Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

Высшего образования

«Национальный исследовательский университет ИТМО»

Факультет Программной Инженерии и Компьютерной Техники

Лабораторная работа 4 по вычислительной математике

Аппроксимация функции методом наименьших квадратов

Вариант №10

Преподаватель: Малышева Татьяна Алексеева

Выполнил: Состанов Тимур Айратович

Группа: P3214

Г. Санкт-Петербург

2024

Оглавление

Цель работы

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

Порядок выполнения работы

Вычислительная реализация задачи

Вычислительная часть лабораторной работы должна быть представлена только в отчете.

Задание:

1. Сформировать таблицу табулирования заданной функции на указанном интервале (см. табл. 1)
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;
6. Привести в отчете подробные вычисления.

Программная реализация задачи

Для исследования использовать:

- линейную функцию,
- полиномиальную функцию 2-й степени,
- полиномиальную функцию 3-й степени,
- экспоненциальную функцию,
- логарифмическую функцию,
- степенную функцию.

Методика проведения исследования:

1. Вычислить меру отклонения $S = \Sigma_{i=1}^{n}[\varphi(x_i) - y_i]^2$для всех исследуемых функций;
2. Уточнить значения коэффициентов эмпирических функций, минимизируя функцию S;
3. Сформировать массивы предполагаемых эмпирических зависимостей $(\varphi(x_i), \varepsilon_i)$;
4. Определить среднеквадратичное отклонение для каждой аппроксимирующей функции. Выбрать наименьшее значение и, следовательно, наилучшее приближение;
5. Построить графики полученных эмпирических функций.

Задание:

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. Вычислить коэффициент детерминации, программа должна выводить соответствующее сообщение в зависимости от полученного значения R^2;
6. Программа должна отображать наилучшую аппроксимирующую функцию;
7. Организовать вывод графиков функций, графики должны полностью отображать весь исследуемый интервал (с запасом);
8. Программа должна быть протестирована при различных наборах данных, в том числе и некорректных;

Рабочие формулы используемых методов

Линейная аппроксимация

Рассмотрим в качестве эмпирической формулы линейную функцию:

$$\varphi(x, a, b) = ax + b$$

Сумма квадратов отклонений запишется следующим образом:

$$S = S(a, b) = \sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n} [\varphi(x_i) - y_i]^2 = \sum_{i=1}^{n} (ax_i + b - y_i)^2 \to min$$

Для нахождения *a* и *b* необходимо найти минимум функции $S(a, b)$.

Необходимое условие существования минимума для функции *S:*

$$\begin{cases} \dfrac{\partial S}{\partial a} = 0 \\ \dfrac{\partial S}{\partial b} = 0 \end{cases} \quad \text{или} \quad \begin{cases} 2\sum_{i=1}^{n} (ax_i + b - y_i)x_i = 0 \\ 2\sum_{i=1}^{n} (ax_i + b - y_i) = 0 \end{cases}$$

Упростим полученную систему:

$$\begin{cases} a\sum_{i=1}^{n} x_i^2 + b\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} x_i y_i \\ a\sum_{i=1}^{n} x_i + bn = \sum_{i=1}^{n} y_i \end{cases}$$

Введем обозначения:

$$SX = \sum_{i=1}^{n} x_i, \quad SXX = \sum_{i=1}^{n} x_i^2, \quad SY = \sum_{i=1}^{n} y_i\,, \quad SXY = \sum_{i=1}^{n} x_i y_i$$

Получим систему уравнений для нахождения параметров *a* и *b:*

$$\begin{cases} aSXX + bSX = SXY \\ aSX + \ bn = SY \end{cases},$$

из которой находим (правило Крамера):

$$\Delta = SXX \cdot n - SX \cdot SX$$

$$\Delta_1 = SXY \cdot n - SX \cdot SY$$

$$\Delta_2 = SXX \cdot SY - SX \cdot SXY$$

$$a = \frac{\Delta_1}{\Delta}, \quad b = \frac{\Delta_2}{\Delta}$$

КВАДРАТИЧНАЯ АППРОКСИМАЦИЯ

Рассмотрим в качестве эмпирической формулы квадратичную функцию:

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

Сумма квадратов отклонений запишется следующим образом:

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

Приравниваем к нулю частные производные S по неизвестным параметрам, получаем систему линейных уравнений:

$$\begin{cases} \dfrac{\partial S}{\partial a_0} = 2\sum_{i=1}^{n} a_2 x_i^2 + a_1 x_i + a_0 - y_i = 0 \\ \dfrac{\partial S}{\partial a_1} = 2\sum_{i=1}^{n} (a_2 x_i^2 + a_1 x_i + a_0 - y_i) x_i = 0 \\ \dfrac{\partial S}{\partial a_2} = 2\sum_{i=1}^{n} (a_2 x_i^2 + a_1 x_i + a_0 - y_i) x_i^2 = 0 \end{cases}$$

$$\begin{cases} a_0 n + a_1 \sum_{i=1}^{n} x_i + a_2 \sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} y_i \\ a_0 \sum_{i=1}^{n} x_i + a_1 \sum_{i=1}^{n} x_i^2 + a_2 \sum_{i=1}^{n} x_i^3 = \sum_{i=1}^{n} x_i y_i \\ a_0 \sum_{i=1}^{n} x_i^2 + a_1 \sum_{i=1}^{n} x_i^3 + a_2 \sum_{i=1}^{n} x_i^4 = \sum_{i=1}^{n} x_i^2 y_i \end{cases}$$

Аппроксимация с помощью других функций

Помимо линейных зависимостей для описания результатов эксперимента используют также показательные, степенные, логарифмические функции. Эти функции легко могут быть приведены к линейному виду, после чего для определения коэффициентов аппроксимирующей функции можно использовать описанный выше алгоритм.

Аппроксимирующая функция задана степенной функцией вида:

$$\boldsymbol{\varphi(x) = ax^b}$$

Для применения метода наименьших квадратов степенная функция ***линеаризуется***:

$$\ln(\varphi(x)) = \ln(ax^b) = \ln(a) + b\ln(x)$$

Введем обозначения: Y=$\ln(\varphi(x))$; A=$\ln(a)$; B=b; X=$\ln(x)$

Получаем линейную зависимость: Y=A+BX.

После определения коэффициентов A и B вернемся к принятым ранее обозначениям:

$$\boldsymbol{a = e^{A}} \quad \boldsymbol{b = B}$$

Аппроксимация с помощью других функций

Аппроксимирующая функция задана экспоненциальной функцией вида:

$$\boldsymbol{\varphi(x) = ae^{bx}}$$

Для применения метода наименьших квадратов экспоненциальная функция линеаризуется:

$$\ln\left(\varphi(x)\right) = \ln\left(ae^{bx}\right) = \ln a + bx$$

Введем обозначения: Y=$\ln\left(\varphi(x)\right)$; A=$\ln(a)$; B=b

Получаем линейную зависимость: Y=A+Bx.

После определения коэффициентов А и В вернемся к принятым ранее обозначениям:

$$\boldsymbol{a = e^{\mathrm{A}} \quad b = B}$$

Аппроксимирующая функция задана логарифмической функцией вида:

$$\boldsymbol{\varphi(x) = aln(x) + b}$$

Аппроксимация с помощью других функций

Вид функции	Табличный X	Табличный Y
Степенная	Ln X	Ln Y
Экспоненциальная	X	Ln Y
Логарифмическая	Ln X	Y

Коэффициент корреляции

Коэффициент корреляции – это степень связи между двумя переменными.

Корреляция помогает найти ответ на два вопроса.

Во-первых, является ли связь между переменными положительной (прямо пропорциональная зависимость) или отрицательной (обратно пропорциональная зависимость).

Во-вторых, насколько сильна зависимость.

Коэффициент корреляции Пирсона позволяет определить наличие или отсутствие **линейной** связи между двумя переменными:

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

Средние значения *x* и *y*:

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} \quad \bar{y} = \frac{\sum_{i=1}^{n} y_i}{n}$$

$$\boldsymbol{-1 \leq r \leq 1}$$

Например, при помощи критерия корреляции Пирсона можно ответить на вопрос о наличии связи между температурой тела и содержанием лейкоцитов в крови при острых респираторных инфекциях, между ростом и весом пациента, между содержанием в питьевой воде фтора и заболеваемостью населения кариесом.

Вычислительная реализация задачи

Функция $y = \frac{18x}{x^4+10}$ на интервале $x \in [0{,}4], h = 0{,}4$

№	X	Y
1	0,0	0,0
2	0,4	$\frac{2250}{3133} \approx 0{,}7182$
3	0,8	$\frac{4500}{3253} \approx 1{,}3833$
4	1,2	$\frac{6750}{3773} \approx 1{,}7890$
5	1,6	$\frac{9000}{5173} \approx 1{,}7398$
6	2,0	$\frac{18}{13} \approx 1{,}3846$
7	2,4	$\frac{13500}{13493} \approx 1{,}0005$
8	2,8	$\frac{15750}{22333} \approx 0{,}7052$
9	3,2	$\frac{18000}{35893} \approx 0{,}5015$
10	3,6	$\frac{20250}{55613} \approx 0{,}3641$
11	4,0	$\frac{36}{133} \approx 0{,}2707$

Линейная аппроксимация:

$$\sum_{i=1}^{n} x_i = 22, \sum_{i=1}^{n} x_i^2 = 61.6, \sum_{i=1}^{n} y_i = 9{,}8569, \sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{x_i y_i} = 17{,}4677$$

Решим систему:

$$\begin{cases} a\sum_{i=1}^{n} x_i^2 + b\sum_{i=1}^{n} x_i = \sum_{i=l}^{n} x_i y_i \\ a\sum_{i=1}^{n} x_i + bn = \sum_{i=1}^{n} y_i \end{cases}$$

Подставим наши значения:

$$\begin{cases} a \cdot 61.6 + b \cdot 22 = 17{,}4677 \\ a \cdot 22 + b \cdot 11 = 9{,}8569 \end{cases}$$

Получаем:

$$\begin{cases} a = -0{,}1276 \\ b = 1{,}1513 \end{cases}$$

Линейная аппроксимация имеет вид:

$$\varphi(x) = ax + b = -0{,}1276 \cdot x + 1{,}1513$$

№	**X**	**Y**	$\varphi(x_i)$	ε_i
1	0,0	0,0	1,1513	1,1513
2	0,4	0,7182	1,1003	0,3821
3	0,8	1,3833	1,0492	-0,3341
4	1,2	1,7890	0,9982	-0,7908
5	1,6	1,7398	0,9471	-0,7927
6	2,0	1,3846	0,8961	-0,4885
7	2,4	1,0005	0,8451	-0,1554
8	2,8	0,7052	0,7940	0,0888
9	3,2	0,5015	0,7430	0,2415
10	3,6	0,3641	0,6919	0,3278
11	4,0	0,2707	0,6409	0,3702

Мера отклонения:

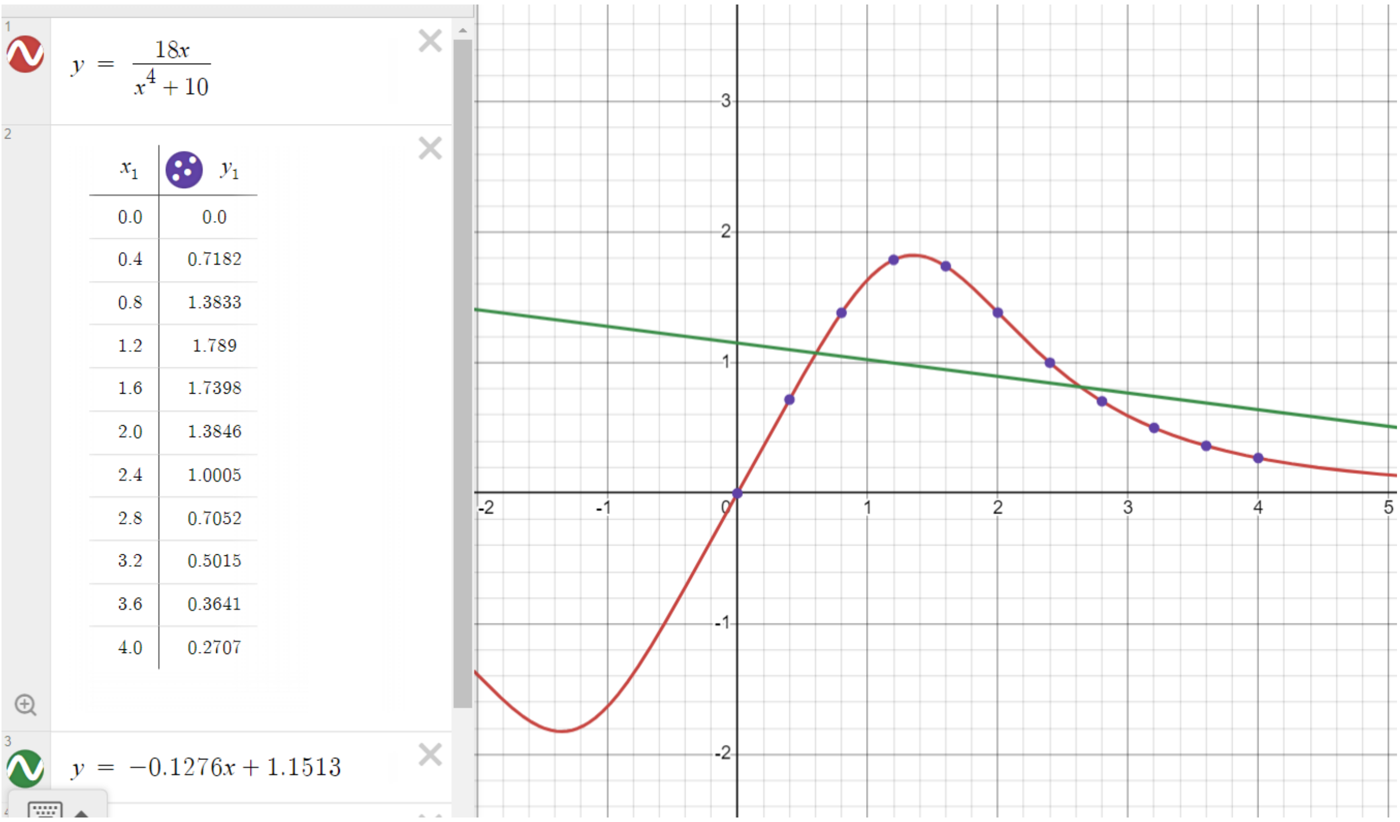
$$S = \sum_{i=1}^{n} \varepsilon_i^2 = 3{,}4103$$

Среднеквадратическое отклонение:

$$\delta = \sqrt{\frac{S}{n}} = 0{,}5568$$

Коэффициент Пирсона:

$$r = -0{,}27845$$



Квадратичная аппроксимация:

$$\sum_{i=1}^{n} x_i = 22, \sum_{i=1}^{n} x_i^2 = 61.6, \sum_{i=1}^{n} x_i^3 = 193.6, \sum_{i=1}^{n} x_i^4 = 648.5248$$

$$\sum_{i=1}^{n} y_i = 9.8569, \sum_{i=1}^{n} x_i y_i = 17.4677, \sum_{i=1}^{n} x_i^2 y_i = 39.0456$$

$$\begin{cases} a_0 n + a_1 \sum_{i=1}^{n} x_i + a_2 \sum_{i=1}^{h} x_i^2 = \sum_{i=1}^{n} y_i \\ a_0 \sum_{i=1}^{n} x_i + a_1 \sum_{i=1}^{n} x_i^2 + a_2 \sum_{i=1}^{n} x_i^3 = \sum_{i=1}^{n} x_i y_i \\ a_0 \sum_{i=1}^{n} x_i^2 + a_1 \sum_{i=1}^{n} x_i^3 + a_2 \sum_{i=1}^{n} x_i^4 = \sum_{i=1}^{n} x_i^2 y_i \end{cases}$$

$$\begin{cases} a_0 \cdot 11 + a_1 \cdot 22 + a_2 \cdot 61.6 = 9.8569 \\ a_0 \cdot 22 + a_1 \cdot 61.6 + a_2 \cdot 193.6 = 17.4677 \\ a_0 \cdot 61.6 + a_1 \cdot 193.6 + a_2 \cdot 648.5248 = 39.0456 \end{cases}$$

Получаем:

$$\begin{cases} a_0 = 0{,}3680 \\ a_1 = 1{,}1779 \\ a_2 = -0{,}3264 \end{cases}$$

Квадратичная аппроксимация имеет вид:

$$\varphi(x) = a_2 x^2 + a_1 x + a_0 = -0{,}3264 \cdot x^2 + 1{,}1779 \cdot x + 0{,}368$$

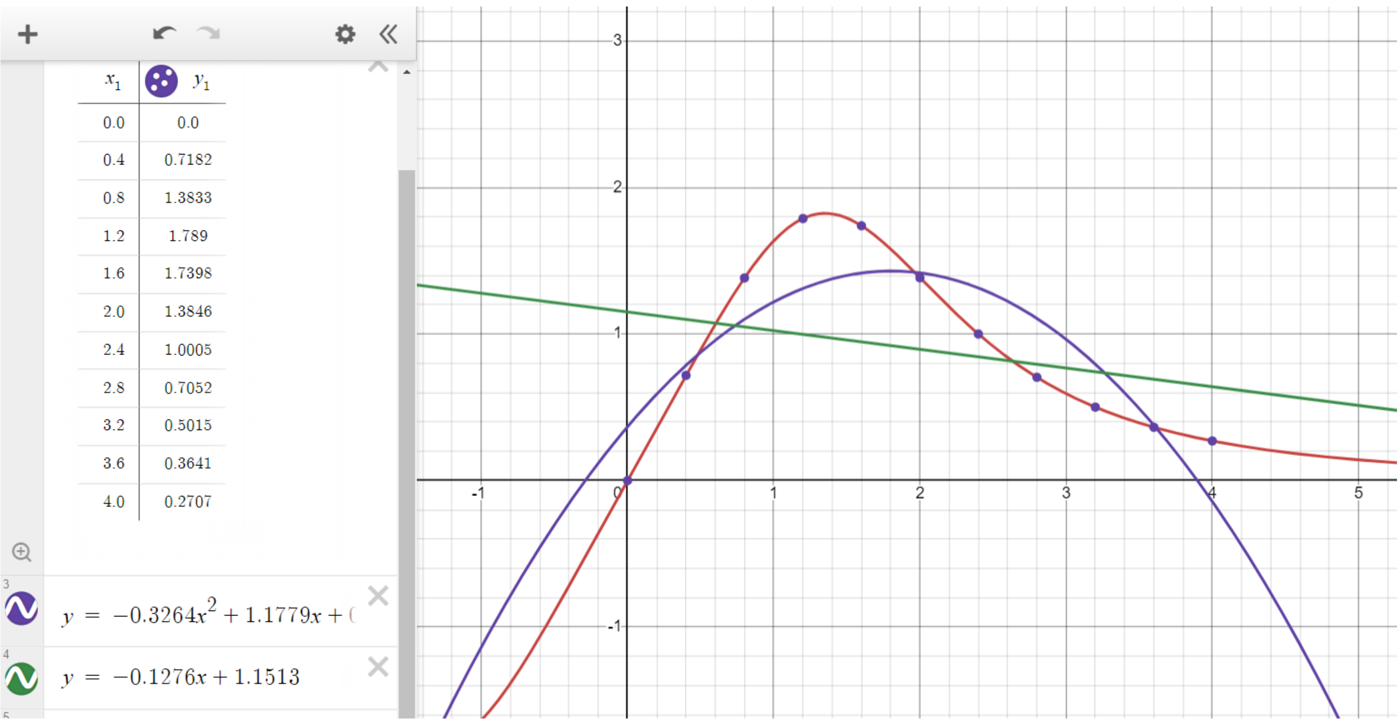
№	X	Y	$\varphi(x_i)$	ε_i
1	0.0000	0.0000	0.3680	0.3680
2	0.4000	0.7182	0.7869	0.0688
3	0.8000	1.3833	1.1014	-0.2819
4	1.2000	1.7890	1.3115	-0.4775
5	1.6000	1.7398	1.4171	-0.3227
6	2.0000	1.3846	1.4183	0.0337
7	2.4000	1.0005	1.3150	0.3145
8	2.8000	0.7052	1.1073	0.4021
9	3.2000	0.5015	0.7952	0.2937
10	3.6000	0.3641	0.3786	0.0145
11	4.0000	0.2707	-0.1424	-0.4132

Мера отклонения:

$$S = \sum_{i=1}^{n} \varepsilon_i^2 = 1{,}07063$$

Среднеквадратическое отклонение:

$$\delta = \sqrt{\frac{S}{n}} = 0{,}31198$$



Ссылка на графики; https://www.desmos.com/calculator/somsw2zkwy

При квадратичной аппроксимации среднеквадратическое отклонение меньше, следовательно данная аппроксимация дает наилучшее приближение (из рассматриваемых аппроксимирующих функциях)

Программная реализация задачи

Листнинг программы:

```python
def linear_approximation(x, y):
    n = len(x)
    SX = sum(x)
    SXX = sum(i ** 2 for i in x)
    SY = sum(y)
    SXY = sum(x[i] * y[i] for i in range(n))

    delta = n * SXX - SX ** 2
    a = (n * SXY - SX * SY) / delta
    b = (SXX * SY - SX * SXY) / delta

    func = lambda x: a * x + b
    return func, a, b
```

```python
def quadratic_approximation(x, y):
    n = len(x)
    SX = sum(x)
    SX2 = sum(p ** 2 for p in x)
    SX3 = sum(p ** 3 for p in x)
    SX4 = sum(p ** 4 for p in x)
    SY = sum(y)
    SXY = sum(x[i] * y[i] for i in range(n))
    SX2Y = sum(x[i] ** 2 * y[i] for i in range(n))

    x_matrix = np.array([[n, SX, SX2], [SX, SX2, SX3], [SX2, SX3, SX4]])
    y_vector = np.array([SY, SXY, SX2Y])
    coefficients = np.linalg.solve(x_matrix, y_vector)

    func = lambda x: coefficients[2] * x ** 2 + coefficients[1] * x + coefficients[0]

    return func, coefficients[2], coefficients[1], coefficients[0]
```

```python
def cubic_approximation(x, y):
    n = len(x)
    SX = sum(x)
    SX2 = sum(p ** 2 for p in x)
    SX3 = sum(p ** 3 for p in x)
    SX4 = sum(p ** 4 for p in x)
    SX5 = sum(p ** 5 for p in x)
    SX6 = sum(p ** 6 for p in x)
    SY = sum(y)
    SXY = sum(x[i] * y[i] for i in range(n))
    SX2Y = sum(x[i] ** 2 * y[i] for i in range(n))
    SX3Y = sum(x[i] ** 3 * y[i] for i in range(n))

    x_matrix = np.array([[n, SX, SX2, SX3], [SX, SX2, SX3, SX4], [SX2, SX3, SX4, SX5], [SX3, SX4, SX5, SX6]])
    y_vector = np.array([SY, SXY, SX2Y, SX3Y])
    coefficients = np.linalg.solve(x_matrix, y_vector)

    func = lambda x: coefficients[3] * x ** 3 + coefficients[2] * x ** 2 + coefficients[1] * x + coefficients[0]

    return func, coefficients[3], coefficients[2], coefficients[1], coefficients[0]
```

```python
def exponential_approximation(x, y):
    if not all([i > 0 for i in x]):
        return None, None, None
    y_ln = [np.log(i) for i in y]
    function, B, A = linear_approximation(x, y_ln)
    a = np.exp(A)
    b = B
    func = lambda x: a * np.exp(b * x)
    return func, a, b
```

```python
def logarithmic_approximation(x, y):
    if not all([i > 0 for i in x]):
        return None, None, None
    x_ln = [np.log(i) for i in x]
    function, a, b = linear_approximation(x_ln, y)
    func = lambda x: a * np.log(x) + b
    return func, a, b
```

```python
def power_approximation(x, y):
    if not (all([p > 0 for p in x]) and all([p > 0 for p in y])):
        return None, None, None
    x_ln = [np.log(p) for p in x]
    y_ln = [np.log(p) for p in y]
    function, B, A = linear_approximation(x_ln, y_ln)
    a = np.exp(A)
    b = B
    func = lambda x: a * x ** b
    return func, a, b
```

Результат работы программы

```
data.txt × main.py × re
1  0.01 0.01
2  0.4 0.7182
3  0.8 1.3833
4  1.2 1.789
5  1.6 1.7398
6  2.0 1.3846
7  2.4 1.0005
8  2.8 0.7052
9  3.2 0.5015
10 3.6 0.3641
11 4.0 0.2707
```

```
Выберите источник данных:
1. Из файла
2. Ввод с клавиатуры
Ваш выбор: 1
Введите имя файла с данными: data.txt
Данные из файла: [(0.01, 0.01), (0.4, 0.7182), (0.8, 1.3833), (1.2, 1.789), (1.6, 1.7398), (2.0, 1.3846), (2.4, 1.0005), (2.8, 0.7052), (3.2, 0.5015), (3.6, 0.3641), (4.0, 0.2707)]

 +-------------------------------------+
|        Линейная аппроксимация       |
+----+-------+-------+-------+--------+
| №  |   X   |   Y   |  Phi  |  eps   |
+----+-------+-------+-------+--------+
| 1  | 0.010 | 0.010 | 1.155 | 1.145  |
| 2  | 0.400 | 0.718 | 1.104 | 0.386  |
| 3  | 0.800 | 1.383 | 1.053 | -0.331 |
| 4  | 1.200 | 1.789 | 1.001 | -0.788 |
| 5  | 1.600 | 1.740 | 0.949 | -0.791 |
| 6  | 2.000 | 1.385 | 0.897 | -0.487 |
| 7  | 2.400 | 1.000 | 0.845 | -0.155 |
| 8  | 2.800 | 0.705 | 0.793 | 0.088  |
| 9  | 3.200 | 0.501 | 0.742 | 0.240  |
| 10 | 3.600 | 0.364 | 0.690 | 0.326  |
| 11 | 4.000 | 0.271 | 0.638 | 0.367  |
+----+-------+-------+-------+--------+
Коэффициент детерминации: 0.08011
Точность аппроксимации недостаточна
Коэффициент Пирсона: -0.28303
Связь слабая
```

Квадратичная аппроксимация

№	X	Y	Phi	eps
1	0.010	0.010	0.381	0.371
2	0.400	0.718	0.787	0.069
3	0.800	1.383	1.101	-0.282
4	1.200	1.789	1.311	-0.478
5	1.600	1.740	1.416	-0.324
6	2.000	1.385	1.417	0.033
7	2.400	1.000	1.314	0.314
8	2.800	0.705	1.107	0.401
9	3.200	0.501	0.795	0.293
10	3.600	0.364	0.379	0.015
11	4.000	0.271	-0.141	-0.412

Коэффициент детерминации: 0.70865

Слабая аппроксимация

Кубическая аппроксимация

№	X	Y	Phi	eps
1	0.010	0.010	-0.073	-0.083
2	0.400	0.718	0.871	0.153
3	0.800	1.383	1.434	0.050
4	1.200	1.789	1.662	-0.127
5	1.600	1.740	1.632	-0.108
6	2.000	1.385	1.420	0.036
7	2.400	1.000	1.103	0.103
8	2.800	0.705	0.758	0.053
9	3.200	0.501	0.460	-0.042
10	3.600	0.364	0.286	-0.078
11	4.000	0.271	0.314	0.043

Коэффициент детерминации: 0.97696

Высокая точность аппроксимации

Экспоненциальная аппроксимация				
№	X	Y	Phi	eps
1	0.010	0.010	0.381	0.371
2	0.400	0.718	0.410	-0.308
3	0.800	1.383	0.442	-0.941
4	1.200	1.789	0.476	-1.313
5	1.600	1.740	0.513	-1.226
6	2.000	1.385	0.553	-0.831
7	2.400	1.000	0.596	-0.404
8	2.800	0.705	0.643	-0.062
9	3.200	0.501	0.693	0.191
10	3.600	0.364	0.747	0.382
11	4.000	0.271	0.805	0.534

Коэффициент детерминации: -0.16649

Точность аппроксимации недостаточна

Логарифмическая аппроксимация				
№	X	Y	Phi	eps
1	0.010	0.010	0.428	0.418
2	0.400	0.718	0.794	0.076
3	0.800	1.383	0.863	-0.521
4	1.200	1.789	0.903	-0.886
5	1.600	1.740	0.932	-0.808
6	2.000	1.385	0.954	-0.431
7	2.400	1.000	0.972	-0.029
8	2.800	0.705	0.987	0.282
9	3.200	0.501	1.000	0.499
10	3.600	0.364	1.012	0.648
11	4.000	0.271	1.023	0.752

Коэффициент детерминации: 0.07875

Точность аппроксимации недостаточна

Степенная аппроксимация				
№	X	Y	Phi	eps
1	0.010	0.010	0.028	0.018
2	0.400	0.718	0.288	-0.431
3	0.800	1.383	0.445	-0.938
4	1.200	1.789	0.575	-1.214
5	1.600	1.740	0.690	-1.050
6	2.000	1.385	0.794	-0.591
7	2.400	1.000	0.891	-0.110
8	2.800	0.705	0.982	0.276
9	3.200	0.501	1.068	0.566
10	3.600	0.364	1.150	0.786
11	4.000	0.271	1.229	0.958

Коэффициент детерминации: -0.50269

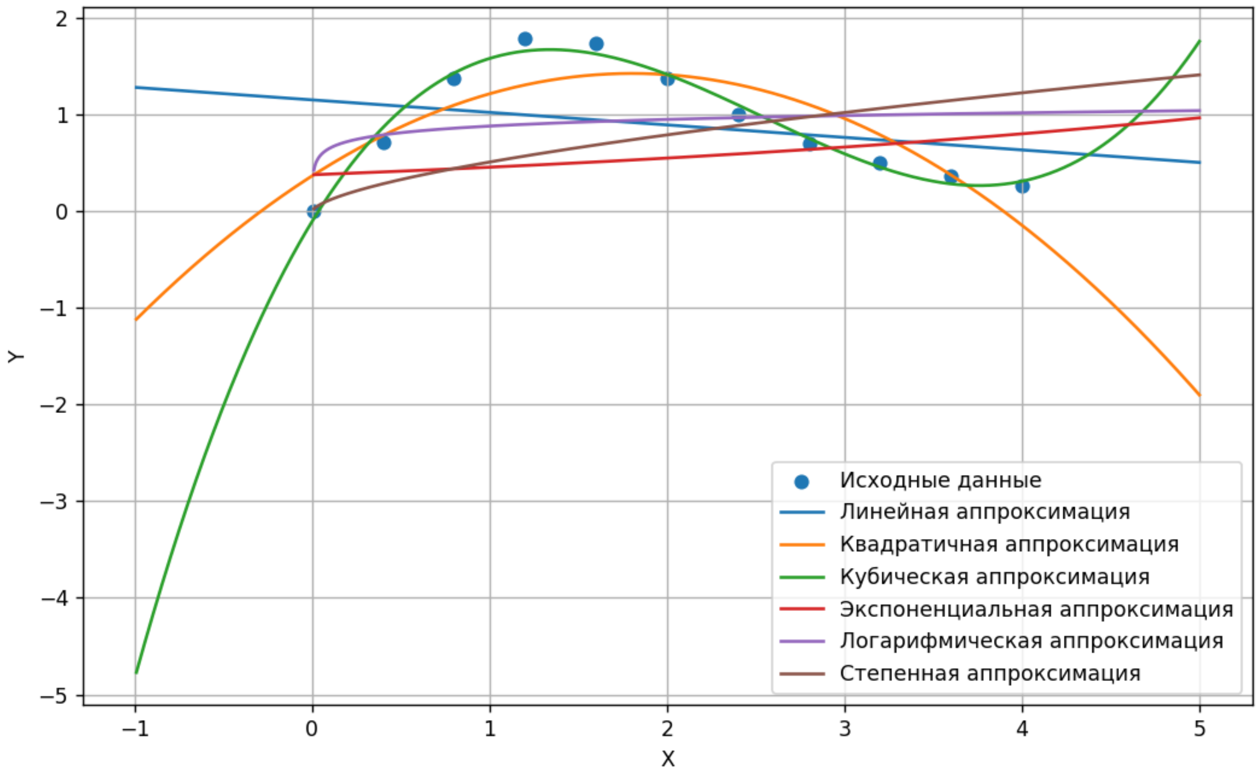
Точность аппроксимации недостаточна

Выбор аппроксимирующей функции							
Вид функции	a	b	c	d	S	stand dev	R^2
ax + b	-0.12955	1.15621	-	-	3.38438	0.55468	0.08011
ax^2 + bx + c	-0.32589	1.17592	0.36906	-	1.07192	0.31216	0.70865
ax^3 + bx^2 + cx + d	0.19917	-1.52371	3.01230	-0.10290	0.08477	0.08779	0.97696
ae^(bx)	0.38061	0.18714	-	-	5.67134	0.71804	-0.16649
alnx + b	0.09925	0.88493	-	-	3.38939	0.55509	0.07875
ax^b	0.51265	0.63082	-	-	5.93659	0.73464	-0.50269

Введите название файла для записи ответа: result.txt

Наилучшая аппроксимирующая функция: Кубическая data.txt | main.py | result.txt | scratch.py

```
Наилучшая аппроксимирующая функция: Кубическая
Коэффициенты аппроксимирующей функции: [0.19916747831739573, -1.5237100273163102, 3.01229974288795, -0.10290466260833266]
Среднеквадратичное отклонение: 0.08778819017421949
Коэффициент детерминации: 0.9769578948159292
```



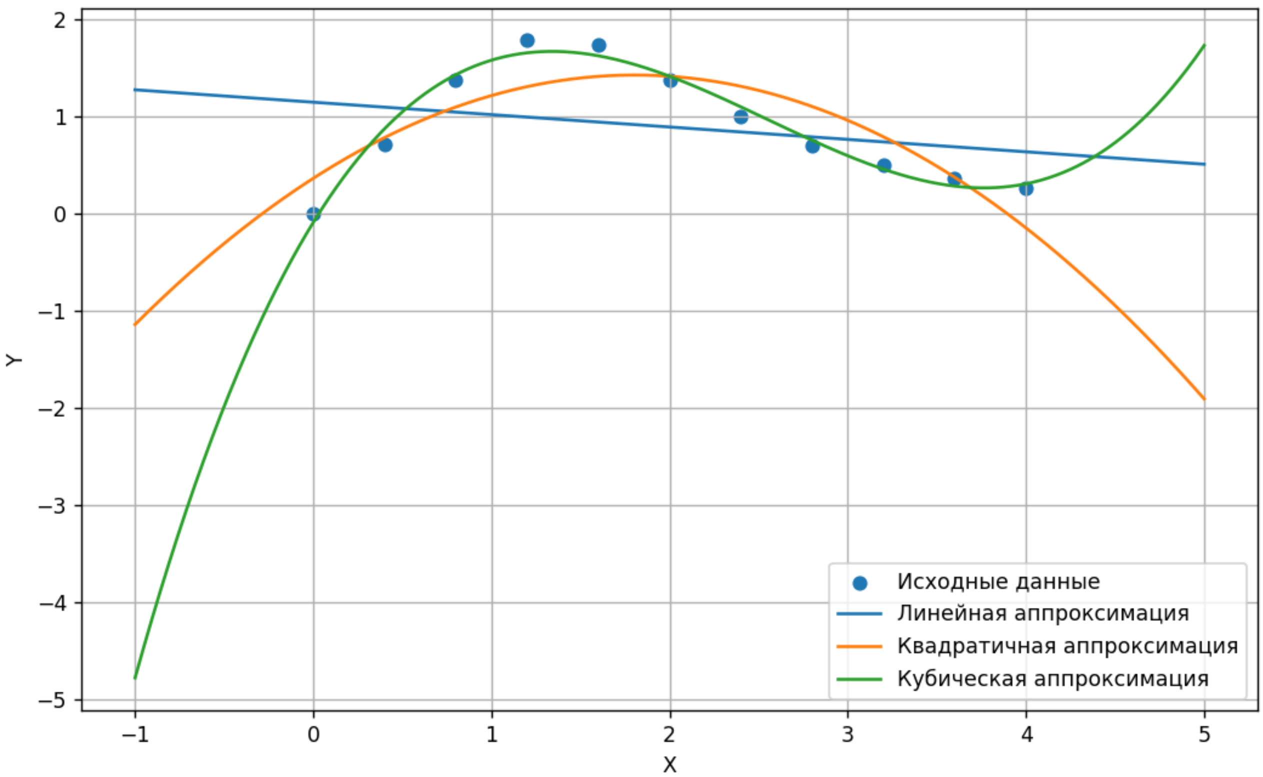
Немного изменим исходные данные

data.txt | main.py

```
0.0 0.0
0.4 0.7182
0.8 1.3833
1.2 1.789
1.6 1.7398
2.0 1.3846
2.4 1.0005
2.8 0.7052
3.2 0.5015
3.6 0.3641
4.0 0.2707
```

(Таблички для каждого отдельного метода практически совпадают)

```
 +------------------------------------------------------------------------------------------------+
|                                  Выбор аппроксимирующей функции                                  |
+----------------------+----------+----------+---------+----------+---------+----------+---------+
|      Вид функции     |    a     |    b     |    c    |    d     |    S    | stand dev |   R^2   |
+----------------------+----------+----------+---------+----------+---------+----------+---------+
|        ax + b        | -0.12762 | 1.15132  |    -    |    -     | 3.41030 |  0.55680  | 0.07753 |
|    ax^2 + bx + c     | -0.32637 | 1.17787  | 0.36802 |    -     | 1.07063 |  0.31198  | 0.71040 |
| ax^3 + bx^2 + cx + d | 0.19705  | -1.50868 | 2.98128 | -0.08598 | 0.08812 |  0.08950  | 0.97616 |
+----------------------+----------+----------+---------+----------+---------+----------+---------+
Введите название файла для записи ответа: answer.txt
Наилучшая аппроксимирующая функция: Кубическая
```



Исходный код: https://github.com/tsostanov/CompMath4

Вывод

В ходе работы были изучены различные виды аппроксимации функции, использующие метод наименьших квадратов (в конечном счете сводящиеся к нему). Были решены задачи аппроксимации вычислительным и программным метода