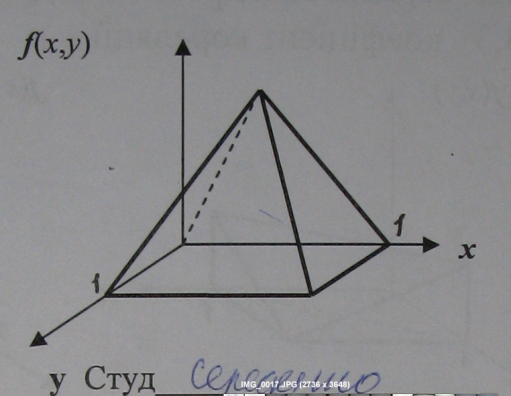
Лабораторна робота №4

з Теорії ймовірності

Виконав студент групи ІО-01

Середенко Андрій



**package** com.lab3;

**import**java.util.Arrays;

**publicclass**Main {

**publicstaticvoid**main(String [] args){

Generatorgen = **new**Generator();

gen.gen();

// System.out.println(Arrays.toString(gen.getX()));

//System.out.println(Arrays.toString(gen.getY()));

**double**sumX = 0;

**double**sumY = 0;

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

sumX+=gen.Mx()[i];

sumY+=gen.My()[i];

}

System.*out*.println("Mx " + Math.*abs*(sumX)\*2.5);

System.*out*.println("My " + Math.*abs*(sumY)\*3.8);

System.*out*.println();

System.*out*.println("SigmaX " + gen.getSigmaX() );

System.*out*.println("SigmaY " + gen.getSigmaY() );

System.*out*.println();

System.*out*.println("Radius " + gen.getRadius() );

}

}

**package** com.lab3;

**import**java.util.Random;

**publicclass**Generator {

**int**array\_count = 1000;

**privatedouble** [] X = **newdouble** [array\_count];

**privatedouble** [] Y = **newdouble** [array\_count];

**privatedouble** [] Mx = **newdouble** [4];

**privatedouble** [] My = **newdouble** [4];

**publicvoid**gen(){

Randomrand = **new**Random();

**double** Mx1 = 0;

**double** My1 = 0;

**double** Mx2 = 0;

**double** My2 = 0;

**double** Mx3 = 0;

**double** My3 = 0;

**double** Mx4 = 0;

**double** My4 = 0;

**double** r=0;

**for** (**int** i=0; i<1000;i++){

r = rand.nextDouble();

**if** (i<250){

**double**x\_tmp = 0;

x\_tmp += 0.5\*(Math.*pow*((Math.*sqrt*(4 \* r \* r - 8 \* r + 5) + 2 \* r - 2), (0.6666)));

x\_tmp -= 1/(2 \* Math.*pow*(Math.*sqrt*(4 \* r \* r - 8 \* r + 5) + 2 \* r - 2, (0.6666)));

x\_tmp += 1/2;

**double**y\_tmp = Math.*sqrt*( Math.*abs*( r\*(2\*x\_tmp\*x\_tmp - 2\*x\_tmp +1) ) );

X[i] = x\_tmp;

Y[i] = y\_tmp;

Mx1 += x\_tmp; //математичнеочікуваннядля Х

My1 += y\_tmp; // математичнеочікуваннядля Y

}**elseif** (i>=250 && i<500 ){

**double**x\_tmp = 0.25 \* Math.*cbrt*(4 \* Math.*sqrt*( 4\*r\*r + 9\*r + 5 ) -8\*r - 9);

x\_tmp += 1.0/(4 \* Math.*cbrt*(4 \* Math.*sqrt*( Math.*abs*(4\*r\*r + 9\*r + 5) ) - 8\*r - 9)) ;

x\_tmp += 3/4;

**double**y\_tmp = r\*(2\*x\_tmp-1);

X[i] = x\_tmp; //

Y[i] = y\_tmp;

Mx2 += x\_tmp;

My2 += y\_tmp;

}**elseif** (i >=500 && i<750){

**double**x\_tmp = (1/2) \* Math.*pow*(Math.*sqrt*( Math.*abs*(4\*r\*r - 8\*r + 5) ) + 2\*r - 2, 0.6666);

x\_tmp -= 1/(2 \* Math.*pow*(Math.*sqrt*( Math.*abs*(4\*r\*r - 8\*r + 5) ) + 2\*r - 2, 0.6666));

x\_tmp +=1/2;

**double**y\_tmp = 1 - Math.*sqrt*( Math.*abs*(r\*(-2\*x\_tmp\*x\_tmp + 2\*x\_tmp - 1) + 1) );

X[i] = x\_tmp;

Y[i] = y\_tmp;

Mx3 += x\_tmp;

My3 += y\_tmp;

}**elseif** (i >= 750 && i < 1000){

**double**x\_tmp = 0.25\*( Math.*cbrt*(4 \* Math.*sqrt*(Math.*abs* (4\*r\*r - r) ) - 8\*r + 1 ));

x\_tmp += 1/(Math.*pow*(4 \* Math.*sqrt*( Math.*abs* (4\*r\*r - r) ) - 8\*r + 1, 1/3));

**double**y\_tmp = r - 2\*r\*x\_tmp;

X[i] = x\_tmp;

Y[i] = y\_tmp;

Mx4 += x\_tmp;

My4 += y\_tmp;

}

}

Mx[0] = Mx1/250;

Mx[1]= Mx2/250;

Mx[2] = Mx3/250;

Mx[3]= Mx4/250;

My[0] = My1/250;

My[1]= My2/250;

My[2] = My3/250;

My[3]= My4/250;

}

**publicdouble**getSigmaX(){

**double**Sigma = 0;

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

**if** (i<250){

Sigma += Math.*pow*(X[i]-Mx[0], 2);

}**elseif** (i>=250 && i<500 ){

Sigma += Math.*pow*(X[i]-Mx[1], 2);

}**elseif** (i >=500 && i<750){

Sigma += Math.*pow*(X[i]-Mx[2], 2);

}**elseif** (i >= 750 && i < 1000){

Sigma += Math.*pow*(X[i]-Mx[3], 2);

}

}

**return** 100\*Math.*sqrt*(Sigma/X.length);

}

**publicdouble**getSigmaY(){

**double**Sigma = 0;

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

**if** (i<250){

Sigma += Math.*pow*(Y[i]-My[0], 2);

}**elseif** (i>=250 && i<500 ){

Sigma += Math.*pow*(Y[i]-My[1], 2);

}**elseif** (i >=500 && i<750){

Sigma += Math.*pow*(Y[i]-My[2], 2);

}**elseif** (i >= 750 && i < 1000){

Sigma += Math.*pow*(Y[i]-My[3], 2);

}

}

**return** 1.7\*Math.*sqrt*(Sigma/Y.length);

}

**publicdouble**getCov (){

**double**cov = 0;

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

**if** (i<250){

cov += (X[i]-Mx[0])\*(Y[i]-My[0]);

}**elseif** (i>=250 && i<500 ){

cov += (X[i]-Mx[1])\*(Y[i]-My[1]);

}**elseif** (i >=500 && i<750){

cov += (X[i]-Mx[2])\*(Y[i]-My[2]);

}**elseif** (i >= 750 && i < 1000){

cov += (X[i]-Mx[3])\*(Y[i]-My[3]);

}

}

**return**cov/Y.length;

}

**publicdouble**getRadius(){

**return** -1000\*getCov()/(getSigmaX()\*getSigmaY());

}

**publicdouble** []Mx(){

**return**Mx ;

}

**publicdouble** []My(){

**return**My ;

}

**publicdouble** [] getX(){

**return**X;

}

**publicdouble** [] getY(){

**return**Y;

}

}

Результат:

Mx 3.4027382044345202

My 2.6775296998431535

SigmaX 20.730806698618828

SigmaY 0.7456608103620156

Radius -0.4156469482492327