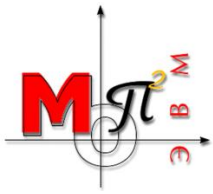
МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное образовательное
учреждение высшего образования
«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт компьютерных технологий и информационной безопасности
Кафедра Математического обеспечения и применения ЭВМ



ОТЧЁТ

по лабораторной работе №2
по курсу «МКиОДП»

Выполнили:
студенты группы КТмо2-3
Куприянова А.А.
Шепель И.О.

Проверил:

ассистент кафедры МОП ЭВМ
Жиглатый А.А.

Оценка

_____

«____» _____ 2017 г.

Таганрог 2017

Задание и цель работы

Цель: ознакомление с новым алгоритмом построения сплайновой кривой, с использованием параметрических дельта-сплайнов.

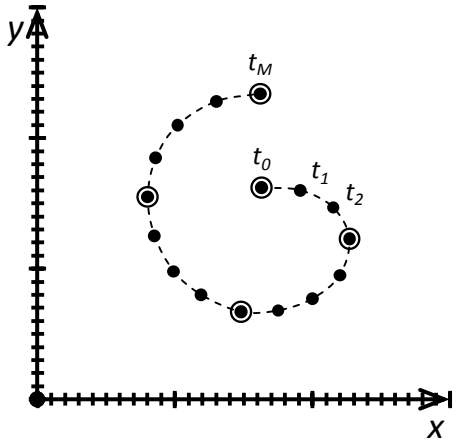
Задание: разработать на основе дельта-преобразований второго порядка программу, реализующую построение и графическое представление двумерной кривой с использованием параметрических дельта-сплайнов.

Вариант №1: параметрические дельта-сплайны с различным изменением шага дискретизации (в функции от t) между узлами интерполяции.

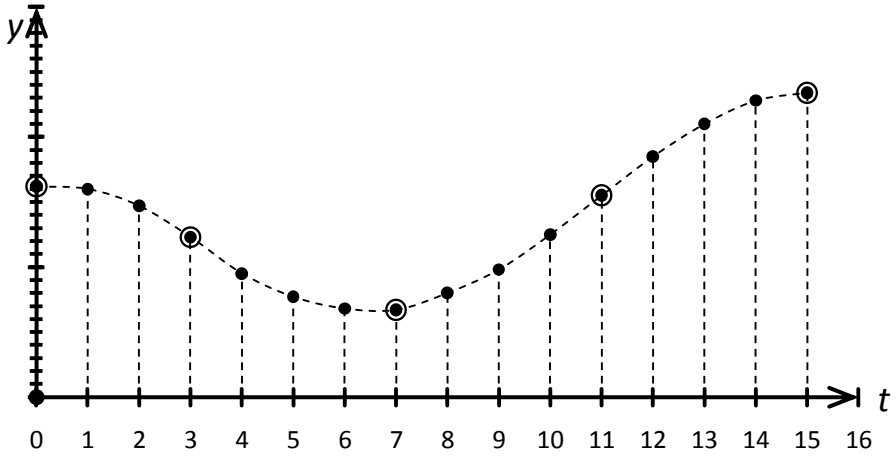
Математические основы построения сплайнов на основе дельта-преобразований второго порядка

В случаях, когда абсциссы и ординаты узлов не образуют строго возрастающие последовательности, построение сплайновой кривой базируется на параметрическом представлении кривой.

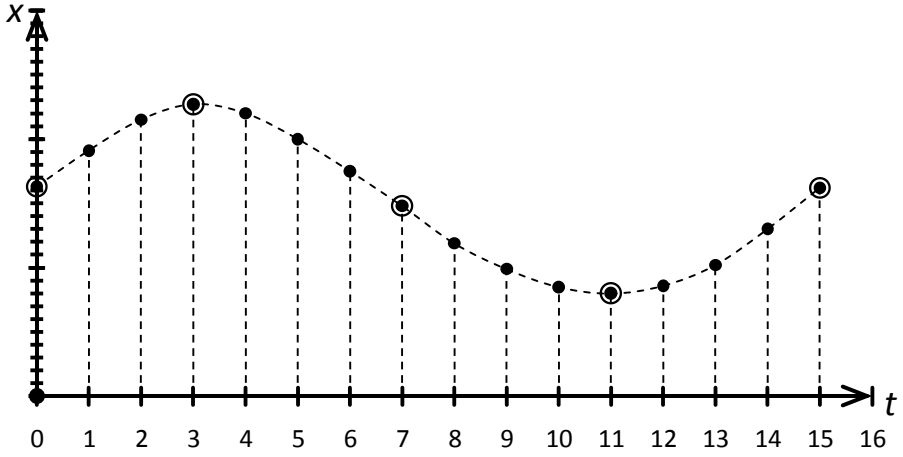
Пусть задан массив узлов $F(x_n, y_n)$ (рис. 3.1, узлы обозначены жирными точками). Формируем для каждой точки x_n и y_n, $t_n = 0,1,..M$ массивы в порядке возрастания n (независимая переменная представлена $t_n \equiv n$). Соответствующие расположение узлов по y_n (рис.3.1,б) и по x_n (рис.3.1,в) принимают вид при возрастающих значениях независимой переменной.



а)



б)



в)

Рис. 1. Параметрическое представление двумерной кривой

На основе дельта-преобразований второго порядка строятся две интерполяционные кривые:

- $y(t)$ с использованием узлов $y(t_n), t_n = 0,1..n;$

- $x(t)$ с использованием узлов $x(t_n), t_n = 0,1..n.$

Затем на основе точек этих двух кривых строится кривая y(x).

Алгоритм построения сплайновой кривой на основе дельта-преобразований второго порядка

В ходе выполнения лабораторной работы был разработан следующий алгоритм построения сплайновой кривой:

1. Задаются (считываются из файла) следующие данные:

t_n, x_n, y_n, $n = \overline{(0, N-1)}$ – базовые отсчёты, t_n – строго возрастающая последовательность.

С консоли вводится значение первой производной в граничных точках.

Количество точек на участок сплайна задаётся пропорционально длине участка сплайна, которая рассчитывается из координат начальной и конечной точки участка по теореме Пифагора.

2. Вычисляется массив t.
3. Производные в граничных точках вводятся в консоль.
4. Строятся интерполяционные кривые y(t) и x(t): массивы Y и X соответственно.
5. Производится запись в файл данных массивов t, X, Y.
6. Строятся три графика:
   - по данным массивов t и X строится график интерполяционной кривой x(t), также на графике отображаются базовые отсчёты и точки переключения кванта преобразования;
   - по данным массивов t и Y строится график интерполяционной кривой y(t), также на графике отображаются базовые отсчёты и точки переключения кванта преобразования;
   - по данным массивов X и Y строится график результирующей интерполяционной кривой y(x), точками на графике отображаются базовые отсчёты.

Выводы о проделанной работе

В ходе выполнения лабораторной работы был разработан и реализован на языке Python алгоритм построения и графического представления двумерной кривой с использованием параметрических дельта-сплайнов.

Результаты работы программы

Для следующих входных данных

t_n = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110],

x_n = [40, 45, 55, 54, 50, 45, 30, 20, 10, 0, -10, -5],

y_n = [60, 30, 5, 0, -20, -10, -30, -10, 10, 10, 30, 40],

$y`(x)_0$ = -5, $y`(x)_{N-1}$=1

программа выдаёт следующий результат в выходной файл:

t	x(t)	y(t)
0.0	40.0	60.0
1.6666666666666667	41.45269869235526	52.30561646704831
3.3333333333333335	42.47746143608773	45.88913253485991
5.0	43.07428823119738	40.7505482034348
6.666666666666667	43.355871897245606	36.88986347277299
8.333333333333334	43.9639679743114	33.94438353295168
10.0	45.0	30.0
12.0	46.830880074906354	24.058452405257736

14.0	49.3235202996254	17.973928352680385
16.0	51.8764797003746	12.76619037896906
18.0	53.76911992509365	8.441547594742264
20.0	55.0	5.0
30.0	54.0	0.0
32.5	53.273872875703134	-4.569283826601964
35.0	52.34549150281253	-11.722864693592145
37.5	51.214855881328184	-17.305716173398036
40.0	50.0	-20.0
45.0	48.549208099778006	-15.0
50.0	45.0	-10.0
52.0	42.69188611699158	-12.407106781186547
54.0	39.767544467966324	-17.62842712474619
56.0	36.232455532033676	-24.37157287525381
58.0	32.80811388300842	-28.59289321881345
60.0	30.0	-30.0
62.5	27.0636104345604	-28.491116523516816
65.0	24.504441738241592	-23.964466094067262
67.5	22.3113895654396	-16.508883476483184
70.0	20.0	-10.0
72.5	17.5	-4.622779130879204
75.0	15.0	0.9911165235168156
77.5	12.5	5.872779130879204
80.0	10.0	10.0
83.33333333333333	6.666666666666671	14.329385121563025
86.66666666666666	3.333333333333343	17.31754048625211
90.0	0.0	20.0
93.33333333333333	-4.309644062711501	23.33333333333333
96.66666666666666	-8.160744349011203	26.666666666666657
100.0	-10.0	30.0
105.0	-9.344056402002256	35.0
110.0	-5.0	40.0

и рисует графики (рисунок 2).

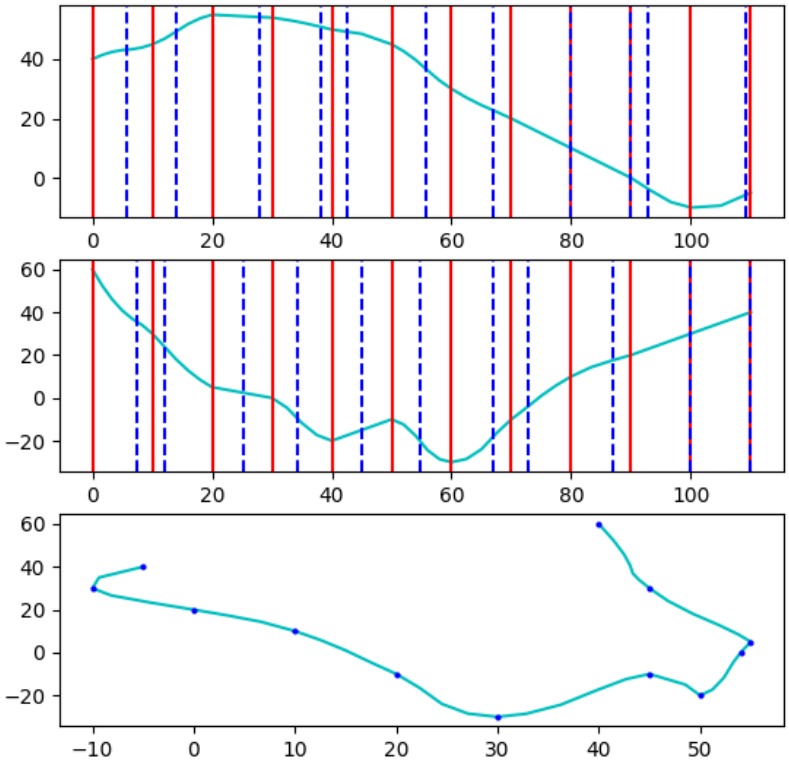


Рисунок 2. – Результат работы программы

Листинг программы

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import math as mt

def read_data(fname):
    df = pd.read_csv(fname, sep=" ", escapechar="#")
    in_val = df.values
    tn = in_val[:, 0]
    xn = in_val[:, 1]
    yn = in_val[:, 2]
    Kn = in_val[:, 3]
    Kn = Kn[~np.isnan(Kn)]
    Kn = np.array(Kn, dtype=np.int16)
    return tn, xn, yn, Kn

def write_data(fname, t, X, Y):
    out_df = pd.DataFrame(
        {"t" : t,
        "x(t)" : X,
        "y(t)" : Y}
    )
    out_df.to_csv(fname, sep="\t", index=False)

def steps(tn): # вычисление длин участков сплайна и величины шага
дискретизации для каждого сплайна
    N = yn.shape[0]
    for i in range(N - 1):
        if i == (N - 1):
            T[i] = tn[N] - tn[N - 1]
        else:
```

```
            T[i] = tn[i+1] - tn[i]
        dt[i] = (T[i]*1.0)/((Kn[i])*1.0)
    return T, dt

def get_t(dt, Kn): #вычисление аргументов сплайна
    ind = 0
    N = dt.shape[0] + 1
    PointCnt = Kn.sum()
    t = np.zeros(PointCnt + 1)
    t[PointCnt] = tn[N - 1]
    for i in range(N - 1):
        for j in range((Kn[i])):
            if ind >= PointCnt:
                break
            if j == 0:
                t[ind] = tn[i]
                ind = ind + 1
                continue
            else:
                t[ind] = t[ind - 1] + dt[i]
            ind = ind+1
    return t

def calc_dy(yn, tn): #вычисление производных
    N = yn.shape[0]
    dy = np.zeros(N)
    for i in range(1, N-1):
        dy[i] = (yn[i] - yn[i-1]) / (tn[i] - tn[i-1]) + (yn[i+1] -
yn[i])/(tn[i+1]-tn[i])
    dy /= 2
    return dy

def calc_spline2D(yn, tn, dy, T, Kn, t, dt): #построение сплайна
    print()
    ind = 0
    N = yn.shape[0]
    P = np.zeros(N - 1)
    D = np.zeros(N - 1)
    PointCnt = Kn.sum()
    Y = np.zeros(PointCnt + 1)
    for i in range(N-1):
        L = 1.0*yn[i+1] - 1.0*yn[i] - 0.5*T[i]*(dy[i+1]+dy[i])
        P[i] = (-L - np.sign(L)*np.sqrt(L*L + 0.25*T[i]*T[i]*pow(dy[i+1]-
dy[i], 2))) / (0.5*T[i]*T[i])
        if P[i] == 0 or L == 0:
            D[i] = 0
        else:
            D[i] = (dy[i+1]-dy[i]+T[i]*P[i]) / (2*P[i])
        tp = tn[i + 1] - D[i]
        for j in range((Kn[i])):
            if ind >= PointCnt:
                break
            if j == 0:
                Y[ind] = yn[i]
                ind = ind + 1
                continue
            tL = 1.0 * (t[ind] - tn[i])
            tt = 1.0*(t[ind] - tn[i])
            if t[ind] < tp:
                Y[ind] = yn[i] + dy[i] * tL - (tL * 1.0 * tL * P[i]) / 2.0
            else:
                tR = T[i] - tt
                if tR < 0:
                   tR = -tR
```

```
                Y[ind] = yn[i + 1] - dy[i + 1] * tR + pow(tR, 2) * P[i] / 2.0
            ind = ind + 1
    Y[PointCnt] = yn[N - 1]
    return Y, D

def print_parametric_spline2D(X, Y, xn, yn, id="111"): #построение графика
    plt.subplot(id)
    plt.plot(X,Y, marker="", markersize="4", c="C")
    plt.plot(xn, yn, "bo", markersize="2")

def print_spline2D(t, Y, tn, D, id="111"): #построение графика
    plt.subplot(id)
    plt.plot(t,Y, marker="", markersize="4", c="C")
    for i, v in enumerate(tn):
        plt.axvline(v,  ls="-", c="R")
        if i == 0:
            continue
        plt.axvline(tn[i] - D[i-1], ls="--", c="B")

tn, xn, yn, Kn = read_data("in.txt")
print("xn=", xn)
print("yn=", yn)
N = yn.shape[0]

#привязка К к длине участка сплайна (теорема Пифагора)
for i in range(N - 1):
    k = 0.2  # количество точек на единицу длины участка
    Kn[i] = round(mt.sqrt(pow((xn[i+1]-xn[i]), 2)+pow((yn[i+1]-yn[i]), 2))*k)
    if Kn[i] == 0:
        Kn[i]=1
dx = np.zeros(N)
dy = np.zeros(N)

dx = calc_dy(xn, tn)
dy = calc_dy(yn, tn)
dy[0]=input("Ender y'(x) in first point:\n")
dy[N-1]=input("Ender y'(x) in last point:\n")
dx[0] = (xn[1]-xn[0])/(tn[1]-tn[0])
dx[N-1] = (xn[N-1]-xn[N-2])/(tn[N-1]-tn[N-2])
dy[0] = dy[0]*dx[0]
dy[N-1] = dy[N-1]*dx[N-1]
#параметры сплайнов
T = np.zeros(N - 1)
dt = np.zeros(N-1, float)
P = np.zeros(N - 1)
D = np.zeros(N - 1)

T, dt = steps(tn)
t = get_t(dt, Kn)
X, D = calc_spline2D(xn, tn, dx, T, Kn, t, dt)
Y, D2 = calc_spline2D(yn, tn, dy, T, Kn, t, dt)
write_data("out.txt", t, X, Y)
plt.figure(1)
print_spline2D(t, X, tn, D, id="311")
print_spline2D(t, Y, tn, D2, id="312")
print_parametric_spline2D(X, Y, xn, yn, id="313")
plt.show()
```