Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет Программной Инженерии и Компьютерной Техники

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

A picture containing logo

Description automatically generated

Лабораторная работа №4 по дисциплине  
«Вычислительная математика»

" АППРОКСИМАЦИЯ ФУНКЦИИ МЕТОДОМ НАИМЕНЬШИХ КВАДРАТОВ"

Выполнил: Дау Конг Туан Ань

Группа: P32151

Преподаватель: Машина Е.А

г. Санкт-Петербург

2023

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

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

Table

Description automatically generated

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| X | 0 | 0.2 | 0.4 | 0.6 | 0.8 | 1 | 1.2 | 1.4 | 1.6 | 1.8 | 2 |
| Y | 0 | 0.089 | 0.177 | 0.263 | 0.34 | 0.4 | 0.433 | 0.436 | 0.412 | 0.369 | 0.32 |

Chart, scatter chart

Description automatically generated

Text, letter

Description automatically generated Text

Description automatically generated

SX = 11, SXX =15.4, SY = 3.24, SXY = 4.61

*Chart, line chart, scatter chart

Description automatically generated*

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| X | 0 | 0.2 | 0.4 | 0.6 | 0.8 | 1 | 1.2 | 1.4 | 1.6 | 1.8 | 2 |
| Y | 0 | 0.089 | 0.177 | 0.263 | 0.34 | 0.4 | 0.433 | 0.436 | 0.412 | 0.369 | 0.32 |
|  | -0.03 | 0.032 | 0.094 | 0.157 | 0.219 | 0.281 | 0.343 | 0.405 | 0.468 | 0.53 | 0.592 |
|  | -0.03 | -0.057 | -0.083 | -0.106 | -0.121 | -0.119 | -0.09 | -0.031 | 0.056 | 0.161 | 0.272 |

*S = 0.163*

*Timeline

Description automatically generated*

*Chart, scatter chart

Description automatically generated*

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| X | 0 | 0.2 | 0.4 | 0.6 | 0.8 | 1 | 1.2 | 1.4 | 1.6 | 1.8 | 2 |
| Y | 0 | 0.089 | 0.177 | 0.263 | 0.34 | 0.4 | 0.433 | 0.436 | 0.412 | 0.369 | 0.32 |
|  | -0.235 | -0.107 | 0.02 | 0.145 | 0.268 | 0.389 | 0.509 | 0.626 | 0.742 | 0.857 | 0.969 |
|  | -0.235 | -0.196 | -0.157 | -0.118 | -0.072 | -0.011 | 0.075 | 0.19 | 0.331 | 0.487 | 0.649 |

*S = 0.947*

1. Программная реализация задачи
2. Linear Function

@Override  
public OutputSet execute(InputSet inputSet) {  
 Double a;  
 Double b;  
 Double SX = 0.0, SY = 0.0, SXX=0.0, SXY=0.0;  
 List<Point> pointList = inputSet.getPointList();  
 for(int i = 0 ;i < inputSet.getPointList().size(); ++i) {  
 SX += pointList.get(i).getX();  
 SY += pointList.get(i).getY();  
 SXX += Math.*pow*(pointList.get(i).getX(), 2);  
 SXY += pointList.get(i).getX()\*pointList.get(i).getY();  
 }  
  
 a = (SXY\* inputSet.getN() - SX\*SY) / (SXX\* inputSet.getN() - Math.*pow*(SX, 2));  
 b = (SXX\* SY - SX\*SXY) / (SXX\* inputSet.getN() - Math.*pow*(SX, 2));  
  
 LinearEquation linearEquation = new LinearEquation(a, b);  
 for(int i = 0 ;i < inputSet.getN(); ++i) {  
 inputSet.getPointList().get(i).setF\_x(  
 linearEquation.getValue(pointList.get(i).getX()));  
 inputSet.getPointList().get(i).setError();  
 }  
  
 return new OutputSet( linearEquation, getR(inputSet), getS(inputSet), getDelta(inputSet));  
}

1. Polynomial Function 2nd

@Override  
 public OutputSet execute(InputSet inputSet) {  
 double[][] matrix = {{0.0, 0.0,0.0}, {0.0, 0.0,0.0}, {0.0, 0.0,0.0}};  
 double[] res = {0.0, 0.0, 0.0};  
 matrix[0][0] = inputSet.getN();  
 for(int i = 0 ;i < matrix[0][0];++i) {  
 matrix[0][1] += Math.*pow*(inputSet.getPointList().get(i).getX(), 1);  
 matrix[0][2] += Math.*pow*(inputSet.getPointList().get(i).getX(), 2);  
 matrix[1][2] += Math.*pow*(inputSet.getPointList().get(i).getX(), 3);  
 matrix[2][2] += Math.*pow*(inputSet.getPointList().get(i).getX(), 4);  
 res[0] += inputSet.getPointList().get(i).getY();  
 res[1] += Math.*pow*(inputSet.getPointList().get(i).getX(), 1) \* inputSet.getPointList().get(i).getY();  
 res[2] += Math.*pow*(inputSet.getPointList().get(i).getX(), 2) \* inputSet.getPointList().get(i).getY();  
 }  
  
 matrix[1][0] = matrix[0][1];  
 matrix[1][1] = matrix[0][2];  
 matrix[2][0] = matrix[1][1];  
 matrix[2][1] = matrix[1][2];  
Matrix A = new Matrix(matrix);  
 Matrix B = new Matrix(3, 1);  
 B.set(0,0,res[0]);  
 B.set(1,0,res[1]);  
 B.set(2,0,res[2]);  
 Matrix X = A.solve(B);  
  
 PolynomialEquation\_2 equation\_2 = new PolynomialEquation\_2(X.get(2, 0), X.get(1,0), X.get(0, 0));  
 for(int i = 0; i < inputSet.getN();++i) {  
 inputSet.getPointList().get(i).setF\_x(equation\_2.getValue(inputSet.getPointList().get(i).getX()));  
 inputSet.getPointList().get(i).setError();  
 }  
  
 return new OutputSet( equation\_2, getR(inputSet), getS(inputSet), getDelta(inputSet));

1. Polynomial Function 3rd

@Override  
public OutputSet execute(InputSet inputSet) {  
 double[][] matrix = {{0.0, 0.0,0.0, 0.0}, {0.0, 0.0,0.0, 0.0}, {0.0, 0.0,0.0, 0.0},{0.0, 0.0,0.0, 0.0} };  
 double[] res = {0.0, 0.0, 0.0, 0.0};  
 matrix[0][0] = inputSet.getN();  
 for(int i = 0 ;i < matrix[0][0];++i) {  
 matrix[0][1] += Math.*pow*(inputSet.getPointList().get(i).getX(), 1);  
 matrix[0][2] += Math.*pow*(inputSet.getPointList().get(i).getX(), 2);  
 matrix[0][3] += Math.*pow*(inputSet.getPointList().get(i).getX(), 3);  
 matrix[1][3] += Math.*pow*(inputSet.getPointList().get(i).getX(), 4);  
 matrix[2][3] += Math.*pow*(inputSet.getPointList().get(i).getX(), 5);  
 matrix[3][3] += Math.*pow*(inputSet.getPointList().get(i).getX(), 6);  
 res[0] += inputSet.getPointList().get(i).getY();  
 res[1] += Math.*pow*(inputSet.getPointList().get(i).getX(), 1) \* inputSet.getPointList().get(i).getY();  
 res[2] += Math.*pow*(inputSet.getPointList().get(i).getX(), 2) \* inputSet.getPointList().get(i).getY();  
 res[3] += Math.*pow*(inputSet.getPointList().get(i).getX(), 3) \* inputSet.getPointList().get(i).getY();  
 }  
  
 matrix[1][0] = matrix[0][1];  
 matrix[1][1] = matrix[0][2];  
 matrix[1][2] = matrix[0][3];  
 matrix[2][0] = matrix[1][1];  
 matrix[2][1] = matrix[1][2];  
 matrix[2][2] = matrix[1][3];  
 matrix[3][0] = matrix[2][1];  
 matrix[3][1] = matrix[2][2];  
 matrix[3][2] = matrix[2][3];  
 Matrix A = new Matrix(matrix);  
 Matrix B = new Matrix(4, 1);  
 B.set(0,0,res[0]);  
 B.set(1,0,res[1]);  
 B.set(2,0,res[2]);  
 B.set(3, 0, res[3]);  
 Matrix X = A.solve(B);  
  
 PolynomialEquation\_3 equation\_3 = new PolynomialEquation\_3(X.get(3, 0), X.get(2,0), X.get(1, 0), X.get(0,0));  
 for(int i = 0; i < inputSet.getN();++i) {  
 inputSet.getPointList().get(i).setF\_x(equation\_3.getValue(inputSet.getPointList().get(i).getX()));  
 inputSet.getPointList().get(i).setError();  
 }  
  
 return new OutputSet( equation\_3, getR(inputSet), getS(inputSet), getDelta(inputSet));  
}

1. Exponential Function

@Override  
public OutputSet execute(InputSet inputSet) {  
 Integer n = inputSet.getN();  
 double[] x = new double[n];  
 double[] y = new double[n];  
 double[] f\_x = new double[n];  
  
 for(int i = 0; i < n; ++i) {  
 x[i] = inputSet.getPointList().get(i).getX();  
 y[i] = Math.*log*(inputSet.getPointList().get(i).getY());  
 }  
  
 Double SX = 0.0, SY = 0.0, SXX=0.0, SXY=0.0;  
 for(int i = 0 ;i < n; ++i) {  
 SX += x[i];  
 SY += y[i];  
 SXX += Math.*pow*(x[i], 2);  
 SXY += x[i]\*y[i];  
 }  
  
 double a = (SXY\* inputSet.getN() - SX\*SY) / (SXX\* inputSet.getN() - Math.*pow*(SX, 2));  
 double b = (SXX\* SY - SX\*SXY) / (SXX\* inputSet.getN() - Math.*pow*(SX, 2));  
  
 ExponentialEquation equation = new ExponentialEquation(Math.*pow*(Math.*E*,b), a);  
  
 for(int i = 0 ;i < n; ++i) {  
 inputSet.getPointList().get(i).setF\_x(equation.getValue(inputSet.getPointList().get(i).getX()));  
 inputSet.getPointList().get(i).setError();  
 }  
  
 return new OutputSet( equation, getR(inputSet), getS(inputSet), getDelta(inputSet));  
}

1. Logarithmic Function

@Override  
public OutputSet execute(InputSet inputSet) {  
 int n = inputSet.getN();  
 double[] x = new double[n];  
 double[] y = new double[n];  
 double[] f\_x = new double[n];  
  
 for(int i = 0; i < n; ++i) {  
 x[i] = Math.*log*(inputSet.getPointList().get(i).getX());  
 y[i] = inputSet.getPointList().get(i).getY();  
 }  
  
 Double SX = 0.0, SY = 0.0, SXX=0.0, SXY=0.0;  
 for(int i = 0 ;i < n; ++i) {  
 SX += x[i];  
 SY += y[i];  
 SXX += Math.*pow*(x[i], 2);  
 SXY += x[i]\*y[i];  
 }  
  
 double a = (SXY\* inputSet.getN() - SX\*SY) / (SXX\* inputSet.getN() - Math.*pow*(SX, 2));  
 double b = (SXX\* SY - SX\*SXY) / (SXX\* inputSet.getN() - Math.*pow*(SX, 2));  
  
 LogarithmicEquation equation = new LogarithmicEquation(a, b);  
  
 for(int i = 0 ;i < n; ++i) {  
 inputSet.getPointList().get(i).setF\_x(equation.getValue(inputSet.getPointList().get(i).getX()));  
 inputSet.getPointList().get(i).setError();  
 }  
  
 return new OutputSet( equation, getR(inputSet), getS(inputSet), getDelta(inputSet));  
 }

1. Power Function
2. @Override  
   public OutputSet execute(InputSet inputSet) {  
    Integer n = inputSet.getN();  
    double[] x = new double[n];  
    double[] y = new double[n];  
     
    for(int i = 0; i < n; ++i) {  
    x[i] = Math.*log*(inputSet.getPointList().get(i).getX());  
    y[i] = Math.*log*(inputSet.getPointList().get(i).getY());  
    }  
     
    Double SX = 0.0, SY = 0.0, SXX=0.0, SXY=0.0;  
    for(int i = 0 ;i < n; ++i) {  
    SX += x[i];  
    SY += y[i];  
    SXX += Math.*pow*(x[i], 2);  
    SXY += x[i]\*y[i];  
    }  
     
    double a = (SXY\* inputSet.getN() - SX\*SY) / (SXX\* inputSet.getN() - Math.*pow*(SX, 2));  
    double b = (SXX\* SY - SX\*SXY) / (SXX\* inputSet.getN() - Math.*pow*(SX, 2));  
     
    PowerEquation equation = new PowerEquation(Math.*pow*(Math.*E*,b), a);  
     
    for(int i = 0 ;i < n; ++i) {  
    inputSet.getPointList().get(i).setF\_x(equation.getValue(inputSet.getPointList().get(i).getX()));  
    inputSet.getPointList().get(i).setError();  
    }  
     
    return new OutputSet( equation, getR(inputSet), getS(inputSet), getDelta(inputSet));  
   }

**Result**

**Do you want to set input by console(1) or File(2) ?**

**2**

**Table Output list:**

**Equation: LinearEquation: 2.547x + -2.247**

**R^2: 0.93**

**S: 47.411**

**Delta: 2.602**

**Equation: PolynomialEquation\_2: 0.601 x^2 + -2.686 x +6.365**

**R^2: 0.992**

**S: 5.189**

**Delta: 0.861**

**Equation: PolynomialEquation\_3: 0.127 x^3 + -1.039 x^2 +3.324 x + 0.775**

**R^2: 0.998**

**S: 0.93**

**Delta: 0.364**

**Equation: ExponentialEquation 2.149\* x^0.279**

**R^2: 0.972**

**S: 17.376**

**Delta: 1.575**

**Equation: LogarithmicEquation 7.455 \* ln(x) + -0.824**

**R^2: 0.836**

**S: 103.499**

**Delta: 3.845**

**Equation: PowerEquation 2.35 \*x^0.868**

**R^2: 0.849**

**S: 77.238**

**Delta: 3.321**

**Chart, line chart

Description automatically generated**

**Вывод**: в ходе данной л.р я познакомился с различными видами аппроксимации функции через метод наименьших квадратов. Данный метод довольно просто программируем.