Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ»

КАФЕДРА «ТЕОРЕТИЧЕСКАЯ ИНФОРМАТИКА И КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ»

Отчёт по лабораторной работе № 3
«Метод наименьших квадратов»

по курсу

«Численные методы»

Студент группы ИУ9-61Б Бакланова А.Д.

Преподаватель Домрачева А.Б.

Москва, 2020 г.

Цель работы

Целью данной работы является реализация метода наименьших квадратов.

Задание

Задача заключается в построении функции, наилучшим образом аппроксимирующей функции по коэффициентам зависимости. Нужно найти коэффициенты a и b, $z_1(x) = ax + b$, для функции $z_3(x) = ae^{bx}$

Дано:

Значения функции в точках $y_i(x_i)$ $i = \overline{1,..,N}$

```
array_x[0] = 1;    array_x[5] = 1.85;
array_x[1] = 1.5;  array_x[6] = 1.86;
array_x[2]= 1.7;   array_x[7] = 1.88;
array_x[3] = 1.75; array_x[8] = 1.89;
array_x[4] = 1.8;  array_x[9] = 1.9;
array_x[10] = 2.0;

array_y[0] = 10;    array_y[5] = 23.5;
array_y[1] = 16.5;  array_y[6] = 24;
array_y[2]= 20.0;   array_y[7] = 24.5;
array_y[3] = 21.5;  array_y[8] = 25;
array_y[4] = 22;    array_y[9] = 25.5;
array_y[10] = 27.0;
```

Метод наименьших квадратов:

$$F(x) = \sum_{i=0}^{n} (g(x_i) - y_i)^2 \to min$$

Пусть $g(x)$ - некоторое приближение исходных данных

$g(x) = ax + b$

$$F(x) = F(a, b, x) = \sum_{i=0}^{n} (ax_i + b - y_i) \to min$$

$$\begin{cases} \delta \frac{F'(a, b, x)}{\delta a} = 0 \\ \delta \frac{F'(a, b, x)}{\delta b} = 0 \end{cases}$$

$$\frac{\delta F'(a, b, x)}{\delta a} = 2 \sum_{i=0}^{n} (ax_i + b - y_i)x_i = 0$$

$$a \sum_{i=0}^{n} x_i^2 + b \sum_{i=0}^{n} x_i - \sum_{i=0}^{n} x_i y_i = 0$$

$$A = \sum_{i=0}^{n} x_i^2, \quad B = \sum_{i=0}^{n} x_i, \quad C = \sum_{i=0}^{n} x_i y_i$$

$$a \sum_{i=0}^{n} (x_i) + b(n + 1) - \sum y_i = 0$$

$$B = \sum_{i=0}^{n} (x_i), \quad D = \sum y_i$$

$$\begin{cases} aA + bB = C \\ aB + (n + 1)b = D \end{cases}$$

Таким образом получаются оценки параметров линейной зависимости $\widehat{a}, \widehat{b}$. Следует отметить, что тренд не обязательно линеен, если есть предположение о виде функции тренда (этап плохо формализуем) данные предварительно линеаризуют.

$$y = ae^{bx}$$

$$ln(y) = ln(ae^{bx})$$

$$ln(y) = ln(a) + b(x)$$

$$y^* = a^* x^* + b^* \rightarrow \widehat{a^*}, \widehat{b^*}$$

$$a = e^{\widehat{b}*}\ b = \widehat{a}*$$

Погрешность:

$$\delta_3 = |z_3(x_a) - y_g|$$

$$x_a = \frac{x_0 + x_n}{2}$$

$$y_g = (y_0 * y_n)^{1/2}$$

Реализация:

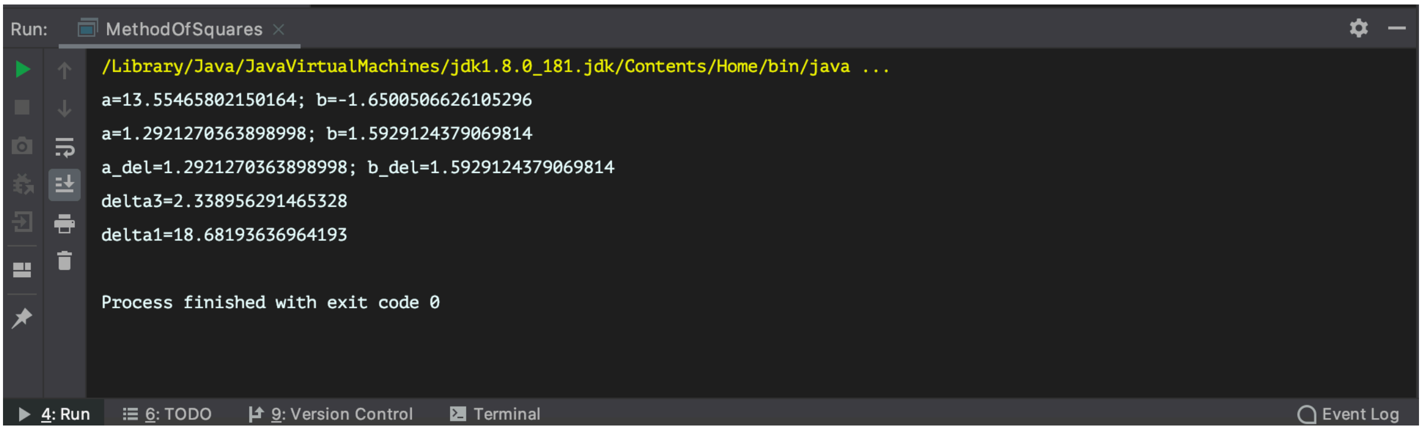
```
Pair koef_z1;
koef_z1 = minimum_squares(array_x, array_y);
System.out.println("a=" + koef_z1.a + "; b=" + koef_z1.b);

double[] array_lny = new double[n];

for (int i=0; i<n; i++) {
    array_lny[i] = log(array_y[i]);
}
```

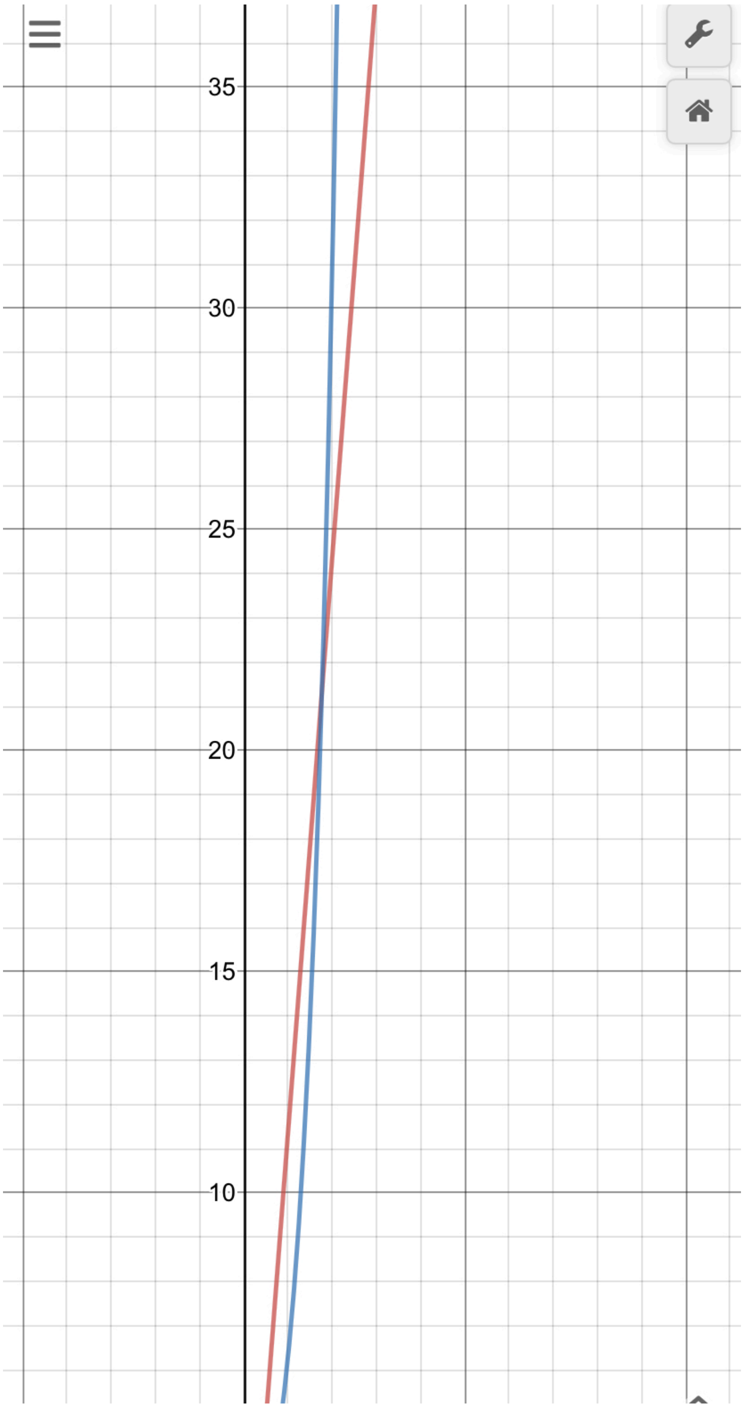
```java
        Pair koef_z3;
        koef_z3 = minimum_squares(array_x, array_lny);
        double aa, bb;
        bb = koef_z3.a;
        aa = exp(koef_z3.b);
        System.out.println("a=" + aa+ "; b=" + bb);


        double a_del;
        double b_del;

        a_del = Math.exp(koef_z3.b);
        b_del = koef_z3.a;

        System.out.println("a_del=" + a_del + "; b_del=" + b_del);

        double y_g;
        y_g = sqrt(array_y[0]*array_y[n-1]);

        double x_a;
        x_a = (array_x[0]+array_x[n-1])/2;

        double array_z3;
        array_z3 = a_del*Math.exp(b_del * x_a);


        double delta3;

        delta3 = abs(array_z3 - y_g);
        System.out.println("delta3=" + delta3);

        double delta1;
        delta1 = abs(koef_z1.a*x_a + koef_z1.b);
        System.out.println("delta1=" + delta1);
```

```java
        double D = 0;
        double C = 0;
        double B = 0;
        double A = 0;

        int n = array_x.length;

        for (int i=0; i<n; i++) {
            B += array_x[i]; //B
            C += array_x[i]*array_y[i]; //C
            D += (array_y[i]); //D
            A += array_x[i]*array_x[i]; //A
        }
```

```java
    double a = (C-D*B/(n+1))/(A-B*B/(n+1));
    double b = (D - a*B)/(n+1);
```

Тестирование:





Вывод:

Итого, на серии тестов было выявлено, что к каждому тесту лучше подходят разные функции. В приведенном выше примере экспонента лучше приближена к данным точкам, чем линейная функция с погрешностью ERR>2.0