Contents

General Linear Congruence Generator

Abhinav Gupta

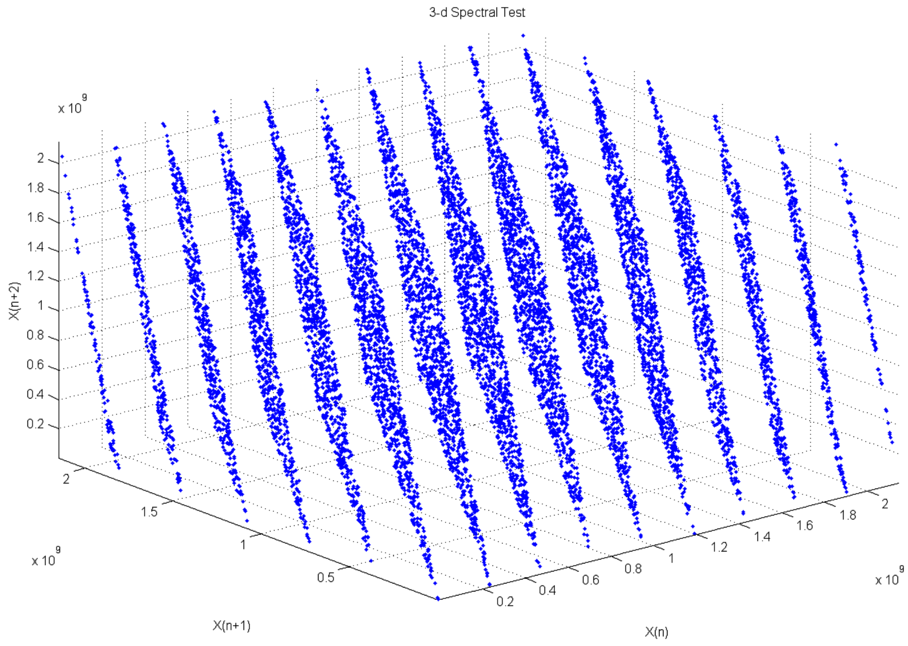
January 25, 2017

1 Short Summary

General Linear Congruence Generator

$$x_{i+1} = (ax_i + b) mod m$$

$$u_{i+1} = x_{i+1}/m$$



x_i is a sequence of pseudo-random values
m is **modulus**
a is **multiplier**
b is **increment**
x_0 is **"Seed" or start value**

A linear congruential generator (LCG) is an algorithm that yields a sequence of pseudo-randomized numbers calculated with a discontinuous piecewise linear equation. The method represents one of the oldest and best-known pseudo-random number generator algorithms. The theory behind them is relatively easy to understand, and they are easily implemented and fast, especially on computer hardware which can provide modulo arithmetic by storage-bit truncation.

2 Question-1

When a=3, b=0 and m=11 $--->$ 5 different values are generated.
When a=6,b=0 and m=11 $--->$ 10 different values are generated.
The reason is as regardless of what **'seed'** we choose same values start getting generated when $x_{i+1} = seed$. So after repetitive modulus of '6' and '3', the following sequences is formed :

$$6\ 3\ 7\ 9\ 10\ 5\ 8\ 4\ 2\ 1\ 6$$

$$3\ 9\ 5\ 4\ 1\ 3$$

So to generate a sequence of random numbers it would be better to use a=6,b=0 and m=11 data-set as they generate more number of different numbers.

- Problems:
  1– Cycle is mostly shorter due to small m.
  2– Generating not much randomness in numbers.

- **"C++ Code is as follows: "**

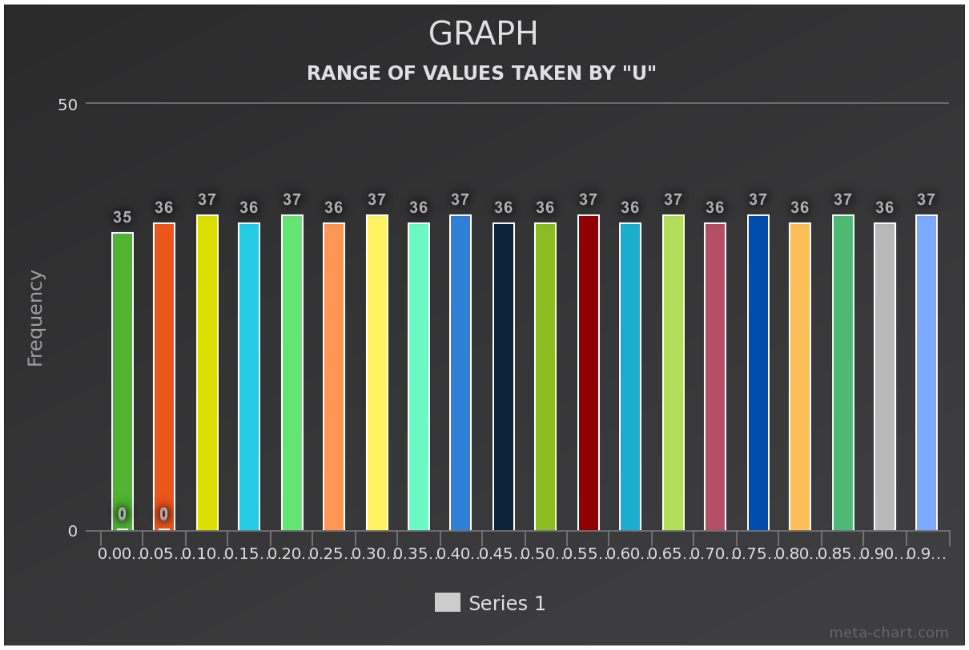
```
#include <math.h>
#include <iostream>
#include <stdlib.h>
#include <vector>
#include <bits/stdc++.h>
using namespace std;
        vector<int> array;
int random_number_generator(int a,int b,int m){
        int i=0,cnt=1;
        int value=(a*array[i++]+b)%m;
        while(!(find(array.begin(),array.end(),value)!=array.end())) {
                array.push_back(value);
                value=(a*array[i++]+b)%m;
                cnt++;
        }
        array.push_back(value);
        cnt++;

        return (cnt-1);
}
```

```
int main(){
        int x0;
        x0=rand()%10;
        array.push_back(x0);
        int cnt=random_number_generator(6,0,11);
        vector<int>::iterator i;
        for(i=array.begin();cnt--;i++){
                cout<<(*i)<<endl;}
}
```

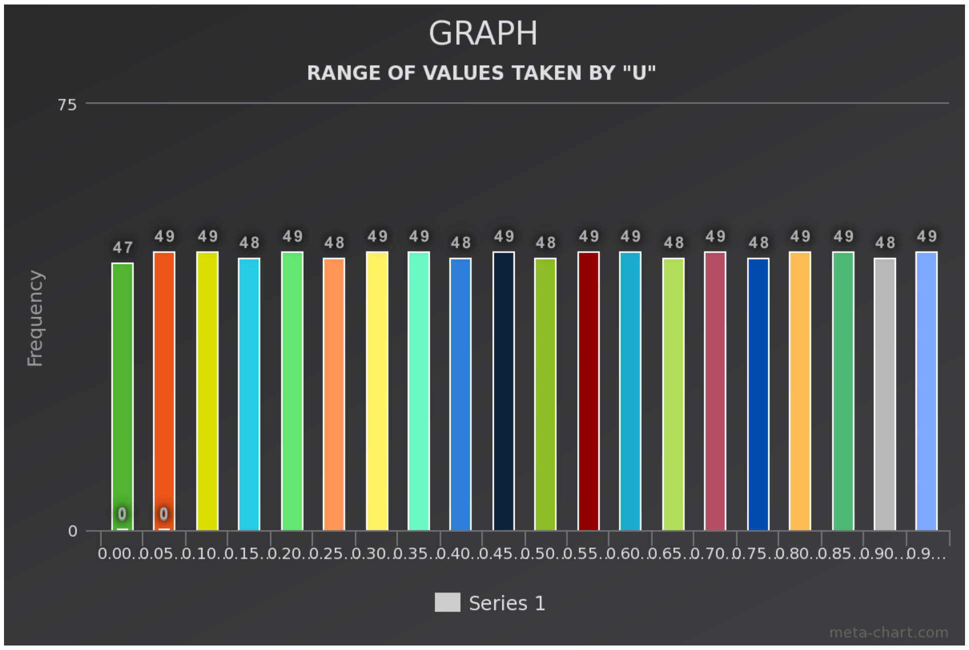
3 Question-2

"Generated bar-graph for frequencies of values falling within different-different ranges : "

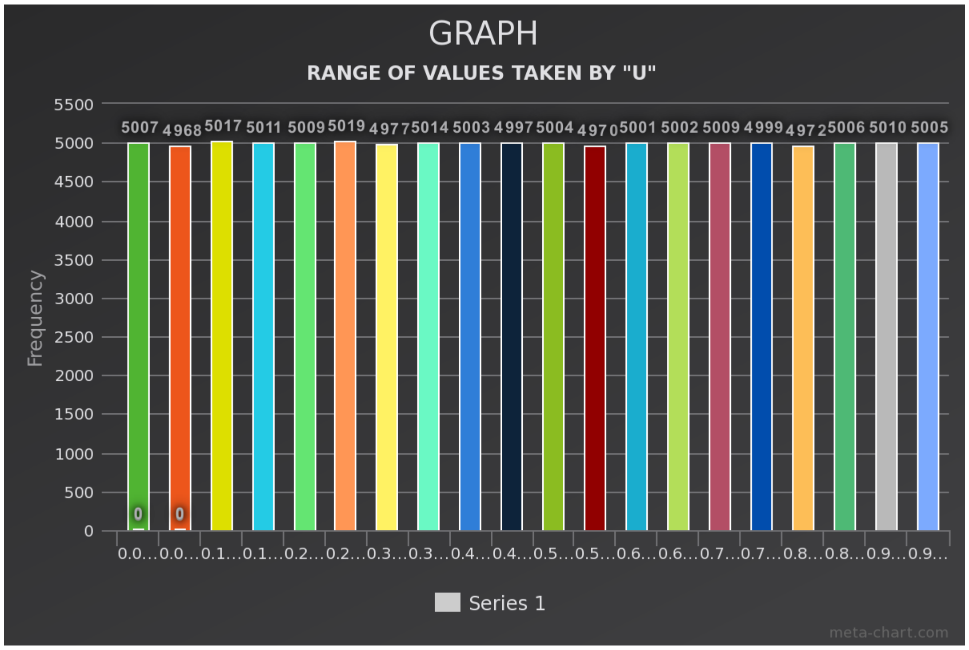
- Observations: In Linear Congruence Generator is look-alike uniform as probability of finding a random number generated in a range is almost equal for all ranges (i.e. random values generated are uniformly generated over the range modulus "m")
- a=1597 and seed=1000 (all different values):



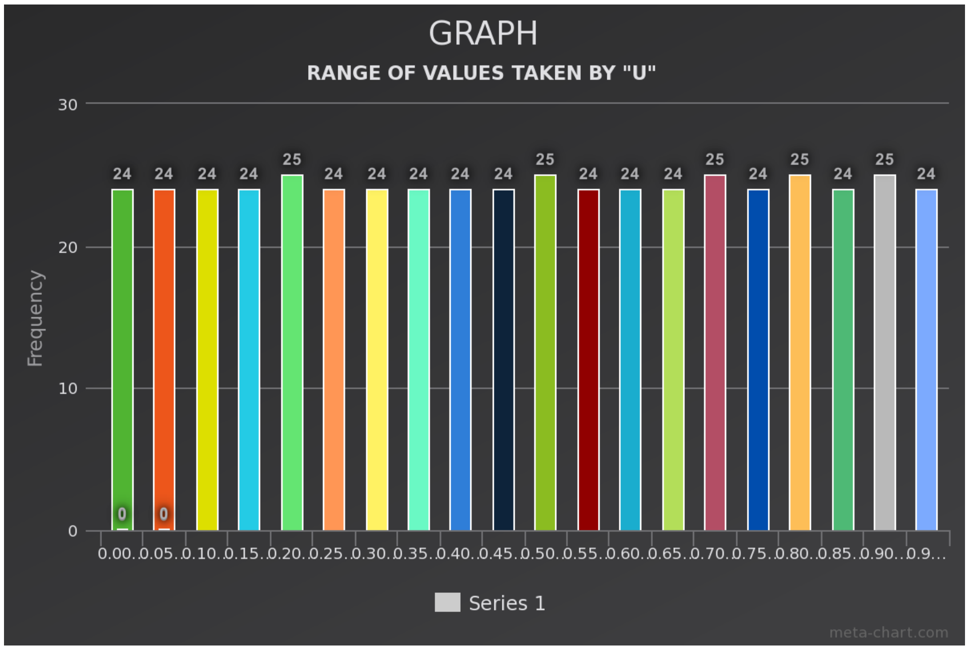
- a=1597 and seed=10005 (all different values):



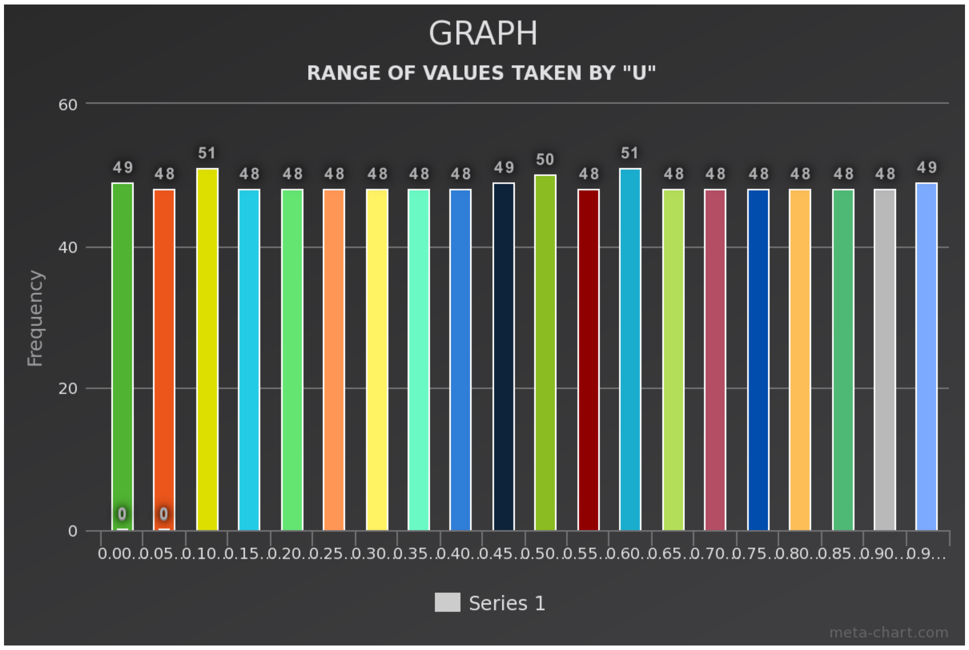
- a=1597 , output- 100000 values (same as well as different), seed=77177:



- a=51749 and seed=2:



- a=51749 and seed=359:

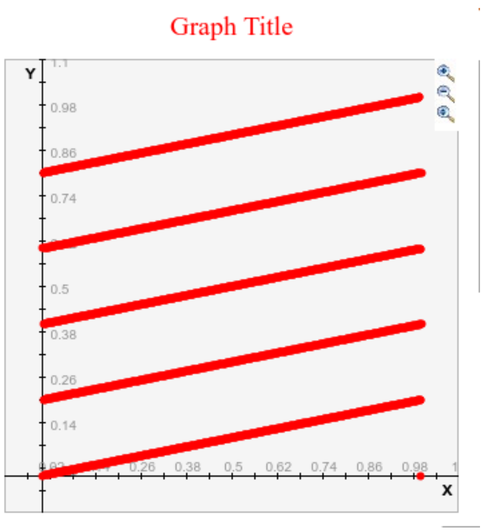


- **”C++ code is as follows : ”**

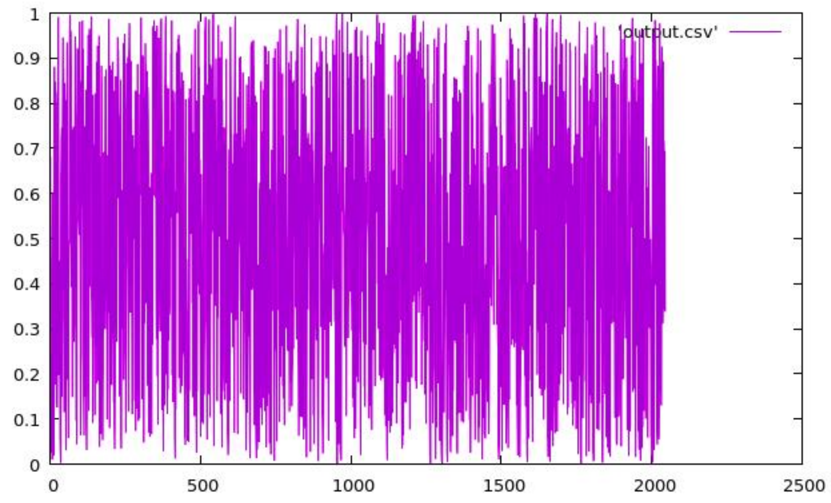
```
#include <math.h>
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <vector>
#include <algorithm>
using namespace std;
int cnt[21]={0};
vector<float> u;
vector<long long int> array;
void random_number_generator(int seed,int a,int b,int m){
        int i=0,k;
        long long int y=(a*array[i++]+b);
        int x=y%m;
        float value=(float)(x)/m;
        while(1){
                if(!(find(array.begin(),array.end(),x)!=array.end()))
                        u.push_back(value);
                else
                        return;
                array.push_back(x);
                float j=.05;
                for(k=1;k<=20;k++){
                        if(value<j){
                                cnt[k]++;
                                break;
                        }
                        j+=.05;
                }
                y=(a*array[i++]+b);
                x=y%m;
                value=(float)(x)/m;
        }
}

int main(){
//      cnt[0]=100000;
        int x0=359;
        array.push_back(x0);
        random_number_generator(x0,51749,0,244944);
        vector<float>::iterator it;
        double j=0;
        for(int i=1;i<=20;i++,j+=.05)
                printf("%g-%g,%d\n",j,j+.05,cnt[i]);
}
```

4 Question-3

- Observation:
  For the Linear Congruence Generator to be truly a **"Uniform Distribution"**, the distribution of graph points would have been uniformly scattered over the unit square (or cube) but that is not the case here as points are lying over some specific hyperplanes and are not scattered.
- Scattered Plot:



- GnuPlot



- **"C++ code is as follows : "**

```
#include <math.h>
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <vector>
#include <algorithm>
using namespace std;
int cnt[21]={0};
vector<float> u;
vector<int> array;
void random_number_generator(int seed,int a,int b,int m){
        int i=0,k;
        long long int y=(a*array[i++]+b);
        int x=y%m;
        float value=(float)(x)/m;
        while(1){
                if(!(find(array.begin(),array.end(),x)!=array.end()))
                        u.push_back(value);
                else
                        return;
                array.push_back(x);
                float j=.05;
                for(k=1;k<=20;k++){
                        if(value<j){
                                cnt[k]++;
                                break;
                        }
                        j+=.05;
                }
                y=(a*array[i++]+b);
                x=y%m;
                value=(float)(x)/m;
        }
}

int main(){
//        cnt[0]=100000;
        int x0=2;
        array.push_back(x0);
        random_number_generator(x0,1229,1,2048);
        vector<float>::iterator it;
        double j=0;
        for(it=u.begin();it<(u.end()-1);it++)
                printf("%f,%f\n",*(it),*(it+1));}
```