

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
 «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Факультет «Информатика и вычислительная техника»

Кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»

|  |  |
| --- | --- |
| Заведующий кафедрой | «ПОВТ и АС» |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | В.В. Долгов |

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_ г.

**ОТЧЕТ**

по лабораторно-практической работе по дисциплине «Исследование операций» по кафедре «Программное обеспечение вычислительной техники и автоматизированных системы»

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_А. А. Кизогян\_\_\_

подпись, дата

Обозначение отчета УП.81.0000.000 Группа \_\_\_\_ВМО31\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Направление 02.03.03 Математическое обеспечение и администрирование информационных систем

Профиль Математическое обеспечение и администрирование информационных систем

Преподаватель: проф. Никитина Алла Валерьевна

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата подпись преподавателя

Ростов-на-Дону

2023

# Содержание

1. Индивидуальная работа №1 3

1.1 Аналитическое решение 3

1.2 Решение программным средством 4

1.3 Вывод 5

1. Индивидуальная работа №2 5

2.1 Аналитическое решение 5

2.2 Решение программным средством 7

2.3 Вывод 8

1. Индивидуальная работа №3 8

1.1 Аналитическое решение 8

1.2 Решение программным средством 9

1.3 Вывод 10

1. Приложение А.1 листинг основной программы индивидуальной работы № 1 11
2. Приложение А.2 листинг подпрограммы индивидуальной работы № 1 12
3. Приложение Б листинг программы индивидуальной работы № 2 13
4. Приложение В листинг программы индивидуальной работы № 3 18

**Индивидуальное задание № 1**

**Цель работы: найти точки экстремума функции**

**Задание:** Вариант 13

**Аналитическое решение**

Найдем частные производные

Производные по x и по y приравниваем к 0 и находим точки

решим систему уравнений

Получим значения

– получаем через дискриминант

Из найденного следует, что существует 2 критические точки:

Найдем частные производные второго порядка:

; ;

Вычислим значения для точки :

и , поэтому в точке имеется минимум .

Вычислим значения для точки :

, поэтому глобального экстремума нет.

Ответ: в точке имеется минимум .

**Решение программным средством**

Благодаря программному средству, можно найти экстремумы функции . Описание программы приведено в Приложении А.1 и А.2. Решение и вывод представлены на рисунках 1 и 2.

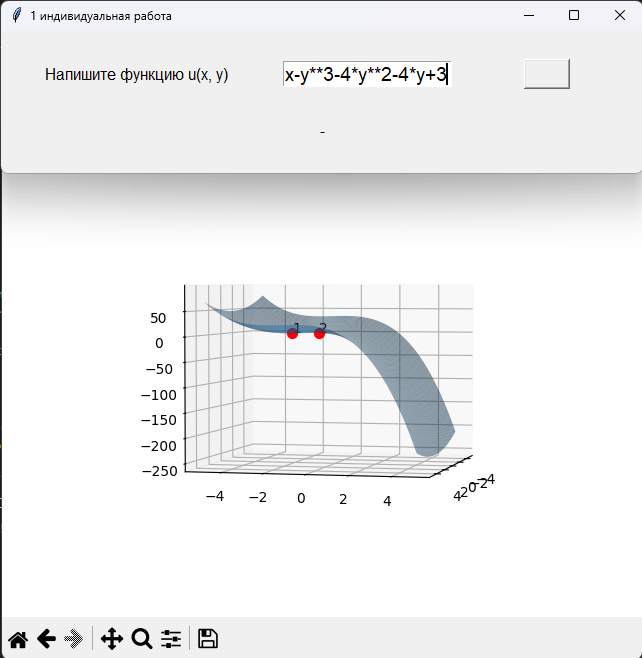


Рисунок 1- Программная реализация и результат работы программы

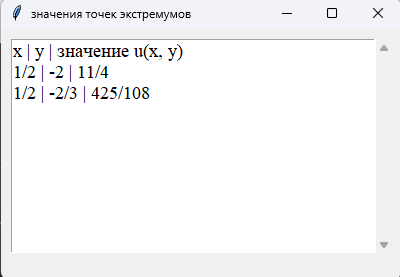


Рисунок 2- Программная реализация и результат работы программы

**Вывод**

Нахождение экстремумов функции является сложной задачей, требующей применения математических методов и инструментов. Она позволяет определить точки, в которых функция достигает минимальных или максимальных значений, что важно для решения многих практических задач.

**Индивидуальное задание № 2**

**Цель работы:** Найдите наибольшее и наименьшее значения функции в области (), заданной указанными неравенствами

**Задание:** Вариант 13

**Аналитическое решение**

Найдем частные производные

Производные по x и по y приравниваем к 0 и находим точки

решим систему уравнений

Получим значения

- существует первая критическая точка, которая не принадлежит из-за условия

Примем значения ограничений:

Существует следующая точка

Но по условию,

Существует следующая точка

Существует следующая точка

Но по условию,

; ; ;

На рисунке 3 изображен график данной функции

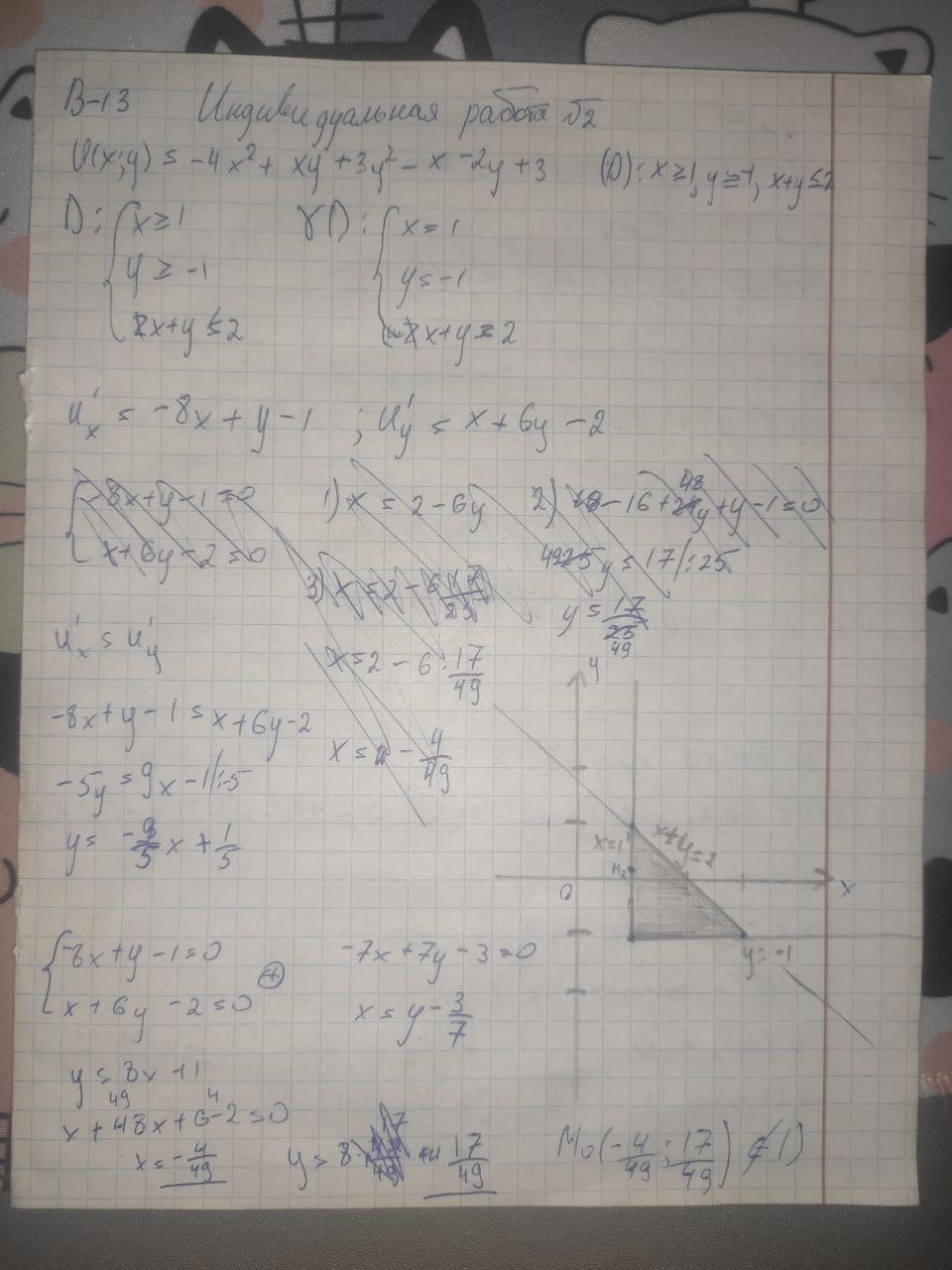


Рисунок 3 – График функции 13 варианта

Ответ:

;

.

**Решение программным средством**

Благодаря программному средству, можно найти наибольшие и наименьшие значения функции при заданных условиях . Описание программы приведено в Приложении Б. Решение и вывод представлены на рисунках 4, 5 и 6.

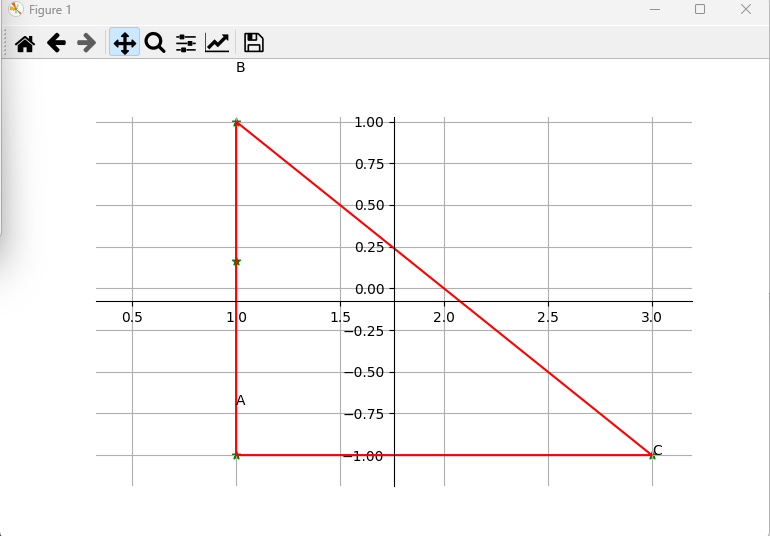


Рисунок 4- График функции, найденный программным средством



Рисунок 5 - Критические точки, найденные программным средством

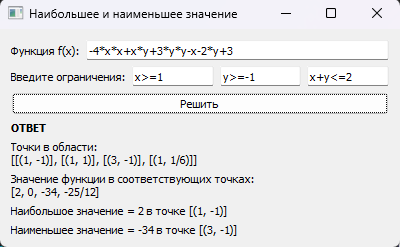
.

Рисунок 6 – Программная реализация и результат работы программы

**Вывод**

В области заданной неравенствами функция может принимать различные значения. Ищем наибольшее и наименьшее значение функции, чтобы определить ее границы в данной области. Для этого необходимо проанализировать все точки, удовлетворяющие неравенствам в области и вычислить значения функции в этих точках. Наибольшее значение будет соответствовать максимальной величине функции, а наименьшее - минимальной.

**Индивидуальное задание № 3**

**Цель работы:** найти условный экстремум функции при заданном уравнении связи.

**Задание:** Вариант 13

**Аналитическое решение**

Находим первые производные в уравнении связи, пригодится в конце для матрицы:

;

Перепишем ограничение задачи в неявном виде:

Составим вспомогательную функцию Лагранжа:

Найдем его первые производные:

Построим систему и приравняем ее к нулю:

В первой системе, первую строку умножаем на 4 и вычитаем вторую строку из первой. Во второй системе все также, только без домножения.

Получим:

Решаем систему и получим значения

; ;

Получим стационарную точку

Находим вторые производные от функции Лагранжа:

; ;

Строим матрицу:

Из этого следует, что в точке функция имеет условный минимум.

**Решение программным средством**

Благодаря программному средству, можно найти условный экстремум функции при заданном уравнении связи. Описание программы приведено в Приложении В. Решение и вывод представлены на рисунках 7, 8.

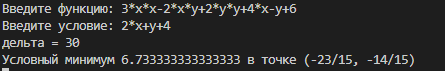


Рисунок 7 - Программная реализация и результат работы программы

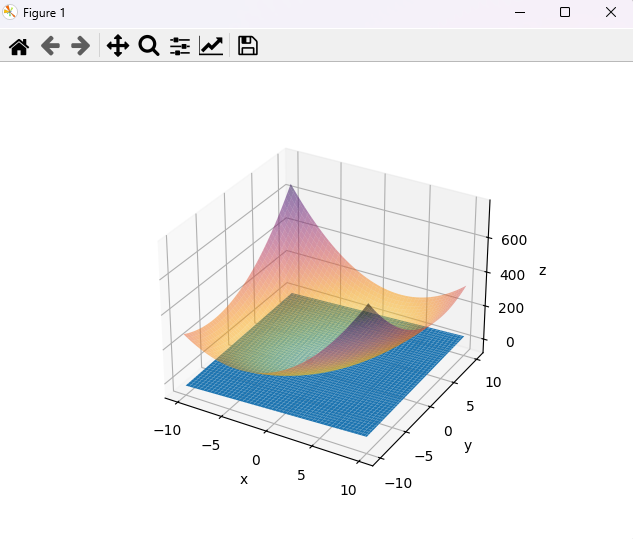


Рисунок 8 – графики функции и уравнения связи

**Вывод**

При поиске условного экстремума функции при заданном уравнении связи необходимо использовать метод множителей Лагранжа. Он позволяет найти такие значения переменных x и y, при которых достигается экстремум функции , учитывая условие связи. Решение задачи заключается в нахождении стационарных точек функции Лагранжа, а затем проверке их на максимальность или минимальность.

**Приложение A.1 Листинг программы**

import sympy as sp

from tkinter import \*

from plot\_file import \*

# [(("x", "y"), "значение u(x, y)")]

res = []

def Start():

    def clicked():

        global res

        x, y = sp.symbols('x y')

        func = txt.get()

        try:

            formula = eval(func)

            grad\_u = [sp.diff(formula, var) for var in (x, y)]

            extrema = sp.solve(grad\_u, (x, y))

        except:

            exc\_txt.configure(text = "Неправильная запись функции")

            return

        gradient = sp.Matrix([sp.diff(func, x), sp.diff(func, y)])

        stationary\_points = sp.solve(gradient, (x, y))

        if len(stationary\_points) == 0:

            exc\_txt.configure("Функция не имеет экстремумов")

            return

        res = []

        for point in extrema:

            temp = formula.subs(x,  point[0])

            res.append((point, temp.subs(y,  point[1])))

        f = sp.lambdify((x, y), formula)

        plot(f, res)

        print\_res(res)

    window = Tk()

    window.geometry('500x150')

    window.title("1 индивидуальная работа")

    txt = Entry(window, width = 15, font = "Arial 14")

    txt.place(relx = 0.57, rely = 0.3, anchor = CENTER)

    but = Button(window, text = "        ", command = clicked, font = "Arial 11")

    but.place(relx = 0.85, rely=0.3, anchor = CENTER)

    txt\_fir = Label(window, text = "Напишите функцию u(x, y)", font = "Arial 12", justify = "left")

    txt\_fir.place(relx = 0.21, rely = 0.3, anchor = CENTER)

    exc\_txt = Label(window, text = "-", font = "Arial 12", justify = "left")

    exc\_txt.place(relx = 0.5, rely = 0.7, anchor = CENTER)

    window.mainloop()

if \_\_name\_\_ == "\_\_main\_\_":

    Start()

**Приложение A.2 Листинг программы**

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D

import numpy as np

from tkinter import \*

from tkinter import scrolledtext

import sympy as sp

def plot(fxy, res):

    X = np.linspace(-5, 5, 100)

    Y = np.linspace(-5, 5, 100)

    X, Y = np.meshgrid(X, Y)

    Z = fxy(X, Y)

    fig = plt.figure()

    ax = fig.add\_subplot(111, projection = '3d')

    ax.plot\_surface(X, Y, Z, alpha=0.5)

    count = 0

    print(res)

    for i in res:

        count += 1

        ax.scatter(i[0][0], i[0][1], i[1], color='r', s=50)

        ax.text(i[0][0], i[0][1], i[1], f"{count}")

    plt.show()

def print\_res(res):

    tk = Tk()

    tk.geometry("400x250")

    tk.title("значения точек экстремумов")

    scr\_txt = scrolledtext.ScrolledText(tk, wrap = WORD, width = 40, height = 10, font = ("Times New Roman", 14))

    scr\_txt.grid(column = 0, pady = 10, padx = 10)

    text = ""

    for i in res:

        text += str(i[0][0]) + " | " + str(i[0][1]) + " | " + str(i[1]) + '\n'

    scr\_txt.insert(INSERT, "x | y | значение u(x, y)\n"  + text)

    tk.mainloop()

**Приложение Б Листинг программы**

import sys

import matplotlib.pyplot as plt

from PyQt5.QtWidgets import QApplication, QWidget, QVBoxLayout, QHBoxLayout, QLabel, QLineEdit, QPushButton, \

QTableWidget, QTableWidgetItem, QScrollArea

from matplotlib.backends.backend\_qt5agg import FigureCanvasQTAgg as FigureCanvas

from matplotlib.figure import Figure

from PyQt5.QtGui import \*

from PyQt5.QtCore import Qt

from math import \*

import numpy as np

from sympy import \*

class MainWindow(QWidget):

    def \_\_init\_\_(self):

        super().\_\_init\_\_()

        # Создаем графические элементы интерфейса

        self.setWindowTitle("Наибольшее и наименьшее значение")  # заголовок окна

        self.move(150, 150)  # положение окна

        self.resize(400, 200)  # размер окна

        self.lbl\_func = QLabel("Функция f(x):")

        self.le\_func = QLineEdit()

        self.rez\_text = QLabel('<b>ОТВЕТ</b>')

        self.lblX0 = QLabel('Введите ограничения:')

        self.txtX0 = QLineEdit()

        self.txtX1 = QLineEdit()

        self.txtX2 = QLineEdit()

        self.rez\_button = QPushButton("Решить")

        self.lbl\_result\_h1 = QLabel()

        self.lbl\_result\_h2 = QLabel()

        self.lbl\_result\_n = QLabel()

        self.lbl\_result\_m = QLabel()

        # Создаем макеты для размещения элементов интерфейса

        self.main\_layout = QVBoxLayout()

        self.head\_layout = QVBoxLayout()

        self.func\_layout = QHBoxLayout()

        self.limit\_layout = QHBoxLayout()

        # Добавляем элементы в левую часть экрана

        self.func\_layout.addWidget(self.lbl\_func)

        self.func\_layout.addWidget(self.le\_func)

        self.limit\_layout.addWidget(self.lblX0)

        self.limit\_layout.addWidget(self.txtX0)

        self.limit\_layout.addWidget(self.txtX1)

        self.limit\_layout.addWidget(self.txtX2)

        self.head\_layout.addLayout(self.func\_layout)

        self.head\_layout.addLayout(self.limit\_layout)

        self.head\_layout.addWidget(self.rez\_button)

        # Добавляем макеты в основной макет

        self.main\_layout.addLayout(self.head\_layout)

        self.main\_layout.addWidget(self.rez\_text)

        self.main\_layout.addWidget(self.lbl\_result\_h1)

        self.main\_layout.addWidget(self.lbl\_result\_h2)

        self.main\_layout.addWidget(self.lbl\_result\_n)

        self.main\_layout.addWidget(self.lbl\_result\_m)

        self.setLayout(self.main\_layout)

        # Назначаем обработчики событий для кнопок

        self.rez\_button.clicked.connect(self.rez)

    def rez(self):

        fun\_value = []

        func\_str = self.le\_func.text()

        lim = [self.txtX0.text(), self.txtX1.text(), self.txtX2.text()]

        func = sympify(replace\_(func\_str))

        toch, changes = limitations(lim)

        oblast = builder\_oblast(toch)

        toch = finding\_points(func, toch)

        for variable in changes:

            temp = func.subs(variable[0][0], variable[0][1])

            toch = finding\_points(temp, toch, variable[0][1])

        belong = []

        for point in toch:

            if belonging\_the\_area(point[0], oblast):

                belong.append(point)

        for research\_point in belong:

            value = function\_value(func, research\_point)

            fun\_value.append(value)

        x\_fig = []

        y\_fig = []

        x\_points = []

        y\_points = []

        for toch in range(len(belong)):

            if toch < 3:

                x\_fig.append(belong[toch][0][0])

                y\_fig.append(belong[toch][0][1])

            else:

                x\_points.append(belong[toch][0][0])

                y\_points.append(belong[toch][0][1])

        x\_fig.append(belong[0][0][0])

        y\_fig.append(belong[0][0][1])

        fig = plt.figure()

        ax1 = fig.add\_subplot(1, 1, 1)

        ax1.grid()

        ax1.spines['left'].set\_position('center')

        ax1.spines['bottom'].set\_position('center')

        ax1.spines['top'].set\_visible(False)

        ax1.spines['right'].set\_visible(False)

        ax1.set\_xlim(-5, 5)

        ax1.set\_ylim(-5, 5)

        ax1.plot(x\_fig, y\_fig, color='red')

        ax1.text(x\_fig[0], y\_fig[0] + 0.3, 'A')

        ax1.text(x\_fig[1], y\_fig[1] + 0.3, 'B')

        ax1.text(x\_fig[2], y\_fig[2], 'C')

        ax1.scatter(x\_points, y\_points, color="green", marker='\*')

        ax1.scatter(x\_fig[0], y\_fig[0], color="green", marker='\*')

        ax1.scatter(x\_fig[1], y\_fig[1], color="green", marker='\*')

        ax1.scatter(x\_fig[2], y\_fig[2], color="green", marker='\*')

        plt.show()

        self.lbl\_result\_h1.setText(f"Точки в области: \n{belong}")

        self.lbl\_result\_h2.setText(f"Значение функции в соответствующих точках: \n{fun\_value}")

        self.lbl\_result\_n.setText(f"Наибольшое значение = {maximum(fun\_value, belong)[0]} в точке {maximum(fun\_value, belong)[1]}")

        self.lbl\_result\_m.setText(f"Наименьшее значение = {minimum(fun\_value, belong)[0]} в точке {minimum(fun\_value, belong)[1]}")

def limitations(lim):

    x, y = symbols("x y")

    limits = []

    limits\_ = []

    extra\_limits = []

    extra\_limits\_ = []

    for i in range(3):

        str\_limit = lim[i]

        if str\_limit.find('x') != -1 and str\_limit.find('y') != -1:

            extra\_limits.append(sympify(str\_limit))

            print(extra\_limits)

            if str\_limit.find(">=") != -1:

                extra\_limits\_.append(sympify(str\_limit.replace(">=", "-")))

                print(extra\_limits\_)

            elif str\_limit.find("<=") != -1:

                extra\_limits\_.append(sympify(str\_limit.replace("<=", "-")))

                print(extra\_limits\_)

        else:

            limits.append(sympify(str\_limit))

            if str\_limit.find(">=") != -1:

                limits\_.append(sympify(str\_limit.replace(">=", "-")))

                print(limits\_)

            elif str\_limit.find("<=") != -1:

                limits\_.append(sympify(str\_limit.replace("<=", "-")))

                print(limits\_)

    limits\_for\_change = []

    limits\_toch = []

    for \_ in limits\_:

        limits\_toch.append(solve(\_))

        limits\_for\_change.append(solve(\_, dict=True))

        print(limits\_toch)

    if len(extra\_limits) != 0:

        for \_ in limits\_:

            limits\_toch.append(solve\_poly\_system([extra\_limits\_[0], \_], x, y))

            print(limits\_toch)

        for \_ in extra\_limits\_:

            limits\_for\_change.append(solve(extra\_limits\_[0], y, dict=True))

            print(limits\_for\_change)

    limits\_toch[0].append(limits\_toch[1][0])

    limits\_toch.pop(1)

    limits\_toch[0] = [tuple(limits\_toch[0])]

    items = []

    for item in limits\_for\_change:

        items.append(list(item[0].items()))

        print(items)

    return [limits\_toch, items]

def finding\_points(expr, points, changed\_variable=0):

    x, y = symbols("x y")

    xdiff = diff(expr, x)

    ydiff = diff(expr, y)

    try:

        point = solve\_poly\_system([xdiff, ydiff], x, y)

        points.append(point)

    except:  # решения на сторонах многоугольника

        if xdiff == 0 and number((str(changed\_variable))):

            point = solve(ydiff, y)

            points.append([(changed\_variable, point[0])])

        elif ydiff == 0 and number(str(changed\_variable)):

            point = solve(xdiff, x)

            points.append([(point[0], changed\_variable)])

        elif not str(changed\_variable).isdigit():

            pointx = solve(xdiff, x)

            pointy = changed\_variable.subs(x, pointx[0])

            points.append([(pointx[0], pointy)])

    print(points)

    return points

def builder\_oblast(points):

    points\_for\_poly = []

    for point in points:

        points\_for\_poly.append(point[0])

    poly = Polygon(\*points\_for\_poly)

    return poly

def belonging\_the\_area(znach, oblast):

    if oblast.encloses\_point(znach) or oblast.intersection(znach) != []:

        return True

    else:

        return False

def function\_value(expr, point):

    x, y = symbols("x y")

    value = expr.subs([(x, point[0][0]), (y, point[0][1])])

    return value

def maximum(values, points):

    value = 0

    point = 0

    for i in range(len(values)):

        if value < values[i]:

            value, point = values[i], points[i]

    return [value, point]

def minimum(values, points):

    value = 0

    point = 0

    for i in range(len(values)):

        if value > values[i]:

            value, point = values[i], points[i]

    return [value, point]

def number(num):

    try:

        float(num)

        return True

    except ValueError:

        return False

def replace\_(function):

    val = function.replace("^", "\*\*")

    return val

if \_\_name\_\_ == "\_\_main\_\_":

    app = QApplication(sys.argv)

    window = MainWindow()

    window.show()

    sys.exit(app.exec\_())

**Приложение В Листинг программы**

from sympy import \*

import matplotlib.pyplot as plt

import numpy as np

x, y, lam = symbols("x y lam")

def extremes\_func():

    t1 = input("Введите функцию: ")

    t2 = input("Введите условие: ")

    u, f = simplify(t1, evaluate=False), simplify(t2, evaluate=False)

    L = u + lam \* f

    Lx = diff(L, x)

    Ly = diff(L, y)

    variables = solve([Lx, Ly, f], [x, y, lam])

    M = {x: variables[x], y: variables[y]}

    Lxx = diff(Lx, x)

    Lyy = diff(Ly, y)

    Lxy = diff(Lx, y)

    fx, fy = diff(f, x), diff(f, y)

    delta = -(Matrix([[0, fx, fy], [fx, Lxx, Lxy], [fy, Lxy, Lyy]]).det())

    print("дельта =",delta)

    u\_res = u.subs([(x, M[x]), (y, M[y])])

    if delta > 0:

        print(f"Условный минимум {str(float(u\_res))} в точке ({str((M[x]))}, {str((M[y]))})")

    elif delta < 0:

        print(f"Условный максимум {str(float(u\_res))} в точке ({str((M[x]))}, {str((M[y]))})")

    else:

        print(f"Нет условного экстремума")

        return

    plot\_suface(float(M[x]), float(M[y]), u, f)

def plot\_suface(px, py, equation, con):

    equation\_lambda = lambdify((x, y), equation)

    con\_lambda = lambdify((x, y), con)

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

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

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

    Z = equation\_lambda(X, Y)

    Z1 = con\_lambda(X, Y)

    fig = plt.figure()

    ax = fig.add\_subplot(111, projection='3d')

    ax.plot\_surface(X, Y, Z, cmap='inferno\_r',alpha = 0.5)

    ax.plot\_surface(X, Y, Z1)

    point\_x = px

    point\_y = py

    point\_z = equation\_lambda(point\_x, point\_y)

    ax.scatter(point\_x, point\_y, point\_z, color='red')

    ax.set\_xlabel('x')

    ax.set\_ylabel('y')

    ax.set\_zlabel('z')

    plt.show()

extremes\_func()