Министерства науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования «Национальный исследовательский университет
ИТМО»

Факультет программной инженерии и компьютерной техники

Вычислительная математика
Лабораторная работа № 4
Вариант: 3

Выполнил:
Вильданов Ильнур Наилевич
Группа №P3212

Проверила:
Машина Екатерина Алексеевна

Санкт-Петербург
2025 г.

Цель работы

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

Задание

Вычислительная реализация задачи:

Функция:

$$y = \frac{4x}{x^4 + 3}, \quad x \in [-2, 0] \quad h = 0{,}2$$

Сформируем таблицу табулирования функции $y = \frac{4x}{x^4+3}$ на интервале $x \in [-2, 0]$ с шагом $h = 0{,}2$:

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	-0,421	-0,533	-0,669	-0,818	-0,946	-1	-0,938	-0,767	-0,528	-0,266	0

Линейная аппроксимация

Найдем линейное приближение данной функции:

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

Введем обозначения:

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

Вычислим суммы:

$$SX = -(2 + 1{,}8 + 1{,}6 + 1{,}4 + 1{,}2 + 1 + 0{,}8 + 0{,}6 + 0{,}4 + 0{,}2) = -11$$
$$SXX = 4{,}00 + 3{,}24 + 2{,}56 + 1{,}96 + 1{,}44 + 1{,}00 + 0{,}64 + 0{,}36 + 0{,}16 + 0{,}04 + 0 = 15{,}4$$
$$SY = -(0{,}421 + 0{,}533 + 0{,}669 + 0{,}818 + 0{,}946 + 1 + 0{,}938 + 0{,}767 + 0{,}528 + 0{,}266) = -6{,}886$$
$$SXY = 0{,}842 + 0{,}9594 + 1{,}0704 + 1{,}1452 + 1{,}1352 + 1 + 0{,}7504 + 0{,}4602 + 0{,}2112 + 0{,}0532 = 7{,}627$$

Получаем систему линейных уравнений:

$$\begin{cases} aSXX + bSX = SXY \\ aSX + bn = SY \end{cases} \rightarrow \begin{cases} 15{,}4a - 11b = 7{,}627 \\ -11a + 11b = -6{,}886 \end{cases}$$

Из которой по правилу Крамера находим:

$$\begin{cases} \Delta = SXX \cdot n - SX \cdot SX \\ \Delta_1 = SXY \cdot n - SX \cdot SY \\ \Delta_2 = SXX \cdot SY - SX \cdot SXY \end{cases} \rightarrow \begin{cases} a = \frac{\Delta_1}{\Delta} = \frac{7{,}627 \cdot (-6{,}886) - (-11) \cdot 7{,}627}{15{,}4 \cdot 11 - (-11) \cdot (-11)} = 0{,}168 \\ b = \frac{\Delta_2}{\Delta} = \frac{15{,}4 \cdot (-6{,}886) - (-11) \cdot 7{,}627}{15{,}4 \cdot 11 - (-11) \cdot (-11)} = -0{,}457 \end{cases}$$

Таким образом, аппроксимирующая линейная функция имеет вид:

$$\varphi(x) = 0{,}168x - 0{,}457$$

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	-0,421	-0,533	-0,669	-0,818	-0,946	-1	-0,938	-0,767	-0,528	-0,266	0
$\varphi(x_i)$	-0,793	-0,759	-0,726	-0,692	-0,659	-0,625	-0.591	-0,558	-0,524	-0,490	-0,457
ε_i	-0,372	-0.226	-0,057	0,126	0,287	0,375	0,347	0,209	0,004	-0,224	-0,457

$\varepsilon_i = \varphi(x_i) - y_i$

Среднеквадратичное отклонение:

$$\delta = \sqrt{\frac{\sum_{i=1}^{n}(\varphi(x_i) - y_i)^2}{n}} = 0{,}279$$

Квадратичная аппроксимация

Найдем квадратичное приближение данной функции:

$$\varphi(x) = a_2x^2 + a_1x + a_0$$

Введем обозначения:

$$SX = \sum_{i=1}^{n} x_i; \quad SXX = \sum_{i=1}^{n} x_i^2; \quad SXXX = \sum_{i=1}^{n} x_i^3; \quad SXXXX = \sum_{i=1}^{n} x_i^4$$

$$SY = \sum_{i=1}^{n} y_i; \quad SXY = \sum_{i=1}^{n} x_i y_i; \quad SXXY = \sum_{i=1}^{n} x_i^2 y_i$$

Вычислим суммы:

$SX = -11$
$SXX = 15{,}4$
$SXXX = -24{,}2$
$SXXXX = 40{,}532$
$SY = -6{,}886$
$SXY = 7{,}627$
$SXXY = -10{,}060$

Получаем систему линейных уравнений:

$$\begin{cases} na_0 + a_1SX + a_2SXX = SY \\ a_0SX + a_1SXX + a_2SXXX = SXY \\ a_0SXX + a_1SXXX + a_2SXXXX = SXXY \end{cases} \rightarrow \begin{cases} 11a_0 - 11a_1 + 15{,}4a_2 = -6{,}886 \\ -11a_0 + 15{,}4a_1 - 24{,}2a_2 = 7{,}627 \\ 15{,}4a_0 - 24{,}2a_1 + 40{,}532a_2 = -10{,}060 \end{cases}$$

Из которой по правилу Крамера находим:

$$\begin{cases} \Delta = 66{,}405 \\ \Delta_0 = 0{,}465 \\ \Delta_1 = 113{,}960 \\ \Delta_2 = 51{,}454 \end{cases} \rightarrow \begin{cases} a_0 = \dfrac{\Delta_0}{\Delta} = \dfrac{0{,}465}{66{,}405} = 0{,}007 \\ a_1 = \dfrac{\Delta_1}{\Delta} = \dfrac{113{,}960}{66{,}405} = 1{,}717 \\ a_2 = \dfrac{\Delta_2}{\Delta} = \dfrac{51{,}454}{66{,}405} = 0{,}774 \end{cases}$$

Таким образом, квадратичная аппроксимация функции имеет вид:

$$\varphi(x) = 0{,}007 + 1{,}717x + 0{,}774x^2$$

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	-0,421	-0,533	-0,669	-0,818	-0,946	-1	-0,938	-0,767	-0,528	-0,266	0
$\varphi(x_i)$	-0,331	-0,576	-0,759	-0,880	-0,939	-0,936	-0,871	-0,745	-0,556	-0,305	0,007
ε_i	0,09	-0,043	-0,09	-0,062	0,007	0,064	0,067	0,022	-0,028	-0,039	0,007

Среднеквадратичное отклонение:

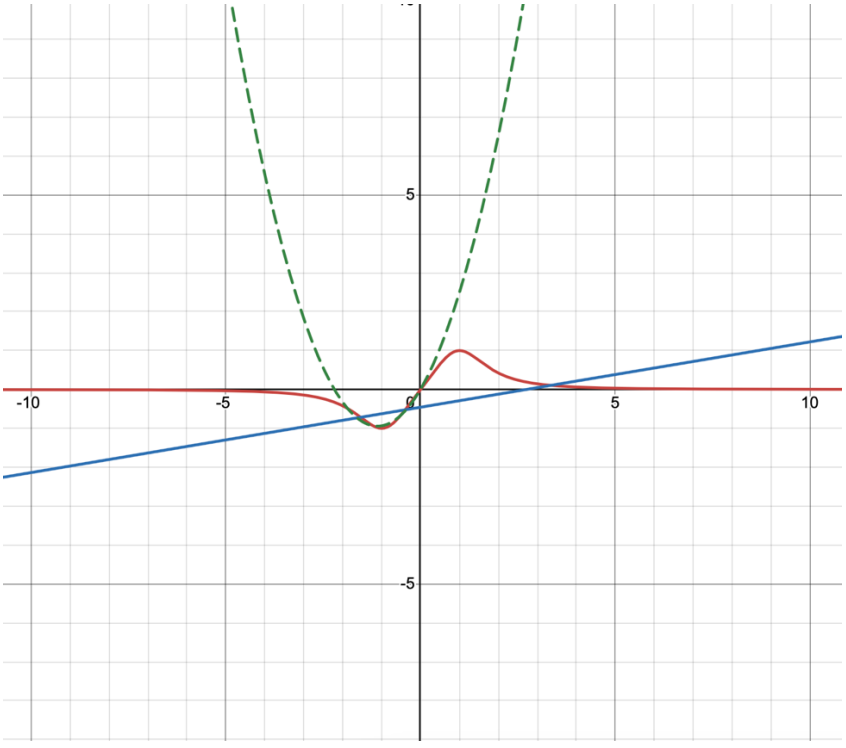
$$\delta = \sqrt{\frac{\sum_{i=1}^{n}(\varphi(x_i) - y_i)^2}{n}} = 0{,}055$$

Выбор наилучшего приближения

Лучшим является квадратичное приближение, поскольку его среднеквадратичное отклонение значительно меньше, чем у линейного:

$$0{,}279 > 0{,}055$$

Графики заданной функции

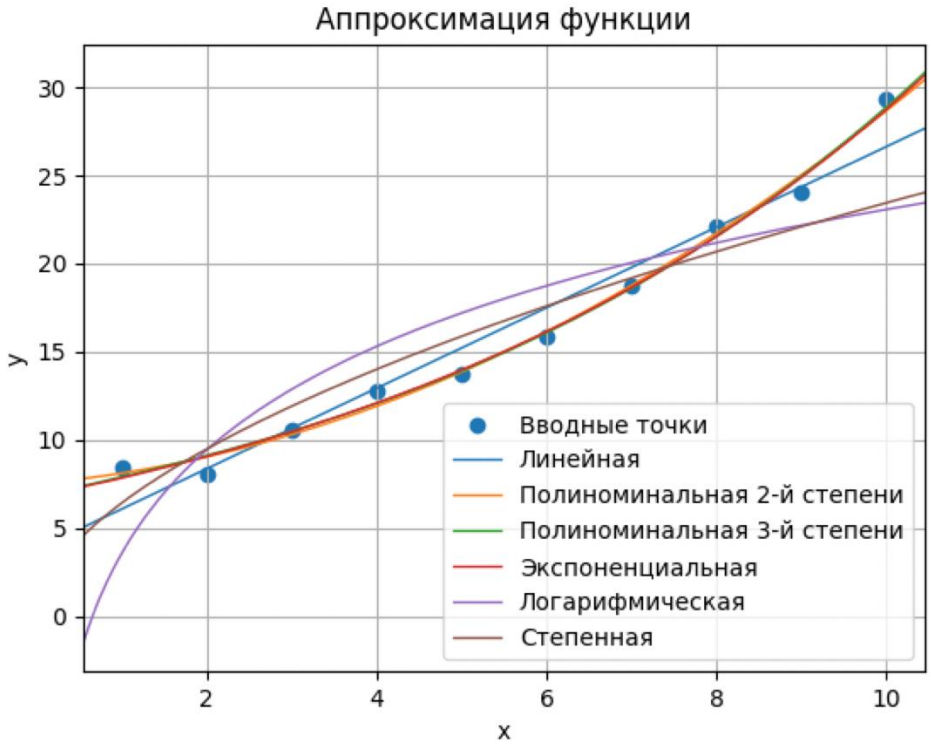


Листинг программы

```
def linear_approximation(X, Y):
    sx = sum(X)
    sxx = sum(x*x for x in X)
    sy = sum(Y)
    sxy = sum(x*y for x, y in zip(X, Y))
    A, B = np.linalg.solve([[len(X), sx], [sx, sxx]], [sy, sxy])
    return A, B

def square_approximation(X, Y):
    sx = sum(X)
    sxx = sum(x*x for x in X)
    sxxx = sum(x**3 for x in X)
    sxxxx = sum(x**4 for x in X)
    sy = sum(Y)
    sxy = sum(x*y for x, y in zip(X, Y))
    sxxy = sum(x*x*y for x, y in zip(X, Y))
    A, B, C = np.linalg.solve(
        [[len(X), sx, sxx],
         [sx, sxx, sxxx],
         [sxx, sxxx, sxxxx]],
        [sy, sxy, sxxy]
    )
    return A, B, C

def cube_approximation(X, Y):
    sx = sum(X)
    sxx = sum(x*x for x in X)
    sxxx = sum(x**3 for x in X)
    sxxxx = sum(x**4 for x in X)
    sxxxxx = sum(x**5 for x in X)
    sxxxxxx = sum(x**6 for x in X)
    sy = sum(Y)
    sxy = sum(x*y for x, y in zip(X, Y))
    sxxy = sum(x*x*y for x, y in zip(X, Y))
    sxxxy = sum(x**3 * y for x, y in zip(X, Y))
    A, B, C, D = np.linalg.solve(
        [[len(X), sx,   sxx,   sxxx],
         [sx,     sxx,  sxxx,  sxxxx],
         [sxx,    sxxx, sxxxx, sxxxxx],
         [sxxx,   sxxxx, sxxxxx, sxxxxxx]],
        [sy, sxy, sxxy, sxxxy]
    )
    return A, B, C, D

def exponential_approximation(X, Y):
    lnY = [math.log(y) for y in Y]
    A, b = linear_approximation(X, lnY)
    return math.exp(A), b

def logarithmic_approximation(X, Y):
    lnX = [math.log(x) for x in X]
    A, b = np.linalg.solve(
        [[len(X), sum(lnX)], [sum(lnX), sum(v*v for v in lnX)]],
        [sum(Y), sum(y*lx for y, lx in zip(Y, lnX))]
    )
    return A, b

def power_approximation(X, Y):
    lnX = [math.log(x) for x in X]
    lnY = [math.log(y) for y in Y]
    A, b = linear_approximation(lnX, lnY)
    return math.exp(A), b
```

```python
def count_correlation(X, Y):
    mx, my = sum(X)/len(X), sum(Y)/len(Y)
    num = sum((x-mx)*(y-my) for x,y in zip(X,Y))
    den = math.sqrt(sum((x-mx)**2 for x in X) * sum((y-my)**2 for y in Y))
    return num/den

def count_R2(Y, PHI):
    m = sum(PHI)/len(PHI)
    ss_res = sum((y - p)**2 for y, p in zip(Y, PHI))
    ss_tot = sum((y - m)**2 for y in Y)
    return 1 - ss_res/ss_tot

def count_sigma(Y, PHI):
    n = len(Y)
    return math.sqrt(sum((y - p)**2 for y,p in zip(Y,PHI)) / n)
```

Примеры и результаты работы программы




```
Выберите способ ввода: консоль — '1', файл — '2': 1
Введите точки в формате `x y`. По окончании ввода введите `q`.
1   8.4
2   8.1
3  10.6
4  12.8
5  13.7
6  15.9
7  18.7
8  22.1
9  24.0
10 29.3
q
Выберите способ вывода: консоль — '1', файл — '2': 1
Аппроксимирующая функция: Линейная
```

Функция: φ(x) = 2.284x + 3.800
Среднеквадратичное отклонение: σ = 1.368
Коэффициент детерминации: R^2 = 0.958 (Высокая аппроксимация)
Мера отклонения (SSE): S = 18.727
Коэффициент корреляции Пирсона: r = 0.979

Массивы значений:
x_i: 1.000, 2.000, 3.000, 4.000, 5.000, 6.000, 7.000, 8.000, 9.000, 10.000
y_i: 8.400, 8.100, 10.600, 12.800, 13.700, 15.900, 18.700, 22.100, 24.000, 29.300
φ(x_i): 6.084, 8.367, 10.651, 12.935, 15.218, 17.502, 19.785, 22.069, 24.353, 26.636
ε_i: -2.316, 0.267, 0.051, 0.135, 1.518, 1.602, 1.085, -0.031, 0.353, -2.664
==========
Аппроксимирующая функция: Полиноминальная 2-й степени
Функция: φ(x) = 0.170x^2 + 0.417x + 7.533
Среднеквадратичное отклонение: σ = 0.593
Коэффициент детерминации: R^2 = 0.992 (Высокая аппроксимация)
Мера отклонения (SSE): S = 3.522

Массивы значений:
x_i: 1.000, 2.000, 3.000, 4.000, 5.000, 6.000, 7.000, 8.000, 9.000, 10.000
y_i: 8.400, 8.100, 10.600, 12.800, 13.700, 15.900, 18.700, 22.100, 24.000, 29.300
φ(x_i): 8.120, 9.046, 10.312, 11.916, 13.861, 16.144, 18.767, 21.730, 25.032, 28.673
ε_i: -0.280, 0.946, -0.288, -0.884, 0.161, 0.244, 0.067, -0.370, 1.032, -0.627
==========
Аппроксимирующая функция: Полиноминальная 3-й степени
Функция: φ(x) = 0.008x^3 + 0.031x^2 + 1.059x + 6.810
Среднеквадратичное отклонение: σ = 0.575
Коэффициент детерминации: R^2 = 0.993 (Высокая аппроксимация)
Мера отклонения (SSE): S = 3.303

Массивы значений:
x_i: 1.000, 2.000, 3.000, 4.000, 5.000, 6.000, 7.000, 8.000, 9.000, 10.000
y_i: 8.400, 8.100, 10.600, 12.800, 13.700, 15.900, 18.700, 22.100, 24.000, 29.300
φ(x_i): 7.908, 9.117, 10.489, 12.073, 13.921, 16.084, 18.610, 21.553, 24.961, 28.885
ε_i: -0.492, 1.017, -0.111, -0.727, 0.221, 0.184, -0.090, -0.547, 0.961, -0.415
==========
Аппроксимирующая функция: Экспоненциальная
Функция: φ(x) = 6.792 * e^0.144x
Среднеквадратичное отклонение: σ = 0.573
Коэффициент детерминации: R^2 = 0.993 (Высокая аппроксимация)
Мера отклонения (SSE): S = 3.284

```
Массивы значений:
xᵢ: 1.000, 2.000, 3.000, 4.000, 5.000, 6.000, 7.000, 8.000,
9.000, 10.000
yᵢ: 8.400, 8.100, 10.600, 12.800, 13.700, 15.900, 18.700,
22.100, 24.000, 29.300
φ(xᵢ): 7.847, 9.065, 10.472, 12.097, 13.975, 16.144, 18.649,
21.544, 24.888, 28.751
εᵢ: -0.553, 0.965, -0.128, -0.703, 0.275, 0.244, -0.051, -0.556,
0.888, -0.549
==================================================
Аппроксимирующая функция: Логарифмическая
Функция: φ(x) = 8.474 * ln(x) + 3.561
Среднеквадратичное отклонение: σ = 3.190
Коэффициент детерминации: R² = 0.773 (Удовлетворительная
аппроксимация)
Мера отклонения (SSE): S = 101.734

Массивы значений:
xᵢ: 1.000, 2.000, 3.000, 4.000, 5.000, 6.000, 7.000, 8.000,
9.000, 10.000
yᵢ: 8.400, 8.100, 10.600, 12.800, 13.700, 15.900, 18.700,
22.100, 24.000, 29.300
φ(xᵢ): 3.561, 9.435, 12.870, 15.308, 17.199, 18.744, 20.050,
21.181, 22.180, 23.072
εᵢ: -4.839, 1.335, 2.270, 2.508, 3.499, 2.844, 1.350, -0.919, -
1.820, -6.228
==================================================
Аппроксимирующая функция: Степенная
Функция: φ(x) = 6.424 * x^0.562
Среднеквадратичное отклонение: σ = 2.382
Коэффициент детерминации: R² = 0.874 (Удовлетворительная
аппроксимация)
Мера отклонения (SSE): S = 56.734

Массивы значений:
xᵢ: 1.000, 2.000, 3.000, 4.000, 5.000, 6.000, 7.000, 8.000,
9.000, 10.000
yᵢ: 8.400, 8.100, 10.600, 12.800, 13.700, 15.900, 18.700,
22.100, 24.000, 29.300
φ(xᵢ): 6.424, 9.486, 11.915, 14.007, 15.880, 17.594, 19.187,
20.683, 22.099, 23.448
εᵢ: -1.976, 1.386, 1.315, 1.207, 2.180, 1.694, 0.487, -1.417, -
1.901, -5.852
==================================================
Лучшая аппроксимирующая функция: Экспоненциальная
Формула: φ(x) = 6.792 * e^0.144x
```

Вывод

В ходе выполнения лабораторной работы я изучил методы нахождения функции, являющейся наилучшим приближением заданной табличной функции по методу

наименьших квадратов, выполнил программную реализацию методов и построение функций на графике на языке Python.