Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**Лабораторна робота № 1**

Виконав студент групи ІО-01

Редько Олександр

# Завдання

Задано дифузійний монотонний потік. Знайти параметри і відтворити потік.

# Програма

## Лістинг коду

**package** lab1;

**import** java.io.BufferedWriter;

**import** java.io.FileWriter;

**import** java.io.IOException;

**import** java.io.PrintWriter;

**import** java.util.Random;

**class** Generator {

**private** **int** n;

**private** **double** [] arrayT;

Generator(**int** n){

**this**.n = n;

}

/\*\*

\* Зберігає дані в файл

\*

\* **@param** name String ім'я файлу, що зберігається

\* **@throws** IOException можливе виключення

\*/

**void** save(String name) **throws** IOException{

PrintWriter pw = **new** PrintWriter(**new** BufferedWriter(

**new** FileWriter(name)));

**for**(**double** element: arrayT){

pw.println(element);

}

pw.close();

}

**int** getN() {

**return** n;

}

**double** [] getArrayT() {

**return** arrayT;

}

**void** setArrayT(**double** [] arrayT) {

**this**.arrayT = arrayT;

}

**void** generateArrayT (**double** tau) {

DmIntegral dmi = **new** DmIntegral(tau);

**double**[] arrayT = **new** **double**[n];

**double**[] randomJ = **new** **double**[arrayT.length];

Random R = **new** Random();

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

randomJ[i] = R.nextDouble();

arrayT[i] = dmi.integrate(randomJ[i]);

}

setArrayT(arrayT);

}

}

package lab1;

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

import java.math.BigDecimal;

import java.util.Arrays;

class Analyzator {

private int n;

private InputData inputData;

private static final int maxIterations = 500000;

private static final double DEFAULT\_XI = 0.01;

Analyzator(int n) {

this.n = n;

inputData = new InputData(n);

}

/\*\*

\* Завантажує дані з файлу

\*

\* @param name ім'я завантажуваного файлу

\* @throws IOException можливе виключення

\*/

void load (String name) throws IOException {

BufferedReader br = new BufferedReader(new FileReader(name));

int i = 0;

while(br.ready()){

inputData.write(i, Double.parseDouble(br.readLine()));

i++;

}

br.close();

}

double maximumLikelihood() {

double[] array;

array = inputData.getData();

double L = .0,

xi = DEFAULT\_XI,

h = .05;

// tau = 3/2 \* xi; //nu = 1

double[][] lArray = new double[maxIterations][2];

Function fct;

int j = 0;

while(true){

L = 0;

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

fct = new Function(xi, 1);

L += Math.log(fct.f(array[i]));

}

lArray[j][0] = L;

lArray[j][1] = xi;

xi += h;

j++;

if(j == maxIterations) {

break;

}

}

xi = DEFAULT\_XI;

for(int k = 0; k <= (int)maxValue(lArray)[1]; k++) {

xi += h;

}

return (xi)\*3/2;//maxValue(lArray)[0]\*3/2; //tau

}

private double[] maxValue(double[][] lArray) {

double[] array = new double[2];

int index = 0;

double max = lArray[0][0];

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

if (lArray[i][0] > max) {

max = lArray[i][0];

index = i;

}

array[0] = max;

array[1] = index;

return array;

}

public double chi2(double[] array1, double[] array2) {

double ch2 = 0;

Arrays.sort(array1);

Arrays.sort(array2);

double [][] M = arrayM(array1, array2);

int n = inputData.getData().length;

for(int i = 0; i < 10; i++) {

ch2 += n \*

(1/(M[0][i] + M[1][i])) \* (M[0][i]/n - M[1][i]/n);

}

return ch2;

}

private double[] separate(double[] sample){

int l = 10;

double[] sep = new double[l + 1];

sep[0] = sample[0];

for(int i = 1; i < l; i++){

sep[i] = (sample[1 + 10 \* i] + sample[10 \* i])/2;

}

sep[l] = sample[sample.length - 1];

return sep;

}

private double[][] arrayM(double[] sample, double[] sample2){

double[] separator = separate(sample);

double[] m1 = new double[separator.length - 1];

for(int i = 0; i < m1.length; i++){

for(int j = 0; j < sample.length; j++){

if(sample[j] <= separator[i + 1] && sample[j] >= separator[i]){

m1[i]++;

}

}

}

double[] m2 = new double[separator.length - 1];

for(int i = 0; i < m2.length; i++){

for(int j = 0; j < sample2.length; j++){

if(sample2[j] <= separator[i + 1] && sample2[j] >= separator[i]){

m2[i]++;

}

}

}

double[][] M = new double[m1.length][2];

M[0] = m1;

M[1] = m2;

return M;

}

public InputData getInputData() {

return inputData;

}

public void setInputData(InputData inputData) {

this.inputData = inputData;

}

public double av(double[] array) {

double s = 0;

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

s += array[i];

}

return s/array.length;

}

}

**package** lab1;

**import** java.io.File;

**import** java.io.IOException;

**import** java.util.Scanner;

**class** Main {

**private** **static** **final** File *FILE* = **new** File("data.txt");

**private** **static** **final** **int** *N* = 100;

**public** **static** **void** main(String[] args) {

Generator g = **new** Generator(*N*);

System.*out*.print("Enter tau: ");

Scanner sc = **new** Scanner(System.*in*);

**double** tau = sc.nextDouble();

g.generateArrayT(tau);

*showArray*(g.getArrayT());

**try** {

g.save(*FILE*.getName());

} **catch** (IOException e) {

e.printStackTrace();

}

Analyzator an = **new** Analyzator(*N*);

**try** {

an.load(*FILE*.getName());

} **catch** (IOException e) {

e.printStackTrace();

}

System.*out*.println();

tau = an.maximumLikelihood();

System.*out*.print("tau: " + tau);

g.generateArrayT(tau);

System.*out*.println();

*showArray*(g.getArrayT());

System.*out*.println();

System.*out*.println(an.chi2(an.getInputData().getData(), g.getArrayT()));

}

**static** **void** showArray(**double**[] array){

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

System.*out*.printf("%9.2f", array[i]);

}

}

**public** **static** File getFile() {

**return** *FILE*;

}

**public** **static** **int** getN() {

**return** *N*;

}

}

**package** lab1;

**class** Function {

**private** **double** nu;

**private** **double** xi;

Function(**double** xi, **double** nu){

**this**.xi = xi;

**this**.nu = nu;

}

**double** f (**double** t){

**return** (t + xi)/(2 \* nu \* t \* Math.*sqrt*(2 \* Math.*PI* \* xi \* t)) \*

Math.*exp*(-(t - xi) \* (t - xi)/(2 \* nu \* nu \* xi \* t));

}

}