ГУАП

КАФЕДРА № 44

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

доцент, к.т.н., доцент		А.А.Сенцов
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №2

МНОГОМЕРНАЯ ЛОКАЛЬНАЯ ОПТИМИЗАЦИЯ

по курсу: МЕТОДЫ ОПТИМИЗАЦИИ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 4242М		М.В.Климов
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2023

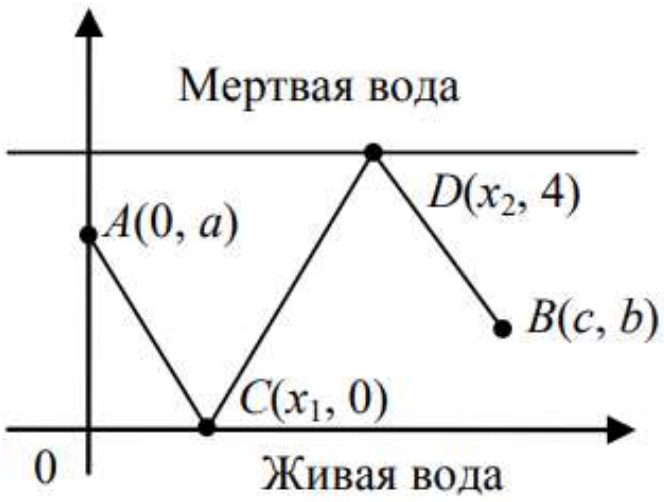
1. Цель работы и вариант

Ознакомиться с известными аналитическими и численными методами одномерной локальной оптимизации, получить опыт решения одномерной оптимизационной задачи, приобрести навыки использования математических программных средств для решения одномерных экстремальных задач.

Вариант 7.

Вариант	Задача	Метод	a	b	c
7	A	g	1	2	5

Задача А. Серый Волк находится в точке А и хочет оживить Ивана-Царевича, находящегося в точке В. Оба они находятся между реками с мертвой и живой водой, текущими параллельно друг другу. Как Волку добраться до Ивана кратчайшим путем, набрав по дороге сначала живой, а затем мертвой воды?



g) методом DFP

2. Вычисления:

Первым делом требуется определить целевую функцию.

В данном случае, расстояние от начальной точки до конечной можно вычислить по формуле:

$$S = S1 + S2 + S3 = \sqrt{a^2 + x1^2} + \sqrt{4^2 + (x2 - x1)^2} + \sqrt{(4 - b)^2 + (c - x2)^2}$$

Для поиска стационарных точек функции найдем ее частные производные и, приравняв каждую производную нулю, получим систему уравнений:

$$\frac{\mathrm{df(x1, x2)}}{\mathrm{dx1}} = \frac{x1}{\sqrt{a^2 + x1^2}} + \frac{x1 - x2}{\sqrt{4^2 + (x2 - x1)^2}} = 0$$

$$\frac{\mathrm{df(x1, x2)}}{\mathrm{dx2}} = \frac{x2 - x1}{\sqrt{4^2 + (x2 - x1)^2}} + \frac{x2 - c}{\sqrt{(4 - b)^2 + (c - x2)^2}} = 0$$

При a = 1, b = 2, c = 5, данная система уравнений имеет решение x1 = 5/7 = 0.714285714285714, x2 = 25/7 = 3.57142857142857. Определим вид полученной стационарной точки. Для этого построим матрицу Гессе и рассчитаем ее значение в стационарной точке:

$$H = \begin{pmatrix} 1715\frac{\sqrt{74}}{21904} & -343\frac{\sqrt{74}}{21904} \\ -343\frac{\sqrt{74}}{21904} & 1029\frac{\sqrt{74}}{21904} \end{pmatrix}$$

Гессиан симметричен и все его угловые миноры положительны:

$1715\frac{\sqrt{74}}{21904} > 0$, detH > 0.

Таким образом, согласно критерию Сильвестра, матрица H в стационарной точке положительно определена, следовательно, стационарная точка является точкой минимума.

Если подставить найденную точку в исходную функцию, то результат будет равен 8.602

Далее требуется найти минимум программным путём, используя метод DFP.

Для этого внесём в программу целевую функцию, и её производные по x1 и x2. Далее нужно указать начальное значение, от которого будет вестись поиск (в данном случае это точка [1;1]) а также коэффициенты rho = 0,55 и sigma = 0,4. Код программы приведён в приложении 1, а результат выполнения представлен на рисунках 1 и 2. По результату можно сделать вывод, что программа добралась до искомой точки от точки 1;1 за 6 тактов.

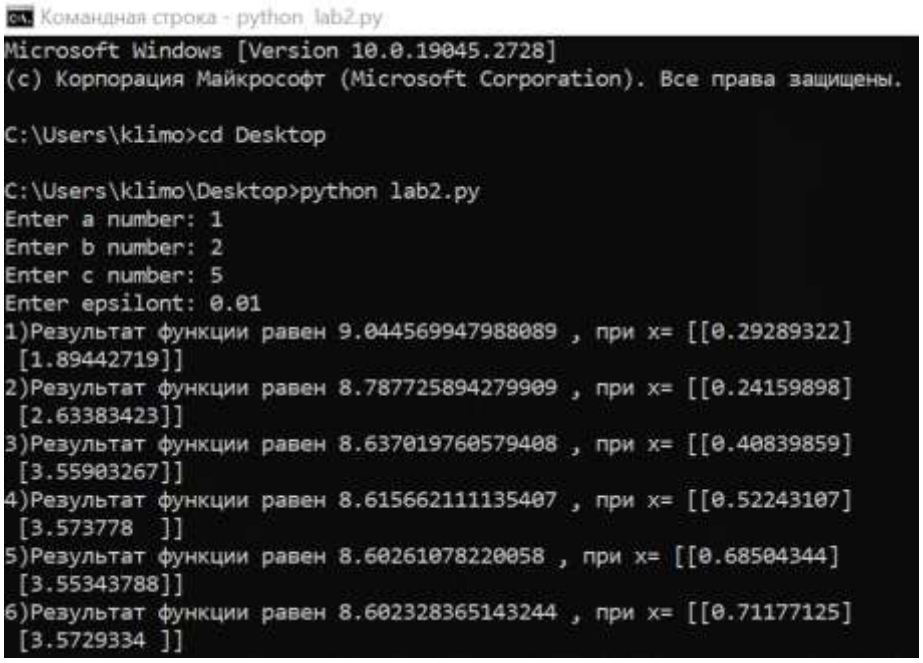


Рисунок 1. Результат выполнения программы в консоли.

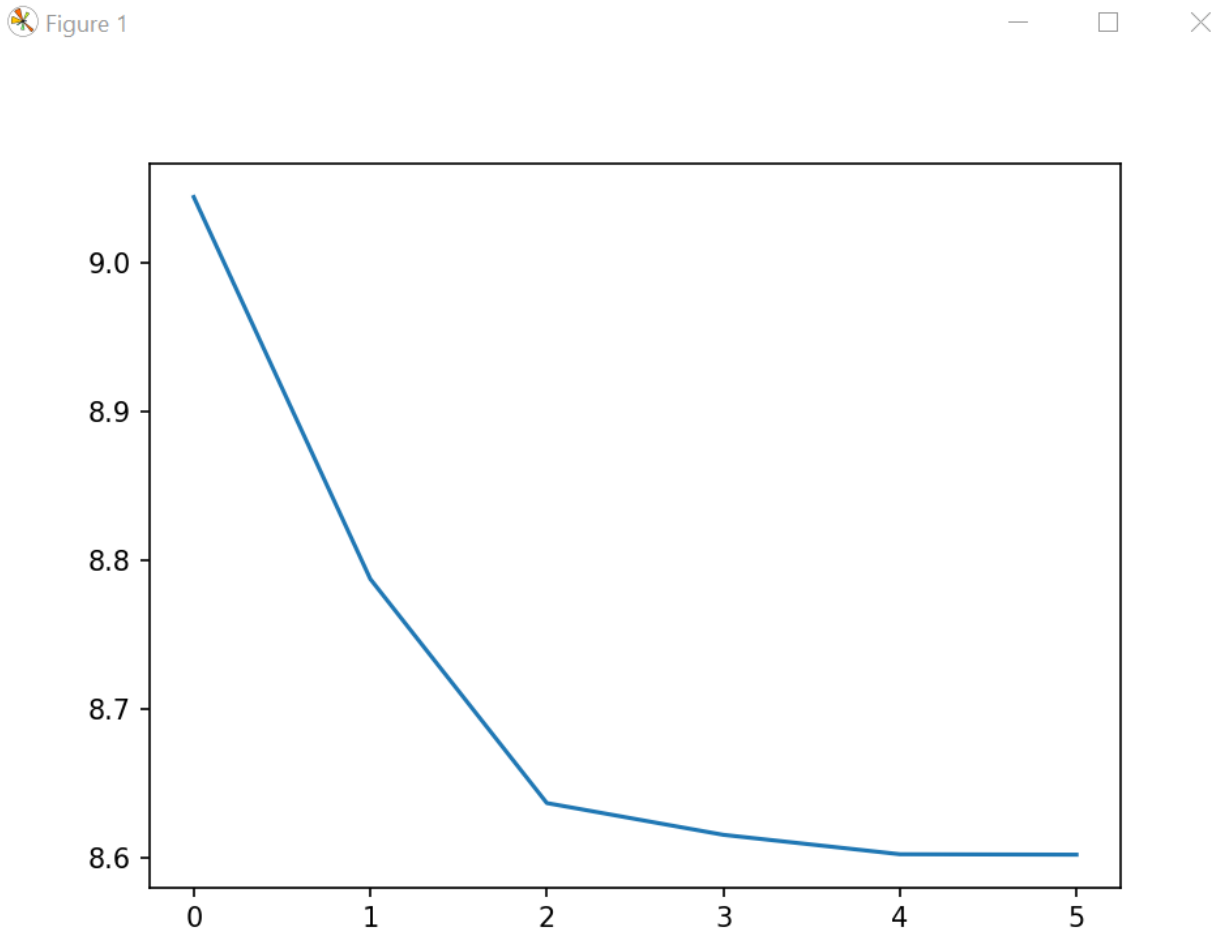


Рисунок 2. Результат выполнения программы на графике.

3. Выводы.

В процессе выполнения лабораторной работы была решена поставленная задача по нахождению минимума двумя способами: аналитическим и программным (Методом DFP). В обоих случаях был получен приблизительно одинаковый ответ, следовательно, при учёте правильности составления целевой функции, оба получившихся решений были правильными.

ПРИЛОЖЕНИЕ 1

Содержимое программы

```
import numpy as np
import matplotlib.pyplot as plt
import math;
def fun(a, b, c, x):
    return (((a ** 2) + (x[0,0] ** 2)) ** (1/2)) + ((((x[1,0]-x[0,0]) ** 2) + (4 ** 2)) ** (1/2)) + ((((c-x[1,0]) ** 2) + ((4-b) ** 2)) ** (1/2))

def gfun(a, b, c, x):
    result = np.zeros((2, 1))
    result[0, 0] = (x[0,0]/(((a ** 2) + (x[0,0] ** 2)) ** (1/2))) + ((x[0,0]-x[1,0])/((((x[1,0]-x[0,0]) ** 2) + (4 ** 2)) ** (1/2)))
    result[1, 0] = ((x[1,0]-x[0,0])/((((x[1,0]-x[0,0]) ** 2) + (4 ** 2)) ** (1/2))) + (x[1,0]-c)/((((c-x[1,0]) ** 2) + ((4-b) ** 2)) ** (1/2))
    return result

def dfp(a, b, c, fun, gfun, x0, e):
    result = []
    rho = 0.55
    n = 0
    sigma = 0.4
    m = np.shape(x0)[0]
    Hk = np.eye(m)
    k = 10*e
    while (k > e):
        gk = np.mat (gfun (a, b, c,x0)) # Рассчитать градиент
        dk = -np.mat(Hk)*gk
        m = 0
        mk = 0
        while (m < 20):
            newf = fun(a, b, c,x0 + rho ** m * dk)
            oldf = fun(a, b, c,x0)
```

```
            if (newf < oldf + sigma * (rho ** m) * (gk.T * dk)[0,0]):
                mk = m
                break
            m = m + 1

                 # Коррекция DFP
        x = x0 + rho ** mk * dk
        sk = x - x0
        yk = gfun(a, b, c,x) - gk
        if (sk.T * yk > 0):
            Hk = Hk - (Hk * yk * yk.T * Hk) / (yk.T * Hk * yk) + (sk *
sk.T) / (sk.T * yk)

        k = abs(fun(a, b, c,x) - fun(a, b, c,x0))
        x0 = x
        result.append(fun(a, b, c,x0))
        n=n+1
        print(f'{n})Результат функции равен {fun(a,b,c,x0)} , при x=
{x0}')

    return result


a = float(input('Enter a number: '))
while (a<0) or (a>4):
    a = float(input('Enter 0<a<4 number: '))
b = float(input('Enter b number: '))
while (b<0) or (b>4):
    b = float(input('Enter 0<b<4 number: '))
c = abs(float(input('Enter c number: ')))
e = float(input('Enter epsilont: '))
n = 0

x0 = np.mat([[1], [1]])
```

```
result = dfp(a, b, c,fun, gfun, x0, e)

n = len(result)
ax = plt.figure().add_subplot(111)
x = np.arange(0, n, 1)
y = result
ax.plot(x,y)

plt.show()
```

ПРИЛОЖЕНИЕ 2

Вопросы:

1. Метод Лагранжа, теория и примеры.

Метод, предложенный Лагранжем, может быть сформулирован в виде следующей теоремы.

Теорема о множителях Лагранжа.

Пусть x0 – решение задачи условной оптимизации

$f(x1, ..., xn) \rightarrow \min (\max)$

$q1(x1, ..., xn) = 0, \ldots, qm(x1, ..., xn) = 0.$

Тогда, если все функции в задаче дифференцируемы в точке x0, то их градиенты в этой точке линейно зависимы, то есть существуют такие не все равные нулю множители $\lambda 0, \lambda 1, \ldots, \lambda m$, для которых: $\lambda 0 \nabla f(x0) + \lambda 1 \nabla q1(x0) + \ldots + \lambda m \nabla qm(x0) = 0$.

Данную теорему можно определить, как следующий список действий (алгоритм) для решения задач оптимизации:

1. Убедиться, что целевая функция в пределах ограничений не принимает бесконечно больших (если ищется максимум) или бесконечно малых (при поиске минимума) значений. В противном случае задача решения не имеет.
2. Составить функцию Лагранжа, найти ее стационарные точки и выбрать точки нужного типа.
3. Рассчитать значения функции f(x) в выбранных точках. Точка с наименьшим (наибольшим) значением является точкой глобального минимума (максимума).

Замечания:

1) если при составлении функции Лагранжа учтены не все ограничения, необходимо проверить, удовлетворяют ли точки, полученные в п. 2, неучтенным ограничениям, а также в п. 3 рассчитать значения функции на границе этих ограничений;

2) при поиске стационарных точек функции Лагранжа удобно рассмотреть два возможных случая: $\lambda 0 = 1$ и $\lambda 0 = 0$;

3) если целевая функция является выпуклой, необходимые условия экстремума являются достаточными.

Этот алгоритм использовался в аналитическом решении задачи данной работы.

2. Генетические алгоритмы и их применение для решения задач оптимизации.

Генетические алгоритмы заключаются в оптимизации решения математических задач на основе анализа данных теории эволюции. Например, Simple Genetic Algorithm – обрабатывает популяцию n строк одинаковой длинны (например, строки из 0 и 1) и скрещивая их (с шансом на мутацию), находит оптимальных «потомков».

ПРИМЕР: N = 4

Особи	f(x)
01101	169
11000	576
01000	64
10011	361

f(x) -> max

Одна итерация алгоритма состоит из N/2 повторений следующих действий.

1. Отбор. На данном шаге осуществляется выбор двух родителей из популяции. При этом наилучшие особи должны выбираться с большей долей вероятности.
    a. Метод рулетки(размер сектора пропорционален f(x))

No.	String	Fitness	% of Total
1	01101	169	14.4
2	11000	576	49.2
3	01000	64	5.5
4	10011	361	30.9
Total		1170	100.0

2. Мутация. С малой долей вероятности изменяется один бит каждого из родителей.
    a. Нормальной вероятностью является один случай на 1000 бит
3. Скрещивание. С некоторой долей вероятности происходит скрещивание родителей: обмен частями строк. В результате скрещивания образуются два потомка, занимающие места родителей в следующей популяции. Если скрещивания не произошло, родители попадают в следующую популяцию.
    a. Для осуществления скрещивания выбирается случайное число k от 1 до l-1 (l – длина сроки). Затем родители обмениваются хвостами строк, начиная с символа k+1.
    b. Например, пусть k = 4 и выбраны родители 01101 и 11000. Тогда потомки будут иметь вид: 01100 и 11001.

Таким образом, по окончании одной итерации будет сформировано новое поклонение из N особей. Алгоритм завершается, кода будет выполнено заданное число итераций.

3. Выполняется ли проверка корректности входных данных? Что будет, если ввести некорректные значения?

Проверка корректности выполняется. При попытке ввести некорректные данные, программа запросит данные снова с указанием ограничений.