Учреждение образования

Белорусский государственный университет информатики

и радиоэлектроники

Факультет компьютерных технологий ИИТ

Лабораторная работа №1

Моделирование

Построение и исследование программного датчика равномерно-

распределенных случайных чисел.

Выполнил: студент гр.180511

Жданко М.И.

Гриневич А.А..

Проверил: преподаватель Лашкевич Е.М.

Минск, 2014

Цель:

1) Разработать программный датчик равномерно-распределенныхслучайных чисел из интервала от 0 до 1 с использованием алгоритмаЛемера;

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

3) Оценить равномерность по косвенным признакам;

4) Найти величину периода и длину участка апериодичности;

5) Варьируя параметрами алгоритма, добиться максимальной длины периода.

Немного теории:

*Непрерывная случайная величина ξ имеет равномерное распределение в интервале (a,b), если ее функции плотности fξ(x) и функция распределения Fξ(x) соответственно имеют вид:*

**

*Числовые характеристики случайной величины ξ, принимающей значения x - математическое ожидание, дисперсию и среднее квадратическое отклонение соответственно:*

**

*Такое распределение имеет математическое ожидание*

*M[ξ]=0.5 и дисперсию D[ξ]= (≈0,083), а σξ= (≈0,3).*

*Это распределение получить на цифровой ЭВМ невозможно, так как машина оперирует с n-разрядными числами. Поэтому на ЭВМ вместо непрерывной совокупности равномерных случайных чисел интервала (0,1) используют дискретную последовательность из 2n случайных чисел того же интервала. Закон распределения такой дискретной последовательности называют квазиравномерным распределением.*

1. **Смешанный метод**

Позволяет вычислить последовательность неотрицательных чисел , не превосходящих , по формуле

,

т.е. в отличие от мультипликативного метода .

С вычислительной точки зрения смешанный метод генерации сложнее мультипликативного на одну операцию сложения, но при этом возможность выбора дополнительного параметра позволяет уменьшить возможную корреляцию получаемых чисел.

Качество конкретной версии такого генератора можно оценить только с помощью соответствующего машинного эксперимента.

В настоящее время почти все библиотеки стандартных, программ универсальных ЭВМ для вычисления последовательностей равномерно распределенных случайных чисел основаны на конгруэнтной процедуре.

Вычислительный алгоритм Д.Лемера - линейно-конгруэнтный метод генерации псевдослучайных чисел.

*Xi+1 = (aXi + c) mod m*

*a*,*c,m* — некоторые положительныецелые числа 

Качество РРСЧ, весьма существенно зависит от выбора *a*, *c, m*,*X0* где параметры  и*X0* влияют на статистические свойства получаемых чисел, а параметр - на период их повторения.

Значение переменной *X0* должно быть:

а) меньше;

в) достаточно большим;

г) желательно простым числом;

г) содержать в двоичном представлена сравнительно большее число единиц.

В таблице ниже приведены наиболее часто используемые параметры линейных конгруэнтных генераторов, в частности, в стандартных библиотеках различных компиляторов (функция rand()).

При реализации выгодно выбирать M = 2n, где n — число битов в машинном слове, поскольку это позволяет избавиться от относительно медленной операции приведения по модулю.

Приведенное соотношение имеет следующий смысл: *Xi+1* равно остатку от деления *(aXi + c)* на *m*.

Запишем алгоритм в виде пошаговой процедуры.

Ш а г 1. Коэффициент *a* умножается на число *Xi*

Ш а г 2. К результату умножения (*aXi*) прибавляется числ*с*

Ш а г 3. Результат сложения *(aXi + c)*  делится на *m*

*aXi + c = q·m+ Xi+1*

где *qm* - целая часть (*q* = 0,1,2,...), *Xi+1* - остаток от деления .

Ш а г 4. Остаток от деления *Xi+1* делится на *m*, чтобы получить искомое случайное число между нулем и единицей:

Следует отметить, что данный метод не обладает криптографической стойкостью, однако входит в большинство современных стандартных библиотек различных компиляторов.

Пример.

х0= 7(0111)

а = 5 (0101)

с = 3 (0011)

m = 2n

n – число бит в машинном слове

Пусть n = 4, тогда m = 24 = 16

*х1* = (*a·х0 + c) modm* = (5·7 + 3)*mod*16 = 38 (00100110) *mod*16 = 6 (0110)

*х1\**= 6 /16 = 0,375

*х2* =*a·х1 + c* = 5·6 + 3 = 33

*х2* = (*a·х1 + c) modm* = (5·6 + 3)*mod*16 = 33 (00100001) *mod*16 = 1 (0001)

*х2\**= 1 /16 = 0,063

**……………….**

**Математическое ожидание** квазиравномерной случайной величины:



N – количество элементов массива, – элемент нашей последовательности.



**Дисперсия квазиравномерной случайной величины - мера разброса случайной величины, т.е. её отклонение от математического ожидания.**



N – количество элементов массива, – математическое ожидание, – элемент нашей последовательности.



СКО квазиравномерной случайной величины - показатель рассеивания значений случайной величины относительно её математического ожидания.



Эффективность статистического моделирования систем на ЭВМ и достоверность получаемых результатов существенным образом зависят от качества исходных (базовых) последовательностей псевдослучайных чисел, которые являются основой для получения стохастических воздействий на элементы моделируемой системы.

Проверка качества последовательностей РРСЧ

1) Проверка равномерности

2) Проверка стохастичности

3) Проверка независимости

Проверка равномерности последовательностей псевдослучайных квазиравномерно распределенных чисел {хi} может быть выполнена по гистограмме или с использованием косвенных признаков.

а) Проверка по гистограмме.

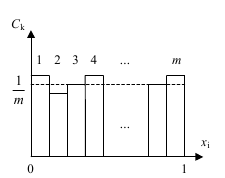
Суть проверки по гистограмме сводится к следующему. Выдвигается гипотеза о равномерности распределения чиел в интервале (0 1). Затем интервал (0 1) разбивается на m равных частей. При генерации последовательности РРСЧ подсчитывается количество попаданий Nk в каждый из m подинтервалов. Вычисляется относительная частота попадания случайных чисел последовательности {хi} в каждый из подинтервалов

Ck= Nk/N,

где N − общее количество чисел в последовательности {хi}.

Очевидно, что при равномерности последовательности чисел, частоты должны быть близкими при достаточно больших N к теоретической вероятности попадания в подинтервалы, равной 1/m.

Оценка степени приближения, т. е. равномерности последовательности{хi}, может быть проведена с использованием критериев согласия. На практике обычно принимается m = 20÷50, N = (102÷103)*m*.



б) Проверка по косвенным признакам.

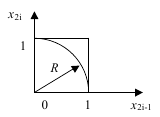
Вся последовательность {хi} разбивается на пары чисел:

(x1, x2), (x3, x4), ... , (x2i-1, x2i), ... , (xN-1, xN).

Затем подсчитывают число пар K, для которых выполняется условие:

<1

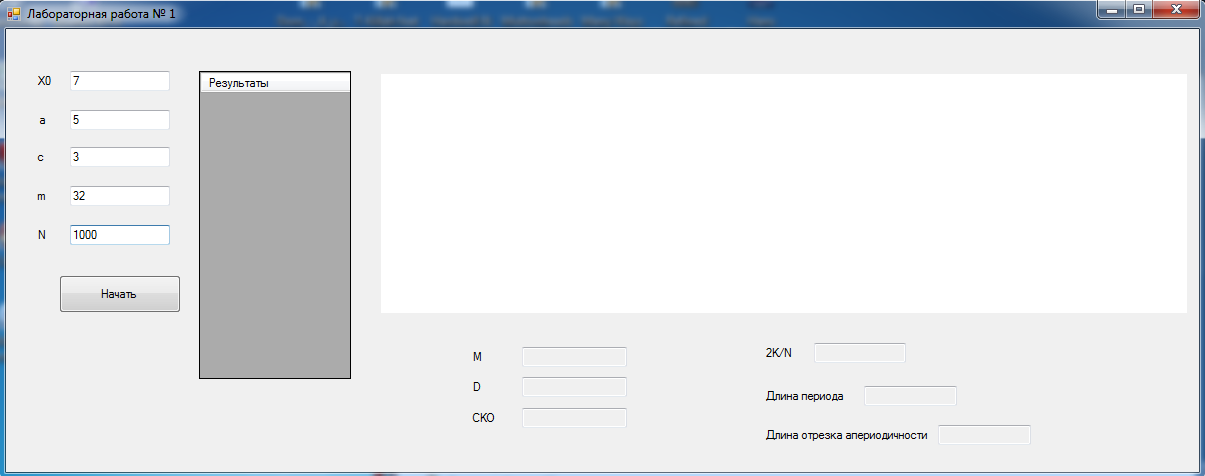
Геометрически это означает, что точка с координатами (x2i-1, x2i)расположена внутри четверти круга радиуса R=1, вписанного в единичный квадрат.



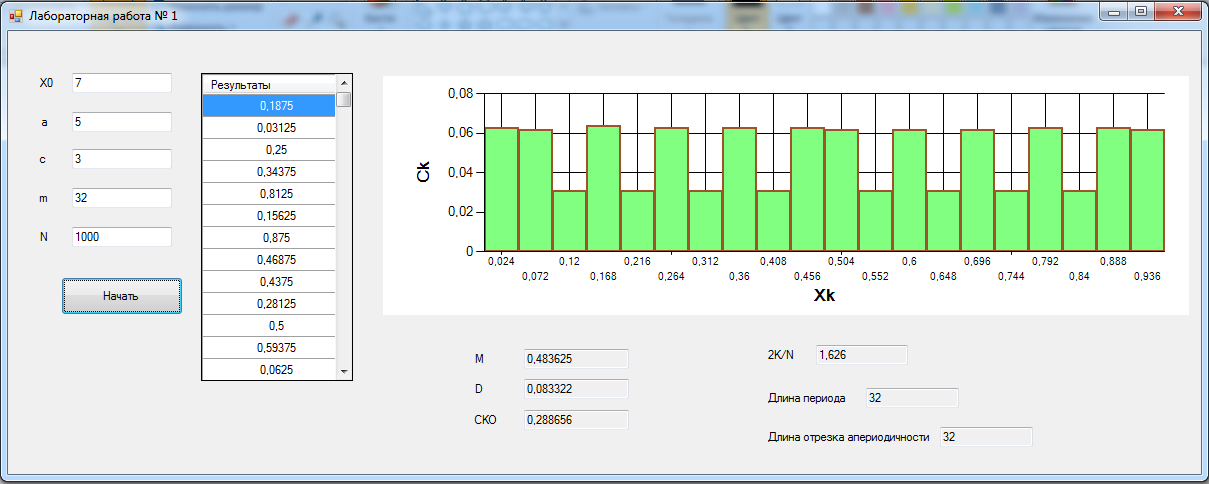
В общем случае точка (x2i-1, x2i) всегда попадет внутрь единичногоквадрата. Тогда теоретическая вероятность попадания этой точки в четвертькруга равна отношению площади четверти круга к площади единичногоквадрата:

P = S1/4 круга/Sквадрата = π/4.

Исходные данные:



Результат:



***Листингпрограммы:***

namespace Lab1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

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

{

int X = Convert.ToInt32(textBox1.Text);

int a = Convert.ToInt32(textBox2.Text);

int c = Convert.ToInt32(textBox3.Text);

int m = Convert.ToInt32(textBox4.Text);

int N = Convert.ToInt32(textBox5.Text);

double[] result=new double[N];

double temp=X;

//Рассчет значений Xi

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

{

double f = (a \* temp + c) % m;

result[i] = f / m;

temp = f;

}

//Выведение результатов из массива на экран

dataGridView1.RowCount = N;

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

{

dataGridView1.Rows[i].Cells[0].Value = result[i];

}

//Расчет мат.ожидания,дисперсии и СКО

double mat = 0, disp = 0, sko = 0;

mat = Math.Round(((1.0/(double)N)\*result.Sum()),6);

textBox6.Text = mat.ToString();

//-------------------------------------------------

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

{

disp += Math.Pow((result[q] - mat), 2);

}

disp = Math.Round(((1.0 / (double)N) \* disp),6);

textBox7.Text = disp.ToString();

//-------------------------------------------------

sko = Math.Round(Math.Sqrt(disp),6);

textBox8.Text = sko.ToString();

//-------------------------------------------------

#region Построение гистограммы

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

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);

#endregion

//Равномерность косвенная

int K = 0;

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

{

if((Math.Pow(result[i-1],2)+Math.Pow(result[i],2))<1)

{

K += 1;

}

}

double kosres=(2.0 \* (double)K) / (double)N;

textBox11.Text = kosres.ToString();

//Определение длины периода

double V = result[N - 1];

int i1=0, i2=0;

int flag=0;

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

{

if(result[v]==V)

{

if (flag == 0)

{

flag = 1;

i1 = v;

}

else { i2 = v; break; }

}

}

int P = i2 - i1;

textBox9.Text = P.ToString();

//Определение длины отрезка апериодичности

int i3 = 0;

while (result[i3] != result[i3 + P])

{

i3++;

}

int aperiod = i3 + P;

textBox10.Text = aperiod.ToString();

}

}

}