**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Компьютерная графика»

**Лабораторная работа № 7**

Тема: Построение плоских полиномиальных кривых.

Студент: Чекушкин Д.И.

Группа: 80-304

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2018

1. Постановка задачи

Задание: Написать программу, строящую полиномиальную кривую по заданным точкам.

12. B-сплайн. n = 6, k = 4. Узловой вектор равномерный.

1. Решения задачи

ЯП: Python

ОС: Ubuntu 16.04

Библиотеки: numpy, matplotlib.pyplot

matplotlib.pyplot.plot(\*args, \*\*kwargs) - Создает график

matplotlib.pyplot.show() - Отображает окно с графиком

numpy.arange(arg1,arg2) - Cоздания последовательностей чисел

Ход работы:

* Создаем массив промежутков для кривой b-spline
* По рекурсивной формуле Кокса - де Бура обходим все промежутки и определем значения сплайна
* Выводим результат

При тестировании были выявлены и исправлены незначительные ошибки, связанные с правильностью рекурсивной формулы

1. Руководство по использованию программы

Файл 7.py содержит код программы .

На вход не подается ничего, на выход – график искомой кривой и контрольные точки.

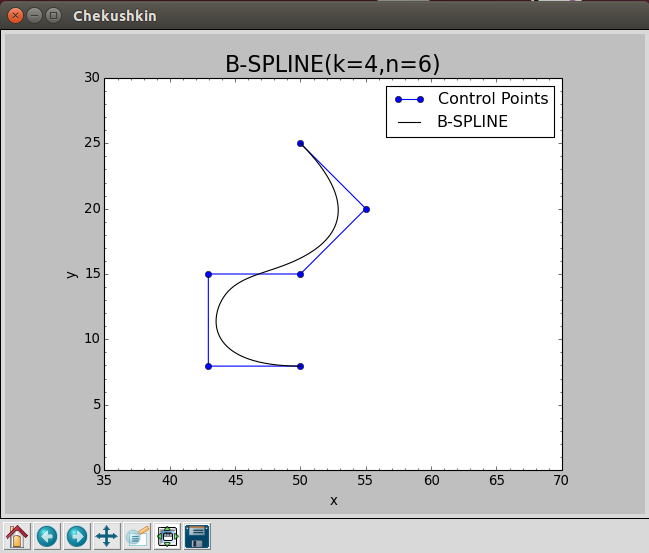


Рис 1 - Вывод программы

1. Листинг программы

#coding=utf-8

import numpy as np

import matplotlib.pyplot as plt

#Чекушкин М8О-304Б

#B-сплайн. n = 6, k = 4. Узловой вектор равномерный.

# cv = массив контрольных точек

# n = количество точек кривой

# d = степень кривой d=k-1, где k - порядок

# closed = кривая замкнута или открыта (TRUE/FALSE)

def bspline(cv, n=100, d=3, closed=False):

# Массив промежутков u

count = len(cv)

knots = None

u = None

if not closed:

u = np.arange(0,n,dtype='float')/(n-1) \* (count-d)

print(len(u))

print(u)

knots = np.array([0]\*d + range(count-d+1) + [count-d]\*d,dtype='int')

print(knots)

else:

u = ((np.arange(0,n,dtype='float')/(n-1) \* count) - (0.5 \* (d-1))) % count

print(len(u))

print(u)

knots = np.arange(0-d,count+d+d-1,dtype='int')

print(knots)

# Рекурсивный аогоритм Кокса - де Бура

def coxDeBoor(u, k, d):

# Входит ли точка в данный промежуток?

if (d == 0):

if (knots[k] <= u and u < knots[k+1]):

return 1

return 0

Den1 = knots[k+d] - knots[k]

Den2 = knots[k+d+1] - knots[k+1]

Eq1 = 0;

Eq2 = 0;

if Den1 > 0:

Eq1 = ((u-knots[k]) / Den1) \* coxDeBoor(u,k,(d-1))

if Den2 > 0:

Eq2 = ((knots[k+d+1]-u) / Den2) \* coxDeBoor(u,(k+1),(d-1))

return Eq1 + Eq2

# Значение кривой для каждого промежутка u

samples = np.zeros((n,3))

for i in xrange(n):

if not closed:

if u[i] == count-d:

samples[i] = np.array(cv[-1])

else:

for k in xrange(count):

samples[i] += coxDeBoor(u[i],k,d) \* cv[k]

else:

for k in xrange(count+d):

samples[i] += coxDeBoor(u[i],k,d) \* cv[k%count]

return samples

#Массив контрольных точек

cv = np.array([[ 50., 25., -0.],

[ 55., 20., -0.],

[ 50., 15., 0.],

[ 42.95, 15., 0.],

[ 42.95, 7.95, 0.],

[ 50., 7.95, -0.]])

p = bspline(cv)

x,y,z = p.T

cv = cv.T

plt.plot(cv[0],cv[1], 'o-', label='Control Points')

plt.plot(x,y,'k-',label='B-SPLINE')

plt.minorticks\_on()

Title = "B-SPLINE(k=4,n=6)"

plt.title(Title,fontsize=20)

plt.gcf().canvas.set\_window\_title("Chekushkin")

plt.legend()

plt.xlabel('x')

plt.ylabel('y')

plt.xlim(35, 70)

plt.ylim(0, 30)

plt.gca().set\_aspect('equal', adjustable='box')

plt.show()

Выводы: благодаря проделанной работе мне удалось освоить создание плоских полиномиальных кривых в Python

Список литературы:

1. Статья об аппроксимации на основе B-сплайнов [Электронный ресурс] Url: http://www.graphicon.ru/oldgr/grafor/gr\_help/chapter\_5\_9.htm
2. B-сплайн - Wikipedia [Электронный ресурс] Url: https://ru.wikipedia.org/wiki/B-%D1%81%D0%BF%D0%BB%D0%B0%D0%B9%D0%BD
3. Cox - De Boor's algorithm [Электронный ресурс] Url: https://en.wikipedia.org/wiki/De\_Boor%27s\_algorithm