Министерство образования и науки РФ
Национальный исследовательский технологический университет
«НИТУ МИСиС»
КИК

Курсовая работа

По дисциплине «Численные методы»

Тема: «Квадратичные сплайны для интерполяции упорядоченной системы точек»

Выполнила:
студентка гр. БПМ-18-1
Нефёдова А.Д.

Москва, 2020 г.

Оглавление

Теоретический обзор

Введём необходимые формальные определения, понятия и постановки.

Интерполяция - способ нахождения промежуточных значений величины по имеющемуся дискретному набору известных значений. При решении задач с научными и инженерными расчётами часто приходится оперировать наборами значений, полученных опытным путём или методом случайной выборки. Как правило, на основании этих наборов требуется построить функцию, на которую могли бы с высокой точностью попадать другие получаемые значения. Интерполяцией называют такую разновидность приближения функций, при которой кривая построенной функции проходит точно через имеющиеся точки данных.

Сплайн - функция, область определения которой разбита на конечное число отрезков, на каждом из которых сплайн совпадает с некоторым алгебраическим полиномом. В случае квадратичных сплайнов, степень каждого полинома равна 2.

Предполагаются заданными n точек на плоскости:

$$P_1 = \langle x_1, y_1 \rangle, P_2 = \langle x_2, y_2 \rangle, \ldots, P_n = \langle x_n, y_n \rangle \tag{1}$$

Предполагается, что задан порядок следования данных точек вдоль будущей интерполирующей кривой. Далее упомянутый порядок будет определяться нумерацией точек. Сама эта кривая не предполагается графиком некоторой функции от переменной x. На каждом отрезке она будет строиться в параметрической форме:

$$x = \varphi_x(t), y = \varphi_y(t) \tag{2}$$

Без потери общности можно предполагать, что

$$\langle \varphi_x(i), \varphi_i(i) \rangle = P_i \ (i = 1, 2, \dots, n) \qquad (3)$$

Функции $\varphi_x(t), \varphi_y(t)$ задают параметризацию искомой кривой. В данном случае рассматриваются квадратичные функции.

Функции, представляющие сплайн на отрезке $[i, i+1]$, таковы:

$$\varphi_x(t) = a_i^x t^2 + b_i^x t + c_i^x \ (i = 1, 2, \dots, n-1), \qquad (4x)$$

$$\varphi_y(t) = a_i^y t^2 + b_i^y t + c_i^y \ (i = 1, 2, \dots, n-1). \qquad (4y)$$

Условия на сплайны записываются в следующем виде:

$$a_i^x i^2 + b_i^x i + c_i^x \ (i = 1, 2, \dots, n-1) = x_i \ (i = 1, 2, \dots, n-1) \qquad (5xl)$$

$$a_i^x (i+1)^2 + b_i^x (i+1) + c_i^x \ (i = 1, 2, \dots, n-1) = x_{i+1} \ (i = 1, 2, \dots, n-1) \qquad (5xr)$$

$$a_i^y i^2 + b_i^y i + c_i^y \ (i = 1, 2, \dots, n-1) = y_i \ (i = 1, 2, \dots, n-1) \qquad (5xl)$$

$$a_i^y (i+1)^2 + b_i^y (i+1) + c_i^y \ (i = 1, 2, \dots, n-1) = y_{i+1} \ (i = 1, 2, \dots, n-1) \qquad (5xr)$$

$$2a_{i-1}^x i + b_{i-1}^x = 2a_i^x i + b_i^x \ (i = 2, 3, \dots, n-1) \qquad (6x)$$

$$2a_{i-1}^y i + b_{i-1}^y = 2a_i^y i + b_i^y \ (i = 2, 3, \dots, n-1) \qquad (6y)$$

Равенства (5*xl*) выражают условие совпадение значения *x*-ой координаты сплайна с *x*-ой координатой заданной точки на левом конце отрезков [*i*; *i*+1], равенства (5*xr*) – на правом конце этих же отрезков. Равенства (5*yl*) и (5*yr*) выражают аналогичные условия для *y*-ой координаты. Наконец, равенства (6*x*) и (6*y*) выражают условие совпадения и непрерывности производных (по параметру *t*) во всех внутренних граничных точках.

Очень важным является то обстоятельство, что все уравнения, относящиеся к *x*-координатам, отличны от всех уравнений, относящимся к у-координатам. Это значит, что системы (5*xl*), (5*xr*), (6*x*) и (5*yl*), (5*yr*), (6*y*) можно рассматривать совершенно независимо друг от друга. Легко увидеть, что в системе (5*xl*), (5*xr*), (6*x*) все переменные линейно выражаются через c_0^x, а в системе (5*yl*), (5*yr*), (6*y*) – через c_0^y. Эти выражения имеют вид:

$$a_i^x = -\frac{x_i}{i} + \frac{x_{i+1}}{i+1} + \frac{c_i}{i(i+1)}\ (i = 1, 2, \dots, n-1) \quad (7)$$

$$b_i^x = \frac{x_i\,(i+1)}{i} - \frac{x_{i+1} i}{i+1} - \frac{c_i(2i+1)}{i(i+1)}\ (i = 1, 2, \dots, n-1) \quad (8)$$

$$c_i^x = \frac{y_{i-1} i^2 (i+1)}{i-1} + y_{i+1} i^2 - 2y_i i(i+1) - \frac{c_{i-1}(i+1)}{i-1}\ (i = 2, 3, \dots, n-1) \quad (9)$$

$$a_i^y = -\frac{y_i}{i} + \frac{y_{i+1}}{i+1} + \frac{c_i}{i(i+1)}\ (i = 1, 2, \dots, n-1) \quad (10)$$

$$b_i^y = \frac{y_i\,(i+1)}{i} - \frac{y_{i+1} i}{i+1} - \frac{c_i(2i+1)}{i(i+1)}\ (i = 1, 2, \dots, n-1) \quad (11)$$

$$c_i^y = \frac{y_{i-1} i^2 (i+1)}{i-1} + y_{i+1} i^2 - 2y_i i(i+1) - \frac{c_{i-1}(i+1)}{i-1}\ (i = 2, 3, \dots, n-1) \quad (12)$$

Таким образом, мы можем найти все коэффициенты (4x), (4y), кроме c_1^x и c_1^y. Эти коэффициенты мы будем находить с помощью метода, описанного далее.

Метод нахождения оптимальной интерполирующей кривой

Для нахождения c_1 воспользуемся следующем алгоритмом:

1. Выберем такой отрезок [c_{left}, c_{right}], что при $c_1 = c_{left}$ и $c_1 = c_{right}$ на двух получившихся интерполирующих кривых существует хотя бы один отрезок [i, i+1], на котором коэффициенты a_i полиномов на этом отрезке имеют противоположный знак, то есть две прямые выпуклы в разные стороны:

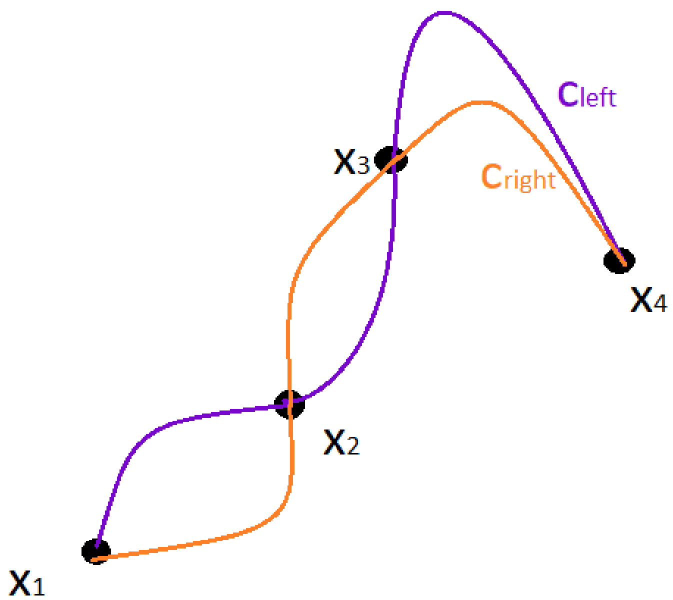


Рис. 1 Пример выбора коэффициентов C_{left} и C_{right}

В случае рисунка 1 условия пункта 1 выполняются – существует 2 отрезка [x_1, x_2] и [x_2, x_3], на которых функции имеют выпуклости разных знаков.

2. Далее выбирается точка $c_{middle} = \frac{c_{right} - c_{left}}{2}$, получаем два отрезка: [c_{left}, c_{middle}], [c_{middle}, c_{right}].

3. Построим для с из [c_{left}, c_{middle}] с шагом $h = \frac{c_{middle} - c_{left}}{100}$ интерполирующие кривые и посчитаем количество отрезков, на которых есть хотя бы одна функцию, имеющая отличный от других знак выпуклости. Если же таких отрезков нет, то решение находится на правом отрезке [c_{middle}, c_{right}].
4. Аналогично считаем количество отрезков, на которых функции имеют разную выпуклость для [c_{middle}, c_{right}].
5. Сравниваем количество отрезков с разной выпуклостью, полученных из п.3 и п.4. Если их больше на [c_{left}, c_{middle}], то в качестве c_{right} берем c_{middle}. Если больше на [c_{middle}, c_{right}], то c_{left} = c_{middle}. Если же их количество совпадает, то меняем любую границу.
6. Повторяем п.2 – п. 5 до тех пор, пока длина отрезка [c_{left}, c_{right}]> 0.0001
7. Таким образом, мы нашли c_1 = c_{middle}.

Данный метод применяется для нахождения $x = \varphi_x(t)$ и $y = \varphi_y(t)$. Далее строится кривая *f* = *y(x)*.

Применение алгоритма к конкретным данных

Дано 27 вариантов, содержащих по 10 точек. Первая колонка – номер точки, вторая – координата x, третья – координата y. Необходимо найти функции $x = \varphi_x(t), y = \varphi_y(t)$ и построить кривую $f = y(x)$. Варианты имеют следующий вид:

Вариант 1

1 -3.001 10.078

2 -2.400 5.556

3 -1.800 3.108

4 -1.198 1.810

5 -0.599 1.186

6 -0.005 0.995

7 0.603 1.191

8 1.197 1.804

9 1.796 3.093

10 2.404 5.581

Вариант 2

1 1.001 0.001

2 0.827 0.563

3 0.366 0.930

4 -0.223 0.974

5 -0.731 0.682

6 -0.989 0.150

7 -0.901 -0.432

8 -0.501 -0.864

9 0.073 -0.996

10 0.621 -0.784

Вариант 3

1 1.001 0.001

2 1.009 0.335

3 1.037 0.679

4 1.085 1.043

5 1.152 1.435

6 1.249 1.874

7 1.380 2.380

8 1.557 2.983

9 1.795 3.726

10 2.126 4.689

Вариант 4

1 -2.697 7.456

2 -2.105 4.159

3 -1.499 2.352

4 -0.901 1.433

5 -0.303 1.043

6 0.300 1.045

7 0.903 1.439

8 1.500 2.352

9 2.103 4.161

10 2.696 7.443

Вариант 5

1 0.957 0.296

2 0.625 0.782

3 0.075 0.997

4 -0.498 0.867

5 -0.899 0.437

6 -0.988 -0.146

7 -0.734 -0.678

8 -0.226 -0.976

9 0.362 -0.933

10 0.824 -0.566

Вариант 6

1 1.002 0.166

2 1.021 0.505

3 1.057 0.857

4 1.114 1.232

5 1.196 1.647

6 1.311 2.118

7 1.463 2.668

8 1.667 3.334

9 1.946 4.173

10 2.340 5.288

Вариант 7

1 -1.499 2.352

2 -1.198 1.810

3 -0.901 1.433

4 -0.599 1.186

5 -0.303 1.043

6 -0.005 0.995

7 0.300 1.045

8 0.603 1.191

9 0.903 1.439

10 1.197 1.804

Вариант 8

1 -0.498 0.867

2 -0.731 0.682

3 -0.899 0.437

4 -0.989 0.150

5 -0.988 -0.146

6 -0.901 -0.432

7 -0.734 -0.678

8 -0.501 -0.864

9 -0.226 -0.976

10 0.073 -0.996

Вариант 9

1 1.152 1.435

2 1.196 1.647

3 1.249 1.874

4 1.311 2.118

5 1.380 2.380

6 1.463 2.668

7 1.557 2.983

8 1.667 3.334

9 1.795 3.726

10 1.946 4.173

Вариант 10

1 5.004 0.001

2 4.830 0.015

3 4.361 0.127

4 3.653 0.406

5 2.818 0.892

6 1.968 1.570

7 1.211 2.386

8 0.630 3.246

9 0.247 4.031

10 0.012 4.831

Вариант 11

1 0.001 0.001

2 0.451 0.069

3 0.899 0.277

4 1.299 0.625

5 1.552 1.057

6 1.548 1.435

7 1.266 1.586

8 0.839 1.452

9 0.438 1.113

10 0.157 0.680

Вариант 12

1 -0.005 -0.005

2 0.032 0.171

3 0.262 0.630

4 0.818 1.221

5 1.711 1.732

6 2.839 1.988

7 4.025 1.906

8 5.047 1.497

9 5.782 0.928

10 6.282 0.049

Вариант 13

1 4.959 0.002

2 4.634 0.055

3 4.035 0.244

4 3.254 0.626

5 2.396 1.212

6 1.578 1.968

7 0.897 2.818

8 0.413 3.655

9 0.134 4.362

10 0.012 4.831

Вариант 14

1 0.225 0.017

2 0.676 0.154

3 1.109 0.434

4 1.451 0.837

5 1.587 1.264

6 1.438 1.548

7 1.061 1.554

8 0.628 1.301

9 0.279 0.901

10 0.068 0.453

Вариант 15

1 0.002 0.042

2 0.114 0.375

3 0.495 0.922

4 1.222 1.495

5 2.259 1.903

6 3.440 1.992

7 4.567 1.735

8 5.465 1.230

9 6.019 0.641

10 6.245 0.173

Вариант 16

1 4.035 0.244

2 3.653 0.406

3 3.254 0.626

4 2.818 0.892

5 2.396 1.212

6 1.968 1.570

7 1.578 1.968

8 1.211 2.386

9 0.897 2.818

10 0.630 3.246

Вариант 17

1 0.676 0.154

2 0.899 0.277

3 1.109 0.434

4 1.299 0.625

5 1.451 0.837

6 1.552 1.057

7 1.587 1.264

8 1.548 1.435

9 1.438 1.548

10 1.266 1.586

Вариант 18

1 -0.005 -0.005

2 0.032 0.171

3 0.262 0.630

4 0.818 1.221

5 1.711 1.732

6 2.259 1.903

7 3.440 1.992

8 4.567 1.735

9 5.465 1.230

10 6.019 0.641

Вариант 19

1 0.999 -0.001

2 1.286 0.194

3 1.464 1.180

4 0.459 2.383

5 -1.536 2.361

6 -2.932 0.599

7 -2.430 -1.644

8 -0.507 -2.598

9 1.134 -1.850

10 1.472 -0.592

Вариант 20

1 0.000 0.000

2 0.496 0.338

3 0.439 1.115

4 -0.397 1.750

5 -1.753 1.628

6 -2.957 0.450

7 -3.237 -1.552

8 -2.100 -3.620

9 0.347 -4.772

10 3.345 -4.217

Вариант 21

1 0.980 0.109

2 0.950 0.180

3 0.901 0.241

4 0.836 0.290

5 0.760 0.330

5 0.669 0.351

7 0.564 0.352

8 0.446 0.326

9 0.303 0.258

10 0.072 0.072

Вариант 22

1 1.085 0.026

2 1.470 0.588

3 1.139 1.843

4 -0.496 2.598

5 -2.423 1.653

6 -2.936 -0.588

7 -1.547 -2.354

8 0.449 -2.388

9 1.290 -0.197

10 1.086 -0.028

Вариант 23

1 0.285 0.088

2 0.559 0.701

3 0.112 1.490

4 -1.044 1.815

5 -2.422 1.172

6 -3.252 -0.484

7 -2.854 -2.638

8 -1.008 -4.371

9 1.845 -4.737

10 4.687 -3.213

Вариант 24

1 0.993 0.075

2 0.968 0.146

3 0.926 0.210

4 0.871 0.267

5 0.800 0.311

6 0.714 0.341

7 0.616 0.353

8 0.505 0.341

9 0.377 0.298

10 0.214 0.197

Вариант 25

1 0.459 2.383

2 -0.496 2.598

3 -1.536 2.361

4 -2.423 1.653

5 -2.932 0.599

6 -2.936 -0.588

7 -2.430 -1.644

8 -1.547 -2.354

9 -0.507 -2.598

10 0.449 -2.388

Вариант 26

1 0.596 0.138

2 0.496 0.338

3 0.439 1.115

4 -0.397 1.750

5 -1.753 1.628

6 -2.422 1.172

7 -3.252 -0.484

8 -2.854 -2.638

9 -1.008 -4.371

10 1.845 -4.737

Вариант 27

1 0.871 0.267

2 0.836 0.290

3 0.800 0.311

4 0.760 0.330

5 0.714 0.341

6 0.669 0.351

7 0.616 0.353

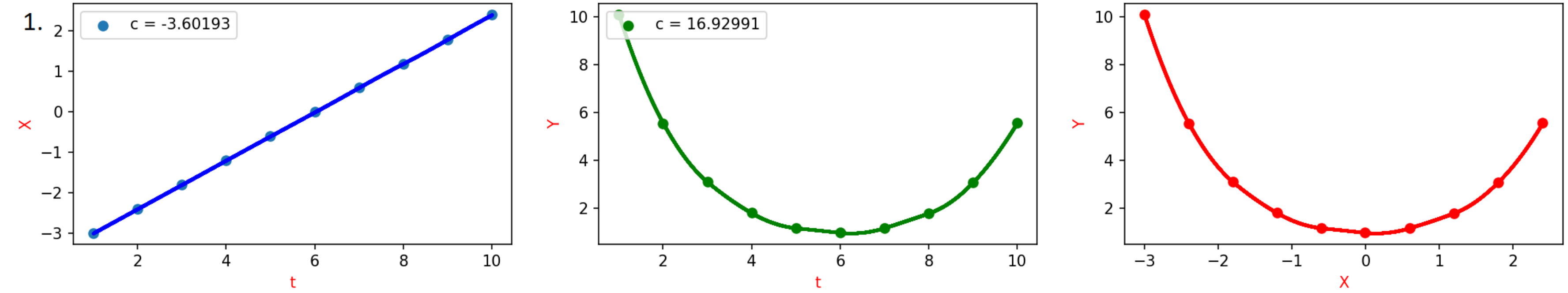
8 0.564 0.352

9 0.505 0.341

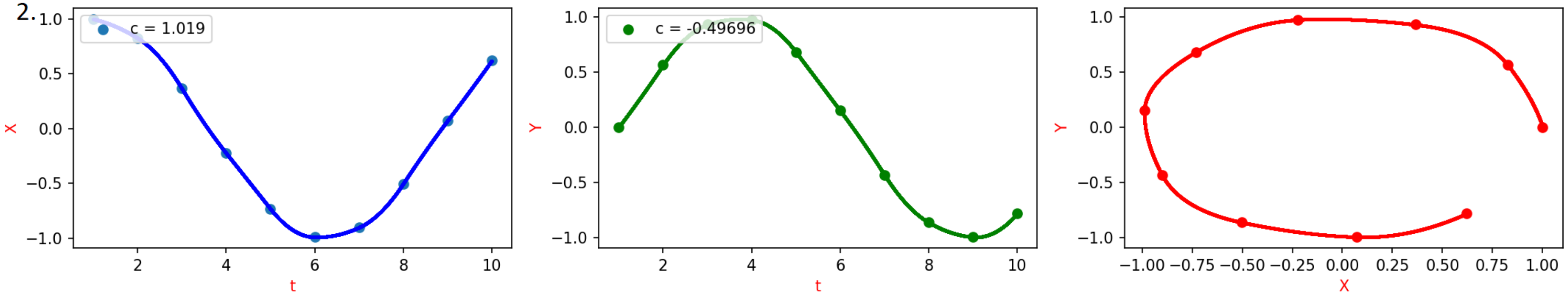
10 0.446 0.

Результаты работы программы для каждого варианты приведены ниже. На первом рисунке изображена зависимость $x = \varphi_x(t)$, на втором $y = \varphi_y(t)$, на третьем $f = y(x)$.

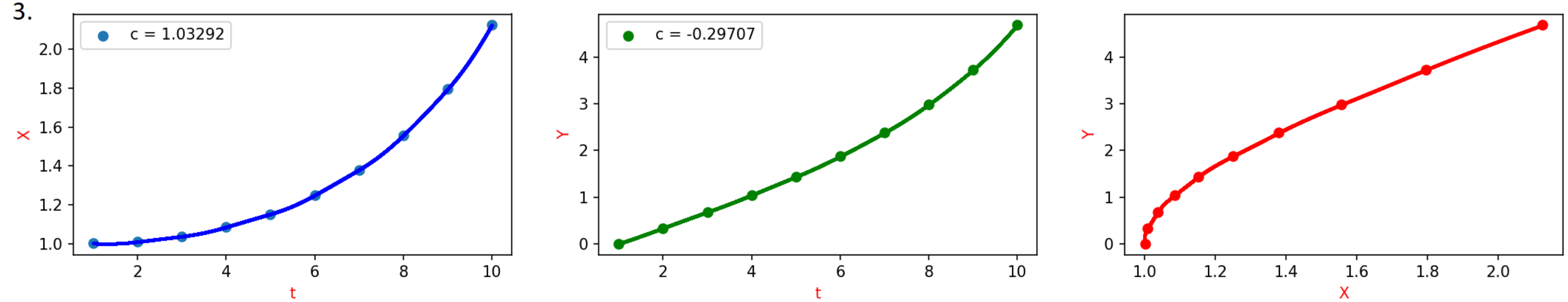
1.

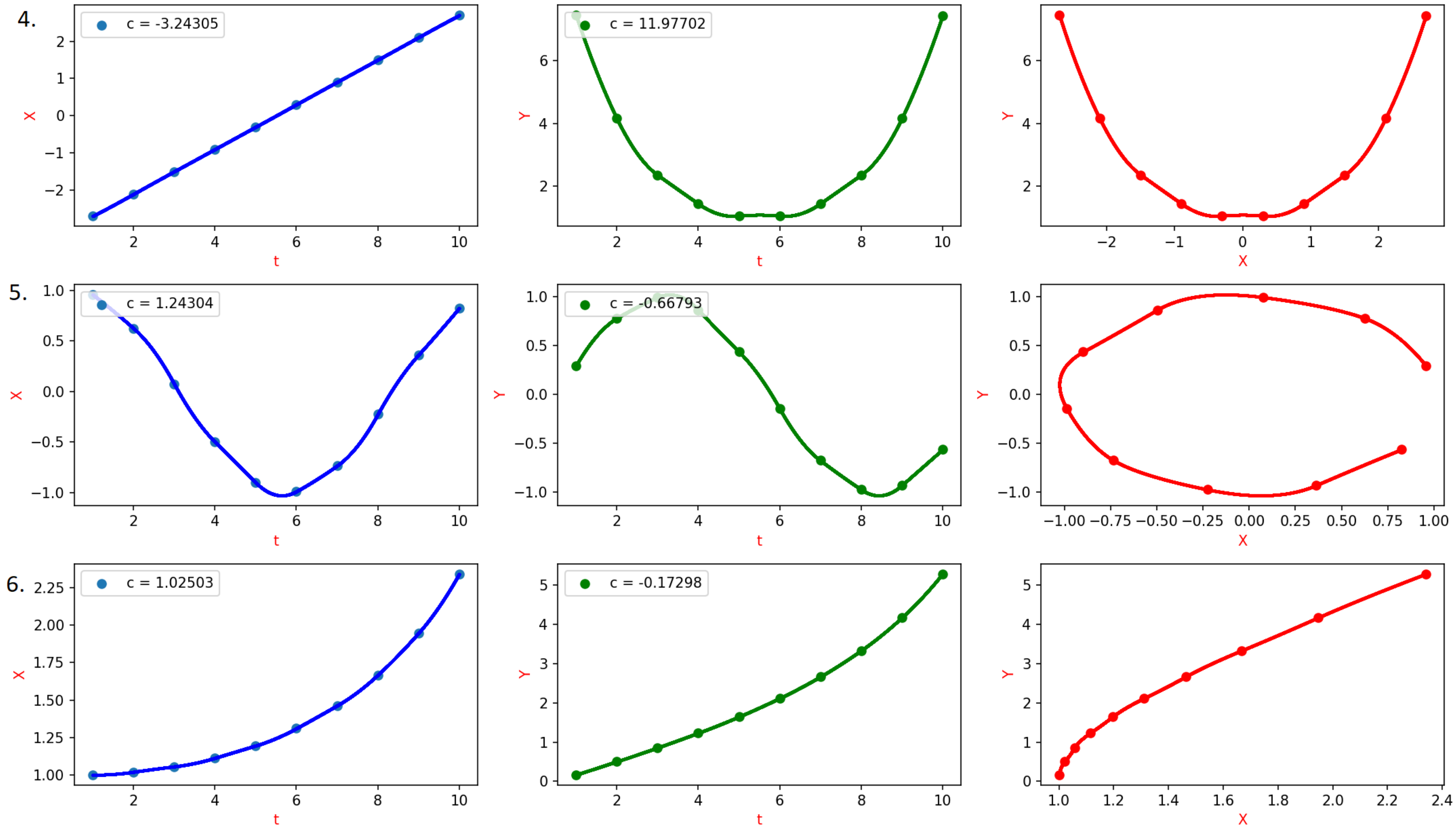


2.

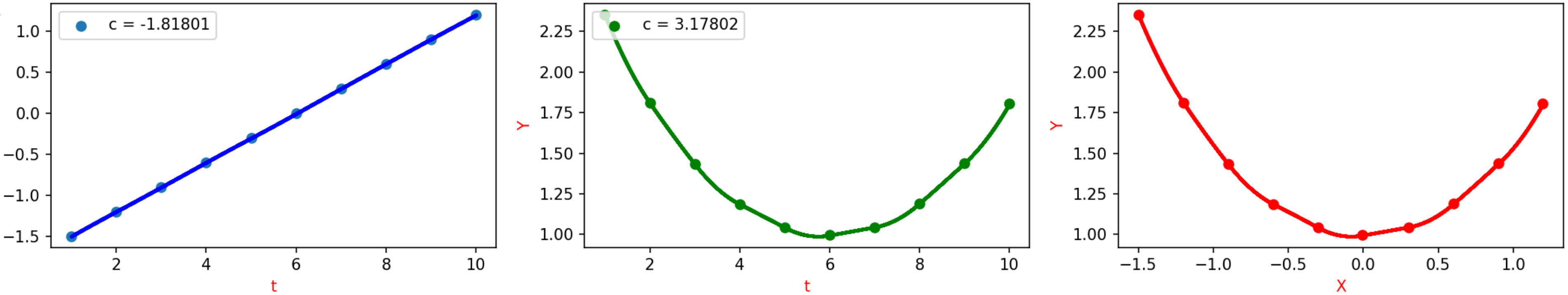


3.

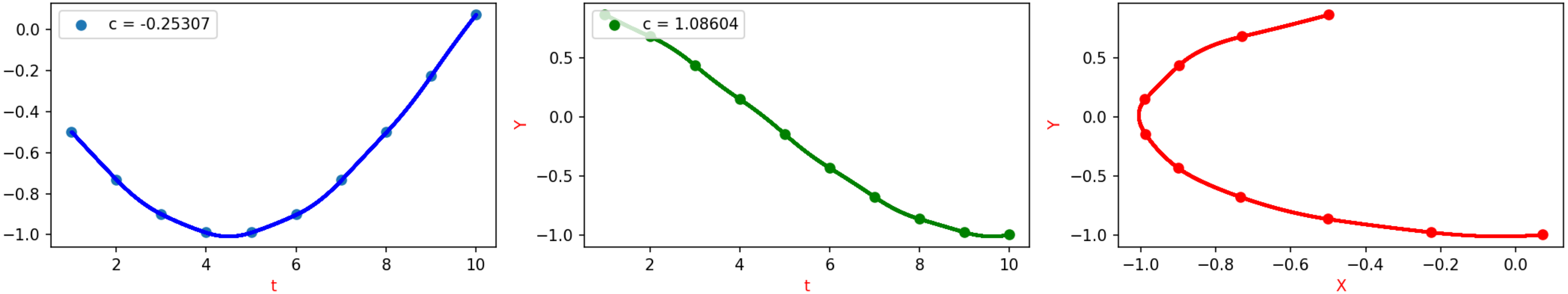



4.
c = -3.24305
c = 11.97702
5.
c = 1.24304
c = -0.66793
6.
c = 1.02503
c = -0.17298
X
Y
t

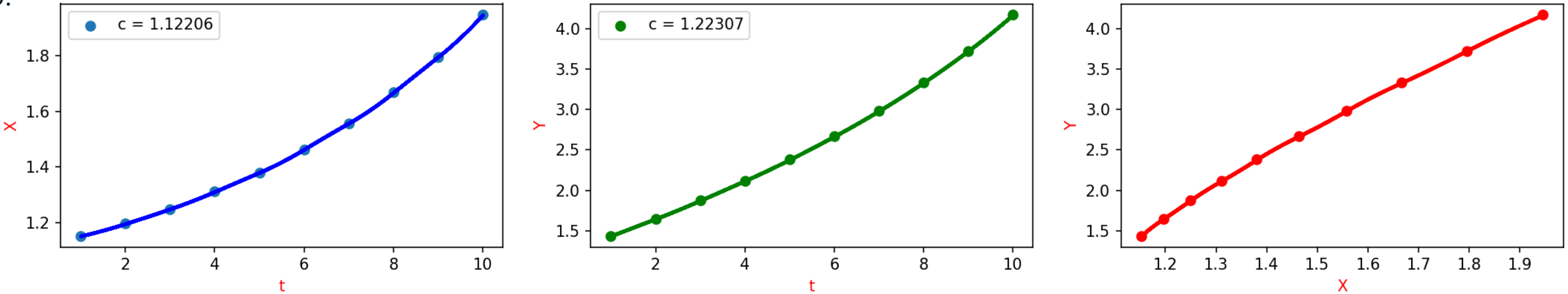
7.


c = -1.81801
c = 3.17802
t
X
Y

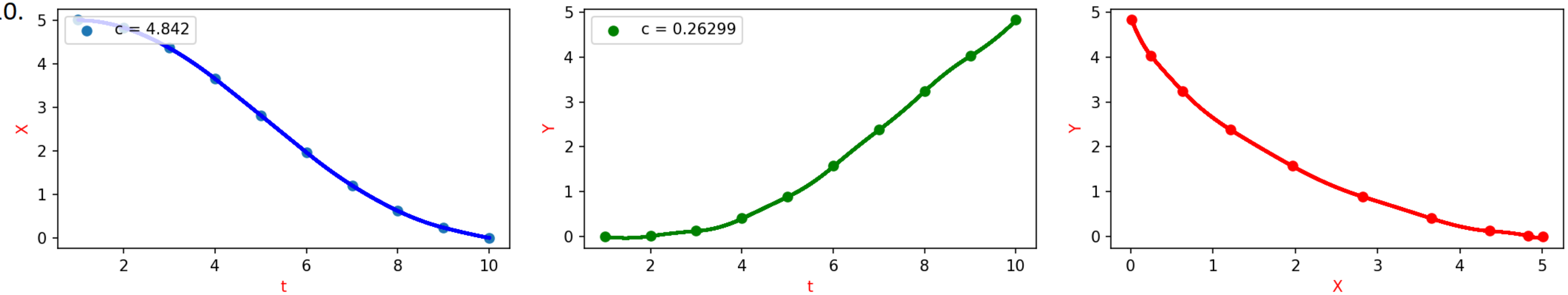
8.


c = -0.25307
c = 1.08604
t
X
Y

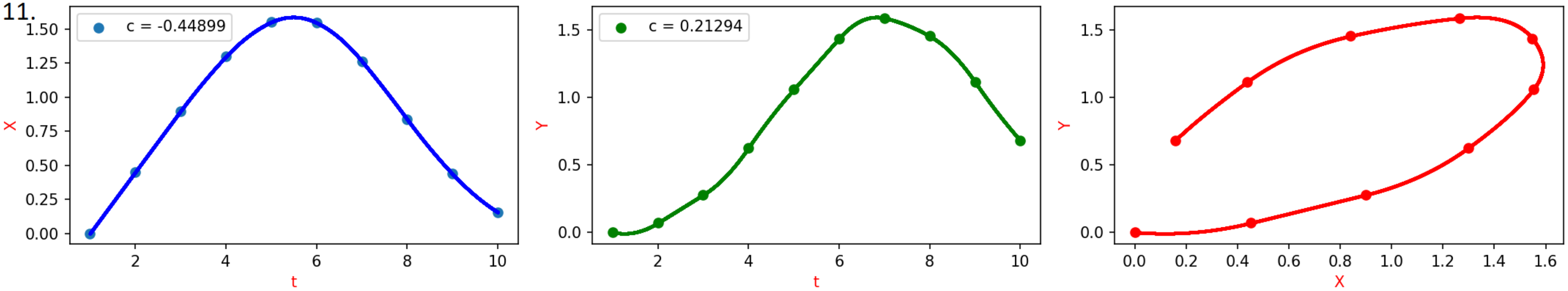
9.


c = 1.12206
c = 1.22307
t
X
Y

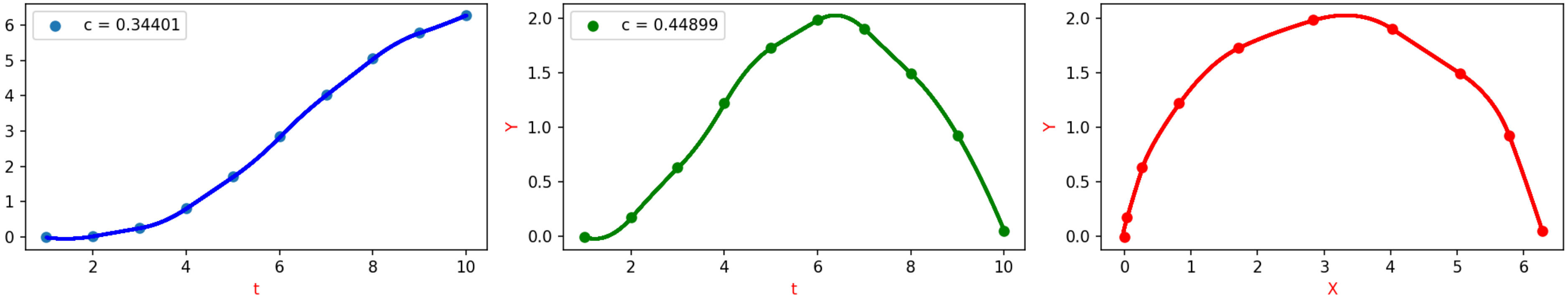
10.



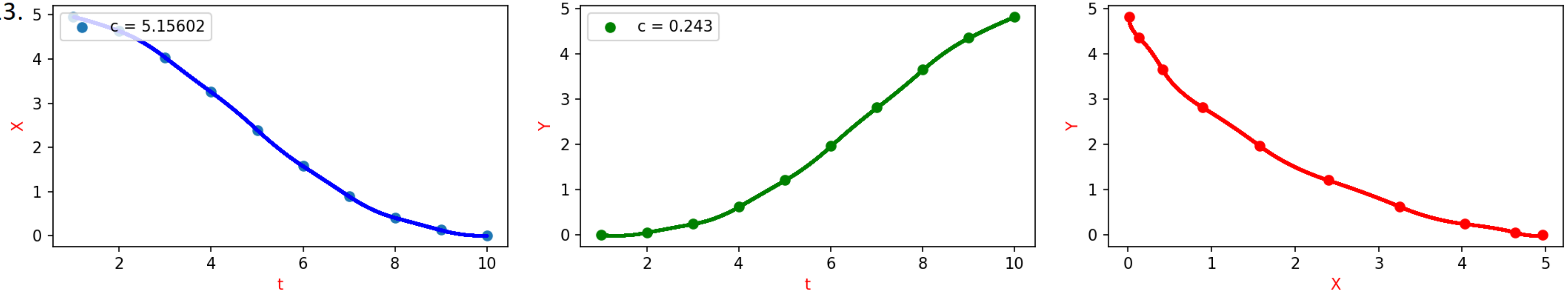
11.



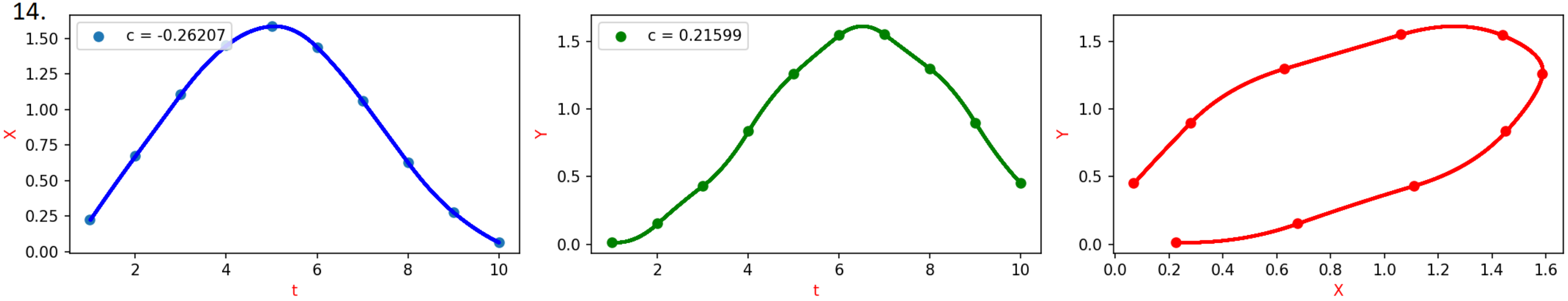
12.



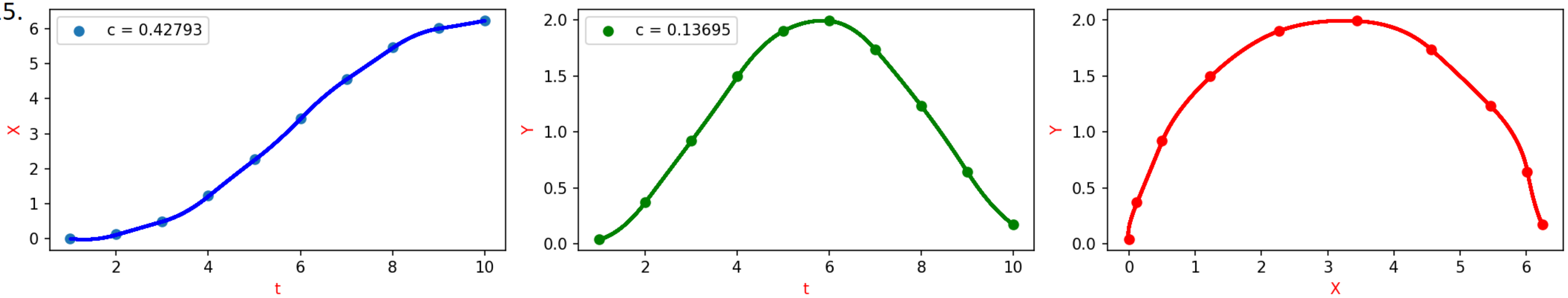
13.


c = 5.15602
c = 0.243
X
Y
t
X

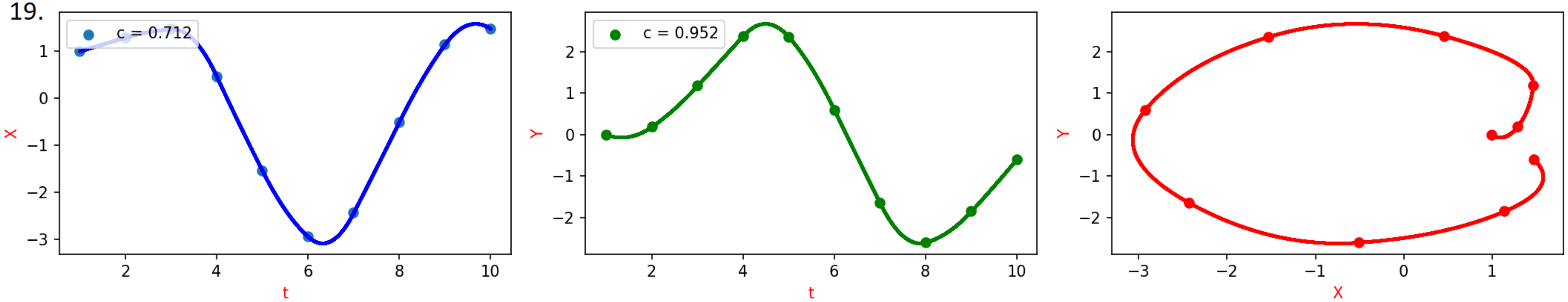
14.


c = -0.26207
c = 0.21599
X
Y
t
X

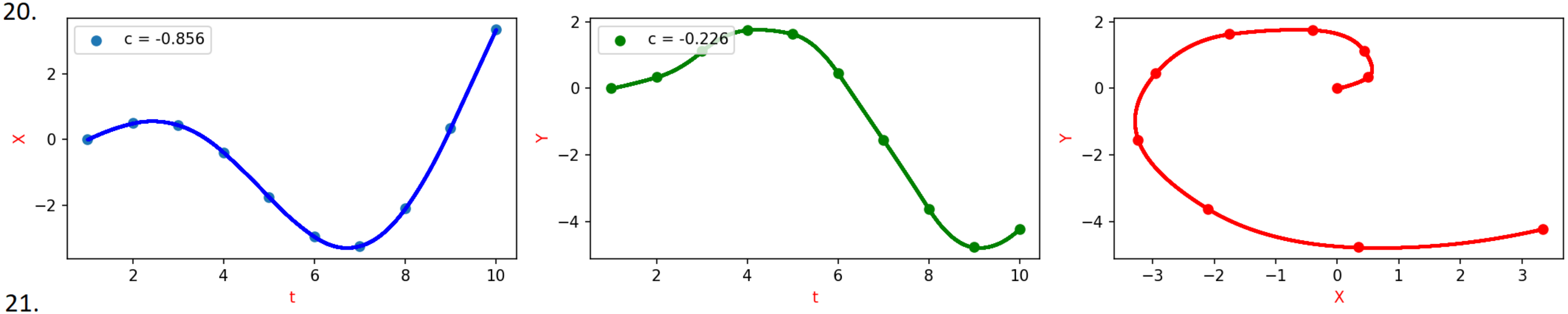
15.


c = 0.42793
c = 0.13695
X
Y
t
X

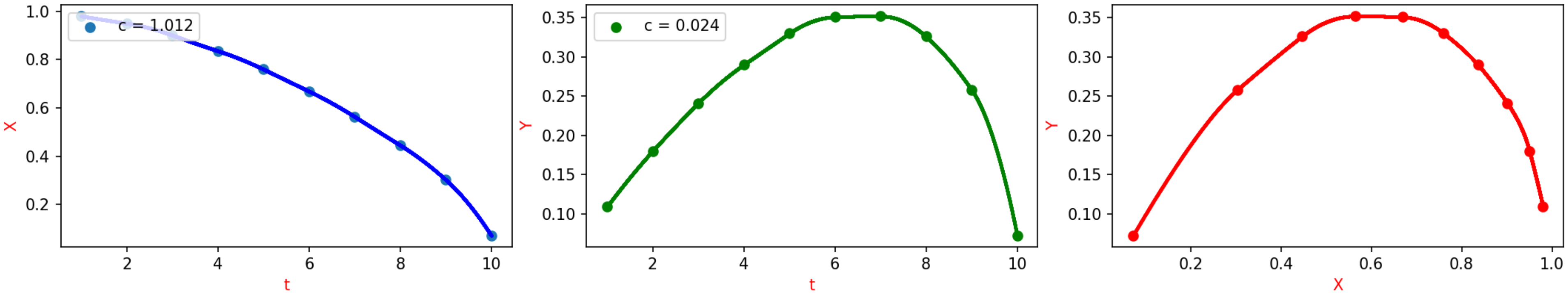
19.



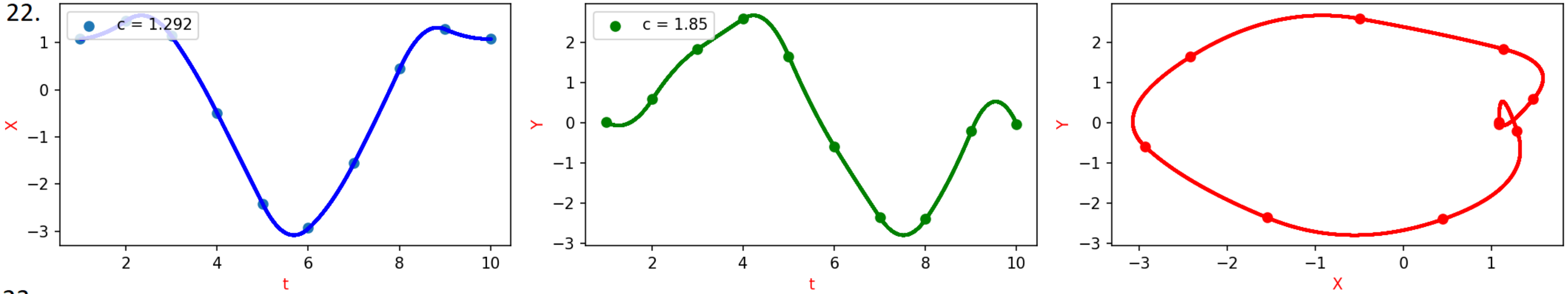
20.



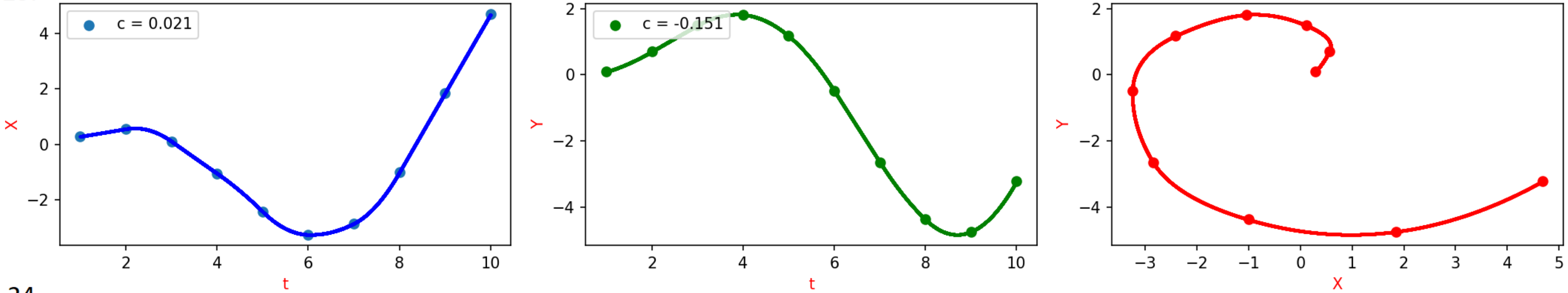
21.



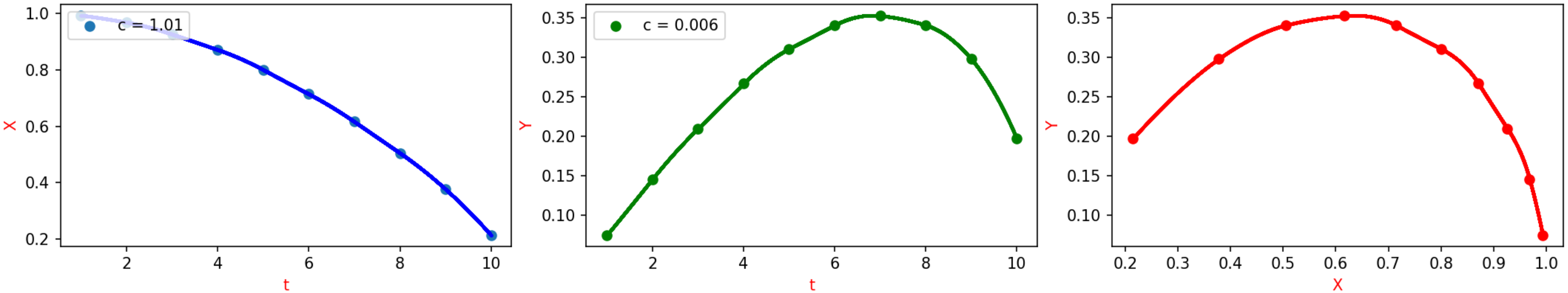
22.



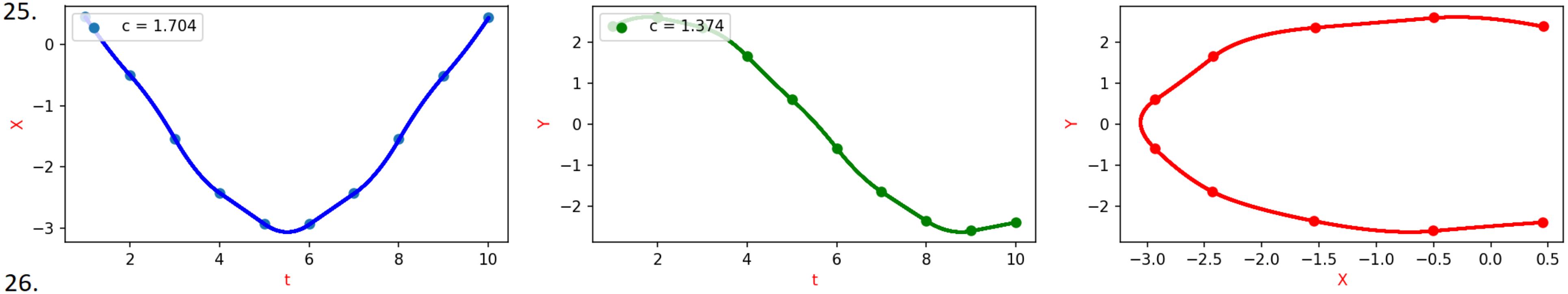
23.



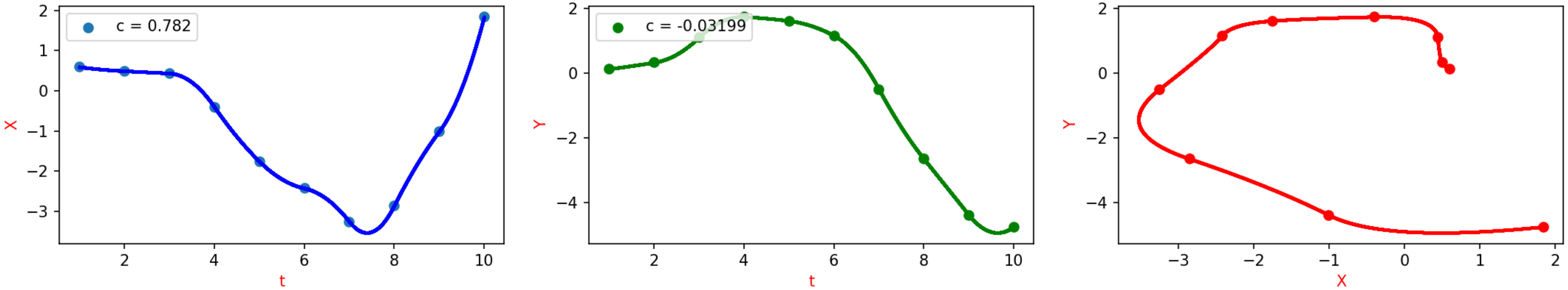
24.



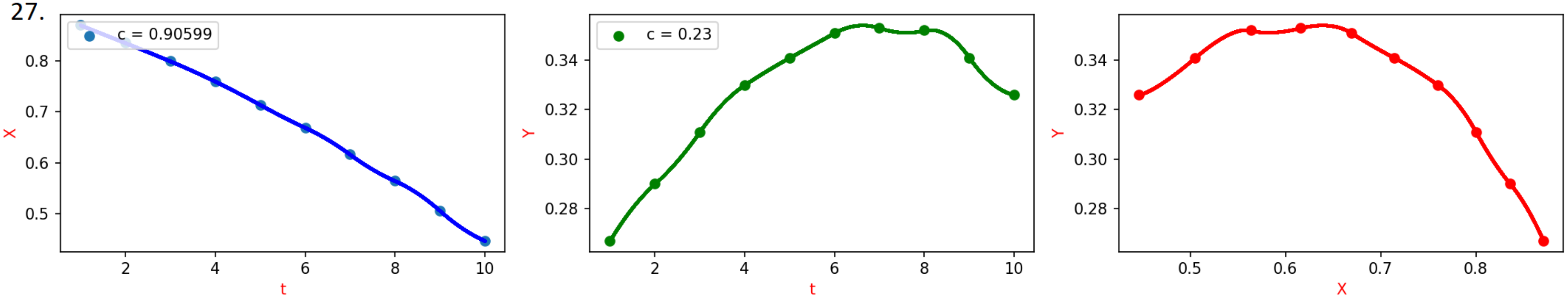
25.



26.



27.



Приложение

Код программы (Python 3.8.6):

```
import matplotlib.pyplot as plt

import numpy as np

from scipy import linalg


lines = []

with open("forSplines.txt", 'r') as file:

    lines = file.readlines()


# получаем данные для варианта

def initData(num):

    X = []

    Y = []

    variant = lines[(num  - 1 ) * 12 + 1: (num  - 1 ) * 12 + 11]


    for line in variant:

        line = line[0:len(line) - 1]

        numbs = line.split(' ')

        X.append(float(numbs[1]))

        Y.append(float(numbs[2]))
```

```
    return(X, Y)

# функция для вычисления значений многочлена второго порядка

def f(x, a, b, c):

    X = []

    for i in x:

        X.append(a * i**2 + b * i + c)

    return(X)

# нахождение коэффициентов интерполяционной кривой

def solve(c, X, Y):

    X01 = [[X[0] ** 2, X[0]], [X[1] ** 2, X[1]]]

    Y01 = [Y[0] - c, Y[1] - c]

    AB = linalg.solve(X01, Y01)


    C = [c]

    A = [AB[0]]

    B = [AB[1]]


    for i in range(1, 9):

        C.append(X[i+1]*X[i]/(X[i]-X[i+1]) *

        (2*A[i-1]*X[i] + B[i-1] + (-X[i]*Y[i+1] + Y[i] * X[i+1])/(X[i+1]*(X[i+1]
- X[i])) - Y[i]/X[i]))

        A.append(Y[i+1]/(X[i+1]*(X[i+1] - X[i])) - Y[i]/(X[i]*(X[i+1] - X[i])) +
C[i]/(X[i+1] * X[i]))
```

```
        B.append(Y[i]/X[i] - A[i]*X[i] - C[i]/X[i])


    return(A, B, C)
# отрисовка сплайна по кусочкам
def paint(X, A, B, C, color):
    for i in range(0, 9):
        plt.scatter(np.arange(X[i], X[i+1], 0.001), f(np.arange(X[i], X[i+1], 0.001), A[i], B[i], C[i]), c = color, s=1)
# определяет, сколько отрезков, где функция имеет разный знак выпуклости. Cnt – число таких отрезков. Если выпуклость нигде не меняется, то q = False и cnt = 0.
def diffSignOfA(A):
    q = False
    cnt = 0
    for j in range(0, len(A) - 1):
        for i in range(0, len(A[j])):
            if (np.sign(A[j][i]) != np.sign(A[j + 1][i])):
                q = True
                cnt += 1
                break
    return(q, cnt)
# нахождение C с помощью деления исходного отрезка.
```

```
def findC(X, Y):
# левая и правая границы отрезка
    arrx = -100
    arry = 100

    eps = 0.000001
    print(X, Y)

    while(arry - arrx > eps):
        # Все коэффициенты A для каждого отрезка
        A1_solutions = []
        A2_solutions = []

        arrm = (arrx + arry) / 2
        step = (arrm - arrx)/100
        arrxm = np.arange(arrx, arrm, step)
        arrmy = np.arange(arrm, arry, step)

        print(arrx, arrm, arry)
        for c in arrxm:
            solution1 = solve(c, X, Y)
            A1_solutions.append(solution1[0])
        for c in arrmy:
```

```
            solution2 = solve(c, X, Y)
            A2_solutions.append(solution2[0])


        signxm = diffSignOfA(A1_solutions)
        signmy = diffSignOfA(A2_solutions)
        print(signxm, signmy)
        # сравниваем знаки выпуклости и количество смен выпуклости
        if (signxm[0] and not signmy[0]):
            arry = arrm
        elif (not signxm[0] and signmy[0]):
            arrx = arrm
        elif (signxm[0] and signmy[0]):
            if (signxm[1] > signmy[1]):
                arry = arrm
            else:
                arrx = arrm
    return(arrm)
# переменная для отрисовки графиков
numb = 1
for variant in range(25, 28):
    colors = ['blue', 'green', 'red']


    Y1, Y2 = initData(variant)
```

```
T = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# коэффициенты c для x = φ_x(t), y = φ_y(t)
answer1 = findC(T, Y1)
answer2 = findC(T, Y2)

# далее идет построение графиков
plt1 = plt.subplot(3, 3, numb)
plt1.set_xlabel('t', color = 'red')
plt1.set_ylabel('X', color = 'red')

plt.scatter(T, Y1, label = 'c = ' + str(round(answer1, 5)))
plt.legend(loc=2, prop={'size': 10})
solutionAns1 = solve(answer1, T, Y1)

X = []
for i in range(0, 9):
    X.append(f(np.arange(T[i], T[i+1], 0.001), solutionAns1[0][i], solutionAns1[1][i], solutionAns1[2][i]))

X = np.array(X)
X = X.reshape((X.shape[0] * X.shape[1]))
paint(T, solutionAns1[0], solutionAns1[1], solutionAns1[2], colors[0])
```

```
        plt2 = plt.subplot(3, 3, numb + 1)
        plt2.set_xlabel('t', color = 'red')
        plt2.set_ylabel('Y', color = 'red')

        plt.scatter(T, Y2, c = 'green', label = 'c = ' + str(round(answer2, 5)))
        plt.legend(loc=2, prop={'size': 10})

        solutionAns2 = solve(answer2, T, Y2)

        Y = []
        for i in range(0, 9):
            Y.append(f(np.arange(T[i], T[i+1], 0.001), solutionAns2[0][i], solutionAns2[1][i], solutionAns2[2][i]))

        Y = np.array(Y)
        Y = Y.reshape(Y.shape[0] * Y.shape[1])
        paint(T, solutionAns2[0], solutionAns2[1], solutionAns2[2], colors[1])

        plt3 = plt.subplot(3, 3, numb + 2)
        plt.scatter(Y1, Y2, c = 'red')
        plt3.set_xlabel('X', color = 'red')
        plt3.set_ylabel('Y', color = 'red')
```

```
    plt.scatter(X, Y, c = 'red', s = 1)
    numb += 3

plt.subplots_adjust(wspace = 2)
plt.show()
```