**Лабораторная работа 1.**

**Условие:**

Используя метод Макларена-Марсальи построить датчик БСВ (1 датчик должен быть мультипликативный конгруэнтный, второй – на выбор). Исследовать точность построенной БСВ.

1) Осуществить моделирование n = 1000 реализаций БСВ с помощью мультипликативного конгруэнтного метода (МКМ) с параметрами a0, β, M = 231 .

2) Осуществить моделирование n = 1000 реализаций БСВ с помощью метода Макларена-Марсальи (один датчик должен быть мультипликативный конгруэнтный (п. 1), второй – на выбор).

K – объем вспомогательной таблицы.

3) Проверить точность моделирования обоих датчиков (п. 1 и п. 2) с помощью критерия согласия Колмогорова и χ2-критерия Пирсона с уровнем значимости ε = 0.05.

**Теория:**

**Мультипликативный конгруэнтный метод:**

Псевдослучайная последовательность строится по следующим рекуррентным формулам:

где - параметры датчика: - множитель (), *M* – модуль, - стартовое значение (нечетное число).

В данной работе брались значения: , .

**Метод Макларена-Марсальи:**

Пусть - псевдослучайные последовательности, порожденные независимо работающими датчиками; - результирующая псевдослучайная последовательность реализация БСВ;

*V={V(0), V(1), …,V(K-1)}* – вспомогательная таблица *K* чисел.

Процесс вычисления включает следующие этапы:

- первоначальное заполнение таблицы

- случайный выбор из таблицы:

-обновление табличных значений:

.

В данной работе в качестве бралась последовательность (из 1000 элементов), полученная мультипликативным конгруэнтным методом, описанным выше. В качестве бралась последовательности (из 1000) элементов, полученная аналогичным способом с тем же *M* и .

**- критерий согласия Пирсона:**

Область возможных значений случайной величины разбивается на интервалы

Рассматривается следующая статистика:

где *n* – объем выборки,

- количество элементов выборки, попавших в *k*-ый интервал,

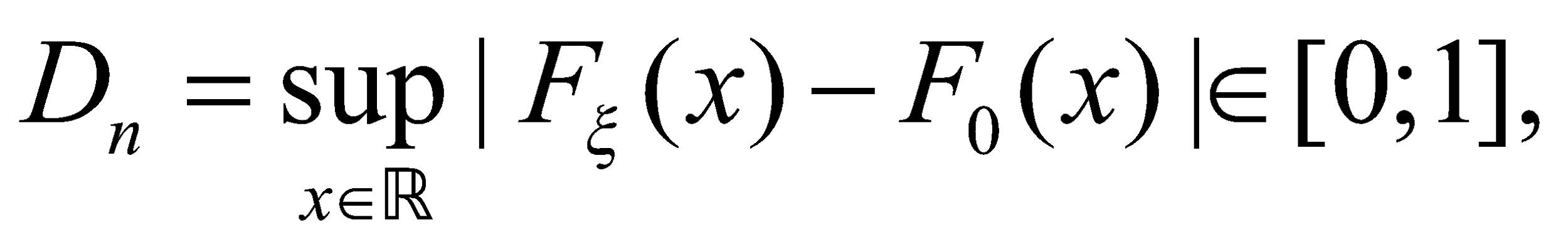
- вероятность попадания случайной величины в *k*-ый интервал.

Проверяется условие , где , - функция распределения распределения**,** - уровень значимости (обычно ).

В данной работе отрезок [0;1] разбивался на 10 интервалов.

**Критерий согласия Колмогорова:**

Рассматривается статистика:



где

Проверяется условие , *K* - функция распределенияраспределения Колмогорова**,** - уровень значимости.

**Код программы:**

#include <iostream>

#include <algorithm>

**using** **namespace** std;

**long** **long** M = 2147483648;

**long** **long** a0 = 262147;

**long** **long** a1 = 50653;

**long** **long** K = 256;

**void** MKM(**long** **double**\* sequence, **int** n, **long** **long** a, **long** **long** b) {

**long** **long** aPrev = a;

**for** (**int** i = 0; i < n; i++) {

sequence[i] = (**long** **double**)aPrev / M;

aPrev = (aPrev \* b) % M;

}

}

**void** MM(**long** **double**\* sequence, **int** n) {

**long** **double**\* bSequence = **new** **long** **double**[1000];

**long** **double**\* cSequence = **new** **long** **double**[1000];

MKM(bSequence, 1000, a0, a0);

MKM(cSequence, 1000, a1, a1);

**long** **double**\* V = **new** **long** **double**[K];

**for** (**int** i = 0; i < K; i++) {

V[i] = bSequence[i];

}

**for** (**int** i = 0; i < n; i++) {

**int** s = (**int**)(cSequence[i] \* K);

sequence[i] = V[s];

V[s] = bSequence[i + K];

}

**delete**[] bSequence, cSequence, V;

}

**void** Pirson(**long** **double**\* sequence, **int** n) {

**double** v[10] = { 0 };

**double** x = 0;

**for** (**int** i = 0; i < n; i++) {

**for** (**int** j = 9; j >= 0; j--) {

**if** (sequence[i] >= 0.1 \* j) {

v[j]++;

**break**;

}

}

}

**for** (**int** i = 0; i < 10; i++) {

x += pow((v[i] - n \* 0.1), 2) / (n \* 0.1);

}

cout << "Критерий Пирсона (порог критерия - 16.92):\n";

cout << "Результат - " << x << "\n";

}

**void** Kolmogorov(**long** **double**\* sequence, **int** n) {

**long** **double**\* sortedSequence = **new** **long** **double**[n];

**for** (**int** i = 0; i < n; i++) {

sortedSequence[i] = sequence[i];

}

sort(sortedSequence, sortedSequence + n);

**long** **double** D = 0;

**for** (**int** i = 0; i < n; i++) {

**long** **double** k = abs((**long** **double**)(i + 1) / n - sortedSequence[i]);

**if** (D < k) {

D = k;

}

}

D \*= sqrt(n);

cout << "Критерий Колмогорова (порог критерия - 1.36):\n";

cout << "Результат - " << D << "\n";

**delete**[] sortedSequence;

}

**int** main()

{

setlocale(LC\_ALL, "Russian");

**int** n = 1000;

**long** **double**\* sequence1 = **new** **long** **double**[n];

MKM(sequence1, n, a0, a0);

**long** **double**\* sequence2 = **new** **long** **double**[n];

MM(sequence2, n);

cout << "Мультипликативный конгруэнтный метод:\n";

Pirson(sequence1, n);

Kolmogorov(sequence1, n);

cout << "Метод Макларена-Марсальи:\n";

Pirson(sequence2, n);

Kolmogorov(sequence2, n);

**delete**[] sequence1, sequence2;

**return** 0;

}

**Результаты:**

Мультипликативный конгруэнтный метод:

Метод Макларена-Марсальи: