|  |  |
| --- | --- |
| **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 №2**

**«Графический метод решения задачи математического программирования»**

**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:** изучение математического аппарата математического программирования на примере задач небольшой размерности, допускающих графическое решение.

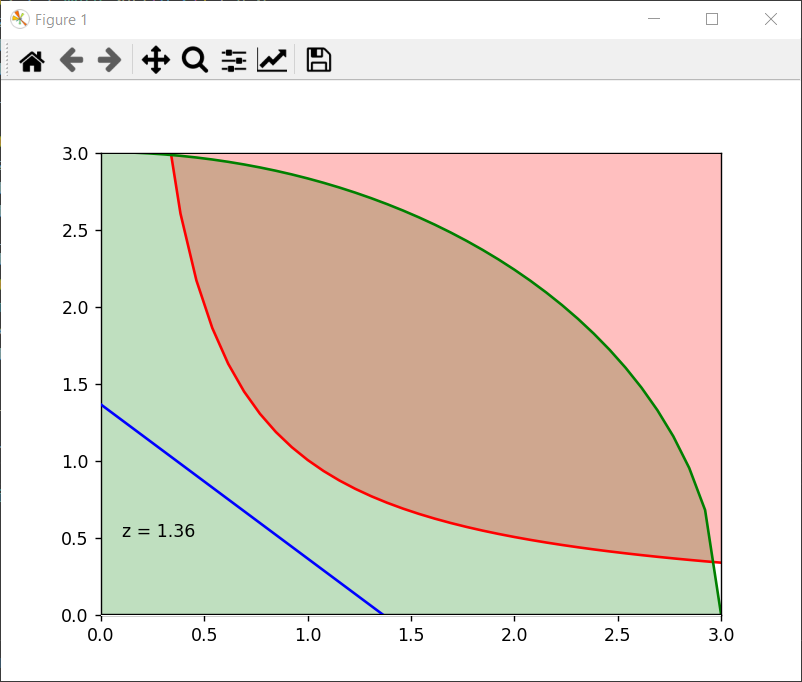
**Задачи:** представить графическое решение, реализованное на языке высокого уровня.

**Вариант 7**

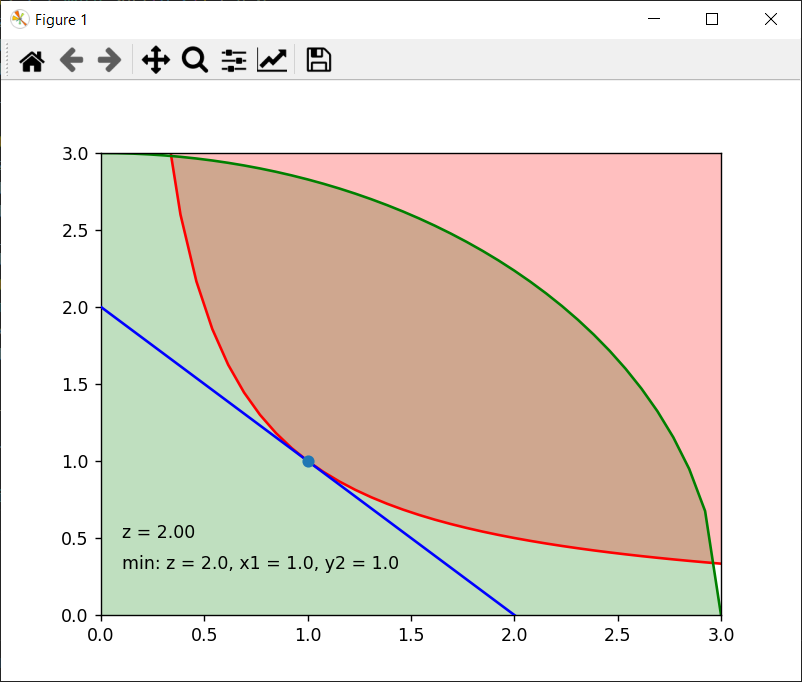
**Задание 1.**

Решить задачу нелинейного программирования графическим методом:

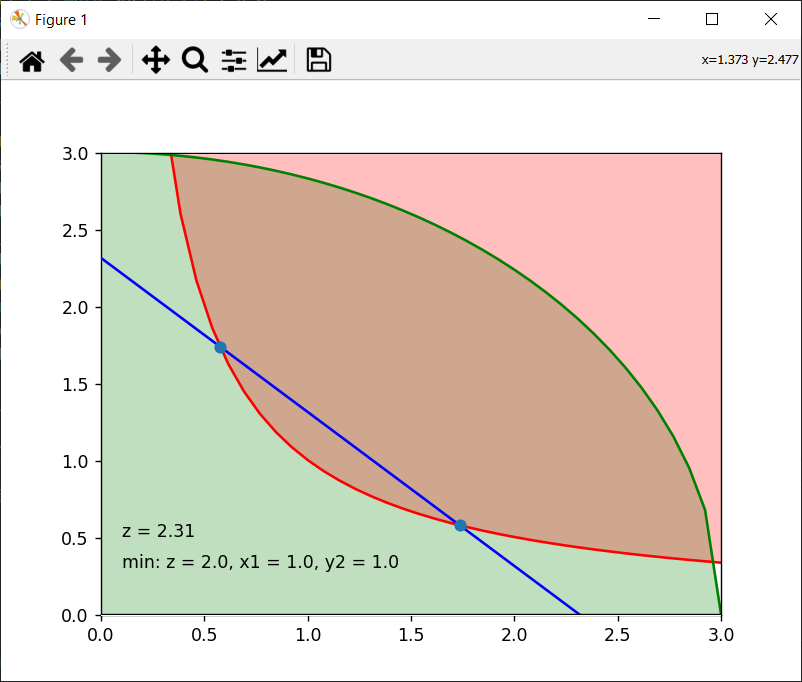
**Решение:**



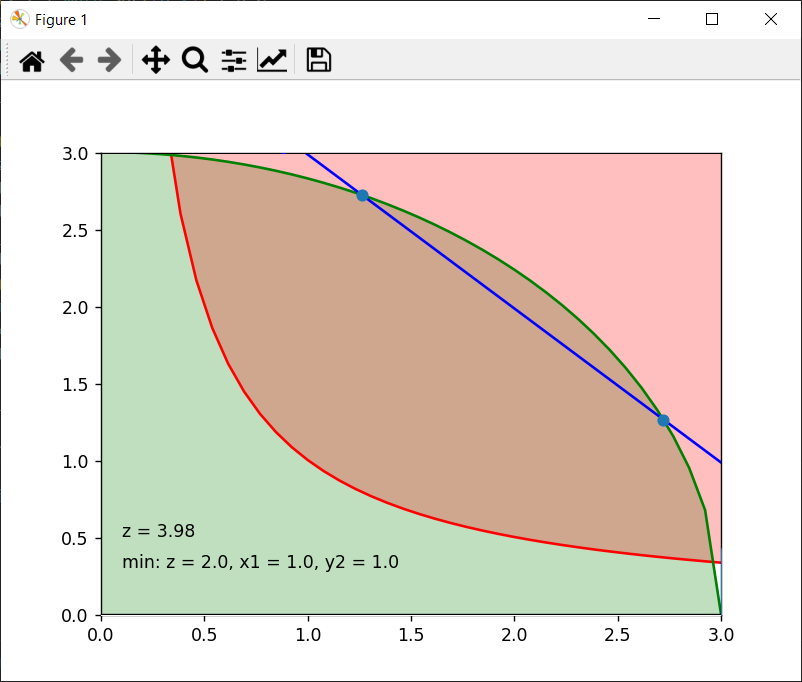
**Рис. 1.1.** Решение



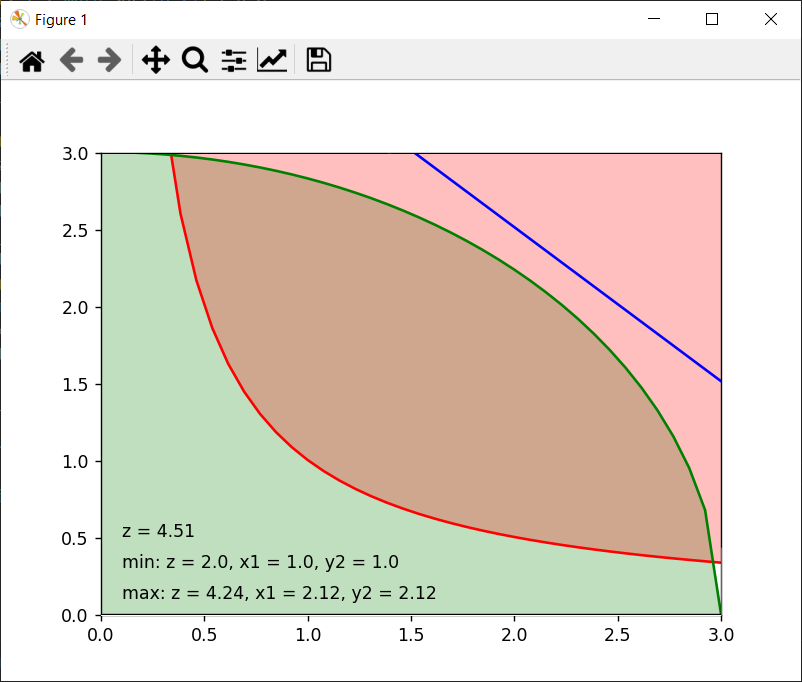
**Рис. 1.2.** Решение



**Рис. 1.3.** Решение



**Рис. 1.4.** Решение

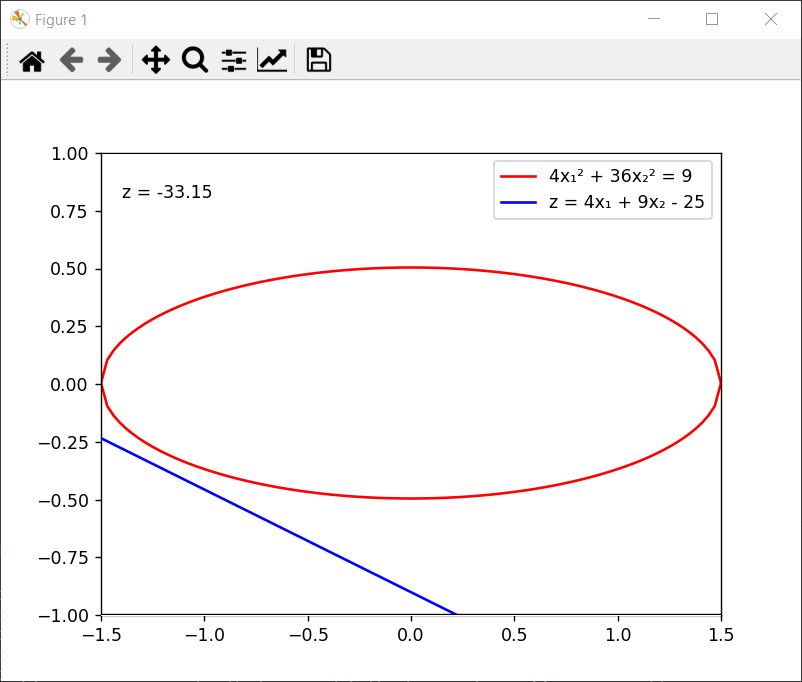


**Рис. 1.5.** Решение

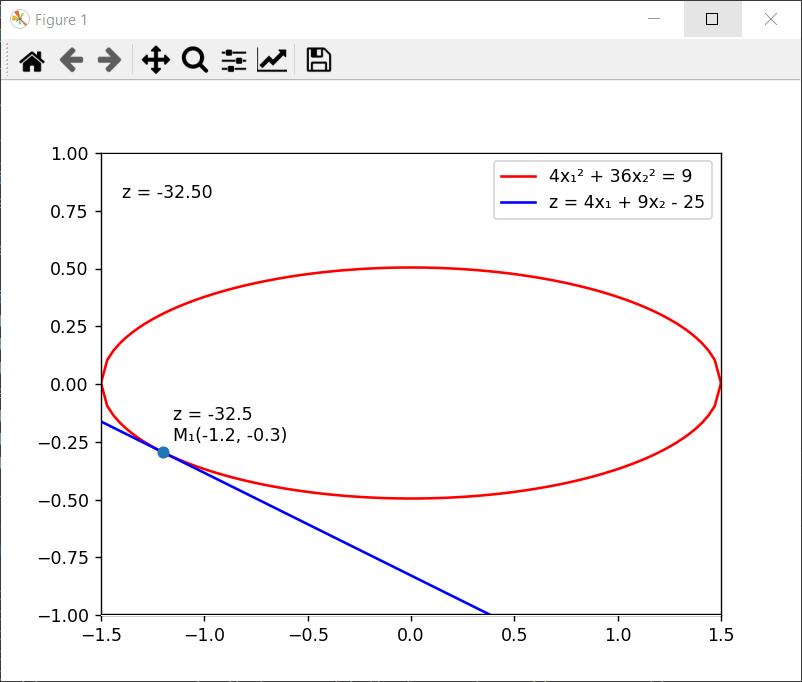
**Задание 2.**

Найти условный экстремум функции методом множителей Лагранжа:

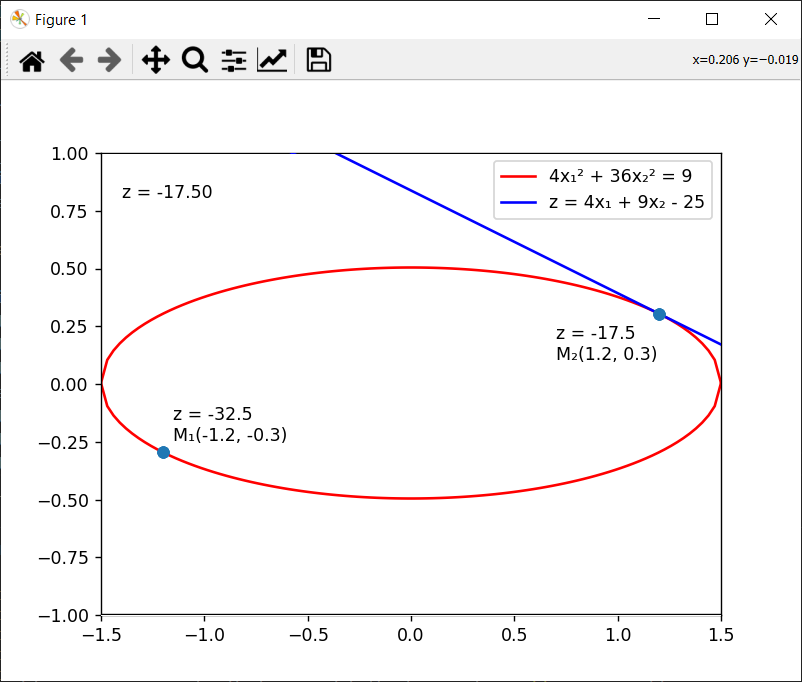
**Решение:**



**Рис. 2.1.** Решение



**Рис. 2.2.** Решение



**Рис. 2.3.** Решение

**Вывод:** в ходе выполнения лабораторной работы был изучен математический аппарат математического программирования на примере задач небольшой размерности, допускающих графическое решение.

**ПРИЛОЖЕНИЯ**

**Листинг:**

***LW2\_1.py:***

import numpy as np

import matplotlib.pyplot as plt

from matplotlib.animation import FuncAnimation

import math

X = np.linspace(0, 3, 40)

Y\_1 = 1 / X

Y\_2 = np.sqrt(9 - X\*\*2)

red\_fill\_x = [X[-1]]

for x in X:

    red\_fill\_x.append(x)

red\_fill\_x.append(X[-1])

red\_fill\_y = [Y\_1[1]]

for y in Y\_1:

    red\_fill\_y.append(y)

red\_fill\_y.append(Y\_1[1])

green\_fill\_x = [X[0]]

for x in X:

    green\_fill\_x.append(x)

green\_fill\_x.append(X[0])

green\_fill\_y = [Y\_2[-1]]

for y in Y\_2:

    green\_fill\_y.append(y)

green\_fill\_y.append(Y\_2[-1])

fig, ax = plt.subplots()

ln, = ax.plot([], [], 'b')

dots, = ax.plot([], [], 'o')

min\_text = ax.text(0.1, 0.3, '', fontsize=10)

max\_text = ax.text(0.1, 0.1, '', fontsize=10)

z\_text = ax.text(0.1, 0.5, '', fontsize=10)

ax.fill(green\_fill\_x, green\_fill\_y, 'green', alpha=0.25)

ax.fill(red\_fill\_x, red\_fill\_y, 'red', alpha=0.25)

ax.plot(X, Y\_1, 'r')

ax.plot(X, Y\_2, 'g')

z = 0

last\_x\_points = None

def init():

    ax.set\_xlim(0, 3)

    ax.set\_ylim(0, 3)

    return ln, dots, min\_text, max\_text, z\_text,

def update(frame):

    global z

    global last\_x\_points

    ydata = frame - X

    ln.set\_data(X, ydata)

    z = frame

    z\_text.set\_text(f'z = {z:.2f}')

    x\_dots = []

    y\_dots = []

    D\_1 = z\*\*2 - 4

    if D\_1 == 0:

        x = z / 2.0

        y = z - x

        x\_dots.append(x)

        y\_dots.append(y)

        min\_text.set\_text(f'min: z = {z}, x1 = {x}, y2 = {y}')

    elif D\_1 > 0:

        x = (z + math.sqrt(D\_1)) / 2.0

        y = z - x

        x\_dots.append(x)

        y\_dots.append(y)

        x = (z - math.sqrt(D\_1)) / 2.0

        y = z - x

        x\_dots.append(x)

        y\_dots.append(y)

    D\_2 = 72 - 4\*z\*\*2

    if D\_2 == 0:

        x = 2\*z / 4.0

        y = z - x

        x\_dots.append(x)

        y\_dots.append(y)

        max\_text.set\_text(f'max: z = {z}, x1 = {x}, y2 = {y}')

    elif D\_2 > 0:

        last\_x\_points = []

        x = (2\*z + math.sqrt(D\_2)) / 4.0

        y = z - x

        x\_dots.append(x)

        y\_dots.append(y)

        last\_x\_points.append(x)

        x = (2\*z - math.sqrt(D\_2)) / 4.0

        y = z - x

        x\_dots.append(x)

        y\_dots.append(y)

        last\_x\_points.append(x)

    elif last\_x\_points != None:

        x = (last\_x\_points[0] + last\_x\_points[1]) / 2.0

        y = math.sqrt(9 - x\*\*2)

        z = x + y

        max\_text.set\_text(f'max: z = {z:.2f}, x1 = {x:.2f}, y2 = {y:.2f}')

    dots.set\_data(x\_dots, y\_dots)

    return ln, dots, min\_text, max\_text, z\_text

ani = FuncAnimation(fig, update, frames=np.linspace(1, 5, 501),

                    init\_func=init, blit=True, interval=10)

plt.show()

***LW2\_2.py:***

import numpy as np

import matplotlib.pyplot as plt

from matplotlib.animation import FuncAnimation

X = np.linspace(-1.5, 1.5, 100)

Y\_1 = np.sqrt(9 - 4\*X\*\*2) / 6

Y\_2 = -np.sqrt(9 - 4\*X\*\*2) / 6

fig, ax = plt.subplots()

ax.plot(X, Y\_1, 'r', label='4x₁² + 36x₂² = 9')

ax.plot(X, Y\_2, 'r')

z\_line, = ax.plot([], [], 'b', label='z = 4x₁ + 9x₂ - 25')

dots, = ax.plot([], [], 'o')

z\_text = ax.text(-1.4, 0.8, '', fontsize=10)

min\_text = ax.text(-1.15, -0.25, '', fontsize=10)

max\_text = ax.text(0.7, 0.1, '', fontsize=10)

x\_dots = []

y\_dots = []

min\_is\_placed = False

max\_is\_placed = False

def init():

    ax.set\_xlim(-1.5, 1.5)

    ax.set\_ylim(-1.0, 1.0)

    return z\_line, dots, z\_text, min\_text, max\_text,

def update(frame):

    global x\_dots

    global y\_dots

    global min\_is\_placed

    global max\_is\_placed

    z\_line.set\_data(X, (frame - 4\*X + 25) / 9)

    if not min\_is\_placed and frame == -32.5:

        x\_dots.append(-1.2)

        y\_dots.append(-0.3)

        min\_text.set\_text(f'z = -32.5\nM₁(-1.2, -0.3)')

    if not max\_is\_placed and frame == -17.5:

        x\_dots.append(1.2)

        y\_dots.append(0.3)

        max\_text.set\_text(f'z = -17.5\nM₂(1.2, 0.3)')

    dots.set\_data(x\_dots, y\_dots)

    z\_text.set\_text(f'z = {frame:.2f}')

    return z\_line, dots, z\_text, min\_text, max\_text,

ani = FuncAnimation(fig, update, frames=np.linspace(-35, -15, 401),

                    init\_func=init, blit=True, interval=5)

ax.legend()

plt.show()