|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **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 №4**

**тема:** «Построение и программная реализация алгоритма наилучшего среднеквадратичного приближения»

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 построения алгоритма метода наименьших квадратов с использованием полинома заданной степени при аппроксимации табличных функций с весами.

**Исходные данные**

1. Таблица функции с количеством узлов N. Задана с помощью формулы в диапазоне [0..10].

2. Значение аргумента x в первом интервале: х = 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()**

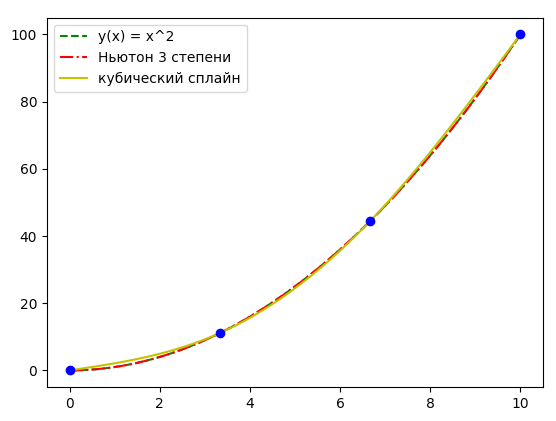
**plt.show()**

**Результаты работы**

1. Значения y(x) для выбранных x:



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 (x1, y1) и (x2, y2), x1 ≠ x2.

Тогда дляимеем:

Условия для значений в узлах:

**(1)**

**(2)**

Дополнительные условия для концов могут быть выбраны произвольно. Для простоты, приравняем вторые производные к нулю для получения функции с минимальной кривизной:

**(3)**

**(4)**

Данный набор из 4-х уравнений позволяет единственным образом определить все необходимые коэффициенты.

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:

Откуда легко выразить недостающую неизвестную:

И записать итоговую формулу для сплайн-функции:

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, поэтому всего неизвестных коэффициентов будет 8.

Условия для значений в узлах:

**(1, 2)**

**(3, 4)**

Условие для значений 1-х производных:

**(5)**

Условие для значений 2-х производных:

**(6)**

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

**(7)**

**(8)**

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

Прогоночные коэффициенты удовлетворяют условиям:

Поэтому, для c1 = c2, начальные значения прогоночных коэффициентов будут равны соответственно:

4. *Написать формулу для определения последнего коэффициента сплайна СN, 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.*