Федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Лабораторная работа № 4

«Аппроксимация функции методом наименьших квадратов»

По дисциплине «Вычислительная математика»

Вариант 12

Выполнила:

Студентка группы P3217

Русакова Е.Д.

Преподаватель:

Малышева Т.А.

Санкт-Петербург

2024

Оглавление

Цель работы:

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

Задание:

Вычислительная реализация задачи

Вычислительная часть лабораторной работы должна быть представлена только в отчете.

Задание:

1. Сформировать таблицу табулирования заданной функции на указанном интервале (см. табл. 1)
2. Построить линейное и квадратичное приближения по 11 точкам заданного интервала;
3. Найти среднеквадратические отклонения для каждой аппроксимирующей функции. Ответы дать с тремя знаками после запятой;
4. Выбрать наилучшее приближение;
5. Построить графики заданной функции, а также полученные линейное и квадратичное приближения;
6. Привести в отчете **подробные вычисления**.

Программная реализация задачи

Для исследования использовать:

- линейную функцию,
- полиномиальную функцию 2-й степени,
- полиномиальную функцию 3-й степени,
- экспоненциальную функцию,
- логарифмическую функцию,
- степенную функцию.

Методика проведения исследования:

1. Вычислить меру отклонения: $S = \sum_{i=1}^{n}[\varphi(x_i) - y_i]^2$ для всех исследуемых функций;
2. Уточнить значения коэффициентов эмпирических функций, минимизируя функцию S;
3. Сформировать массивы предполагаемых эмпирических зависимостей $(\varphi(x_i), \varepsilon_i)$;
4. Определить среднеквадратичное отклонение для каждой аппроксимирующей функции. Выбрать наименьшее значение и, следовательно, наилучшее приближение;
5. Построить графики полученных эмпирических функций.

Задание:

1. Предусмотреть ввод исходных данных из файла/консоли (таблица $y = f(x)$ должна содержать от 8 до 12 точек);
2. Реализовать метод наименьших квадратов, исследуя все указанные функции;
3. Предусмотреть вывод результатов в файл/консоль: коэффициенты аппроксимирующих функций, среднеквадратичное отклонение, массивы значений $\boldsymbol{x_i, y_i, \varphi(x_i), \varepsilon_i}$;
4. Для линейной зависимости вычислить коэффициент корреляции Пирсона;
5. Вычислить коэффициент детерминации, программа должна выводить соответствующее сообщение в зависимости от полученного значения R^2;
6. Программа должна отображать наилучшую аппроксимирующую функцию;
7. Организовать вывод графиков функций, графики должны полностью отображать весь исследуемый интервал (с запасом);
8. Программа должна быть протестирована при различных наборах данных, в том числе и некорректных;

Задание для варианта 12:

Для вычислительной реализации задачи:

$$y = \frac{4x}{x^4 + 12}, \qquad x \in [-2,0], \qquad h = 0.2$$

Рабочие формулы:

Мера отклонения – сумма квадратов отклонений:

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

Система уравнений для поиска коэффициентов приближенной функции:

$$\begin{cases} \frac{\partial S}{\partial a_0} = 0 \\ \frac{\partial S}{\partial a_1} = 0 \\ \dots \\ \frac{\partial S}{\partial a_n} = 0 \end{cases} \Rightarrow \begin{cases} a_0 n + a_1 \sum_{i=1}^{n} x_i + \dots + a_{m-1} \sum_{i=1}^{n} x_i^{m-1} + a_m \sum_{i=1}^{n} x_i^m = \sum_{i=1}^{n} y_i \\ a_0 \sum_{i=1}^{n} x_i + a_1 \sum_{i=1}^{n} x_i^2 + \dots + a_{m-1} \sum_{i=1}^{n} x_i^m + a_m \sum_{i=1}^{n} x_i^{m+1} = \sum_{i=1}^{n} x_i y_i \\ \dots \\ a_0 \sum_{i=1}^{n} x_i^m + a_1 \sum_{i=1}^{n} x_i^{m+1} + \dots + a_{m-1} \sum_{i=1}^{n} x_i^{2m-1} + a_m \sum_{i=1}^{n} x_i^{2m} = \sum_{i=1}^{n} x_i^m y_i \end{cases}$$

Коэффициент корреляции:

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

Среднеквадратичное отклонение:

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

Достоверность аппроксимации:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \varphi_i)^2}{\sum_{i=1}^{n}(y_i - \bar{\varphi})^2}, \quad \bar{\varphi} = \frac{1}{n}\sum_{i=1}^{n} \varphi_i$$

Вычислительная часть:

Таблица табулирования заданной функции $y = \frac{4x}{x^4+12}$ на интервале $x \in [-2,0]$

n	1	2	3	4	5	6	7	8	9	10	11
x_i	-2,0	-1,8	-1,6	-1,4	-1,2	-1,0	-0,8	-0,6	-0,4	-0,2	0,0
y_i	-0,286	-0,320	-0,345	-0,353	-0,341	-0,308	-0,258	-0,198	-0,133	-0,067	0,000

$SX = \sum_{i=1}^{n} x_i = -11$; $SX^2 = \sum_{i=1}^{n} x_i^2 = 15{,}4$; $SX^3 = \sum_{i=1}^{n} x_i^3 = -24{,}2$; $SX^4 = \sum_{i=1}^{n} x_i^4 = 40{,}533$;
$SY = \sum_{i=1}^{n} y_i = -2{,}608$; $SXY = \sum_{i=1}^{n} x_i y_i = 3{,}303$; $SX^2Y = \sum_{i=1}^{n} x_i^2 y_i = -4{,}815$

Линейное приближение:

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

$$S = \sum_{i=1}^{n}(ax_i + b - y_i)^2$$

$$\begin{cases} \frac{\partial S}{\partial a} = 0 \\ \frac{\partial S}{\partial b} = 0 \end{cases} \Rightarrow \begin{cases} a * SX^2 + b * SX = SXY \\ a * SX + b * n = SY \end{cases}$$

По правилу Крамера:

$$\Delta = SX^2 * n - SX * SX = 15{,}4 * 11 - (-11)^2 = 48{,}4$$

$$\Delta_1 = SXY * n - SX * SY = 3{,}303 * 11 - (-11) * (-2{,}608) = 7{,}645$$

$$\Delta_2 = SX^2 * SY - SX * SXY = 15{,}4 * (-2{,}608) - (-11) * 3{,}303 = -3{,}8302$$

$$a = \frac{\Delta_1}{\Delta} = \frac{7{,}645}{48{,}4} = 0{,}158$$

$$b = \frac{\Delta_2}{\Delta} = \frac{-3{,}8302}{48{,}4} = -0{,}079$$

$$\varphi(x) = 0{,}158x - 0{,}079$$

Среднеквадратичное отклонение:

$$\delta_1 = \sqrt{\frac{\sum_{i=1}^{n}(\varphi(x_i) - y_i)^2}{n}} = \sqrt{\frac{0{,}039}{11}} = 0{,}0595$$

Квадратичное приближение:

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

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

$$\begin{cases} a_0 * n + a_1 * SX + a_2 * SX^2 = SY \\ a_0 * SX + a_1 * SX^2 + a_2 * SX^3 = SXY \\ a_0 * SX^2 + a_1 * SX^3 + a_2 * SX^4 = SX^2 Y \end{cases} \Rightarrow \begin{cases} 11 * a_0 - 11 * a_1 + 15{,}4 * a_2 = -2{,}608 \\ -11 * a_0 + 15{,}4 * a_1 - 24{,}2 * a_2 = 3{,}303 \\ 15{,}4 * a_0 - 24{,}2 * a_1 + 40{,}533 * a_2 = -4{,}815 \end{cases}$$

По правилу Крамера:

$$\Delta = \begin{vmatrix} 11 & -11 & 15{,}4 \\ -11 & 15{,}4 & -24{,}2 \\ 15{,}4 & -24{,}2 & 40{,}533 \end{vmatrix} = 66{,}4532$$

$$\Delta_1 = \begin{vmatrix} -2{,}608 & -11 & 15{,}4 \\ 3{,}303 & 15{,}4 & -24{,}2 \\ -4{,}815 & -24{,}2 & 40{,}533 \end{vmatrix} = 1{,}3099834$$

$$\Delta_2 = \begin{vmatrix} 11 & -2{,}608 & 15{,}4 \\ -11 & 3{,}303 & -24{,}2 \\ 15{,}4 & -4{,}815 & 40{,}533 \end{vmatrix} = 32{,}392745$$

$$\Delta_3 = \begin{vmatrix} 11 & -11 & -2{,}608 \\ -11 & 15{,}4 & 3{,}303 \\ 15{,}4 & -24{,}2 & -4{,}815 \end{vmatrix} = 10{,}94808$$

$$a_0 = \frac{\Delta_1}{\Delta} = \frac{1{,}3099834}{66{,}4532} = 0{,}0197$$

$$a_1 = \frac{\Delta_2}{\Delta} = \frac{32{,}392745}{66{,}4532} = 0{,}487$$

$$a_2 = \frac{\Delta_3}{\Delta} = \frac{10{,}94808}{66{,}4532} = 0{,}165$$

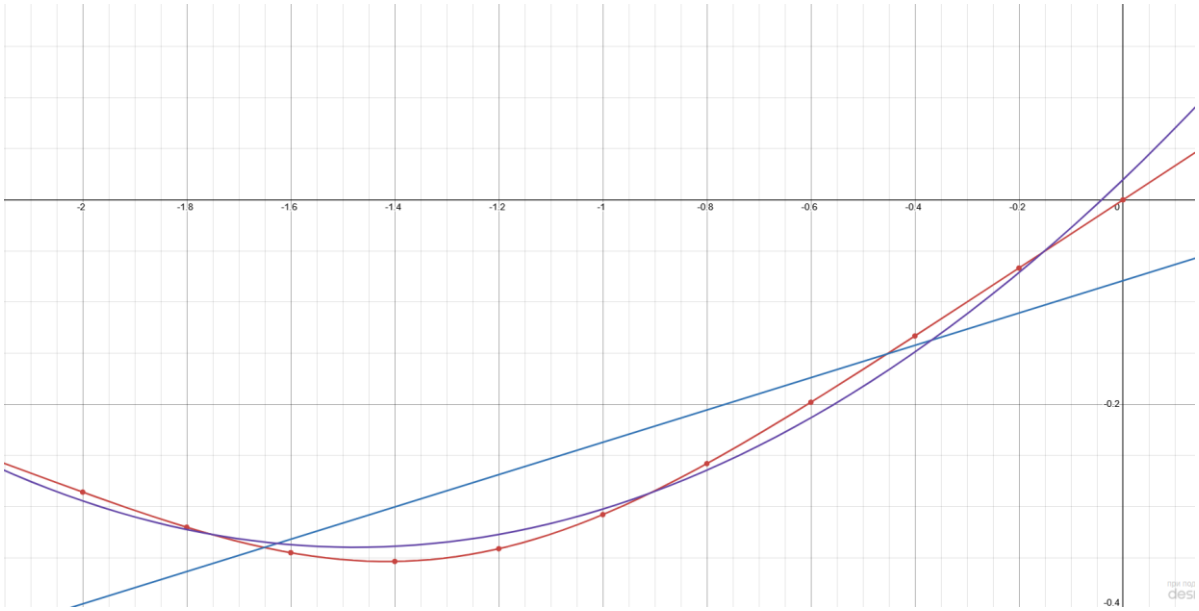
$$\varphi(x) = 0{,}0197 + 0{,}487 * x + 0{,}165 * x^2$$

Среднеквадратичное отклонение:

$$\delta_2 = \sqrt{\frac{\sum_{i=1}^{n}(\varphi(x_i) - y_i)^2}{n}} = \sqrt{\frac{0{,}002}{11}} = 0{,}0135$$

$\delta_2 < \delta_1$, следовательно, квадратичная аппроксимация является более точной.

Графики исходной функции и полученных приближений:



Красный – исходная функция, синий – линейная аппроксимация, фиолетовый – квадратичная аппроксимация.

Программная реализация:

Описание разработанной программы:

Разработанная программа позволяет найти 6 приближенных к исходной, функций, различного вида. Пользователь вводит исходные значения точек с клавиатуры или из файла. Программа находит аппроксимированные функции, меры отклонений, среднеквадратичное отклонение, достоверность для каждой аппроксимации и коэффициент корреляции для линейной аппроксимации, и определяет наилучшую аппроксимацию по значениям среднеквадратичных отклонений. Также программа строит графики исходной и полученных функций и выводит все результаты на экран или в файл по выбору пользователя.

Исходный код программы:

Полный код программы выложен на Github и доступен по ссылке [lenapochemy/comp-math-lab4: вычмат лаба 4 аппроксимация по мнк (github.com)]

Далее приведен код классов, которые отвечают за нахождение приближенных функций, и код абстрактного класса, в котором вычисляются общие для всех видов функций характеристики.

AbstractApproximation.java – класс с общими вычислениями

```
package approximations;

import java.io.FileWriter;
import java.io.IOException;
import java.math.BigDecimal;
import java.math.RoundingMode;
import java.util.function.DoubleFunction;

public abstract class AbstractApproximation {
    public final double[] x, y;
    public final int n;
    public double[] phi, eps; //значения аппроксимирующей функции и
отклонения
    public double S, standardDeviation, R2;
    public final DoubleFunction<Double> phiFunc; //аппроксимирующая функция
    public final boolean outputMode, negativeData;
    public final FileWriter file;
    public String name, coef, type;

    public AbstractApproximation(int n, double[] x, double[] y, String
name,String type, boolean outputMode, FileWriter file, boolean negativeData){
        this.outputMode = outputMode;
        this.negativeData = negativeData;
        this.file = file;
        this.n = n;
        this.x = x;
        this.y = y;
        this.name = name;
        this.type = type;
        writeResult("\n-------------------------------------------------");
        writeResult(name + ": " + type + "\n");
        this.phiFunc = findFunc();
        if(negativeData){
            writeResult("Данная аппроксимация не возможна с отрицательными
числами");
        } else {
            phi = new double[n];
            eps = new double[n];
            S = 0;
            double phi_sr = 0;

            for (int i = 0; i < n; i++) {
                phi[i] = phiFunc.apply(x[i]);
                eps[i] = phi[i] - y[i];
                S += eps[i] * eps[i];
                phi_sr += phi[i];
            }

            standardDeviation = Math.sqrt(S / n); //СКО

            phi_sr = phi_sr / n; //phi среднее
            double sum1 = 0, sum2 = 0;
            for(int i = 0; i < n; i++){
                sum1 += Math.pow((y[i] - phi[i]), 2);
                sum2 += Math.pow((y[i] - phi_sr), 2);
            }
            R2 = 1 - sum1 / sum2; //достоверность
```

```
            printAllResults();
        }
    }

    public abstract DoubleFunction<Double> findFunc();

    public void printAllResults(){
        writeResult("Коэффициенты: " + coef);
        writeResult("Мера отклонения: " + S);
        writeResult("Среднеквадратичное отклонение: " + standardDeviation);
        String accuracy;
        if(R2 >= 0.95) {
            accuracy = "Высокая точность аппроксимации";
        } else if(R2 >= 0.75){
            accuracy = "Удовлетворительная точность аппроксимации";
        } else if(R2 >= 0.5){
            accuracy = "Слабая точность аппроксимации";
        } else {
            accuracy = "Недостаточная точность аппроксимации";
        }
        writeResult("Достоверность аппроксимации: " + R2 + " -> " + accuracy
+ "\n");
        String res = String.format("%-6s|", "x_i");
        for(int i = 0; i < n; i++){
            res += String.format(" %-20s|", rounding(x[i]));
        }
        writeResult(res);
        res = String.format("%-6s|", "y_i");
        for(int i = 0; i < n; i++){
            res += String.format(" %-20s|", rounding(y[i]));
        }
        writeResult(res);
        res = String.format("%-6s|", "phi_i");
        for(int i = 0; i < n; i++){
            res += String.format(" %-20s|", rounding(phi[i]));
        }
        writeResult(res);
        res = String.format("%-6s|", "eps_i");
        for(int i = 0; i < n; i++){
            res += String.format(" %-20s|", rounding(eps[i]));
        }
        writeResult(res);
    }
    private double rounding(double number){
        BigDecimal help = new BigDecimal(number);
        help = help.setScale(15, RoundingMode.HALF_UP);
        return help.doubleValue();
    }
    public void writeResult(String string){
        if(outputMode){
            System.out.println(string);
        } else {
            try {
                file.write(string + "\n");
            } catch (IOException e){
                System.out.println("Проблемы с файлом");
            }
        }
    }
```

```
}
```

LinearApproximation.java – линейная аппроксимация

```
package approximations;

import java.io.FileWriter;
import java.util.function.DoubleFunction;

public class LinearApproximation extends AbstractApproximation{

    public double r;

    public LinearApproximation(int n, double[] x, double[] y, boolean outputMode,
FileWriter file){
        super(n, x, y, "Линейная аппроксимация", "ax+b", outputMode, file, false);
        double x_sr = 0;
        double y_sr = 0;
        for(int i = 0; i < n; i++){
            x_sr += x[i];
            y_sr += y[i];
        }
        x_sr = x_sr / n;
        y_sr = y_sr / n;

        double s1 = 0, s2 = 0, s3 = 0;
        for(int i = 0; i < n; i++){
            s1 += (x[i] - x_sr) * (y[i] - y_sr);
            s2 += Math.pow((x[i] - x_sr), 2);
            s3 += Math.pow((y[i] - y_sr), 2);
        }
        r = s1 / Math.sqrt(s2 * s3);
        writeResult("\nКоэффициент корреляции: " + r);
    }

    @Override
    public DoubleFunction<Double> findFunc() {
        double sx = 0, sx2 = 0, sy = 0, sxy = 0;
        for (int i = 0; i < n; i++) {
            sx += x[i];
            sx2 += x[i] * x[i];
            sy += y[i];
            sxy += x[i] * y[i];
        }

        double det = sx2 * n - sx * sx;
        double det1 = sxy * n - sx * sy;
        double det2 = sx2 * sy - sx * sxy;
        double a = det1 / det;
        double b = det2 / det;
        coef = "a = " + a + " b = " + b;
        return x -> a * x + b;
    }
}
```

QuadraticApproximation.java – квадратичная аппроксимация

```
package approximations;

import utils.Det;

import java.io.FileWriter;
```

```java
import java.util.function.DoubleFunction;

public class QuadraticApproximation extends AbstractApproximation{

    public QuadraticApproximation(int n, double[] x, double[] y, boolean outputMode,
FileWriter file){
        super(n, x, y, "Квадратичная аппроксимация", "a2*x²+a1*x+a0", outputMode,
file, false);
    }

    @Override
    public DoubleFunction<Double> findFunc(){
        double sx = 0, sx2 = 0, sx3 = 0, sx4 = 0, sy = 0, sxy = 0, sx2y = 0;

        for (int i = 0; i < n; i++) {
            sx += x[i];
            sx2 += x[i] * x[i];
            sx3 += Math.pow(x[i], 3);
            sx4 += Math.pow(x[i], 4);
            sy += y[i];
            sxy += x[i] * y[i];
            sx2y += x[i] * x[i] * y[i];
        }

        double det = Det.determinant(new double[][]{{n, sx, sx2},
                                                    {sx, sx2, sx3},
                                                    {sx2, sx3, sx4}});
        double det1 = Det.determinant(new double[][]{{sy, sx, sx2},
                                                     {sxy, sx2, sx3},
                                                     {sx2y, sx3, sx4}});
        double det2 = Det.determinant(new double[][]{{n, sy, sx2},
                                                     {sx, sxy, sx3},
                                                     {sx2, sx2y, sx4}});
        double det3 = Det.determinant(new double[][]{{n, sx, sy},
                                                     {sx, sx2, sxy},
                                                     {sx2, sx3, sx2y}});
        double a0 = det1 / det;
        double a1 = det2 / det;
        double a2 = det3 / det;
        coef = "a2 = " + a2 + " a1 = " + a1 + " a0 = " + a0;
        return x -> a0 + a1 * x + a2 * x * x;
    }
}
```

CubicApproximation.java – кубическая аппроксимация

```java
package approximations;

import utils.Det;

import java.io.FileWriter;
import java.util.function.DoubleFunction;

public class CubicApproximation extends AbstractApproximation{

    public CubicApproximation(int n, double[] x, double[] y, boolean outputMode,
FileWriter file){
        super(n, x, y, "Кубическая аппроксимация", "a3*x³+a2*x²+a1*x+a0", outputMode,
file, false);
    }
```

```
    @Override
    public DoubleFunction<Double> findFunc(){
        double sx = 0, sx2 = 0, sx3 = 0, sx4 = 0, sx5 = 0, sx6 = 0,  sy = 0, sxy = 0,
sx2y = 0, sx3y = 0;

        for (int i = 0; i < n; i++) {
            sx += x[i];
            sx2 += x[i] * x[i];
            sx3 += Math.pow(x[i], 3);
            sx4 += Math.pow(x[i], 4);
            sx5 += Math.pow(x[i], 5);
            sx6 += Math.pow(x[i], 6);
            sy += y[i];
            sxy += x[i] * y[i];
            sx2y += x[i] * x[i] * y[i];
            sx3y += Math.pow(x[i], 3) * y[i];
        }

        double det = Det.determinant(new double[][]{
                {n, sx, sx2, sx3},
                {sx, sx2, sx3, sx4},
                {sx2, sx3, sx4, sx5},
                {sx3, sx4, sx5, sx6}});
        double det1 = Det.determinant(new double[][]{
                {sy, sx, sx2, sx3},
                {sxy, sx2, sx3, sx4},
                {sx2y, sx3, sx4, sx5},
                {sx3y, sx4, sx5, sx6}});
        double det2 = Det.determinant(new double[][]{
                {n, sy, sx2, sx3},
                {sx, sxy, sx3, sx4},
                {sx2, sx2y, sx4, sx5},
                {sx3, sx3y, sx5, sx6}});
        double det3 = Det.determinant(new double[][]{
                {n, sx, sy, sx3},
                {sx, sx2, sxy, sx4},
                {sx2, sx3, sx2y, sx5},
                {sx3, sx4, sx3y, sx6}});
        double det4 = Det.determinant(new double[][]{
                {n, sx, sx2, sy},
                {sx, sx2, sx3, sxy},
                {sx2, sx3, sx4, sx2y},
                {sx3, sx4, sx5, sx3y}});
        double a0 = det1 / det;
        double a1 = det2 / det;
        double a2 = det3 / det;
        double a3 = det4 / det;
        coef = "a3 = " + a3 + " a2 = " + a2 + " a1 = " + a1 + " a0 = " + a0;
        return x -> a0 + a1 * x + a2 * x * x + a3 * x * x * x;
    }
}
```

PowerApproximation.java – степенная аппроксимация

```
package approximations;

import utils.Det;

import java.io.FileWriter;
import java.util.function.DoubleFunction;
```

```
public class PowerApproximation extends AbstractApproximation{


    public PowerApproximation(int n, double[] x, double[] y, boolean outputMode,
FileWriter file, boolean negativeData){
        super(n, x, y, "Степенная аппроксимация", "a*x^b", outputMode, file,
negativeData);
    }

    @Override
    public DoubleFunction<Double> findFunc(){
        if(!negativeData) {
            double sx = 0, sx2 = 0, sy = 0, sxy = 0;
            for (int i = 0; i < n; i++) {
                double xi = Math.log(x[i]);
                double yi = Math.log(y[i]);
                sx += xi;
                sx2 += xi * xi;
                sy += yi;
                sxy += xi * yi;
            }

            double det = Det.determinant(new double[][]{{n, sx}, {sx, sx2}});
            double det1 = Det.determinant(new double[][]{{sy, sx}, {sxy, sx2}});
            double det2 = Det.determinant(new double[][]{{n, sy}, {sx, sxy}});
            double a = Math.exp(det1 / det);
            double b = det2 / det;

            coef = "a = " + a + " b = " + b;
            return x -> a * Math.pow(x, b);
        } else return null;
    }
}
```

LogarithmicApproximation.java – логарифмическая аппроксимация

```
package approximations;

import utils.Det;

import java.io.FileWriter;
import java.util.function.DoubleFunction;

public class LogarithmicApproximation extends AbstractApproximation{
    public LogarithmicApproximation(int n, double[] x, double[] y, boolean outputMode,
FileWriter file, boolean negativeData){
        super(n, x, y, "Логарифмическая аппроксимация", "a*ln(x)+b", outputMode, file,
negativeData);
    }

    @Override
    public DoubleFunction<Double> findFunc(){
        if(!negativeData) {
            double sx = 0, sx2 = 0, sy = 0, sxy = 0;
            for (int i = 0; i < n; i++) {
                double xi = Math.log(x[i]);
                double yi = y[i];
                sx += xi;
                sx2 += xi * xi;
                sy += yi;
                sxy += xi * yi;
```

```
            }

            double det = Det.determinant(new double[][]{{n, sx}, {sx, sx2}});
            double det1 = Det.determinant(new double[][]{{sy, sx}, {sxy, sx2}});
            double det2 = Det.determinant(new double[][]{{n, sy}, {sx, sxy}});
            double b = det1 / det;
            double a = det2 / det;

            coef = "a = " + a + " b = " + b;
            return x -> a * Math.log(x) + b;
        } else return null;
    }
}
```

ExponentialApproximation.java – экспоненциальная аппроксимация

```
package approximations;

import utils.Det;

import java.io.FileWriter;
import java.util.function.DoubleFunction;

public class ExponentialApproximation extends AbstractApproximation{
    public ExponentialApproximation(int n, double[] x, double[] y, boolean outputMode,
FileWriter file, boolean negativeData){
        super(n, x, y,"Экспоненциальная аппроксимация", "a*e^(bx)", outputMode, file,
negativeData);
    }

    @Override
    public DoubleFunction<Double> findFunc(){
        if(!negativeData) {
            double sx = 0, sx2 = 0, sy = 0, sxy = 0;
            for (int i = 0; i < n; i++) {
                double xi = x[i];
                double yi = Math.log(y[i]);
                sx += xi;
                sx2 += xi * xi;
                sy += yi;
                sxy += xi * yi;
            }

            double det = Det.determinant(new double[][]{{n, sx}, {sx, sx2}});
            double det1 = Det.determinant(new double[][]{{sy, sx}, {sxy, sx2}});
            double det2 = Det.determinant(new double[][]{{n, sy}, {sx, sxy}});
            double a = Math.exp(det1 / det);
            double b = det2 / det;

            coef = "a = " + a + " b = " + b;
            return x -> a * Math.exp(b * x);
        } return null;
    }
}
```

Примеры работы программы:

Пример 1

```
Вы хотите вводить данные с клавиатуры или из файла? (k/f) f
Введите путь к файлу:
C:\Users\Elena\IdeaProjects\compMath\lab4\src\files\data2
Введите количество точек от 7 до 12: 8
Введите исходные точки парами (x, y) через пробел
1.2 7.4
2.9 9.5
4.1 11.1
5.5 12.9
6.7 14.6
7.8 17.3
9.2 18.2
10.3 20.7
Результаты вывести на экран или в записать в файл? (s/f) f
Введите путь к файлу:
C:\Users\Elena\IdeaProjects\compMath\lab4\src\files\res2
```

data2 | data5

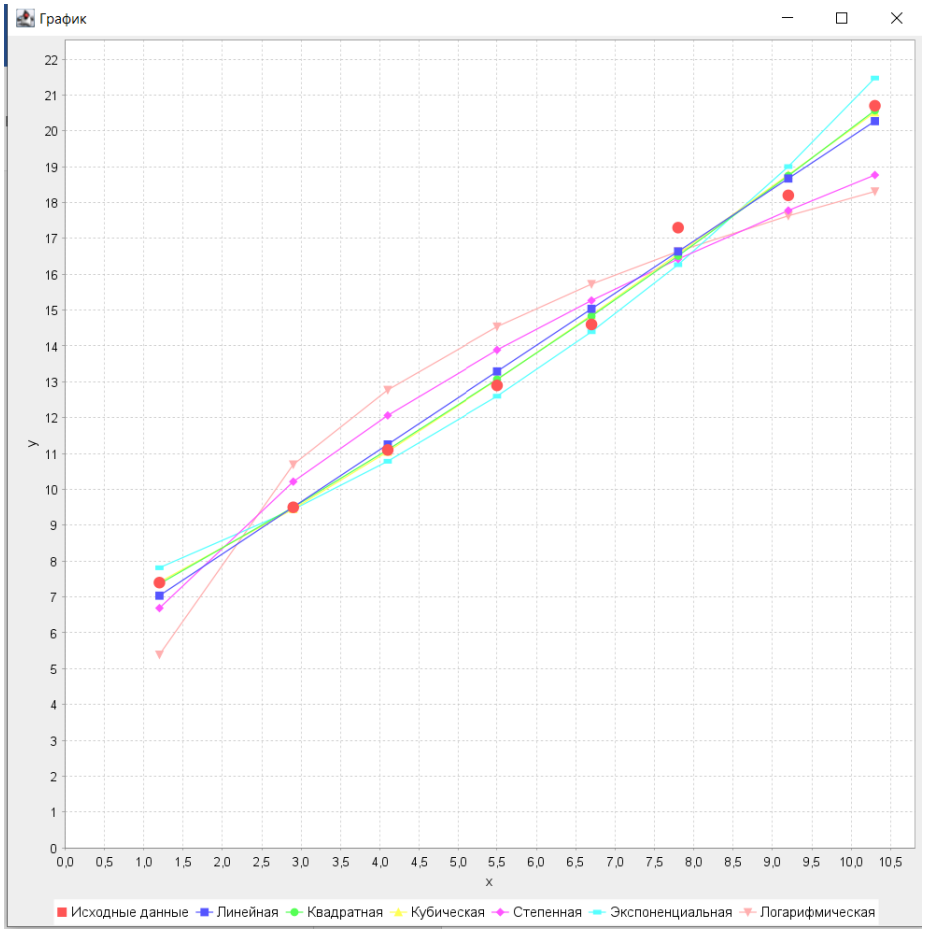
```
8
1.2 7.4
2.9 9.5
4.1 11.1
5.5 12.9
6.7 14.6
7.8 17.3
9.2 18.2
10.3 20.7
```

data2 | res2 | Main.java

```
--------------------------------------------------
Линейная аппроксимация: ax+b

Коэффициенты: a = 1.4543295810901447 b = 5.29105987275002
Мера отклонения: 1.34585422444577
Среднеквадратичное отклонение: 0.41016067346312135
Достоверность аппроксимации: 0.9908568909420374 -> Высокая точность аппроксимации

x_i   | 1.2               | 2.9               | 4.1               | 5.5               | 6.7               | 7.8               | 9.2               | 10.3              |
y_i   | 7.4               | 9.5               | 11.1              | 12.9              | 14.6              | 17.3              | 18.2              | 20.7              |
phi_i | 7.036255370058194 | 9.50861565791144  | 11.253811155219612 | 13.289872568745816 | 15.03506806605399 | 16.63483060525315 | 18.670892018779348 | 20.27065455797851 |
eps_i | -0.363744629941806 | 0.00861565791144  | 0.153811155219612 | 0.389872568745815 | 0.43506806605399  | -0.665169394746851 | 0.470892018779349 | -0.429345442021489 |

Коэффициент корреляции: 0.9954179478701582

--------------------------------------------------
Квадратичная аппроксимация: a2*x²+a1*x+a0

Коэффициенты: a2 = 0.025973674183274002 a1 = 1.1525630586368418 a0 = 5.943053187109902
Мера отклонения: 1.0158903343995724
Среднеквадратичное отклонение: 0.3563513600366169
Достоверность аппроксимации: 0.9930985124914473 -> Высокая точность аппроксимации

x_i   | 1.2               | 2.9               | 4.1               | 5.5               | 6.7               | 7.8               | 9.2               | 10.3              |
y_i   | 7.4               | 9.5               | 11.1              | 12.9              | 14.6              | 17.3              | 18.2              | 20.7              |
phi_i | 7.363530868298027 | 9.503924577038077 | 11.105179110541789 | 13.067853573656572 | 14.831183834063914 | 16.51328330178766 | 18.745045029441158 | 20.569999705172915 |
eps_i | -0.036469131701973 | 0.003924577038077 | 0.005179110541789 | 0.167853573656572 | 0.231183834063915 | -0.786716698212341 | 0.545045029441159 | -0.130000294827084 |

--------------------------------------------------
Кубическая аппроксимация: a3*x³+a2*x²+a1*x+a0

Коэффициенты: a3 = -0.0023713802937302907 a2 = 0.06687472776417426 a1 = 0.9547538603099531 a0 = 6.1778789143426875
Мера отклонения: 0.9995870041086208
Среднеквадратичное отклонение: 0.3534803750048616
Достоверность аппроксимации: 0.9932092697518925 -> Высокая точность аппроксимации

x_i   | 1.2               | 2.9               | 4.1               | 5.5               | 6.7               | 7.8               | 9.2               | 10.3              |
y_i   | 7.4               | 9.5               | 11.1              | 12.9              | 14.6              | 17.3              | 18.2              | 20.7              |
phi_i | 7.415785409547476 | 9.45124597575447  | 11.05309601410508 | 13.057447264544324 | 14.863512856469955 | 16.568274200782387 | 18.775326008989712 | 20.515312269809435 |
eps_i | 0.015785409547476 | -0.048754024245531 | -0.04690398589492 | 0.157447264544324 | 0.263512856469955 | -0.731725799217614 | 0.575326008989713 | -0.184687730190564 |
```

data2 | res2 | Main.java

```
-----------------------------------------------
Степенная аппроксимация: a*x^b

Коэффициенты: a = 6.128671326925076 b = 0.47989414784078466
Мера отклонения: 8.046183833187773
Среднеквадратичное отклонение: 1.0028823356448484
Достоверность аппроксимации: 0.9453549748702458 -> Удовлетворительная точность аппроксимации

x_i   | 1.2                | 2.9                | 4.1                | 5.5                | 6.7                | 7.8                | 9.2                | 10.3               |
y_i   | 7.4                | 9.5                | 11.1               | 12.9               | 14.6               | 17.3               | 18.2               | 20.7               |
phi_i | 6.689057787663377  | 10.215706909944638 | 12.062510878866085 | 13.888714587459742 | 15.268444500323294 | 16.423938511963527 | 17.777982297088284 | 18.768129359102097 |
eps_i | -0.710942212336623 | 0.7157069099446638 | 0.962510878866086  | 0.988714587459741  | 0.668444500323295  | -0.876061488036473 | -0.422017702917159 | -1.931870640897902 |

-----------------------------------------------
Экспоненциальная аппроксимация: a*e^(bx)

Коэффициенты: a = 6.839635206780496 b = 0.11107656939440223
Мера отклонения: 2.7188698653067 58
Среднеквадратичное отклонение: 0.5829740415861968
Достоверность аппроксимации: 0.9815293565436891 -> Высокая точность аппроксимации

x_i   | 1.2                | 2.9                | 4.1                | 5.5                | 6.7                | 7.8                | 9.2                | 10.3               |
y_i   | 7.4                | 9.5                | 11.1               | 12.9               | 14.6               | 17.3               | 18.2               | 20.7               |
phi_i | 7.814853995829316  | 9.439060473706753  | 10.784914287050675 | 12.59947982707062  | 14.395957137354634 | 16.266888138117974 | 19.003797664995642 | 21.47357399486653  |
eps_i | 0.414853995829316  | -0.060939526293247 | -0.315085712949324 | -0.30052017292938  | -0.204042862645366 | -1.033111861882027 | 0.803797664995642  | 0.773573994866531  |

-----------------------------------------------
Логарифмическая аппроксимация: a*ln(x)+b

Коэффициенты: a = 6.008624101281969 b = 4.29586618473261
Мера отклонения: 18.692764043319315
Среднеквадратичное отклонение: 1.5285926551619022
Достоверность аппроксимации: 0.8730100354566916 -> Удовлетворительная точность аппроксимации

x_i   | 1.2                | 2.9                | 4.1                | 5.5                | 6.7                | 7.8                | 9.2                | 10.3               |
y_i   | 7.4                | 9.5                | 11.1               | 12.9               | 14.6               | 17.3               | 18.2               | 20.7               |
phi_i | 5.391367805068015  | 10.693312699919002 | 12.773956441562998 | 14.539056578370087 | 15.724915231070975 | 16.638323478030976 | 17.630225644674 37 | 18.30884212130279  |
eps_i | -2.008632194931986 | 1.193312699919002  | 1.673956441562998  | 1.639056578370869  | 1.124915231070975  | -0.661676521969024 | -0.569774355325631 | -2.391157878697211 |

-----------------------------------------------

Наилучшее приближение - Кубическая аппроксимация
```



Пример 2

```
Вы хотите вводить данные с клавиатуры или из файла? (k/f) f
Введите путь к файлу:
C:\Users\Elena\IdeaProjects\compMath\lab4\src\files\data1
Введите количество точек от 7 до 12: 7
Введите исходные точки парами (x, y) через пробел
-1.1 2.73
2.3 5.12
3.7 7.74
4.5 8.91
5.4 10.59
6.8 12.75
7.5 13.43
Результаты вывести на экран или в записать в файл? (s/f) f
Введите путь к файлу:
C:\Users\Elena\IdeaProjects\compMath\lab4\src\files\res1
```

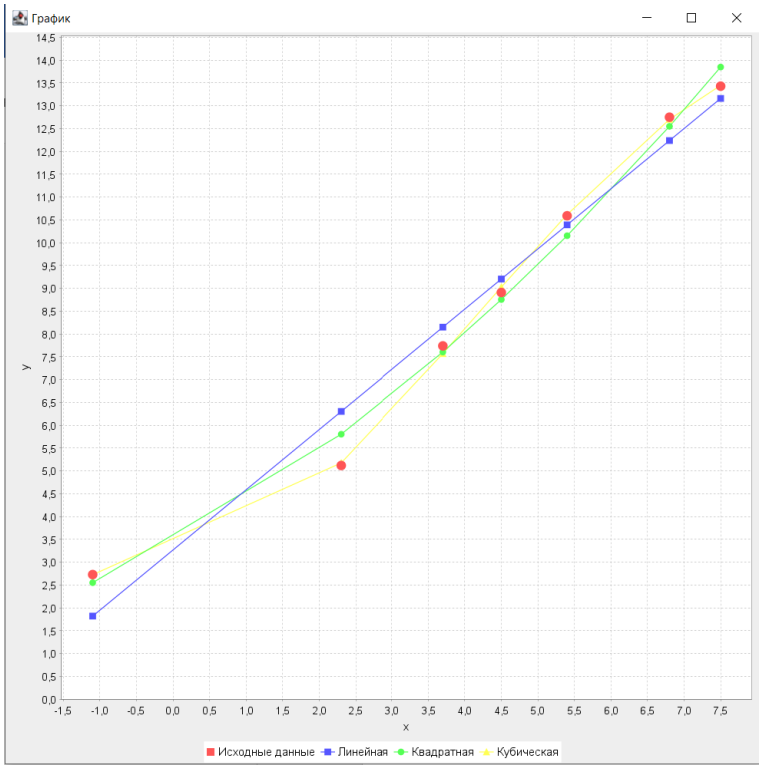
data1

```
7
-1.1 2.73
2.3 5.12
3.7 7.74
4.5 8.91
5.4 10.59
6.8 12.75
7.5 13.43
```

data1 res1

```
--------------------------------------------------
Линейная аппроксимация: ax+b

Коэффициенты: a = 1.3187133195461396 b = 3.270777485886759
Мера отклонения: 2.8579850874741486
Среднеквадратичное отклонение: 0.6389707222751903
Достоверность аппроксимации: 0.9688506188767557 -> Высокая точность аппроксимации

x_i   | -1.1               | 2.3               | 3.7               | 4.5               | 5.4               | 6.8               | 7.5               |
y_i   | 2.73               | 5.12              | 7.74              | 8.91              | 10.59             | 12.75             | 13.43             |
phi_i | 1.820192834386005  | 6.30381812084288  | 8.150016768207475 | 9.204987423844386 | 10.391829411435914 | 12.238028058800507 | 13.161127382482805 |
eps_i | -0.909807165613995 | 1.18381812084288  | 0.410016768207475 | 0.294987423844386 | -0.198170588564086 | -0.511971941199493 | -0.268872617517195 |

Коэффициент корреляции: 0.9843020973648058

--------------------------------------------------
Квадратичная аппроксимация: a2*x²+a1*x+a0

Коэффициенты: a2 = 0.06860643625262437 a1 = 0.8744882745535918 a0 = 3.4308442279680467
Мера отклонения: 0.9487938195509027
Среднеквадратичное отклонение: 0.36816025616782366
Достоверность аппроксимации: 0.989659029215338 -> Высокая точность аппроксимации

x_i   | -1.1               | 2.3               | 3.7               | 4.5               | 5.4               | 6.8               | 7.5               |
y_i   | 2.73               | 5.12              | 7.74              | 8.91              | 10.59             | 12.75             | 13.43             |
phi_i | 2.551920913824771  | 5.80509530721769  | 7.605672956114764 | 8.755321797574853 | 10.15364459168397 | 12.549726107253823 | 13.848618326330106 |
eps_i | -0.178079086175229 | 0.68509530721769  | -0.134327043885236 | -0.154678202425147 | -0.436355408316031 | -0.200273892746177 | 0.418618326330106 |

--------------------------------------------------
Кубическая аппроксимация: a3*x³+a2*x²+a1*x+a0

Коэффициенты: a3 = -0.027592932681470346 a2 = 0.3412295237245461 a1 = 0.42067827396223684 a0 = 2.7398434791200836
Мера отклонения: 0.04939454443802542
Среднеквадратичное отклонение: 0.08400216360991239
Достоверность аппроксимации: 0.9994616453749698 -> Высокая точность аппроксимации

x_i   | -1.1               | 2.3               | 3.7               | 4.5               | 5.4               | 6.8               | 7.5               |
y_i   | 2.73               | 5.12              | 7.74              | 8.91              | 10.59             | 12.75             | 13.43             |
phi_i | 2.726711294867361  | 5.176784477800628 | 7.57012045345488  | 9.028387576773223 | 10.616865518568881 | 12.702807910186223 | 13.448322768347278 |
eps_i | -0.0032887051326390 | 0.056784477800628 | -0.16987954654512 | 0.1183875767732 | 0.026865518568881 | -0.047192089813777 | 0.018322768347279 |
```

```
-----------------------------------------------
Степенная аппроксимация: a*x^b

Данная аппроксимация не возможна с отрицательными числами

-----------------------------------------------
Экспоненциальная аппроксимация: a*e^(bx)

Данная аппроксимация не возможна с отрицательными числами

-----------------------------------------------
Логарифмическая аппроксимация: a*ln(x)+b

Данная аппроксимация не возможна с отрицательными числами

-----------------------------------------------

Наилучшее приближение - Кубическая аппроксимация
```



Пример 3

```
Вы хотите вводить данные с клавиатуры или из файла? (k/f) f
Введите путь к файлу:
C:\Users\Elena\IdeaProjects\compMath\lab4\src\files\data4
Введите количество точек от 7 до 12: 7
Введите исходные точки парами (x, y) через пробел
1 1
2 4
3 9
4 16
5 25
6 36
7 49
Результаты вывести на экран или в записать в файл? (s/f) f
Введите путь к файлу:
C:\Users\Elena\IdeaProjects\compMath\lab4\src\files\res4
```

data4 × res4

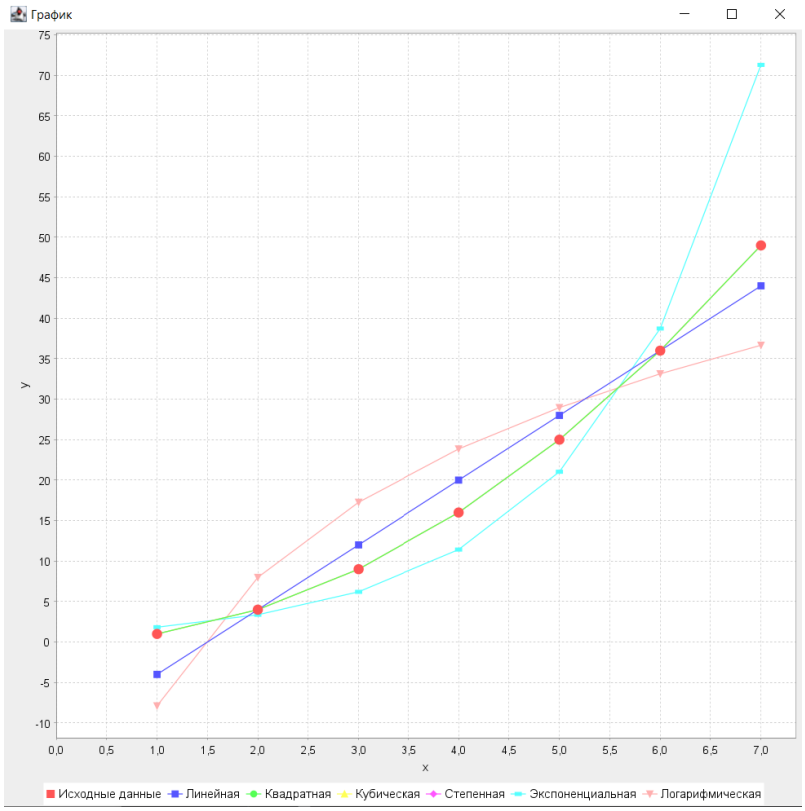
```
7
1 1
2 4
3 9
4 16
5 25
6 36
7 49
```

data4 res4 ×

```
-------------------------------------------------
Линейная аппроксимация: ax+b

Коэффициенты: a = 8.0 b = -12.0
Мера отклонения: 84.0
Среднеквадратичное отклонение: 3.4641016151377544
Достоверность аппроксимации: 0.9552238805970149 -> Высокая точность аппроксимации

x_i   | 1.0               | 2.0               | 3.0               | 4.0               | 5.0               | 6.0               | 7.0               |
y_i   | 1.0               | 4.0               | 9.0               | 16.0              | 25.0              | 36.0              | 49.0              |
phi_i | -4.0              | 4.0               | 12.0              | 20.0              | 28.0              | 36.0              | 44.0              |
eps_i | -5.0              | 0.0               | 3.0               | 4.0               | 3.0               | 0.0               | -5.0              |

Коэффициент корреляции: 0.9773555548504418

-------------------------------------------------
Квадратичная аппроксимация: a2*x²+a1*x+a0

Коэффициенты: a2 = 1.0 a1 = 0.0 a0 = 0.0
Мера отклонения: 0.0
Среднеквадратичное отклонение: 0.0
Достоверность аппроксимации: 1.0 -> Высокая точность аппроксимации

x_i   | 1.0               | 2.0               | 3.0               | 4.0               | 5.0               | 6.0               | 7.0               |
y_i   | 1.0               | 4.0               | 9.0               | 16.0              | 25.0              | 36.0              | 49.0              |
phi_i | 1.0               | 4.0               | 9.0               | 16.0              | 25.0              | 36.0              | 49.0              |
eps_i | 0.0               | 0.0               | 0.0               | 0.0               | 0.0               | 0.0               | 0.0               |

-------------------------------------------------
Кубическая аппроксимация: a3*x³+a2*x²+a1*x+a0

Коэффициенты: a3 = 0.0 a2 = 1.0 a1 = 0.0 a0 = 0.0
Мера отклонения: 0.0
Среднеквадратичное отклонение: 0.0
Достоверность аппроксимации: 1.0 -> Высокая точность аппроксимации

x_i   | 1.0               | 2.0               | 3.0               | 4.0               | 5.0               | 6.0               | 7.0               |
y_i   | 1.0               | 4.0               | 9.0               | 16.0              | 25.0              | 36.0              | 49.0              |
phi_i | 1.0               | 4.0               | 9.0               | 16.0              | 25.0              | 36.0              | 49.0              |
eps_i | 0.0               | 0.0               | 0.0               | 0.0               | 0.0               | 0.0               | 0.0               |
```

data4 res4

```
------------------------------------------------
Степенная аппроксимация: a*x^b

Коэффициенты: a = 1.0 b = 2.0
Мера отклонения: 0.0
Среднеквадратичное отклонение: 0.0
Достоверность аппроксимации: 1.0 -> Высокая точность аппроксимации

x_i   | 1.0               | 2.0               | 3.0               | 4.0               | 5.0               | 6.0               | 7.0               |
y_i   | 1.0               | 4.0               | 9.0               | 16.0              | 25.0              | 36.0              | 49.0              |
phi_i | 1.0               | 4.0               | 9.0               | 16.0              | 25.0              | 36.0              | 49.0              |
eps_i | 0.0               | 0.0               | 0.0               | 0.0               | 0.0               | 0.0               | 0.0               |

------------------------------------------------
Экспоненциальная аппроксимация: a*e^(bx)

Коэффициенты: a = 0.9941260943232314 b = 0.610412903447725
Мера отклонения: 550.6212414227206
Среднеквадратичное отклонение: 8.869057297486748
Достоверность аппроксимации: 0.7107481233107174 -> Слабая точность аппроксимации

x_i   | 1.0               | 2.0               | 3.0               | 4.0               | 5.0               | 6.0               | 7.0               |
y_i   | 1.0               | 4.0               | 9.0               | 16.0              | 25.0              | 36.0              | 49.0              |
phi_i | 1.830376491096148 | 3.370073593569845 | 6.204950774512638 | 11.42450247602495 | 21.03469657822574 | 38.728904043440785 | 71.30732800580105 |
eps_i | 0.830376491096148 | -0.629926406430155 | -2.795049225487362 | -4.57549752397505 | -3.965303421774259 | 2.728904043440785 | 22.307328005801054 |

------------------------------------------------
Логарифмическая аппроксимация: a*ln(x)+b

Коэффициенты: a = 22.896218323946353 b = -7.884850824260766
Мера отклонения: 400.8775254955616
Среднеквадратичное отклонение: 7.567576754016927
Достоверность аппроксимации: 0.7863126196718755 -> Удовлетворительная точность аппроксимации

x_i   | 1.0               | 2.0               | 3.0               | 4.0               | 5.0               | 6.0               | 7.0               |
y_i   | 1.0               | 4.0               | 9.0               | 16.0              | 25.0              | 36.0              | 49.0              |
phi_i | -7.884850824260766 | 7.985598352467606 | 17.26921599045465 | 23.856047529195976 | 28.965190997666845 | 33.13966516718302 | 36.669132787292675 |
eps_i | -8.884850824260766 | 3.985598352467606 | 8.26921599045465 | 7.856047529195976 | 3.965190997666845 | -2.860334832816982 | -12.330867212707325 |

------------------------------------------------

Наилучшее приближение - Квадратичная аппроксимация, Кубическая аппроксимация, Степенная аппроксимация
```



Вывод:

При выполнении лабораторной работы я познакомилась с методом наименьших квадратов для нахождения приближенных функций.