



WYDZIAŁ  
**ELEKTROTECHNIKI  
I INFORMATYKI**  
POLITECHNIKI RZESZOWSKIEJ

**Sztuczna inteligencja  
Laboratorium**

*Synteza układu wnioskującego*

**Stanislau Antanovich**  
nr. indeksu: 173590  
gr. lab: L04

## Spis treści

<b>1</b>	<b>Wstęp</b>	<b>2</b>
1.1	Cel ćwiczenia . . . . .	2
<b>2</b>	<b>System ekspertowy typu Mamdaniego</b>	<b>2</b>
2.1	Problem . . . . .	2
2.2	Realizacja . . . . .	3
2.2.1	Inicjowanie modułów . . . . .	3
2.2.2	Tworzenie zmiennych stanu poprzednika “obsługa” oraz następnika “napiwek” . . . . .	4
2.2.3	Dodanie zbiorów rozmytych . . . . .	4
2.2.4	Podgląd zbiorów rozmytych . . . . .	4
2.2.5	Definicja reguł . . . . .	4
2.2.6	Dodanie reguł do systemu rozmytego . . . . .	4
2.2.7	Sprawdzenie działania systemu . . . . .	4
2.2.8	Sprawdzenie działania systemu dla wartości obsługi . . . . .	5
2.2.9	Dodanie drugiej wejściowej . . . . .	6
2.2.10	Dodanie reguł 4 i 5 . . . . .	6
2.2.11	Sprawdzenie działania systemu dla wartości “obsługi” . . . . .	6
2.2.12	Sprawdzenie działania systemu dla wartości “obsługi” i “jedzenia” . . . . .	6
<b>3</b>	<b>Modyfikacja systemu</b>	<b>7</b>
<b>4</b>	<b>Przykładowe zadania zaliczeniowe</b>	<b>7</b>
<b>5</b>	<b>Wnioski</b>	<b>7</b>

## Spis rysunków

1	<i>Obsługa</i> . . . . .	2
2	<i>Napiwek</i> . . . . .	3
3	<i>Jedzenie</i> . . . . .	3
4	<i>Wyostrzenie metodą środka ciężkości dla wejścia obsługa = 0</i> . . . . .	5
5	<i>Wyostrzenie metodą środka ciężkości dla wejścia obsługa = 10</i> . . . . .	5
6	<i>Powierzchnia przejścia systemu “napiwek” o jednym wejściu</i> . . . . .	6
7	<i>Powierzchnia przejścia systemu “napiwek” o dwóch wejściach</i> . . . . .	7

# 1 Wstęp

## 1.1 Cel ćwiczenia

Laboratorium składa się z trzech zasadniczych części. Część 2 ma na celu zapoznanie się ze sposobem syntezy rozmytego systemu ekspertowego typu Mamdaniego z wykorzystaniem biblioteki **scikit-fuzzy**. W części 3, należy zapoznać się z ideą działania systemu Mamdaniego a następnie dokonać modyfikacji systemu wykonanego w części 2. Część 4 laboratorium polega na wykonaniu przykładowego zadania zaliczeniowego.

# 2 System ekspertowy typu Mamdaniego

## 2.1 Problem

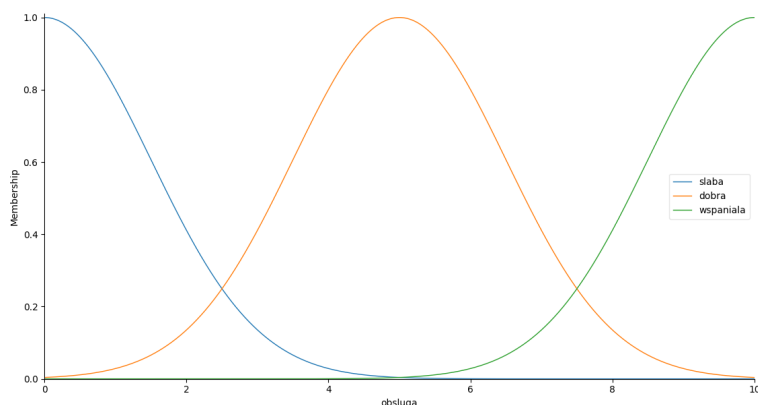
Zaprojektować rozmyty układ ekspertowy doradzający ile napiwku pozostawić w restauracji na podstawie oceny jakości obsługi oraz jakości jedzenia. Jakość obsługi i jakość jedzenia będzie oceniana w skali od 1 do 10, gdzie 10 reprezentuje ocenę maksymalną, natomiast napiwek będzie liczbą z przedziału  $[0,30]$  reprezentującą procent wartości rachunku.

Baza reguł będzie składała się z 5 reguł. System zostanie wykonany w dwóch etapach. W **etapie pierwszym**(rys. 1 i 2) system będzie zbudowany z jednego wejścia(*obsługa*) i jednego wyjścia(*napiwek*) oraz 3 reguł postaci:

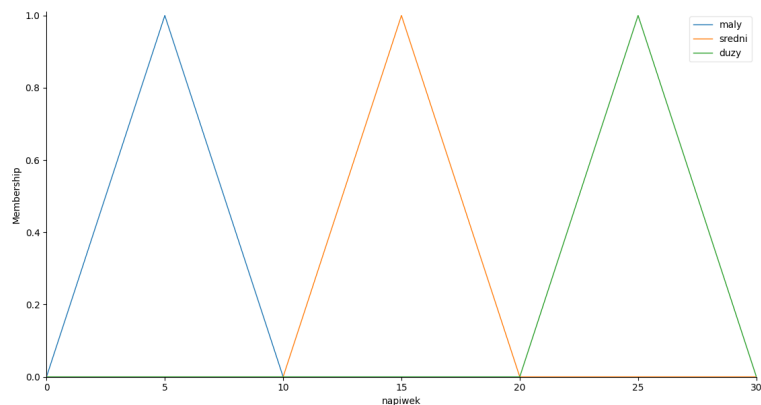
R1 *jeżeli obsługa jest słaba, to napiwek jest mały*

R2 *jeżeli obsługa jest dobra, to napiwek jest średni*

R3 *jeżeli obsługa jest wspaniała, to napiwek jest duży*



Rysunek 1: *Obsługa*

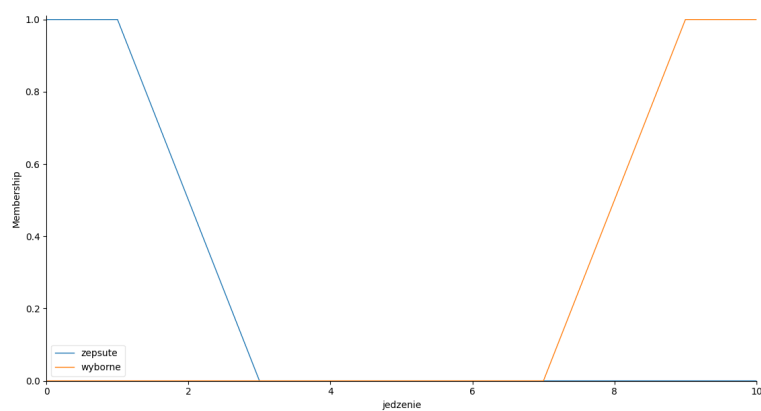


Rysunek 2: *Napiwek*

W **etapie drugim** do systemu zostanie dodana gruga zmienna wejściowa *jedzenie* (rys. 3) oraz 2 dodatkowe reguły

R4 *jeżeli jedzenie jest zepsute, to napiwek jest mały*

R5 *jeżeli jedzenie jest wyborne, to napiwek jest duży*



Rysunek 3: *Jedzenie*

## 2.2 Realizacja

### 2.2.1 Inicjowanie modułów

W tej sekcji inicjujemy wszystkie wymagane biblioteki dla prawidłowego działania programu.

```
1 import numpy as np
2 import skfuzzy as fuzz
3 from skfuzzy import control as ctrl
4 import matplotlib.pyplot as plt
```

### 2.2.2 Tworzenie zmiennych stanu poprzednika “obsługa” oraz następnika “napiwek”

W tej sekcji tworzymy zmienne stanu poprzednika “obsługa” oraz następnika “napiwek”.

```
1 obsluga = ctrl.Antecedent(np.arange(0, 10.01, 0.01), 'obsługa')
2 napiwek = ctrl.Consequent(np.arange(0, 30.01, 0.01), 'napiwek')
```

### 2.2.3 Dodanie zbiorów rozmytych

W tej sekcji do zmiennej *obsługa* dodajemy następujące zbiory: *slaba*, *dobra*, *wspaniala*.

Zbiór *slaba* o centrum umieszczonym w punkcie uniwersum równym **0** i rozpiętości wynoszącej **1.5**. Dla zbiorów *dobra* i *wspaniala* o centrach ułożonych w punktach odpowiednio **5** oraz **10** i rozpiętości wynoszącej **1.5**

Dla zmiennej *napiwek* dodajemy zbiory trójkątne: *maly*, *sredni*, *duzy* o parametrach [0, 5, 10], [10, 15, 20] i [20, 25, 30] odpowiednio.

```
1 obsluga['slaba'] = fuzz.gaussmf(obsluga.universe, 0, 1.5)
2 obsluga['dobra'] = fuzz.gaussmf(obsluga.universe, 5, 1.5)
3 obsluga['wspaniala'] = fuzz.gaussmf(obsluga.universe, 10, 1.5)
4
5 napiwek['maly'] = fuzz.trimf(napiwek.universe, [0, 5, 10])
6 napiwek['sredni'] = fuzz.trimf(napiwek.universe, [10, 15, 20])
7 napiwek['duzy'] = fuzz.trimf(napiwek.universe, [20, 25, 30])
```

### 2.2.4 Podgląd zbiorów rozmytych

Podgląd zbiorów rozmytych można zrealizować metodą `view()` dla poszczególnych zmiennych stanu.

```
1 obsluga.view()
2 napiwek.view()
```

### 2.2.5 Definicja reguł

W tej sekcji definiujemy reguły.

```
1 regula1 = ctrl.Rule(obsluga['slaba'], napiwek['maly'])
2 regula2 = ctrl.Rule(obsluga['dobra'], napiwek['sredni'])
3 regula3 = ctrl.Rule(obsluga['wspaniala'], napiwek['duzy'])
```

### 2.2.6 Dodanie reguł do systemu rozmytego

W tej sekcji dodajemy powyżej zdefiniowane reguły do systemu rozmytego.

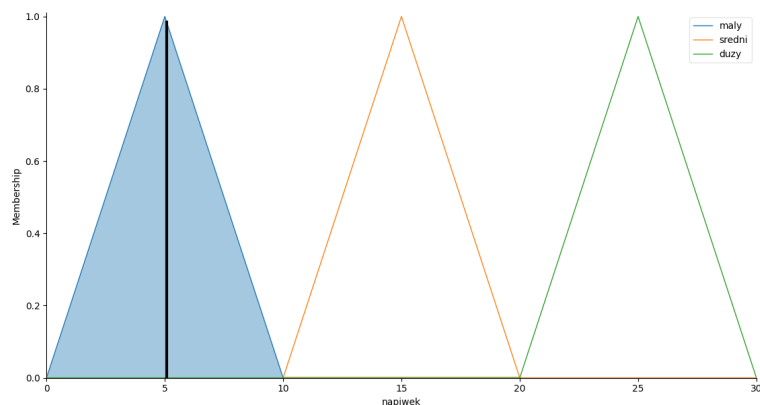
```
1 napiwek_ctr = ctrl.ControlSystem([regula1, regula2, regula3, regula4,
2   regula5])
3 napiwek_sym = ctrl.ControlSystemSimulation(napiwek_ctr)
```

### 2.2.7 Sprawdzenie działania systemu

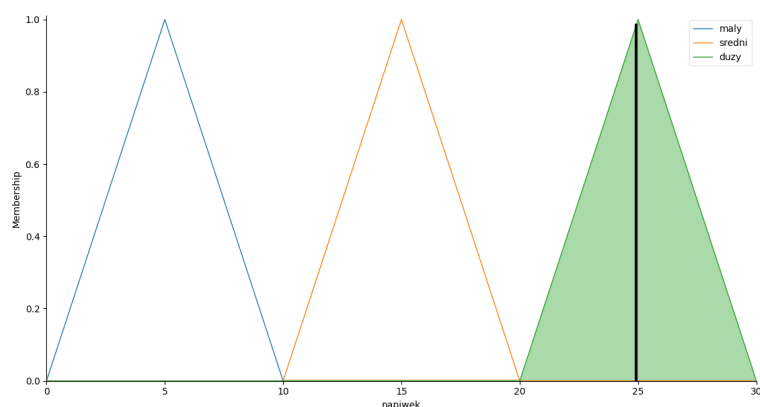
W tej sekcji sprawdzamy działanie systemu dla wartości obsługi równej **0** (rys. 4) oraz dla wartości równej **10** (rys. 5).

```
1 napiwek_sym.input['obsługa'] = 0
2 #napiwek_sym.input['obsługa'] = 10 # dla wartosci rownej 10
3 napiwek_sym.compute()
4 print('Wynik', napiwek_sym.output['napiwek'])
5 napiwek.view(sim=napiwek_sym)
```

Wartość napiwku dla wartości obsługi wynosi: 5.07657801  
 Wartość napiwku dla wartości obsługi wynosi: 24.9234219



Rysunek 4: Wyostrzenie metodą środka ciężkości dla wejścia obsługa = 0



Rysunek 5: Wyostrzenie metodą środka ciężkości dla wejścia obsługa = 10

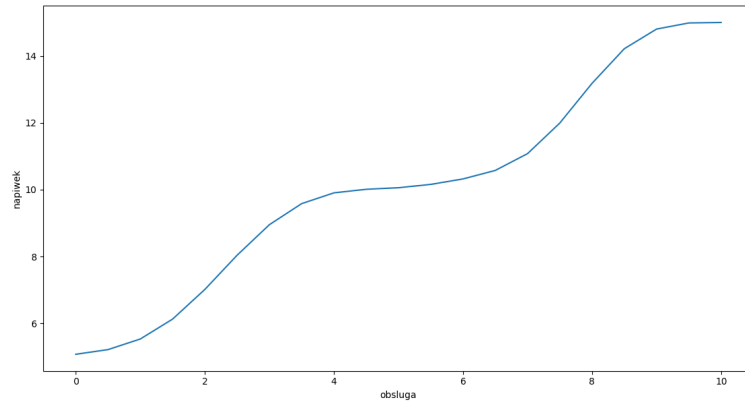
## 2.2.8 Sprawdzenie działania systemu dla wartości obsługi

W tej sekcji sprawdzamy działanie systemu dla wartości obsługi od 0 do 10 (rys. 6) wykres funkcji  $napiwek = f(obsługa)$ .

```

1 n_points = 21
2 x = np.linspace(0, 10, n_points)
3 z = np.zeros_like(x)
4 for i in range(n_points):
5     napiwek_sym.input['obsługa'] = x[i]
6     napiwek_sym.compute()
7     z[i] = napiwek_sym.output['napiwek']
8
9 fig, ax = plt.subplots()
10 ax.set_xlabel('obsługa')
11 ax.set_ylabel('napiwek')
12 ax.plot(x, z)
13 plt.show()

```



Rysunek 6: Powierzchnia przejścia systemu “napiwek” o jednym wejściu

### 2.2.9 Dodanie drugiej wejściowej

W tej sekcji dodajemy drugą wejściową zmienną stanu “jedzenie” o trapezoidalnych(trapmf) zbiorach rozmytych “zepsute” oraz “wyborne”.

```
1 jedzenie = ctrl.Antecedent(np.arange(0, 10.01, 0.01), 'jedzenie')

1 jedzenie['zepsute'] = fuzz.trapmf(jedzenie.universe, [-2, 0, 1, 3])
2 jedzenie['wyborne'] = fuzz.trapmf(jedzenie.universe, [7, 9, 10, 12])
```

### 2.2.10 Dodanie reguł 4 i 5

W tej sekcji dodajemy reguły 4 i 5 analogicznie do punktu 2.2.5.

```
1 regula4 = ctrl.Rule(jedzenie['zepsute'], napiwek['maly'])
2 regula5 = ctrl.Rule(jedzenie['wyborne'], napiwek['duzy'])
```

### 2.2.11 Sprawdzenie działania systemu dla wartości “obsługi”

W tej sekcji sprawdzamy działanie systemu dla wartości obsługi równej 0 oraz wartości jedzenia równej 0.

```
1 napiwek_sym.input['jedzenie'] = 0
2 napiwek_sym.input['obsługa'] = 0
3 #napiwek_sym.input['obsługa'] = 10 # dla wartosci rownej 10
4 napiwek_sym.compute()
5 print('Wynik', napiwek_sym.output['napiwek'])
6 napiwek.view(sim=napiwek_sym)
```

### 2.2.12 Sprawdzenie działania systemu dla wartości “obsługi” i “jedzenia”

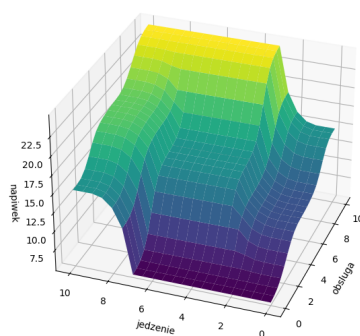
W tej sekcji sprawdzamy działanie systemu dla wartości obsługi i jedzenia od 0 do 10(rys. 7)

```
1 n_points = 21
2 upsampled = np.linspace(0, 10, n_points)
3 x, y = np.meshgrid(upsampled, upsampled)
4 z = np.zeros_like(x)
5
6 for i in range(n_points):
7     for j in range(n_points):
```

```

8         napiwek_sym.input['obsługa'] = x[i, j]
9         napiwek_sym.input['jedzenie'] = y[i, j]
10        napiwek_sym.compute()
11        z[i, j] = napiwek_sym.output['napiwek']
12
13    fig = plt.figure(figsize=(8, 8))
14    ax = fig.add_subplot(111, projection='3d')
15    surf = ax.plot_surface(x, y, z, cmap='viridis')
16
17    ax.set_xlabel('obsługa')
18    ax.set_ylabel('jedzenie')
19    ax.set_zlabel('napiwek')
20    ax.view_init(30, 200)
21    plt.show()

```



Rysunek 7: Powierzchnia przejścia systemu “napiwek” o dwóch wejściach

- 3 Modyfikacja systemu
- 4 Przykładowe zadania zaliczeniowe
- 5 Wnioski