**ФПИиКТ**

**Рабочий протокол и отчет по**

**домашней работе №4**

**Вариант №6**

Ибадуллаев Алибаба Эльбрус оглы

Группа: P3215

Преподаватель: Малышева Татьяна Алексеевна

Санкт-Петербург

2022г.

**1. Цель лабораторной работы**: Найти функцию, являющуюся наилучшим приближением заданной табличной функции по методу наименьших квадратов.

**Для исследования использовать:**

1. линейную функцию;
2. полиномиальную функцию 2-й степени;
3. полиномиальную функцию 3-й степени;
4. экспоненциальную функцию;
5. логарифмическую функцию;
6. степенную функцию.
7. **Методика проведения исследования:**
8. Вычислить меру отклонения: для всех исследуемых функций.
9. Уточнить значения коэффициентов эмпирических функций, минимизируя функцию S.
10. Сформировать массивы предполагаемых эмпирических зависимостей (.
11. Определить среднеквадратичное отклонение для каждой аппроксимирующей функции. Выбрать наименьшее значение и, следовательно, наилучшее приближение.
12. Построить графики полученных эмпирических функций.

|  |  |  |
| --- | --- | --- |
| 6 |  |  |

[*https://www.desmos.com/calculator/gtwet8sd99*](https://www.desmos.com/calculator/gtwet8sd99)

1. **Вычислительная реализация задачи:**

x0 = 0.0 y0 = 0.00

x1 = 0.2 y1 = 0.40

x2 = 0.4 y2 = 0.78

x3 = 0.6 y3 = 1.06

x4 = 0.8 y4 = 1.14

x5 = 1.0 y5 = 1.00

x6 = 1.2 y6 = 0.78

x7 = 1.4 y7 = 0.58

x8 = 1.6 y8 = 0.42

x9 = 1.8 y9 = 0.31

x10 = 2.0 y10 = 0.24

SX = 11.0

SXX = 15.4

SXXX = 24.2

SXXXX = 40.532

SY = 6.707

SXY = 6.395

SXXY = 7.546

FIRST APPROXIMATION(linear)

<<RESULTS>>

<<Triangle Matrix>>

A = [11.0, 11.0]

[0.0, 4.4]

B = [6.71, -0.312]

X = [0.6809, -0.0709]

S = 1.351

Theta = 0.3504

SECOND APPROXIMATION(2nd polynomial)

<<INPUT MATRIX>>

A = [11.0, 11.0, 15.4]

[11.0, 15.4, 24.2]

[15.4, 24.2, 40.5328]

B = [6.71, 6.398, 7.5516]

<<Iteration №1>>

A = [11.0, 11.0, 15.4]

[0.0, 4.4, 8.79]

[0.0, 8.799, 18.9728]

B = [6.71, -0.312, -1.8423]

<<Iteration №2>>

A = [11.0, 11.0, 15.4]

[0.0, 4.4, 8.799]

[0.0, 0.0, 1.3728]

B = [6.71, -0.312, -1.2183]

<<RESULTS>>

<<Triangle Matrix>>

A = [11.0, 11.0, 15.4]

[0.0, 4.4, 8.799]

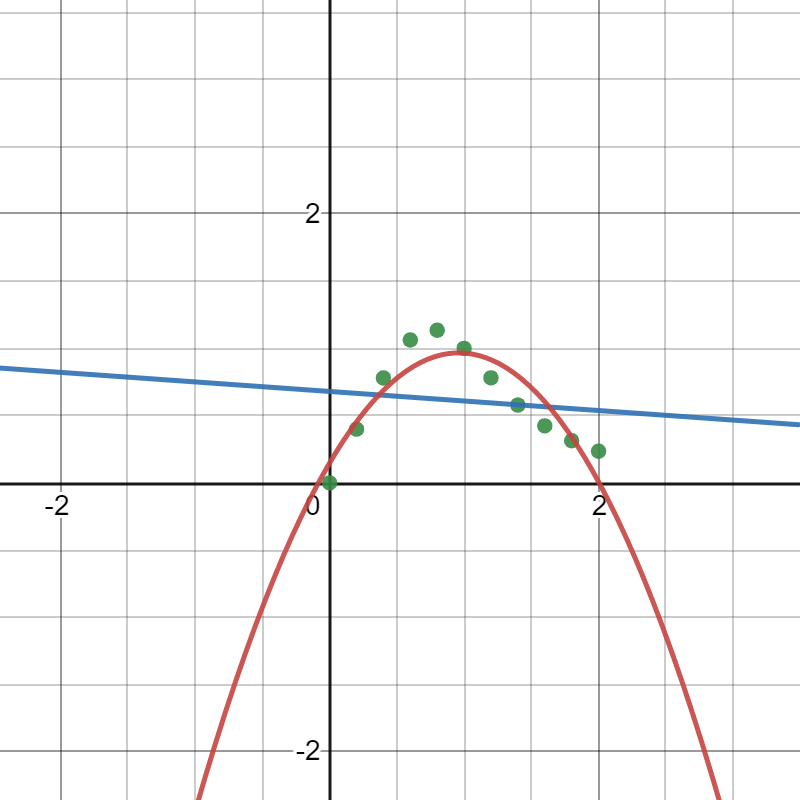
[0.0, 0.0, 1.372]

B = [6.71, -0.312, -1.2183]

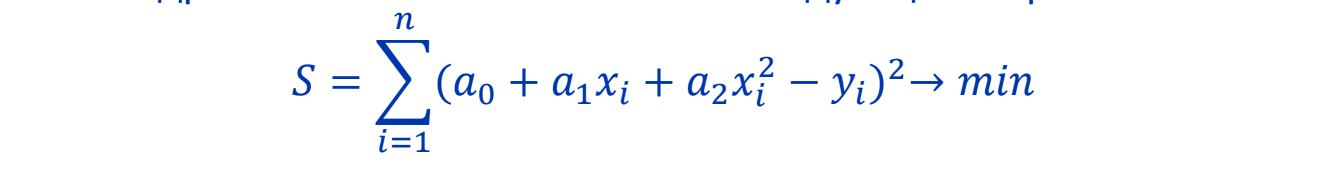
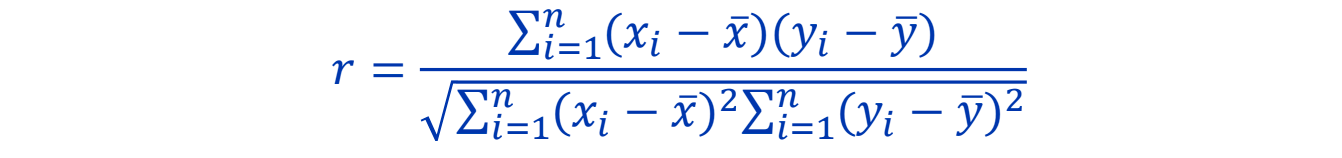
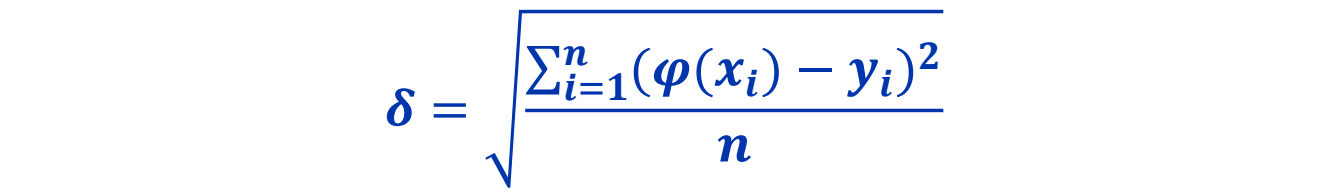
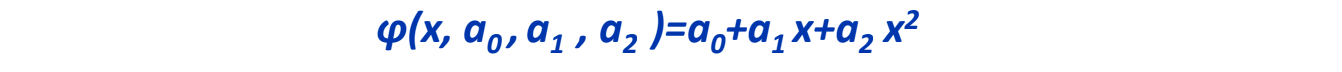
X = [0.1483, 1.704, -0.887]

S = 0.269

Theta = 0.156

****

1. **Программная реализация задачи:**
   1. Предусмотреть ввод исходных данных из файла/консоли (таблица *y=f(x)* должна содержать 10 - 12 точек).
   2. Реализовать метод наименьших квадратов, исследуя все функции п.1.
   3. Предусмотреть вывод результатов в файл/консоль.
   4. Для линейной зависимости вычислить коэффициент корреляции Пирсона.
   5. Программа должна отображать наилучшую аппроксимирующую функцию.
   6. Организовать вывод графиков функций, графики должны полностью отображать весь исследуемый интервал (с запасом).
2. **Основные рабочие формулы**



1. **Листинг программы**

**Основной код:**

double[] arr1 = {n, SX, SX,SXX,SY, SXY};  
double[] arr2 = {n, SX, SXX,SX, SXX, SXXX, SXX, SXXX, SXXXX, SY, SXY, SXXY};  
double[] arr3 = {n, SX, SXX, SXXX, SX, SXX, SXXX, SXXXX, SXX, SXXX, SXXXX, SXXXXX, SXXX, SXXXX, SXXXXX, SXXXXXX, SY, SXY, SXXY, SXXXY};  
double[] arr4 = {n, SLN\_X, SLN\_X,SLN\_XX, SLN\_Y, SLN\_XY};  
double[] arr5 = {n, SX, SX,SXX, SLN\_Y, SXLN\_Y};  
double[] arr6 = {n, SLN\_X, SLN\_X,SLN\_XX, SY, SYLN\_X};  
  
System.*out*.println("FIRST APPROXIMATION(linear)");  
double[] x1 = *gaussianMethod*(arr1, 2);  
DoubleFunction<Double> f\_X1 = (x) -> x1[0] + x1[1]\*x;  
thetas.put(1,*io*.showResults(points, f\_X1));  
  
double SX\_X\_AVG\_Y\_Y\_AVG = 0;  
double SX\_X\_AVG2 = 0;  
double SY\_Y\_AVG2 = 0;  
for (double key: points.keySet()){  
 SX\_X\_AVG\_Y\_Y\_AVG += ((key-AVG\_X)\*(points.get(key)-AVG\_Y));  
 SX\_X\_AVG2 += Math.*pow*(key-AVG\_X,2);  
 SY\_Y\_AVG2 += Math.*pow*(points.get(key)-AVG\_Y,2);  
}  
R = SX\_X\_AVG\_Y\_Y\_AVG/Math.*sqrt*(SX\_X\_AVG2\*SY\_Y\_AVG2);  
System.*out*.format("r = %.3f\n", R); // Pirson coefficient  
*io*.showResultsLinear(points, f\_X1);  
  
  
System.*out*.println("SECOND APPROXIMATION(2nd polynomial)");  
double[] x2 = *gaussianMethod*(arr2, 3);  
DoubleFunction<Double> f\_X2 = (x) -> x2[0] + x2[1]\*x + x2[2]\*x\*x;  
thetas.put(2,*io*.showResults(points, f\_X2));  
  
System.*out*.println("THIRD APPROXIMATION(3rd polynomial)");  
double[] x3 = *gaussianMethod*(arr3, 4);  
DoubleFunction<Double> f\_X3 = (x) -> x3[0] + x3[1]\*x + x3[2]\*x\*x + x3[3]\*x\*x\*x;  
thetas.put(3,*io*.showResults(points, f\_X3));  
  
System.*out*.println("FOURTH APPROXIMATION(power function)");  
double[] x4 = *gaussianMethod*(arr4, 2);  
DoubleFunction<Double> f\_X4 = (x) -> Math.*exp*(x4[0])\*Math.*pow*(x,x4[1]);  
thetas.put(4,*io*.showResults(points, f\_X4));  
  
System.*out*.println("FIFTH APPROXIMATION(exponential function)");  
double[] x5 = *gaussianMethod*(arr5, 2);  
DoubleFunction<Double> f\_X5 = (x) -> Math.*exp*(x5[0])\*Math.*exp*(x\*x5[1]);  
thetas.put(5,*io*.showResults(points, f\_X5));  
  
System.*out*.println("SIXTH APPROXIMATION(logarithm function)");  
double[] x6 = *gaussianMethod*(arr6, 2);  
DoubleFunction<Double> f\_X6 = (x) -> x6[0] + x6[1]\*Math.*log*(x);  
thetas.put(6,*io*.showResults(points, f\_X6));  
  
for(int tht : thetas.keySet()){  
 if (thetas.get(tht)<theta) {  
 theta = thetas.get(tht);  
 min = tht;  
 }  
}  
System.*out*.print("THE BEST APPROXIMATION FUNCTION IS: ");  
switch (min){  
 case 1:  
 System.*out*.println("Linear");  
 break;  
 case 2:  
 System.*out*.println("Polynom 2nd");  
 break;  
 case 3:  
 System.*out*.println("Polynom 3rd");  
 break;  
 case 4:  
 System.*out*.println("Power function");  
 break;  
 case 5:  
 System.*out*.println("Exponential function");  
 break;  
 case 6:  
 System.*out*.println("Logarithm function");  
 break;  
}

**Метод Гаусса (для вычисления коэффициентов формул апроксимаций):**

public static double[] gaussianMethod(double [] arr\_res, int n){  
 double[] x = new double[n];  
 double [][] arr = new double[n][n];  
 double[] b = new double[n];  
 int stop = 0;  
 double c;  
 double s;  
 int swaps = 0;  
  
  
 // Fulling matrix by result array which has got from data input  
 for (int i = 0; i < n; i++){  
 for (int k = 0; k < n; k++){  
 arr[i][k] = arr\_res[k+(i\*n)];  
 }  
 b[i] = arr\_res[(n\*n)+i];  
 }  
 double [][] arr\_copy = new double[n][n];  
 double [] b\_copy = new double[n];  
  
 for (int i = 0; i < n; i++) {  
 System.*arraycopy*(arr[i], 0, arr\_copy[i], 0, arr[i].length);  
 }  
  
 System.*arraycopy*(b, 0, b\_copy, 0, arr.length);  
  
 System.*out*.println("\t<<INPUT MATRIX>>");  
 *io*.showExtendedMatrix(arr, b);  
  
 // straight stroke (process of exception an elements of matrix to get triangle-matrix)  
 for(int i = 0; i < n - 1 ; i++){  
 while (arr[i][i] == 0 ){  
 if (stop == n - i) {  
 System.*err*.println("The error in swapping lines!");  
 System.*exit*(0);  
 }  
 *swapLines*(arr, b, i);  
 stop++;  
 swaps++;  
 }  
  
 for (int k = i+1; k < n; k++){  
 c = arr[k][i] / arr[i][i];  
 arr[k][i] = 0;  
 for (int j = i+1; j < n; j++){  
 arr[k][j] = arr[k][j] - c \* arr[i][j];  
 }  
 b[k] = b[k] - c \* b[i];  
 }  
 stop = 0;  
 System.*out*.println();  
 System.*out*.println("\t<<Iteration №" + (i+1)+">>");  
 *io*.showExtendedMatrix(arr, b);  
 }  
  
  
 // reverse stroke (process of getting an unknown variables)  
 for (int i = n - 1 ; i >= 0 ; i--){  
 s = 0;  
 for (int j = i+1; j<n; j++){  
 if (Double.*isNaN*(x[j])) s=s+0;  
 else s = s + arr[i][j]\*x[j];  
 }  
 if (Double.*isInfinite*(s)) {  
 System.*err*.println("This system has no solution!\nBecause: "+"0!="+b[i+1]);  
 System.*exit*(-1);  
 }  
 x[i] = (b[i] - s)/arr[i][i] == -0 ? 0:(b[i] - s)/arr[i][i] ;  
 }  
 System.*out*.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");  
 System.*out*.print("X =\t");  
 System.*out*.println(Arrays.*toString*(x));  
  
 return x;  
}

1. **Анализ результатов работы: апробация и тестирование.**

**Входные данные:**

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

**Выходные данные:**

You has selected file: input.txt

SX = 31,300

SXX = 172,090

SXXX = 1049,047

SXXXX = 6779,433

SXXXXX = 45466,149

SXXXXXX = 312660,209

SY = 61,270

SXY = 328,123

SXXY = 1970,578

SXXXY = 12612,240

Sln(X) = 9,359

Sln(XX) = 14,374

Sln(Y) = 14,374

Sln(XY) = 9,359

FIRST APPROXIMATION(linear)

<<INPUT MATRIX>>

A = [7.0, 31.300000000000004]

[31.300000000000004, 172.09]

B = [61.269999999999996, 328.123]

<<Iteration №1>>

A = [7.0, 31.300000000000004]

[0.0, 32.13428571428568]

B = [61.269999999999996, 54.15857142857135]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

X = [1.216788476927183, 1.6853827687383294]

S = 0,473

Theta = 0,260

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

r = 0,997

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

x0 = 1.1 y0 = 2,73 phi0 = 3,07

x1 = 2.3 y1 = 5,12 phi1 = 5,09

x2 = 3.7 y2 = 7,74 phi2 = 7,45

x3 = 4.5 y3 = 8,91 phi3 = 8,80

x4 = 5.4 y4 = 10,59 phi4 = 10,32

x5 = 6.8 y5 = 12,75 phi5 = 12,68

x6 = 7.5 y6 = 13,43 phi6 = 13,86

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

SECOND APPROXIMATION(2nd polynomial)

<<INPUT MATRIX>>

A = [7.0, 31.300000000000004, 172.09]

[31.300000000000004, 172.09, 1049.047]

[172.09, 1049.047, 6779.4325]

B = [61.269999999999996, 328.123, 1970.5781]

<<Iteration №1>>

A = [7.0, 31.300000000000004, 172.09]

[0.0, 32.13428571428568, 279.55885714285694]

[0.0, 279.55885714285705, 2548.722771428571]

B = [61.269999999999996, 54.15857142857135, 464.2989142857143]

<<Iteration №2>>

A = [7.0, 31.300000000000004, 172.09]

[0.0, 32.13428571428568, 279.55885714285694]

[0.0, 0.0, 116.6427400906905]

B = [61.269999999999996, 54.15857142857135, -6.864766391037222]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

X = [0.37425996066520434, 2.197385944878237, -0.058852924628655164]

S = 0,069

Theta = 0,099

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

THIRD APPROXIMATION(3rd polynomial)

<<INPUT MATRIX>>

A = [7.0, 31.300000000000004, 172.09, 1049.047]

[31.300000000000004, 172.09, 1049.047, 6779.4325]

[172.09, 1049.047, 6779.4325, 45466.14943]

[1049.047, 6779.4325, 45466.14943, 312660.209029]

B = [61.269999999999996, 328.123, 1970.5781, 12612.23965]

<<Iteration №1>>

A = [7.0, 31.300000000000004, 172.09, 1049.047]

[0.0, 32.13428571428568, 279.55885714285694, 2088.6937714285705]

[0.0, 279.55885714285705, 2548.722771428571, 19676.07825428571]

[0.0, 2088.6937714285705, 19676.078254285712, 155445.97928485717]

B = [61.269999999999996, 54.15857142857135, 464.2989142857143, 3430.081122857144]

<<Iteration №2>>

A = [7.0, 31.300000000000004, 172.09, 1049.047]

[0.0, 32.13428571428568, 279.55885714285694, 2088.6937714285705]

[0.0, 0.0, 116.6427400906905, 1505.0552897892667]

[0.0, 0.0, 1505.0552897892776, 19683.146104313142]

B = [61.269999999999996, 54.15857142857135, -6.864766391037222, -90.16736867964391]

<<Iteration №3>>

A = [7.0, 31.300000000000004, 172.09, 1049.047]

[0.0, 32.13428571428568, 279.55885714285694, 2088.6937714285705]

[0.0, 0.0, 116.6427400906905, 1505.0552897892667]

[0.0, 0.0, 0.0, 263.2368706868692]

B = [61.269999999999996, 54.15857142857135, -6.864766391037222, -1.5904631477168039]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

X = [0.6397716678970694, 1.9118771220533892, 0.019107039841767988, -0.006041946721091072]

S = 0,059

Theta = 0,092

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

FOURTH APPROXIMATION(power function)

<<INPUT MATRIX>>

A = [7.0, 9.358854105460436]

[9.358854105460436, 15.255173133797939]

B = [14.373964419085535, 21.516072189105785]

<<Iteration №1>>

A = [7.0, 9.358854105460436]

[0.0, 2.742580252755989]

B = [14.373964419085535, 2.2983813440628005]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

X = [0.9329866476654225, 0.8380361310314192]

S = 0,154

Theta = 0,149

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

FIFTH APPROXIMATION(exponential function)

<<INPUT MATRIX>>

A = [7.0, 31.300000000000004]

[31.300000000000004, 172.09]

B = [14.373964419085535, 71.80926739856876]

<<Iteration №1>>

A = [7.0, 31.300000000000004]

[0.0, 32.13428571428568]

B = [14.373964419085535, 7.537112210372001]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

X = [1.004647760759436, 0.23455048222905692]

S = 10,707

Theta = 1,237

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

SIXTH APPROXIMATION(logarithm function)

<<INPUT MATRIX>>

A = [7.0, 9.358854105460436]

[9.358854105460436, 15.255173133797939]

B = [61.269999999999996, 97.41239291917515]

<<Iteration №1>>

A = [7.0, 9.358854105460436]

[0.0, 2.742580252755989]

B = [61.269999999999996, 15.495679913237879]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

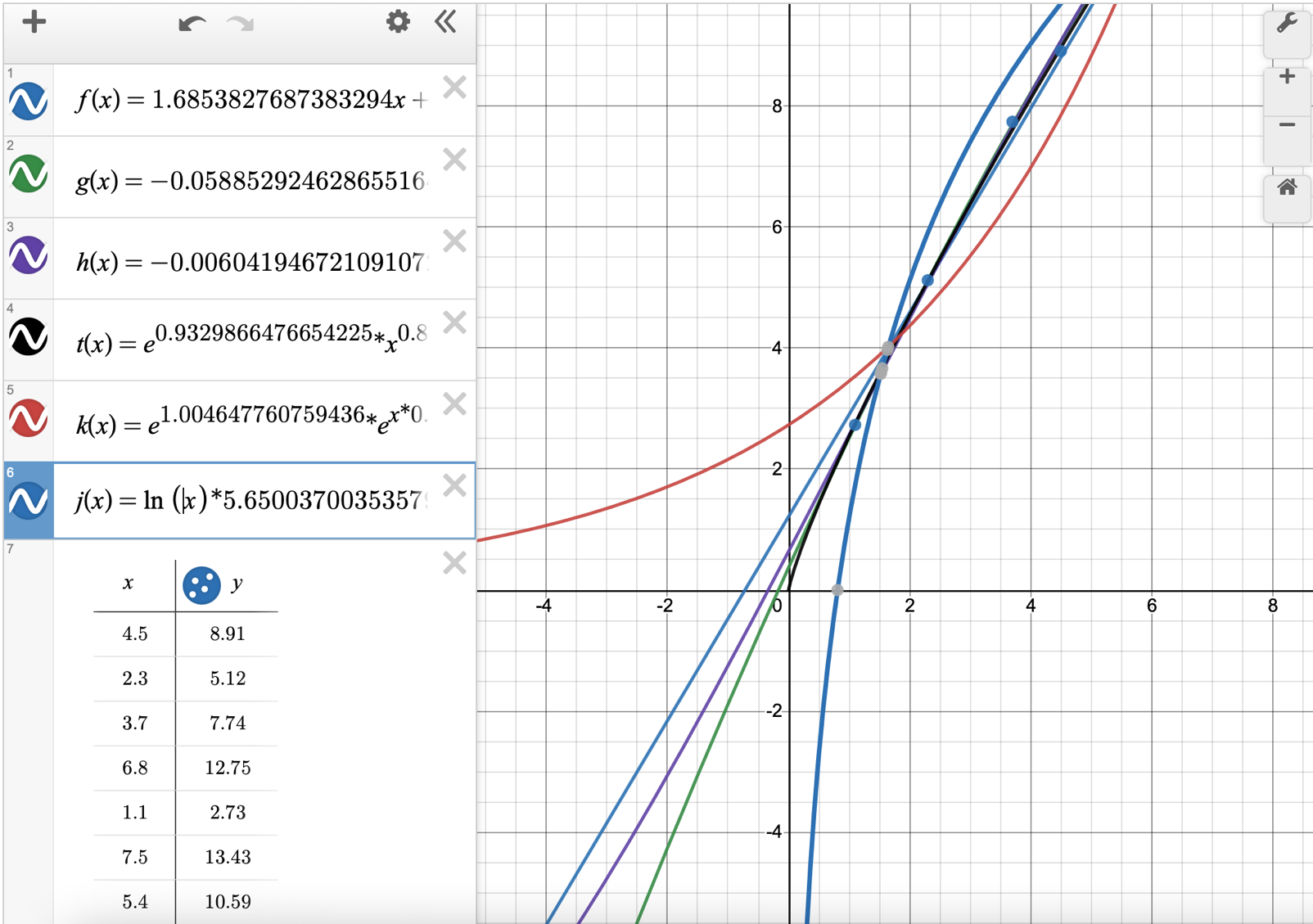
X = [1.1988754276365259, 5.6500370035357905]

S = 4,200

Theta = 0,775

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

THE BEST APPROXIMATION FUNCTION IS: Polynom 3rd



1. **Вывод работы:**

Я использовал разные апроксимирующие функции для входного набора точек, реализовав их с помощью метода Гаусса (для вычисления коэффициентов аппроксимирующих функций). Также смог вывести графики и вычислять ко/ффициенты Пирсона и т.д.