|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

**ФАКУЛЬТЕТ** ***ИУК «Информатика и управление»***

**КАФЕДРА** \_\_***ИУК4 «Программное обеспечение ЭВМ, информационные технологии»***

**ЛАБОРАТОРНАЯ РАБОТА №1.1**

**«Минимизация функций»**

**ДИСЦИПЛИНА: «Моделирование»**

|  |  |  |
| --- | --- | --- |
| Выполнил: студент гр. ИУК4-72Б | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Карельский М.К. )  (Подпись) |
| Проверил: | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Никитенко У.В. )  (Подпись) |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | | |
|  | - Балльная оценка:  - Оценка: | |

Калуга, 2023

**Цель:** сформировать практические навыки анализа возможностей построения и выделения наиболее важных свойств объектов моделей для моделирования и использования специализированных программных пакетов и библиотек для решения задачи минимизации функции и визуализации результатов решения.

**Задачи:** найти минимум функции, указанной в варианте предложенным методом, сравнить результаты, выдвинуть и обосновать гипотезу целесообразности использования того или иного метода в зависимости от предложенной задачи и ее вариаций, точности результата, трудоемкости, сложности алгоритма, сложности обоснования применимости метода, вычислительной эффективности алгоритма. Визуализировать результаты.

**Вариант 7**

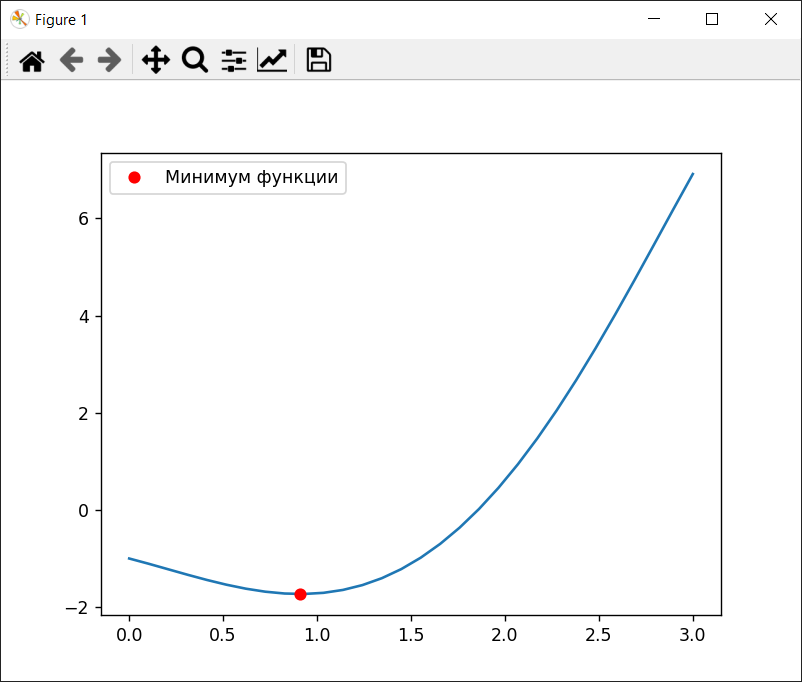
**Задание 1.7.**

Методом Ньютона найти минимум и максимум унимодальной на отрезке функции с точностью . Предусмотреть подсчет числа итераций, потребовавшихся для достижения заданной точности.

**Решение:**

В качестве начального значения возьмем:

Результаты вычислений:



**Рис. 1.** Метод Ньютона

**Задание 2.3.**

Указанным в индивидуальном варианте методом найти минимумы и максимумы функции на отрезке с точностью . Предусмотреть подсчет числа итераций, потребовавшихся для достижения заданной точности.

Метод – Фибоначчи

**Решение:**

В качестве константы различимости выберем:

Установим, что:

Тогда общее число вычислений n определим из:

Вычислим:

Шаг 1: если:

то:

иначе:

Шаг 2: если:

то увеличить k на 1 и перейти к первому шагу.

Шаг 3:

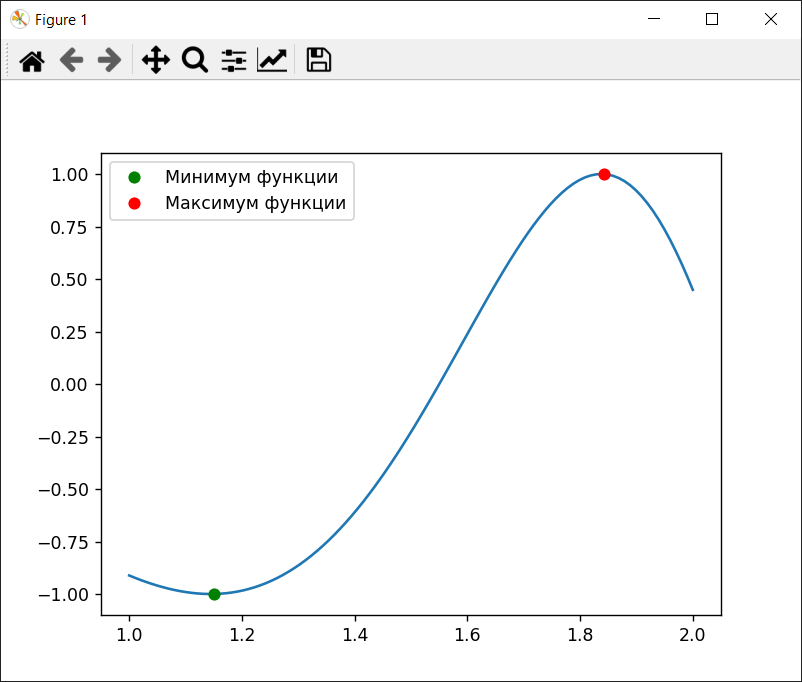
Если:

то:

иначе:

В результате имеем:

Результаты вычислений:



**Рис. 2.** Метод Фибоначчи

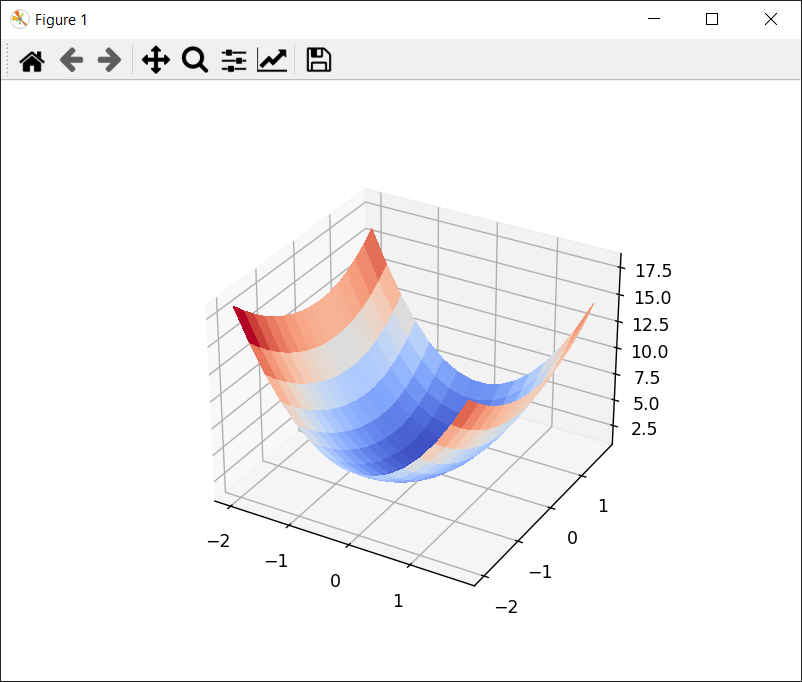
**Задание 5.7.**

Найти минимум функции 2-х переменных с точностью на прямоугольнике .

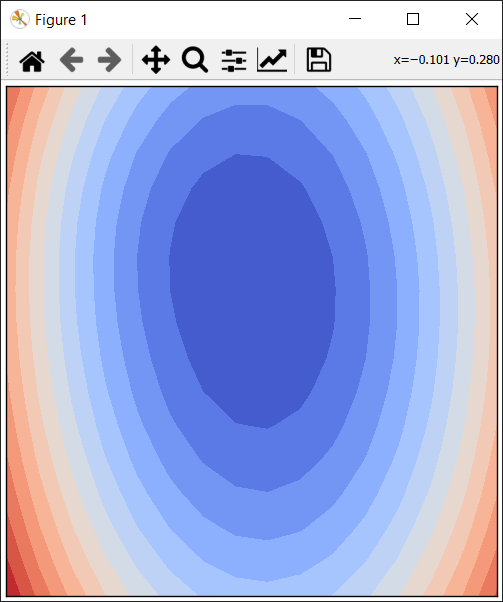
Порядок решения задачи:

1. Задать указанную в варианте функцию .
2. Построить графики функции и поверхностей уровня .
3. По графикам найти точки начального приближения к точкам экстремума.
4. Найти экстремумы функции c заданной точностью.

**Решение:**



**Рис. 3.** График функции



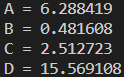
**Рис. 4.** График поверхностей уровня

Исходя из графиков, можно предположить, что в указанных границах функция имеет точку минимума M примерно в .

Вычислим точное значение:



**Рис. 5.** Решение системы



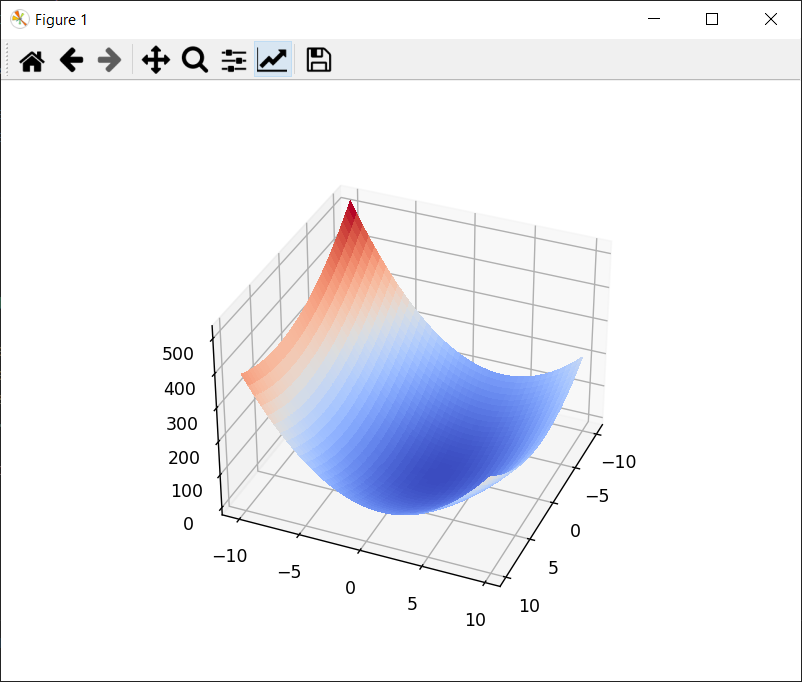
**Рис. 6.** Проверка достаточного условия экстремумы

**Задание 6.7.**

Указанным в индивидуальном варианте методом найти минимум квадратичной функции с точностью . Для решения задачи многомерной минимизации использовать метод Ньютона. Построить график функции f. Предусмотреть подсчет числа итераций, потребовавшихся для достижения заданной точности.

Метод – сопряженных направлений

**Решение:**



**Рис. 6.** График функции

Возьмем за исходную точку .

Следующие вычисления будут проводиться в цикле, пока , где k – текущая итерация

Результаты вычислений:

Решение методом Ньютона:

Результаты вычислений:

**Вывод:** в ходе выполнения лабораторной работы были получены практические навыки анализа возможностей построения и выделения наиболее важных свойств объектов моделей для моделирования и использования специализированных программных пакетов и библиотек для решения задачи минимизации функции и визуализации результатов решения.

**ПРИЛОЖЕНИЯ**

**Листинг:**

**Task\_1.py**

import math, numpy

import matplotlib.pyplot as plt

f = lambda x: x\*\*3 - math.exp(x)

a = 0

b = 3

eps = 10\*\*-6

df = lambda x: 3\*x\*\*2 - math.exp(x)

d2f = lambda x: 6\*x - math.exp(x)

x = (a + b) / 2

err = None

n = 0

while err == None or err > eps:

    if err == None:

        err = abs(df(x))

        continue

    err = abs(df(x))

    x = x - df(x)/d2f(x)

    n += 1

print(f"x = {x:.6f}; f(x) = {f(x):.6f}; n = {n}")

X = numpy.linspace(a, b, (b-a)\*10)

Y = [f(x) for x in X]

plt.plot(X, Y)

plt.plot(x, f(x), 'ro', label='Минимум функции')

plt.legend()

plt.show()

**Task\_2.py**

import math, numpy

import matplotlib.pyplot as plt

f = lambda x: math.cos(math.exp(x))

x\_1 = 1

x\_2 = 2

vareps = 10\*\*-6

eps = 0.01

l = 2\*vareps

def Fibonacci(f):

    a = x\_1

    b = x\_2

    F\_x = (b - a)/l

    F = [1, 1]

    while F[-1] < F\_x:

        F.append(F[-1] + F[-2])

    n = len(F) - 1

    lmb = a + F[-3] / F[-1] \* (b - a)

    mu = a + F[-2] / F[-1] \* (b - a)

    k = 0

    while k != n - 2:

        k += 1

        if f(lmb) > f(mu):

            a = lmb

            lmb = mu

            mu = a + F[-k-2] / F[-k-1] \* (b - a)

        else:

            b = mu

            mu = lmb

            lmb = a + F[-k-3] / F[-k-1] \* (b - a)

    mu = lmb + eps

    if f(lmb) == f(mu):

        a = lmb

    else:

        b = mu

    print(f"n = {n}")

    return (a + b) / 2

x\_min = Fibonacci(f)

print(f"x = {x\_min:.6f}; f(x) = {f(x\_min):.6f}")

x\_max = Fibonacci(lambda x: -f(x))

print(f"x = {x\_max:.6f}; f(x) = {f(x\_max):.6f}")

X = numpy.linspace(x\_1, x\_2, (x\_2-x\_1)\*100)

Y = [f(x) for x in X]

plt.plot(X, Y)

plt.plot(x\_min, f(x\_min), 'go', label='Минимум функции')

plt.plot(x\_max, f(x\_max), 'ro', label='Максимум функции')

plt.legend()

plt.show()

**Task\_3.py**

import matplotlib.pyplot as plt

from matplotlib import cm

import numpy as np

from scipy import optimize

x\_1 = -2

x\_2 = 2

y\_1 = -2

y\_2 = 2

eps = 10\*\*-6

fig, ax = plt.subplots(subplot\_kw={"projection": "3d"})

X = np.arange(x\_1, x\_2, 0.25)

Y = np.arange(y\_1, y\_2, 0.25)

X, Y = np.meshgrid(X, Y)

Z = 3\*X\*\*2 + Y\*\*2 + np.log(X\*\*2 + Y\*\*2 + 2\*X - 2\*Y + 3)

surf = ax.plot\_surface(X, Y, Z, cmap=cm.coolwarm,

                       linewidth=0, antialiased=False)

plt.show()

plt.style.use('\_mpl-gallery-nogrid')

levels = np.linspace(Z.min(), Z.max(), 15)

fig, ax = plt.subplots()

ax.contourf(X, Y, Z, levels=levels, cmap=cm.coolwarm, antialiased=False)

plt.show()

def equations(vars):

    x, y = vars

    eq1 = 6\*x + (2\*x + 2) / (x\*\*2 + y\*\*2 + 2\*x - 2\*y + 3)

    eq2 = 2\*y + (2\*y - 2) / (x\*\*2 + y\*\*2 + 2\*x - 2\*y + 3)

    return [eq1, eq2]

x, y = optimize.fsolve(equations, (-0.1, 0.3), xtol=eps)

print(f'x = {x:.6f}')

print(f'y = {y:.6f}')

A = 6 + (-2\*x\*\*2 + 2\*y\*\*2 - 4\*x - 4\*y + 2) / (x\*\*2 + y\*\*2 + 2\*x - 2\*y + 3)\*\*2

B = -(2\*x + 2) \* (2\*y - 2) / (x\*\*2 + y\*\*2 + 2\*x - 2\*y + 3)\*\*2

C = 2 + (2\*x\*\*2 - 2\*y\*\*2 + 4\*x + 4\*y + 2) / (x\*\*2 + y\*\*2 + 2\*x - 2\*y + 3)\*\*2

print(f'A = {A:.6f}')

print(f'B = {B:.6f}')

print(f'C = {C:.6f}')

D = A\*C - B\*\*2

print(f'D = {D:.6f}')

**Task\_4.py**

import numpy as np

from matplotlib import cm

import matplotlib.pyplot as plt

import math

eps = 10\*\*-6

f = lambda x, y: x\*\*2 + 0.5\*x\*y + 2.5\*y\*\*2 - 2\*x - 10.5\*y

df\_x = lambda x, y: 2\*x + 0.5\*y - 2

df\_y = lambda x, y: 0.5\*x + 5\*y - 10.5

d2f\_x = 2

d2f\_y = 5

det = d2f\_x \* d2f\_y

x = [0]

y = [3]

p = [(df\_x(x[0], y[0]), df\_y(x[0], y[0]))]

lmd = (df\_x(x[0], y[0]) \* p[0][0] + df\_y(x[0], y[0]) \* p[0][1]) / ((p[0][0]\*\*2 + p[0][1]\*\*2) \* det)

x.append(x[0] - lmd \* p[0][0])

y.append(y[0] - lmd \* p[0][1])

n = 0

while math.sqrt(p[-1][0]\*\*2 + p[-1][1]\*\*2) > eps:

    frac = (df\_x(x[-1], y[-1])\*\*2 + df\_y(x[-1], y[-1])\*\*2) / (df\_x(x[-2], y[-2])\*\*2 + df\_y(x[-2], y[-2])\*\*2)

    p\_1 = df\_x(x[-1], y[-1]) + p[-1][0] \* frac

    p\_2 = df\_y(x[-1], y[-1]) + p[-1][1] \* frac

    p.append((p\_1, p\_2))

    lmd = (df\_x(x[-1], y[-1]) \* p\_1 + df\_y(x[-1], y[-1]) \* p\_2) / ((p\_1\*\*2 + p\_2\*\*2) \* det)

    x.append(x[-1] - lmd \* p\_1)

    y.append(y[-1] - lmd \* p\_2)

    n += 1

print(f"x = {x[-1]:.6f}")

print(f"y = {y[-1]:.6f}")

print(f"f(x, y) = {f(x[-1], y[-1]):.6f}")

print(f"n = {n}")

x = [0]

y = [3]

det\_inv = 1 / 9.75

H\_inv = [[det\_inv \* 5, det\_inv \* -0.5],

         [det\_inv \* -0.5, det\_inv \* 2]]

n = 0

while math.sqrt(df\_x(x[-1], y[-1])\*\*2 + df\_y(x[-1], y[-1])\*\*2) > eps:

    x\_k = x[-1] - (H\_inv[0][0] \* df\_x(x[-1], y[-1]) + H\_inv[0][1] \* df\_y(x[-1], y[-1]))

    y\_k = y[-1] - (H\_inv[1][0] \* df\_x(x[-1], y[-1]) + H\_inv[1][1] \* df\_y(x[-1], y[-1]))

    x.append(x\_k)

    y.append(y\_k)

    n += 1

print(f"x = {x[-1]:.6f}")

print(f"y = {y[-1]:.6f}")

print(f"f(x, y) = {f(x[-1], y[-1]):.6f}")

print(f"n = {n}")

fig, ax = plt.subplots(subplot\_kw={"projection": "3d"})

X = np.arange(-10, 10, 0.25)

Y = np.arange(-10, 10, 0.25)

X, Y = np.meshgrid(X, Y)

Z = X\*\*2 + 0.5\*X\*Y + 2.5\*Y\*\*2 - 2\*X - 10.5\*Y

surf = ax.plot\_surface(X, Y, Z, cmap=cm.coolwarm,

                       linewidth=0, antialiased=False)

plt.show()