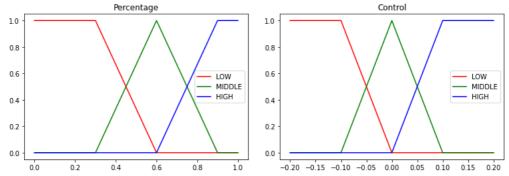
V1

1.1) I created 3 partition for percentage and control.

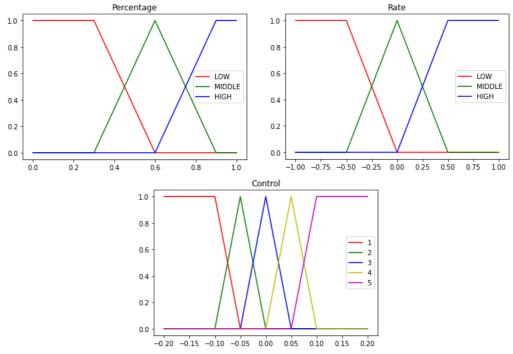


- 1.2) Control rules decided on the idea of that if you are giving one of them as high then the other will be low which means if percentage is high then control is low.
- 1.3) Fuzzification can be explained as precise data is converted into imprecise data, whereas defuzzification is the conversion of imprecise data to precise data. Defuzzification is the inverse of fuzzification.

1.4) vaccinated population percentage over days % vaccination rate (%/day) vaccinated population 11 13 19 vaccination rate over days vaccination rate control (%/day) 0 1 2 3 4 5 6 7 vaccination rate control over days 10 11 13 14 15 12 16 0.10 0.05 0.00 -0.05 Ó 10 11 12 13 14 day

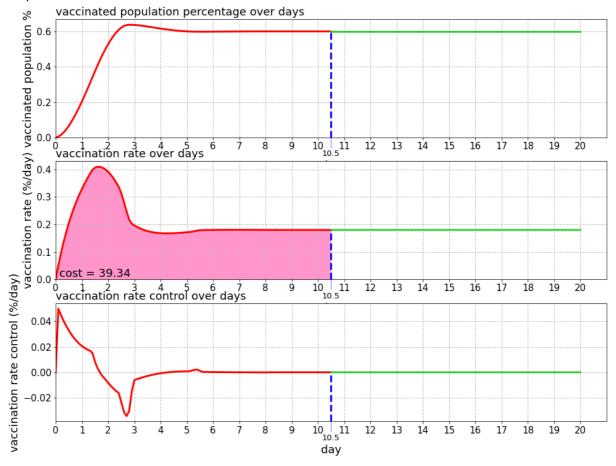
V2

2.1) The percentage and rate partitions are set as in v1. Partition of the output set into 5 fuzzy sets between [-0.2, 0.2] as in the graph.



2.2) Rules are created as to give highest control to the lowest percentage and lowest rate whereas lowest control is given to the highest percentage and highest rate.

2.3)



2.4) Comparing the v1 and v2, in v2 case of vaccination we have earlier convergence and lower cost. In that sense v2 is greater choice instead of v1. Vaccination rate and vaccination rate control are also more stable in v2.

9.06.2022 17:52 untitled1.py

```
1 #!/usr/bin/env python3
 2 \# -*- coding: utf-8 -*-
 3
 4 Created on Tue Jun 7 20:59:44 2022
 5
 6 @author: bahadir
 7 | """
8
9 #% VACCINATION V1
10 from vaccination import Vaccination
11 import os
12 import numpy as np
13 import matplotlib.pyplot as plt
14 import skfuzzy as fuzzy
15 import skfuzzy.membership as skfm
16
17 def Fuzzy_Model1(current_percent_vacc_people,plot):
18
      #set partitioning
19
      #creating percentages and control variables
20
      #giving 1.1 to see clearly in graph
21
       percent_vacc_people = np.arange( 0 , 1.1 , 0.1 )
22
       control_variable = np.arange(-0.2, 0.21, 0.05)
23
      #percent low is 1 from 0 and after 0.3-0.6 interval it becomes 0
24
       percent_vacc_low = skfm.trapmf(percent_vacc_people, [0, 0, 0.3, 0.6])
25
       percent_vacc_mid = skfm.trimf(percent_vacc_people, [0.3, 0.6, 0.9])
26
      percent_vacc_high = skfm.trapmf(percent_vacc_people, [0.6, 0.9, 1, 1])
27
       control low = skfm.trapmf(control_variable, [-0.20, -0.20, -0.1, 0])
28
29
       control_mid = skfm.trimf(control_variable, [-0.1, 0, 0.1])
30
       control_high = skfm.trapmf(control_variable, [0, 0.1, 0.25, 0.25])
31
32
       if plot == 199:
33
           #plotting
34
           plt.plot(percent_vacc_people, percent_vacc_low , 'r')
35
           plt.plot(percent_vacc_people, percent_vacc_mid ,
36
           plt.plot(percent_vacc_people, percent_vacc_high, 'b')
           plt.legend(["LOW", "MIDDLE" ,"HIGH"])
37
38
           plt.title("Percentage")
39
           plt.show()
40
           plt.savefig("percetage1.png")
41
42
           plt.plot(control_variable, control_low , 'r')
           plt.plot(control_variable, control_mid ,
43
           plt.plot(control_variable, control_high,
44
           plt.legend(["LOW", "MIDDLE", "HIGH"])
45
46
           plt.title("Control")
47
           plt.show()
           plt.savefig("control1.png")
48
49
50
      #Fuzzyfying
51
       percent_fit1 = fuzzy.interp_membership(percent_vacc_people,
   percent_vacc_low , current_percent_vacc_people)
52
       percent_fit2 = fuzzy.interp_membership(percent_vacc_people,
   percent_vacc_mid , current_percent_vacc_people)
53
       percent_fit3 = fuzzy.interp_membership(percent_vacc_people,
   percent_vacc_high, current_percent_vacc_people)
54
55
      #Fuzzy Control Rules--- with low percentage we have higher control or
   vice versa
```

localhost:4649/?mode=python 1/4

9.06.2022 17:52 untitled1.py

```
56
       control_rule1 = np.fmin(percent_fit1 , control_high)
        control_rule2 = np.fmin(percent_fit2 , control_mid )
 57
 58
        control_rule3 = np.fmin(percent_fit3, control_low )
 59
 60
        final control = np.fmax(control rule1,control rule2)
 61
        final_control = np.fmax(final_control,control_rule3)
 62
63
       #Deffuzify the control variable and final variable
64
        defuzzed = fuzzy.defuzz(control variable, final control, 'centroid')
65
        return defuzzed
66
67 #start vaccination
68 vaccination = Vaccination()
69 equilibrium day = 0
70 \cos t = 0
 71 final cost = 0
72 percent_temp_previous = -5
 73 percent_temp_current = 0
74 s=0
75 for i in range (200):
       percentage , rate = vaccination.checkVaccinationStatus()
76
 77
       #Vacinate people according to the control variable comes from the Fuzzy
 78
       vaccination.vaccinatePeople(Fuzzy_Model1(percentage,i))
 79
        cost = cost + vaccination.vaccination_rate_curve_[-1]
 80
81
        if abs(percent_temp_previous -
   vaccination.vaccinated_percentage_curve_[-1]) < 0.000001 and s==0 :
82
            equilibrium day = i
83
            final cost = cost
 84
            s=1
 85
 86
        percent_temp_previous = percent_temp_current
87
        percent_temp_current = vaccination.vaccinated_percentage_curve_[-1]
88
89 vaccination.viewVaccination(equilibrium_day,cost)
90
91 #% VACCINATION V2
92 from vaccination import Vaccination
93 import os
94 import numpy as np
95 import matplotlib.pyplot as plt
96 import skfuzzy as fuzzy
97 import skfuzzy.membership as skfm
98
99 def Fuzzy_Model2(current_percent_vacc_people,current_rate, plot):
       #set partitioning
100
101
       #creating percentages and control variables
102
       #giving 1.1 to see clearly in graph
103
       percent_vacc_people = np.arange( 0 , 1.1 , 0.1 )
        rate = np.arange(-1, 1.1, 0.1)
104
105
        control variable = np.arange(-0.2, 0.21, 0.05)
       #percent low is 1 from 0 and after 0.3-0.6 interval it becomes 0
106
       percent vacc_low = skfm.trapmf(percent_vacc_people, [0, 0, 0.3, 0.6])
107
108
       percent_vacc_mid = skfm.trimf(percent_vacc_people, [0.3, 0.6, 0.9])
109
       percent_vacc_high = skfm.trapmf(percent_vacc_people, [0.6, 0.9, 1, 1])
110
111
        rate_low = skfm.trapmf(rate, [-1, -1, -0.5, 0])
        rate_mid = skfm.trimf(rate, [-0.5, 0, 0.5])
112
        rate_high = skfm.trapmf(rate, [0, 0.5, 1, 1])
113
114
```

localhost:4649/?mode=python 2/4

9.06.2022 17:52 untitled1.py

```
control1 = skfm.trapmf(control_variable, [-0.20, -0.20, -0.1, -0.05])
115
        control2 = skfm.trimf(control_variable, [-0.1, -0.05 , 0])
116
        control3 = skfm.trimf(control_variable, [-0.05, 0 , 0.05])
117
118
        control4 = skfm.trimf(control_variable, [0, 0.05, 0.1])
119
        control5 = skfm.trapmf(control variable, [0.05, 0.1, 0.25, 0.25])
120
121
        if plot == 199:
122
123
            #plotting
124
            plt.plot(percent_vacc_people, percent_vacc_low ,
125
            plt.plot(percent_vacc_people, percent_vacc_mid , 'g')
126
            plt.plot(percent_vacc_people, percent_vacc_high, 'b')
            plt.legend(["LOW", "MIDDLE", "HIGH"])
127
128
            plt.title("Percentage")
129
            plt.show()
            plt.savefig("percetage2.png")
130
131
132
            plt.plot(rate, rate_low , 'r')
            plt.plot(rate, rate_mid , 'g')
133
134
            plt.plot(rate, rate_high, 'b')
            plt.legend(["LOW", "MIDDLE" ,"HIGH"])
135
136
            plt.title("Rate")
137
            plt.show()
138
            plt.savefig("rate2.png")
139
            plt.plot(control_variable, control1 , 'r')
140
            plt.plot(control_variable, control2 ,
141
            plt.plot(control_variable, control3,
142
143
            plt.plot(control variable, control4, 'y')
144
            plt.plot(control_variable, control5, 'm')
            plt.legend(["1", "2", "3", "4", "5"])
145
146
            plt.title("Control")
147
            plt.show()
            plt.savefig("control2.png")
148
149
150
        #Fuzzyfying
        percent fit1 = fuzzy.interp membership(percent vacc people,
151
    percent_vacc_low , current_percent_vacc_people)
        percent_fit2 = fuzzy.interp_membership(percent_vacc_people,
152
    percent_vacc_mid , current_percent_vacc_people)
153
        percent_fit3 = fuzzy.interp_membership(percent_vacc_people,
    percent_vacc_high, current_percent_vacc_people)
154
155
        rate_fit1 = fuzzy.interp_membership(rate, rate_low , current_rate)
156
        rate fit2 = fuzzy.interp membership(rate, rate mid , current rate)
157
        rate fit3 = fuzzy.interp membership(rate, rate high, current rate)
158
159
       #Fuzzy Control Rules--- with low percentage we have higher control or
    vice versa
160
        control_rule1 = np.fmin(np.fmin(percent_fit1,rate_fit1),control5)
161
        control rule2 =
    np.fmin(np.fmax(np.fmin(percent_fit2, rate_fit1), np.fmin(percent_fit1, rate_fit
    2)),control4)
162
        control_rule3 = np.fmin(np.fmax(np.fmax(np.fmin(percent_fit1, rate_fit3),
    np.fmin(percent fit3, rate fit1)), np.fmin(percent fit2,
    rate_fit2)),control3)
163
        control rule4 =
    np.fmin(np.fmax(np.fmin(percent_fit3,rate_fit2),np.fmin(percent_fit2,rate_fit
    3)),control2)
        control rule5 = np.fmin(np.fmin(percent fit3, rate fit3), control1)
164
```

localhost:4649/?mode=python 3/4

9.06.2022 17:52 untitled1.py

```
165
166
        final control =
    np.fmax(np.fmax(control_rule1,control_rule2),np.fmax(control_rule3,co
    ntrol rule4)),control rule5)
167
        #Deffuzify the control variable and final variable
168
169
        defuzzed = fuzzy.defuzz(control_variable, final_control, 'centroid')
        return defuzzed
170
171
172 #start vaccination
173 vaccination = Vaccination()
174 \text{ equilibrium\_day} = -1
175 | cost = 0
176 | final cost = 0
177 percent_temp_previous = -5
178 percent_temp_current = 0
179 s=0
180 for i in range (200):
181
        percentage , rate = vaccination.checkVaccinationStatus()
182
        #Vacinate people according to the control variable comes from the Fuzzy
        vaccination.vaccinatePeople(Fuzzy_Model2(percentage, rate,i))
183
        cost = cost + vaccination.vaccination_rate_curve_[-1]
184
185
186
        if abs(percent_temp_previous -
    vaccination.vaccinated_percentage_curve_[-1]) < 0.000001 and s==0:
187
            equilibrium_day = i
188
            final_cost = cost
189
            s=1
190
191
        percent_temp_previous = percent_temp_current
192
        percent_temp_current = vaccination.vaccinated_percentage_curve_[-1]
193
194 vaccination.viewVaccination(equilibrium day,cost)
195
196
```

4/4 localhost:4649/?mode=python