|  |  |
| --- | --- |
| **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 | |  |  | ( | Сафронов Н.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:** изучение математического аппарата математического программирования на примере задач небольшой размерности, допускающих графическое решение.

**Постановка задачи**

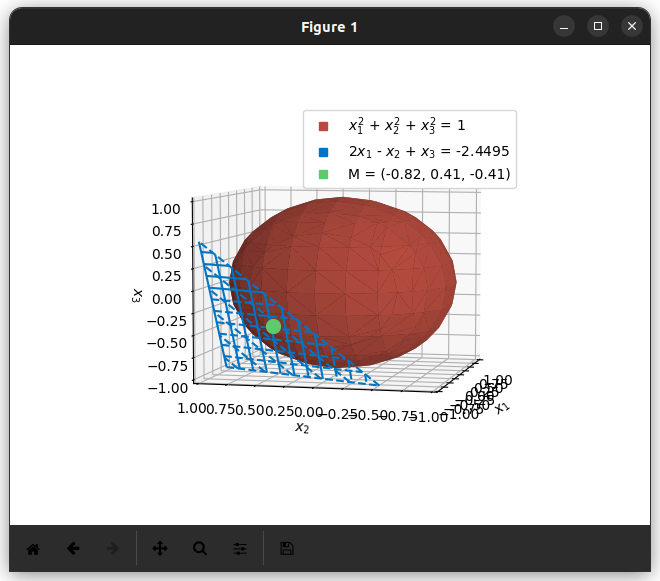
**Вариант 14**

Найти условный экстремум функции методом множителей Лагранжа:

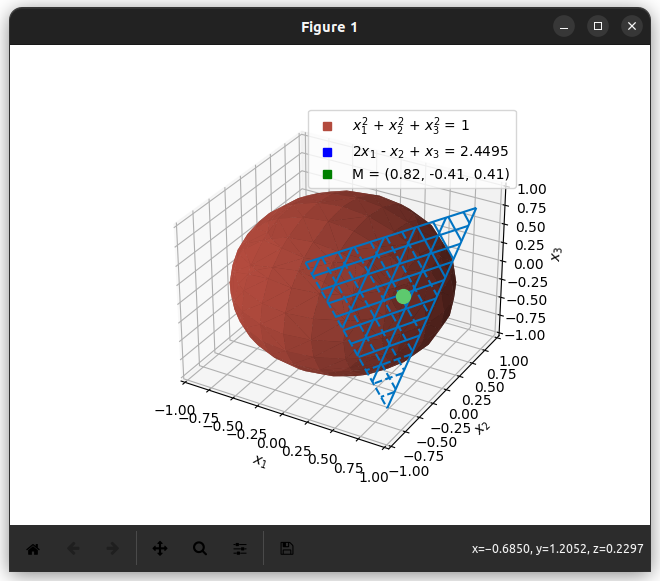
при условии

**Решение**

Построим графики функций и обозначим на них полученные точки:

****

**Рисунок 1 –** Найденная точка условного локального минимума

****

**Рисунок 2 –** Найденная точка условного локального максимума

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

**ПРИЛОЖЕНИЯ**

**Листинг программы**

import matplotlib  
import matplotlib.pyplot as plt  
import numpy as np  
from skimage import measure  
  
  
def function(x\_1: float, x\_2: float, x\_3: float, z: float) -> float:  
 return 2 \* x\_1 - x\_2 + x\_3 - z  
  
  
def condition(x\_1: float, x\_2: float, x\_3: float) -> float:  
 return np.power(x\_1, 2) + np.power(x\_2, 2) + np.power(x\_3, 2) - 1  
  
  
def plot\_implicit(  
 axis: plt.Axes, fn, bbox=(-2.5, 2.5), color="red",  
 order=1  
):  
 xl = np.linspace(-3, 3, 25)  
 X, Y, Z = np.meshgrid(xl, xl, xl)  
 F = fn(X, Y, Z)  
 verts, faces, normals, values = measure.marching\_cubes(  
 F, 0, spacing=[np.diff(xl)[0]] \* 3  
 )  
 verts -= 3  
 axis.plot\_trisurf(  
 verts[:, 0], verts[:, 1], verts[:, 2], triangles=faces,  
 color=color, lw=0, zorder=order  
 )  
  
  
def plot\_implicit\_contour(  
 axis: plt.Axes, fn, bbox=(-2.5, 2.5), color="red",  
 order=1  
):  
 A = np.linspace(-1, 1, 100)   
 B = np.linspace(-1, 1, 15)   
 A1, A2 = np.meshgrid(A, A)   
 for z in B:

X, Y = A1, A2  
 Z = fn(X, Y, z)  
 cset = axis.contour(  
 X, Y, Z + z, [z], zdir="z", colors=[color],  
 zorder=order  
 )  
  
 for y in B:   
 X, Z = A1, A2  
 Y = fn(X, y, Z)  
 cset = axis.contour(  
 X, Y + y, Z, [y], zdir="y", colors=[color],  
 zorder=order  
 )  
  
 for x in B:   
 Y, Z = A1, A2  
 X = fn(x, Y, Z)  
 cset = axis.contour(  
 X + x, Y, Z, [x], zdir="x", colors=[color],  
 zorder=order  
 )  
  
  
def show\_result\_plot(  
 function,  
 condition,  
 z: float,  
 point: list,  
 colors: dict,  
 legend: list  
):  
 figure = plt.figure()  
 axis = figure.add\_subplot(111, projection="3d", computed\_zorder=False)  
 plot\_implicit(axis, condition, color=colors["red"], order=1)  
 plot\_implicit\_contour(  
 axis, lambda x\_1, x\_2, x\_3: function(x\_1, x\_2, x\_3, z),  
 color=colors["blue"], order=2  
 )  
 axis.plot(  
 point[0], point[1], point[2], ".", c=colors["black"],  
 markersize=20, zorder=3  
 )  
 axis.set\_xlabel("$x\_1$")  
 axis.set\_ylabel("$x\_2$")  
 axis.set\_zlabel("$x\_3$")  
 axis.set\_xlim(-1, 1)  
 axis.set\_ylim(-1, 1)  
 axis.set\_zlim(-1, 1)  
  
 plt.legend(  
 handles=legend  
 )  
 plt.show()  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 colors = {  
 "red": "#B34B3E",  
 "blue": "#0174C3",  
 "black": "#5DCA6E"  
 }  
  
 min\_z = -6 \* np.sqrt(1 / 6)  
 red\_legend\_handle = matplotlib.lines.Line2D(  
 [], [], color=colors["red"], marker="s", ls="",  
 label="$x\_1^2$ + $x\_2^2$ + $x\_3^2$ = 1"  
 )  
 blue\_legend\_handle = matplotlib.lines.Line2D(  
 [], [], color=colors["blue"], marker="s", ls="",  
 label=f"2$x\_1$ - $x\_2$ + $x\_3$ = {min\_z:.4f}"  
 )  
 green\_legend\_handle = matplotlib.lines.Line2D(  
 [], [], color=colors["black"], marker="s", ls="",  
 label=f"M = ({-np.sqrt(2 / 3):.2f}, "  
 f"{np.sqrt(2 / 12):.2f}, {-np.sqrt(2 / 12):.2f})"  
 )  
 legend = [red\_legend\_handle, blue\_legend\_handle, green\_legend\_handle]  
 point = [-np.sqrt(2 / 3), np.sqrt(2 / 12), -np.sqrt(2 / 12)]  
 show\_result\_plot(function, condition, min\_z, point, colors, legend)  
  
 max\_z = 6 \* np.sqrt(1 / 6)  
 blue\_legend\_handle = matplotlib.lines.Line2D(  
 [], [], color="blue", marker="s", ls="",  
 label=f"2$x\_1$ - $x\_2$ + $x\_3$ = {max\_z:.4f}"  
 )  
 green\_legend\_handle = matplotlib.lines.Line2D(  
 [], [], color="green", marker="s", ls="",  
 label=f"M = ({np.sqrt(2 / 3):.2f}, "  
 f"{-np.sqrt(2 / 12):.2f}, {np.sqrt(2 / 12):.2f})"  
 )  
 legend = [red\_legend\_handle, blue\_legend\_handle, green\_legend\_handle]  
 point = [np.sqrt(2 / 3), -np.sqrt(2 / 12), np.sqrt(2 / 12)]  
 show\_result\_plot(function, condition, max\_z, point, colors, legend)