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

«Построение и программная реализация алгоритма наилучшего среднеквадратичного приближения»

Студент Маслова Марина Дмитриевна
фамилия, имя, отчество

Группа ИУ7-43Б

Оценка (баллы) ____________________

Преподаватель Градов Владимир Михайлович
фамилия, имя, отчество

2021 г.

Оглавление

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 весами ρ_i с количеством узлов N, сформированная случайным образом.
2. Степень аппроксимирующего полинома — n.

Описание алгоритма

1. Выбирается степень полинома $n << N$.
2. Составляется СЛАУ:

$$\sum_{m=0}^{n} (x^k, x^m)\, a_m = (y, x^k),\ 0 \le k \le n,$$

где $(x^k,\ x^m) = \sum_{i=1}^{N} \rho_i x_i^{k+m}$, $(y,\ x^k) = \sum_{i=1}^{N} \rho_i y_i x_i^k$.

3. Решением СЛАУ находятся коэффициенты полинома a_k.
4. С помощью найденных коэффициентов рассчитываются точки кривой, по которым строится график.

Код программы

Код программы представлен на листингах 1-4.

Листинг 1. rms.py

```
"""
    Модуль среднеквадратичного приближения
"""
EPS = 1e-6

def getEquationSystem(table, degree):
    """
        Поиск коэффициентов СЛАУ для
        метода наименьших квадратов
```

```
    """
    slae = [[0] * (degree + 2) for i in range(degree + 1)]

    for i in range(degree + 1):
        for j in range(degree + 1):
            curA = 0
            for row in table:
                curA += row[2] * pow(row[0], i) * pow(row[0], j)
            slae[i][j] = curA

        curB = 0
        for row in table:
            curB += row[2] * row[1] * pow(row[0], i)
        slae[i][degree + 1] = curB

    return slae


def solveSLAE(slae):
    """
        Решение СЛАУ
    """
    length = len(slae)
    for j in range(length):
        for i in range(j + 1, length):
            if abs(slae[j][j]) < EPS:
                continue

            curCoef = slae[i][j] / slae[j][j]

            for k in range(j, length + 1):
                slae[i][k] -= curCoef * slae[j][k]

    answer = [0. for i in range(length)]

    for i in range(length - 1, -1, -1):
        for j in range(length - 1, i, -1):
            slae[i][length] -= slae[i][j] * answer[j]

        if abs(slae[i][i]) < EPS:
            answer[i] = slae[i][length]
            continue

        answer[i] = slae[i][length] / slae[i][i]

    return answer
```

Листинг 2. graphics.py

```
"""
    Модуль получения данных для графиков
"""
import rms


def getPointsFunc(polinomial, span):
    """
        Получение точек для построения графика
    """
    step = (span[1] - span[0]) / 1000
```

```python
    xData = []
    yData = []

    xCur = span[0] - step

    while xCur < span[1] + step:
        yCur = 0

        for i, coef in enumerate(polinomial):
            yCur += coef * pow(xCur, i)
        xData.append(xCur)
        yData.append(yCur)

        xCur += step

    return xData, yData


def getXs(table):
    """ Получение списка координат x """
    return [rec[0] for rec in table]


def getYs(table):
    """ Получение списка координат y """
    return [rec[1] for rec in table]


def findMaximumX(table):
    """ Получение максимума из координат x """
    return max(getXs(table))


def findMinimumX(table):
    """ Получение минимума из координат x """
    return min(getXs(table))


def getPlot(table, degree):
    """
        Получение списка точек по
        исходной таблице и степени полинома
    """
    slae = rms.getEquationSystem(table, degree)
    polinomial = rms.solveSLAE(slae)
    minX = findMinimumX(table)
    maxX = findMaximumX(table)
    pointsFunc = getPointsFunc(polinomial, [minX, maxX])

    return pointsFunc
```

Листинг 3. points.py

```python
"""
    Модуль для генерации табличной
    функции с весами узлов
"""
from numpy import random

def generateRandomList(left, right, num):
    """
```

```
        Генерация списка со случайными
        значениями в заданном диапазоне
        заданного размера
    """
    return [round(el[0], 2) for el in
            random.uniform(left, right, size=(num, 1)).tolist()]



def generateTable(num, equal=[False, 0.]):
    """
        Генерация таблицы точек заданного размера
    """
    table = []

    xlist = generateRandomList(-100, 100, num)
    ylist = generateRandomList(-100, 100, num)

    if equal[0]:
        rolist = [equal[1]] * num
    else:
        rolist = generateRandomList(0, 100, num)

    for i in range(num):
        rec = [xlist[i], ylist[i], rolist[i]]
        table.append(rec)

    return table
```

Листинг 4. main.py

```
"""
    Модуль для запуска программы
    ЛАБОРАТОРНАЯ РАБОТА №4
    ПОСТРОЕНИЕ И ПРОГРАММНАЯ РЕАЛИЗАЦИЯ АЛГОРИТМА
    НАИЛУЧШЕГО СРЕДНЕКВАДРАТИЧНОГО ПРИБЛИЖЕНИЯ
"""

import sys
from PyQt5 import QtWidgets, uic
from PyQt5.QtWidgets import QTableWidget, QTableWidgetItem, QHeaderView
import matplotlib.pyplot as plt

from MainWindow import Ui_MainWindow
import points
import graphics

class doubleDelegate(QtWidgets.QItemDelegate):
    """
        Класс настройки полей для ввода
        вещественных чисел
    """

    def createEditor(self, parent, option, index):
        """
            Настройка поля ввода на
            получение только вещественных чисел
        """
        self.doubleSpin = QtWidgets.QDoubleSpinBox(parent)
        self.doubleSpin.setMaximum(1000)
```

```
        if index.column() == 2:
            self.doubleSpin.setMinimum(0)
        else:
            self.doubleSpin.setMinimum(-1000)
        return self.doubleSpin


def callError(title, text):
    msg = QtWidgets.QMessageBox()
    msg.setIcon(QtWidgets.QMessageBox.Critical)
    msg.setWindowTitle(title)
    msg.setText(text)
    msg.exec_()


class MainWindow(QtWidgets.QMainWindow, Ui_MainWindow):
    """
        Класс главного окна
    """

    def __init__(self, *args, **kwargs):
        """
            Инициализация главного окна
        """
        super(MainWindow, self).__init__(*args, **kwargs)
        self.setupUi(self)

        self.table_init()
        self.generateBtn.clicked.connect(self.generateTable)
        self.weightsRadioBtn.clicked.connect(self.switch)
        self.plotBtn.clicked.connect(self.getPlots)


    def table_init(self):
        """
            Начальные настройки таблицы
        """
        self.pointsTable.setColumnCount(3)
        delegate = doubleDelegate()
        for i in range(3):
            self.pointsTable.horizontalHeader().setSectionResizeMode(i,
QHeaderView.Stretch)
            self.pointsTable.setItemDelegateForColumn(i, delegate)


    def switch(self):
        """
            Блокировка/разблокировка поля ввода веса
        """
        self.weightsDSpin.setDisabled(self.weightsDSpin.isEnabled())


    def generateTable(self):
        """
            Генерация таблицы по введенным данным
        """
        num = self.pointsNumSpin.value()

        equal = [False, 0.]
        if self.weightsRadioBtn.isChecked():
```

```
            equal = [True, self.weightsDSpin.value()]

        table = points.generateTable(num, equal)

        self.pointsTable.setRowCount(num)
        for i, rec in enumerate(table):
            for j, value in enumerate(rec):
                self.pointsTable.setItem(i, j,
QTableWidgetItem(str(value)))


    def getPlots(self):
        """
            Получение графиков по таблице
        """
        checks = [
            self.degree1Check,
            self.degree2Check,
            self.degree3Check,
            self.degree4Check,
            self.degree5Check
            ]

        degrees = []
        for i, check in enumerate(checks):
            if check.isChecked():
                if i + 1 >= self.pointsTable.rowCount():
                    callError("n >= N!", "Полином не может быть
построен")
                    return

                degrees.append(i + 1)

        plt.close()

        table = self.getTable()
        plt.plot(graphics.getXs(table), graphics.getYs(table),
                "o", label="Исходные")

        for degree in degrees:
            plot = graphics.getPlot(table, degree)
            plt.plot(plot[0], plot[1], label="{:d}-я
степень".format(degree))

        plt.get_current_fig_manager().window.move(700, 100)
        plt.get_current_fig_manager().resize(1000, 758)
        plt.grid()
        plt.legend()
        plt.show()


    def getTable(self):
        """
            Получение таблицы
        """
        table = []
        for i in range(self.pointsTable.rowCount()):
            table.append([float(self.pointsTable.item(i, j).text())
                              for j in range(3)])
        return table
```

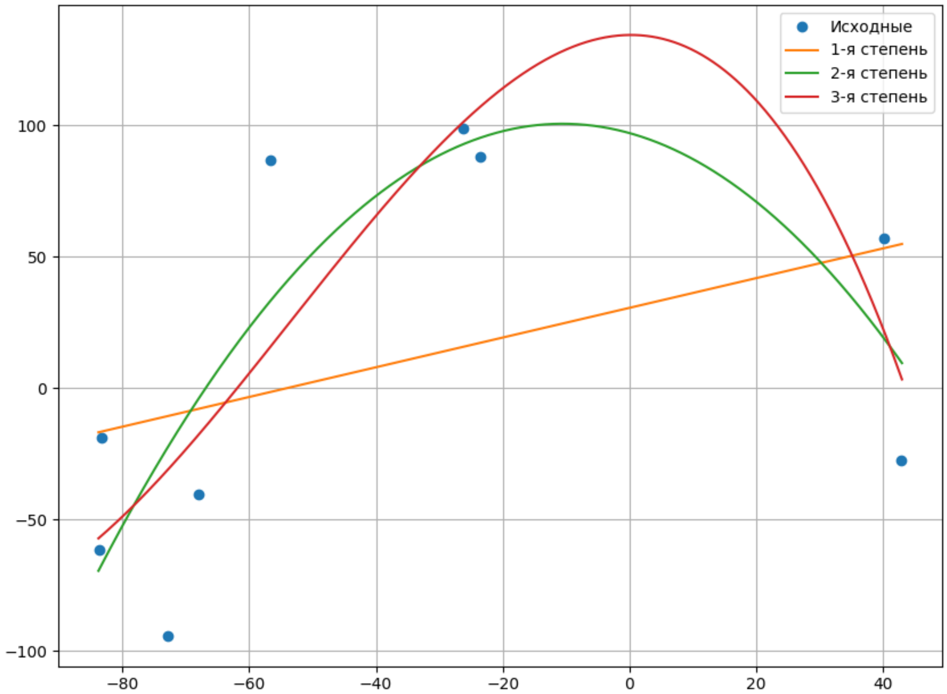
```
if __name__ == '__main__':
    app = QtWidgets.QApplication(sys.argv)
    main = MainWindow()
    main.move(200, 100)
    main.show()
    sys.exit(app.exec_())
```

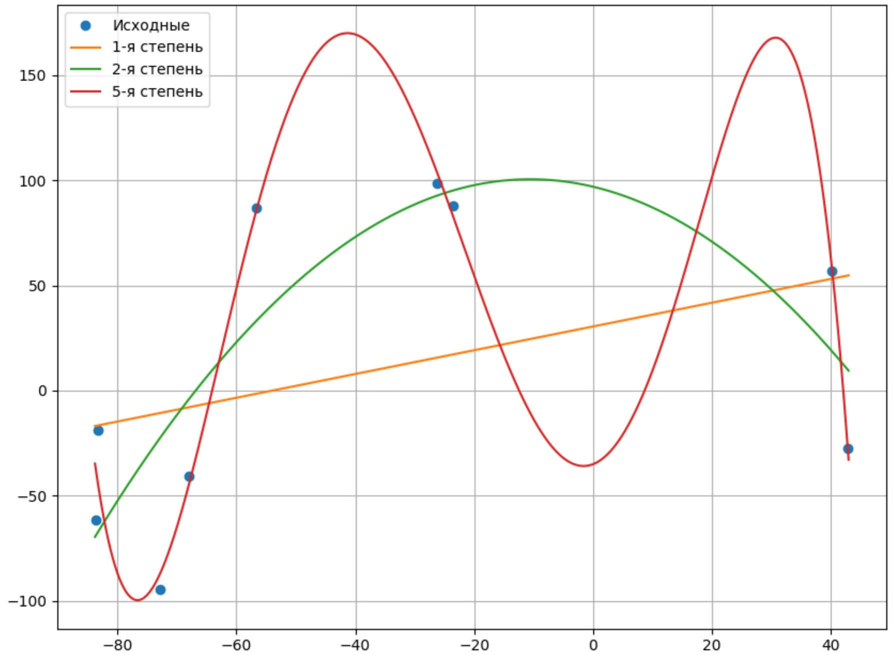
Результат работы

Таблица для демонстрации работы программы при одинаковых весах точек:

	x	y	ρ
1	-72.77	-94.34	1.0
2	-56.63	86.68	1.0
3	-68.04	-40.44	1.0
4	40.26	56.82	1.0
5	-83.22	-18.73	1.0
6	42.89	-27.66	1.0
7	-23.52	87.81	1.0
8	-83.67	-61.53	1.0
9	-26.31	98.74	1.0

Полученные графики для степеней $n = 1; 2; 3; 5$:





Демонстрация изменения угла наклона прямой при изменении весов точек. Таблицы:

	x	y	ρ
1	84.87	58.36	1.0
2	5.74	-83.41	1.0
3	-56.07	67.76	1.0
4	-62.7	82.42	1.0
5	52.41	-22.54	1.0
6	-52.78	7.96	1.0
7	18.58	-33.38	1.0
8	-24.53	17.89	1.0
9	-4.9	12.83	1.0
10	7.79	-95.42	1.0

Одинаковые веса точек
(оранжевая прямая)

	x	y	ρ
1	84.87	58.36	10
2	5.74	-83.41	1.0
3	-56.07	67.76	0.5
4	-62.7	82.42	0.5
5	52.41	-22.54	0.5
6	-52.78	7.96	1.0
7	18.58	-33.38	1.0
8	-24.53	17.89	0.5
9	-4.9	12.83	1.0
10	7.79	-95.42	1.0

Разные веса точек
(зеленая прямая)

Полученные по таблицам графики:

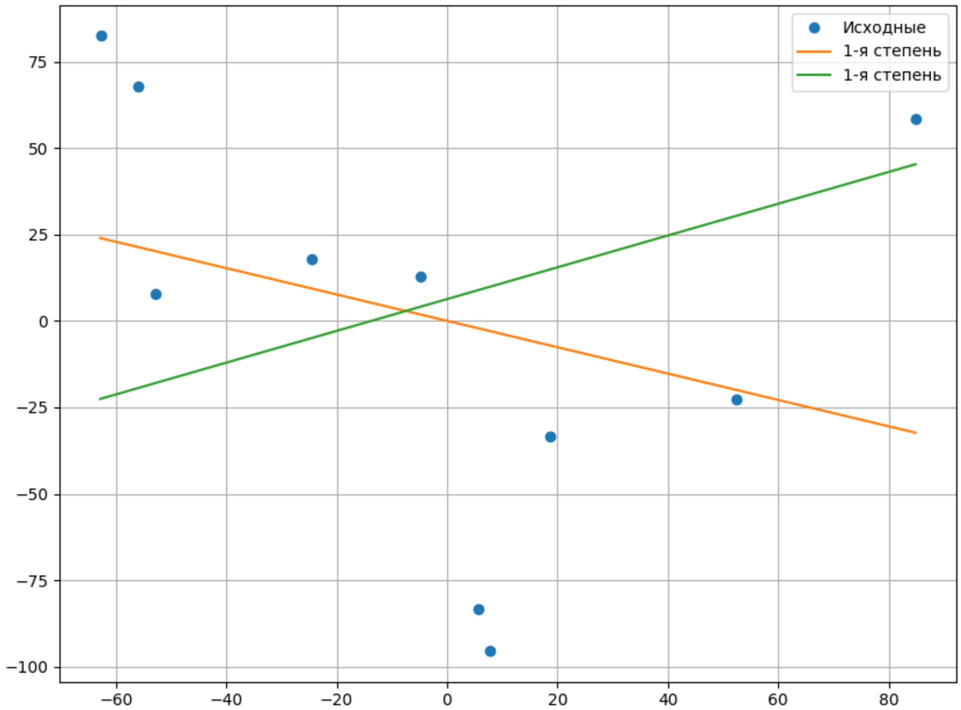
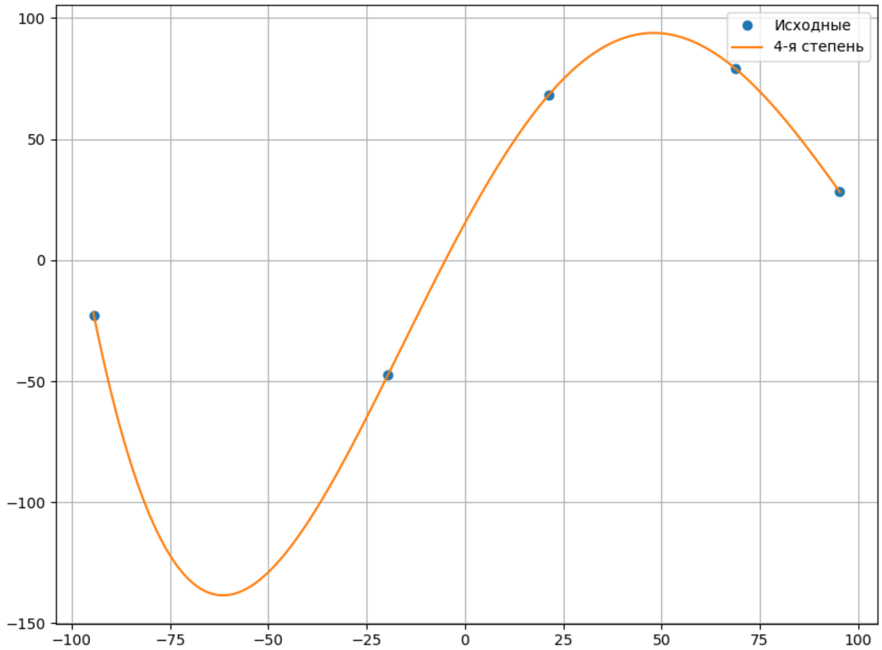


Таблица и график при $n = N - 1$:

	x	y	ρ
1	-94.33	-22.95	45.92
2	21.28	68.18	61.18
3	68.68	79.22	97.06
4	95.21	28.44	99.91
5	-19.57	-47.42	24.01



Контрольные вопросы

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

При $n = N - 1$ будет достаточно точек для однозначного определения полинома, так как для построения полинома степени n требуется $n + 1$узел. Поэтому график, построенный при данных условиях будет проходить через все заданные точки, причем при любых весах точек. Последнее утверждение вытекает из того, что при $n = N - 1$в соотношении:

$$\sum_{i=1}^{N} \rho_i \left[y(x_i) - \varphi(x_i)\right]^2 = \min$$

— выражение в скобках обращается в нуль, следовательно правая часть соотношения не зависит от узлов и всегда принимает минимальное значение.

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

Работа программы при $n \geq N$ будет некорректной, так как будет не хватать узлов для построения полинома. В таком случае построенная СЛАУ будет иметь определитель, равный нулю, и либо иметь бесконечное количество решений, либо не иметь их. Оба случая при решении СЛАУ методом Гаусса приводят к делению на нуль. Ошибку можно проверять непосредственно при решении СЛАУ либо при получении данных от пользователя (в случае, если $n \geq N$ сразу возвращать ошибку).

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

$$\varphi(x) = a_0$$

$$(x^0, x^0)\, a_0 = (y, x^0)$$

$$(x^0, x^0) = \sum_{i=1}^{N} \rho_i$$

$$(y, x^0) = \sum_{i=1}^{N} \rho_i y_i$$

$$\sum_{i=1}^{N} \rho_i \cdot a_0 = \sum_{i=1}^{N} \rho_i y_i$$

$$a_0 = \frac{\sum_{i=1}^{N} \rho_i y_i}{\sum_{i=1}^{N} \rho_i}$$

$$a_0 = \frac{\rho_1 y_1 + \rho_2 y_2 + \ldots + \rho_N y_N}{\sum_{i=1}^{N} \rho_i}$$

$$a_0 = \frac{\rho_1}{\sum\limits_{i=1}^{N} \rho_i} y_1 + \frac{\rho_2}{\sum\limits_{i=1}^{N} \rho_i} y_2 + \ldots + \frac{\rho_N}{\sum\limits_{i=1}^{N} \rho_i} y_N$$

$$a_0 = p_1 y_1 + p_2 y_2 + \ldots + p_N y_N = \sum_{i=1}^{N} p_i y_i = M(Y)$$

Коэффициент a_0 представляет математическое ожидание, которое приближенно равно среднему значению случайной величины.

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

$n = N = 2$

Пусть даны точки:

x	y	ρ
x_1	y_1	1
x_2	y_2	1

Матрица СЛАУ примет вид:

$$\begin{pmatrix} 2 & x_1 + x_2 & x_1^2 + x_2^2 \\ x_1 + x_2 & x_1^2 + x_2^2 & x_1^3 + x_2^3 \\ x_1^2 + x_2^2 & x_1^3 + x_2^3 & x_1^4 + x_2^4 \end{pmatrix}$$

Найдем определитель матрицы:

$$\begin{vmatrix} 2 & x_1 + x_2 & x_1^2 + x_2^2 \\ x_1 + x_2 & x_1^2 + x_2^2 & x_1^3 + x_2^3 \\ x_1^2 + x_2^2 & x_1^3 + x_2^3 & x_1^4 + x_2^4 \end{vmatrix} = 2((x_1^2 + x_2^2)(x_1^4 + x_2^4) - (x_1^3 + x_2^3)(x_1^3 + x_2^3)) -$$

$$- (x_1 + x_2)((x_1 + x_2)(x_1^4 + x_2^4) - (x_1^3 + x_2^3)(x_1^2 + x_2^2)) + (x_1^2 + x_2^2)((x_1 +$$

$$+ x_2)(x_1^3 + x_2^3) - (x_1^2 + x_2^2)(x_1^2 + x_2^2)) = 0$$

Определитель матрицы равен нулю, что соответствует рассуждениям в вопросе 2.

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

$$\begin{cases} (x^0, x^0)a_0 + (x^0, x^m)a_1 + (x^0, x^n)a_2 = (y, x^0) \\ (x^m, x^0)a_0 + (x^m, x^m)a_1 + (x^m, x^n)a_2 = (y, x^m) \\ (x^n, x^0)a_0 + (x^n, x^m)a_1 + (x^n, x^n)a_2 = (y, x^n) \end{cases}$$

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.

Задачу можно решить перебором всевозможных пар n и m. Для каждой пары находятся коэффициенты полинома a_k, строится полином $\varphi(x)$ и выбирается та пара, полином которой наилучшим образом аппроксимирует заданную функцию, т. е. выполняется условие:

$$\sum_{i=1}^{N} \rho_i \, [y(x_i) - \varphi(x_i)]^2 = \min$$