ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

ФАКУЛЬТЕТ ИНФОРМАТИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Кафедра вычислительных систем

ОТЧЁТ

по лабораторной работе №1
по дисциплине «Моделирование»
на тему «Генерация непрерывных случайных величин
с заданной плотностью распределения»

Выполнил:
студент группы ИВ-222
Терешков Р. В.

Проверил:
д.т.н., доцент
Родионов А. С.

Новосибирск – 2016

Задание

В рамках выполнения лабораторной работы предлагается реализовать датчик непрерывно распределённых псевдослучайных чисел с заданной функцией плотности распределения. Функция плотности распределения представлена на рисунке 1.

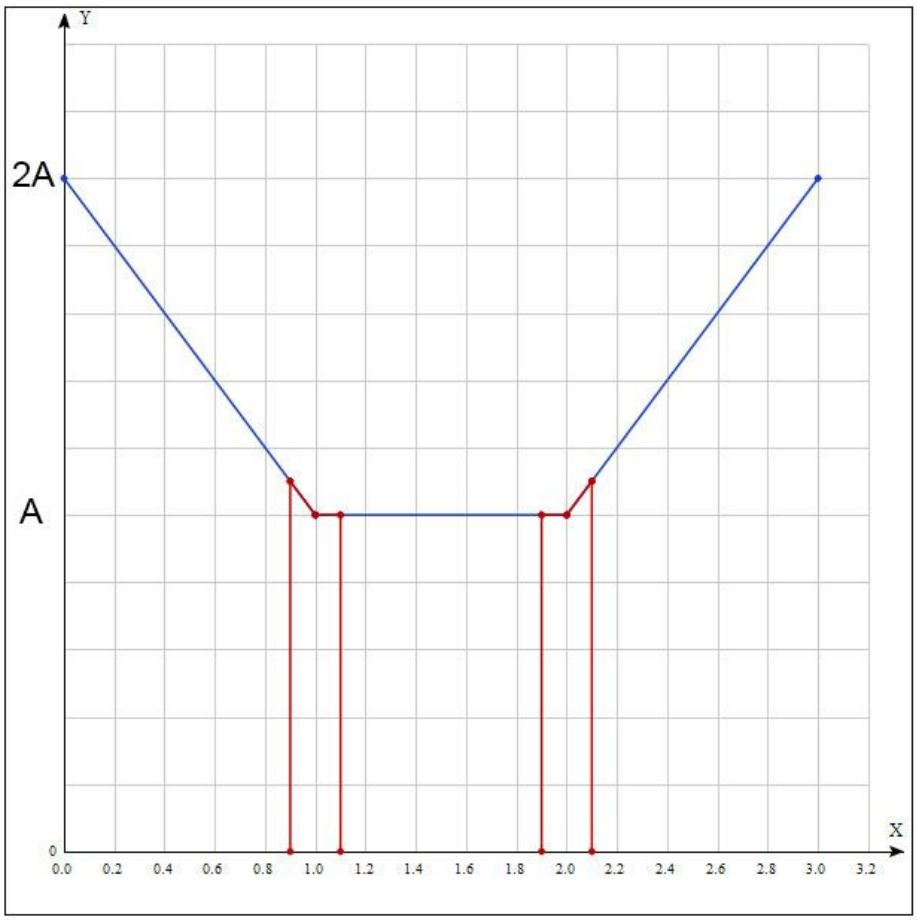

Рис. 1 – Функция плотности распределения случайной величины

Ход работы

В ходе выполнения лабораторной работы была реализована программа на языке программирования C++, выполняющая моделирование псевдослучайных величин по заданному закону распределения.

Для вычисления закона распределения случайной величины необходимо найти неизвестный параметр A. Логически разделим площадь всей фигуры на равные участки:

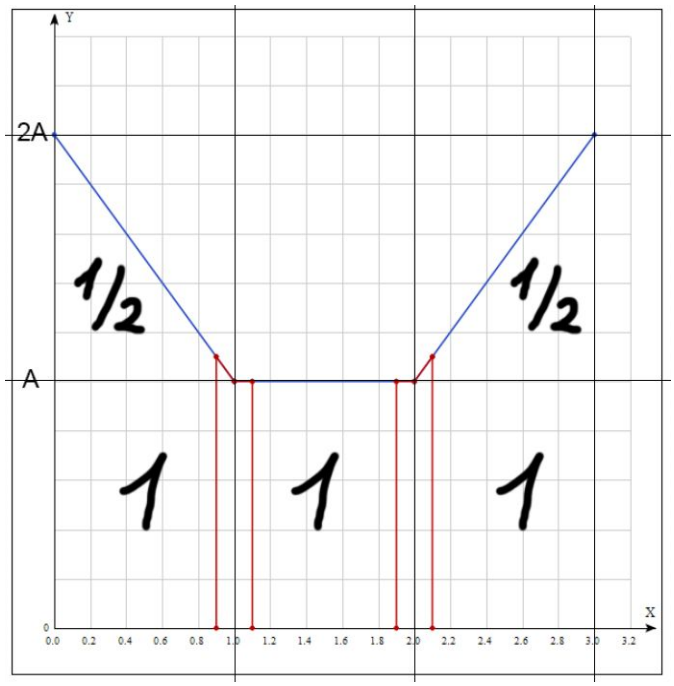

Рис. 2 – Логическое разбиение фигуры на составляющие

Площадь прямоугольников: 3 * 1 * A = 3A;
Площадь треугольников: 2 * ½ * A = A;
Общая площадь: 3A + A = 1 => 4A = 1 => A = ¼.

Вероятность попадания в трапецеидальную область находится по формуле площади трапеции, в прямоугольную – по площади прямоугольника соответственно (Рисунок 3). В результате получаем: $\mathbf{p_1}$ = ⅜, $\mathbf{p_2}$ = ¼.

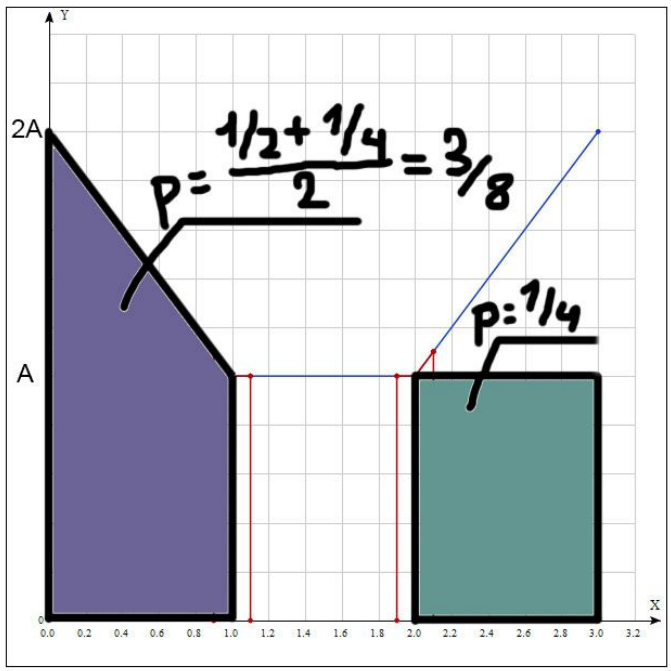

Рис. 3 – Вероятности попадания в выделенные области

Рассмотрим область для генерации псевдослучайных чисел. Данная фигура представляет из себя два одинаковых прямоугольника и прямоугольный треугольник:

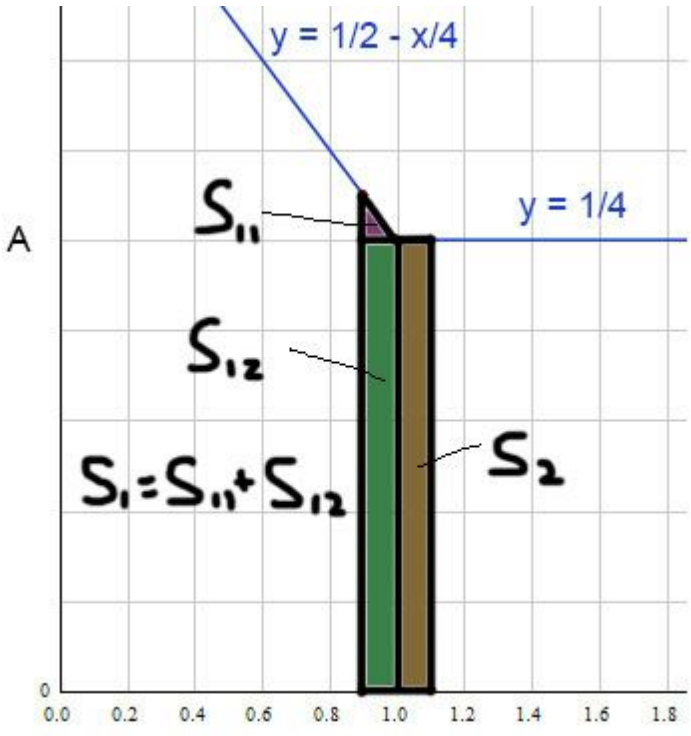

Рис. 4 – Область для генерации псевдослучайных чисел

Площадь прямоугольников: 0.1 * A = 0.1 * ¼ = 0.025;
Площадь треугольника (находим по подобию треугольников): 0.0125.
В результате получаем: $\mathbf{S_1}$ = 0.0375, $\mathbf{S_2}$ = 0.025, $\mathbf{S_{общ}}$ = 0.0625.

Далее находим вероятности u_{11}, u_{12} попадания случайной величины в данные области, а также вероятности u_{21} и u_{22}, которые указывают на зеркальные области (правая часть графика):

$\mathbf{u_{11}}$ = $\mathbf{u_{22}}$ = ½ * 0.0375 / (0.0625) = 0.3;
$\mathbf{u_{12}}$ = $\mathbf{u_{21}}$ = ½ * 0.0250 / (0.0625) = 0.2.

В конечном итоге псевдослучайные величины для каждого интервала генерируются по следующим законам:

$\mathbf{x_{11}}$ = (4 - √(16 - 12 * (ʂ * 0.1 + 0.9))) / 2;
$\mathbf{x_{12}}$ = ʂ * 0.1 + 1;
$\mathbf{x_{21}}$ = ʂ * 0.1 + 1.9;
$\mathbf{x_{22}}$ = √(3 * (ʂ * 0.1 + ⅓)) + 1.

Результаты моделирования

На основе результатов выполнения программы была построена частотная гистограмма, отражающая распределение смоделированных псевдослучайных величин. Содержимое полученной гистограммы повторяет график заданной функции плотности распределения случайной величины (Рисунок 5). Листинг программы представлен ниже.

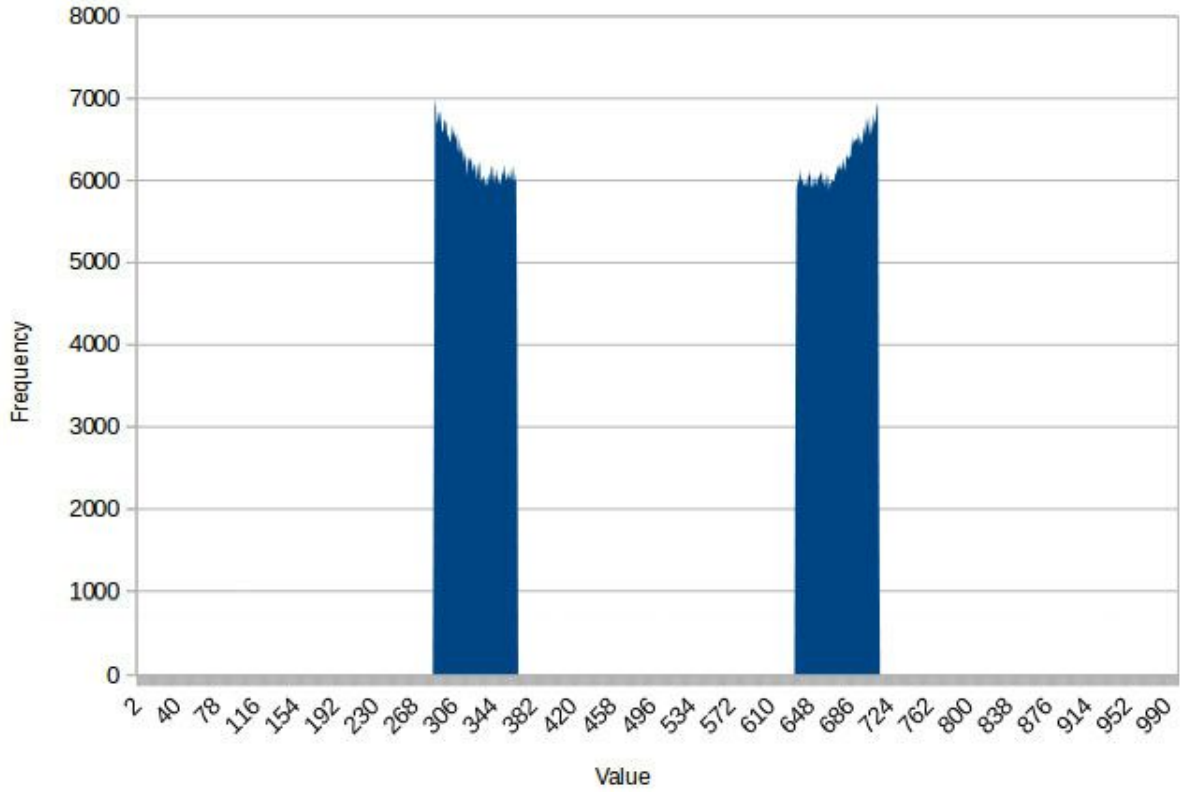

Рис. 5 – Частотная гистограмма распределения псевдослучайной величины

Листинг программы

```
#include <iostream>
#include <fstream>
#include <ctime>
#include <cstdlib>
#include <cmath>

using namespace std;

enum vars { N = 1000000, n = 1000};

double a[4] = { 0.9, 1.0, 1.9, 2.0 }, b[4] = { 1.0, 1.1, 2.0, 2.1 };
int histogram[n] = { 0 };
double numbers[N];

int main()
{
	srand(time(0));

	for (int i = 0; i < N; ++i) {
		double ksi = drand48();
		if (ksi <= 0.3) {
			ksi = drand48() * (b[0] - a[0]) + a[0];
			numbers[i] = (4 - sqrt(16 - 12 * ksi)) / 2.0;
		} else if (ksi <= 1 - 0.3) {
			ksi = drand48();
			if (ksi <= 0.5) {
				ksi = drand48() * (b[1] - a[1]) + a[1];
				numbers[i] = ksi;
			} else {
				ksi = drand48() * (b[2] - a[2]) + a[2];
				numbers[i] = ksi;
			}
		} else {
			ksi = drand48() * (b[3] - a[3]) + 1.0 / 3.0;
			numbers[i] = sqrt(3 * ksi) + 1;
		}
		++histogram[int((numbers[i] * n) / 3)];
	}

	ofstream os("res.out");
	for (int i = 0; i < n; ++i) os << histogram[i] << endl;
	os.close();

	return 0;
}
```