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: Ковалец Кирилл

Группа: ИУ7-43Б

Оценка (баллы): _____

Преподаватель: Градов Владимир Михайлович

Москва
2021 г

Задание

Цель работы.

Получение навыков владения методами интерполяции таблично заданных функций с помощью кубических сплайнов.

Исходные данные.

1. Таблица функции с количеством узлов N. Задать с помощью формулы $y = x^2$ в диапазоне [0..10] с шагом 1.
2. Значение аргумента x в первом интервале, например, при x=0.5 и в середине таблицы, например, при x= 5.5. Сравнить с точным значением.

Результат работы программы.

1. Значения y(x).
2. Сравнить результаты интерполяции кубическим сплайном и полиномом Ньютона 3-ей степени.

Теоретическая часть

Для кубического сплайна значение y вычисляется следующим образом

$$\psi(x_i) = y_i = a_i + b_i h_i + c_i h_i^2 + d_i h_i^3,$$

$$h_i = x_i - x_{i-1}, 1 \le i \le N.$$

Нахождение коэффициентов

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

где ξ_{i+1}, η_{i+1} - некоторые, не известные пока прогоночные коэффициенты;

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

В этих формулах введено обозначение

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

$$b_i = \frac{y_i - y_{i-1}}{h_i} - h_i \frac{c_{i+1} - 2c_i}{3}, \quad 1 \le i \le N-1.$$

$$b_N = \frac{y_N - y_{N-1}}{h_N} - h_N \frac{2c_N}{3}$$

$$d_i = \frac{c_{i+1} - c_i}{3h_i}, \quad 1 \le i \le N-1,$$

$$d_N = -\frac{c_N}{3h_N}.$$

Код программы

main.py

```
from newton_polynom import newton_polynomial
from spline import spline

def read_file(name_file):
    try:
        with open(name_file, "r") as f:
            table = [list(map(int, string.split())) for string in list(f)]
        return table

    except:
        print("Ошибка чтения файла!\n")
        return []

def read_data(size_table):
    try:
        x = float(input("Введите значение аргумента, для которого выполняется
интерполяция: "))
        return 0, x

    except:
        print("Ошибка ввода данных!\n")
        return 1, 0

def print_table(table):
    print("\n{:^5}{:^11}\n".format("x", "y"))

    for i in range(len(table)):
        for j in range(len(table[i])):
            print("%-8.2f" %(table[i][j]), end = '')
        print()

    print()
```

```
def main():
    name_file = "../data.txt"

    table = read_file(name_file)
    if (table == []):
        return

    table.sort(key = lambda array: array[0])
    print_table(table)

    r, x = read_data(len(table))
    if (r):
        return

    spl = spline(table, x)
    np = newton_polynomial(table, 3, x)

    print("\nSpline = %.6f" %(spl))
    print("\nNewton_p = %.6f\n" %(np))

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

newton_polynom.py

```
EPS = 1e-6

def search_index(table, x, n):
    index = 0

    for i in table:
        if (i[0] > x):
            break
        index += 1
```

```
    if index >= len(table) - n:
        return len(table) - n - 1

    l_border = index
    r_border = index

    while (n > 0):
        if (r_border - index == index - l_border):
            if (l_border > 0):
                l_border -= 1
            else:
                r_border += 1
        else:
            if (r_border < len(table) - 1):
                r_border += 1
            else:
                l_border -= 1
        n -= 1

    return l_border

def divided_difference(x0, y0, x1, y1):
    if (abs(x0 - x1) > EPS):
        return (y0 - y1) / (x0 - x1)

def newton_polynomial(table, n, x):
    index = search_index(table, x, n)
    np = table[index][1]

    for i in range(n):
        for j in range(n - i):
            table[index + j][1] = divided_difference(
                table[index + j][0],         table[index + j][1],
                table[index + j + i + 1][0], table[index + j + 1][1])
```

```
        mult = 1
        for j in range(i + 1):
            mult *= (x - table[index + j][0])

        mult *= table[index][1]
        np += mult

    return np
```

spline.py

```
def calc_A(y_arr):
    return y_arr[:-1]

def calc_C(x_arr, y_arr):
    size = len(x_arr)

    C = [0] * (size - 1)

    ksi_arr = [0, 0]
    teta_arr = [0, 0]

    for i in range(2, size):
        h1 = x_arr[i]     - x_arr[i - 1]
        h2 = x_arr[i - 1] - x_arr[i - 2]

        fi = 3 * ((y_arr[i]     - y_arr[i - 1]) / h1 -
                  (y_arr[i - 1] - y_arr[i - 2]) / h2)

        ksi_cur = - h1                          / (h2 * ksi_arr[i - 1] + 2 *
(h2 + h1))
        teta_cur = (fi - h1 * teta_arr[i - 1]) / (h1 * ksi_arr[i - 1] + 2 *
(h2 + h1))
```

```
        ksi_arr.append(ksi_cur)
        teta_arr.append(teta_cur)

    C[-1] = teta_arr[-1]

    for i in range(size - 2, 0, -1):
        C[i - 1] = ksi_arr[i] * C[i] + teta_arr[i]

    return C

def calc_B_and_D(x_arr, y_arr, C):
    B = []
    D = []

    size = len(x_arr)

    for i in range(1, size - 1):
        h = x_arr[i] - x_arr[i - 1]

        B.append((y_arr[i] - y_arr[i - 1]) / h -  (h  * (C[i] + 2 * C[i -
1])) / 3)
        D.append((C[i] - C[i - 1]) / (3 * h))

    h = x_arr[-1] - x_arr[-2]

    B.append((y_arr[-1] - y_arr[-2]) / h - (h * 2 * C[-1]) / 3)
    D.append(-C[-1] / (3 * h))

    return B, D

def calculate_koefs_spline(x_arr, y_arr):
    A = calc_A(y_arr)
    C = calc_C(x_arr, y_arr)
    B, D = calc_B_and_D(x_arr, y_arr, C)
```

```
    return A, B, C, D

def fined_index(x_arr, x):
    size = len(x_arr)
    index = 1

    while (index < size and x_arr[index] < x):
        index += 1

    return index - 1

def count_polynom(x, x_arr, index, koefs):
    h = x - x_arr[index]
    y = 0

    for i in range(4):
        y += koefs[i][index] * (h ** i)

    return y

def spline(table, x):
    x_arr = [i[0] for i in table]
    y_arr = [i[1] for i in table]

    koefs = calculate_koefs_spline(x_arr, y_arr)

    index = fined_index(x_arr, x)

    y = count_polynom(x, x_arr, index, koefs)

    return y
```

Пример работы программы

```
  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
10.00   100.00

Введите значение аргумента, для которого выполняется интерполяция: 5.5

Spline = 30.250345

Newton_p = 30.250000
```

Результаты работы программы при данных значениях (Полином Ньютона 3-ей степени)

```
N = 3


-------------------------------------------
|   X    |     Y    |   Spline   |  Newton_p  |
-------------------------------------------
|  0.5   |   0.25   |  0.341506  |  0.250000  |
-------------------------------------------
|  5.5   |   30.25  |  30.250345 |  30.250000 |
-------------------------------------------
```

Вопросы при защите лабораторной работы.

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

Пусть заданы две точки: (x1, y1), (x2, y2). Тогда кубический полином будет иметь вид:

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

Коэффициент A = y1

Если сплайн строится по двум точкам и C(N) = 0, то C(N) = C = 0

Коэффициент D = - C(N) / 3h(N) = 0

коэффициент B = $\frac{y_N - y_{N-1}}{h_N} - h_N \frac{2c_N}{3}$ = (y2 - y1) / (x2 - x1)

В итоге функция выродилась в уравнение прямой.

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

Пусть дано три точки : (x1, y1), (x2, y2), (x3, y3)

Будет 8 коэффициентов:

1. Через значения узлов: w1(x1) = y1, w2(x2) = y2, w1(x2) = y2, w2(x3) = y3:
   - a1 = y1
   - a2 = y2
   - a1 + b1*(x2 - x1) + c1*(x2 - x1)^2 + d1*(x2 - x1)^3 = y2
   - a2 + b2*(x3 - x2) + c2*(x3 - x2)^2 + d2*(x3 - x2)^3 = y3
   -
2. Через производные:
   - w1’(x2) = w2’(x2) => b1+ 2*c1*(x2 - x1) + 3*d1*(x2 - x1)^2 = b2
   - w1’’(x2) = w2’’(x2) => c1 + 3*d1*(x2 - x1) = c2
   - w1’’(x1) = 0
   - w2’’(x3) = 0

3. Определить начальные значения прогоночных коэффициентов, если принять, что для коэффициентов сплайна справедливо $C_1=C_2$.

$$c_{i-1} = \xi_i c_i + \eta_i$$

Если C1 = C2, то: $\xi_2 = 1$, $\eta_2 = 0$

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(N) * C(N) + n(N)

k*C(N-1) + m*C(N) = p

C(N) = (p - k*n(N)) / (k*e(N) + m)