MA226 : Monte-Carlo Simulation
Linear Congruential Generator
Assignment 2

Turkhade Hrushikesh Pramod
150123044

25-01-2017

1 Problem 1

Our Linear congruential generator has following form:

$x_{i+1} = (a * x_i + b) \bmod m$

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

We are given two different linear congruentials here:

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

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

Let's examin them one by one

1.1 Source code of the solution

```
int arr[100];
double arr2[100];

void genRan(int a, int b, int m,int seed ,int length)
{
        int index=1;
        arr[0]=seed;

        for(int i=1;i<=length;i++)
        {
                arr[i]= (a*arr[i-1]+b)%m;
        }
        for(int i=0;i<=length;++i)
        {
                arr2[i]=arr[i]*1.0/m;
        }


}

int main()
{
        int a,b,m,seed=5,length=50;

        a=6;
        b=0;
        m=11;
        for(int i=0;i<=10;i++)
        {
                seed=i;
                genRan(a,b,m,seed,length);
                cout<<"Numbers generated for seed = "<<seed<<" : "<<endl;
                for(int i=0;i<=length;i++)
                {
                        cout<<arr2[i]<<" ";
```

```
                }
                cout<<endl;
        }

        a=3;
        b=0;
        m=11;
        for(int i=0;i<=10;i++)
        {
                seed=i;
                genRan(a,b,m,seed,length);

                cout<<"Numbers generated for seed = "<<seed<<" : "<<endl;
                for(int i=0;i<=length;i++)
                {
                        cout<<arr2[i]<<" ";
                }
                cout<<endl;
        }
}
```

1.2 Analysis

1.2.1 Case: a=6

The data of the first linear congruence equation for various seeds is as follow:

Tabelle 1: Data

seed	Generated Values
0	0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
1	0.09 0.55 0.27 0.64 0.82 0.91 0.45 0.73 0.36 0.18 0.09 0.55
2	0.18 0.09 0.55 0.27 0.64 0.82 0.91 0.45 0.73 0.36 0.18 0.09
3	0.27 0.64 0.82 0.91 0.45 0.73 0.36 0.18 0.09 0.55 0.27 0.64
4	0.36 0.18 0.09 0.55 0.27 0.64 0.82 0.91 0.45 0.73 0.36 0.18
5	0.45 0.73 0.36 0.18 0.09 0.55 0.27 0.64 0.82 0.91 0.45 0.73
6	0.55 0.27 0.64 0.82 0.91 0.45 0.73 0.36 0.18 0.09 0.55 0.27
7	0.64 0.82 0.91 0.45 0.73 0.36 0.18 0.09 0.55 0.27 0.64 0.82
8	0.73 0.36 0.18 0.09 0.55 0.27 0.64 0.82 0.91 0.45 0.73 0.36
9	0.82 0.91 0.45 0.73 0.36 0.18 0.09 0.55 0.27 0.64 0.82 0.91
10	0.91 0.45 0.73 0.36 0.18 0.09 0.55 0.27 0.64 0.82 0.91 0.45

According to the above table, for each seed the value of the random numbers generated repeats itself after 10^{th} iteration. Hence, it is a full period generator.

1.2.2 Case: a=3

The data of the first linear congruence equation for various seeds is as follow:

Tabelle 2: Data

seed	Generated Values
0	0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
1	0.09 0.27 0.82 0.45 0.36 0.09 0.27 0.82 0.45 0.36 0.09 0.27
2	0.18 0.55 0.64 0.91 0.73 0.18 0.55 0.64 0.91 0.73 0.18 0.55
3	0.27 0.82 0.45 0.36 0.09 0.27 0.82 0.45 0.36 0.09 0.27 0.82
4	0.36 0.09 0.27 0.82 0.45 0.36 0.09 0.27 0.82 0.45 0.36 0.09
5	0.45 0.36 0.09 0.27 0.82 0.45 0.36 0.09 0.27 0.82 0.45 0.36
6	0.55 0.64 0.91 0.73 0.18 0.55 0.64 0.91 0.73 0.18 0.55 0.64
7	0.64 0.91 0.73 0.18 0.55 0.64 0.91 0.73 0.18 0.55 0.64 0.91
8	0.73 0.18 0.55 0.64 0.91 0.73 0.18 0.55 0.64 0.91 0.73 0.18
9	0.82 0.45 0.36 0.09 0.27 0.82 0.45 0.36 0.09 0.27 0.82 0.45
10	0.91 0.73 0.18 0.55 0.64 0.91 0.73 0.18 0.55 0.64 0.91 0.73

According to the above table, for each seed the value of the random number generated repeats itself after 5^{th} iteration.

1.3 Conclusion

Comparing the two linear congruential generators, we obseve that the first one is better than the second one. This is primarily due the fact that first one is a full period generator, while the second one repeats its values halfways through the generation.

2 Problem 2

In this problem, we have two linear congrential generators.

$a = 1597 b = 0 m = 244944$

$a = 51749 b = 0 m = 244944$

2.1 Source code for generating frequencies

```
int seed[6]={1,9999,6879,443};
int arr[100010];
int count[6][100010];

void genRan(int a, int b, int m,int length,int k)
{
        double div=1.0/k;
        for(int j=1;j<=5;++j)
        {
                arr[0]=seed[j];
```

```
                for(int i=1;i<=length;++i)
                {
                        arr[i]=(a*arr[i-1]+b)%m;
                }
                for(int i=1;i<=length;++i )
                {
                        int a=floor((arr[i]*1.0/m)/div);
                        count[j][a+1]++;
                }
        }
}


int main()
{
        int k=20;
        int a,b,m,length=100000;
        m=244944;
        a=51749;
        // a=1597
        b=0;

        // ofstream f;
        // f.open("lab2_que2_data.csv");

        genRan(a,b,m,length,k);

        // cout<<setprecision()

        for(int i=1;i<=5;i++)
        {
                // f<<seed[i]<<",";
        }
        // f<<"\n";

        cout<<"count         "<<seed[1]<<"  "<<seed[2]<<"  "<<seed[3]<<endl;
        for(int i=1;i<=k;++i)
        {
                cout<<setprecision(2)<<fixed;
                cout<<1.0*(i-1)/k<<"-"<<1.0*i/k<<"     ";
                for(int j=1;j<=3;++j)
                {
                        // f<<count[j][i]<<",";
                        cout<<count[j][i]<<" ";
                }
                // f<<"\n";
                cout<<endl;
        }

}
```

2.2 Analysis

2.2.1 Equation 1

Following are the histograms for seeds $9999, 6879 and 443$. The linear congruential genarator used is $a = 1597 b = 0 m = 244944$.

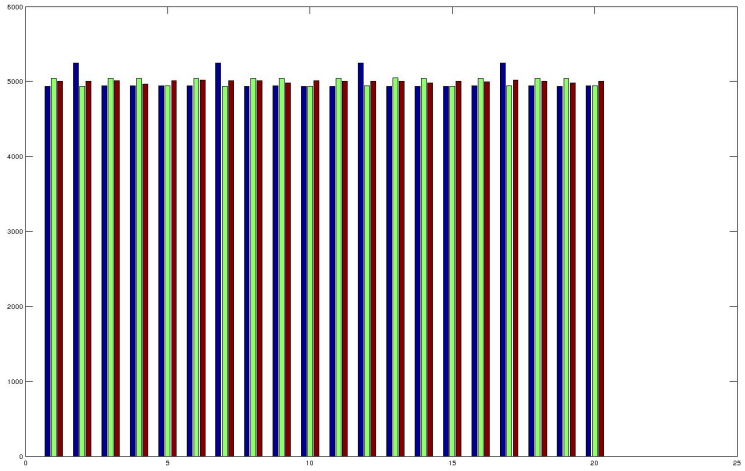

Abbildung 1: Here Blue is 9999,Green is 6879 and Red is 443

2.3 Analysis

2.3.1 Equation 2

Following are the histograms for seeds $9999, 6879 and 443$. The linear congruential genarator used is $a = 51749 b = 0 m = 244944$.

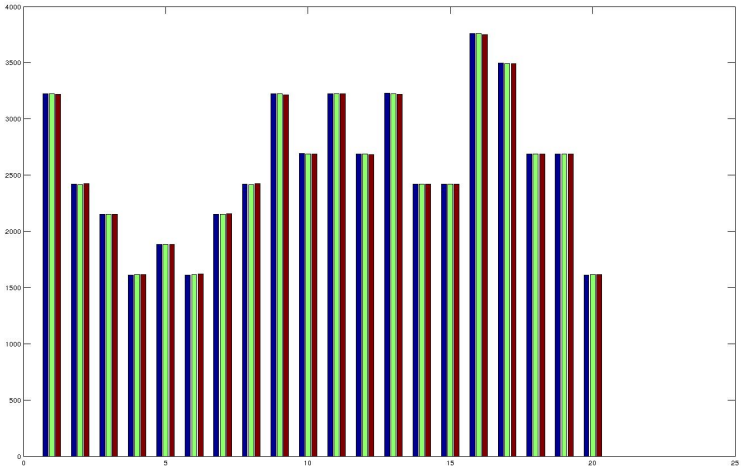

Abbildung 2: Here Blue is 9999,Green is 6879 and Red is 443

2.4 Conclusion

From above we can conclude that the first random number genertor is fairly good in comparison to the second one. The frequencies in case of the first one are quite uniform and that's what we want. Hence, the first one is better.

3 Problem 3

In this problem, we plot graph for (u_{i-1}, u_i) on X-Y axis.

3.1 Source code for generation data points

```
int main()
{
        int a,b,m,length=100000;
        ofstream f;
        f.open("lab2_que3_data.csv");
        a=1229;
        b=1;
        m=2048;

        int seed=7;

        f<<seed<<",";

        for(int i=0;i<=length;i++)
        {
```

```
                seed=(a*seed+b)%m;
                f<<seed<<"\n";
                f<<seed<<",";
        }
        f<<(a*seed+b)%m<<"\n";
}
```

3.2 Plot of the points

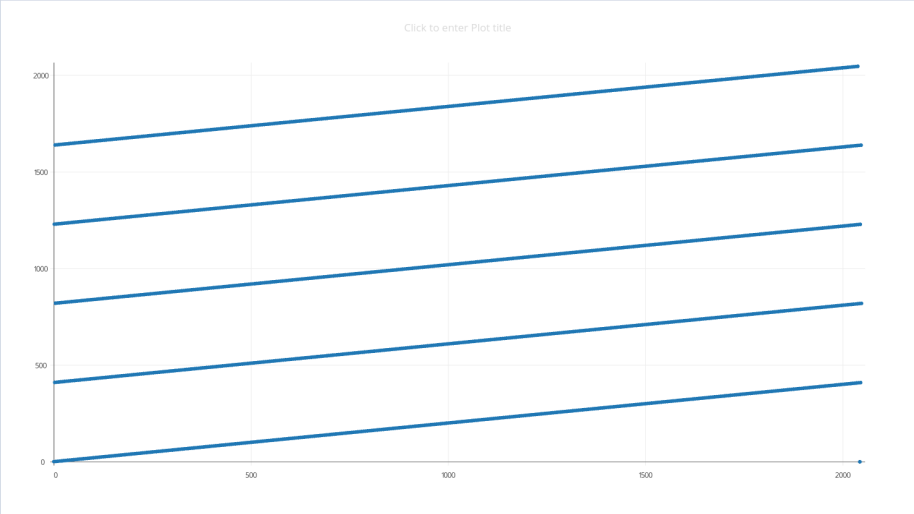

Abbildung 3: 2D plot for (u_{i-1}, u_i) , seed=7

3.3 Conclusion

From this plot we find that all the generated points are aligned in the straight lines. So, we can safely conclude that for a given seed the straight lines are fixed and all the generated points lie on the straight lines.