Assignment 4

Total points - 40 + 20 bonus.

Name:
UNCC email:
Total points: 40 (+ 20 bonus)
(Grading Rubrics are below each question)

Submission Format: Submit your solution as **a single .zip file** containing (1) a single PDF file and (2) other files for the implementation portions of the assignment. Your files should run without having to download any third-party package and should be able to reproduce all figures and results reported in your PDF file.
Ps. Please include a readme file containing the instructions to run your program.

List of Collaborators and Acknowledgements: List the names of all people you have collaborated with and for which question(s).

Questions about this assignment should be directed to TA Yash Ahire, yahire@uncc.edu

Contents:

1 . Questions ==========

1. Consider the initial value problem

$y' = x^3 * e^{-2x} - 2 * y$

$y(0) = 1$

Use the following methods with the step size h=0.1 to find approximate values of y(x) at x = 0.1 to 10 with an increment of 0.1:

- Euler integration method
- Mid-point method
- 4th-order Runge Kutta method

Write a Python program to implement the above methods and report a figure with plotted y values by Euler integration method, Mid-point method, and 4th-order Runge Kutta method, respectively, for x = 0.1 to 10.
Comment on the accuracy of the methods.

2 . You are tasked to plan a path for a car pulling one trailer using the RRT algorithm and RRT* algorithm (bonus). Each trailer is attached to the center of the rear axle of the body in front of it. Please refer to the lecture slides for the kinematics model: https://drive.google.com/file/d/1wpvMfWx8wGfRRdGNwEvgDwwfCipp6P0f/view. The environment has obstacles and the car has the following properties:

Maximum speed: 5 m/s
Maximum steering angle: 30 degrees
Length of the car: 4 meters
Length of the trailer: 8 meters

- Start point: (0, 0)
- End point: (50, 50)
- Obstacles:
    - Circle with center at (20, 20) and radius of 5
    - Rectangle with corners at (30, 10), (40, 10), (40, 20), and (30, 20)
    - Triangle with vertices at (10, 30), (15, 35), and (20, 30)

(a) Implement the RRT algorithm to generate a collision-free path for the car pulling one trailer. The path should be plotted using a suitable visualization tool, and the final path length and computation time should be reported.

Bonus -

· Implement the RRT* algorithm to generate a collision-free path for the car pulling one trailer. Compare the performance of RRT and RRT* in terms of the path length and computation time.

· Plan a path for a car pulling *three* trailers in an environment with obstacles using the RRT algorithm and RRT* algorithm. The car and trailer properties are the same as in part (a). The initial position of the car is (0,0) and the goal position is (50,50).

2. Explanations ==========

Sample code for question 2 :

1. Define the environment and obstacles
2. Define the start and end points for the car and trailer
3. Define the maximum steering angle, car and trailer lengths, and maximum speed
4. Define the RRT algorithm with the following steps:
5. Plot the path using a suitable visualization tool and report the final path length and computation time

```
import matplotlib.pyplot as plt
from matplotlib.patches import Circle, Rectangle, Polygon
import math

def plot_car_with_trailers(x, y, theta, thetas):
        fig, ax = plt.subplots()

        # plot car
        car_width = 2
        car_length = 4
        car = plt.Rectangle((x, y), car_length, car_width, angle=theta*180/math.pi, fc='blue')
        ax.add_patch(car)

        # plot trailers
        trailer_width = 2
        trailer_length = 4
        for i, trailer_theta in enumerate(thetas):
        trailer_x = x - (i+1)*trailer_length*math.cos(trailer_theta)
        trailer_y = y - (i+1)*trailer_length*math.sin(trailer_theta)
        trailer = plt.Rectangle((trailer_x, trailer_y), trailer_length, trailer_width, angle=trailer_theta*180/math.pi,
fc='green')
        ax.add_patch(trailer)

        # set plot limits
        ax.set_xlim([0, 100])
        ax.set_ylim([0, 100])

        # show plot
        plt.show()
```

```
def plot_obstacles(obstacles):
    fig, ax = plt.subplots()

    for obstacle in obstacles:
        if len(obstacle) == 1:  # obstacle is a circle
            center = obstacle[0]
            circle = Circle(center, radius=5, fc='red')
            ax.add_patch(circle)
        elif len(obstacle) == 4:  # obstacle is a rectangle
            rect = Rectangle(obstacle[0], width=obstacle[2][0]-obstacle[0][0],
height=obstacle[2][1]-obstacle[0][1], fc='red')
            ax.add_patch(rect)
        elif len(obstacle) == 3:  # obstacle is a triangle
            tri = Polygon(obstacle, closed=True, fc='red')
            ax.add_patch(tri)

    ax.set_xlim([0, 50])
    ax.set_ylim([0, 50])

    plt.show()




x = 50
y = 50
theta = 0
thetas = [0.1, 0.2, 0.3]
obstacles = [[[20, 30], [30, 30], [30, 40], [20, 40]],
         [[60, 60], [70, 60], [70, 70], [60, 70]]]

# plot car with trailers
plot_car_with_trailers(x, y, theta, thetas)

# plot obstacles
plot_obstacles(obstacles)
```

4. Deliverables ==========

1. . Python/Matlab code files for both the problems.
2. . Readme files with instructions to execute the scene/program.
3. . Write-up to include the planning results from your algorithms.

5. Rubrics ==========

Question 1 :
Executable program - 20 points
Question 2 :
Executable program - 20 points
Bonus 10 (RRT*) + 10 (RRT and RRT* for a car pulling three trailers).

6. Additional References ==========

Refer to our ppt slides as the main reference.

https://drive.google.com/file/d/1wpvMfWx8wGfRRdGNwEvgDwwfCipp6P0f/view.