МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КУБГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**Отчет**

**по лабораторной работе №2 по курсу**

**«МЕТОДЫ ПОИСКОВОЙ ОПТИМИЗАЦИИ»**

Работу выполнили

Студены 36 группы

Воробьев А.Д.

Андреев А.П.

Преподаватель:

Полупанова Е.Е

Краснодар 2024

**Цель работы:** изучить задачи квадратичного программирования. Решить задачу КП на примере функции.

**Ход работы:** для решение задачи КП используются необходимые условия Куна – Таккера и использование частных производных.

Задача квадратичного программирования является оптимизационной задачей, в которой требуется найти значения переменных, минимизирующие квадратичную целевую функцию при наличии линейных и квадратичных ограничений. Распишем алгоритм решения задачи КП по шагам:

1. **Формулировка задачи:**

Задать квадратичную целевую функцию вида:

где x - вектор переменных, Q - матрица квадратичных коэффициентов, c - вектор линейных коэффициентов.

Определить ограничения, если они есть, в виде линейных и квадратичных равенств и/или неравенств.

1. **Подготовка задачи:**

Проверить матрицу на симметричность и положительную (или отрицательную) определенность. Если не является симметричной и положительно определенной, задачу КП можно сделать симметричной путем взятия средней матрицы и.

Проверить наличие ограничений и привести их к стандартной форме КП, если это необходимо.

1. **Формирование Лагранжиана:**

Ввести множители Лагранжа для каждого ограничения.

Сформировать Лагранжиан задачи:

Где λ - вектор множителей Лагранжа, A - матрица коэффициентов ограничений, b - вектор правых частей ограничений.

1. **Нахождение стационарных точек:**

Найти производные Лагранжиана по переменным x и приравнять их к нулю:

Это уравнение называется условием стационарности. Решение этой системы уравнений даст значения переменных x в стационарной точке.

1. **Проверка условий оптимальности:**

Проверить, что полученное решение удовлетворяет ограничениям:

Проверить условие ККТ (условия Каруша-Куна-Таккера), включая неотрицательность множителей Лагранжа:

1. **Вычисление значения целевой функции:**

Вычислить значение квадратичной целевой функции в найденной оптимальной точке:

1. **Интерпретация результата:**

Полученное значение x является оптимальным решением задачи КП, а f(x) - минимальным значением целевой функции при заданных ограничениях.

Для создания программы используется язык программирования Python 3.11 и среда разработки Intellij Idea Ultimate. Для графической визуализации используется графический фреймворк Tkinter и Matplotlib.

В созданной программе одно главное активное окно.

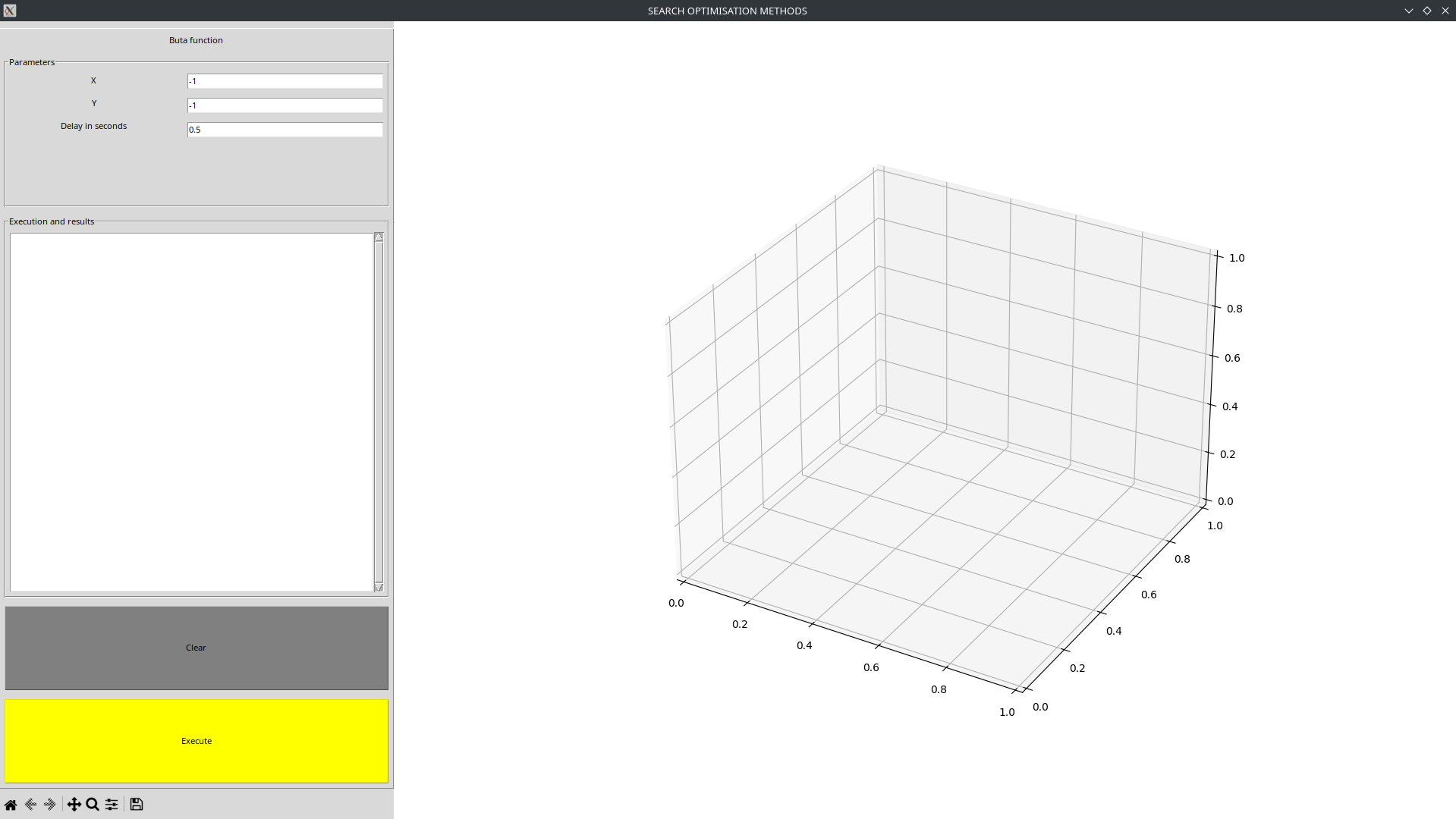


Рисунок 1 – Главное окно программы.

В данном окне можно выбрать необходимую лабораторную работу, в данном случае «2», ввести начальную точку метода градиентного спуска с координатами «X» и «Y», задать начальный шаг алгоритма, число итераций (шагов) алгоритма, и задержку между этими итерациями в секундах (используется для наглядного представления работы алгоритма).

В поле «Выполнение и результаты» динамически выводятся результаты работы алгоритма в виде шагов, представленных координатами и значениями оптимизируемой функции в этих координатах. На рисунке 2 показано поле «Выполнение и результаты» до запуска программы.

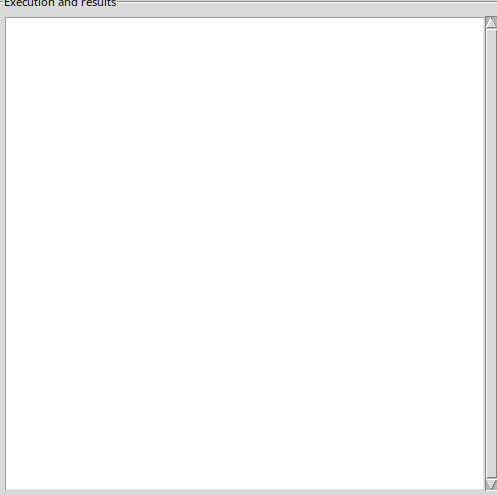


Рисунок 2 – Выполнение и результаты.

При нажатии на кнопку “Выполнить” на главном окне программы отображается необходимая нам функция, как показано на рисунке 4. Причем, более высокие значения функции показана ярко желтым цветом, а самые низкие темно серым цветом.

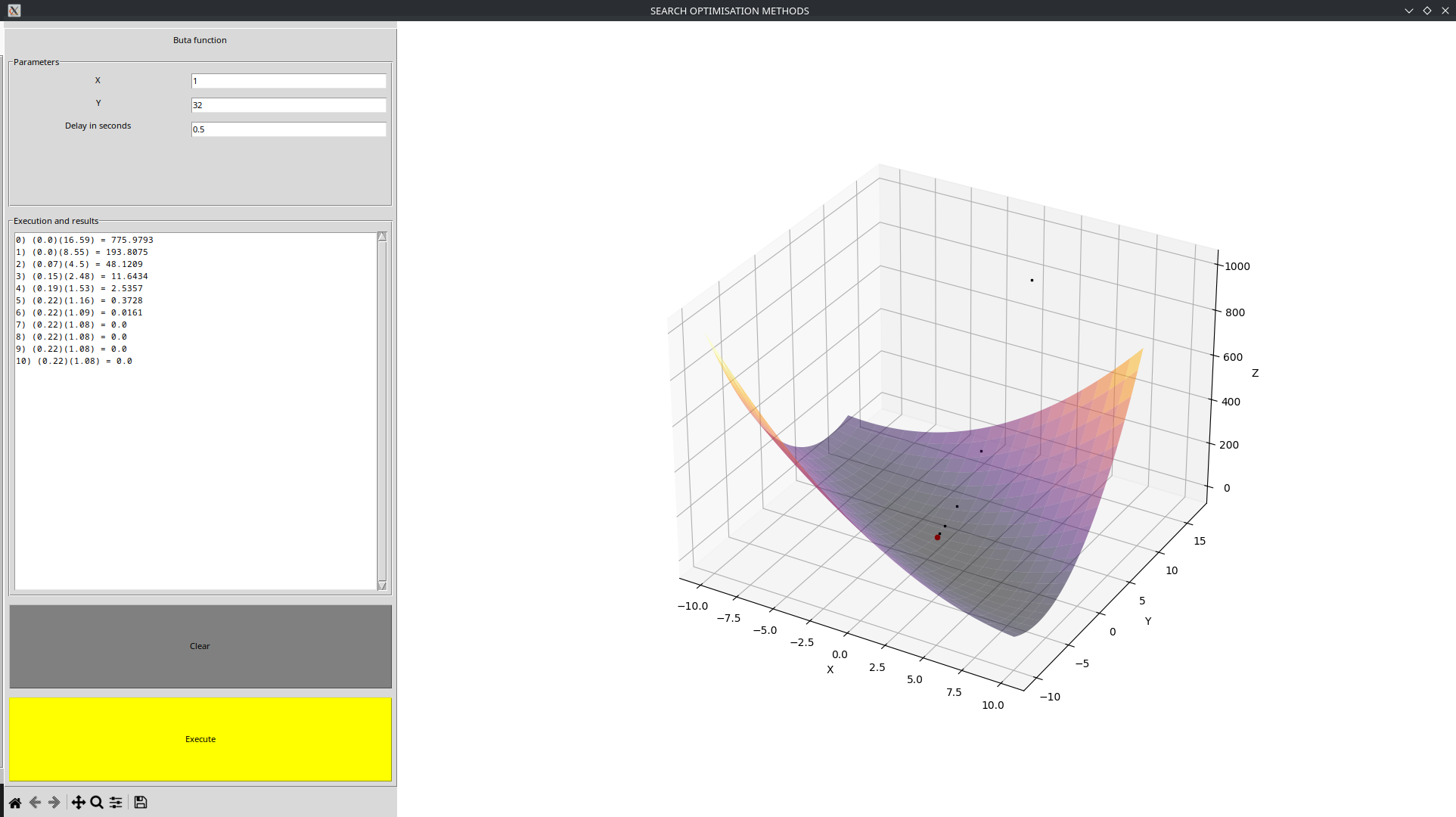


Рисунок 4 – Функция 2x₁² + 3x₂² + 4x₁x₂ - 6x₁ - 3x₂.

Для запуска алгоритма необходимо нажать кнопку «Выполнить», при этом поле «Выполнение и результаты» начнёт динамически заполнятся, а на отображаемой функции можно наглядно увидеть функционирование алгоритма - искомые точки. При полном выполнении программа выводит окно, уведомляющие нас об окончании работы алгоритма.

**Вывод:** в ходе работы были изучены задачи квадратичного программирования, реализован метод решения данных задач.

**Листинг программы:**

**Файл utils.py**

from scipy.optimize import minimize

import numpy as np

import numpy

def make\_data\_lab\_2():

x = numpy.linspace(-10, 10, 100)

y = numpy.linspace(-10, 10, 100)

x\_grid, y\_grid = numpy.meshgrid(x, y)

z = 2 \* x\_grid \* x\_grid + 3 \* y\_grid \* y\_grid + 4 \* x\_grid \* y\_grid - 6 \* x\_grid - 3 \* y\_grid

return x\_grid, y\_grid, z

def kp(x, y):

# global points

points = []

def fun(x\_i): # Функция

x1 = x\_i[0]

x2 = x\_i[1]

return 2 \* x1 \* x1 + 3 \* x2 \* x2 + 4 \* x1 \* x2 - 6 \* x1 - 3 \* x2

def callback(x\_w):

g\_list = np.ndarray.tolist(x\_w)

g\_list.append(fun(x\_w))

points.append(g\_list)

b = (0, float("inf")) # диапазон поиска

bounds = (b, b)

x0 = (x, y) # начальная точка

con = {'type': 'eq', 'fun': fun}

# основной вызов

res = minimize(fun, x0, method="SLSQP", bounds=bounds,

constraints=con, callback=callback)

glist = np.ndarray.tolist(res.x)

glist.append(res.fun)

points.append(glist)

for iteration, point in enumerate(points):

yield iteration, point

**Файл main.py**

import tkinter

import time

from tkinter import \*

from tkinter import scrolledtext, messagebox

from tkinter.ttk import Combobox, Notebook, Style

from matplotlib import pyplot as plt

from matplotlib.backends.backend\_tkagg import (FigureCanvasTkAgg, NavigationToolbar2Tk)

from lab2.utils import make\_data\_lab\_2, kp

def main():

window = Tk()

width = window.winfo\_screenwidth()

height = window.winfo\_screenheight()

window.geometry("%dx%d" % (width, height))

window.title("SEARCH OPTIMISATION METHODS")

fig = plt.figure(figsize=(14, 14))

fig.add\_subplot(projection='3d')

canvas = FigureCanvasTkAgg(fig, master=window)

canvas.draw()

canvas.get\_tk\_widget().pack(side=tkinter.RIGHT, fill=tkinter.BOTH)

toolbar = NavigationToolbar2Tk(canvas, window)

toolbar.update()

canvas.get\_tk\_widget().pack(side=tkinter.RIGHT, fill=tkinter.BOTH)

style = Style()

style.theme\_create("dummy", parent="alt", settings={

"TNotebook": {"configure": {"tabmargins": [2, 5, 2, 0]}},

"TNotebook.Tab": {

"configure": {"padding": [5, 1], "background": "#808080"},

"map": {"background": [("selected", "FFFF00")],

"expand": [("selected", [1, 1, 1, 0])]}}})

style.theme\_use("dummy")

tab\_control = Notebook(window)

def draw\_lab\_2():

fig.clf()

x, y, z = make\_data\_lab\_2()

res\_x = txt\_1\_tab\_1.get()

res\_y = txt\_2\_tab\_1.get()

ax = fig.add\_subplot(projection='3d')

ax.plot\_surface(x, y, z, rstride=5, cstride=5, alpha=0.5, cmap="inferno")

canvas.draw()

x\_cs = []

y\_cs = []

z\_cs = []

for i, point in kp(res\_x, res\_y):

x\_cs.append(point[0])

y\_cs.append(point[1])

z\_cs.append(point[2])

for i in range(len(x\_cs)):

if i < (len(x\_cs) - 1):

ax.scatter(x\_cs[i - 1], y\_cs[i - 1], z\_cs[i - 1], c="black", s=1, marker="s")

else:

ax.scatter(x\_cs[i - 1], y\_cs[i - 1], z\_cs[i - 1], c="red")

txt\_tab\_1.insert(INSERT, f"{i}) ({round(x\_cs[i], 2)})({round(y\_cs[i], 2)}) = {round(z\_cs[i], 4)}\n")

canvas.draw()

ax.set\_xlabel('X')

ax.set\_ylabel('Y')

ax.set\_zlabel('Z')

window.update()

delay = txt\_3\_tab\_1.get()

time.sleep(float(delay))

messagebox.showinfo('Уведомление', 'Готово')

def delete\_lab\_2():

txt\_tab\_1.delete(1.0, END)

tab\_1 = Frame(tab\_control)

tab\_control.add(tab\_1, text="LAB\_1")

main\_f\_tab\_1 = LabelFrame(tab\_1, text="Parameters")

left\_f\_tab\_1 = Frame(main\_f\_tab\_1)

right\_f\_tab\_1 = Frame(main\_f\_tab\_1)

txt\_f\_tab\_1 = LabelFrame(tab\_1, text="Execution and results")

lbl\_1\_tab\_1 = Label(left\_f\_tab\_1, text="X")

lbl\_2\_tab\_1 = Label(left\_f\_tab\_1, text="Y")

lbl\_3\_tab\_1 = Label(left\_f\_tab\_1, text="Delay in seconds")

lbl\_5\_tab\_1 = Label(tab\_1, text="Buta function")

txt\_1\_tab\_1 = Entry(right\_f\_tab\_1)

txt\_1\_tab\_1.insert(0, "-1")

txt\_2\_tab\_1 = Entry(right\_f\_tab\_1)

txt\_2\_tab\_1.insert(0, "-1")

txt\_3\_tab\_1 = Entry(right\_f\_tab\_1)

txt\_3\_tab\_1.insert(0, "0.5")

txt\_tab\_1 = scrolledtext.ScrolledText(txt\_f\_tab\_1)

btn\_del\_tab\_1 = Button(tab\_1, text="Clear", command=delete\_lab\_2, foreground="black", background="grey")

btn\_tab\_1 = Button(tab\_1, text="Execute", foreground="black", background="yellow", command=draw\_lab\_2)

lbl\_5\_tab\_1.pack(side=TOP, padx=5, pady=5)

main\_f\_tab\_1.pack(side=TOP, padx=5, pady=5, fill=BOTH, expand=True)

left\_f\_tab\_1.pack(side=LEFT, fill=BOTH, expand=True)

right\_f\_tab\_1.pack(side=RIGHT, fill=BOTH, expand=True)

lbl\_1\_tab\_1.pack(side=TOP, padx=5, pady=5, fill=BOTH)

lbl\_2\_tab\_1.pack(side=TOP, padx=5, pady=5, fill=BOTH)

lbl\_3\_tab\_1.pack(side=TOP, padx=5, pady=5, fill=BOTH)

txt\_1\_tab\_1.pack(side=TOP, padx=5, pady=5, fill=BOTH)

txt\_2\_tab\_1.pack(side=TOP, padx=5, pady=5, fill=BOTH)

txt\_3\_tab\_1.pack(side=TOP, padx=5, pady=5, fill=BOTH)

txt\_tab\_1.pack(padx=5, pady=5, fill=BOTH, expand=True)

btn\_tab\_1.pack(side=BOTTOM, padx=5, pady=5, fill=BOTH, expand=True)

btn\_del\_tab\_1.pack(side=BOTTOM, padx=5, pady=5, fill=BOTH, expand=True)

txt\_f\_tab\_1.pack(side=BOTTOM, padx=5, pady=5, fill=BOTH, expand=True)

tab\_control.pack(side=RIGHT, fill=BOTH, expand=True)

window.mainloop()

if \_\_name\_\_ == '\_\_main\_\_':

main()