Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Кафедра ПІ

Лабораторна робота №3

з дисципліни: «Теорія ймовірностей, математична статистика та емпіричні методи програмної інженерії»

на тему «Закони розподілу»

Виконали:

ст. групи ПИ-15-5 Пігнастий М.О.

Майборода В.А.

Шопинський М.В.

Перевірив: Дрюк О.Д.

Харків 2016

1. Мета роботи

Ознайомитись з існуючими законами розподілу, навчитись будувати графічне відображення наведених вхідних даних, навчитись оцінювати відповідність наведених вхідних даних існуючим законам розподілу, спираючись на їх графічне відображення.

2. Хід роботи

2.1 Розрахунок значень (скріншоти коду)

2.1.1 Рівномірний розподіл

Random r = new Random();

double Uniform(double a, double b)

{

return a + r.NextDouble() \* (b - a);

}

2.1.2 Експоненційний розподіл

double Exponential(double lambda)

{

return -Math.Log(Uniform(0, 1)) / lambda;

}

2.1.3 Нормальний розподіл метода Бокса-Мюлера

double BoxMuller(bool isFirst, double mean, double dev)

{

double u, v, s;

do {

u = Uniform(-1, 1);

v = Uniform(-1, 1);

s = u \* u + v \* v;

} while (s >= 1);

double r = Math.Sqrt(-2 \* Math.Log(s) / s);

if (isFirst)

return r \* v \* dev + mean;

else

return r \* u \* dev + mean;

}

2.1.4 Нормальний розподіл (зиккурат-алгоритм)

#region Ziggurat

static double[] stairWidth = new double[257], stairHeight = new double[256];

const double x1 = 3.6541528853610088;

const double A = 4.92867323399e-3;

void setupNormalTables()

{

// coordinates of the implicit rectangle in base layer

stairHeight[0] = Math.Exp(-0.5 \* x1 \* x1);

stairWidth[0] = A / stairHeight[0];

// implicit value for the top layer

stairWidth[256] = 0;

for (int i = 1; i <= 255; ++i)

{

// such x\_i that f(x\_i) = y\_{i-1}

stairWidth[i] = Math.Sqrt(-2 \* Math.Log(stairHeight[i - 1]));

stairHeight[i] = stairHeight[i - 1] + A / stairWidth[i];

}

}

double NormalZiggurat()

{

int iter = 0;

do

{

int B = r.Next();

int stairId = B & 255;

double x = Uniform(0, stairWidth[stairId]);

if (x < stairWidth[stairId + 1])

return (B > 0) ? x : -x;

if (stairId == 0)

{

double z = -1;

double y;

if (z > 0)

{

x = Exponential(x1);

z -= 0.5 \* x \* x;

}

if (z <= 0)

{

do

{

x = Exponential(x1);

y = Exponential(1);

z = y - 0.5 \* x \* x;

} while (z <= 0);

}

x += x1;

return (B > 0) ? x : -x;

}

if (Uniform(stairHeight[stairId - 1], stairHeight[stairId]) < Math.Exp(-.5 \* x \* x))

return (B > 0) ? x : -x;

} while (++iter <= 1e9);

return -1;

}

double Normal(double mu, double sigma)

{

return mu + NormalZiggurat() \* sigma;

}

#endregion