Solution and Grading Scheme of Midsem

Dr. Arabin Kumar Dey

3 March 2013

Q 1 Implement the linear congruence generator $x_{i+1} = ax_i \mod m$ to generate a sequence x_i and hence uniform random numbers u_i . Make use of the following set of values of a and m: $a = 16807$ and $m = 2^{31} - 1$ (a) Give first 10 numbers. Set your seed at $X_0 = 5$. (b) Calculate mean based on 10000 generated samples.

Code for C

```
# include <iostream>
# include <cmath>

using namespace std;

int main()
{
long long int x;
int i;
float u[10000],mean=0;

x=5;

for (i=0; i<10000; i++)
{
u[i]=x/(pow(2,31)-1);

if (i<10)
    cout<<u[i]<<endl;

mean+=u[i];
x=(16807*x)%((long long int)(pow(2,31)-1));
}

mean/=10000;
cout <<"\nMean = "<<mean<<endl;

return 0;
}
```

Output:

(a) The ten generated numbers are : 2.32831e-09

3.91318e-05

0.657689

0.778027

0.293251

0.663836

0.0947959

0.235223

0.394324

0.396482

(b) The calculated mean = 0.501091; which is close to the theoretical mean 0.5.

Q 2 Consider the extended Fibonacci generator : $X_i = (X_{i-17} + X_{i-5}) \mod 2^{31}$

Use the above linear congruence generator to generate the first 17 values of X_i, but taking $a = 16807$, $m = 2^{31}$.

(a) Then generate the values of uniform random numbers U_i (say for 1000, 10000 and 100000 values) and draw the histograms.

(b) For $n = 10000$, plot (U_i, U_{i+1}).

Q 3 Compare the above LGC and Fibonacci generator in terms of periodicity and auto-correlation with lag - 1.

The code for generation of histogram is shown below for n =1000. Same code will generate frequecies varying over n = 10000 and 100000 :

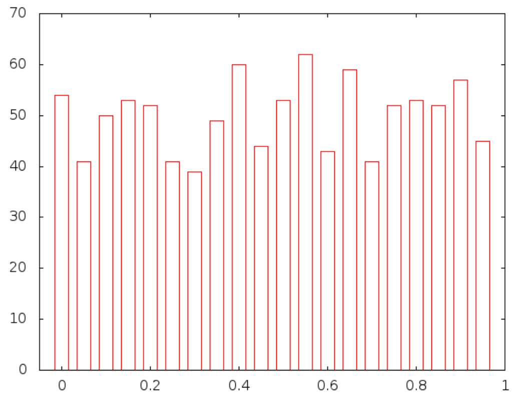
Q 2 (a) Code:

```
# include <iostream>
# include <cmath>
# include <cstring>
# include <cstdio>

using namespace std;

int main()
{
long long int x[1000];
float u[1000];
```

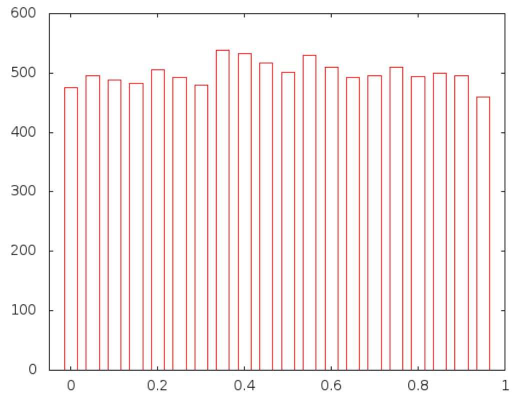
```
int i,k,freq[20];

for (i=0; i<20; i++)
    freq[i]=0;

x[0]=5;

for (i=0; i<17; i++)
    {
    u[i]=x[i]/(pow(2,31));
    x[i+1]=(16807*x[i])%((long long int)(pow(2,31)));

    for (k=0; k<20; k++)
        {
        if (u[i]>=k*0.05 && u[i]<(k+1)*0.05)
            {
            freq[k]++;
            break;
            }
        }
    }

for (i=17; i<1000; i++)
    {
    x[i]=(x[i-17]+x[i-5])%((long long int)(pow(2,31)));
    u[i]=x[i]/(pow(2,31));

    for (k=0; k<20; k++)
        {
        if (u[i]>=k*0.05 && u[i]<(k+1)*0.05)
            {
            freq[k]++;
            break;
            }
        }
    }

for (i=0; i<20; i++)
    cout<<i*0.05<<' '<<freq[i]<<endl;

return 0;
```

```
}
```

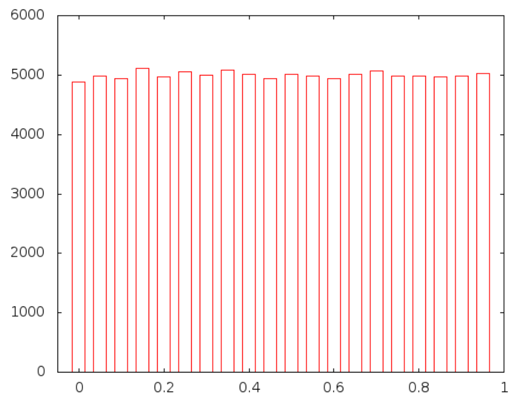
The histograms can be shown as :



(a) ξ_1



(b) ξ_2

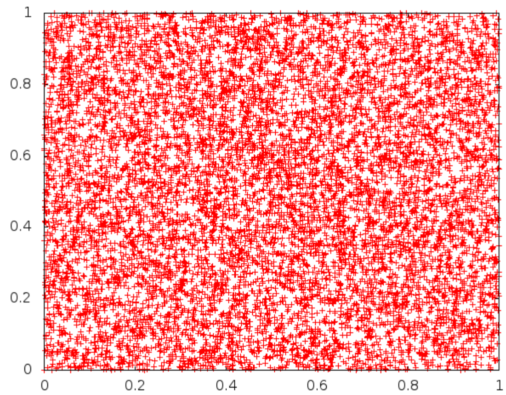


(c) ξ_3

Figure 1: *Histograms for (a) $n = 1000$, (b) $n = 10000$ and (c) $n = 100000$*

The picture shows that the generated numbers are correct, since and n increasing we are getting the proper shape of uniform distribution.

(b) For $n = 10000$, plot (U_i, U_{i+1}) is as follows :



(a) ξ_1

Figure 2: *Histograms for (a) n = 10000*

Code for Q3 : The following the code for periodicity and auto-correlation with lag - 1 for LCG and Fibonocci generator.

```
// Code for autocorrelation of Fibonocci Generator.
#include<iostream>
#include<fstream>
#include<cmath>

using namespace std;

int main()
{

    long long int m, t[100000], x, mat[20];
    int i, l;
    double u[100000], mean=0, var=0, cor=0, autocor;
    m=pow(2, 31);
    x=5;
    for(i=0; i<17; i++)
    {
        x=(16807*x)%m;
        t[i]=x;
        u[i]=(double)x/m;
        mean=mean+u[i]/100000;
        var=var+u[i]*u[i]/100000;
        if(i<11)
        {
            cout<<" "<<u[i];
```

```
        }
        if(i>=1)
        {
            //outf<<u[i-1]<<"\t"<<u[i]<<endl;
        }
    }
    for(i=17; i<100000; i++)
    {
        t[i]=(t[i-17]+t[i-5])%m;
        u[i]=(double)t[i]/m;
        mean=mean+u[i]/100000;
        var=var+u[i]*u[i]/100000;
        if(i<11)
        {
            cout<<" "<<u[i];
        }
        //outf<<u[i-1]<<"\t"<<u[i]<<endl;
    }
    var=var-mean*mean;
    cout<<"\nmean is"<<mean;
    cout<<"\n var is"<<var;
    i=1;
    for(i=1; i<100000; i++)
    {
        cor=cor+(u[i]-mean)*(u[i-1]-mean)/100000;
    }
    autocor=cor/var;
    cout<<"\n autocor is"<< autocor;



}
```

```
//Code for autocorrelation of LCG
#include<iostream>
#include<fstream>
#include<cmath>

using namespace std;

int main()
```

```
{

    long long int m, x;
    int i;
    double u[10000], mean=0, var=0, cor=0, autocor;
    m=pow(2, 31)-1;

    x=5;
    for(i=0; i<10000; i++)
    {
        x=(16807*x)%m;
        u[i]=(double)x/m;
        mean=mean+u[i]/10000;
        var=var+u[i]*u[i]/10000;
        if(i<11)
        {
            cout<<" "<<u[i];
        }
    }
    var=var-mean*mean;
    cout<<"\nmean is"<<mean;
    cout<<"\n var is"<<var;
    i=1;
    for(i=1; i<10000; i++)
    {
        cor=cor+(u[i]-mean)*(u[i-1]-mean)/10000;
    }
    autocor=cor/var;
    cout<<"\n autocor is"<< autocor;



}
```

Results

Autocorrelation of lag 1 by LCG : 0.0141241 Autocorrelation of lag 1 by Fibonacci generator : 0.00125989

Therefore autocorrelation of Fibonacci generator is more close to zero than that of LCG. Therefore Fibonacci generator is better than the given LCG in terms of autocorrelation.

```
//Code for calculating period of LCG
#include<iostream>
#include<cmath>
using namespace std;
int main()
{
    {
    long int a=16807;
    long long int m=pow(2,31)-1;
    long long int x=5,n=1;
    float u=0,mean=0;
    x=(a*x)%m;
    n+=1;
    int i=0;
    while(x!=5)
    {
        x=(a*x)%m;
        n+=1;
        u=((float)x/(float)m);
        if(i==0)
            mean=u;
        else
            mean=((float)(mean*i)/(float)(i+1))+((float)u/(float)(i+1));
        i+=1;
    }
    n=n+1;
    cout<<"period\t\t"<<n<<endl;
    cout<<"mean\t\t"<<mean<<endl;
    for(int i=0;i<1000;i++)
    {}
    return 0;
    }
    /*{
    long int a=16807;
    long long int m=pow(2,31)-1;
    long long int x=5,y,n=1;
    float u=0;
    long int b[17]={0};
    for(int i=0;i<17;i++)
    {
        x=(a*x)%m;
```

```
        b[i]=x;
        //cout<<x<<endl<<endl;
    }
   m=m+1;
   y=x;
   int i=0;
   while((x!=y)||(n==1))
   {
        x=((b[i]%m)+(b[(i+12)%17]%m))%m;
        b[i]=x;
        i=(i+1)%17;
        n=n+1;
   }
   cout<<"period\t\t"<<n<<endl;
   }*/

}
```

Results: Period of the LCG = $2^{31} - 2 = 2147483648$ as expected, whereas period of Fibonocci generator that we have used here is equal to $2^{31}(2^{17} - 1)$ is much larger than LCG. Therefore Fibonocci is better than LCG in terms of periodicity.

Q 4 The following is the probability density function for the Weibull distribution

$$f(x;\beta,\theta) = \beta\theta^{\beta}x^{\beta-1}e^{-(\theta x)^{\beta}}, \quad x > 0, \theta, \beta > 0$$

Generate random number from the following Weibull distribution by inverse transform method. Take $\theta = 1.5$, $\beta = 2$ and draw histogram for n = 100, 500, 1000. [5 marks, Use R]

Solution: We can write the cdf of Weibull distribution as

$$F_{WE}(x;\beta,\theta) = (1 - e^{-\theta x})^{\beta}.$$

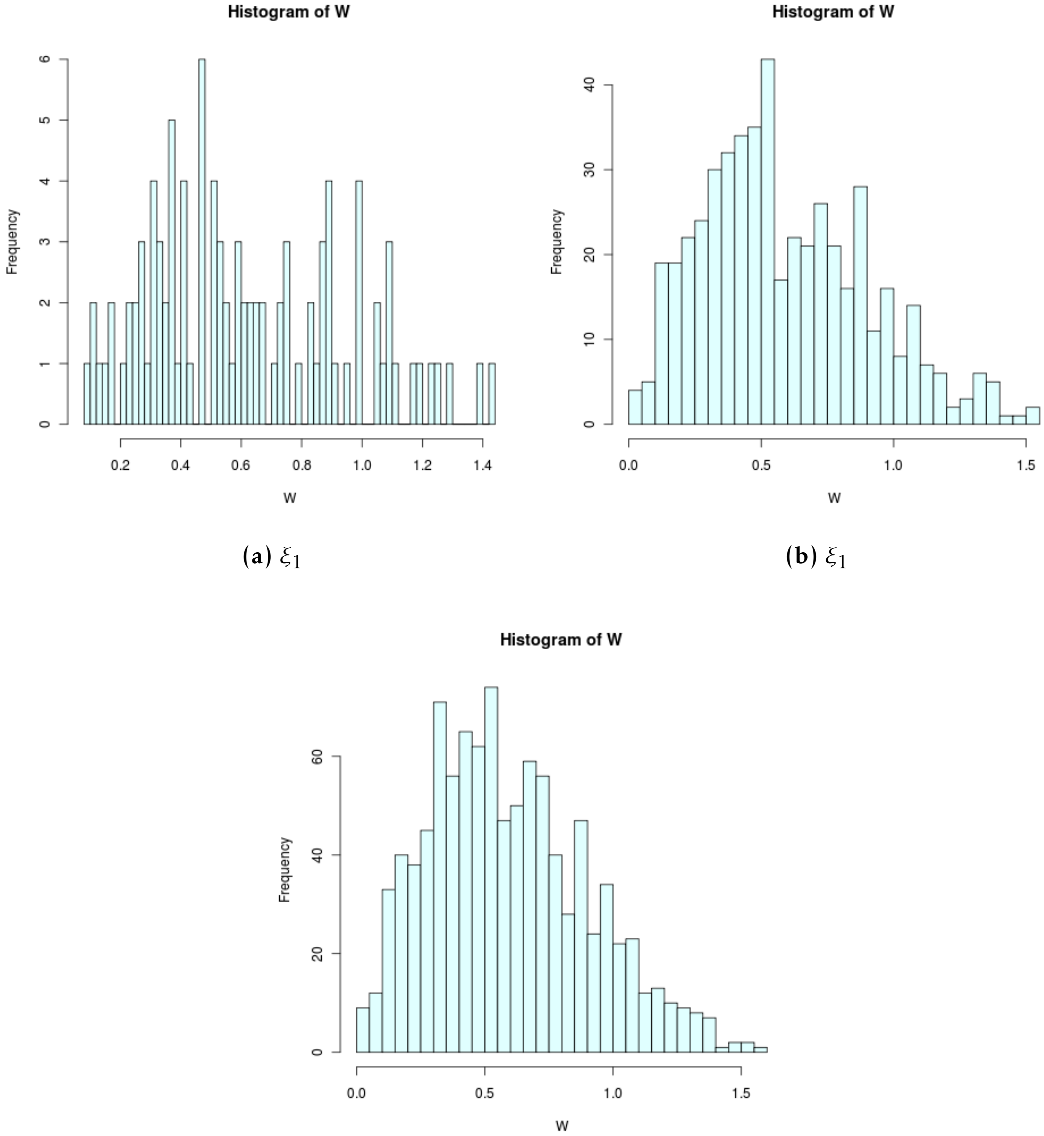
To apply inverse transform method we equate $u = (1 - e^{-\theta x})^{\beta} \Rightarrow x = -\frac{1}{\theta}\log(1 - u^{1/\beta})$ where u is an observation from Uniform(0,1).

Algorithm 1 Generating Random number from Weibull distribution

1: Generate U from $\mathcal{U}[0,1]$.

2: Generate X from the following relation $X = -\frac{1}{\theta}\log(1 - U^{1/\beta})$.

```
genWeib<-function(n)
{
W<-vector(length=n);
 set.seed(5);

u<-runif(n,0,1);
W=(1/1.5)*((-log(1- u))^0.5);

png("Q4_3.png");
 hist(W, breaks=50, col="light cyan",plot=TRUE);
 dev.off();

}
```



(a) ξ_1 (b) ξ_1

(c) ξ_1

Figure 3: *Histograms for (a) n = 100 (b) n = 500 (c) n = 1000*

Q 5 Generate 1000 random number from gamma(shape $=\frac{3}{2}$, scale =1) by acceptance-rejection method. Find mean and variance. How do you say your generated random number is correct ? Please set your seed at 151, before generating random number from uniform distribution. [Hints: The pdf of gamma(shape $=\frac{3}{2}$, scale =1) can be given by

$$f(x) = \frac{1}{\Gamma(\frac{3}{2})} e^{-x} x^{\frac{3}{2}-1}; \quad x > 0$$

]

For generating random number from gamma distribution(shape $=\frac{3}{2}$, scale =1) by acceptance-rejection method we choose exponential with mean = $\frac{3}{2}$ as our candidate distribution. We write the pdf of exponential as

$$g(x) = \frac{2}{3}e^{-\frac{2}{3}x}$$

We choose c maximizing $\frac{f(x)}{g(x)} = \frac{3}{\sqrt{\pi}}x^{\frac{1}{2}}e^{-\frac{1}{3}x}$

Maximum of c will attain at $x = \frac{3}{2}$ i.e. $c = \frac{3}{\sqrt{\pi}}(\frac{3}{2})^{\frac{1}{2}}e^{-\frac{1}{2}}$.

Algorithm 2 Generating random number from Gamma distribution by acceptance-rejection method

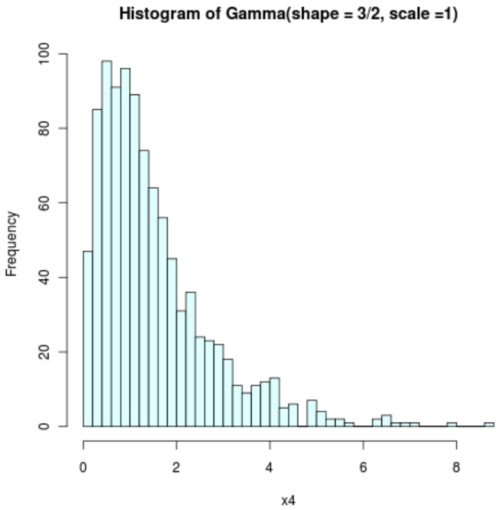
1: Generate U from $\mathcal{U}[0,1]$ so that $Y = \frac{-3}{2}\log U_1$
2: Generate U_2.
3:
4: **if** $U_2 < \frac{f(Y)}{cg(Y)} = \sqrt{\frac{2e}{3}}\sqrt{Y}e^{-Y/3}$ **then**
5: X = Y
6: **end if**

```
f<-function(n)
{

 set.seed(151);
 u<-runif(2*n,0,1);
 #con<-(3/sqrt(pi))*(sqrt(3/2))*(1/sqrt(exp(1)));
 u1<-u[1:n];
 u2<-u[(n + 1):(2*n)];
 y<--(3/2)*log(u1);
 w2<-y;

 x1<-y[u2 < (sqrt((2*exp(1))/3))*(sqrt(y))*(exp(-y/3))];



len<-length(x1);
redu<-(n - len);
aprob<-(len/n);
v<-round((n - len)/aprob);
u3<-runif(v,0,1);
y1<--(3/2)*log(u3);
```

```
x3<-y1[u3 < (sqrt((2*exp(1))/3))*(sqrt(y1))*(exp(-y1/3))]
x4<-c(x1, x3);

hist(c(x1, x4), breaks=20);

me<-mean(x4);
va<-var(x4)

return(c(me,va));

}
```

Results:

Mean and variance are 1.509025 and 1.510892 respectively which is equal to the theoretical mean and variance 1.5. Moreover shape of histogram also ensures that generated random numbers are correct.



(a) ξ_1

Figure 4: *Histograms for (a) n = 1000*

1 Grading Scheme

1. Code properly written :

   Q1) 3 marks

   Q2) 3 marks

Q3) 2 marks

Q4) 2 marks

Q5) set seed= 1 marks; correct code = 3 marks;

2. Results and Interpretation properly stated and graphs are given-

   Q1) 2 marks

   Q2) 4 marks

   Q3) 1 marks

   Q4) 3 marks

   Q5) calculation of c and choice of candidate distribution = 3 marks; other interpretation and graphs = 3 marks;