Національний технічний університет України «КПІ ім. Ігоря Сікорського»
Факультет Інформатики та Обчислювальної Техніки
Кафедра інформаційних систем та технологій

Лабораторна робота № 2
з дисципліни «Методи та технології штучного інтелекту»
На тему:
«Моделювання функції з двох змінних засобами нечіткої математики»

Виконала:
студентка групи ІС-12.
Мельникова К.О.

Перевірив:
Шимкович В. М.

Київ – 2023

Мета роботи: Промоделювати засобами нечіткої логіки функцію з двох змінних. Провести дослідження форми функції приналежності на якість моделювання.

Завдання:

1. Побудувати нечітку модель функції двох змінних згідно з варіантом з додатку А, що містить 6 функцій приналежності для вхідних змінних і не менше 9 для вихідної. Варіант з додатку А обрати за номер в списку журналу, якщо ви N = 27-й в списку Ваш варіант М27. Якщо N = 57-й номер в списку, Ваш варіант N-30 = 27, якщо N = 87, відповідно N-60 = 27 і т.п.
2. Дослідити вплив форми функції приналежності (трикутник, трапеція, Гауса) на якість моделювання (порівняти відносні помилки моделювання).
3. Дослідити можливість зменшення числа правил за рахунок виключення деяких (перевірити достатність використання правил, що представляють тільки діагональ таблиці).
4. Зробити висновки.

Виконання роботи:

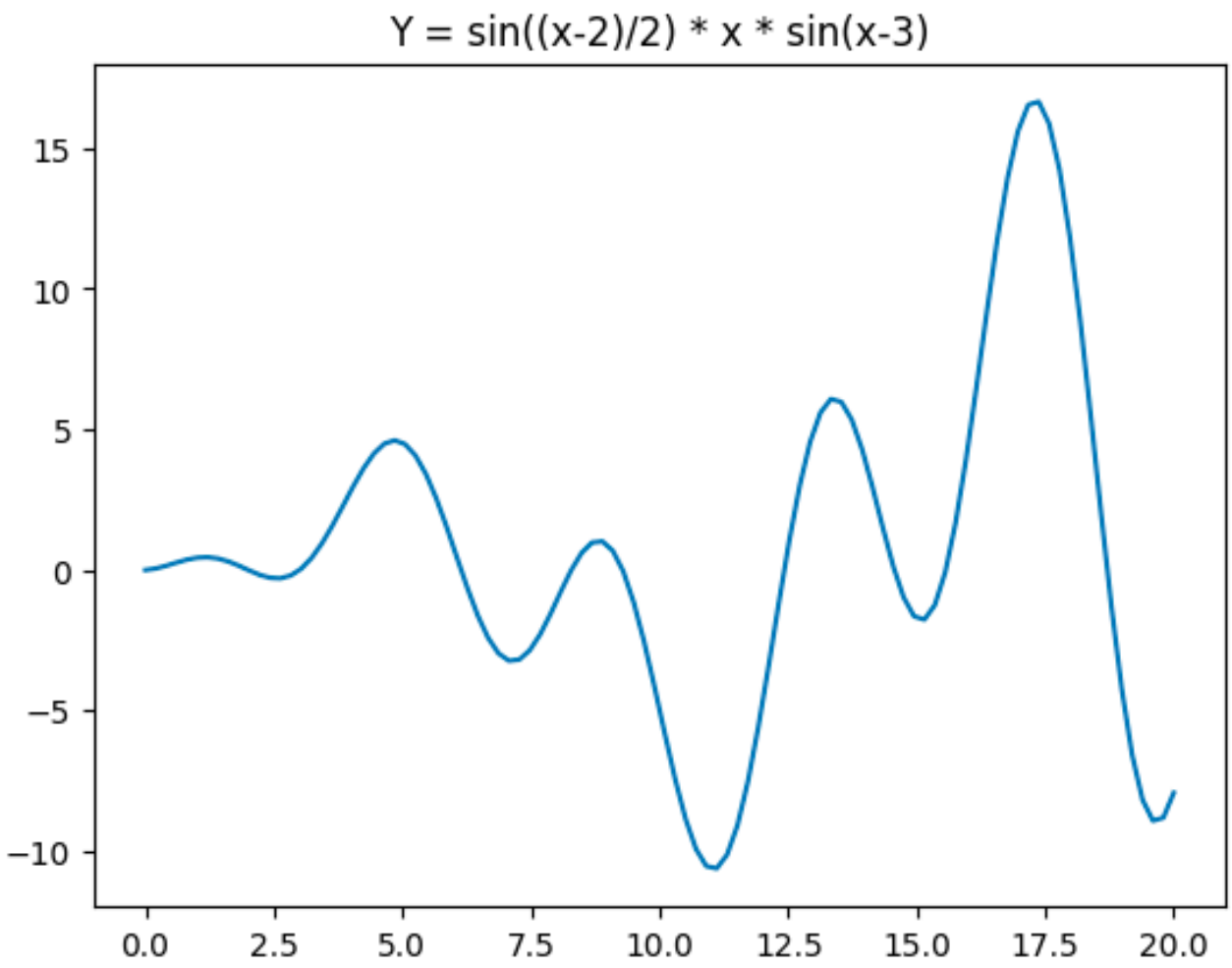
Варіант:

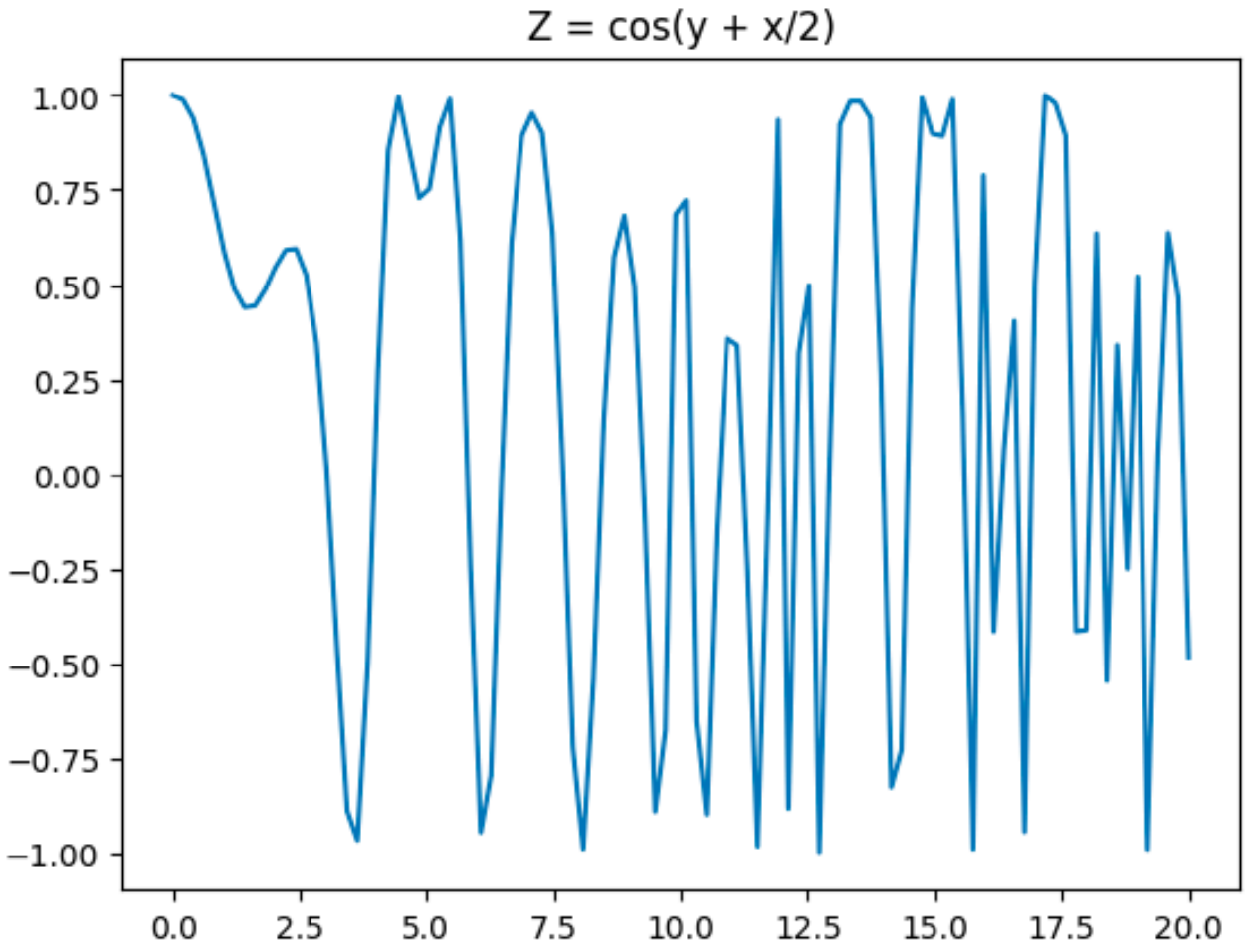
11.	$y=\sin((x-2)/2)\cdot x\cdot\sin(x-3)$ $z=\cos(y+x/2)$	11.	20.	9.

Побудуємо графіки функцій, діапазон значень X від 0 до 20:

```python
import numpy as np
import matplotlib.pyplot as plt
import skfuzzy as fuzz
from sklearn.metrics import mean_squared_error, mean_absolute_error
import pandas as pd
import tabulate as tabulate

x_range = np.linspace(0, 20, 100)
y_func = np.sin((x_range-2)/2)*x_range*np.sin(x_range-3)
z_func = np.cos(y_func+x_range/2)

plt.plot(x_range, y_func)
plt.title("Y = sin((x-2)/2) * x * sin(x-3)")
plt.show()

plt.plot(x_range, z_func)
plt.title("Z = cos(y + x/2)")
plt.show()
```

✓ 0.1s Python





Тепер розіб’ємо дані на проміжки відповідно кількостям функцій приналежності: x – 6 проміжків, y – 6 проміжків, z – 9 проміжків.

```
Межі функцій приналежності x:
[0.0, 1.6666666666666667, 3.3333333333333335]
[3.3333333333333335, 5.0, 6.666666666666667]
[6.666666666666667, 8.333333333333334, 10.0]
[10.0, 11.666666666666668, 13.333333333333334]
[13.333333333333334, 15.0, 16.666666666666668]
[16.666666666666668, 18.333333333333336, 20.0]

Межі функцій приналежності y:
[-10.613936341819958, -8.341562953166921, -6.069189564513883]
[-6.069189564513883, -3.796816175860846, -1.5244427872078088]
[-1.5244427872078088, 0.7479306014452289, 3.0203039900982667]
[3.0203039900982667, 5.2926773787513035, 7.56505076740434]
[7.56505076740434, 9.837424156057377, 12.109797544710414]
[12.109797544710414, 14.382170933363453, 16.65454432201649]

Межі функцій приналежності z:
[-0.9989575810205715, -0.8879043820749842, -0.7768511831293969]
[-0.7768511831293969, -0.6657979841838096, -0.5547447852382223]
[-0.5547447852382223, -0.443691586292635, -0.33263838734704765]
[-0.33263838734704765, -0.22158518840146035, -0.11053198945587306]
[-0.11053198945587306, 0.0005212094897142916, 0.11157440843530164]
[0.11157440843530164, 0.22262760738088894, 0.33368080632647623]
[0.33368080632647623, 0.4447340052720635, 0.5557872042176508]
[0.5557872042176508, 0.6668404031632381, 0.7778936021088254]
[0.7778936021088254, 0.8889468010544127, 1.0]
```

```python
# порахуємо інтервали
def calculate_intervals(values, n_intervals):
    s = min(values)
    e = max(values)
    intervals = []
    val = np.linspace(s, e, n_intervals + 1)
    for i in range(n_intervals):
        start = val[i]
        end = val[i + 1]
        ser = (start + end) / 2
        intervals.append([start, ser, end])
    return intervals

# Створимо функцію для виведення проміжків
def print_intervals(intervals, name):
    print('\nМежі функцій приналежності ' + str(name) + ':')
    for i in intervals:
        print(i)

# Виведемо межі функцій приналежності:
x_intervals = calculate_intervals(x_range, 6)
print_intervals(x_intervals, 'x')

y_intervals = calculate_intervals(y_func, 6)
print_intervals(y_intervals, 'y')

z_intervals = calculate_intervals(z_func, 9)
print_intervals(z_intervals, 'z')
```

✓ 0.0s

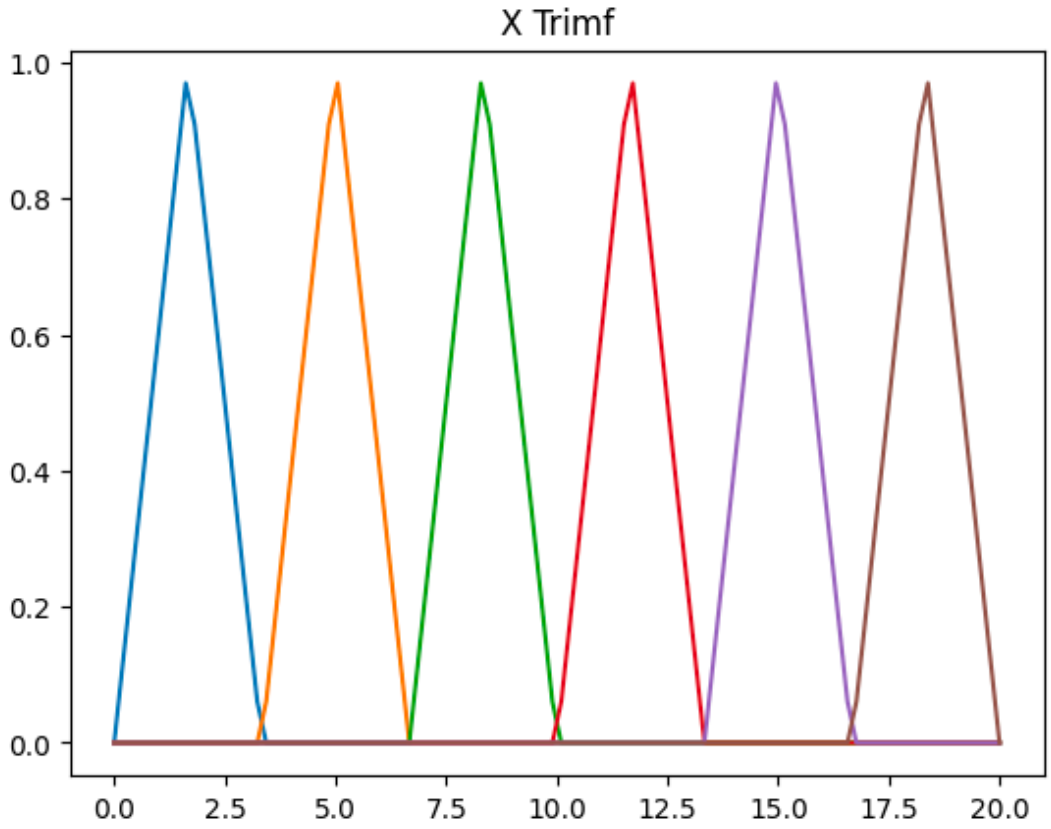
Тепер створимо функції приналежності:

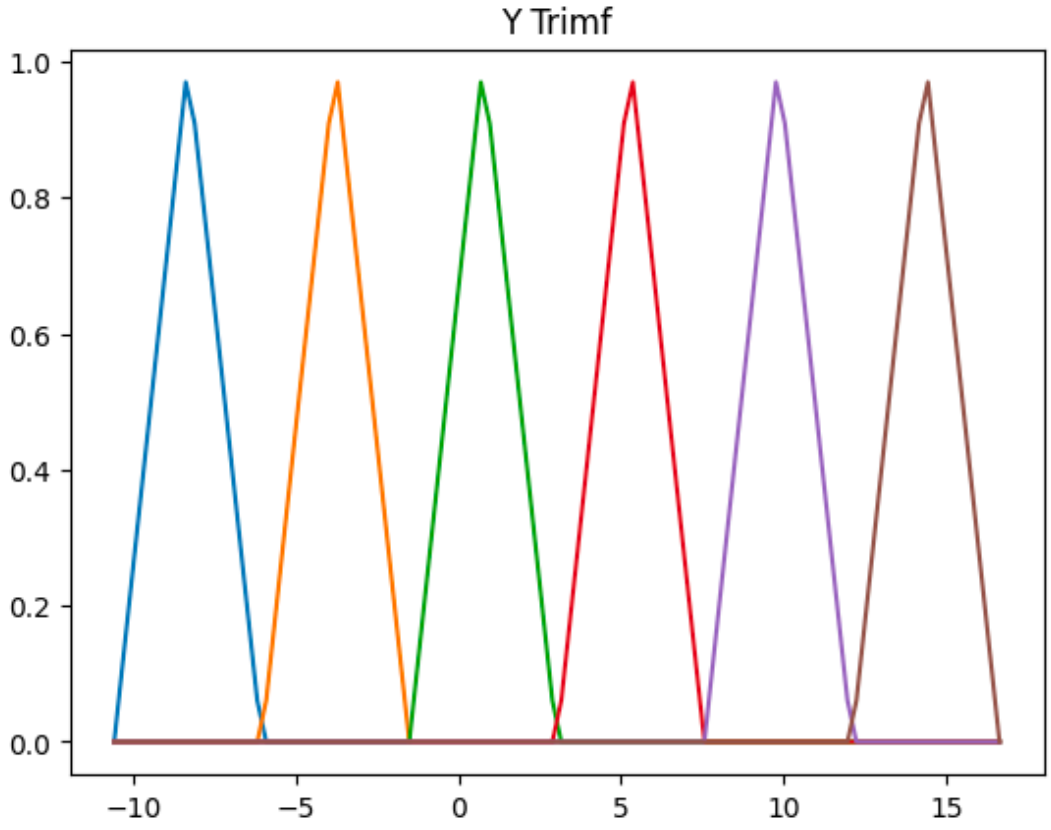
Трикутна функція приналежності:

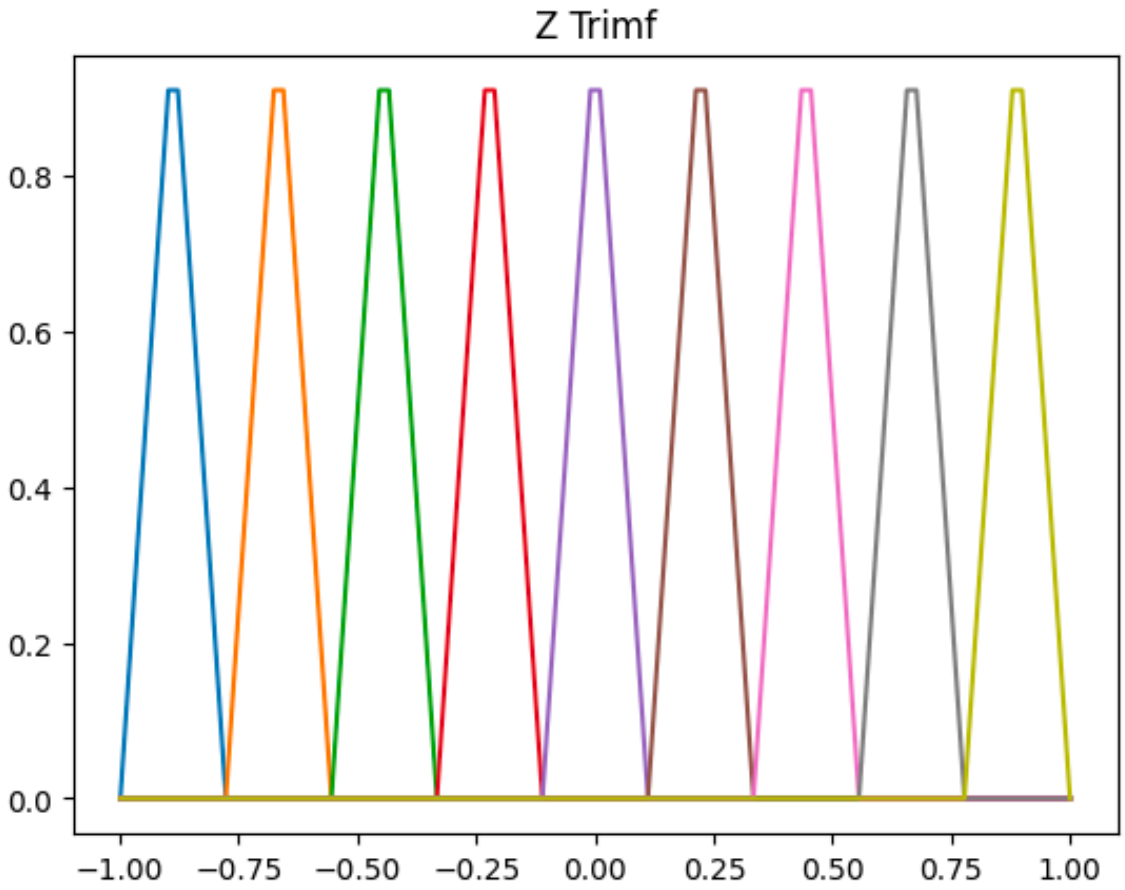
Побудуємо графіки:

```python
# Створимо функцію для виведення графіків функцій приналежності

def show_graphs(space, function, text=""):
    for i in range(len(function)):
        plt.title(text)
        plt.plot(space, function[i])
    plt.show()

graph_x = [fuzz.trimf(x_range, x_intervals[i]) for i in range(6)]
show_graphs(x_range, graph_x, "X Trimf")


y_space = np.linspace(min(y_func), max(y_func), 100)
graph_y = [fuzz.trimf(y_space, y_intervals[i]) for i in range(6)]
show_graphs(y_space, graph_y, "Y Trimf")

z_space = np.linspace(min(z_func), max(z_func), 100)
graph_z = [fuzz.trimf(z_space, z_intervals[i]) for i in range(9)]
show_graphs(z_space, graph_z, "Z Trimf")
```

✓ 0.2s Python


X Trimf
1.0
0.8
0.6
0.4
0.2
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


Y Trimf
1.0
0.8
0.6
0.4
0.2
0.0
−10
−5
0
5
10
15



Тепер створимо і виведемо таблиці приналежності та правила:

Тепер створимо і виведемо таблиці приналежності та правила:

Для спрощення процесу побудови таблиці використаємо бібліотеку `tabulate`

```python
from tabulate import tabulate

# функція для обчислення значень функції
def values_count(x, a, b, c):
    result = 0
    if x >= a and x < b:
        result = (x - a)/(b - a)
    elif x >= b and x <= c:
        result = (c - x)/(c - b)
    return result

# функція що знаходить індекс найкращого значення
def best_value(x, space, step):
    best = float("-inf")
    best_ind = -1
    for i, mid in enumerate(space):
        current_trimf_value = values_count(x, mid - step, mid, mid + step)
        if current_trimf_value > best:
            best = current_trimf_value
            best_ind = i
    return best_ind

# розіб'ємо значення функцій на відповідну к-кість підпроміжків
x_trimf_middle = np.linspace(min(x_range), max(x_range), 6)
y_trimf_middle = np.linspace(min(y_func), max(y_func), 6)
z_trimf_middle = np.linspace(min(z_func), max(z_func), 9)
```

```python
# виведемо таблицю значень
print ("Таблиця значень:")
table = [["y\\x"] + [str(x) for x in x_trimf_middle]]
for y_value in y_trimf_middle:
    row = [round(y_value, 2)]
    for x in x_trimf_middle:
        z = np.cos(y_value+x/2)
        row.append(round(z, 2))
    table.append(row)

print(tabulate(table, tablefmt="grid"))

# виведемо таблицю з назвами функцій
rules = {}
print ("Таблиця з назвами функцій")
table = [["y\\x"] + ["mx" + str(i) for i in range (1, 7)]]
for i in range(6):
    row = ["my" + str(i + 1)]
    for j in range(6):
        z = np.cos(y_trimf_middle[i] + x_trimf_middle[j]/2)
        best_func = best_value(z, z_trimf_middle, 4)
        row.append("mf" + str(best_func + 1))
        rules [(j, i)] = best_func # одночасно записуватимемо значення у словник rules, для подальшого виведення правил
    table.append(row)

print(tabulate(table, tablefmt="grid"))

# виведемо правила
print("\nRules:")
for rule in rules.keys():
    print(f"if (x is mx{rule[0] + 1}) and (y is my{rule[1] + 1}) then (z is mf{rules[rule] + 1})")
```

✓ 0.0s Python

```
Таблиця значень:
+--------+-------+-------+-------+-------+-------+-------+
| y\x    |  0    |  4    |  8    | 12    | 16    | 20    |
+--------+-------+-------+-------+-------+-------+-------+
| -10.61 | -0.37 | -0.69 |  0.95 | -0.1  | -0.86 |  0.82 |
+--------+-------+-------+-------+-------+-------+-------+
| -5.16  |  0.43 | -1    |  0.4  |  0.67 | -0.95 |  0.13 |
+--------+-------+-------+-------+-------+-------+-------+
| 0.29   |  0.96 | -0.66 | -0.41 |  1    | -0.43 | -0.65 |
+--------+-------+-------+-------+-------+-------+-------+
| 5.75   |  0.86 |  0.11 | -0.95 |  0.68 |  0.38 | -1    |
+--------+-------+-------+-------+-------+-------+-------+
| 11.2   |  0.2  |  0.81 | -0.87 | -0.08 |  0.94 | -0.7  |
+--------+-------+-------+-------+-------+-------+-------+
| 16.65  | -0.58 |  0.98 | -0.23 | -0.79 |  0.89 |  0.05 |
+--------+-------+-------+-------+-------+-------+-------+
Таблиця з назвами функцій
+-----+-----+-----+-----+-----+-----+-----+
| y\x | mx1 | mx2 | mx3 | mx4 | mx5 | mx6 |
+-----+-----+-----+-----+-----+-----+-----+
| my1 | mf4 | mf2 | mf9 | mf5 | mf2 | mf8 |
+-----+-----+-----+-----+-----+-----+-----+
| my2 | mf7 | mf1 | mf7 | mf8 | mf1 | mf6 |
+-----+-----+-----+-----+-----+-----+-----+
| my3 | mf9 | mf2 | mf3 | mf9 | mf3 | mf2 |
+-----+-----+-----+-----+-----+-----+-----+
| my4 | mf8 | mf5 | mf1 | mf8 | mf7 | mf1 |
+-----+-----+-----+-----+-----+-----+-----+
```

```
+-----+-----+-----+-----+-----+-----+-----+
| my4 | mf8 | mf5 | mf1 | mf8 | mf7 | mf1 |
+-----+-----+-----+-----+-----+-----+-----+
| my5 | mf6 | mf8 | mf1 | mf5 | mf9 | mf2 |
+-----+-----+-----+-----+-----+-----+-----+
| my6 | mf3 | mf9 | mf4 | mf2 | mf9 | mf5 |
+-----+-----+-----+-----+-----+-----+-----+

Rules:
if (x is mx1) and (y is my1) then (z is mf4)
if (x is mx2) and (y is my1) then (z is mf2)
if (x is mx3) and (y is my1) then (z is mf9)
if (x is mx4) and (y is my1) then (z is mf5)
if (x is mx5) and (y is my1) then (z is mf2)
if (x is mx6) and (y is my1) then (z is mf8)
if (x is mx1) and (y is my2) then (z is mf7)
if (x is mx2) and (y is my2) then (z is mf1)
if (x is mx3) and (y is my2) then (z is mf7)
if (x is mx4) and (y is my2) then (z is mf8)
if (x is mx5) and (y is my2) then (z is mf1)
if (x is mx6) and (y is my2) then (z is mf6)
if (x is mx1) and (y is my3) then (z is mf9)
if (x is mx2) and (y is my3) then (z is mf2)
if (x is mx3) and (y is my3) then (z is mf3)
if (x is mx4) and (y is my3) then (z is mf9)
if (x is mx5) and (y is my3) then (z is mf3)
if (x is mx6) and (y is my3) then (z is mf2)
if (x is mx1) and (y is my4) then (z is mf8)
```

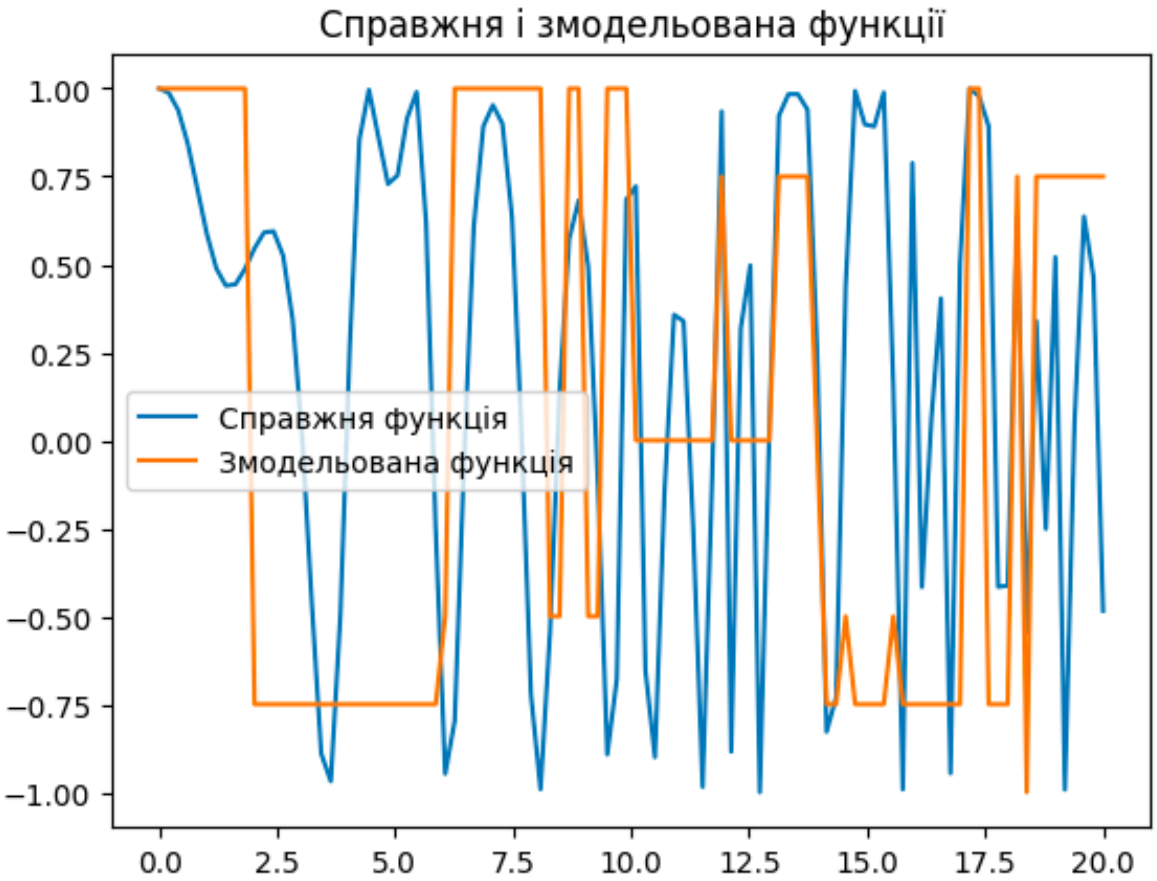
```
if (x is mx1) and (y is my2) then (z is mf7)
if (x is mx2) and (y is my2) then (z is mf1)
if (x is mx3) and (y is my2) then (z is mf7)
if (x is mx4) and (y is my2) then (z is mf8)
if (x is mx5) and (y is my2) then (z is mf1)
if (x is mx6) and (y is my2) then (z is mf6)
if (x is mx1) and (y is my3) then (z is mf9)
if (x is mx2) and (y is my3) then (z is mf2)
if (x is mx3) and (y is my3) then (z is mf3)
if (x is mx4) and (y is my3) then (z is mf9)
if (x is mx5) and (y is my3) then (z is mf3)
if (x is mx6) and (y is my3) then (z is mf2)
if (x is mx1) and (y is my4) then (z is mf8)
if (x is mx2) and (y is my4) then (z is mf5)
if (x is mx3) and (y is my4) then (z is mf1)
if (x is mx4) and (y is my4) then (z is mf8)
if (x is mx5) and (y is my4) then (z is mf7)
if (x is mx6) and (y is my4) then (z is mf1)
if (x is mx1) and (y is my5) then (z is mf6)
if (x is mx2) and (y is my5) then (z is mf8)
if (x is mx3) and (y is my5) then (z is mf1)
if (x is mx4) and (y is my5) then (z is mf5)
if (x is mx5) and (y is my5) then (z is mf9)
if (x is mx6) and (y is my5) then (z is mf2)
if (x is mx1) and (y is my6) then (z is mf3)
if (x is mx2) and (y is my6) then (z is mf9)
if (x is mx3) and (y is my6) then (z is mf4)
if (x is mx4) and (y is my6) then (z is mf2)
if (x is mx5) and (y is my6) then (z is mf9)
if (x is mx6) and (y is my6) then (z is mf5)
```

Тепер змоделюємо функцію використовуючи словник rules та функцію best_value з попереднього пункту:

```
output = []

for x in x_range:
    best_x = best_value(x, x_trimf_middle, 3)
    best_y = best_value(np.sin((x-2)/2) * x * np.sin(x-3), y_trimf_middle, 0.5)
    func = rules[(best_x, best_y)]
    output.append(z_trimf_middle[func])

plt.plot(x_range, output, label="Змодельована функція")
plt.title("Змодельована функція")
plt.legend()
plt.show()

plt.plot(x_range, z_func, label="Справжня функція")
plt.plot(x_range, output, label="Змодельована функція")
plt.title("Справжня і змодельована функції")
plt.legend()
plt.show()
```

[169] ✓ 0.2s Python


Змодельована функція
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
Змодельована функція
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


Справжня і змодельована функції
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
Справжня функція
Змодельована функція
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

Тепер обрахуємо похибки: середньоквадратичну і середню абсолютну:

Тепер обрахуємо похибки: середньоквадратичну і середню абсолютну.

Для спрощення задачі використаємо методи бібліотеки `sklearn.metrics`.

```
print("Середньоквадратична похибка: ", mean_squared_error(z_func, output))
print("Середня абсолютна похибка: ", mean_absolute_error(z_func, output))
```

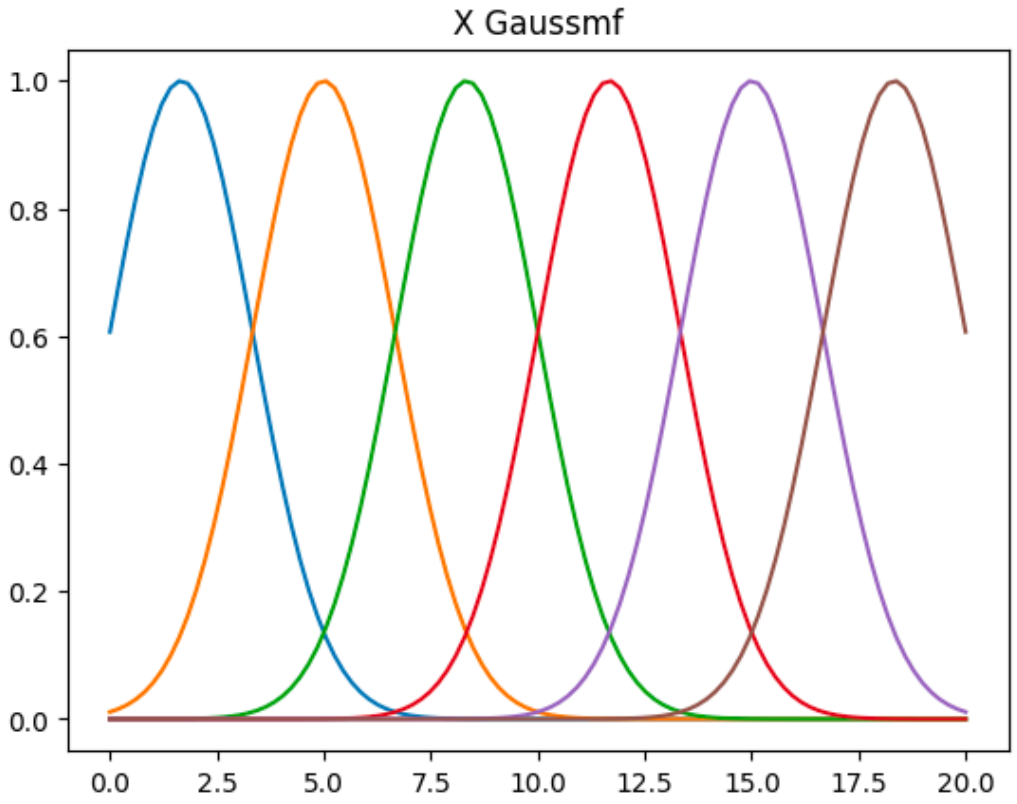
[73] ✓ 0.0s Python

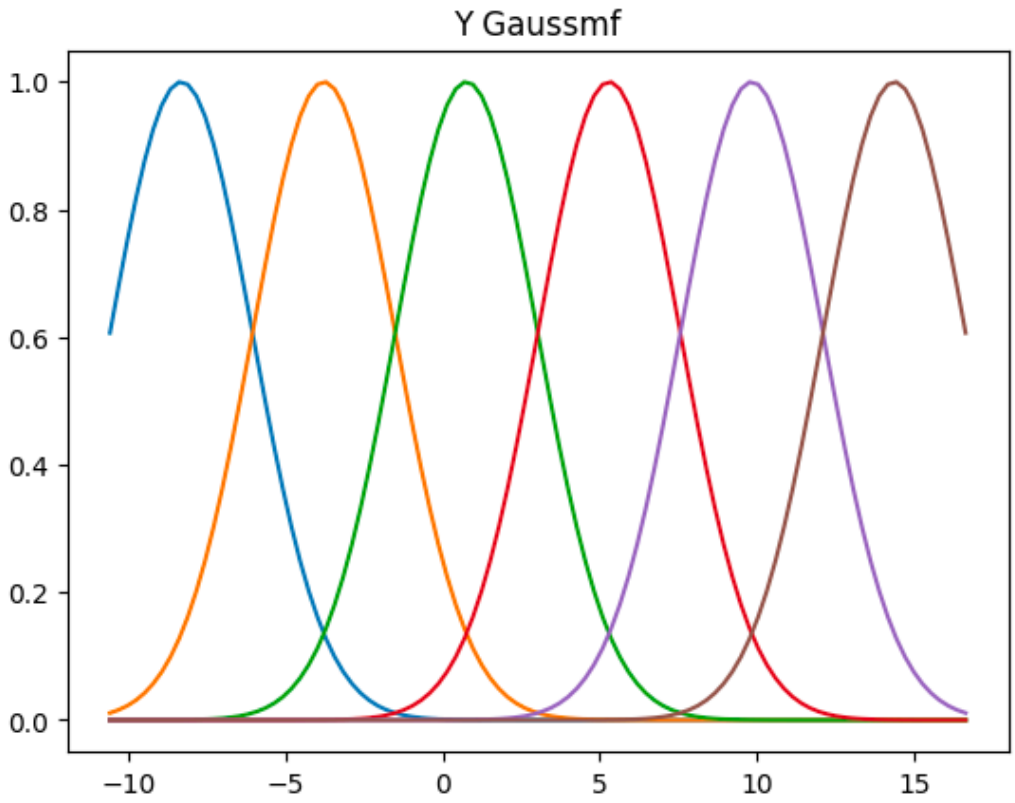
```
Середньоквадратична похибка:  0.8685168170790353
Середня абсолютна похибка:  0.7205464082298682
```

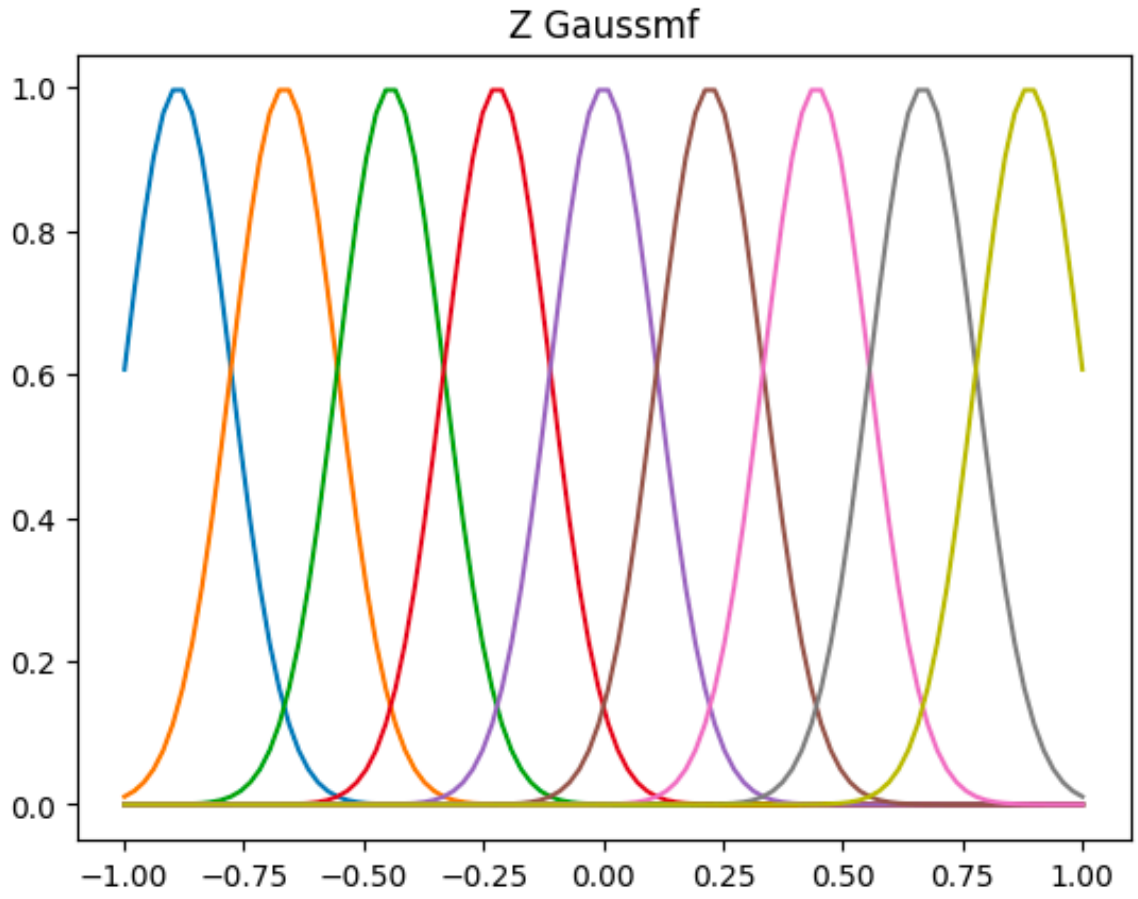
Повторимо ті ж кроки для функції приналежності Гауса:

```
sigma_x = (max(x_range) - min(x_range))/12
graph_x = [fuzz.gaussmf(x_range, x_intervals[i][1], sigma_x) for i in range(6)]
show_graphs(x_range, graph_x, "X Gaussmf")

sigma_y = (max(y_func) - min(y_func))/12
graph_y = [fuzz.gaussmf(y_space, y_intervals[i][1], sigma_y) for i in range(6)]
show_graphs(y_space, graph_y, "Y Gaussmf")

sigma_z = (max(z_func) - min(z_func))/18
graph_z = [fuzz.gaussmf(z_space, z_intervals[i][1], sigma_z) for i in range(9)]
show_graphs(z_space, graph_z, "Z Gaussmf")
```

[178] ✓ 0.2s Python



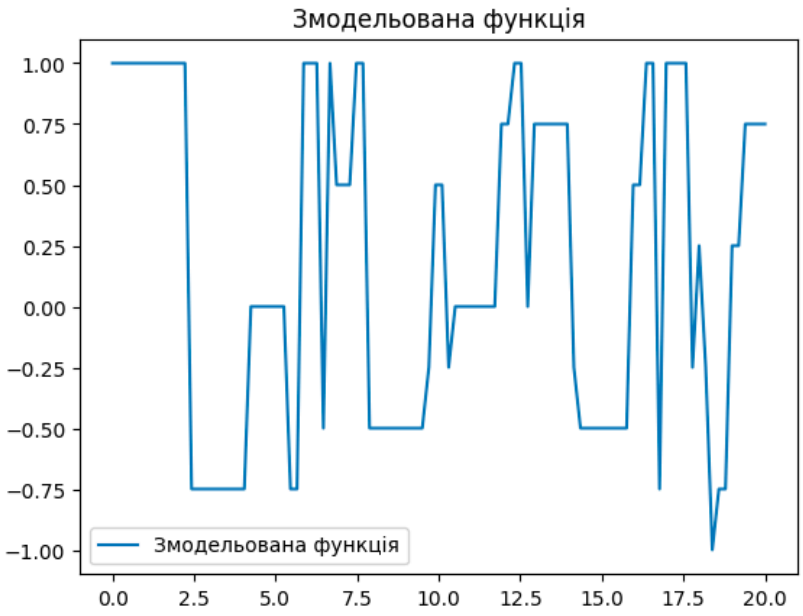

Y Gaussmf
1.0
0.8
0.6
0.4
0.2
0.0
−10
−5
0
5
10
15

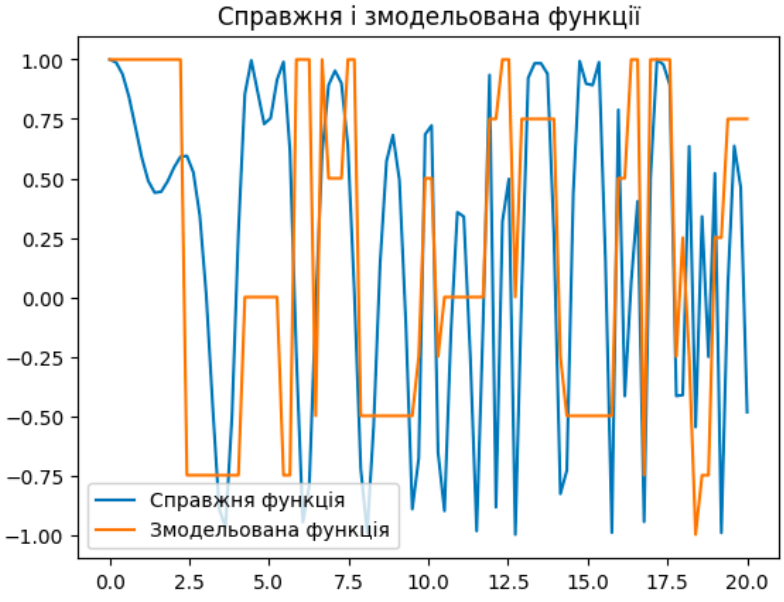

Z Gaussmf
1.0
0.8
0.6
0.4
0.2
0.0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

Змоделюємо функцію:

```python
output = []

for x in x_range:
    best_x = best_value(x, x_trimf_middle, sigma_x)
    best_y = best_value(np.sin((x-2)/2) * x * np.sin(x-3), y_trimf_middle, sigma_y)
    func = rules[(best_x, best_y)]
    output.append(z_trimf_middle[func])

plt.plot(x_range, output, label="Змодельована функція")
plt.title("Змодельована функція")
plt.legend()
plt.show()

plt.plot(x_range, z_func, label="Справжня функція")
plt.plot(x_range, output, label="Змодельована функція")
plt.title("Справжня і змодельована функції")
plt.legend()
plt.show()
```

✓ 0.1s Python





Обрахуємо похибки:

```
Обрахуємо похибки:

    print("Середньоквадратична похибка: ", mean_squared_error(z_func, output))
    print("Середня абсолютна похибка: ", mean_absolute_error(z_func, output))
[213]  ✓ 0.0s

...  Середньоквадратична похибка:  0.5985458381967094
     Середня абсолютна похибка:  0.6256143790242386
```

Як бачимо, похибки навіть нижчі. Що означає, що модель при функції приналежності Гауса працює навіть краще.

3. Проведемо експеримент:

Зменшимо кількість правил залишивши тільки ті, що розміщені на діагоналі таблиці:

```
output = []

rules2 = dict((key, value) for key, value in rules.items() if key[0] == key[1])

for x in x_range:
    best_x = best_value(x, x_trimf_middle, sigma_x)
    best_y = best_value(np.sin((x-2)/2) * x * np.sin(x-3), y_trimf_middle, sigma_y)
    try:
        func = rules2[(best_x, best_y)]
        output.append(z_trimf_middle[func])
    except:
        continue

# plt.plot(x_range, output, label="Змодельована функція")
# plt.title("Змодельована функція")
# plt.legend()
# plt.show()

# plt.plot(x_range, z_func, label="Справжня функція")
# plt.plot(x_range, output, label="Змодельована функція")
# plt.title("Справжня і змодельована функції")
# plt.legend()
# plt.show()
 ✓ 0.0s

Обрахуємо похибки:

    print("Середньоквадратична похибка: ", mean_squared_error(z_func[0:21], output))
    print("Середня абсолютна похибка: ", mean_absolute_error(z_func[0:21], output))
 ✓ 0.0s

Середньоквадратична похибка:  1.1740897183190424
Середня абсолютна похибка:  0.9489317998426705
```

Як бачимо, вище описані дії призвели до підвищення значення похибки.

Висновок:

Отже, під час виконання даної лабораторної роботи було змодельовано функцію двох змінних з використанням засобів нечіткої логіки. Крім цього, було проведено дослідження форми функції приналежності на якість моделювання. Була проаналізована та порівняна ефективність двох ФП: трикутної та Гауссівської. Функція приналежності Гаусса дала меншу похибку, що свідчить про вищу ефективність. Хоча, трикутна ФП також чудово впоралася із задачею. Також було проведено експеримент зі зменшення кількості правил, що призвів до підвищення значення похибки.