Міністерство освіти і науки України

Харківський національний університет імені В.Н. Каразіна

Факультет комп’ютерних наук

# ЛАБОРАТОРНА РОБОТА № 4

з дисципліни «Чисельні методи»

Тема: «Інтерполяція та аппроксімація»

Виконала:

студентка 3 курсу

групи КС-32

Дібцева Анна Миколаївна

Перевірила: доцент

Чуб Ольга Ігорівна

Харків – 2021

# ХОД РАБОТЫ

Целью данной работы является исследование алгоритмов интерполяции и аппроксимации, а так же программная реализация.

Входными данными является набор значений x и значений функции в этих точках, а так же, значение x, для которого нужно узнать значение функции.

Программа на основе введенных данных вычисляет значение коэффициентов при х и с их использованием вычисляет значение функции в точке. Для поиска коэффициентов при х в методе интерполяции используется полином Лагранжа, а в методе аппроксимации используется метод Гаусса, рассмотренный в лабораторной работе 1.

Программа скомпилирована в исполняемый файл.

Результат работы программы приведен ниже.

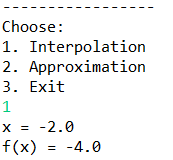


Рисунок 1 – Результат работы метода интерполяции

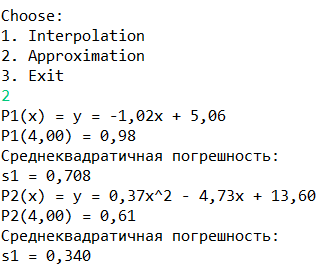


Рисунок 2 – Результат работы метода аппроксимации

# ВЫВОДЫ

Таким образом, в ходе выполнения данной лабораторной работы была написана программа, которая выполняет интерполяцию и аппроксимацию.

# ЛИСТИНГ ПРОГРАММЫ

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

printMenu();

}

private static void printMenu() {

Scanner scan = new Scanner(System.in);

while (true) {

System.out.println("-----------------\n"

+ "Choose:\n"

+ "1. Interpolation\n"

+ "2. Approximation\n"

+ "3. Exit");

int choise = scanInt(scan);

switch (choise) {

case 1:

interpolate();

break;

case 2:

approximate();

break;

case 3:

scan.close();

return;

default:

System.out.println("You didn't choose anything");

}

}

}

private static int scanInt(Scanner scan) {

int number = 0;

while (true) {

try {

number = scan.nextInt();

break;

} catch (Exception e) {

scan.next();

System.out.println("Wrong number. Enter another: ");

}

}

return number;

}

public static void interpolate() {

DataSet dataSet = new DataSet();

// dataSet.addPoint(-2, -4);

// dataSet.addPoint(-1, 0);

// dataSet.addPoint(1.2, 3.8);

// dataSet.addPoint(3.5, -2);

// dataSet.addPoint(5, 7);

dataSet.addPoint(-4, -2);

dataSet.addPoint(-3, 0);

dataSet.addPoint(-2, 1);

dataSet.addPoint(-1, -1);

dataSet.addPoint(0, -3);

double x = 0;

LagrangePolynomial lagrange = new LagrangePolynomial(dataSet);

double f\_x = lagrange.countFunc(x);

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

System.out.println("f(x) = " + f\_x);

}

public static void approximate () {

DataSet dataSet = new DataSet();

// dataSet.addPoint(3, 2.5);

// dataSet.addPoint(4, 1.1);

// dataSet.addPoint(5, -0.8);

// dataSet.addPoint(6, -1.9);

// dataSet.addPoint(7, -1,1);

dataSet.addPoint(-2, -4.8);

dataSet.addPoint(-1, 0);

dataSet.addPoint(0, 3.2);

dataSet.addPoint(1, 4);

dataSet.addPoint(2, 2.8);

double x = -2;

Approximation approximation = new Approximation(dataSet);

approximation.countFunc(x);

}

}

public class Point {

private double x;

private double y;

Point(){}

Point(double x, double y){

this.x = x;

this.y = y;

}

public double getX() {

return x;

}

public void setX(double x) {

this.x = x;

}

public double getY() {

return y;

}

public void setY(double y) {

this.y = y;

}

}

import java.util.ArrayList;

public class DataSet {

ArrayList<Point> points;

DataSet(){

points = new ArrayList();

}

DataSet(ArrayList<Point> points){

this.points = points;

}

public void addPoint(double x, double y) {

points.add(new Point(x, y));

}

public int removePoint(double x) {

for (int i = 0; i < points.size(); i++) {

if (points.get(i).getX() == x) {

points.remove(i);

return 0;

}

}

return -1;

}

public Point getPoint(int index) {

return points.get(index);

}

public int getSize() {

return points.size();

}

}

public class LagrangePolynomial {

private DataSet dataSet;

LagrangePolynomial() {}

LagrangePolynomial(DataSet dataSet) {

this.dataSet = dataSet;

}

public double countFunc(double x) {

dataSetSizeCheck();

return countPolynomial(x);

}

public double countFunc(double x, DataSet dataSet) {

this.dataSet = dataSet;

dataSetSizeCheck();

return countPolynomial(x);

}

private void dataSetSizeCheck() {

if (dataSet.getSize() == 0) {

System.out.println("You didn't inputed the number of points!");

System.exit(0);

}

}

private double countPolynomial(double x) {

double res = 0;

for (int i = 0; i < dataSet.getSize(); i++) {

Point pointI = dataSet.getPoint(i);

double numProduct = 1;

for (int j = 0; j < dataSet.getSize(); j++) {

if (j == i) continue;

Point pointJ = dataSet.getPoint(j);

numProduct \*= (x - pointJ.getX())/(pointI.getX() - pointJ.getX());

}

res += pointI.getY() \* numProduct;

}

return res;

}

}

import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;

public class GaussMethod {

public GaussMethod() { }

public static void gaussMethod(int dimension, double AMatrix[][], double BMatrix[]) {

//проверка, что определитель матрицы не равен 0

double determinant = determinant(dimension, AMatrix);

if (determinant == 0) {

System.out.println("The determinant = 0");

System.exit(-1);

}

reverse(AMatrix, BMatrix);

for(int i = 0; i < dimension; i++) {

//делим строку так, чтобы по диагонали стояла единица

double diagonalVal = AMatrix[i][i];

stringDivision(AMatrix[i], diagonalVal);

BMatrix[i] /= diagonalVal;

for(int j = 0; j < dimension; j++) {

if (i == j) continue;

double factor = AMatrix[j][i];

for(int n = 0; n < dimension; n++) {

AMatrix[j][n] -= AMatrix[i][n] \* factor;

}

BMatrix[j] -= BMatrix[i] \* factor;

}

}

}

private static void reverse(double AMatrix[][], double BMatrix[]) {

int dimension = AMatrix.length;

for (int i = 0; i < dimension; i++) {

if (AMatrix[i][i] == 0) {

int suitableIndex = -1;

for (int j = (i+1)%dimension; j < dimension; j = (j+1)) {

if (AMatrix[j][i] != 0) {

suitableIndex = j;

break;

}

}

if (suitableIndex < 0) {

System.out.println("Error in the equation system!");

System.exit(-1);

}

double[] tempRow = AMatrix[i].clone();

double tempB = BMatrix[i];

AMatrix[i] = AMatrix[suitableIndex].clone();

AMatrix[suitableIndex] = tempRow.clone();

BMatrix[i] = BMatrix[suitableIndex];

BMatrix[suitableIndex] = tempB;

}

}

}

private static double determinant(int dimension, double matrix[][]) {

double matrixDeterminant = 0;

if(dimension > 2) {

for(int i = 0; i < dimension; i++) {

List<double[]> submatrix = new ArrayList<>();

for(int j = 0; j < dimension; j++) {

if(j != i) {

submatrix.add((double[])Arrays.copyOfRange(matrix[j], 1, matrix[j].length));

}

}

matrixDeterminant += Math.pow(-1, i) \* matrix[i][0] \*

determinant(dimension-1, submatrix.toArray(new double[dimension-1][dimension-1]));

}

return matrixDeterminant;

}

else {

return matrix[0][0] \* matrix[1][1] - matrix[0][1] \* matrix[1][0];

}

}

private static void stringDivision(double matrix[], double divider) {

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

matrix[i] /= divider;

}

}

import static java.lang.Math.\*;

public class Approximation {

private DataSet dataSet;

private double aCoef;

private double bCoef;

private double cCoef;

Approximation() {}

Approximation(DataSet dataSet) {

this.dataSet = dataSet;

}

public void countFunc(double x) {

dataSetSizeCheck();

approximate(x);

}

public void countFunc(double x, DataSet dataSet) {

this.dataSet = dataSet;

dataSetSizeCheck();

approximate(x);

}

private void dataSetSizeCheck() {

if (dataSet.getSize() == 0) {

System.out.println("You didn't inputed the number of points!");

System.exit(0);

}

}

private void approximate (double x) {

double y\_x = firstDegreeApproximation(x);

System.out.printf("P1(%4.2f) = %4.2f\n", x, y\_x);

double error = countError(new double[] {aCoef, bCoef});

System.out.println("Среднеквадратичная погрешность: ");

System.out.printf("s1 = %4.3f\n", error);

y\_x = secondDegreeApproximation(x);

System.out.printf("P2(%4.2f) = %4.2f\n", x, y\_x);

error = countError(new double[] {aCoef, bCoef, cCoef});

System.out.println("Среднеквадратичная погрешность: ");

System.out.printf("s1 = %4.3f\n", error);

}

private double firstDegreeApproximation(double x) {

int dimension = 2;

double[][] AMatrix = new double[dimension][dimension];

double[] BMatrix = new double[dimension];

abFinding(AMatrix, BMatrix);

if (signum(bCoef) == -1)

System.out.printf("P1(x) = y = %4.2fx - %4.2f\n", aCoef, abs(bCoef));

else

System.out.printf("P1(x) = y = %4.2fx + %4.2f\n", aCoef, bCoef);

double y\_x = countEquation(x, new double[] {aCoef, bCoef});

return y\_x;

}

private double secondDegreeApproximation(double x) {

int dimension = 3;

double[][] AMatrix = new double[dimension][dimension];

double[] BMatrix = new double[dimension];

abcFinding(AMatrix, BMatrix);

System.out.printf("P2(x) = y = %4.2fx^2", aCoef);

if(signum(bCoef) == -1)

System.out.printf(" - %4.2fx", abs(bCoef));

else

System.out.printf(" + %4.2fx", bCoef);

if(signum(cCoef) == -1)

System.out.printf(" - %4.2f\n", abs(cCoef));

else

System.out.printf(" + %4.2f\n", cCoef);

double y\_x = countEquation(x, new double[] {aCoef, bCoef, cCoef});

return y\_x;

}

private void abFinding(double AMatrix[][], double BMatrix[]) {

coefficientsForABFinding(AMatrix, BMatrix);

GaussMethod.gaussMethod(2, AMatrix, BMatrix);

aCoef = BMatrix[0];

bCoef = BMatrix[1];

}

private void abcFinding(double AMatrix[][], double BMatrix[]) {

coefficientsForABСFinding(AMatrix, BMatrix);

GaussMethod.gaussMethod(3, AMatrix, BMatrix);

aCoef = BMatrix[0];

bCoef = BMatrix[1];

cCoef = BMatrix[2];

}

private void coefficientsForABFinding(double AMatrix[][], double BMatrix[]) {

AMatrix[0][0] = sumUpX();

AMatrix[0][1] = dataSet.getSize();

AMatrix[1][0] = sumUpXInPower(2);

AMatrix[1][1] = AMatrix[0][0];

BMatrix[0] = sumUpY();

BMatrix[1] = sumUpXY();

}

private void coefficientsForABСFinding(double AMatrix[][], double BMatrix[]) {

AMatrix[0][0] = sumUpXInPower(2);

AMatrix[0][1] = sumUpX();

AMatrix[0][2] = dataSet.getSize();

BMatrix[0] = sumUpY();

AMatrix[1][0] = sumUpXInPower(3);

AMatrix[1][1] = AMatrix[0][0];

AMatrix[1][2] = AMatrix[0][1];

BMatrix[1] = sumUpXY();

AMatrix[2][0] = sumUpXInPower(4);

AMatrix[2][1] = AMatrix[1][0];

AMatrix[2][2] = AMatrix[1][1];

BMatrix[2] = 0;

for (int i = 0; i < dataSet.getSize(); i++) {

Point point = dataSet.getPoint(i);

BMatrix[2] += pow(point.getX(), 2) \* point.getY();

}

// System.out.printf("Sum x^2y = %4.2f\n", BMatrix[2]);

}

private double sumUpX() {

double res = 0;

for (int i = 0; i < dataSet.getSize(); i++) {

res += dataSet.getPoint(i).getX();

}

// System.out.println("Sum xI = "+res);

return res;

}

private double sumUpXInPower(int power) {

double res = 0;

for (int i = 0; i < dataSet.getSize(); i++) {

res += pow(dataSet.getPoint(i).getX(), power);

}

// System.out.println("Sum xI^"+power+" = "+res);

return res;

}

private double sumUpY() {

double res = 0;

for (int i = 0; i < dataSet.getSize(); i++) {

res += dataSet.getPoint(i).getY();

}

// System.out.println("Sum yI = "+res);

return res;

}

private double sumUpXY() {

double res = 0;

for (int i = 0; i < dataSet.getSize(); i++) {

Point point = dataSet.getPoint(i);

res += point.getX() \* point.getY();

}

// System.out.println("Sum xy = "+res);

return res;

}

private double countError(double coefficients[]) {

double res = 0;

double sum = 0;

double length = dataSet.getSize();

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

double p1\_x = countEquation(dataSet.getPoint(i).getX(), coefficients);

sum += pow((p1\_x - dataSet.getPoint(i).getY()),2);

}

res = sqrt(sum / length);

return res;

}

private double countEquation(double x, double coefficients[]) {

double equationDegree = coefficients.length;

double f\_x = 0;

for (int i = 0; i < equationDegree; i++)

f\_x += coefficients[i] \* pow(x, equationDegree-i-1);

return f\_x;

}

}