ГУАП

КАФЕДРА № 42

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ _____

ПРЕПОДАВАТЕЛЬ

Доцент		А.В. Аграновский
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ № 1

ИНТЕРПОЛЯЦИОННАЯ КРИВАЯ CATMULL-ROM

Вариант 5

по курсу: КОМПЬЮТЕРНАЯ ГРАФИКА

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 4128		В.А. Воробьев
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2023

СОДЕРЖАНИЕ

1 Цель работы

Изучение интерполяционной кривой Catmull-Rom, построение интерполяционной кривой Catmull-Rom с помощью математического пакета и/или языка программирования высокого уровня.

Задание:

1) Построить график гармонических колебаний.
2) На периоде гармонических колебаний взять N точек, где N равно 4 плюс номер студента в группе.
3) По опорным точкам из пункта 2 построить кривую Catmull-Rom (на том же графике, что и в пункте 1).
4) Рассчитать ошибку восстановления гармонических колебаний кривой Catmull-Rom.
5) Уменьшить число точек на периоде в 2 раза и повторить пункты 1-4.
6) Увеличить число точек на периоде в 2 раза и повторить пункты 1-4.
7) Построить кривую Catmull-Rom на основе полинома N-го порядка (где N берется из пункта 2) и рассчитать ошибку.
8) На форме должно быть 3 кнопки: отображение графика гармонических колебаний, интерполяционной кривой Catmull-Rom на основе гармонических колебаний и интерполяционной кривой Catmull-Rom на основе N-го полинома

Группа	Функция для лабораторных работ 1 и 2	Интервал
4128	*f(x) = 2 sin(x) + 1.5 sin(2x)*	*один период f(x)*

Рисунок 1 – вариант задания

2 Теоретические сведения

Сплайн – кривая, удовлетворяющая некоторым критериям гладкости.

Базовые (опорные) точки – набор точек, на основе которых выполняется построение кривой.

Интерполяция – построение кривой, точно проходящей через набор базовых точек.

Полиномом называется функция вида:

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 = y$$

Существует большое количество разных вариантов сплайновых кривых, отличающихся своими свойствами.

Сплайновая кривая *Catmull-Rom* (см рис. 2) по заданному массиву точек P0, P1, P2, P3 определяется при помощи уравнения, имеющего следующий вид:

$$R(t) = 1/2\,(-t(1-t)^2 P_0 + (2 - 5t^2 + 3t^3)P_1 + t(1 + 4t - 3t^2)P_2 - t^2(1-t)P_3),$$

$$0 \le t \le 1.$$

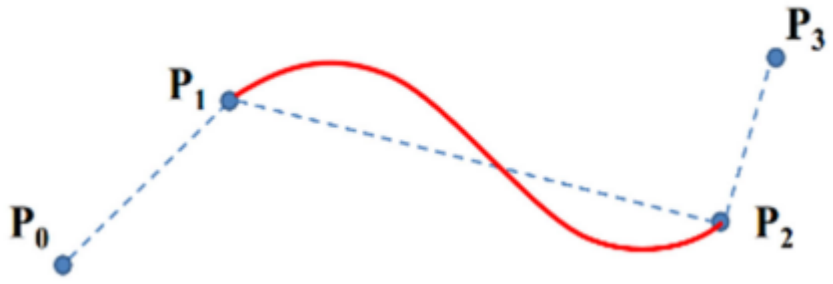

Рисунок 2 – пример кривой Catmull-Rom

Поскольку сплайновая кривая Catmull-Rom является интерполяционной, то она проходит через каждую из базовых точек.

Однако при построении составной сплайновой кривой каждый сегмент рассчитывается на участке между парой внутренних точек очередной четверки из набора базовых точек. Поэтому для построения составной сплайновой интерполяционной кривой, начинающейся в первой базовой точке и заканчивающейся в последней базовой точке, достаточно дополнить набор копиями первой и последней точек. Копия начальной точки при этом добавляется в начало набора, а копия последней точки – в конец набора.

Каждый сегмент кривой строится на основе четырех точек. Построение кривой осуществляется только между двумя внутренними точками каждой четверки.

Четверки выбираются с перекрытием, т.е. первой точкой очередной четверки выбирается вторая точка предыдущей четверки. Например, сегмент 1 – строится на основе точек P0, P1, P2, P3, а сегмент 2 – на основе точек P1, P2, P3, P4.

3 Выполнение работы

Для выполнения лабораторной работы было решено выбрать высокоуровневый язык Python 3 и библиотеки matplotlib (для отображения графиков) и NumPy (работа с матрицами). Код, выполняющий поставленную задачу, показан в приложении А, а также доступен по ссылке на GitHub(URL: https://github.com/vladcto/SUAI_homework/blob/8e15da3e3e5cb70da0215ea3d1bf0f553278730b/4_semester/CG/1%D0%BB%D1%80/solution.py). Код снабжен комментариями и легок для понимания. Ниже приведены скриншоты с результатом работы программы. В соответствии с 5 вариантом нужно было взять 9, 4 и 18 точек, а также полином 9-го порядка.

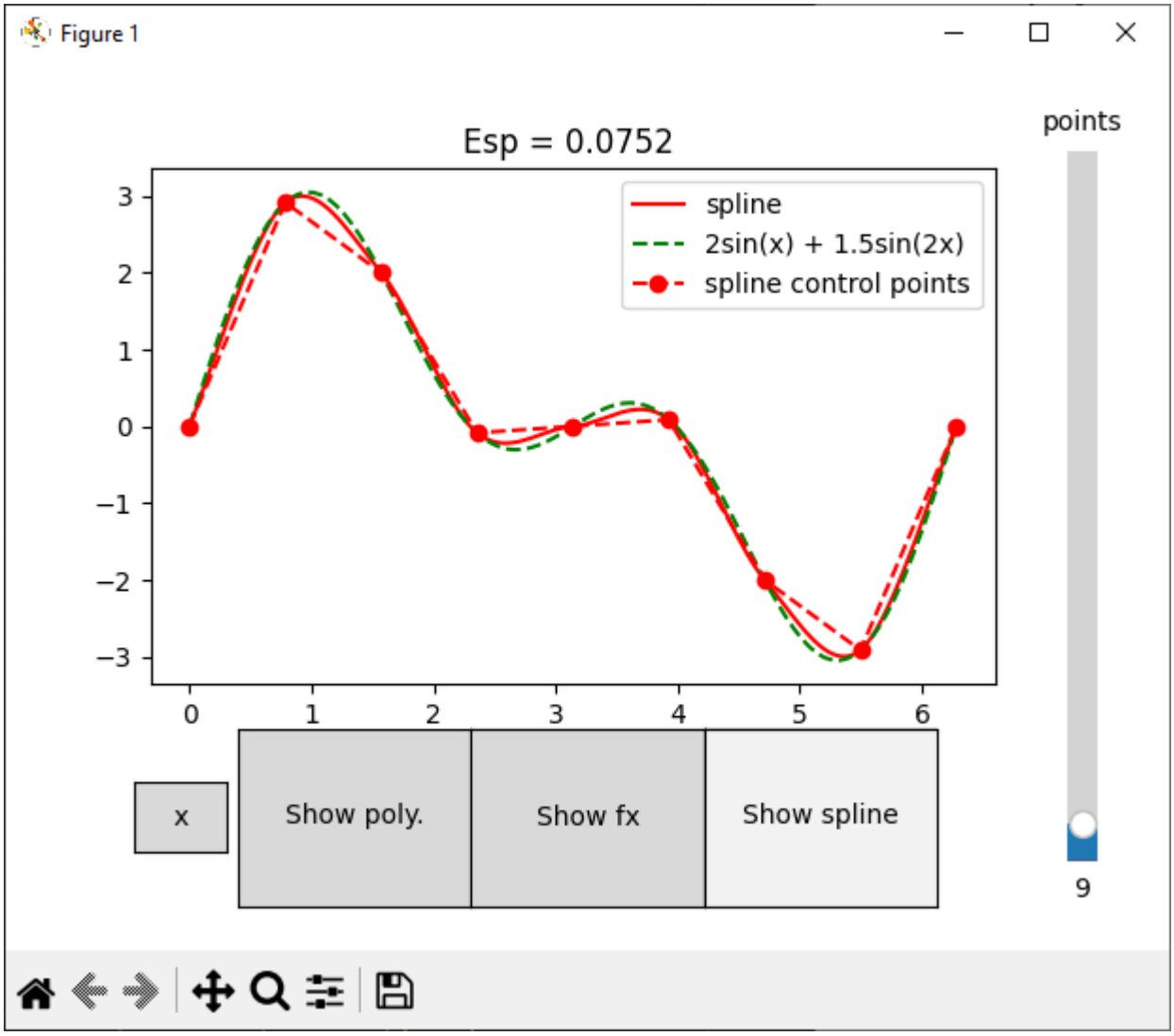

Рисунок 3 – сплайн Catmull-Rom для 9 контрольных точек

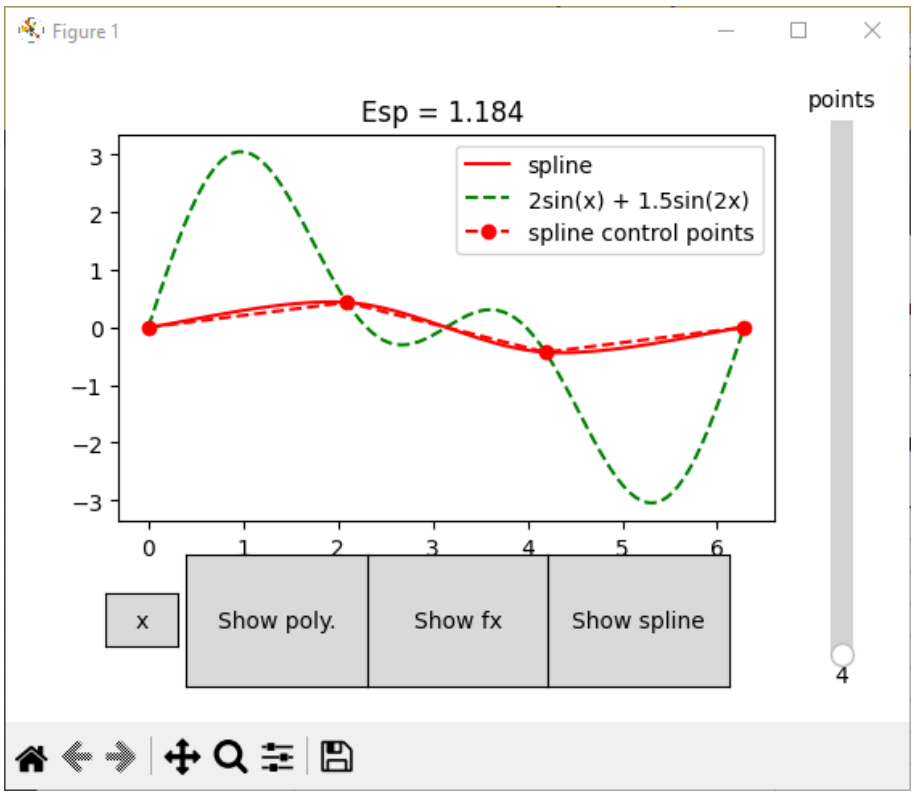

Рисунок 4 – сплайн Catmull-Rom для 4 точек

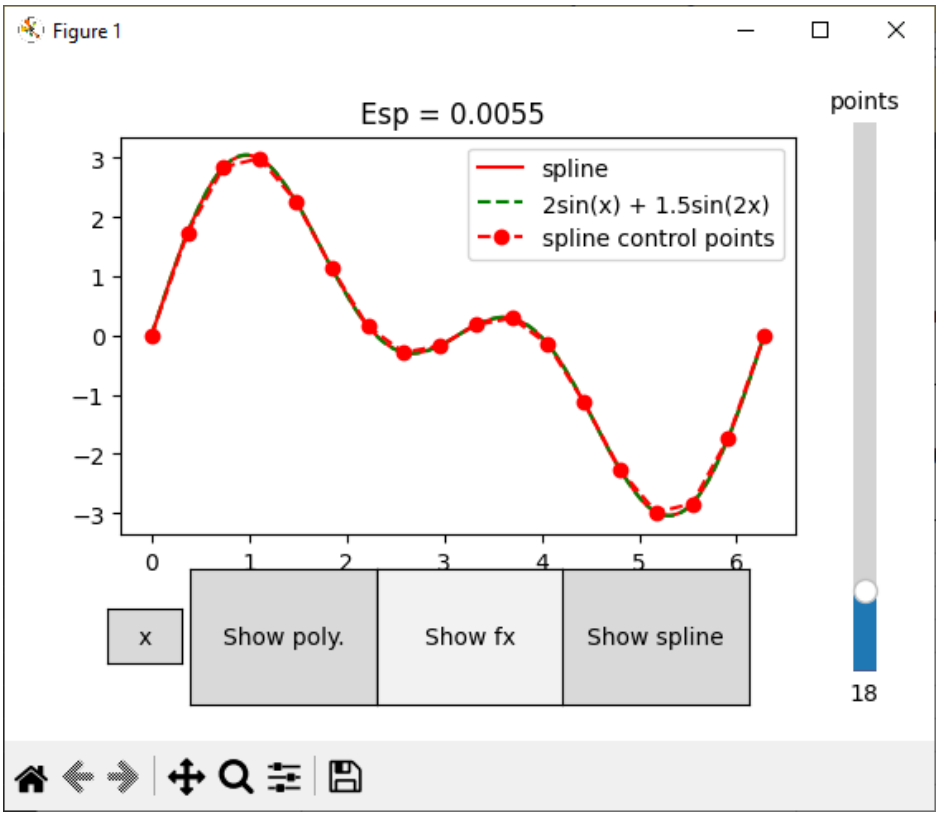

Рисунок 5 – сплайн Catmull-Rom для 18 точек

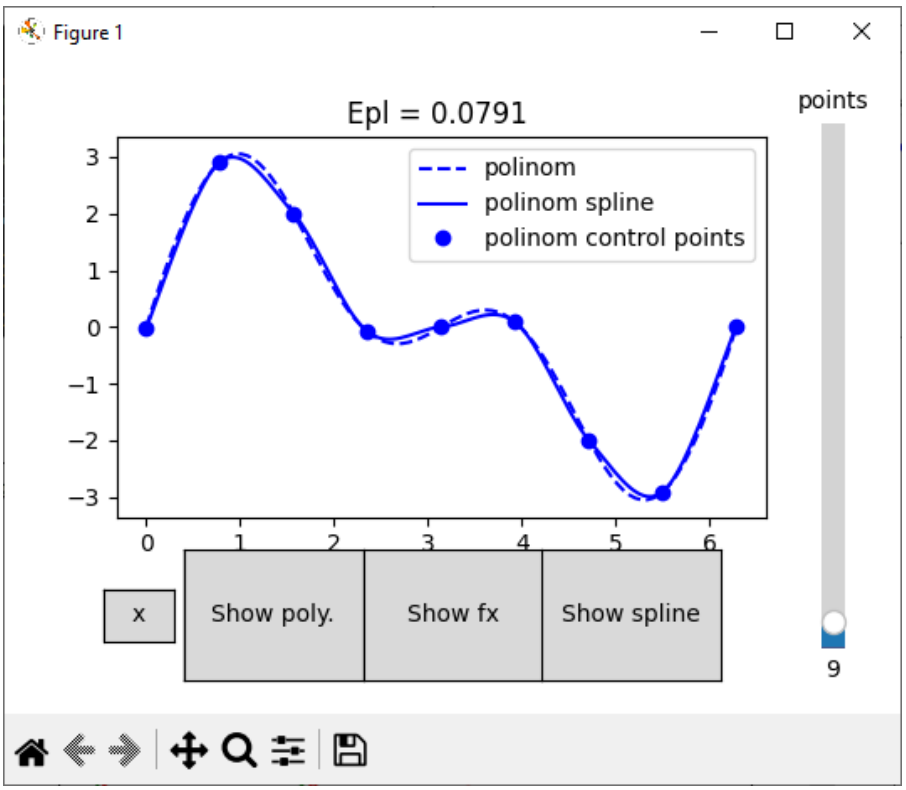

Рисунок 6 – сплайн Catmull-Rom для полинома 9-го порядка

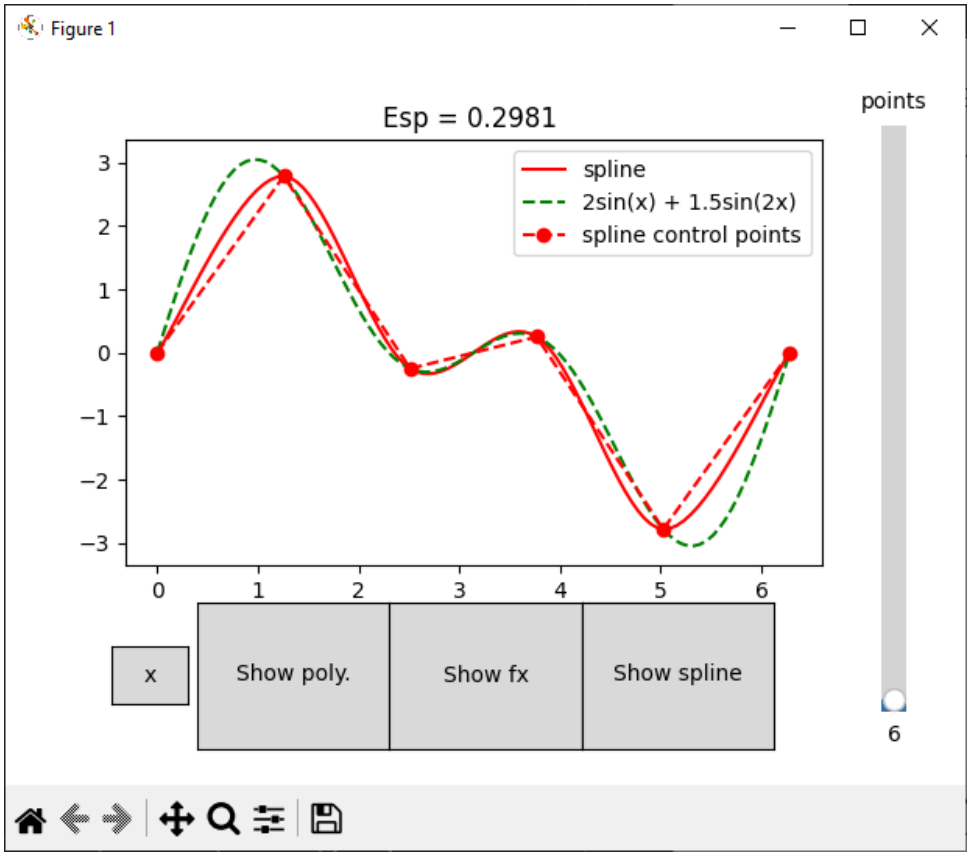

Рисунок 7 – сплайн Catmull-Rom для произвольного количества точек

4 Вывод

В результате выполнения лабораторной работы были получены навыки построения кривой Catmull-Rom, используя язык Python. С помощью библиотеки matplotlib и NumPy был реализован алгоритм построения кривой для гармонической функции и полинома 9-го порядка, а также кривой Catmull-Rom для заданного количества опорных точек.

Функционал программы был протестирован, в результате чего мы получили значения отклонения кривой Catmull-Rom от графика гармонической функции, для которой был построен сплайн:

1) 4 точки – 1.184
2) 9 точек – 0.075
3) 18 точек – 0.006

По полученным значениям можем утверждать, что с увеличением количества опорных точек увеличивается точность кривой Catmull-Rom. Причем вне зависимости от количества опорных точек кривая Catmull-Rom всегда проходит через них.

Одним из недостатков кривой Catmull-Rom можно выделить, что у нас нет явного способа регулировать кривизну кривой, разве только если увеличивать число опорных точек, что может сказаться на производительности.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1) Mika's Coding Bits: Использование кривых Catmull-Rom: сайт. – URL: https://qroph.github.io/2018/07/30/smooth-paths-using-catmull-rom-splines.html (дата обращения: 01.03.2023)
2) NumPy: Документация NumPy: сайт. – URL: https://numpy.org/doc/stable/index.html (дата обращения: 01.03.2023)
3) Matplotlib: Документация Matplotlib: сайт. – URL: https://matplotlib.org/stable/index.html (дата обращения: 01.03.2023)

ПРИЛОЖЕНИЕ А

ЛИСТИНГ ПРОГРАММЫ

```
from math import sin
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.widgets import Button, Slider


# Catmull Row function
def catmull_equation(points, t):
    p1 = points[0]
    p2 = points[1]
    p3 = points[2]
    p4 = points[3]
    q1 = -t * ((1 - t) ** 2) * p1
    q2 = (2 - 5 * (t ** 2) + 3 * (t ** 3)) * p2
    q3 = t * (1 + 4 * t - 3 * (t ** 2)) * p3
    q4 = -(t ** 2) * (1 - t) * p4
    return 0.5 * (q1 + q2 + q3 + q4)



# Get points for spline
def catmull_row(points, step=0.05):
    # copy one point to start and one to end
    res = []
```

```
    points = np.vstack((points[0], points, points[-1]))
    # take 4 points
    for i in range(points.shape[0] - 3):
        points_now = points[i:i+4]
        for t in np.arange(0, 1, step):
            res.append(catmull_equation(points_now, t))
    return np.array(res)



def update_plot(state, fig):
    fig.clear()
    # Control points for spline and other curves.
    x_points = np.linspace(0, 2 * np.pi, 100)
    x_controls = np.linspace(0, 2 * np.pi, state["points"])
    y_controls = 2 * np.sin(x_controls) + 1.5 * np.sin(x_controls * 2)
    label_text = ""

    if (state["poly"]):
        # Polynom points
        y_f = 2 * np.sin(x_points) + 1.5 * np.sin(x_points * 2)
        # Polynom coefs
        p = np.polyfit(x_points,y_f,9)
        poli_y = np.polyval(p,x_points)
        poli_controls_y = np.polyval(p,x_controls)
```

```
    poli_points = catmull_row(np.column_stack([x_controls, poli_controls_y]))


    fig.plot(x_points, poli_y, "b--", label="polinom")
    fig.plot(poli_points[:, 0], poli_points[:, 1],
             "b-", label="polinom spline")
    fig.plot(x_controls, poli_controls_y, "bo", label="polinom control points")
    # Calculate error
    y_predict = np.polyval(p,poli_points[:,0])
    error_dif = np.abs(poli_points[:, 1] - y_predict).mean()
    label_text += f"Epl = {round(error_dif,4)} "

if (state["catmull"]):
    # Spline points
    splain = catmull_row(np.column_stack((x_controls, y_controls)))
    x_splain = splain[:, 0]
    y_splain = splain[:, 1]


    fig.plot(x_splain, y_splain, "r-", label="spline")
    # Calculate error
    y_predict = np.asarray(
        [2 * sin(x) + 1.5*sin(2 * x) for x in x_splain])
    error_dif = np.abs(y_splain - y_predict).mean()
    label_text += f"Esp = {round(error_dif,4)} "
```

```
    if (state["fx"]):
        y_f = 2 * np.sin(x_points) + 1.5 * np.sin(x_points * 2)

        fig.plot(x_points, y_f, "g--", label="2sin(x) + 1.5sin(2x)")
        fig.plot(x_controls, y_controls, "ro--", label="spline control points")

    handles, labels = fig.get_legend_handles_labels()
    by_label = dict(zip(labels, handles))
    fig.legend(by_label.values(), by_label.keys())
    fig.set_title(label_text)
    plt.draw()


def draw_polinom(state, fig):
    state["poly"] = True
    update_plot(state, fig)


def draw_spline(state, fig):
    state["catmull"] = True
    update_plot(state, fig)


def draw_function(state, fig):
```

```
    state["fx"] = True
    update_plot(state, fig)


def clear(state, fig):
    state["fx"] = False
    state["catmull"] = False
    state["poly"] = False
    update_plot(state, fig)


def change_control_points(state, fig, num):
    state["points"] = int(num)
    update_plot(state, fig)


# Draw graphics
fig, ax = plt.subplots()
my_state = {"fx": False, "catmull": False, "poly": False, "points": 4}
fig.subplots_adjust(bottom=0.3)
fig.subplots_adjust(right=0.85)

# Buttons and slider
ax_poli_btn = fig.add_axes([0.2, 0.05, 0.2, 0.2])
```

```
poli_btn = Button(ax_poli_btn, "Show poly.")
poli_btn.on_clicked(lambda _: draw_polinom(my_state, ax))

ax_fx_btn = fig.add_axes([0.4, 0.05, 0.2, 0.2])
fx_btn = Button(ax_fx_btn, "Show fx")
fx_btn.on_clicked(lambda _: draw_function(my_state, ax))

ax_splain_btn = fig.add_axes([0.6, 0.05, 0.2, 0.2])
splain_btn = Button(ax_splain_btn, "Show spline")
splain_btn.on_clicked(lambda _: draw_spline(my_state, ax))

ax_clear_btn = fig.add_axes([0.11, 0.11, 0.08, 0.08])
clear_btn = Button(ax_clear_btn, "x")
clear_btn.on_clicked(lambda _: clear(my_state, ax))

ax_slider = fig.add_axes([0.9, 0.1, 0.05, 0.8])
slider = Slider(ax_slider,
                valmin=4,
                valmax=100,
                orientation="vertical",
                label="points",
                valstep=1)
slider.on_changed(lambda num: change_control_points(my_state, ax, num))
plt.show()
```