Санкт-Петербургский Государственный университет ИТМО
Факультет програмной инженерии и компьютерной техники

Лабораторная работа №4 по дисциплине "Вычислительная математика"

Аппроксимация функции методом наименьших квадратов

Вариант №7

Работу выполнила:
Д. А. Карасева
Группа: P3217
Преподаватель:
Т. А. Малышева

Санкт-Петербург
2024

Содержание

1. Цель работы 3

2. Описание методов. Расчетные формулы 4

3. Вычислительная часть 6

4. Листинг программы 8

5. Примеры и результаты работы программы 9

6. Вывод 11

1. Цель работы

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

Предусмотреть ввод исходных данных из файла/консоли (таблица должна содержать от 8 до 12 точек); Реализовать метод наименьших квадратов, исследуя все указанные функции; Предусмотреть вывод результатов в файл/консоль: коэффициенты аппроксимирующих функций, среднеквадратичное отклонение, массивы значений; Для линейной зависимости вычислить коэффициент корреляции Пирсона; Вычислить коэффициент детерминации, программа должна выводить соответствующее сообщение в зависимости от полученного значения R; Программа должна отображать наилучшую аппроксимирующую функцию; Организовать вывод графиков функций, графики должны полностью отображать весь исследуемый интервал (с запасом); Программа должна быть протестирована при различных наборах данных, в том числе и некорректных;

2. Описание методов. Расчетные формулы

Метод наименьших квадратов

Задача заключается в нахождении коэффициентов линейной зависимости, при которых функция двух переменных a и b $F(a,b) = \sum_{i=1}^{n}(y_i-(ax_i+b))^2$ принимает наименьшее значение. То есть, при данных a и b сумма квадратов отклонений экспериментальных данных от найденной прямой будет наименьшей. В этом вся суть метода наименьших квадратов.

Таким образом, решение примера сводится к нахождению экстремума функции двух переменных.

Метод наименьших квадратов с точки зрения линейной алгебры и матриц. Цитируется "INTRODUCTION TO LINEAR ALGEBRA" GILBERT STRANG (библиотека в моем коде считает МНК именно так)

Решение по методу наименьших квадратов одномерной задачи ax = b имеет вид

$$\overline{x} = \frac{a^T b}{a^T a}$$

Геометрически это решение совпадает с проекцией: p =$\overline{x}$a является точкой на прямой, определенной вектором a, ближайшей к b.

Решение по методу наименьших квадратов для несовместной системы Ax=b, состоящей из m уравнений с n неизвестными, удовлетворяет соотношению

$$A^T A\overline{x} = A^T b$$

Оно известно под названием "нормальных уравнений". Если столбцы матрицы A являются линейно независимымы, то матрица $A^T A$ обратима и единственное решение дается формулой

$$\overline{x} = (A^T A)^{-1} A^T b$$

.

Проекция точки b на пространство столбцов, таким образом, имеет вид

$$p = A\overline{x} = A(A^T A)^{-1} A^T b$$

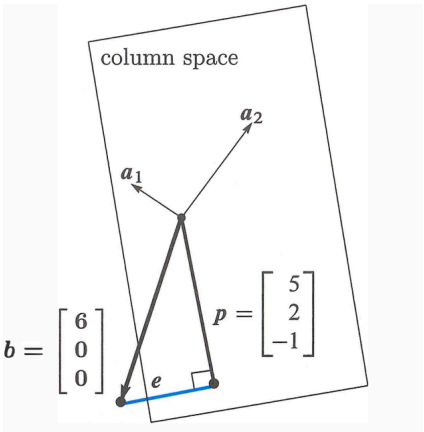
column space
a_2
a_1
$b = \begin{bmatrix} 6 \\ 0 \\ 0 \end{bmatrix}$
$p = \begin{bmatrix} 5 \\ 2 \\ -1 \end{bmatrix}$
e

Рисунок 2.1

3. Вычислительная часть

$$y = \frac{23x}{x^4 + 7}, x \in [-2; 0], h = 0.2$$

Сформировать таблицу табулирования заданной функции на указанном интервале

i	1	2	3	4	5	6	7	8	9	10	11
x_i	-2	-1.8	-1.6	-1.4	-1.2	-1	-0.8	-0.6	-0.4	-0.2	0
y_i	-2	-2.36	-2.71	-2.97	-3.04	-2.875	-2.48	-1.93	-1.30	-0.65	0

Линейная аппроксимация: $\phi(x) = a + bx$. Вычислим суммы SX = -11, SXX = 15.4, SY = -22.3, SXY = 27.08

$$\begin{cases} n * a + SX * b = SY \\ SX * a + SXX * b = SXY \end{cases} \begin{cases} 11 * a - 11 * b = -22.3 \\ -11 * a + 15.4 * b = 27.08 \end{cases} \begin{cases} a = -0.94 \\ b = 1.086 \end{cases}$$

$$\phi(x) = -0.94 + 1.086x$$

i	1	2	3	4	5	6	7	8	9	10	11
x_i	-2	-1.8	-1.6	-1.4	-1.2	-1	-0.8	-0.6	-0.4	-0.2	0
y_i	-2	-2.36	-2.71	-2.97	-3.04	-2.87	-2.48	-1.93	-1.30	-0.65	0
$\phi(x_i)$	-3,11	-2,89	-2,67	-2,46	-2,24	-2,026	-1,80	-1,59	-1,37	-1,15	-0,94
$(\phi(x_i) - y_i)^2$	1,23	0,27	0,0014	0,25	0,63	0,72	0,45	0,118	0,0042	0,25	0,88

$$\sigma = \sqrt{\frac{\sum((\phi(x_i) - y_i)^2)}{n}} = 0.66$$

Квадратичная аппроксимация: $\phi(x) = a + bx + cx^2$. Вычислим суммы SX = -11, SXX = 15.4, SXXX = -24.2, SXXXX = 40.53, SY = -22.3, SXY = 27.08, SXXY = -40.75

$$\begin{cases} n * a + SX * b + SXX * c = SY \\ SX * a + SXX * b + SXXX * c = SXY \\ SXX * a + SXXX * b + SXXXX * c = SXXY \end{cases} \begin{cases} 11 * a - 11 * b + 15.4 * c = -22.3 \\ -11 * a + 15.4 * b + -24.2 * c = 27.08 \\ 15.4 * a + -24.2 * b + 40.53 * c = -40.75 \end{cases}$$

$$\begin{cases} a = -0.92 \\ b = 1.13 \\ c = 0.02 \end{cases}$$

$$\phi(x) = -0.92 + 1.13x + 0.02x^2$$

i	1	2	3	4	5	6	7	8	9	10	11
x_i	-2	-1.8	-1.6	-1.4	-1.2	-1	-0.8	-0.6	-0.4	-0.2	0
y_i	-2	-2.36	-2.71	-2.97	-3.04	-2.87	-2.48	-1.93	-1.30	-0.65	0
$\phi(x_i)$	-3,1	-2,88	-2,67	-2,46	-2,24	-2,03	-1,81	-1,59	-1,36	-1,14	-0,92
$(\phi(x_i) - y_i)^2$	1,21	0,27	0,0014	0,25	0,63	0,71	0,45	0,11	0,0035	0,23	0,84

$$\sigma = \sqrt{\frac{\sum((\phi(x_i) - y_i)^2)}{n}} = 0.65$$

Среднеквадратичное отклонение квадратичной аппроксимации меньше, соответственно приближение лучше.

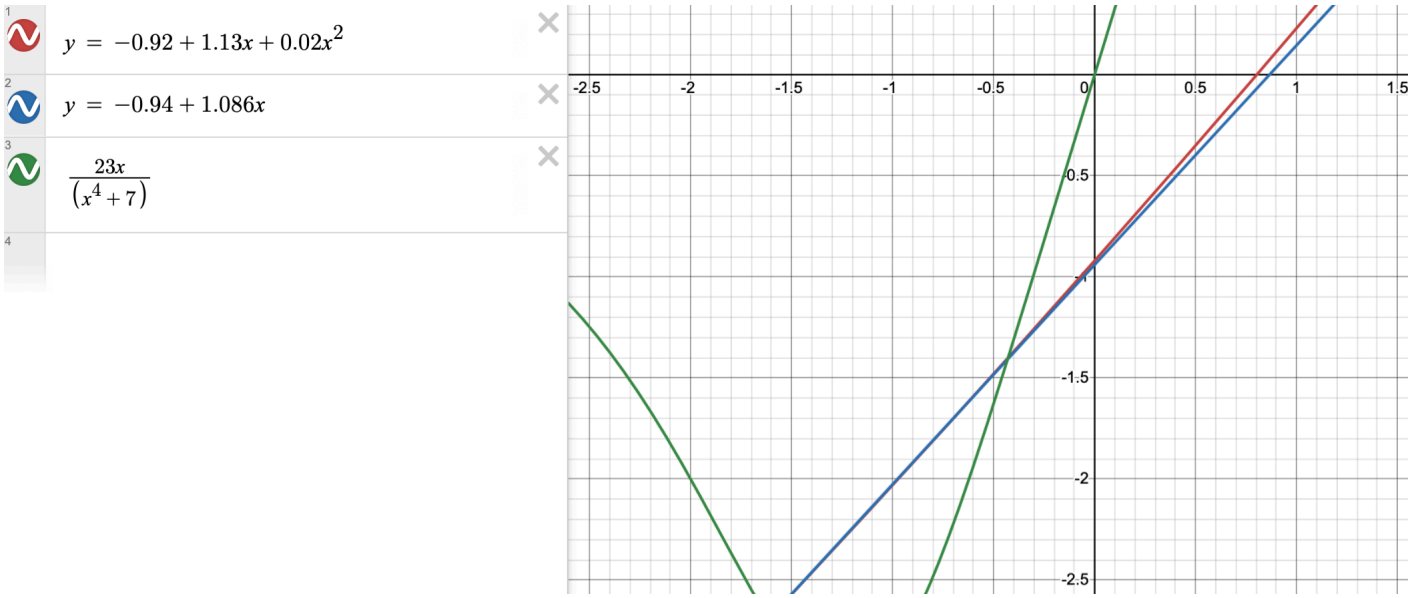

Рисунок 3.1

4. Листинг программы

Репозиторий на GitHub

Функция, отвечающая за реализацию метода наименьших квадратов

```
def least_squares_fit(x, y, function):
    n = len(x)
    if function == linear_function:
        # Linear function: y = ax + b
        A = np.vstack([x, np.ones(n)]).T
        coeffs, _, _, _ = np.linalg.lstsq(A, y, rcond=None)
        a, b = coeffs
        f_fitted = linear_function(x, a, b)
        # print(A)
    elif function == quadratic_function:
        # Quadratic function: y = ax^2 + bx + c
        A = np.vstack([x ** 2, x, np.ones(n)]).T
        coeffs, _, _, _ = np.linalg.lstsq(A, y, rcond=None)
        a, b, c = coeffs
        f_fitted = quadratic_function(x, a, b, c)
    elif function == cubic_function:
        # Cubic function: y = ax^3 + bx^2 + cx + d
        A = np.vstack([x ** 3, x ** 2, x, np.ones(n)]).T
        coeffs, _, _, _ = np.linalg.lstsq(A, y, rcond=None)
        a, b, c, d = coeffs
        f_fitted = cubic_function(x, a, b, c, d)
    elif function == exponential_function:
        # Exponential function: y = a * exp(bx)
        A = np.vstack([np.exp(x), np.ones(n)]).T
        coeffs, _, _, _ = np.linalg.lstsq(A, y, rcond=None)
        a, b = coeffs
        f_fitted = exponential_function(x, a, b)
    elif function == logarithmic_function:
        # Logarithmic function: y = a * log(x) + b
        A = np.vstack([np.log(x), np.ones(n)]).T
        coeffs, _, _, _ = np.linalg.lstsq(A, y, rcond=None)
        a, b = coeffs
        f_fitted = logarithmic_function(x, a, b)
    elif function == power_function:
        # Power function: y = a * x^b
        A = np.vstack([x, np.log(x)]).T
        coeffs, _, _, _ = np.linalg.lstsq(A, np.log(y), rcond=None)
        a, b = coeffs
        f_fitted = power_function(x, np.exp(a), b)
        print(A)
    else:
        raise ValueErrorНеверный(" типфункции ")
    rms_error = np.sqrt(np.mean((y - f_fitted) ** 2))
    return [float(i) for i in coeffs], rms_error
```

5. Примеры и результаты работы программы

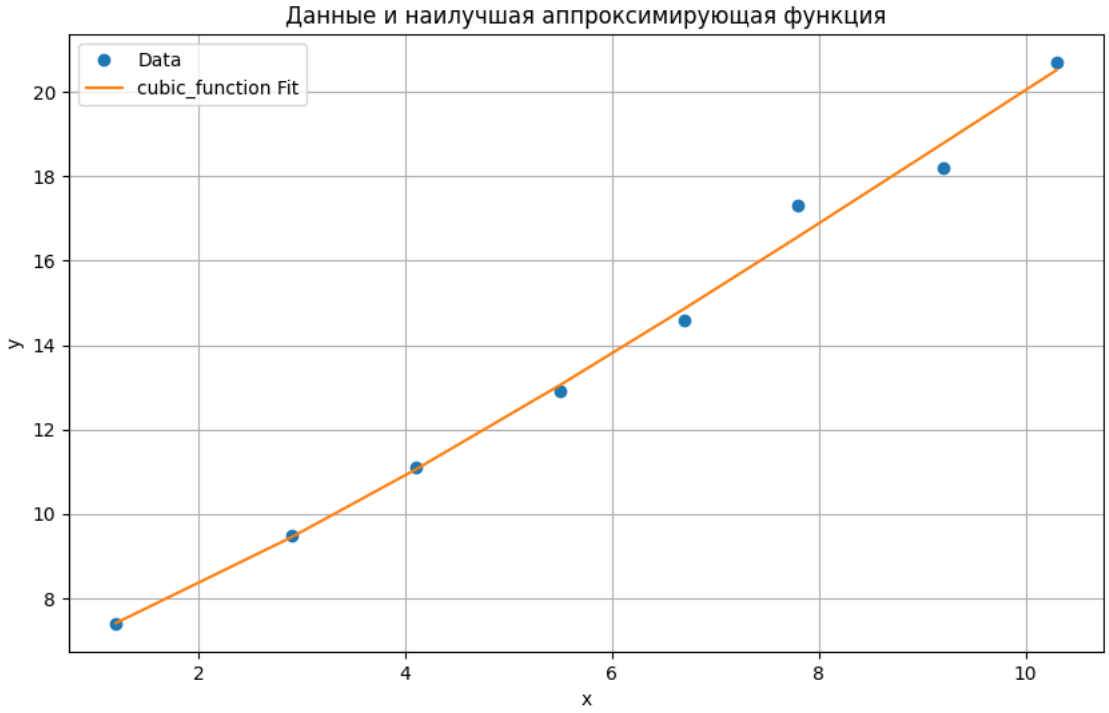

Рисунок 5.1

Введите название файла (или оставьте пустым для консольного ввода): 1.txt
[[1.2 0.18232156]
[2.9 1.06471074]
[4.1 1.41098697]
[5.5 1.70474809]
[6.7 1.90210753]
[7.8 2.05412373]
[9.2 2.21920348]
[10.3 2.3321439]]
---------- Результаты ----------
Функция: $\text{linear}_{f}unction$
Коэффициенты: [1.4543295810901444, 5.291059872750014]
Среднеквадратичная ошибка: 0.41016067346312185
Функция: $\text{quadratic}_{f}unction$
Коэффициенты: [0.025973674183265644, 1.15256305863677, 5.943053107110677]
Среднеквадратичная ошибка: 0.35635136003661705
Функция: $\text{cubic}_{f}unction$
Коэффициенты: [-0.0023713802938570587, 0.06687472776413161, 0.9547538603232554, 6.177878914311304]
Среднеквадратичная ошибка: 0.35348037500486135
Функция: $\text{exponential}_{f}unction$
Коэффициенты: [0.00033151914967839203, 12.171911938496969]

Среднеквадратичная ошибка: 3.2867280713001745e+50
Функция: logarithmic$_{function}$
Коэффициенты: [6.008624101281966, 4.295866104732609]
Среднеквадратичная ошибка: 1.528592655161903
Функция: power$_{function}$
Коэффициенты: [0.8457308598240646, 2.0745143764766403]
Среднеквадратичная ошибка: 42.99491458916805

Наилучшая аппроксимирующая функция: cubic$_{function}$
Коэффициенты: [-0.0023713802938570587, 0.06687472776413161, 0.9547538603232554, 6.177878914311304]
Среднеквадратичная ошибка: 0.35348037500486135

6. Вывод

В результате выполнения лабораторной работы я изучила алгоритм аппроксимация функции методом наименьших квадратов, смогла программно реализовать его на языке Python.