|  |  |
| --- | --- |
| **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**

**«Минимизация функции»**

**ДИСЦИПЛИНА: «Моделирование»**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Выполнил: студент гр. ИУК4-72Б | |  |  | ( | Калашников А.С. | ) |
|  |  |  | (подпись) |  | (Ф.И.О.) |  |
| Проверил: | |  |  | ( | Никитенко У.В. | ) |
|  |  |  | (подпись) |  | (Ф.И.О.) |  |

|  |  |
| --- | --- |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | |
|  | - Балльная оценка:  - Оценка: |

Калуга, 2023

**Цель работы:** 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 №6**

|  |  |
| --- | --- |
| **№** | **Выполняемые задачи** |
| 6 | 1.6, 4.2, 5.6, 6.6 |

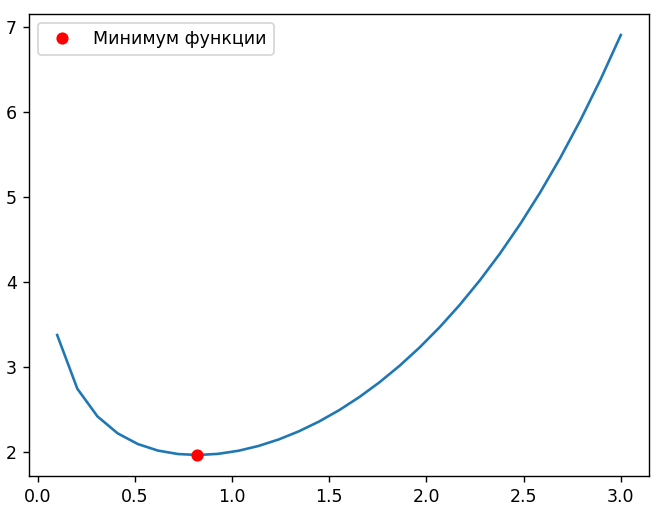
**Задание №1.6**

Методом Ньютона найти минимум и максимум унимодальной на отрезке [a, b] функции f(x) с точностью ε = 10-6. Предусмотреть подсчет числа итераций, потребовавшихся для достижения заданной точности.

|  |  |  |
| --- | --- | --- |
| **F(x)** | **a** | **b** |
| **2^x-ln(x)** | **0.1** | **3** |



**Рис**. 1. Результат работы алгоритма

****

**Рис**. 2. График функции

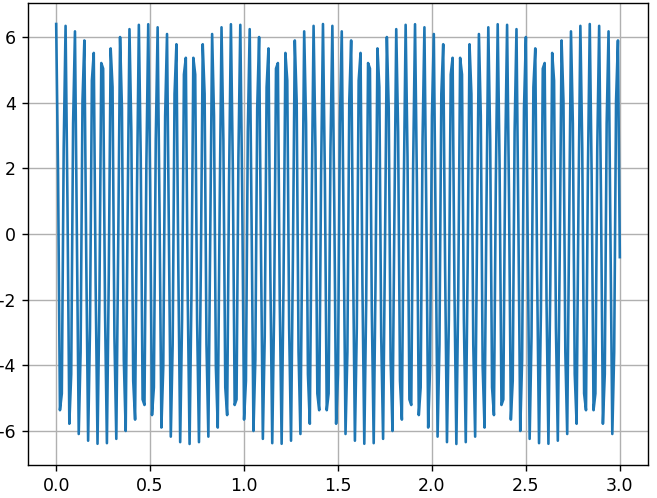
**Задание №4.2**

Функция f(x) = (-1)^n\*(cos(2nx)/n^3) представлена частичной суммой ряда. Построить график функции на заданном отрезке [0, 3] и найти ee минимумы и максимумы с указанной точностью 0,0001, n = 250, Методом минимизации деления отрезка пополам.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **u(x)** | **X1** | **X2** | **n** | **ℰ** | **Метод минимизации** |
| 4.2 |  | 0 | 3 | 250 | 0.0001 | Деления отрезка пополам |



**Рис. 3**. Результат работы алгоритма



**Рис. 4** *–* График функции

**Задание №5.6**

Найти минимум функции 2-х переменных f(x,y) с точностью ε = 10-6 на прямоугольнике [x1, x2] × [y1, y2].

ПОРЯДОК РЕШЕНИЯ ЗАДАЧИ:

1. Задать указанную в варианте функцию f(x,y).

2. Построить графики функции и поверхностей уровня f(x,y).

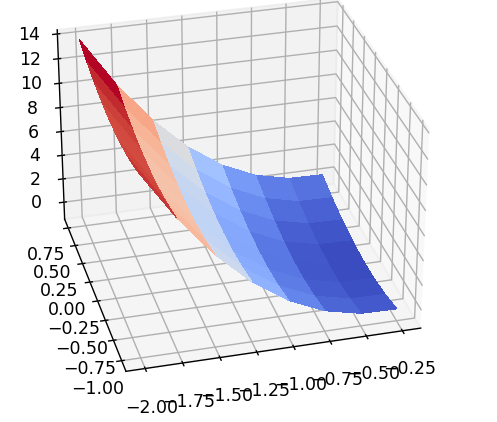
3. По графикам найти точки начального приближения к точкам экстремума.

4. Найти экстремумы функции c заданной точностью.

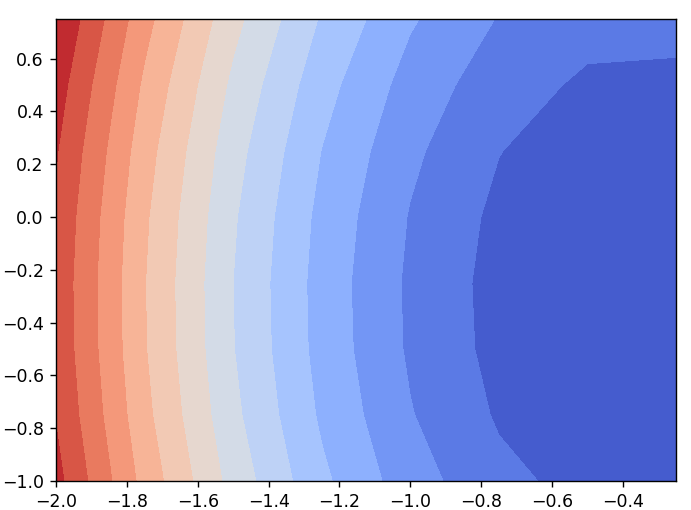
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **f(x)** | **x1** | **x2** | **y1** | **y2** |
| 5.6 |  | -2 | 0 | -1 | 1 |



**Рис. 6.**Результат работы алгоритма

**

**Риc. 6.** *–* Вывод графика

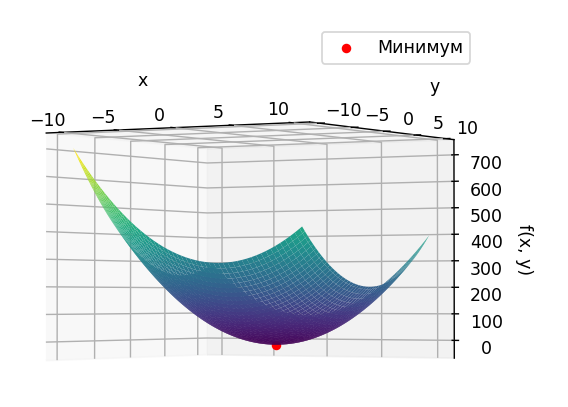
**

**Риc. 6.** *–* Вывод графика

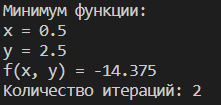
**Задание №6.6**

Методом наискорейший спуск найти минимум квадратичной функции f(x,y)=a(11)x^2+2a(12)xy+a(22)y^2+2a(13)x+2a(23)y с точностью e=10^-6 Для решения задачи многомерной минимизации использовать метод Ньютона. Построить график функции f. Предусмотреть подсчет числа итераций, потребовавшихся для достижения заданной точности. a(11)=2.5, 2a(12)=1, a(22)=2, 2a(13)=-5, 2a(23)=-10.5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A11 | 2A12 | A22 | 2A13 | 2A23 | Метод |
| 2.5 | 1 | 2 | -5 | -10.5 | Наискорейший спуск |



**Рис. 7** *–* Вывод графика



**Рис. 8** *–* Результат работы алгоритма

**Вывод:** в ходе выполнения работы были сформированы 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 программы**

**Ex. 1**

import math, numpy

import matplotlib.pyplot as plt

f = lambda x: 2\*\*x - numpy.log(x)

a = 0.1

b = 3

eps = 10\*\*-6

df = lambda x: 2\*\*x\*numpy.log(2) - (1/x)

d2f = lambda x: 2\*\*x\*(numpy.log(2)\*\*2) + (1/x\*\*2)

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, int((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()

**Ex. 2**

import math

import matplotlib.pyplot as plt

# Определяем функцию

def f(x, n):

    return (-1)\*\*n \* (math.cos(2\*n\*x) / n\*\*3)

# Определяем метод минимизации деления отрезка пополам

def bisection\_method(f, a, b, eps):

    while (b - a) / 2 > eps:

        c = (a + b) / 2

        if f(c) == 0:

            return c

        elif f(c) \* f(a) < 0:

            b = c

        else:

            a = c

    return (a + b) / 2

# Задаем параметры

n = 250

a = 0

b = 3

eps = 0.0001

# Строим график функции

x\_values = [i / 100 for i in range(301)]

y\_values = [f(x, n) for x in x\_values]

plt.plot(x\_values, y\_values)

plt.xlabel('x')

plt.ylabel('f(x)')

plt.title('Graph of f(x)')

plt.show()

# Ищем минимумы и максимумы функции

min\_values = []

max\_values = []

for i in range(1, 301):

    a = (i - 1) / 100

    b = i / 100

    if f(a, n) < f(b, n):

        min\_value = bisection\_method(lambda x: f(x, n), a, b, eps)

        min\_values.append(min\_value)

    elif f(a, n) > f(b, n):

        max\_value = bisection\_method(lambda x: -f(x, n), a, b, eps)

        max\_values.append(max\_value)

# Выводим результаты

print('Минимумы функции:', min\_values)

print('Максимумы функции:', max\_values)

**Ex. 3**

import matplotlib.pyplot as plt

from matplotlib import cm

from matplotlib.ticker import LinearLocator

import numpy as np

import math

x\_1 = -2

x\_2 = 0

y\_1 = -1

y\_2 = 1

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 = 4\*X\*\*2 + Y\*\*2 + 3\* np.sin(X)-np.cos(Y+1)

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

                       linewidth=0, antialiased=False)

plt.show()

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()

**Ex. 4**

import numpy as np

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D

def f(x, y):

    a11 = 2.5

    a12 = 0.5

    a22 = 2

    a13 = -2.5

    a23 = -5.25

    return a11 \* x\*\*2 + 2 \* a12 \* x \* y + a22 \* y\*\*2 + 2 \* a13 \* x + 2 \* a23 \* y

def grad(x, y):

    a11 = 2.5

    a12 = 0.5

    a22 = 2

    a13 = -2.5

    a23 = -5.25

    grad\_x = 2 \* a11 \* x + 2 \* a12 \* y + 2 \* a13

    grad\_y = 2 \* a12 \* x + 2 \* a22 \* y + 2 \* a23

    return np.array([grad\_x, grad\_y])

def hessian(x, y):

    a11 = 2.5

    a12 = 0.5

    a22 = 2

    hess\_xx = 2 \* a11

    hess\_xy = 2 \* a12

    hess\_yy = 2 \* a22

    return np.array([[hess\_xx, hess\_xy], [hess\_xy, hess\_yy]])

def line\_search(x, y, grad, direction):

    t = 1.0

    alpha = 0.5

    beta = 0.8

    while f(x + t \* direction[0], y + t \* direction[1]) > f(x, y) + alpha \* t \* np.dot(grad, direction):

        t \*= beta

    return t

def newton\_method(x0, y0, epsilon):

    x\_current = np.array([x0, y0])

    iteration = 0

    while True:

        iteration += 1

        grad\_current = grad(x\_current[0], x\_current[1])

        hessian\_current = hessian(x\_current[0], x\_current[1])

        direction = -np.linalg.inv(hessian\_current) @ grad\_current

        t = line\_search(x\_current[0], x\_current[1], grad\_current, direction)

        x\_next = x\_current + t \* direction

        if np.linalg.norm(x\_next - x\_current) < epsilon:

            break

        x\_current = x\_next

    return x\_next, iteration

# Начальная точка

x0 = 0

y0 = 0

# Точность

epsilon = 1e-6

# Запуск метода Ньютона

solution, iteration = newton\_method(x0, y0, epsilon)

# Результат

x\_min, y\_min = solution

min\_value = f(x\_min, y\_min)

print("Минимум функции:")

print("x =", x\_min)

print("y =", y\_min)

print("f(x, y) =", min\_value)

print("Количество итераций:", iteration)

# Построение графика функции

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

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

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

Z = f(X, Y)

fig = plt.figure()

ax = plt.axes(projection='3d')

ax.plot\_surface(X, Y, Z, cmap='viridis')

ax.scatter(x\_min, y\_min, min\_value, color='red', label='Минимум')

ax.set\_xlabel('x')

ax.set\_ylabel('y')

ax.set\_zlabel('f(x, y)')

ax.legend()

plt.show()