

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Дальневосточный федеральный университет»**

|  |
| --- |
| **ШКОЛА ЕСТЕСТВЕННЫХ НАУК**  **кафедра информатики, математического и компьютерного моделирования** |

**О Т Ч Е Т**

о выполнении лабораторной работы №3 по предмету «Методы сплайн-функций»

Направление подготовки 01.03.02 «Прикладная математика и информатика»

|  |  |  |
| --- | --- | --- |
|  |  | Выполнил студент гр. Б8116-01.03.02-ПМИ  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Глушков В.К. |
|  |  | (*подпись*) *(Ф.И.О.)* |
|  |  |  |

г. Владивосток

2020

## Цель работы

Построить параметрический рациональный сплайн с четвертым типом краевых условий через параметризацию по суммарной длине хорд для функции «Логарифмическая спираль».

## Задание

Интерполировать функцию «Логарифмическая спираль»

|  |  |
| --- | --- |
|  | (1) |

параметрическим рациональным сплайном с четвертым типом краевых условий.

## Выполнение задание

Интерполяционный рациональный сплайн есть совокупность двух рациональных сплайнов , интерполирующих соответственно координаты точек , кривой Если в качестве параметра взять суммарную длину хорд , то рациональный параметрический сплайн может быть записан в виде

|  |  |
| --- | --- |
|  | (2) |

где – заданные числа , Таким образом, заменяется на или , а заменяется на для вычисления коэффициентов сплайна.

Четвертое краевое условие выглядит следующим образом

|  |  |
| --- | --- |
|  | (3) |

где

|  |  |
| --- | --- |
|  | (4) |

Учитывая условия интерполяции, выразим через , определим коэффициенты так, чтобы были непрерывны первая и вторая производные, и найдем формулы для их вычисления

|  |  |
| --- | --- |
|  | (5) |

Опуская несложные выкладки, выпишем системы уравнений относительно неизвестных для четвертого типа краевых условий

|  |  |
| --- | --- |
|  | (6) |

|  |  |
| --- | --- |
|  | (7) |

Сплайн для строится таким же образом с заменой на .

Решив эту систему, можно вычислить коэффициенты параметрического сплайна.

## Код программы

|  |
| --- |
| main.py |
| import numpy as np import matplotlib.pyplot as plt import math   def func\_x(a, b, t) -> float:  return a \* np.exp(b \* t) \* np.cos(t)  # return 5 \* np.tan(t)   def func\_y(a, b, t) -> float:  return a \* np.exp(b \* t) \* np.sin(t)  # return 5 \* np.power(np.cos(t), 2)   class Lab3:  def \_\_init\_\_(self, p, q, points):  self.p = p  self.q = q  self.points = points  self.n = len(points) - 1  self.is\_x = True  self.m\_res = []   n = self.n   d = lambda i: self.\_\_calc\_dist\_\_(i)  c = lambda i: self.\_\_calc\_c\_term\_\_(i)  P = lambda i: self.\_\_calc\_p\_\_(i)  Q = lambda i: self.\_\_calc\_q\_\_(i)   self.s = np.zeros(n + 1)  for i in range(n + 1):  self.s[i] = sum([d(j) for j in range(0, i - 1)])   y0 = d(0) / d(1)  yn = d(n - 1) / d(n - 2)   term1 = 2 \* (self.\_\_get\_node\_\_(1) - self.\_\_get\_node\_\_(0)) / d(0)  term2 = 2 \* np.power(y0, 2)  term3 = (self.\_\_get\_node\_\_(2) - self.\_\_get\_node\_\_(1)) / d(1)  c1\_star = term1 - term2 \* term3   term4 = 2 \* (self.\_\_get\_node\_\_(n) - self.\_\_get\_node\_\_(n - 1)) / d(n - 1)  term5 = 2 \* np.power(yn, 2)  term6 = (self.\_\_get\_node\_\_(n - 1) - self.\_\_get\_node\_\_(n - 2)) / d(n - 2)  c2\_star = term4 - term5 \* term6 # c\_(n-1)\*   m\_a = np.zeros((n + 1) \*\* 2).reshape(n + 1, n + 1)  m\_a[0][0] = 1  m\_a[0][1] = -(np.power(y0, 2) - 1)  m\_a[0][2] = -np.power(y0, 2)   term7 = self.\_\_calc\_lmbd\_\_(1) \* P(0) \* (1 + np.power(y0, 2) + self.\_\_get\_q\_\_(0))  term8 = self.\_\_calc\_mu\_\_(1) \* Q(1) \* (2 + self.\_\_get\_p\_\_(1))  m\_a[1][1] = term7 + term8  m\_a[1][2] = self.\_\_calc\_mu\_\_(1) \* Q(1) + self.\_\_calc\_lmbd\_\_(1) \* P(0) \* np.power(y0, 2)   for i in range(2, n - 1):  m\_a[i][i - 1] = self.\_\_calc\_lmbd\_\_(i) \* P(i - 1)  iterm1 = self.\_\_calc\_lmbd\_\_(i) \* P(i - 1) \* (2 + self.\_\_get\_q\_\_(i - 1))  iterm2 = self.\_\_calc\_mu\_\_(i) \* Q(i) \* (2 + self.\_\_get\_p\_\_(i))  m\_a[i][i] = iterm1 + iterm2  m\_a[i][i + 1] = self.\_\_calc\_mu\_\_(i) \* Q(i)   m\_a[n - 1][n - 2] = self.\_\_calc\_lmbd\_\_(n - 1) \* P(n - 2) + self.\_\_calc\_mu\_\_(n - 1) \* np.power(yn, 2) \* Q(n - 1)   term13 = self.\_\_calc\_lmbd\_\_(n - 1) \* P(n - 2) \* (2 + self.\_\_get\_q\_\_(n - 2))  term14 = self.\_\_calc\_mu\_\_(n - 1) \* Q(n - 1) \* (1 + np.power(yn, 2) + self.\_\_get\_p\_\_(n - 1))  m\_a[n - 1][n - 1] = term13 + term14   m\_a[n][n - 2] = -np.power(yn, 2)  m\_a[n][n - 1] = -(np.power(yn, 2) - 1)  m\_a[n][n] = 1   m\_b = np.zeros(n + 1)  m\_b[0] = c1\_star  m\_b[1] = c(1) - self.\_\_calc\_lmbd\_\_(1) \* P(0) \* c1\_star  for i in range(2, n - 1):  m\_b[i] = c(i)  m\_b[n - 1] = c(n - 1) - self.\_\_calc\_mu\_\_(n - 1) \* Q(n - 1) \* c2\_star  m\_b[n] = c2\_star   self.m\_res = np.linalg.solve(m\_a, m\_b)   def s\_x(self, t):  return self.\_\_s\_\_(t, True)   def s\_y(self, t):  return self.\_\_s\_\_(t, False)   def \_\_calc\_a\_\_(self, i):  return self.\_\_get\_node\_\_(i + 1) - self.\_\_calc\_c\_\_(i)   def \_\_calc\_b\_\_(self, i):  return self.\_\_get\_node\_\_(i) - self.\_\_calc\_d\_\_(i)   def \_\_calc\_c\_\_(self, i):  term1 = (3 + self.\_\_get\_q\_\_(i)) \* (self.\_\_get\_node\_\_(i + 1) - self.\_\_get\_node\_\_(i))  term2 = self.\_\_calc\_dist\_\_(i) \* self.\_\_get\_m\_\_(i)  term3 = (2 + self.\_\_get\_q\_\_(i)) \* self.\_\_calc\_dist\_\_(i) \* self.\_\_get\_m\_\_(i + 1)  term4 = (2 + self.\_\_get\_q\_\_(i)) \* (2 + self.\_\_get\_p\_\_(i)) - 1  return (-term1 + term2 + term3) / term4   def \_\_calc\_c\_term\_\_(self, i):  term1 = self.\_\_calc\_lmbd\_\_(i) \* self.\_\_calc\_p\_\_(i - 1) \* (3 + self.\_\_get\_q\_\_(i - 1))  term2 = (self.\_\_get\_node\_\_(i) - self.\_\_get\_node\_\_(i - 1)) / self.\_\_calc\_dist\_\_(i - 1)  term3 = self.\_\_calc\_mu\_\_(i) \* self.\_\_calc\_q\_\_(i) \* (3 + self.\_\_get\_p\_\_(i))  term4 = (self.\_\_get\_node\_\_(i + 1) - self.\_\_get\_node\_\_(i)) / self.\_\_calc\_dist\_\_(i)  return term1 \* term2 + term3 \* term4   def \_\_calc\_d\_\_(self, i):  term1 = (3 + self.\_\_get\_p\_\_(i)) \* (self.\_\_get\_node\_\_(i + 1) - self.\_\_get\_node\_\_(i))  term2 = self.\_\_calc\_dist\_\_(i) \* self.\_\_get\_m\_\_(i + 1)  term3 = (2 + self.\_\_get\_p\_\_(i)) \* self.\_\_calc\_dist\_\_(i) \* self.\_\_get\_m\_\_(i)  term4 = (2 + self.\_\_get\_q\_\_(i)) \* (2 + self.\_\_get\_p\_\_(i)) - 1  return (term1 - term2 - term3) / term4   def \_\_calc\_dist\_\_(self, i):  return np.sqrt(np.power(self.\_\_get\_x\_\_(i + 1) - self.\_\_get\_x\_\_(i), 2) +  np.power(self.\_\_get\_y\_\_(i + 1) - self.\_\_get\_y\_\_(i), 2))   def \_\_calc\_i\_\_(self, s):  for i in range(0, len(self.s) - 1):  s1 = self.s[i]  s2 = self.s[i + 1]  if s1 <= s <= s2:  # print(i)  return i  assert False, "ERROR: 'i' not found!"   def \_\_calc\_lmbd\_\_(self, i):  return self.\_\_calc\_dist\_\_(i) / (self.\_\_calc\_dist\_\_(i - 1) + self.\_\_calc\_dist\_\_(i))   def \_\_calc\_mu\_\_(self, i):  return 1 - self.\_\_calc\_lmbd\_\_(i)   def \_\_calc\_p\_\_(self, i):  term1 = 3 + 3 \* self.\_\_get\_p\_\_(i) + math.pow(self.\_\_get\_p\_\_(i), 2)  term2 = (2 + self.\_\_get\_q\_\_(i)) \* (2 + self.\_\_get\_p\_\_(i)) - 1  return term1 / term2   def \_\_calc\_q\_\_(self, i):  term1 = 3 + 3 \* self.\_\_get\_q\_\_(i) + math.pow(self.\_\_get\_q\_\_(i), 2)  term2 = (2 + self.\_\_get\_q\_\_(i)) \* (2 + self.\_\_get\_p\_\_(i)) - 1  return term1 / term2   def \_\_get\_m\_\_(self, i):  return self.m\_res[i]   def \_\_get\_p\_\_(self, i):  return self.p[i]   def \_\_get\_q\_\_(self, i):  return self.q[i]   def \_\_get\_node\_\_(self, i):  \_, x\_, y\_ = self.points[i]  return x\_ if self.is\_x else y\_   def \_\_get\_x\_\_(self, i):  \_, x\_, \_ = self.points[i]  return x\_   def \_\_get\_y\_\_(self, i):  \_, \_, y\_ = self.points[i]  return y\_   def \_\_s\_\_(self, s, is\_x: bool):  self.is\_x = is\_x  i = self.\_\_calc\_i\_\_(s)  t = (s - self.s[i]) / self.\_\_calc\_dist\_\_(i)  term1 = self.\_\_calc\_a\_\_(i) \* t  term2 = self.\_\_calc\_b\_\_(i) \* (1 - t)  term3 = (self.\_\_calc\_c\_\_(i) \* np.power(t, 3)) / (1 + self.\_\_get\_p\_\_(i) \* (1 - t))  term4 = (self.\_\_calc\_d\_\_(i) \* np.power(1 - t, 3)) / (1 + self.\_\_get\_q\_\_(i) \* t)  return term1 + term2 + term3 + term4   def main():  n = 100  t1 = -3 \* np.pi  t2 = 3 \* np.pi  a\_param = 0.01  b\_param = 0.15  t = np.linspace(t1, t2, num=n + 1)  x = func\_x(a\_param, b\_param, t)  y = func\_y(a\_param, b\_param, t)   k = np.linspace(-0.9, 1000, 100)  err = []  for k\_ in k:  p = [k\_ for i in range(0, n)]  q = [k\_ for i in range(0, n)]   solver = Lab3(p, q, list(zip(t, x, y)))   n2 = 1000  tt = np.linspace(t1, t2, num=n2)  f\_x\_dots = np.array([func\_x(a\_param, b\_param, t\_) for t\_ in tt])  f\_y\_dots = np.array([func\_y(a\_param, b\_param, t\_) for t\_ in tt])  s\_x\_dots = np.array([solver.s\_x(s\_) for s\_ in np.linspace(0, solver.s[len(solver.s) - 1], num=n2)])  s\_y\_dots = np.array([solver.s\_y(s\_) for s\_ in np.linspace(0, solver.s[len(solver.s) - 1], num=n2)])  sum = 0  for i in range(n2):  sum += math.sqrt(math.pow(f\_x\_dots[i] - s\_x\_dots[i], 2) + math.pow(f\_y\_dots[i] - s\_y\_dots[i], 2))  sum = sum / n2  err.append(sum)   plt.plot(f\_x\_dots, f\_y\_dots, label="func", lw=3)  plt.plot(s\_x\_dots, s\_y\_dots, label="spline")  plt.legend()  plt.show()   fig, ax = plt.subplots()  ax.grid()  ax.plot(k, err, label="error")  ax.legend()  ax.set\_xlabel('p, q')  ax.set\_ylabel('error')  plt.show()  return 0   if \_\_name\_\_ == "\_\_main\_\_":  main() |

## Результат

Для упрощения расчётов примем:

|  |  |  |
| --- | --- | --- |
|  | (8) |  |

|  |  |  |
| --- | --- | --- |
| Логарифмическая спираль | | |
| Параметры |  |  |
| График |  |  |

|  |  |  |
| --- | --- | --- |
| Параметр | 1 | 5 |
| График |  |  |
| Параметр | 10 | 100 |
| График |  |  |
| Параметр | 1000 | 10000 |
| График |  |  |

## Вывод

Был построен интерполяционный параметрический рациональный сплайн с четвертым краевым условием для функции «Логарифмическая спираль». Отрицательные коэффициенты ухудшают результаты интерполяции. При значении коэффициентов, равных нулю, получим интерполяцию кубическим сплайном. Для наилучшего результата интерполирования этой функции рекомендуется выбирать большие положительные коэффициенты .