Федеральное государственное автономное образовательное учреждение высшего образования Национальный исследовательский университет ИТМО

Факультет программной инженерии и компьютерной техники

Вычислительная математика

Лабораторная работа №4

Вариант № 1

Выполнил: студент группы P3208,
Васильев Н. А.

Преподаватель: Машина Е.А.

Санкт-Петербург 2025

Текст задания

Аппроксимация функции методом наименьших квадратов.

Цель работы

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

Описание метода, расчётные формулы

$$S = \sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n} [\varphi(x_i) - y_i]^2 \rightarrow min$$

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

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

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

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

Вычислительная реализация задачи:

Функция: $y = \frac{12x}{x^4+1}$

Исследуемый интервал: $x \in [0, 2]$; $h = 0,2$

i	1	2	3	4	5	6	7	8	9	10	11
x_i	0	0,2	0,4	0,6	0,8	1	1,2	1,4	1,6	1,8	2
y_i	0	2,396	4,680	6,374	6,810	6	4,685	3,470	2,542	1,879	1,412

Линейное приближение:

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

Введем обозначения:

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

Получаем:

$$SX = 11;\;\; SXX = 15{,}4;\;\; SY = 40{,}248;\;\; SXY = 38{,}376$$

Тогда:

$$\begin{cases} aSXX + bSX = SXY \\ aSX + b \times 11 = SY \end{cases} = \begin{cases} 15{,}4a + 11b = 38{,}376 \\ 11a + 11b = 40{,}248 \end{cases} \to \begin{cases} a = -0{,}425 \\ b = 4{,}084 \end{cases}$$

Вычислим значения аппроксимирующей функции:

$$\varphi(x) = -0{,}425x + 4{,}084$$

i	1	2	3	4	5	6	7	8	9	10	11
x_i	0	0,2	0,4	0,6	0,8	1	1,2	1,4	1,6	1,8	2
y_i	0	2,396	4,680	6,374	6,810	6	4,685	3,470	2,542	1,879	1,412
$\varphi(x_i)$	4,084	3,999	3,914	3,829	3,744	3,659	3,574	3,489	3,404	3,319	3,234
ε_i	4,084	1,603	−0,766	−2,545	−3,066	−2,341	−1,111	0,019	0,862	1,44	1,822

$$\delta = \sqrt{\frac{\sum_{i=1}^{n}(\varphi(x_i) - y_i)^2}{n}} = 2{,}101$$

Квадратичное приближение:

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

Введем обозначения:

$$SX = \sum_{i=1}^{n} x_i\,;\;\; SXX = \sum_{i=1}^{n} x_i^2\,;\; SXXX = \sum_{i=1}^{n} x_i^3\,;\; SXXX = \sum_{i=1}^{n} x_i^4\,;\; SY = \sum_{i=1}^{n} y_i\,;\;\; SXY = \sum_{i=1}^{n} x_i y_i\,;\;\; SXXY = \sum_{i=1}^{n} x_i^2 y_i$$

Получаем:

$$SX = 11;\;\; SXX = 15{,}4;\; SXXX = 24{,}2; SXXXX = 40{,}533;\;\; SY = 40{,}248;\;\; SXY = 38{,}376;\; SXXY = 45{,}287$$

Тогда:

$$\begin{cases} na_0 + a_1 SX + a_2 SXX = SY \\ a_0 SX + a_1 SXX + a_2 SXXX = SXY \\ a_0 SXX + a_1 SXXX + a_2 SXXXX = SXXY \end{cases} = \begin{cases} 11a_0 + 11a_1 + 15{,}4a_2 = 40{,}248 \\ 11a_0 + 15{,}4a_1 + 24{,}2a_2 = 38{,}376 \\ 15{,}4a_0 + 24{,}2a_1 + 40{,}533a_2 = 45{,}287 \end{cases} \to \begin{cases} a_0 = 0{,}887 \\ a_1 = 10{,}232 \\ a_2 = -5{,}329 \end{cases}$$

$$\varphi(x) = 0{,}887 + 10{,}232x - 5{,}329x^2$$

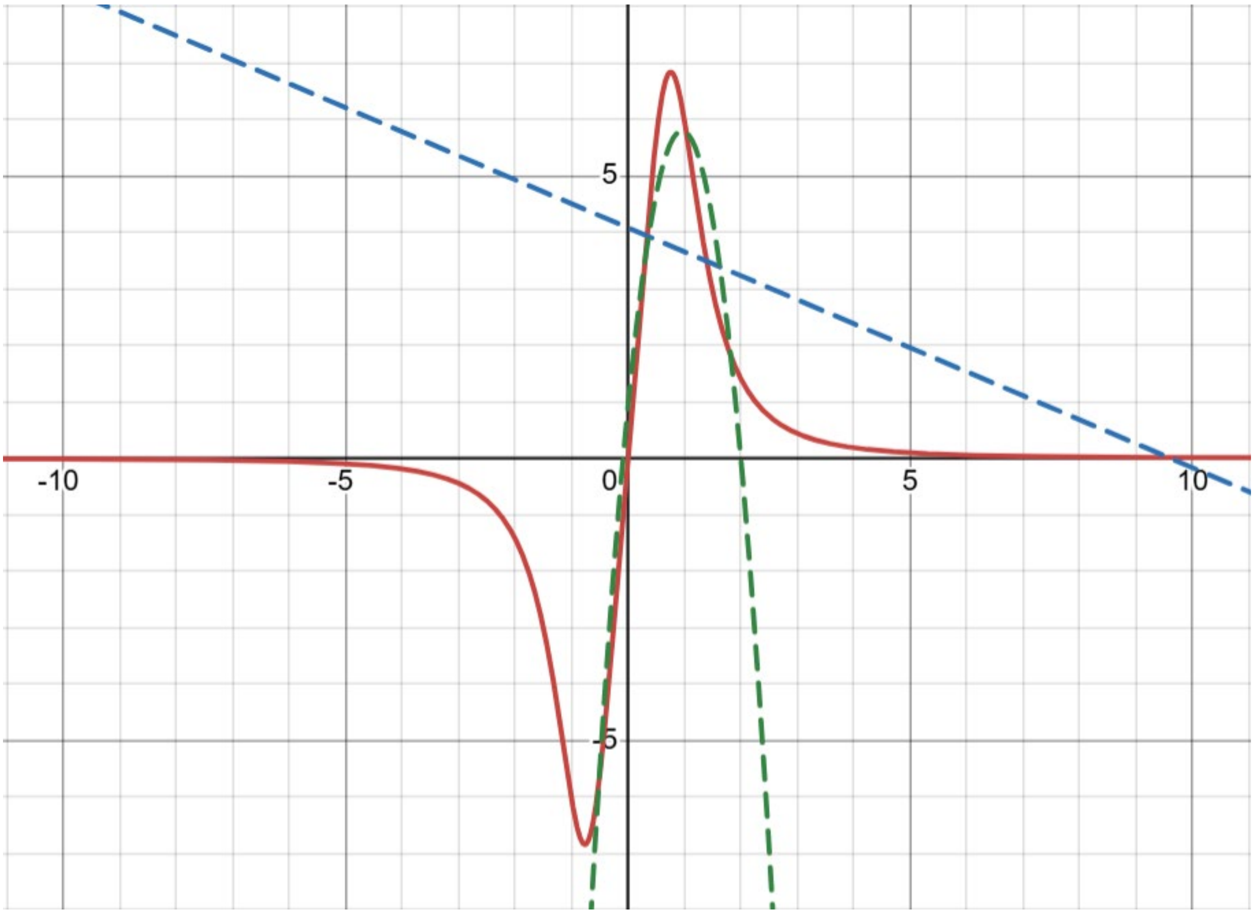
i	1	2	3	4	5	6	7	8	9	10	11
x_i	0	0,2	0,4	0,6	0,8	1	1,2	1,4	1,6	1,8	2
y_i	0	2,396	4,680	6,374	6,810	6	4,685	3,470	2,542	1,879	1,412
$\varphi(x_i)$	0,887	0,324	−0,553	−1,266	−1,148	−0,210	0,807	1,297	1,074	0,160	−1,377
ε_i	0,787	0,105	0,306	1,603	1,319	0,044	0,651	1,682	1,154	0,026	1,895

$$\delta = \sqrt{\frac{\sum_{i=1}^{n}(\varphi(x_i) - y_i)^2}{n}} = 0,933$$

Сравним полученные результаты:

$$0,933 < 2,101$$

Следовательно, квадратичное приближение лучше.



Листинг программы

```
import numpy as np


def linear_approximation(x, y, n):
    x = np.array(x)
    y = np.array(y)

    sx = np.sum(x)
```

```python
    sxx = np.sum(x ** 2)
    sy = np.sum(y)
    sxy = np.sum(x * y)

    a, b = np.linalg.solve(
        [
            [n, sx],
            [sx, sxx]
        ],
        [sy, sxy])
    return lambda xi: a + b * xi, a.item(), b.item()


def quadratic_approximation(x, y, n):
    x = np.array(x)
    y = np.array(y)

    sx = np.sum(x)
    sxx = np.sum(x ** 2)
    sxxx = np.sum(x ** 3)
    sxxxx = np.sum(x ** 4)
    sy = np.sum(y)
    sxy = np.sum(x * y)
    sxxy = np.sum(x ** 2 * y)

    a, b, c = np.linalg.solve(
        [
            [n, sx, sxx],
            [sx, sxx, sxxx],
            [sxx, sxxx, sxxxx]
        ],
        [sy, sxy, sxxy]
    )
    return lambda xi: a + b * xi + c * xi ** 2, a.item(), b.item(), c.item()


def cubic_approximation(x, y, n):
    x = np.array(x)
    y = np.array(y)

    sx = np.sum(x)
    sy = np.sum(y)
    sxy = np.sum(x * y)
    sxx = np.sum(x ** 2)
    sxxx = np.sum(x ** 3)
    sxxxx = np.sum(x ** 4)
    sxxxxx = np.sum(x ** 5)
    sxxxxxx = np.sum(x ** 6)
    sxxy = np.sum(x ** 2 * y)
    sxxxy = np.sum(x ** 3 * y)

    a, b, c, d = np.linalg.solve(
        [
            [n, sx, sxx, sxxx],
            [sx, sxx, sxxx, sxxxx],
```

```python
            [sxx, sxxx, sxxxx, sxxxxx],
            [sxxx, sxxxx, sxxxxx, sxxxxxx]
        ],
        [sy, sxy, sxxy, sxxxy]
    )
    return lambda xi: a + b * xi + c * xi ** 2 + d * xi ** 3, a.item(),
b.item(), c.item(), d.item()

def exponential_approximation(x, y, n):
    y_log = np.log(y)
    _, a_log, b_log = linear_approximation(x, y_log, n)

    a = np.exp(a_log)
    b = b_log

    return lambda xi: a * np.exp(b * xi), a.item(), b


def logarithmic_approximation(x, y, n):
    x_log = np.log(x)
    _, a, b = linear_approximation(x_log, y, n)

    return lambda xi: a + b * np.log(xi), a, b


def power_approximation(x, y, n):
    x_log = np.log(x)
    y_log = np.log(y)

    _, a_log, b_log = linear_approximation(x_log, y_log, n)

    a = np.exp(a_log)
    b = b_log

    return lambda xi: a * xi ** b, a.item(), b


import numpy as np


def compute_pearson_correlation(x, y):
    av_x = np.mean(x)
    av_y = np.mean(y)

    numerator = np.sum((x - av_x) * (y - av_y))
    denominator = np.sqrt(np.sum((x - av_x) ** 2) * np.sum((y - av_y) ** 2))

    return numerator / denominator


def compute_mean_squared_error(x, y, phi):
    x = np.array(x)
    y = np.array(y)

    errors = phi(x) - y
```

```
    mse = np.mean(errors ** 2)
    return np.sqrt(mse)


def compute_measure_of_deviation(x, y, phi):
    x = np.array(x)
    y = np.array(y)

    deviations = phi(x) - y
    return np.sum(deviations ** 2)


def compute_coefficient_of_determination(x, y, phi, n):
    x = np.array(x)
    y = np.array(y)

    av_phi = np.sum(phi(x)) / n
    total_variation = np.sum((y - av_phi) ** 2)
    unexplained_variation = np.sum((y - phi(x)) ** 2)

    return 1 - (unexplained_variation / total_variation)
```

Примеры и результаты работы программы

Пример 1: Точки: (0.2; 2.396), (0.4; 4.68), (0.6; 6.374), (0.8; 6.810), (1; 6), (1.2; 4.685), (1.4; 3.47), (1.6; 2.542), (1.8; 1.879), (2; 1.412):

Линейная аппроксимация:

Коэффициенты: [5.9901, -1.7866]

Среднеквадратичное отклонение: 1.55241

Коэффициент детерминации: 0.30414

Мера отклонения: S = 24.09984

Коэффициент корреляции Пирсона: r = -0.5514913913498515

Полиноминальная 2-й степени аппроксимация:

Коэффициенты: [2.1126, 7.9071, -4.4063]

Среднеквадратичное отклонение: 0.87738

Коэффициент детерминации: 0.77773

Мера отклонения: S = 7.69802

Полиноминальная 3-й степени аппроксимация:

Коэффициенты: [-2.0762, 26.4832, -24.5446, 6.1025]

Среднеквадратичное отклонение: 0.18335

Коэффициент детерминации: 0.99029

Мера отклонения: S = 0.33616

Экспоненциальная аппроксимация:

Коэффициенты: [6.4761, -0.5459]

Среднеквадратичное отклонение: 1.72108

Коэффициент детерминации: 0.16569

Мера отклонения: S = 29.62102

Логарифмическая аппроксимация:

Коэффициенты: [3.9504, -0.7519]

Среднеквадратичное отклонение: 1.78604

Коэффициент детерминации: 0.07894

Мера отклонения: S = 31.89952

Степенная аппроксимация:

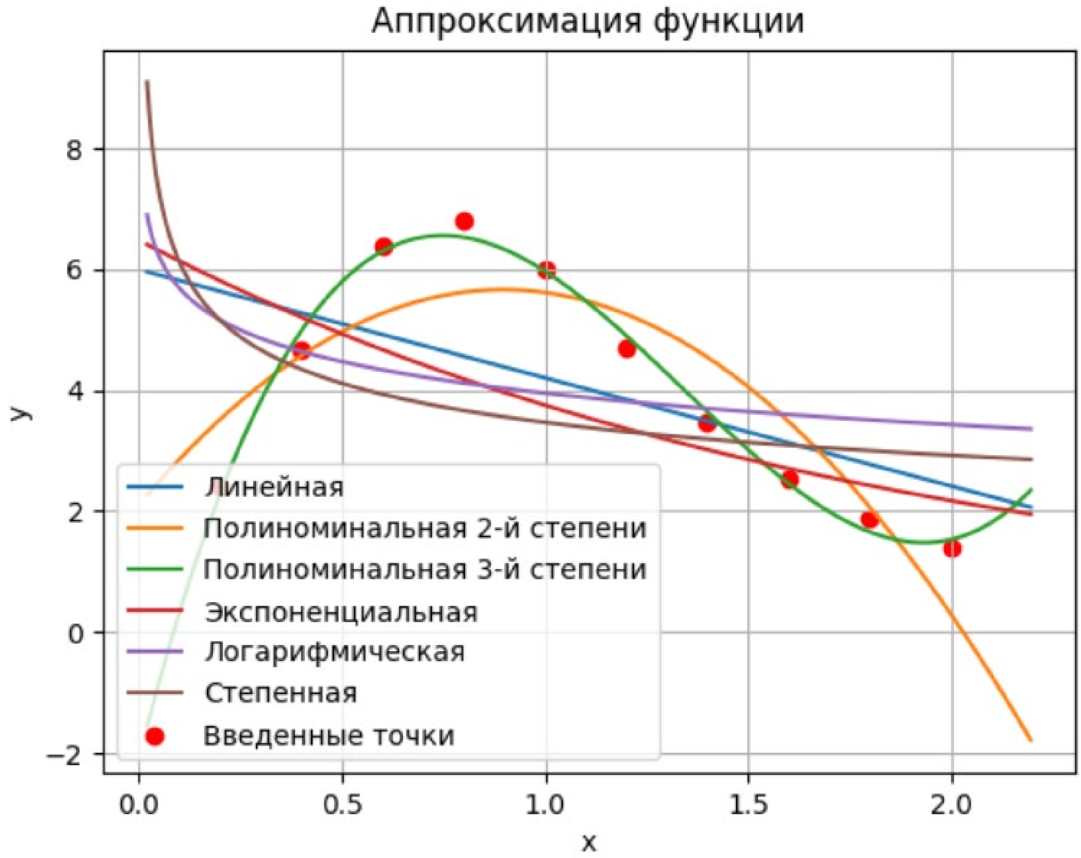
Коэффициенты: [3.467, -0.2463]

Среднеквадратичное отклонение: 1.89193

Коэффициент детерминации: 0.01593

Мера отклонения: S = 35.79388

Лучшая функция приближения: Полиноминальная 3-й степени



Вывод

В ходе выполнения лабораторной работы была реализована аппроксимация таблично заданной функции с использованием метода наименьших квадратов. Были подобраны коэффициенты приближающей функции заданного вида (линейной, квадратичной), минимизирующей сумму квадратов отклонений между значениями аппроксимирующей функции и исходными данными.

Результатом работы стала функция, наилучшим образом аппроксимирующая исходные данные в смысле наименьших квадратов. Полученные значения коэффициентов и графическое сравнение аппроксимирующей функции с табличными данными подтвердили адекватность выбранной модели.

Таким образом, была достигнута поставленная цель — построена аппроксимирующая функция, максимально приближённая к заданным данным по критерию наименьших квадратов.

Была реализована программа, позволяющая вычислять аппроксимацию для функции по заданным точкам.