ИНБИКСТ МФТИ

Карпов В.Э., Сорокоумов П.С.

Указания по выполнению практических работ
по курсу “Управление в технических системах”

Работа 4. Нечёткие системы управления

2019 г.

Цель работы
Освоить методы создания и оценки качества нечётких регуляторов.

Задачи

1. Освоить работу со средствами Scikit для разработки нечётких систем управления.
2. Разработать управляющий компонент с нечёткой логикой для заданной системы управления.

Порядок выполнения работы

1. Установите scikit-fuzzy - пакет для работы с нечёткой логикой. Сделать это можно с помощью менеджера пакетов pip. В зависимости от конфигурации имеющегося ПО можно сделать это командой
pip install scikit-fuzzy
для установки в общую системную директорию либо
pip install --user scikit-fuzzy
для установки в директорию текущего пользователя.
Руководство пользователя установленного пакета находится по адресу https://pythonhosted.org/scikit-fuzzy/user_guide.html . Для ознакомления с возможностями пакета полезен справочник https://pythonhosted.org/scikit-fuzzy/api/skfuzzy.html . Названия всех методов создания функций принадлежности оканчиваются на mf (membership function).

2. Создадим нечёткую систему управления для того же маятника, с которым работал в прошлый раз.

```
import math
```

```
import numpy as np
from scipy import signal
import skfuzzy as fuzz
from skfuzzy import control as ctrl
import matplotlib.pyplot as plt
```

```
dT = 0.1
TotalTime = 5
Tin = np.linspace(0, TotalTime, int(TotalTime/dT) + 1)
m = 0.1
L = 0.5
g = 9.81
sys_tf = signal.tf2ss([1], [m*L**2, 0, -m*g*L])
```

```
(array([[-0. , 19.62], [ 1. , 0. ]]),
 array([[1.], [0.]]),
 array([[0., 40.]]),
 array([[0.]]))
```

3. Нечёткий регулятор будет формировать входной сигнал для управляемой системы по разнице между желаемым и действительным положением маятника, то есть будет занимать в итоговой системе то же место, что и PID-регулятор ранее. Тогда на его вход будет поступать угол отклонения маятника от желаемого положения, а на выходе следует формировать момент силы. Пусть и на выходе, и на входе у него будут по две лингвистические переменные: <<угол велик>>, <<угол мал>>, <<сигнал велик>>, <<сигнал мал>>. Выберем,

например, для входных сигналов форму функции в виде гауссиана, для выходных --- треугольную (это решение в данном случае абсолютно произвольно и не вызвано никакой технической необходимостью):

```
max_angle = 15.0 / 180 * math.pi
inp_discr = 1.0 / 180 * math.pi
inp_mean = 10.0 / 180 * math.pi
inp_sigma = 5.0 / 180 * math.pi

inp_values = np.linspace(-max_angle, max_angle, int(2 * max_angle
/ inp_discr) + 1)
input = ctrl.Antecedent(inp_values, 'input')
input['low'] = fuzz.gaussmf(input.universe, -inp_mean, inp_sigma)
input['high'] = fuzz.gaussmf(input.universe, inp_mean, inp_sigma)

input.view()
plt.show()

max_output = 20
output_discr = 1
outp_mean = 10
outp_hbreadth = 3

outp_values = np.linspace(-max_output, max_output, int(2 *
max_output / output_discr) + 1)
output = ctrl.Consequent(outp_values, 'output')
output['low'] = fuzz.trimf(output.universe, [-outp_mean -
outp_hbreadth, -outp_mean, -outp_mean + outp_hbreadth])
output['high'] = fuzz.trimf(output.universe, [outp_mean -
outp_hbreadth, outp_mean, outp_mean + outp_hbreadth])

output.view()
plt.show()
```

4. Создадим правила, которые определяют зависимости выходов от входов:

```
rule1 = ctrl.Rule(input['low'] , output['high'])
rule2 = ctrl.Rule(input['high'] , output['low'])
```

5. Создадим систему как совокупность правил.

```
control_system =  ctrl.ControlSystem([rule1, rule2])
```

6. Выполним вычисления для заданного входного значения.

```
simulation = ctrl.ControlSystemSimulation(control_system)
simulation.input['input'] = 0.1
simulation.compute()
print(simulation.output['output'])
```

7. Теперь, когда работоспособность нечёткой системы проверена, можно формировать входные сигналы для системы управления маятником с её помощью. Так как передаточной

функции у данной системы нет, придётся вручную передавать её сигналы на вход симулятора системы; его состояние также будем сохранять вручную. Для удобства код нечёткого регулятора вынесен в отдельный класс (см. прилагаемый файл fuzzy_reg.py).

```
import math

import numpy as np
from scipy import signal
import matplotlib.pyplot as plt

import fuzzy_reg

# создаём управляемую систему
dT = 0.01
TotalTime = 5
moments_num = int(TotalTime/dT) + 1
Tin = np.linspace(0, TotalTime, moments_num)
m = 0.1
L = 0.5
g = 9.81
A,B,C,D = signal.tf2ss([1], [m*L**2, 0, -m*g*L])
sys_tf = signal.StateSpace(A,B,C,D)

# создаём нечёткий регулятор
fc = fuzzy_reg.FuzzyController()
# выведем зависимость выхода от входа
fc.plot()

# задаём начальные условия
init_angle = 5.0 / 180.0 * math.pi
out_pos = np.zeros(moments_num + 1)
out_pos[0] = init_angle
curr_state = [0, out_pos[0]/40]

# для каждого момента времени
for i in range(moments_num):
    # вычислим сигнал нечёткого регулятора
    inp = fc.calc(curr_state[1]*40) # передаём угол, а не
переменную состояния
    # вычислим выход управляемой системы по входу
    Tout,yout,xout = signal.lsim(sys_tf, [inp,inp], [0,dT],
X0=curr_state)
    curr_state = xout[-1]
    out_pos[i+1] = yout[-1]

plt.plot(Tin, out_pos[:-1],'b')
limit = 15 / 180.0 * np.pi
plt.plot([0, TotalTime], [limit, limit], 'r')
plt.plot([0, TotalTime], [-limit, -limit], 'r')
plt.show()
```

8. Варьируйте параметры нечёткого регулятора, отмечая, что изменяется в его поведении.

Контрольные задания и вопросы

1. Что такое «лингвистическая переменная»?

2. В чём разница между функцией принадлежности лингвистической переменной и функцией распределения случайной величины?

3. Отражает ли функция принадлежности вероятность того, что объект, характеризуемый входным параметром, принадлежит к определённому классу?

4. Является ли линейной полученная нами нечёткая система управления?

5. Какими свойствами должны обладать методы вычисления логических функций от нечётких величин?

6. Как задать вид функции принадлежности лингвистической переменной? Могут ли при фаззификации одного входного значения сочетаться разные типы функций принадлежности (например, линейные аппроксимации и гауссианы)?

7. Могут ли несколько правил выдавать ненулевую достоверность одновременно для одного набора входных переменных?

8. Как выполняется дефаззификация по алгоритму Мамдани?

9. Как поведёт себя нечёткая система управления, если значение входной переменной не принадлежит допустимому диапазону?

Задания для самостоятельной работы

1. В качестве управляемой системы используйте тот же обратный маятник, который моделировали в работе 3. Используйте при создании нечёткого регулятора указанное число лингвистических переменных:

    1. 3 входных, 2 выходных.
    2. 2 входных, 4 выходных.
    3. 2 входных, 2 выходных.
    4. 4 входных, 2 выходных.
    5. 3 входных, 3 выходных.
    6. 3 входных, 3 выходных.
    7. 4 входных, 2 выходных.
    8. 2 входных, 2 выходных.
    9. 2 входных, 4 выходных.
    10. 3 входных, 2 выходных.

2. Задайте функции принадлежности, описывающие каждую входную и выходную лингвистическую переменную.
3. Задайте правила работы нечёткого компонента.
4. Оцените правильность работы нечёткого регулятора, передавая на него разные комбинации входных сигналов. Постройте зависимость его выхода от какого-либо входного параметра при фиксированных остальных.
5. Реализуйте требуемую систему управления. Является ли её реакция на сигнал ограниченной во времени и величине?
6. Постарайтесь настроить нечёткий регулятор так, чтобы он приводил маятник в вертикальное положение за достаточно малое время при любых отклонениях от равновесия в пределах 15°. Для некоторых комбинаций лингвистических переменных и управляемых

систем корректная настройка может быть затруднена, поэтому для сдачи достаточно продемонстрировать верную работу нечёткого регулятора совместно с управляемой системой.