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

Студенты 46 группы

Стасюк К.В.

Баева Д.Н.

Преподаватель:

Нигодин Е.А.

Краснодар 2023

**Цель работы:** изучить задачи квадратичного программирования, а затем решить задачу КП на примере функции.

**Ход работы:** для реализации задачи КП используются условия Куна – Таккера и частные производные.

**Шаги алгоритма**

*Шаг 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.

*Шаг 2.* Проверить матрицу на симметричность и положительную (или отрицательную) определенность. Если не является симметричной и положительно определенной, то задачу КП можно сделать симметричной путем взятия средней матрицы и. Далее необходимо проверить наличие ограничений и привести их к стандартной форме КП, если это необходимо.

*Шаг 3.* **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.

*Шаг 4.* **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.

*Шаг 5.* **Проверка условий оптимальности.** Проверить, что полученное решение удовлетворяет ограничениям: . Затем проверить условие ККТ (условия Каруша-Куна-Таккера), включая неотрицательность множителей Лагранжа: .

*Шаг 6*. **Вычисление значения целевой функции.** Вычислить значение квадратичной целевой функции в найденной оптимальной точке: .

**Особенности реализации алгоритма градиентного спуска с постоянным шагом**

Для создания программы использовался язык программирования Python 3.10 и среда разработки PyCharm Professional 2023.2. Для графической визуализации были подключены графические фреймворки Tkinter и Matplotlib.

Интерфейс программы имеет вид:

Изображение выглядит как текст, снимок экрана, диаграмма, План

Автоматически созданное описаниеРисунок 1 – Главное окно программы.

В данном окне можно выбрать необходимую вкладку соответствующей лабораторной работы, в данном случае «ЛР2», ввести начальную точку для функции с координатами «X» и «Y» и задержку между 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.

Изображение выглядит как текст, снимок экрана, Прямоугольник

Автоматически созданное описание

Рисунок 2 – Выполнение и результаты.

Также, в интерфейсе программы есть панель управления «Функция и отображения её графика», в котором можно выбрать необходимую функцию и задать её параметры:

1. X интервал (интервал на котором строится и минимизируется функция);
2. Y интервал (интервал на котором строится и минимизируется функция);
3. Ось X интервал (размер отображаемой сетки по оси X);
4. Ось Y интервал (размер отображаемой сетки по оси Y).

На рисунке 3 показана панель «Функция и отображение её графика»

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описаниеРисунок 3 – Функция и отображение её графика.

Для запуска алгоритма необходимо нажать кнопку «Выполнить», при этом поле «Выполнение и результаты» начнёт динамически заполняться точками, а на отображаемой функции можно наглядно увидеть функционирование алгоритма - искомые точки, которые также отображаются динамически как показано на рисунке 4. Результирующая точка выделена на графике крестом. Значения результирующей функции имеют разный окрас в зависимости от положения: самые высокие точки показаны ярко-жёлтым цветом, а самые низкие – тёмно-синим.

Изображение выглядит как текст, снимок экрана, диаграмма, Графическое программное обеспечение

Автоматически созданное описаниеРисунок 4 – Результат работы программы.

**Вывод:** в ходе работы были изучены различные методы безусловной поисковой оптимизации с использованием производных, реализован метод градиентного спуска с постоянным шагом.

**Листинг программы**

Файл LR1.py

import tkinter as tk  
from tkinter import ttk  
import numpy as np  
import matplotlib.pyplot as plt  
from mpl\_toolkits.mplot3d import Axes3D  
from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg  
import time  
import numdifftools as nd  
from tkinter import scrolledtext  
  
# Определение функций, которые мы можем оптимизировать  
def target\_function(x, y):  
 return ((x \*\* 2 + y - 11) \*\* 2) + ((x + y \*\* 2 - 7) \*\* 2)  
  
# Функция для градиента  
def gradient(function, input):  
 ret = np.empty(len(input))  
 for i in range(len(input)):  
 fg = lambda x: partial\_function(function, input, i, x)  
 ret[i] = nd.Derivative(fg)(input[i])  
 return ret  
  
# Функция для частной производной  
def partial\_function(f\_\_\_, input, pos, value):  
 tmp = input[pos]  
 input[pos] = value  
 ret = f\_\_\_(\*input)  
 input[pos] = tmp  
 return ret  
  
# Функция, которая будет выполнена при нажатии кнопки "Выполнить"  
def run\_optimization():  
 x0 = x\_var.get()  
 y0 = y\_var.get()  
 step = step\_var.get()  
 max\_iterations = iterations\_var.get()  
 delay = delay\_var.get()  
 ax.cla()  
 x\_range = np.linspace(x\_interval\_min.get(), x\_interval\_max.get(), 100)  
 y\_range = np.linspace(y\_interval\_min.get(), y\_interval\_max.get(), 100)  
 X, Y = np.meshgrid(x\_range, y\_range)  
 Z = target\_function(X, Y)  
 ax.plot\_surface(X, Y, Z, cmap='viridis',alpha=0.7)  
 ax.set\_xlabel('X')  
 ax.set\_ylabel('Y')  
 ax.set\_zlabel('Z')  
 ax.set\_xticks(np.arange(x\_interval\_min.get(), x\_interval\_max.get() + 1, x\_axis\_interval.get()))  
 ax.set\_yticks(np.arange(y\_interval\_min.get(), y\_interval\_max.get() + 1, y\_axis\_interval.get()))  
 ax.set\_title("Алгоритм градиентного спуска с постоянным шагом")  
  
 function\_choice = function\_var.get()  
 if function\_choice == "Функция Химмельблау":  
 target\_func = target\_function  
  
 results = []  
 results\_text.config(state=tk.NORMAL)  
 results\_text.delete(1.0, tk.END)  
 for k in range(max\_iterations):  
 (gx, gy) = gradient(target\_func, [x0, y0])  
  
 if np.linalg.norm((gx, gy)) < 0.0001:  
 break  
 x1, y1 = x0 - step \* gx, y0 - step \* gy  
 f1 = target\_func(x1, y1)  
 f0 = target\_func(x0, y0)  
  
 while not f1 < f0:  
 step = step / 2  
 x1, y1 = x0 - step \* gx, y0 - step \* gy  
 f1 = target\_func(x1, y1)  
 f0 = target\_func(x0, y0)  
  
 if np.sqrt((x1 - x0) \*\* 2 + (y1 - y0) \*\* 2) < 0.0001 and abs(f1 - f0) < 0.0001:  
 x0, y0 = x1, y1  
 break  
 else:  
 x0, y0 = x1, y1  
  
 results.append((x0, y0, k,f1))  
 ax.scatter([x0], [y0], [f1], color='red',s=10)  
 results\_text.insert(tk.END,  
 f"Шаг {k}: Координаты ({x0:.2f}, {y0:.2f}), Значение функции: {f1:.7f}\n")  
 results\_text.yview\_moveto(1)  
 canvas.draw()  
 root.update()  
 time.sleep(delay)  
  
 length=len(results)-1  
 ax.scatter(results[length][0], results[length][1], results[length][3], color='black',marker='x',s=60)  
 results\_text.insert(tk.END,  
 f"Результат:\nКоординаты ({results[length][0]:.8f}, {results[length][1]:.8f})\nЗначение функции: {results[length][3]:.8f}\n")  
 results\_text.yview\_moveto(1)  
 results\_text.config(state=tk.DISABLED)  
  
# Создание окна приложения  
root = tk.Tk()  
root.title("Методы поисковой оптимизации")  
  
notebook = ttk.Notebook(root)  
notebook.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)  
  
# Вкладка для лр1  
param\_frame = ttk.Frame(notebook,padding=(15, 0))  
notebook.add(param\_frame, text="ЛР1")  
  
param\_frame2 = ttk.Frame(notebook)  
notebook.add(param\_frame2, text="ЛР2")  
  
param\_frame3 = ttk.Frame(notebook)  
notebook.add(param\_frame3, text="ЛР3")  
  
param\_frame4 = ttk.Frame(notebook)  
notebook.add(param\_frame4, text="ЛР4")  
  
param\_frame5 = ttk.Frame(notebook)  
notebook.add(param\_frame5, text="ЛР5")  
  
param\_frame6 = ttk.Frame(notebook)  
notebook.add(param\_frame6, text="ЛР6")  
  
param\_frame7 = ttk.Frame(notebook)  
notebook.add(param\_frame7, text="ЛР7")  
  
param\_frame8 = ttk.Frame(notebook)  
notebook.add(param\_frame8, text="ЛР8")  
  
  
# Параметры задачи  
ttk.Label(param\_frame, text="Инициализация значений", font=("Helvetica", 12)).grid(row=0, column=0,pady=15)  
ttk.Label(param\_frame, text="X начальное", font=("Helvetica", 10)).grid(row=1, column=0)  
ttk.Label(param\_frame, text="Y начальное", font=("Helvetica", 10)).grid(row=2, column=0)  
ttk.Label(param\_frame, text="Шаг", font=("Helvetica", 10)).grid(row=3, column=0)  
ttk.Label(param\_frame, text="Число итераций", font=("Helvetica", 10)).grid(row=4, column=0)  
ttk.Label(param\_frame, text="Задержка (сек)", font=("Helvetica", 10)).grid(row=5, column=0)  
  
x\_var = tk.DoubleVar(value=-1)  
y\_var = tk.DoubleVar(value=-1)  
step\_var = tk.DoubleVar(value=0.5)  
iterations\_var = tk.IntVar(value=100)  
delay\_var = tk.DoubleVar(value=0.5)  
  
x\_entry = ttk.Entry(param\_frame, textvariable=x\_var)  
y\_entry = ttk.Entry(param\_frame, textvariable=y\_var)  
step\_entry = ttk.Entry(param\_frame, textvariable=step\_var)  
iterations\_entry = ttk.Entry(param\_frame, textvariable=iterations\_var)  
delay\_entry = ttk.Entry(param\_frame, textvariable=delay\_var)  
  
x\_entry.grid(row=1, column=1)  
y\_entry.grid(row=2, column=1)  
step\_entry.grid(row=3, column=1)  
iterations\_entry.grid(row=4, column=1)  
delay\_entry.grid(row=5, column=1)  
  
separator = ttk.Separator(param\_frame, orient="horizontal") # Горизонтальная полоса разделения  
separator.grid(row=7, column=0, columnspan=2, sticky="ew",pady=10)  
  
# Параметры функции  
ttk.Label(param\_frame, text="Функция и отображение ее графика", font=("Helvetica", 12)).grid(row=9, column=0,pady=10)  
ttk.Label(param\_frame, text="Выберите функцию", font=("Helvetica", 10)).grid(row=10, column=0)  
function\_choices = ["Функция Химмельблау"]  
function\_var = tk.StringVar(value=function\_choices[0])  
function\_menu = ttk.Combobox(param\_frame, textvariable=function\_var, values=function\_choices,width=22)  
function\_menu.grid(row=10, column=1,pady=5)  
ttk.Label(param\_frame, text="X интервал (min)", font=("Helvetica", 10)).grid(row=11, column=0)  
ttk.Label(param\_frame, text="X интервал (max)", font=("Helvetica", 10)).grid(row=12, column=0)  
ttk.Label(param\_frame, text="Y интервал (min)", font=("Helvetica", 10)).grid(row=13, column=0)  
ttk.Label(param\_frame, text="Y интервал (max)", font=("Helvetica", 10)).grid(row=14, column=0)  
ttk.Label(param\_frame, text="Ось X интервал", font=("Helvetica", 10)).grid(row=16, column=0)  
ttk.Label(param\_frame, text="Ось Y интервал", font=("Helvetica", 10)).grid(row=17, column=0)  
  
separator = ttk.Separator(param\_frame, orient="horizontal") # Горизонтальная полоса разделения  
separator.grid(row=18, column=0,columnspan=2, sticky="ew",pady=10)  
  
x\_interval\_min = tk.DoubleVar(value=-5)  
x\_interval\_max = tk.DoubleVar(value=5)  
y\_interval\_min = tk.DoubleVar(value=-5)  
y\_interval\_max = tk.DoubleVar(value=5)  
x\_axis\_interval = tk.IntVar(value=2)  
y\_axis\_interval = tk.IntVar(value=2)  
  
x\_interval\_min\_entry = ttk.Entry(param\_frame, textvariable=x\_interval\_min)  
x\_interval\_max\_entry = ttk.Entry(param\_frame, textvariable=x\_interval\_max)  
y\_interval\_min\_entry = ttk.Entry(param\_frame, textvariable=y\_interval\_min)  
y\_interval\_max\_entry = ttk.Entry(param\_frame, textvariable=y\_interval\_max)  
x\_axis\_interval\_entry = ttk.Entry(param\_frame, textvariable=x\_axis\_interval)  
y\_axis\_interval\_entry = ttk.Entry(param\_frame, textvariable=y\_axis\_interval)  
  
x\_interval\_min\_entry.grid(row=11, column=1)  
x\_interval\_max\_entry.grid(row=12, column=1)  
y\_interval\_min\_entry.grid(row=13, column=1)  
y\_interval\_max\_entry.grid(row=14, column=1)  
x\_axis\_interval\_entry.grid(row=16, column=1)  
y\_axis\_interval\_entry.grid(row=17, column=1)  
  
# Создание кнопки Выполнить  
button\_style = ttk.Style()  
button\_style.configure("My.TButton", font=("Helvetica", 14))  
  
# Создание кнопки Выполнить  
apply\_settings\_button = ttk.Button(param\_frame, text="Выполнить", command=run\_optimization, style="My.TButton")  
apply\_settings\_button.grid(row=21, column=1, padx=10, pady=10)  
  
ttk.Label(param\_frame, text="Выполнение и результаты", font=("Helvetica", 12)).grid(row=20, column=0,pady=10)  
results\_text = scrolledtext.ScrolledText(param\_frame, wrap=tk.WORD, height=18, width=40,padx=2, state=tk.DISABLED)  
results\_text.grid(row=21, column=0,padx=10)  
  
# Инициализация графика при запуске программы  
fig = plt.figure(figsize=(8, 9)) # Установка размеров фигуры (ширина, высота)  
ax = fig.add\_subplot(111, projection='3d')  
ax.set\_title("Алгоритм градиентного спуска с постоянным шагом")  
canvas = FigureCanvasTkAgg(fig, master=root)  
canvas\_widget = canvas.get\_tk\_widget()  
canvas\_widget.pack(side=tk.RIGHT, padx=20)  
  
root.mainloop()