«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Вычислительная математика»

Отчет

По лабораторной работе №4

Вариант 1

Студент:

Алхимовици А.

P3210

Преподаватель:

Наумова Н. А.

Санкт-Петербург, 2025 г.

Цель лабораторной работы:

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

1 Вычислительная реализация задачи:

Линейная аппроксимация:

y = $\frac{12x}{x^4+1}$ **;** n = 11 ; x ∈ [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	1.0	1.2	1.4	1.6	1.8	2.0
yi	0	2.396	4.68	6.374	6.81	6	4.685	3.47	2.542	1.879	1.412

φ(x) = a + bx

Вычисляем суммы: sx = 11, sxx = 15.4, sy = 40.25, sxy = 38.38

$$\begin{cases} n*a+sx*b=sy \\ sx*a+sxx*b=sxy \end{cases} \quad \begin{cases} 11*a+11*b=40.25 \\ 11*a+15.4*b=38.38 \end{cases}$$

$$\begin{cases} a=4.084 \\ b=-0.425 \end{cases}$$

$$\varphi(x)=4.084-0.425*x$$

i	1	2	3	4	5	6	7	8	9	10	11
x_i	0	0.2	0.4	0.6	0.8	1.0	1.2	1.4	1.6	1.8	2.0
yi	0	2.396	4.68	6.374	6.81	6	4.685	3.47	2.542	1.879	1.412
φ(xi)	4.084	3.999	3,914	3,829	3,744	3,659	3,574	3,489	3.404	3.319	3.234
(φ(xi)-yi)^2	16.679	2.569	0.587	6,477	9,403	5,480	1,234	0	0,743	2.075	3.321

$$\boldsymbol{\sigma=\sqrt{\frac{\sum(\varphi\ (xi)-yi)^2}{n}}=2.101}$$

Квадратичная аппроксимация:

y = $\frac{12x}{x^4+1}$ **;** n = 11 ; x ∈ [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	1.0	1.2	1.4	1.6	1.8	2.0
yi	0	2.396	4.68	6.374	6.81	6	4.685	3.47	2.542	1.879	1.412

φ(x) = a + bx + cx^2

Вычисляем суммы:

sx = 11, sxx = 15.4, sxxx = 24,2; sxxxx = 40,53; sy = 40,25; sxy = 38,38; sxxy = 45,29

$$\begin{cases} n * a + sx * b + sxx * c = sy \\ sx * a + sxx * b + sxxx * c = sxy \\ sxx * a + sxxx * b + sxxxx * c = sxxy \end{cases}$$

$$\begin{cases} 11 * a + 11 * b + 15.4 * c = 40{,}25 \\ 11 * a + 15.4 * b + 24.2 * c = 38{,}38 \\ 15.4 * a + 24.2 * b + 40.53 * c = 45{,}29 \end{cases}$$

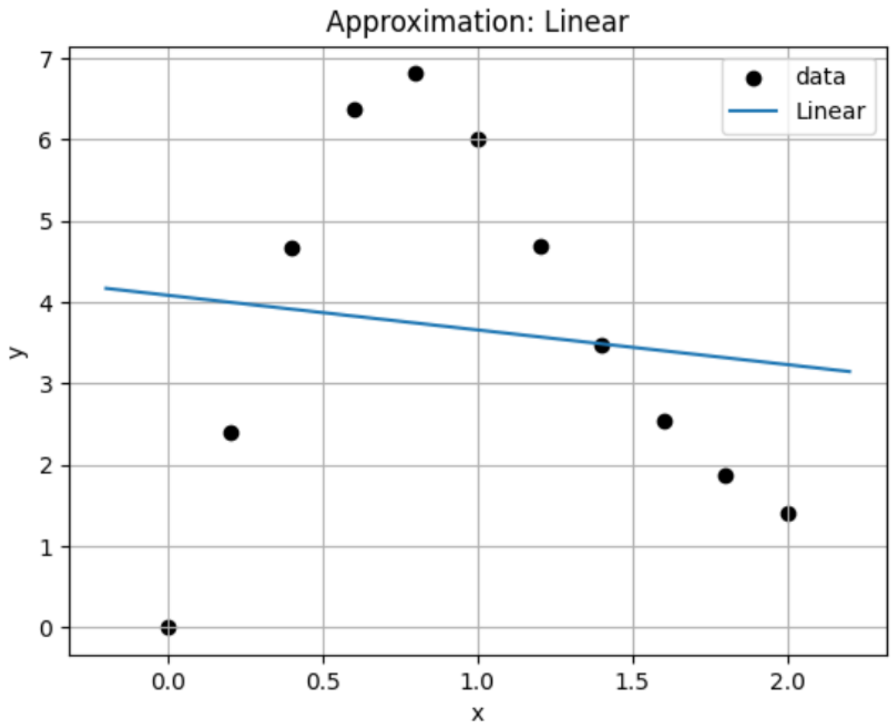
$$\begin{cases} a = 0.878 \\ b = 10.261 \\ c = -5.343 \end{cases}$$

$$\varphi(\mathrm{x}) = 0.878 + 10.261x - 5.343x^2$$

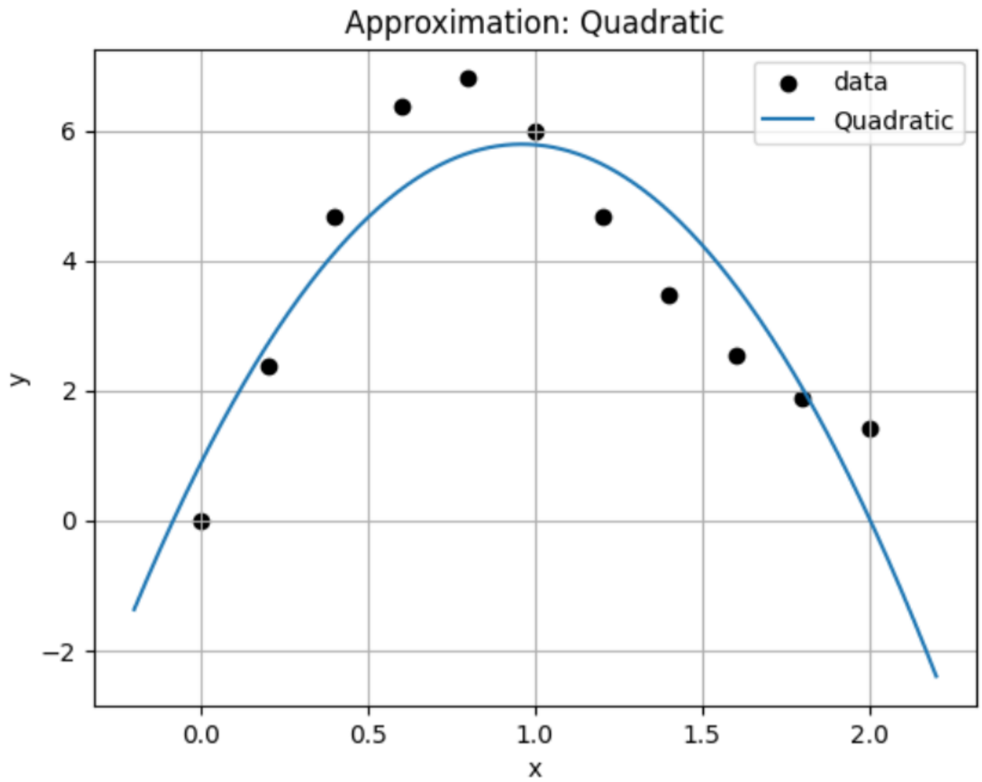
i	1	2	3	4	5	6	7	8	9	10	11
x_i	0	0.2	0.4	0.6	0.8	1.0	1.2	1.4	1.6	1.8	2.0
yi	0	2.396	4.68	6.374	6.81	6	4.685	3.47	2.542	1.879	1.412
φ(xi)	0.878	2.716	4.128	5.111	5.667	5.796	5.497	4.771	3.618	2.036	0.028
(φ(xi)-yi)^2	0.771	0.103	0.305	1.595	1.307	0.042	0.660	1.693	1.157	0.025	1.915

$$\sigma = \sqrt{\frac{\sum(\varphi\ (\mathrm{xi}) - \mathrm{yi})^2}{n}} = \mathbf{0.933}$$

0,933 < 2,101 у квадратичной аппроксимации среднеквадратичное отклонение меньше, поэтому это приближение лучше.



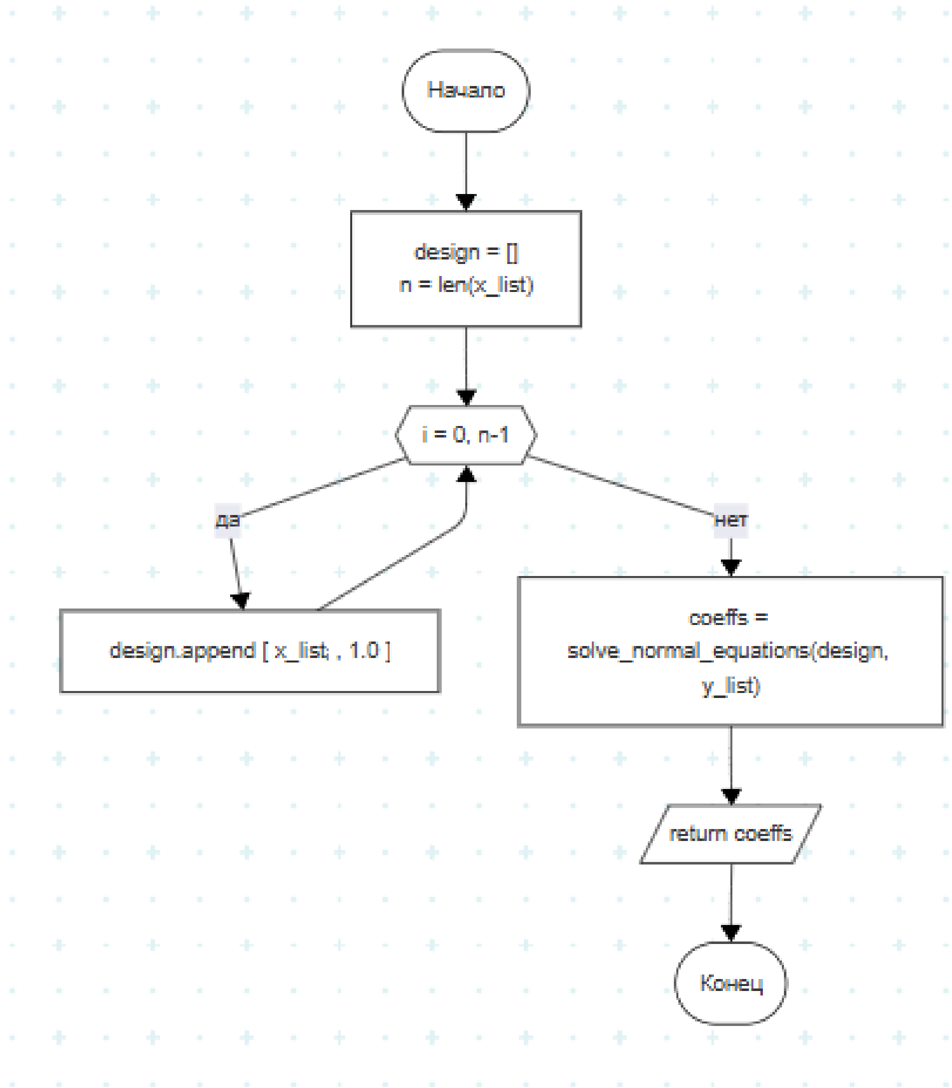
Quadratic Model



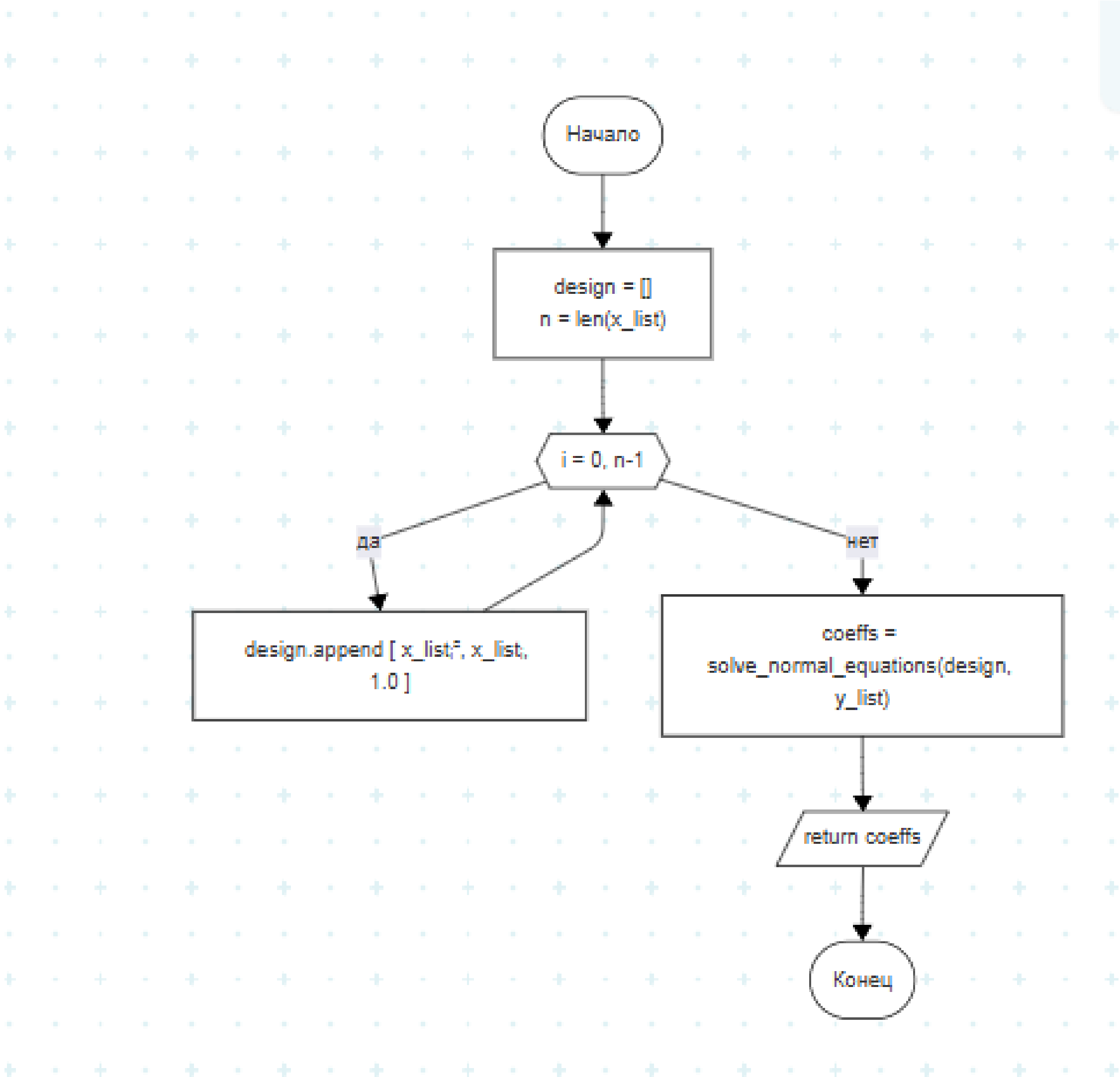
2. Программная реализация

Блок схемы

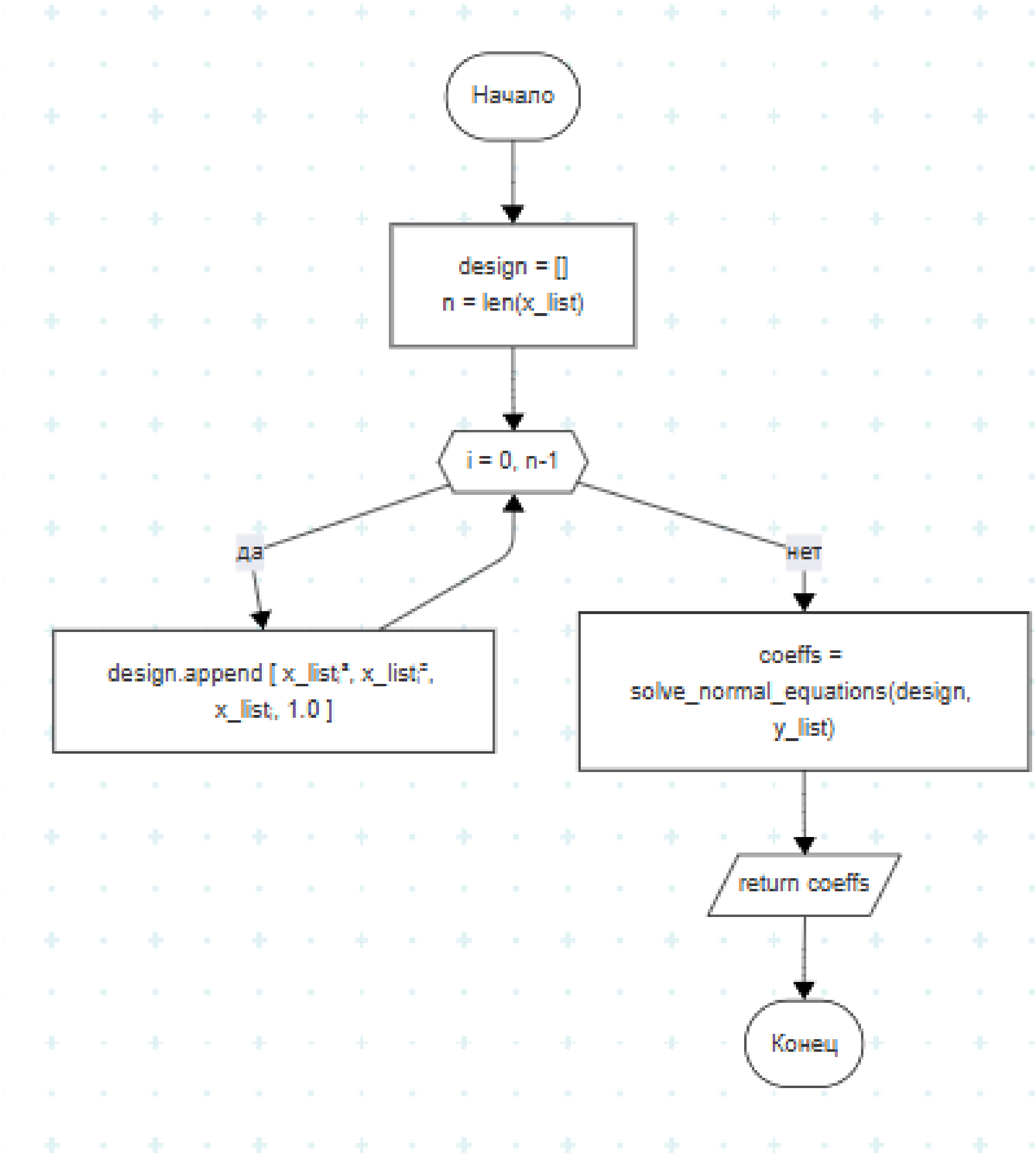
Линейная фукнция



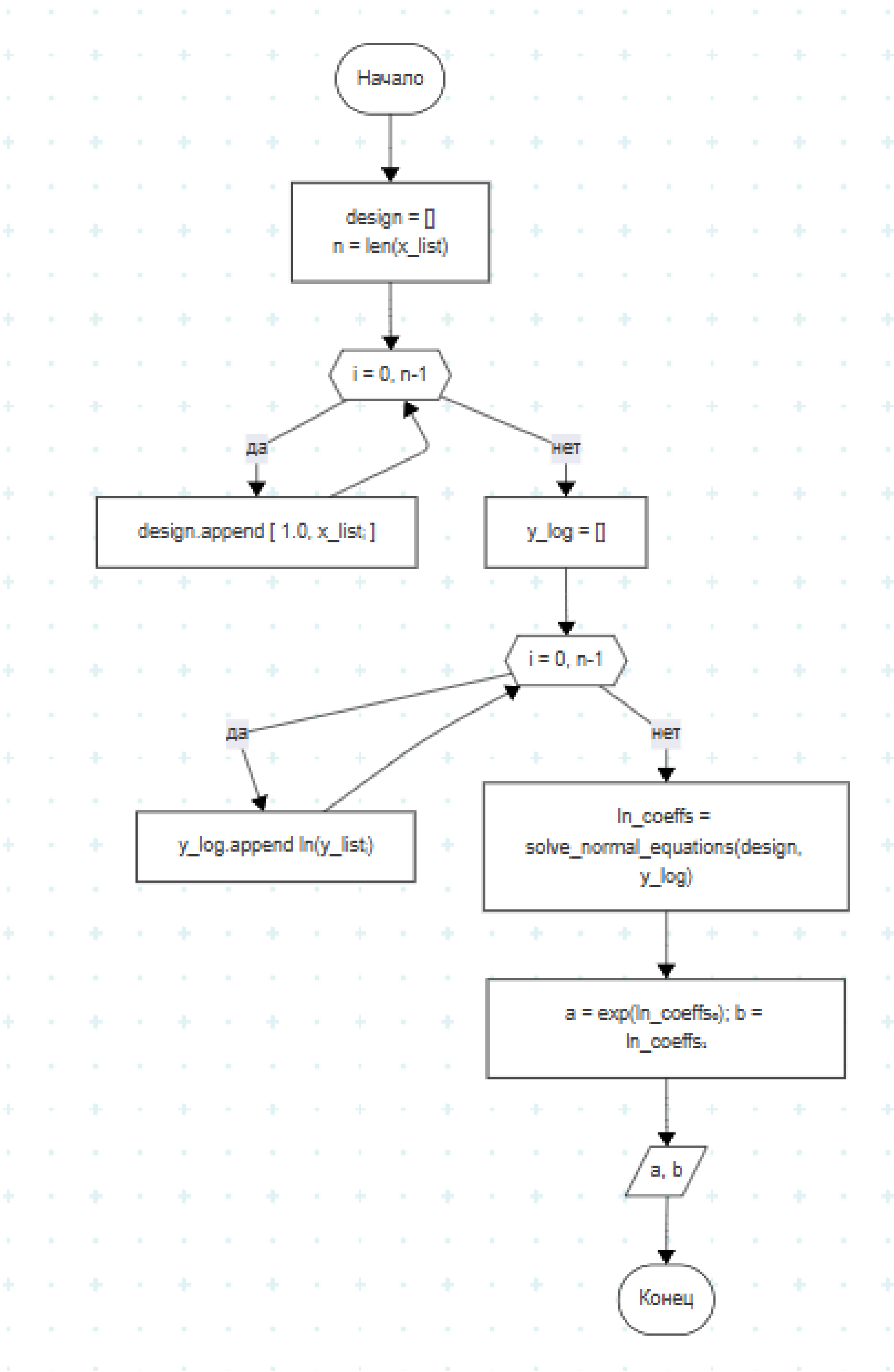
Полиномиальная функция 2-й степени



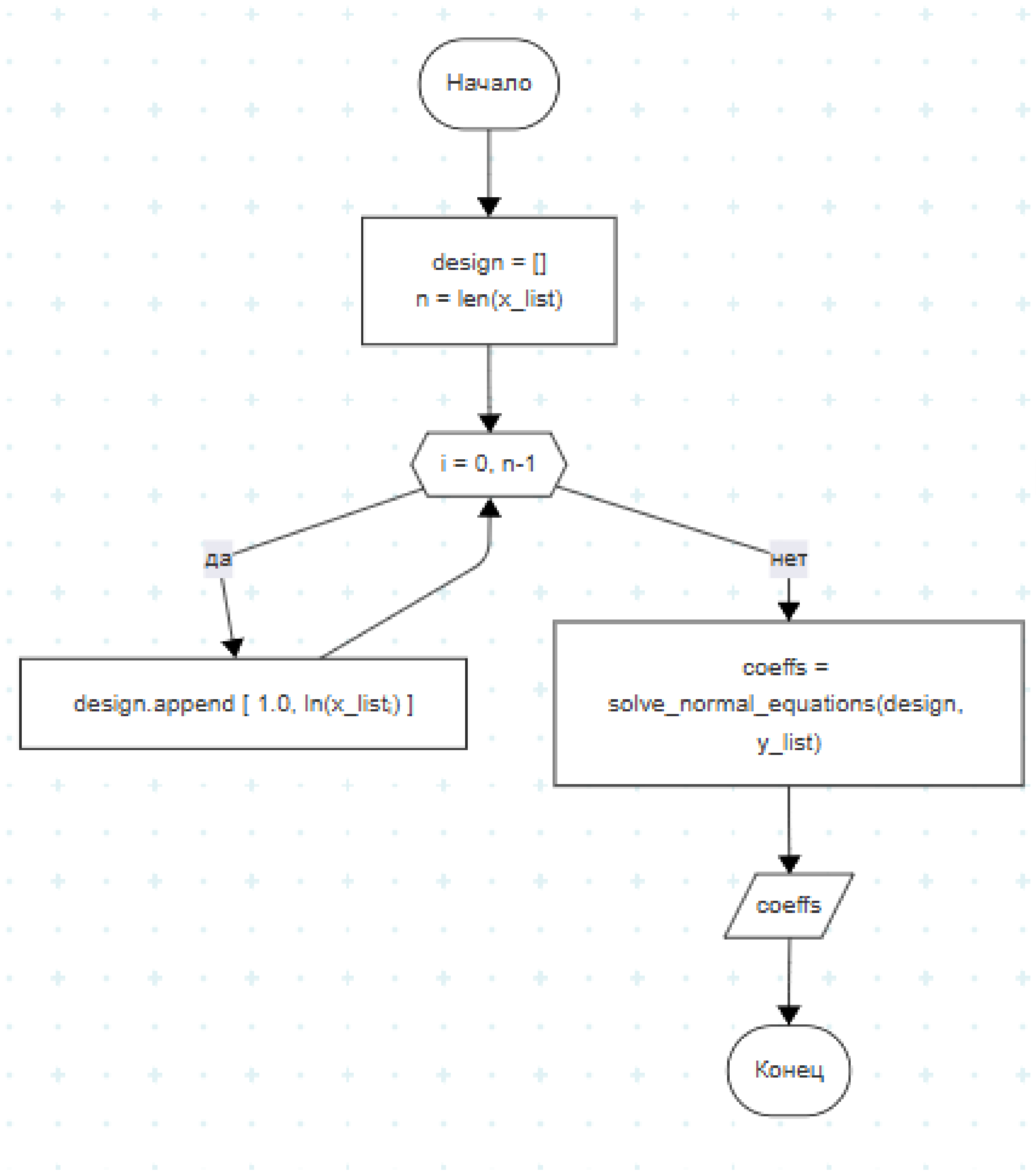
Полиномиальная функция 3-й степени



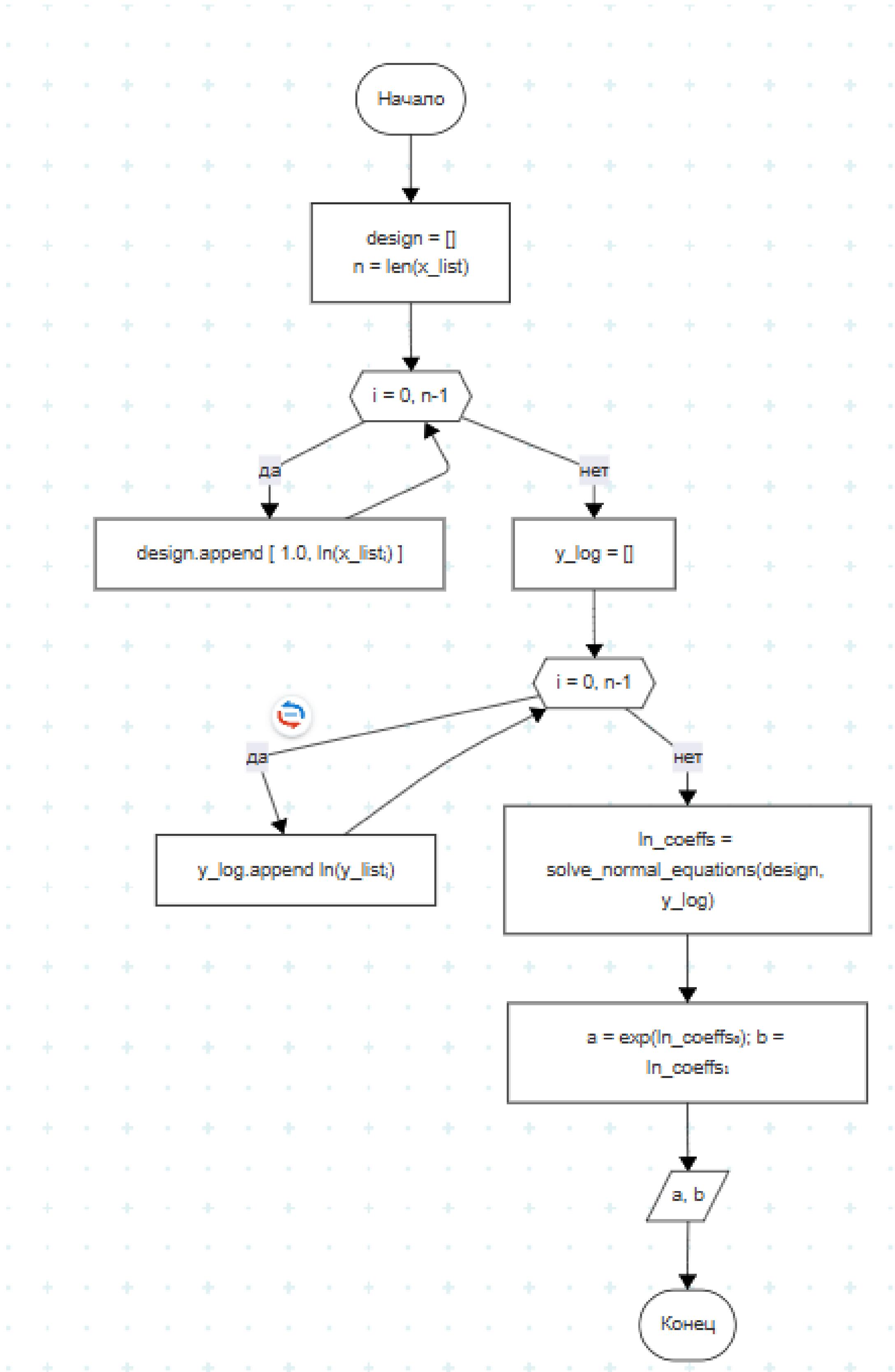
Экспоненциальная функция: y = a * exp(b*x)



Логарифмическая функция



Степенная функция



Листинг программы

https://github.com/senya-2011/Vu4Math/tree/main/lab4

```
def calc_measure_of_deviation(errors):
    squared = [error ** 2 for error in errors]
    return sum(squared) / len(squared)

# линейная аппроксимация: y = a*x + b
def approx_linear(x_list, y_list):
    design = [[x, 1.0] for x in x_list]
    return logic.solve_normal_equations(design, y_list)


# квадратичная аппроксимация: y = a*x^2 + b*x + c
def approx_quadratic(x_list, y_list):
    design = [[x ** 2, x, 1.0] for x in x_list]
    return logic.solve_normal_equations(design, y_list)


# кубическая аппроксимация: y = a*x^3 + b*x^2 + c*x + d
def approx_cubic(x_list, y_list):
    design = [[x ** 3, x ** 2, x, 1.0] for x in x_list]
    return logic.solve_normal_equations(design, y_list)


# экспоненциальная аппроксимация: y = a * exp(b*x)
def approx_exponential(x_list, y_list):
    # логарифмируем y, чтобы получить линейную зависимость ln(y) = ln(a) + b*x
    design = [[1.0, x] for x in x_list]
    y_log = [math.log(y_i) for y_i in y_list]
    ln_coeffs = logic.solve_normal_equations(design, y_log)
    a = math.exp(ln_coeffs[0])  # восстановление a
    b = ln_coeffs[1]
    return [a, b]


# логарифмическая аппроксимация: y = a + b*ln(x)
def approx_logarithmic(x_list, y_list):
    design = [[1.0, math.log(x)] for x in x_list]
    return logic.solve_normal_equations(design, y_list)


# степенная аппроксимация: y = a * x^b
def approx_power(x_list, y_list):
    # берем логарифмы: ln(y) = ln(a) + b*ln(x)
    design = [[1.0, math.log(x)] for x in x_list]
    y_log = [math.log(y_i) for y_i in y_list]
    ln_coeffs = logic.solve_normal_equations(design, y_log)
    a = math.exp(ln_coeffs[0])
    b = ln_coeffs[1]
    return [a, b]


def solve_normal_equations(design_matrix, y_vector):
    Xt = transpose_matrix(design_matrix)
    XtX = mat_mat_mul(Xt, design_matrix)
    XtY = mat_vec_mul(Xt, y_vector)
    return solve_gauss(XtX, XtY)


def solve_gauss(A, b, epsilon=1e-10):
    n = len(A)
    # Формируем расширенную матрицу
    M = [row[:] + [b_i] for row, b_i in zip(A, b)]
    # Прямой ход
```

```
    for i in range(n):
        # Поиск ведущего элемента
        max_row, max_val = i, abs(M[i][i])
        for r in range(i+1, n):
            if abs(M[r][i]) > max_val:
                max_row, max_val = r, abs(M[r][i])
        M[i], M[max_row] = M[max_row], M[i]
        pivot = M[i][i]
        for j in range(i, n+1): M[i][j] /= pivot
        for k in range(i+1, n):
            factor = M[k][i]
            for j in range(i, n+1):
                M[k][j] -= factor * M[i][j]
    # Обратный ход
    x = [0.0]*n
    for i in range(n-1, -1, -1):
        x[i] = M[i][n]
        for j in range(i+1, n):
            x[i] -= M[i][j]*x[j]
        x[i] /= M[i][i]
    return x

def calc_r2_and_pearson(x_list, y_list):
    predictions = generate_all_predictions(x_list, y_list)
    mean_y = sum(y_list) / len(y_list)
    stats = []
    for name, coeffs, y_pred in predictions:
        ss_res = sum((y - yp) ** 2 for y, yp in zip(y_list, y_pred))
        ss_tot = sum((y - mean_y) ** 2 for y in y_list)
        r2 = 1 - ss_res / ss_tot if ss_tot != 0 else None
        pearson_r = None
        if name == 'Linear':
            mean_x = sum(x_list) / len(x_list)
            cov = var_x = var_y = 0.0
            for x, y in zip(x_list, y_list):
                dx = x - mean_x
                dy = y - (sum(y_list) / len(y_list))
                cov += dx * dy
                var_x += dx ** 2
                var_y += dy ** 2
            pearson_r = cov / math.sqrt(var_x * var_y) if var_x and var_y else None
        if r2 is None:
            reliability = 'Невозможно вычислить R²'
        elif r2 >= 0.95:
            reliability = 'Высокая точность аппроксимации (модель хорошо описывает
явление)'
        elif r2 >= 0.75:
            reliability = 'Удовлетворительная аппроксимация (модель в целом
адекватно описывает явление)'
        elif r2 >= 0.5:
            reliability = 'Слабая аппроксимация (модель слабо описывает явление)'
        else:
            reliability = 'Точность аппроксимации недостаточна и модель требует
изменения'
        stats.append({
            'name': name,
            'r2': r2,
            'pearson_r': pearson_r,
            'reliability': reliability
        })
    return stats
```

Примеры и результаты работы программы

Лабораторная работа №4

Значения X (через пробел 8-12 чисел):

Значения Y (через пробел 8-12 чисел):

Или загрузите файл (txt):

Выберите файл test_2.txt

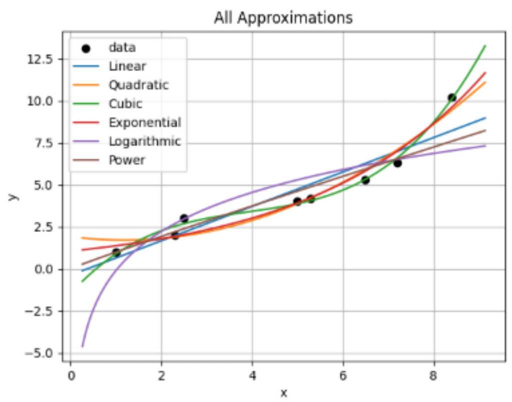
Формат файла:

- Только текстовые файлы (.txt)
- Первая строка - значения X через пробел
- Вторая строка - значения Y через пробел
- Пример:
  1.2 3.4 5.6 7.8 9.0 2.1 4.3 6.5 8.7 0.9
  1.2 3.4 5.6 7.8 9.0 2.1 4.3 6.5 8.7 0.9

Рассчитать

Результаты анализа

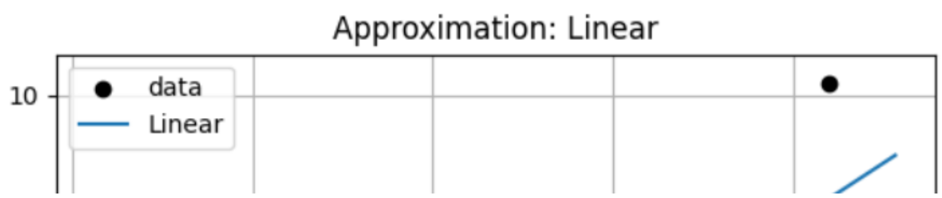
Комбинированный график



Скачать отчет

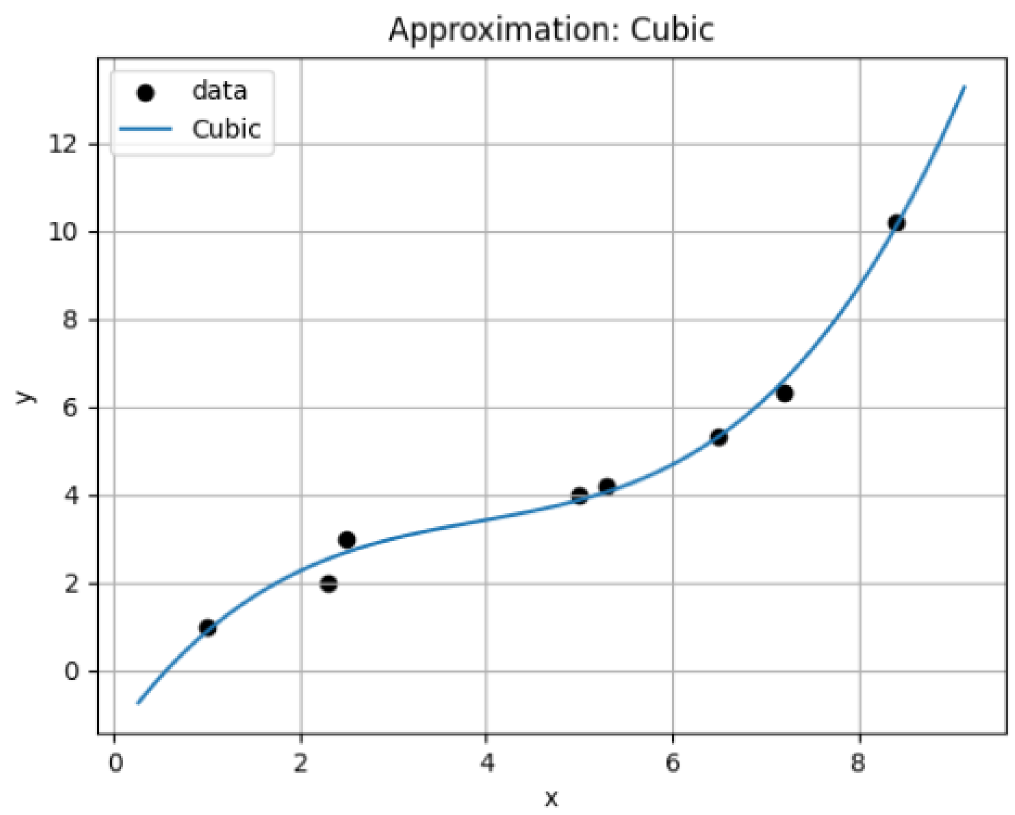
Linear Quadratic Cubic ★ Exponential Logarithmic Power

Linear Model



Linear | Quadratic | Cubic ★ | Exponential | Logarithmic | Power

Cubic Model ★ Лучшая модель



Параметры модели

Коэффициенты: [0.05557866382285918, -0.6543762342226507, 2.9487890968643438, -1.4656918641747616]
MSE: 0.06655625990479004
R^2: 0.9907335523975231
Высокая точность аппроксимации (модель хорошо описывает явление)

Таблица данных

x_i	y_i	y_pred	error
1.0000	1.0000	0.8843	0.1157
2.3000	2.0000	2.5311	-0.5311
2.5000	3.0000	2.6848	0.3152
5.0000	4.0000	3.8662	0.1338
5.3000	4.2000	4.0558	0.1442
6.5000	5.3000	5.3173	-0.0173
7.2000	6.3000	6.5874	-0.2874
8.4000	10.2000	10.0730	0.1270

Выводы:

В ходе работы мы реализовали шесть методов аппроксимации (линейный, квадратичный, кубический, экспоненциальный, логарифмический и степенной), вычислили для каждой модели MSE, а для линейной — ещё и коэффициент Пирсона. По итогам тестов степенная функция наиболее точная и универсальная, но для каждой из функций можно подобрать тест, где она покажет себя лучше всего. В итоге получен удобный инструмент для быстрого подбора и визуализации оптимальной модели приближения данных.