Университет ИТМО, факультет ПИиКТ

Лабораторная работа №3 по

Вычислительной математике

“Аппроксимация функции методом наименьших квадратов”

Вариант 7

Выполнил: Клюев Андрей Викторович  
Группа: №3214

Преподаватель: Малышева Татьяна Алексеевна

Санкт-Петербург, 2020 год

Текст задания:

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

Для исследования использовать:

a)Линейную функцию.

b)Полиноминальную функцию 2-й степени.

c)Экспоненциальную функцию.

d)Логарифмическую функцию.

f)Степенную функцию.

Решение:

Логическая часть программы:

public static void **linearApproximation**(double[] arrX, double[] arrY){

double SX = sumArr(x -> x, arrX);

double SXX = sumArr(x -> Math.pow(x, 2), arrX);

double SY = sumArr(y -> y, arrY);

double SXY = sumArrXY((x,y) -> x\*y, arrX, arrY);

double n = arrX.length;

double a = (SXY\*n - SX\*SY) / (SXX\*n - SX\*SX);

double b = (SXX\*SY - SX\*SXY) / (SXX\*n - SX\*SX);

double S = 0;

double 𝜹 = 0;

for(int i = 0; i < arrX.length; i++){

double e = (a \* arrX[i] + b) - arrY[i];

S += e \* e;

}

𝜹 = Math.sqrt(S / arrX.length);

System.out.println("Мера отклонения S = " + S);

System.out.println("Среднеквадратичное отклонение \uD835\uDF39 = " + 𝜹);

System.out.println("a = " + a);

System.out.println("b = " + b);

}

public static void **polynominalApproximation**(double[] arrX, double[] arrY){

double SX = sumArr(x -> x, arrX);

double SXX = sumArr(x -> Math.pow(x, 2), arrX);

double SXXX = sumArr(x -> Math.pow(x, 3), arrX);

double SXXXX = sumArr(x -> Math.pow(x, 4), arrX);

double SY = sumArr(y -> y, arrY);

double SXY = sumArrXY((x,y) -> x\*y, arrX, arrY);

double SXXY = sumArrXY((x, y) -> x\*x\*y, arrX, arrY);

int m = 3, n = arrX.length;

double[][] a = new double[3][3];

a[0][0] = n;

a[0][1] = SX;

a[0][2] = SXX;

a[1][0] = SX;

a[1][1] = SXX;

a[1][2] = SXXX;

a[2][0] = SXX;

a[2][1] = SXXX;

a[2][2] = SXXXX;

double[] b = new double[3];

b[0] = SY;

b[1] = SXY;

b[2] = SXXY;

n = 3;

MatrixSolver ms = new MatrixSolver();

double[] result = NewGauss.solveMatrix(ms, a, b, n, m);

double S = 0;

double 𝜹 = 0;

for(int i = 0; i < arrX.length; i++){

double e = ((result[2] \* arrX[i]\* arrX[i]) + (result[1] \* arrX[i]) + result[0]) - arrY[i];

S += e \* e;

}

𝜹 = Math.sqrt(S / arrX.length);

System.out.println("\n\nМера отклонения S = " + S);

System.out.println("Среднеквадратичное отклонение \uD835\uDF39 = " + 𝜹);

System.out.println("a = " + result[2]);

System.out.println("b = " + result[1]);

System.out.println("c = " + result[0]);

}

public static void **exponentialApproximation**(double[] arrX, double[] arrY){

double [] ARR\_Y = Arrays.stream(arrY).map(Math::log).toArray();

double SX = sumArr(x -> x, arrX);

double SXX = sumArr(x -> x\*x, arrX);

double SY = sumArr(y -> y, ARR\_Y);

double SXY = sumArrXY((x,y) -> x\*y, arrX, ARR\_Y);

double n = arrX.length;

double tempA = (SXY\*n - SX\*SY) / (SXX\*n - SX\*SX);

double tempB = (SXX\*SY - SX\*SXY) / (SXX\*n - SX\*SX);

double a = Math.exp(tempB);

double b = tempA;

double S = 0;

double 𝜹 = 0;

for(int i = 0; i < arrX.length; i++){

double e = (a \* Math.pow(Math.E, b \* arrX[i])) - arrY[i];

S += e \* e;

}

𝜹 = Math.sqrt(S / arrX.length);

System.out.println("Мера отклонения S = " + S);

System.out.println("Среднеквадратичное отклонение \uD835\uDF39 = " + 𝜹);

System.out.println("a = " + a);

System.out.println("b = " + b);

}

public static void **logarithmicApproximation**(double[] arrX, double[] arrY){

double [] ARR\_X = Arrays.stream(arrX).map(Math::log).toArray();

double SX = sumArr(x -> x, ARR\_X);

double SXX = sumArr(x -> x\*x, ARR\_X);

double SY = sumArr(y -> y, arrY);

double SXY = sumArrXY((x,y) -> x\*y, ARR\_X, arrY);

double n = arrX.length;

double a = (SXY\*n - SX\*SY) / (SXX\*n - SX\*SX);

double b = (SXX\*SY - SX\*SXY) / (SXX\*n - SX\*SX);

double S = 0;

double 𝜹 = 0;

for(int i = 0; i < arrX.length; i++){

double e = (a \* Math.log(arrX[i]) + b) - arrY[i];

S += e \* e;

}

𝜹 = Math.sqrt(S / arrX.length);

System.out.println("Мера отклонения S = " + S);

System.out.println("Среднеквадратичное отклонение \uD835\uDF39 = " + 𝜹);

System.out.println("a = " + a);

System.out.println("b = " + b);

}

public static void **powerApproximation**(double[] arrX, double[] arrY){

double [] logArrY = Arrays.stream(arrY).map(Math::log).toArray();

double [] logArrX = Arrays.stream(arrX).map(Math::log).toArray();

double SX = sumArr(x -> x, logArrX);

double SXX = sumArr(x -> x\*x, logArrX);

double SY = sumArr(y -> y, logArrY);

double SXY = sumArrXY((x,y) -> x\*y, logArrX, logArrY);

int n = arrX.length;

double tempA = (SXY\*n - SX\*SY) / (SXX\*n - SX\*SX);

double tempB = (SXX\*SY - SX\*SXY) / (SXX\*n - SX\*SX);

double a = Math.exp(tempB);

double b = tempA;

double S = 0;

double 𝜹 = 0;

for(int i = 0; i < arrX.length; i++){

double e = (a \* Math.pow(arrX[i], b)) - arrY[i];

S += e \* e;

}

𝜹 = Math.sqrt(S / arrX.length);

System.out.println("Мера отклонения S = " + S);

System.out.println("Среднеквадратичное отклонение \uD835\uDF39 = " + 𝜹);

System.out.println("a = " + a);

System.out.println("b = " + b);

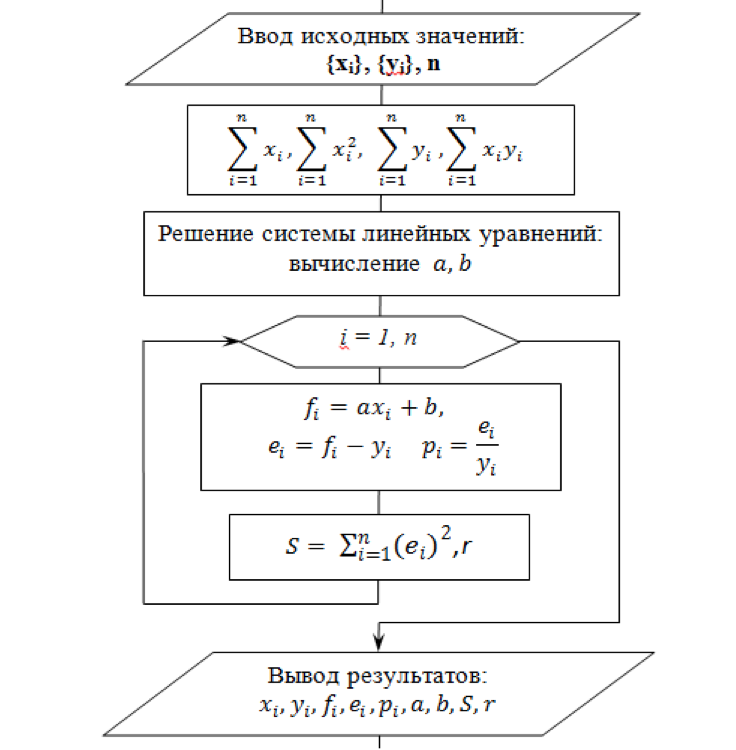
}

**ВЕСЬ КОД ПРОГРАММЫ:**

https://github.com/Amur27RUS/CompMath\_Lab4

Блок-схемы:

Блок схема метода наименьших квадратов:

****

Заполненные таблицы:

|  |  |  |
| --- | --- | --- |
| Вид 𝜹 (x) | | Линейная |
| X | Y | F = ax + b |
| 1.1 | 2.22 | 1.774 |
| 1.3 | 2.57 | 2.355 |
| 1.5 | 2.89 | 2.935 |
| 1.7 | 3.76 | 3.516 |
| 1.9 | 3.98 | 3.097 |
| 2.1 | 4.34 | 4.6773 |
| 2.3 | 4.79 | 5.258 |
| 2.5 | 5.67 | 5.839 |
| 2.7 | 6.21 | 6.419 |
| 2.9 | 6.88 | 6.9995 |
| 3.1 | 7.49 | 7.581 |
| 3.33 | 8.03 | 8.1612 |
| 3.5 | 8.85 | 8.7419 |
| 3.7 | 9.67 | 9.323 |
| 3.9 | 10.23 | 9.9031 |
| S | | 1.004 |
| 𝜹 | | 0.259 |
| a | | 2.9032 |
| b | | -1.4194 |
| c | | - |

|  |  |  |
| --- | --- | --- |
| Вид 𝜹 (x) | | Полиноминальная |
| X | Y | F = ax^2 + bx + c |
| 1.1 | 2.22 | 2.194 |
| 1.3 | 2.57 | 2.595 |
| 1.5 | 2.89 | 3.023 |
| 1.7 | 3.76 | 3.479 |
| 1.9 | 3.98 | 3.963 |
| 2.1 | 4.34 | 4.4743 |
| 2.3 | 4.79 | 5.0134 |
| 2.5 | 5.67 | 5.5802 |
| 2.7 | 6.21 | 6.1747 |
| 2.9 | 6.88 | 6.7969 |
| 3.1 | 7.49 | 7.4468 |
| 3.33 | 8.03 | 8.1243 |
| 3.5 | 8.85 | 9.9296 |
| 3.7 | 9.67 | 9.5625 |
| 3.9 | 10.23 | 10.3231 |
| S | | 0.2137 |
| 𝜹 | | 0.119 |
| a | | 0.3461 |
| b | | 1.17326 |
| c | | 0.485406 |

|  |  |  |
| --- | --- | --- |
| Вид 𝜹 (x) | | экспоненциальная |
| X | Y | F = ae^bx |
| 1.1 | 2.22 | 2.4631 |
| 1.3 | 2.57 | 2.7451 |
| 1.5 | 2.89 | 3.059 |
| 1.7 | 3.76 | 3.4098 |
| 1.9 | 3.98 | 3.800 |
| 2.1 | 4.34 | 4.235 |
| 2.3 | 4.79 | 4.7203 |
| 2.5 | 5.67 | 5.2608 |
| 2.7 | 6.21 | 5.8632 |
| 2.9 | 6.88 | 6.535 |
| 3.1 | 7.49 | 7.283 |
| 3.33 | 8.03 | 8.117 |
| 3.5 | 8.85 | 9.046 |
| 3.7 | 9.67 | 10.0818 |
| 3.9 | 10.23 | 11.2362 |
| S | | 1.967 |
| 𝜹 | | 0.3622 |
| a | | 1.357 |
| b | | 0.5420 |
| c | | - |

|  |  |  |
| --- | --- | --- |
| Вид 𝜹 (x) | | Логарифмическая |
| X | Y | F = a\*lnx + b |
| 1.1 | 2.22 |  |
| 1.3 | 2.57 |  |
| 1.5 | 2.89 |  |
| 1.7 | 3.76 |  |
| 1.9 | 3.98 |  |
| 2.1 | 4.34 |  |
| 2.3 | 4.79 |  |
| 2.5 | 5.67 |  |
| 2.7 | 6.21 |  |
| 2.9 | 6.88 |  |
| 3.1 | 7.49 |  |
| 3.33 | 8.03 |  |
| 3.5 | 8.85 |  |
| 3.7 | 9.67 |  |
| 3.9 | 10.23 |  |
| S | |  |
| 𝜹 | |  |
| a | |  |
| b | |  |
| c | |  |

|  |  |  |
| --- | --- | --- |
| Вид 𝜹 (x) | | Полиноминальная |
| X | Y | F = ax^2 + bx + c |
| 1.1 | 2.22 |  |
| 1.3 | 2.57 |  |
| 1.5 | 2.89 |  |
| 1.7 | 3.76 |  |
| 1.9 | 3.98 |  |
| 2.1 | 4.34 |  |
| 2.3 | 4.79 |  |
| 2.5 | 5.67 |  |
| 2.7 | 6.21 |  |
| 2.9 | 6.88 |  |
| 3.1 | 7.49 |  |
| 3.33 | 8.03 |  |
| 3.5 | 8.85 |  |
| 3.7 | 9.67 |  |
| 3.9 | 10.23 |  |
| S | |  |
| 𝜹 | |  |
| a | |  |
| b | |  |
| c | |  |

Вывод:  
В ходе выполнения работы я разобрался с тем, как можно находить решения нелинейных уравнений, а также сделал их реализацию на языке программирования Java.