ГУАП

КАФЕДРА № 42

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ _____

ПРЕПОДАВАТЕЛЬ

канд. техн. наук, доцент		А.В. Аграновский
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №6

Сплайновая кривая Безье

по курсу: КОМПЬЮТЕРНАЯ ГРАФИКА

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 4329		Д.С. Шаповалова
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2024

Содержание

1. Цель работы:

Изучение сплайновой кривой Безье, построение сплайновой кривой Безье с помощью математического пакета и/или языка программирования высокого уровня.

2. Задание:

Написать программу на языке программирования высокого уровня или с помощью математического пакета, которая выполняет построение кривой Безье и вычисляет ошибку восстановления. На форме должен находиться график, таблица с координатами опорных точек, а также три кнопки. При нажатии на кнопку 1 – выполнить построение графика гармонических колебаний, опорных точек и таблицы с координатами базовых точек. Кнопка 2 – построение кривой Безье на основе гармонических колебаний. Кнопка 3 – построение кривой Безье на основе полинома.

Для этого необходимо:

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.

Вариант 4329 - $f(x) = 1 - e^{-x^5}$, x = [0; 1,5], номер студента – 17.

3. Теоретические сведения:

Сплайн — это двумерные геометрические объекты, которые совершенно самостоятельны и могут служить основой для построения более сложных трехмерных тел. Внешне сплайны представляют собой разнообразные линии, форма линии определяется типом вершин, через которые она проходит. Сплайнами могут быть как простейшие геометрические фигуры: прямоугольники, звезды, эллипсы и пр., так и сложные ломаные или кривые, а также контуры текстовых символов.

Аппроксимация — это научный метод, который состоит в замене одного объекта другим, близким к исходному, но более простым. Это позволяет исследовать числовые характеристики и качественные свойства объекта, сводя задачу к изучению более простых или более удобных объектов. В различных областях науки, включая теорию чисел,

геометрию и вычислительную математику, используются различные виды аппроксимации, такие как диофантовы приближения, аппроксимация кривых ломаными и теория приближения функций. Также существует понятие интерполяции, которое представляет собой особый вид аппроксимации, когда кривая построенной функции проходит точно через известные точки данных.

Полином:

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

Кривая Безье — это частный случай многочленов Бернштейна, используется в компьютерной графике для рисования плавных изгибов. Обладает рядом интересных свойств:

- проходит внутри выпуклой оболочки, заданной опорными точками;
- набор базовых функций однозначно определяет кривую, т.е. нет возможности регулировать ее форму.
- точки не всегда на кривой.

Для создания кривой Безье требуется набор из четырёх точек P_0, P_1, P_2 и P_3, которые задают сегмент кривой между центральными точками P_1 и P_2. При построении кривой используется параметр t, который варьируется от 0 до 1 и позволяет получить промежуточные значения между P_1 и P_2. Уравнение для каждой точки на сегменте задается следующим образом (1):

$$R(t) = \Big(\big((1-t)P_0 + 3tP_1\big)(1-t) + 3t^2 P_2\Big)(1-t) + t^3 P_3, \quad 0 \le t \le 1. \quad (2)$$

где P_0, P_1, P_2, P_3 — координаты точек, определяющих сегмент;

Элементарная кривая начинается в точке P_0 и заканчивается в точке P_3, касаясь при этом отрезков P_0P_1 и P_2P_3.

Этот метод применим к любому участку кривой. Он позволяет объединить несколько участков в единый сплайн, который проходит через все опорные точки.

Для построения кривой Безье можно использовать любое количество точек. Однако вычисление полиномов, имеющих степень более 10 000, представляет собой сложную задачу.

Поэтому применяется следующий подход: точки делятся на группы по четыре, для каждой группы строится кривая Безье, а затем полученные сегменты соединяются в одну кривую. Это значительно упрощает процесс поддержки и вычислений.

Однако стоит отметить, что полученная кривая может быть не слишком гладкой на границах сегментов.

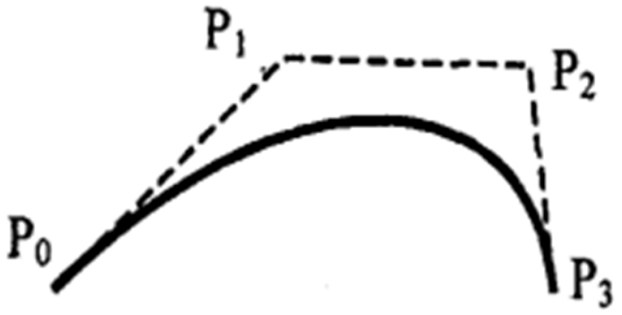

Рисунок 1 – Сплайновая кривая Безье

Чтобы составная кривая Безье была непрерывной, необходимо, чтобы три точки в месте стыка лежали на одной прямой.

Составную кривую можно построить из наборов элементарных кривых Безье для четырёх вершин.

Предположим, у нас есть шесть точек P_1, P_2, P_3, P_4, P_5 и P_6 с координатами $(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4), (x_5, y_5)$ и (x_6, y_6).

Допустим, что координаты третьей и четвёртой точки — (x_3, y_3) и (x_4, y_4).

Добавим между ними точку P' с координатами (x', y'), где $x' = \frac{(x_3 + x_4)}{2}$ и $y' = \frac{(y_3 + y_4)}{2}$. Затем проведём одну кривую через точки P_1, P_2, P_3 и P', а вторую — через точки P', P_4, P_5 и P_6. В результате получим одну гладкую кривую для шести точек.

Если учесть поведение кривой Безье и то, что точки $(x_3, y_3), (x', y')$ и (x_4, y_4) лежат на одной прямой, это становится очевидным.

Если точек больше шести, их можно разбить на группы по той же схеме и связать эти группы с помощью дополнительных точек, как описано выше.

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

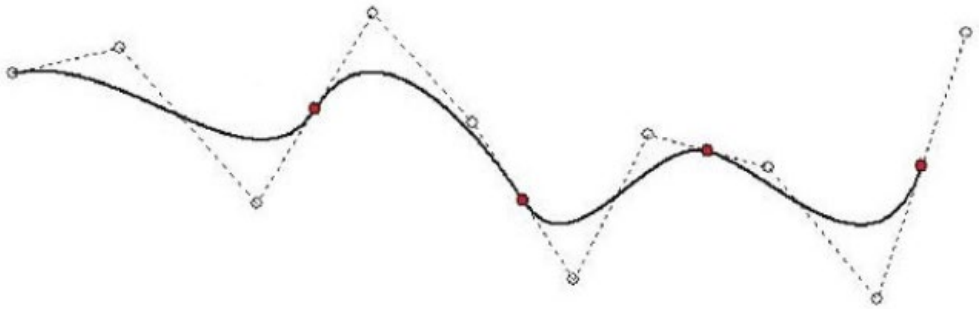

Рисунок 2 – Пример аппроксимации кривой Безье

Для оценки точности построенной кривой в работе применяют метод средней абсолютной ошибки (MAE). Он рассчитывается как среднее значение абсолютных отклонений между значениями исходной функции и значениями аппроксимации.

Ошибка восстановления считается по формуле (2):

$$\frac{1}{N}\sum_{i=1}^{N}\left|y_{\text{истинное}}\,(x_i) - \,y_{\text{апрокс}}(x_i)\right|, \tag{3}$$

где $y_{\text{истинное}}$ — значения исходной функции;

$y_{\text{аппрокс}}$ — значения, полученные с помощью кривой Безье.

Чтобы построить функцию, которая будет похожа на исходные данные, используют полиномиальную аппроксимацию. Потом с помощью этой функции строят кривую Безье. Так можно понять, насколько точно работает кривая Безье, если использовать не настоящие значения функции, а приближённые.

4. Листинг с кодом программы:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import PchipInterpolator
import tkinter as tk
from tkinter import ttk
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import math

N = 21  # Количество точек (4+17)
Z = 10  # Количество точек после запятой для ошибки восстановления
x_range = np.linspace(0, 1.5, 500)  # Диапазон значений x для построения графиков


def f(x):
    return 1 - np.exp(-x ** 5)


def calculate_error(true_vals, approx_vals):
    return np.mean(np.abs(true_vals - approx_vals))  # Среднее абсолютное отклонение


# Функция для кубической кривой Безье
def bezier_curve(control_points, num_points=100):
    n = len(control_points) - 1
    t_values = np.linspace(0, 1, num_points)
    bezier_points = np.zeros((num_points, 2))

    for i in range(n + 1):
        binomial_coefficient = math.factorial(n) / (math.factorial(i) * math.factorial(n - i))
        bezier_points[:, 0] += binomial_coefficient * ((1 - t_values) ** (n - i)) * (t_values ** i) * control_points[
            i, 0]
        bezier_points[:, 1] += binomial_coefficient * ((1 - t_values) ** (n - i)) * (t_values ** i) * control_points[
            i, 1]

    return bezier_points[:, 0], bezier_points[:, 1]


# Функция для построения графика гармонических колебаний
def plot_f(points_count):
    fig.clear()
    x_points = np.linspace(0, 1.5, points_count)
    y_points = f(x_points)
    y_true = f(x_range)
    ax = fig.add_subplot(111)
    ax.plot(x_range, y_true, label="Гармонические колебания", color='cyan', linewidth=1.5)
    ax.scatter(x_points, y_points, color='red')
    ax.legend()
    ax.set_title("Гармонические колебания")
    ax.set_xlabel("x")
    ax.set_ylabel("y")
    for i in table.get_children():
        table.delete(i)
    for x, y in zip(x_points, y_points):
        table.insert("", "end", values=(f"{x:.3f}", f"{y:.3f}"))
    canvas.draw()
```

```
# Функция для построения кривой Безье
def plot_bezier_curve(points_count):
    fig.clear()
    x_points = np.linspace(0, 1.5, points_count)
    y_points = f(x_points)
    control_points = np.vstack((x_points, y_points)).T
    bezier_x, bezier_y = bezier_curve(control_points)
    y_true = f(bezier_x)
    error = calculate_error(y_true, bezier_y)
    ax = fig.add_subplot(111)
    ax.plot(bezier_x, bezier_y, label="Кривая Безье на основе функции",
color='orange', linestyle='--')
    ax.scatter(x_points, y_points, color='blue')
    ax.legend()
    ax.set_title(f"Кривая Безье на основе функции\nОшибка восстановления:
{error:.{Z}f}")
    ax.set_xlabel("x")
    ax.set_ylabel("y")
    canvas.draw()


# Функция для построения кривой Безье на основе полинома
def plot_bezier_curve_polynomial(points_count):
    fig.clear()
    x_points = np.linspace(0, 1.5, points_count)
    coefficients = np.polyfit(x_points, f(x_points), points_count - 1)
    polynomial = np.poly1d(coefficients)
    y_points = polynomial(x_points)
    control_points = np.vstack((x_points, y_points)).T
    bezier_x, bezier_y = bezier_curve(control_points)
    y_true = f(bezier_x)
    error = calculate_error(y_true, bezier_y)
    ax = fig.add_subplot(111)
    ax.plot(bezier_x, bezier_y, label="Кривая Безье на основе полинома",
color='green', linestyle='--')
    ax.scatter(x_points, y_points, color='violet')
    ax.legend()
    ax.set_title(f"Кривая Безье на основе полинома\nОшибка восстановления:
{error:.{Z}f}")
    ax.set_xlabel("x")
    ax.set_ylabel("y")
    canvas.draw()


# Создание интерфейса
root = tk.Tk()
root.title("Интерполяционная кривая Безье")
root.geometry("800x800")

# График
fig = plt.Figure(figsize=(6, 4), dpi=100)
canvas = FigureCanvasTkAgg(fig, master=root)
canvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1, pady=(0, 5))

# Таблица
table_frame = tk.Frame(root)
table_frame.pack(side=tk.TOP, fill=tk.BOTH, padx=10, pady=(0, 10))

table = ttk.Treeview(table_frame, columns=("X", "Y"), show='headings')
table.heading("X", text="X")
table.heading("Y", text="Y")
```

```
table.pack()


def button1_action():
    plot_f(points_count=N)


def button2_action():
    plot_bezier_curve(points_count=N)


def button3_action():
    plot_bezier_curve_polynomial(points_count=N)


# Панель с кнопками
button_frame = tk.Frame(root)
button_frame.pack(side=tk.BOTTOM, fill=tk.X, padx=10, pady=(0, 10))

# Кнопки
btn1 = tk.Button(button_frame, text="1 - e^(-x^5)", command=button1_action)
btn1.pack(side=tk.LEFT, padx=(0, 5))

btn2 = tk.Button(button_frame, text="Кривая Безье на основе функции",
command=button2_action)
btn2.pack(side=tk.LEFT, padx=(0, 5))

btn3 = tk.Button(button_frame, text="Кривая Безье на основе полинома",
command=button3_action)
btn3.pack(side=tk.LEFT)

# Запуск интерфейса
root.mainloop()
```

5. Скриншоты, иллюстрирующие результаты работы программы:

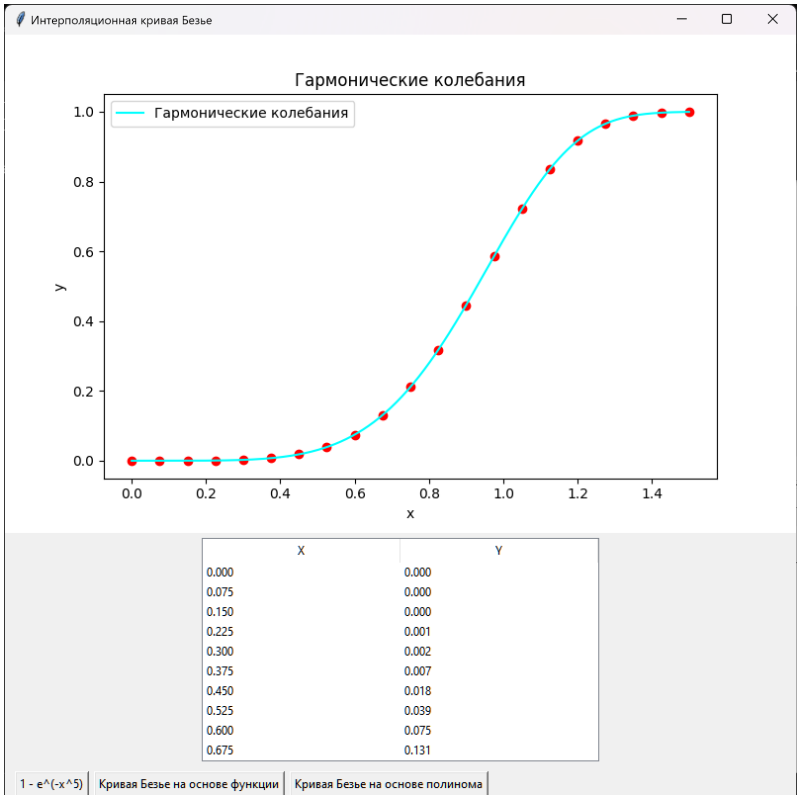

Рисунок 3.1 – Построение графика гармонических колебаний, 21 опорная точка

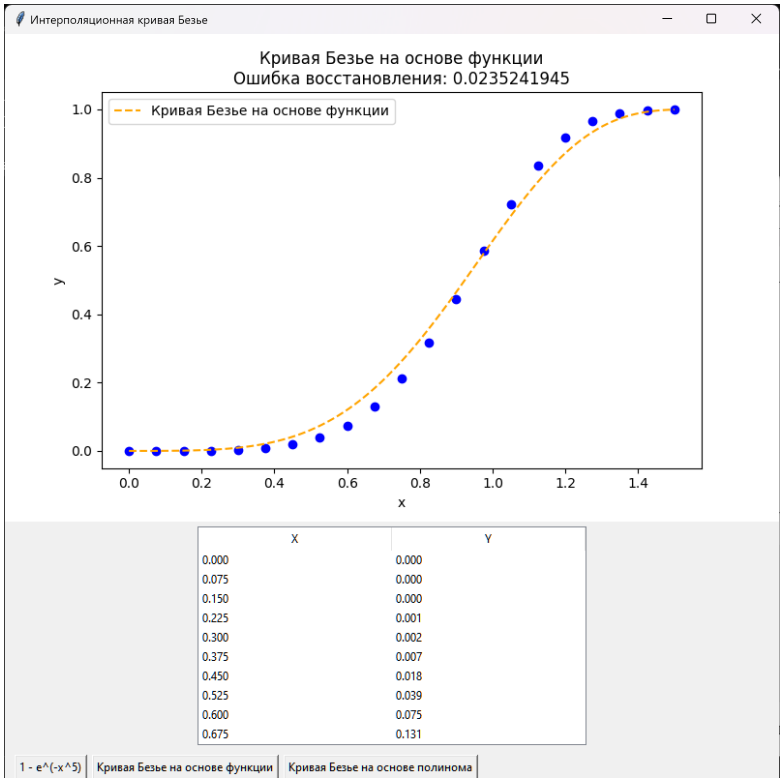

Рисунок 3.2 – График кривой Безье на основе функции, 21 опорная точка

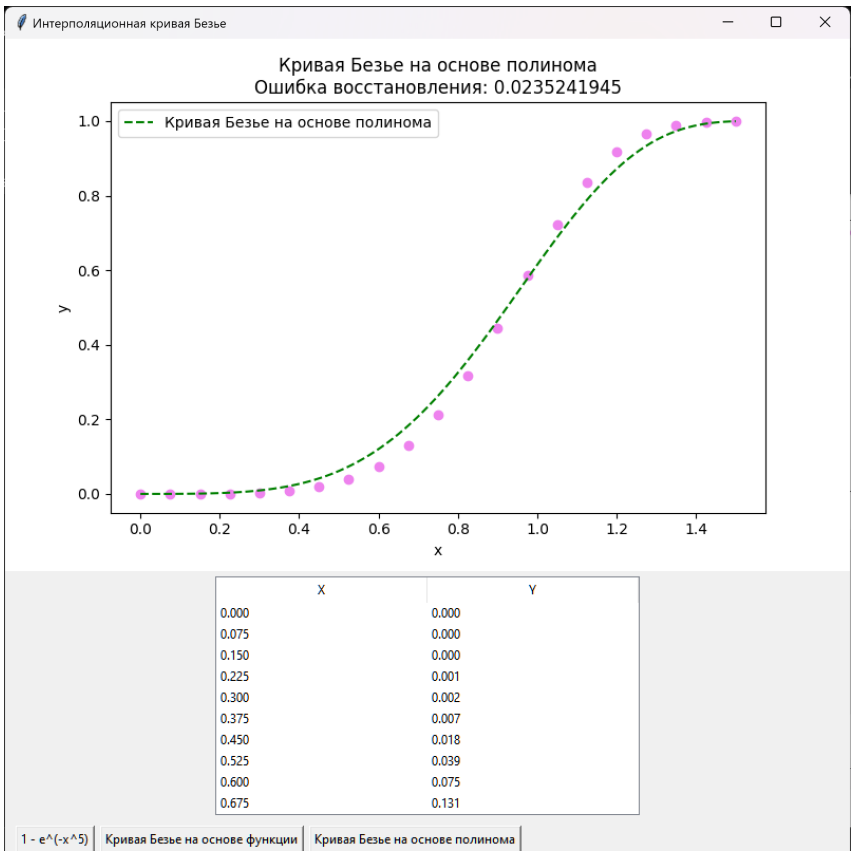

Рисунок 3.3 – График кривой на основе полинома, 21 опорная точка

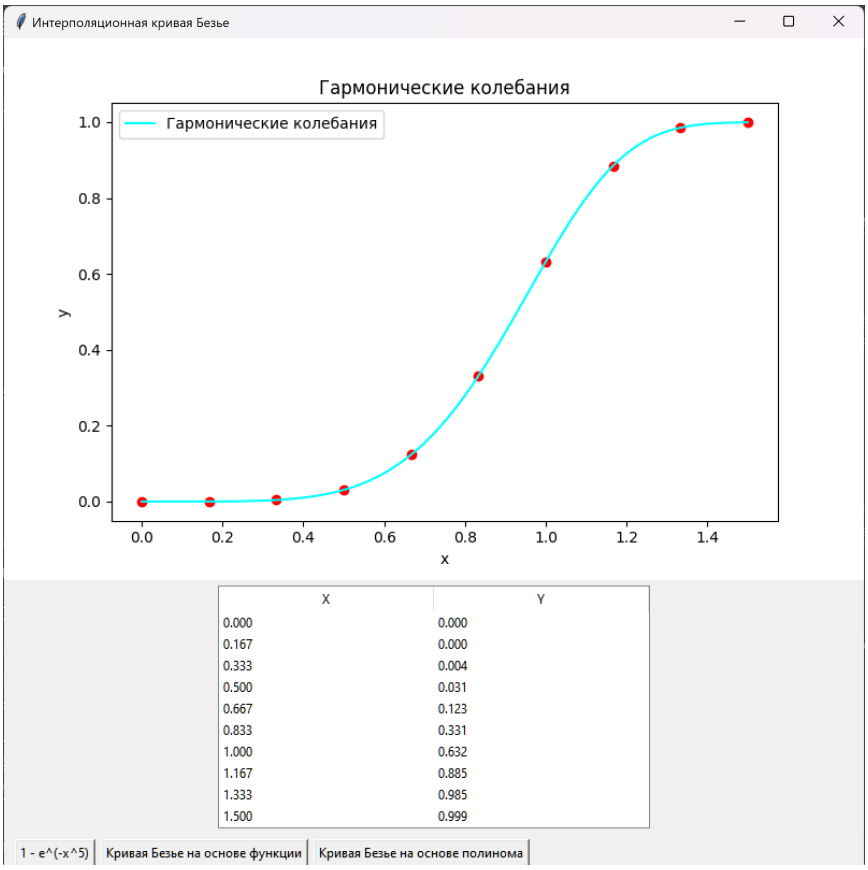

Рисунок 3.4 – Построение графика гармонических колебаний, 10 опорных точек

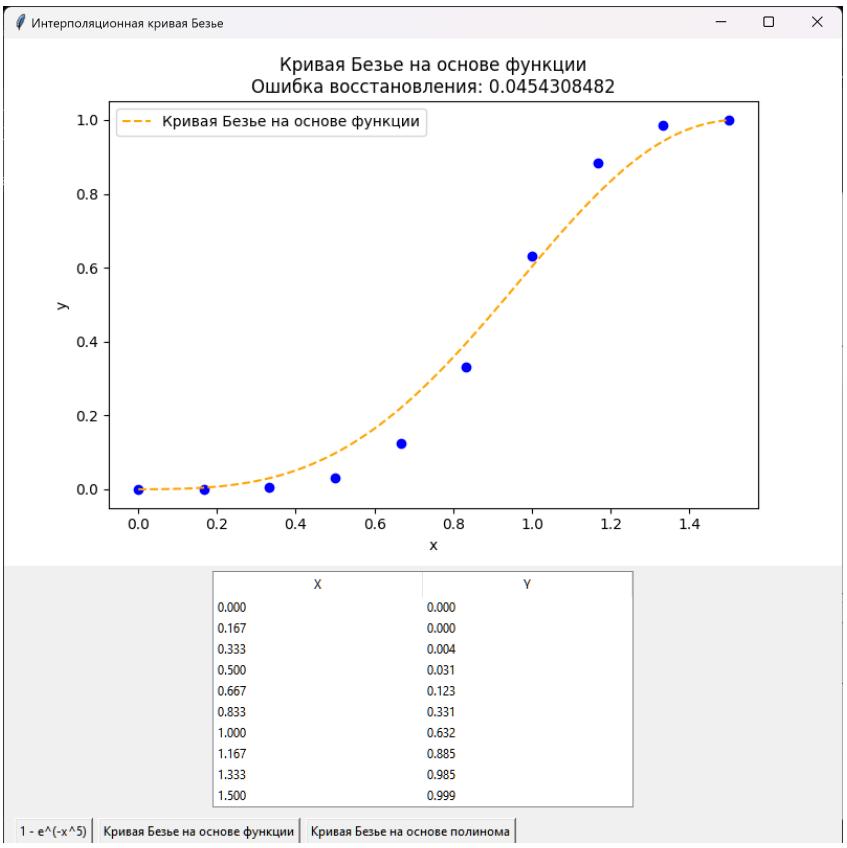

Рисунок 3.5 – График кривой Безье на основе функции, 10 опорных точек

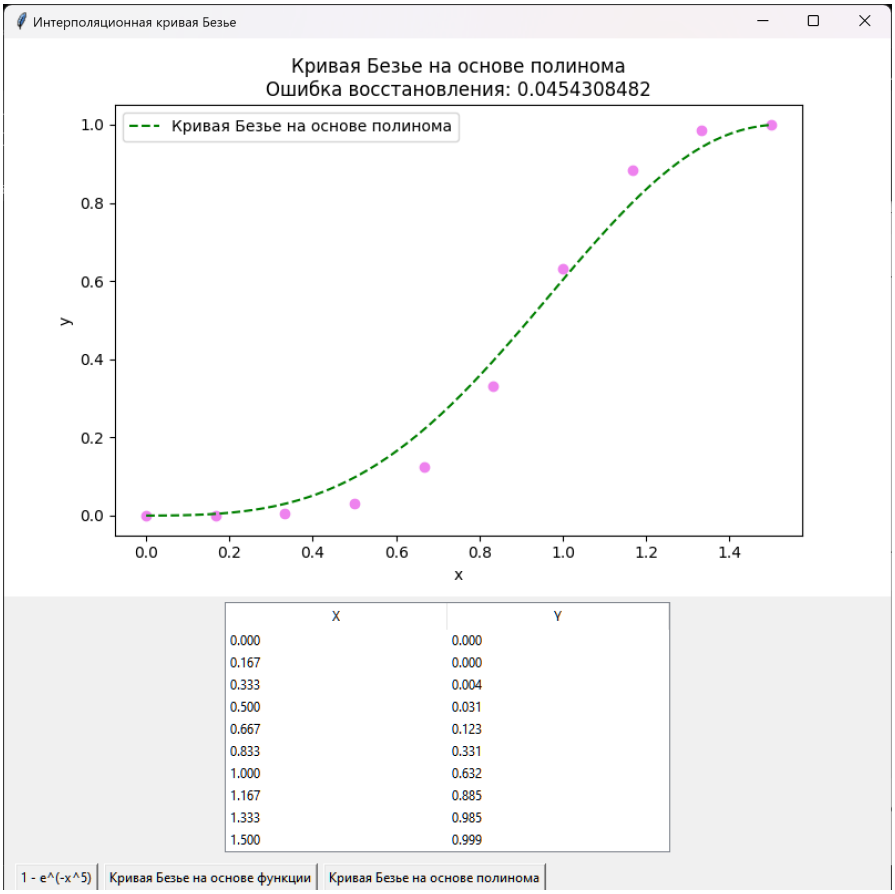

Рисунок 3.6 – График кривой на основе полинома, 10 опорных точек

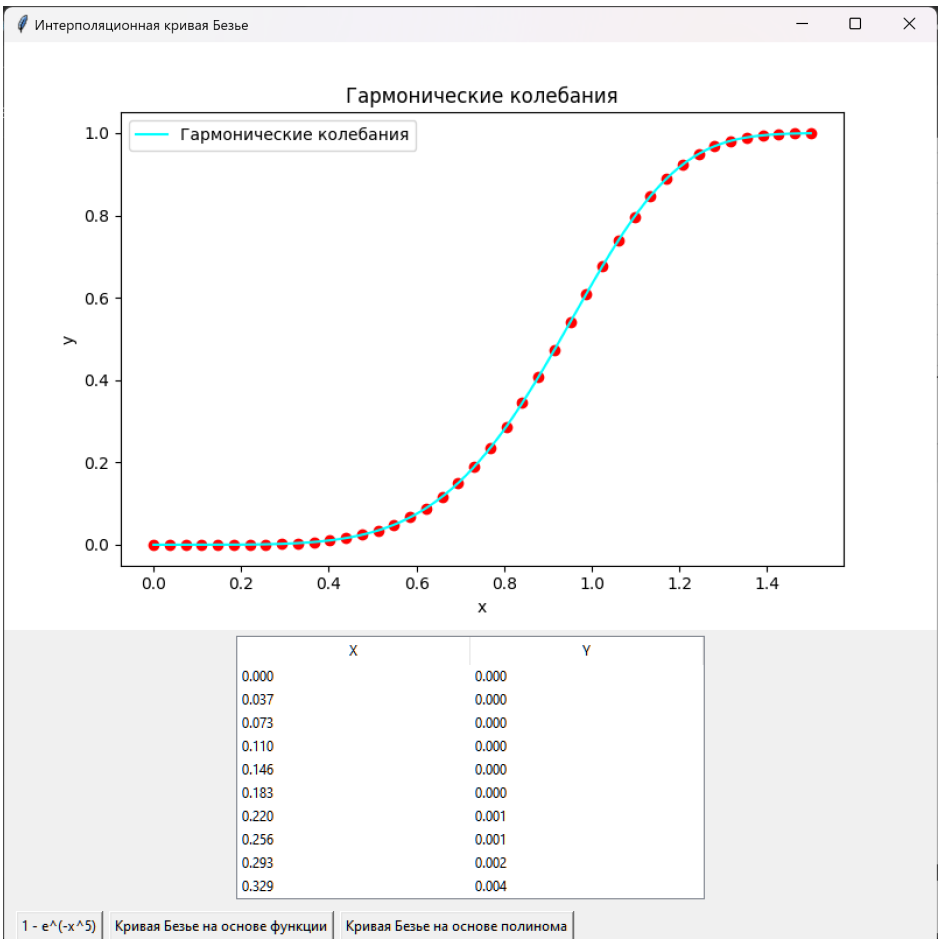

Рисунок 3.7 – Построение графика гармонических колебаний, 42 опорные точки

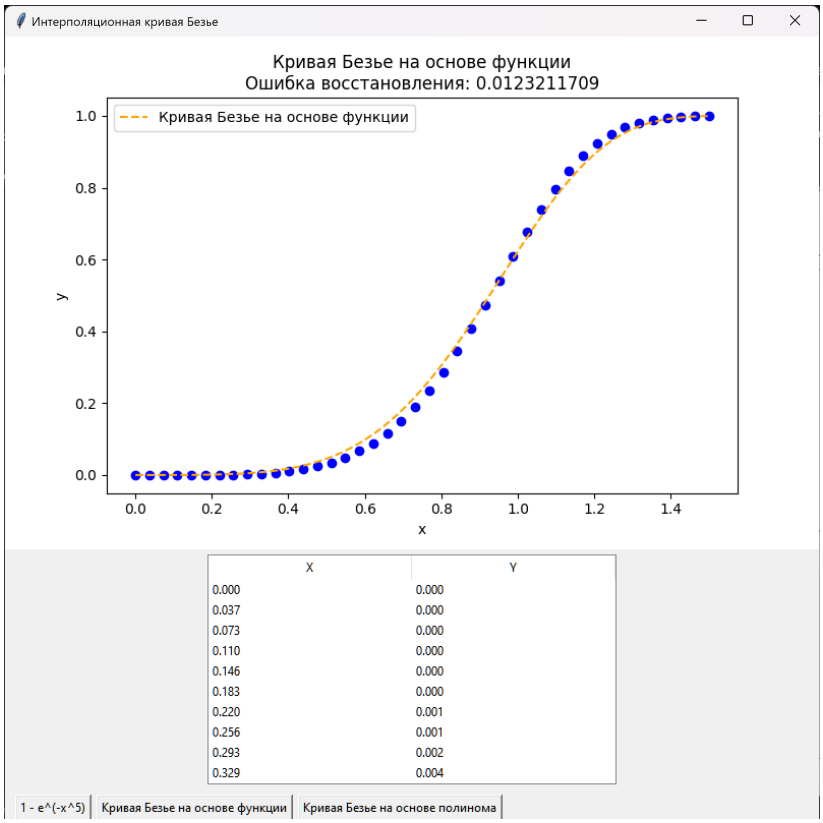

Рисунок 3.8 – График кривой Безье на основе функции, 42 опорных точки

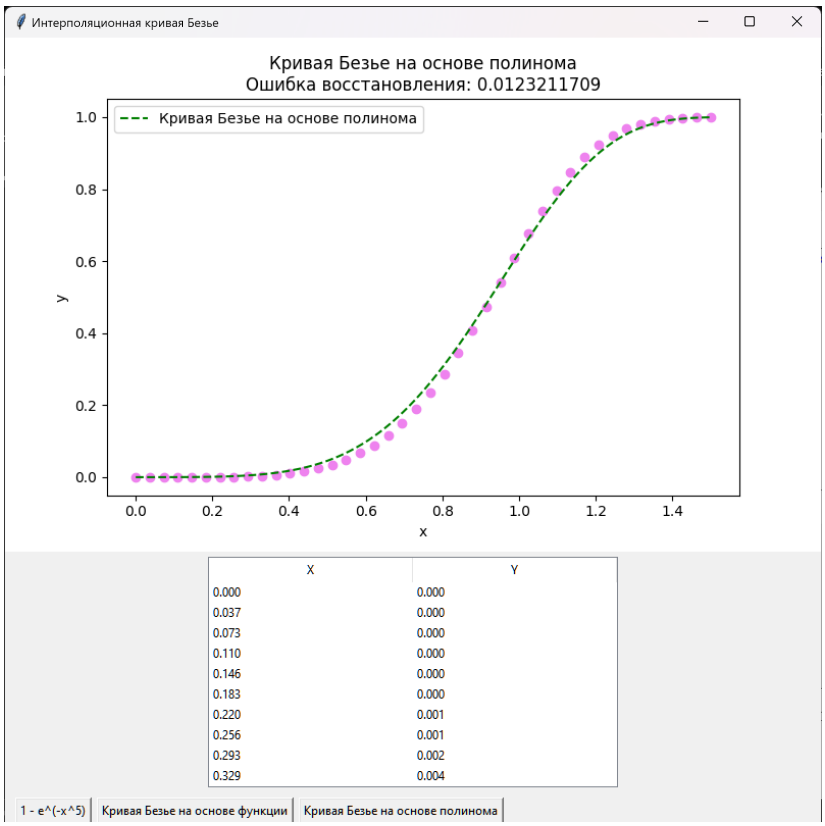

Рисунок 3.9 – График кривой на основе полинома, 42 опорных точки

6. Вывод:

В данной работе мы подробно рассмотрели, как создавать кривые, которые точно проходят через заданные точки, используя метод сплайнов. Мы также проанализировали способы построения сплайнов на основе исходной функции и её приближения многочленами.

Одним из ключевых элементов нашего исследования стала кривая Безье, точки разбивают на группы по 4. Для каждой группы строят кривую Безье. Потом соединяют все кривые в одну. Чтобы кривая была непрерывной, нужно, чтобы три точки в месте стыка лежали на одной прямой. Кривую Безье строят из наборов кривых для 4 вершин. Эта кривая нашла широкое применение в анимации, дизайне и других областях, где требуются плавне изгибы.

Мы изучили два метода построения сплайнов: на основе исходной функции и её приближения многочленами, что позволило нам сравнить их и определить наиболее эффективный подход.

Мы также освоили метод оценки точности приближения, используя среднюю абсолютную ошибку. Этот показатель позволяет понять, насколько точно кривая соответствует исходной функции, что имеет важное значение для достижения наилучших результатов.

Мы осознали, что полиномиальная аппроксимация может сделать кривую более гладкой, однако иногда она не всегда точно передаёт форму исходной функции. В зависимости от количества опорных точек, использование полиномиальной аппроксимации может помочь уменьшить ошибку, но если точек достаточно, то лучше использовать саму функцию для вычисления опорных точек, что обеспечит более точный и естественный результат.

Анализ ошибок восстановления:

Были проведены эксперименты с изменением количества опорных точек для построения кривой Безье с целью проанализировать точность и ошибки восстановления.

Для 21 точки ошибка восстановления составила 0,0235241945.

Для 10 точек ошибка восстановления составила 0,0454308482.

Для 42 точек ошибка восстановления составила 0,0123211709.

Это говорит о том, что чем больше опорных точек, тем точнее кривая Безье.

В общем:

Исследование показало, что кривая Безье — это хороший способ сделать линию или фигуру плавной. Было выяснено, что полиномиальная аппроксимация позволяет упростить расчёты и слабо влияет на ошибку восстановления. Экспериментальным путём было найдена закономерность – чем больше точек, тем меньше ошибка восстановления, тем точнее строится кривая Безье.

В ходе работы мы узнали о методах интерполяции и аппроксимации функций, изучили кривую Безье и научились оценивать ошибки восстановления.

Кривые Безье полезны в компьютерной графике, дизайне, потому что они позволяют сделать плавные изгибы. Это важно для создания анимации, шрифтов, иконок и других векторных картинок с необходимостью просто масштабировать фигуру без потери качества.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Шикин Е.В., Плис Л.И. Кривые и поверхности на экране компьютера. Руководство по сплайнам для пользователей. М.: ДИАЛОГ-МИФИ, 1996. – 286 с.
2. Интерполяция – URL: https://help.fsight.ru/ru/mergedProjects/lib/03_transformations/uimodelling_interpolation.htm (дата обращения 10.11.2024)
3. Что такое сплайны – URL: http://cpu3d.com/lesson/chto-takoe-splayny/ (дата обращения 10.11.2024)
4. Что такое интерполяция и зачем она нужна? // Хабр. URL: https://habr.com/ru/articles/323442/ (дата обращения: 09.11.2024).
5. ru.wikipedia.org:Аппроксимация - https://ru.wikipedia.org/wiki?curid=42664 (дата обращения 16.11.2024)
6. Кривые Безье – URL: https://learn.javascript.ru/bezier-curve (дата обращения 17.11.2024)
7. Аппроксимация функции с помощью сплайна Безье | Записки программиста – URL: https://eax.me/bezier-spline/ (дата обращения 17.11.2024)