Linear Congruence Generator

Ayush Sharma

[150123046]

ayush.2015@iitg.ac.in

Instructor: Arabin Kumar Dey

Table of Contents

Abstract

A linear congruential generator (LCG) is an algorithm that yields a sequence of pseudo-randomized numbers calculated with a discontinuous piecewise linear equation. The generator is defined by the recurrence relation:

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

$$u_{i+1} = \frac{x_{i+1}}{m}$$

where u is the sequence of pseudorandom values, and

$m, 0 < m$ - the "modulus"

$a, 0 < a < m$ - the "multiplier"

$b, 0 \leq b < m$ - the "increment"

$x_0, 0 \leq x_0 < m$ - the "seed" or "start value"

are integer constants that specify the generator.

If $b = 0$, the generator is often called a multiplicative congruential generator (MCG), or Lehmer RNG. If $b \neq 0$, the method is called a mixed congruential generator.

Problem Statement 1

Generate the sequence of numbers x_i for a = 6, b = 0, m = 11 and x_0 ranging from 0 to10. Also, generate the sequence of numbers x_i for a = 3, b = 0, m = 11, and x_0 ranging from 0 to 10.

Observe the sequence of numbers generated and observe the repetition of values. Tabulate these for each group of values.

How many distinct values are appearing before repetitions? Which, in your view, are the best choices and why?

(Use only C or C++ code)

Program

Code in C

```c
#include<stdio.h>
void gen(int a, int b, int m, int x0, int * value, int v){
	int x = x0, i = 0;
	printf("[");
	do{
		printf(" %f",(x/(float)m));
		x = (((a * x) + b) % m);
		i += 1;
	}while(x != x0);
	printf(" ]\tRepetition after %d terms.\n", i);
	value[v] = i;
}
void main(){
	int x0, value[22] = {0}, v = 0;
	for(x0 = 0; x0 <= 10; x0++){
		printf("For a = 6, b = 0, m = 11, x0 = %d :\n\t",x0);
		gen(6, 0, 11, x0, value, v);
		v++;
	}
	for(x0 = 0; x0 <= 10; x0++){
		printf("For a = 3, b = 0, m = 11, x0 = %d :\n\t",x0);
		gen(3, 0, 11, x0, value, v);
		v++;
	}

	int max = 0;
	for(v = 0; v < 22; v++){
		if (max < value[v]){
			max = value[v];
		}
	}

	printf("The best choice/choices are (generating maximum, i.e. %d,terms without repetition):\n", max);

	for(v = 0; v < 22; v++){
		if (max == value[v]){
			if (v < 11){
				printf("\ta = 6, b = 0, m = 11, x0 = %d\n",(v));
			}
			else{
				printf("\ta = 3, b = 0, m = 11, x0 = %d\n",(v-11));
			}
		}
	}
}
```

Output

```
For a = 6, b = 0, m = 11, x0 = 0 :
      [ 0.000000 ]      Repetition after 1 terms.
For a = 6, b = 0, m = 11, x0 = 1 :
      [ 0.090909 0.545455 0.272727 0.636364 0.818182 0.909091 0.454545 0.727273
0.363636 0.181818 ]     Repetition after 10 terms.
For a = 6, b = 0, m = 11, x0 = 2 :
      [ 0.181818 0.090909 0.545455 0.272727 0.636364 0.818182 0.909091 0.454545
0.727273 0.363636 ]     Repetition after 10 terms.
For a = 6, b = 0, m = 11, x0 = 3 :
      [ 0.272727 0.636364 0.818182 0.909091 0.454545 0.727273 0.363636 0.181818
0.090909 0.545455 ]     Repetition after 10 terms.
For a = 6, b = 0, m = 11, x0 = 4 :
      [ 0.363636 0.181818 0.090909 0.545455 0.272727 0.636364 0.818182 0.909091
0.454545 0.727273 ]     Repetition after 10 terms.
For a = 6, b = 0, m = 11, x0 = 5 :
      [ 0.454545 0.727273 0.363636 0.181818 0.090909 0.545455 0.272727 0.636364
0.818182 0.909091 ]     Repetition after 10 terms.
For a = 6, b = 0, m = 11, x0 = 6 :
      [ 0.545455 0.272727 0.636364 0.818182 0.909091 0.454545 0.727273 0.363636
0.181818 0.090909 ]     Repetition after 10 terms.
For a = 6, b = 0, m = 11, x0 = 7 :
      [ 0.636364 0.818182 0.909091 0.454545 0.727273 0.363636 0.181818 0.090909
0.545455 0.272727 ]     Repetition after 10 terms.
For a = 6, b = 0, m = 11, x0 = 8 :
      [ 0.727273 0.363636 0.181818 0.090909 0.545455 0.272727 0.636364 0.818182
0.909091 0.454545 ]     Repetition after 10 terms.
For a = 6, b = 0, m = 11, x0 = 9 :
      [ 0.818182 0.909091 0.454545 0.727273 0.363636 0.181818 0.090909 0.545455
0.272727 0.636364 ]     Repetition after 10 terms.
For a = 6, b = 0, m = 11, x0 = 10 :
      [ 0.909091 0.454545 0.727273 0.363636 0.181818 0.090909 0.545455 0.272727
0.636364 0.818182 ]     Repetition after 10 terms.
For a = 3, b = 0, m = 11, x0 = 0 :
      [ 0.000000 ]      Repetition after 1 terms.
For a = 3, b = 0, m = 11, x0 = 1 :
      [ 0.090909 0.272727 0.818182 0.454545 0.363636 ]      Repetition  after  5
terms.
For a = 3, b = 0, m = 11, x0 = 2 :
      [ 0.181818 0.545455 0.636364 0.909091 0.727273 ]      Repetition  after  5
terms.
For a = 3, b = 0, m = 11, x0 = 3 :
      [ 0.272727 0.818182 0.454545 0.363636 0.090909 ]      Repetition  after  5
terms.
For a = 3, b = 0, m = 11, x0 = 4 :
      [ 0.363636 0.090909 0.272727 0.818182 0.454545 ]      Repetition  after  5
terms.
For a = 3, b = 0, m = 11, x0 = 5 :
      [ 0.454545 0.363636 0.090909 0.272727 0.818182 ]      Repetition  after  5
terms.
For a = 3, b = 0, m = 11, x0 = 6 :
      [ 0.545455 0.636364 0.909091 0.727273 0.181818 ]      Repetition  after  5
terms.
For a = 3, b = 0, m = 11, x0 = 7 :
```

```
      [ 0.636364 0.909091 0.727273 0.181818 0.545455 ]      Repetition  after  5
terms.
For a = 3, b = 0, m = 11, x0 = 8 :
      [ 0.727273 0.181818 0.545455 0.636364 0.909091 ]      Repetition  after  5
terms.
For a = 3, b = 0, m = 11, x0 = 9 :
      [ 0.818182 0.454545 0.363636 0.090909 0.272727 ]      Repetition  after  5
terms.
For a = 3, b = 0, m = 11, x0 = 10 :
      [ 0.909091 0.727273 0.181818 0.545455 0.636364 ]      Repetition  after  5
terms.
The  best  choice/choices  are  (generating  maximum,  i.e.  10,terms  without
repetition):
      a = 6, b = 0, m = 11, x0 = 1
      a = 6, b = 0, m = 11, x0 = 2
      a = 6, b = 0, m = 11, x0 = 3
      a = 6, b = 0, m = 11, x0 = 4
      a = 6, b = 0, m = 11, x0 = 5
      a = 6, b = 0, m = 11, x0 = 6
      a = 6, b = 0, m = 11, x0 = 7
      a = 6, b = 0, m = 11, x0 = 8
      a = 6, b = 0, m = 11, x0 = 9
      a = 6, b = 0, m = 11, x0 = 10
```

Problem Statement 2

Generate a sequence u_i with m = 244944, a = 1597, b = 51749 (take x_0 as per your choice). Try to group the values in the ranges 0 - 0.05, 0.05 - 0.10, 0.10 - 0.15, ... and see their frequencies (i.e. the number of values falling in a group).

For at least 5 different values of the number of values generated, tabulate the frequencies in each case, draw bar diagrams of these data and put in your observations.

(Use both R and C / C ++ code)

Program

Code in C

```
#include<stdio.h>
void gen(int x0, int * f){
      int x = x0, i;
      do{
            x = (((1597 * x) + 51749) % 244944);
            i = ((x/(float)244944)/0.05);
            if (i == 20) {i--;}
            *(f + i) += 1;
      }while(x != x0);
      for(i = 0; i < 20; i++){
            *(f + 20) += *(f + i);
      }
}
void main(){
      int x0[5], f[5][21] = {0}, i;
      for(i = 0; i < 5; i++){
            printf("x0_%d ? ", (i+1));
            scanf("%d", &x0[i]);
            gen(x0[i], *(f + i));
      }
      printf("With a = 1597, b = 51749, m = 244944 :\n");
      for(i = 0; i < 5; i++){
            printf("And, with the value of x0 = %d, the frequency distribution
is:\n", x0[i]);
            for(int j = 0; j < 20; j++){
                  printf("\t%.2f-%.2f\t:\t%d\n",    (j/(float)20),    ((j    +
1)/(float)20), f[i][j]);
            }
            printf("Total numbers : %d\n\n", f[i][20]);
      }
}
```

Output

Table

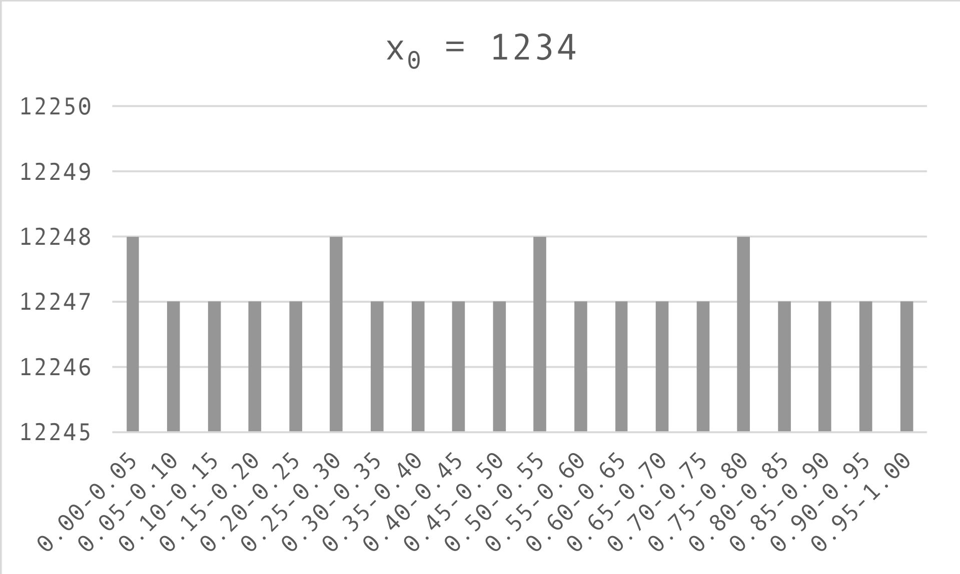
With the values of a = 1597, b = 51749 and m = 244944, the frequency distributions at different values of x_0 are:

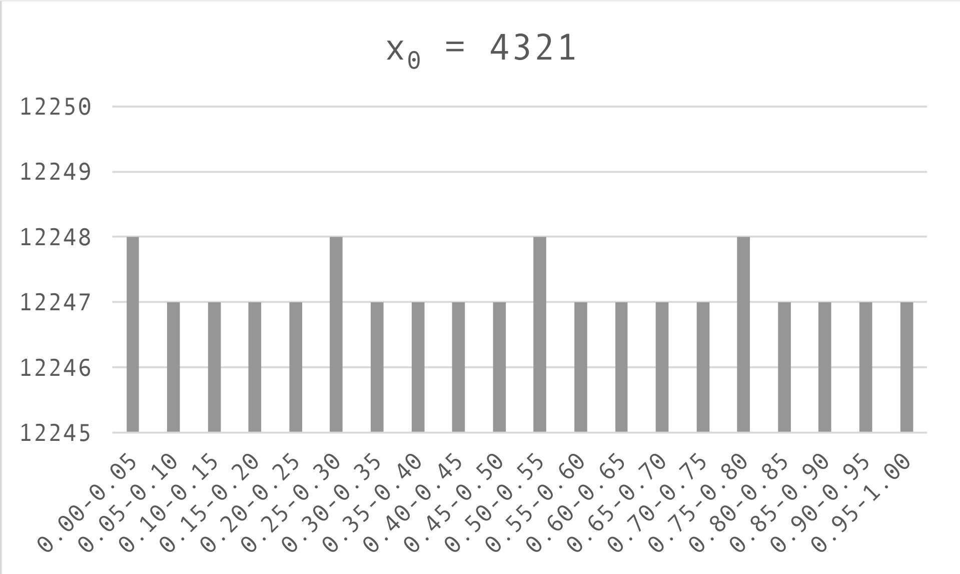
x_0 → Range ↓	1234	4321	12345	54321	123456
0.00 - 0.05	12248	12248	12248	12248	12248
0.05 - 0.10	12247	12247	12247	12247	12247
0.10 - 0.15	12247	12247	12247	12247	12247
0.15 - 0.20	12247	12247	12247	12247	12247
0.20 - 0.25	12247	12247	12247	12247	12247
0.25 - 0.30	12248	12248	12248	12248	12248
0.30 - 0.35	12247	12247	12247	12247	12247
0.35 - 0.40	12247	12247	12247	12247	12247
0.40 - 0.45	12247	12247	12247	12247	12247
0.45 - 0.50	12247	12247	12247	12247	12247
0.50 - 0.55	12248	12248	12248	12248	12248
0.55 - 0.60	12247	12247	12247	12247	12247
0.60 - 0.65	12247	12247	12247	12247	12247
0.65 - 0.70	12247	12247	12247	12247	12247
0.70 - 0.75	12247	12247	12247	12247	12247
0.75 - 0.80	12248	12248	12248	12248	12248
0.80 - 0.85	12247	12247	12247	12247	12247
0.85 - 0.90	12247	12247	12247	12247	12247
0.90 - 0.95	12247	12247	12247	12247	12247
0.95 - 1.00	12247	12247	12247	12247	12247
Total	244944	244944	244944	244944	244944

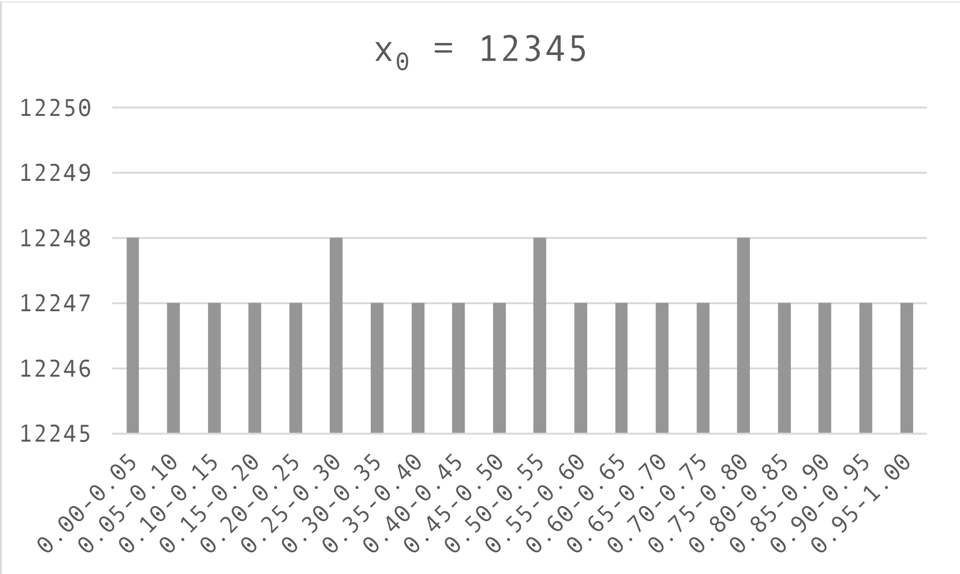
*The frequency distributions at all the taken values of x_0 are same.

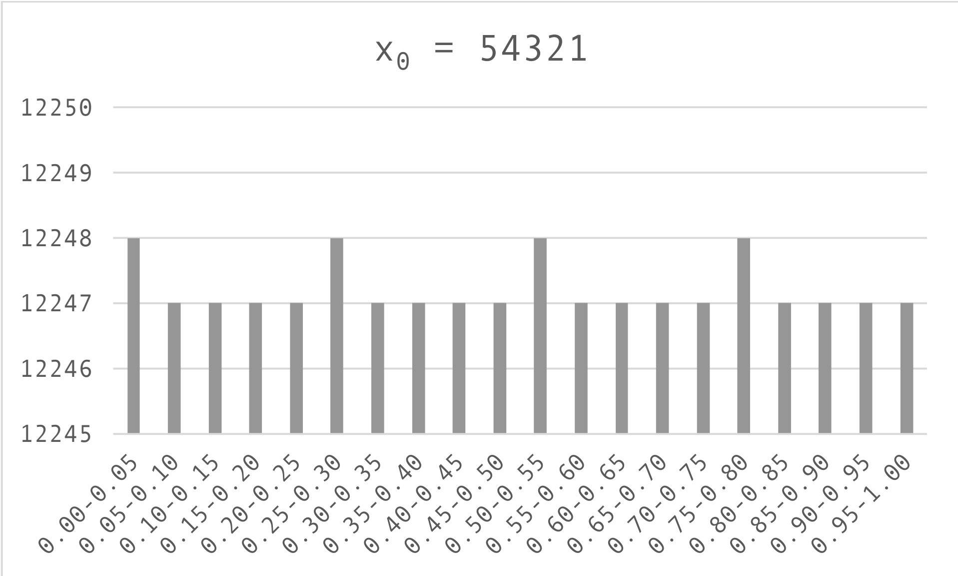
Bar-Graphs

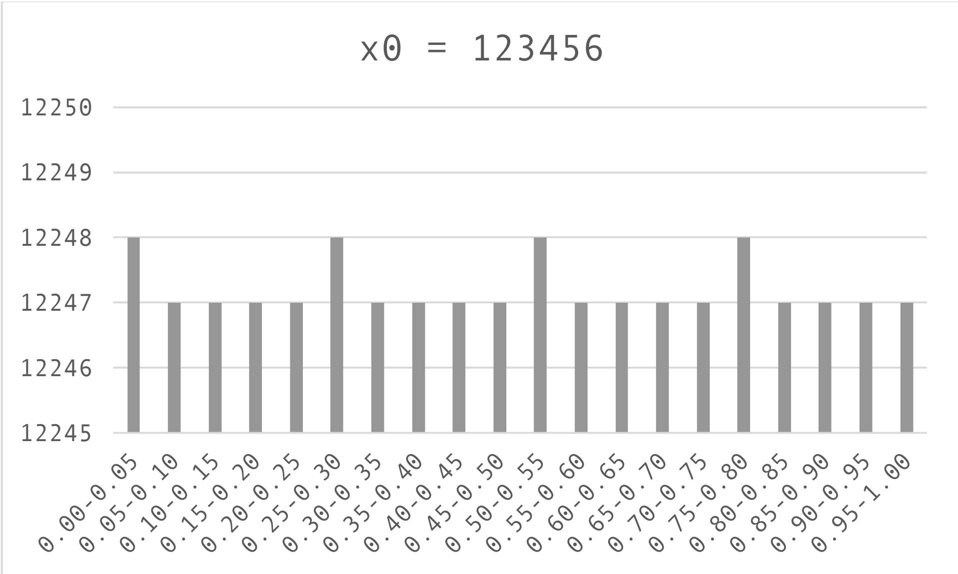
With the values of a = 1597, b = 51749 and m = 244944, the bar-diagrams/bar-graphs of the frequency distributions at different values of x_0 are:


x_0 = 1234
12250
12249
12248
12247
12246
12245
0.00-0.05
0.05-0.10
0.10-0.15
0.15-0.20
0.20-0.25
0.25-0.30
0.30-0.35
0.35-0.40
0.40-0.45
0.45-0.50
0.50-0.55
0.55-0.60
0.60-0.65
0.65-0.70
0.70-0.75
0.75-0.80
0.80-0.85
0.85-0.90
0.90-0.95
0.95-1.00


x_0 = 4321
12250
12249
12248
12247
12246
12245
0.00-0.05
0.05-0.10
0.10-0.15
0.15-0.20
0.20-0.25
0.25-0.30
0.30-0.35
0.35-0.40
0.40-0.45
0.45-0.50
0.50-0.55
0.55-0.60
0.60-0.65
0.65-0.70
0.70-0.75
0.75-0.80
0.80-0.85
0.85-0.90
0.90-0.95
0.95-1.00


x_0 = 12345
12250
12249
12248
12247
12246
12245
0.00-0.05
0.05-0.10
0.10-0.15
0.15-0.20
0.20-0.25
0.25-0.30
0.30-0.35
0.35-0.40
0.40-0.45
0.45-0.50
0.50-0.55
0.55-0.60
0.60-0.65
0.65-0.70
0.70-0.75
0.75-0.80
0.80-0.85
0.85-0.90
0.90-0.95
0.95-1.00





*The bar-diagrams/bar-graphs of the frequency distributions at all the taken values of x_0 are same, since, the frequency distributions at all the taken values of x_0 are same.

Observation

The numbers generated are distributed evenly (almost) across the range [0,1), as observed in the tables of frequency distributions and their bar-diagrams/bar-graphs.

Problem Statement 3

Generate a sequence u_i with a = 1229, b = 1, m = 2048. Plot in a two-dimensional graph the points (u_{i-1}, u_i), i.e., the points (u_1, u_2), (u_2, u_3), (u_3, u_4), · · · What are your observations?

Program

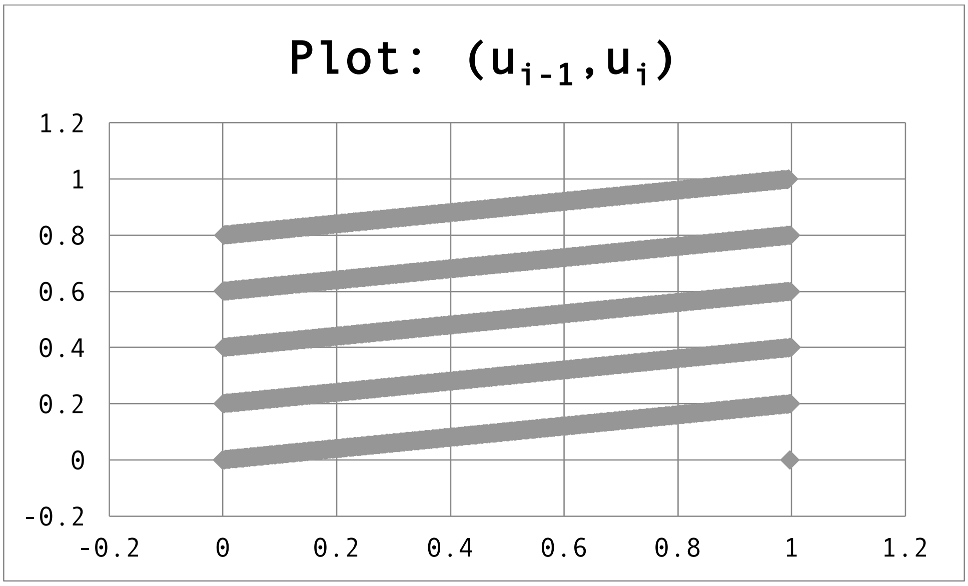
Code in C

```
#include<stdio.h>
void main(){
	int x0, x;
	float u, u1;
	printf("x0 ? ");
	scanf("%d", &x0);
	x = x0;
	u1 = (x/(float)2048);
	do{
		u = u1;
		x = (((1229 * x) + 1) % 2048);
		u1 = (x/(float)2048);
		printf("(%g,%g)\t", u, u1);
	}while(x != x0);
	printf("\n");
}
```

Output

Plot

With the values of a = 1229, b = 1 m = 2048, the plot of the sequence (u_{i-1}, u_i) is:



Observation

The sequence of the numbers u_i generated are pseudorandom instead of random.

If they were truly i.i.d. Uniform(0,1) we’d see points randomly dispersed in the unit square, in the plot of sequence (u_{i-1}, u_i). But instead the points fall entirely on some definite region (lines).