Министерство науки и высшего образования 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»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа № 3

Тема: Построение и программная реализация алгоритма сплайн-интерполяции табличных функций.

Студент: Хамзина Р.Р.

Группа: ИУ7-43Б

Оценка (баллы): _____

Преподаватель: Градов В.М.

Москва
2021 г

Цель работы: получение навыков владения методами интерполяции таблично заданных функций с помощью кубических сплайнов.

1 Исходные данные

1. Таблица функции с количеством узлов N.

   Для отладки:

x	y
0	0
1	1
2	4
3	9
4	16
5	25
6	36
7	49
8	64
9	81
10	100

2. Значение аргумента x.

   Для отладки: x = 0.5 - в первом интервале, x = 5.5 в середине таблицы.

2 Код программы

```
from dataclasses import dataclass

from newton_interpolation import newton_interpolation

@dataclass
class Constants:
    count_x = 11


def print_results(argument, spline_result, newton_result):
    """
        Печать результатов.
    """
    print("")
    print("Значение аргумента: ", argument)
    print("Результат интерполяции кубическим сплайном: {:.5f}".format(spline_result))
    print("Результат интерполяции полиномом Ньютона (3 степень): {:.5f}".format(newton
_result))
    print("")


def get_argument_position(arguments, argument):
    """
       Поиск позиции правой границы интервала,
       в который входит искомый аргумент.
    """

    for i in range(len(arguments)):
        if arguments[i] >= argument:
            return i


def get_value_polynom(value_table, factors, argument):
    """
       Получить значение полинома.
    """

    position = get_argument_position(value_table[0], argument) - 1
    step = argument - value_table[0][position]

    polynom = factors[0][position] + factors[1][position] * step + \
              factors[2][position] * step ** 2 + \
              factors[3][position] * step ** 2 * step
    return polynom


def get_factors_a(values):
    """
        Получить коэффициенты a.
    """

    factors_a = []

    for i in range(1, Constants.count_x):
        factors_a.append(values[i - 1])

    return factors_a


def get_factors_b(value_table, factors_c):
    """
        Получить коэффициенты b.
    """

    factors_b = []

    for i in range(1, Constants.count_x - 2):
        now_step = value_table[0][i] - value_table[0][i - 1]
        now_values_diff = value_table[1][i] - value_table[1][i - 1]

        now_b = now_values_diff / now_step - \
                now_step * (factors_c[i] + 2 * factors_c[i - 1]) / 3
        factors_b.append(now_b)
```

```
    # b_n

    step_n = value_table[0][Constants.count_x - 1] - \
             value_table[0][Constants.count_x - 2]
    values_diff_n = value_table[1][Constants.count_x - 1] - \
                    value_table[1][Constants.count_x - 2]

    b_n = values_diff_n / step_n - \
          step_n * 2 * factors_c[Constants.count_x - 2] / 3
    factors_b.append(b_n)

    return factors_b


def get_factors_e(arguments):
    """
        Получить коэффициенты e.
    """

    factors_e = [0]

    for i in range(2, Constants.count_x):
        now_step = arguments[i] - arguments[i - 1]
        prev_step = arguments[i - 1] - arguments[i - 2]

        now_e = -now_step / (prev_step * factors_e[i - 2] + \
                2 * (prev_step + now_step))
        factors_e.append(now_e)

    return factors_e


def get_factors_g(value_table, factors_e):
    """
        Получить коэффициенты g.
    """

    factors_g = [0]

    for i in range(2, Constants.count_x):
        now_step = value_table[0][i] - value_table[0][i - 1]
        prev_step = value_table[0][i - 1] - value_table[0][i - 2]

        now_values_diff = value_table[1][i] - value_table[1][i - 1]
        prev_values_diff = value_table[1][i - 1] - value_table[1][i - 2]
        now_f = 3 * (now_values_diff / now_step - prev_values_diff / prev_step)

        now_g = (now_f - now_step * factors_g[i - 2]) / \
                (now_step * factors_e[i - 2] + 2 * (prev_step + now_step))
        factors_g.append(now_g)

    return factors_g


def straight_run(value_table):
    """
        Прямой ход.
    """

    factors_e = get_factors_e(value_table[0])
    factors_g = get_factors_g(value_table, factors_e)

    return factors_e, factors_g


def reverse(factors_e, factors_g):
    """
        Обратный ход.
    """

    factors_c = [0, 0]
    now_end_diff = 0

    for i in range(Constants.count_x - 2, 0, -1):
        now_c = factors_e[i] * factors_c[len(factors_c) - 1 - now_end_diff] + factors_
g[i]
        now_end_diff += 1
        factors_c.insert(1, now_c)

    return factors_c
```

```python
def get_factors_c(value_table):
    """
        Получить коэффициенты c.
    """

    factors_e, factors_g = straight_run(value_table)
    factors_c = reverse(factors_e, factors_g)

    return factors_c


def get_factors_d(arguments, factors_c):
    """
       Получить коэффициенты d.
    """

    factors_d = []

    for i in range(1, Constants.count_x - 2):
        now_step = arguments[i] - arguments[i - 1]
        now_d = (factors_c[i] - factors_c[i - 1]) / (3 * now_step)
        factors_d.append(now_d)

    # d_n

    step_n = arguments[Constants.count_x - 1] - \
             arguments[Constants.count_x - 2]
    d_n = -factors_c[Constants.count_x - 2] / (3 * step_n)
    factors_d.append(d_n)

    return factors_d


def spline_interpolation(value_table, argument):
    """
       Интерполяция кубическим сплайном.
    """

    factors = []

    factors_a = get_factors_a(value_table[1])
    factors.append(factors_a)
    factors_c = get_factors_c(value_table)
    factors_b = get_factors_b(value_table, factors_c)
    factors.append(factors_b)
    factors.append(factors_c)
    factors_d = get_factors_d(value_table[0], factors_c)
    factors.append(factors_d)

    spline_polynom = get_value_polynom(value_table, factors, argument)

    return spline_polynom
```

```python
def form_value_table():
    """
        Формирование таблицы значений
        y = x^2, x в диапазоне [0...10].
    """

    value_table = [[], []]

    for x in range(Constants.count_x):
        value_table[0].append(x)
        value_table[1].append(x ** 2)

    return value_table
```

```
if __name__ == "__main__":
    """
        Сравнение результатов интерполяции
        - кубическим сплайном;
        - полиномом Ньютона 3-й степени.
    """

    value_table = form_value_table()

    argument = 0.5
    spline_result = spline_interpolation(value_table, argument)
    newton_result = newton_interpolation(value_table, Constants.count_x, 3, argument)
    print_results(argument, spline_result, newton_result)


    argument = 5.5
    spline_result = spline_interpolation(value_table, argument)
    newton_result = newton_interpolation(value_table, Constants.count_x, 3, argument)
    print_results(argument, spline_result, newton_result)
```

3 Результаты работы

1) Необходимо найти полином вида:

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

$$x_{i-1} \le x \le x_i, 0 \le i \le N.$$

1. Найдем коэффициенты a_i: $a_i = y_{i-1}$, i в диапазоне [1...N].
2. Найдем коэффициенты c_i для i в диапазоне [2...N] методом прогонки $c_{i-1} = \xi_i c_i + \eta_i$. Найдем прогоночные коэффициенты.

а) Прямой ход: $c_1 = 0$, значит $\xi_2 = 0, \eta_2 = 0$. Для i в диапазоне [2...N] прогоночные коэффициенты находятся так:

$$h_i = x_i - x_{i-1}$$

$$f_i = 3\left(\frac{y_i - y_{i-1}}{h_i} - \frac{y_{i-1} - y_{i-2}}{h_{i-1}}\right)$$

$$\xi_{i+1} = -\frac{h_i}{h_{i-1}\xi_i + 2(h_{i-1} + h_i)}, \eta_{i+1} = \frac{f_i - h_{i-1}\eta_i}{h_{i-1}\xi_i + 2(h_{i-1} + h_i)}.$$

б) Обратный ход: $c_{N+1} = 0$. Зная прогоночные коэффициенты, найдем c_i:

$$c_i = \xi_{i+1} c_{i+1} + \eta_{i+1}$$

3. Найдем коэффициенты b_i для i в диапазоне [1...N-1]:

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

$b_N = (y_N - y_{N-1}) / h_N - h_N 2c_N / 3$

4. Найдем коэффициенты d_i для i в диапазоне [1...N-1]:

$d_i = (c_{i+1} - c_i) / 3h_i$

$d_N = -c_N / 3h_N$

5. Подставим найденные коэффициенты и значение аргумента в формулу из пункта 1).

Полученные результаты:

x	0.5	5.5
y(x)	0.34151	30.25035

2) Сравним результаты интерполяции кубическим сплайном и полиномом Ньютона третьей степени.

x	Кубический сплайн	Полином Ньютона 3-ей степени
0.5	0.34151	0.25
5.5	30.25035	30.25

4 Вопросы при защите лабораторной работы

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

Начальные условия:

x	y
x_0	y_0
x_1	y_1

Коэффициенты:

1. $a = [y_0]$
2. $c_1 = 0$, поэтому $e_2 = 0$, $g_2 = 0$, $c_{N+1=3} = 0$, поэтому $e_3 = 0$, $g_3 = 0$.

   Тогда $c_2 = e_3c_3 + g_3 = 0$

   $c = [0]$
3. $h_1 = x_1 - x_0$

   $b_1 = (y_1 - y_0) / h_1 - h_1 2c_1 / 3 = (y_1 - y_0) / (x_1 - x_0)$

   $b = [(y_1 - y_0) / (x_1 - x_0)]$
4. $h_1 = x_1 - x_0$

   $d_1 = -c_1 / 3h_1 = 0$

   $d = [0]$

Получившиеся коэффициенты: $a = y_0$, $b = (y_1 - y_0) / (x_1 - x_0)$, $c = 0$, $d = 0$

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

Начальные условия:

x	y
x_0	y_0
x_1	y_1
x_2	y_2

В узлах значения многочлена и интерполируемой функции совпадают:

$f_0(x_0) = y_0$,

$f_1(x_1) = y_1$,

$f_0(x_1) = y_1$,

$f_1(x_2) = y_2$.

Тогда:

$a_0 = y_0$, $a_1 = y_1$,

$a_0 + b_0*(x_1 - x_0) + c_0*(x_1 - x_0)^2 + d_0*(x_1 - x_0)^3 = y_1$,

$a_1 + b_1*(x_2 - x_1) + c_1*(x_2 - x_1)^2 + d_1*(x_2 - x_1)^3 = y_2$,

Число уравнений меньше числа неизвестных в два раза. Недостающие уравнения получим, приравняв во внутренних узлах первые и вторые производные,вычисляемые по коэффициентам на соседних участках:

$f_0'(x_1) = f_1'(x_1)$

$f_0''(x_1) = f_1''(x_1)$

$f_0''(x_0) = 0$

$f_1'(x_2) = 0$

Тогда:

$b_0 + 2*c_0*(x_1 - x_0) + 3*d_0*(x_1 - x_0)^2 = b_1$

$c_0 + 3*d_0*(x_1 - x_0) = c_1$

Ответ: $a_0 = y_0$,

$a_1 = y_1$,

$a_0 + b_0*(x_1 - x_0) + c_0*(x_1 - x_0)^2 + d_0*(x_1 - x_0)^3 = y_1$,

$a_1 + b_1*(x_2 - x_1) + c_1*(x_2 - x_1)^2 + d_1*(x_2 - x_1)^3 = y_2$,

$b_0 + 2*c_0*(x_1 - x_0) + 3*d_0*(x_1 - x_0)^2 = b_1$,

$c_0 + 3*d_0*(x_1 - x_0) = c_1$,

$f_0''(x_0) = 0$

$f_1'(x_2) = 0$

3. Определить начальные значения прогоночных коэффициентов, если принять, что для коэффициентов сплайна справедливо $c_1 = c_2$.

$c_i = e_{i+1}c_{i+1} + g_{i+1}$

Тогда:

$c_1 = e_2c_2+g_2 = c_2$. Значит, $e_2 = 1$, $g_2 = 0$.

Ответ: $e_2 = 1$, $g_2 = 0$.

4. Написать формулу для определения последнего коэффициента сплайна c_N, 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.

$c_{N-1} = e_Nc_N+g_N$

Из условия: $c_N = (p - kc_{N-1}) / m$, $c_N = (p - ke_Nc_N - kg_N) / m$

Тогда: $c_N = (p - kg_N) / (m + ke_N)$

Ответ: $c_N = (p - kg_N) / (m + ke_N)$