Федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский университет ИТМО»
Факультет программной инженерии и компьютерной техники

Лабораторная работа 4 по дисциплине

«Вычислительная математика»

Вариант № 10

Выполнил:
Мамонтов Г. А.

Преподаватели:
Машина Е. А.
Малышева Т. А.

Санкт-Петербург, 2025 г

Цель работы

Цель : найти функцию, являющуюся наилучшим приближением заданной табличной функции по методу наименьших квадратов.

Рабочие формулы метода

Вычислительная часть лабораторной работы

Функция: $y = \frac{18x}{x^4 + 10}$; исследуемый интервал: $x \in [0, 4]$; $h = 0,4$

1. Сформировать таблицу табулирования заданной функции на указанном интервале

x_i	0	0,4	0,8	1,2	1,6	2	2,4	2,8	3.2	3,6	4
y_i	0	0.718	1.383	1.789	1.740	1.385	1.001	0.705	0.501	0.364	0.271

2. Построить линейное и квадратичное приближения по 11 точкам заданного интервала

Линейное приближение

Форма приближающей функции: $f(x) = a_0 + a_1 x$
Система уравнений для нахождения коэффициентов:

$$\Sigma y = a_0 n + a_1 \Sigma x$$
$$\Sigma xy = a_0 \Sigma x + a_1 \Sigma x^2$$

Отсюда:

$$a_1 = \frac{n\Sigma xy - \Sigma x \Sigma y}{n\Sigma x^2 - (\Sigma x)^2} = \frac{11 * 17.468 - 22 * 9.857}{11 * 61.6 - 22^2} \approx -0.128$$
$$a_0 = \frac{\Sigma y - a_1 \Sigma x}{n} = \frac{9.857 + 0.128 * 22}{11} \approx 1,152$$

$$f(x) = 1.152 - 0.128x$$

Квадратичное приближение

Форма приближающей функции: $f(x) = a_0 + a_1x + a_2x^2$

Система уравнений для нахождения коэффициентов:

$\Sigma y = a_0 n + a_1 \Sigma x + a_2 \Sigma x^2$

$\Sigma xy = a_0 \Sigma x + a_1 \Sigma x^2 + a_2 \Sigma x^3$

$\Sigma x^2 y = a_0 \Sigma x^2 + a_1 \Sigma x^3 + a_2 \Sigma x^4$

Рассчитаем коэффициенты:

$\Sigma y = 9.857$; $\Sigma x = 22$; $\Sigma x^2 = 61.6$; $\Sigma x^3 = 193.6$; $\Sigma x^4 = 648.525$; $\Sigma xy = 17.468$; $\Sigma x^2 y = 39.046$

Получаем систему:

$11a_0 + 22a_1 + 61.6a_2 = 9.857$

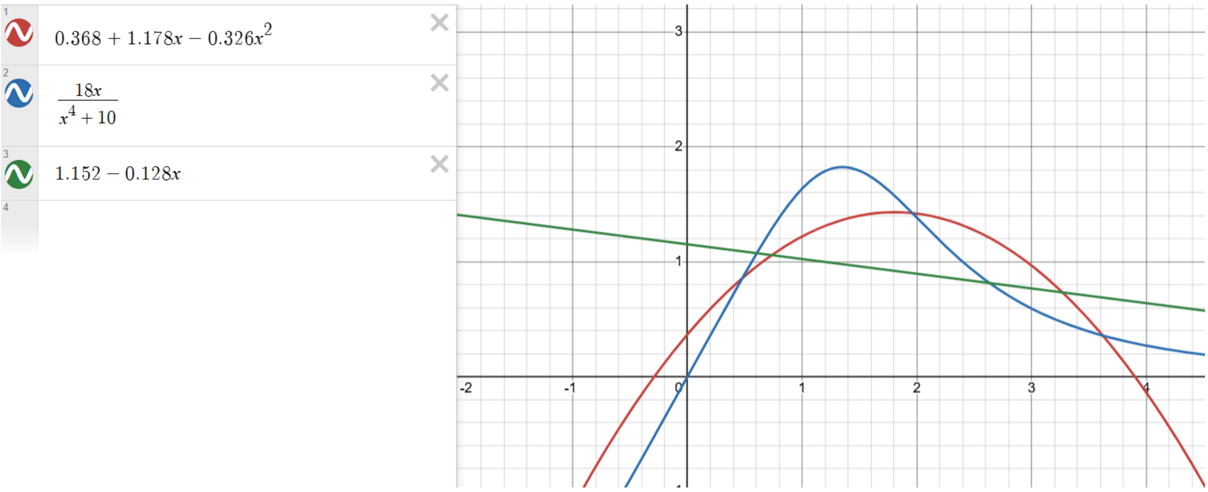
$22a_0 + 61.6a_1 + 193.6a_2 = 17.468$

$61.6a_0 + 193.6a_1 + 648.525a_2 = 39.046$

Решение системы:

$a_0 = 0.368$; $a_1 = 1,178$; $a_2 \approx -0.326$

Квадратичное приближение эффективнее.



Листинг программы

```
import math
import matplotlib.pyplot as plt

# Ввод с клавиатуры
def read_data_console():
    n = int(input("Введите количество точек: "))
    x, y = [], []
    for _ in range(n):
        xi, yi = map(float, input("Введите x и y: ").split())
        x.append(xi)
        y.append(yi)
    return x, y

# Ввод из файла: формат — по одной паре x y на строку
def read_data_file():
    filename = input("Введите имя файла: ")
    x, y = [], []
    with open(filename, 'r') as f:
        for line in f:
            if line.strip():
                xi, yi = map(float, line.strip().split())
                x.append(xi)
                y.append(yi)
    return x, y



def mean(values):
    return sum(values) / len(values)



# Решение системы линейных уравнений методом Гаусса
def solve_slae(A, b):
    n = len(b)

    for i in range(n):
        max_row = max(range(i, n), key=lambda k: abs(A[k][i]))
        A[i], A[max_row] = A[max_row], A[i]
        b[i], b[max_row] = b[max_row], b[i]
```

```
        for j in range(i + 1, n):
            coef = A[j][i] / A[i][i]
            for k in range(i, n):
                A[j][k] -= coef * A[i][k]
            b[j] -= coef * b[i]

    x = [0] * n
    for i in reversed(range(n)):
        s = sum(A[i][j] * x[j] for j in range(i + 1, n))
        x[i] = (b[i] - s) / A[i][i]
    return x



# Линейная: y = a·x + b
def lin_approx(x, y):

    #Строим приближающую прямую y = a·x + b по МНК.
    n = len(x)
    sx = sum(x)
    sy = sum(y)
    sxx = sum(xi**2 for xi in x)
    sxy = sum(xi * yi for xi, yi in zip(x, y))

    # Решаем систему:
    # a * Σx^2 + b * Σx = Σx*y
    # a * Σx   + b * n  = Σy

    a = (n * sxy - sx * sy) / (n * sxx - sx**2)
    b = (sy - a * sx) / n

    return lambda t: a * t + b, (a, b)


# Квадратичная: y = a2·x² + a1·x + a0
def quad_approx(x, y):

    #Строим параболу по методу наименьших квадратов.
    n = len(x)

    # Вычисляем все необходимые суммы:
    sx = sum(x)
    sx2 = sum(xi**2 for xi in x)
```

```
    sx3 = sum(xi**3 for xi in x)
    sx4 = sum(xi**4 for xi in x)
    sy = sum(y)
    sxy = sum(xi * yi for xi, yi in zip(x, y))
    sx2y = sum(xi**2 * yi for xi, yi in zip(x, y))

    # Составляем систему:
    # a2*Σx^4 + a1*Σx^3 + a0*Σx^2 = Σx^2·y
    # a2*Σx^3 + a1*Σx^2 + a0*Σx   = Σx·y
    # a2*Σx^2 + a1*Σx   + a0*n    = Σy

    A = [
        [sx4, sx3, sx2],
        [sx3, sx2, sx],
        [sx2, sx,  n]
    ]
    b = [sx2y, sxy, sy]

    a2, a1, a0 = solve_slae(A, b)

    return lambda t: a2 * t**2 + a1 * t + a0, (a2, a1, a0)


# Кубическая: y = a3·x³ + a2·x² + a1·x + a0
def cubic_approx(x, y):

    # Строим кубическую кривую по методу наименьших квадратов.
    n = len(x)

    # Вычисляем все суммы до шестой степени
    sx = sum(x)
    sx2 = sum(xi**2 for xi in x)
    sx3 = sum(xi**3 for xi in x)
    sx4 = sum(xi**4 for xi in x)
    sx5 = sum(xi**5 for xi in x)
    sx6 = sum(xi**6 for xi in x)
    sy = sum(y)
    sxy = sum(xi * yi for xi, yi in zip(x, y))
    sx2y = sum(xi**2 * yi for xi, yi in zip(x, y))
    sx3y = sum(xi**3 * yi for xi, yi in zip(x, y))

    # Составляем систему на 4 неизвестных: a3, a2, a1, a0
    A = [
```

```
        [sx6, sx5, sx4, sx3],
        [sx5, sx4, sx3, sx2],
        [sx4, sx3, sx2, sx],
        [sx3, sx2, sx,  n]
    ]
    b = [sx3y, sx2y, sxy, sy]

    a3, a2, a1, a0 = solve_slae(A, b)

    return lambda t: a3*t**3 + a2*t**2 + a1*t + a0, (a3, a2, a1, a0)


# Экспоненциальная: y = a·e^(b·x)
def exp_approx(x, y):
    # Линеаризуем модель: ln(y) = ln(a) + b·x, решаем как линейную.

    Y = [math.log(yi) for yi in y]  # ln(y)
    f, (b, ln_a) = lin_approx(x, Y)
    a = math.exp(ln_a)
    return lambda t: a * math.exp(b * t), (a, b)


# Логарифмическая: y = a·ln(x) + b
def log_approx(x, y):

    #Аппроксимация функции вида y = a * ln(x) + b
    #Работает только для x > 0
    # Проверка на допустимость значений

    if any(xi <= 0 for xi in x):
        raise ValueError("Логарифмическая аппроксимация применима
только при x > 0")

    # Логарифмируем x
    X = [math.log(xi) for xi in x]

    # Аппроксимируем линейной функцией: ln(x) → X, y — без изменений
    f_lin, (a, b) = lin_approx(X, y)

    # Возвращаем восстановленную функцию по x: y = a * ln(x) + b
    def f_log(t):
        if t <= 0:
```

```
            return float('nan')  # безопасное значение при попытке
log(0) или log(<0)
        return a * math.log(t) + b

    return f_log, (a, b)

# Степенная: y = a·x^b
def power_approx(x, y):
    # Логарифмируем обе переменные: ln(y) = ln(a) + b·ln(x)

    X = [math.log(xi) for xi in x]
    Y = [math.log(yi) for yi in y]
    f, (b, ln_a) = lin_approx(X, Y)
    a = math.exp(ln_a)
    return lambda t: a * t**b, (a, b)


# Среднеквадратическое отклонение
def deviation(f, x, y):
    return math.sqrt(sum((f(xi) - yi)**2 for xi, yi in zip(x, y)) /
len(x))


# Коэффициент корреляции Пирсона
def correlation(x, y):
    mx, my = mean(x), mean(y)
    num = sum((xi - mx) * (yi - my) for xi, yi in zip(x, y))
    den = math.sqrt(sum((xi - mx)**2 for xi in x) * sum((yi - my)**2
for yi in y))
    return num / den

# Коэффициент детерминации
def determination(r):
    R2 = r**2
    if R2 > 0.9:
        msg = "сильная зависимость"
    elif R2 > 0.7:
        msg = "умеренная зависимость"
    else:
        msg = "слабая зависимость"
    return R2, msg
```

```
def main():
    print("Выберите способ ввода данных:")
    print("1 - Ввод с клавиатуры")
    print("2 - Ввод из файла")
    choice = input("Ваш выбор: ")

    if choice == "1":
        x, y = read_data_console()
    elif choice == "2":
        x, y = read_data_file()
    else:
        print("Неверный выбор.")
        return

    funcs = {
        "Линейная": lin_approx,
        "Квадратичная": quad_approx,
        "Кубическая": cubic_approx,
        "Экспоненциальная": exp_approx,
        "Логарифмическая": log_approx,
        "Степенная": power_approx
    }

    results = {}
    for name, approx in funcs.items():
        try:
            f, coeffs = approx(x, y)
            dev = deviation(f, x, y)
            results[name] = (f, coeffs, dev)
            print(f"{name}: коэффициенты = {coeffs}, отклонение =
{dev:.3f}")
        except Exception as e:
            print(f"{name}: ошибка ({e})")

    best = min(results.items(), key=lambda item: item[1][2])
    print(f"\n Лучшая аппроксимация: {best[0]}")

    if "Линейная" in results:
        r = correlation(x, y)
        R2, msg = determination(r)
        print(f"Коэф. корреляции: {r:.3f}, детерминации: {R2:.3f} —
{msg}")
```

```
    # Построение графиков
    plt.scatter(x, y, label="Исходные данные", color="black")
    min_x, max_x = min(x), max(x)
    t_vals = [min_x - 1 + i * 0.1 for i in range(int((max_x - min_x +
2) * 10))]
    for name, (f, _, _) in results.items():
        y_vals = [f(t) for t in t_vals]
        plt.plot(t_vals, y_vals, label=name)

    plt.legend()
    plt.grid()
    plt.title("Аппроксимации методом наименьших квадратов")
    plt.show()

if __name__ == "__main__":
    main()
```

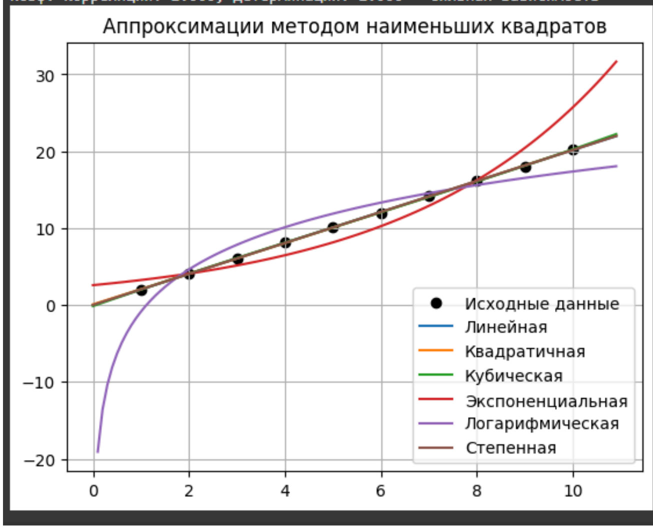
Untitled5.ipynb

Графики аппроксимирующих функций

Результаты выполнения программы при различных исходных данных

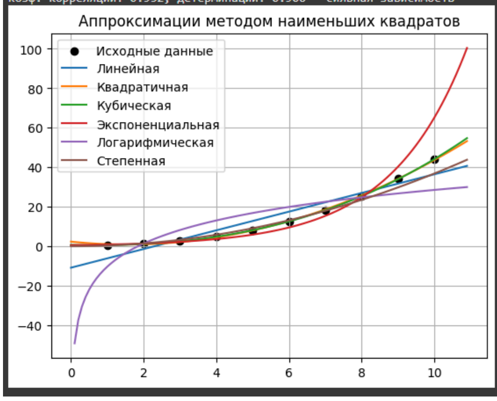
1:
1 2
2 4.1
3 6
4 8.2
5 10.1
6 11.9
7 14.2
8 16.1
9 18
10 20.3

```
Выберите способ ввода данных:
1 - Ввод с клавиатуры
2 - Ввод из файла
Ваш выбор: 2
Введите имя файла: /content/sample_data/data.txt
Линейная: коэффициенты = (2.013939393939395, 0.013333333333326891), отклонение = 0.106
Квадратичная: коэффициенты = (0.0026515151515131867, 1.9847727272727504, 0.07166666666661413), отклонение = 0.105
Кубическая: коэффициенты = (0.0025058275058301967, -0.03869463869468437, 2.17546620046642, -0.1433333333335975), отклонение = 0.095
Экспоненциальная: коэффициенты = (2.577696419837357, 0.23005807430172437), отклонение = 2.212
Логарифмическая: коэффициенты = (7.914523265189008, -0.8644224716619562), отклонение = 1.784
Степенная: коэффициенты = (2.0192828656302115, 0.9993590346924752), отклонение = 0.107

 Лучшая аппроксимация: Кубическая
Коэф. корреляции: 1.000, детерминации: 1.000 — сильная зависимость
```



2:
1 0.5
2 1.2
3 2.7
4 4.8
5 8.0
6 12.5
7 18.2
8 25.5
9 34.1
10 44.0

```
Выберите способ ввода данных:
1 - Ввод с клавиатуры
2 - Ввод из файла
Ваш выбор: 2
Введите имя файла: /content/sample_data/data2.txt
Линейная: коэффициенты = (4.730303030303030305, -10.866666666666669), отклонение = 4.374
Квадратичная: коэффициенты = (0.5999999999999986, -1.8696969696969543, 2.333333333333301), отклонение = 0.356
Кубическая: коэффициенты = (0.01923076923077241, 0.28269230769225356, -0.4062354312351709, 0.6833333333330242), отклонение = 0.110
Экспоненциальная: коэффициенты = (0.5296299503253954, 0.48119211908895854), отклонение = 7.126
Логарифмическая: коэффициенты = (16.868224547904973, -10.328462294683694), отклонение = 8.132
Степенная: коэффициенты = (0.3567677038467478, 2.0137495023670935), отклонение = 2.781

 Лучшая аппроксимация: Кубическая
Коэф. корреляции: 0.952, детерминации: 0.906 — сильная зависимость
```



3:

1 0

2 0.69

3 1.10

4 1.39

5 1.61

6 1.79

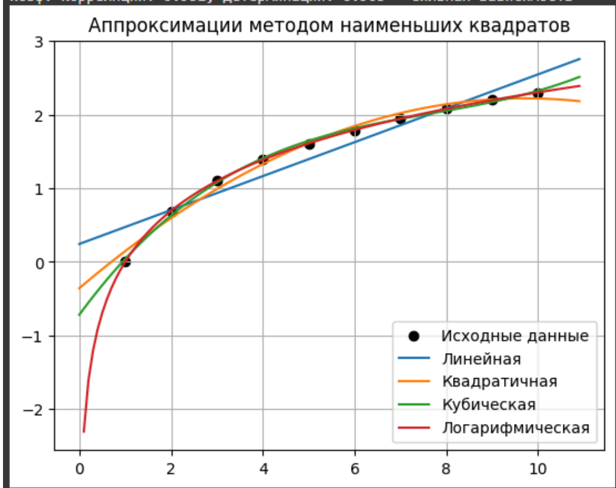
7 1.95

8 2.08

9 2.20

10 2.30

```
Выберите способ ввода данных:
1 - Ввод с клавиатуры
2 - Ввод из файла
Ваш выбор: 2
Введите имя файла: /content/sample_data/log_data.txt
Линейная: коэффициенты = (0.2304848484848486, 0.24333333333333265), отклонение = 0.214
Квадратичная: коэффициенты = (-0.02734848484848535, 0.5313181818181878, -0.3583333333333471), отклонение = 0.080
Кубическая: коэффициенты = (0.004203574203574202, -0.09670745920745895, 0.8512101787101757, -0.718999999999993), отклонение = 0.032
Экспоненциальная: ошибка (math domain error)
Логарифмическая: коэффициенты = (1.0006324380267895, -0.0003965177959047494), отклонение = 0.002
Степенная: ошибка (math domain error)

 Лучшая аппроксимация: Логарифмическая
Коэф. корреляции: 0.951, детерминации: 0.905 — сильная зависимость
```



Выводы

Во время выполнения работы был изучен принцип аппроксимации табличных функций и реализован на практике.