**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №4**

**по дисциплине «Компьютерная графика»**

Тема: «Кубические сплайн»

|  |  |  |
| --- | --- | --- |
| Студентка гр. 6382 |  | Черкасова Е.И. |
| Студент гр. 6303 |  | Доброхвалов М.О. |
| Преподаватель |  | Герасимова Т.В. |

Санкт-Петербург

2019

**Задание.**

Интерполяционный многочлен Лагранжа по 5 точкам.

**Общие сведения.**

**Сплайны** - это гладкие (имеющие несколько непрерывных производных) кусочно-полиномиальные функции, которые могут быть использованы для представления функций, заданных большим количеством значений и для которых неприменима аппроксимация одним полиномом. Так как сплайны гладки, экономичны и легки в работе, они используются при построении произвольных функций для:

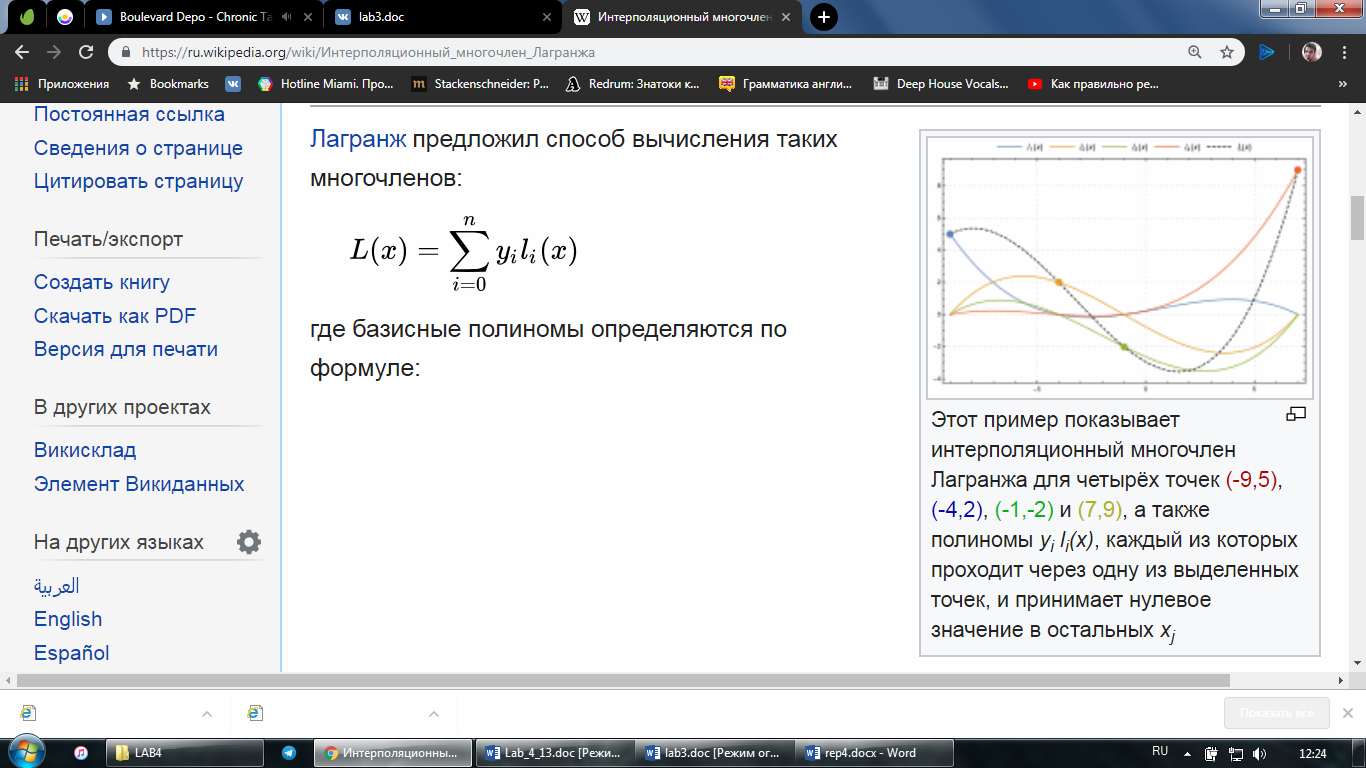
* моделирования кривых;
* аппроксимации данных с помощью кривых;
* выполнения функциональных аппроксимаций;
* решения функциональных уравнений.

Здесь кратко излагаются некоторые основные положения и использования сплайнов в 3D графики.

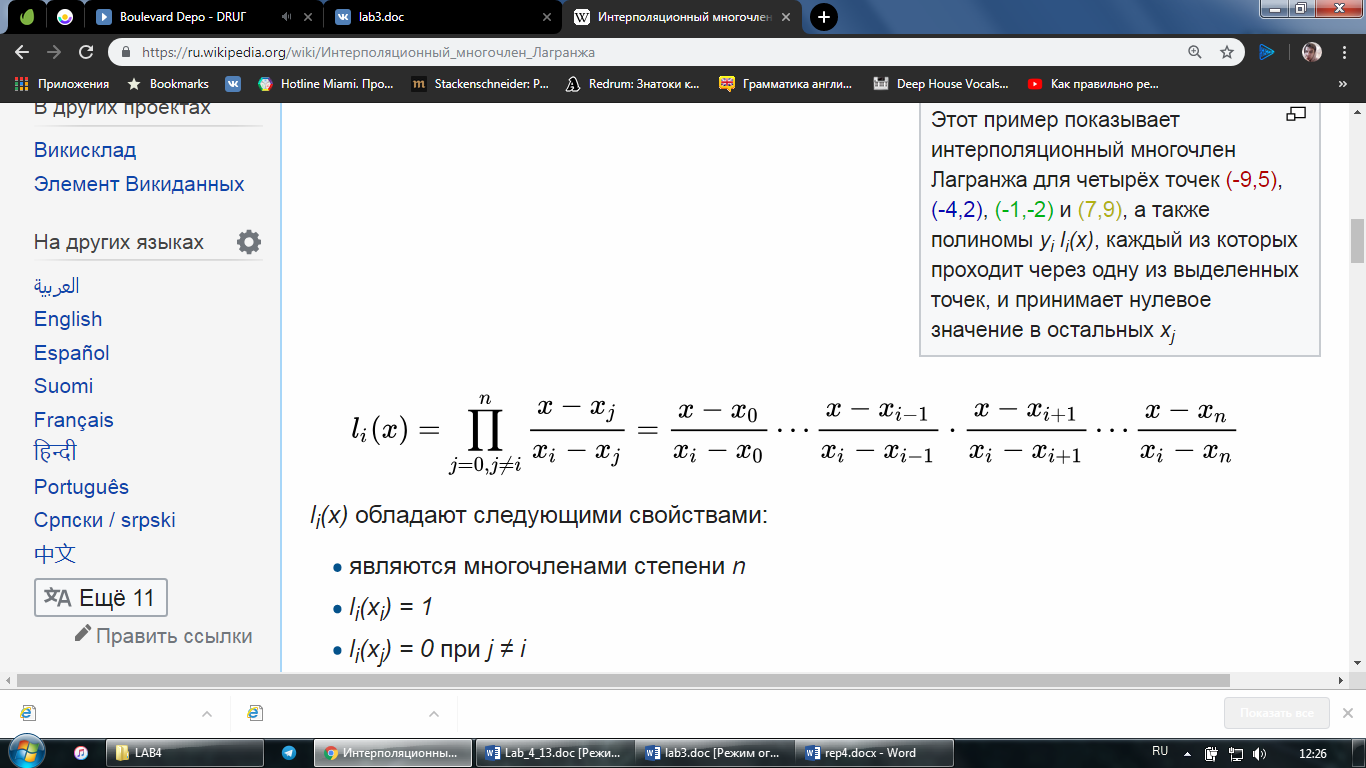
Важным их свойством является простота вычислений. На практике часто используют сплайны вида полиномов третьей степени. С их помощью довольно удобно проводить кривые, которые интуитивно соответствуют человеческому субъективному понятию гладкости.

**Интерполяцио́нный многочле́н Лагра́нжа** — многочлен минимальной степени, принимающий данные значения в данном наборе точек. Для *n+1* пар чисел (*x0*, *y0*), (*x1*, *y1*),…, (*xn*, *yn*), где все *xj* различны, существует единственный многочлен *L(x)* степени не более *n*, для которого *L(xj) = yj*.

В простейшем случае (*n=1*)— это линейный многочлен, график которого — прямая, проходящая через две заданные точки.

,

где li



**Ход работы.**

Функция lagrange используется из библиотеки scipy.interpolate, она позволяет найти коэффициенты интерполяционного многочлена Лагранжа.

 def get\_polynomial(self):

        points\_x = [point[0] for point in self.int\_points]

        points\_y = [point[1] for point in self.int\_points]

        return lagrange(points\_x, points\_y)

Далее выполняется функция расчета точек полученной функции, и ее отрисовка по массиву рассчитанных точек. Запускается таймер обновления экрана и обработчик событий мыши. Когда пользователь кликает на точку, проверяется, «точность» нажатия и точка начинает движение. Когда пользователь отпускает кнопку мыши точка устанавливается на данную позицию и расчет происходит заново.

Функции обработчики мыши:

  def mousePressEvent(self, event):

        on\_pressed = event.pos()

        for index, point in enumerate(self.int\_points):

            if not self.in\_move and abs(point[0]-on\_pressed.x()) < 5 and \

                    abs(point[1]-on\_pressed.y()) < 5:

                self.focus\_point = index

                self.in\_move = True

    def mouseMoveEvent(self, event):

        if self.in\_move:

            new\_position = event.pos()

            self.int\_points[self.focus\_point] = [new\_position.x(), new\_position.y()]

            self.polinomial = self.get\_polynomial()

            self.update()

    def mouseReleaseEvent(self, \*args, \*\*kwargs):

        self.in\_move = False

        self.focus\_point = -1

Рисование точек происходит на интервале [100; 500].

Модуль **controller.py,** в котором находится основная логика программы, имеет следующий вид:

from OpenGL.GL import \*

from PyQt5.QtOpenGL import QGLWidget

from scipy.interpolate import lagrange

class GLWidget(QGLWidget):

    def \_\_init\_\_(self, parent):

        super(GLWidget, self).\_\_init\_\_(parent)

        self.in\_move = False

        self.focus\_point = -1

        self.int\_points = [[100., 100.], [200., 200.], [300., 300.], [400., 400.], [500., 500.]]

        self.polinomial = self.get\_polynomial()

    def initializeGL(self):

        """It is called once before the first call to paintGL() or resizeGL(),

        and then once whenever the widget has been assigned a new QGLContext """

        glMatrixMode(GL\_PROJECTION)

        glLoadIdentity()

        glOrtho(0, self.width(), self.height(), 0, -1, 1)

        glMatrixMode(GL\_MODELVIEW)

    def paintGL(self):

        glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT | GL\_STENCIL\_BUFFER\_BIT)

        glLoadIdentity()

        glLineWidth(2)

        interp\_x = [value for value in range(1, self.width())]

        glColor3f(1,0,0)

        glBegin(GL\_LINE\_STRIP)

        for i in range(len(interp\_x)):

            glVertex2f(interp\_x[i], self.polinomial(interp\_x[i]))

        glEnd()

        glPointSize(10)

        glEnable(GL\_POINT\_SMOOTH)

        glBegin(GL\_POINTS)

        glColor3f(1,1,1)

        for point in self.int\_points:

            glVertex2f(\*point)

        glEnd()

    def resizeGL(self, w, h):

        glViewport(0, 0, w, h)

        glOrtho(0, w, 0, h, -1.0, 1.0)

        glLoadIdentity()

    def mousePressEvent(self, event):

        on\_pressed = event.pos()

        for index, point in enumerate(self.int\_points):

            if not self.in\_move and abs(point[0]-on\_pressed.x()) < 5 and \

                    abs(point[1]-on\_pressed.y()) < 5:

                self.focus\_point = index

                self.in\_move = True

    def mouseMoveEvent(self, event):

        if self.in\_move:

            new\_position = event.pos()

            self.int\_points[self.focus\_point] = [new\_position.x(), new\_position.y()]

            self.polinomial = self.get\_polynomial()

            self.update()

    def mouseReleaseEvent(self, \*args, \*\*kwargs):

        self.in\_move = False

        self.focus\_point = -1

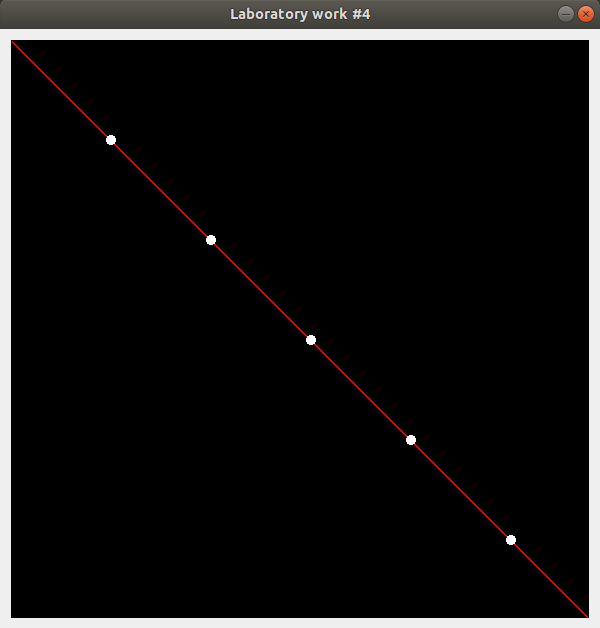
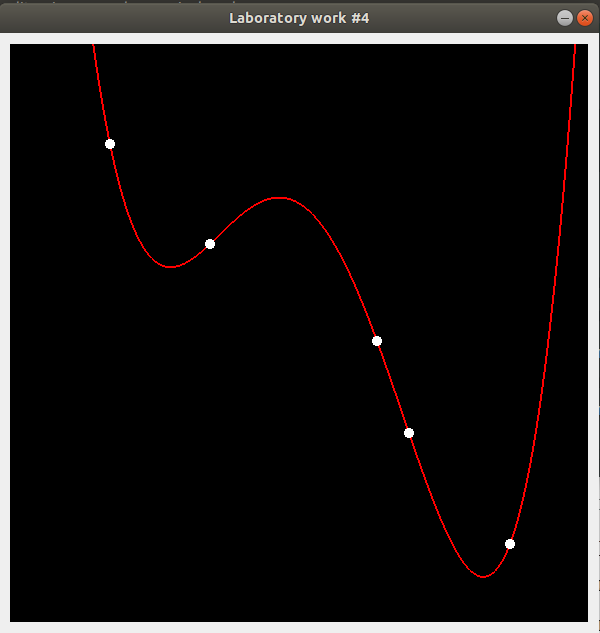
    def get\_polynomial(self):

        points\_x = [point[0] for point in self.int\_points]

        points\_y = [point[1] for point in self.int\_points]

        return lagrange(points\_x, points\_y)

**Тестирование.**

После перемещения точки пользователем:

**Вывод.**

В процессе выполнения лабораторной работы была разработана программа, строящая интерполяционный многочлен Лагранжа. Написана на языке программирования Python, протестирована в операционной системе Ubuntu. При выполнении работы были приобретены навыки работы с графической библиотекой OpenGL.