Национальный исследовательский университет «МЭИ»

Институт информационных и вычислительных технологий

**Кафедра вычислительных технологий**

Практическое задание

**по курсу «Геометрическое моделирование»**

|  |  |
| --- | --- |
|  | **Выполнил** |
| **Студент** | Зинченко Семен Константинович |
| **Группа** | А-06-20 |
| **Дата** |  |

|  |  |
| --- | --- |
|  | **Принял** |
| **Преподаватель** | Лешихина Ирина Евгеньевна |
| **Оценка** |  |
| **Дата** |  |

Москва 2023

1. **ЗАДАНИЕ**

Построить двузначную кривую на основе периодического В-сплайна 4-ой степени. Число опорных точек не меньше 21. Для визуализации использовать ортографическую проекцию.

1. **РЕШЕНИЕ**
   1. **Вектор параметризации**

p = 4

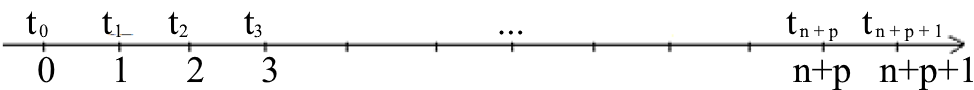
n >= 20

Длина вектора параметризации: m = n + p + 2 >= 26

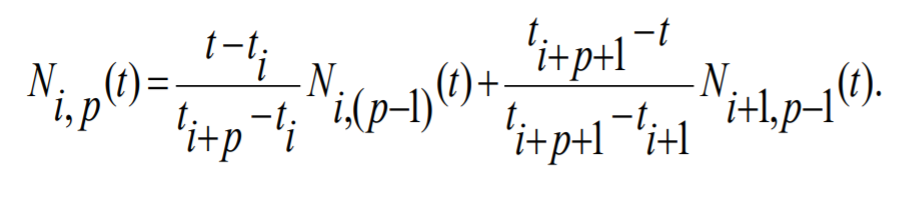
Число существенных интервалов: int = n – p + 1 >= 17

Число собственных интервалов: p + 1 = 5

Вектор параметризации: T = {0, 1, 2, 3, … , n + p, n + p + 1}



* 1. **Элементарные B-сплайны**



Формула базисной функции

Код для нахождения элементарных B-сплайнов:

# Рекурсивное получение интервала параметризации  
def get\_part(self, power, region, parameters): # Передаём степень сплайна, номер интервала и параметры  
 # Базис рекурсии  
 if region < 0 or region > power:  
 return [0] \* (power + 1)  
 if power == 0:  
 return [1]  
 rez = [0] \* (power + 1) # Список коэффициентов при t от p до 0  
 # Получение коэффициентов параметризации меньшей степени  
 prev\_left = self.get\_part(power - 1, region, parameters[:len(parameters) - 1])  
 prev\_right = self.get\_part(power - 1, region - 1, parameters[1:])  
 # Коэффициенты при t и и не t в формуле Кокса де Бура  
 t\_left = 1 / (parameters[power] - parameters[0])  
 left = -1 \* parameters[0] / (parameters[power] - parameters[0])  
 right = parameters[power + 1] / (parameters[power + 1] - parameters[1])  
 t\_right = -1 / (parameters[power + 1] - parameters[1])  
 # Получаем список коэффициентов по формуле Кокса де Бура  
 for i in range(power):  
 # Так как умножаем на коэффициент при t прибавляем к большей степени  
 rez[i] += prev\_left[i] \* t\_left + prev\_right[i] \* t\_right  
 # Так как умножаем на коэффициент без t прибавляем к текущей степени  
 rez[i + 1] += prev\_left[i] \* left + prev\_right[i] \* right  
 return rez

В результате будет получен массив длины (p+1), в котором будут коэффициенты при степенях t от p-ой, до нулевой.

1. **Построение сплайна**

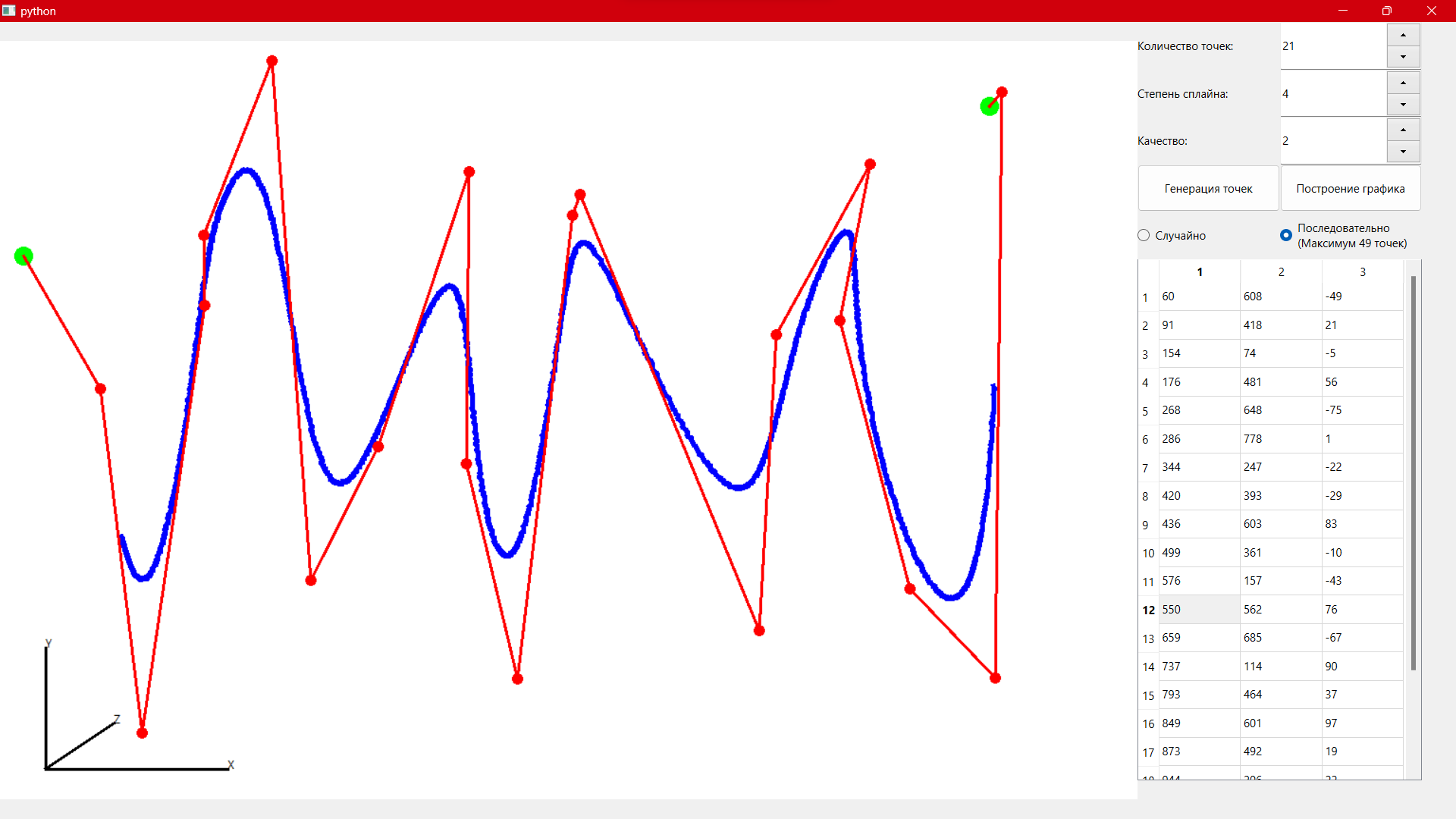
В процессе работы я понял, что могу не ограничиваться построением B-сплайна только 4 степени и только по 21 точке, поэтому в своей программе добавил возможность выбора степени и количества точек.

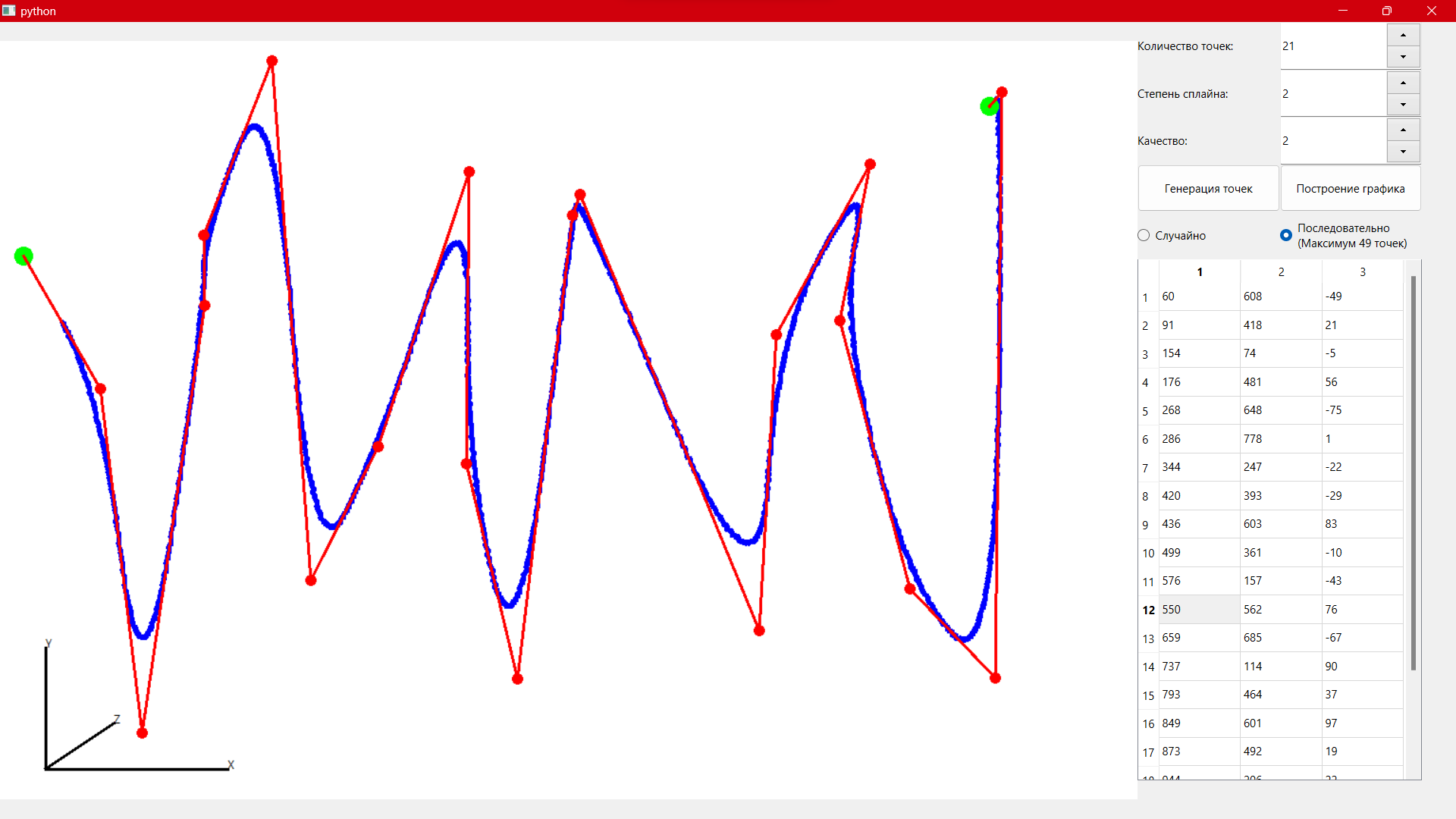
Также был добавлен изменяемый параметр «Качество», от которого зависит количество промежуточных точек между параметрами при построении.

Для удобства добавлена кнопка «Генерация точек» в двух вариациях:  
а) точки будут равномерно (последовательно) распределены по оси x;  
б) положение точек будет абсолютно случайным.  
Точки также можно изменить вручную в таблице

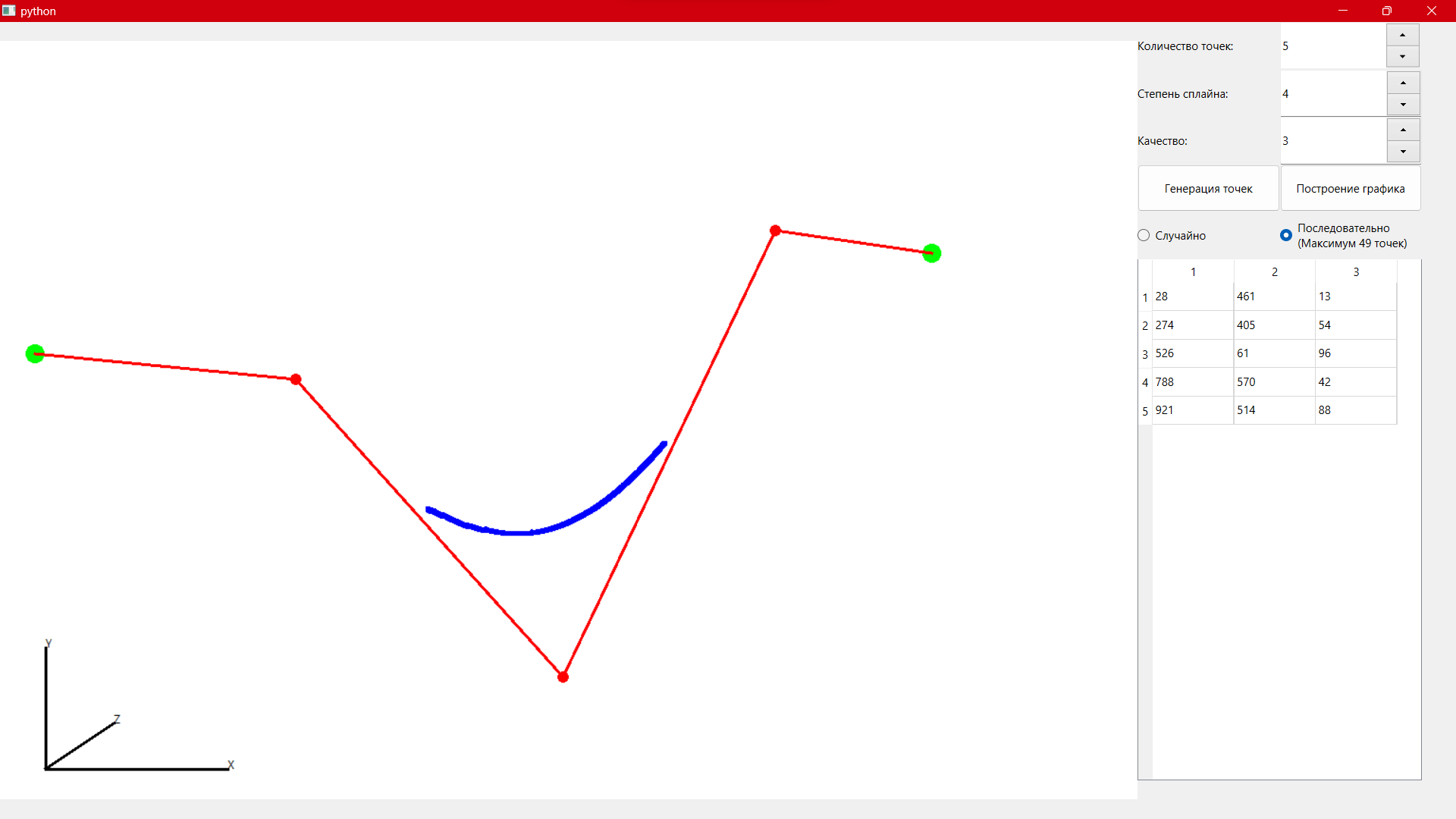
По кнопке «Построение графика» будет построен сплайн по точкам из таблицы и указанным параметрам.

Примеры построенных сплайнов:

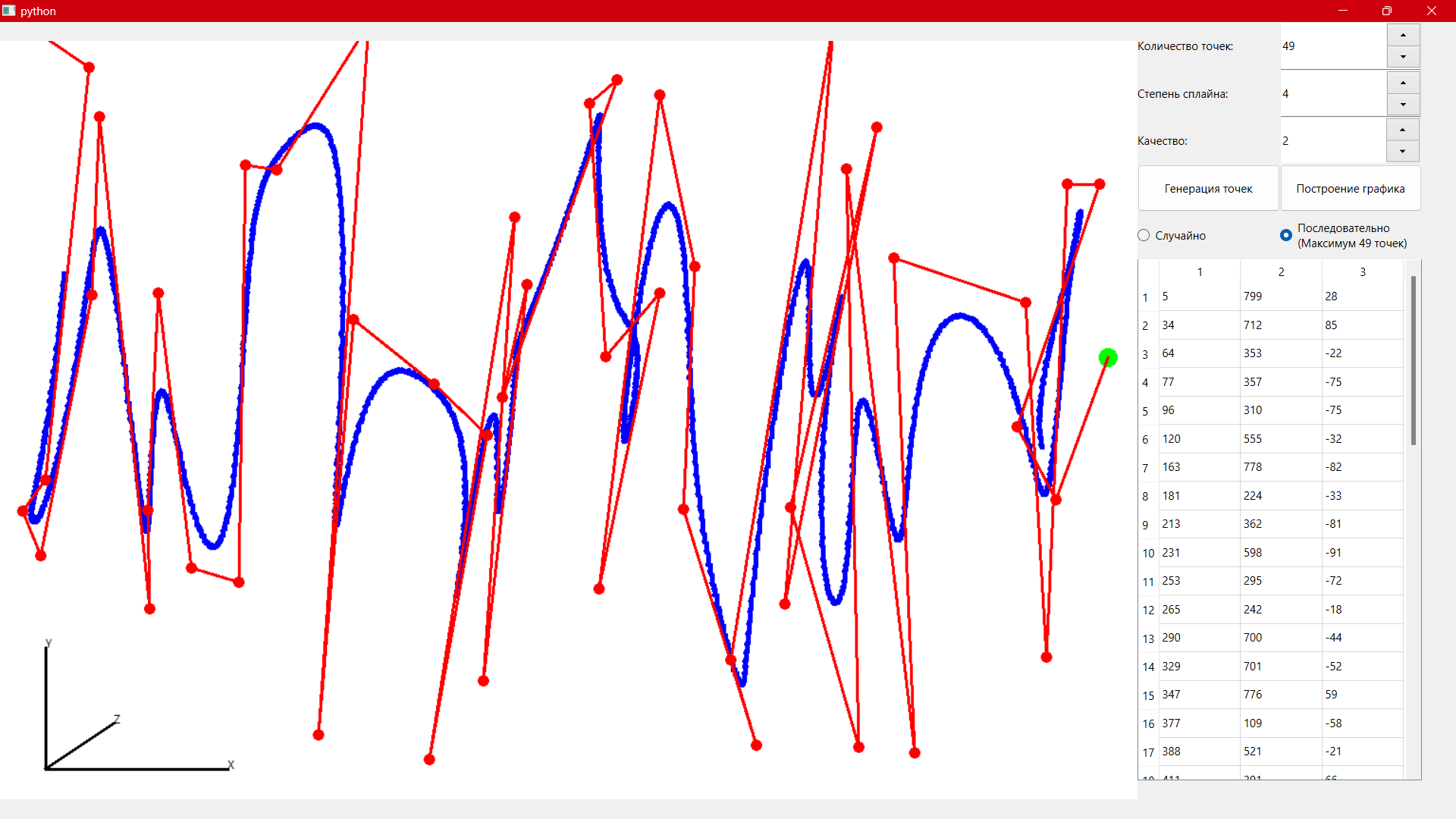
  
4 степень, 21 опорная точка (условия гз)



Те же точки, но вторая степень

  
Сплайн четвёртой степени по пяти опорным точкам

И ещё 3 не совсем информативных, но интересных сплайна по большому количеству точек:



49 точек и сплайн 4 степени



Те же 49 точек, но случайно распределённые по области построения

100 точек случайно распределённые по области построения

Полный код программы:

from random import randrange  
from PyQt6.QtWidgets import QMainWindow, QApplication  
from PyQt6.QtWidgets import QLabel, QSpinBox, QPushButton, QRadioButton, QTableWidget, QTableWidgetItem  
from PyQt6.QtGui import QPen, QPixmap, QPainter, QColor  
from PyQt6.QtCore import Qt, QPoint  
from math import floor  
  
  
class MainWindow(QMainWindow):  
 # Функция инициализации окна, только графика  
 def \_\_init\_\_(self, width, height, draw\_width):  
 self.draw\_width, self.draw\_height = draw\_width, height  
 widg\_width, widg\_height = floor((width - draw\_width) / 2), floor(height / 16)  
  
 super().\_\_init\_\_()  
 self.setGeometry(0, 0, width, height)  
  
 self.label\_draw = QLabel()  
 self.setCentralWidget(self.label\_draw)  
 self.clear\_graph()  
  
 self.label\_np = QLabel("Количество точек:", self)  
 self.label\_np.setGeometry(draw\_width, 0, widg\_width, widg\_height)  
 self.SB\_np = QSpinBox(self)  
 self.SB\_np.setGeometry(draw\_width + widg\_width, 0, widg\_width, widg\_height)  
 self.SB\_np.setRange(2, 100) # 49  
 self.SB\_np.setValue(21)  
 self.SB\_np.textChanged.connect(self.clear\_table)  
  
 self.label\_pow = QLabel("Степень сплайна:", self)  
 self.label\_pow.setGeometry(draw\_width, widg\_height, widg\_width, widg\_height)  
 self.SB\_pow = QSpinBox(self)  
 self.SB\_pow.setGeometry(draw\_width + widg\_width, widg\_height, widg\_width, widg\_height)  
 self.SB\_pow.setRange(0, 5)  
 self.SB\_pow.setValue(4)  
 self.SB\_pow.textChanged.connect(self.clear\_graph)  
  
 self.label\_qual = QLabel("Качество:", self)  
 self.label\_qual.setGeometry(draw\_width, 2 \* widg\_height, widg\_width, widg\_height)  
 self.SB\_qual = QSpinBox(self)  
 self.SB\_qual.setGeometry(draw\_width + widg\_width, 2 \* widg\_height, widg\_width, widg\_height)  
 self.SB\_qual.setRange(0, 3)  
 self.SB\_qual.setValue(2)  
 self.SB\_qual.textChanged.connect(self.clear\_graph)  
  
 self.PB\_gen = QPushButton("Генерация точек", self)  
 self.PB\_gen.setGeometry(draw\_width, 3 \* widg\_height, widg\_width, widg\_height)  
 self.PB\_gen.clicked.connect(self.gen\_points)  
  
 self.PB\_draw = QPushButton("Построение графика", self)  
 self.PB\_draw.setGeometry(draw\_width + widg\_width, 3 \* widg\_height, widg\_width, widg\_height)  
 self.PB\_draw.clicked.connect(self.drawing)  
  
 self.RB\_rand = QRadioButton("Случайно", self)  
 self.RB\_rand.setGeometry(draw\_width, 4 \* widg\_height, widg\_width, widg\_height)  
 self.RB\_ser = QRadioButton("Последовательно\n(Максимум 49 точек)", self)  
 self.RB\_ser.setGeometry(draw\_width + widg\_width, 4 \* widg\_height, widg\_width, widg\_height)  
 self.RB\_ser.setChecked(True)  
  
 self.T = QTableWidget(self)  
 self.T.setColumnCount(3)  
 for i in range(3):  
 self.T.setColumnWidth(i, 86)  
 self.T.setGeometry(draw\_width, 5 \* widg\_height, 2 \* widg\_width, 11 \* widg\_height)  
  
 # Очистка таблицы точек, только графика  
 def clear\_table(self):  
 self.T.clear()  
 self.T.setRowCount(0)  
 self.clear\_graph()  
  
 # Очистка области рисования, только графика  
 def clear\_graph(self):  
 canvas = QPixmap(1200, 800)  
 canvas.fill(Qt.GlobalColor.white)  
 self.label\_draw.setPixmap(canvas)  
 canvas = self.label\_draw.pixmap()  
 painter = QPainter(canvas)  
 pen = QPen()  
 pen.setColor(QColor(0, 0, 0))  
 pen.setWidth(3)  
 painter.setPen(pen)  
 point0 = QPoint(floor(self.draw\_width / 25), floor(24 \* self.draw\_height / 25))  
 point1 = QPoint(floor(self.draw\_width / 5), floor(24 \* self.draw\_height / 25))  
 point2 = QPoint(floor(self.draw\_width / 25), floor(4 \* self.draw\_height / 5))  
 point3 = QPoint(floor(self.draw\_width / 10), floor(9 \* self.draw\_height / 10))  
 painter.drawLine(point0, point1)  
 painter.drawText(point1, "X")  
 painter.drawLine(point0, point2)  
 painter.drawText(point2, "Y")  
 painter.drawLine(point0, point3)  
 painter.drawText(point3, "Z")  
 painter.end()  
 self.label\_draw.setPixmap(canvas)  
  
 # Рекурсивное получение интервала параметризации  
 def get\_part(self, power, region, parameters): # Передаём степень сплайна, номер интервала и параметры  
 # Базис рекурсии  
 if region < 0 or region > power:  
 return [0] \* (power + 1)  
 if power == 0:  
 return [1]  
 rez = [0] \* (power + 1) # Список коэффициентов при t от p до 0  
 # Получение коэффициентов параметризации меньшей степени  
 prev\_left = self.get\_part(power - 1, region, parameters[:len(parameters) - 1])  
 prev\_right = self.get\_part(power - 1, region - 1, parameters[1:])  
 # Коэффициенты при t и и не t в формуле Кокса де Бура  
 t\_left = 1 / (parameters[power] - parameters[0])  
 left = -1 \* parameters[0] / (parameters[power] - parameters[0])  
 right = parameters[power + 1] / (parameters[power + 1] - parameters[1])  
 t\_right = -1 / (parameters[power + 1] - parameters[1])  
 # Получаем список коэффициентов по формуле Кокса де Бура  
 for i in range(power):  
 # Так как умножаем на коэффициент при t прибавляем к большей степени  
 rez[i] += prev\_left[i] \* t\_left + prev\_right[i] \* t\_right  
 # Так как умножаем на коэффициент без t прибавляем к текущей степени  
 rez[i + 1] += prev\_left[i] \* left + prev\_right[i] \* right  
 return rez  
  
 # Отрисовка сплайна  
 def draw\_graph(self, points, status=0):  
 canvas = self.label\_draw.pixmap()  
 painter = QPainter(canvas)  
 # Прибавляем к координатам x и y координату z умноженную на cos(45) или sin(45) (sqrt(2) / 2)  
 for i in range(len(points)):  
 koef = round(points[i][2] / pow(2, (1 / 2)))  
 points[i][0] += koef  
 points[i][1] += koef  
 pen = QPen()  
 # Установка цвета и толщины линий, отрисовка точек  
 if status == 0:  
 pen.setColor(QColor(0, 255, 0))  
 pen.setWidth(10)  
 painter.setPen(pen)  
 painter.drawEllipse(points[0][0] - 5, (800 - points[0][1]) - 5, 10, 10)  
 painter.drawEllipse(points[len(points) - 1][0] - 5, (800 - points[len(points) - 1][1]) - 5, 10, 10)  
 pen.setColor(QColor(255, 0, 0))  
 pen.setWidth(6)  
 painter.setPen(pen)  
 for i in range(1, len(points) - 1):  
 painter.drawEllipse(points[i][0] - 3, (800 - points[i][1]) - 3, 6, 6)  
 pen.setWidth(3)  
 painter.setPen(pen)  
 else:  
 pen.setColor(QColor(0, 0, 255))  
 pen.setWidth(5)  
 painter.setPen(pen)  
 # Отрисовка графика  
 for i in range(len(points) - 1):  
 painter.drawLine(points[i][0], (800 - points[i][1]), points[i + 1][0], (800 - points[i + 1][1]))  
 painter.end()  
 self.label\_draw.setPixmap(canvas)  
  
 # Генерация опорных точек  
 def gen\_points(self):  
 if (self.SB\_np.value() < 50 or self.RB\_rand.isChecked()) and self.SB\_np.value() >= self.SB\_pow.value() + 1:  
 self.T.setRowCount(self.SB\_np.value())  
 for i in range(self.SB\_np.value()):  
 if self.RB\_ser.isChecked():  
 x\_lower = floor(self.draw\_width / (self.SB\_np.value() + 1) \* i)  
 x\_upper = floor(self.draw\_width / (self.SB\_np.value() + 1)) \* (i + 1)  
 self.T.setItem(i, 0, QTableWidgetItem(str(randrange(x\_lower, x\_upper))))  
 self.T.setItem(i, 1, QTableWidgetItem(str(randrange(0, self.draw\_height))))  
 self.T.setItem(i, 2, QTableWidgetItem(str(randrange(-100, 100))))  
 else:  
 self.T.setItem(i, 0, QTableWidgetItem(str(randrange(100, self.draw\_width - 100))))  
 self.T.setItem(i, 1, QTableWidgetItem(str(randrange(100, self.draw\_height - 100))))  
 self.T.setItem(i, 2, QTableWidgetItem(str(randrange(-100, 100))))  
  
 # Обработка нажатия на кнопку "Построения графика"  
 def drawing(self):  
 # Только если в таблице есть точки  
 if self.T.rowCount() != 0:  
 # Очистка поля графика  
 self.clear\_graph()  
 # Количество точек  
 num\_points = self.SB\_np.value()  
 # Степень  
 power = self.SB\_pow.value()  
 # Качество (количество точек для отрисовки одного элементарного сплайна)  
 quality = pow(10, self.SB\_qual.value())  
 # Число существенных интервалов (int - зарезервированное слово, его использовать нельзя)  
 imp = num\_points - power  
 # Интервал параметризации  
 parameters = [i for i in range(power + num\_points + 1)]  
 # Получаем опорные точки из таблицы  
 pts = []  
 for i in range(self.T.rowCount()):  
 pts.append(  
 [int(self.T.item(i, 0).text()), int(self.T.item(i, 1).text()), int(self.T.item(i, 2).text())])  
 # Получение точек для отрисовки сплайна в количестве imp \* quality, у точек три координаты: x, y, z  
 pts2 = [[0] \* 3 for \_ in range(imp \* quality)]  
 for t in range(power \* quality, num\_points \* quality):  
 # Получение коэффициентов для каждого интервала параметризации  
 vector = [[0] \* (power + 1) for \_ in range(power + 1)]  
 for i in range(power + 1):  
 left\_cap = i + floor(t / quality) - power  
 right\_cap = i + floor(t / quality) + 2  
 vector[i] = self.get\_part(power, (power - i), parameters[left\_cap: right\_cap])  
 # Находим текущую опорную точку  
 t\_cur = (t / quality)  
 # Получаем координаты текущей точки сплайна,

# для чего перемножаем вектор степеней t на матрицу коэффициентов t, на вектор координат опорных точек  
 for i in range(power + 1):  
 for j in range(power + 1):  
 degree = (power - j)  
 for k in range(3):  
 point\_cur = pts[i + floor(t\_cur) - power][k]  
 pts2[t - (power \* quality)][k] += vector[i][j] \* (t\_cur \*\* degree) \* point\_cur  
 # Округляем полученные координаты для отрисовки  
 for i in range(3):  
 pts2[t - (power \* quality)][i] = round(pts2[t - (power \* quality)][i])  
 # Отрисовка сплайна и ломанной по опорным точкам  
 window.draw\_graph(pts2, 1)  
 window.draw\_graph(pts)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 # Инициализация, только графика  
 window\_size\_x, window\_size\_y = 1500, 800  
 draw\_size\_x, draw\_size\_y = 1200, 800  
 app = QApplication([])  
 window = MainWindow(window\_size\_x, window\_size\_y, draw\_size\_x)  
 window.show()  
 app.exec()