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

Решить задачу нелинейного программирования графическим методом:

$$z = x_1 + 2x_2 \rightarrow (max, min)$$

при ограничениях

$$z = \begin{cases} (x_1 - 2)(x_2 + 1) \geq 4 \\ x_1 + x_2 \leq 6 \\ x_1 \geq, x_2 \geq 0 \end{cases}$$

Листинг программы

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.colors
from matplotlib.animation import FuncAnimation
import matplotlib.lines

from typing import Union


def first_condition(
    xs: Union[np.array, float], ys: Union[np.array, float]
) -> Union[np.array, float]:
    return (xs - 2) * (ys + 1)


def second_condition(
    xs: Union[np.array, float], ys: Union[np.array, float]
) -> Union[np.array, float]:
    return xs + ys


def first_condition_inequality(
    xs: Union[np.array, float], ys: Union[np.array, float]
) -> Union[np.array, float]:
    return first_condition(xs, ys) >= 4


def second_condition_inequality(
    xs: Union[np.array, float], ys: Union[np.array, float]
) -> Union[np.array, float]:
    return second_condition(xs, ys) <= 6


class Animation:
```

```
    def __init__(self, xs, ys, line, points, min_label, max_label,
root_label):
        self._xs = xs
        self._ys = ys
        self._line = line
        self._points = points
        self._min_label = min_label
        self._max_label = max_label
        self._root_label = root_label
        self._previous_points = []
        self._pause = False
        self._previous_root = 5

    def to_tuple(self) -> tuple:
        return (
            self._line,
            self._points,
            self._min_label,
            self._max_label,
            self._root_label,
        )

    def get_frames(self):
        MAX_ROOT = 10
        STEP = 0.01
        while self._previous_root < MAX_ROOT:
            if not self._pause:
                self._previous_root += STEP
            yield self._previous_root

    def on_click(self, event):
        self._pause ^= True

    def get_animation_frame(self, z: float) -> tuple:
        y = (z - self._xs) / 2
        self._line.set_data(self._xs, y)
        self._root_label.set_text(f"z = {z:.2f}")

        intersections = {}

        y_1 = ((z / 2) + np.sqrt(z**2 / 4 - 8)) / 2 - 1
        x_1 = z - 2 * y_1

        if (
            abs(first_condition(x_1, y_1) - 4) <= 1e-6
            and second_condition(x_1, y_1) <= 6
        ):
            intersections[x_1] = y_1

        y_2 = z - 6
        x_2 = z - 2 * y_2
        if (
            abs(second_condition(x_2, y_2) - 6) <= 1e-6
            and first_condition(x_2, y_2) >= 4
        ):
            intersections[x_2] = y_2

        if len(self._previous_points) == 0 and len(intersections) > 0:
            xs, ys = ([*intersections.keys()], [*intersections.values()])
```

```
            min_label.set_text(f"min $z = $f({xs[0]:.2f}, {ys[0]:.2f}) =
{z:.2f}")
            self._pause = True
        elif len(self._previous_points) > 0 and len(intersections) == 0:
            xs, ys = self._previous_points
            z = xs[0] + 2 * ys[0]
            max_label.set_text(f"max $z = $f({xs[0]:.2f}, {ys[0]:.2f}) =
{z:.2f}")
            self._pause = True

        self._points.set_data([*intersections.keys()],
[*intersections.values()])
        if len(intersections) > 0:
            self._previous_points = ([*intersections.keys()],
[*intersections.values()])
        else:
            self._previous_points = []

        return self.to_tuple()


if __name__ == "__main__":
    x_range = np.linspace(0, 7, 250)
    y_range = np.linspace(0, 4, 250)
    xs, ys = np.meshgrid(x_range, y_range)

    figure, axis = plt.subplots()

    matplotlib.pyplot.contour(xs, ys, (first_condition(xs, ys) - 4), [0],
colors="k")
    matplotlib.pyplot.contour(xs, ys, (second_condition(xs, ys) - 6), [0],
colors="k")

    first_region = first_condition_inequality(xs, ys)
    second_region = second_condition_inequality(xs, ys)
    extent = (xs.min(), xs.max(), ys.min(), ys.max())
    plt.imshow(
        first_region.astype(int), extent=extent, origin="lower",
cmap="Blues", alpha=0.5
    )

    plt.imshow(
        second_region.astype(int),
        extent=extent,
        origin="lower",
        cmap="Greens",
        alpha=0.5,
    )

    line = axis.plot([], [], "r")[0]
    points = axis.plot([], [], "ro")[0]
    min_label = axis.text(0.1, 0.3, "", fontsize=10)
    max_label = axis.text(0.1, 0.1, "", fontsize=10)
    root_label = axis.text(0.1, 0.5, "", fontsize=10)

    animation = Animation(xs, ys, line, points, min_label, max_label,
root_label)
    figure.canvas.mpl_connect('button_press_event', animation.on_click)
    ani = FuncAnimation(
```

```
    figure,
    animation.get_animation_frame,
    frames=animation.get_frames,
    init_func=animation.to_tuple,
    blit=True,
    interval=10,
)

green_legend_handle = matplotlib.lines.Line2D(
    [], [], color="green", marker="s", ls="",
    label="$x_1$ + $x_2$ <= 6"
)
blue_legend_handle = matplotlib.lines.Line2D(
    [], [], color="blue", marker="s", ls="",
    label="($x_1$ - 2)($x_2$ + 1) >= 4"
)

plt.legend(handles=[green_legend_handle, blue_legend_handle])

plt.show()
```

Результаты выполнения программы

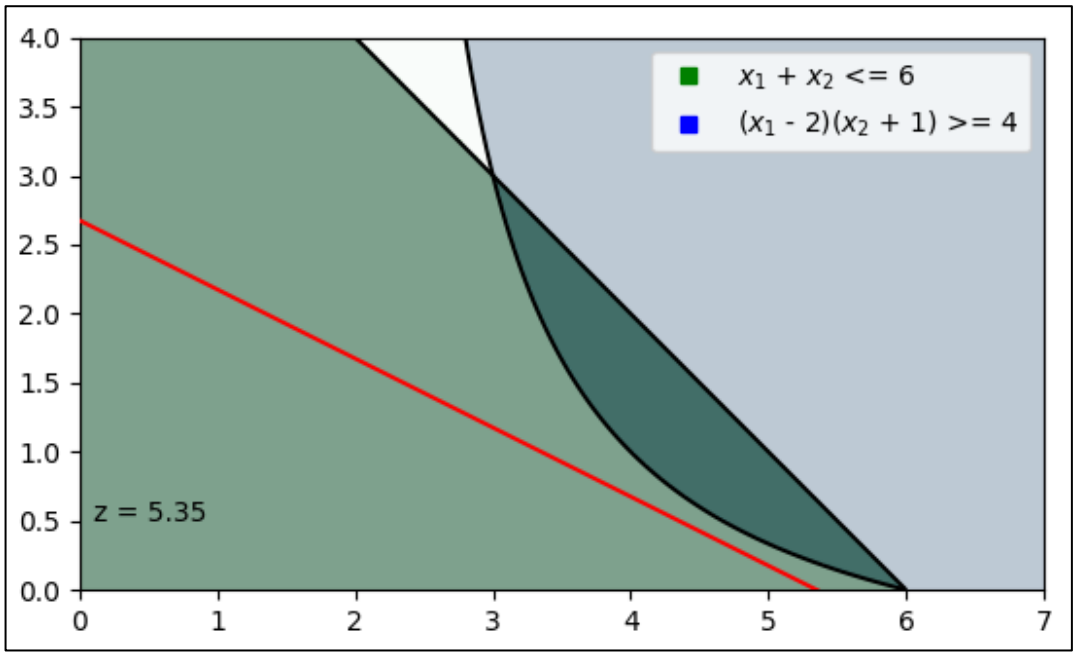

Рисунок 1 – Анимация нахождения решения графическим методом в начальный момент

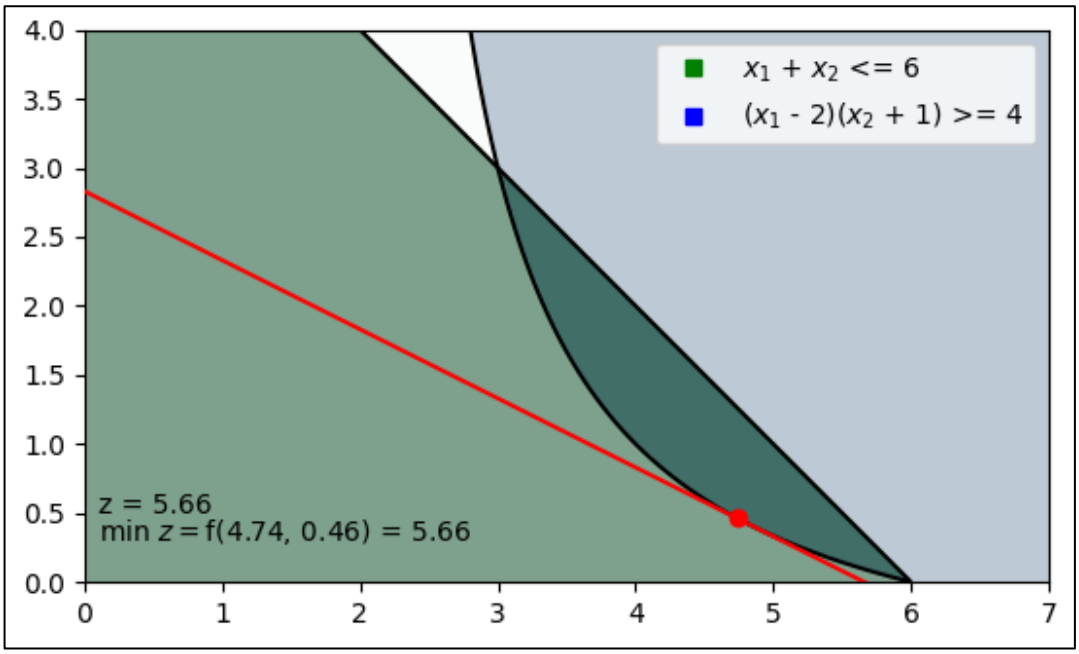

Рисунок 2 – Найденная точка минимума

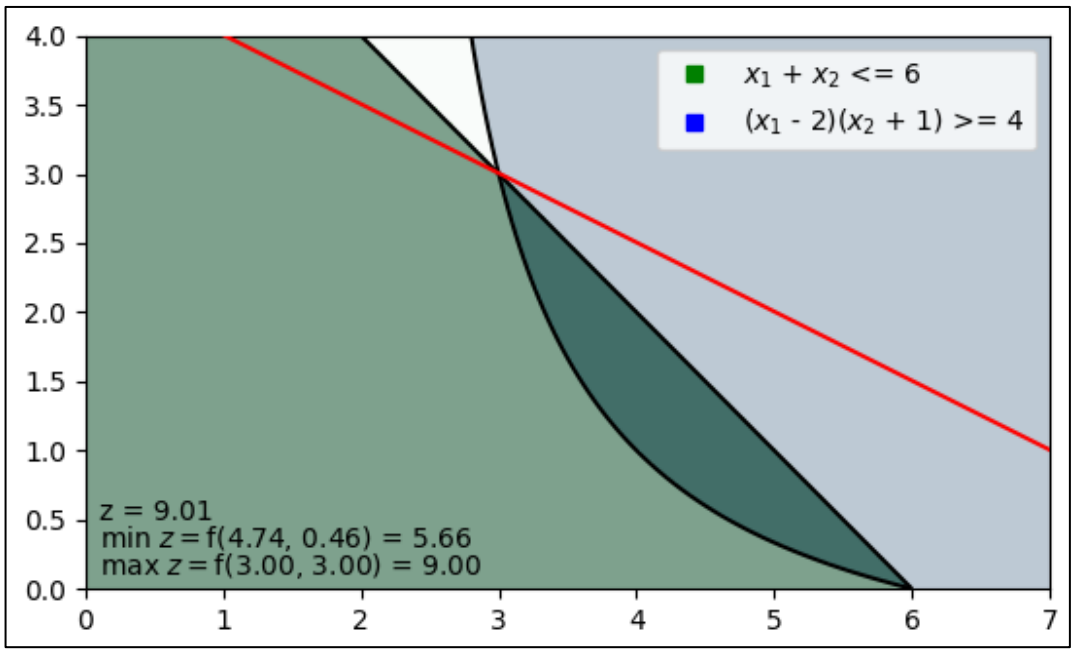

Рисунок 3 – Найденная точка максимума

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