Учреждение образования

Белорусский государственный университет информатики

и радиоэлектроники

Факультет компьютерных технологий ИИТ

Лабораторная работа №2

Моделирование

Формирование случайных чисел с заданным распределением

Выполнил: студент гр.180511

Жданко М.И.

Гриневич А.А..

Проверил: преподаватель Лашкевич Е.М.

Минск, 2014

Цель работы:  
 1) Разработать программный датчик последовательности случайных чисел с заданным законом распределения:

− равномерное распределение в заданном интервале;

− нормальное распределение;

− показательное распределение;

− гамма-распределение;

− треугольное-распределение;

− распределение Симпсона и т.д.

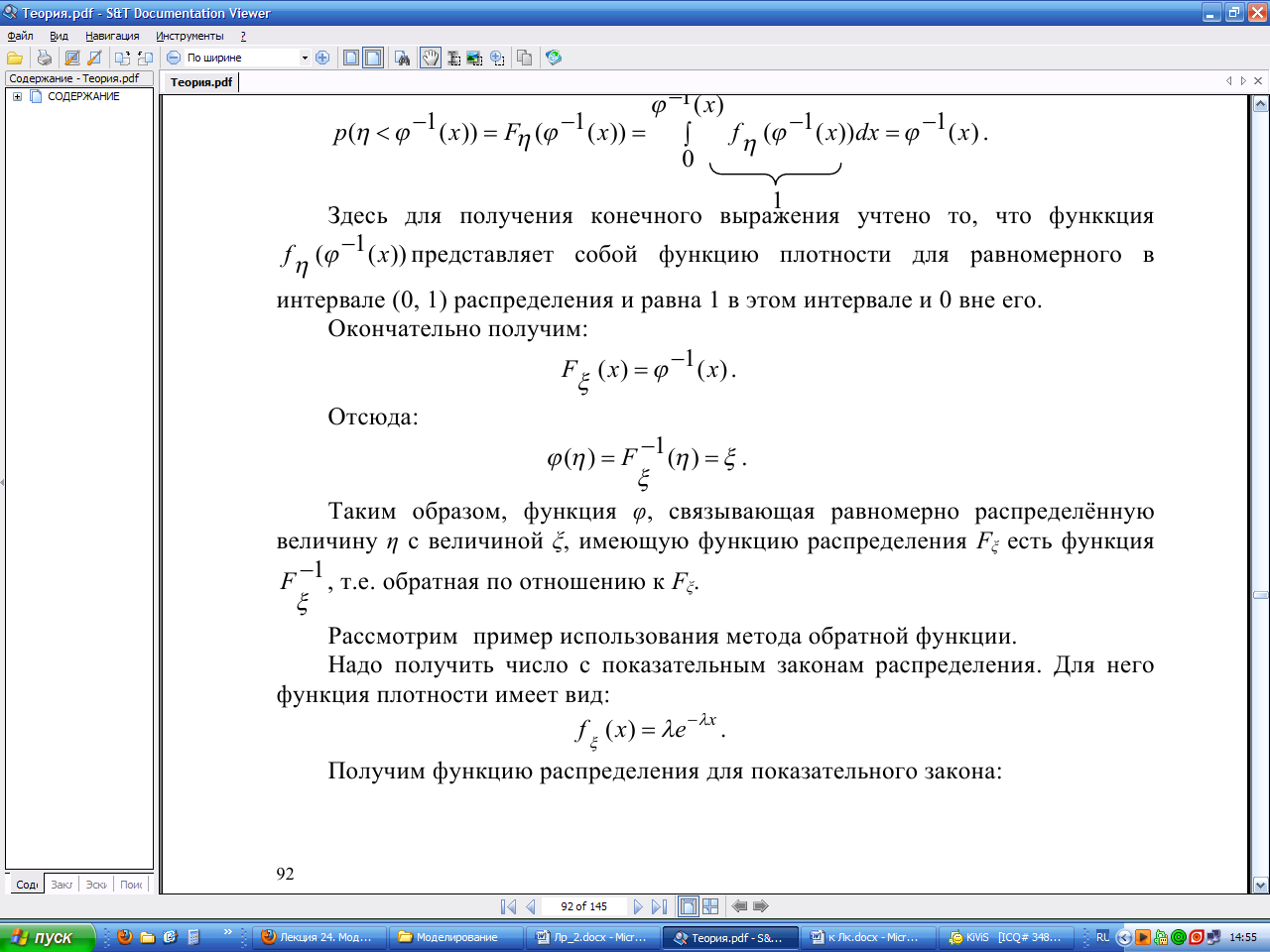
2) По полученной выборке построить гистограмму, определить значения математического ожидания , дисперсии и среднего квадратического отклонения;

**Теория.**

Случайные числа с заданным распределением, как правило, формируются в результате преобразования последовательность РРСЧ, принимающих значения из диапазона от 0 до 1.

Методы, позволяющие имитировать непрерывные и дискретные вероятностные распределения:

– метод обратных функций,

**

– метод исключения,

– метод композиции

– и т.д.

Рассмотрим содержание двух наиболее распространенных на практике процедур.

*1. Для непрерывных распределений*

*Реализуется метод обратных функций*.

Имитации подлежит непрерывная случайная величина *x*, которая описывается плотностью распределения. Плотность распределения *f*(*x*) связана с функцией распределения *F*(*x*) соотношением



Требуется разработать машинный алгоритм. Для этого последовательно выполняются следующие операции.

1. Осуществляется переход от плотности распределения *f*(*x*) к функции распределения *F*(*x*) на основе соотношения:



2. Составляется исходное уравнение.

F (*x*) = *r*

*r* — число, генерируемое ГСЧрр в интервале от 0 до 1

3. Данное уравнение решается относительно *x*:

*x* = F–1(*r*) - искомый машинный алгоритм

*x* – генерируемая в итоге случайная величина с заданным распределением

F–1() - функция, обратная по отношению к функции F().

*2. Для дискретных распределений*

*Реализуется метод обратных функций.*

Имитации подлежит дискретная случайная величина , которая описывается рядом распределения

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *xi* | *x1* | *x2* | *…* | *xn* |
| *pi* | *p1* | *p2* | *…* | *pn* |

где 

Для имитации значения дискретной случайной величины *x* используется СЧрр *r* на интервале [0,1].

Очевидно, что в этом случае

P(0 ≤ *r* < p1) = p1;

P(p1≤ *r* < p1 + p2 ) = p2;

P(p1+ p2 ≤ *r* < p1 + p2 + p3) = p3;

. . .

P(p1+ p2 + ...+ pn-1 ≤ *r* < p1+ p2 + ...+ pn ) = pn;

Машинный алгоритм, имитирующий значение дискретной случайной величины *x*:

1. Берется случайное СЧрр *r*.

2. Проверяется логическое условие:



где k принимает целочисленные значения, возрастающие от 1 до n.

3. При некотором k условие начинает выполняться. Это определяет имитируемое значение xk - дискретной случайной величины X.

1. Имитация равномерного распределения

Равномерное распределение непрерывной случайной величины  описывается плотностью распределения



Математическое ожидание и дисперсия случайной величины  определяется соотношениями

 и 

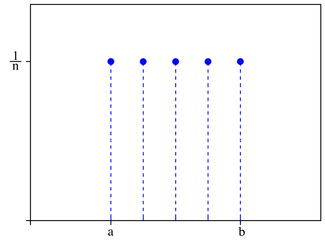
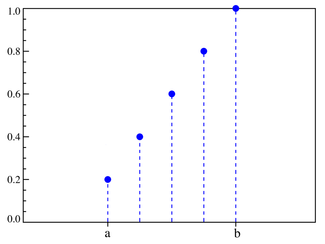
*Используем метод обратных функций* для имитации равномерного распределения:



 **-** искомый машинный алгоритм.

Дискретное равномерное распределение

Функция вероятности Функция распределения

[](http://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB:DUniform_distribution_PDF.p) [](http://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB:DUniform_distribution_CDF.p)

n=5

**Параметры**

a \in (...,-2,-1,0,1,2,...)\,  
b \in (...,-2,-1,0,1,2,...)\,  
n=b-a+1\,

2. Имитация распределения Симпсона (треугольное распределение)

Распределение Симпсона непрерывной случайной величины  описывается плотностью распределения

Распределение Симпсона имеет случайная величина , которая представляет собой следующую сумму:

X = y+z ,

где  и  - независимые случайные величины, распределенные равномерно на интервале . Следовательно, распределение Симпсона можно рассматривать как композицию двух одинаковых законов равномерного распределения.

Машинный алгоритм для имитации распределения Симпсона базируется на применении формулы X = y+z. Согласно этой формуле необходимо получить два случайных числа y и z, распределенных равномерно на интервале , и просуммировать их. Найденное таким образом число  будет иметь распределение Симпсона.

3. Имитация экспоненциального распределения

Экспоненциальное распределение непрерывной случайной величины *x* описывается плотностью распределения



где  - параметр экспоненциального распределения.

Математическое ожидание и дисперсия случайной величины  определяются соотношениями

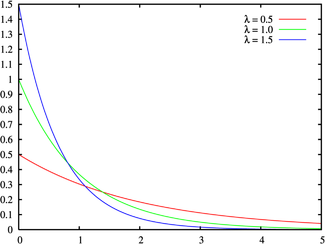
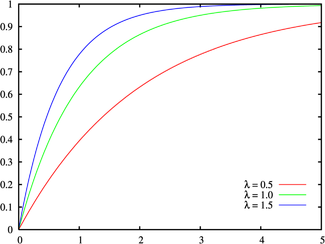
 и .

*Используем метод обратных функций* для имитации равномерного распределения:



 - искомый машинный алгоритм.

Плотность вероятности Функция распределения

[](http://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB:Exponential_distribution_pdf.p) [](http://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB:Exponential_distribution_cdf.p)

**Параметры** \lambda > 0 \,- интенсивность или обратный коэффициент масштаба

4. Имитация гамма-распределения

Гамма-распределение непрерывной случайной величины  описывается плотностью распределения:



где  и  - параметры гамма-распределения (η>0, λ>0)).

При η, принимающем целочисленные значения, гамма-распределение называется распределением Эрланга.

Гамма-распределение сводится к экспоненциальному распределению, если положить .

Математическое ожидание и дисперсия случайной величины *x* определяются соотношениями

 и .

Случайная величина *x* может быть представлена в виде суммы независимых случайных величин *x*i

.

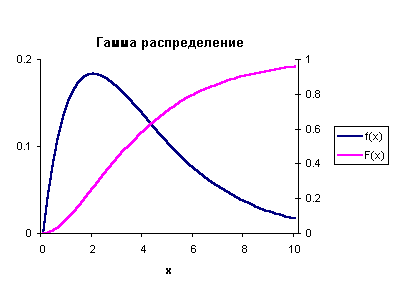
Если гамма-распределение случайной величины свести к экспоненциальному распределению, получим машинный алгоритм для имитации гамма-распределения.



или

 - искомый машинный алгоритм,

где  - СЧрр *r*.



5. Имитация нормального (гауссовского) распределения

Гауссовское распределение является одним из наиболее распространенных непрерывных распределений.

Гауссовское распределение непрерывной случайной величины *x* описывается плотностью распределения:



где  и  - соответственно математическое ожидание и среднее квадратическое отклонение гаусовского распределения.

Машинный алгоритм для имитации гауссовского распределения можно получить, *базируясь на центральной предельной теореме* (сумма независимых, случайных величин с произвольными распределениями имеет асимптотически гауссовское распределение). Сходимость к гауссовскому распределению осуществляется наиболее быстро, если суммируются величины с одинаковым распределением. В этом случае даже небольшое число слагаемых приводит к гауссовскому распределению.

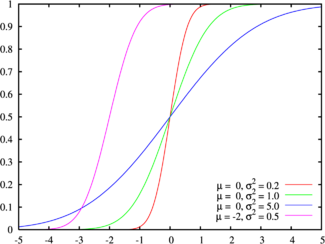
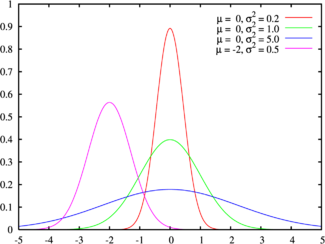
В основе машинного алгоритма для имитации гауссовского распределения лежит суммирование СЧрр *r*.

- искомый машинный алгоритм.

С возрастанием , т.е. числа суммируемых случайных СЧрр *r*, повышается точность имитации гауссовского распределения. Обычно  выбирают в пределах от 6 до 12. При этом достаточная для многих приложений точность обеспечивается при использовании всего шести СЧрр *r*. Для случая, когда  = 6:



Плотность вероятности Функция распределения

[](http://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB:Normal_distribution_cdf.p)[](http://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB:Normal_distribution_pdf.p)

Зеленая линия соответствует стандартному нормальному распределению

* - коэффициент масштаба (вещественный, строго положительный)

6. Имитация треугольного распределения

Может быть использован метод исключения И.Неймана.

Треугольное распределение непрерывной случайной величины *x* описывается плотностями распределения:

а) 

или

б) 

Машинный алгоритм

1. Формируются два СЧрр *r1* и *r2.*

2. Проверяется условие *r2 < r1*. Если условие выполняется, то находится искомое число

.

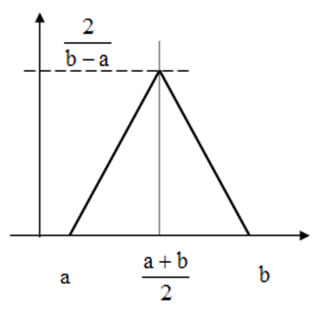
В противном случае пара чисел  отбрасывается и осуществляется переход к шагу 1.

* Приведенный алгоритм имеет существенный недостаток: часть пар чисел , приходится отбрасывать. Принимая во внимание независимость СЧрр *r1* и *r2*, можно предложить более экономичные алгоритмы, основанные на использовании следующих формул:

|  |  |
| --- | --- |
|  | **-** машинные алгоритмы  для имитации треугольного распределения |

где  - взятие максимального числа из совокупности двух СЧрр *r1* и *r2*;

 - взятие минимального числа на совокупности двух СЧрр *r1* и *r2*.



*Построение гистограммы*

Построение гистограммы распределения состоит в последовательном выполнении следующих этапов.

1. Находится минимальное  и максимальное  значения массива реализаций.

2. Определяется размах варьирования



3. Определяется длина интервала



где  - число интервалов.

4. Определяются граничные значения для каждого -го интервала 

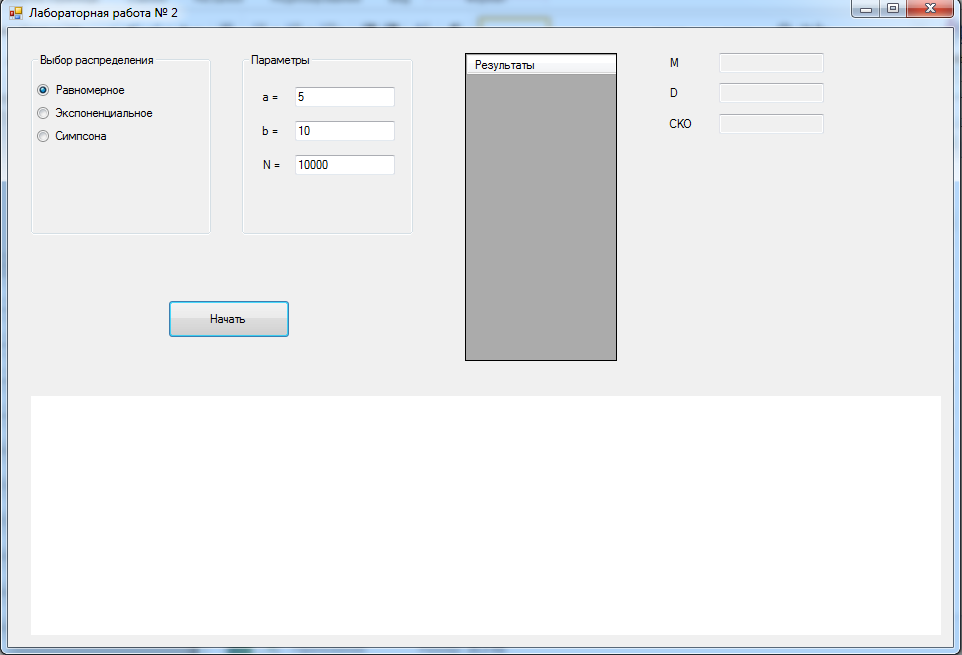
5. Фиксируется количество попаданий  в каждый -й интервал 

6. Вычисляются ординаты гистограммы распределения

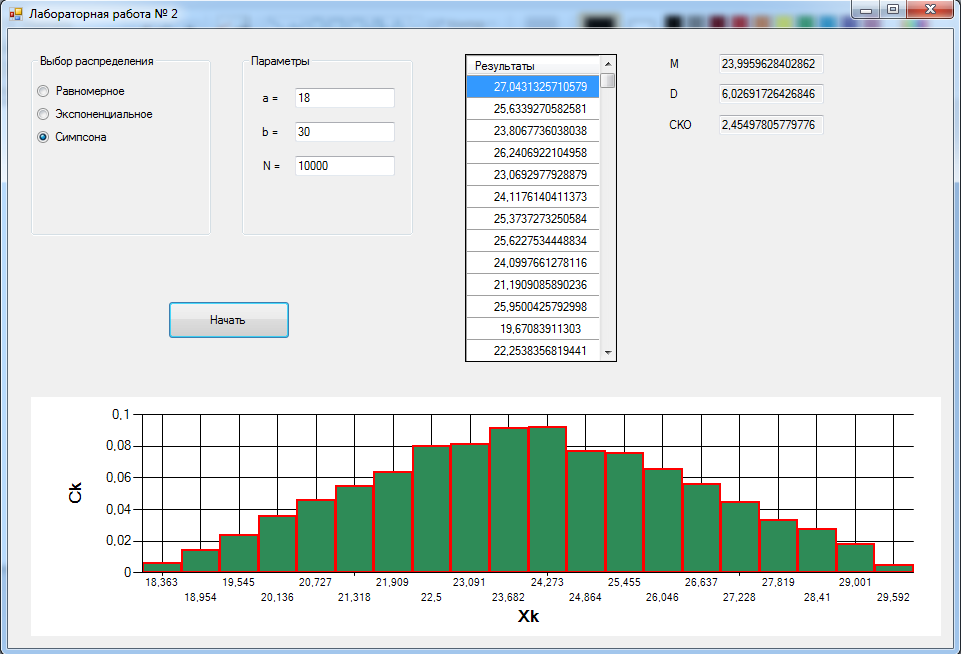
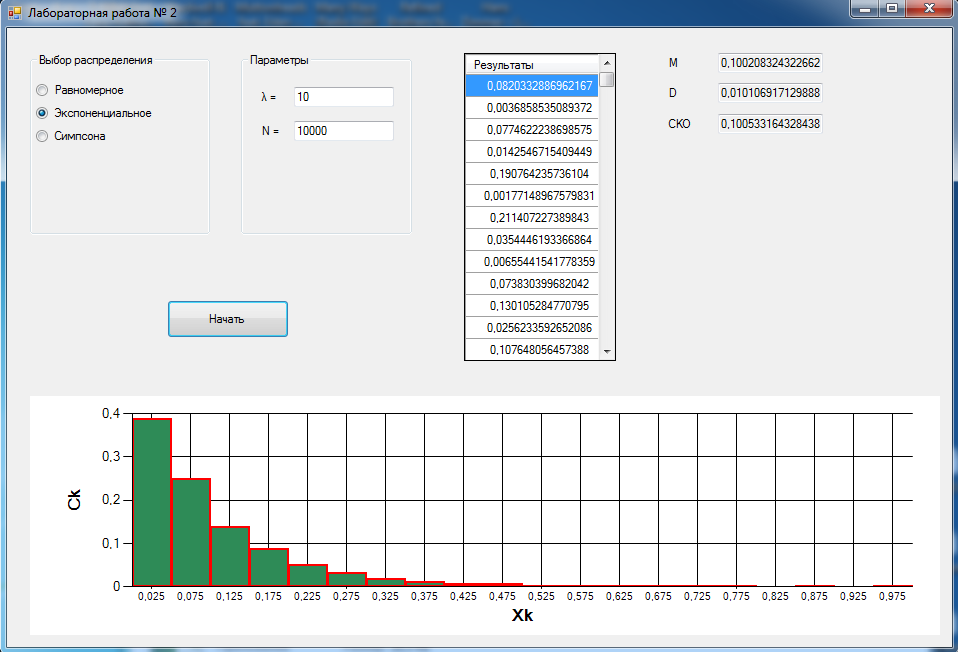


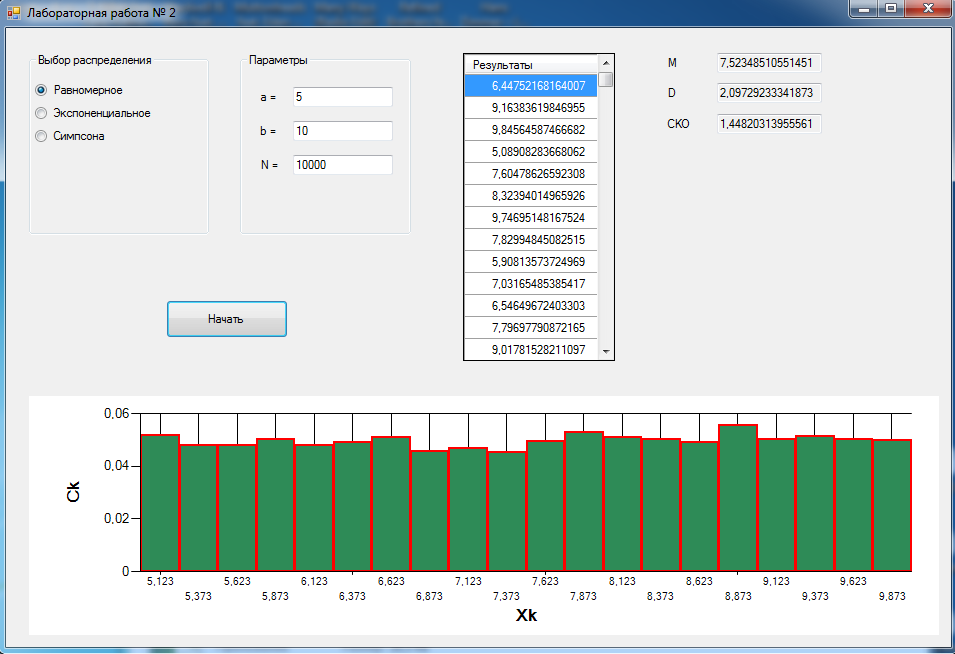
где  - число выполненных испытаний (объем массива реализаций).

Исходные данные:



Результат:





***Листингпрограммы:***

namespace Lab1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

radioButton\_uniform.Checked = true;

}

private double[] randArray;

private void radioButton\_uniform\_CheckedChanged(object sender, EventArgs e)

{

if (!radioButton\_uniform.Checked) return;

label1.Text = "a = "; textBox1.Text = "5";

label2.Text = "b = "; textBox2.Text = "10";

label3.Text = "N = "; textBox3.Text = "10000";

}

private void radioButton\_exp\_CheckedChanged(object sender, EventArgs e)

{

if (radioButton\_exp.Checked)

{

label1.Text = "λ = "; textBox1.Text = "10";

label2.Text = "N = "; textBox2.Text = "10000";

label3.Visible = false;

textBox3.Visible = false;

}

else

{

label3.Visible = true;

textBox3.Visible = true;

}

}

private void radioButton\_simpson\_CheckedChanged(object sender, EventArgs e)

{

if (!radioButton\_simpson.Checked) return;

label1.Text = "a = "; textBox1.Text = "18";

label2.Text = "b = "; textBox2.Text = "30";

label3.Text = "N = "; textBox3.Text = "10000";

}

private void UniformDistribution()

{

Random rand = new Random();

int N = int.Parse(textBox3.Text);

double a = double.Parse(textBox1.Text);

double b = double.Parse(textBox2.Text);

randArray = new double[N];

for (int i = 0; i < N; i++)

randArray[i] = a + (b - a) \* rand.NextDouble();

CalculateStatValues();

DrawHistogram();

dataGridView1.RowCount = N;

for (int j = 0; j < N; j++)

{

dataGridView1.Rows[j].Cells[0].Value = randArray[j];

}

}

// Экспоненциальное распределение

private void ExponentialDistribution()

{

Random rand = new Random();

int N = int.Parse(textBox2.Text); // Количество генерируемых чисел

double λ = double.Parse(textBox1.Text); // Параметр экспоненциального распределения

randArray = new double[N];

for (int i = 0; i < N; i++)

randArray[i] = - Math.Log(rand.NextDouble()) / λ;

CalculateStatValues();

DrawHistogram();

dataGridView1.RowCount = N;

for (int j = 0; j < N; j++)

{

dataGridView1.Rows[j].Cells[0].Value = randArray[j];

}

}

// Распределение Симпсона

private void SimpsonDistribution()

{

Random rand = new Random();

int N = int.Parse(textBox3.Text);

double a = double.Parse(textBox1.Text);

double b = double.Parse(textBox2.Text);

randArray = new double[N];

for (int i = 0; i < N; i++)

randArray[i] = a/2 + (b/2 - a/2)\*rand.NextDouble() + a/2 + (b/2 - a/2)\*rand.NextDouble();

CalculateStatValues();

DrawHistogram();

dataGridView1.RowCount = N;

for (int j = 0; j < N; j++)

{

dataGridView1.Rows[j].Cells[0].Value = randArray[j];

}

}

// Вычисление математического ожидания, дисперсии и СКО

private void CalculateStatValues()

{

double Mx = randArray.Sum() / randArray.Length;

textBox6.Text = Mx.ToString();

double Dx = randArray.Sum(t => (t - Mx) \* (t - Mx)) / (randArray.Length - 1);

textBox7.Text = Dx.ToString();

textBox8.Text = (Math.Sqrt(Dx)).ToString();

}

private void DrawHistogram()

{

List<double> numbers = new List<double>(randArray);

numbers.Sort();

const int intervalsCount = 20;

double width = numbers.Last() - numbers.First();

double widthOfInterval = width / intervalsCount;

double[] heights = new double[intervalsCount]; // Высота столбцов гистограммы

double[] X\_values = new double[intervalsCount]; // Значение по оси x

X\_values[0] = Math.Round(0.0245 \* width + numbers.First(),3);

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

X\_values[i] = Math.Round(X\_values[i - 1] + widthOfInterval,3);

double xLeft = numbers.First(); // Начало диаграммы по оси x

double xRight = xLeft + widthOfInterval; // Конец текущего интервала по оси x

int j = 0;

for (int i = 0; i < intervalsCount; i++)

{

while (j < numbers.Count && xLeft <= numbers[j] && xRight > numbers[j])

{

heights[i] ++;

j++;

}

heights[i] /= numbers.Count;

xLeft = xRight;

xRight += widthOfInterval;

}

chart1.ChartAreas["ChartArea1"].AxisX.Interval = 1;

chart1.Series["Series1"]["PointWidth"] = "1";

chart1.Series["Series1"].Points.DataBindXY(X\_values,heights);

}

private void button1\_Click(object sender, EventArgs e)

{

if (radioButton\_uniform.Checked)

UniformDistribution();

else if (radioButton\_exp.Checked)

ExponentialDistribution();

else if (radioButton\_simpson.Checked)

SimpsonDistribution();

}

private void chart1\_Click(object sender, EventArgs e)

{

}

}

}