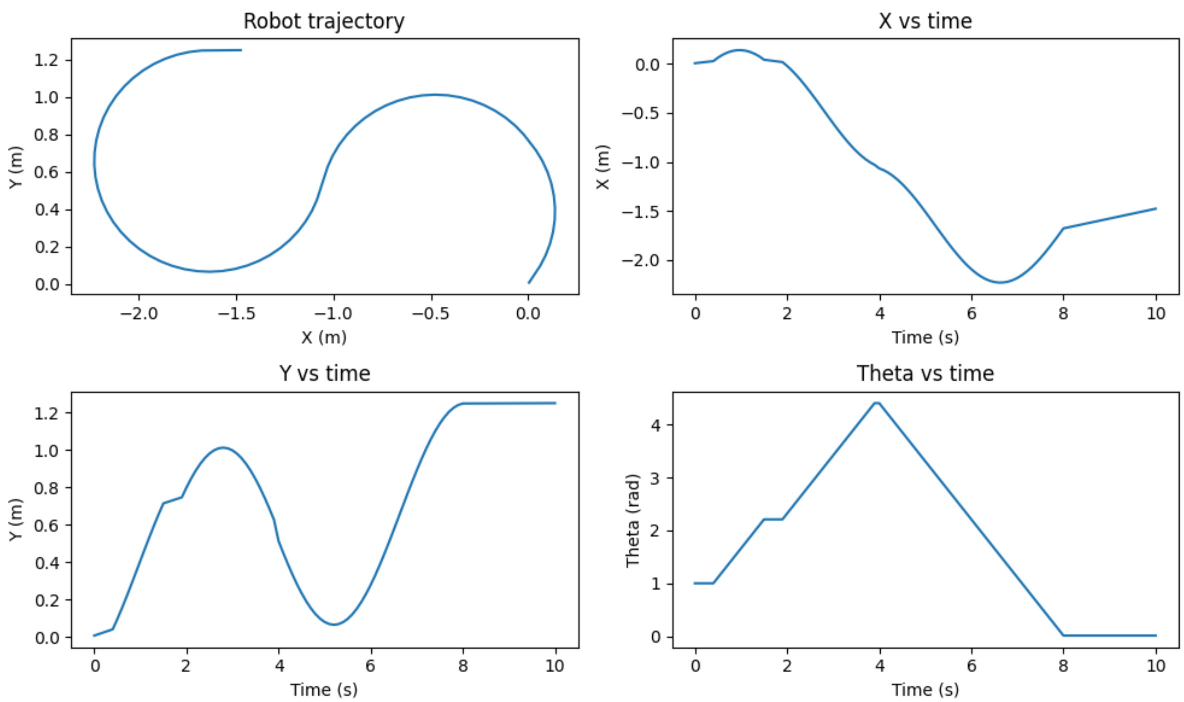
```
from google.colab import drive

# Mount Google Drive
drive.mount('/content/drive')

import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation

# Define control inputs
def control_input(t):
    wl = np.where((t >= 4) & (t <= 6), 12, np.where((t >= 6) & (t <= 8),
12, 1))
```

```
    wr = np.where(((t >= 0.5) & (t <= 1.5)) | ((t >= 2) & (t <= 4)), 12,
1)
    return wl, wr

# Define differential equations
def f(x, y, theta, wl, wr):
    r = 0.1
    L = 1

    v = (wl + wr) * r / 2
    w = (wr - wl) * r / L

    x_dot = v * np.cos(theta)
    y_dot = v * np.sin(theta)
    theta_dot = w
    return x_dot, y_dot, theta_dot

# Simulation parameters
dt = 0.1
T = 10

# Initial conditions
x = 0
y = 0
theta = 1

# Time array
t_data = np.arange(0, T + dt, dt)
# Control inputs
wl, wr = control_input(t_data)
# Initialize data storage
x_data = np.zeros(len(t_data))
y_data = np.zeros(len(t_data))
theta_data = np.zeros(len(t_data))

# Euler's method loop
for i in range(len(t_data)):
    x_dot, y_dot, theta_dot = f(x, y, theta, wl[i], wr[i])
    x += x_dot * dt
    y += y_dot * dt
    theta += theta_dot * dt
    x_data[i] = x
    y_data[i] = y
    theta_data[i] = theta

# Plot the results
plt.figure(figsize=(10, 6))

plt.subplot(2, 2, 1)
plt.plot(x_data, y_data)
plt.xlabel('X (m)')
plt.ylabel('Y (m)')
```

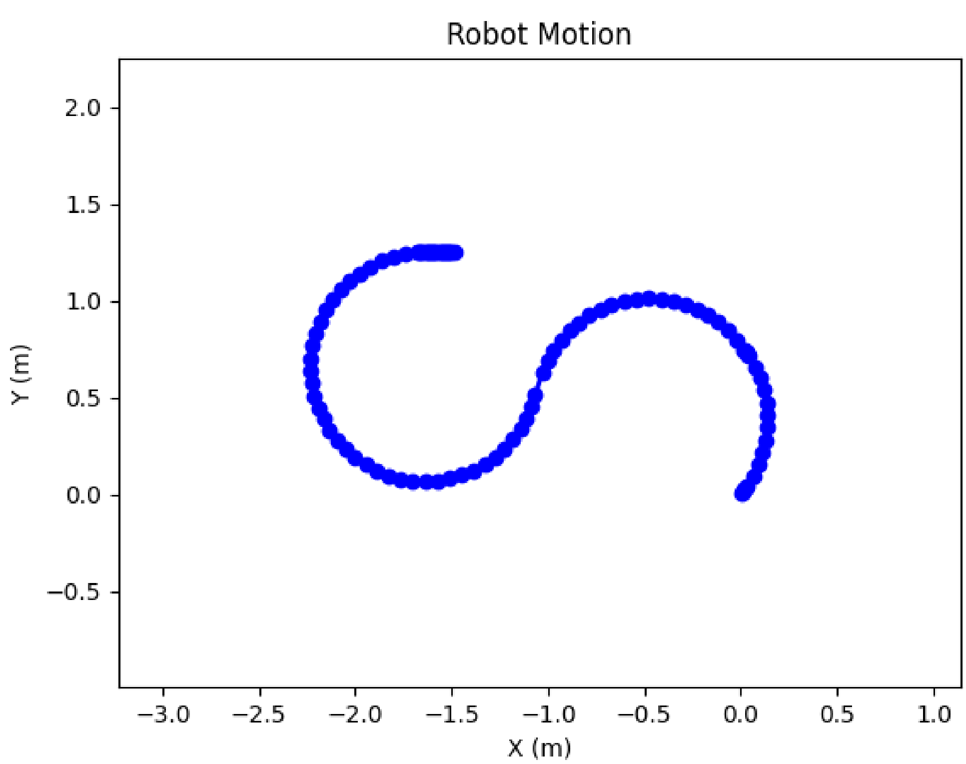
```python
plt.title('Robot trajectory')

plt.subplot(2, 2, 2)
plt.plot(t_data, x_data)
plt.xlabel('Time (s)')
plt.ylabel('X (m)')
plt.title('X vs time')

plt.subplot(2, 2, 3)
plt.plot(t_data, y_data)
plt.xlabel('Time (s)')
plt.ylabel('Y (m)')
plt.title('Y vs time')

plt.subplot(2, 2, 4)
plt.plot(t_data, theta_data)
plt.xlabel('Time (s)')
plt.ylabel('Theta (rad)')
plt.title('Theta vs time')

plt.tight_layout()
plt.show()

# Animation
fig, ax = plt.subplots()
line, = ax.plot([], [], 'bo-', lw=2)

def init():
    ax.set_xlim(min(x_data) - 1, max(x_data) + 1)
    ax.set_ylim(min(y_data) - 1, max(y_data) + 1)
    ax.set_xlabel('X (m)')
    ax.set_ylabel('Y (m)')
    ax.set_title('Robot Motion')
    return line,

def update(frame):
    line.set_data(x_data[:frame], y_data[:frame])
    return line,

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

plt.show()
```

Output:

3(a):



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



Updated version:

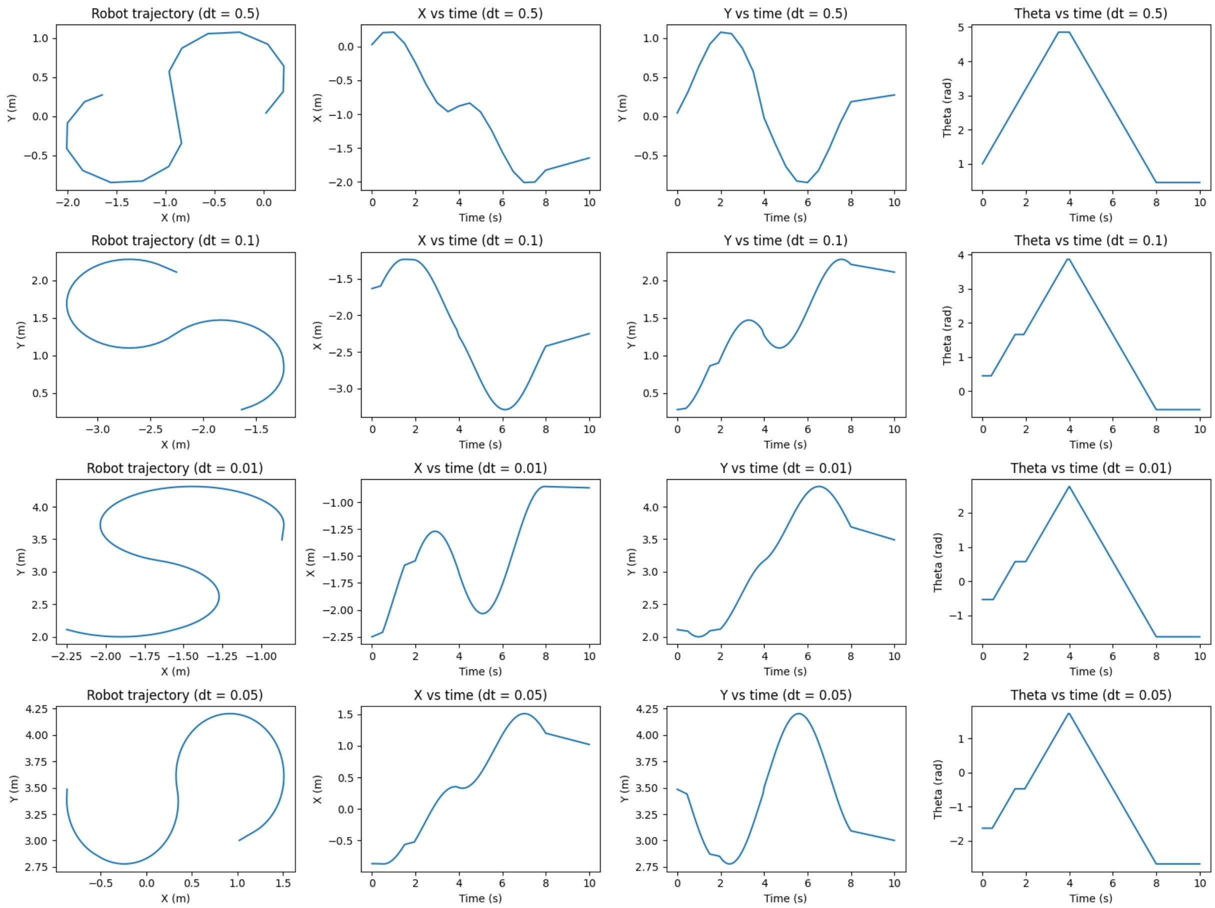
Δt = 0.5, 0.1, 0.01, 0.05.

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

```
# If time step Δt = 0.5, 0.1, 0.01, 0.05.

import numpy as np
import matplotlib.pyplot as plt

# Define control inputs
def control_input(t):
    wl = np.where((t >= 4) & (t <= 6), 12, np.where((t >= 6) & (t <= 8),
12, 1))
    wr = np.where(((t >= 0.5) & (t <= 1.5)) | ((t >= 2) & (t <= 4)), 12,
1)
    return wl, wr

# Define differential equations
def f(x, y, theta, wl, wr):
    r = 0.1
    L = 1

    v = (wl + wr) * r / 2
    w = (wr - wl) * r / L

    x_dot = v * np.cos(theta)
    y_dot = v * np.sin(theta)
    theta_dot = w
    return x_dot, y_dot, theta_dot

# Simulation parameters
dt_values = [0.5, 0.1, 0.01, 0.05]
T = 10

# Initial conditions
x = 0
y = 0
theta = 1

# Plotting setup
plt.figure(figsize=(16, 12))
plot_idx = 1

for dt in dt_values:
    # Time array
    t_data = np.arange(0, T + dt, dt)

    # Control inputs
```

```
    wl, wr = control_input(t_data)

    # Initialize data storage
    x_data = np.zeros(len(t_data))
    y_data = np.zeros(len(t_data))
    theta_data = np.zeros(len(t_data))

    # Euler's method loop
    for i in range(len(t_data)):
        x_dot, y_dot, theta_dot = f(x, y, theta, wl[i], wr[i])
        x += x_dot * dt
        y += y_dot * dt
        theta += theta_dot * dt
        x_data[i] = x
        y_data[i] = y
        theta_data[i] = theta

    # Plotting
    plt.subplot(4, 4, plot_idx)
    plt.plot(x_data, y_data)
    plt.xlabel('X (m)')
    plt.ylabel('Y (m)')
    plt.title(f'Robot trajectory (dt = {dt})')

    plt.subplot(4, 4, plot_idx + 1)
    plt.plot(t_data, x_data)
    plt.xlabel('Time (s)')
    plt.ylabel('X (m)')
    plt.title(f'X vs time (dt = {dt})')

    plt.subplot(4, 4, plot_idx + 2)
    plt.plot(t_data, y_data)
    plt.xlabel('Time (s)')
    plt.ylabel('Y (m)')
    plt.title(f'Y vs time (dt = {dt})')

    plt.subplot(4, 4, plot_idx + 3)
    plt.plot(t_data, theta_data)
    plt.xlabel('Time (s)')
    plt.ylabel('Theta (rad)')
    plt.title(f'Theta vs time (dt = {dt})')

    plot_idx += 4

plt.tight_layout()
plt.show()
```

Output:


Robot trajectory (dt = 0.5)
X vs time (dt = 0.5)
Y vs time (dt = 0.5)
Theta vs time (dt = 0.5)
Robot trajectory (dt = 0.1)
X vs time (dt = 0.1)
Y vs time (dt = 0.1)
Theta vs time (dt = 0.1)
Robot trajectory (dt = 0.01)
X vs time (dt = 0.01)
Y vs time (dt = 0.01)
Theta vs time (dt = 0.01)
Robot trajectory (dt = 0.05)
X vs time (dt = 0.05)
Y vs time (dt = 0.05)
Theta vs time (dt = 0.05)
X (m)
Y (m)
Time (s)
Theta (rad)