Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Ульяновский государственный технический университет»

Кафедра «Вычислительная техника»

**Системы искусственного интеллекта**

**Лабораторная работа №2**

**«Нечеткая логика»**

**Вариант №4**

Выполнил:

cтудент группы ИВТАСбд-41

Рубцов Денис Алексеевич

Проверил:

Хайруллин И. Д.

Ульяновск

2025

# **СОДЕРЖАНИЕ**

[**СОДЕРЖАНИЕ** 2](#_Toc212824160)

[**Постановка задачи** 3](#_Toc212824161)

[**Ход работы** 4](#_Toc212824162)

[**Тестирование** 6](#_Toc212824163)

[**Заключение** 8](#_Toc212824164)

[**Приложение А.** 9](#_Toc212824165)

## **Постановка задачи**

**Общее задание:**

Необходимо разработать программу на языке python, которая реализует предложенное вариантом задание. Предметную область можно выбрать из предложенного списка, либо выбрать свою.

Вариант 4: на языке Python разработайте скрипт, позволяющий выполнить операцию пересечения заданных пользователем нечетких множеств с трапециевидными функциями принадлежности. Входными данными будут параметры функций принадлежности и четкие объекты для каждого из множеств. Выходными – пересечение данных нечетких множеств.

Предметная область – авиация.

Высота полета: низкая, средняя, высокая

Скорость полета: медленная, нормальная, быстрая

## **Ход работы**

Первым делом в начале программы задаются универсумы (области определения) для входных переменных и нечёткие переменные (Рис.1)

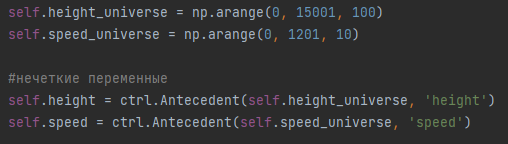


Рис. 1.

Для каждой переменной определены три нечётких множества с использованием трапециевидных функций принадлежности (Рис.2)

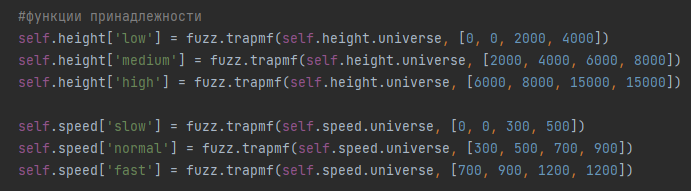


Рис. 2.

Это одна из самых часто используемых функций принадлежности в нечёткой логике. Она описывает плавный переход элемента из состояния «не принадлежит множеству» к состоянию «полностью принадлежит множеству» и обратно. Задается в виде следующей формулы (Рис.3)

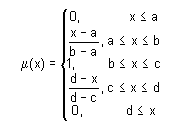


Рис. 3.

Для введённых пользователем значений высоты и скорости выполняется вычисление степени принадлежности каждого значения всем нечётким множествам. Результат выражается числом от 0 до 1 — степени принадлежности (Рис.4)

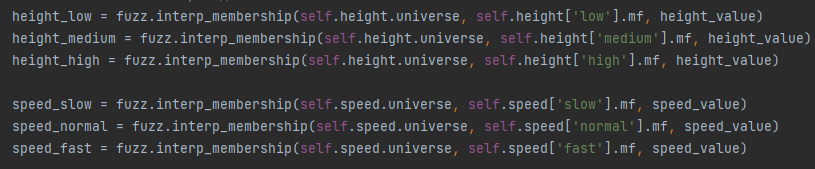


Рис. 4.

Далее для всех комбинаций термов (например, «низкая высота и высокая скорость») вычисляются степени пересечения с использованием операции минимума (Рис.5)

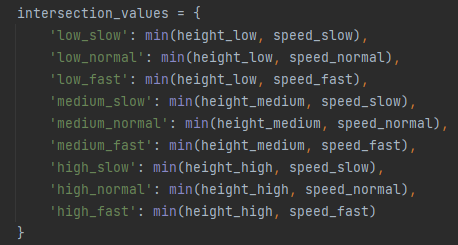


Рис. 5.

После вычисления всех пересечений программа выбирает комбинации, имеющие наибольшую степень пересечения.

Для лучшей комбинации строится график с наглядным положением четкого объекта на трапециевидных функциях принадлежности.

## **Тестирование**

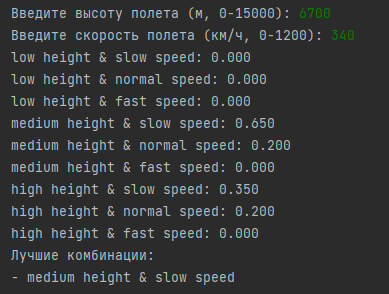


Рис. 6. Высота полета 6700, скорость 340.

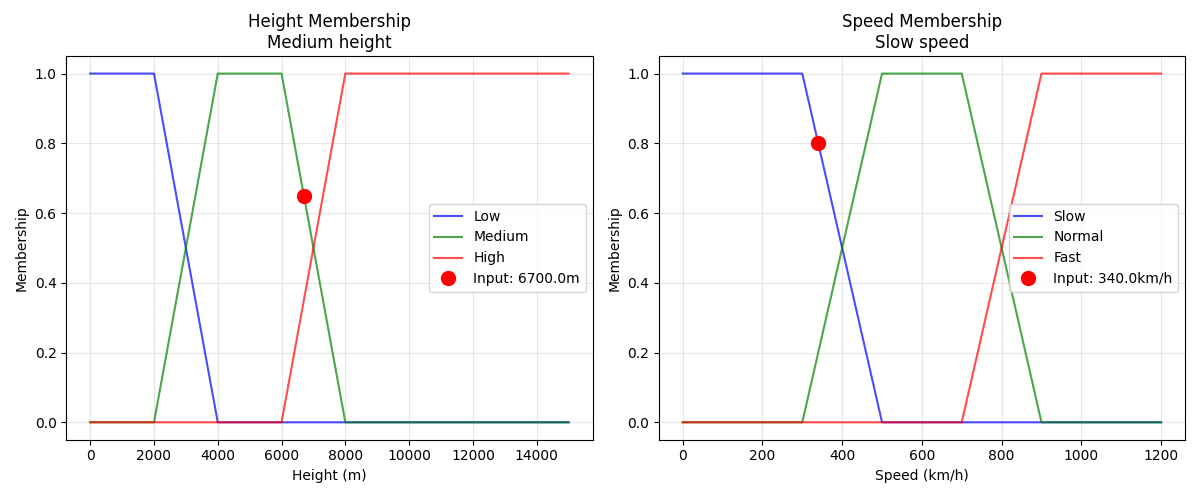


Рис. 7. График: высота полета 6700, скорость 340.

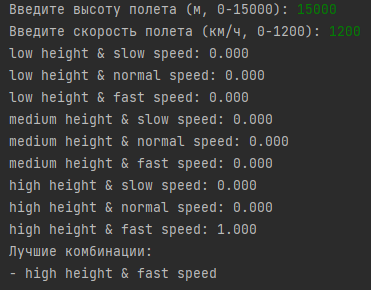


Рис. 8. Высота полета 15000, скорость 1200.

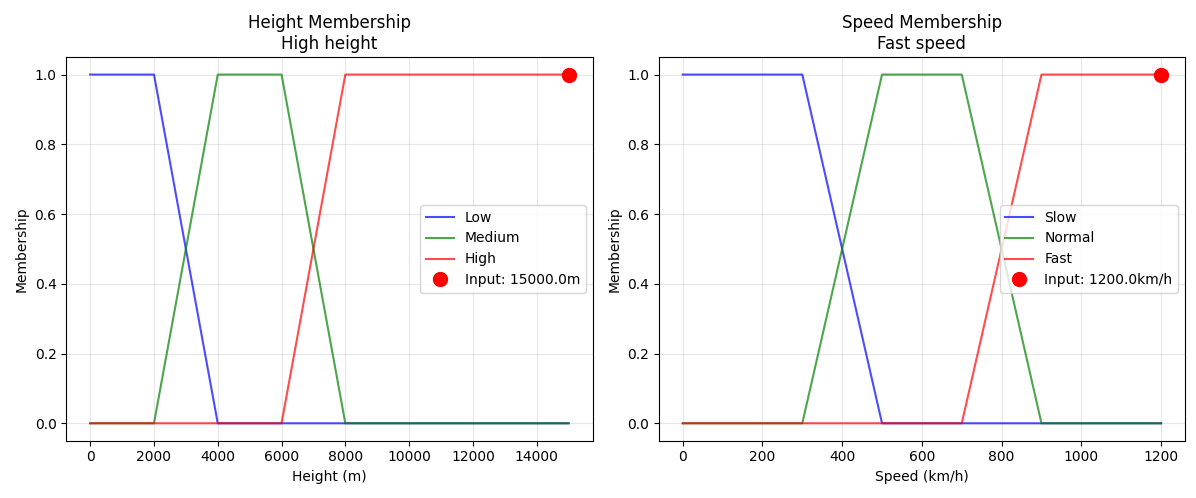


Рис. 9. График: высота полета 15000, скорость 1200.

# **Заключение**

В рамках выполнения лабораторной работы были приобретены умения и навыки в реализации нечеткой логики на языке программирования python.

Также была написана программа, реализующая все пункты задания

# **Приложение А.**

import numpy as np

import skfuzzy as fuzz

from skfuzzy import control as ctrl

import matplotlib.pyplot as plt

class AviationSystem:

def \_\_init\_\_(self):

self.height\_universe = np.arange(0, 15001, 100)

self.speed\_universe = np.arange(0, 1201, 10)

#нечеткие переменные

self.height = ctrl.Antecedent(self.height\_universe, 'height')

self.speed = ctrl.Antecedent(self.speed\_universe, 'speed')

#функции принадлежности

self.height['low'] = fuzz.trapmf(self.height.universe, [0, 0, 2000, 4000])

self.height['medium'] = fuzz.trapmf(self.height.universe, [2000, 4000, 6000, 8000])

self.height['high'] = fuzz.trapmf(self.height.universe, [6000, 8000, 15000, 15000])

self.speed['slow'] = fuzz.trapmf(self.speed.universe, [0, 0, 300, 500])

self.speed['normal'] = fuzz.trapmf(self.speed.universe, [300, 500, 700, 900])

self.speed['fast'] = fuzz.trapmf(self.speed.universe, [700, 900, 1200, 1200])

def calculate\_intersection(self, height\_value, speed\_value):

#вычисление степеней принадлежности

height\_low = fuzz.interp\_membership(self.height.universe, self.height['low'].mf, height\_value)

height\_medium = fuzz.interp\_membership(self.height.universe, self.height['medium'].mf, height\_value)

height\_high = fuzz.interp\_membership(self.height.universe, self.height['high'].mf, height\_value)

speed\_slow = fuzz.interp\_membership(self.speed.universe, self.speed['slow'].mf, speed\_value)

speed\_normal = fuzz.interp\_membership(self.speed.universe, self.speed['normal'].mf, speed\_value)

speed\_fast = fuzz.interp\_membership(self.speed.universe, self.speed['fast'].mf, speed\_value)

#операция пересечения (минимум) для всех комбинаций

intersection\_values = {

'low\_slow': min(height\_low, speed\_slow),

'low\_normal': min(height\_low, speed\_normal),

'low\_fast': min(height\_low, speed\_fast),

'medium\_slow': min(height\_medium, speed\_slow),

'medium\_normal': min(height\_medium, speed\_normal),

'medium\_fast': min(height\_medium, speed\_fast),

'high\_slow': min(height\_high, speed\_slow),

'high\_normal': min(height\_high, speed\_normal),

'high\_fast': min(height\_high, speed\_fast)

}

return intersection\_values

def get\_intersection\_result(self, height\_value, speed\_value):

results = self.calculate\_intersection(height\_value, speed\_value)

max\_intersection = max(results.values())

best\_combinations = [comb for comb, value in results.items() if value == max\_intersection]

return best\_combinations, results

def plot(self, height\_value, speed\_value, best\_combination):

height\_term, speed\_term = best\_combination.split('\_')

#график с двумя подграфиками

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 5))

#Высота

ax1.plot(self.height\_universe, self.height['low'].mf, 'b', linewidth=1.5, label='Low', alpha=0.7)

ax1.plot(self.height\_universe, self.height['medium'].mf, 'g', linewidth=1.5, label='Medium', alpha=0.7)

ax1.plot(self.height\_universe, self.height['high'].mf, 'r', linewidth=1.5, label='High', alpha=0.7)

#Точка для высоты

height\_membership = fuzz.interp\_membership(self.height.universe, self.height[height\_term].mf, height\_value)

ax1.plot(height\_value, height\_membership, 'ro', markersize=10, label=f'Input: {height\_value}m')

ax1.set\_title(f'Height Membership\n{height\_term.capitalize()} height')

ax1.set\_xlabel('Height (m)')

ax1.set\_ylabel('Membership')

ax1.legend()

ax1.grid(True, alpha=0.3)

#Скорость

ax2.plot(self.speed\_universe, self.speed['slow'].mf, 'b', linewidth=1.5, label='Slow', alpha=0.7)

ax2.plot(self.speed\_universe, self.speed['normal'].mf, 'g', linewidth=1.5, label='Normal', alpha=0.7)

ax2.plot(self.speed\_universe, self.speed['fast'].mf, 'r', linewidth=1.5, label='Fast', alpha=0.7)

# Точка для скорости

speed\_membership = fuzz.interp\_membership(self.speed.universe, self.speed[speed\_term].mf, speed\_value)

ax2.plot(speed\_value, speed\_membership, 'ro', markersize=10, label=f'Input: {speed\_value}km/h')

ax2.set\_title(f'Speed Membership\n{speed\_term.capitalize()} speed')

ax2.set\_xlabel('Speed (km/h)')

ax2.set\_ylabel('Membership')

ax2.legend()

ax2.grid(True, alpha=0.3)

plt.tight\_layout()

plt.show()

def main():

aviation\_system = AviationSystem()

try:

height\_val = float(input("Введите высоту полета (м, 0-15000): "))

speed\_val = float(input("Введите скорость полета (км/ч, 0-1200): "))

best\_combinations, all\_results = aviation\_system.get\_intersection\_result(height\_val,speed\_val)

for combination, value in all\_results.items():

height\_term, speed\_term = combination.split('\_')

print(f"{height\_term} height & {speed\_term} speed: {value:.3f}")

print("Лучшие комбинации:")

for comb in best\_combinations:

height\_term, speed\_term = comb.split('\_')

print(f"- {height\_term} height & {speed\_term} speed")

if best\_combinations:

aviation\_system.plot(height\_val, speed\_val, best\_combinations[0])

except ValueError:

print("Ошибка: введите корректные числовые значения.")

except Exception as e:

print(f"Произошла ошибка: {e}")

if \_\_name\_\_ == "\_\_main\_\_":

main()