Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота №3

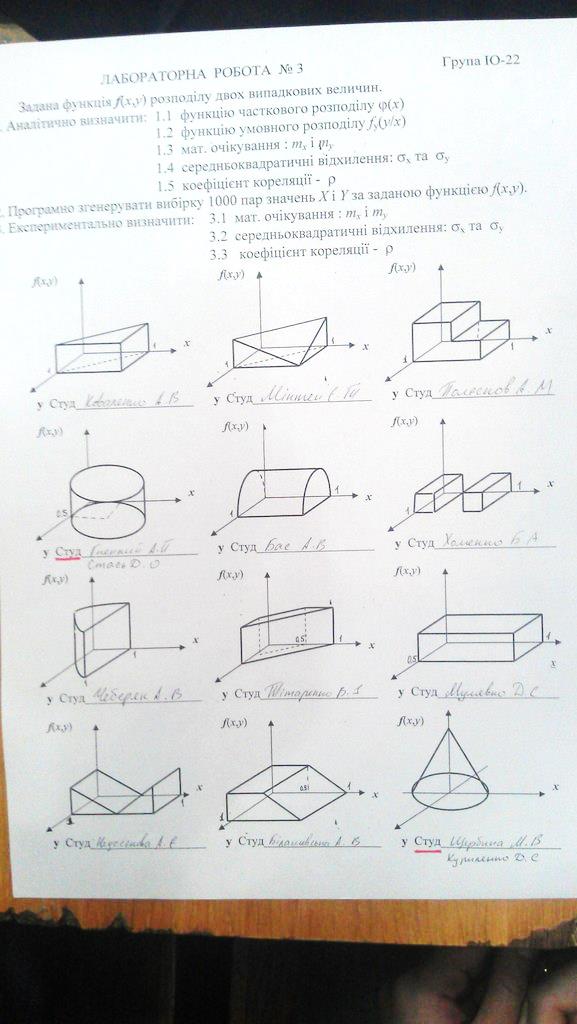
з предмету: «Теорія ймовірності»

Виконав:

студент групи ІО-32

Попенко Руслан

Київ – 2014



Фігура – круговий конус з радіусом R=1 та об’ємом V=1.

Знайдемо висоту:

1. функція часткового розподілу:

Загальна формула:

1. умовна ймовірність

Загальна формула:

1. Математичне очікування

Загальні формули:

1. Середньо квадратичні відхилення

Загальні формули:

1. Коефіцієнт кореляції

Загальні формули:

=0

import java.util.Random;

import java.lang.Math;

interface Func {

double f(double x, double y);

}

class X implements Func {

@Override

public double f(double x, double y) {

return Main.w(x, y);

}

}

class Y implements Func {

@Override

public double f(double x, double y) {

return Main.z(x);

}

}

public class Main {

static final double h = 3 / Math.PI;

static int n = 1000;// кількість елементів

static double[] x;// значення величини Х

static double[] y;// значення величини Y

static double Mx;// математичне очікування величини Х

static double My;// математичне очікування величини Y

static double Dx;// дисперсія величини Х

static double Dy;// дисперсія величини Y

static double Zx;// середнє квадратичне відхилення величини Х

static double Zy;// середнє квадратичне відхилення величини Y

static double cov;// коваріація

static double R;// корелляція

static Random random = new Random();

static double f(double x, double y) {

return h \* (1 - Math.sqrt(x \* x + y \* y));

}

/\*\*

\* функція часткового розподілу У

\*/

static double z(double y) {

return h

\* (Math.sqrt(1 - y \* y) - y \* y

\* Math.log((1 + Math.sqrt(1 - y \* y)) / Math.abs(y)));

}

/\*\*

\* функція умовного розподілу х

\*/

static double w(double x, double y) {

return f(x, y) / z(y);

}

/\*\*

\* @param s

\* @param res

\* @return

\*/

static double solution(double[][] s, double res) {

double result = 1;

for (int i = 0; i < s[0].length; i++)

if (s[0][i] > res) {

result = s[1][i];

break;

}

return result;

}

/\*\*

\* знаходить інтеграли на проміжку [a;b]s

\*

\* @param f

\* - функція

\* @param y

\* - значення другого агрументу функції

\* @param n

\* - кількість точок розбиття

\* @param a

\* - ліва межа відрізку

\* @param b

\* - права межа відрізку

\* @return

\*/

static double[][] integrate(Func f, double y, int n, double a, double b) {

double[][] result = new double[2][n + 1];

double step = (b - a) / n;

double step2 = step / 2;

double x = a;

result[0][0] = 0;

result[1][0] = a;

for (int i = 1; i < result[0].length; i++) {

result[0][i] = result[0][i - 1] + f.f(x + step2, y) \* step;

result[1][i] = x + step2;

x = x + step;

}

return result;

}

/\*\*

\* математичне очікування

\*

\* @param x

\* - послідовність чисел

\* @return

\*/

static double mathematicalExpectation(double[] x) {

double summa = 0;

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

summa += x[i];

}

double result = summa / x.length;

return result;

}

/\*\*

\* дисперсія

\*

\* @param x

\* - послідовність чисел

\* @return

\*/

static double dispersion(double[] x) {

double m = mathematicalExpectation(x);

double summa = 0;

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

summa += (x[i] - m) \* (x[i] - m);

}

double result = summa / x.length;

return result;

}

/\*\*

\* коваріація

\*/

static double covariance(double[] x, double[] y) {

double result = 0;

double mx = mathematicalExpectation(x);

double my = mathematicalExpectation(y);

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

result += (x[i] - mx) \* (y[i] - my);

}

result = result / y.length;

return result;

}

/\*\*

\* випадковий знак

\*/

static int randomSign() {

int sing = random.nextInt(2);

if (sing == 0)

sing = -1;

return sing;

}

public static void main(String[] args) {

x = new double[n];

y = new double[n];

Y yF = new Y();

X xF = new X();

int countPoint = 100;

double[][] sX;

double[][] sY = integrate(yF, 0, countPoint, 0, 1);

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

double rr = random.nextDouble();

double yi = rr \* sY[0][sY[0].length - 1];

y[i] = randomSign() \* solution(sY, yi);

sX = integrate(xF, y[i], countPoint, 0, 1);

double xi = random.nextDouble() \* sX[0][sX[0].length - 1];

x[i] = randomSign() \* solution(sX, xi);

}

Mx = mathematicalExpectation(x);

My = mathematicalExpectation(y);

Dx = dispersion(x);

Dy = dispersion(y);

Zx = Math.sqrt(Dx);

Zy = Math.sqrt(Dy);

cov = covariance(x, y);

R = cov / Zx / Zy;

System.out.println("Результати:");

System.out.println("математичне очікування x = " + Mx);

System.out.println("математичне очікування y = " + My);

System.out.println("середнє квадватичне відхилення x = " + Zx);

System.out.println("середнє квадватичне відхилення y= " + Zy);

System.out.println("коваріація = " + cov);

System.out.println("колеряція = " + R);

}

}

**Результати:**

математичне очікування x = 0.003404999999999986

математичне очікування y = 0.008490000000000011

середнє квадватичне відхилення x = 0.3654073493719031

середнє квадватичне відхилення y= 0.3769829172522276

коваріація = -0.00194093345

колеряція = -0.014090022799669895