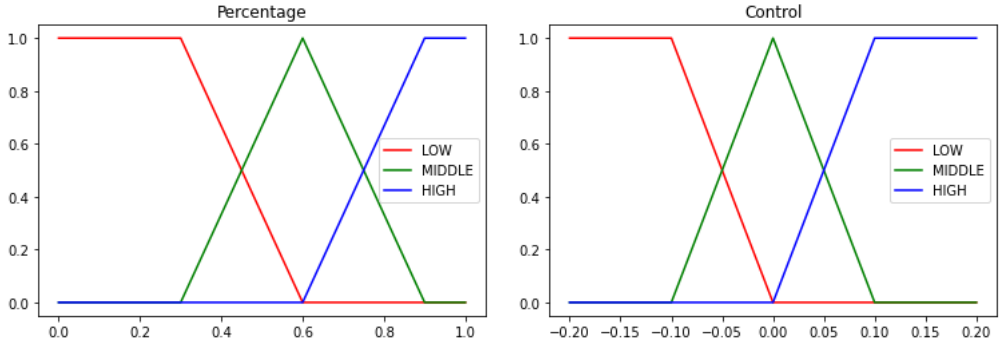
EE449 HW3

Muhammed Bahadır Demiryakan
2304426

V1

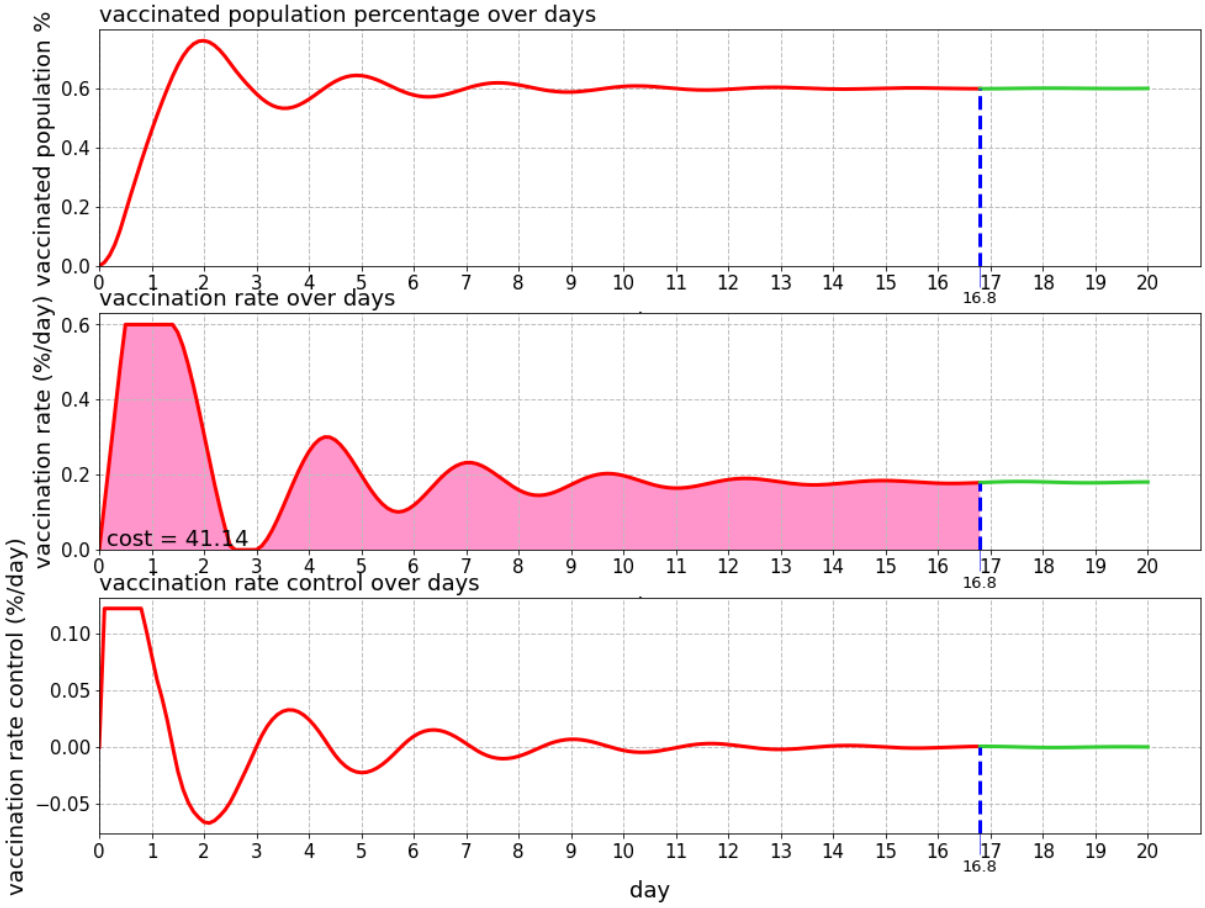
1.1) I created 3 partition for percentage and control.



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.

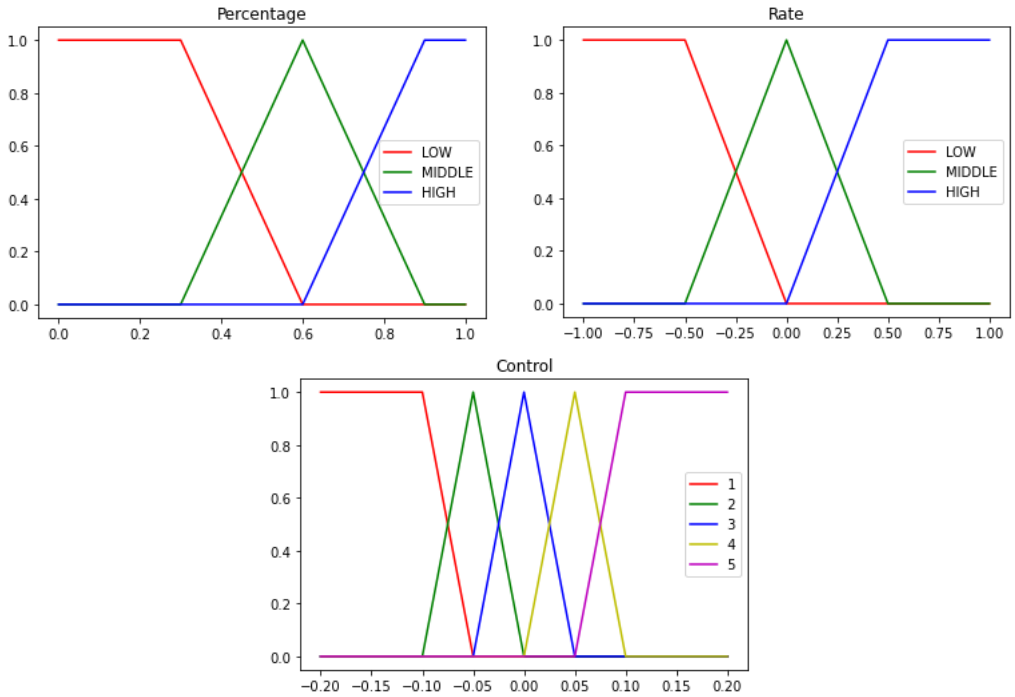
1.3) Fuzzification can be explained as precise data is converted into imprecise data, whereas defuzzification is the conversion of imprecise data to precise data. Defuzzification is the inverse of fuzzification.

1.4)



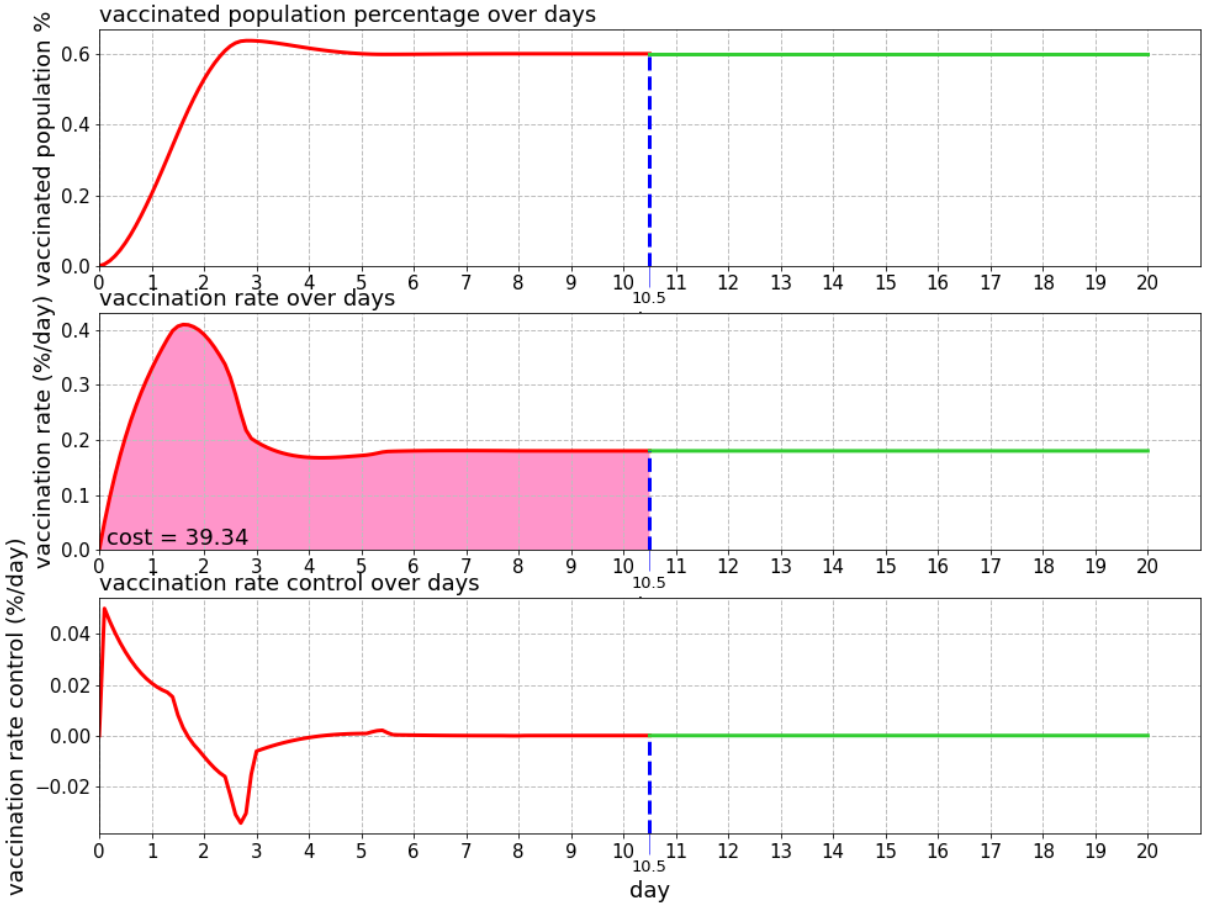
V2

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.

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue Jun  7 20:59:44 2022

@author: bahadir
"""

#%% VACCINATION V1
from vaccination import Vaccination
import os
import numpy as np
import matplotlib.pyplot as plt
import skfuzzy as fuzzy
import skfuzzy.membership as skfm

def Fuzzy_Model1(current_percent_vacc_people,plot):
    #set partitioning
    #creating percentages and control variables
    #giving 1.1 to see clearly in graph
    percent_vacc_people = np.arange( 0 , 1.1 , 0.1 )
    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
    percent_vacc_low = skfm.trapmf(percent_vacc_people, [0, 0, 0.3, 0.6])
    percent_vacc_mid = skfm.trimf(percent_vacc_people, [0.3, 0.6, 0.9])
    percent_vacc_high = skfm.trapmf(percent_vacc_people, [0.6, 0.9, 1, 1])

    control_low = skfm.trapmf(control_variable, [-0.20, -0.20, -0.1, 0])
    control_mid = skfm.trimf(control_variable, [-0.1, 0, 0.1])
    control_high = skfm.trapmf(control_variable, [0, 0.1, 0.25, 0.25])

    if plot == 199:
        #plotting
        plt.plot(percent_vacc_people, percent_vacc_low , 'r')
        plt.plot(percent_vacc_people, percent_vacc_mid , 'g')
        plt.plot(percent_vacc_people, percent_vacc_high, 'b')
        plt.legend(["LOW", "MIDDLE" ,"HIGH"])
        plt.title("Percentage")
        plt.show()
        plt.savefig("percetage1.png")

        plt.plot(control_variable, control_low , 'r')
        plt.plot(control_variable, control_mid , 'g')
        plt.plot(control_variable, control_high, 'b')
        plt.legend(["LOW", "MIDDLE" ,"HIGH"])
        plt.title("Control")
        plt.show()
        plt.savefig("control1.png")

    #Fuzzyfying
    percent_fit1 = fuzzy.interp_membership(percent_vacc_people, percent_vacc_low , current_percent_vacc_people)
    percent_fit2 = fuzzy.interp_membership(percent_vacc_people, percent_vacc_mid , current_percent_vacc_people)
    percent_fit3 = fuzzy.interp_membership(percent_vacc_people, percent_vacc_high, current_percent_vacc_people)

    #Fuzzy Control Rules--- with low percentage we have higher control or vice versa
```

```
    control_rule1 = np.fmin(percent_fit1 , control_high)
    control_rule2 = np.fmin(percent_fit2 , control_mid )
    control_rule3 = np.fmin(percent_fit3, control_low )

    final_control = np.fmax(control_rule1,control_rule2)
    final_control = np.fmax(final_control,control_rule3)

    #Deffuzify the control variable and final variable
    defuzzed = fuzzy.defuzz(control_variable, final_control, 'centroid')
    return defuzzed

 #start vaccination
vaccination = Vaccination()
equilibrium_day = 0
cost = 0
final_cost = 0
percent_temp_previous = -5
percent_temp_current = 0
s=0
for i in range (200):
    percentage , rate = vaccination.checkVaccinationStatus()
    #Vacinate people according to the control variable comes from the Fuzzy
    vaccination.vaccinatePeople(Fuzzy_Model1(percentage,i))
    cost = cost + vaccination.vaccination_rate_curve_[-1]

    if abs(percent_temp_previous -
vaccination.vaccinated_percentage_curve_[-1]) < 0.000001 and s==0 :
        equilibrium_day = i
        final_cost = cost
        s=1

    percent_temp_previous = percent_temp_current
    percent_temp_current = vaccination.vaccinated_percentage_curve_[-1]

vaccination.viewVaccination(equilibrium_day,cost)

#%% VACCINATION V2
from vaccination import Vaccination
import os
import numpy as np
import matplotlib.pyplot as plt
import skfuzzy as fuzzy
import skfuzzy.membership as skfm

def Fuzzy_Model2(current_percent_vacc_people,current_rate, plot):
    #set partitioning
    #creating percentages and control variables
    #giving 1.1 to see clearly in graph
    percent_vacc_people = np.arange( 0 , 1.1 , 0.1 )
    rate = np.arange(-1, 1.1, 0.1)
    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
    percent_vacc_low = skfm.trapmf(percent_vacc_people, [0, 0, 0.3, 0.6])
    percent_vacc_mid = skfm.trimf(percent_vacc_people, [0.3, 0.6, 0.9])
    percent_vacc_high = skfm.trapmf(percent_vacc_people, [0.6, 0.9, 1, 1])

    rate_low  = skfm.trapmf(rate, [-1, -1, -0.5, 0])
    rate_mid  = skfm.trimf(rate, [-0.5, 0, 0.5])
    rate_high = skfm.trapmf(rate, [0, 0.5, 1, 1])

```

```
    control1 = skfm.trapmf(control_variable, [-0.20, -0.20, -0.1, -0.05])
    control2 = skfm.trimf(control_variable, [-0.1, -0.05 , 0])
    control3 = skfm.trimf(control_variable, [-0.05, 0 , 0.05])
    control4 = skfm.trimf(control_variable, [0, 0.05, 0.1])
    control5 = skfm.trapmf(control_variable, [0.05, 0.1, 0.25, 0.25])


    if plot == 199:
        #plotting
        plt.plot(percent_vacc_people, percent_vacc_low , 'r')
        plt.plot(percent_vacc_people, percent_vacc_mid , 'g')
        plt.plot(percent_vacc_people, percent_vacc_high, 'b')
        plt.legend(["LOW", "MIDDLE" ,"HIGH"])
        plt.title("Percentage")
        plt.show()
        plt.savefig("percetage2.png")

        plt.plot(rate, rate_low , 'r')
        plt.plot(rate, rate_mid , 'g')
        plt.plot(rate, rate_high, 'b')
        plt.legend(["LOW", "MIDDLE" ,"HIGH"])
        plt.title("Rate")
        plt.show()
        plt.savefig("rate2.png")

        plt.plot(control_variable, control1 , 'r')
        plt.plot(control_variable, control2 , 'g')
        plt.plot(control_variable, control3, 'b')
        plt.plot(control_variable, control4, 'y')
        plt.plot(control_variable, control5, 'm')
        plt.legend(["1", "2" ,"3", "4", "5"])
        plt.title("Control")
        plt.show()
        plt.savefig("control2.png")

    #Fuzzyfying
    percent_fit1 = fuzzy.interp_membership(percent_vacc_people,
percent_vacc_low , current_percent_vacc_people)
    percent_fit2 = fuzzy.interp_membership(percent_vacc_people,
percent_vacc_mid , current_percent_vacc_people)
    percent_fit3 = fuzzy.interp_membership(percent_vacc_people,
percent_vacc_high, current_percent_vacc_people)

    rate_fit1 = fuzzy.interp_membership(rate, rate_low , current_rate)
    rate_fit2 = fuzzy.interp_membership(rate, rate_mid , current_rate)
    rate_fit3 = fuzzy.interp_membership(rate, rate_high, current_rate)

    #Fuzzy Control Rules--- with low percentage we have higher control or
vice versa
    control_rule1 = np.fmin(np.fmin(percent_fit1,rate_fit1),control5)
    control_rule2 =
np.fmin(np.fmax(np.fmin(percent_fit2,rate_fit1),np.fmin(percent_fit1,rate_fit
2)),control4)
    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)
    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)
```

```
165
166     final_control =
np.fmax(np.fmax(np.fmax(control_rule1,control_rule2),np.fmax(control_rule3,co
ntrol_rule4)),control_rule5)
167
168     #Deffuzify the control variable and final variable
169     defuzzed = fuzzy.defuzz(control_variable, final_control, 'centroid')
170     return defuzzed
171
172  #start vaccination
173 vaccination = Vaccination()
174 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
183     vaccination.vaccinatePeople(Fuzzy_Model2(percentage,rate,i))
184     cost = cost + vaccination.vaccination_rate_curve_[-1]
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
```