МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное образовательное учреждение высшего образования **«Национальный исследовательский университет ИТМО»**

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

ЛАБОРАТОРНАЯ РАБОТА №4

‘ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА’

Вариант №25

Студент:
Хоанг Ван Куан
Группа P3266

Преподаватель:
Машина Екатерина Александровна

Санкт-Петербург, 2024

1. Цель работы

Найти функцию, являющуюся наилучшим приближением заданной табличной функции по методу наименьших квадратов.

2. Порядок выполнения работы

- 1 часть: Вычислительная реализация задачи
    1. Сформировать таблицу табулирования заданной функции на указанном интервале
       $y = \frac{28x}{x^4 + 25}, \quad x \in [0{,}4], \quad h = 0{,}4$
    2. Построить линейное и квадратичное приближения по 11 точкам заданного интервала
    3. Найти среднеквадратические отклонения для каждой аппроксимирующей функции. Ответы дать с тремя знаками после запятой
    4. Выбрать наилучшее приближение
    5. Построить графики заданной функции, а также полученные линейное и квадратичное приближения

- 2 часть: Программная реализация задачи
    1. Предусмотреть ввод исходных данных из файла/консоли (таблица $y = f(x)$должна содержать от 8 до 12 точек)
    2. Реализовать метод наименьших квадратов, исследуя все указанные функции
    3. Предусмотреть вывод результатов в файл/консоль: коэффициенты аппроксимирующих функций, среднеквадратичное отклонение, массивы значений $x_i, y_i, \varphi(x_i), \varepsilon_i$
    4. Для линейной зависимости вычислить коэффициент корреляции Пирсона
    5. Программа должна отображать наилучшую аппроксимирующую функцию
    6. Организовать вывод графиков функций, графики должны полностью отображать весь исследуемый интервал (с запасом)
    7. Программа должна быть протестирована при различных наборах данных, в том числе и некорректных

3. Рабочие формулы

Аппроксимировать f(x) функцией $\varphi(x)$

$$\varphi(x) = a_0 + a_1 x + \cdots + a_n x^n$$

МЕТОД НАИМЕНЬШИХ КВАДРАТОВ

Преобразуем полученную линейную систему уравнений: раскроем скобки и перенесем свободные слагаемые в правую часть выражения.

$$\begin{cases} a_0 n + a_1 \sum_{i=1}^{n} x_i + \cdots + a_{m-1} \sum_{i=1}^{n} x_i^{m-1} + a_m \sum_{i=1}^{n} x_i^m = \sum_{i=1}^{n} y_i \\ a_0 \sum_{i=1}^{n} x_i + a_1 \sum_{i=1}^{n} x_i^2 + \cdots + a_{m-1} \sum_{i=1}^{n} x_i^m + a_m \sum_{i=1}^{n} x_i^{m+1} = \sum_{i=1}^{n} x_i y_i \\ \cdots \cdots \cdots \\ a_0 \sum_{i=1}^{n} x_i^m + a_1 \sum_{i=1}^{n} x_i^{m+1} + \cdots + a_{m-1} \sum_{i=1}^{n} x_i^{2m-1} + a_m \sum_{i=1}^{n} x_i^{2m} = \sum_{i=1}^{n} x_i^m y_i \end{cases}$$

в матричном виде:

$$\begin{vmatrix} n & \sum_{i=1}^{n} x_i & \cdots & \sum_{i=1}^{n} x_i^m \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 & \cdots & \sum_{i=1}^{n} x_i^{m+1} \\ \cdots & \cdots & \cdots & \cdots \\ \sum_{i=1}^{n} x_i^m & \sum_{i=1}^{n} x_i^{m+1} & \cdots & \sum_{i=1}^{n} x_i^{2m} \end{vmatrix} \cdot \begin{vmatrix} a_0 \\ a_1 \\ \cdots \\ \cdots \\ a_m \end{vmatrix} = \begin{vmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i y_i \\ \cdots \\ \sum_{i=1}^{n} x_i^m y_i \end{vmatrix}$$

Коэффициент корреляции

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

Среднеквадратичное отклонение

$$\delta = \sqrt{\frac{\sum_{i=1}^{n}(\varphi(x_i) - y_i)^2}{n}}$$

Выбор аппроксимирующей функции

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \varphi_i)^2}{\sum_{i=1}^{n} \varphi_i^2 - \frac{1}{n}\left(\sum_{i=1}^{n} \varphi_i\right)^2}$$

4. Вычислительная часть

- Сформировать таблицу табулирования функции

  $y = \frac{28x}{x^4 + 25}, \quad x \in [0,4], \quad h = 0,4$

x	0.0	0.4	0.8	1.2	1.6	2.0	2.4	2.8	3.2	3.6	4.0
y	0	0.448	0.882	1.241	1.42	1.366	1.155	0.907	0.69	0.522	0.399

- Построить линейное и квадратичное приближения по 11 точкам заданного интервала

+ Линейное приближения:

Для определения вида зависимости. Выбираем многочлен первой степени и строим линейную модель $\mathrm{P}_1(x) = ax + b$

Вычисляем суммы:

$$SX = \sum_{i=1}^{n} x_i = 22$$

$$SXX = \sum_{i=1}^{n} {x_i}^2 = 61{,}6$$

$$SY = \sum_{i=1}^{n} y_i = 9{,}03$$

$$SXY = \sum_{i=1}^{n} x_i y_i = 18{,}3728$$

Получим систему управнений для нахождения параметров a и b

$$\begin{cases} aSXX + bSX = SXY \\ aSX + bn = SY \end{cases} \rightarrow \begin{cases} 61{,}6a + 22b = 18{,}3728 \\ 22a + 11b = 9{,}03 \end{cases}$$

Решая систему, получим значения коэффициентов:

$a \approx 0{,}0178$

$b \approx 0{,}785$

Проверим правильность выбора линейной модели. Для этого вычислим значения аппроксимирующей финкции $P_1(x) = 0.0178x + 0.785$

№ п.п	1	2	3	4	5	6	7	8	9	10	11
x	0	0.4	0.8	1.2	1.6	2	2.4	2.8	3.2	3.6	4.0
y	0	0.448	0.882	1.241	1.42	1.366	1.155	0.907	0.69	0.522	0.399
$P_1(x)=ax+b$	0.785	0.79212	0.79924	0.80636	0.81348	0.8206	0.82772	0.83484	0.84196	0.84908	0.8562
ε_i	-0.785	-0.34412	0.08276	0.43464	0.60652	0.5454	0.32728	0.07216	-0.15196	-0.32708	-0.4572

Определим меру отклонения $S = \sum_{i=1}^{n} {\varepsilon_i}^2 = 2.047$

Среднеквадратичное отклонение $\delta = \sqrt{\frac{\sum_{i=1}^{n} (\varphi(x_i) - y_i)^2}{n}} = 0.4314$

+ Квадратичное приближения:

Для определения вида зависимости. Выбираем многочлен второй степени и строим линейную модель $\mathrm{P}_2(x) = a_0 + a_1 x + a_2 x^2$

Сумма квадратов отклонений запишется следующим образом:

$$S = \sum_{i=1}^{n} (a_0 + a_1 x + a_2 x^2 - y_i)^2 \rightarrow min$$

Вычислим:

$\sum_{i=1}^{n} x_i = 22$ $\quad$ $\sum_{i=1}^{n} {x_i}^2 = 61{,}6$

$\sum_{i=1}^{n} {x_i}^3 = 193{,}6$ $\quad$ $\sum_{i=1}^{n} {x_i}^4 = 648{,}5248$

$\sum_{i=1}^{n} y_i = 9{,}03$ $\quad$ $\sum_{i=1}^{n} x_i y_i = 18{,}3728$

$\sum_{i=1}^{n} {x_i}^2 y_i = 45{,}5008$

Получим систему линейных управнений, решив которую, определим значения коэффициентов эмпирической формулы:

$$\begin{cases} 11a_0 + 22a_1 + 61{,}6a_2 = 9{,}03 \\ 22a_0 + 61{,}6a_1 + 193{,}6a_2 = 18{,}3728 \\ 61{,}6a_0 + 193{,}6a_1 + 648{,}5248a_2 = 45{,}5008 \end{cases} \rightarrow \begin{cases} a_0 = 0.095 \\ a_1 = 1.168 \\ a_2 = -0.288 \end{cases}$$

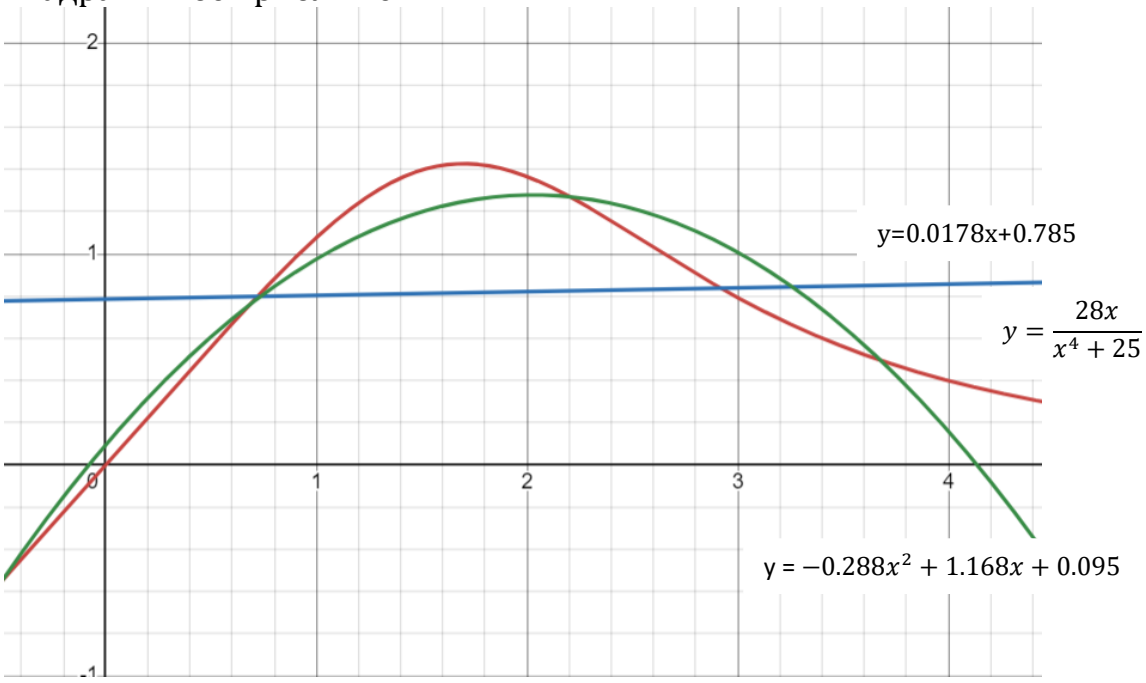
Проверим правильность выбора линейной модели. Для этого вычислим значения аппроксимирующей финкции $P_2(x) = -0.288x^2 + 1.168x + 0.095$

№ п.п	1	2	3	4	5	6	7	8	9	10	11
x	0	0.4	0.8	1.2	1.6	2	2.4	2.8	3.2	3.6	4.0
y	0	0.448	0.882	1.241	1.42	1.366	1.155	0.907	0.69	0.522	0.399
$P_2(x)= a_0 + a_1x + a_2x^2$	0.095	0.51612	0.84508	1.08188	1.22652	1.279	1.23932	1.10748	0.88348	0.56732	0.159
ε_i	-0.095	-0.06812	0.03692	0.15912	0.19348	0.087	-0.08432	-0.20048	-0.19348	-0.04532	0.24

Определим меру отклонения $S = \sum_{i=1}^{n} {\varepsilon_i}^2 = 0.229$

Среднеквадратичное отклонение $\delta = \sqrt{\frac{\sum_{i=1}^{n}(\varphi(x_i)-y_i)^2}{n}} = 0.1445$

- Выбрать наилучшее приближение
  Наилучшее приближение: Квадратичное приближения
- Построить графики заданной функции, а также полученные линейное и квадратичное приближения



5. Листинг программы

```
from dataclasses import dataclass
import math
```

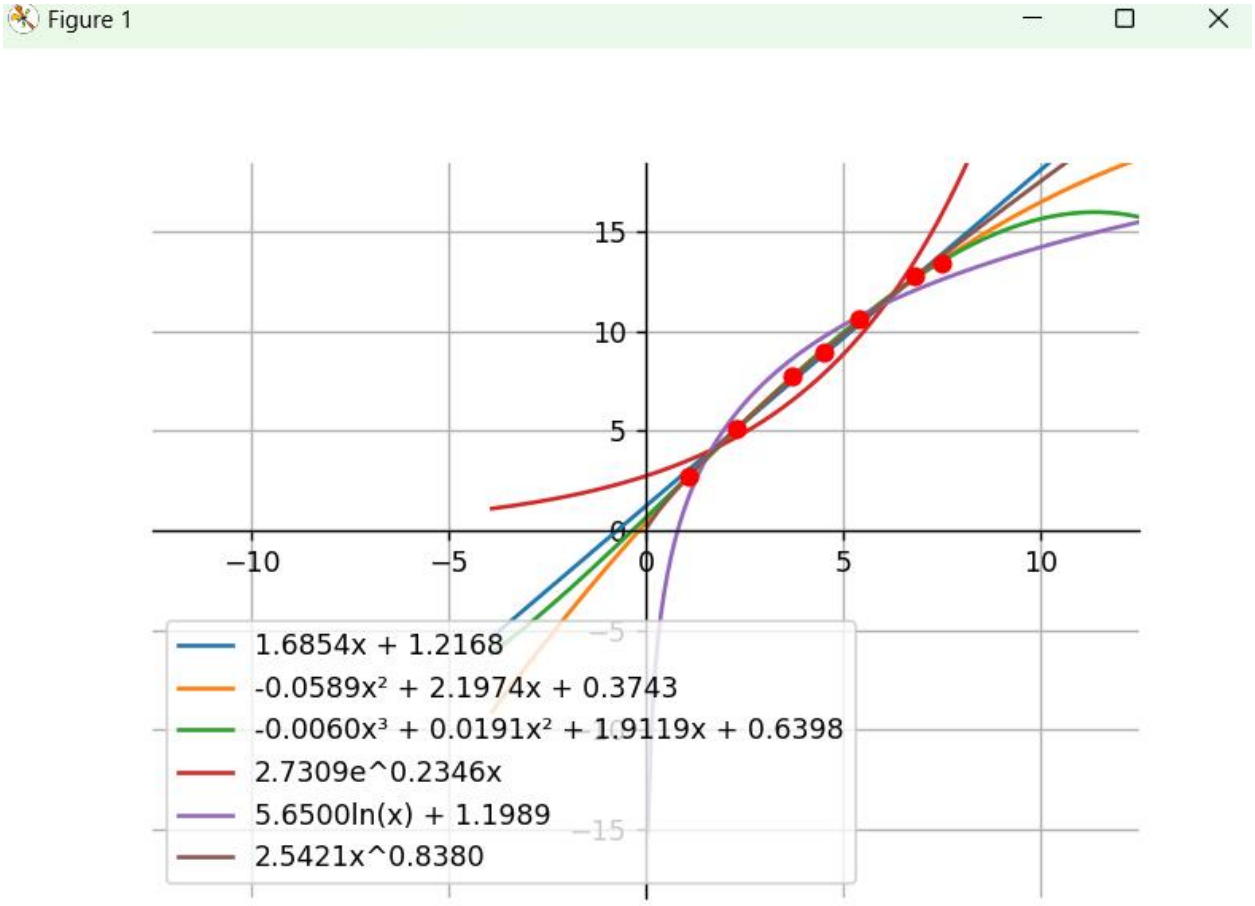
```
import numpy as np


@dataclass
class Result:
    coefficients: iter
    apply: callable
    function: str
    S: float
    deviation: float
    confidence: float
    r: float | None = None
    def __lt__(self, other):
        return self.deviation < other.deviation
    def __le__(self, other):
        return self.deviation == other.deviation

def getData(x, y):
    x = np.array(x)
    y = np.array(y)
    return x, y

# среднеквадратичное отклонение
def msr(phi, y):
    return (np.sum((phi - y) ** 2) / len(y)) ** 0.5

#  достоверность аппроксимации
def confidence(phi, y):
    return 1 - np.sum((y - phi) ** 2) / (np.sum(phi ** 2) - np.sum(phi) ** 2 /
len(y))

# Коэффициент корреляции
def r(x, y):
    x0, y0 = np.mean(x), np.mean(y)
    return np.sum((x - x0) * (y - y0)) / (np.sum((x - x0) ** 2) * np.sum((y - y0)
** 2)) ** 0.5

# линейная функция - done
def Approximation_linear(x, y):
    x, y = getData(x, y)
    SX, SXX, SY, SXY = np.sum(x), np.sum(x ** 2), np.sum(y), np.sum(x * y)
    b, a = np.linalg.solve(
        np.array([[len(x), SX], [SX, SXX]]),
        np.array([SY, SXY])
    )
    # Сумма квадратов отклонений
    phi = a * x + b
    S = np.sum((phi - y) ** 2)

    return Result(
        coefficients = (a, b),
```

```
        apply = lambda x: a * x + b,
        function = f'{a:.4f}x + {b:.4f}',
        S = S,
        r = r(x, y),
        confidence = confidence(phi, y),
        deviation = msr(phi, y)
    )

# полиномиальная функция 2-й степени - done
def Approximation_degree2(x, y):
    x, y = getData(x, y)
    SX, SXX, SXXX, SXXXX, SY, SXY, SXXY = np.sum(x), np.sum(x**2), np.sum(x**3),
np.sum(x**4), np.sum(y), np.sum(x*y), np.sum(x*x*y)
    a0, a1, a2 = np.linalg.solve(
        np.array([[len(x), SX, SXX], [SX, SXX, SXXX], [SXX, SXXX, SXXXX]]),
        np.array([SY, SXY, SXXY])
    )

     # Сумма квадратов отклонений
    phi = a2*x**2 + a1*x + a0
    S = np.sum((phi - y) ** 2)

    return Result(
        coefficients = (a0, a1, a2),
        apply = lambda x: a2*x**2 + a1*x + a0,
        function = f'{a2:.4f}x² + {a1:.4f}x + {a0:.4f}',
        S = S,
        deviation = msr(phi, y),
        confidence = confidence(phi, y)
    )

# полиномиальная функция 3-й степени - done
def Approximation_degree3(x, y):
    x, y = getData(x, y)
    SX,SX2, SX3, SX4, SX5, SX6, SY,SXY, SX2Y, SX3Y = np.sum(x), np.sum(x**2),
np.sum(x**3), np.sum(x**4), np.sum(x**5), np.sum(x**6) , np.sum(y), np.sum(x*y),
np.sum(x*x*y), np.sum(x*x*x*y)
    a0, a1, a2, a3 = np.linalg.solve(
        np.array([[len(x), SX, SX2, SX3], [SX, SX2, SX3, SX4], [SX2, SX3, SX4,
SX5], [SX3, SX4, SX5, SX6]]),
        np.array([SY, SXY, SX2Y, SX3Y])
    )

     # Сумма квадратов отклонений
    phi = a3*x**3 + a2*x**2 + a1*x + a0
    S = np.sum((phi - y) ** 2)

    return Result(
        coefficients = (a0, a1, a2),
        apply = lambda x: a3*x**3 + a2*x**2 + a1*x + a0,
        function = f'{a3:.4f}x³ + {a2:.4f}x² + {a1:.4f}x + {a0:.4f}',
```

```
        S = S,
        deviation = msr(phi, y),
        confidence = confidence(phi, y)
    )

class LnException(Exception):
    pass

# логарифмическая функция - done
def Approximation_logarith(x, y):
    x, y = getData(x, y)
    if x[x < 0]:
        raise LnException('x должен быть больше чем 0')
    X = np.log(x)
    a, b = Approximation_linear(X, y).coefficients
    phi = a*np.log(x) + b
    S = np.sum((phi - y)** 2)

    return Result(
        coefficients = (a, b),
        apply = lambda x: a*math.log(x) + b,
        function = f'{a:.4f}ln(x) + {b:.4f}',
        S = S,
        deviation = msr(phi, y),
        confidence = confidence(phi, y)
    )

#  степенная функция - done
def Approximation_power(x, y):
    x, y = getData(x,y)
    X, Y = np.log(x), np.log(y)
    B, A = Approximation_linear(X, Y).coefficients
    a, b = math.exp(A), B

    phi = a*x**b
    S = np.sum((phi - y)** 2)
    return Result(
        coefficients = (a, b),
        apply = lambda x: a*math.pow(x, b),
        function = f'{a:.4f}x^{b:.4f}',
        S = S,
        deviation = msr(phi, y),
        confidence = confidence(phi, y)
    )

#  экспоненциальная функция - done
def Approximation_exp(x, y):
    x, y = getData(x, y)
    Y = np.log(y)
    B, A = Approximation_linear(x, Y).coefficients
    a, b = math.exp(A), B
```

```
    phi = a*np.exp(b*x)
    S = np.sum((phi - y)** 2)

    return Result(
        coefficients = (a, b),
        apply = lambda x: a*math.exp(b*x),
        function = f'{a:.4f}e^{b:.4f}x',
        S = S,
        deviation = msr(phi, y),
        confidence = confidence(phi, y)
    )
```

6. Результаты выполнения программы



```
Введите '+' или '-' для выбора способа ввода: +
Вывод в файл (+/-): -
--- Линейная функция
ϕ(x) = 1.6854x + 1.2168
S = 0.4730
δ = 0.2600
R² = 0.9948
r = 0.9974
--- Полиномиальная функция 2-й степени
ϕ(x) = -0.0589x² + 2.1974x + 0.3743
S = 0.0690
δ = 0.0993
R² = 0.9992
--- Полиномиальная функция 3-й степени
ϕ(x) = -0.0060x³ + 0.0191x² + 1.9119x + 0.6398
S = 0.0594
δ = 0.0921
R² = 0.9994
--- Экспоненциальная функция
ϕ(x) = 2.7309e^0.2346x
S = 10.7071
δ = 1.2368
R² = 0.9131
--- Логарифмическая функция
ϕ(x) = 5.6500ln(x) + 1.1989
S = 4.1998
δ = 0.7746
R² = 0.9520
--- Степенная функция
ϕ(x) = 2.5421x^0.8380
S = 0.1544
δ = 0.1485
R² = 0.9984
Лучше всего аппроксимирует, Полиномиальная функция 3-й степени: δ = 0.092
```

7. Выводы

В результате выполнения данной лабораторной работой я познакомился с аппроксимация функции методом наименьших квадратов и реализовал их на языке программирования Python, закрепив знания.

Аппроксимация может потребоваться, например, в случае, если из эксперимента известны лишь некоторые значения функции и требуется найти неизвестное. Или же, если изначальная функция слишком сложна для регулярного использования.

Можно выделить следующие достоинства метода: расчеты довольны просты необходимо лишь найти коэффициенты, полученная функция также проста, разнообразие возможных аппроксимирующих функций.

Основным недостатком МНК является чувствительность оценок к резким выбросам, которые встречаются в исходных данных.