Министерство науки высшего образования

Российской Федерации

Новосибирский государственный технический университет

Кафедра Автоматизированных систем управления

Лабораторная работа №1

**Построение и оценка качества генераторов псевдослучайных чисел**

Факультет: АВТФ Преподаватель:

Группа: АВТ-113 Лыгина Н.И.

Студенты:

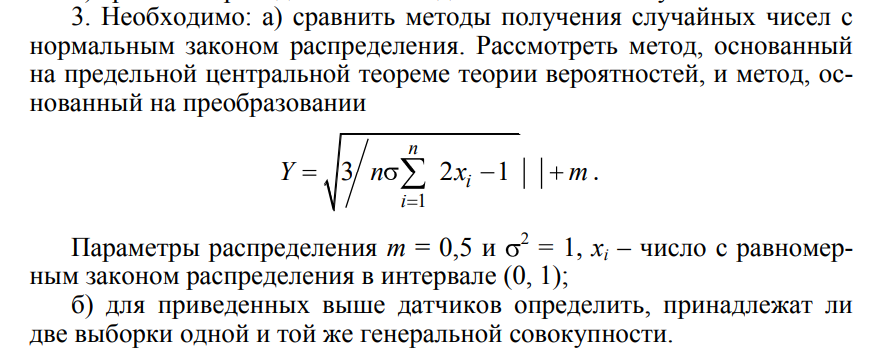
Зеленев П.А.

Осокин Д.М.

Салихова Е.А.

Новосибирск 2023

1. **Задание**



1. **Алгоритм моделирования случайных чисел с заданным законом распределения и используемые критерии согласия**.

В нашей работе используется алгоритм моделирования чисел на основе центральной предельной теоремы и метод из задания. Для проверки принадлежности этих двух методов к одной генеральной совокупности будем использовать критерий согласия Колмогорова-Смирнова.

1. **Программа, реализующая соответствующий алгоритм**.

|  |
| --- |
| using System;  using System.Runtime.InteropServices;  using System.Windows.Forms;  using System.Windows.Forms.DataVisualization.Charting;  namespace ModelLab1  {  public partial class Form1 : Form  {  private const int NumberOfSampleElements = 20; // количество элементов одной выборки  private double[] X = new double[NumberOfSampleElements]; // элементы выборки  private Random rnd = new Random();    private const int NumberOfSamples = 10000; // количество выборок  private double[] CLT\_Y = new double[NumberOfSamples]; // результаты преобразования выборок по формулам  private double[] Y = new double[NumberOfSamples]; // результаты преобразования выборок по формулам  private double[] Fx = new double[NumberOfSamples];    private const double m = 0.5; // матожидание  private const double dispersion = 1;    public Form1()  {  InitializeComponent();  methods();  }  void GenerateY()  {  for (int i = 0; i < NumberOfSamples; i++)  {  generateXSample();  double sumOfXElements = 0;  for (int j = 0; j < NumberOfSampleElements; j++)  {  sumOfXElements += Math.Abs(2 \* X[j] - 1);  }    Y[i] = Math.Sqrt(3 / (NumberOfSampleElements \* dispersion \* sumOfXElements)) + m;  }  }  void GenerateCLT\_Y()  {  for (int i = 0; i < NumberOfSamples; i++)  {  generateXSample();  double sumOfXElements = 0;  for (int j = 0; j < NumberOfSampleElements; j++)  {  sumOfXElements += X[j];  }  CLT\_Y[i] = (sumOfXElements - m \* NumberOfSampleElements) / (dispersion \* Math.Sqrt(NumberOfSampleElements));  }  }  void methods()  {  GenerateY();  Array.Sort(Y);  GenerateCLT\_Y();  Array.Sort(CLT\_Y);    CreateGistogramms(Y, firstChart, distributionFirst, 0);  CreateGistogramms(CLT\_Y, secondChart, distributionSecond, 0);    CreateGistogramms(CLT\_Y, density, distribution, 0);  CreateGistogramms(Y, density, distribution, 1);    }  void CreateGistogramms(double[] YNow, Chart chart, Chart chart1, int t)  {  double minValue, maxValue, intervalLength;  int countOfIntervals;  countOfIntervals = (int)Math.Floor(1 + 3.322 \* Math.Log10(NumberOfSamples));  // нахождение длины интервала  minValue = YNow[0];  maxValue = YNow[NumberOfSamples - 1];  intervalLength = (maxValue - minValue) / countOfIntervals;  double[] ProbabilitiesOfHittingTheInerval = new double[countOfIntervals];  int[] CounterOfNumbersOnInterval = new int[countOfIntervals];  double[] MiddlesOfTheIntervals = new double[countOfIntervals];    for (int i =0; i < countOfIntervals; i++)  {  CounterOfNumbersOnInterval[i] = 0;  }  // нахождение середин интервалов  for (int i = 0; i < countOfIntervals; i++)  {  double leftBorderOfInterval = minValue + (i \* intervalLength);  double rightBorderOfInterval = minValue + ((i+1) \* intervalLength);  MiddlesOfTheIntervals[i] = (leftBorderOfInterval + rightBorderOfInterval) / 2;  }  for (int i = 0; i < NumberOfSamples; i++)  {  double leftBorderOfInterval = minValue;  int j = 0;  while(true)  {  if (YNow[i] >= leftBorderOfInterval && YNow[i] < leftBorderOfInterval + intervalLength)  {  CounterOfNumbersOnInterval[j]++;  break;  }  // случай попадания варианта на крайнюю правую границу  if (j == countOfIntervals - 1 && YNow[i] >= leftBorderOfInterval + intervalLength)  {  CounterOfNumbersOnInterval[j]++;  break;  }    j++;  if (j == countOfIntervals)  break;  leftBorderOfInterval += intervalLength;  }  }  int summ = 0; // для проверки (должна быть равна количеству выборок)  for (int j = 0; j < countOfIntervals; j++)  {  summ += CounterOfNumbersOnInterval[j];  }  double summs = 0; // для проверки (должна быть равна 1)  for (int i = 0; i < countOfIntervals; i++)  {  ProbabilitiesOfHittingTheInerval[i] = (double)CounterOfNumbersOnInterval[i] / NumberOfSamples;  summs += ProbabilitiesOfHittingTheInerval[i];  chart.Series[t].Points.AddXY(Math.Round(MiddlesOfTheIntervals[i], 4), Math.Round(ProbabilitiesOfHittingTheInerval[i], 4));  }  summs = 0;  for (int i = 0; i < countOfIntervals; i++)  {  ProbabilitiesOfHittingTheInerval[i] = (double)CounterOfNumbersOnInterval[i] / NumberOfSamples;  summs += ProbabilitiesOfHittingTheInerval[i];  chart1.Series[t].Points.AddXY(Math.Round(MiddlesOfTheIntervals[i], 4), Math.Round(summs, 4));  }  }  void generateXSample()  {  // Создаём набор чисел с равномерным законом распределения на интервале (0,1)  for (int i = 0; i < NumberOfSampleElements; i++)  {  X[i] = rnd.NextDouble();  }  }  private void button1\_Click(object sender, EventArgs e)  {  firstChart.Series[0].Points.Clear();  secondChart.Series[0].Points.Clear();  distributionFirst.Series[0].Points.Clear();  distributionSecond.Series[0].Points.Clear();  density.Series[0].Points.Clear();  density.Series[1].Points.Clear();  distribution.Series[0].Points.Clear();  distribution.Series[1].Points.Clear();  methods();  }  private void button2\_Click(object sender, EventArgs e)  {  firstChart.Series[0].Points.Clear();  secondChart.Series[0].Points.Clear();  distributionFirst.Series[0].Points.Clear();  distributionSecond.Series[0].Points.Clear();  density.Series[0].Points.Clear();  density.Series[1].Points.Clear();  distribution.Series[0].Points.Clear();  distribution.Series[1].Points.Clear();  methods();  }  private void button3\_Click(object sender, EventArgs e)  {  firstChart.Series[0].Points.Clear();  secondChart.Series[0].Points.Clear();  distributionFirst.Series[0].Points.Clear();  distributionSecond.Series[0].Points.Clear();  density.Series[0].Points.Clear();  density.Series[1].Points.Clear();  distribution.Series[0].Points.Clear();  distribution.Series[1].Points.Clear();  methods();  }  }  } |

1. **Проверка и результаты гипотезы о распределении случайных чисел с помощью критериев согласия.**

В критерии Колмогорова-Смирнова в качестве H0 рассматривается гипотеза о том, что две выборки принадлежат одной и той же генеральной совокупности. Для этого критерия надо найти максимальную разницу числа D на оси Y при одинаковом значении Х.

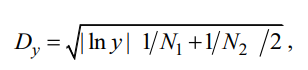


Найдем это значение на графике программы рис.1:

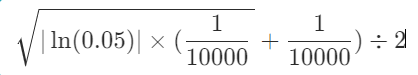


Рис.1 – Гистограммы, отображающая функции распределения

Здесь видно, что максимальное D ≈ 0.95. Теперь при заданном уровне значимости у (возьмем 0.05) находим допустимое отклонение по этой формуле:



Где N1 и N2 - количество выборки и равны 10000.



Получаем Dy = 0.0122

Так как 0.95 > 0.0122, то нулевая гипотеза H0 о тождественности законов распределения наших методов с доверительной вероятностью 0.95 отвергается.

1. **График экспериментальной функции распределения случайной величины или гистограммы.**

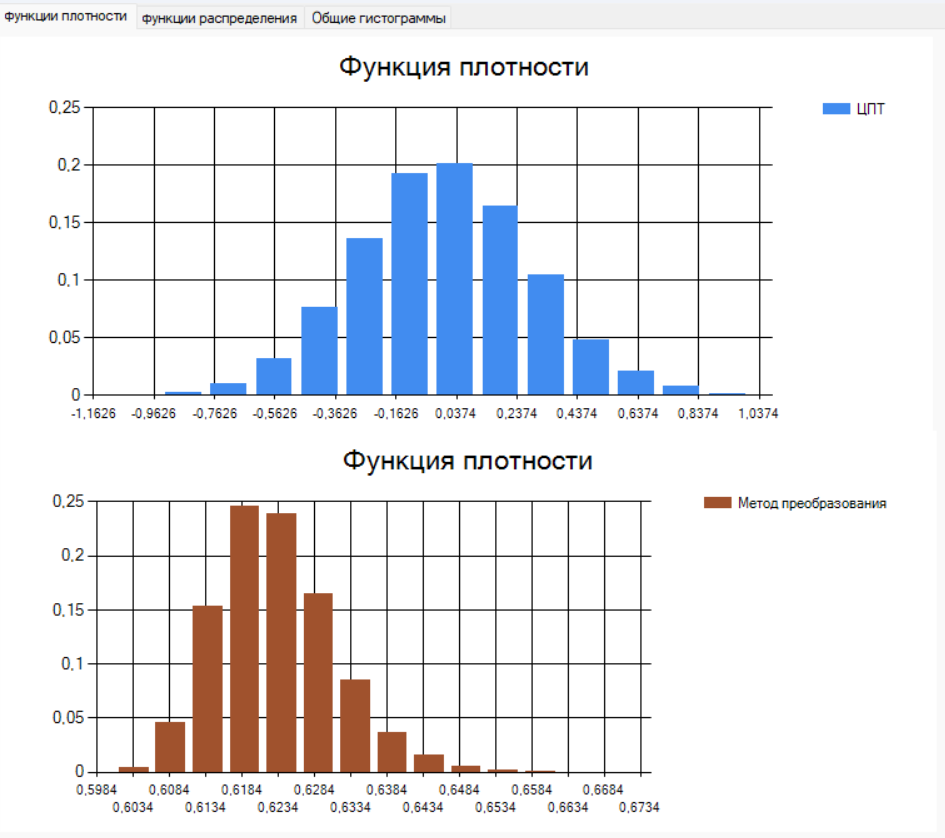


Рис. 2 – Гистограммы плотности по отдельности.

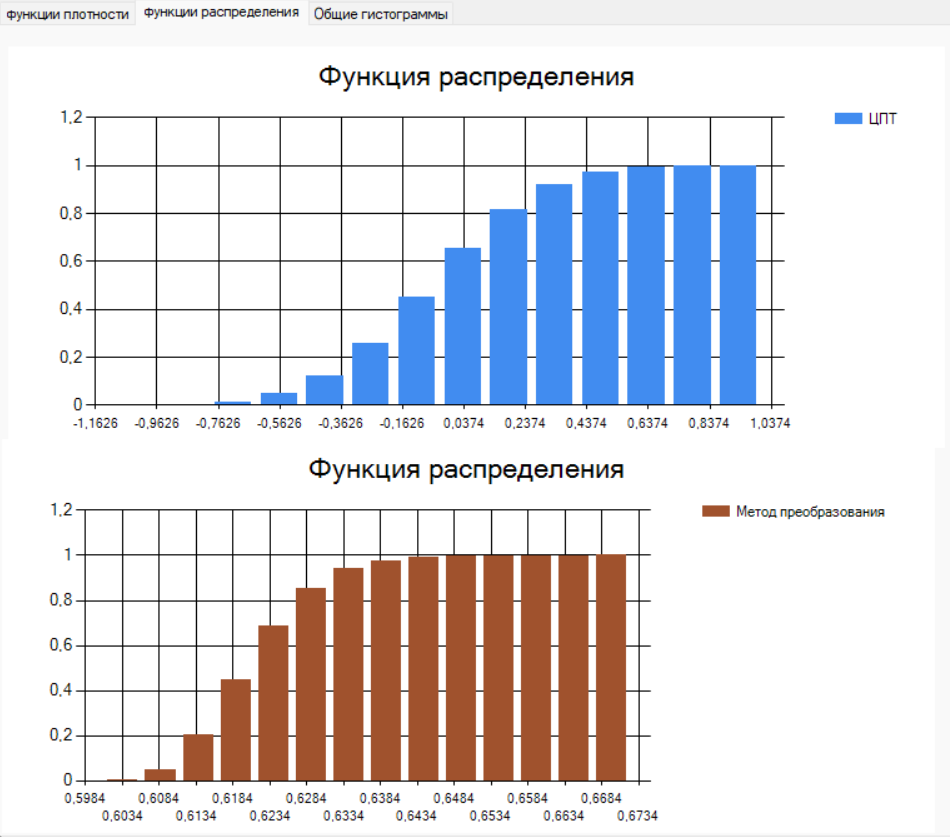


Рис. 3 – Гистограммы функций распределения по отдельности.

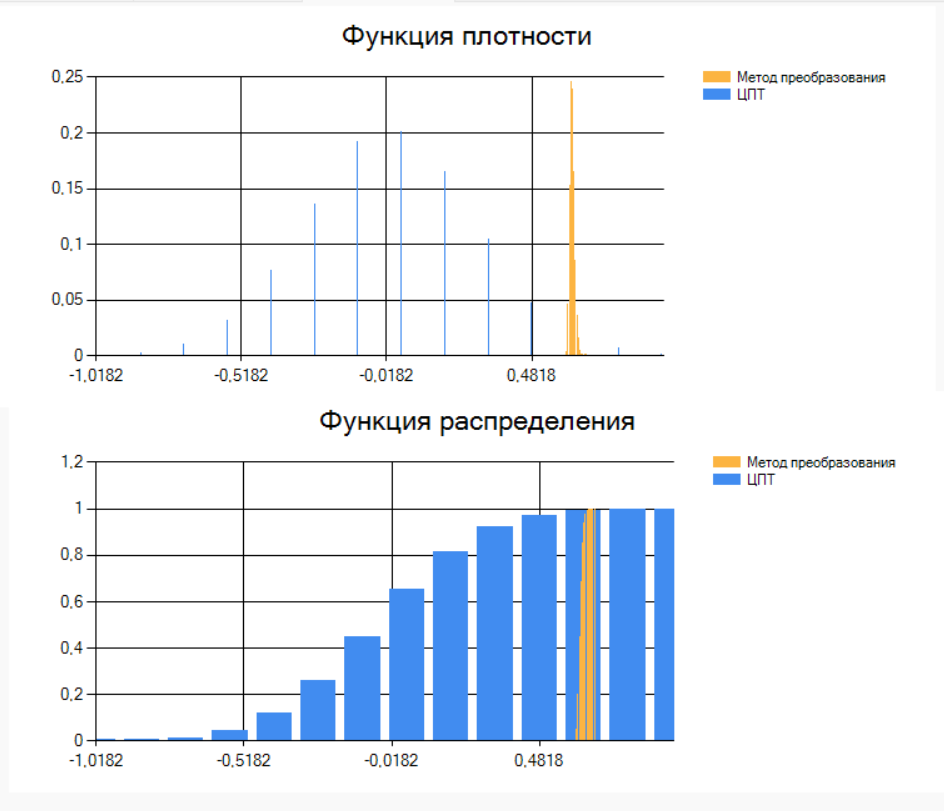


Рис. 4 – Гистограммы плотности и распределения вместе.

1. **Анализ полученных результатов**

Из имеющихся результатов можно сделать вывод, что наши методы совершенно не соответствуют одной генеральной совокупности.