```python
from plague import Plague
import numpy as np
import random
import matplotlib.pyplot as plt


def fuzzy_controller(infected_percentage):
    # Calculate the memberships of the infected bots rate for 3 different regions
    memberships = [calculate_membership_low_infected(infected_percentage),
                   calculate_membership_good_infected(infected_percentage),
                   calculate_membership_high_infected(infected_percentage)]

    best_member = memberships.index(max(memberships))
    # According to the highest memberships, evaluate the output control variable by using The Max
Criterion method
    if best_member == 0:
        return output_controller_low(memberships[0])
    elif best_member == 1:
        return output_controller_good(memberships[1])
    else:
        return output_controller_high(memberships[2])


# Calculating the membership for low infected bots
def calculate_membership_low_infected(x):
    if 0.4 >= x > 0:
        return_value = 1
    elif 0.55 >= x >= 0.4:
        return_value = -(x / 0.15) + 3.6666
    else:
        return_value = 0
    return return_value


# Calculating the output control variable from membership (Low)
def output_controller_low(x):
    first_point = (0.15 * x / 2)
    second_point = -((x - 2) * 0.15 / 2)
    return_value = random.uniform(first_point, second_point)
    return return_value


# Calculating the membership for good rate infected bots
def calculate_membership_good_infected(x):
    if 0.7 >= x >= 0.6:
        return_value = -10 * x + 7
    elif 0.6 >= x >= 0.5:
        return_value = (10 * x) - 5
    else:
        return_value = 0
    return return_value


# Calculating the output control variable from membership (Good)
def output_controller_good(x):
    first_point = (x - 1) * 0.15 / 2
    second_point = -((x - 1) * 0.15 / 2)
    return_value = random.uniform(first_point, second_point)
    return return_value


# Calculating the membership for High rate infected bots
def calculate_membership_high_infected(x):
    if 1 >= x >= 0.8:
        return_value = 1
    elif 0.8 >= x >= 0.65:
        return_value = (x / 0.15) - 4.33
    else:
        return_value = 0
    return return_value


# Calculating the output control variable from membership (High)
def output_controller_high(x):
    first_point = ((x - 2) * 0.15 / 2)
    second_point = -(0.15 * x / 2)
    return_value = random.uniform(first_point, second_point)
    return return_value


plague = Plague()

number_iterations = 200
counter = 0
infected_bots_plot = np.empty(shape=(200,))
effective_infection_rates = np.empty(shape=(200,))
# Loop for spreading virus for 20 days
while counter != number_iterations:
    # Get the rates for the infected bot and effective infection
    infected_bots, effective_infection_rate = plague.checkInfectionStatus()
    # Add them to array for debugging and plotting
    infected_bots_plot[counter] = infected_bots
    effective_infection_rates[counter] = effective_infection_rate
    # Use infected bots rate in order to generate a control variable
    control_variable = fuzzy_controller(infected_bots)
    # Spread virus
    plague.spreadPlague(control_variable)
    # Increase the loop iteration
    counter += 1

# Plotting
plt.ylabel("Infection rate")
plt.xlabel("Steps")
plt.plot(plague.infected_percentage_curve_)
plt.show()

plt.ylabel("Infection cost")
plt.xlabel("Steps")
plt.plot(plague.infection_rate_curve_)
plt.show()

plt.ylabel("effective infection rate")
plt.xlabel("Steps")
plt.plot(effective_infection_rates)
plt.show()

# Computation of the total cost until equilibrium
cost_sum = sum(plague.infected_percentage_curve_[1:100])

plague.viewPlague(100, cost_sum)

```