```
1 import matplotlib.pyplot as plt
2 import numpy as np
5 # Plotting the output control variable (Good)
6 def output controller good(x):
      if 0 >= x >= -0.15 / 2:
          return_value = (x * 2 / 0.15) + 1
8
       elif x \le -0.15 / 2 or x \ge 0.15 / 2:
9
         return_value = 0
10
11
      else:
12
          return value = -(x * 2 / 0.15) + 1
13
      return return value
14
15
16 # Plotting the output control variable (High)
17 def output controller high(x):
18
      if x >= 0 or x < -0.15:
          return_value = 0
19
20
      elif -0.15 / 2 >= x >= -0.15:
21
         return_value = 2 * x / 0.15 + 2
22
      else:
          return_value = - (2 * x / 0.15)
23
24
      return return_value
25
26
27 # Plotting the output control variable (Low)
28 def output_controller_small(x):
29
      if x \le 0 or x > 0.15:
30
         return_value = 0
31
      elif 0.15 / 2 >= x >= 0:
32
         return_value = 2 * x / 0.15
33
      else:
34
          return value = -(2 * x / 0.15) + 2
35
      return return_value
36
37 # X-axis definition for the output partition plot
38 t = np.linspace(-0.2, 0.2, 200)
39 counter = 0
40 # Define intervals for the output control variable partition
41 good triangle = np.zeros(shape=t.shape)
42 high_triangle = np.zeros(shape=t.shape)
43 small triangle = np.zeros(shape=t.shape)
45 for i in t:
      good triangle[counter] = output_controller_good(i)
46
47
       high_triangle[counter] = output_controller_high(i)
48
       small triangle[counter] = output controller small(i)
49
      counter += 1
50
51 plt.ylabel("Membership")
52 plt.xlabel("$\delta$")
53 plt.title('Fuzzy Partition for the control variable')
54 plt.plot(t, small_triangle, label='Low Infection rate')
55 plt.plot(t, good_triangle, label='Good Infection rate')
56 plt.plot(t, high triangle, label='High Infection rate')
57 plt.legend(bbox_to_anchor=(0.98, 1), loc="upper left")
58
59 plt.show()
60
61
62 # Plotting the input partition for the rate of infected bots (High)
63 def controller_high(x):
64
     if 1 >= x >= 0.8:
65
          return value = 1
66
      elif 0.8 >= x >= 0.65:
67
         return_value = (x / 0.15) - 4.33
68
      else:
69
         return value = 0
70
      return return_value
71
73 # Plotting the input partition for the rate of infected bots (Good)
74 def controller good(x):
75
      if 0.7 >= x >= 0.6:
76
          return value = -10 * x + 7
77
      elif 0.6 >= x >= 0.5:
```

File - C:\Users\berkenutku\Desktop\496_hw3\plot_of_partioning.py

```
return value = (10 * x) - 5
 79
        else:
80
            return_value = 0
81
        return return_value
 82
83
84 \# Plotting the input partition for the rate of infected bots (Low)
85 def controller small(x):
 86 if 0.4 >= x > 0:
            return_value = 1
87
     elif 0.55 >= x >= 0.4:
88
 89
         return_value = -(x / 0.15) + 3.666
 90
       else:
 91
        return value = 0
 92
      return return_value
 93
 94 # X-axis definition for the input measurement partition
 95 k = np.linspace(-0.05, 1, 400)
96
97 \text{ counter} = 0
98 good_triangle = np.zeros(shape=k.shape)
99 high_triangle = np.zeros(shape=k.shape)
100 small_triangle = np.zeros(shape=k.shape)
101 for x in k:
      good triangle[counter] = controller_good(x)
102
103
       high_triangle[counter] = controller_high(x)
small_triangle[counter] = controller_small(x)
104
105
      counter += 1
106
107 plt.ylabel("Membership")
108 plt.xlabel("$\pi$")
109 plt.title('Fuzzy partition of the current percentage of the infected bots')
110 plt.plot(k, small_triangle, label='Low Infection rate')
111 plt.plot(k, good_triangle, label='Good Infection rate')
112 plt.plot(k, high_triangle, label='High Infection rate')
113 plt.xticks(np.arange(0, 1.1, 0.1))
114 plt.legend(bbox_to_anchor=(0.98, 1), loc="upper left")
115
116 plt.show()
117
```