1. **Vaccination v1**
   1. **Set Partitioning**

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

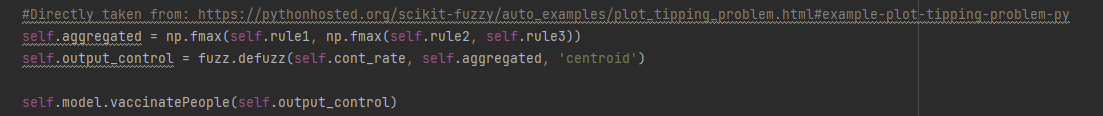
As the desired output is 0.6, average is adjusted with respect to that. In addition, as the range of control rate increases, we obtain more precise result in a higher convergence speed. Hence, in order to obtain optimum result, average control rate is chosen as the 50% of the given range.

* 1. **Fuzzy Control Rules**

metin içeren bir resim

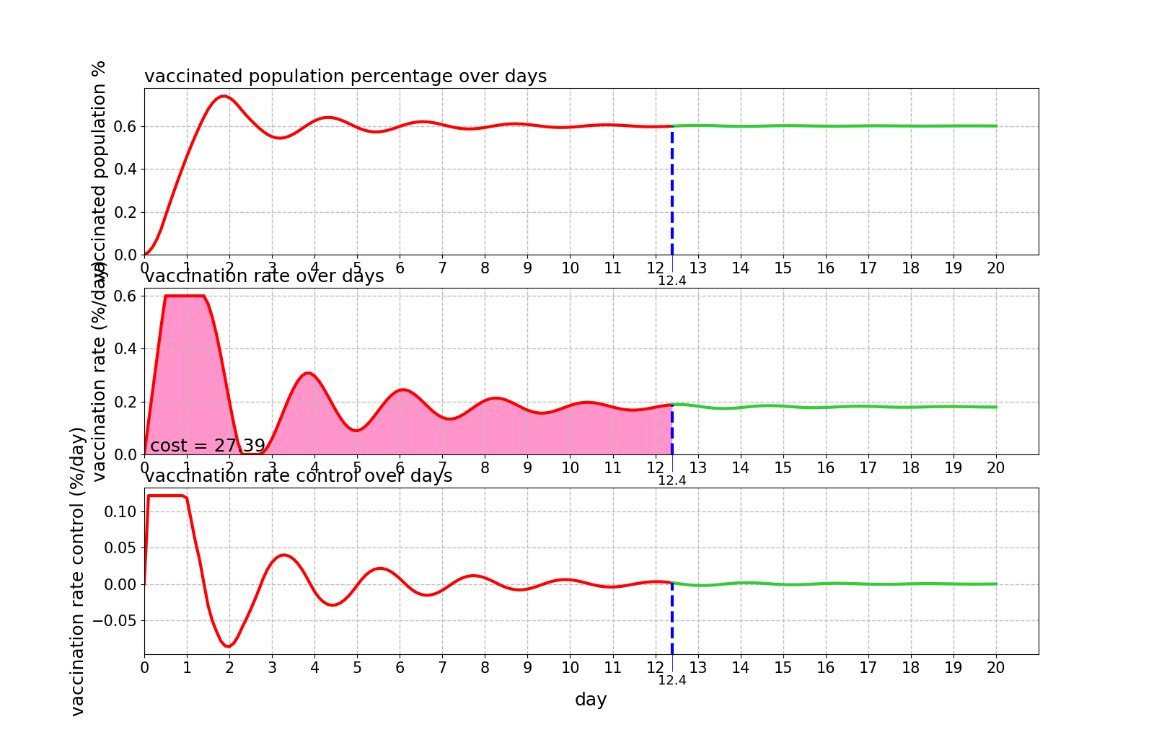
Açıklama otomatik olarak oluşturuldu

* 1. **Fuzzification and Defuzzification Interface**

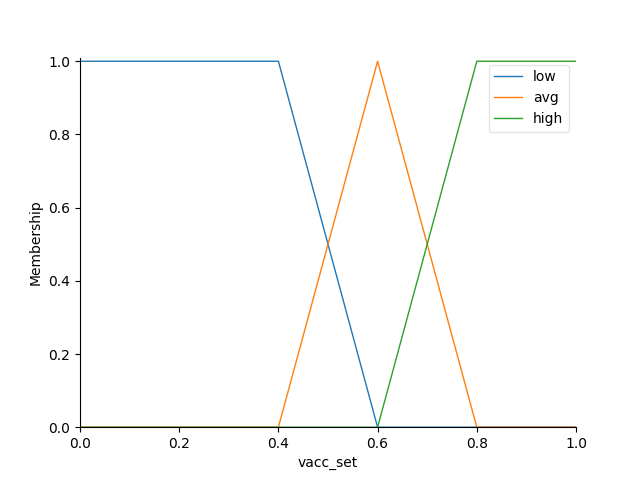


Aggregate all three output membership functions together in the first line. Then, calculate the defuzzified result with centroid method. Then, take necessary actions.

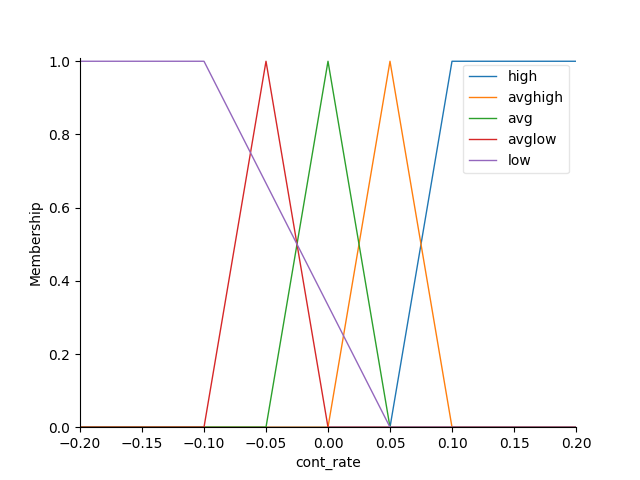
* 1. **Simulation**



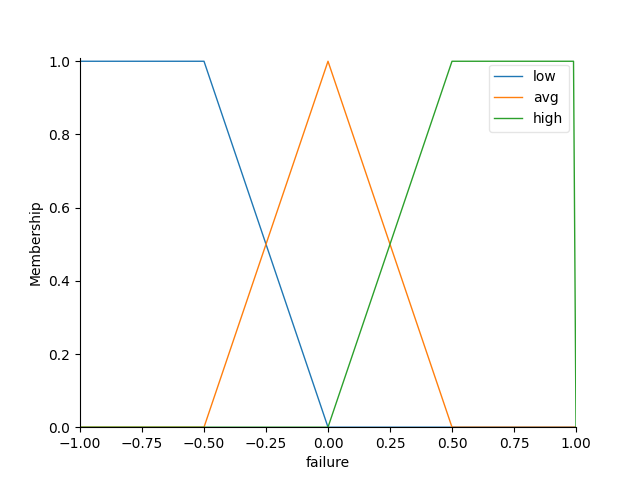
1. **Vaccination v2**
   1. **Set Partitioning**



As the desired output is 0.6, average is adjusted with respect to that.

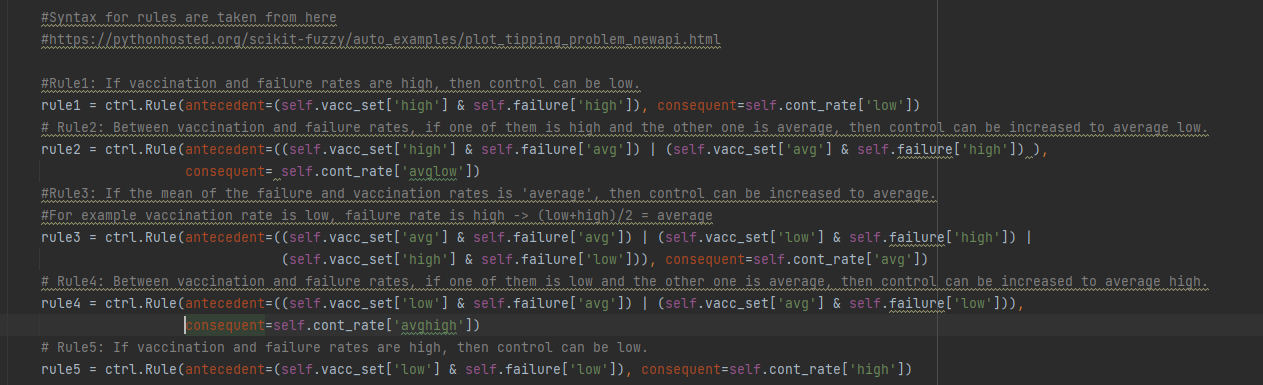


Medium is adjusted to the mean value of the given range. Then, midium values of the range [-0.2,0.2] is divided into equal parts and control rate is increased with respect to that. As in the previous part 50% of the given range is chosen, I divided between [-0.1,0.1].



As the range of control rate increases, we obtain more precise result in a higher convergence speed. Hence, in order to obtain optimum result, average control rate is chosen as the 50% of the given range.

* 1. **Fuzzy Control Rules**



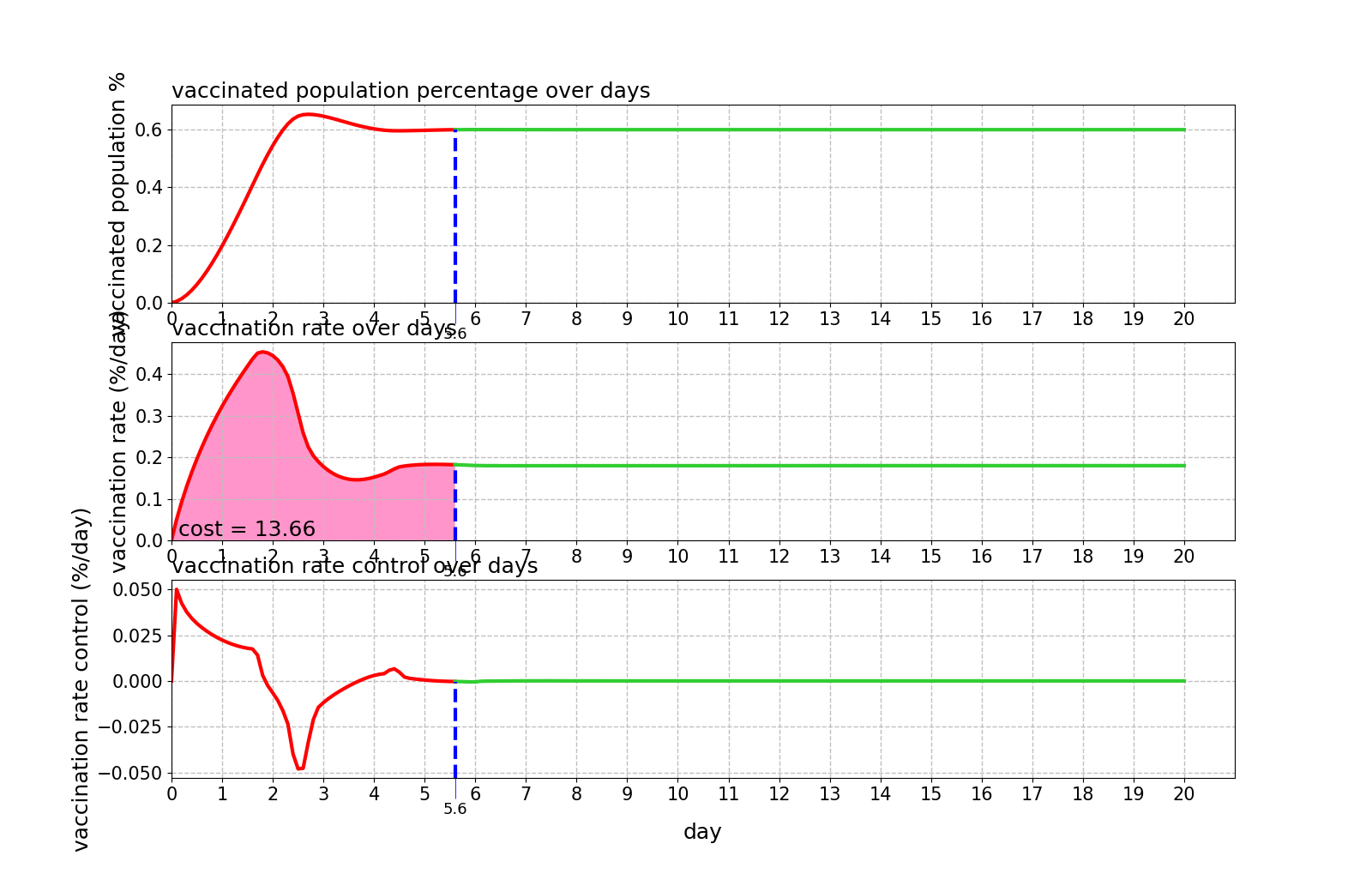
* 1. **Fuzzification and Defuzzification Interface**

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Pass the rule set to the ControlSystemSimulation and update the variables. Then, give input with respect to their labels. After compute(), take the output and take necessary actions given in vaccinatePeople.

* 1. **Simulation**



Cost is decreased and convergence is increased as we observe the simulation results. Division of the control rate and increase in the inputs have improved the system and we observed better results in vaccination v2.

**Vaccination v1 Codes**

import numpy as np  
import skfuzzy as fuzz  
from vaccination import Vaccination  
# Create universal variables to use in Antecedent and Consequent functions  
# vacc\_per = np.linspace(0, 1.0, 0.05)  
# sigma = np.linspace(-0.2, 0.2, 0.05)  
# linspace caused the following error.  
# TypeError: 'float' object cannot be interpreted as an integer  
# Hence, it is replaced with np.arange which is taken from:  
# https://pythonhosted.org/scikit-fuzzy/auto\_examples/plot\_tipping\_problem\_newapi.html  
  
#References: https://pythonhosted.org/scikit-fuzzy/auto\_examples/plot\_tipping\_problem.html#example-plot-tipping-problem-py  
#https://pythonhosted.org/scikit-fuzzy/auto\_examples/plot\_tipping\_problem\_newapi.html  
#https://pythonhosted.org/scikit-fuzzy/auto\_examples/plot\_control\_system\_advanced.html  
#https://stackoverflow.com/questions/65596610/fuzzy-system-valueerror  
#Some parts are taken directly from the above websites. Just changed the names.  
  
#Class declaration  
class myFuzzy():  
 def \_\_init\_\_(self):  
 #Call vaccination.py  
 self.model = Vaccination()  
 #Update variable  
 self.UpdatePercent()  
 self.UpdateCost()  
 # Update variable  
  
 self.vacc\_set = np.arange(0, 1.01, 0.01)  
 self.cont\_rate = np.arange(-0.2, 0.21, 0.05)  
 #https://pythonhosted.org/scikit-fuzzy/auto\_examples/plot\_tipping\_problem.html#example-plot-tipping-problem-py  
 #above website is used and some parts are directly taken  
 self.low\_vacc = fuzz.trapmf(self.vacc\_set, [0, 0, 0.4, 0.6])  
 self.avg\_vacc = fuzz.trimf(self.vacc\_set, [0.4, 0.6, 0.8])  
 self.high\_vacc = fuzz.trapmf(self.vacc\_set, [0.6, 0.8, 1, 1])  
 #https://pythonhosted.org/scikit-fuzzy/auto\_examples/plot\_tipping\_problem.html#example-plot-tipping-problem-py  
 #above website is used and some parts are directly taken  
 self.high\_cont = fuzz.trapmf(self.cont\_rate, [0, 0.1, 0.2, 0.2])  
 self.avg\_cont = fuzz.trimf(self.cont\_rate, [-0.1, 0, 0.1])  
 self.low\_cont = fuzz.trapmf(self.cont\_rate, [-0.2, -0.2, -0.1, 0])  
  
 def FuzzyLogic(self):  
 #https://pythonhosted.org/scikit-fuzzy/auto\_examples/plot\_tipping\_problem.html#example-plot-tipping-problem-py  
 #Directly taken from here. I just adjusted the variable names.  
 self.low\_level = fuzz.interp\_membership(self.vacc\_set, self.low\_vacc, self.vacc\_perc)  
 self.avg\_level = fuzz.interp\_membership(self.vacc\_set, self.avg\_vacc, self.vacc\_perc)  
 self.high\_level = fuzz.interp\_membership(self.vacc\_set, self.high\_vacc, self.vacc\_perc)  
  
  
 #Overall rule is that the vaccination rate and control rate is inversely proportional.  
 #As one of them increases, decrease the other one.  
  
 #Rule1: If vaccination is low, then we need the have high control.  
 self.rule1 = np.fmin(self.low\_level, self.high\_cont)  
 #Rule2: If vaccination is average, then we can decrease the control to average.  
 self.rule2 = np.fmin(self.avg\_level, self.avg\_cont)  
 #Rule3: If vaccination is high, then we can decrease the control to low.  
 self.rule3 = np.fmin(self.high\_level, self.low\_cont)  
 # https://pythonhosted.org/scikit-fuzzy/auto\_examples/plot\_tipping\_problem.html#example-plot-tipping-problem-py  
 # Directly taken from here. I just adjusted the variable names.  
  
  
 #Directly taken from: https://pythonhosted.org/scikit-fuzzy/auto\_examples/plot\_tipping\_problem.html#example-plot-tipping-problem-py  
 self.aggregated = np.fmax(self.rule1, np.fmax(self.rule2, self.rule3))  
 self.output\_control = fuzz.defuzz(self.cont\_rate, self.aggregated, 'centroid')  
  
 self.model.vaccinatePeople(self.output\_control)  
 def UpdatePercent(self):  
 #Update variable  
 self.vacc\_perc = self.model.checkVaccinationStatus()[0]  
 def UpdateCost(self):  
 # Update variable  
 self.cost = self.model.vaccination\_rate\_curve\_[-1]  
  
#Create a variable for check to eq. point.  
checker = 1  
#Initialize (declare) cost  
cost = 0  
#Define error value  
error = 0.0001  
#Call the fuzzy system  
Pop = myFuzzy()  
for i in range(200):  
 # Initialize logic  
 Pop.FuzzyLogic()  
 #Update variable  
 Pop.UpdatePercent()  
 Pop.UpdateCost()  
 # Update variable  
 if checker == 1:  
 cost += Pop.cost #Update the cost until eq. point  
 diff = abs(Pop.vacc\_perc - 0.6) #Check the difference  
 if (diff < error) and (checker == 1):#If eq. conditions are satisfied  
 checker = 0  
 checker = 0  
 point\_ss = i #Get the step that reaches eq. point  
  
Pop.model.viewVaccination(point\_ss = point\_ss, vaccination\_cost=cost, filename='vaccination1')

**Vaccination v2 Codes**

import numpy as np  
import skfuzzy as fuzz  
from vaccination import Vaccination  
from skfuzzy import control as ctrl  
  
# Create universal variables to use in Antecedent and Consequent functions  
# vacc\_per = np.linspace(0, 1.0, 0.05)  
# sigma = np.linspace(-0.2, 0.2, 0.05)  
# linspace caused the following error.  
# TypeError: 'float' object cannot be interpreted as an integer  
# Hence, it is replaced with np.arange which is taken from:  
# https://pythonhosted.org/scikit-fuzzy/auto\_examples/plot\_tipping\_problem\_newapi.html  
  
  
#References: https://pythonhosted.org/scikit-fuzzy/auto\_examples/plot\_tipping\_problem.html#example-plot-tipping-problem-py  
#https://pythonhosted.org/scikit-fuzzy/auto\_examples/plot\_tipping\_problem\_newapi.html  
#https://pythonhosted.org/scikit-fuzzy/auto\_examples/plot\_control\_system\_advanced.html  
#https://stackoverflow.com/questions/65596610/fuzzy-system-valueerror  
#Some parts are taken directly from the above websites. Just changed the names.  
  
#Class definition  
class myFuzzy():  
 def \_\_init\_\_(self):  
 #Partition definitions are take directly from here  
 #https://stackoverflow.com/questions/65596610/fuzzy-system-valueerror  
 #https://pythonhosted.org/scikit-fuzzy/auto\_examples/plot\_tipping\_problem\_newapi.html  
 #Just adjusted the variable names and values  
 self.vacc\_set = ctrl.Antecedent(np.arange(0, 1.01, 0.01), 'vacc\_set')  
 self.failure = ctrl.Antecedent(np.arange(-1, 1.01, 0.01), 'failure')  
 self.cont\_rate = ctrl.Consequent(np.arange(-0.2, 0.21, 0.05), 'cont\_rate')  
  
 # Define the partitions for vaccination  
 self.vacc\_set['low'] = fuzz.trapmf(self.vacc\_set.universe, [0, 0, 0.4, 0.6])  
 self.vacc\_set['avg'] = fuzz.trimf(self.vacc\_set.universe, [0.4, 0.6, 0.8])  
 self.vacc\_set['high'] = fuzz.trapmf(self.vacc\_set.universe, [0.6, 0.8, 1, 1])  
 self.vacc\_set.view()  
  
 # Define the partitions for control rate  
 self.cont\_rate['high'] = fuzz.trapmf(self.cont\_rate.universe, [0.05, 0.1, 0.2, 0.2])  
 self.cont\_rate['avghigh'] = fuzz.trimf(self.cont\_rate.universe, [0, 0.05, 0.1])  
 self.cont\_rate['avg']= fuzz.trimf(self.cont\_rate.universe, [-0.05, 0, 0.05])  
 self.cont\_rate['avglow'] = fuzz.trimf(self.cont\_rate.universe, [-0.1, -0.05, 0])  
 self.cont\_rate['low']= fuzz.trapmf(self.cont\_rate.universe, [-0.2, -0.2, -0.1, 0.05])  
 self.cont\_rate.view()  
  
 #Define the partitions for failure  
 self.failure['low'] = fuzz.trapmf(self.failure.universe, [-1, -1, -0.5, 0])  
 self.failure['avg'] = fuzz.trimf(self.failure.universe, [-0.5, 0, 0.5])  
 self.failure['high'] = fuzz.trapmf(self.failure.universe, [0, 0.5, 1, 1])  
 self.failure.view()  
  
 #Syntax for rules are taken from here  
 #https://pythonhosted.org/scikit-fuzzy/auto\_examples/plot\_tipping\_problem\_newapi.html  
  
 #Rule1: If vaccination and failure rates are high, then control can be low.  
 rule1 = ctrl.Rule(antecedent=(self.vacc\_set['high'] & self.failure['high']), consequent=self.cont\_rate['low'])  
 # Rule2: Between vaccination and failure rates, if one of them is high and the other one is average, then control can be increased to average low.  
 rule2 = ctrl.Rule(antecedent=((self.vacc\_set['high'] & self.failure['avg']) | (self.vacc\_set['avg'] & self.failure['high']) ),  
 consequent= self.cont\_rate['avglow'])  
 #Rule3: If the mean of the failure and vaccination rates is 'average', then control can be increased to average.  
 #For example vaccination rate is low, failure rate is high -> (low+high)/2 = average  
 rule3 = ctrl.Rule(antecedent=((self.vacc\_set['avg'] & self.failure['avg']) | (self.vacc\_set['low'] & self.failure['high']) |  
 (self.vacc\_set['high'] & self.failure['low'])), consequent=self.cont\_rate['avg'])  
 # Rule4: Between vaccination and failure rates, if one of them is low and the other one is average, then control can be increased to average high.  
 rule4 = ctrl.Rule(antecedent=((self.vacc\_set['low'] & self.failure['avg']) | (self.vacc\_set['avg'] & self.failure['low'])),  
 consequent=self.cont\_rate['avghigh'])  
 # Rule5: If vaccination and failure rates are high, then control can be low.  
 rule5 = ctrl.Rule(antecedent=(self.vacc\_set['low'] & self.failure['low']), consequent=self.cont\_rate['high'])  
  
 #Set control rules  
 self.cont\_rule = ctrl.ControlSystem(rules=[rule1, rule2, rule3, rule4, rule5])  
 #Call Vaccination.py  
 self.model = Vaccination()  
  
  
 def FuzzyLogic(self):  
 #Directly taken from here  
 #https://pythonhosted.org/scikit-fuzzy/auto\_examples/plot\_tipping\_problem\_newapi.html  
 self.fuzz = ctrl.ControlSystemSimulation(self.cont\_rule)  
 self.percentage = self.model.checkVaccinationStatus()[0]  
 self.failure\_inp = self.model.checkVaccinationStatus()[1]  
 #Give inputs  
 self.fuzz.input['vacc\_set'] = self.percentage  
 self.fuzz.input['failure'] = self.failure\_inp  
 #Compute the system  
 self.fuzz.compute()  
 #Get the output  
 self.output\_Control = self.fuzz.output['cont\_rate']  
 self.model.vaccinatePeople(self.output\_Control)  
 def UpdatePercent(self):  
 #Update the variable  
 self.vacc\_perc = self.model.checkVaccinationStatus()[0]  
 def UpdateFail(self):  
 # Update the variable  
 self.failure = self.model.checkVaccinationStatus()[1]  
 def UpdateCost(self):  
 # Update the variable  
 self.cost = self.model.vaccination\_rate\_curve\_[-1]  
  
#Create a variable for check to eq. point.  
checker = 1  
#Initialize cost  
cost = 0  
#Define error value  
error = 0.0001  
#Call the fuzzy system  
Pop = myFuzzy()  
for i in range(200):  
 #Initialize logic  
 Pop.FuzzyLogic()  
 #Update variables  
 Pop.UpdatePercent()  
 Pop.UpdateCost()  
 Pop.UpdateFail()  
 # Update variables  
 if checker == 1:  
 cost += Pop.cost #Update the cost until eq. point  
 diff = abs(Pop.vacc\_perc - 0.6) #Check the difference  
 if (diff < error) and (checker == 1): #If eq. conditions are satisfied  
 checker = 0  
 point\_ss = i #Get the step that reaches eq. point  
  
Pop.model.viewVaccination(point\_ss = point\_ss, vaccination\_cost=cost, filename='vaccination2')