Федеральное агентство связи
Федеральное государственное бюджетное образовательное учреждение
высшего образования «Сибирский государственный университет
телекоммуникаций и информатики»

Кафедра вычислительных систем

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к нулевой лабораторной работе по дисциплине
«Моделирование»

Выполнил:
студент гр. ИВ-621
Антипова Е.П.

Проверила:
Ассистент кафедры ВС
Петухова Я.В.

Новосибирск, 2020

Содержание

Задание

Убедиться в равномерности трех различных генераторов псевдослучайных чисел (ГПСЧ), используя параметры «хи-квадрат» (критерий согласия Пирсона) и автокорреляцию.

Теория

- *Генератор псевдослучайных чисел (ГПСЧ)*

ГПСЧ – это программа, которая принимает стартовое/начальное значение и выполняет с ним определённые математические операции, чтобы конвертировать его в другое число, которое совсем не связанное со стартовым. Затем программа использует новое сгенерированное значение и выполняет с ним те же математические операции, что и с начальным числом, чтобы конвертировать его в ещё в одно новое число — третье, которое не связано ни с первым, ни со вторым. Применяя этот алгоритм к последнему сгенерированному значению, программа может генерировать целый ряд новых чисел, которые будут казаться случайными (при условии, что алгоритм будет достаточно сложным).

Все ГПСЧ являются циклическими, т.е. в какой-то момент последовательность генерируемых чисел начнёт повторяться. ГПСЧ являются детерминированными, и с одним значением ввода мы получим одно и то же значение вывода. Что произойдёт, когда ГПСЧ сгенерирует число, которое уже ранее было сгенерировано. С этого момента начнётся дублирование последовательности чисел между первым и последующим появлением этого числа. Длина этой последовательности называется периодом.

- *Критерий согласия Пирсона*

С помощью критерия согласия определяют, удовлетворяет ли ГПСЧ требования равномерного распределения или нет.

Формулы для вычисления «хи-квадрат»

N - общее количество сгенерированных чисел

p_i - теоретическая вероятность попадания чисел в i-ый интервал ($p_i = \frac{1}{k}$)

k – общее количество интервалов

n_i - попадание чисел в каждый интервал

x^2 –критерий согласия

Сгенерировать псевдослучайные числа в диапазоне от 0 до 1. Разбить диапазон на k равных интервала. Запустить ГПСЧ N раз (N должно быть велико, например, $N/k > 5$). Выяснить количество случайных чисел, попавший в каждый интервал. Вычислить хи-квадрат.

$$x^2 = \frac{1}{N}\sum_{i=1}^{k}\left(\frac{n_i^2}{p_i}\right) - N$$

- *Автокорреляция*

Статистическая взаимосвязь между случайными величинами из одного ряда, но взятых со сдвигом.

Формулы для вычисления автокорреляции

E_x - математическое ожидание

S^2 - выборочная дисперсия

$\hat{a}(\tau)$ – автокорреляция

x_i - множество псевдослучайных чисел

τ - смещение

$$E_x = \sum_{i=1}^{N}\frac{x_i}{N};\ \ S^2 = \sum\frac{x_i^2}{N} - (E_x)^2;$$

$$\hat{a}(\tau) = \frac{1}{(N-\tau)*S^2}\sum_{i=1}^{N-\tau}(x_i - E_x)(x_{i+\tau} - E_x);$$

Генераторы

1) *get_rand()* – функция из стандартной библиотеки C, которая генерирует псевдослучайные числа в диапазоне от 0 до RAND_MAX. RAND_MAX это специальная константа языка Си, в которой содержится максимальное целое число, которое может быть возвращено функцией *get_rand()* . При генерации случайных чисел типа с плавающей точкой используется Линейный Конгруэнтный Метод(ЛКМ).
2) *mt19937* - алгоритм Вихрь Мерсенна. Генератор пвседослучайных чисел (ГПСЧ), основывающийся на свойствах простых чисел Мерсенна и обеспечивающий быструю генерацию высококачественных по критерию случайности псевдослучайных чисел. «Вихрь» - преобразование, которое обеспечит равномерное распределение генерируемых псевдослучайных чисел в 623 измерениях (для линейных конгруэнтных генераторов оно ограничено 5 измерениями).
3) *Uniform_real_distribution<float>* – Формирует равномерное распределение (каждое значение одинаково вероятно) случайных значений с плавающей запятой. Производит случайные значения с плавающей запятой, равномерно распределенные на интервале.

Результаты

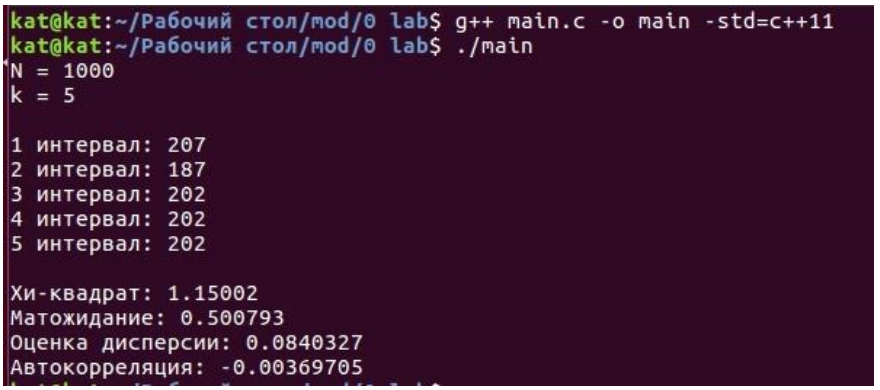

Рис.1. Запуск программы с использованием get_rand() при k = 5 и N = 1000

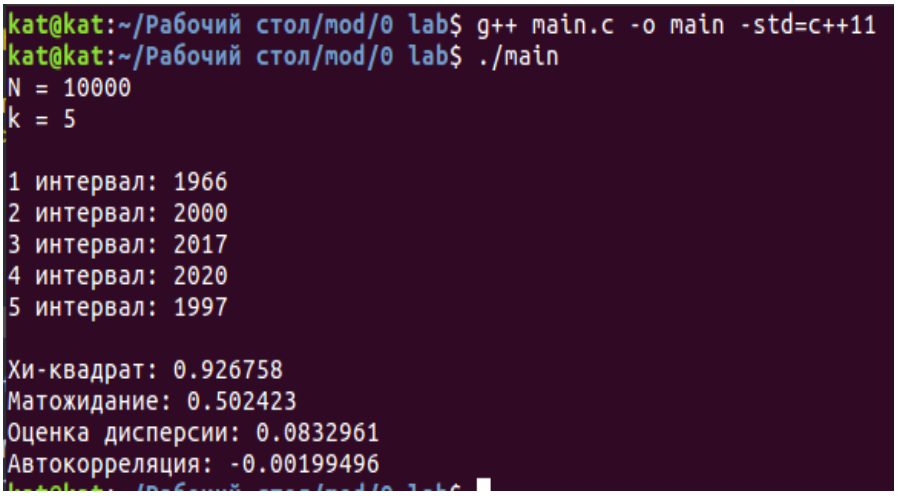

Рис.2. Запуск программы с использованием get_rand() при k = 5 и N = 10000

Рис.3. Запуск программы с использованием mt19937 при k = 5 и N = 1000

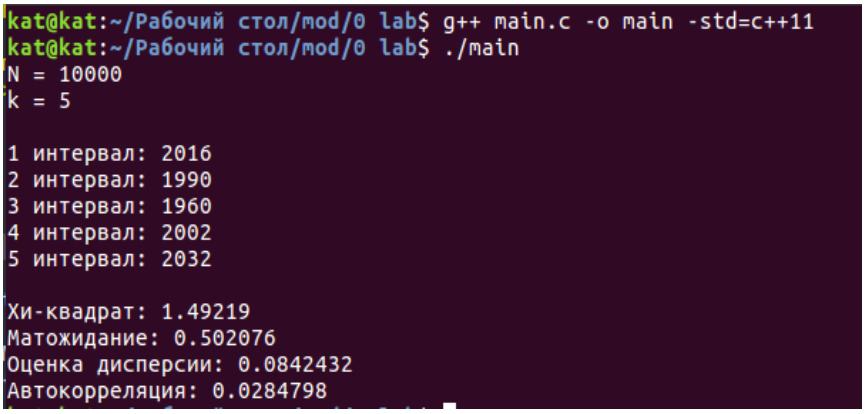

Рис.4. Запуск программы с использованием
mt19937 при k = 5 и N = 10000

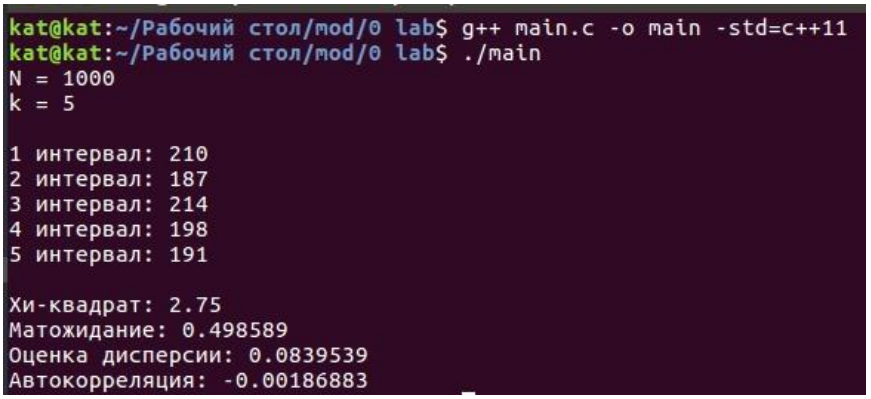

Рис.5. Запуск программы с использованием
Uniform_real_distribution<float> при k = 5 и N = 1000

Рис.6. Запуск программы с использованием
Uniform_real_distribution<float> при k = 5 и N = 10000

Получив значения Хи-квадрат и зная число степеней свободы:

$$k = m - r - 1,$$

m – это количество интервалов,

r – это количество параметров в конкретной функции распределения),

a = 0.05 (уровень значимости, выбранное случайным образом).

Можно обратиться к таблице значений Хи-квадрат и определить значение нормали. При k = 2 и a = 0.05 будет равна 6.0.

Сравним фактические значения, полученные при трех запусках (при k = 5 и N=1 000) 1.15, 4.22, 2.75 с табличным значением.
Расчетные критерии оказались меньше.

Коэффициент автокорреляции колеблется в диапазоне от -1 до 1.

По результатам можно увидеть, что в первом случае значения автокорреляции равны: -0.003, 0.0006, -0.001, а в последующих при увеличении N: -0.001, 0.028, -0.178.

Вывод

В ходе выполнения лабораторной работы были изучены три различных генератора псевдослучайных чисел и методы тестирования качества работы ГПСЧ: критерий согласия Пирсона и автокорреляция.

У нас есть нулевая гипотеза, она заключается в том, что мы ожидаем, что ГПСЧ имеет равномерное распределение, то есть те результаты, которые мы хотим получить (фактические) не будут противоречить ожидаемым. Если этот так, то разброс будет относительно небольшим, в пределах случайных колебаний. По результатам работы можно отметить, что ГПСЧ действительно имеет равномерное распределение, потому что, как мы отметили выше - расчетные критерии оказались меньше, значит гипотеза о равенстве (согласии) частот не отклоняется. Нулевая гипотеза подтверждена.

Чем выше значение коэффициента автокорреляции, тем выше связь. По результатам можно увидеть:

1. При увеличении N заметно увеличивается и количество точек, попавших в интервалы;
2. Прямой зависимости автокорреляции от смещения нету. Оно помогает выявить зависимость между значениями;
3. Значения автокорреляции колеблются около нуля. Следовательно, можно сказать, что числа полученными такими генераторами, близки к случайным.

Стоит отметить что все протестированные генераторы показали значения хи-квадрат не превышающие критического. Следовательно, что для всех генераторов гипотезу о равномерном распределении нельзя опровергнуть. Но mt19937 в проведенных экспериментах выдал наименьшее значение хи-квадрата из всех имеющихся. Это говорит о том, что данный генератор выдавал более равномерное распределение, чем другие генераторы.

Листинг программы

```
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#include <math.h>
#include <iostream>
#include <iomanip>
#include <random>
#include <ctime>
#include <cmath>
#include <chrono>

#define N 1000

using namespace std;
const float RAND_MAX_F = RAND_MAX;

float get_rand() {
	return rand() / RAND_MAX_F;
}
//первый генератор
float get_rand_range(const float min, const float max) {
	return get_rand() * (max - min) + min;
}
int main(){
	int k = 5;
	int count[k];
	int index =0;
	float numeric[N];

	//второй генератор
	mt19937 gen(time(0));
	uniform_real_distribution<> urd(0, 1);
	//третий генератор
	unsigned seed =
	std::chrono::system_clock::now().time_since_epoch().count();
	std::uniform_real_distribution<float> dist(0, 1);
	std::mt19937_64 rng(seed);

	for(int i = 0; i < k; i++)
	{
		count[i] = 0;
	}
	srand(time(NULL));
	float Q = 0.0;
	float sum = 0, sum_kvdr = 0;

	for(int i = 0; i < N; i++)
	{
		//float gch = get_rand_range(0, 1); //для первого генератора
		//float gch = urd(gen); //для второго генератора
		float gch = dist(rng); //для третьего генератора
		numeric[i] = gch;
		sum += gch;
		sum_kvdr += gch * gch;
		//cout << "Rand: " << gch << "\n";
		Q = 0.0;
		for(index = 0; index < k; index++){
			if(gch > Q && gch <= Q + (float)1/k){
				count[index]++;
```

```
                        //cout << "INDEX: "<<count[index] << "\n";
                        break;
                  }
                  Q += (float)1/k;
                  //cout << "Q: " << Q << "\n";
            }
            //cout << get_rand_range(0,1) << "\n";
      }
      cout << "N = " << N << "\nk = " << k << "\n\n";
      int interv = 1;
      for(int i = 0; i < k; i++){
            cout << interv << " интервал: " << count[i]<< "\n";
            interv++;
      }
      float HI = 0.0;
      for(int i = 0; i < k; i++){
            HI += (float)count[i] * count[i] / (1 / (float)k);
      }
      HI /= (float)N;
      HI -= (float)N;

      cout << "\n" << "Хи-квадрат: " << HI << "\n";
      float Ex = 0.0;
      Ex = sum / (float) N;
      cout << "Матожидание: " << Ex << "\n";
      float S = 0.0;
      S = (sum_kvdr / (float) N) - (Ex * Ex);
      cout << "Оценка дисперсии: " << S << "\n";
      float autoc = 0.0;
      for(int taaay = 1; taaay < 500; taaay++)
      {
            for(int i = 0; i < N - taaay; i++)
      {
      autoc +=(float) ((numeric[i] - Ex) * (float) (numeric[i + taaay] - Ex));
      }
            autoc /= (float)(N - taaay) * S ;
      }
      cout << "Автокорреляция: " << autoc << "\n";
      return 0;
}
```

Список использованной литературы

1. http://stratum.ac.ru/education/textbooks/modelir/lection22.html
2. https://ru.wikipedia.org/wiki/Генератор_псевдослучайных_чисел
3. https://statanaliz.info/statistica/proverka-gipotez/kriterij-soglasiya-pirsona-khi-kvadrat/
4. http://www.quizful.net/post/programming-random-number-generator-on-cpp
5. https://en.cppreference.com/w/cpp/numeric/random
6. https://ravesli.com/urok-71-generatsiya-sluchajnyh-chisel-funktsii-srand-i-rand/
7. https://wiki2.org/ru/Вихрь_Мерсенна
8. http://www.quizful.net/post/random-number-generation-in-cpp11
9. https://ru.cppreference.com/w/cpp/numeric/random/poisson_distribution
10. https://mf.grsu.by/Kafedry/sp_cs/academic_process/th_inf/lect_07