|  |  |
| --- | --- |
| **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**

**# 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 = []**

**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):**

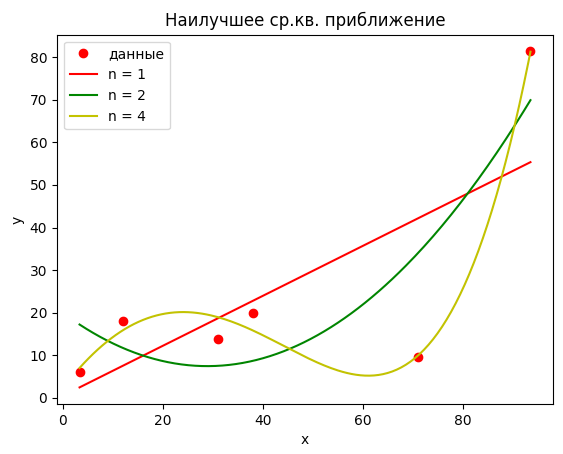
**self.data\_table.randomize\_data(0, 0, 100, 100)**

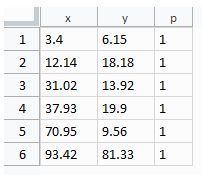
**self.\_\_display\_data()**

**App().feed\_approximator(best\_sqr\_approx).mainloop()**

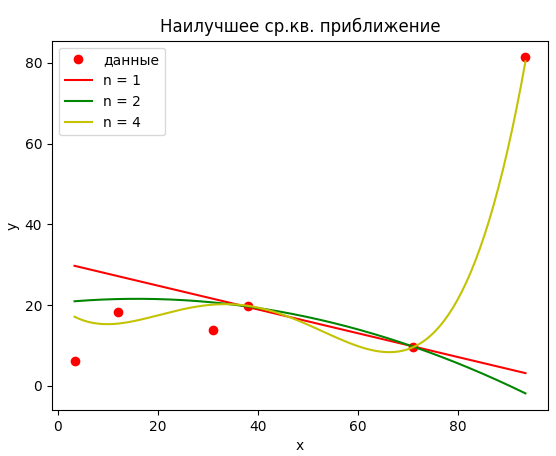
**Результаты работы**

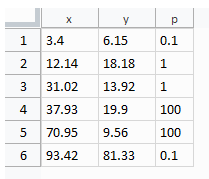
1. Веса точек одинаковы и равны единице:





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





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

**Контрольные вопросы**

1. *Что произойдет при задании степени полинома (числу узлов таблицы минус 1)?*

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

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

При поиске полинома степени **n** необходимо не менее условий для нахождения неизвестных коэффициентов. При превышении степени дополнительные условия могут быть введены естественным способом . Тогда степень полученного многочлена будет не более .

На практике (в моей реализации алгоритма) если введённая степень превышает величину , то ей присваивается .

3. *Получить формулу для коэффициента полинома a0 при степени полинома . Какой смысл имеет величина, которую представляет данный коэффициент?*

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

4. *Записать и вычислить определитель матрицы СЛАУ для нахождения коэффициентов полинома для случая, когда . Принять все .*

5. *Построить СЛАУ при выборочном задании степеней аргумента полинома , причем степени n и m в этой формуле известны.*

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

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

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.