ГУАП

КАФЕДРА № 42

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ПРЕПОДАВАТЕЛЬ

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

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ № 2 |
| **Сплайновая кривая Безье**  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 3](#_Toc128758293)

[2 Теоретические сведения 4](#_Toc128758294)

[3 Выполнение работы 6](#_Toc128758295)

[4 Вывод 13](#_Toc128758296)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 14](#_Toc128758297)

[ПРИЛОЖЕНИЕ А. ЛИСТИНГ ПРОГРАММЫ 15](#_Toc128758298)

**1 Цель работы**

Изучение сплайновой кривой Безье, построение сплайновой кривой Безье 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 Безье (на том же графике, что и в пункте 1).
4. Рассчитать ошибку восстановления гармонических колебаний кривой Безье.
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 Безье 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, кривой Безье на основе гармонических колебаний и кривой Безье на основе N-го полинома

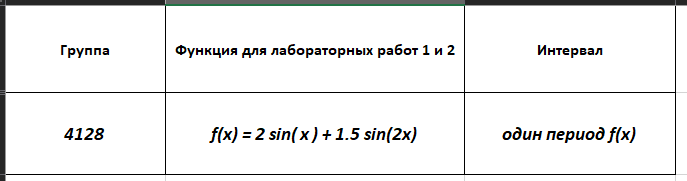


Рисунок 1 – вариант задания

**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.

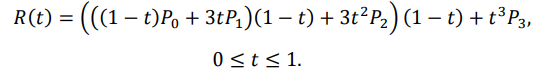
*Полиномом* называется функция вида:



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

*Кривая Безье* является частным случаем многочленов Бернштейна, описанных Сергеем Натановичем Бернштейном в 1912 году.

По заданному массиву точек P0, P1, P2, P3 сплайновая кубическая элементарная кривая Безье (По заданному массиву точек P0, P1, P2, P3 сплайновая кубическая элементарная кривая Безье (рисунок 1) описывается уравнением (3): см. рис. 2) описывается уравнением:



Элементарная кривая начинается в точке P0 и заканчивается в точке P3,

касаясь при этом отрезков P0 P1 и P2 P3.

Свойства составной кривой Безье:

1. проходит внутри выпуклой оболочки, заданной опорными точками;
2. набор базовых функций однозначно определяет кривую, т.е. нет возможности регулировать ее форму.

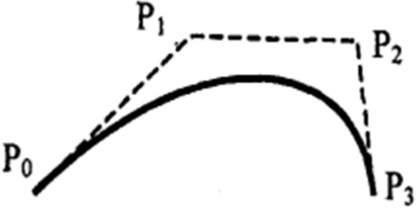


Рисунок 2 – пример кривой Catmull-Rom

Чтобы составная кривая Безье была геометрически непрерывной, необходимо, чтобы каждые три точки в месте стыковки лежали на одной прямой. Составную кривую построим из наборов элементарных кривых Безье для четверок вершин.

Допустим, у нас есть шесть точек P1, P2, P3, P4, P5, P6, где Pi(xi, yi). Пусть (x3; y3) и (x4; y4) — координаты третьей и четвертой точки соответственно. Вставляем между ними дополнительную точку P’ с координатами x’=(x3+x4)/2 и y’=(y3+y4)/2, после чего проводим одну кривую через точки P1, P2, P3, P’ а вторую — через точки P’, P4, P5, P6. В результате получим одну гладкую кривую для шести точек.

Если точек больше шести, их нужно разбить по такой же схеме и связать полученные группы с помощью дополнительных точек, как описано выше.

Последнюю точку можно повторить несколько раз, если множество точек не делится на целое число групп, чтобы кривая доходила до последней точки.

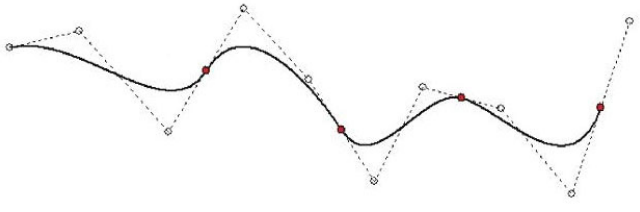
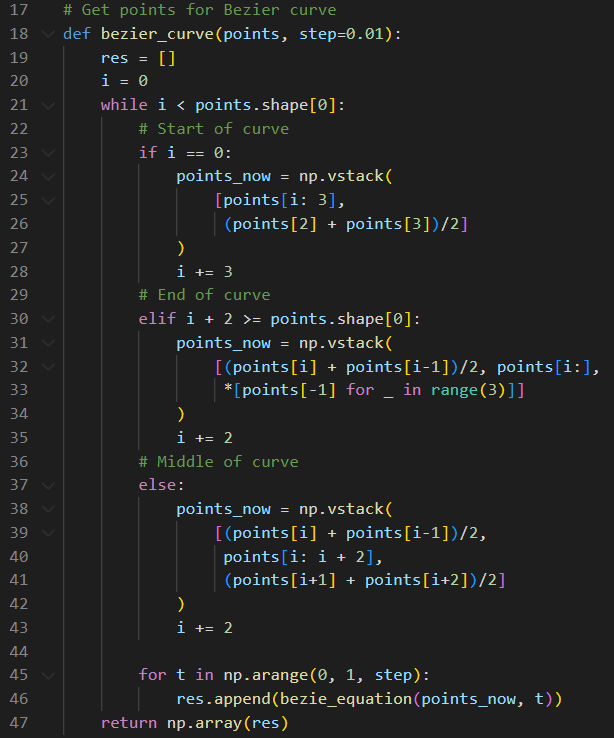


Рисунок 3 – пример аппроксимации кривой Безье

**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: ). Код снабжен комментариями и легок для понимания. Вся логика вычисления точек кривой Безье на основе опорных точек сосредоточена в функции bezier\_curve(см. рис. 4) .



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.

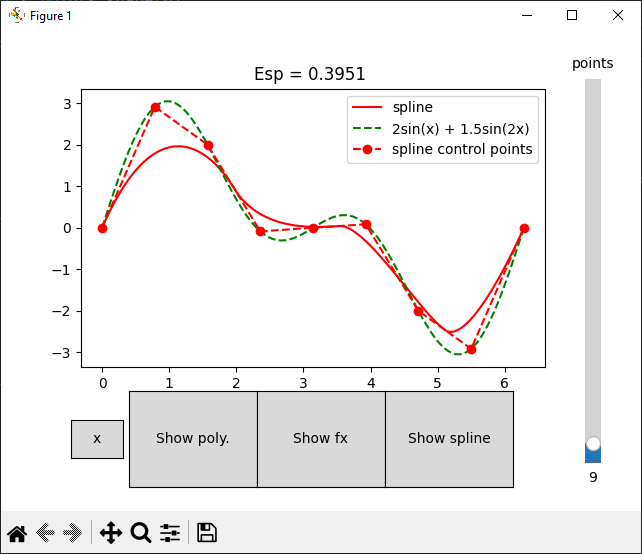


Рисунок 3 – кривая Безье для 9 контрольных точек

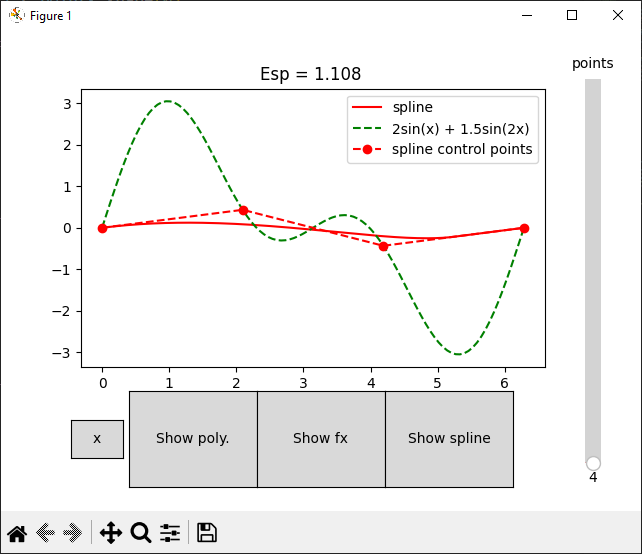


Рисунок 4 – кривая Безье для 4 точек

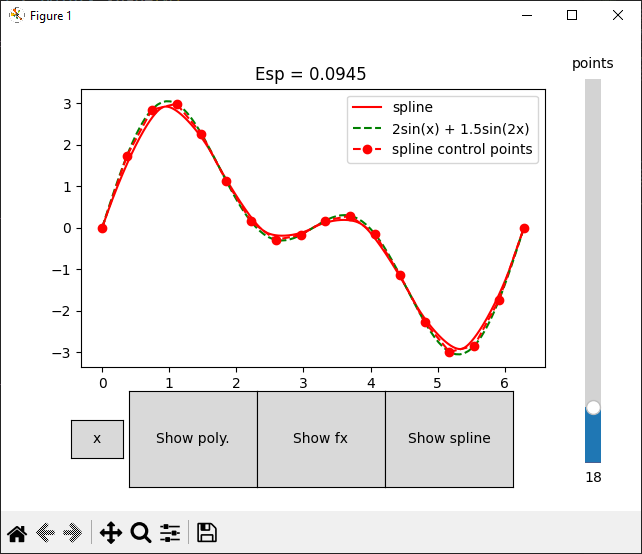


Рисунок 5 – кривая Безье для 18 точек

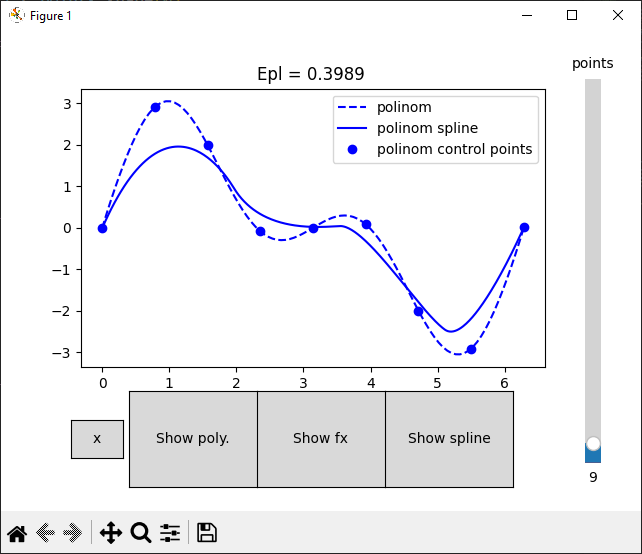


Рисунок 6 – кривая Безье для полинома 9-го порядка

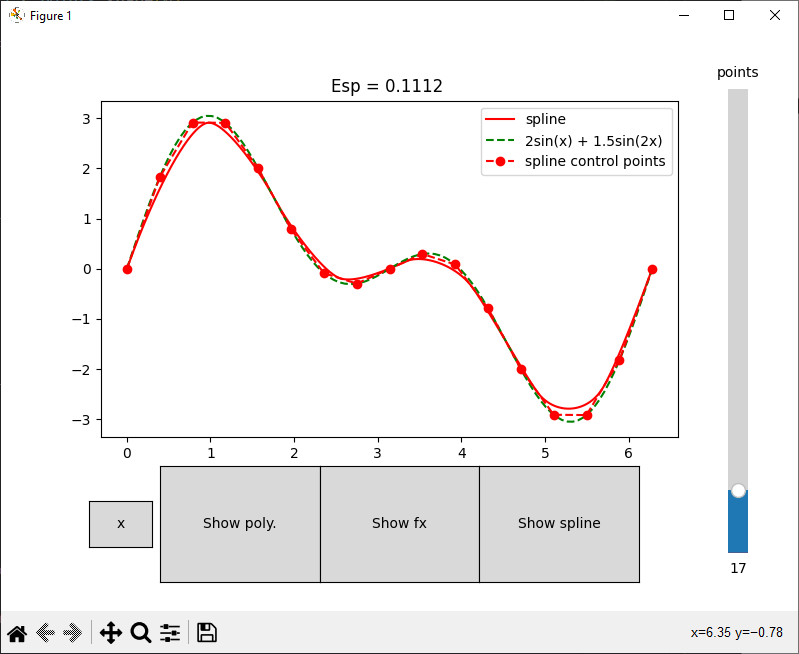


Рисунок 7 – кривая Безье для произвольного количества точек

# 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:

|  |  |
| --- | --- |
| N точек | Ошибка восстановления |
| 4 точки | 1.108 |
| 9 точек | 0.3951 |
| 18 точек | 0.0945 |

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. Половко А.М., Бутусов П.Н. Интерполяция. Методы и компьютерные

технологии их реализации. – СПб.: БХВ-Петербург, 2004. – 320 с. (дата обращения: 02.03.2023)

1. NumPy: Документация NumPy: сайт. – URL: https://numpy.org/doc/stable/index.html (дата обращения: 01.03.2023)
2. Учебник JavaScript: Кривые Безье: сайт. – URL: https://learn.javascript.ru/bezier-curve (дата обращения: 03.03.2023)
3. Matplotlib: Документация Matplotlib: сайт. – URL: https://matplotlib.org/stable/index.html (дата обращения: 01.03.2023)
4. StackOverflow: Кривые Безье для N точек: сайт. – URL: https://stackoverflow.com/questions/7715788/find-bezier-control-points-for-curve-passing-through-n-points (дата обращения: 03.03.2023)

# ПРИЛОЖЕНИЕ А. ЛИСТИНГ ПРОГРАММЫ

from math import sin

import numpy as np

import matplotlib.pyplot as plt

from matplotlib.widgets import Button, Slider

# Bezier function

def bezie\_equation(points, t):

p1 = points[0]

p2 = points[1]

p3 = points[2]

p4 = points[3]

q1 = ((1 - t) \* p1 + 3 \* t \* p2) \* (1 - t) + 3 \* (t \*\* 2) \* p3

return q1 \* (1 - t) + (t \*\* 3) \* p4

# Get points for Bezier curve

def bezier\_curve(points, step=0.01):

res = []

i = 0

while i < points.shape[0]:

# Start of curve

if i == 0:

points\_now = np.vstack(

[points[i: 3],

(points[2] + points[3])/2]

)

i += 3

# End of curve

elif i + 2 >= points.shape[0]:

points\_now = np.vstack(

[(points[i] + points[i-1])/2, points[i:],

\*[points[-1] for \_ in range(3)]]

)

i += 2

# Middle of curve

else:

points\_now = np.vstack(

[(points[i] + points[i-1])/2,

points[i: i + 2],

(points[i+1] + points[i+2])/2]

)

i += 2

for t in np.arange(0, 1, step):

res.append(bezie\_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"]):

# Polinom points

# 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 = bezier\_curve(

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["bezier"]):

# Spline points

splain = bezier\_curve(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["bezier"] = 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["bezier"] = 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, "bezier": 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()