**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра математического обеспечения и применения ЭВМ**

отчет

**По лабораторной работе № 4**

**по дисциплине «Компьютерная графика»**

Тема: **Кубические сплайны**

|  |  |  |
| --- | --- | --- |
| Студент гр. 0303 |  | Архипов В.А. |
| Преподаватель |  | Герасимова Т.В. |

Санкт-Петербург

2023

**Цель работы.**

Реализовать интерактивное приложение, отображающее заданную полиномиальную кривую.

**Задание.**

Реализовать сплайн Коханека-Бартелса (Вариант 39).

**Выполнение работы.**

Работа была выполнена в среде разработки PyCharm на языке программирования Python 3.10. Для реализации пользовательского интерфейса были использованы библиотека PyQt6 и программа Qt Designer, создающая пользовательский интерфейс по построенному в ней макету. Для работы с графикой была использована библиотека PyOpenGL. Подключение графической библиотеки к пользовательскому интерфейсу было осуществлено с помощью виджета QOpenGLWidget из библиотеки PyQt6.

Макет пользовательского интерфейса был разработан в программе Qt Designer (см. рис. 1).

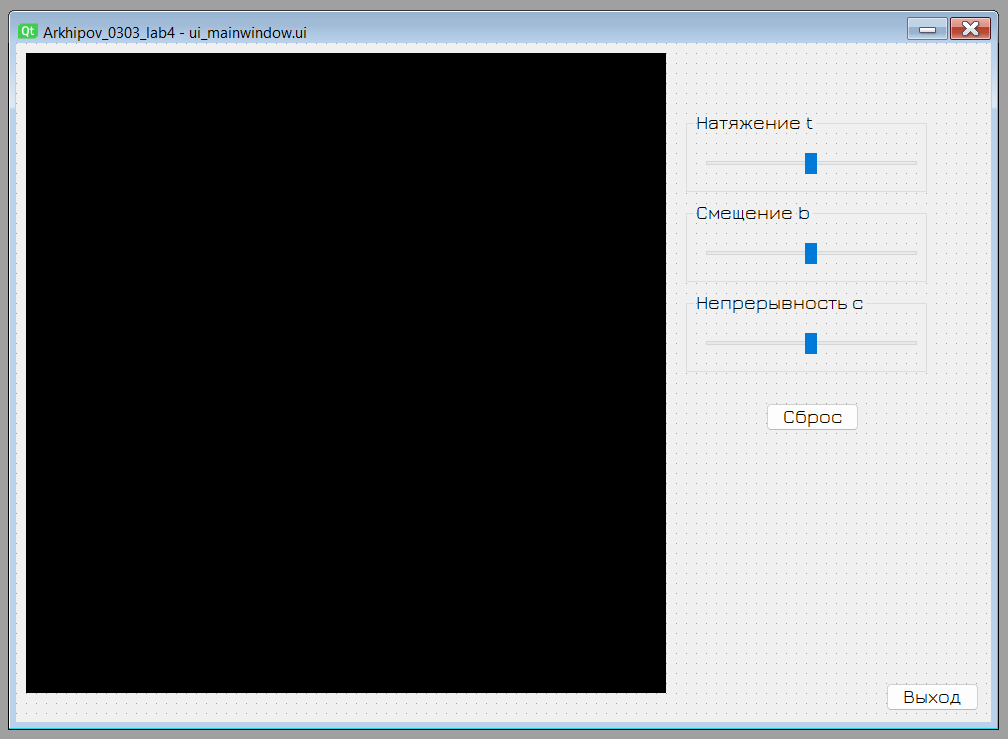


Рисунок 1 – разработанный в Qt Designer макет пользовательского интерфейса

Класс главного окна MainWindow наследуется от базового для PyQt класса QMainWindow и содержит внутри себя виджет glWidget унаследованный от QOpenGLWidget, для отображения графики, а также меню для настройки параметров изображения (в конкретном случае параметров три – натяжение, смещение и непрерывность; установить каждое из этих значений можно, передвинув соответствующий ползунок).

Теперь перейдем к рассмотрению математической модели сплайна, который необходимо реализовать.

Сплайн Коханека-Бартелса представляет собой кубический сплайн Эрмита с дополнительными параметрами натяжения, смещения и непрерывности, которые определяют поведение касательных. В общем случае для построения сплайна необходимы как минимум 4 точки – первая и последняя точки всегда будут опорными, а через точки, находящиеся между первой и последней будет проведена кривая. Для каждого сегмента будущей кривой определена начальная и конечная точки и соответственно. На основании этих точек определяется значение начальной и конечной касательных и по следующим формулам:

(1)

(2)

где – натяжение, – смещение, – непрерывность. Стоит отметить, что значения лежат в пределах .

Помимо сведений о сплайне Коханека-Бартелса необходимо ознакомиться с тем, как происходит построение кубических сплайнов Эрмита. Для вычисления значения кубического сплайна в любой точке , принадлежащей интервалу (, ), можно воспользоваться формулой

где , , ,

,

,

,

.

Для того, чтобы нарисовать кривую, пользователю необходимо установить точки, через которые должен проходить сплайн. Сделать это можно, щелкнув мышкой внутри виджета OpenGL по месту, куда пользователь хочет установить точку. Когда количество выбранных точек будет больше или равно 4, будет построена кривая.

Для вычисления искомой кривой был реализован модуль interpolation.py.

Внутри этого модуля реализована функция kochanek\_bartels\_spline, которая принимает на вход массив точек, установленных пользователем на экране, и значения параметров . Далее для каждой пары точек, начиная со второй и третьей и заканчивая пред-предпоследней и предпоследней определяются значения . Первая и последняя точки массива являются опорными и требуются только для вычисления значений касательных, поэтому в основном цикле функции не рассматриваются как точки, через которые будет проводиться кривая. Далее для каждой рассматриваемой на текущей итерации пары точек вычисляются значения начальной и конечной касательных по формулам (1) и (2). После этого интервал ( ) разбивается на равные отрезки 20 точками. Для каждой из таких точек вычисляется значение функции сплайна по формуле (3). Координаты получившейся точки записываются в итоговый массив. Когда все сегменты кривой, образованные парами точек, будут рассмотрены, функция завершит свою работу и вернет массив точек, которые нужно будет отобразить. Эти точки будут отображены на экране и с помощью примитива GL\_LINE\_STRIP будет построена итоговая кривая.

**Демонстрация.**

Ниже приведены примеры работы реализованного приложения.

На рис. 2 – 4 представлено влияние на кривую значения параметра натяжения . Как мы можем наблюдать, чем ближе это значение к -1, тем более выпуклой получается линия, и наоборот, чем ближе это значение к 1, тем кривая больше начинает напоминать прямую натянутую струну.

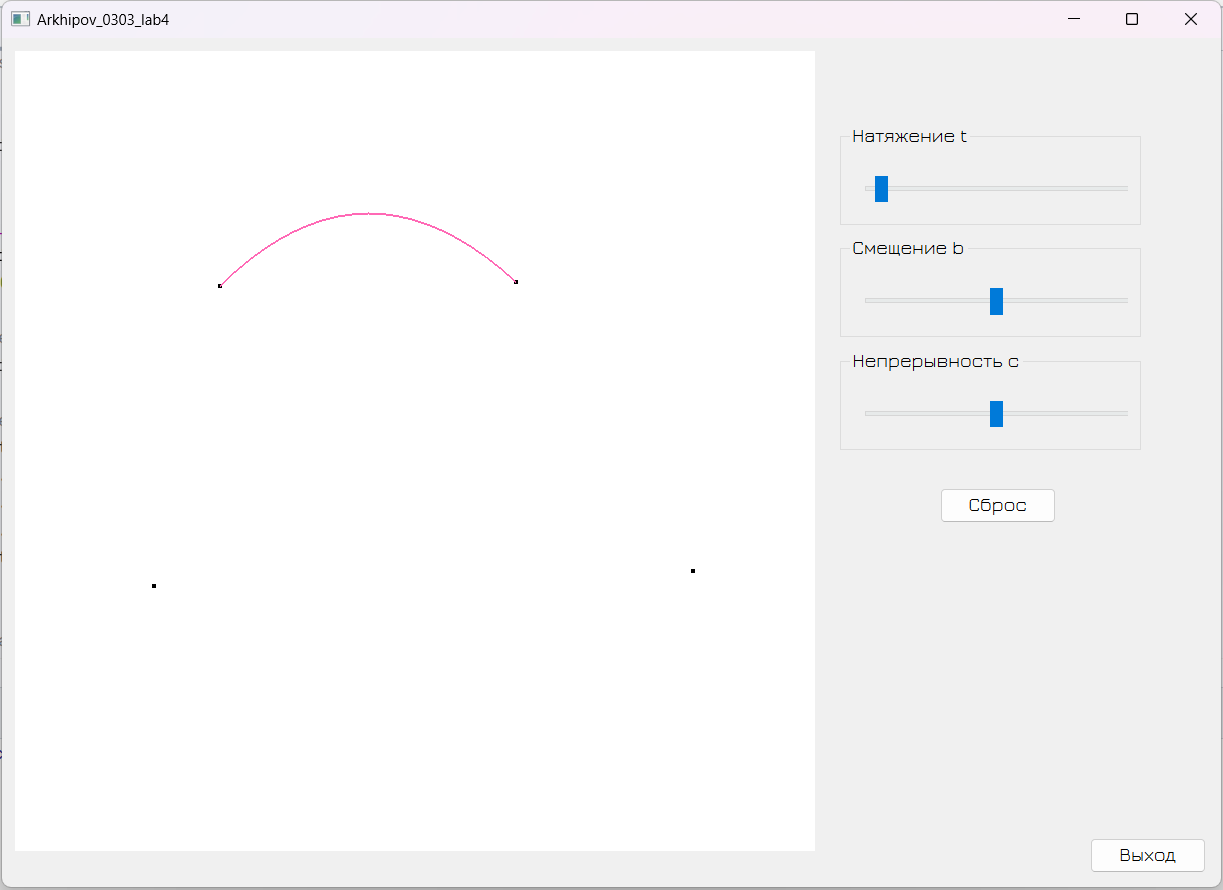


Рисунок 2 – параметр близок к -1

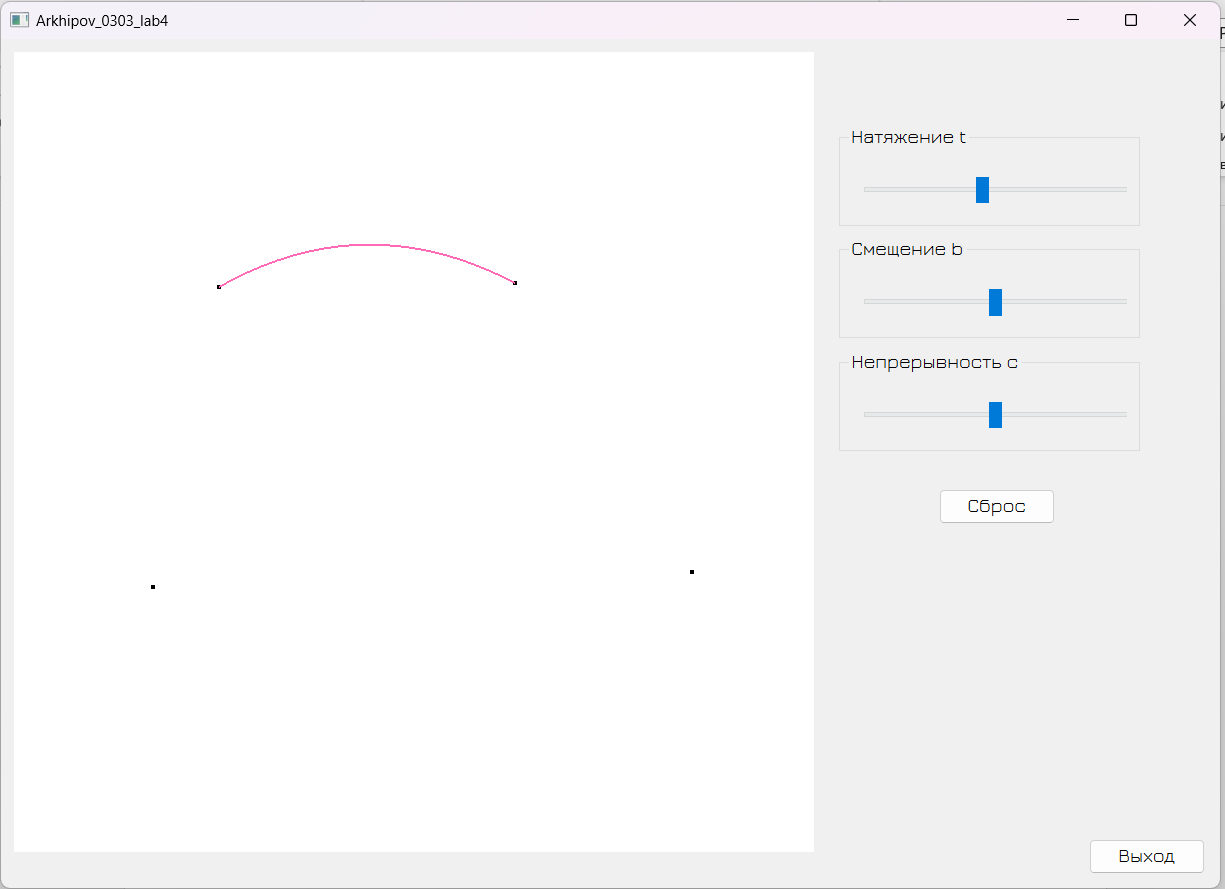


Рисунок 3 – параметр близок к 0

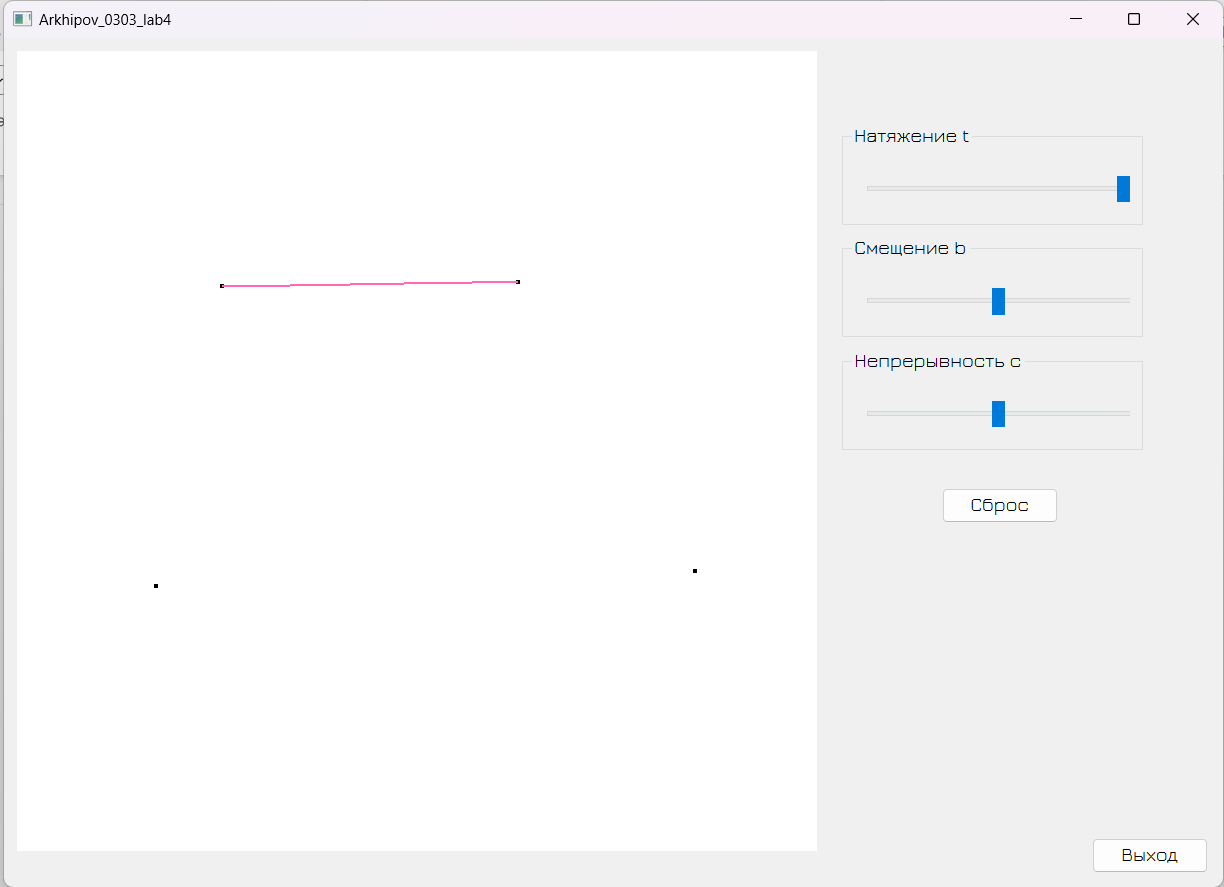


Рисунок 4 – параметр близок к 1

Теперь рассмотрим влияние на итоговый вид кривой параметра смещения (см. рис. 5 – 7). При варьировании этого значения мы можем заметить, как меняется положение точек экстремума кривой относительной выставленных пользователем точек. При плавном изменении этого параметра синусоида как будто движется вдоль горизонтальной оси.

Изображение выглядит как диаграмма

Автоматически созданное описание

Рисунок 5 – параметр близок к -1

Изображение выглядит как диаграмма

Автоматически созданное описание

Рисунок 6 – параметр близок к 0

Изображение выглядит как диаграмма

Автоматически созданное описание

Рисунок 7 – параметр близок к 1

При изменении параметра мы видим, что функция постепенно становится более извилистой (см. рис. 8 – 10). При мы видим «изломы» в местах стыка соседних фрагментов. Это говорит о том, что производная в данной точке стыка не определена и построенная функция не является непрерывной. Поэтому данный сплайн не следует применять в компьютерной графике.

Изображение выглядит как диаграмма

Автоматически созданное описание

Рисунок 8 – параметр близок к -1

Изображение выглядит как диаграмма

Автоматически созданное описание

Рисунок 9 – параметр близок к 0

Изображение выглядит как диаграмма

Автоматически созданное описание

Рисунок 0 – параметр близок к 1

Также не стоит применять данную кривую в графике, так как при некоторой конфигурации параметров данная кривая может вылезать за пределы области, которую определил пользователь. На рис. 11 были отмечены точки прямоугольника и видно, что функция вышла за пределы сформированной фигуры.

Изображение выглядит как диаграмма

Автоматически созданное описание

Рисунок 11 – кривая вышла за обозначенный границы

Помимо этого, в некоторых случаях эта кривая может образовывать петли, что также является нежелательным, когда мы говорим о кривых, пригодных к использованию в компьютерной графике (см. рис. 12).

Изображение выглядит как диаграмма

Автоматически созданное описание

Рисунок 1 – кривая может образовывать петли

На основании данной демонстрации можно сделать вывод о том, что сплайн Коханека-Бартелса нежелателен для использования в компьютерной графике, так как при некоторых конфигурациях может иметь изломы, способен давать петли и вылезать за границы сформированной фигуры.

Разработанный программный код см. в приложении А.

**Вывод.**

В результате выполнения лабораторной работы была разработана программа, реализующая построение заданной полиномиальной кривой. Также было установлено, что заданная кривая не всегда может быть применена для нужд компьютерной графики.

**ПРИЛОЖЕНИЕ А**

**РАЗРАБОТАННЫЙ ПРОГРАММНЫЙ КОД**

Название файла: glWidget.py

from PyQt6.QtOpenGLWidgets import QOpenGLWidget  
from OpenGL import GL as gl  
from PyQt6 import QtCore  
from interpolation import kochanek\_bartels\_spline  
  
class glWidget(QOpenGLWidget):  
 def \_\_init\_\_(self, main\_window):  
 super().\_\_init\_\_(parent=main\_window.ui.centralwidget)  
 *# Список, отмеченных пользователем точек* self.points = []  
  
 main\_window.ui.openGLWidget = self  
 main\_window.ui.openGLWidget.setGeometry(QtCore.QRect(10, 10, 640, 640))  
 main\_window.ui.openGLWidget.setObjectName("openGLWidget")  
 *# Ссылка на родительское окно* self.mw = main\_window  
  
 *# Настройка состояния. Вызывается перед каждым обновлением кадра* def initializeGL(self):  
 gl.glClearColor(1,1,1,1)  
 gl.glMatrixMode(gl.GL\_PROJECTION)  
 gl.glClear(gl.GL\_COLOR\_BUFFER\_BIT | gl.GL\_DEPTH\_BUFFER\_BIT)  
  
 def paintGL(self):  
 gl.glColor3f(1, 0, 0)  
 gl.glPointSize(4)  
 gl.glBegin(gl.GL\_POINTS)  
 if not self.points:  
 gl.glColor3f(1,1,1)  
 gl.glVertex2f(0,0)  
 gl.glColor3f(0,0,0)  
 for point in self.points:  
 gl.glVertex2fv(point)  
 gl.glEnd()  
  
 if len(self.points) >= 4:  
 gl.glColor3f(1.00,0.41,0.71)  
 gl.glPointSize(2)  
 gl.glLineWidth(2)  
 gl.glBegin(gl.GL\_LINE\_STRIP)  
  
 for point in kochanek\_bartels\_spline(self.points, self.mw.ui.tSlider.value()/100,  
 self.mw.ui.bSlider.value()/100, self.mw.ui.cSlider.value()/100):  
 gl.glVertex2fv(point)  
 gl.glEnd()  
  
 *# Клик мышки создаёт в этом месте точку* def mousePressEvent(self, event):  
 origin = event.pos()  
 x = origin.x()  
 y = origin.y()  
  
 if x < 320:  
 x = -(320 - x)/320  
 else:  
 x = (x - 320)/320  
  
 if y < 320:  
 y = (320 - y)/320  
 else:  
 y = -(y - 320) / 320  
  
 point = [x, y]  
 self.points.append(point)  
 self.update()

Название файла: MainWindow.py

import sys  
from PyQt6.QtWidgets import QMainWindow  
from glWidget import glWidget  
from ui\_mainwindow import Ui\_MainWindow  
  
  
class MainWindow(QMainWindow):  
  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.ui = Ui\_MainWindow()  
 self.ui.setupUi(self)  
  
 *# Создание виджета для отображения графики* self.ui.openGLWidget = glWidget(self)  
  
 *# Подписка элементов интерфейса на события* self.ui.pushButton.clicked.connect(self.leave)  
 self.ui.tSlider.valueChanged.connect(self.param\_changed)  
 self.ui.bSlider.valueChanged.connect(self.param\_changed)  
 self.ui.cSlider.valueChanged.connect(self.param\_changed)  
 self.ui.pushButton\_2.clicked.connect(self.restart)  
  
 *# Демонстрация окна* self.show()  
  
  
 def leave(self):  
 print('Выполнен выход из приложения')  
 sys.exit()  
  
 def param\_changed(self):  
 self.ui.openGLWidget.update()  
  
 def restart(self):  
 print('Выполним сброс точек')  
 self.ui.openGLWidget.points = []  
 self.ui.tSlider.setValue(0)  
 self.ui.cSlider.setValue(0)  
 self.ui.bSlider.setValue(0)  
 self.ui.openGLWidget.update()

Название файла: interpolation.py

h00 = lambda t: 2 \* t \*\* 3 - 3 \* t \*\* 2 + 1  
h10 = lambda t: t \*\* 3 - 2 \* t \*\* 2 + t  
h01 = lambda t: - 2 \* t \*\* 3 + 3 \* t \*\* 2  
h11 = lambda t: t \*\* 3 - t \*\* 2  
  
  
def kochanek\_bartels\_spline(points, t, b, c):  
 new\_points = []  
 for i in range(1, len(points) - 2):  
 x\_i, p\_i = points[i]  
 p\_i\_prev = points[i-1][1]  
 x\_i1, p\_i1 = points[i+1]  
 p\_i1\_next = points[i + 2][1]  
  
 *# Считаем касательные* d\_i = (1-t)\*(1+b)\*(1+c)/2 \* (p\_i - p\_i\_prev) + (1-t)\*(1-b)\*(1-c)/2 \* (p\_i1 - p\_i)  
 d\_i1 = (1-t)\*(1+b)\*(1-c)/2 \* (p\_i1 - p\_i) + (1-t)\*(1-b)\*(1+c)/2 \* (p\_i1\_next - p\_i1)  
  
 delta = (x\_i1 - x\_i)/20  
 for i in range(21):  
 current\_point\_x = x\_i + i\*delta  
 t\_p = (i\*delta) / (x\_i1 - x\_i)  
 *#current\_point\_y = h00(t\_p)\*p\_i + h10(t\_p)\*(x\_i1 - x\_i)\*d\_i + h01(t\_p)\*p\_i1 + h11(t\_p)\*(x\_i1 - x\_i)\*d\_i1* current\_point\_y = h00(t\_p)\*p\_i + h10(t\_p)\*(x\_i1 - x\_i)\*d\_i/ abs(x\_i - x\_i1) + h01(t\_p)\*p\_i1 + h11(t\_p)\*(x\_i1 - x\_i)\*d\_i1/ abs(x\_i1 - x\_i)  
 new\_points.append([current\_point\_x, current\_point\_y])  
  
 return new\_points

Название файла: main.py

import sys  
from PyQt6.QtWidgets import QApplication  
from MainWindow import MainWindow  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 app = QApplication(sys.argv)  
 main\_window = MainWindow()  
 main\_window.setFixedSize(975, 679)  
 sys.exit(app.exec())

Название файла: ui\_mainwindow.py

*# Form implementation generated from reading ui file 'ui\_mainwindow.ui'  
#  
# Created by: PyQt6 UI code generator 6.4.2  
#  
# WARNING: Any manual changes made to this file will be lost when pyuic6 is  
# run again. Do not edit this file unless you know what you are doing.*from PyQt6 import QtCore, QtGui, QtWidgets  
  
  
class Ui\_MainWindow(object):  
 def \_\_init\_\_(self):  
 self.openGLWidget = None  
  
 def setupUi(self, MainWindow):  
 MainWindow.setObjectName("MainWindow")  
 MainWindow.resize(975, 679)  
 self.centralwidget = QtWidgets.QWidget(parent=MainWindow)  
 self.centralwidget.setObjectName("centralwidget")  
 self.pushButton = QtWidgets.QPushButton(parent=self.centralwidget)  
 self.pushButton.setGeometry(QtCore.QRect(870, 640, 93, 28))  
 font = QtGui.QFont()  
 font.setFamily("Jura")  
 font.setPointSize(12)  
 self.pushButton.setFont(font)  
 self.pushButton.setObjectName("pushButton")  
 self.pushButton\_2 = QtWidgets.QPushButton(parent=self.centralwidget)  
 self.pushButton\_2.setGeometry(QtCore.QRect(750, 360, 93, 28))  
 font = QtGui.QFont()  
 font.setFamily("Jura")  
 font.setPointSize(12)  
 self.pushButton\_2.setFont(font)  
 self.pushButton\_2.setObjectName("pushButton\_2")  
 self.groupBox = QtWidgets.QGroupBox(parent=self.centralwidget)  
 self.groupBox.setGeometry(QtCore.QRect(670, 70, 241, 80))  
 font = QtGui.QFont()  
 font.setFamily("Jura")  
 font.setPointSize(12)  
 self.groupBox.setFont(font)  
 self.groupBox.setObjectName("groupBox")  
 self.tSlider = QtWidgets.QSlider(parent=self.groupBox)  
 self.tSlider.setGeometry(QtCore.QRect(20, 40, 211, 22))  
 self.tSlider.setMinimum(-100)  
 self.tSlider.setMaximum(100)  
 self.tSlider.setOrientation(QtCore.Qt.Orientation.Horizontal)  
 self.tSlider.setInvertedAppearance(False)  
 self.tSlider.setInvertedControls(False)  
 self.tSlider.setTickPosition(QtWidgets.QSlider.TickPosition.NoTicks)  
 self.tSlider.setObjectName("tSlider")  
 self.groupBox\_3 = QtWidgets.QGroupBox(parent=self.centralwidget)  
 self.groupBox\_3.setGeometry(QtCore.QRect(670, 250, 241, 80))  
 font = QtGui.QFont()  
 font.setFamily("Jura")  
 font.setPointSize(12)  
 self.groupBox\_3.setFont(font)  
 self.groupBox\_3.setObjectName("groupBox\_3")  
 self.cSlider = QtWidgets.QSlider(parent=self.groupBox\_3)  
 self.cSlider.setGeometry(QtCore.QRect(20, 40, 211, 22))  
 self.cSlider.setMinimum(-100)  
 self.cSlider.setMaximum(100)  
 self.cSlider.setOrientation(QtCore.Qt.Orientation.Horizontal)  
 self.cSlider.setInvertedAppearance(False)  
 self.cSlider.setInvertedControls(False)  
 self.cSlider.setTickPosition(QtWidgets.QSlider.TickPosition.NoTicks)  
 self.cSlider.setObjectName("cSlider")  
 self.groupBox\_2 = QtWidgets.QGroupBox(parent=self.centralwidget)  
 self.groupBox\_2.setGeometry(QtCore.QRect(670, 160, 241, 80))  
 font = QtGui.QFont()  
 font.setFamily("Jura")  
 font.setPointSize(12)  
 self.groupBox\_2.setFont(font)  
 self.groupBox\_2.setObjectName("groupBox\_2")  
 self.bSlider = QtWidgets.QSlider(parent=self.groupBox\_2)  
 self.bSlider.setGeometry(QtCore.QRect(20, 40, 211, 22))  
 self.bSlider.setMinimum(-100)  
 self.bSlider.setMaximum(100)  
 self.bSlider.setOrientation(QtCore.Qt.Orientation.Horizontal)  
 self.bSlider.setInvertedAppearance(False)  
 self.bSlider.setInvertedControls(False)  
 self.bSlider.setTickPosition(QtWidgets.QSlider.TickPosition.NoTicks)  
 self.bSlider.setObjectName("bSlider")  
 MainWindow.setCentralWidget(self.centralwidget)  
  
 self.retranslateUi(MainWindow)  
 QtCore.QMetaObject.connectSlotsByName(MainWindow)  
  
 def retranslateUi(self, MainWindow):  
 \_translate = QtCore.QCoreApplication.translate  
 MainWindow.setWindowTitle(\_translate("MainWindow", "Arkhipov\_0303\_lab4"))  
 self.pushButton.setText(\_translate("MainWindow", "Выход"))  
 self.pushButton\_2.setText(\_translate("MainWindow", "Сброс"))  
 self.groupBox.setTitle(\_translate("MainWindow", "Натяжение t"))  
 self.groupBox\_3.setTitle(\_translate("MainWindow", "Непрерывность c"))  
 self.groupBox\_2.setTitle(\_translate("MainWindow", "Смещение b"))  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 import sys  
 app = QtWidgets.QApplication(sys.argv)  
 MainWindow = QtWidgets.QMainWindow()  
 ui = Ui\_MainWindow()  
 ui.setupUi(MainWindow)  
 MainWindow.show()  
 sys.exit(app.exec())