Федеральное государственное автономное образовательное
учреждение высшего образования Национальный
исследовательский университет ИТМО

Факультет Программной Инженерии и Компьютерной Техники

Лабораторная работа №4
Вычислительная математика

Вариант: №1

Группа	P3208
Студент	Абдуллин И.Э.
Преподаватель	Машина Е.А.

Цель работы:

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

Вычислительная реализация задачи:

Исследуемый интервал:

$$x \subseteq [0, 2]; h = 0.2$$

Функция:

$$y = \frac{12x}{x^4 + 1}$$

1. ***Сформировать таблицу табулирования заданной функции на указанном***

x_i	0	0.2	0.4	0.6	0.8	1	1.2	1.4	1.6	1.8	2
y_i	0	2.396	4.68	6.374	6.81	6	4.69	3.47	2.541	1.879	1.412

2. ***Построить линейное и квадратичное приближения по 11 точкам заданного интервала***

Линейное приближение:

$$\phi_1(x) = ax + b$$

$$SX = \sum_{i=1}^{n} x_i = 11 \qquad SXX = \sum_{i=1}^{n} (x_i)^2 = 15.4$$

$$SY = \sum_{i=1}^{n} y_i = 40.248 \qquad SXY = \sum_{i=1}^{n} x_i * y_i = 38.376$$

$$\begin{cases} a = \dfrac{SXY \cdot n - SX \cdot SY}{SXX \cdot n - SX \cdot SX} \\ b = \dfrac{SXX \cdot SY - SX \cdot SXY}{SXX \cdot n - SX \cdot SX} \end{cases} = \begin{cases} a \approx -0.425 \\ b \approx 4.084 \end{cases}$$

$$\phi_1(x) = -0.425x + 4.084$$

x_i	0	0.2	0.4	0.6	0.8	1	1.2	1.4	1.6	1.8	2
y_i	0	2.396	4.68	6.374	6.81	6	4.69	3.47	2.541	1.879	1.412
ϕ_{1i}	4.084	4	3.914	3.829	3.744	3.659	3.574	3.489	3.404	3.319	3.234
ε_i	4.084	1.603	-0.766	-2.545	-3.066	-2.341	-1.111	0.019	0.862	1.44	1.822

$$\sigma_1 = \sqrt{\frac{\sum_{i=1}^{n} \varepsilon_i^2}{n}} \simeq 2.101$$

Квадратичное приближение:

$$\phi_2(x) = a + bx + cx^2$$

$$SX = \sum_{i=1}^{n} x_i = 11 \qquad SXX = \sum_{i=1}^{n} (x_i)^2 = 15.4$$

$$SXX = \sum_{i=1}^{n} (x_i)^2 = 15.4$$

$$\sum_{i=1}^{n} (x_i)^3 = 24.2$$

$$\sum_{i=1}^{n} (x_i)^4 = 40.533 \qquad SY = \sum_{i=1}^{n} y_i = 40.248$$

$$SXY = \sum_{i=1}^{n} x_i * y_i = 38.376$$

$$\sum_{i=1}^{n} (x_i)^2 * y_i = 45.287$$

$$\begin{cases} n \cdot a + SX \cdot b + SXX \cdot c = SY \\ SX \cdot a + SXX \cdot b + S3X \cdot c = SXY \\ SXX \cdot a + S3X \cdot b + S4X \cdot c = SXXY \end{cases}$$

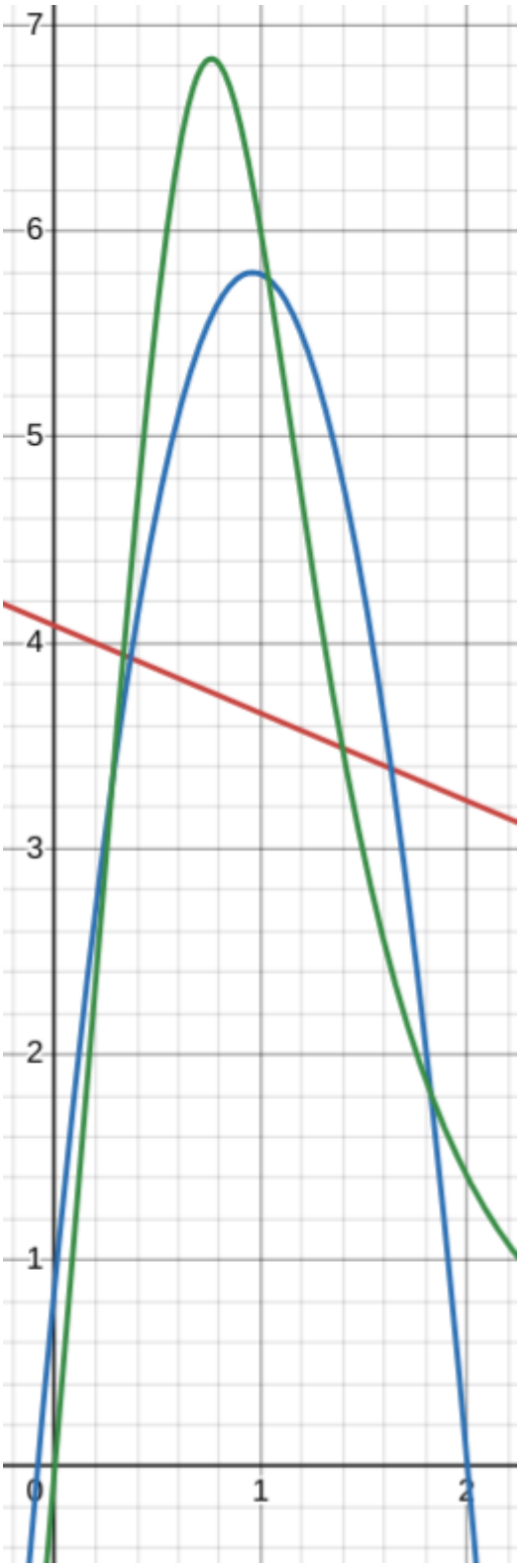
$$\begin{cases} a \approx 0.887 \\ b \approx 10.232 \\ c \approx -5.327 \end{cases}$$

$$\phi_2(x) = 0.887 + 10.232x - 5.327x^2$$

x_i	0	0.2	0.4	0.6	0.8	1	1.2	1.4	1.6	1.8	2
y_i	0	2.396	4.68	6.374	6.81	6	4.69	3.47	2.541	1.879	1.412
ϕ_{2i}	0.887	2.72	4.127	5.108	5.663	5.792	5.494	4.77	3.621	2.045	0.043
ε_i	0.887	0.324	-0.552	-1.265	-1.147	-0.208	0.809	1.3	1.079	0.166	-1.368

$$\sigma_2 = \sqrt{\frac{\sum_{i=1}^{n} \varepsilon_i^2}{n}} \simeq 0.933$$

$\sigma_2 < \sigma_1 \Rightarrow$ квадратичное приближение – наилучшее



Красная - линейное приближение

Зеленая - исходная функция

Синий - квадратичное приближение

Программная реализация задачи:

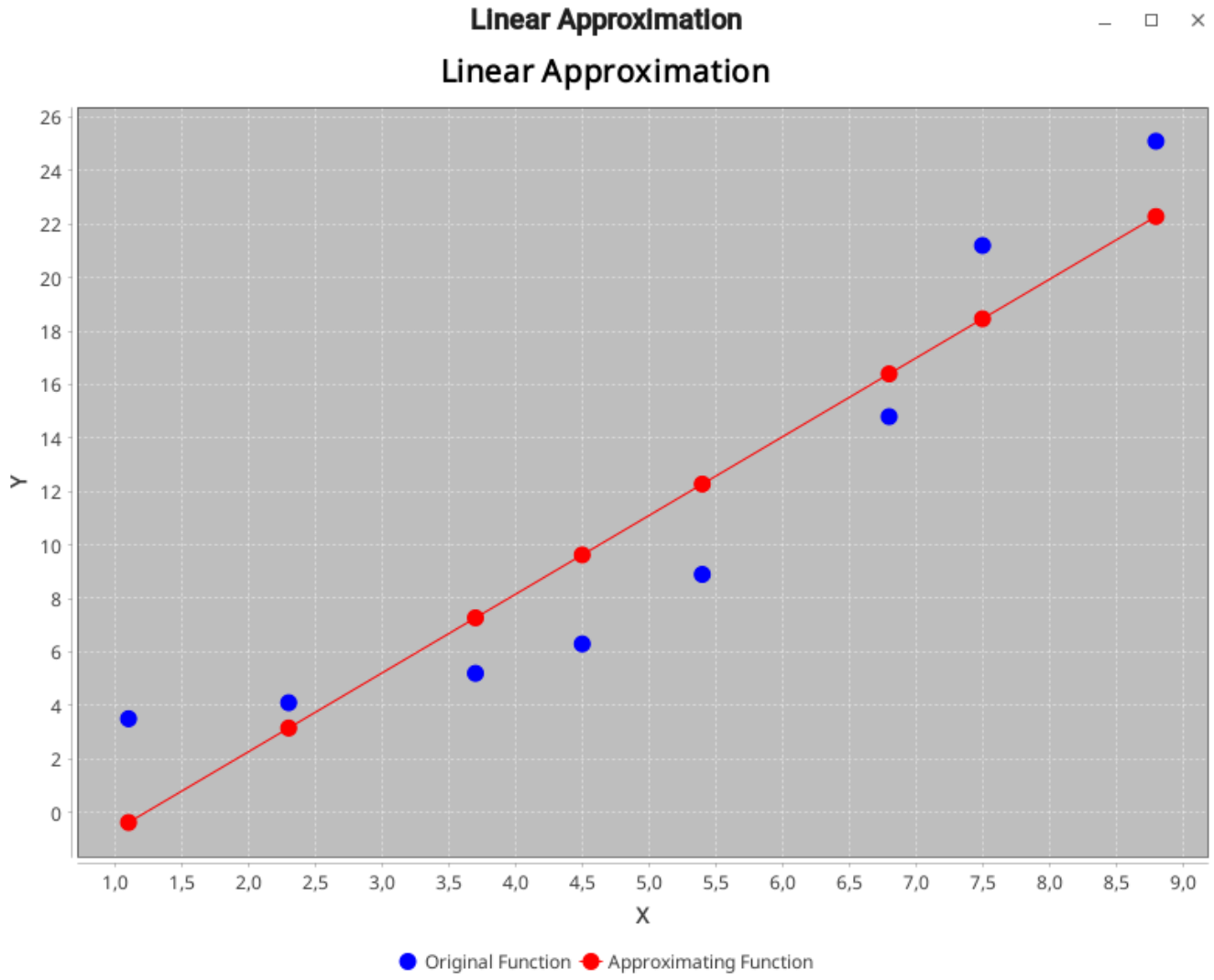
```
package Abdullin_367039.lab4;

import javax.swing.*;

public class LinearApproximation {
 private final int number = 1;
 private double a = 0;
 private double b = 0;
 private double sko = 0;
 private double[] epsilon;

 public double[] solve(double[] x, double[] y, int amount) {
   double sx = 0;
   double sxx = 0;
   double sy = 0;
   double sxy = 0;
   for (int i = 0; i < amount; i++) {
     sx += x[i];
   }
   for (int i = 0; i < amount; i++) {
     sxx += x[i] * x[i];
   }
   for (int i = 0; i < amount; i++) {
     sy += y[i];
   }
   for (int i = 0; i < amount; i++) {
     sxy += x[i] * y[i];
   }
   a = (sxy * amount - sx * sy) / (sxx * amount - sx * sx);
   b = (sxx * sy - sx * sxy) / (sxx * amount - sx * sx);
   double[] result = new double[amount];
   for (int i = 0; i < amount; i++) {
     result[i] = a * x[i] + b;
   }
   epsilon = new double[amount];
   for (int i = 0; i < amount; i++) {
     epsilon[i] = result[i] - y[i];
   }
   for (int i = 0; i < amount; i++) {
     sko += epsilon[i] * epsilon[i];
   }
   sko = Math.sqrt(sko / amount);
   return result;
 }

 public void draw(double[] x, double[] y, double[] result, int amount) {
   String title = "Linear Approximation";
   FunctionDrawer functionDrawer = new FunctionDrawer(title, amount, x, y, result);
   functionDrawer.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   functionDrawer.pack();
   functionDrawer.setVisible(true);
 }

 public double getPearsonCoefficient(double[] x, double[] y, int amount) {
   double midX = 0;
   double midY = 0;
   double r = 0;
   for (int i = 0; i < amount; i++) {
     midX += x[i];
     midY += y[i];
   }
   midX = midX / amount;
   midY = midY / amount;
   double chisl = 0;
   double znam = 0;
   for (int i = 0; i < amount; i++) {
     chisl += (x[i] - midX) * (y[i] - midY);
     znam += (x[i] - midX) * (x[i] - midX) * (y[i] - midY) * (y[i] - midY);
```

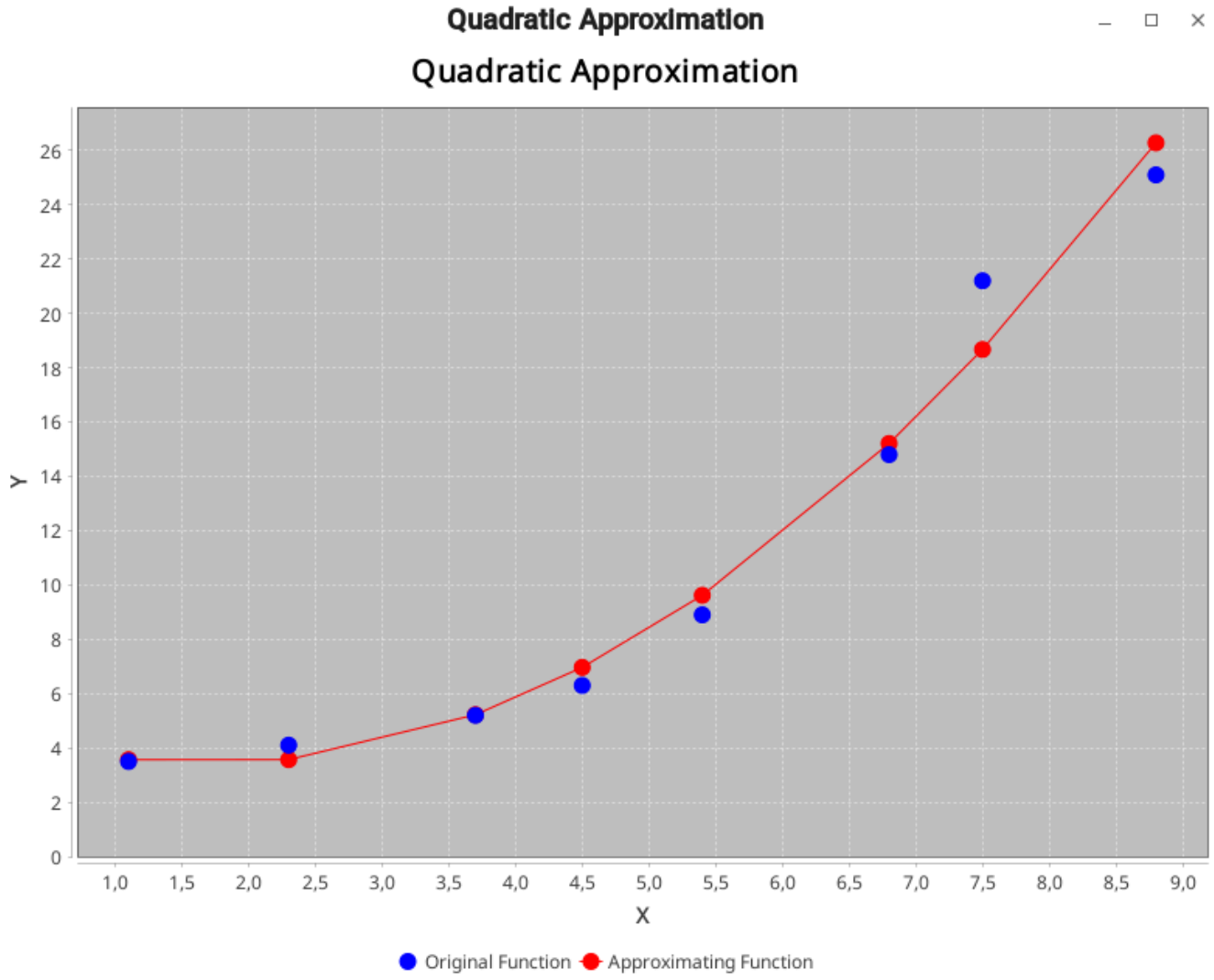
```
    }
    r = chisl / znam;
    return r;
  }

  public double getDeterminationCoefficient(double[] result, double y[], int amount) {
    double midPhi = 0;
    double r2 = 0;
    for (int i = 0; i < amount; i++) {
      midPhi += result[i];
    }
    midPhi = midPhi / amount;
    double chisl = 0;
    double znam = 0;
    for (int i = 0; i < amount; i++) {
      chisl += (y[i] - result[i]) * (y[i] - result[i]);
      znam += (y[i] - midPhi) * (y[i] - midPhi);
    }
    r2 = 1 - chisl / znam;
    return r2;
  }

  public double getA() {
    return a;
  }

  public double getB() {
    return b;
  }

  public double[] getEpsilon() {
    return epsilon;
  }

  public double getSko() {
    return sko;
  }
}
```

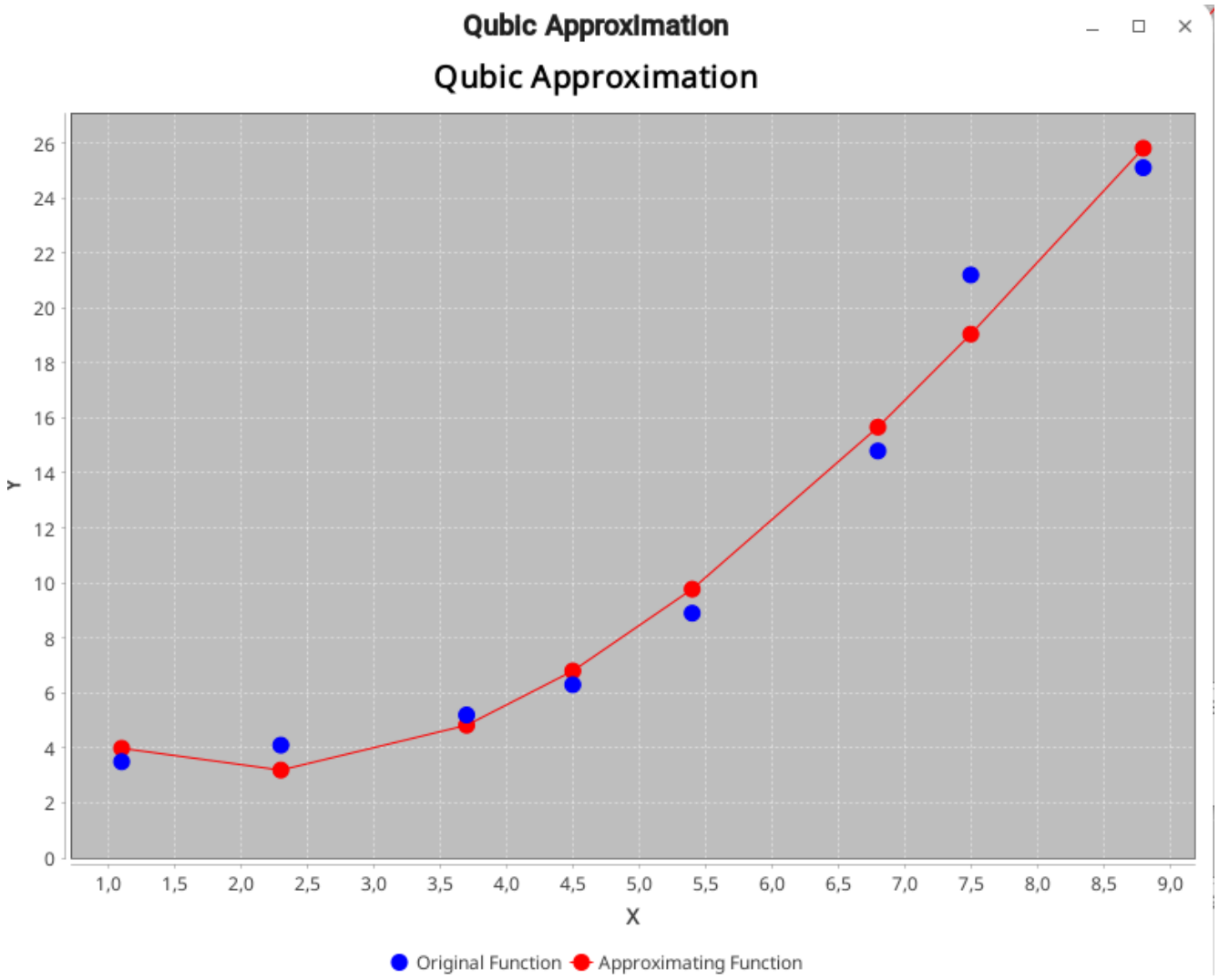
```
package Abdullin_367039.lab4;

public class QuadraticApproximation {
  private final int number = 2;
  private double[] epsilon;
  double a0 = 0;
  double a1 = 0;
  double a2 = 0;
  double sko = 0;

  public double[] solve(double[] x, double[] y, int amount) {
    double sx = 0;
    double sxx = 0;
    double sxxx = 0;
    double sxxxx = 0;
    double sy = 0;
    double sxy = 0;
    double sxxy = 0;
    for (int i = 0; i < amount; i++) {
      sx += x[i];
    }
    for (int i = 0; i < amount; i++) {
      sxx += x[i] * x[i];
    }
    for (int i = 0; i < amount; i++) {
      sxxx += x[i] * x[i] * x[i];
    }
    for (int i = 0; i < amount; i++) {
      sxxxx += x[i] * x[i] * x[i] * x[i];
    }
    for (int i = 0; i < amount; i++) {
```

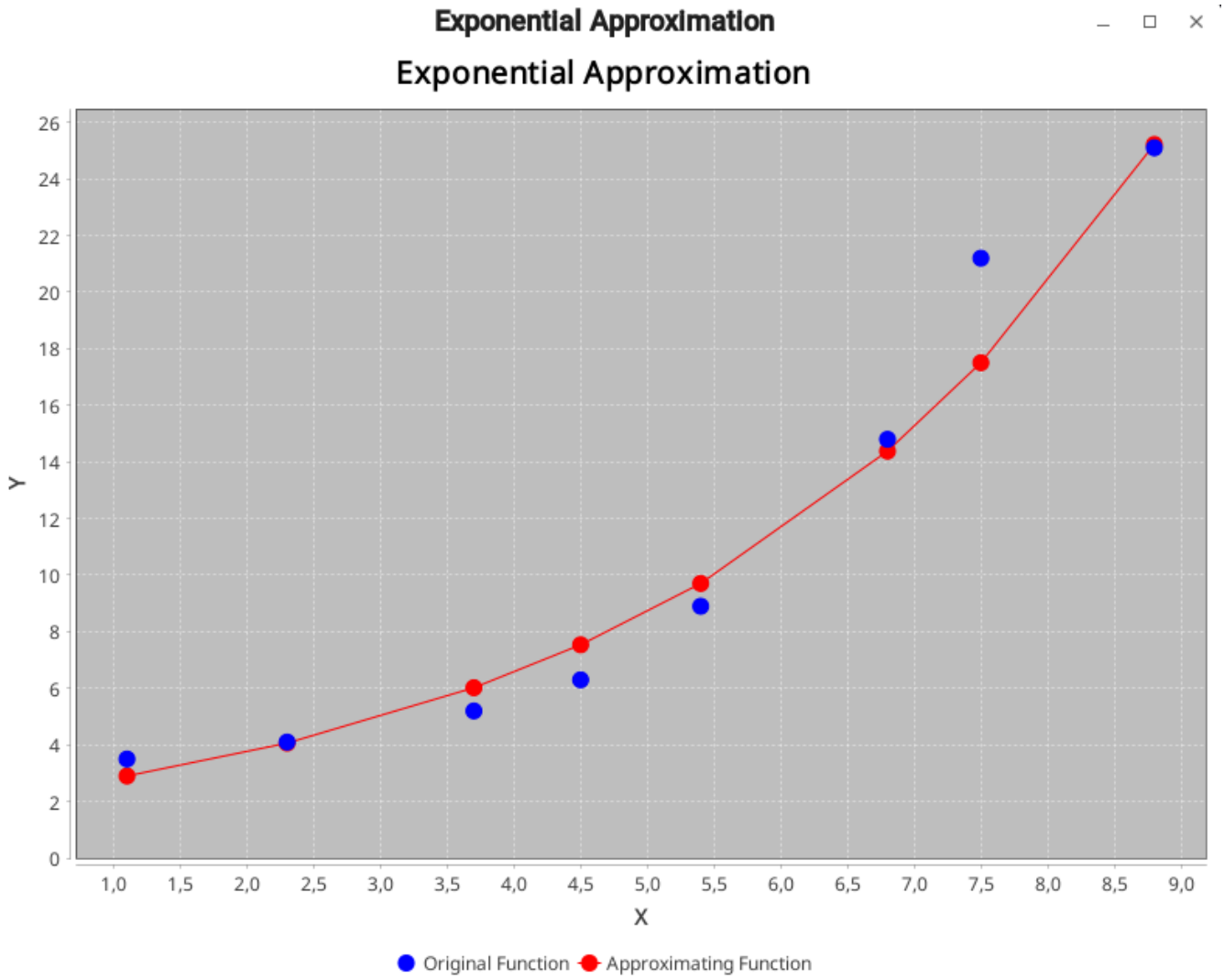
```
    sy += y[i];
  }
  for (int i = 0; i < amount; i++) {
    sxy += x[i] * y[i];
  }
  for (int i = 0; i < amount; i++) {
    sxxy += x[i] * x[i] * y[i];
  }
  a0 =
      (sy * sxx * sxxxx
              + sxxy * sx * sxxx
              + sxy * sxx * sxxx
              - sxxy * sxx * sxx
              - sx * sxy * sxxxx
              - sy * sxxx * sxxx)
          / (amount * sxx * sxxxx
              + sx * sxx * sxxx
              + sxx * sx * sxxx
              - sxx * sxx * sxx
              - sx * sx * sxxxx
              - amount * sxxx * sxxx);
  a1 =
      (amount * sxy * sxxxx
              + sx * sxx * sxxy
              + sxx * sy * sxxx
              - sxx * sxy * sxx
              - sx * sy * sxxxx
              - amount * sxxy * sxxx)
          / (amount * sxx * sxxxx
              + sx * sxx * sxxx
              + sxx * sx * sxxx
              - sxx * sxx * sxx
              - sx * sx * sxxxx
              - amount * sxxx * sxxx);
  a2 =
      (amount * sxx * sxxy
              + sx * sy * sxxx
              + sxx * sx * sxy
              - sxx * sxx * sy
              - sx * sx * sxxy
              - amount * sxxx * sxy)
          / (amount * sxx * sxxxx
              + sx * sxx * sxxx
              + sxx * sx * sxxx
              - sxx * sxx * sxx
              - sx * sx * sxxxx
              - amount * sxxx * sxxx);
  double[] result = new double[amount];
  for (int i = 0; i < amount; i++) {
    result[i] = a0 + a1 * x[i] + a2 * x[i] * x[i];
  }
  epsilon = new double[amount];
  for (int i = 0; i < amount; i++) {
    epsilon[i] = result[i] - y[i];
  }
  for (int i = 0; i < amount; i++) {
    sko += epsilon[i] * epsilon[i];
  }
  sko = Math.sqrt(sko / amount);
  return result;
}

public void draw(double[] x, double[] y, double[] result, int amount) {
  String title = "Quadratic Approximation";
}

public double getDeterminationCoefficient(double[] result, double y[], int amount) {
  double midPhi = 0;
  double r2 = 0;
  for (int i = 0; i < amount; i++) {
    midPhi += result[i];
  }
  midPhi = midPhi / amount;
```

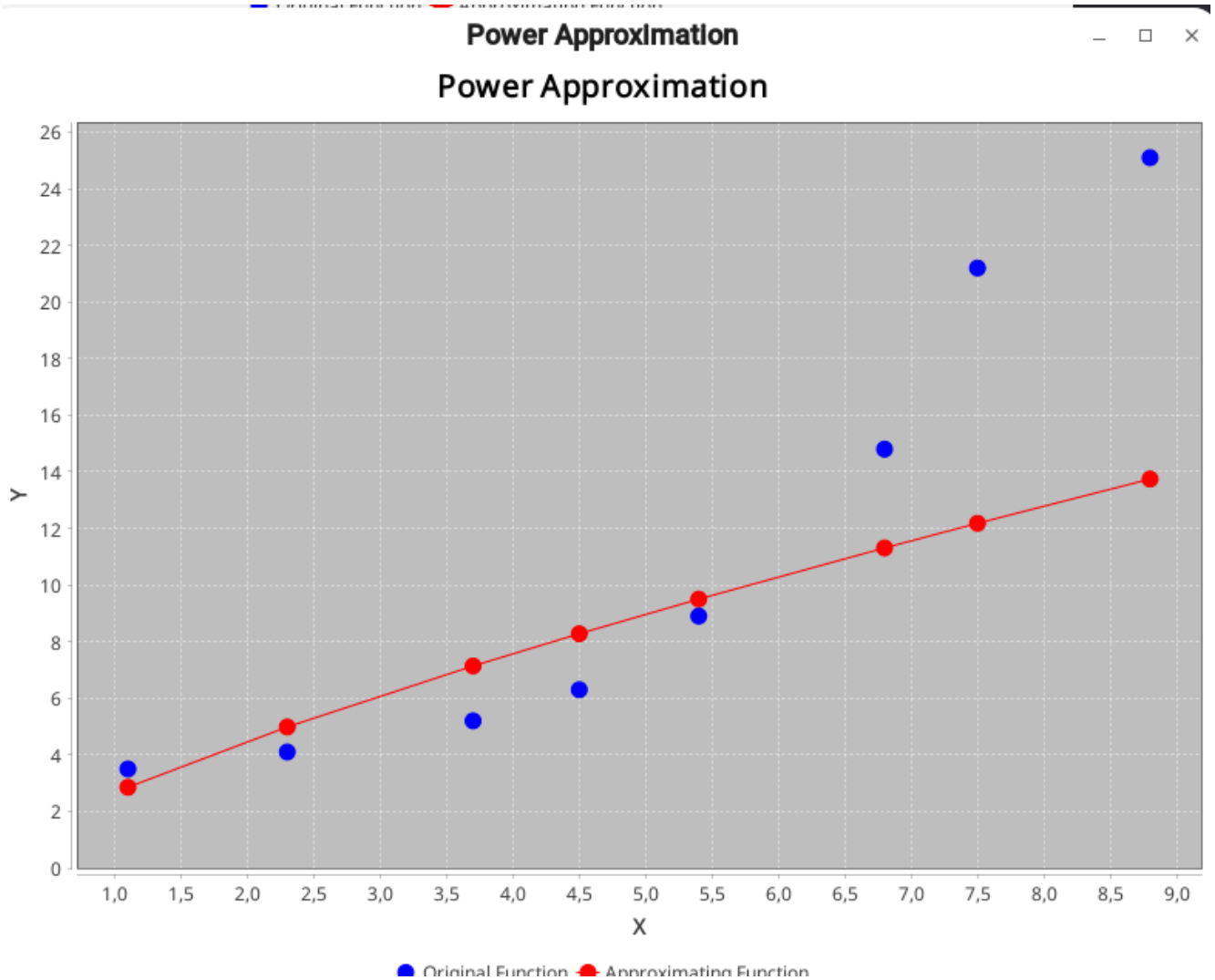
```
    double chisl = 0;
    double znam = 0;
    for (int i = 0; i < amount; i++) {
      chisl += (y[i] - result[i]) * (y[i] - result[i]);
      znam += (y[i] - midPhi) * (y[i] - midPhi);
    }
    r2 = 1 - chisl / znam;
    return r2;
  }

  public double getA0() {
    return a0;
  }

  public double getA1() {
    return a1;
  }

  public double getA2() {
    return a2;
  }

  public double[] getEpsilon() {
    return epsilon;
  }

  public double getSko() {
    return sko;
  }
}
```

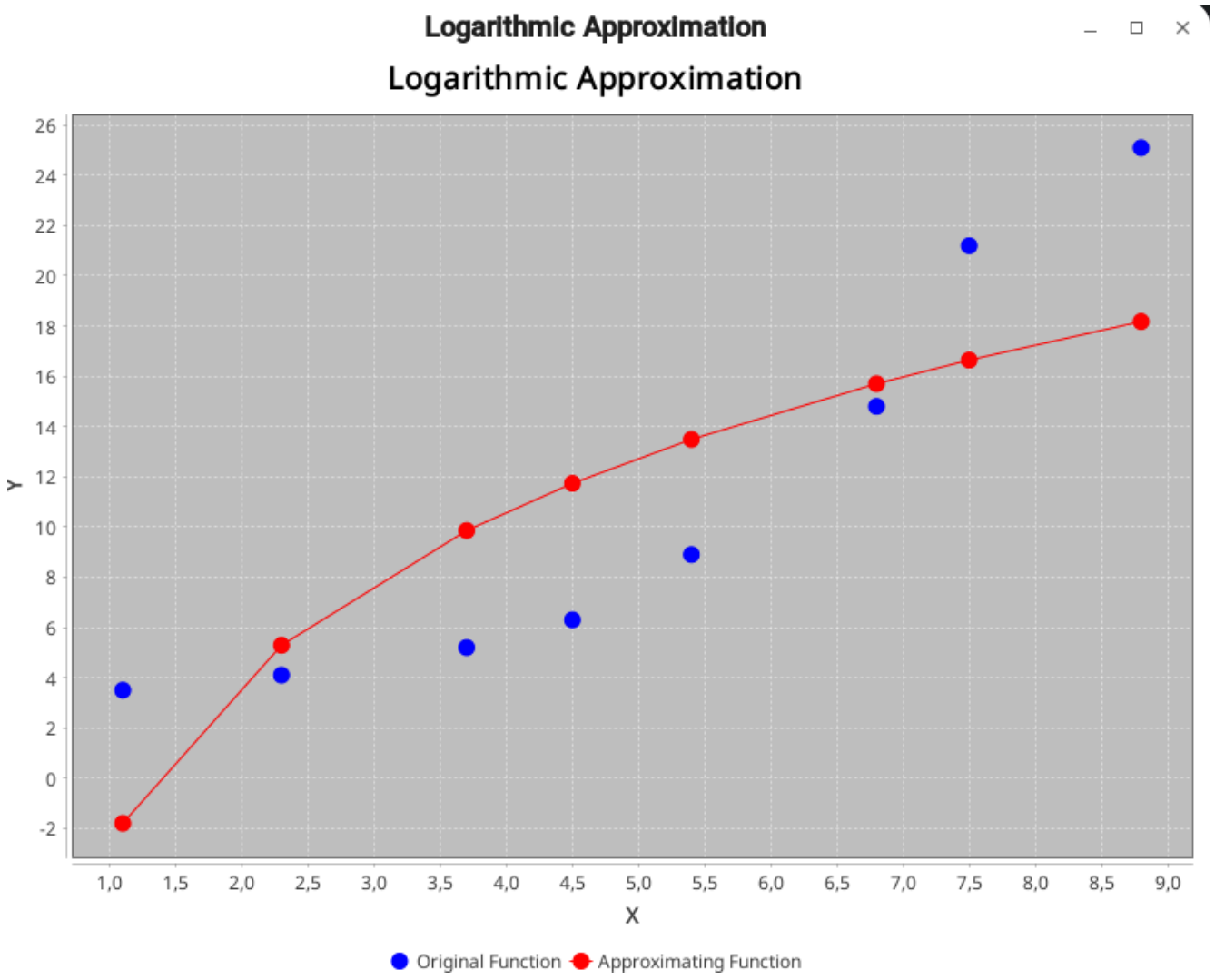

Linear Approximation
Linear Approximation
26
24
22
20
18
16
14
12
10
8
6
4
2
0
Y
1,0
1,5
2,0
2,5
3,0
3,5
4,0
4,5
5,0
5,5
6,0
6,5
7,0
7,5
8,0
8,5
9,0
X
Original Function
Approximating Function


Quadratic Approximation
Quadratic Approximation
26
24
22
20
18
16
14
12
10
8
6
4
2
0
Y
1,0
1,5
2,0
2,5
3,0
3,5
4,0
4,5
5,0
5,5
6,0
6,5
7,0
7,5
8,0
8,5
9,0
X
Original Function
Approximating Function


Qubic Approximation
Qubic Approximation
Y
X
0
2
4
6
8
10
12
14
16
18
20
22
24
26
1,0
1,5
2,0
2,5
3,0
3,5
4,0
4,5
5,0
5,5
6,0
6,5
7,0
7,5
8,0
8,5
9,0
Original Function
Approximating Function


Exponential Approximation
Exponential Approximation
Y
X
0
2
4
6
8
10
12
14
16
18
20
22
24
26
1,0
1,5
2,0
2,5
3,0
3,5
4,0
4,5
5,0
5,5
6,0
6,5
7,0
7,5
8,0
8,5
9,0
Original Function
Approximating Function


Power Approximation
Power Approximation
26
24
22
20
18
16
14
12
10
8
6
4
2
0
Y
1,0
1,5
2,0
2,5
3,0
3,5
4,0
4,5
5,0
5,5
6,0
6,5
7,0
7,5
8,0
8,5
9,0
X
Original Function
Approximating Function



Выводы:

Во время выполнения работы мне удалось изучить различные виды аппроксимации: линейную, квадратичную, кубическую, логарифмическую, экспоненциальную и степенную.
Нельзя однозначно сказать, какая аппроксимирующая функция лучше, так как это зависит от самих экспериментальных данных.