Федеральное государственное автономное образовательное учреждение высшего образования
Национальный исследовательский университет
"Высшая школа экономики"

Московский институт электроники и математики имени А. Н. Тихонова
Программа "Прикладная математика"

Лабораторная работа №5

МИНИМИЗАЦИЯ ФУНКЦИЙ

Группа БПМ213
Вариант 8
Номера выполняемых задач: 9.1.8, 9.3.3, 9.5.8, 9.6.8

Выполнила:
Варфоломеева Анастасия Андреевна
Преподаватель:
Токмачев Михаил Геннадьевич

Москва, 2024 г.

1 Метод Ньютона

1.1 Формулировка задачи

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

Критерий останова:

$$\left| \frac{x^{(n+1)} - x^{(n)}}{1 - \left| \frac{x^{(n+1)} - x^{(n)}}{x^{(n)} - x^{(n-1)}} \right|} \right| < \varepsilon$$

$$f(x) = \sin x - 2\cos x$$

$$[a, b] = [1, 4]$$

1.2 Аналитическое решение

Найдем $f'(x)$ и ее нули на отрезке $[a, b] = [1, 4]$:

$$f'(x) = \cos x + 2\sin x$$

$$f'(x) = 0$$

$$\cos x + 2\sin x = 0$$

$$\tan x = -1/2$$

$$x_{max} = \arctan(-1/2) + \pi \approx 2.6779$$

Через вторую производную определяем чем является точка. Для этого представим функцию в виде $f(x) = \sqrt{5}\left(\cos\left(x - \arcsin\left(-\frac{1}{\sqrt{5}}\right)\right)\right)$. Так как $f''(x) < 0$ на отрезке, тогда $f(x)$ вогнута и $x_{max} = 2.6779$ - точка максимума функции $f(x)$ на $[1, 4]$, а также минимальное значение она принимает в одном из концов отрезка

$$f(1) = -0.2391$$

$$f(4) = 0.5505$$

Следовательно $x_{min} = 1$ - точка минимума $f(x)$ на $[1, 4]$.

1.3 Код на Python

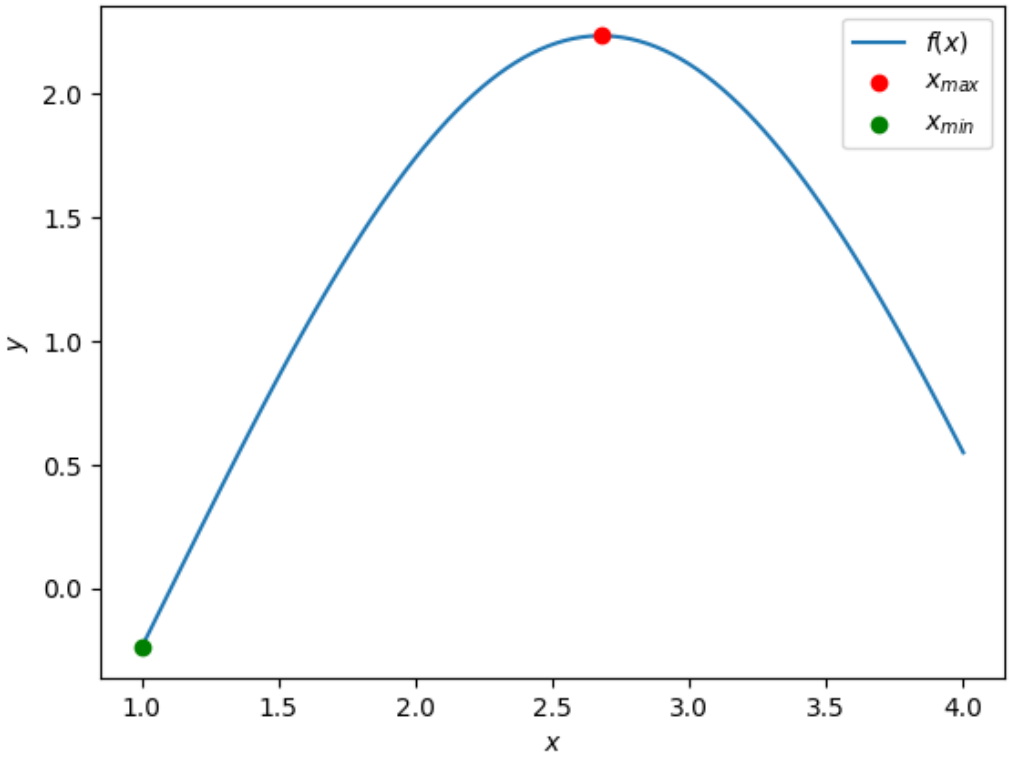
```
import numpy as np
import matplotlib.pyplot as plt

def f(x):
    return np.sin(x) - 2*np.cos(x)

def df(f, x, h=1e-4):
    return (f(x + h) - f(x - h))/2/h

def d2f(f, x, h=1e-4):
```

```
    return (f(x + 2 * h)- 2 * f(x) + f(x-2* h))/(4*h*h)

def newtons_step(f, x):
    return x - df(f, x)/d2f(f, x)

def p(x2, x1, x0):
    x2_x1 = np.abs(x2 - x1)
    x1_x0 = np.abs(x1 - x0)
    return np.abs(x2_x1 / (1 - x2_x1/ x1_x0))

def newtons_method(f, x0, eps=1e-6):
    x1 = newtons_step(f, x0)
    x2 = newtons_step(f, x1)
    i = 2
    while p(x2, x1, x0) > eps:
        x0 = x1
        x1 = x2
        x2 = newtons_step(f, x2)
        i += 1
    return x2, i

a = 1
b = 4
x_points = np.linspace(a, b, 100)
x1, i1 = newtons_method(f, 2.5)
x2, i2 = newtons_method(f,  3)
print(f'Start in x0 = {2.5}:\nFinds x_max = {x1:.6f} in {i1} iterations')
print(f'f(x_max) = {f(x1):.6f}')
print('------------------')
print(f'Start in x0 = {3}:\nFinds x_max = {x2:.6f} in {i2} iterations')
print(f'f(x_max) = {f(x2):.6f}')
print('------------------')
x_min = [a, b][np.argmin([f(a), f(b)])]
print(f'x_min = {x_min}')
print(f'f(x_min) = {f(x_min)}')
plt.plot(x_points, f(x_points), label="$f(x)$")
plt.scatter(x1, f(x1), label="$x_{max}$", c="r", zorder=10)
plt.scatter(x_min, f(x_min), label="$x_{min}$", c="g", zorder=10)
plt.xlabel("$x$")
plt.ylabel("$y$")
plt.legend()
plt.show()
```

1.4 Результат работы программы

```
Start in x0 = 2.5:
Finds x_max = 2.677945 in 3 iterations
f(x_max) = 2.236068
-----------------
Start in x0 = 3:
Finds x_max = 2.677945 in 3 iterations
f(x_max) = 2.236068
-----------------
x_min = 1
f(x_min) = -0.2391336269283828
```

1.5 График функции с отмеченными минимумом и максимумом



1.6 Вывод

В задаче представлен метод Ньютона для нахождения минимума и максимума унимодальной функции на отрезке. В приведенной задаче дана периодическая функция $f(x)$, то есть без удачного выбора начального приблежения метод может найти любой максимум функции. В связи с этим были рассмотрены два приближения: $x = 2.5$

и $x = 3$. Оба сошлись к $x_{max} = 2.6779$ - точке максимума, которая совпала с полученной в аналитическом решении. Максимум функции $f(x_{max}) = 2.2361$. Метод сошелся оба раза за 3 итерации. Минимум функции был найден исходя из вогнутости функции $f(x)$ на данном отрезке, сравненивая значения функции на его концах: $x_{min} = 1, f(x_{min}) = -0.2391$, что также совпало с аналитическим решением и видно из графика.

2 Метод минимизации Фибоначчи

2.1 Формулировка задачи

Функция $x(t)$ задана неявно уравнением $F(x,t) = 0, t_1 < t < t_2, x_1 < x < x_2$. Построить график зависимости функции $x(t)$ на заданном отрезке $[t_1, t_2]$ и найти ее минимум и максимум с точностью $\varepsilon = 10^{-6}$.

Метод минимизации - Фибоначчи

$$F(x,t) = e^{2x+t} - e^{t^2} - 2\cos x$$

$$0 < t < 2$$

$$0 < x < 2$$

2.2 Порядок решения задачи

1. Найти число N - количество итераций, блягодаря которому достигается необходимая точность $\varepsilon = 10^{-6}$ с помощью формулы $\frac{\Delta}{F_{N+1}} < \varepsilon$. В представленной задаче число Фибоначчи должно быть больше $2 * 10^6$, то есть $F_{31} = 2178309$
2. Выразить функцию явно $x(t)$.
3. Найти начальные точки отрезка

$$x_1 = a + (b-a)\frac{F_{n-2}}{F_n}, \quad x_2 = a + (b-a)\frac{F_{n-1}}{F_n}$$

и их значения в целевой функции $y_1 = f(x_1), y_2 = f(x_2)$.
4. Если $y_1 > y_2$, то $a = x_1, x_1 = x_2, x_2 = b - (x_1 - a), y_1 = y_2, y_2 = f(x_2)$
Иначе $b = x_2, x_2 = x_1, x_1 = a + (b - x_2), y_2 = y_1, y_1 = f(x_1)$
5. Производить вычисления до $N = 1$, тогда

$$x = \frac{x_1 + x_2}{2}$$

и остановка

2.3 Код на Python

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import fsolve

def fibonacci(n):
    if n in (0, 1):
        return 1
    return fibonacci(n - 1) + fibonacci(n - 2)
```

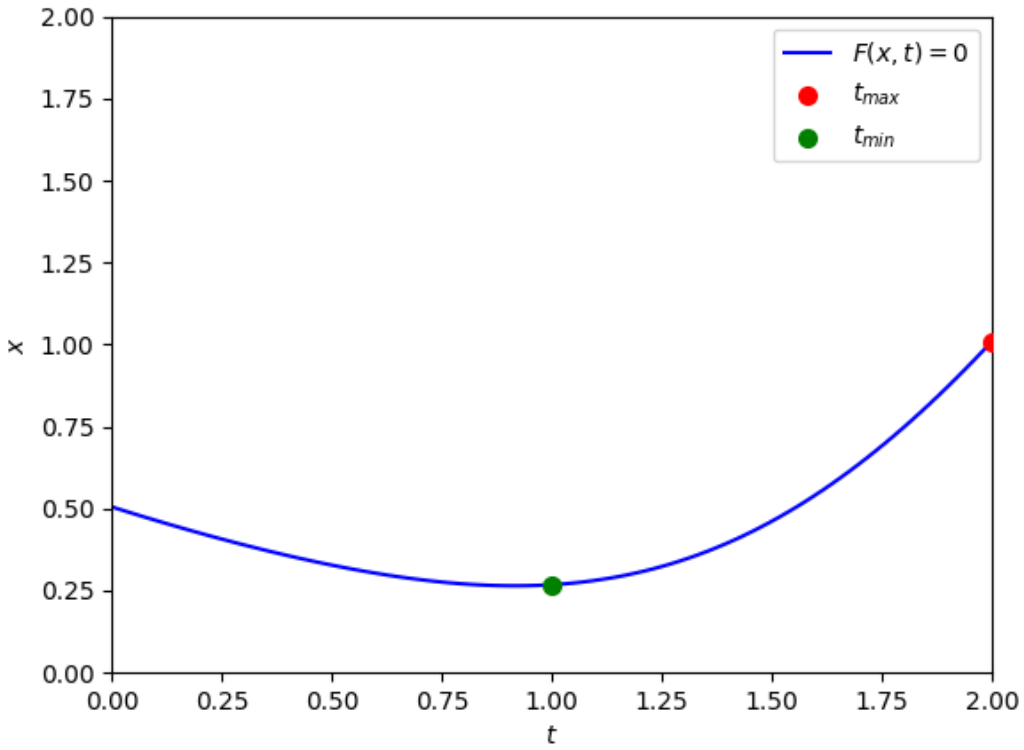
```
def f(x, t):
    return np.exp(2*x+t) - np.exp(t**2) - 2*np.cos(x)

def x(t):
    return  fsolve(lambda x: f(x, t), 1)[0]

def fib_meth(f, a, b, n, eps=1e-6):
    x1, x2 = a + (b - a)*fibonacci(n-2)/fibonacci(n), a + (b - a)*fibonacci(n-1)/fibo
    y1, y2 = f(x1), f(x2)
    for ind in range(1, n, -1):
        if y1 > y2:
            a = x1
            x1 = x2
            x2 = b - (x1 - a)
            y1 = y2
            y2 = f(x2)
        else:
            b = x2
            x2 = x1
            x1 = a + (b - x2)
            y2 = y1
            y1 = f(x1)
    return (x1 + x2)/2

t1, t2 =  0, 2
x1, x2 =  0, 2
t_points = np.linspace(t1, t2, 100)
x_points = np.linspace(x1, x2, 100)
X, T = np.meshgrid(x_points, t_points)
contour1 = plt.contour(T, X, f(X, T),[0], colors='b')
h1,_ = contour1.legend_elements()

eps = 10**(-6)
n = 0
while 2/fibonacci(n + 1) >= eps:
    n += 1
print(f'Accuracy achives with N = {n}: {2/fibonacci(31)} < {eps}')
print('-----------------')

t_min = fib_meth(x, t1, t2, n)
print(f"t_min = {t_min:.4f}")
print(f"x(t_min) = {x(t_min):.4f}")
print('-----------------')
t_max = [t1, t2][np.argmax([x(t1), x(t2)])]
print(f"t_max = {t_max:.4f}")
print(f"x(t_max) = {x(t_max):.4f}")
p1 = plt.scatter(t_max, x(t_max), c="r", s=49, zorder=10, label="$t_{max}$")
p2 = plt.scatter(t_min, x(t_min), c="g", s=49, zorder=10, label="$t_{min}$")
plt.legend([h1[0], p1, p2], ['$F(x,t)=0$', '$t_{max}$', "$t_{min}$"])
```

```
plt.xlabel("$t$")
plt.ylabel("$x$")
plt.show()
```

2.4 Результат работы программы

```
Accuracy achives with N = 30: 9.1814338553437555e-07 < 1e-06
-----------------
t_min = 1.000000
x(t_min) = 0.268093
-----------------
t_max = 2.000000
x(t_max) = 1.009653
```

2.5 График функции с отмеченными минимумом и максимумом



2.6 Вывод

В данной задаче представлен метод минимизации Фибоначчи. Метод обеспечивает максимальное гарантированное сокращение отрезка локализации при заданном числе N вычисления функции и основан на использовании чисел Фибоначчи. В конкретно этой задаче была дана точность, с которой необходимо найти решение, то есть сначала нужно было посчитать необходимое количество итераций по формуле

$$\frac{\Delta}{F_{N+1}} < \varepsilon$$

Как и в аналитическом решении точность достигается при $N = 30$. Найден минимум $t_{min} = 1, \quad x(t_{min}) = 0.2681$. Максимум находился сравнением значений на концах и равен $t_{max} = 2 \quad x(t_{max}) = 1.0097$. Эти же данный видно на представленном графике.

3 Минимум функции 2-х переменных

3.1 Формулировка задачи

Найти минимум функции 2-х переменных $f(x, y)$ с точностью $\varepsilon = 10^{-6}$ на прямоугольнике $[x_1, x_2] \times [y_1, y_2]$.

3.2 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. Используя встроенные функции, найти экстремумы функции с заданной точностью.

$$f(x, y) = x^2 + 2y^2 + e^x \cos(y - 1)$$
$$[x_1, x_2] \times [y_1, y_2] = [-2, 1] \times [-1, 1]$$

3.3 Аналитическое решение

Градиент функции $f(x, y)$:

$$\nabla f(x, y) = \begin{pmatrix} 2x + e^x \cos(y - 1) \\ 4y - e^x \sin(y - 1) \end{pmatrix}$$

Приравняем градиент к нулю:

$$\begin{cases} 2x + e^x \cos(y - 1) = 0 \\ 4y - e^x \sin(y - 1) = 0 \end{cases}$$

Решение этой системы уравнений $x_{min} = (-0.1553, -0.1994)$, принадлежащая прямоугольнику $[-2, 1] \times [-1, 1]$. $f(x_{min}) = 0.4144$
Найдем матрицу Гессе функции $f(x, y)$:

$$f'' = \begin{pmatrix} 2 + e^x \cos(y - 1) & -e^x \sin(y - 1) \\ -e^x \sin(y - 1) & 4 - e^x \cos(y - 1) \end{pmatrix}$$

Она является положительно определенной, следовательно x_{min} - точка минимума функции $f(x, y)$.

3.4 Код на Python

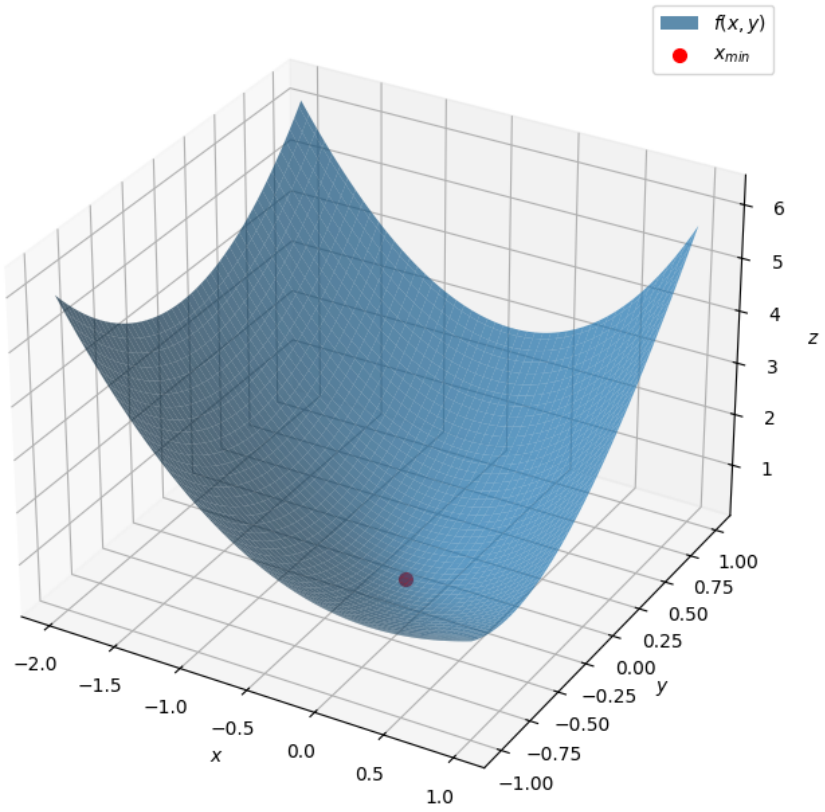
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import minimize
```

```
def f(x):
    x, y = x
    return  x ** 2 + 2 * (y**2) + np.exp(x) * np.cos(y-1)
x1, x2 = -2, 1
y1, y2 = -1, 1

x_points = np.linspace(x1, x2, 100)
y_points = np.linspace(y1, y2, 100)
r1 = minimize(f, [-1.5, -1], tol=1e-6)
r2 = minimize(f, [1, 1], tol=1e-6)
X, Y = np.meshgrid(x_points, y_points)
fig = plt.figure(figsize=(10, 18))
ax_3d = fig.add_subplot(projection='3d')
ax_3d.plot_surface(X, Y, f([X,Y]), alpha=0.7, label="$f(x, y)$")
ax_3d.scatter(*r1.x, r1.fun, s=49, c="r", alpha=1, label="$x_{min}$")
ax_3d.set_xlabel("$x$")
ax_3d.set_ylabel("$y$")
ax_3d.set_zlabel("$z$")
plt.legend()
plt.show()
plt.contour(X, Y, f([X, Y]), 20)
plt.colorbar()
plt.scatter(*r1.x, c="r", label="$x_{min}$", zorder=10)
plt.xlabel("$x$")
plt.ylabel("$y$")
plt.legend()
plt.show()
with np.printoptions(6):
    print("Start in x0 = [-1.5, -1]")
    print(f"Finds x_min = {r1.x} in {r1.nit} iterations")
    print(f"f(x_min) = {r1.fun:.6f}")
    print('----------------')
    print("Start in x0 = [1, 1]")
    print(f"Finds x_min = {r2.x} in {r2.nit} iterations")
    print(f"f(x_min) = {r1.fun:.6f}")
    print('----------------')
```

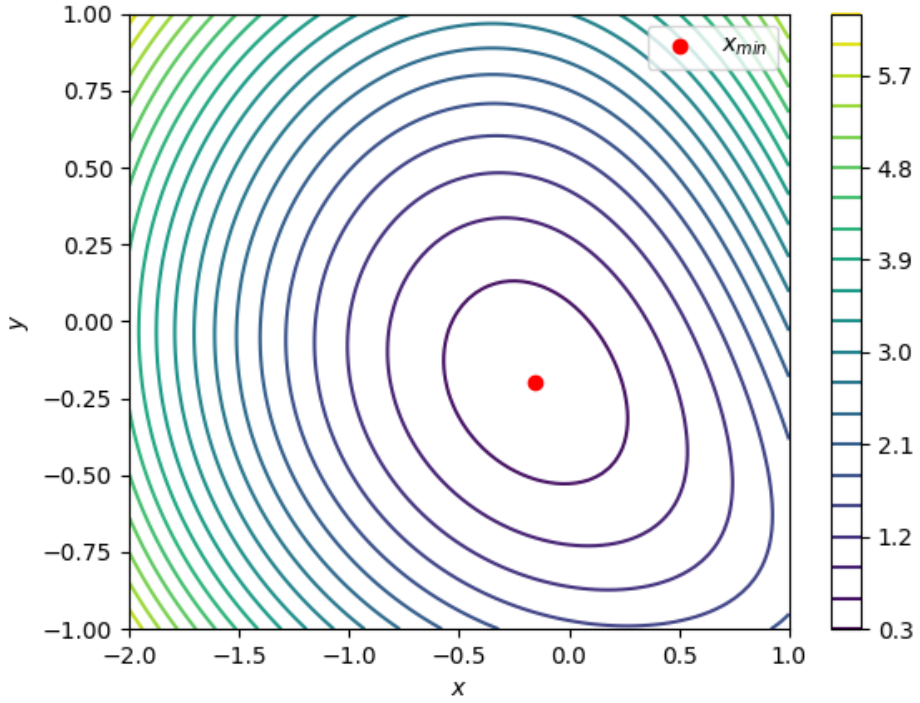
3.5 Результат работы программы

```
Start in x0 = [-1.5, -1]
Finds x_min = [-0.155335 -0.199443] in 6 iterations
f(x_min) = 0.414353
----------------
Start in x0 = [1, 1]
Finds x_min = [-0.155335 -0.199443] in 7 iterations
f(x_min) = 0.414353
----------------
```

3.6 График функции и ее минимум



3.7 График поверхностей уровней и минимум



3.8 Вывод

В данной задаче производился поиск минимума функции $f(x)$ в заданном квадрате. С помощью встроенной функции для двух разных приближений (-1.5, -1) и (1, 1) был найден минимум $x_{min} = (-0.1553, -0.1994)$ за 6 и 7 итераций соответственно. Значение функции в точке равно $f(x_{min} = 0.4144$. Такой же ответ был получен в аналитическом решении. Также были построены графики функции и поверхностей уровней из который тоже видно, что минимум находится в полеченной точке.

4 Минимум квадратичной функции

4.1 Формулировка задачи

Указанным в индивидуальном варианте методом 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 $\varepsilon = 10^{-6}$. Для решения задачи одномерной минимизации использовать метод Ньютона. Построить график функции f. Предусмотреть подсчет числа итераций, потребовавшихся для достижения заданной точности.

Метод сопряженных градиентов

$$f(x, y) = x^2 + 0.5xy + 2.5y^2 - 3.5x - 6.5y$$

4.2 Аналитическое решение

Градиент функции $f(x, y)$:

$$\nabla f(x, y) = \begin{pmatrix} 2x + 0.5y - 3.5 \\ 0.5x + 5y - 6.5 \end{pmatrix}$$

Приравняем градиент к нулю:

$$\begin{cases} 2x + 0.5y - 3.5 = 0 \\ 0.5x + 5y - 6.5 = 0 \end{cases}$$

Решение этой системы уравнений $x_{min} = (1.46, 1.15)$. $f(x_{min}) = -6.3077$
Найдем матрицу Гессе функции $f(x, y)$:

$$f'' = \begin{pmatrix} 2 & 0.5 \\ 0.5 & 5 \end{pmatrix}$$

Эта матрица является положительно определенной, следовательно $x_{min}-$ точка минимума функции $f(x, y)$.

4.3 Код на Python

```
import numpy as np
import matplotlib.pyplot as plt

a11 = 1
a12 = 0.5
a22 = 2.5
```

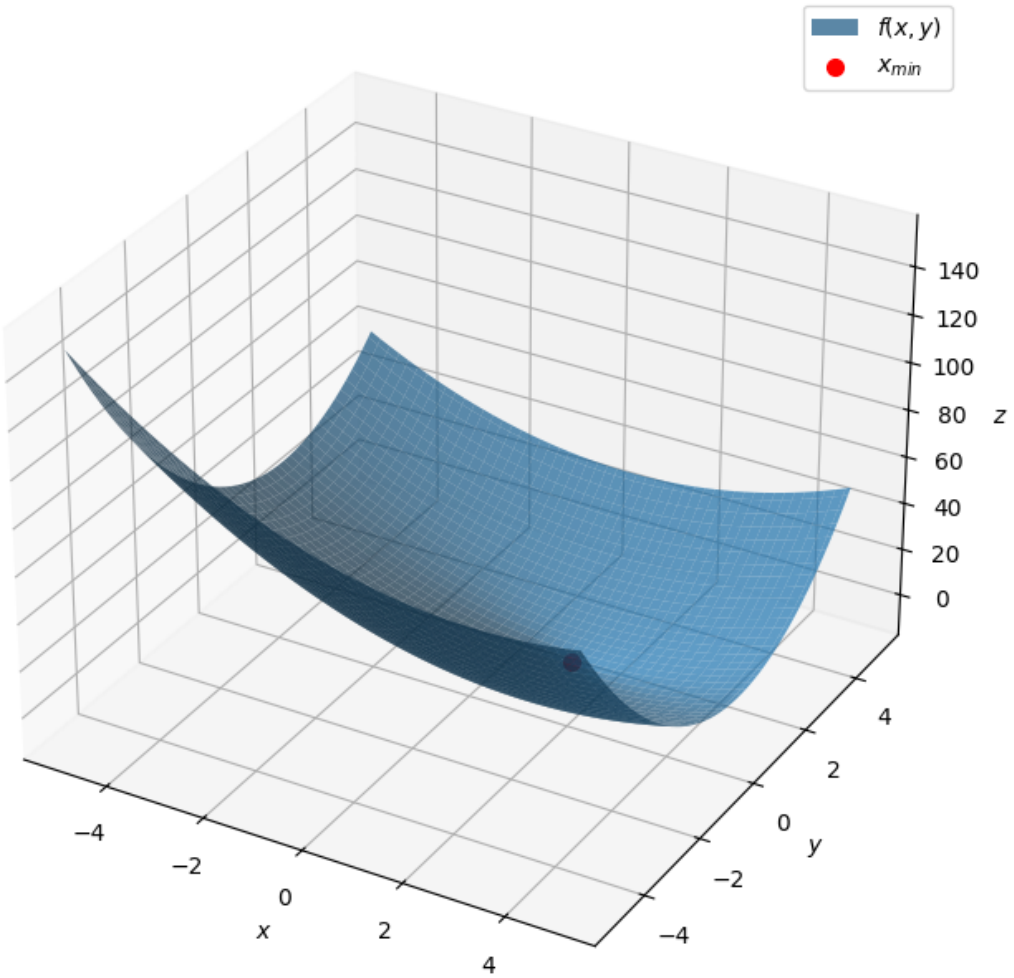
```
a13 = -3.5
a23 = -6.5
def f(x):
    x, y = x
    return (
        a11 * x**2 + a12 * x * y + a22 * y**2 + a13 * x + a23 * y
    )

def grad(f, x, h = 1e-4):
    x, y = x
    return np.array(
        [
            (f([x + h, y]) - f([x - h, y]))/ h / 2,
            (f([x, y + h]) - f([x, y - h]))/ h / 2
        ]
    )
def df(f, x, h=1e-4):
    return (f(x + h) - f(x - h))/2/h

def d2f(f, x, h=1e-4):
    return (f(x + 2 * h)- 2 * f(x) + f(x-2* h))/(4*h*h)

def newtons_step(f, x):
    return x - df(f, x)/d2f(f, x)

def p(x2, x1, x0):
    x2_x1 = np.abs(x2 - x1)
    x1_x0 = np.abs(x1 - x0)
    return np.abs(x2_x1 / (1 - x2_x1/ x1_x0))

def newtons_method(f, x0 = 1.0, eps = 1e-6):
    x1 = newtons_step(f, x0)
    x2 = newtons_step(f, x1)
    while p(x2, x1, x0)>eps:
        x0 = x1
        x1 = x2
        x2 = newtons_step(f, x2)
    return x2


def congruent_gradients(f, x0, eps=1e-6):
    s = -grad(f, x0)
    prev_grad = s
    x = x0.copy()
    i = 0
    while np.max(np.abs(-grad(f, x))) > eps:
        i += 1
        def foo(a):
            return f(x + a * s)
        a = newtons_method(foo)
```

```
        x += a * s
        new_grad = -grad(f, x)
        s = new_grad + np.sum(new_grad **2)/np.sum(prev_grad ** 2) * s
        prev_grad = new_grad
    return x, i


x_points = np.linspace(-5, 5, 100)
y_points = np.linspace(-5, 5, 100)
X, Y = np.meshgrid(x_points, y_points)
fig = plt.figure(figsize=(8, 16))
ax_3d = fig.add_subplot(projection='3d')
x01 = np.array([1.5, -3.0])
x02 = np.array([-40.0, 60.0])
x_min1, iter1 = congruent_gradients(f, x01)
x_min2, iter2 = congruent_gradients(f, x02)
ax_3d.plot_surface(X, Y, f([X,Y]), alpha=0.7, label="$f(x,y)$")
ax_3d.scatter(*x_min1, f(x_min1), s=49, c="r", alpha=1, label="$x_{min}$")
ax_3d.set_xlabel("$x$")
ax_3d.set_ylabel("$y$")
ax_3d.set_zlabel("$z$")
ax_3d.legend()
plt.show()
plt.contour(X, Y, f([X, Y]), 20)
plt.colorbar()
plt.scatter(*x_min1, c="r", label="$x_{min}$", zorder=10)
plt.legend()
plt.xlabel("$x$")
plt.ylabel("$y$")
plt.show()
with np.printoptions(6):
    print("Start in x0 = [1.5, -3]")
    print(f"Finds x_min = {x_min1} in {iter1} iterations")
    print(f"f(x_min) = {f(x_min1):.4f}")
    print('----------------')
    print("Start in x0 = [-40, 60]")
    print(f"Finds x_min = {x_min2} in {iter2} iterations")
    print(f"f(x_min) = {f(x_min2):.4f}")
    print('----------------')
```

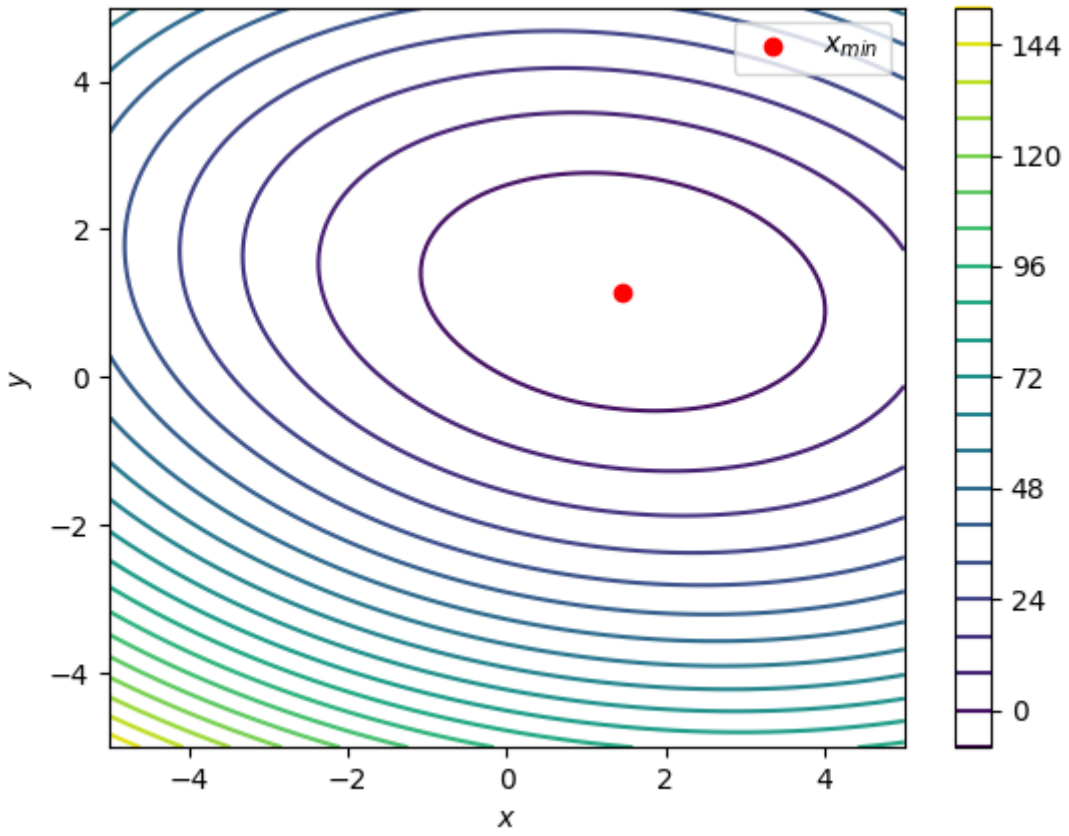
4.4 Результат работы программы

```
Start in x0 = [1.5, -3]
Finds x_min = [1.461538 1.153846] in 2 iterations
f(x_min) = -6.307692
----------------
Start in x0 = [-40, 60]
Finds x_min = [1.461538 1.153846] in 2 iterations
f(x_min) = -6.307692
----------------
```

4.5 График функции и ее минимум



4.6 График поверхностей уровней и минимум



4.7 Вывод

В представленной работе был реализован метод сопряженных градиентов для решения задачи одномерной минимизации при заданной точности. Благодаря тому, что использовался метод сопряженных градиентов, при разных приближениях ответ был найдет за одинаковое количество итераций, а именно - за размерность пространства, в котором работаем (метод сходится не более чем за размерность пространства итераций). Было выбрано два приближения: (1.5, -3) и (-40, 60) и они сошлись за 2 итерации к минимуму $x_{min} = (1.4615, 1.1538)$. Значение функции в этой точке равно $f(x_{min}) = -6.3077$. Ответ сходится с тем, что был получен аналитически. Также были построены графики функции и поверхностей уровней из который тоже видно, что минимум находится в полеченной точке.