*Національний технічний університет України*

*«Київський політехнічний інститут»*

***Факультет інформатики та обчислювальної техніки***

Лабораторна робота №3

*з курсу "****Теорія ймовірностей****"*

**Тема: “Системи випадкових величин”**

***Виконав:***

*Долинний О.В.*

***Група*** *ІО-31*

***Перевірив:***

*Марковський О.П.*

***Київ - 2014р.***

**Клас Main**

**package** lab\_3;

**public** **class** Main {

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

Tests t = **new** Tests(0.5, 1);

t.generateNumbers(1000);

System.***out***.println("Математическое ожидание х = " + t.mathX());

System.***out***.println("Математическое ожидание у = " +t.mathY());

System.***out***.println("Дисперсия х = " +t.dispX());

System.***out***.println("Дисперсия у = " +t.dispY());

System.***out***.println("Ковариация = " +t.cov());

System.***out***.println("Коеффициент кореляции = " +t.corelation());

}

}

**Клас Shape**

**package** lab\_3;

**import** java.util.Random;

**public** **class** Shape {

**private** **double** a;

**private** **double** b;

**public** **static** **double** *epsilon* = 0.000001;

**public** Shape(**double** a, **double** b) {

**this**.a = a;

**this**.b = b;

}

**public** Vector generateVector(){

**double** x = generateX();

**double** y = generateY(x);

**return** **new** Vector(x,y);

}

**private** **double** generateX(){

Random rand = **new** Random();

**double** r = rand.nextDouble();

**double** square = 0;

**double** step = 0.000001;

**double** xResult = 0.0;

**double** trapeciaSquare = 0;

**while**(square < r-*epsilon*){

trapeciaSquare = ((functionPhiX(xResult) + functionPhiX(xResult + step))/2)\*step;

square += trapeciaSquare;

xResult += step;

**if**(xResult > a){

System.***err***.println("X is grater than " + a);

System.*exit*(1);

}

}

**return** xResult;

}

**private** **double** generateY(**double** x){

Random rand = **new** Random();

**double** r = rand.nextDouble();

**double** square = 0;

**double** step = 0.000001;

**double** yResult = 0.0;

**double** trapeciaSquare = 0;

**while**(square < r-*epsilon*){

trapeciaSquare = (functionF\_if\_X(yResult, x) + functionF\_if\_X(yResult+step, x))/2\*step;

square += trapeciaSquare;

yResult += step;

**if**(yResult > b){

System.***err***.println("Y is grater than " + b);

System.*exit*(1);

}

}

**return** yResult;

}

**private** **double** functionPhiX(**double** x){

**return** 6\*x\*(a-x)/(a\*a\*a);

}

**private** **double** main\_function(**double** x, **double** y){

**if**(x >= 0 && x <= a/2 && y >= b\*x/a && y <= -b\*x/a + b){

**return** functionOne(x, y); }

**if**(y >= 0 && y <= b/2 && x <= -a\*y/b + a && x >= a\*y/b){

**return** functionTwo(x,y);

}

**if**(x >= a/2 && x <= a && y <= b\*x/a && y >= -b\*x/a +b){

**return** functionThree(x,y);

}

**return** functionFour(x,y);

}

**private** **double** functionOne(**double** x, **double** y){

**return** 6\*x/(a\*a\*b);

}

**private** **double** functionTwo(**double** x, **double** y){

**return** 6\*y/(a\*b\*b);

}

**private** **double** functionThree(**double** x, **double** y){

**return** 6\*(a-x)/(a\*a\*b);

}

**private** **double** functionFour(**double** x, **double** y){

**return** 6\*(b-y)/(a\*b\*b);

}

**private** **double** functionF\_if\_X(**double** y, **double** x){

**double** result = main\_function(x, y)/functionPhiX(x);

**return** result;

}

}

**Клас Tests**

**package** lab\_3;

**import** java.util.ArrayList;

**public** **class** Tests {

**private** Shape shape;

**private** ArrayList<Vector> numbers;

**private** **double** mathX;

**private** **double** mathY;

**private** **double** dispersionX;

**private** **double** dispersionY;

**private** **double** cov;

**public** Tests(**double** a, **double** b){

shape = **new** Shape(a, b);

numbers = **new** ArrayList<Vector>();

}

**public** **void** generateNumbers(**int** count){

**for**(**int** i = 0; i < count; i++){

numbers.add(shape.generateVector());

}

}

**public** **double** mathX(){

mathX = 0;

**for**(Vector v : numbers){

mathX += v.getX();

}

mathX /= numbers.size();

**return** mathX;

}

**public** **double** mathY(){

mathY = 0;

**for**(Vector v : numbers){

mathY += v.getY();

}

mathY /= numbers.size();

**return** mathY;

}

**public** **double** dispX(){

dispersionX = 0;

**for**(Vector v: numbers){

dispersionX += (v.getX() - mathX)\*(v.getX() - mathX);

}

dispersionX /= numbers.size();

**return** dispersionX;

}

**public** **double** dispY(){

dispersionY = 0;

**for**(Vector v: numbers){

dispersionY += (v.getY() - mathY)\*(v.getY() - mathY);

}

dispersionY /= numbers.size();

**return** dispersionY;

}

**public** **double** cov(){

cov = 0;

**for**(Vector v : numbers){

cov += (v.getX() - mathX) \* (v.getY() - mathY);

}

cov /= numbers.size();

**return** cov;

}

**public** **double** corelation(){

**return** cov/(Math.*sqrt*(dispersionX)\*Math.*sqrt*(dispersionY));

}

}

**Клас Vector**

**package** lab\_3;

**public** **class** Vector {

**private** **double** x;

**private** **double** 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;

}

**public** Vector(**double** x, **double** y) {

**super**();

**this**.x = x;

**this**.y = y;

}

}