1. Vaccination v1

1.1 Set Partitioning

```
#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])
```

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.

1.2 Fuzzy Control Rules

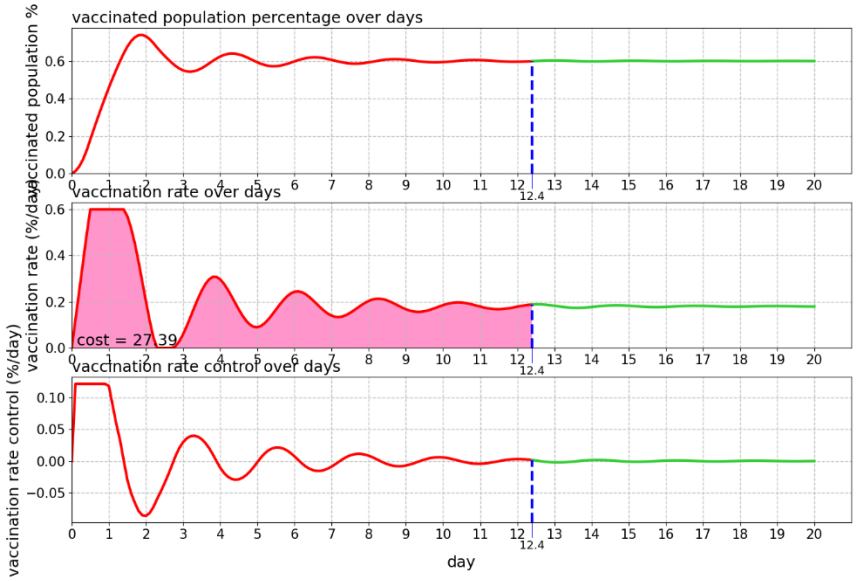
```
#Overall rule is that the vaccination rate and control rate is inversely proportional.
#As one of them increases, decrease the other one.

#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)
```

1.3 Fuzzification and Defuzzification Interface

```
#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)
```

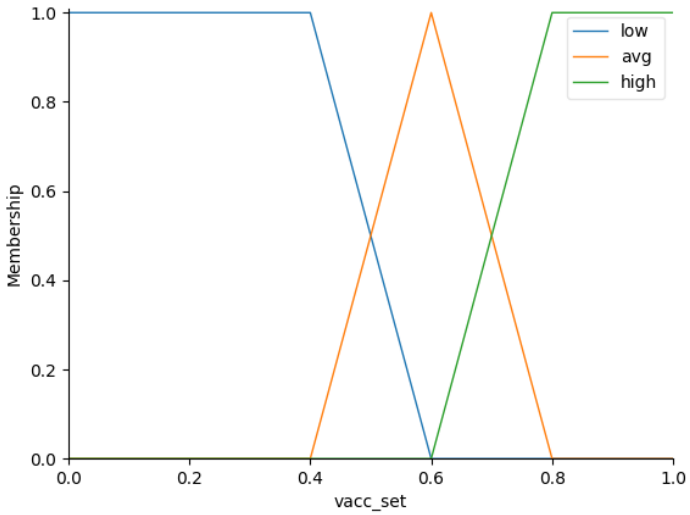
Aggregate all three output membership functions together in the first line. Then, calculate the defuzzified result with centroid method. Then, take necessary actions.

1.4 Simulation

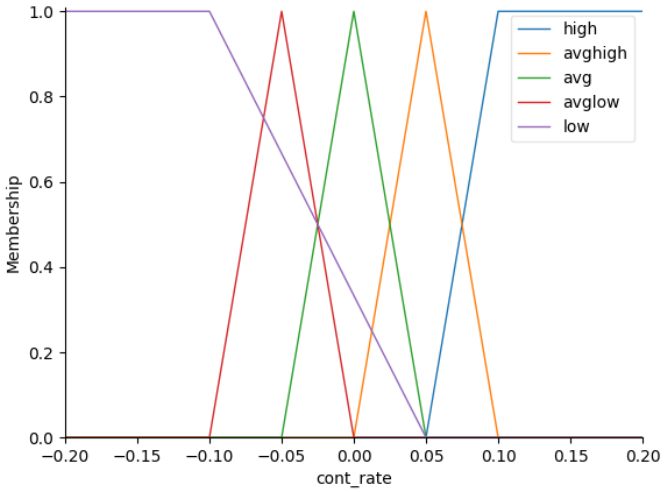


2. Vaccination v2

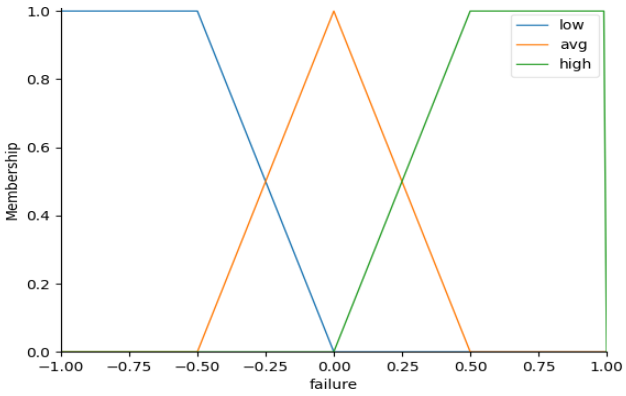
2.1 Set Partitioning



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.

2.2 Fuzzy Control Rules

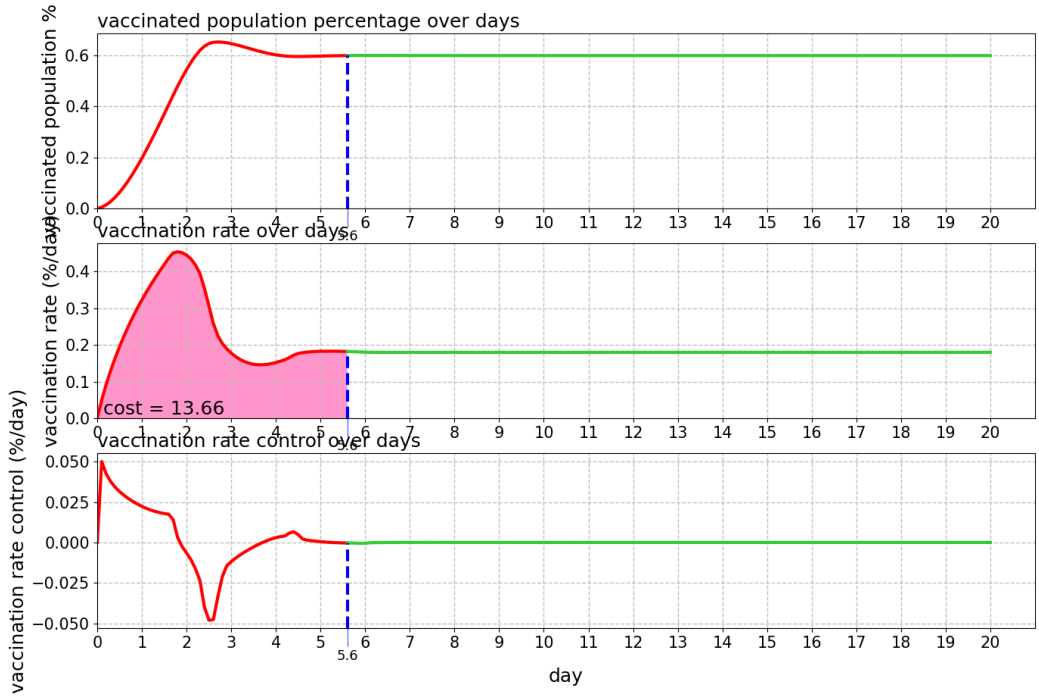
```
#Syntax for rules are taken from here
#https://pythonhosted.org/scikit-fuzzy/auto_examples/plot_tipping_problem_newapi.html


#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'])
```

2.3 Fuzzification and Defuzzification Interface

```
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)
```

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.

2.4 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')
```