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»

Выполнил студент: __*Клименко Алексей Константинович*______________

фамилия, имя, отчество

Группа: _______________________*ИУ7-45Б*____________________________

Проверил, к.п.н.: ____

подпись, дата

Оценка __________________________________ Дата ________________

2020 г.

Цель работы

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

Исходные данные

1. Таблица функции с количеством узлов N. Задана с помощью формулы $y(x)=x^2$ в диапазоне [0..10].

2. Значение аргумента x в первом интервале: x = 0.5 и в середине таблицы: x = 5.5.

Код программы

```
from numpy import linspace

# исходные данные
N = 4
X = linspace(0, 10, N)
Y = X ** 2

# интерполяция полиномом Ньютона
def newton(X, Y, n, x):
  def fill_high_order_diffs(diffs, X, n):
    for order in range(1, n):
      for i in range(n - order):
        xx = tuple(X[i:i + 2 + order])
        diffs[xx] = (diffs[xx[:-1]] - diffs[xx[1:]]) / (xx[0] - xx[-1])

  def make_y(X, Y, diffs, n, x):
    y, x_prod = Y[X[0]], x - X[0]
    for i in range(n):
      xx = tuple(X[:i + 2])
      y += x_prod * diffs[xx]
      x_prod *= x - X[i + 1]
    return y

  Y = dict(zip(X, Y))
  X = sorted(X, key=lambda xi: abs(x - xi))
  X = sorted(X[:n + 1])

  diffs = dict()
  for i in range(n):
    diffs[tuple(X[i:i + 2])] = (Y[X[i]] - Y[X[i + 1]]) / (X[i] - X[i + 1])

  fill_high_order_diffs(diffs, X, n)
  return make_y(X, Y, diffs, n, x)

# кубический сплайн
def calc_qubic_spline_data(X, Y):
  n = len(X)
  A = Y[:-1]

  ksi, eta = [0, 0], [0, 0]
  for i in range(2, n): # прямой проход
    hi, him1 = X[i] - X[i - 1], X[i - 1] - X[i - 2]
    fi = 3 * ((Y[i] - Y[i - 1]) / hi - (Y[i - 1] - Y[i - 2]) / him1)
    ksi.append(- hi / (him1 * ksi[i - 1] + 2 * (him1 + hi)))
```

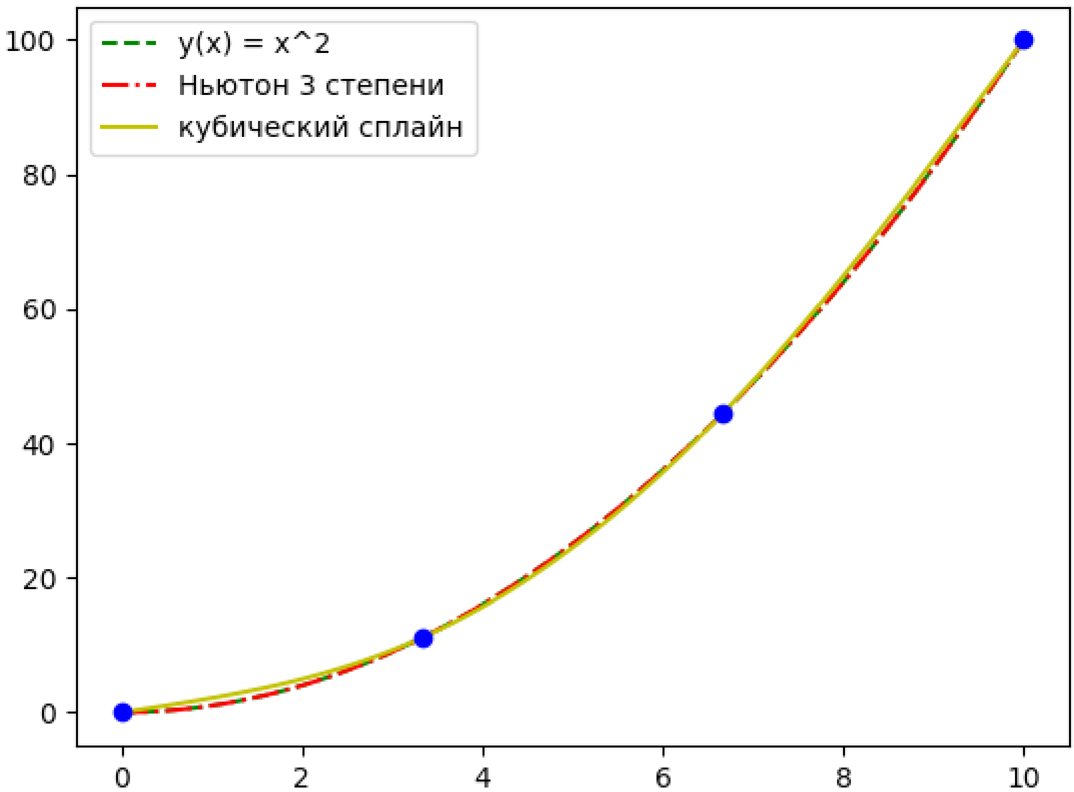
```
    eta.append((fi - him1 * eta[i - 1]) / (him1 * ksi[i - 1] + 2 * (him1 + hi)))

  C = [0] * (n - 1)
  C[n - 2] = eta[-1]
  for i in range(n - 2, 0, -1): # обратный проход
    C[i - 1] = ksi[i] * C[i] + eta[i]

  B, D = [], []
  for i in range(1, n - 1):
    hi = X[i] - X[i - 1]
    B.append((Y[i] - Y[i - 1]) / hi - hi / 3 * (C[i] + 2 * C[i - 1]))
    D.append((C[i] - C[i - 1]) / 3 / hi)
  B.append((Y[-1] - Y[-2]) / (X[-1] - X[-2]) - (X[-1] - X[-2]) / 3 * 2 * C[-1])
  D.append(- C[n - 2] / 3 / (X[-1] - X[-2]))

  return A, B, C, D

# использование коэф.-тов интерполяции для нахождения
# интерполированного значения табличной функции
def apply_interp_data(X, data, x):
  i = max([0] + list(filter(lambda i: X[i] < x, range(len(X)))))
  y, h = 0, x - X[i]
  for k, row in enumerate(data):
    y += row[i] * h ** k
  return y

x1, x2 = 0.5, 5.5           # точки интерполирования
y1, y2 = x1 ** 2, x2 ** 2   # актуальные значения

# коэффициенты интерполяции
data = calc_qubic_spline_data(X, Y)

# интерполированные значения
y1_i = apply_interp_data(X, data, x1)
y2_i = apply_interp_data(X, data, x2)

print("  x  |   y   |  y интерп.  ")
print("=====|=======|=============")
print(f" {x1} | {y1:5.2f} | {y1_i:7.2f}")
print(f" {x2} | {y2:5.2f} | {y2_i:7.2f}")

# вывод графика интерполированной функции
from matplotlib import pyplot as plt

X_ext = linspace(min(X), max(X), 100)
Y_ext = X_ext ** 2
Y_newton = [newton(X, Y, 3, x) for x in X_ext]
Y_interp = [apply_interp_data(X, data, x) for x in X_ext]

plt.plot(X_ext, Y_ext, 'g--', label="y(x) = x^2")
plt.plot(X_ext, Y_newton, 'r-.', label="Ньютон 3 степени")
plt.plot(X_ext, Y_interp, 'y-', label="кубический сплайн")

plt.plot(X, Y, 'bo')
plt.legend()
plt.show()
```

Результаты работы

1. Значения y(x) для выбранных x:

```
MSI_PC@MSI /cygdrive/d/Dev/ca-lab/lab_3
$ make
py main.py data.txt
  x   |    y    |  y интерп.
=====|=======|=============
 0.5 |   0.25 |     1.01
 5.5 | 30.25 |   29.74
```

2. Сравнение результатов интерполяции кубическим сплайном и полиномом Ньютона 3-ей степени:



Как видно, интерполяция полиномом Ньютона 3 степени идеально аппроксимирует заданную функцию, в отличае от интерполяции сплайнами.

Контрольные вопросы

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

Пусть заданные точки имеют координаты (x_1, y_1) и (x_2, y_2), $x_1 \neq x_2$.

Тогда для $\phi(x) = a + b(x-x_1) + c(x-x_1)^2 + d(x-x_1)^3$ имеем:

Условия для значений в узлах: $\phi(x_1) = y_1$, $\phi(x_2) = y_2$

$a = y_1$ **(1)**

$$a + b(x_2 - x_1) + c(x_2 - x_1)^2 + d(x_2 - x_1)^3 = y_2 \qquad \textbf{(2)}$$

Дополнительные условия для концов могут быть выбраны произвольно. Для простоты, приравняем вторые производные к нулю для получения функции с минимальной кривизной: $\phi''(x_1) = 0$, $\phi''(x_2) = 0$

$$2c = 0 \qquad \textbf{(3)}$$

$$2c + 6d(x_2 - x_1) = 0 \qquad \textbf{(4)}$$

<u>Данный набор из 4-х уравнений позволяет единственным образом определить все необходимые коэффициенты.</u>

Очевидно, что решение уравнений **(3, 4)** тривиально: **c = 0**, **d = 0**.

В следствие этого, можем подставить полученные значения коэффициентов c и d в уравнение **(2)** и привести его к следующему виду:

$$a + b(x_2 - x_1) = y_2$$

Откуда легко выразить недостающую неизвестную:

$$a = y_1 \,,\ \ b = \frac{y_2 - y_1}{x_2 - x_1} \,,\ \ c = 0 \,,\ \ d = 0$$

И записать итоговую формулу для сплайн-функции:

$$\phi(x) = y_1 + \frac{y_2 - y_1}{x_2 - x_1}(x - x_1)$$

Получили, что функция выродилась в уравнение прямой, проходящей через две точки.

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

Сплайн по трём точкам будет состоять из двух функций (ϕ_1 , ϕ_2) , поэтому всего неизвестных коэффициентов будет 8.

Условия для значений в узлах: $\phi_i(x_i) = y_i$, $\phi_i(x_{i+1}) = y_{i+1}$, $i = \overline{1,2}$

$$a_i = y_i \qquad \textbf{(1, 2)}$$

$$a_i + b_i(x_{i+1} - x_i) + c_i(x_{i+1} - x_i)^2 + d_i(x_{i+1} - x_i)^3 = y_{i+1} \qquad \textbf{(3, 4)}$$

Условие для значений 1-х производных: $\phi_1'(x_2) = \phi_2'(x_2)$

$$b_1 + 2c_1(x_2 - x_1) + 3d_1(x_2 - x_1)^2 = b_2 \qquad \textbf{(5)}$$

Условие для значений 2-х производных: $\phi_1''(x_2) = \phi_2''(x_2)$

$$c_1 + 3d_1(x_2 - x_1) = c_2 \qquad \textbf{(6)}$$

Еще два уравнения выбираются произвольно для задания поведения сплайна в области граничных точек. Выберем следующие:

$$\phi_1''(x_1) = 0 \quad \rightarrow \quad c_1 = 0 \qquad \textbf{(7)}$$

$$\phi_2''(x_3) = 0 \quad \rightarrow \quad c_2 + 3d_2(x_3 - x_2) = 0 \qquad \textbf{(8)}$$

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

Прогоночные коэффициенты удовлетворяют условиям:

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

Поэтому, для $c_1 = c_2$, начальные значения прогоночных коэффициентов будут равны соответственно:

$$\xi_2 = 1\ ,\ \ \eta_2 = 0$$

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

$$c_i = \xi_{i+1} c_{i+1} + \eta_{i+1} \quad \rightarrow \quad c_{N-1} = \xi_N c_N + \eta_N$$

$$k c_{N-1} + m c_N = p \quad \rightarrow \quad k(\xi_N c_N + \eta_N) + m c_N = p$$

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