Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления (ИУ)

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии (ИУ7)

Лабораторная работа №3

Тема: Сплайн интерполяция

Студент: Миронов Г.А.

Группа: ИУ7-43Б

Оценка (баллы): _________

Преподаватель: Градов В.М.

Москва.
2020 г.

Задание

1. Задана матрица значений функции вида X, F(X). С помощью интерполяции, используя метод полинома сплайнов, найти приближенное значение функции от введенного X
2. Сравнить результат интерполяции кубическим сплайном и полиномом Ньютона 3-ей степени

Входные данные

1. Таблица координат
2. Координата точки по оси абсцисс
3. Степень искомого полинома

Выходные данные

1. Значение функции в точке X, найденное методом кубического сплайна

Описание алгоритма

Кубический сплайн - это кривая, состоящая из состыкованных полиномов третьей степени $(y^{(IV)}(x) = 0)$. В точках стыковки значения и производные двух соседних полиномов равны.

Интерполяционный полином на участке между каждой парой соседних точек имеет вид:

$$\phi(x) = a_i + b_i(x - x_{i-1}) + c_i(x - x_{i-1})^2 + d_i(x - x_{i-1})^3$$

В узлах значения многочлена и интерполируемой функции совпадают:

$$f(x_{i-1}) = y_{i-1}$$

$$f(x_i) = y_i = a_i + b_i(x - x_{i-1}) + c_i(x - x_{i-1})^2 + d_i(x - x_{i-1})^3$$

Формулы для определения коэффициентов:

$$a_i = y_{i-1}$$

$$d_i = \frac{c_{i+1} - c_i}{3h_i}, h_i = x_i - x_{i-1}$$

$$b_i = \frac{(y_i - y_{i-1})}{h_i} - \frac{h_i(c_{i+1} + 2c_i)}{3}$$

Система уравнений для определения коэффициента c_i:

$$\begin{cases} c_1 = 0 \\ h_{i-1}c_{i-1} + 2(h_{i-1} + h_i)c_i + h_i c_{i+1} = 3(\frac{y_i - y_{i-1}}{h_i} - \frac{y_{i-1} - y_{i-1}}{h_{i-2}}) \\ c_{N+1} = 0 \end{cases}$$

Матрица этой системы трёхдиагональна. Такая система решается *методом прогонки*.

Алгоритм метода прогонки

Прямой ход: при заданных начальных значениях прогоночных коэффициентов ξ_i и η_i определяются все прогоночные коэффициенты:

$$\xi_{i+1} = \frac{D_i}{B_i - A_i \xi_i}$$

$$\eta_{i+1} = \frac{F_i + A_i \eta_i}{B_i - A_i \xi_i}$$

Обратный ход: при известном c_N определяются все $c_i, i = \overline{1, N}$

$$c_1 = 0$$

$$c_1 = \xi_2 c_2 + \eta_2$$

$$\begin{cases} \xi_2 = 0 \\ \eta_2 = 0 \end{cases}$$

Имея граничные условия, находим начальные коэффициенты (прямой ход).

Нахождение c_i (обратный ход):

$$c_i = \xi_{i+1} c_{i+1} + \eta_{i+1}, c_{N+1} = 0, c_N = \eta_{i+1}$$

Входные данные

Функция вида y = x * x

X	Y
0.00	0.00
1.00	1.00
2.00	4.00
3.00	9.00
4.00	16.00
5.00	25.00
6.00	36.00
7.00	49.00
8.00	64.00
9.00	81.00

Результаты

X	Кубический сплайн	Полином Ньютона 3 степени
0.5	0.3415	0.25
5.5	30.2481	30.25

Код программы

Листинг 1. `utils.py`

```
from spline import Dot


def read_dots(fname: str) -> list[Dot]:
   dots = []

   with open(fname) as fin:
       for line in fin.readlines():
           dots += [Dot(*list(map(float, line.split()[:2])))]

   return dots


def print_dots(dots: list[Dot]) -> None:
   print("{:^8} {:^8}".format("X", "Y"))
   for i in dots:
       print("{:<8.2f} {:<8.2f}".format(i.x, i.y))


def read_x() -> float:
   return float(input())
```

Листинг 2. spline.py

```
from __future__ import annotations


class Dot(object):
   x: float
   y: float

   def __init__(self, _x: float, _y: float) -> None:
       super().__init__()

       self.x, self.y = _x, _y

   def __lt__(self, other: Dot):
       return self.x < other.x


class Spline(object):
   dots: list[Dot]

   def __init__(self, _dots: list[Dot]) -> None:
       super().__init__()
       self.dots = _dots

   def get_pos(self, d: Dot) -> int:
       i = 1

       while i < len(self.dots) and self.dots[i].x < d.x:
           i += 1

       return i - 1

   def solve(self, x: float) -> Dot:
       arg_x = [d.x for d in self.dots]
       arg_y = [d.y for d in self.dots]

       a = arg_y[:-1]

       c = [0] * (len(self.dots) - 1)

       # Начальные известные значения коэффициентов
```

```python
ksi_coef, eta_coef = [0, 0], [0, 0]

# Прямой проход
for i in range(2, len(self.dots)):
    xhi, xhi_1 = arg_x[i] - arg_x[i - 1], arg_x[i - 1] - arg_x[i - 2]
    yhi, yhi_1 = arg_y[i] - arg_y[i - 1], arg_y[i - 1] - arg_y[i - 2]

    fi = 3 * (yhi / xhi - yhi_1 / xhi_1)

    # Вычисление прогоночных коэффициентов
    ksi_coef.append(-xhi /
                    (xhi_1 * ksi_coef[i - 1] + 2 * (xhi_1 + xhi)))
    eta_coef.append(
        (fi - xhi_1 * eta_coef[i - 1]) / (
                xhi_1 * ksi_coef[i - 1] + 2 * (xhi_1 + xhi)))

c[len(self.dots) - 2] = eta_coef[-1]

# Обратный проход
for i in range(len(self.dots) - 2, 0, -1):
    c[i - 1] = ksi_coef[i] * c[i] + eta_coef[i]

b, d = [], []
for i in range(1, len(self.dots) - 1):
    xhi = arg_x[i] - arg_x[i - 1]
    yhi = arg_y[i] - arg_y[i - 1]
    b.append(yhi / xhi - (xhi * (c[i] + 2 * c[i - 1])) / 3)
    d.append((c[i] - c[i - 1]) / (3 * xhi))

b.append((arg_y[-1] - arg_y[-2]) / (arg_x[-1] - arg_x[-2]) -
         ((arg_x[-1] - arg_x[-2]) * 2 * c[-1]) / 3)
d.append(-c[len(self.dots) - 2] / (3 * (arg_x[-1] - arg_x[-2])))

pos = get_pos(self.dots, Dot(x, 0))

res = a[pos] + \
      b[pos] * (x - self.dots[pos].x) + \
      c[pos] * (x - self.dots[pos].x) ** 2 + \
      d[pos] * (x - self.dots[pos].x) ** 3

return Dot(x, res)
```

Листинг 3. `polynomial.py`

```python
from __future__ import annotations
from math import prod


class Polynomial(object):
   terms: list[callable[float]:float]

   def __init__(self, terms: list[callable[float]:float]):
       self.terms = terms

   def __call__(self, arg: float) -> float:
       return sum([term(arg) for term in self.terms])


class NewtonPolynomial(Polynomial):

   @staticmethod
   def build(points: list[list[float]], arg: float, n: int) -> NewtonPolynomial:
       table = NewtonPolynomial._make_table(points, arg, n)

       return NewtonPolynomial(
           [lambda x:table[1][0]] +
           [NewtonPolynomial._term(table[i][0],  table[0][:i - 1])
            for i in range(2, len(table))]
       )

   @staticmethod
   def _term(va: float, vl: list[float]) -> callable:
       return lambda x: va * prod(map(lambda a: (x - a), vl))

   @staticmethod
   def _make_table(points: list[list[float]], arg: float, n: int) -> list[list[float]]:
       base = sorted(sorted(points, key=lambda p: abs(p[0] - arg))[:n+1])

       t = [[0.0 for i in range(len(base))] for j in range(n + 2)]

       for i in range(len(t[0])):
           t[0][i], t[1][i] = base[i][0], base[i][1]

       for i in range(2, len(t)):
           for j in range(len(base) - i + 1):
               t[i][j] = (t[i - 1][j] - t[i - 1][j + 1]) / \
```

```
            (t[0][j] - t[0][j + i - 1])

    return t
```

Листинг 3. `main.py`

```python
from sys import argv

from polynomial import NewtonPolynomial
from spline import Spline
from utils import *


def main() -> None:
   dots = read_dots(argv[1])

   print("Table loaded from file\n")
   print_dots(dots)

   print("\nEnter X value: ")
   x = read_x()

   res = Spline(dots).solve(x)
   poly = Dot(x, NewtonPolynomial.build([[d.x, d.y] for d in dots], x, 3)(x))

   print_res(res, poly)


if __name__ == "__main__":
   main()
```

Контрольные вопросы

1. Получить выражения для коэффициентов кубического сплайна, построенного на двух точках.

Пусть даны x_0, x_1, y_0, y_1 - аргументы и значения функции в двух точках

Кубический сплайн:

$$\psi(x) = a_i + b_i(x - x_{i-1}) + c_i(x - x_{i-1})^2 + d_i(x - x_{i-1})^3, x_{i-1} \le x \le x_i, 1 \le i \le N$$

N = 1 т.к. даны только две точки.
Неизвестные: a_1, b_1, c_1, d_1.

$$\psi(x_0) = y_0 = a_1$$

$$\psi(x_1) = y_1 = a_1 + b_1(x_1 - x_0) + c_1(x_1 - x_0)^2 + d_1(x_1 - x_0)^3 \quad (1)$$

$$\psi'(x) = b_1 + 2c_1(x - x_0) + 3d_1(x - x_0)^2$$

$$\psi''(x) = 2c_1 + 6d_1(x - x_0)$$

На концах участка интерполирования вторую производную положим равной нулю:

$$\psi''(x_0) = 0 = 2c_1 \Rightarrow c_1 = 0$$

$$\psi''(x_1) = 0 = 2c_1 + 6d_1(x_1 - x_0) \Rightarrow d_1 = \frac{-c_1}{3(x_1 - x_0)} = 0$$

Подставляя вычисленные значения a_1, c_1, d_1 в (1) находим b_1:

$$y_1 = y_0 + b_1(x_1 - x_0) + 0 \cdot (x_1 - x_0)^2 + 0 \cdot (x_1 - x_0)^3 \Rightarrow b_1 = \frac{y_1 - y_0}{x_1 - x_0}$$

Имеем: $a_1 = y_0$; $b_1 = \frac{y_1 - y_0}{x_1 - x_0}$; $c_1 = d_1 = 0$

2. Выписать все условия для определения коэффициентов сплайна, построенного на 3-х точках

   - Совпадение значения сплайна и интерполируемой функции во всех 3 точках
   - Совпадение значения 1ых и 2ых производных во внутреннем узле между “левой” и “правой” частями
   - Из-за нехватки условий, можно, например, положить величину второй производной равной нулю на краях участка интерполирования

   Запишем в виде системы

   $\psi(x_0) = y_0 = a_1$

   $\psi(x_1) = y_1 = a_1 + b_1(x_1 - x_0) + c_1(x_1 - x_0)^2 + d_1(x_1 - x_0)^3 = a_2$ - здесь 2 уравнения

   $\psi(x_2) = y_2 = a_2 + b_2(x_2 - x_1) + c_2(x_2 - x_1)^2 + d_2(x_2 - x_1)^3$

   $\psi'(x_1) = b_1 + 2c_1(x_1 - x_0) + 3d_1(x_1 - x_0)^2 = b_2$

   $\psi''(x_1) = 2c_1 + 6d_1(x_1 - x_0) = c_2$

   $\psi''(x_0) = 0$

   $\psi''(x_2) = 0$

3. Определить начальные значения прогоночных коэффициентов, если принять, что для коэффициентов сплайна справедливо C1=C2

   $c_{i-1} = \xi_i c_i + \eta_i \Rightarrow c_1 = \xi_2 c_2 + \eta_2 \Rightarrow \xi_2 = 1;\ \eta = 0$

4. Написать формулу для определения последнего коэффициента сплайна CN , чтобы можно было выполнить обратный ход метода прогонки, если в качестве граничного условия задано $kC_{N-1}+mC_N=p$, где k,m и p - заданные числа.

   $c_{N-1} = \xi_N c_N + \eta_N$

   Т.к. $kc_{N-1} + mc_N = p$, то:

   $$c_N = \frac{p - k\eta_N}{k\xi_N + m}$$