Assignment II (MA226)

Name: Sourav Bikash

Roll No.:11012338

Submission Date: 18/01/2013 Time:23:59 hrs.

Aim of the Problem:

The problem involves the use of a linear congruence generator. The basic aim lies in the fact that although it generates a large number of values, the values are not randomly distributed. And there exists a definition to these distributions.

Mathematical Analysis/Theory:

The problem uses the following linear congruence generator:

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

It generates a sequence of x_i and a dependent sequence u_i.

Part I:

We first generate a sequence of x_is. We see that the values are cyclic and so we obtain each of these cycles for some specific values of x_0.(in this case 1 ->10)
Using values: (a,b,m) as (6,0,11) and (3,0,11) we obtain a sequence of x_i.

Implementation using C++:

```
#include<iostream>
using namespace std;
int main()
{
	int i;int x,x1;int initial; // initializing variables
	for(i=0;i<11;i++)// taking valyes from 1 to 10
	{
		cout<<"x0="<<i<<"\n";
		initial=(6*i)%11;// for a=6,b=0,m=11
		x=initial;
		do
		{
			cout<<x<<",";
			x=(6*x)%11;

		}while(x!=initial);//printing each cycle as it repeats itself
		cout<<"\n----------------------"<<"\n";
	}cout<<"----------------------"<<"\n";
	x1=0;x=0;
	for(i=0;i<11;i++)
	{
		cout<<"x0="<<i<<"\n";
		initial=(3*i)%11;//for a=3,b=0,m=11
		x=initial;
		do
		{
```

```
                        cout<<x<<",";
                        x=(3*x)%11;

                }while(x!=initial);
                cout<<"\n----------------------"<<"\n";
        }
        return 0;
}
```

The following output was obtained:

For a=6, b=0,m=11

X_0	1	2	3	4	5	6	7	8	9	10
1	6	3	7	9	10	5	8	4	2	1
2	1	6	3	7	9	10	5	8	4	2
3	7	9	10	5	8	4	2	1	6	3
4	2	1	6	3	7	9	10	5	8	4
5	8	4	2	1	6	3	7	9	10	5
6	3	7	9	10	5	8	4	2	1	6
7	9	10	5	8	4	2	1	6	3	7
8	4	2	1	6	3	7	9	10	5	8
9	10	5	8	4	2	1	6	3	7	9
10	5	8	4	2	1	6	3	7	9	10

For a=3, b=0, m=11

X_0	1	2	3	4	5
1	3	9	5	4	1
2	6	7	10	8	2
3	9	5	4	1	3
4	1	3	9	5	4
5	4	1	3	9	5
6	7	10	8	2	6
7	10	8	2	6	7
8	2	6	7	10	8
9	5	4	1	3	9
10	8	2	6	7	10

Analysis: We observe permutation cycles being generated. For the first value one 10-cycle<10,5,8,4,2,1,6,3,7,9> is being permuted, while the second one has two distinct 5-cycles<1,3,9,5,4> &<6,7,10,8,2>.
The first choice has 10 distinct values in each cycle and the second one has two cycles of five distinct values each.

Part II:

Here we have to generate a sequence of u_i. The following implementation was done taking x_0=5;
The following program generates first 100 $u_{i.}$

Implementation using C++:

(using a=1597,b=1,m=244944)

```
#include<cstdio>
#include<iostream>
using namespace std;
int main()
{
	int xo=5;long count=1;
	int initial=((1597*xo)+1)%244944;
	double u=(double)initial/244944;
	cout<<u<<"\n";
	int x;
	x=((1597*initial)+1)%244944;
	while(count<101)//running for 100 values
	{
		u=(double)x/244944;
		cout<<u<<"\n";
		x=((1597*x)+1)%244944;
		count++;
	}
	return 0;
}
```

(using a=51749,b=1,m=244944)

```
#include<cstdio>
#include<iostream>
using namespace std;
int main()
{
	int xo=5;long count=1;
	long long int initial=((51749*xo)+1)%244944;
	double u=(double)initial/244944;
	cout<<u<<"\n";
	long long int x;
	x=((51749*initial)+1)%244944;
	while(count<101)
	{
		u=(double)x/244944;
		cout<<u<<"\n";
		x=((51749*x)+1)%244944;
		count++;
	}
	return 0;
}
```

Implementation using R:

(using a=1597,b=1,m=244944)

```
xo<-5
count<-1
initial<-((1597*xo+1))%% 244944
u<-initial/244944
print (paste(u))
x<-initial
x<-((1597*x)+1)%% 244944
while (count<101){
	u<-x/244944
	print (paste(u))
	x <- ((1597*x)+1)%% 244944
```

```
      count<-count+1
}#using a while loop for the problem
```

(using a=51749,b=1,m=244944)

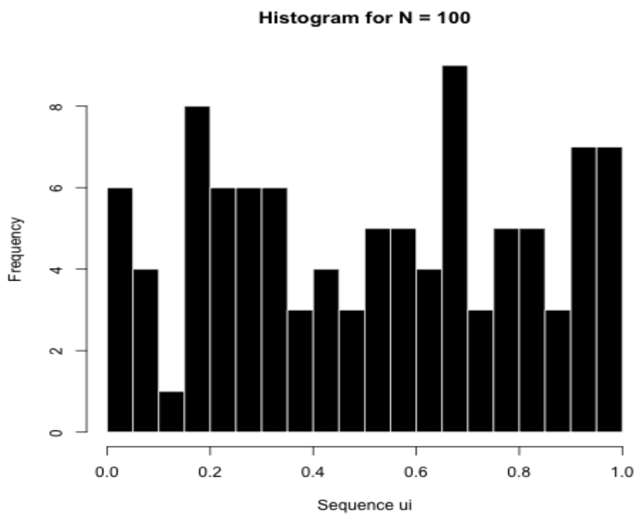
```
xo<-5
count<-1
initial<-((51749*xo+1))%% 244944
u<-initial/244944
print (paste(u))
x<-initial
x<-((51749*x)+1)%% 244944
while (count<100){
      u<-x/244944
      print (paste(u))
      x <- ((51749*x)+1)%% 244944
      count<-count+1
}#using a while loop for the problem
```

The Output of these programs are attached. Pl. see output 2.1.txt and output2.2.txt. the first 100 values are generated.
Modify the above program and count the frequency of the occurrences in class- width = 0.05.

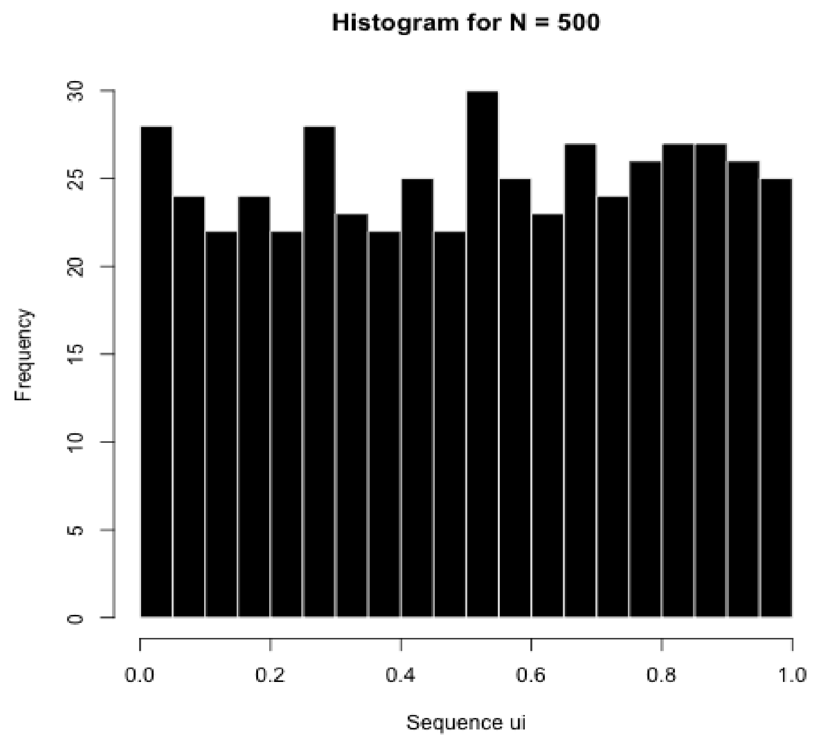
Final C++ code:

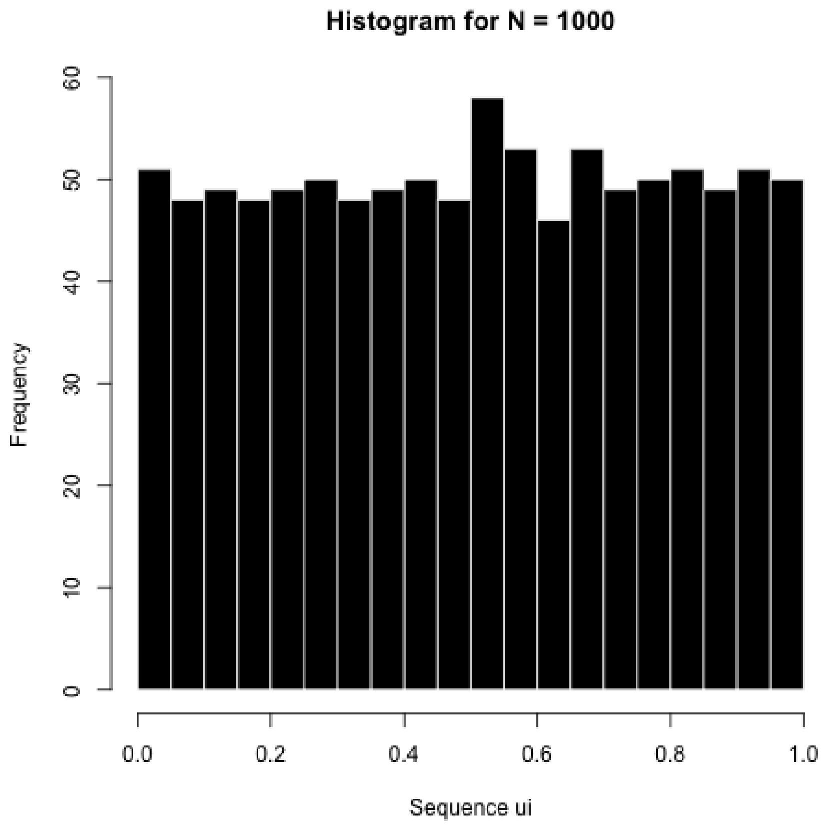
```
#include<cstdio>
#include<iostream>
#include<cmath>
using namespace std;
int main()
{
       int xo=5;long count=1;
       int initial=((1597*xo)+1)%244944;
       double u=(double)initial/244944;

       int x;int k;
       int a[20][2];// initializing array to store data
       x=((1597*initial)+1)%244944;
       while(count<101)
       {
              u=(double)x/244944;

              x=((1597*x)+1)%244944;
              count++;
              if(u>=0.0 && u<0.05)
                     a[0][1]=a[0][1]+1;
              if(u>0.05 && u<0.10)
                     a[1][1]=a[1][1]+1;
              if(u>=0.95 && u<1.0)
                     a[19][1]=a[19][1]+1;
              if(u>=0.1 && u<0.15)
                     a[2][1]=a[2][1]+1;
              if(u>=0.15 && u<0.2)
                     a[3][1]=a[3][1]+1;
              if(u>=0.2 && u<0.25)
                     a[4][1]=a[4][1]+1;
              if(u>=0.25 && u<0.30)
                     a[5][1]=a[5][1]+1;
              if(u>=0.30 && u<0.35)
                     a[6][1]=a[6][1]+1;
```

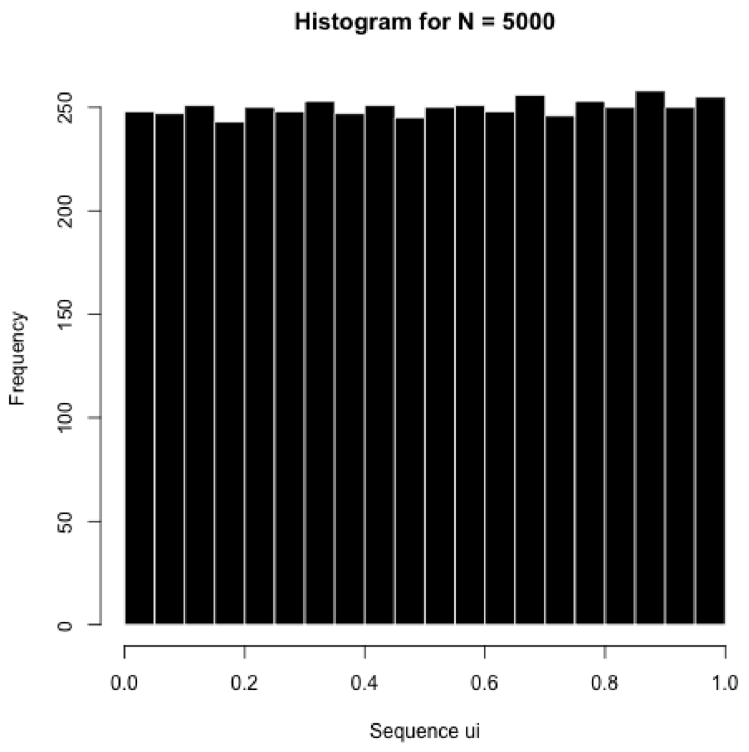
```
                if(u>=0.35 && u<0.40)
                        a[7][1]=a[7][1]+1;
                if(u>=0.40 && u<0.45)
                        a[8][1]=a[8][1]+1;
                if(u>=0.45 && u<0.5)
                        a[9][1]=a[9][1]+1;
                if(u>=0.5 && u<0.55)
                        a[10][1]=a[10][1]+1;
                if(u>=0.55 && u<0.60)
                        a[11][1]=a[11][1]+1;
                if(u>=0.60 && u<0.65)
                        a[12][1]=a[12][1]+1;
                if(u>=0.65 && u<0.70)
                        a[13][1]=a[13][1]+1;
                if(u>=0.70 && u<0.75)
                        a[14][1]=a[14][1]+1;
                if(u>=0.75 && u<0.80)
                        a[15][1]=a[15][1]+1;
                if(u>=0.80 && u<0.85)      //setting frequency
                        a[16][1]=a[16][1]+1;
                if(u>=0.85 && u<0.90)
                        a[17][1]=a[17][1]+1;
                if(u>=0.90 && u<0.95)
                        a[18][1]=a[18][1]+1;


        }int c=1;int i;int j;
        for(i=0;i<20;i++)
        {
                a[i][0]=i+1;
        }
        for(i=0;i<20;i++)
        {
                for(j=0;j<2;j++)
                {
                        cout<<a[i][j]<<" ";
                }
                cout<<"\n";
        }
        return 0;
}
```
The bare output values of the frequency can be viewed in output2.txt file enclosed. Using these values a Histogram was plotted.

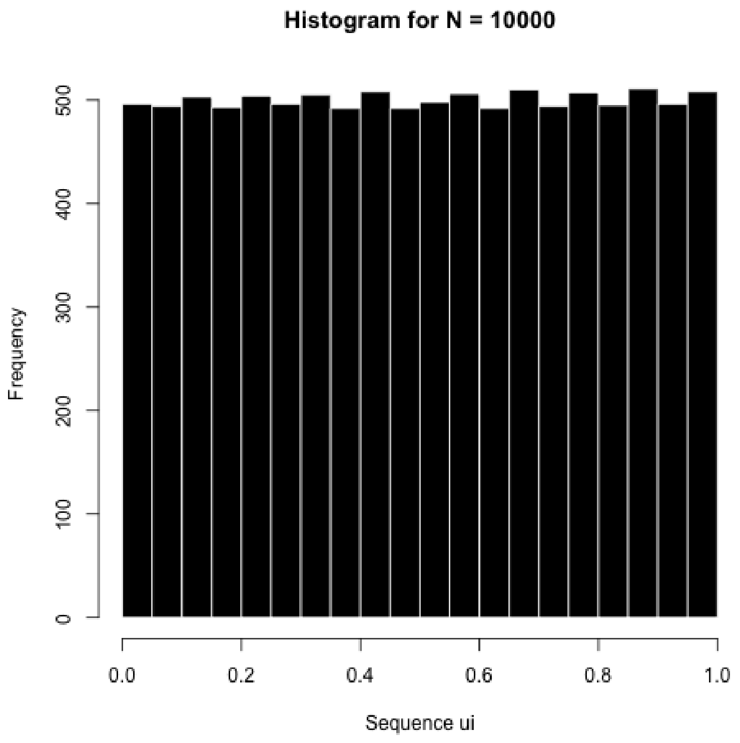


Increasing the number of values to 500, 1000, 5000, 10000, we obtain the following histograms:









Analysis:

Frequency of the values tends to a constant value. Observing thus that the constant is (N/20). As the value of N increases (or the number of iterations increases), the frequencies tend to stabilize.

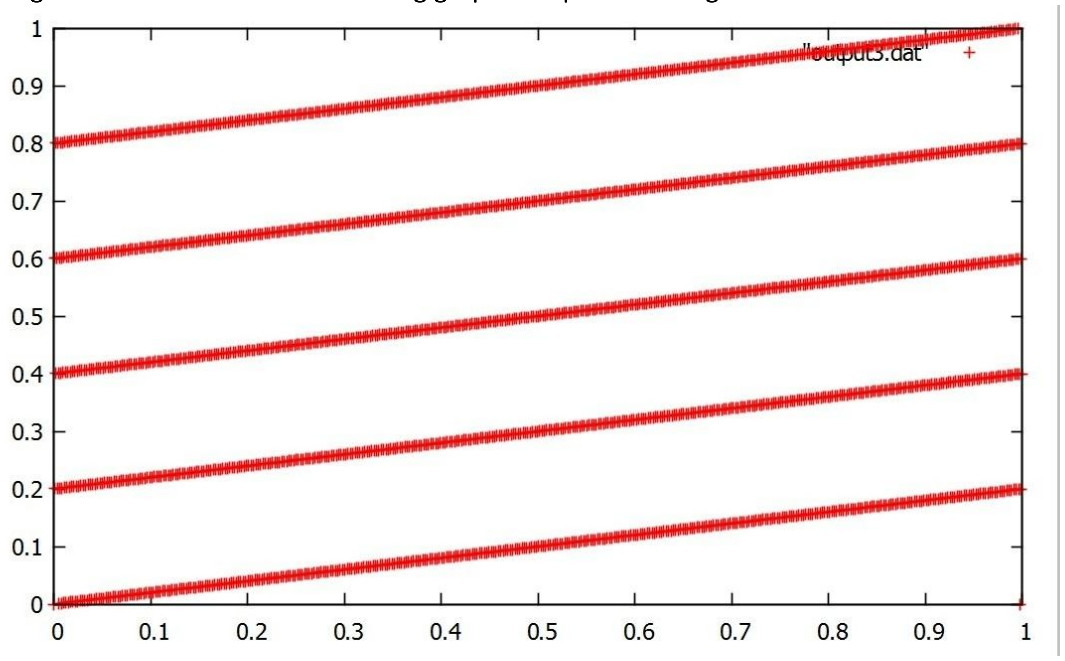
Part III:

Here we generate the set of points(u_{i-1},u_i), from the sequence of u_i.
The values used:
a=1229
b=1
m=2048

Implementation using C++:

```
#include<cstdio>
#include<iostream>
using namespace std;
int main()
{
        int xo=5;
        int initial=((1229*xo)+1)%2048;
        double u=(double)initial/2048;
        int x;
        x=((1229*initial)+1)%2048;
        while(x!=initial)
        {
                cout<<u<<" ";
                u=(double)x/2048;
                cout<<u<<"\n";
                x=((1229*x)+1)%2048;
        }
        return 0;
}
```

Using the data obtained the following graph was plotted using GNUPLOT :



Analysis: The data lies on specific lines. This indicates the lack of randomness of the generator.