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 $y(x) = x^2$ в диапазоне [0..10].

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

Код программы

```
# алгоритм наилучшего среднеквадратичного приближения
from numpy import array
from numpy.linalg import inv

def best_sqr_approx(X: list, Y: list, W: list, n: int):
    N = len(X)
    n += 1
    if n > N: n = N

    Y_hat = array([[
        sum(W[k] * Y[k] * X[k] ** i for k in range(N))
        for i in range(n)
    ]])

    mat = [[sum(W[k] * X[k] ** (i + j) for k in range(N))
            for j in range(n)
        ] for i in range(n)
    ]

    mat = inv(mat)
    A = Y_hat.dot(mat).flatten()

    def functor(x):
        y = 0
        for i, a in enumerate(A):
            y += a * x ** i
        return y

  return functor

# отрисовка интерфейса
import numpy as np
from random import random
import tkinter as tk
import tkinter.ttk as ttk
from tksheet import Sheet
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.figure import Figure


class DataTable(Sheet):
    def __init__(self, master):
        self.__data = []
```

```python
        super().__init__(master, data=self.__data, headers=["x", "y", "p"])
        self.sheet_display_dimensions(total_columns=3)
        self.enable_bindings()
        self.extra_bindings([("end_edit_cell", self.__execute_callbacks)])
        self.__callbacks = []

    def add_point(self, x, y, p=1):
        self.__data.append([x, y, p])
        self.set_sheet_data(self.__data)
        self.__execute_callbacks()

    def add_callback(self, callback):
        self.__callbacks.append(callback)

    def __execute_callbacks(self, event=None):
        for c in self.__callbacks: c()

    def initialize_data(self, n):
        self.__data.clear()
        for i in range(n):
            self.__data.append([0.0, 0.0, 1])
        self.set_sheet_data(self.__data)

    def randomize_data(self, xmin, ymin, xmax, ymax):
        for row in self.__data:
            row[0] = round(xmin + random() * (xmax - xmin), 2)
            row[1] = round(ymin + random() * (ymax - ymin), 2)
            row[2] = 1
        self.__data.sort(key=lambda d: d[0])
        self.set_sheet_data(self.__data)

    def get_data(self) -> list:
        return self.__data


class Plotter(tk.Frame):
    def __init__(self, master):
        super().__init__(master)

        self.__figure = Figure()
        self.__plot = self.__figure.add_subplot()
        self.__canvas = FigureCanvasTkAgg(self.__figure, master=self)
        self.__canvas.get_tk_widget().grid(row=0, column=0, sticky=tk.NSEW)
        self.rowconfigure(0, weight=1)
        self.columnconfigure(0, weight=1)

        self.plot_data()

    def plot_data(self, data=None, approx=None):
        self.__plot.clear()
        self.__plot.set_title("Наилучшее ср.кв. приближение")
        self.__plot.set_xlabel("x")
        self.__plot.set_ylabel("y")
        if data is not None:
            X = [row[0] for row in data]
            Y = [row[1] for row in data]
            self.__plot.plot(X, Y, "ro", label="данные")
            if approx is not None:
                colors = (None, "r", "g", "b", "y")
                styles = (None, "-", "-", "-", "-")
                for n, X, Y in approx:
                    self.__plot.plot(X, Y, colors[n] + styles[n], label=f"n = {n}")
```

```
            self.__plot.legend()
        self.__canvas.draw()


class App(tk.Tk):
    def __init__(self):
        super().__init__()
        self.title("Лабораторная работа №4")
        self.__configure_grid()

        self.plotter = Plotter(self)
        self.plotter.grid(row=0, column=0, columnspan=4, sticky=tk.NSEW)

        self.data_table = DataTable(self)
        self.data_table.add_callback(self.__display_data)
        self.data_table.grid(row=0, column=4, rowspan=2, sticky=tk.NSEW)

        label = ttk.Label(self, text="N = ")
        label.grid(row=1, column=0, padx=0, pady=10, sticky=tk.E)

        self.n_value_string = tk.StringVar(self, value="10")
        self.editbox = ttk.Entry(self, textvariable=self.n_value_string)
        self.editbox.grid(row=1, column=1, padx=10, pady=10)

        self.initialize_button = ttk.Button(self, text="инициализировать")
        self.initialize_button.grid(row=1, column=2, padx=0, pady=10)
        self.initialize_button.configure(command=self.initialize_action)

        self.randomize_button = ttk.Button(self, text="случайное заполнение")
        self.randomize_button.grid(row=1, column=3, padx=10, pady=10)
        self.randomize_button.configure(command=self.randomize_action)

    def __configure_grid(self):
        rows = [1, 1]
        cols = [1, 0, 0, 0, 1]

        for row, weight in enumerate(rows):
            self.rowconfigure(row, weight=weight)

        for col, weight in enumerate(cols):
            self.columnconfigure(col, weight=weight)

    def feed_approximator(self, approx_factory):
        self.approx_factory = approx_factory
        return self

    def __display_data(self):
        data = self.data_table.get_data()
        approx = []

        if self.approx_factory is not None:
            X = [float(row[0]) for row in data]
            Y = [float(row[1]) for row in data]
            W = [float(row[2]) for row in data]
            for n in 1, 2, 4:
                func = self.approx_factory(X, Y, W, n)
                X_ext = np.linspace(min(X), max(X), 200)
                Y_ext = [func(x) for x in X_ext]
                approx.append([n, X_ext, Y_ext])

        self.plotter.plot_data(data, approx)
```

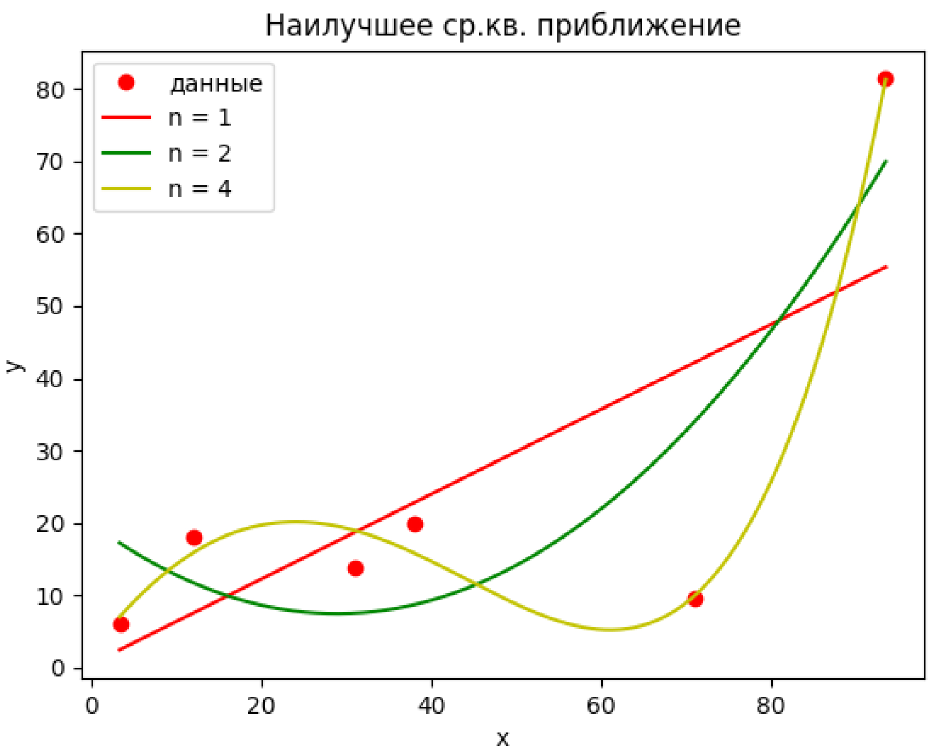
```
    def initialize_action(self):
        try:
            n = int(self.n_value_string.get())
            self.data_table.initialize_data(n)
        except Exception:
            print("bad n")

    def randomize_action(self):
        self.data_table.randomize_data(0, 0, 100, 100)
        self.__display_data()

App().feed_approximator(best_sqr_approx).mainloop()
```

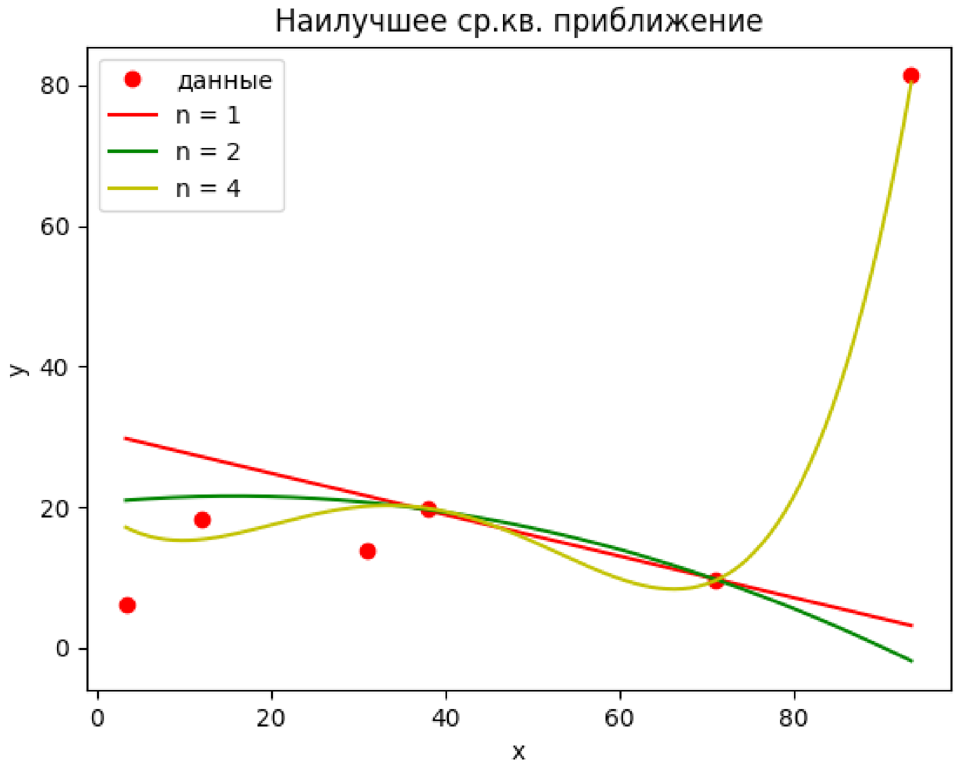
Результаты работы

1. Веса точек одинаковы и равны единице:



	x	y	p
1	3.4	6.15	1
2	12.14	18.18	1
3	31.02	13.92	1
4	37.93	19.9	1
5	70.95	9.56	1
6	93.42	81.33	1

2. Веса точек разные (сами точки те же):



	x	y	p
1	3.4	6.15	0.1
2	12.14	18.18	1
3	31.02	13.92	1
4	37.93	19.9	100
5	70.95	9.56	100
6	93.42	81.33	0.1

Как видно, благодаря коррекции весов удалось добиться отрицательного коэффициента наклона аппроксимирующей прямой.

Контрольные вопросы

1. *Что произойдет при задании степени полинома* $n = N - 1$ *(числу узлов таблицы минус 1)?*

При такой ситуации построенный полином будет проходить точно через все узлы и будет являться полиномом Ньютона степени n.

2. *Будет ли работать Ваша программа при* $n \le N$ *? Что именно в алгоритме требует отдельного анализа данного случая и может привести к аварийной остановке?*

При поиске полинома степени **n** необходимо не менее $n+1$ условий для нахождения неизвестных коэффициентов. При превышении степени n дополнительные условия могут быть введены естественным способом $a_i = 0\,, i > n$. Тогда степень полученного многочлена будет не более n.

На практике (в моей реализации алгоритма) если введённая степень n превышает величину $N-1$, то ей присваивается $N-1$.

3. *Получить формулу для коэффициента полинома a_0 при степени полинома $n=0$. Какой смысл имеет величина, которую представляет данный коэффициент?*

$$\varphi(x) = a_0$$

$$I = \sum_{i=0}^{N} \rho_i [y_i - a_0]^2$$

$$\frac{\partial I}{\partial a_0} = -2 \sum_{i=0}^{N} \rho_i (y_i - a_0)$$

$$\frac{\partial I}{\partial a_0} = 0: \qquad \sum_{i=0}^{N} \rho_i a_0 = \sum_{i=0}^{N} \rho_i y_i$$

$$a_0 = \frac{\sum_{i=0}^{N} \rho_i y_i}{\sum_{i=0}^{N} \rho_i}$$

Полученный коэффициент a_0 является взвешенным средним арифметическим значением для ординат исходного набора точек.

4. *Записать и вычислить определитель матрицы СЛАУ для нахождения коэффициентов полинома для случая, когда $n = N = 2$. Принять все $\rho_i = 1$.*

$$\varphi(x) = a_0 + a_1 x + a_2 x^2$$

$$\begin{cases} (\varphi, \varphi_0) = (y, \varphi_0) \\ (\varphi, \varphi_1) = (y, \varphi_1) \\ (\varphi, \varphi_2) = (y, \varphi_2) \end{cases} \qquad \begin{cases} a_0 X_0 + a_1 X_1 + a_2 X_2 = Y_0 \\ a_0 X_1 + a_1 X_2 + a_2 X_3 = Y_1 \\ a_0 X_2 + a_1 X_3 + a_2 X_4 = Y_2 \end{cases}$$

$$X_k \stackrel{\text{def}}{=} \sum_{i=0}^{2} x_i^k \qquad Y_k \stackrel{\text{def}}{=} \sum_{i=0}^{2} y_i\, x_i^k$$

$$\begin{pmatrix} X_0 & X_1 & X_2 \\ X_1 & X_2 & X_3 \\ X_2 & X_3 & X_4 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} Y_0 \\ Y_1 \\ Y_2 \end{pmatrix} \qquad XA = Y$$

$$\det(X) = X_0 X_2 X_4 + 2 X_1 X_2 X_3 - X_0 X_3 X_3 - X_1 X_1 X_4 - (X_2)^3$$

5. *Построить СЛАУ при выборочном задании степеней аргумента полинома* $\varphi(x) = a_0 + a_1 x^n + a_2 x^m$, *причем степени n и m в этой формуле известны.*

$$\begin{cases} (\varphi, 1) = (y, 1) \\ (\varphi, x^n) = (y, x^n) \\ (\varphi, x^m) = (y, x^m) \end{cases} \qquad \begin{cases} a_0 X_0 + a_1 X_n + a_2 X_m = Y_0 \\ a_0 X_n + a_1 X_{2n} + a_2 X_{n+m} = Y_n \\ a_0 X_m + a_1 X_{n+m} + a_2 X_{2m} = Y_m \end{cases}$$

$$X_k \stackrel{\text{def}}{=} \sum_{i=0}^{2} x_i^k \qquad Y_k \stackrel{\text{def}}{=} \sum_{i=0}^{2} y_i\, x_i^k$$

6. *Предложить схему алгоритма решения задачи из вопроса 5, если степени n и m подлежат определению наравне с коэффициентами* a_k , *т. е. количество неизвестных равно 5.*

Можно составить цикл по всем сочетаниям значений переменных n и m, в котором рассчитывать на их основе значения коэффициентов a_k. Для каждого найденного набора вычислять значение ошибки I, и среди всех ошибок выбрать наименьшую и использовать соответствующие степени n, m и коэффициенты a_k.

Важно понимать при этом, что максимально возможные значения степеней не следует брать более 6-7, так как это может привести к значительным отклонениям аппроксимирующей функции.