НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.03.04 - Программная инженерия

Дисциплина – Вычислительная математика

Лабораторная работа №4

Вариант №11

Выполнил: Мухсинов С.П

Группа: P3217

Преподаватель: Малышева Т.А

Санкт-Петербург

2024

Цель работы:

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

Задание:

Вычислительная реализация задачи:

1. Сформировать таблицу табулирования заданной функции на указанном интервале.
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.
2. Реализовать метод наименьших квадратов, исследуя все указанные функции.
3. Предусмотреть вывод результатов в файл/консоль: коэффициенты аппроксимирующих функций, среднеквадратичное отклонение, массивы значений $x_i, y_i, \varphi(x_i), \varepsilon_i$.
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.

Используемые формулы и методы

Линейная аппроксимация

Рассмотрим в качестве эмпирической формулы линейную функцию:

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

Сумма квадратов отклонений запишется следующим образом:

$$S = S(a,b) = \sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n} [\varphi(x_i) - y_i]^2 = \sum_{i=1}^{n} (ax_i + b - y_i)^2 \to min$$

Для нахождения a и b необходимо найти минимум функции $S(a, b)$.

Необходимое условие существования минимума для функции S:

$$\begin{cases} \dfrac{\partial S}{\partial a} = 0 \\ \dfrac{\partial S}{\partial b} = 0 \end{cases} \quad \text{или} \quad \begin{cases} 2\sum_{i=1}^{n} (ax_i + b - y_i)x_i = 0 \\ 2\sum_{i=1}^{n} (ax_i + b - y_i) = 0 \end{cases}$$

Упростим полученную систему:

$$\begin{cases} a\sum_{i=1}^{n} x_i^2 + b\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} x_i y_i \\ a\sum_{i=1}^{n} x_i + bn = \sum_{i=1}^{n} y_i \end{cases}$$

Линейная аппроксимация

Введем обозначения:

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

Получим систему уравнений для нахождения параметров *a* и *b:*

$$\begin{cases} aSXX + bSX = SXY \\ aSX + \; bn = SY \end{cases},$$

из которой находим (правило Крамера):

$$\Delta = SXX \cdot n - SX \cdot SX$$
$$\Delta_1 = SXY \cdot n - SX \cdot SY$$
$$\Delta_2 = SXX \cdot SY - SX \cdot SXY$$
$$a = \frac{\Delta_1}{\Delta}, \quad b = \frac{\Delta_2}{\Delta}$$

КВАДРАТИЧНАЯ АППРОКСИМАЦИЯ

Рассмотрим в качестве эмпирической формулы квадратичную функцию:

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

Сумма квадратов отклонений запишется следующим образом:

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

Приравниваем к нулю частные производные *S* по неизвестным параметрам, получаем систему линейных уравнений:

$$\begin{cases} \dfrac{\partial S}{\partial a_0} = 2\sum_{i=1}^{n} a_2 x_i^2 + a_1 x_i + a_0 - y_i = 0 \\ \dfrac{\partial S}{\partial a_1} = 2\sum_{i=1}^{n} (a_2 x_i^2 + a_1 x_i + a_0 - y_i) x_i = 0 \\ \dfrac{\partial S}{\partial a_2} = 2\sum_{i=1}^{n} (a_2 x_i^2 + a_1 x_i + a_0 - y_i) x_i^2 = 0 \end{cases} \qquad \begin{cases} a_0 n + a_1 \sum_{i=1}^{n} x_i + a_2 \sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} y_i \\ a_0 \sum_{i=1}^{n} x_i + a_1 \sum_{i=1}^{n} x_i^2 + a_2 \sum_{i=1}^{n} x_i^3 = \sum_{i=1}^{n} x_i y_i \\ a_0 \sum_{i=1}^{n} x_i^2 + a_1 \sum_{i=1}^{n} x_i^3 + a_2 \sum_{i=1}^{n} x_i^4 = \sum_{i=1}^{n} x_i^2 y_i \end{cases}$$

Аппроксимация с помощью других функций

Помимо линейных зависимостей для описания результатов эксперимента используют также показательные, степенные, логарифмические функции. Эти функции легко могут быть приведены к линейному виду, после чего для определения коэффициентов аппроксимирующей функции можно использовать описанный выше алгоритм.

Аппроксимирующая функция задана степенной функцией вида:

$$\boldsymbol{\varphi(x) = ax^b}$$

Для применения метода наименьших квадратов степенная функция ***линеаризуется***:

$$\ln(\varphi(x)) = \ln(ax^b) = \ln(a) + b\ln(x)$$

Введем обозначения: Y=$\ln(\varphi(x))$; A=$\ln(a)$; B=b; X=$\ln(x)$

Получаем линейную зависимость: Y=A+BX.

После определения коэффициентов A и B вернемся к принятым ранее обозначениям:

$$\boldsymbol{a = e^{\mathrm{A}}} \quad \boldsymbol{b = B}$$

Аппроксимация с помощью других функций

Аппроксимирующая функция задана экспоненциальной функцией вида:

$$\boldsymbol{\varphi(x) = ae^{bx}}$$

Для применения метода наименьших квадратов экспоненциальная функция линеаризуется:

$$\ln(\varphi(x)) = \ln(ae^{bx}) = \ln a + bx$$

Введем обозначения: Y=$\ln(\varphi(x))$; A=$\ln(a)$; B=b

Получаем линейную зависимость: Y=A+Bx.

После определения коэффициентов A и B вернемся к принятым ранее обозначениям:

$$\boldsymbol{a = e^{\mathrm{A}}} \quad \boldsymbol{b = B}$$

Аппроксимирующая функция задана логарифмической функцией вида:

$$\boldsymbol{\varphi(x) = aln(x) + b}$$

Коэффициент корреляции

Коэффициент корреляции – это степень связи между двумя переменными.

Корреляция помогает найти ответ на два вопроса.

Во-первых, является ли связь между переменными положительной (прямо пропорциональная зависимость) или отрицательной (обратно пропорциональная зависимость).

Во-вторых, насколько сильна зависимость.

Коэффициент корреляции Пирсона позволяет определить наличие или отсутствие **линейной** связи между двумя переменными:

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

Средние значения *x* и *y:*

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} \quad \bar{y} = \frac{\sum_{i=1}^{n} y_i}{n}$$

$$-\mathbf{1} \le \boldsymbol{r} \le \mathbf{1}$$

Например, при помощи критерия корреляции Пирсона можно ответить на вопрос о наличии связи между температурой тела и содержанием лейкоцитов в крови при острых респираторных инфекциях, между ростом и весом пациента, между содержанием в питьевой воде фтора и заболеваемостью населения кариесом.

Среднеквадратичное отклонение

В случае, когда экспериментальные данные могут быть описаны несколькими уравнениями, выбор наилучшего из них можно осуществить по величине среднеквадратичного отклонения:

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

Наилучшим считается уравнение, для которого значение $\boldsymbol{\delta}$ минимально.

Вычислительная реализация

$$\frac{5x}{x^4 + 11} \quad x \in [-2, 0] \quad h = 0{,}2.$$

Сформировать таблицу табулирования заданной функции на указанном интервале:

X	Y
-2	-0,3737
-1,8	-0,4187
-1,6	-0,4557
-1,4	-0,4716
-1,2	-0,4589
-1	-0,4167
-0,8	-0,3506
-0,6	-0,2696
-0,4	-0,1814
-0,2	-0,0909
0	0

Линейное приближение:

$$\begin{cases} a\sum_{i=1}^{n} x_i^2 + b\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} x_i y_i \\ a\sum_{i=1}^{n} x_i + bn = \sum_{i=1}^{n} y_i \end{cases}$$

$$\sum_{i=1}^{n} x_i = SX = -11$$

$$\sum_{i=1}^{n} x_i^2 = SXX = 15{,}4$$

$$\sum_{i=1}^{n} y_i = SY = -3,4878$$

$$\sum_{i=1}^{n} x_i y_i = SXY = 4,3908$$

Правило Крамера:

$$\Delta = SXX \cdot n - SX \cdot SX = 48,4$$

$$\Delta_1 = SXY \cdot n - SX \cdot SY = 9,9328$$

$$\Delta_2 = SXX \cdot SY - SX \cdot SXY = -5,4135$$

$$a = \frac{\Delta_1}{\Delta} = 0,2052$$

$$b = \frac{\Delta_2}{\Delta} = -0,1119$$

Полученное приближение: $0,2052x - 0,1119$

Среднеквадратичное отклонение:

$$\delta_1 = \sqrt{\frac{\sum_{i=1}^{n} (\varphi(x_i) - y_i)^2}{n}} = 0.0825$$

Квадратичное приближение:

$$\sum_{i=1}^{n} x_i = -11$$

$$\sum_{i=1}^{n} x_i^2 = 15,4$$

$$\sum_{i=1}^{n} x_i^3 = -24,2$$

$$\sum_{i=1}^{n} x_i^4 = 40,5328$$

$$\sum_{i=1}^{n} y_i = -3,4878$$

$$\sum_{i=1}^{n} x_i y_i = 4,3908$$

$$\sum_{i=1}^{n} x_i^2 y_i = -6,3739$$

Решив следующую систему:

$$\begin{cases} a_0 n + a_1 \sum_{i=1}^{n} x_i + a_2 \sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} y_i \\ a_0 \sum_{i=1}^{n} x_i + a_1 \sum_{i=1}^{n} x_i^2 + a_2 \sum_{i=1}^{n} x_i^3 = \sum_{i=1}^{n} x_i y_i \\ a_0 \sum_{i=1}^{n} x_i^2 + a_1 \sum_{i=1}^{n} x_i^3 + a_2 \sum_{i=1}^{n} x_i^4 = \sum_{i=1}^{n} x_i^2 y_i \end{cases}$$

Получим:

$$a_0 = 0,0258$$

$$a_1 = 0,6642$$

$$a_2 = 0,2295$$

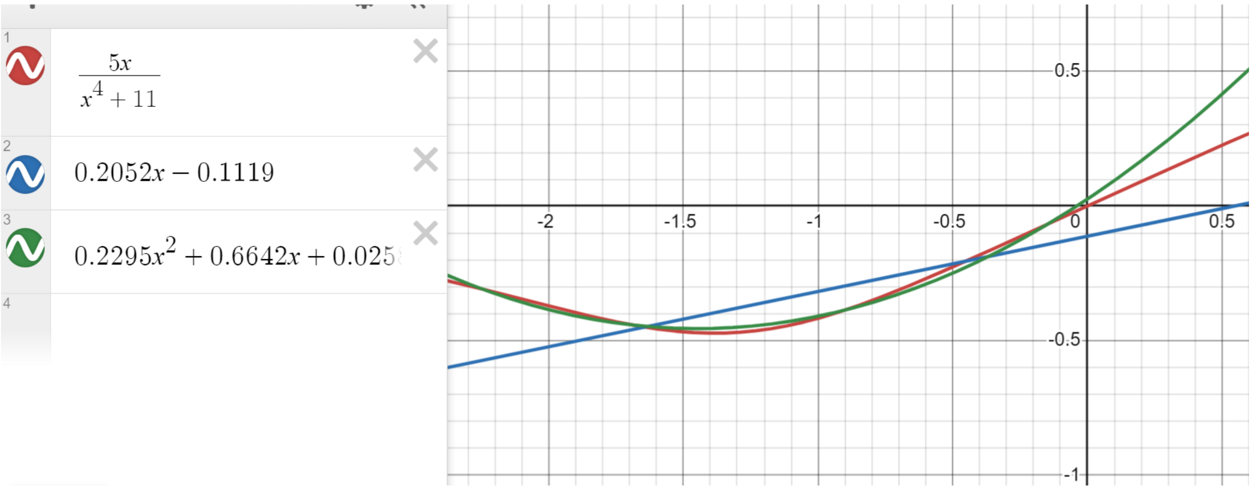
Полученное приближение: $0,2295x^2 + 0,6642x + 0,0258$

Среднеквадратичное отклонение:

$$\delta_2 = \sqrt{\frac{\sum_{i=1}^{n} (\varphi(x_i) - y_i)^2}{n}} = 0.0153$$

Так как $\delta_2 = 0.0153 < 0.0825 = \delta_1$ **, то наилучшим является квадратичное приближение.**

Графики:



Программная реализация

Экспоненциальная функция

```
  public Exponential(double[] x, double[] y) {
        super(x, y);
    }

    @Override
    public Result solve() {
        boolean isNan = false;
        for(int i = 0; i < n; i++){
            if(this.y[i] > 0){
                this.y[i] = Math.log(this.y[i]);
            }else{
                isNan = true;
            }
        }
        double[] r = new Linear(x, y).solve().getR();
        r[1] = Math.exp(r[1]);
        MathFunction f  = new MathFunction(x -> r[1] * Math.exp(x*r[0]));
        String function = r[1] + " * " + "e^(" + r[0] + ")x";
        return new Result(MethodType.EXPONENTIAL, function, f, deviation(f),
determination(f), isNan);
    }
}
```

Линейная функция

```
   public Linear(double[] x, double[] y){
        super(x, y);
        this.a = new double[2];
    }
    @Override
    public Result solve() {
        double a, b;
        double sx = 0, sxx = 0, sy = 0, sxy = 0;
        for (int i = 0; i < n; i++) {
```

```
            sx += x[i];
            sxx += x[i] * x[i];
            sy += y[i];
            sxy += x[i] * y[i];

        }
        double delta = sxx * n - sx * sx;
        double delta1 = sxy * n - sx * sy;
        double delta2 = sxx * sy - sx * sxy;
        a = delta1 / delta;
        b = delta2 / delta;
        MathFunction f = new MathFunction(x -> a * x + b);
        String function = a + "x" + " + " + b;
        return new Result(MethodType.LINEAR, function, f, deviation(f),
correlation(), new double[]{a, b}, determination(f));
    }

    private double correlation() {
        double x_avg = Arrays.stream(x).sum() / n;
        double y_avg = Arrays.stream(y).sum() / n;
        double numerator = 0;
        double denominator1 = 0;
        double denominator2 = 0;
        for (int i = 0; i < n; i++) {
            numerator += (x[i] - x_avg) * (y[i] - y_avg);
            denominator1 += Math.pow(x[i] - x_avg, 2);
            denominator2 += Math.pow(y[i] - y_avg, 2);
        }
        double denominator = Math.sqrt(denominator1*denominator2);
        return numerator/denominator;
    }
}
```

Логарифмическая функция

```
  public Logarithmic(double[] x, double[] y) {
        super(x, y);
    }

    @Override
    public Result solve() {
        boolean isNan = false;
        for(int i = 0; i < n; i++){
            if(this.x[i] < 0){
                isNan = true;
            }
            this.x[i] = Math.log(this.x[i]);
        }
        double[] r = new Linear(x,y).solve().getR();
        MathFunction f = new MathFunction(x -> r[0] * Math.log(x) + r[1]);
        String function = r[0] + "ln(x)" + " + " + r[1];
        double deviation = deviation(f);
        return new Result(MethodType.LOGARITHMIC, function, f, deviation(f),
determination(f), isNan);
    }
}
```

Кубическая функция

```
  public Cubic(double[] x, double[] y) {
        super(x, y);
    }

    @Override
    public Result solve() {
        double sx = 0, sx2 = 0, sx3 = 0, sx4 = 0, sx5 = 0, sx6 = 0;
        double sy = 0, sxy = 0, sx2y = 0, sx3y = 0;
        for(int i = 0; i < n; i++){
            sx += x[i];
            sx2 += Math.pow(x[i], 2);
            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 += Math.pow(x[i], 2) * y[i];
            sx3y += Math.pow(x[i], 3) * y[i];
        }
        double[][] x_i = {
                {n, sx, sx2, sx3},
                {sx, sx2, sx3, sx4},
                {sx2, sx3, sx4, sx5},
                {sx3, sx4, sx5, sx6}
        };
        double[] y_i = {sy, sxy, sx2y, sx3y};
        double[] result = solveLinearSystem(x_i, y_i);
        double a0 = result[0], a1 = result[1], a2 = result[2], a3 =
result[3];
        MathFunction f= new MathFunction(x -> a3*x*x*x + a2*x*x + a1*x + a0);
        String function = a3 + "x^3" + " + " + a2 +"x^2" + " + " + a3 + "x" +
" + " + a0;
        return new Result(MethodType.CUBIC, function,f, deviation(f),
determination(f), false);
    }
}
```

Степенная функция

```
   public Power(double[] x, double[] y) {
        super(x, y);
    }

    @Override
    public Result solve() {
        boolean isNan = false;
        for(int i = 0; i < n;i++){
            if(x[i] < 0 || y[i] < 0){
                isNan = true;
            }
            this.x[i] = Math.log(this.x[i]);
            this.y[i] = Math.log(this.y[i]);
        }
        double[] r  = new Linear(x , y).solve().getR();
        r[1] = Math.exp(r[1]);
        MathFunction f = new MathFunction(x -> r[1]*Math.pow(x, r[0]));
        String function = r[1] + " * " + "x^(" + r[0] + ")";
        return new Result(MethodType.POWER, function, f, deviation(f),
determination(f), isNan);
```

```
    }
}
```

Квадратичная функция

```
  public Quadratic(double[] x, double[] y) {
        super(x, y);
    }

    @Override
    public Result solve() {
        double sx = 0, sx2 = 0, sx3 = 0, sx4 = 0;
        double sy = 0, sxy = 0, sx2y = 0;
        for(int i = 0; i < n; i++){
            sx += x[i];
            sx2 += Math.pow(x[i], 2);
            sx3 += Math.pow(x[i], 3);
            sx4 += Math.pow(x[i], 4);
            sy += y[i];
            sxy += x[i]*y[i];
            sx2y += Math.pow(x[i], 2) * y[i];
        }
        double[][] x_i = {{n, sx, sx2}, {sx, sx2, sx3}, {sx2, sx3, sx4}};
        double[] y_i = {sy, sxy, sx2y};
        double[] result = solveLinearSystem(x_i, y_i);
        double a0 = result[0], a1 = result[1], a2 = result[2];
        MathFunction f= new MathFunction(x -> a2*x*x + a1*x + a0);
        String function = a2 + "x^2" + " + " + a1 +"x" + " + " + a0;
        return new Result(MethodType.QUADRATIC, function, f, deviation(f),
determination(f), false);
    }
}
```

Результаты работы программы

Входные данные:

```
11
1 1
1.2 1.0955
1.5 1.2247
2 1.41421
3.5 1.87083
4 2
4.5 2.1213
5 2.236
5.5 2.3452
6 2.4495
6.5 2.5495
```

Результат:

```
Вы хотите прочитать данные с файла или консоли?
1) С консоли
2) С файла
2
Введите путь к файлу:
C:\Users\Microsoft\Desktop\Current\Fourth semester\Computational mathematics\Lab4\src\main\resources\input1.txt
Введите количество входных данных (от 8 до 12):
Введите таблицу значений функции в виде пар (x, y) через пробел:
Аппроксимация: Линейная
Функция: 0.28145229426433965x + 0.8046937839492159
Отклонение: 0.0354329559571095
Достоверность: 0.9889684669645966
Корреляция: 0.9944689371541963


Аппроксимация: Квадратная
Функция: -0.020604830082476856x^2 + 0.42971971871069115x + 0.6132984088093067
Отклонение: 0.0017435385757416813
Достоверность: 0.999457174743759


Аппроксимация: Кубическая
Функция: 0.003429091307505627x^3 + -0.05991170778487325x^2 + 0.003429091307505627x + 0.5024910349195607
Отклонение: 1.6541858059993049E-4
Достоверность: 0.9999484993422856


Аппроксимация: Степенная
Функция: 1.000006825978482 * x^(0.4999914179578249)
Отклонение: 9.127362620790008E-9
Достоверность: 0.999999997158329


Аппроксимация: Показательная
Функция: 0.9484936966614004 * e^(0.1666476228819394)x
Отклонение: 0.17611710818268306
Достоверность: 0.9451685217758957


Аппроксимация: Логарифмическая
Функция: 0.8254914360915603ln(x) + 0.9153769923832971
Отклонение: 0.03661630265351023
Достоверность: 0.9886000492635874


Лучшие аппроксимации:
Степенная
Среднеквадратичное отклонение: 9.127362620790008E-9S
```

Входные данные:

```
11
-2 -0.3737
-1.8 -0.4187
-1.6 -0.4557
-1.4 -0.4716
-1.2 -0.4589
-1 -0.4167
-0.8 -0.3506
-0.6 -0.2696
-0.4 -0.1814
-0.2 -0.0909
0 0
```

Результат:

```
Вы хотите прочитать данные с файла или консоли?
1) С консоли
2) С файла
2
Введите путь к файлу:
input2.txt
Введите количество входных данных (от 8 до 12):
Введите таблицу значений функции в виде пар (x, y) через пробел:
Аппроксимация: Линейная
Функция: 0.2052227272727274x + -0.11185
Отклонение: 0.07483354354545456
Достоверность: 0.7123397261808508
Корреляция: 0.8440022074502239


Аппроксимация: Квадратная
Функция: 0.22942016317016353x^2 + 0.6640630536130546x + 0.025802097902098545
Отклонение: 0.0025781219953379957
Достоверность: 0.9900896944875044


Аппроксимация: Кубическая
Функция: -0.07169289044289291x^3 + 0.014341491841485591x^2 + -0.07169289044289291x + 0.005154545454545862
Отклонение: 5.4598988344988E-4
Достоверность: 0.9979012139217986


Аппроксимация: Степенная
Введенные данные меньше нуля! Поэтому логарифм взять то не удалось!



Аппроксимация: Показательная
Введенные данные меньше нуля! Поэтому логарифм взять то не удалось!



Аппроксимация: Логарифмическая
Введенные данные меньше нуля! Поэтому логарифм взять то не удалось!



Лучшие аппроксимации:
Кубическая
Среднеквадратичное отклонение: 5.4598988344988E-4
```

Входные данные:

```
11
-2 -0.3737
-1.8 gde
-1.6 -0.4557
-1.4 -0.4716
-1.2 -0.4589
-1 -0.4167
-0.8 -0.3506
-0.6 -0.2696
-0.4 -0.1814
-0.2 -0.0909
0 0
```

Результат:

```
Вы хотите прочитать данные с файла или консоли?
1) С консоли
2) С файла
2
Введите путь к файлу:
C:\Users\Microsoft\Desktop\Current\Fourth semester\Computational mathematics\Lab4\src\main\resources\input4.txt
Введенные данные должны быть числом!
```

Вывод

В ходе выполнения лабораторной работы мы познакомились с различными способами аппроксимации функции и реализовали программу, находящую приближение функции через метод наименьших квадратов.