# Отчет по лабораторной работе

Выполнила: Годорожа Оксана

Группа: I2302

## Тема:

Визуализация данных о риске ожога с помощью трехмерных графиков на Python.

## Цель работы:

Целью данной лабораторной работы является освоение построения и анимации трёхмерных графиков с помощью библиотек matplotlib и numpy. Также требуется визуализировать изменение риска ожога в зависимости от температуры и времени.

## Введение:

Визуализация данных является важным инструментом анализа, позволяющим лучше понять закономерности и зависимости в наборе данных. В данной работе демонстрируется зависимость риска ожога от температуры и времени с помощью трёхмерного графика. Для реализации задачи использовались библиотеки Python: NumPy (для вычислений и создания сетки данных) и Matplotlib (для визуализации).

## Ход работы:

1. Импортированы необходимые библиотеки: numpy, matplotlib и mpl\_toolkits.mplot3d.  
2. С помощью numpy создана двумерная сетка данных, моделирующая диапазон температур (20–100°C) и времени (0–10 секунд).  
3. Определена функция burn\_risk(T, t), вычисляющая вероятность ожога на основе температуры и времени.  
4. Сформирована 3D-поверхность с использованием метода plot\_surface(), где оси обозначают:  
 • X — температура (°C)  
 • Y — время (с)  
 • Z — риск ожога  
5. Добавлена функция анимации, изменяющая угол обзора графика во времени, чтобы лучше продемонстрировать объёмную зависимость.

## Код программы:

import numpy as np

import matplotlib.pyplot as plt

from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg

from mpl\_toolkits.mplot3d import Axes3D

import tkinter as tk

class FuzzyBurnRiskSystem:

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

self.temp\_range = np.linspace(0, 45, 100)

self.uv\_range = np.linspace(0, 12, 100)

self.humidity\_range = np.linspace(0, 100, 100)

self.risk\_range = np.linspace(0, 100, 100)

# --- Температура ---

def temp\_low(self, t):

if t <= 15: return 1

elif t >= 25: return 0

else: return (25 - t) / 10

def temp\_medium(self, t):

if t <= 15 or t >= 35: return 0

elif 15 < t <= 25: return (t - 15) / 10

else: return (35 - t) / 10

def temp\_high(self, t):

if t <= 25: return 0

elif t >= 35: return 1

else: return (t - 25) / 10

# --- UV ---

def uv\_low(self, u):

if u <= 2: return 1

elif u >= 5: return 0

else: return (5 - u) / 3

def uv\_moderate(self, u):

if u <= 2 or u >= 7: return 0

elif 2 < u <= 5: return (u - 2) / 3

else: return (7 - u) / 2

def uv\_high(self, u):

if u <= 5 or u >= 10: return 0

elif 5 < u <= 7: return (u - 5) / 2

else: return (10 - u) / 3

def uv\_extreme(self, u):

if u <= 7: return 0

elif u >= 10: return 1

else: return (u - 7) / 3

# --- Влажность ---

def humidity\_low(self, h):

if h <= 30: return 1

elif h >= 60: return 0

else: return (60 - h) / 30

def humidity\_high(self, h):

if h <= 30: return 0

elif h >= 60: return 1

else: return (h - 30) / 30

# --- Расчёт риска ---

def calculate\_risk(self, temperature, uv\_index, humidity):

t\_low = self.temp\_low(temperature)

t\_med = self.temp\_medium(temperature)

t\_high = self.temp\_high(temperature)

u\_low = self.uv\_low(uv\_index)

u\_mod = self.uv\_moderate(uv\_index)

u\_high = self.uv\_high(uv\_index)

u\_ext = self.uv\_extreme(uv\_index)

h\_low = self.humidity\_low(humidity)

h\_high = self.humidity\_high(humidity)

rules = [

(min(u\_high, t\_high), 90),

(min(u\_ext, t\_high), 98),

(min(u\_mod, h\_high), 50),

(min(u\_low, t\_low), 15),

(min(u\_high, h\_low), 90),

(min(u\_ext, t\_med), 85),

(min(u\_mod, t\_low), 30),

(min(u\_low, h\_high), 20),

(min(u\_mod, t\_med, h\_low), 55),

(min(u\_high, t\_med), 75),

(min(u\_ext, h\_high), 80),

(min(u\_low, t\_high), 35)

]

weighted = sum((s \*\* 2) \* v for s, v in rules)

total = sum((s \*\* 2) for s, \_ in rules)

risk = weighted / total if total > 0 else 0

strong = max(s for s, \_ in rules)

max\_risk = max(v for \_, v in rules)

risk = max(risk, strong \* max\_risk \* 0.7)

return risk

def get\_recommendation(self, risk):

if risk < 30:

return ("НИЗКИЙ", "#4caf50", "Риск минимален. Крем SPF 15+.")

elif risk < 60:

return ("СРЕДНИЙ", "#ff9800", "Умеренный риск. Крем SPF 30+, избегайте солнца в полдень.")

else:

return ("ВЫСОКИЙ", "#f44336", "⚠️ Высокий риск! SPF 50+, избегайте солнца 10:00–16:00.")

class BurnRiskGUI:

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

self.root = root

self.root.title("☀️ Прогноз риска солнечного ожога")

self.root.geometry("1400x900")

self.system = FuzzyBurnRiskSystem()

self.temperature = tk.DoubleVar(value=25)

self.uv = tk.DoubleVar(value=5)

self.humidity = tk.DoubleVar(value=50)

self.create\_ui()

self.update\_all()

def create\_ui(self):

header = tk.Label(self.root, text="Прогнозирование риска солнечного ожога",

font=("Arial", 20, "bold"), bg="#ff9800", fg="white")

header.pack(fill=tk.X, pady=10)

main = tk.Frame(self.root)

main.pack(fill=tk.BOTH, expand=True)

left = tk.Frame(main, width=300)

left.pack(side=tk.LEFT, fill=tk.Y, padx=10)

right = tk.Frame(main)

right.pack(side=tk.RIGHT, fill=tk.BOTH, expand=True)

# --- Ползунки ---

tk.Label(left, text="🌡 Температура (°C)", font=("Arial", 10, "bold")).pack(pady=(10, 0))

tk.Scale(left, from\_=0, to=45, variable=self.temperature, orient=tk.HORIZONTAL, command=self.on\_change).pack()

tk.Label(left, text="☀️ UV-индекс", font=("Arial", 10, "bold")).pack(pady=(10, 0))

tk.Scale(left, from\_=0, to=12, resolution=0.1, variable=self.uv, orient=tk.HORIZONTAL, command=self.on\_change).pack()

tk.Label(left, text="💧 Влажность (%)", font=("Arial", 10, "bold")).pack(pady=(10, 0))

tk.Scale(left, from\_=0, to=100, variable=self.humidity, orient=tk.HORIZONTAL, command=self.on\_change).pack()

# --- Результаты ---

self.result\_label = tk.Label(left, text="", font=("Arial", 14, "bold"), wraplength=250)

self.result\_label.pack(pady=20)

# --- Графики ---

self.fig = plt.Figure(figsize=(11, 7))

self.ax\_t = self.fig.add\_subplot(2, 2, 1)

self.ax\_u = self.fig.add\_subplot(2, 2, 2)

self.ax\_h = self.fig.add\_subplot(2, 2, 3)

self.ax\_3d = self.fig.add\_subplot(2, 2, 4, projection="3d")

self.canvas = FigureCanvasTkAgg(self.fig, master=right)

self.canvas.get\_tk\_widget().pack(fill=tk.BOTH, expand=True)

def on\_change(self, event=None):

self.update\_all()

def update\_all(self):

t = self.temperature.get()

u = self.uv.get()

h = self.humidity.get()

risk = self.system.calculate\_risk(t, u, h)

level, color, msg = self.system.get\_recommendation(risk)

self.result\_label.config(

text=f"Риск: {risk:.1f}%\nУровень: {level}\n\n{msg}", fg=color

)

self.plot\_temp(t)

self.plot\_uv(u)

self.plot\_humidity(h)

self.plot\_3d(h)

self.canvas.draw()

def plot\_temp(self, val):

x = self.system.temp\_range

self.ax\_t.clear()

self.ax\_t.plot(x, [self.system.temp\_low(v) for v in x], 'g', label="Низкая")

self.ax\_t.plot(x, [self.system.temp\_medium(v) for v in x], 'orange', label="Средняя")

self.ax\_t.plot(x, [self.system.temp\_high(v) for v in x], 'r', label="Высокая")

self.ax\_t.axvline(val, color='black', linestyle='--')

self.ax\_t.set\_title("Температура (°C)")

self.ax\_t.legend()

def plot\_uv(self, val):

x = self.system.uv\_range

self.ax\_u.clear()

self.ax\_u.plot(x, [self.system.uv\_low(v) for v in x], 'g', label="Низкий")

self.ax\_u.plot(x, [self.system.uv\_moderate(v) for v in x], 'orange', label="Умеренный")

self.ax\_u.plot(x, [self.system.uv\_high(v) for v in x], 'r', label="Высокий")

self.ax\_u.plot(x, [self.system.uv\_extreme(v) for v in x], 'purple', label="Экстремальный")

self.ax\_u.axvline(val, color='black', linestyle='--')

self.ax\_u.set\_title("UV-индекс")

self.ax\_u.legend()

def plot\_humidity(self, val):

x = self.system.humidity\_range

self.ax\_h.clear()

self.ax\_h.plot(x, [self.system.humidity\_low(v) for v in x], 'g', label="Низкая")

self.ax\_h.plot(x, [self.system.humidity\_high(v) for v in x], 'b', label="Высокая")

self.ax\_h.axvline(val, color='black', linestyle='--')

self.ax\_h.set\_title("Влажность (%)")

self.ax\_h.legend()

def plot\_3d(self, humidity):

self.ax\_3d.clear()

t\_vals = np.linspace(10, 40, 30)

u\_vals = np.linspace(0, 12, 30)

T, U = np.meshgrid(t\_vals, u\_vals)

R = np.zeros\_like(T)

for i in range(T.shape[0]):

for j in range(T.shape[1]):

R[i, j] = self.system.calculate\_risk(T[i, j], U[i, j], humidity)

surf = self.ax\_3d.plot\_surface(T, U, R, cmap='inferno', alpha=0.8)

self.ax\_3d.set\_title("3D поверхность риска (зависит от влажности)")

self.ax\_3d.set\_xlabel("Температура (°C)")

self.ax\_3d.set\_ylabel("UV-индекс")

self.ax\_3d.set\_zlabel("Риск (%)")

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

root = tk.Tk()

app = BurnRiskGUI(root)

root.mainloop()

## Пояснение к коду:

• Функция np.linspace() создаёт равномерно распределённые значения температур и времени.  
• np.meshgrid() формирует двумерную сетку, которая позволяет рассчитать Z для каждой пары (T, t).  
• Функция burn\_risk() моделирует уменьшение риска ожога при удалении от критической температуры (60°C) и с течением времени.  
• Метод plot\_surface() визуализирует трёхмерную поверхность с цветовой схемой 'plasma'.  
• Функция update() отвечает за анимацию вращения, изменяя угол обзора графика.  
• FuncAnimation() создаёт плавную анимацию вращения, чтобы визуализировать поверхность с разных углов.

## Результат работы программы:

После запуска программы отображается трёхмерная поверхность, где яркость и цвет отражают уровень риска ожога. График динамически вращается, что даёт возможность наблюдать зависимость под разными углами. Максимальный риск ожога наблюдается при температуре около 60°C и в начале временного интервала.

## Вывод:

В ходе лабораторной работы была изучена техника построения и анимации трёхмерных графиков с помощью библиотеки Matplotlib. Созданная визуализация помогает наглядно проанализировать зависимость риска ожога от температуры и времени. Использование анимации позволяет более наглядно представить форму и поведение поверхности.